

WESTERN SYDNEY
UNIVERSITY



Digital Neuromorphic Auditory Systems

by

Ram Kuber Singh

A dissertation submitted in fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Supervisor: Professor André van Schaik

Co-Supervisor: Professor Sue Denham

Associate Professor Tara Julia Hamilton

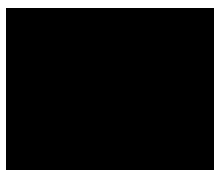
Associate Professor Gregory Cohen

**International Centre for Neuromorphic Systems (ICNS),
The MARCS Institute for Brain, Behaviour and Development,
Western Sydney University
Werrington South, NSW, Australia**

2020

Statement of Authentication

The work presented in this dissertation is, to the best of my knowledge and belief, original except as acknowledged in the text. I hereby declare that I have not submitted this material, either in full or in part, for a degree at this or any other institution.



(Ram Kuber Singh)

To Śrīla Prabhupāda, Śrī-Śrī Gaura-Nitāi, and Śrī-Śrī Rādhā-Kṛṣṇa

Acknowledgements

I would like to offer my thanks to my principal PhD supervisor, André van Schaik, for mentoring me. Under his tutelage, I have been encouraged to think analytically as a researcher and an engineer – a trait, which I have used on other students. I wish to thank my co-supervisor, Sue Denham, for steering me in solving problems and presenting ideas from an alternative perspective besides the neuromorphic lens. I also want to thank Tara Hamilton for her guidance from technical issues to conference selection to just spewing jokes. My final thanks to my supervisory panel go to Greg Cohen, who encouraged me on surviving the PhD journey through his own experiences. My sincere thanks to all of you for your willingness to guide me and tolerating my peculiarities.

I would also like to thank Mark Wang for his help on getting me up to speed with FPGA-related development and Upul Gunawardana for sharpening my presentation skills by allowing me to be his teaching assistant. I also wish to thank the following good people for their support and acquaintances: Kate Stevens, Paul Breen, Gaetano Gargiulo, Yossi Buskila, Ganesh Naik, Travis Monk, Andrew Nicholson, Gough Lui, James Wright, Chetan Singh, Patrick Kasi, Ying Xu, Elham Shabanivaraki, David Karpul Saeed Afshar, Hossein Moinsadeh, Titus Jayarathna, Mohd Atiqul, Nick Ralph, Sally Longmore, Grahame Andrews, Colin Symons, Paras Karki. My thanks to my two examiners, Shihab Shamma and Richard F. Lyon for their comments to further improve the contents of my PhD dissertation.

I am grateful to my sister, Laj, and her husband, Joseph, and her family, who helped with my accommodation during my PhD journey. I am deeply thankful to my mother, Bhasanthy Singh, for her love and steadfast support to make this academic journey possible.

Finally, a special and heart-felt thank you to my spiritual teacher, Śrīla Prabhupāda, whose wisdom and teachings kept me going on regularly (and still does), Śrī-Śrī Gaura-Nitāi, and Śrī-Śrī Rādhā-Kṛṣṇa for their transcendental support, who made this endeavour possible for me to achieve.

Abstract

This dissertation presents several digital neuromorphic auditory systems. Neuromorphic systems are capable of running in real-time at a smaller computing cost and consume lower power than on widely available general computers. These auditory systems are considered neuromorphic as they are modelled after computational models of the mammalian auditory pathway and are capable of running on digital hardware, or more specifically on a field-programmable gate array (FPGA). The models introduced are categorised into three parts: a cochlear model, an auditory pitch model, and a functional primary auditory cortical (A1) model. The cochlear model is the primary interface of an input sound signal and transmits the 2D time-frequency representation of the sound to the pitch models as well as to the A1 model. In the pitch model, pitch information is extracted from the sound signal in the form of a fundamental frequency. From the A1 model, timbre information in the form of time-frequency envelope information of the sound signal is extracted.

Since the computational auditory models mentioned above are required to be implemented on FPGAs that possess fewer computational resources than general-purpose computers, the algorithms in the models are optimised so that they fit on a single FPGA. The optimisation includes using simplified hardware-implementable signal processing algorithms. Computational resource information of each model on FPGA is extracted to understand the minimum computational resources required to run each model. This information includes the quantity of logic modules, register quantity utilised, and power consumption. Similarity comparisons are also made between the output responses of the computational auditory models on software and hardware using pure tones, chirp signals, frequency-modulated signal, moving ripple signals, and musical signals as input. The limitation of the responses of the models to musical signals at multiple intensity levels is also presented along with the use of an automatic gain control algorithm to alleviate such limitations.

With real-world musical signals as their inputs, the responses of the models are also tested using classifiers – the response of the auditory pitch model is used for the classification of monophonic musical notes, and the response of the A1 model is used for the classification of musical instruments with their respective monophonic signals. Classification accuracy results are shown for model output responses on both software and hardware. With the hardware implementable auditory pitch model, the classification score stands at 100% accuracy for musical notes from the 4th and 5th octaves containing 24 classes of notes. With the hardware implementation auditory timbre model, the classification score is 92% accuracy for 12 classes musical instruments. Also presented is the difference in memory requirements of the model output responses on both software and hardware – pitch and timbre responses used for the classification exercises use 24 and 2 times less memory space for hardware than software.

Contents

Statement of Authentication	i
Acknowledgements	iii
Abstract	iv
Contents	v
Abbreviations	ix
List of Figures	xii
List of Tables	xv
1. Introduction	1
1.1. Aims and Chapter Synopses	2
1.2. Bibliography	5
2. The Auditory Pathway: A Modelling Perspective	7
2.1. A Survey of Auditory Pathway Models	7
2.1.1. MAP Model	10
2.1.2. CAR-FAC Model	16
2.1.3. Model Selection	20
2.2. Auditory Pitch	21
2.2.1. Pitch Perception	21
2.2.2. A Survey of Auditory Pitch Models	27
2.3. Auditory Timbre	35
2.3.1. Timbre Perception	36
2.3.2. A Survey of Auditory Cortical Models	37
2.4. Chapter Summary and Conclusion	42
2.5. Bibliography	42
3. CAR-Lite: A Multi-Rate Cochlear Model	54
3.1. Motivation	54
3.2. A Multi-Rate Cochlear Model	54
3.2.1. CAR Model Revisited	55
3.2.2. CAR-Lite	55
3.2.3. Fixed-Point Implementation	58
3.2.4. FPGA Implementation	61
3.2.5. Software Floating-point vs. Hardware Fixed-point	65
3.2.6. CAR vs. CAR-Lite	66
3.2.7. Response to Log Chirp	68

3.2.8.	Response to Music	73
3.3.	Encoding Sound Intensity (SI).....	77
3.3.1.	CAR-Lite-SI	78
3.3.2.	The New Auditory Nerve Algorithm	78
3.3.3.	Fixed-Point Implementation	81
3.3.4.	FPGA Implementation.....	83
3.3.5.	Response to Pure Tones	88
3.3.6.	Iso-Intensity Response	89
3.3.7.	Noise Effect on Real-World Signals	91
3.4.	Chapter Summary and Conclusion	95
3.5.	Bibliography	95
4.	Auditory Pitch Model: Autocorrelogram Generation	102
4.1.	Motivation	102
4.2.	Algorithm Characteristics	103
4.3.	General Model Characteristics	106
4.4.	CAR-Lite-ACF Model	108
4.4.1.	FPGA Implementation.....	110
4.4.2.	Response to Complex Tones.....	121
4.4.3.	Response to Missing Fundamental Frequency	126
4.4.4.	Response to Harmonics Phase Change	128
4.5.	Chapter Summary and Conclusion	130
4.6.	Bibliography	130
5.	A Functional Primary Auditory Cortical Model	133
5.1.	Motivation	133
5.2.	CAR-Lite-A1 Model.....	134
5.2.1.	Input Sampling Rate	135
5.2.2.	Filter Configuration Survey for Spectro-Temporal Modulation Filters	136
5.3.	Spectro-Temporal Modulation Directionality	152
5.3.1.	NSL Model.....	152
5.3.2.	CAR-Lite-A1 Model.....	153
5.4.	Circuit	159
5.5.	Fixed-Point Implementation	161
5.5.1.	Filter Stability	162
5.6.	FPGA Implementation.....	165
5.6.1.	Operation of Modules.....	168
5.6.2.	Hardware Resource Utilisation.....	169

5.6.3.	Software Floating-Point vs. Hardware Fixed-Point	170
5.7.	Model Responses	171
5.7.1.	Response to Moving Ripple	174
5.7.2.	Response to Frequency-Modulated and Log Chirp Signals.....	180
5.8.	Chapter Summary and Conclusion	184
5.9.	Bibliography	184
6.	Pitch Estimation and Classification of Musical Notes.....	189
6.1.	Pathway to Pitch Estimation: Model Settings and Ground Truth.....	189
6.2.	Pitch (f_0) Estimation from Autocorrelogram (AC)	191
6.2.1.	Peak-Picking.....	193
6.2.2.	Weighting High Peaks.....	195
6.2.3.	Threshold-Bound Search	197
6.2.4.	Summary of Algorithms.....	199
6.2.5.	Classifier Algorithm	200
6.2.6.	FPGA Implementation.....	200
6.3.	Results and Evaluation	203
6.3.1.	Accuracy of Pitch Estimation (0 dBFS Intensity Level)	204
6.3.2.	Varying Intensity and Noise Levels	208
6.3.3.	Autocorrelogram File Sizes	213
6.3.4.	Comparison with Other Models: Accuracy	214
6.4.	Summary and Conclusion	216
6.5.	Bibliography	216
7.	Classification of Musical Instruments.....	219
7.1.	Timbre Representation	219
7.2.	Musical Notes Selection.....	220
7.2.1.	Timbre Invariance	220
7.2.2.	Dynamics and Articulation.....	221
7.3.	Feature Representation	222
7.3.1.	Temporal Invariance	222
7.3.2.	Summary Profiles	223
7.4.	Input Similarity and Classification Algorithms	225
7.4.1.	2D Correlation Coefficient (CC).....	225
7.4.2.	Timbre Distance (TD).....	227
7.4.3.	Classification.....	230
7.4.4.	FPGA Implementation.....	230
7.4.5.	Application to Musical Signals.....	233

7.5.	Results and Evaluation	235
7.5.1.	Accuracy Comparison of 0 dBFS Input Signals.....	236
7.5.2.	Varying Intensity and Noise Levels	241
7.5.3.	In-model Comparison: 4D and 2D Matrix File Sizes	247
7.5.4.	Comparison with Other Models: Accuracy	248
7.6.	Summary and Conclusion	250
7.7.	Bibliography	251
8.	Summary, Conclusion, and Future Work	255
8.1.	Summary and Conclusion	255
8.2.	Future Work	257
8.3.	Publications	259
8.4.	Bibliography	259
A.	Automatic Gain Control (AGC)	261
	Bibliography	263
B.	Classification of Musical Notes (0 dBFS, No Noise)	264
C.	Classification of Musical Notes based on Varying Intensity and SNR Levels	267
D.	Classification of Musical Instruments (0 dBFS, No Noise)	277
E.	Classification of Musical Instruments based on Varying Intensity and SNR Levels.....	289

Abbreviations

A1	Primary Auditory Cortex
AAC	AND-Accumulate
ACF	Autocorrelation Function
AF	Apoyando / Finger
AGC	Automatic Gain Control
ALM	Adaptive Logic Module
ALU	Arithmetic Logic Unit
AM	Amplitude Modulation
AN	Auditory Nerve
ANN	Artificial Neural Network
AP	Apoyando / Pick
AR	Asymmetric Resonator
ASA	Acoustical Society of America
ASIC	Application-Specific Integrated Circuit
BF	Best Frequency
BM	Basilar Membrane
BPF	Bandpass Filter
CAR-FAC	Cascade of Asymmetric Resonator with Fast Acting Compression
CC	Correlation Coefficient
CF	Centre Frequency
CN	Cochlear Nucleus
CPU	Central Processing Unit
DRNL	Dual resonance nonlinear
DSP	Digital Signal Processor
ERB	Equivalent Rectangular Bandwidth
f_0	Fundamental Frequency
FIR	Finite Impulse Response

fMRI	Functional Magnetic Resonance Imaging
FSM	Finite State Machine
HN	Hard mallet / Normal
HPF	High-Pass Filter
HSR	High Spontaneous Rate
HWR	Half-Wave Rectifier
HT	Harmonic Template
IC	Inferior Colliculus
IHC	Inner Hair Cell
IIR	Infinite Impulse Response
LF	Legato / Finger
LFSR	Linear Feedback Shift Register
LIF	Leaky-integrate-and-fire
LP	Legato / Pick
LPF	Low-pass Filter
LSR	Low Spontaneous Rate
MAC	Multiply-Accumulate
MFCC	Mel-Frequency Cepstral Coefficient
MIC	Musical Instruments Classification
MNC	Musical Notes Classification
MOC	Medial Olivo-Cochlear
MRSAC	Multiply, Right-Shift, and Accumulate
MSR	Medium Spontaneous Rate
MTF	Modulation Transfer Function
NLF	Nonlinear Function
NO	Normal
OPC	Octave Processing Control
OS	Operating System

PBNG	Pseudorandom Binary Number Generator
PCA	Principal Component Analysis
PEQ	Peaking Equaliser
PN	Normal / Pick
PZ	Pole-Zero
QMHT	Quadrature Mirror Hilbert Transformer
RF	AI aire / Finger
RMS	Root-Mean-Square
RP	AI aire / Pick
SH	Slapping thumb
SI	Sound Intensity
SN	Soft mallet / Normal
SP	Spiccato
SPC	Section Processing Control
SR	Spontaneous Rate
ST	Staccato
STRF	Spectro-Temporal Receptive Field
SVM	Support Vector Machine
TI	Temporal Integration
TO	Tonguing
VCN	Ventral Cochlear Nucleus
VLSI	Very Large-Scale Integrated
WTA	Winner-Take-All
ZC	Zero-Crossings

List of Figures

Figure 1-1: Flowchart of chapters in this dissertation.....	3
Figure 2-1: Initial stages of the human auditory pathway.....	8
Figure 2-2: Sound energy transmitted from the outer ear to the inner ear.	9
Figure 2-3: Monoaural auditory models reviewed in this chapter.	9
Figure 2-4: Outer and middle ear filters.	10
Figure 2-5: Travelling wave of the basilar membrane (BM).	11
Figure 2-6: A dual resonant nonlinear (DRNL) filter.	12
Figure 2-7: Inner hair cell (IHC) cilia bundle motion.....	13
Figure 2-8: Neurotransmitter release model.	15
Figure 2-9: The CAR-FAC model.	17
Figure 2-10: The CAR model.	18
Figure 2-11: CAR-FAC digital inner hair cell (IHC) model.....	19
Figure 2-12: Automatic gain control (AGC) loop-filter.	20
Figure 2-13: Pitch siren experiment setup used by Seebeck [44] and Strutt [45].	22
Figure 2-14: Pulses recorded from Seebeck's siren experiment.....	22
Figure 2-15: Residue theory demonstration.	25
Figure 2-16: Residue pitch from fine-structured temporal waveform.....	27
Figure 2-17: Virtual pitch model.	29
Figure 2-18: Filterbank of autocorrelation functions (ACFs).....	31
Figure 2-19: Operation of an autocorrelation function (ACF).	32
Figure 2-20: The SPINET model.	34
Figure 2-21: An abstract of the temporal modulation model.	39
Figure 2-22: The multiresolution spectro-temporal auditory cortical model.	41
Figure 3-1: The CAR-Lite model.	57
Figure 3-2: BM signal gain response.	60
Figure 3-3: BM gain response similarity.	60
Figure 3-4: Architecture of the CAR-Lite model implemented on FPGA.	62
Figure 3-5: FPGA output vector waveforms of the CAR-Lite model.....	63
Figure 3-6: Octave processing map.	64
Figure 3-7: Gain difference between floating-point and fixed-point responses.....	66
Figure 3-8: 16 bits AR coefficients.....	67
Figure 3-9: 8 bits AR coefficients.....	68
Figure 3-10: BM and BMd gain and phase responses.....	70
Figure 3-11: New BMd gain response.	71
Figure 3-12: BMd and IHC responses.	72
Figure 3-13: AN spike response.....	72
Figure 3-14: Musical signal at various intensities.	74
Figure 3-15: AGC effect on BM responses.....	75
Figure 3-16: AGC effect on clipped amplitudes.	76
Figure 3-17: Bit widths of the BM signal at multiple intensities.	77
Figure 3-18: The CAR-Lite-SI model.	78
Figure 3-19: Spontaneous-rate (SR) – auditory nerve (AN) algorithm.	79
Figure 3-20: FPGA architecture of the CAR-Lite-SI model.	84
Figure 3-21: FPGA output vector waveform of the CAR-Lite-SI model.	85
Figure 3-22: Linear feedback shift register (LFSR).	86
Figure 3-23: Cochleagram representation of an IHC signal.....	88

Figure 3-24: SR-AN spike response.....	89
Figure 3-25: Iso-intensity responses.	91
Figure 3-26: Sound intensity representation of real-world sound signals.....	93
Figure 3-27: Signal-to-noise (SNR) ratio of real-world signals.	94
Figure 4-1: Bit width differences of elements in coincidence matrices.	105
Figure 4-2: (a) The CAR-Lite-ACF model.	109
Figure 4-3: LIF neuron firing threshold.	110
Figure 4-4: FPGA architecture of the CAR-Lite-ACF (MAC) model.	112
Figure 4-5: FPGA output vector waveforms of the CAR-Lite-ACF (MAC) model.	113
Figure 4-6: Simulation of a MAC-based ACF algorithm.	116
Figure 4-7: SystemVerilog simulation of a MAC-based ACF algorithm.....	117
Figure 4-8: FPGA architecture of the CAR-Lite-ACF (AAC) model.	118
Figure 4-9: FPGA output vector waveforms of the CAR-Lite-ACF (AAC) model.	119
Figure 4-10: Simulation of an AAC-based ACF algorithm.....	120
Figure 4-11: SystemVerilog simulation of the AAC-based algorithm.....	121
Figure 4-12: Autocorrelogram matrices.	122
Figure 4-13: Magnified temporal profiles.	125
Figure 4-14: Degrees of similarity between autocorrelograms.....	126
Figure 4-15: Autocorrelogram representation of a harmonic signal.	127
Figure 4-16: Stimulus with (a) sine (0°) phase, and (b) alternating (90°) phase.....	129
Figure 4-17: Temporal profiles of sine-phase and alternating-phase stimuli.	129
Figure 5-1: The CAR-Lite-A1 model.	135
Figure 5-2: 2 nd -order IIR standard bandpass filter (BPF) direct-form-1 configuration.	140
Figure 5-3: 2 nd -order IIR peaking equaliser filter.....	141
Figure 5-4: 2 nd -order cascade LPF-HPF BPF.	143
Figure 5-5: 4 th -order cascade LPF-HPF BPF.....	144
Figure 5-6: Coupled-form configuration of a 2 nd -order asymmetric resonator.	146
Figure 5-7: Gain and phase responses of a rate filterbank.	146
Figure 5-8: Gain and phase responses of a scale filterbank.....	147
Figure 5-9: Seed functions.	150
Figure 5-10: Pole-zero (PZ) map of the rate filterbank.....	151
Figure 5-11: Pole-zero (PZ) map of the scale filterbank.	152
Figure 5-12: Quadrature mirror Hilbert transformer (QMHT) configuration.	157
Figure 5-13: Gains and phases of a 2 Hz QMHT using a 1 st -order LPF and HPF.....	157
Figure 5-14: Gains and phases of a 32 Hz QMHT using a 1 st -order LPF and HPF.....	158
Figure 5-15: Gain and phase responses of a modified rate QMHT.....	159
Figure 5-16: Gain and phase responses of a modified scale QMHT.....	159
Figure 5-17: CAR-Lite-A1 model circuit.	160
Figure 5-18: A1 neuron directionality filter.....	161
Figure 5-19: PZ map of a 2 nd -order AR with 9 bits coefficients.	163
Figure 5-20: PZ map of a 2 nd -order HPF with 16 bits coefficients.	164
Figure 5-21: Magnified region of the unit circle in the PZ map.....	165
Figure 5-22: FPGA architecture of the CAR-Lite-A1 model.	166
Figure 5-23: FPGA vector waveform of the CAR-Lite-A1 model.....	167
Figure 5-24: Degree of similarity between CAR-Lite-A1 model responses.	171
Figure 5-25: Degree of similarity between 2D summary profiles.....	171
Figure 5-26: CAR-Lite-A1 4D response.....	172
Figure 5-27: CAR-Lite-A1 response of a downward moving ripple signal.	177

Figure 5-28: CAR-Lite-A1 response of an upward moving ripple signal.....	178
Figure 5-29: Summary profiles calculated using sum-absolute operations.	180
Figure 5-30: CAR-Lite-A1 response of an FM signal.	182
Figure 5-31: CAR-Lite-A1 response of a log chirp signal.....	183
Figure 6-1: Starting point locations to calculate an autocorrelogram (AC).	192
Figure 6-2: Starting points comparison to calculate an AC.	192
Figure 6-3: Autocorrelogram (AC) response of an A4 piano note.	193
Figure 6-4: Temporal profile of an A4 piano note.	195
Figure 6-5: Temporal profile of an E4 piano note.	197
Figure 6-6: Temporal profile of a D#4 piano note.	199
Figure 6-7: Demonstration of temporal profile peak detection on FPGA.....	202
Figure 6-8: FPGA output vector waveform of the AC f_0 estimation and classifier.	203
Figure 6-9: Classification of musical notes in octave groups 2 and 3.....	206
Figure 6-10: Classification of musical notes in octave groups 4 and 5.....	206
Figure 6-11: Classification of musical notes in octave groups 6 and 7.....	207
Figure 6-12: Classification of musical notes across all three-octave groups.	207
Figure 6-13: AC representation of low pitch (C2) and high pitch (C7).	208
Figure 6-14: AGC effect on CAR-Lite-ACF response.	211
Figure 6-15: Pitch estimation and classification across multiple intensity levels.	212
Figure 6-16: Pitch estimation and classification for varying SNR levels.....	212
Figure 6-17: Mean of the standard deviation of classification accuracy scores.....	213
Figure 6-18: AC file sizes.	214
Figure 7-1: CAR-Lite-A1 summary profiles of an A3 piano note.	224
Figure 7-2: Approximation results of calculating correlation coefficients (CCs).....	227
Figure 7-3: FPGA output vector waveform of the timbre distance and KNN classifier.....	233
Figure 7-4: Algorithm for classifying musical instruments.	235
Figure 7-5: Confusion matrix from the classification of musical instruments.	239
Figure 7-6: AGC effect on CAR-Lite-A1 response.	244
Figure 7-7: Accuracy scores of varying intensity levels.	245
Figure 7-8: Accuracy scores of varying SNR levels.....	246
Figure 7-9: Mean of the standard deviation of the classification accuracy.	247
Figure 7-10: File sizes of the CAR-Lite-A1 output response.....	248
Figure A-1: Demonstration of the AGC algorithm on a speech signal.....	263

List of Tables

Table 2-1: Summary of computational models of the auditory pathway.....	21
Table 2-2: Summary of auditory pitch models.	35
Table 2-3: Summary of auditory timbre models.	42
Table 3-1: Performance of cochlear filters on FPGA.	65
Table 3-2: Sub-sampling factors in the SR-AN stage.	82
Table 3-3: CAR-Lite-SI model settings.	83
Table 3-4: General performance of cochlear filters.....	87
Table 3-5: Mean spike rate of each AN fibre.	89
Table 3-6: Coincidence matching ratio of a speech signal.....	95
Table 3-7: Coincidence matching ratio of a musical signal.	95
Table 4-1: Survey of computational resources.	106
Table 4-2: Settings of the CAR-Lite-ACF model.	107
Table 4-3: FPGA computational resources used by the CAR-Lite-ACF model.	121
Table 4-4: Phase insensitivity effect on an autocorrelogram.....	129
Table 5-1: 2 nd -order filter coefficients.	139
Table 5-2: A review of four filter configurations.....	148
Table 5-3: Latencies of modules in the CAR-Lite-A1 model.	169
Table 5-4: FPGA computational resources used by A1 models.....	169
Table 5-5: Input signal settings.	174
Table 6-1: Point-based hierarchically structured algorithms.	199
Table 6-2: Demonstration of the AC f_0 estimation and classification algorithms.	201
Table 6-3: Comparison of the results of classifying musical notes.....	215
Table 7-1: Twelve classes of musical instruments.....	222
Table 7-2: Computational resources used by the timbre distance and KNN classifier.	231
Table 7-3: Summary of classifying musical instruments with a linear classifier.....	240
Table 7-4: Summary of classifying musical instruments with a KNN classifier.....	241
Table 7-5: Comparison of the classification of musical instruments.....	250
Table 7-6: Features used for the classification of musical instruments.	250
Table B-1: Classification of musical notes for octaves 2 and 3.	265
Table B-2: Classification of musical notes for octaves 4 and 5.	266
Table B-3: Classification of musical notes for octaves 6 and 7.	266
Table C-1: Pitch estimation for musical notes without white Gaussian noise.....	267
Table C-2: Pitch estimation for musical notes with 20 dB SNR and AGC disabled.	269
Table C-3: Pitch estimation for musical notes with 20 dB SNR and AGC enabled.....	270
Table C-4: Pitch estimation for musical notes with 0 dB SNR and AGC disabled.	272
Table C-5: Pitch estimation for musical notes with 0 dB SNR and AGC enabled.....	273
Table C-6: Pitch estimation for musical notes with -20 dB SNR and AGC disabled.....	275
Table C-7: Pitch estimation for musical notes with -20 dB SNR and AGC enabled.	276
Table D-1: Linear classification of musical instruments (floating-point).....	279
Table D-2: Linear classification of musical instruments (fixed-point).	282
Table D-3: KNN classification of musical instruments (floating-point).	285
Table D-4: KNN classification of musical instruments (fixed-point).	288
Table E-1: Classification of musical instruments without noise.....	289
Table E-2: Classification of musical instruments at 20 dB SNR.....	291
Table E-3: Classification of musical instruments at 0 dB SNR.....	292
Table E-4: Classification of musical instruments at -20 dB SNR.....	294

1. Introduction

Carver Mead first suggested the term ‘neuromorphic’ for analogue circuits that mimicked biological systems. Since their inception, neuromorphic circuits, have also encompassed digital circuits as well as a mix of analogue-digital circuits into its paradigm. Biological systems that have been translated to neuromorphic systems include vision pathway (see), auditory pathway (hear), olfactory pathway (smell), and tactile pathway (touch). The first neuromorphic auditory system was an analogue electronic circuit developed by Lyon and Mead, which mimicked a mammalian cochlea [1]. The characteristics of this circuit were backed up by a computational model developed by Lyon [2]. Other notable works in analogue circuits by van Schaik [3], Hamilton [4] and others [5], [6] have shown that neuromorphic auditory systems provide an alternative and intuitive platform for studying biological auditory pathway in real time besides the conventional manner of using software-based computational auditory models [7].

Digital electronic circuits have also been used to build neuromorphic auditory systems [8]–[10]. Although analogue circuits are lower-powered and have smaller areas than the former, as well as having mismatches contributing to biological signal processing, they are not used widely as they do not perform robustly under changing environmental conditions as opposed to digital electronic circuits [11]. Digital electronic circuits range from general-purpose central processing units (CPU) on computers capable of running sophisticated auditory models to customised digital circuitry capable of running a scaled-down auditory model. The difference between these two circuit types is that the general-purpose CPU uses more computational resources such as memory and faster clock speed, but enables models to be customised quicker than a customised digital circuit. In contrast, the operation of a customised digital circuitry is often simulated on a computer before being fabricated as an application-specific integrated circuit (ASIC) chip. As its name suggests, ASIC chips are designed and developed for specific use in research as well as commercial applications. Some of these applications include hearing prostheses [12], coding strategies improvement [13] to increase the sensitivity of hearing prostheses to musical signals in addition to speech signals [14], [15], and audio computing processors for multimedia and entertainment [16].

One major advantage of ASIC chips is their low power consumption, which enables mobile devices to have long battery lifetimes and reduces the cost of power utilities for electronic devices connected to the power grid [17]. Another advantage is the small size dimensions of the chips, which enables portability of electronic devices due to their light weights. The auditory models running on ASIC chips should ideally be stripped of complex operations and maintain only basic operations to minimise the size and power consumed by the ASIC chip. However, this may mean sacrificing functionalities that may significantly lessen a model’s impact. To alleviate this situation, one can adopt the Occam’s Razor principle – when two theories can capably describe some observed data, the simpler theory is selected [18], [19]. Using Occam’s Razor principle on models [20], [21], digital neuromorphic auditory systems can be designed and developed using simplified abstract methods instead of complex operations. This notion means that functionalities that may otherwise be required to be omitted can be implemented in digital hardware with reduced computational resources as opposed to significant resources that are required using sophisticated methods.

1.1. Aims and Chapter Synopses

In this research project, I aim to develop and investigate the performance of hardware-implementable auditory models adapted from existing computational models. Here, hardware refers explicitly to a field-programmable gate array (FPGA), and the computational model refers to a biologically inspired auditory model implemented solely on software such as Matlab. The hardware used is an Altera Cyclone V GX starter kit because it is an affordable and off-the-shelf FPGA, which is readily acquirable for model reproducibility. FPGA implementation of the auditory models in this dissertation provides a feasible platform to demonstrate that these models are fully implementable on ASIC chips. Although an FPGA consumes more power and has more speed limitation than an ASIC chip, it is reconfigurable [22]. This attribute is advantageous when working with models that require constant tuning of variables. The FPGA enables changes to be introduced at a shorter development time than an ASIC chip. Moreover, the emphasis of the hardware implementation of the models is based on the optimisation of conventional algorithms from software models. Thus, the emphasis is on the comparison of power consumption between the hardware implementation of the optimised and conventional algorithms rather than between FPGA and ASIC implementations.

Since an FPGA has limited computational resources, including logic modules, registers, and clock speed, it is necessary to design auditory models to run on an FPGA with limited computational resources. Hence, this limitation enforces auditory models to be optimally designed to use as few computational resources as possible, to conserve FPGA logic elements. If this notion is adhered, then an ASIC chip implementation of the optimally designed circuit of an auditory model will be of small size and consume low power leading to running cost savings as opposed to the outcome of a non-optimally designed auditory model. Hence, Occam's Razor principle plays a significant role in the design of the auditory models, i.e. when two models can characterise some observed data satisfactorily, the model with 'simpler' algorithm is selected for implementation on hardware. Here, 'simpler' refers to algorithms requiring fewer computational resources to run on an FPGA as opposed to algorithms requiring significant computational resources to run on the same platform.

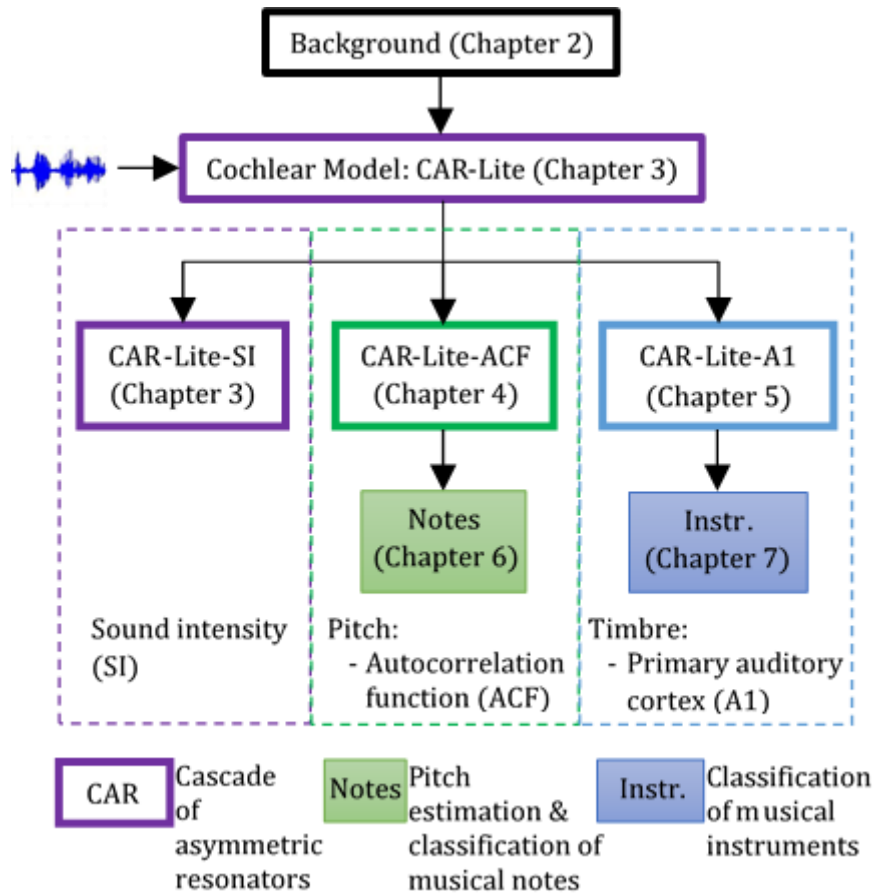


Figure 1-1: Flowchart of chapters in this dissertation.

The auditory models will include various stages of a mammalian auditory pathway capable of generating responses for pitch estimation as well as loudness and timbre representations. Figure 1-1 displays the chapters in the structure of this dissertation. The aims are further broken down based on the following chapter numbers:

Chapter 2: This chapter presents a survey of auditory computational models for hardware implementation. The models presented are based on three stages of the mammalian auditory pathway as follows: cochlear models, auditory pitch models, and auditory timbre models. The cochlea (inner ear) is the first interface in our hearing that receives sound with the exception of the outer and middle ear. It comprises the functionality of a basilar membrane, inner hair cells, and auditory nerves that converts sound to a 2D time-frequency representation and sends this information higher in the auditory pathway for processing. In the higher regions of the auditory pathway, pitch and timbre information are extracted, which are described by the auditory pitch models and timbre models, respectively. From the models reviewed for each of the three stages, one model is selected for hardware implementation.

Chapter 3: This chapter presents a simple cochlear model capable of running on hardware (FPGA). In the first half of this chapter, a cochlear model selected from chapter 2 is modified with reduced computational resources to run on hardware. This notion results in a cochlear model running at multiple sampling rates and using nine times fewer coefficients than a model running at a single sampling rate. The limitation of the dynamic range of the model is presented with musical signals at multiple intensity levels, and an automatic gain control

algorithm (AGC) is utilised to improve this dynamic range. In the second half of this chapter, the model is fitted with several biologically inspired algorithms to understand if the new cochlear model is capable of characterising auditory features. One algorithm is the use of binary spike trains at multiple firing thresholds for the representation of sound intensity to depict loudness information on hardware. The new model is tested with real-world signals such as speech and music. For both models, computational resource usage on hardware is also presented.

Chapter 4: This chapter presents a system that, using the new cochlear model introduced in the first half of chapter 3, is capable of extracting pitch information using time and frequency information of a sound signal on an FPGA-based (hardware-based) auditory model. This is achievable by using the cochlear model from the first half of chapter 3 with a pitch model reviewed in chapter 2 to produce a hardware-implementable model. The pitch model generates an autocorrelogram by using correlations over time per cochlear section for the entire range of cochlear sections. Computation of the autocorrelogram requires significant computational resources. A similar autocorrelogram can be generated using a binary spiking algorithm, as used in the model generating sound intensity response in the second half of chapter 3. This novel model uses fewer computational resources than originally defined in the computational model. Hence, this chapter also presents how many computational resources these novel algorithms conserve on hardware as opposed to the conventional computation of the original model to generate an autocorrelogram.

Chapter 5: This chapter presents the extraction of modulating envelopes of a sound signal on an FPGA-based (hardware-based) system. From the first half of chapter 3, the new simple cochlear model is used with a mammalian functional primary auditory cortical (A1) model (also known as auditory timbre model). The A1 model comprises modulation filterbanks that extract temporal and spectral envelope information from a sound signal as well as spectral directional envelope changes describing either frequency increase or decrease, which are all essential to timbre. Changes are required to accommodate this new model on hardware. Hence, the temporal and spectral filterbanks are replaced as the original filterbanks do not possess hardware-implementable attributes. As such, this chapter presents a survey of hardware-implementable filter configurations to be used in place of the original. The results of the new model are also presented, which includes its computational resource utilisation on hardware.

Chapter 6: In this chapter, two exercises are performed to understand how the hardware-based (FPGA-based) pitch model from chapter 4 performs with real-world sound signals. The first of the chapter presents algorithms for extracting pitch information represented by fundamental frequency estimation. A classification algorithm is also presented to determine if the estimated fundamental frequency matches the ground truth fundamental frequency of a musical note calculated with an equation. Both the fundamental frequency estimation and classification algorithms are implementable on hardware (FPGA). The second half of the chapter presents the results of the classification of musical notes from several musical instruments. Pitch is estimated from the autocorrelogram response of the model from chapter 4 before being classified. Then a comparison of classification accuracies based on software and hardware implementations of the models is made. To showcase the effects of the limitation and the improvement of the dynamic range of the model's responses on pitch estimation and classification performance, musical signals at multiple intensity levels are used with and without an automatic gain control algorithm (AGC). Performance impact

based on noise in the musical signals is also presented. Finally, the significant difference in memory sizes for storing the output responses of both the software- and hardware-implemented models are presented. This attribute is essential, as a reduced memory size lowers power consumption during runtime as well as operational costs corresponding to hardware and power utilities.

Chapter 7: This chapter demonstrates how the hardware-based (FPGA-based) mammalian functional auditory cortical model from chapter 5 performs with real-world sound signals by using its representation of musical notes from various musical instruments for the classification of these instruments. The responses of the cortical model are divided into two different sets based on different manufacturers. So, each set contains the same notes from the same instruments but from different manufacturers. The notes are further divided based on intensity and pitch levels. Two separate classification algorithms are then used to classify the musical instruments. One is a software-based classifier, and the other is a hardware-based (FPGA-based) classifier. Classification accuracies are presented based on the responses of the software- and hardware-implemented cortical models. Also, the impact of the variation of intensity levels of the musical signals on classification is presented similar to the classification of musical notes in chapter 6. This exercise is performed with the musical signals conditioned with and without an automatic gain control (AGC) algorithm. Classification performance is also presented with various noise levels added to the musical signals. Lastly, the difference in memory sizes for storing the output responses of both the software- and hardware-implemented models are presented because a lower memory usage reduces runtime power consumption and operational costs.

Chapter 8: This chapter presents a summary and conclusion of this dissertation, where the main results are summarised. Additionally, recommendations are also presented here for future research along with publications from work presented in this dissertation.

1.2. Bibliography

- [1] R. F. Lyon and C. Mead, "An Analog Electronic Cochlea," *IEEE Trans. Acoust.*, vol. 36, no. 7, pp. 1119–1134, 1988, doi: 10.1109/29.1639.
- [2] R. F. Lyon, "A Computational Model of Filtering, Detection, and Compression in the Cochlea," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82.*, 1982, pp. 1282–1285, doi: 10.1109/ICASSP.1982.1171644.
- [3] A. van Schaik, "Analogue VLSI Building Blocks for an Electronic Auditory Pathway," École Polytechnique Fédérale de Lausanne, 1997.
- [4] T. J. Hamilton, "Analogue VLSI Implementations of Two Dimensional , Nonlinear , Active Cochlea Models," The University of Sydney, 2008.
- [5] J. Lazzaro, J. Wawrzynek, and A. Kramer, "Systems Technologies for Silicon Auditory Models," *IEEE Micro*, vol. 14, no. 3, pp. 7–15, 1994, doi: 10.1109/40.285219.
- [6] R. Sarpeshkar, M. W. Baker, C. D. Salthouse, J.-J. Sit, L. Turicchia, and S. M. Zhak, "An Analog Bionic Ear Processor with Zero-Crossing Detection," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, 2005, pp. 2004–2005, doi: 10.1109/ISSCC.2005.1493877.
- [7] A. van Schaik, T. J. Hamilton, and C. Jin, "Silicon Models of the Auditory Pathway," in

- Computational Models of the Auditory System*, vol. 35, R. Meddis, E. A. Lopez-Poveda, R. R. Fay, and A. N. Popper, Eds. New York, Dordrecht, Heidelberg, London: Springer, 2010, pp. 261–276.
- [8] S. Mandal, S. M. Zhak, and R. Sarpeshkar, “A Bio-Inspired Active Radio-Frequency Silicon Cochlea,” *IEEE J. Solid-State Circuits*, vol. 44, no. 6, pp. 1814–1828, 2009, doi: 10.1109/JSSC.2009.2020465.
 - [9] M. Yang, C. Chien, T. Delbruck, and S.-C. Liu, “A 0.5 V 55 μ W 64 \times 2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing,” *IEEE J. Solid-State Circuits*, vol. 51, no. 11, pp. 2554–2569, 2016, doi: 10.1109/JSSC.2016.2604285.
 - [10] S. Wang, T. J. Koickalt, G. Enemali, L. Gouveia, L. Wang, and A. Hamilton, “Design of a Silicon Cochlea System with Biologically Faithful Response,” in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–7, doi: 10.1109/IJCNN.2015.7280828.
 - [11] Y. Xu, “A Digital Neuromorphic Auditory Pathway,” Western Sydney University, 2018.
 - [12] B. S. Wilson, E. A. Lopez-Poveda, and R. Schatzer, “Use of Auditory Models in Developing Coding Strategies for Cochlear Implants,” in *Computational Models of the Auditory System*, R. Meddis, E. A. Lopez-Poveda, R. R. Fay, and A. N. Popper, Eds. New York, Dordrecht, Heidelberg, London: Springer, 2010, pp. 237–260.
 - [13] D. Pressnitzer, J. Bestel, and B. Fraysse, “Music to Electric Ears: Pitch and Timbre Perception by Cochlear Implant Patients,” *Neurosci. Music II From Percept. to Perform.*, vol. 1060, no. 1, pp. 343–345, 2005, doi: 10.1196/annals.1360.050.
 - [14] M. Yitao and X. Li, “Music and Cochlear Implants,” *J. Otol.*, vol. 8, no. 1, pp. 32–38, 2013, doi: 10.1016/S1672-2930(13)50004-3.
 - [15] S. M. Prentiss, D. R. Friedland, T. Fullmer, A. Crane, T. Stoddard, and C. L. Runge, “Temporal and spectral contributions to musical instrument identification and discrimination among cochlear implant users,” *World J. Otorhinolaryngol. Neck Surg.*, vol. 2, no. 3, pp. 148–156, 2016, doi: 10.1016/j.wjorl.2016.09.001.
 - [16] S. Jones, R. Meddis, S. C. Lim, and A. R. Temple, “Toward a Digital Neuromorphic Pitch Extraction System,” *IEEE Trans. Neural Networks*, vol. 11, no. 4, pp. 978–987, 2000, doi: 10.1109/72.857777.
 - [17] D. Chinnery and K. Keutzer, “Introduction,” in *Closing the Power Gap Between ASIC and Custom: Tools and Techniques for Low Power Design*, D. Chinnery and K. Keutzer, Eds. NY, USA: SpringerScience, 2007, pp. 1–10.
 - [18] P. Gibbs, “What is Occam’s Razor?,” 1996. <http://math.ucr.edu/home/baez/physics/General/occam.html> (accessed Jul. 30, 2019).
 - [19] J. Sheffer, “Occam’s Razor,” *Biomed. Instrum. Technol.*, vol. 48, no. 2, p. 1, 2014.
 - [20] L. L. Beranek and T. J. Mellow, *Acoustics: Sound Fields and Transducers*. Academic Press, 2012.
 - [21] J. B. J. Smeets, E. Brenner, and J. Martin, “Grasping Occam’s Razor,” in *Progress in Motor Control*, D. Sternad, Ed. Springer, 2009, pp. 499–522.
 - [22] A. Amara, F. Amiel, and T. Ea, “FPGA vs. ASIC for low power applications,” *Microelectronics J.*, vol. 37, no. 8, pp. 669–677, 2006, doi: 10.1016/j.mejo.2005.11.003.

2. The Auditory Pathway: A Modelling Perspective

This chapter presents a review of the mammalian biophysical sensory and perceptual stages of the auditory pathway in three sections. Section 2.1 describes the early stage of audition, with two computational models of the mammalian auditory pathway. Section 2.2 covers pitch perception studies and models for calculating pitch information. Finally, section 2.3 covers timbre perception studies and models for calculating timbre features.

From the models reviewed in each section, a model is selected to be implemented on FPGA, which is covered from chapters 3 to 5. The capabilities of the pitch and timbre models described in chapters 4 and 5, respectively, are further explored in chapter 6 for the classification of monophonic musical notes and in chapter 7 for the classification of musical instruments.

2.1. A Survey of Auditory Pathway Models

Figure 2-1 illustrates a cutaway diagram of the frontend auditory pathway of a human, comprising an outer ear, a middle ear, and an inner ear (cochlea). Figure 2-2 illustrates the transmission of sound energy from the outer ear to the middle ear and to the inner ear, the latter known alternatively as a cochlea. These three components form the initial stages of the auditory pathway. In the following two subsections, a description of the auditory pathway is presented using two models with a focus on the initial stages for implementation on the FPGA. These models include the Matlab Auditory Periphery (MAP) model and a model comprising a cascade of asymmetric resonators with fast-acting compression (CAR-FAC). The MAP model is considered as it is used for extracting pitch information, which serves one of my research aims of extracting pitch from monophonic signals. The CAR-FAC model is reviewed as it has been developed primarily for machine hearing applications. Its design is ideal for bridging the gap between a software [1]–[3] and digital hardware implementation of a functional cochlear model [4]–[6].

Figure 2-3 depicts the auditory pathway (AP) stages covered by the two models, each demarcated with a unique coloured box. The intricacies of each stage within the models is explored in the next three subsections with subsection 2.1.3 discussing the model selected for FPGA implementation.

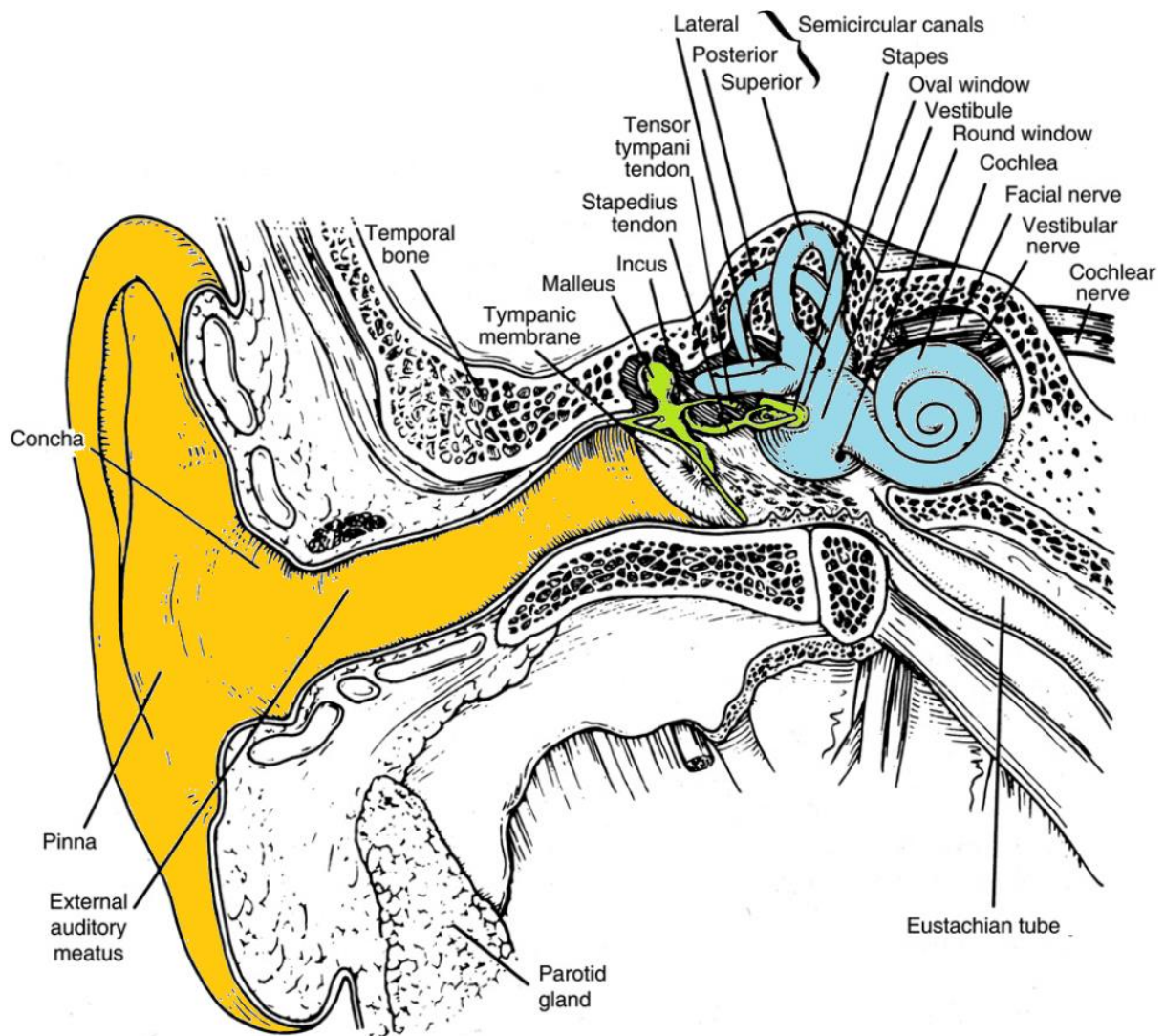


Figure 2-1: The outer ear (yellow), middle ear (green), and inner ear (blue) forming the initial stages of the auditory pathway of a human. Modified from Pickles [7].

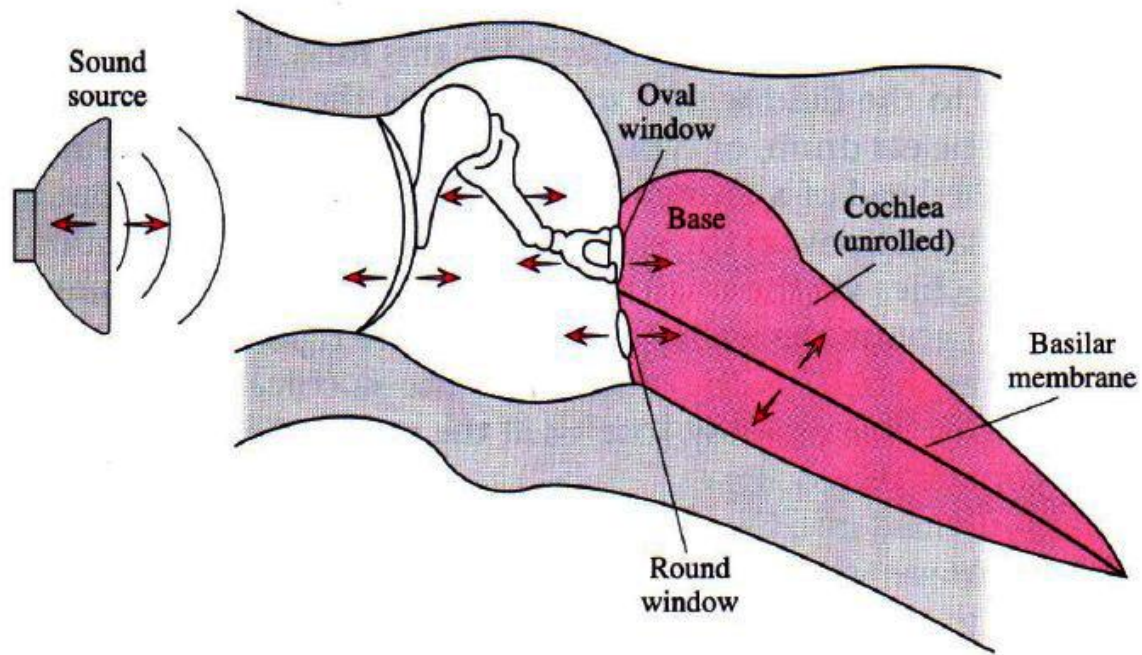


Figure 2-2: Transmission of sound energy from the outer ear to the basilar membrane (BM) in the inner ear via the tympanic membrane (depicted as a crescent shape) and the three bones in the middle ear. The BM is coiled in the inner ear as illustrated in Figure 2-1, but for illustration of sound transmission within the auditory pathway, the BM is shown as uncoiled in this figure. The region shaded in pink represents fluid-filled cochlea. Adapted from Matthews [8].

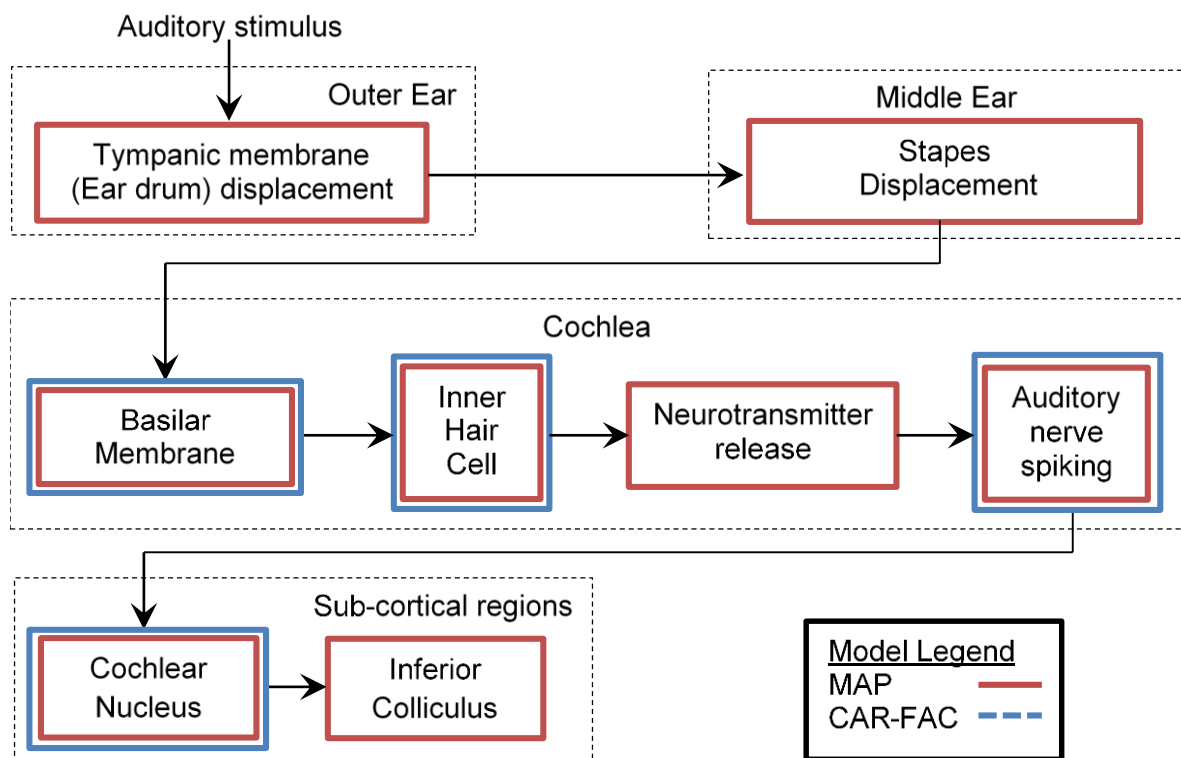


Figure 2-3: Monaural auditory models reviewed in this chapter.

2.1.1. MAP Model

The Matlab Auditory Periphery (MAP) model is a biophysical model describing the mammalian auditory pathway, which has been developed by Meddis and his colleagues at the University of Essex in the UK. Besides comprising the outer and middle ear stages, the model has a unique BM filter that characterises auditory phenomenon such as two-tone suppression, whereby the presence of a salient component in a stimulus reduces the responses of a less salient component in the same stimulus. This characteristic is advantageous in increasing the contrast between the two components in the context of sound source segregation [9]. Another essential trait of the model is auditory nerve spiking which is beneficial in explaining pitch and timbre in terms of spike rate and spatial placement in the auditory nerves.

Sound arriving at the outer ear is channelled via the ear canal and vibrates the tympanic membrane (eardrum) [7]. The outer ear amplifies sound at a specific frequency range to the tympanic membrane with the aid of the pinna and the concha [10], as illustrated in Figure 2-1. The MAP model characterises this increased pressure effect on the tympanic membrane displacement in the range of 1 kHz to 4 kHz in two stages. Firstly, a sound signal is input to a 2nd-order bandpass filter made of two parallel branches. One branch contains a 1st-order low-pass filter (LPF) with a cut-off of 1 kHz, and the other branch has a high-pass filter (HPF) with a cut-off of 4 kHz. In the second stage, the resonances in this range are added to the original sound.

The tympanic membrane vibration displaces three bones held in conjunction in the middle ear comprising malleus, incus, and stapes [7]. The middle ear functions as a mechanical impedance matching transformer that relays sound signals to a higher impedance fluid-filled cochlea. It is an essential feature without which much of the sound is reflected out to the outer ear. The displacement of the three bones in the middle ear is modelled using an LPF with a 50 Hz cut-off that converts sound pressure from the tympanic membrane stage to displacement. The LPF is cascaded with an HPF with a cut-off at 2 kHz to reflect the stiffness of the basilar membrane (BM) as the input sound is transmitted from the stapes bone to the cochlea [11]. Figure 2-4 illustrates the outer and middle ear stages in the MAP model.

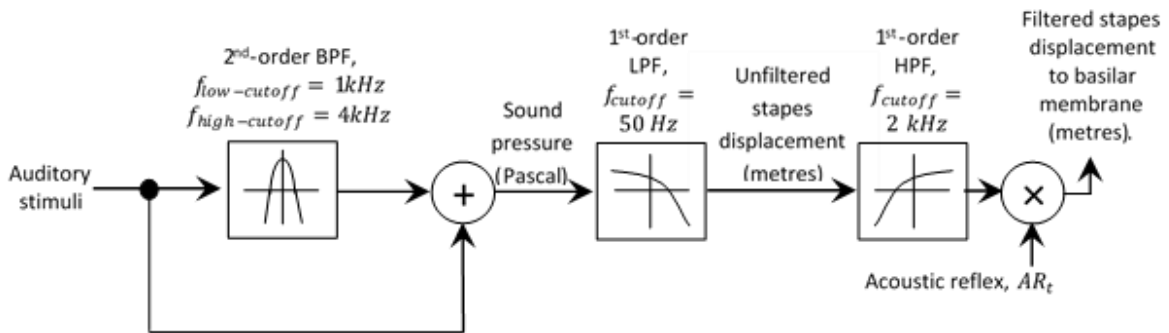


Figure 2-4: Outer and middle ear filters characterised in the MAP model. Adapted from Singh [12].

The mechanical motion of the stapes, which is dependent on stimulus frequencies [13], propagates the mechanical sound vibrations to the oval window disturbing the cochlea fluids within the cochlea. This characteristic mechanically influences the basilar membrane (BM) and induces a travelling wave along its coiled trapezoidal length of approximately 35mm for

humans, as illustrated in Figure 2-5. The BM is thick at the base (connected to the oval window) and gradually thins at the apex. High frequency sound mechanically vibrates the BM closer to the base, while low frequency mechanically vibrates the BM closer to the apex. Hence, for sounds with both high and low frequency, the vibrations start at the base. They travel to specific locations along the BM with increasing amplitudes before ending at the apex. Each of these locations on the BM corresponds and reacts to a specific frequency component present in the sound. Every BM location along the length of the BM has a bell-shaped curve response with the peak of the curve corresponding to a unique best frequency (BF) and bandwidth. Each response slope has a gradual pre-BF rise and a steep post-BF fall and its sidebands overlap with response curves of adjacent BF sites [14]. Hence, the BM is considered as a spectrum analyser, where the travelling waves capture the attributes of a time-varying input sound and represent them in two dimensions (2D) in time and frequency [15].

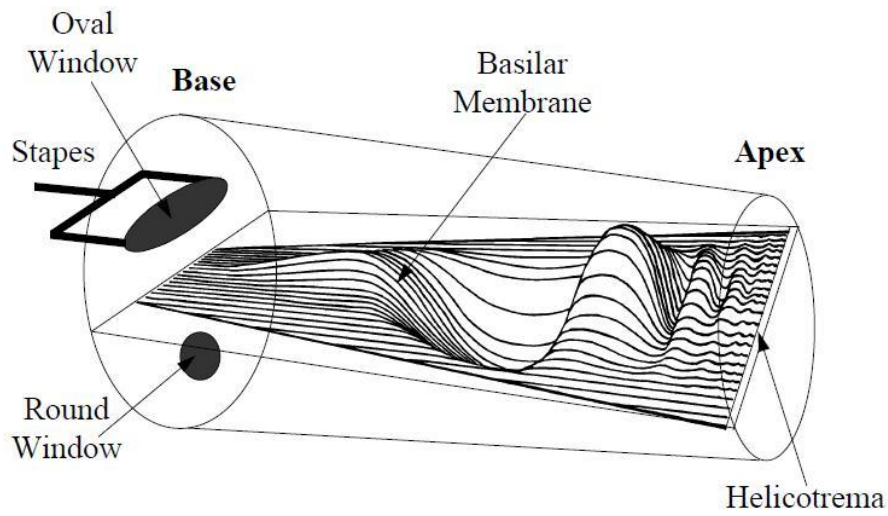


Figure 2-5: Travelling wave of the basilar membrane (BM) from the base to the apex. Adapted from van Schaik [16].

In the MAP model, BM displacement is modelled using a bank of dual resonant nonlinear (DRNL) filters [17] as displayed in Figure 2-6. Each DRNL filter has two parallel paths comprising a linear and nonlinear branch. The linear pathway has a cascade of three 1st-order gammatone filter with an impulse response of:

$$h(t) = kt^{n-1}\exp(-2\pi Bt) \cos(2\pi f_c t + \varphi) \quad (2-1)$$

where n is the filter order; B is the filter bandwidth; φ is the filter phase; k is the filter gain.

The nonlinear pathway has two sets of three cascaded 1st-order gammatone filters such as the one for the linear pathway, i.e. two 3rd-order gammatone filters. In between these two 3rd-order filters in the nonlinear pathway, there is an input level-dependent and memoryless compressive function, which results in a nonlinear signal:

$$h(t) = \text{sign}[x(t)] \cdot \min[a|x(t)|, b|x(t)|^c] \quad (2-2)$$

where $x(t)$ and $y(t)$ are the input and output of the nonlinear function; a , and b are frequency dependent constants; c is a constant set at 0.25. The two pathways are summed,

resulting in BM displacement. The advantage of the DRNL is its capability to characterise level dependence depicting shifts in best frequency (BF) responses based on sound intensity. At very low and high sound levels, the linear pathway dominates the DRNL responses while outside this range, the nonlinear pathway dominates the responses.

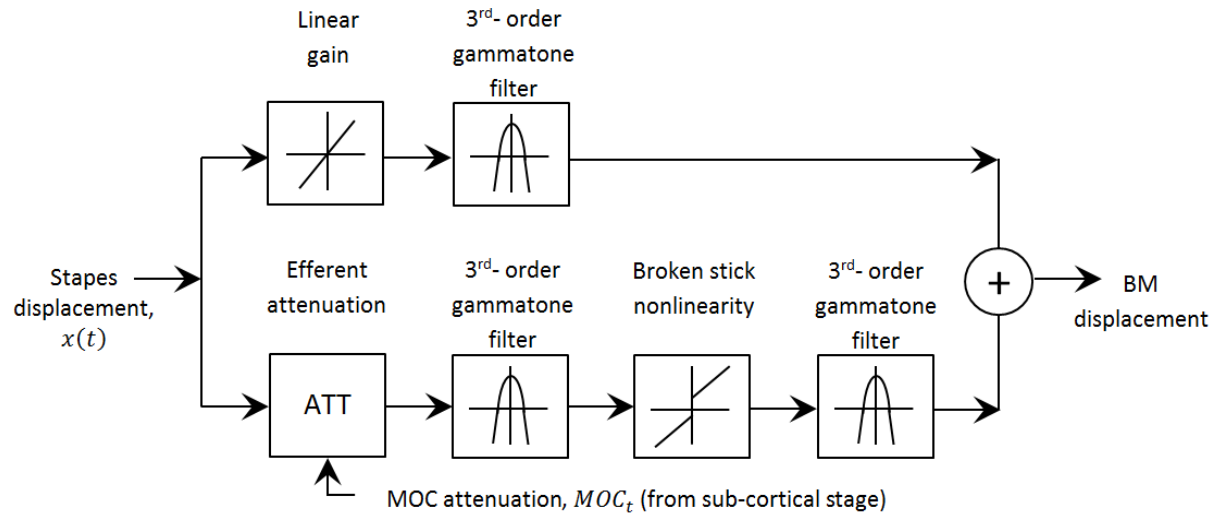


Figure 2-6: A dual resonant nonlinear (DRNL) filter that characterises a discrete point along the basilar membrane. The top parallel branch is the linear pathway, and the bottom parallel branch is the nonlinear pathway. Adapted from Meddis et al. [2].

In humans, there are approximately 3,500 inner hair cells (IHC) and 12,000 outer hair cells (OHC) situated along the length of the BM spanning from its base to its apex [18]. The IHCs transmit the mechanical vibration information of the travelling wave on the BM to the auditory afferent nerve (AN) fibres in the spiral ganglion, where the information is then transmitted to the auditory brainstem [9]. On top of an IHC, there is a bundle of hair-like structure of gradually increasing length called cilia. The travelling wave of the BM deflects the cilia on IHCs. When the cilia move in the direction of the longest cilium strand, ions flow into the IHC via the tips of the cilia. When the cilia move in the direction of the shortest cilium strand, potassium ions are prohibited from flowing into the IHC [19]. The high concentration of potassium in the ions flowing into an IHC leads to a rise of intracellular potential in the IHC. This rise enables the release of neurotransmitters from the base of the IHC and enables the excitation of auditory neurons, thus generating spike trains as seen in Figure 2-7. While the IHCs detect the motion of the BM, the OHCs adjust the motion of the BM by efferent input connections descending from the higher regions of the auditory brainstem to the cochlea. In the presence of loud sounds, the OHCs reduce the amplitudes of the BM travelling waves and in the presence of soft sounds, the OHCs increase the amplitudes of the BM travelling wave [20], [21].

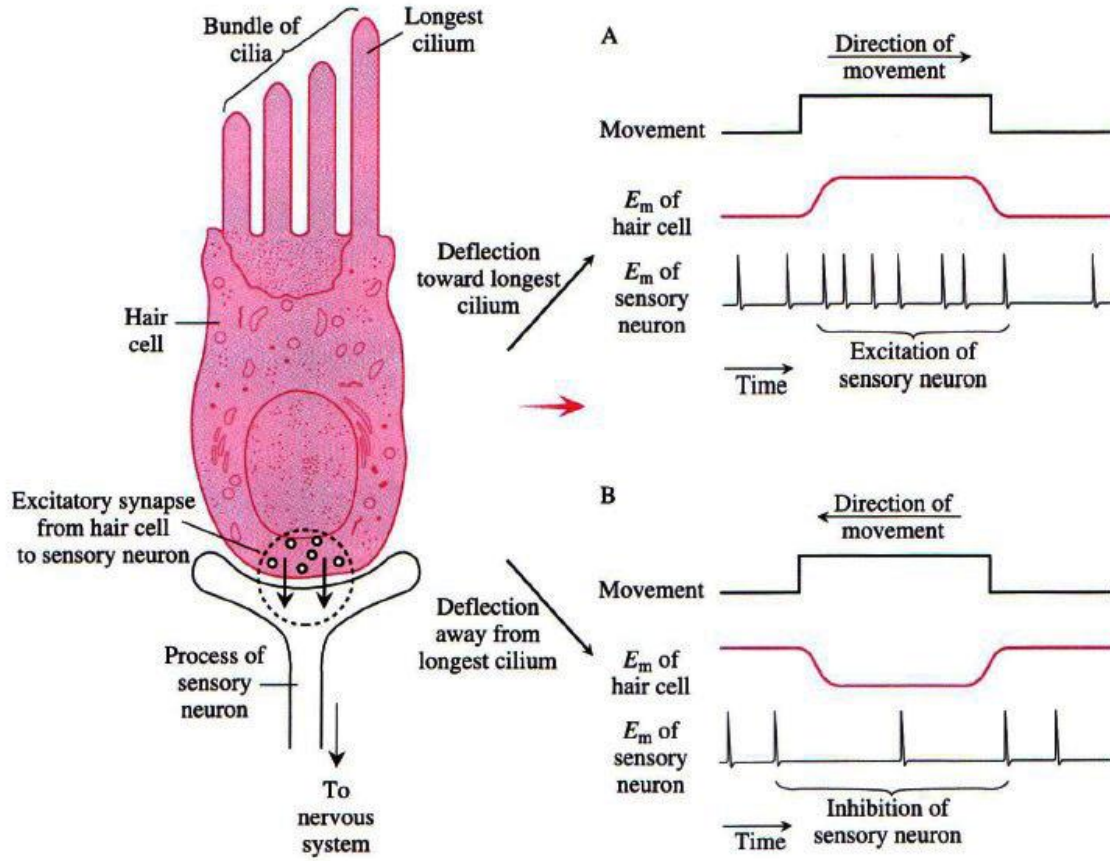


Figure 2-7: The motion of a bundle of cilia on top of an inner hair cell (IHC) [in pink] (A) towards the longest cilium strand leading to an increase in action potentials (voltage spikes) output from an auditory neuron (AN) fibre due to a build-up of potassium ions in the IHC; (B) towards the shortest cilium leading to a decrease in action potentials output from an AN fibre due to the stoppage of potassium ion flow into the IHC. Adapted from Matthews [8].

In the MAP model, the output of the DRNL filter is the input to a biophysical model of an IHC [1]. The first stage includes a high-pass filter (HPF) that characterises fluid-cilia coupling describing IHC cilia bundle motion in phase with BM displacement at high frequencies and cilia motion in phase with BM velocity at low frequencies. The cilia displacement affects the potassium ion levels in the IHC, which in turn affects the intracellular potential. When the cilia deflect to the direction of its longest cilium strand, incoming potassium ions increase intracellular potential. The deflection in the opposite direction stops the flow of incoming potassium ions, decreasing intracellular potential. This behaviour is modelled with an analogue circuit and the intracellular potential, v_m is, calculated using Kirchhoff current law by rearranging the following formula accordingly in terms of v_m :

$$C_m \frac{dv_m(t)}{dt} + G(u)(v_m(t) - E_t) + G_k \left(v_m(t) - \left(E_k + E_t \frac{R_p}{R_t + R_p} \right) \right) = 0 \quad (2-3)$$

where C_m is the IHC capacitance at 4 pF; G_k is potassium conductance at 20 nS; E_t and E_p are endocochlea and potassium potentials, respectively; R_t and R_p are epithelium and endocochlea resistances, respectively.

Neurotransmitters are released from the base of an IHC across a small area known as synaptic cleft to the auditory nerve. This release is dependent on the intracellular potential of the IHC and the release of calcium ions [22]. The calcium current is a measure of the concentration of calcium ions, which is analogous to intracellular potential and is determined by:

$$I_{Ca}(t) = G_{Ca}^{max} \cdot m_{I_{Ca}}^3(t) \cdot (v_m(t) - E_{Ca}) \quad (2-4)$$

where E_{Ca} is the reversal potential of calcium; G_{Ca}^{max} is the calcium conductance in the synapse and; $m_{I_{Ca}}$ is the ratio of opened calcium channels. The calcium concentration variable, $[Ca^{2+}](t)$ is established with a first-order low-pass filter (LPF):

$$\tau_{Ca} \frac{d[Ca^{2+}](t)}{dt} + [Ca^{2+}](t) = I_{Ca}(t) \quad (2-5)$$

where τ_{Ca} is the filter time constant. The neurotransmitter release rate can then be calculated as a probability variable by:

$$k(t) = \max\left([Ca^{2+}]^3(t) - [Ca^{2+}]_{thr}^3, 0\right) \quad (2-6)$$

where $[Ca^{2+}]_{thr}$ is a threshold constant; z is a scalar for converting calcium concentration levels to release rate.

Neurotransmitter flow across the synapse and evoke a corresponding auditory nerve to fire an action potential. This flow is modelled as a bidirectional flow of neurotransmitter at the synaptic cleft [23] as illustrated in Figure 2-8. It relies on the availability of a finite amount of neurotransmitter made available by the reuptake process from the synapse to the IHC and new neurotransmitter from a neurotransmitter factory that compensates lost ones across the synapse. Synaptic adaptation occurs when there are insufficient neurotransmitter in the free transmitter pool to be released, rendering the auditory nerve unable to fire [24]. The amount of neurotransmitters in the free transmitter pool is defined by:

$$\frac{dq}{dt} = y(1 - q(t)) + xw(t) - k(t)q(t) \quad (2-7)$$

where $q(t)$ is a time-varying amount of neurotransmitter in the free transmitter pool; $w(t)$ is a time-varying amount of neurotransmitters in the reprocessing store; x is the transfer rate between the reprocessing store and the free transmitter pool; $1 - q(t)$ is the new neurotransmitter release rate from the factory; $k(t)$ is the neurotransmitter release rate derived from equation (2-6). The amount of neurotransmitter in the synapse is determined by the difference between the time-varying release and the numbers lost as well as recycled ones:

$$\frac{dc}{dt} = k(t)q(t) - lc(t) - rc(t) \quad (2-8)$$

where l is the amount of neurotransmitters lost in the synaptic cleft; r is the reuptake (recycle) rate of the neurotransmitters from the synapse to the reprocessing store. The amount of neurotransmitters in the reprocessing store is defined by:

$$\frac{dw}{dt} = rc(t) - xw(t) \quad (2-9)$$

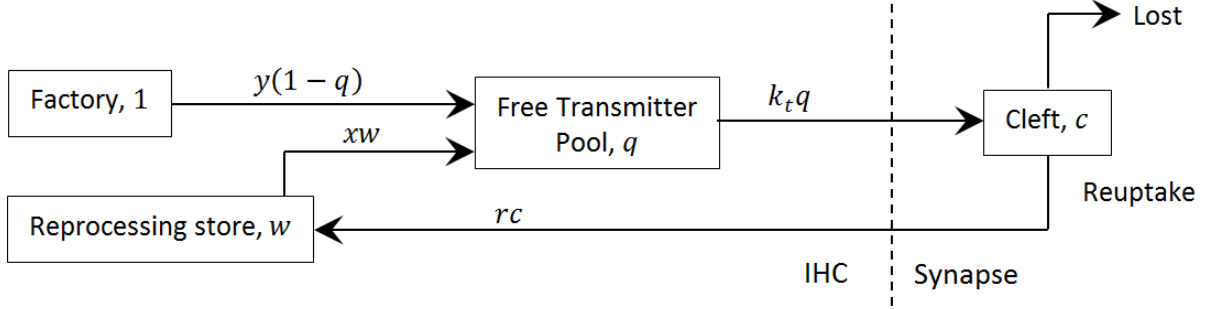


Figure 2-8: Neurotransmitters release model. Adapted from Meddis et al [23].

Neurotransmitters release generates a spike in an auditory nerve (AN) fibre [25]. There are two manners of generating spikes: a quantal model resulting in precise spike generation with high computational cost and a probabilistic model resulting in spikes approximation at low computational cost. The probabilistic model is described herein over the quantal model to maintain low computational cost. The amount of neurotransmitters residing in the synapse determines the spiking rate at the auditory nerve:

$$AN_{fr} = \frac{c(t)}{dt} \quad (2-10)$$

The refractory period is 0.75 ms, which indicates that a spike signal can only be generated after 0.75 ms from the hyperpolarisation of the preceding spike. The probability of occurrence of a spike is dependent on the release of a neurotransmitter and if $p(t)$ is larger than a random number between 0 and 1:

$$p(t) = 1 - c_r \cdot \exp(-(t - t_l - R_A)/s_r) \quad (2-11)$$

where c_r is the maximum relative refractory period at 0.55 ms; R_A is the absolute refractory period at 0.75 ms; t_l is the time of occurrence of the preceding spike; s_r is the refraction time constant at 0.8 ms.

The generated spikes are used as part of a hypothetical model of a mammalian sub-cortical region for the gain control of stapes motion at the middle ear as well as the basilar membrane motion via the outer hair cell. The subsequent destinations of the spike trains from the AN fibres are the ventral cochlear nucleus (VCN) and inferior colliculus (IC), which are subjected to temporal and rate modulation transfer functions (MTFs) [26]. Firstly, the spike trains from the AN fibres are low-pass filtered with temporal-MTF. After that, another temporal-MTF representing VCN chopper units condition the signals. At this stage, low-level signals are low-pass filtered, and for other levels, they are bandpass filtered. At the IC stage, the signals are conditioned by a final temporal-MTF, whereby low-level signals are low-pass filtered and medium and high-level signals are filtered with a broadly tuned bandpass filter.

Additionally, a rate MTF shapes the signals with sharply tuned bandpass filters at low and medium signal levels. At high levels, flat-shaped bandpass filters shape the signals.

Low spontaneous rate (LSR) fibre output signals from the IC stage are summed, and smoothed with an LPF and finally scaled to generate an acoustic reflex gain, AR_t , which is applied to the stapes displacement calculation at the middle ear stage:

$$AR_t = 1 - 0.008z_t^{(1)} \quad (2-12)$$

where $z_t^{(1)}$ is the summed IC output filtered through a LPF with a 0.6 Hz cut-off. Alternatively, the summed LPF-smoothed, and scaled IC stage output signal is used to generate a medial olivo-cochlear (MOC) reflex gain, MOC_t , which is applied as a feedback to attenuate BM displacement via the outer hair cell (OHC) [27]. This gain is applied to the start of the nonlinear pathway of the DRNL filter, as illustrated in Figure 2-6 and is calculated as:

$$MOC_t = 1 - 0.00625z_t^{(2)} \quad (2-13)$$

where $z_t^{(2)}$ is the summed IC signal filtered through a LPF with a cut-off at 6 Hz.

2.1.2. CAR-FAC Model

The cascade of asymmetric resonators with fast-acting compression (CAR-FAC) model encompasses BM, IHC, outer hair cell (OHC) and automatic gain control (AGC) segments. It does not characterise the outer and middle ear like the more biologically plausible MAP model. Instead, it emphasises on the characteristics of critical functional components of the cochlea (inner ear) and the cochlear nucleus crucial for extracting features from a sound signal.

The CAR-FAC model can be divided into two halves, as displayed in Figure 2-9: (a) CAR (b) FAC [3]. The CAR model is the primary interface to an incoming sound signal. It characterises the BM motion travelling only in the forward direction from the base of the cochlea to its apex and not its reflective motion travelling in the opposite direction. A transmission-line model characterises such bidirectional BM motions. Hence, the CAR model is an abstract of a transmission-line structure [28]. The CAR model is capable of depicting wave mechanics more accurately and efficiently, albeit with more complexity than a gammatone filterbank such as the ones used in the DRNL filterbank of the MAP model [29]. Each filter section is characterised by a pole-zero-filter-cascade, which is a 2nd-order filter with two poles and two zeros and has the following transfer function:

$$H(z) = \frac{Y}{X} = g \left(\frac{z^2 + (-2a + hc)rz + r^2}{z^2 - 2arz + r^2} \right) \quad (2-14)$$

where x and y are the inputs and outputs, respectively; a and c are the real and imaginary components of a complex signal represented by $z = a \pm jc$, whereby $a = \cos \theta_R$ and $c = \sin \theta_R$; θ_R is the normalised pole ringing frequency or the pole angle in the z -plane; g is the overall gain; h represents the proximity of poles from zeros, i.e. low h means zeros are closer to poles resulting in a higher degree of asymmetry in response peak and vice versa. Generally, h is set to c resulting in zeros maintained at approximately half octave above pole frequencies.

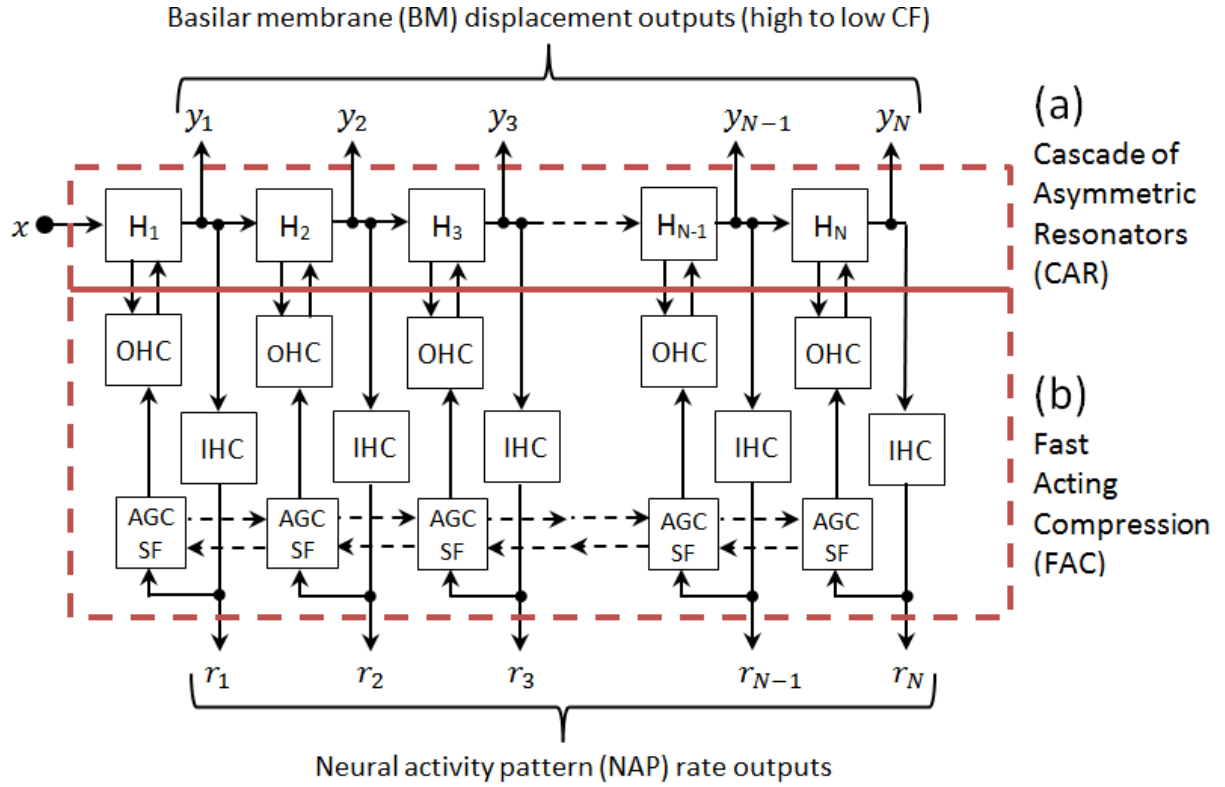


Figure 2-9: The CAR-FAC model. Adapted from Lyon et al. [30].

The travelling wave amplitudes of the BM generated from the CAR model is modulated by a fast-acting compression (FAC), referring to rapid BM gain damping. This characteristic is achieved by modulating r in the transfer function of the CAR model in equation (2-14) by the outer hair cell (OHC) block, as illustrated in Figure 2-10. The rate of change of the internal state of a linear filter in the CAR model, v , is varied by a nonlinear function, NLF and scaled by gains d_{rz} and $1 - b$. The term, d_{rz} , defines the rate at which NLF influences the pole radius r , thereby resulting in a compressive gain change in the linear filter of the CAR model while the term, $1 - b$, is an AGC feedback that is capped at 1 indicating saturation in the hair cell. The NLF is characterised by an inverse-square law that saturates at zero:

$$NLF(v) = \frac{1}{1 + (v \cdot scale + offset)^2} \quad (2-15)$$

where $scale = 0.1$ and $offset = 0.04$. The saturation of the NLF is analogous to transduction with sigmoidal nonlinearity whereby, the response of the NLF approaches zero as it saturates either positively or negatively. This saturation describes two-tone suppression, whereby one tone suppresses another tone [31]. Therefore, the OHC exhibits active undamping whereby, it increases its energy feedback to the BM exciting the magnitude of the travelling waves at low levels, and for loud sounds, the OHC suppresses the magnitude of the travelling waves.

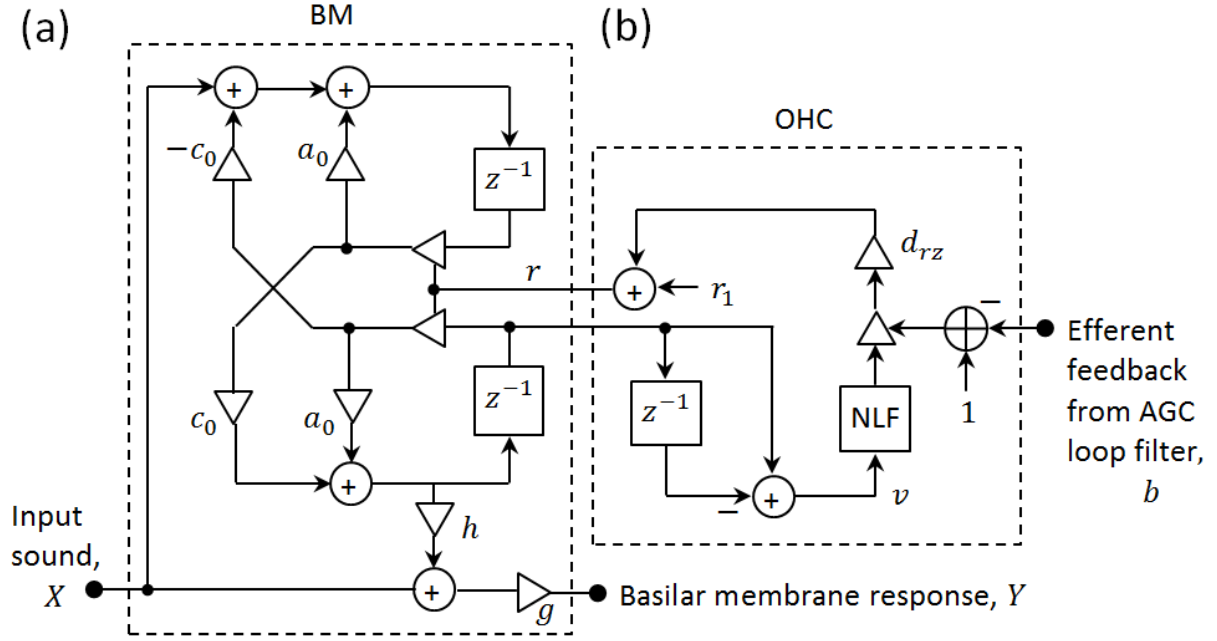


Figure 2-10: The CAR model representing (a) travelling wave response from a discretely located spot along the BM appended with (b) OHC algorithm. Adapted from Lyon et al. [32].

The output of the CAR model is conditioned by an IHC algorithm illustrated in Figure 2-11. The IHC is modelled with four first-order infinite impulse response (IIR) low pass filters (LPF). The first LPF implements a high pass filter (HPF) subtracted from unity. It serves the purpose of removing quadratic distortion below 20 Hz that mimics the helicotrema suppressing BM mechanical wave propagation at the apex. The motion of hair-like cilia bundle motion atop an IHC, which is affected by the BM travelling wave, is nonlinear. It is depicted as an NLF (nonlinear function) block in Figure 2-11 and has different characteristics from the NLF defined by equation (2-15) for the OHC. The IHC nonlinearity is defined by a soft rectifying rational-function sigmoid characterised by a cascade of equations from (2-16) to (2-19). The first equation in the cascade is a half-wave rectified (HWR) form of the high-pass filtered BM motion, x , which signifies the unidirectional effect of the deflection of the cilia atop an IHC to the longest cilium:

$$z = \text{HWR}(x + 0.175) \quad (2-16)$$

The HWR output, z , is then placed in a rational function comprising cubic polynomials, which indicates the electrical conductance of an IHC:

$$g = \frac{z^3}{z^3 + z^2 + 0.1} \quad (2-17)$$

The conductance, g , is scaled by a gain defined by the voltage across a capacitor, v , which produces an output current, y :

$$y = gv \quad (2-18)$$

The output current, y , is then used for updating the capacitor voltage, v_+ :

$$v_+ = v - c_{OUT}y + c_{IN}(1 - v) \quad (2-19)$$

where c_{OUT} and c_{IN} are the discharge and recharge rates of a capacitor used in a Schroeder-Hall hair cell analogue model [33], respectively. The output current, y is smoothed by two LPFs generating neural activity patterns that are potentially observed at the auditory nerve.

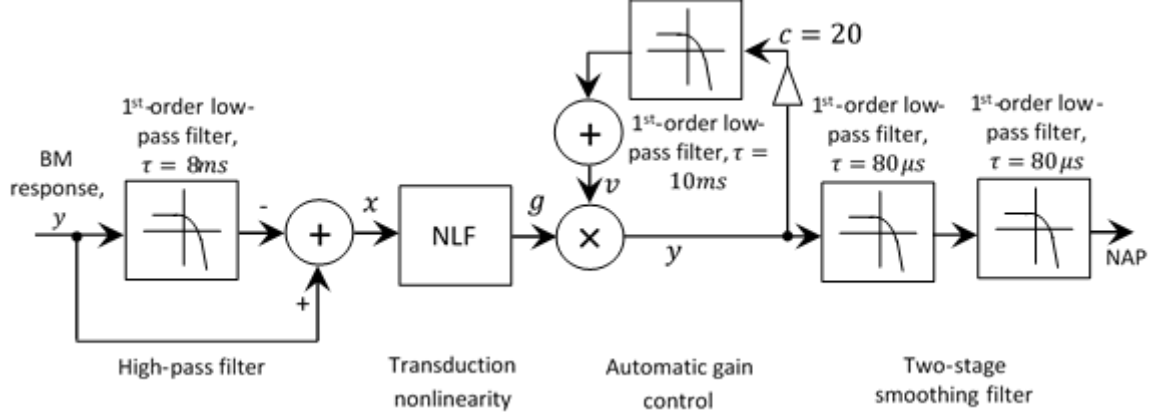


Figure 2-11: CAR-FAC digital inner hair cell (IHC) model. Adapted from Lyon et al. [34].

The output of the IHC algorithm drives an automatic gain control (AGC) loop-filter representing the mediating gain response of the medial olivo-cochlear (MOC). It consists of four LPFs with four distinct time constants illustrated in Figure 2-12(a). These are known as temporal smoothing filters (SFs), structured in a parallel-of-cascades form to ensure that the response of the AGC is fast, stable and non-ringing over a wide range of conditions. The transfer function is given by:

$$H(s) = \frac{a_3s^3 + a_2s^2 + a_1s + a_0}{4096 + \tau_1^4s^4 + 5440\tau_1^3s^3 + 1428\tau_1^2s^2 + 85\tau_1s + 1} \quad (2-20)$$

where τ is the time constant of an LPF. Solving equation (2-20) for poles and zeros location result in:

$$\text{Poles location: } \left[-\frac{1}{\tau_1}, -\frac{1}{4\tau_1}, -\frac{1}{16\tau_1}, -\frac{1}{64\tau_1} \right] \quad (2-21)$$

$$\text{Zeros location: } \left[-\frac{1}{1.54\tau_1}, -\frac{1}{7.20\tau_1}, -\frac{1}{24.59\tau_1} \right] \quad (2-22)$$

With the zeros and poles interleaved, the filter has a moderate roll-off slope and phase shift over a wide frequency range. Also, the AGC loop-filter operates at lower sampling rates than the CAR model. Through decimation, the fastest AGC stage is only updated every eighth sample period, and subsequent stages are updated less often at factors of two. The AGC filter-loop also encompasses spatial smoothing filters, where the gains of adjacent channels are maintained close to each other while reducing its dynamic range. A smoothing filter is implemented with a 3-point finite impulse response (FIR) smoothing filter, which is integrated with the earlier mentioned first-order temporal smoothing LPF as displayed in Figure 2-12(b). Coefficients c is the temporal smoothing LPF time constant, and g is the

input gain, while a and b are weights that define the degree of influence of the left and the right neighbouring temporal smoothing LPFs.

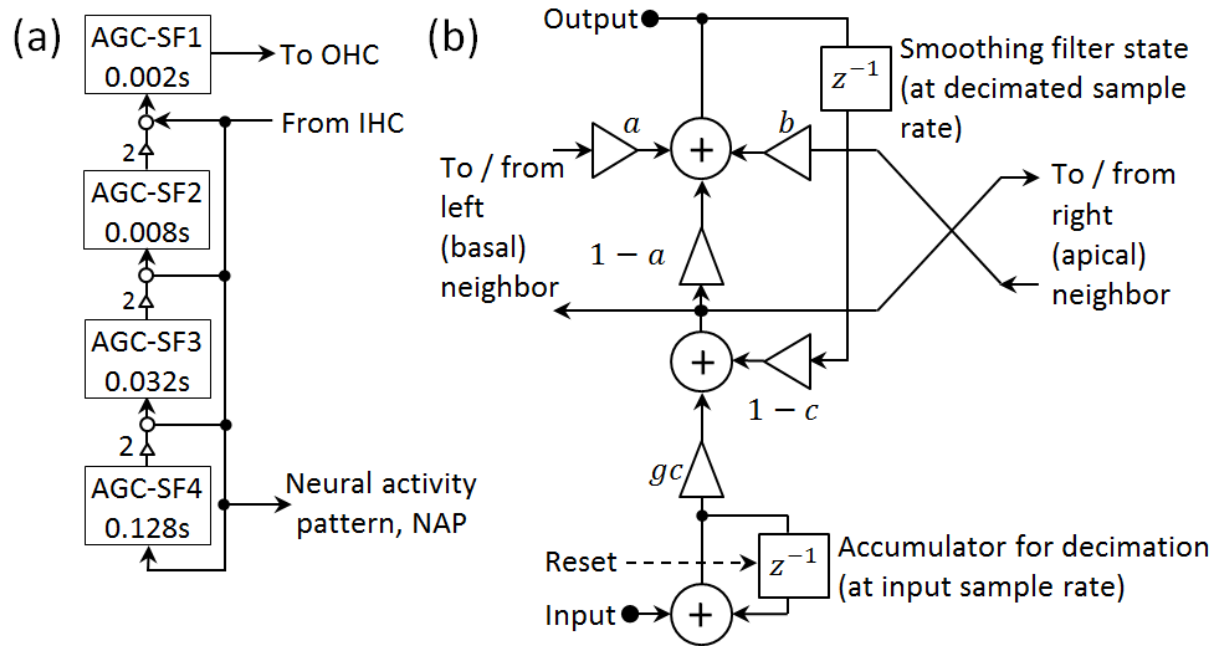


Figure 2-12: (a) Automatic gain control (AGC) loop-filter of the CAR-FAC model. (b) Connection of each of the four nodes in (a) is made to a spatial smoothing filter. Adapted from Lyon [35].

2.1.3. Model Selection

Table 2-1 summarises the features of the two auditory models presented above. To select one of the two auditory models to be implemented on FPGA, the Occam's Razor principle is invoked, which states that when two models are capable of describing some observed data, the less complicated model is selected. The following compares the features of the two models to understand the complexities of both models:

- 1) A DRNL filter in the MAP model requires eight parameters to generate the output of one cochlear section, while a CAR filter in the CAR-FAC model requires only six.
- 2) According to Saremi [36], the CAR-FAC model is capable of reproducing experimental data more frequently than the MAP model.
- 3) Also, according to Saremi [36], the CAR-FAC model has quicker computation time than the MAP model.

The three factors above place the CAR-FAC model in an advantageous position over the MAP model over selection. However, the CAR-FAC model has already been implemented on the FPGA [4]–[6], [37], which rules it out of contention for implementation on an FPGA. The MAP model is only implemented in software in Matlab [38] and C [39], [40] but not on FPGA. Nonetheless, according to Saremi [36], the linear gammatone filter, which makes up the DRNL filter structure in the MAP model, can capture excitation patterns of a mammalian cochlea and is, therefore, capable of representing features in a sound signal. Inspired by this finding, a similar arrangement can be made by using only the linear filters in the CAR segment of the CAR-FAC model. Although the CAR model has already been implemented on FPGA [4], further modifications are performed to simplify its design and more importantly, reduce its size in terms of filter coefficients utilisation and storage in

accordance with the Occam Razor's principle. The details of this implementation are presented in chapter 3.

Cochlear Models	Features
MAP	BM modelled with parallel gammatone filterbank with memoryless nonlinear compression [17].
	Nonlinear biophysical cochlea model capable of characterising psychophysical features [24].
	Capable of computing pitch from simple sound [41].
	Real-time software implementation [39], [40] bridges the gap to hardware implementation.
CAR-FAC	BM modelled with “transmission-line” cascade filterbank simulating travelling wave of BM [3].
	Amplitude-level dependent automatic gain control (AGC) to regulate the amplitude of BM response [42].
	Accounts for psychophysical features such as two-tone suppression and cubic distortion tone and as such can potentially be used for general sound processing as well as biologically inspired research.
	Hardware implementation is available [4]–[6], [37] and can be integrated with subcortical and cortical filters.

Table 2-1: Summary of computational models of the auditory pathway.

2.2. Auditory Pitch

Pitch is defined as the auditory sensation, where sounds are ordered on a scale used for melody in music. Alternatively, the pitch of a sound may also be described by the frequency of a pure tone at a specific sound pressure level that is perceived by a listener to match the perceived pitch of a complex sound [43]. In the next two subsections, two aspects of pitch are presented: studies of human pitch perception and pitch perception models.

2.2.1. Pitch Perception

Pitch perception studies are reviewed in this subsection to appreciate the inner workings of pitch perception models. The following paragraphs offer a brief history of early pitch perception experiments, which advocates algorithms used in current pitch perception models.

One early pitch perception experiment was conducted by Seebeck, who used sirens to produce pitched sounds [44]. His first of two experiments included rotating a disk with equidistant-spaced holes near its circumference and channelling air through the holes as illustrated in Figure 2-13. He concluded that pitch is determined by the time it took for the compressed air to blow through from one hole to the next as the disk rotated. This duration is known as the period, and the inverse of this is the pitch of the sound perceived. When the amount of holes was doubled, the frequency of the siren also doubled, and the perceived pitch was an octave higher (first column of Figure 2-14A and Figure 2-14B). In the second half of his experiment, he used a disc with non-equidistantly spaced holes, such that the durations between the air puffs were regulated in the following order t_1 , t_2 , t_1 , t_2 , etc. He noticed that the pitch was equal to its highest periodicity ($T = t_1 + t_2$), as observed in the first column of Figure 2-14C. This attribute corresponds to the lowest perceivable periodic frequency known as the fundamental frequency ($1/T$ Hz). However, the power at the

fundamental frequency was not significant to define the pitch. He concluded that the period of a sound signal is more dominant in determining pitch than its fundamental frequency.

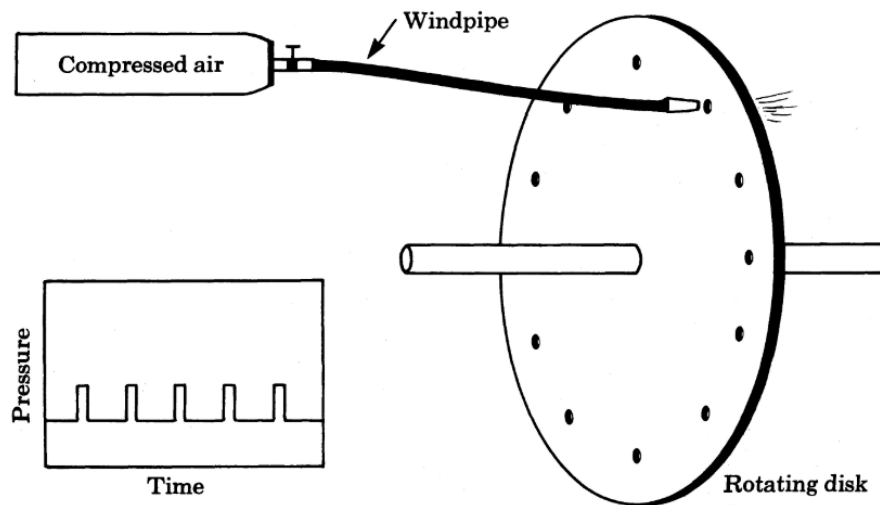


Figure 2-13: Pitch siren experiment setup used by Seebeck [44] and Strutt [45].

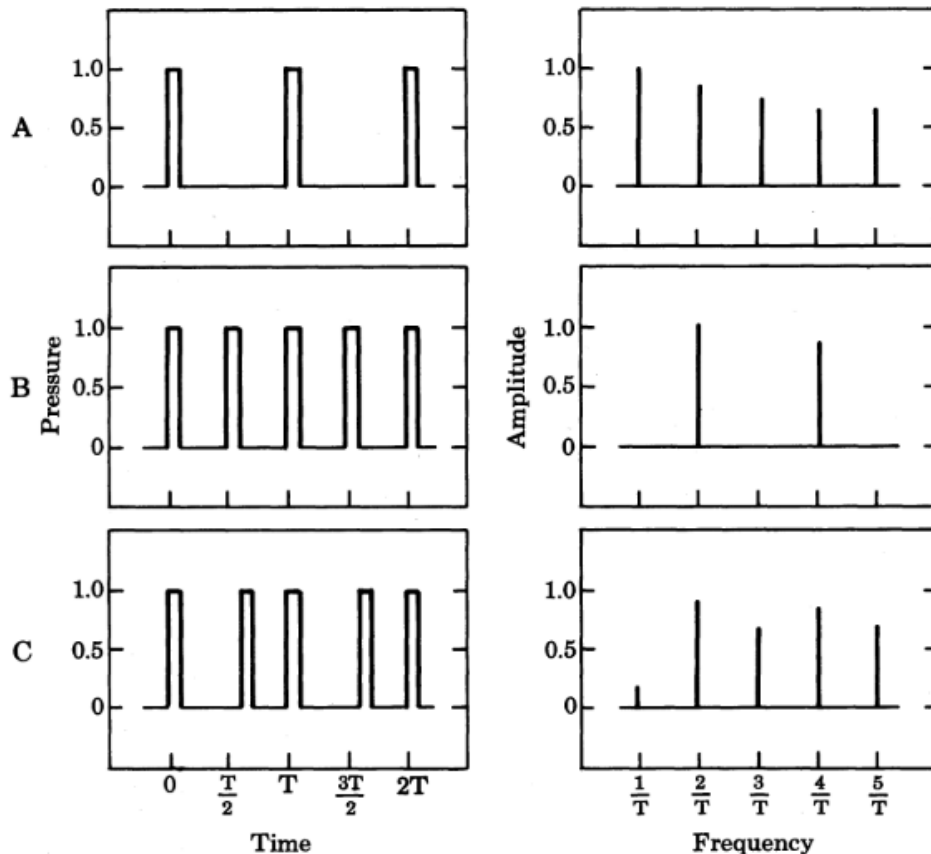


Figure 2-14: Pulses recorded from Seebeck's siren experiment corresponding to the perceived pitch in the first column on the left, and Ohm's application of Fourier transform showing the power spectrum of the frequency components in the second column on the right.

Ohm devised the acoustical law [46] to explain the observations of Seebeck as seen in the second column of Figure 2-14. He applied Fourier's theorem to decompose the waveform into frequency components and showed the power spectra of the components matched the periodic components Seebeck had observed. However, Seebeck maintained

that the low power spectra of the fundamental frequency determined by Ohm's acoustical law was insufficient in explaining the strong pitch he perceived in the second half of his experiment as observed in the second column of Figure 2-14C (labelled as ' $1/T$ ') [47]. Ohm later called this effect an acoustical illusion [48]. Helmholtz attributed this effect to a superposition of nonlinear distortion introduced to the waveforms in the cochlea [49], which can be applied to the three types of stimuli in Figure 2-14. For a pure tone stimulus in Figure 2-14A, nonlinear distortion is added to the sole frequency component, which enhances the perceived pitch. The same applies to the stimulus in Figure 2-14B. For the third stimulus containing two frequencies in Figure 2-14C, nonlinear distortion is added to the difference in frequency of the two components, i.e. $1/T$. This characteristic means that the perceived pitch would still be significant at the fundamental frequency even if the fundamental frequency component itself is weak. For a complex periodic tone comprising multiple harmonic pure tones, the addition of nonlinearity to the frequency difference between any two neighbouring frequency components would still be at $1/T$. In other words, a large harmonic content would ensure a considerable power added at the fundamental frequency.

Fletcher devised an electronic circuit to repeat Seebeck's experiments and filtered out low harmonics [49]. He found the perceived pitch to match the fundamental frequency as well as the frequency difference, thus reinforcing Seebeck's and Helmholtz's findings as well as putting forth the issue of weak fundamental frequency. Helmholtz's theory was further strengthened indirectly by von Békésy's observation of travelling waves on the basilar membranes (BM) in cadavers [50]. However, Schouten disproved Helmholtz's frequency-difference nonlinear distortion hypothesis to perceived pitch, described in the preceding paragraph. He generated a waveform from pulses with a filtered fundamental frequency of 200 Hz and a 206 Hz pure tone. Based on Helmholtz's frequency-difference hypothesis, a 6 Hz repetitive beat is expected to be heard. Instead, no beats were heard, and the perceived pitch was still at 200 Hz. Furthermore, Schouten found that Helmholtz's frequency-difference hypothesis also cannot account for shifted frequency components. He used an amplitude-modulated waveform with components at 1,000 Hz, 1,200 Hz, and 1,400 Hz, and reported the pitch is perceived to be at 200 Hz. When the components were shifted upwards by 40 Hz (1,040 Hz, 1,240 Hz, and 1,440 Hz), it was found that the pitch also shifted upwards to 205 Hz [51]. This attribute was further validated by Ritsma, who mapped several conditions for pitch shifts [52], [53].

Licklider's two experiments further weakened Helmholtz's nonlinear distortion hypothesis. In the first experiment, he developed a stimulus comprising only high harmonics, without any fundamental frequency component. A Fourier analysis of the stimulus confirmed the presence of only the high harmonic components and no indication of the fundamental frequency component. Despite the missing fundamental frequency, low pitch was still perceived from the complex tone of high harmonics. In the second experiment, Licklider formed a simple melody by changing the frequencies of the individual component of the high harmonics. He then added low frequency noise to the stimulus to mask nonlinear distortion effects that may have been contributed by the missing fundamental frequencies of the sequence of complex tones, forming the melody. He used Fourier analysis to ensure the absence of the missing fundamental frequencies. Despite the high-intensity levels of the noise, a low pitch melody can still be perceived [54]. He concluded that the frequency components found using Fourier transform, while valid in defining the spectral composition of a sound, is insufficient in defining perceived pitch, notably when fundamental frequencies

are missing. The results of Licklider's experiment have been repeated and reinforced by Thurlow [55] and Patterson [56].

Schouten used von Békésy's findings of the mammalian basilar membrane (BM) to explain the phenomenon of missing fundamental frequency [57]. This explanation is known as the residue pitch theory, which comprises two stages. The first stage is the BM stage, where low frequency components of a complex sound are perceived as separate pure tones as opposed to complex tones at high frequency. The frequency resolution of the BM is lesser at high frequencies than at low frequencies, which means that a BM filter at low centre frequency has a narrower bandwidth than a BM filter at high centre frequency. As a result, low frequency components of a complex tone are output from the BM filters, which are close to pure tones because the narrow bandwidths of these BM filters profoundly suppress frequency components outside their respective centre frequency (CF) ratings. Therefore, sinusoidal signals output from BM filters at low frequency are resolved – one sinusoidal signal output from a single unique BM filter. On the contrary, high frequency components are unresolved. This notion is valid because BM filters at high frequencies have large bandwidths that integrate unsuppressed high frequency components close to their respective CF ratings.

The temporal patterns in the output of the BM filters are also maintained in the auditory nerve firings, which are transmitted to the next stage. This stage of the mammalian auditory pathway comprises the cochlear nucleus, the primary auditory cortex as well as other cortical centres for pitch processing. How each cortical centre works individually to contribute to pitch perception has yet to be discovered, but there are several hypotheses on the collective operation between the cortical centres. Schouten's explanation is one such hypothesis [57]. He explains that the temporal patterns in the unresolved components, known as residual components, which are preserved as fine-coded information from the auditory nerve firings interact with one another to formulate a periodicity in the cortical centres corresponding to the perceived pitch. In other words, this interaction of residual components results in a single combined output signal, and the perceived pitch is the inverse of the time duration between two significant peaks found in this output signal. Based on Schouten's explanation whereby, since the temporal patterns are maintained throughout the cochlear and cortical centres, one may use the temporal pattern in an input sound signal directly for pitch perception analysis.

An example of the residue pitch theory is illustrated in Figure 2-15, with a fine-structured waveform of a complex tone comprising 4th (800 Hz), 5th (1 kHz), and 6th (1.2 kHz) harmonics of a missing fundamental frequency of 200 Hz. By taking the inverse of the time between the two highest peaks in Figure 2-15(a), the residue pitch is calculable. This calculation equals to the missing fundamental frequency of 200 Hz. In Figure 2-15(c), when all the harmonic components are shifted slightly upwards (frequency increased slightly), using the same principle, the pitch is defined by t_2 . A secondary pitch, t_1 , is also perceived due to the proximity between its highest peak and second highest peak. The distances of the highest peak with smaller peaks also contributes to perceived pitch away from the fundamental frequency. However, the dominant pitch is defined primarily by the highest peaks. Hence, Schouten's residue pitch theory is capable of addressing cases where fundamental frequency is missing in sound, as well as the perceived pitch shift effect.

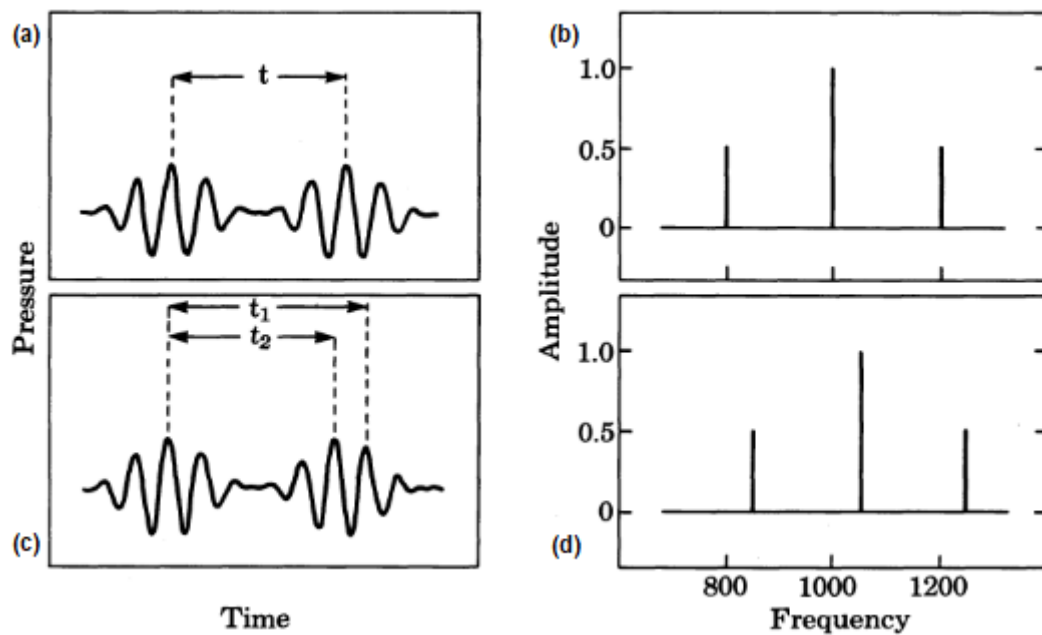


Figure 2-15: Residue theory using the time duration between peaks of a complex sound to determine pitch: (a) Sound waveform missing its fundamental frequency with 3 harmonics and its (b) Fourier frequency components unable to indicate the fundamental frequency; (c) Sound waveform missing its fundamental frequency with shifted harmonic components and its (d) shifted Fourier frequency components not showing the fundamental frequency. Adapted from Wightman et al. [58].

A fundamental frequency calculated with residue pitch theory, using a fine-structured temporal waveform, is affected when the phase of a sound signal changes. This effect is observable from the waveform in Figure 2-16, which is made up of signals ranging from 5th (1,000 Hz) to 10th (2,000 Hz) harmonics with a fundamental frequency of 200 Hz. The harmonic signals with cosine phases are added to produce the bottom-placed waveform in Figure 2-16(a). Figure 2-16(b) displays the effect of adding random-phased harmonic signals. Using the residue theory, the pitches of the relative phase of these two waveforms are different. However, Patterson reported that the pitch perceived is the same despite relative phase changes, although the sound roughness altered accordingly [59]. Wightman had the same findings and developed a pitch model to explain the relative phase phenomenon over the residue theory [60]. Carlyon and Shackleton further found that this phase insensitivity is relegated only to low and mid-level harmonics, and when high-level harmonics above 3.9 kHz are introduced at alternating phase, the pitch doubled [61]. Meddis proceeded to explain these new findings as well as the phase sensitivity phenomenon, with the auditory model described in section 2.1.1 (MAP model) equipped with an autocorrelation function [62], which is presented in subsection 2.2.2.2. Meddis also designed a hypothetical physiological model [41], described in subsection 2.2.2.1, in an attempt to explain the capability of the auditory-autocorrelation model to showcase pitch perception information.

An alternative pitch perception theory to residue pitch theory is global pitch, which contributes to the missing fundamental frequency, f_0 as well as pitch shift effect. According to Brunstrom and Roberts, global pitch is the f_0 , of a periodic complex tone that best fits a harmonic template containing a distribution of resolved frequency components [63]. The harmonic template is a central pitch mechanism in the mammalian brain that is used as a reference for matching the harmonic content of a sound stimulus [64]–[66]. In terms of a missing f_0 , Bendor and Wang showed there exist pitch selective neurons in marmoset

monkeys that react to a pure tone as well as to a complex tone with missing f_0 corresponding to the pure tone frequency [67]. These neurons reside in the anterolateral border of the marmoset's primary auditory cortex that are also found in humans [68]. The experiments of Bendor and Wang suggest that the activations of pitch-selective neurons corresponding to the missing f_0 occur from the presence of the resolved harmonic contents in the stimulus. The resolved harmonics in the stimulus evoke harmonic-selective neurons in a template, which also collectively evokes the neurons corresponding to the f_0 regardless of the presence of f_0 in the stimulus.

Regarding the pitch shift effect, Lin and Hartmann conducted several psychoacoustic pitch matching experiments and found that the perceived f_0 is changed when a harmonic of the f_0 , known as a partial, is mistuned [69]. They concluded that the pitch shift perceived is due to a mismatch between the stimulus with mistuned partial and a tuned partial in a harmonic template in the brain. These pitch-shift effects have also been reported extensively by Roberts and Brunstrom [63], [70]–[72]. These psychoacoustic results have been reinforced by Feng and Wang [73], who found harmonic template-sensitive neurons in marmoset monkeys. Regarding computational models, an early model of the central processing of pitch template was suggested by Goldstein, who used complex tones stimuli [74]. Duifhuis et al. used Goldstein's model on speech stimuli [65]. Scheffer modified Goldstein's model to adhere to auditory frequency analysis [66]. Shamma and Klein attempted to model the central processing of harmonic templates using the contents of the cross-correlation between cochlear filter outputs [75]. An alternative central processing of pitch template is the SPINET model [76], which is described in subsection 2.2.2.3.

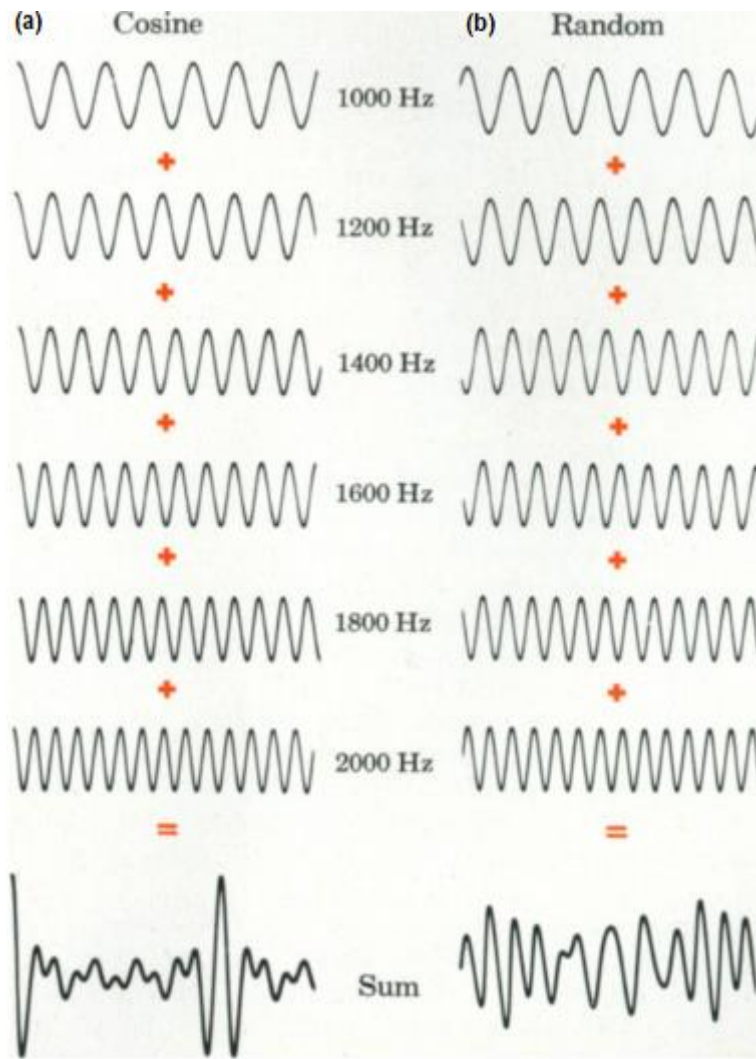


Figure 2-16: Residue pitch from fine-structured temporal waveform (bottom plots labelled as 'Sum') effect due to harmonic signals added in (a) cosine phase, and (b) random phase. Adapted from Wightman et al. [58].

2.2.2. A Survey of Auditory Pitch Models

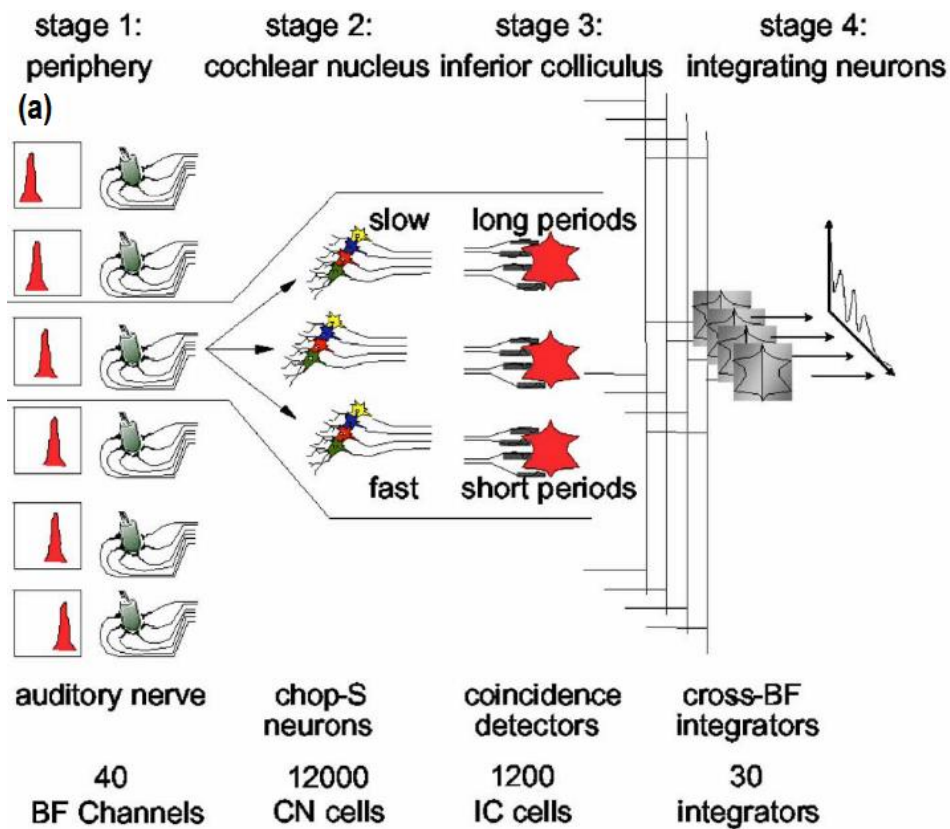
Pitch perception models can be categorised into two theories: spectral and temporal [77]. The spectral theory of pitch perception translates the place coding of the basilar membrane to account for the extraction of pitch information. Temporal theory involves the calculation of distances between peaks in the conditioned output signals represented in the time-domain from a cochlear model.

Carlyon and Shackleton found that temporal models are better equipped than spectral models to extract pitch information from high-numbered harmonics, while spectral models are more capable of extracting pitch from low-numbered harmonics [61]. Their findings agree with Licklider's proposal of determining pitch with a duplex model [54]. Sounds with low frequency components in the range of low pitch perceptual limit of 30 Hz [78], [79] and high perceptual pitch limit of 800 Hz are resolved and pitch can be determined by place coding attributes of spectral theory. When the fundamental frequency is weak or missing, a temporal model can be used to extract pitch from the harmonics. For resolved harmonics in the range of 100 Hz to 400 Hz, pitch information can be reliably extracted from 3rd to 5th harmonics [80]. This range is known as the dominant region of pitch perception. The amount

of harmonics required to determine pitch increases beyond this range, which includes unresolved harmonics [81]. Despite the differences between spectral and temporal models of pitch perception, three such models are reviewed in the subsections below for consideration to be implemented on FPGA.

2.2.2.1. Computational Physiological Model of Virtual Pitch

Temporal regularity or periodicities can characterise virtual pitch or periodicity pitch in a sound signal as part of temporal pitch theory. A physiological model of virtual pitch [41] uses the auditory nerve (AN) signal as an input signal generated from the MAP cochlear model described in subsection 2.1.1. The model comprises four stages, as shown in Figure 2-17. Thirty AN fibres are connected to a ventral cochlear nucleus (VCN) units, and ten VCN units are connected to a single inferior colliculus (IC) unit, which are all corresponding to a unique best frequency (BF). The VCN unit enhances the periodicities in the sound signal, and the IC unit contains coincidence units that fire when it receives synchronous inputs from the VCN. At the final stage, the rate-based spikes are integrated to project the overall periodicity present in the sound signal.



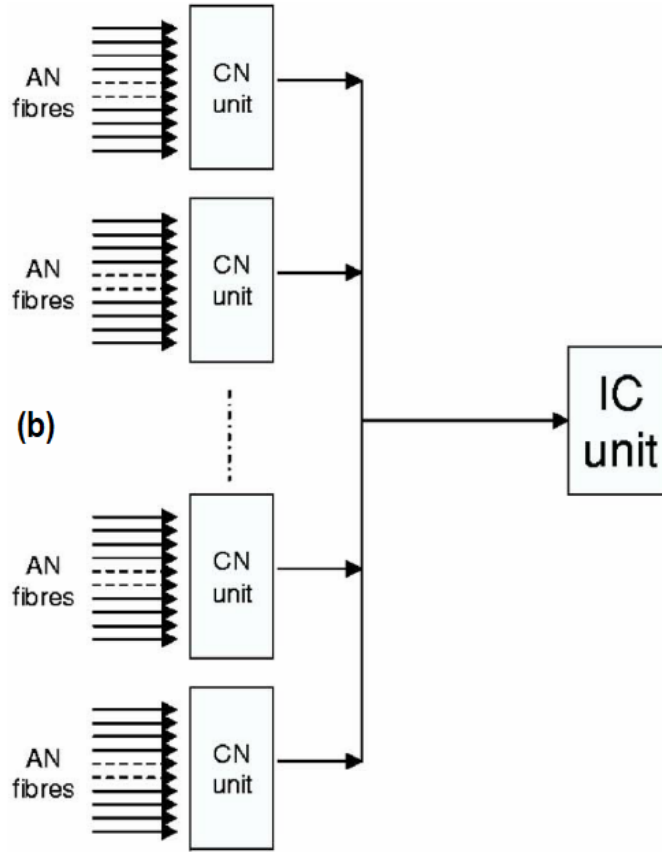


Figure 2-17: Virtual pitch model showing (a) 4 stages from the auditory nerve (AN) to the cross-BF integrators, and (b) the connection between the cochlear nucleus (CN) and the inferior colliculus (IC) units. Adapted from Meddis et al. [41].

Each VCN unit contains multiple chop-S type chopper neurons [82], each of which fires at a fixed rate. Chopper neurons at the cochlear nucleus are also connected to onset neurons that contribute to timbre attributes, but the latter has been omitted from this model to ensure exclusivity to pitch extraction. The chopper neurons have thirty discrete chopping rates that are equally spaced between 60 and 350 spikes/s, and each chopper neuron is characterised by a simplified Hodgkin-Huxley model [83] of spike generation using a point neuron model [84]. The point neuron is influenced by four variables [85]: (2-23) transmembrane potential as a deviation from the cell resting potential; (2-24) a potassium conductance; (2-25) the time-varying threshold; (2-26) a binary spiking variable.

Aside from the potassium levels, the parameters in the point neuron model are also heavily influenced by results from in vitro experiments, where stellate cells response in the cochlear nucleus are obtained by injecting electrical current to the AN fibres [86], [87]. With respect to the applied electrode input current, the change in the potential of a stellate chopper cell is calculated as:

$$\frac{dE(t)}{dt} = \frac{-E(t) + \{V(t) + G_k(t)[E_k - E(t)]\}}{\tau_m} \quad (2-23)$$

where $E(t)$ is the instantaneous cell-membrane potential above resting level E_0 ; G_k is the cell potassium conductance; τ_m is the membrane time constant; E_k is the equilibrium

potential of potassium conductance with respect to the cell resting level; $V(t)$ is the instantaneous change in voltage due to an applied or synaptic current.

The second variable, potassium conductance, is defined by:

$$\frac{dG_k(t)}{dt} = \frac{-G_k(t) + (bs)}{\tau_{Gk}} \quad (2-24)$$

where b is the delayed rectifier potassium conductance strength; s is a binary spiking variable either at 0 or 1; τ_{Gk} is the time constant of potassium conductance decay. The third variable is the rise in the time-varying threshold:

$$\frac{dTh}{dt} = \frac{-[Th(t) - Th_0] + cE(t)}{\tau_{Th}} \quad (2-25)$$

where $Th(t)$ is the time-varying threshold of the cell; Th_0 is the resting threshold; c is an accommodation constant; τ_{Th} is the time constant of the threshold rise. The fourth variable is the main output, which is a combination of transmembrane potential and the binary spiking variable:

$$p(t) = E(t) + s[E_b - E(t)] \quad (2-26)$$

where E_b is the reversal potential of the cell such that

$$s = \begin{cases} 0, & E(t) < Th \\ 1, & E(t) > Th \end{cases} \quad (2-27)$$

To attain the different fixed spike rates, the frequency of firing is adjusted by changing either the potassium conductance (τ_{Gk} or b), membrane time constant, τ_m , or the accommodation term, c .

The VCN units of the same chopping rates are connected to a single IC unit, which is made up of the same algorithm as the VCN but with different parameters. Each IC unit has a coincidence detector, which upon receiving synchronous spike inputs from the ten channelled inputs connected to the VCN units, fire a spike. A single 10-VCN-to-1-IC module as shown in Figure 2-17(b) corresponds to a peak modulation gain at a specific best-modulation frequency that is usually characterised by a bandpass filter if the signals are not spike-based [26]. The final stage of the model is equipped with integrators that sum the spike-based output signals from the IC units to a single dimension waveform. This waveform projects the periodicities in the sound signal, where the reciprocal of time duration between peak magnitudes indicate fundamental frequencies corresponding to pitch attributes present in the sound.

2.2.2.2. Auditory - Autocorrelation Function (ACF) Model

Under the temporal pitch theory, periodicity pitch can be determined using delays and coincidence detection as proposed by Licklider [88]. This mathematical method is known as the autocorrelation function (ACF) and is highly similar to Meddis's physiological model used for deriving virtual pitch described in subsection 2.2.2.1 and in [41]. A variant of the ACF is

the cancellation model where multiplication operation in the ACF is replaced with a subtraction operation [89].

In terms of biological evidence, a delay line with coincidence detection has been found for sound localisation [90], [91] but not for pitch detection. Phase interactions between cochlear sections might account for synthesised delays of resolved partials but not for unresolved partials [92]. Hence, the ACF suffers from the lack of convincing biological evidence to support its hypothesis for its use in the brain for pitch detection, specifically with the use of the delay mechanism. Though, the virtual pitch model [41], described in subsection 2.2.2.1, surmises that the delays are attributable to the slow responses of chopper neurons in the cochlear nucleus.

Regardless of the lack of biological support, the ACF model is used as an analytical tool in speech [93] and music [94] signal processing. In speech processing, vowel identification requires delays at approximately 10 ms [95], [96], and the ACF is an ideal algorithm for this task. It has also been used in computational auditory scene analysis, especially to find multiple fundamental frequencies in polyphonic musical signals from multiple sound sources at an instance [97]–[99].

The input to the ACF model can either be a single vector sound signal or a two-dimensional time-frequency sound image in the form of either a spectrogram (calculated from Fourier transform) or a cochleagram (calculated from a cochlear model). Figure 2-18 depicts a filterbank of ACFs connected to a cochlear model, whereby every row of the cochlear model is connected to a single ACF.

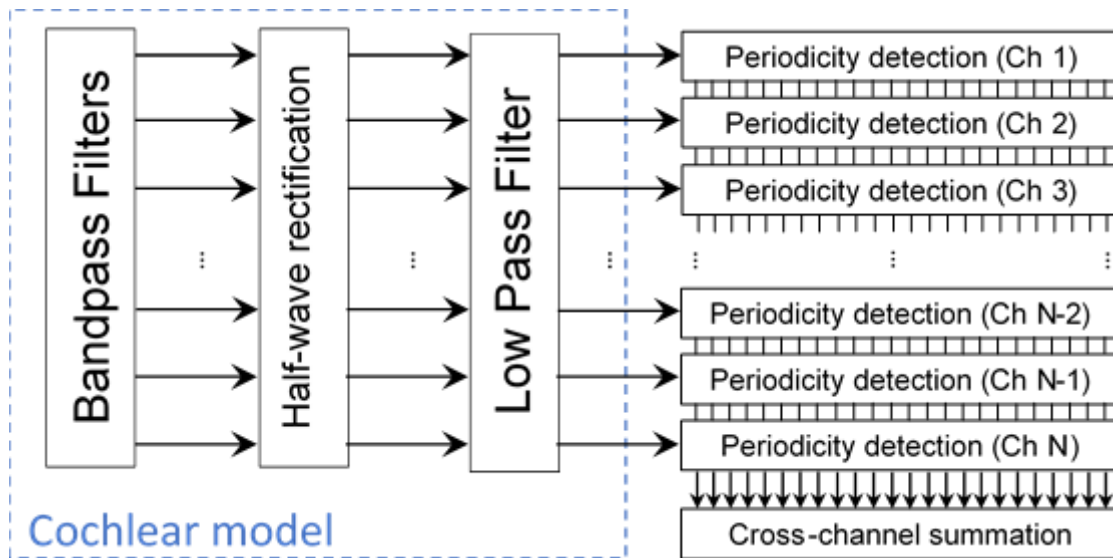


Figure 2-18: Filterbank of autocorrelation functions (ACFs) operating in parallel on the cochlear model output (one ACF connected to one output section of the cochlear model). Adapted from Meddis et al. [62].

As my interest lies in auditory models, I will describe the operation of an ACF with respect to inputs from a cochlear model [100]. An ACF defines the similarity or correlation of a signal to the delayed version of itself. It is characterised by:

$$r_p(c, t, \tau) = \sum_{t=n}^{n+N-\tau} p(c, t) \cdot p(c, t + \tau) \quad (2-28)$$

where p is an input signal of a cochlear section c from a cochlear model; τ is the delay or lag of the cloned original signal; N is the total amount of output data points, r , corresponding to the amount of lags that is encapsulated in a window. Hence, $r_p(c, t, \tau)$ is the output signal of a single vector of samples pertaining to the delayed similarity at time, t . Stacking $r_p(c, t, \tau)$ over one another from the first cochlear section to the last, results in a 2D image known as an autocorrelogram or a stabilised auditory image. The ACF operation per cochlear section is graphically illustrated in Figure 2-19.

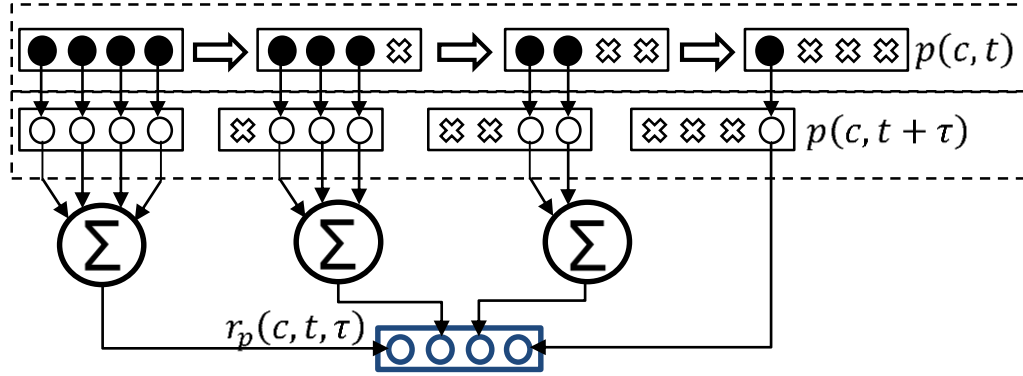


Figure 2-19: Operation of an autocorrelation function (ACF). Vector of dark spots represents the original input signal, whereas white spots represent the cloned original signal that undergoes a delay process. A downward arrow represents a multiply operation, 'x' represents no operation and ' Σ ' represents a sum operation.

The periodicity of a signal can be extracted from an autocorrelated signal by finding the distance between adjacent peaks for every cochlear section. An alternative method is to use a summarised ACF signal [100]. This signal can be calculated by averaging $r_p(c, t, \tau)$ across the entire range of cochlear sections (totalling K), thereby generating a single temporal profile instead of K amount of parallel temporal profiles for pitch extraction:

$$r_s(t, \tau) = \frac{1}{K} \sum_{k=1}^K r_p(c, t, \tau) \quad (2-29)$$

With the use of summary autocorrelograms, pitch extraction is possible by two methods. One method is by calculating the distance between either two significant neighbouring peaks or one peak relative to the peak at 0 delay, which equates to the pitch periodicity and its inverse is the pitch frequency [101]. The second method is to use a pitch matching algorithm by calculating the squared Euclidean distance, D^2 , between a template summary autocorrelograms containing pitch information reference and a summary autocorrelogram of the input sound signal, where the required pitch possibly resides [62]:

$$D^2 = \sum_{i=a}^b (r_s(t, \tau) - r_s'(t, \tau))^2 \quad (2-30)$$

where $\tau = i \cdot dt$; a and b are the range of lags used in the comparison. A small D^2 specifies that the pitch information in the two summary autocorrelograms is similar, whereas a large D^2 specifies dissimilarity between the two summary autocorrelograms.

2.2.2.3. SPINET Model

Spectral theories of pitch perception often use a central pattern recognition hypothesis to match spectral templates [74], [102]–[104]. There is physiological evidence supporting this hypothesis [73]. One model that characterises this hypothesis is the SPINET model by Cohen et al. [76].

Figure 2-20 displays the seven stages of the SPINET model. In stages 1 and 2, the model takes a sound signal as its input and streams the signal into a gammatone filterbank simulating the characteristic of the basilar membrane (BM), as illustrated in Figure 2-20. In stage 3, each gammatone filter output corresponding to a specific centre frequency is averaged over a 5 ms window to generate a short-term energy spectrum. In stage 4, a broad bandpass filter is then applied to the averaged values, which removes low and high frequency components of the perceived hearing range akin to the outer and middle ear filtering.

Stage 5 models the spatial interactions of channels, whereby channels in closer proximity to a specific channel have a larger effect on that channel than channels further away. The interactions between the channels are defined by:

$$S(f_i, n) = \sum_{j=1}^J Y(f, n) \left(\frac{|H(f_i, f_j, \kappa_{ex})|^2}{A_{ex}(f_i)} - \frac{|H(f_i, f_j, \kappa_{in})|^2}{A_{in}(f_i)} \right) \quad (2-31)$$

where n is the sample number; f_i is the centre frequency of the channel; f_j is the centre frequency of other gammatone filters; J is the total number of gammatone filters; κ_{ex} is the excitatory region, and κ_{in} is the inhibitory region set as a constant of the equivalent rectangular bandwidth (ERB) of a frequency channel; A_{ex} and A_{in} are the areas of the excitatory and inhibitory regions, which are power spectrums, $|H(f_i, f_j, \kappa_{ex})|^2$ and $|H(f_i, f_j, \kappa_{in})|^2$ summed over all the centre frequencies of the gammatone filterbank. The power spectrum of a gammatone filter is calculated as:

$$|H(f_i, f_j, \kappa)|^2 = \left[1 + \left(\frac{(f_j - f_i)}{\kappa b(f_i)} \right)^2 \right]^{-4} \quad (2-32)$$

where $b(f_i)$ is the bandwidth of a gammatone filter. With a flat power spectrum, the excitatory and inhibitory segments combine to output zero across the frequency spectrum, simulating the equilibrium response of neurons.

Finally, in stages 6 and 7, a weighted harmonic summation is done:

$$P(p, n) = \sum_m [S(mp, n)]^+ h(m) \quad (2-33)$$

$$[x]^+ = \begin{cases} x, & \text{for } x > 0 \\ 0, & \text{otherwise} \end{cases} \quad (2-34)$$

$$h(m) = \begin{cases} 1 - M \log_2(m), & \text{for } M \log_2(m) < 1 \\ 0, & \text{otherwise} \end{cases} \quad (2-35)$$

where P is the pitch strength; $S(mp, n)$ is the non-negative spectral strength calculated in stage 5 and weighted by the distance between nominal pitch p and harmonic frequency mp ; M defines the slope of the decay with harmonic number m .

The output of the model is the pitch with the strongest activation. In other words, pitch is determined to be the value of p , with the largest pitch strength P .

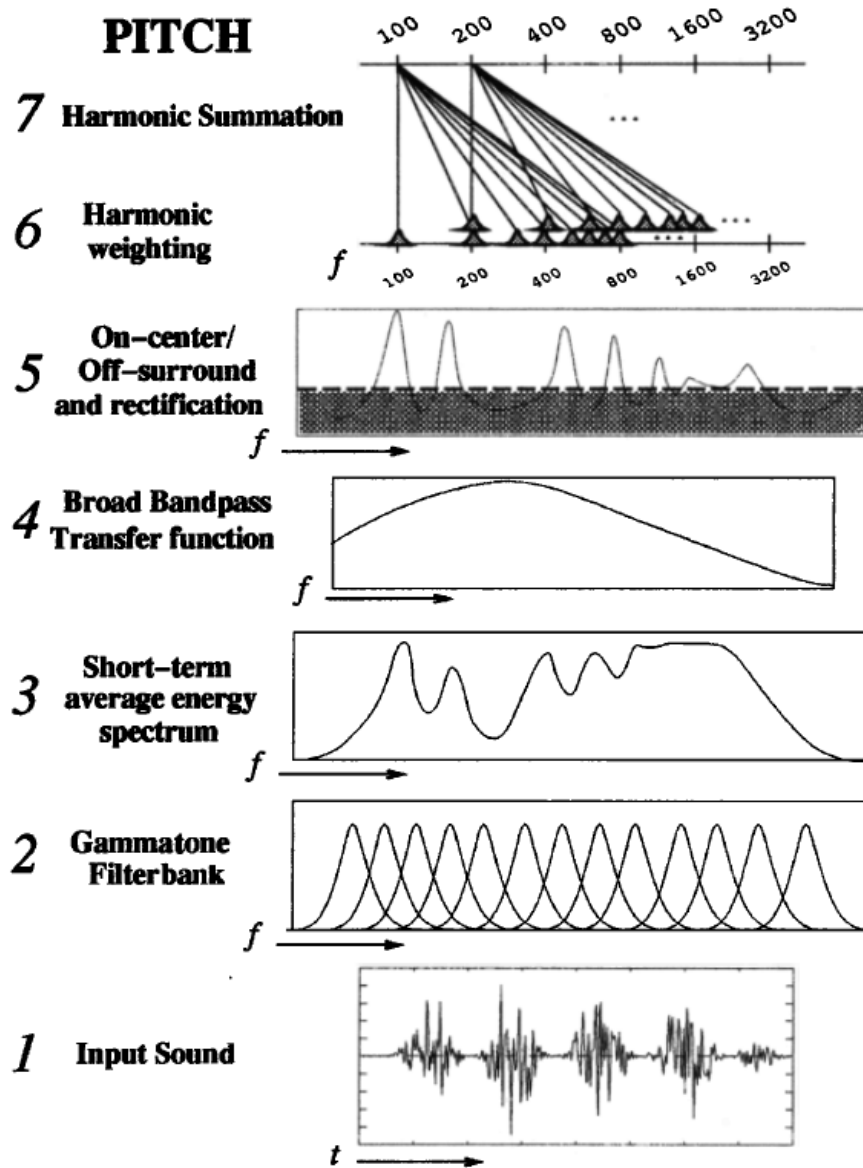


Figure 2-20: The SPINET model. Adapted from Cohen et al. [76].

2.2.2.4. Model Selection

Table 2-2 presents a summary of the three pitch models. The virtual pitch model has already been implemented in real-time on an analogue very large-scale integrated (VLSI) chip [105] and so is not considered for FPGA implementation. Both the SPINET and autocorrelation function (ACF) models are calculable on an FPGA. However, both models are computationally intensive as they rely heavily on multiplication and accumulate (MAC) operations.

The ACF and SPINET models differ in the way they use MAC operations. For the ACF model, the multiply and accumulate operations are evoked consecutively for every input sample. However, this is not the same for the SPINET model, as the equations for calculating pitch combine multiply and accumulate operations are non-consecutively. In other words, these two operations are distributed separately throughout the model and do not appear in the order of the MAC sequence that is then processed iteratively. Iterative MAC operations are crucial in model selection since an alternative method is presented in chapter 4 to approximate MAC operations, which means the model with a higher iterative MAC operations usage is required to demonstrate the difference in computational resource utilisation between the MAC and the novel approximation operations. Thus, the ACF model is selected over the SPINET model for FPGA implementation.

Although the ACF model is selected for FPGA implementation, its reliance on significant MAC operations to calculate an autocorrelogram poses a problem. This heavy reliance on MAC operations will undoubtedly, consume significant memory and power as well as utilise significant digital signal processors (DSP) on an FPGA, mainly if parallel computation is used. However, these computational resources can be minimised by pipelining the MAC operations, instead of computing the operations in parallel. Furthermore, and more importantly, a spike-based approach alleviates the use of MAC operations and offers a cheaper computing alternative. These are addressed with respect to the ACF models in chapter 4.

Pitch Model	Features
Physiological Model	Physiological model based on temporal pitch theory, which can be regarded as a general pitch model.
	Encompasses a chopper neuron algorithm to simulate a functional part of the ventral cochlear nucleus (VCN) and a functional part of the inferior colliculus (IC) to strengthen periodicities across cochlear sections.
Autocorrelation Function (ACF)	Mathematical equivalent of the virtual pitch physiological model.
	Correlation across delayed samples per cochlear section for every cochlear section.
SPINET model	Based on spectral pitch theory and central template matching hypothesis.
	Pitch determined by the strongest frequency channel activation.

Table 2-2: Summary of auditory pitch models.

2.3. Auditory Timbre

According to the Acoustical Society of America (ASA) [106], timbre is defined as a multidimensional auditory sensation that enables a listener to distinguish two non-identical sounds of the same loudness, pitch, spatial location, and duration [107], [108]. It is an essential cue in the identification of a sound source, e.g. musical instrument, human

speaker, animal, other natural and man-made sound sources [43]. The dimensions of timbre are defined in subsection 2.3.1.

In the next subsections, two aspects of timbre are presented: studies of timbre perception in subsection 2.3.1 and biologically-inspired auditory cortical models that may influence timbre perception in subsection 2.3.2.

2.3.1. Timbre Perception

One of the earliest studies of timbre perception was by Helmholtz. He gave a descriptive account of the physics of musical instruments, their playing styles and the types of tones generated [109]. In his PhD dissertation, Lichte reported that the timbre of complex tones has at least three dimensions: brightness, roughness, and fullness [110]. Brightness is the midpoint of energy distribution in the frequency domain. It has similar physical properties as sharpness [111], which is defined as the intensity levels of the high frequency components of a sound signal [112]. Roughness is the perception of an unpleasant sound [113] due to the presence of a low fundamental frequency component [114], [115], which produces a ripple in its amplitude over time in the range of 20 Hz to 200 Hz. Fullness is the presence of a broad range of frequency components, particularly at lower frequencies [116].

Timbre is also heavily influenced by the time-varying contour, especially at the onset (attack) or initial transient of a signal [117], [118]. Using early digital computers and musical notes from fourteen musical instruments, Luce found invariances in the attack phases unique to each instrument. This response was validated by Grey [119], who found that temporal cues such as attack as well as spectral cues such as brightness of monophonic musical signals (signals originating from a single musical instrument) are necessary to define timbre. However, Iversen found that by removing the attack phase of a monophonic musical signal, musical instruments can still be identified by human listeners [120]. In spite of the progress of timbre perceptual research over time, it is beneficial to understand the underlying mechanics of the mammalian auditory system that allows us to perceive timbre.

2.3.1.1. Modulation Filterbank

Since the envelope of a sound signal carries vital information of the temporal dimensions of timbre, applying a temporal modulation filter is a sensible solution for representing the envelope of a carrier signal. In psychoacoustical studies, it has been found that low envelope frequencies with modulation peak rates of 3 to 4 Hz are associated with sequence rates of words and frequencies up to 20 Hz are associated with rhythm [121]. From 10 Hz to 200 Hz, modulation components define the roughness of a sound signal [122]. Such temporal modulation dynamics are found in a mammalian primary auditory cortex (A1) [123]. In addition, the same cortical region is also reactive to spectral envelope frequencies found in the range of 0.2 to 3 cycles/octave [124]. This form of spectral modulation is analogous to cepstral representation [125], which is the inverse Fourier transform of the logarithm of a sound signal spectrum. Spectral modulation is applied on a local scale of the spectrum, whereas the cepstral calculation is applied to the entire spectrum. It has been postulated by Shamma that the dimensions of spectral and temporal (spectro-temporal) modulations form the basis of representing timbre [126].

Modulation filterbanks can also characterise the responses of a group of neuronal cells in a mammalian primary auditory cortex (A1). The manner a group of A1 neuronal cells

respond to a stimulus is known as the spectro-temporal receptive field (STRF). The STRF responses of A1 neuronal cells are related to the Fourier transform, i.e. cells that are tuned broadly correspond to low ripple densities and cells that are sharply tuned are most sensitive to high ripple densities [127]. Similarly, A1 cells that best respond to slower dynamics are better tuned to low ripple velocities than cells sensitive to fast dynamics, which are tuned to high ripple velocities. The STRFs can be modelled from a functional perspective as selective modulation filters. Each filter is sensitive to a specific spectral resolution, known as scale, as well as to a temporal modulation, known as rate. Together, these STRFs form a filterbank corresponding to a wide range of psychoacoustical recordings of rate [128] and scale [129] sensitivities found in humans [130], [131] and animals [132].

2.3.1.2. *Effect of Stimulus Type on Modulation Filterbank Response*

In an auditory computational model, the capability of a modulation filterbank emulating A1 neuronal response is dependent on the type of stimulus used and the linearity of the filterbank. In an experiment by Sahani and Linden [133], A1 neuronal responses of mice and rats were recorded with a presentation of random chord stimuli to the animals' outer ear. Here, a random chord is a combination of pure tones with randomly selected frequencies [134]. The researchers then compared the A1 neuronal responses from the animal with responses from a linear STRF model. The input to the model was the same random chord stimuli used in the animal experiment. The stimuli were transformed into a spectrogram, which were then processed by the linear STRF modulation filters. It was found that the linear STRF model could only account for 18% to 40% of the A1 neuronal responses recorded from the animals. Using natural-sounding stimuli, inclusive of narrowband sound signals such as cricket calls to broadband sound signals such as gurgling creek, Machens et al. reported that only an average of 11% of the A1 neuronal responses of rats is accountable by a linear STRF model [135].

Yet another stimulus type is a broadband signal with sinusoidally modulated spectro-temporal envelopes called moving ripples. Moving ripples have identical functions as regular sinusoids (pure tones) in measuring the transfer functions of linear filters [136]–[138]. The exception is that moving ripples are two-dimensional, covering spectral and temporal dimensions. They comprise a combination of pure tones having frequencies that are spaced logarithmically. Shamma, Versnel, and Kowalski found that approximately 90% of selected neurons in A1 of ferrets respond to moving ripples [136]. Transfer functions of an STRF modulation filterbank are derived as a result of the experiment with moving ripples stimuli, which are used in a predictive model to correlate the A1 neuronal response of ferrets in another experiment using naturally- and artificially-voiced vowels as stimuli [139]. It was found in the latter experiment that the moving ripple-derived transfer function of the STRF filterbank is capable of accounting 71% of the selected A1 neuronal responses to moving ripples. Aside from moving ripples, other selected methods of measuring A1 neuronal response include naturally- and artificially-generated bird chirps [140].

2.3.2. A Survey of Auditory Cortical Models

Two biologically-inspired models of timbre perception are presented in this subsection to represent signals for the recognition of sound sources. Both these models use wavelet transform through a filterbank comprising of either multiple time or time-frequency components to extract either a temporal or spectro-temporal multiresolution representation of the envelopes present in the sound signal.

2.3.2.1. *Dau's Temporal Modulation Model*

The temporal modulation model is a psychoacoustical model that utilises a temporal modulation transfer function (TMTF) [128] to extract the envelope of an input sound signal at multiple temporal modulation rates [141]. It is a functional abstract of a mammalian auditory cortex determined from perceptual experiments. Figure 2-21 illustrates the model. The bandpass characteristics of the basilar membrane (BM) is modelled using a gammatone filterbank with centre frequencies ranging from 100 Hz to 4 kHz [142]. A gammatone filter is used widely as auditory filters, which has impulse response shaped like gamma distribution that mimics the response of a specific location on a BM [29], [143].

The output of each of the 24 gammatone filters is half-wave rectified and low-pass filtered at 1 kHz to simulate inner hair cell (IHC) characteristics. This processing preserves the envelope of high frequency components of the signal. The IHC output is transmitted to the adaptation stage to reduce large-signal envelope variations. The adaptation stage is a nonlinear model comprising five low-pass filters (LPFs) connected in series with a feedback loop on each filter. The time constants are different for each filter ranging from 5 ms to 500 ms. The combined output of each of the five LPF fed back determines the amount of attenuation applied to the input signal [144].

The output of each of the adaptation stages is connected to a modulation filterbank, comprising 12 filters. The lowest modulation frequency filter is a low-pass filter (LPF) with a 2.5 Hz cut-off. All other filters are bandpass filters (BPFs). BPFs ranging from 0 to 10 Hz are linearly scaled, and those between 10 Hz to 1 kHz are logarithmically scaled. The output of the temporal modulation filterbank is interpreted as a three-dimensional representation of an input sound signal, comprising amplitude, time and modulation centre frequency information.

Internal noise with a constant variance is added to each modulation filter output to simulate the limitations of the resolution. This internal representation of the input sound signal is cross-correlated with several pre-acquired internal representation of the sound signal with high signal-to-noise (SNR). This form of optimal detector uses high correlation coefficients as a decision device to match and predict the performances of subjects on a trial-by-trial basis in a temporal modulation psychoacoustic experiment.

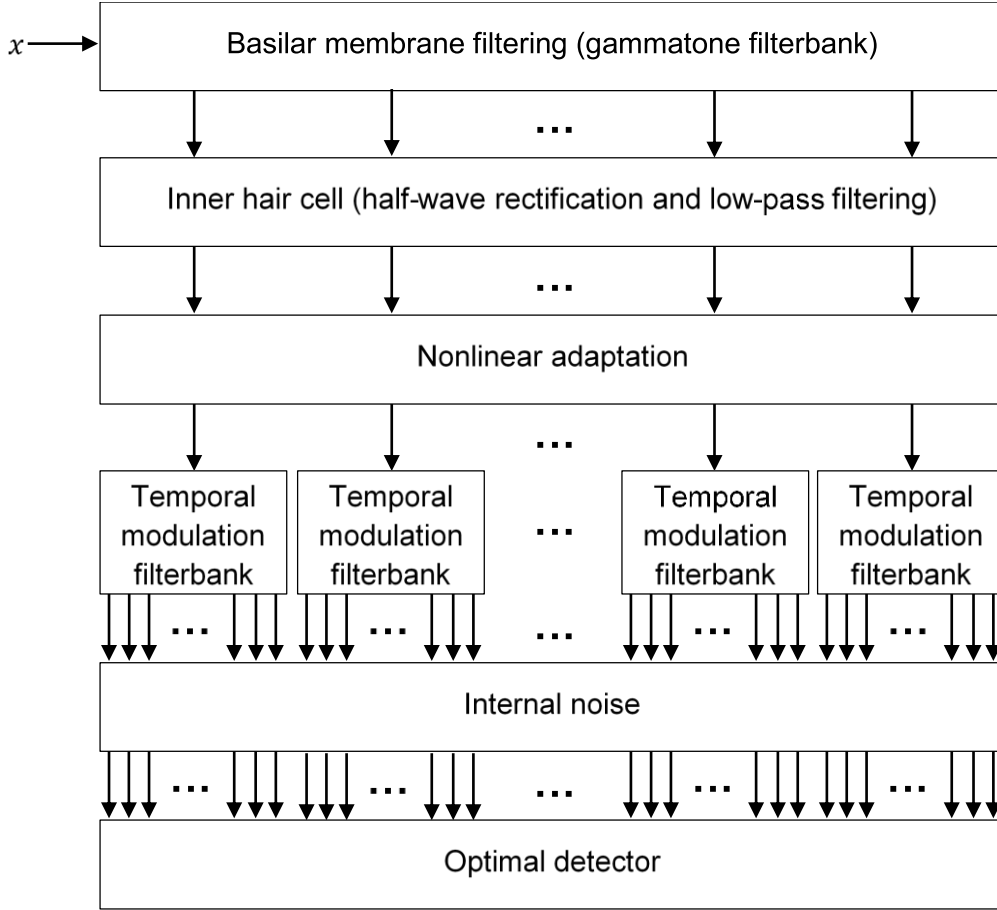


Figure 2-21: An abstract of the temporal modulation model. Adapted from Dau et al. [141].

2.3.2.2. Multiresolution Spectro-temporal NSL Model

The multiresolution spectro-temporal model [127] is an abstract formulation of physiological recordings of the primary auditory cortex (A1) of the ferret [136], [145]. The spectral and temporal modulation contents of the auditory spectrogram are extracted using a modulation filterbank. The filterbank parameters correspond to spectro-temporal envelopes ranging from slow to fast rates temporally and from narrow to broad scales spectrally. The spectro-temporal receptive fields (STRF) are distributed at various frequencies along the frequency axis as displayed in Figure 2-22. The STRF can be defined using seed functions in the time domain, t , and frequency domain parameter, x , as follow:

$$h_t(t) = t^2 e^{-3.5t} \sin(2\pi t) \quad (2-36)$$

$$h_s(x) = (1 - x^2) e^{-x^2/2} \quad (2-37)$$

where h_t is a gammatone function [146]; h_s is a second-order Gaussian Gabor-like function that is usually used for describing spatial receptive field in vision literature [147]. For different rates, ω , and scales, Ω , the seed functions are:

$$h_t(t; \omega) = \omega h_t(\omega t) \quad (2-38)$$

$$h_s(x; \Omega) = \Omega h_s(\Omega x) \quad (2-39)$$

The seed function, h_t , is used in a temporal function, h_{irt} , and h_s is used in a spectral function, h_{irs} . These temporal and spectral functions are each represented in an analytic signal equation format, which comprises two real-valued terms: the output of the seed function and a 90° phase-shifted output of the seed function. The temporal and spectral functions are defined as:

$$h_{irt}(t; \omega, \theta) = h_t(t; \omega) \cos \theta + \hat{h}_t(t; \omega) \sin \theta \quad (2-40)$$

$$h_{irs}(x; \Omega, \phi) = h_s(x; \Omega) \cos \phi + \hat{h}_s(x; \Omega) \sin \phi \quad (2-41)$$

where \hat{h}_t and \hat{h}_s represent Hilbert transform (90° phase-shifted) of the two seed functions, h_t and h_s , respectively; Ω and ω are the spectral density and temporal velocity parameters of the filter defining discrete rates of change of spectral and temporal envelopes; ϕ and θ are characteristic phases. From h_{irt} and h_{irs} , the complex temporal impulse response, h_{IRT} , and the complex spectral impulse response, h_{IRS} , can then be defined as:

$$h_{IRT}(t; \omega, \theta) = h_{irt}(t; \omega, \theta) + j\hat{h}_{irt}(t; \omega, \theta) \quad (2-42)$$

$$h_{IRS}(x; \Omega, \phi) = h_{irs}(x; \Omega, \phi) + j\hat{h}_{irs}(x; \Omega, \phi) \quad (2-43)$$

These complex impulse responses are capable of representing excitatory and inhibitory neuronal responses observed in a mammalian A1 [124], [136], [145]. The real components of the product of equations (2-42) and (2-43) result in the STRF:

$$STRF = \text{Re}\{h_{IRT}(t; \omega, \theta) \cdot h_{IRS}(x; \Omega, \phi)\} \quad (2-44)$$

The differential sensitivity of A1 neuronal cells demonstrate the modulation drifts of spectral envelopes in two directions [148]: upward and downward. Neuronal cells exhibiting this property have spectral peaks that move upward and downward in frequency for an input signal with changing frequencies. However, in reality, most A1 cells are not strongly bidirectionally (upward and downward) sensitive. Instead, they are more responsive to either one direction (upward) or the other (downward). Regardless, the mathematical formulation defining these characteristics captures frequency modulation (FM) sweeps and amplitude modulations (AM) from the input auditory spectrogram [127]:

$$STRF_{\Downarrow} = \text{Re}\{h_{IRT}(t; \omega, \theta) \cdot h_{IRS}(x; \Omega, \phi)\} \quad (2-45)$$

$$STRF_{\Uparrow} = \text{Re}\{h_{IRT}^*(t; \omega, \theta) \cdot h_{IRS}(x; \Omega, \phi)\} \quad (2-46)$$

where \Downarrow and \Uparrow denotes downwards and upwards modulations drifts, respectively and $*$ defines complex conjugate.

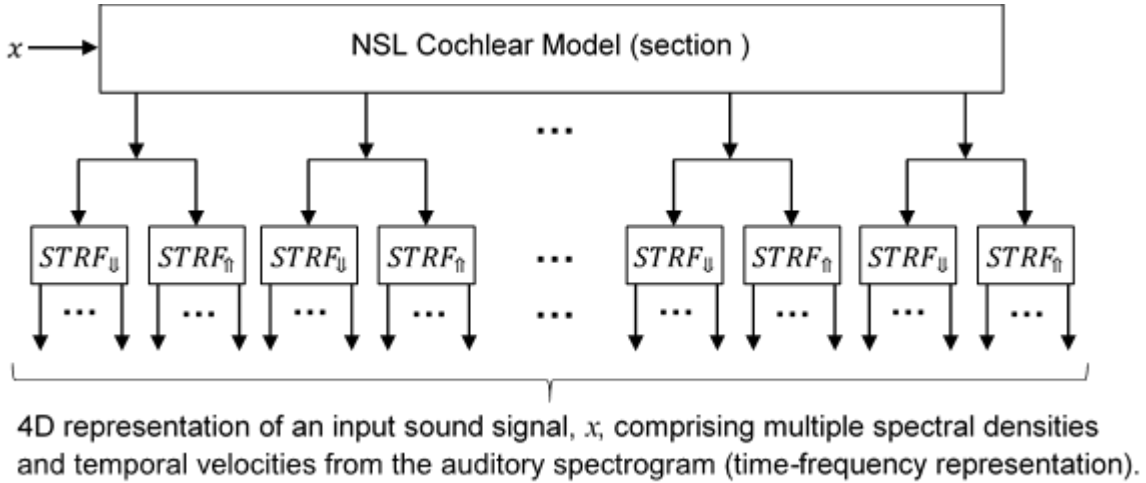


Figure 2-22: The multiresolution spectro-temporal auditory cortical model. Adapted from Shamma et al. [126], [127].

2.3.2.3. Model Selection

A summary of the two auditory cortical models are projected in Table 2-3. Dau's temporal modulation model has been used for automatic speech recognition [149], assessing speech quality [150], predicting speech intelligibility in hearing-impaired subjects [151], detecting across-channel sound fluctuations [152], audio quality assessment [153], audio decoding for audio signal transmission [154], and a binaural sound signal detector [155]–[157].

Like Dau's model, the NSL model extracts temporal modulation information from a sound signal. In addition, the NSL model also extracts spectral modulation information. The combination of spectro-temporal modulation information from the NSL model has several features. It can be resynthesised to restore perceptually intelligible speech, advantageous for speech compression [158] and as a result, has also been used in speech intelligibility experiments [159]. Other applications of the NSL model include source separation of speech signals by gender [160], speech detection amid animal vocals, music and environmental sounds [161], noise suppression in speech signals [162] or speech enhancement [163], phoneme discrimination for automatic speech recognition [164], automatic speech emotion recognition [165], and investigation of speech signal reconstruction quality under various spectro-temporal fluctuations [166].

Aside from speech signals, the NSL model is also used for musical timbre classification, which has resulted in a musical instrument classification accuracy as high as 98.7% [167] using a support vector machine (SVM) and 11 musical instrument classes. Burred et al. used an alternative method of extracting spectro-temporal envelopes using sinusoidal modelling, frequency interpolation, and principal component analysis to attain a score of 94.9% musical instrument classification accuracy from five instruments [168]. Eronen and Klapuri used temporal features such as rise-time and decay-time in addition to temporal envelope information as well as spectral features using mel-frequency cepstral coefficients (MFCC), representing discrete magnitudes of short-term power distributed on a nonlinear frequency scale, to achieve 80% classification of 30 musical instruments [169]. Out of all the models mentioned above, the NSL model has the highest classification accuracy of musical instruments. Hence, the NSL model is selected to be implemented on FPGA.

Auditory Cortical Model	Features
Dau's Temporal Modulation Model	Nonlinear adaptation.
	Temporal modulation filterbank implemented.
	Features extracted from temporal modulation filterbank output to match the performances of subjects in a psychoacoustic experiment.
Multiresolution Spectro-temporal NSL Model	Spectral and temporal (spectro-temporal) modulation filterbanks implemented.
	Neuronal cell directionality capturing upward and downward modulation drifts.
	Features extracted from spectro-temporal receptive fields (STRFs) under various spectral densities and temporal velocities capable of representing features for classifying musical instruments accurately up to 98.7% [167].

Table 2-3: Summary of auditory timbre models.

2.4. Chapter Summary and Conclusion

In this chapter, a brief description of a mammalian auditory pathway is presented along with psychoacoustical studies of human pitch and timbre perceptions. Two cochlear models, three pitch perception models, and two timbre perception models are reviewed. Out of the two cochlear models reviewed, the CAR segment of the CAR-FAC model is selected to be modified and implemented on FPGA, which is presented in chapter 3. A pitch perception model, the ACF model, is selected to be fitted with a novel algorithm to run on an FPGA, which is presented in chapter 4. For timbre perception, the multiresolution spectro-temporal auditory cortical model is selected to be modified and implemented on FPGA with selected algorithm segments from the other reviewed timbre models to be included as well. The details of this design and FPGA implementation are covered in chapter 5. The capabilities of the selected pitch models and timbre model are also showcased in chapters 6 and 7 for the classifications of monophonic musical notes and musical instruments, respectively.

2.5. Bibliography

- [1] S. A. Shamma, R. S. Chadwick, W. J. Wilbur, K. A. Morrish, and J. Rinzel, "A Biophysical Model of Cochlear Processing: Intensity Dependence of Pure Tone Responses," *J. Acoust. Soc. Am.*, vol. 80, no. 1, pp. 133–145, 1986, doi: 10.1121/1.394173.
- [2] R. Meddis, L. P. O'Mard, and E. A. Lopez-Poveda, "A Computational Algorithm for Computing Nonlinear Auditory Frequency Selectivity," *J. Acoust. Soc. Am.*, vol. 109, no. 6, pp. 2852–2861, 2001, doi: 10.1121/1.1370357.
- [3] R. F. Lyon, "Cascades of two-pole–two-zero asymmetric resonators are good models of peripheral auditory function," *J. Acoust. Soc. Am.*, vol. 130, no. 6, p. 3893, 2011, doi: 10.1121/1.3658470.
- [4] C. S. Thakur, T. J. Hamilton, J. Tapson, A. van Schaik, and R. F. Lyon, "FPGA Implementation of the CAR Model of the Cochlea," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1853–1856, doi: 10.1109/ISCAS.2014.6865519.
- [5] Y. Xu, C. S. Thakur, R. K. Singh, R. Wang, J. Tapson, and A. Van Schaik, "Electronic Cochlea : CAR-FAC Model on FPGA," in *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2016, pp. 564–567, doi: 10.1109/BioCAS.2016.7833857.

- [6] Y. Xu, C. S. Thakur, R. K. Singh, T. J. Hamilton, R. Wang, and A. van Schaik, "A FPGA Implementation of the CAR-FAC Cochlear Model," *Front. Neurosci.*, vol. 12, no. April, pp. 1–14, 2018, doi: 10.3389/fnins.2018.00198.
- [7] J. O. Pickles, "The Outer and Middle Ears," in *An Introduction to the Physiology of Hearing*, 4th ed., Bingley, U.K.: Brill, 2012, pp. 11–24.
- [8] G. G. Matthews, "Hearing and Other Vibration Senses," in *Neurobiology Molecules, Cells and Systems*, 1st ed., Cambridge, Massachusetts: Blackwell Science, 1997, pp. 433–457.
- [9] J. O. Pickles, "The Auditory Nerve," in *An Introduction to the Physiology of Hearing*, 4th ed., Bingley, U.K.: Brill, 2012, pp. 73–100.
- [10] J. J. Rosowski, "The effects of external- and middle-ear filtering on auditory threshold and noise-induced hearing loss," *J. Acoust. Soc. Am.*, vol. 90, no. 1, pp. 124–135, 1991, doi: 10.1121/1.401306.
- [11] M. A. Ruggero and A. N. Temchin, "The Roles of the External , Middle , and Inner Ears in Determining the Bandwidth of Hearing," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 99, no. 20, pp. 13206–13210, 2002.
- [12] R. K. Singh, "A Real-time Implementation of the Primary Auditory Neuron Activities," University of Western Sydney, 2012.
- [13] A. Huber, T. Linder, M. Ferrazzini, S. Schmid, N. Dillier, S. Stoeckli, and U. Fisch, "Intraoperative Assessment of Stapes Movement," *Ann. Otol. Rhinol. Laryngol.*, vol. 110, no. 1, pp. 31–35, 2001, doi: 10.1177/000348940111000106.
- [14] J. O. Pickles, "The Cochlea," in *An Introduction to the Physiology of Hearing*, 3rd ed., Bingley, U.K.: Emerald Group, 2008, pp. 25–72.
- [15] F. Mammano and R. Nobili, "Biophysics of the cochlea : Linear approximation Biophysics of the cochlea : Linear approximation," *J. Acoust. Soc. Am.*, vol. 93, no. 6, pp. 3320–3332, 1993, doi: 10.1121/1.405716.
- [16] A. van Schaik, "Analogue VLSI Building Blocks for an Electronic Auditory Pathway," École Polytechnique Fédérale de Lausanne, 1997.
- [17] E. A. Lopez-Poveda and R. Meddis, "A Human Nonlinear Cochlear Filterbank," *J. Acoust. Soc. Am.*, vol. 110, no. 6, pp. 3107–3118, 2001, doi: 10.1121/1.1416197.
- [18] R. S. Swenson, "Chapter 7D - Auditory System," *Review of Clinical and Functional Neuroscience*, 2006. https://www.dartmouth.edu/~rswenson/NeuroSci/chapter_7D.html (accessed Mar. 21, 2019).
- [19] J. O. Pickles, "Mechanisms of transduction and excitation in the cochlea," in *An Introduction to the Physiology of Hearing*, 4th ed., Bingley, U.K.: Brill, 2012, pp. 101–154.
- [20] J. J. Guinan, "Olivocochlear Efferents: Anatomy, Physiology, Function, and the Measurement of Efferent Effects in Humans," *Ear Hear.*, vol. 27, no. 6, pp. 589–607, 2006, doi: 10.1097/01.aud.0000240507.83072.e7.
- [21] J. Ashmore, "Cochlear Outer Hair Cell Motility," *Physiol. Rev.*, vol. 88, no. 1, pp. 173–210, 2008, doi: 10.1152/physrev.00044.2006.

- [22] R. C. Kidd and T. F. Weiss, "Mechanisms that Degrade Timing Information in the Cochlea," *Hear. Res.*, vol. 49, no. HEARES 01421, pp. 181–208, 1990.
- [23] R. Meddis, "Simulation of Mechanical to Neural Transduction in the Auditory Receptor," *J. Acoust. Soc. Am.*, vol. 79, no. 3, pp. 702–711, 1986, doi: 10.1121/1.393460.
- [24] R. Meddis and E. A. Lopez-Poveda, "Auditory Periphery: From Pinna to Auditory Nerve," in *Computational Models of the Auditory System*, R. Meddis, E. A. Lopez-Poveda, R. R. Fay, and A. N. Popper, Eds. New York, Dordrecht, Heidelberg, London: Springer, 2010, pp. 7–38.
- [25] C. J. Sumner, E. A. Lopez-Poveda, L. P. O'Mard, and R. Meddis, "A Revised Model of the Inner-Hair Cell and Auditory-Nerve Complex," *J. Acoust. Soc. Am.*, vol. 111, no. 5, pp. 2178–2188, 2002, doi: 10.1121/1.1453451.
- [26] M. J. Hewitt and R. Meddis, "A computer model of amplitude-modulation sensitivity of single units in the inferior colliculus," *J. Acoust. Soc. Am.*, vol. 95, no. 4, pp. 2145–2159, 1994, doi: 10.1121/1.408676.
- [27] R. T. Ferry and R. Meddis, "A Computer Model of Medial Efferent Suppression in the Mammalian Auditory System," *J. Acoust. Soc. Am.*, vol. 122, no. 6, pp. 3519–3526, 2007, doi: 10.1121/1.2799914.
- [28] R. F. Lyon, "Auditory Filter Models," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 239–264.
- [29] R. F. Lyon, A. G. Katsiamis, and E. M. Drakakis, "History and Future of Auditory Filter Models," *ISCAS 2010 - 2010 IEEE Int. Symp. Circuits Syst. Nano-Bio Circuit Fabr. Syst.*, pp. 3809–3812, 2010, doi: 10.1109/ISCAS.2010.5537724.
- [30] R. F. Lyon, "The CARFAC Digital Cochlear Model," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 293–298.
- [31] C. D. Geisler, G. K. Yates, R. B. Patuzzi, and B. M. Johnstone, "Saturation of Outer Hair Cell Receptor Currents Causes Two-Tone Suppression," *Hear. Res.*, vol. 44, no. 2–3, pp. 241–256, 1990, doi: 10.1016/0378-5955(90)90084-3.
- [32] R. F. Lyon, "The Outer Hair Cell," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 309–319.
- [33] M. R. Schroeder and J. L. Hall, "Model for Mechanical to Neural Transduction in the Auditory Receptor," *J. Acoust. Soc. Am.*, vol. 55, no. 5, pp. 1055–1060, 1974, doi: 10.1121/1.1914647.
- [34] R. F. Lyon, "The Inner Hair Cell," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 320–330.
- [35] R. F. Lyon, "The AGC Loop Filter," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 331–344.
- [36] A. Saremi, R. Beutelmann, M. Dietz, G. Ashida, J. Kretzberg, and S. Verhulst, "A Comparative Study of Seven Human Cochlear Filter Models," *J. Acoust. Soc. Am.*, vol. 140, no. 3, pp. 1618–1634, 2017, doi: 10.1121/1.4960486.
- [37] C. S. Thakur, T. J. Hamilton, J. Tapson, A. van Schaik, and R. F. Lyon, "Live Demonstration: FPGA implementation of the CAR Model of the cochlea," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1853–

- 1856, doi: 10.1109/ISCAS.2014.6865519.
- [38] R. Meddis, "MAP 1_14 Model description." Meddis, Ray - University of Essex, Colchester, pp. 1–32, 2012.
 - [39] R. K. Singh, "RTAP: Towards a Real-Time Auditory Periphery Simulation," in *International Conference on Future Computational Technologies*, 2015, pp. 52–57, doi: 10.17758/UR.U0315218.
 - [40] R. K. Singh, "A Real-Time Implementation of a Dual Resonance Nonlinear Filterbank," in *International Conference on Engineering and Natural Sciences*, 2015, pp. 36–41, doi: <http://iierdl.org/proceeding.php?pid=24>.
 - [41] R. Meddis and L. P. O'Mard, "Virtual Pitch in a Computational Physiological Model," *J. Acoust. Soc. Am.*, vol. 120, no. 6, pp. 3861–3869, 2006, doi: 10.1121/1.2372595.
 - [42] R. F. Lyon, "A Pole-Zero Filter Cascade Provides Good Fits to Human Masking Data and to Basilar Membrane and Neural Data," in *AIP Conference Proceedings*, 2011, vol. 1403, pp. 224–229, doi: 10.1063/1.3658090.
 - [43] ASA, "ASA Standard Term Database," 2016. <https://asastandards.org/asa-standard-term-database/> (accessed Sep. 26, 2018).
 - [44] A. Seebeck, "Beobachtungen über einige Bedingungen der Entstehung von Tönen," *Ann. Phys.*, vol. 129, no. 7, pp. 417–436, 1841, doi: 10.1002/andp.18411290702.
 - [45] J. W. Strutt, *The Theory of Sound: Volume 1*, 1st ed. London, UK: Macmillan and Co., 1877.
 - [46] G. S. Ohm, "Ueber die Definition des Tones, nebst daran geknüpfter Theorie der Sirene und ähnlicher tonbildender Vorrichtungen," *Ann. Phys.*, vol. 135, no. 8, pp. 513–565, 1843, doi: 10.1002/andp.18431350802.
 - [47] A. Seebeck, "Ueber die Sirene," *Ann. Phys.*, vol. 136, no. 12, pp. 449–481, 1843, doi: 10.1002/andp.18431361202.
 - [48] G. S. Ohm, "Noch ein Paar Worte über die Definition des Tones," *Ann. Phys.*, vol. 138, no. 5, pp. 1–18, 1844, doi: 10.1002/andp.18441380503.
 - [49] H. F. Helmholtz, "On the Sensations of Tone as a Physiological Basis for the Theory of Music," *Nature*, no. September, pp. 449–452, 1875, doi: 10.1038/012449a0.
 - [50] G. von Békésy, "Zur Theorie des Hörens; die Schwingungsform der Basilarmembran," *Phys. Zeits.*, vol. 29, pp. 793–810, 1928.
 - [51] J. F. Schouten, R. J. Ritsma, and B. L. Cardozo, "Pitch of the Residue," *J. Acoust. Soc. Am.*, vol. 34, no. 9B, pp. 1418–1424, 1962, doi: 10.1121/1.1918360.
 - [52] R. J. Ritsma, "Existence Region of the Tonal Residue II," *J. Acoust. Soc. Am.*, vol. 35, no. 8, pp. 1241–1244, 1963, doi: 10.1121/1.1918679.
 - [53] R. J. Ritsma, "Existence Region of the Tonal Residue. I," *J. Acoust. Soc. Am.*, vol. 34, no. 9A, pp. 1224–1229, 1962, doi: 10.1121/1.1918307.
 - [54] J. C. R. Licklider, "'Periodicity' Pitch and 'Place' Pitch," *J. Acoust. Soc. Am.*, vol. 26, no. 5, pp. 945–945, 1954, doi: 10.1121/1.1928005.
 - [55] W. R. Thurlow and A. M. Small, "Pitch perception for Certain Periodic Auditory

- Stimuli," *J. Acoust. Soc. Am.*, vol. 27, no. 1, pp. 132–137, 1955, doi: 10.1121/1.1907473.
- [56] R. D. Patterson, "Noise Masking of a Change in Residue Pitch," *J. Acoust. Soc. Am.*, vol. 45, no. 6, pp. 1520–1524, 1969, doi: 10.1121/1.1911632.
 - [57] J. F. Schouten, "The Perception of Pitch," *Philips Tech. Rev.*, vol. 5, no. 10, pp. 286–295, 1940.
 - [58] F. L. Wightman and D. M. Green, "The Perception of Pitch," *Am. Sci.*, vol. 62, no. 2, pp. 208–215, 1974, doi: 10.1093/oxfordhpb/9780199298457.013.0005.
 - [59] R. D. Patterson, "The effects of relative phase and the number of components on residue pitch," *J. Acoust. Soc. Am.*, vol. 53, no. 6, pp. 1565–1572, 1973, doi: 10.1121/1.1913504.
 - [60] F. L. Wightman, "Pitch and stimulus fine structure," *J. Acoust. Soc. Am.*, vol. 54, no. 2, pp. 397–406, 1973, doi: 10.1121/1.1913591.
 - [61] R. P. Carlyon and T. M. Shackleton, "Comparing the fundamental frequencies of resolved and unresolved harmonics: Evidence for two pitch mechanisms?," *J. Acoust. Soc. Am.*, vol. 95, no. 6, pp. 3541–3554, 1994, doi: 10.1121/1.409971.
 - [62] R. Meddis and L. O'Mard, "A unitary model of pitch perception," *J. Acoust. Soc. Am.*, vol. 102, no. 3, pp. 1811–1820, 1997, doi: 10.1121/1.420088.
 - [63] J. M. Brunstrom and B. Roberts, "Separate mechanisms govern the selection of spectral components for perceptual fusion and for the computation of global pitch," *J. Acoust. Soc. Am.*, vol. 107, no. 3, pp. 1566–1577, 2001, doi: 10.1121/1.428441.
 - [64] A. Gerson and J. L. Goldstein, "Evidence for a general template in central optimal processing for pitch of complex tones," *J. Acoust. Soc. Am.*, vol. 63, no. 2, pp. 498–510, 1978, doi: 10.1121/1.381750.
 - [65] H. Duifhuis, L. F. Willems, and R. J. Sluyter, "Measurement of pitch in speech: An implementation of Goldstein's theory of pitch perception," *J. Acoust. Soc. Am.*, vol. 71, no. 6, pp. 1568–1580, 1982, doi: 10.1121/1.387811.
 - [66] M. T. M. Scheffers, "Simulation of auditory analysis of pitch: An elaboration on the DWS pitch meter," *J. Acoust. Soc. Am.*, vol. 74, no. 6, pp. 1716–1725, 1983, doi: 10.1121/1.390280.
 - [67] D. Bendor and X. Wang, "The neuronal representation of pitch in primate auditory cortex," *Nature*, vol. 436, no. August, pp. 1161–1165, 2005, doi: 10.1038/nature03867.
 - [68] H. Penagos, J. R. Melcher, and A. J. Oxenham, "A Neural Representation of Pitch Salience in Nonprimary Human Auditory Cortex Revealed with Functional Magnetic Resonance Imaging," *Journal Neurosci.*, vol. 24, no. 30, pp. 6810–6815, 2004, doi: 10.1523/JNEUROSCI.0383-04.2004.
 - [69] J. Lin and W. M. Hartmann, "The pitch of a mistuned harmonic: Evidence for a template model," *J. Acoust. Soc. Am.*, vol. 103, no. 5, pp. 2608–2617, 1998, doi: 10.1121/1.422781.
 - [70] J. M. Brunstrom and B. Roberts, "Effects of asynchrony and ear of presentation on the pitch of mistuned partials in harmonic and frequency-shifted," *J. Acoust. Soc. Am.*, vol. 110, no. 1, pp. 391–401, 2001, doi: 10.1121/1.1379079.

- [71] B. Roberts and J. M. Brunstrom, "Perceptual fusion and fragmentation of complex tones made inharmonic by applying different degrees of frequency shift and spectral stretch Perceptual fusion and fragmentation of complex tones made inharmonic by applying different degrees of frequency shif," *J. Acoust. Soc. Am.*, vol. 110, no. 5, pp. 2479–2490, 2001, doi: 10.1121/1.1410965.
- [72] B. Roberts and J. M. Brunstrom, "Spectral pattern, harmonic relations, and the perceptual grouping of low-numbered components," *J. Acoust. Soc. Am.*, vol. 114, no. 4, pp. 2118–2134, 2003, doi: 10.1121/1.1605411.
- [73] L. Feng and X. Wang, "Harmonic template neurons in primate auditory cortex underlying complex sound processing," *Proc. Natl. Acad. Sci. United States Am.*, vol. 114, no. 5, pp. E840–E848, 2017, doi: 10.1073/pnas.1607519114.
- [74] J. L. Goldstein, "An optimum processor theory for the central formation of the pitch of complex tones," *J. Acoust. Soc. Am.*, vol. 54, no. 6, pp. 1496–1516, 1973, doi: 10.1121/1.1914448.
- [75] S. Shamma and D. Klein, "The case of the missing pitch templates: How harmonic templates emerge in the early auditory system," *J. Acoust. Soc. Am.*, vol. 107, no. 5, pp. 2631–2644, 2000, doi: <http://dx.doi.org/10.1121/1.428649>.
- [76] M. A. Cohen, S. Grossberg, and L. L. Wyse, "A spectral network model of pitch perception," *J. Acoust. Soc. Am.*, vol. 98, no. 2, pp. 862–879, 1995, doi: 10.1121/1.413512.
- [77] J. C. R. Licklider, "A Duplex Theory of Pitch Perception," *J. Acoust. Soc. Am.*, vol. 23, no. 1, p. 147, 1951, doi: 10.1121/1.1917296.
- [78] K. Krumbholz, R. D. Patterson, and D. Pressnitzer, "The lower limit of pitch as determined by rate discrimination," *J. Acoust. Soc. Am.*, vol. 108, no. 3, pp. 1170–1180, 2000, doi: 10.1121/1.1287843.
- [79] D. Pressnitzer, R. D. Patterson, and K. Krumbholz, "The lower limit of melodic pitch," *J. Acoust. Soc. Am.*, vol. 109, no. 5, pp. 2074–2084, 2001, doi: 10.1121/1.1359797.
- [80] R. J. Ritsma, "Frequencies Dominant in the Perception of the Pitch of Complex Sounds," *J. Acoust. Soc. Am.*, vol. 42, no. 1, pp. 191–198, 1967, doi: 10.1121/1.1910550.
- [81] R. Renken, J. E. C. Wiersinga-Post, S. Tomaskovic, and H. Duifhuis, "Dominance of missing fundamental versus spectrally cued pitch: Individual differences for complex tones with unresolved harmonics," *J. Acoust. Soc. Am.*, vol. 115, no. 5, pp. 2257–2263, 2004, doi: 10.1121/1.1690076.
- [82] C. C. Blackburn and M. B. Sachs, "Classification of Unit Types in the Anteroventral Cochlear Nucleus: PST Histograms and Regularity Analysis," *J. Neurophysiol.*, vol. 62, no. 6, pp. 1303–1329, 1989, doi: 10.1152/jn.1989.62.6.1303.
- [83] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, 1952, doi: 10.1113/jphysiol.1952.sp004764.
- [84] R. J. Macgregor, *Neural and Brain Modeling*. Academic Press, 1987.
- [85] M. J. Hewitt, R. Meddis, and T. M. Shackleton, "A computer model of a cochlear-nucleus stellate cell: responses to amplitude-modulated and pure-tone stimuli," *J. Acoust. Soc. Am.*, vol. 91, no. 4, pp. 2096–2109, 1992, doi: 10.1121/1.403696.

- [86] D. Oertel, "Synaptic Responses and Electrical Properties of Cells in Brain Slices of the Mouse Anteroventral Cochlear Nucleus," *J. Neurosci.*, vol. 3, no. 10, pp. 2043–2053, 1983, doi: 10.1523/JNEUROSCI.03-10-02043.1983.
- [87] D. Oertel, "Use of brain slices in the study of the auditory system: Spatial and temporal summation of synaptic inputs in cells in the anteroventral cochlear nucleus of the mouse," *J. Acoust. Soc. Am.*, vol. 78, no. 1, pp. 328–333, 1985, doi: 10.1121/1.392494.
- [88] G. Langner, "Periodicity coding in the auditory system," *Hear. Res.*, vol. 60, no. 2, pp. 115–142, 1992, doi: 10.1016/0378-5955(92)90015-F.
- [89] A. de Cheveigné, "Cancellation model of pitch perception," *J. Acoust. Soc. Am.*, vol. 103, no. 3, pp. 1261–1271, 1998, doi: 10.1121/1.423232.
- [90] S. J. Jones and J. C. Van der Poel, "Binaural interaction in the brain-stem auditory evoked potential: evidence for a delay line coincidence detection mechanism," *Electroencephalogr. Clin. Neurophysiol.*, vol. 77, no. 3, pp. 214–224, 1990, doi: 10.1016/0168-5597(90)90040-K.
- [91] P. X. Joris, P. H. Smith, and T. C. T. Yin, "Coincidence Detection in the Auditory System: 50 Years after Jeffress," *Neuron*, vol. 21, no. 6, pp. 1235–1238, 1998, doi: 10.1016/S0896-6273(00)80643-1.
- [92] A. de Cheveigné and D. Pressnitzer, "The case of the missing delay lines: Synthetic delays obtained by cross-channel phase interaction," *J. Acoust. Soc. Am.*, vol. 119, no. 6, pp. 3908–3918, 2006, doi: 10.1121/1.2195291.
- [93] L. R. Rabiner, "On the Use of Autocorrelation Analysis for Pitch Detection," *IEEE Trans. Acoust.*, vol. 25, no. 1, pp. 24–33, 1977, doi: 10.1109/TASSP.1977.1162905.
- [94] Y. Ando, T. Okano, and Y. Takezoe, "The running autocorrelation function of different music signals relating to preferred temporal parameters of sound fields," *J. Acoust. Soc. Am.*, vol. 86, no. 2, pp. 644–649, 1989, doi: 10.1121/1.398242.
- [95] A. de Cheveigné, "Separation of concurrent harmonic sounds: Fundamental frequency estimation and a time-domain cancellation model of auditory processing," *J. Acoust. Soc. Am.*, vol. 93, no. 6, pp. 3271–3290, 1993, doi: 10.1121/1.405712.
- [96] A. de Cheveigné, "Concurrent vowel identification. III. A neural model of harmonic interference cancellation," *J. Acoust. Soc. Am.*, vol. 101, no. 5, pp. 2857–2865, 1997, doi: 10.1121/1.419480.
- [97] K. D. Martin, "Automatic Transcription of Simple Polyphonic Music: Robust Front End Processing," 1996. [Online]. Available: http://www.music.mcgill.ca/~ich/classes/mumt611_05/transcription/kdm-TR399.pdf.
- [98] A. Klapuri, "Multipitch Analysis of Polyphonic Music and Speech Signals Using an Auditory Model," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 16, no. 2, pp. 255–266, 2008, doi: 10.1109/TASL.2007.908129.
- [99] A. Klapuri, "Auditory Model-Based Methods for Multiple Fundamental Frequency Estimation," in *Signal Processing Methods for Music Transcription*, A. Klapuri and M. Davy, Eds. New York, USA: Springer US, 2006, pp. 229–265.
- [100] R. Meddis and M. J. Hewitt, "Virtual pitch and phase sensitivity of a computer model of the auditory periphery. I: Pitch identification," *J. Acoust. Soc. Am.*, vol. 89, no. 6, pp. 2866–2882, 1991, doi: 10.1121/1.400725.

- [101] A. de Cheveigné, "Pitch Perception Models," in *Pitch: Neural Coding and Perception*, C. J. Plack, A. J. Oxenham, R. R. Fay, and A. N. Popper, Eds. New York, NY, USA: Springer Science+Business Media LLC, 2005, pp. 169–233.
- [102] F. L. Wightman, "The pattern-transformation model of pitch," *J. Acoust. Soc. Am.*, vol. 54, no. 2, pp. 407–416, 1973, doi: 10.1121/1.1913592.
- [103] F. A. Bilsen, "Pitch of noise signals: Evidence for a 'central spectrum,'" *J. Acoust. Soc. Am.*, vol. 61, no. 1, pp. 150–161, 1977, doi: 10.1121/1.381276.
- [104] E. Terhardt, "Pitch, consonance, and harmony," *J. Acoust. Soc. Am.*, vol. 55, no. 5, pp. 1061–1069, 1974, doi: 10.1121/1.1914648.
- [105] A. van Schaik and R. Meddis, "Analog very large-scale integrated (VLSI) implementation of a model of amplitude-modulation sensitivity in the auditory brainstem," *J. Acoust. Soc. Am.*, vol. 105, no. 2, pp. 811–821, 1999, doi: 10.1121/1.426270.
- [106] ASA, "Timbre." <https://asastandards.org/Terms/timbre/> (accessed Apr. 23, 2019).
- [107] J. Marozeau, A. de Cheveigné, S. Mcadams, and S. Winsberg, "The dependency of timbre on fundamental frequency," *J. Acoust. Soc. Am.*, vol. 114, no. 5, pp. 2946–2957, 2003, doi: 10.1121/1.1618239.
- [108] M. Fabiani and A. Friberg, "Influence of pitch, loudness, and timbre on the perception of instrument dynamics," *J. Acoust. Soc. Am.*, vol. 130, no. 4, pp. EL193–EL199, 2011, doi: 10.1121/1.3633687.
- [109] H. L. F. Helmholtz, "On the Differences in the Quality of Musical Tones.," in *The Sensations of Tone as a Physiological Basis for the Theory of Music.*, New York, NY, US: Longmans, Green and Co, 1875, pp. 106–173.
- [110] W. H. Lichte, "Attributes of Complex Tones," The University of Iowa, 1940.
- [111] A. Gabrielsson and H. Sjögren, "Perceived sound quality of sound-reproducing systems," *J. Acoust. Soc. Am.*, vol. 65, no. 4, pp. 1019–1033, 1979, doi: 10.1121/1.382579.
- [112] P. Goad, "Sharpness: A perceptually based measure of the spectral dimension of musical timbre," *J. Acoust. Soc. Am.*, vol. 95, no. 5, p. 2958, 1994, doi: 10.1121/1.409051.
- [113] W. De Baene, A. Vandierendonck, M. Leman, A. Widmann, and M. Tervaniemi, "Roughness perception in sounds : behavioral and ERP evidence," *Biol. Psychol.*, vol. 67, no. 3, pp. 319–330, 2004, doi: 10.1016/j.biopsycho.2004.01.003.
- [114] H. Fastl and E. Zwicker, "Critical Bands and Excitation," in *Psychoacoustics: Facts and Models*, Springer-Verlag Berlin Heidelberg, 2007, pp. 149–173.
- [115] H. Fastl and E. Zwicker, "Roughness," in *Psychoacoustics: Facts and Models*, 3rd ed., Springer-Verlag Berlin Heidelberg, 2007, pp. 257–264.
- [116] A. Gabrielsson, B. Hagerman, T. Bech-Kristensen, and G. Lundberg, "Perceived sound quality of reproductions with different frequency responses and sound levels," *J. Acoust. Soc. Am.*, vol. 88, no. 3, pp. 1359–1366, 1990, doi: 10.1121/1.399713.
- [117] K. W. Berger, "Some Factors in the Recognition of Timbre," *J. Acoust. Soc. Am.*, vol. 36, no. 10, pp. 1888–1891, 1964, doi: 10.1121/1.1919287.

- [118] L. Wedin and G. Goude, "Dimension Analysis of the Perception of Instrumental Timbre," *Scand. J. Psychol.*, vol. 13, no. 1, pp. 228–240, 1972, doi: 10.1111/j.1467-9450.1972.tb00071.x.
- [119] J. M. Grey, "Timbre discrimination in musical patterns," *J. Acoust. Soc. Am.*, vol. 64, no. 2, pp. 467–472, 1978, doi: 10.1121/1.382018.
- [120] P. Iverson and C. L. Krumhansl, "Isolating the dynamic attributes of musical timbre," *J. Acoust. Soc. Am.*, vol. 94, no. 5, pp. 2595–2603, 1993, doi: 10.1121/1.407371.
- [121] R. Plomp, "The Role of Modulation in Hearing," in *Hearing-Physiological Bases and Psychophysics*, Springer, Berlin, Heidelberg, 1983, pp. 270–276.
- [122] E. Terhardt, "On the Perception of Periodic Sound fluctuations (Roughness)," *Acta Acust. united with Acust.*, vol. 30, no. 4, pp. 201–213, 1974.
- [123] C. E. Schreiner, J. Mendelson, M. W. Raggio, M. Brosch, and K. Krueger, "Temporal processing in cat primary auditory cortex.," *Acta Otolaryngol. Suppl.*, vol. 532, pp. 54–60, 1997, doi: 10.3109/00016489709126145.
- [124] S. A. Shamma and H. Versnel, "Ripple Analysis in Ferret Primary Auditory Cortex . II . Prediction of Unit Responses to Arbitrary Spectral Profiles," *Audit. Neurosci.*, vol. 1, pp. 255–270, 1995.
- [125] L. R. Rabiner and R. W. Schafer, *Introduction to Digital Speech Processing*, vol. 1. Hanover, MA, USA: Publishers Inc., 2007.
- [126] S. Shamma, "Encoding Sound Timbre in the Auditory System," *IETE J. Res.*, vol. 49, no. 2, pp. 145–156, 2003, doi: 10.1080/03772063.2003.11416333.
- [127] T. Chi, P. Ru, and S. A. Shamma, "Multiresolution Spectrotemporal Analysis of Complex Sounds," *J. Acoust. Soc. Am.*, vol. 118, no. 2, pp. 887–906, 2005, doi: 10.1121/1.1945807.
- [128] N. F. Viemeister, "Temporal modulation transfer functions based upon modulation thresholds," *J. Acoust. Soc. Am.*, vol. 66, no. 5, pp. 1364–1380, 1979, doi: 10.1121/1.383531.
- [129] D. M. Green, "'Frequency' and the Detection of Spectral Shape Change," in *Auditory Frequency Selectivity*, B. C. J. Moore and R. D. Patterson, Eds. New York, London: Plenum Press, 1986, pp. 351–359.
- [130] T. Chi, Y. Gao, M. C. Guyton, P. Ru, and S. Shamma, "Spectro-temporal modulation transfer functions and speech intelligibility," *J. Acoust. Soc. Am.*, vol. 106, no. 5, pp. 2719–2732, 1999, doi: 10.1121/1.428100.
- [131] T. Dau, B. Kollmeier, and A. Kohlrausch, "Modeling auditory processing of amplitude modulation . I. Detection and masking with narrow-band carriers," *J. Acoust. Soc. Am.*, vol. 102, no. 5, pp. 2892–2905, 1997, doi: 10.1121/1.420344.
- [132] S. Amagai, R. J. Dooling, S. Shamma, T. L. Kidd, and B. Lohr, "Detection of modulation in spectral envelopes and linear-rippled noises by budgerigars (*Melopsittacus undulatus*)," *J. Acoust. Soc. Am.*, vol. 105, no. 3, pp. 2029–2035, 1999, doi: 10.1121/1.426736.
- [133] M. Sahani and J. F. Linden, "How Linear are Auditory Cortical Responses?," in *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2002, pp. 109–116, doi: 10.1124/dmd.105.005157.concerning.

- [134] R. C. DeCharms, D. T. Blake, and M. M. Merzenich, "Optimizing Sound Features for Cortical Neurons," *Science* (80-.), vol. 280, no. 5368, pp. 1439–1443, 1998, [Online]. Available: <https://www.jstor.org/stable/2895918>.
- [135] C. K. Machens, M. S. Wehr, and A. M. Zador, "Linearity of Cortical Receptive Fields Measured with Natural Sounds," *J. Neurosci.*, vol. 24, no. 5, pp. 1089–1100, 2004, doi: 10.1523/JNEUROSCI.4445-03.2004.
- [136] S. A. Shamma, H. Versnel, and N. Kowalski, "Ripple Analysis in Ferret Primary Auditory Cortex. I. Response Characteristics of Single Units to Sinusoidally Rippled Spectra," Maryland, USA, 1994.
- [137] N. Kowalski, D. A. Depireux, and S. A. Shamma, "Analysis of Dynamic Spectra in Ferret Primary Auditory Cortex. I. Characteristics of Single-Unit Responses to Moving Ripple Spectra," *J. Neurophysiol.*, vol. 76, no. 5, pp. 3503–3523, 1996, [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/8930290><http://www.ncbi.nlm.nih.gov/pubmed/8930289>.
- [138] D. J. Klein, D. A. Depireux, J. Z. Simon, and S. A. Shamma, "Robust spectrotemporal reverse correlation for the auditory system: Optimizing stimulus design," *J. Comput. Neurosci.*, vol. 9, no. 1, pp. 85–111, 2000, doi: 10.1023/A:1008990412183.
- [139] H. Versnel and S. A. Shamma, "Spectral-ripple representation of steady-state vowels in primary auditory cortex," *J. Acoust. Soc. Am.*, vol. 103, no. 5, pp. 2502–2514, 1998, doi: 10.1121/1.422771.
- [140] O. Bar-yosef, Y. Rotman, and I. Nelken, "Responses of Neurons in Cat Primary Auditory Cortex to Bird Chirps: Effects of Temporal and Spectral Context," *J. Neurosci.*, vol. 22, no. 19, pp. 8619–8632, 2002, doi: 10.1523/JNEUROSCI.22-19-08619.2002.
- [141] T. Dau, B. Kollmeier, and A. Kohlrausch, "Modeling auditory processing of amplitude modulation. II. Spectral and temporal integration," *J. Acoust. Soc. Am.*, vol. 102, no. 5, pp. 2906–2919, 1997, doi: 10.1121/1.420345.
- [142] R. Patterson, I. Nimmo-smith, J. Holdsworth, and P. Rice, "An efficient auditory filterbank based on the gammatone function." Speech-Group meeting of Institute of Acoustics on Auditory Modelling, RSRE, Malvern, pp. 1–33, 1987.
- [143] M. Slaney, "An Efficient Implementation of the Patterson-Holdsworth Auditory Filter Bank," 1993.
- [144] T. Dau, D. Püschel, and A. Kohlrausch, "A quantitative model of the "effective" signal processing in the auditory system. I. Model structure," *J. Acoust. Soc. Am.*, vol. 99, no. 6, pp. 3615–3622, 1996, doi: 10.1121/1.414959.
- [145] S. A. Shamma and H. Versnel, "Ripple Analysis in Ferret Primary Auditory Cortex . III . Prediction of Unit Responses to Arbitrary Spectral Profiles," 1995.
- [146] M. Slaney, "Auditory Toolbox Version 2," 1998. [Online]. Available: <https://engineering.purdue.edu/~malcolm/interval/1998-010/AuditoryToolboxTechReport.pdf>.
- [147] J. P. Jones and L. A. Palmer, "An Evaluation of the Two-Dimensional Gabor Filter Model of Simple Receptive Fields in Cat Striate Cortex," *J. Neurophysiol.*, vol. 58, no. 6, pp. 1233–1258, 1987.

- [148] D. A. Depireux, J. Z. Simon, D. J. Klein, and S. A. Shamma, "Spectro-Temporal Response Field Characterization with Dynamic Ripples in Ferret Primary Auditory Cortex," *J. Neurophysiol.*, vol. 85, no. 3, pp. 1220–1234, 2001.
- [149] J. Tchorz and B. Kollmeier, "A model of auditory perception as front end for automatic speech recognition," *J. Acoust. Soc. Am.*, vol. 106, no. 4, pp. 2040–2050, 1999, doi: 10.1121/1.427950.
- [150] M. Hansen and B. Kollmeier, "Continuous assessment of time-varying speech quality," *J. Acoust. Soc. Am.*, vol. 106, no. 5, pp. 2888–2899, 1999, doi: 10.1121/1.428136.
- [151] I. Holube and B. Kollmeier, "Speech intelligibility prediction in hearing-impaired listeners based on a psychoacoustically motivated perception model Speech intelligibility prediction in hearing-impaired listeners based on a psychoacoustically motivated perception," *J. Acoust. Soc. Am.*, vol. 100, no. 3, pp. 1703–1716, 1996, doi: 10.1121/1.417354.
- [152] T. Piechowiak, S. D. Ewert, T. Dau, T. Piechowiak, S. D. Ewert, and T. Dau, "Modeling comodulation masking release using an equalization-cancellation mechanism Modeling comodulation masking release using an equalization-cancellation mechanism," *J. Acoust. Soc. Am.*, vol. 121, no. 4, pp. 2111–2126, 2007, doi: 10.1121/1.2534227.
- [153] R. Huber and B. Kollmeier, "PEMO-Q — A New Method for Objective Audio Quality Assessment Using a Model of Auditory Perception," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 14, no. 6, pp. 1902–1911, 2006, doi: 10.1109/TASL.2006.883259.
- [154] J. H. Plasberg and W. B. Kleijn, "The Sensitivity Matrix: Using Advanced Auditory Models in Speech and Audio Processing," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 15, no. 1, pp. 310–319, 2007, doi: 10.1109/TASL.2006.876722.
- [155] J. Breebaart, S. Van De Par, and A. Kohlrausch, "Binaural processing model based on contralateral inhibition. I. Model structure," *J. Acoust. Soc. Am.*, vol. 110, no. 2, pp. 1074–1088, 2001, doi: 10.1121/1.1383297.
- [156] J. Breebaart, S. Van De Par, and A. Kohlrausch, "Binaural processing model based on contralateral inhibition. II. Dependence on spectral parameters," *J. Acoust. Soc. Am.*, vol. 110, no. 2, pp. 1089–1104, 2001, doi: 10.1121/1.1383298.
- [157] J. Breebaart, S. Van De Par, and A. Kohlrausch, "Binaural processing model based on contralateral inhibition. III. Dependence on temporal parameters," *J. Acoust. Soc. Am.*, vol. 110, no. 2, pp. 1105–1117, 2001, doi: 10.1121/1.1383299.
- [158] T. Chi and S. A. Shamma, "Spectrum Restoration From Multiscale Auditory Phase Singularities by Generalized Projections," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 14, no. 4, pp. 1179–1192, 2006, doi: 10.1109/TSA.2005.860828.
- [159] M. Elhilali and S. Shamma, "Information-bearing components of speech intelligibility under babble-noise and bandlimiting distortions," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 4205–4208, doi: 10.1109/ICASSP.2008.4518582.
- [160] G. Wolf, S. Mallat, and S. Shamma, "Audio source separation with time-frequency velocities," in *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2014, pp. 1–6, doi: 10.1109/MLSP.2014.6958893.

- [161] M. S. Modulations, N. Mesgarani, S. Member, M. Slaney, S. Member, S. A. Shamma, and S. Member, "Discrimination of Speech From Nonspeech Based on Multiscale Spectro-Temporal Modulations," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 14, no. 3, pp. 920–930, 2006, doi: 10.1109/TSA.2005.858055.
- [162] N. Mesgarani and S. Shamma, "Denoising in the Domain of Spectrotemporal Modulations," *EURASIP J. Audio, Speech, Music Process.*, vol. 2007, pp. 1–8, 2007, doi: 10.1155/2007/42357.
- [163] N. Mesgarani and S. Shamma, "Speech processing with a cortical representation of audio," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5872–5875, doi: 10.1109/ICASSP.2011.5947697.
- [164] N. Mesgarani, S. David, and S. Shamma, "Representation of Phonemes in Primary Auditory Cortex: How the Brain Analyzes Speech," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, 2007, pp. 765–768, doi: 10.1109/ICASSP.2007.367025.
- [165] S. Wu, T. H. Falk, and W. Chan, "Automatic speech emotion recognition using modulation spectral features," *Speech Commun.*, vol. 53, no. 5, pp. 768–785, 2011, doi: 10.1016/j.specom.2010.08.013.
- [166] B. N. Pasley, S. V. David, N. Mesgarani, A. Flinker, S. A. Shamma, E. Nathan, R. T. Knight, and E. F. Chang, "Reconstructing Speech from Human Auditory Cortex," *PLOS Biol.*, vol. 10, no. 1, pp. 1–13, 2012, doi: 10.1371/journal.pbio.1001251.
- [167] K. Patil, D. Pressnitzer, S. Shamma, and M. Elhilali, "Music in Our Ears: The Biological Bases of Musical Timbre Perception," *PLoS Comput. Biol.*, vol. 8, no. 11, pp. 1–16, 2012, doi: 10.1371/journal.pcbi.1002759.
- [168] J. Burred, A. Robel, and T. Sikora, "Dynamic Spectral Envelope Modeling for Timbre Analysis of Musical Instrument Sounds," *Audio, Speech, Lang. ...*, vol. 18, no. 3, pp. 663–674, 2010, doi: 10.1109/TASL.2009.2036300.
- [169] A. Eronen and A. Klapuri, "Musical Instrument Recognition using Cepstral Coefficients and Temporal Features," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, 2000, pp. 753–756, doi: 10.1109/ICASSP.2000.859069.

3. CAR-Lite: A Multi-Rate Cochlear Model

The cochlear models reviewed in chapter 2 operate at a single sampling rate. This chapter presents a cochlear model operating at multiple sampling rates, which is implementable on hardware. Out of the two cochlear models reviewed in chapter 2, the cascade of asymmetric resonator (CAR) filter configuration from the CAR-FAC model is selected to be implemented with multi-rate operation. The reason behind this selection is because the CAR configuration [1] characterises most biological features when compared to other configurations [2] and enables real-time implementation [3]–[5].

3.1. Motivation

The degree of sophistication of an auditory cochlear model configuration is dependent on the signal processing algorithm used in the acquisition of spectral information from a sound signal. With a sophisticated cochlear model, digital hardware such as an FPGA running an algorithm using a number of filters running in the time domain to capture these spectra is affected by the significant number of filter coefficients that require non-volatile storage. This characteristic, in turn, impacts memory and silicon area costs as well as power consumed by the hardware, which is a significant factor for signal processing engineers to avoid using auditory models to develop real-world applications. Instead, they rely heavily on computationally cost-saving algorithm such as the fast Fourier transform (FFT) [6] for hardware implementation [7], [8], which represents spectral information of a sound signal in the frequency domain [9], [10].

In this chapter, an attempt is made to reduce the computational costs of an auditory model by introducing a multi-sampling-rate cochlear model, as per Occam’s Razor principle. As observed in section 3.2, the multi-sampling-rate used across octaves enables coefficients from one octave to be reused for the other octaves, which leads to the reduction of filter coefficients as well as digital hardware computational resources on the FPGA. Because of its small scale and simplicity, the cochlear model output can be fitted with other sophisticated algorithms to process perceptual sound cues. As an illustration, the multi-rate cochlear model from section 3.2 is modified with a novel spiking algorithm to capture sound intensities, which is presented in section 3.3. The multi-rate cochlear model is also used as a frontend model cascaded with other models to characterise pitch and timbre cues, which are described from chapters 4 to 7.

3.2. A Multi-Rate Cochlear Model

This section presents a multi-sampling rate cochlear model. Subsection 3.2.1 revisits the CAR model, where more of the model’s characteristics are presented before subsection 3.2.2 introduces the multi-rate model. Subsections 3.2.3 and 3.2.4 present an implementation of the model on FPGA. Subsection 3.2.5 discusses the software and hardware characteristics of the model, and subsection 3.2.6 compares filter coefficients usage of the model with the CAR model. Subsection 3.2.7 presents the response of the model to a log chirp signal while subsection 3.2.8 presents the model response to a musical signal.

3.2.1. CAR Model Revisited

Chapter 2 provides an overview of the CAR-FAC model. This section presents an expansive view of the CAR segment of the CAR-FAC model, which is a precursor to the next section. As described in chapter 2, the CAR model [11], [12] uses a cascade filterbank to model sound wave propagation of the basilar membrane in the cochlea [13]. Each filter in the filterbank is known as an asymmetric resonator (AR) [14], which is configured as a two-pole-two-zero filter with an infinite impulse response (IIR). The AR has an asymmetric gain response shaped like a skewed bell-shaped curve, which is defined by a shallow gradient before its centre frequency (CF) and a steep gradient after the CF.

Coefficients of the AR placed at the start of the cascade filterbank result in a resonance corresponding to a high CF. ARs placed after that, have coefficients resulting in decreasing CFs. The transfer function of an AR [1], $H(z)$, is defined in chapter 2 but is reiterated here for clarity:

$$H(z) = \frac{y}{x} = g \left(\frac{z^2 + (-2a + hc)rz + r^2}{z^2 - 2arz + r^2} \right) \quad (3-1)$$

where x is the input signal; y is the output signal; coefficients a and c are the real and imaginary components of a complex signal. They are computed as follow:

$$a = \cos\left(2\pi \frac{f_c}{f_s}\right) \quad (3-2)$$

$$c = \sin\left(2\pi \frac{f_c}{f_s}\right) \quad (3-3)$$

where f_c is the centre frequency of the filter and f_s is the sampling rate. Coefficients h controls the distance of zeros from the frequency of the poles and is set to the real components of the ringing frequency, c . r is the radius of the poles and zeros in the z -plane and g is an overall unity gain at DC defined by:

$$g = \frac{1 - 2ar + r^2}{1 - (2a - h)r + r^2} \quad (3-4)$$

3.2.2. CAR-Lite

The CAR-Lite cochlear model is based on the single sampling rate CAR model but has multiple sampling rates. Figure 3-1 displays its configuration. Each filter in CAR-Lite is organised in a closed-couple form [1] comprising two split first-order difference equations from the transfer function defined in equation (3-1). Calculation of the coefficients in CAR-Lite is identical to that of the CAR model [1], [3]–[5], [12] for the first 12 sections, i.e. for the highest octave, o_1 . We used 12 sections as there are 12 musical notes in an octave from A to G# (7 major notes from A to G and 5 sharp notes from A# to G#). This note range applies to all octaves for any pitched musical instrument in the RWC database [15], which are used in the classification exercises in chapters 6 and 7. The CFs of the 12 ARs in an octave are equally spaced on a log scale spanning from $f_s/4$ to $f_s/8$. The ratio of the CF to the

bandwidth, known as Q , of every AR remains the same. In other words, the CAR-Lite model has a logarithmically spaced constant- Q filterbank.

Coefficients a , c , and g of the 12 ARs in o_1 are pre-computed with the settings in 3.2.3. Coefficient r is calculated for only the 12 ARs in o_1 , which are then reused for all other octaves:

$$r = 1 - \zeta 2\pi f_c / f_s \quad (3-5)$$

where ζ is the damping factor set at 0.28 to ensure minimum peak gain response error between neighbouring filter sections calculated from equation (3-1) to maintain constant gain responses across the 12 ARs. The 12 r values are empirically modified to reduce the errors further. This modification is done by iteratively adjusting r for every neighbouring pair of filter sections up to 3 decimal places until the errors of their respective gains are minimal.

At the next lower octave, o_2 , the same set of coefficients, namely a , c , g , and r , are reused, but the sampling rate is halved. This technique ensures that the ARs in o_2 have the same gain response as the ARs in o_1 . However, the centre frequencies (CFs) of the ARs in o_2 are halved with respect to the CFs of the ARs in o_1 in accordance with sound wave propagation in BM in decreasing CFs. On FPGA, this technique brings forth three advantages. Firstly, only coefficients from ARs in o_1 require storage, which conserves memory by as many as N times. Here, N refers to the number of octaves. Secondly, a halving sampling rate can be implemented simply by ignoring every second input sample to o_2 . This technique is applied to all other octaves, with the sampling rate reduced according to:

$$f_s(n) = f_s(1) / (2^{n-1}) \quad (3-6)$$

where n is the octave index number and $f_s(1)$ is the sampling rate at the highest octave, o_1 . Hence, no expensive division circuit [16], [17] is required to be implemented on FPGA.

The last advantage is that only a single AR circuit needs to be implemented on FPGA to process all the cochlear section. The use of this circuit can be pipelined through time-slicing – an allocated amount of time is given for a cochlear section to be processed by this circuit with the output sample fed to the input of the next section, which is processed by the same AR circuit for the same duration of allocated time. In this way, no extra circuit is required to process multiple cochlear sections.

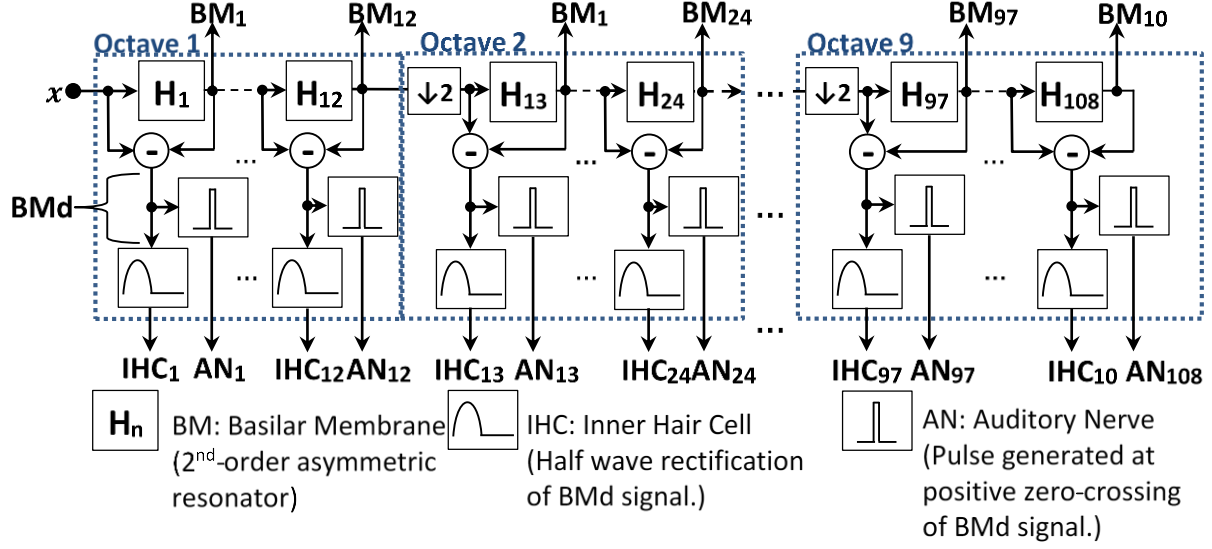


Figure 3-1: The CAR-Lite model with the sampling rate halved at the start of each octave.

The ARs used in CAR-Lite are low-pass filters with a resonant gain around CF, but they all have a 0-dB gain at DC as observed for the low-frequency slopes in Figure 3-10(a). This characteristic can be used to represent the basilar membrane (BM) displacements. However, inner hair cells (IHCs) are sensitive to BM velocity, which means 6 dB per octave pre-CF low-frequency slopes are required. One way to compute BM velocity is to implement a temporal differentiation step by taking the difference of the output at previous and current discrete times from the same section. However, this reliance on previous time values requires additional storage. A simple solution is to take the difference between neighbouring sections since the output of the previous section is the input to the current one. Therefore, there is no need to store the previous time AR filter output, which conserves memory size equivalent to the total number of cochlear sections for the BMd signal. This lateral section difference technique is applied to the output of the CAR filters to calculate the BM velocity, BMd :

$$BMd(s, t) = y(s, t) - y(s - 1, t) \quad (3-7)$$

where $y(s, t)$ is the AR output of section s at time t . As a result, the pre-CF low-frequency slopes have the desired gains of 6 dB/octave as observed in Figure 3-10(c).

Biologically plausible models characterise the mechanical-to-electrical transduction process in the IHCs in several ways. Sumner et al. [18] modelled an IHC with a sigmoid function based on an analogue circuit to generate a voltage output to initiate the release of neurotransmitters for spike generation in the auditory nerve (AN). Carney [19], Zhang et al. [20], and Zilany et al. [21] modelled the IHC voltage output using a logarithmic compression function and a low-pass filter (LPF). Here, the LPF is a smoothing filter to remove high-frequency artefacts generated from the nonlinear functions. The nonlinear functions used in these models are fundamentally sigmoidal functions that can be abstracted as half-wave rectifiers (HWR) [22]. Such an HWR is used to model the IHC in the CAR-Lite model:

$$IHC(s, t) = \begin{cases} BMd(s, t), & BMd(s, t) \geq 0 \\ 0, & BMd(s, t) < 0 \end{cases} \quad (3-8)$$

An LPF is not used after the HWR because the IHC signal is not used as input for auditory spike generation at the next stage. Hence, introducing an LPF after the HWR needlessly adds redundant computation. Although this makes the IHC stage redundant for the CAR-Lite model, it is retained here for completeness of the output of the CAR-Lite model and to ensure that the size of the model remains small. Furthermore, the IHC signal from the CAR-Lite model is used as input to the auditory models described in section 3.3 and chapters 4 and 5, which are then low-pass filtered in adherence to the aforementioned biologically plausible models.

The auditory nerve (AN) spikes output after the IHC stage is generated at positive zero-crossings of the BMd signal, which is different from biologically plausible models, where the IHC signal is used instead of the BMd signal:

$$AN(s, t) = \begin{cases} 1, & BMd(s, t) < 0 \text{ and } BMd(s, t) \geq 0.01 \\ 0, & \text{otherwise} \end{cases} \quad (3-9)$$

The BMd signal is selected over the IHC signal at this stage because positive zero-crossings is achievable by checking only the most significant bit of the BMd signal, when it changes from 1 to 0, instead of the entire bit width when an IHC signal is used. Hence, this algorithm reduces runtime logic elements on hardware. An arbitrary amplitude threshold of 0.01 is introduced to equation (3-9) to avoid spike generation by sound signals with very low amplitude out of a full-scale amplitude range of -1 and +1. Additionally, binary spikes from equation (3-9) are capable of characterising periodicity in the input sound signal, which indicates a simple manner of accounting for pitch information as presented in chapters 4 and 6. However, the intensity level of the input sound signal cannot be characterised by such spikes. To accommodate the intensity level, section 3.3 describes a modified CAR-Lite model extended with a new algorithm.

3.2.3. Fixed-Point Implementation

The CAR-Lite model was implemented initially in floating-point arithmetic to acquire a benchmark response and subsequently, converted to fixed-point numbers in Matlab. The fixed-point model is used as the reference model for an FPGA implementation via SystemVerilog. From the floating-point implementation, the CAR-Lite model is simulated with nine octaves totalling 108 sections. The centre frequencies (CFs) from the filters span from 50 Hz to 24 kHz, which cover the range of human hearing. The sampling rate for the first octave, o_1 , is set at 96 kHz and is halved for every octave down the cascade. Therefore, the second octave, o_2 , operates at a sampling rate of 48 kHz, and the final octave, o_9 , operates at 375 Hz sampling rate.

Subsection 3.2.3.1 presents the determination of the bit widths of the coefficients of the CAR-Lite model as well as bit widths of its input and output signals. Subsection 3.2.3.2 presents the dynamic range of the model based on the set bit widths from subsection 3.2.3.1.

3.2.3.1. Bit Widths of Signals and Coefficients

In the FPGA implementation of the CAR [3] and CAR-FAC [25] models with a single sampling rate, the coefficients are set to 16 bits. Since CAR-Lite is aimed to be a simple and

small implementation of a silicon cochlea, a review of the bit widths of the fixed-point implementation is required to ensure that the coefficients are set at the minimum bit widths to generate output responses close to the floating-point model. The input signal used for testing the bit widths is a log chirp. The details of this signal are given in subsection 3.2.7. The output signal used is the AR output, representing the basilar membrane (BM) output. With this signal, the gain response of the BM output signal (from 108 ARs) is generated and compared with the floating-point gain response. Details of the gain response calculation are given in subsection 3.2.7. The AR coefficients are set separately at 8 bits, 12 bits, and 16 bits. The input signal is set at five different bit widths: 8 bits, 10 bits, 12 bits, 14 bits, and 16 bits.

Figure 3-2 displays the gain responses of the floating-point and fixed-point implementations for 8 bits AR coefficients and input signal ranging from 8 bits to 16 bits. Figure 3-3 displays the 2D correlation coefficients (CC) of the comparison between the floating-point implementation and the combinations of bit widths of the AR coefficients and the input signal cited in the previous paragraph. The gain responses are treated as 2D images to quantify the degree of similarity, which is calculated with the following equation [26]:

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \quad (3-10)$$

where r is a correlation coefficient; m is the row number; n is the column number; A and B are the two 2D matrices to be compared; \bar{A} and \bar{B} are the mean of the 2D matrices.

When 8 bits AR coefficients and 8 bits input signal are used, the gain response appears to be the noisiest as observed in Figure 3-2(b). This characteristic is corroborated by the lowest 2D correlation coefficient (CC) score of 0.569 in Figure 3-3. As the bit width of the input signal is increased, the gain response begins to appear more like the floating-point gain response. The increase of the CC also corroborates this characteristic. It is expectedly the highest for 16 bits coefficients and 16 bits input signal. Using a small bit width for the input signal at 8 bits, and scaling up the bit widths of the AR coefficients from 8 bits to 16 bits does not result in any improvement to the CC – the CC remains at its lowest at approximately 0.57 similar to when 8 bits AR coefficients and 8 bits input signal are used.

Conversely, when the input signal is fixed to 16 bits, and the bit widths of the AR coefficients are increased from 8 bits to 16 bits, the CC improves significantly by 38% as opposed to when 8 bits input signal is used. So, to ensure a high similarity between the benchmark software floating-point and hardware fixed-point responses, the bit width of the input signal should be set high, while the bit widths of the AR coefficients can be set low. Although the CCs reflected in Figure 3-3 for 8 bits AR coefficients is lower than 12 bits and 16 bits AR coefficients for 16 bits input signal, the conservation of memory is prioritised over the small differences observed in the CCs. Hence, the AR coefficients are set to 8 bits, and the input, the BM output, BMd output, and the IHC signals are set to 16 bits. The AN spike is a single bit signal.

BM Gain Responses

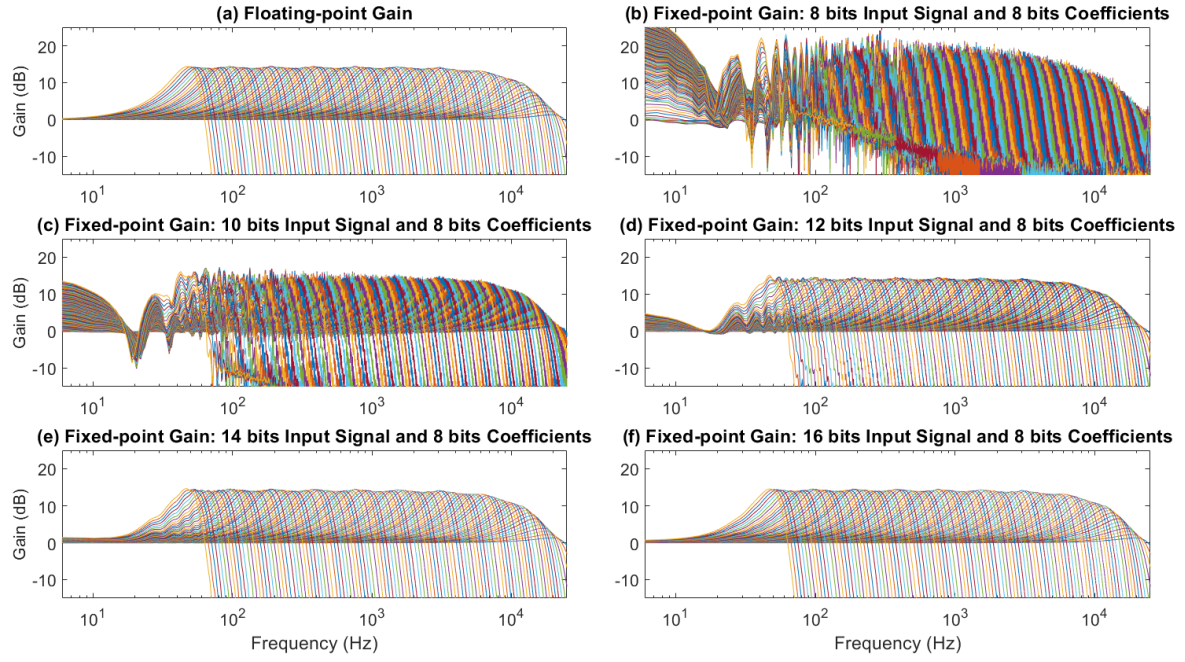


Figure 3-2: BM signal gain response of the (a) floating-point implementation and fixed-point implementation with 8 bits coefficients and (b) 8 bits, (c) 10 bits, (d) 12 bits, (e) 14 bits, and (f) 16 bits input signal.

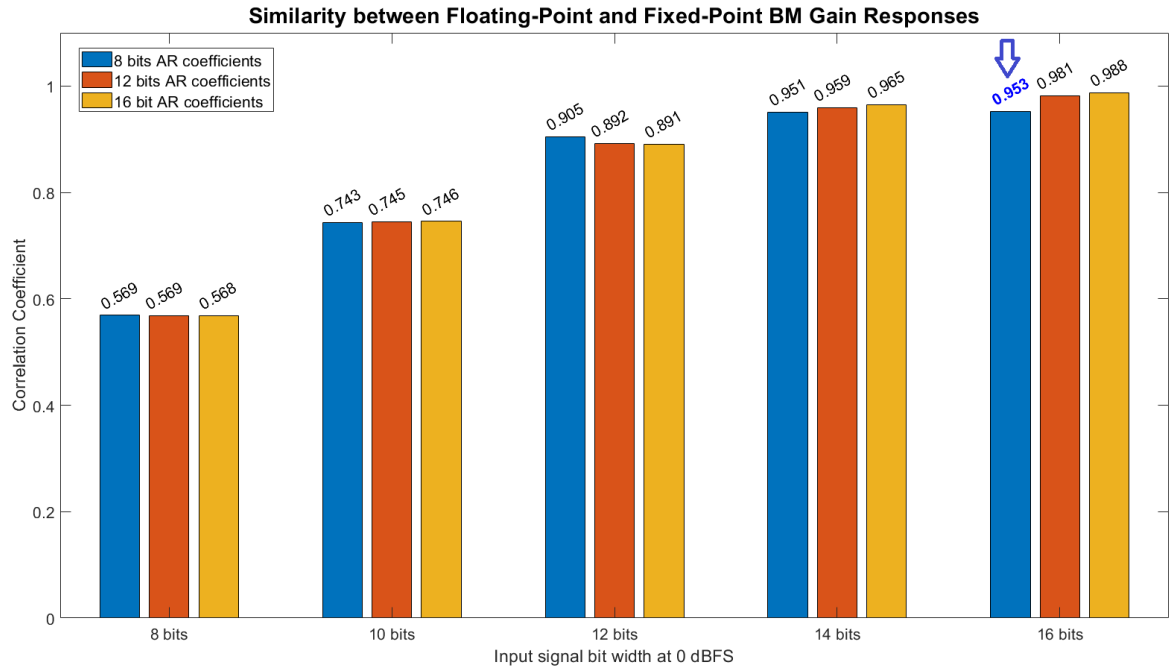


Figure 3-3: Correlation coefficients showing the similarity between the floating-point and the fixed-point BM gain responses for various AR coefficients and input signal bit widths. The input signal is at 0 dB full-scale (FS). CC score (of 0.953) in bold blue font shows the selected bit widths of the AR coefficients and input signal at 8 bits and 16 bits, respectively.

3.2.3.2. Dynamic Range

Dynamic range, DR , is the ratio of the maximum (x_{max}) and minimum (x_{min}) value that can be represented in a signal [34]. In music, the DR signifies the quality [35] of a musical signal affecting its emotional influence to a listener [36]. The DR also affects the quality of a speech signal, which impacts both normal-hearing and hearing-impaired listeners [37]. The

DR of a human auditory system is approximately 140 dB [38]. However, the DRs are lower for different categories of sound signals, i.e. 30 dB to 60 dB for speech [37], and 20 dB to 60 dB for music [39].

DR is expressed in decibels (dB) as:

$$DR = 20 \log_{10} \left(\frac{x_{max}}{x_{min}} \right) \quad (3-11)$$

where for an N bit fixed-point signal, x_{max} is calculated as 2^N , and x_{min} is the interval between two consecutive N bit numbers, i.e. 1 for 16 bits fixed-point numbers. Since the bit widths of the BM, BMd, and IHC signals are all dependent on the input signal, the input signal determines the dynamic range of the CAR-Lite model.

Using the values of the last paragraph in section 3.2.2 as an illustration, let us assume an input signal is having $x_{max} = 2$ based on the difference between the peak-to-peak amplitudes of +1 and -1, and $x_{min} = 0.01$ based on the arbitrary amplitude threshold of 0.01. Then the dynamic range of the model is 46 dB [= $20 \log(2/0.01)$ using equation (3-11)]. Consequently, each sample in the BM, BMd, and IHC signal requires a minimum of 8 bits $\approx \lceil \log_2(2/0.01) \rceil$ to be represented. However, without considering an automatic gain control (AGC), this dynamic range is insufficient in representing music [24] as well as speech [23] range of approximately 60 dB sound pressure level. Thus, a higher BMd bit width must be used.

With the input signal, BM, BMd, and IHC signal set a bit width of $N = 16$ bits, then $x_{max} = 2^N = 65,536$ and $x_{min} = 1$. The dynamic range of the model is 96 dB [= $20 \log(2^{16}/1)$]. In this case, the DR of the CAR-Lite model is larger than the DRs of speech and musical signals mentioned in the preceding paragraph. Hence, the model is capable of representing such signals, which will be seen throughout this dissertation, starting in subsection 3.2.8.

3.2.4. FPGA Implementation

An Altera Cyclone V FPGA (target FPGA chip: 5CGXFC5C6F27C7N) is used to implement the CAR-Lite model with a system clock speed of 250 MHz and an audio codec sampling rate set at 96 kHz. Figure 3-4 displays the architecture of the FPGA implementation of the CAR-Lite model. The model is characterised by two modules: a supervisor module and an equation module. The supervisor module invokes the equation module using time-multiplexing, as well as manages the flow of computation by controlling resources. These resources include the cochlear section to be processed and the transmission of its corresponding coefficients and data into the equation module. The equation module implements the AR difference equations defined by the transfer function in equation (3-1), as well as equations (3-7), (3-8), and (3-9) for BMd, IHC, and AN stages, respectively. The operations between these equations are serialised but are not pipelined to ensure a simple supervisor module circuitry.

Figure 3-5(a) illustrates the FPGA operations of the CAR-Lite model upon the arrival of an audio sample. The octave processing control (OPC) and the section processing control (SPC) in the supervisor module determines the cochlear octave number and cochlear section number to run at T0 and T1, respectively. Once determined, the supervisor module

transmits the relevant coefficients and signal samples to the equation module before going into *IDLE* mode. At T3, the equation module is awakened from an *IDLE* state and processes the cochlear equations mentioned in the preceding paragraph. Before the equation module finishes, the supervisor module pre-empt the next cochlear section to be processed and prepares its corresponding coefficients. As soon as the equation module finishes, it goes briefly into an *IDLE* state (not shown). During this time, the SPC transmits the coefficients and the input samples for the next section to be processed at T4. The reason behind this waiting is because the input to the next section is the output of the current section and thus, requires the completion of the equation module. The same operation shown between T0 to T4 is applied to all other sections until the end of an octave, as shown in Figure 3-5(b), is reached. At this time, either a new octave is invoked with the same procedure, or the operation is ceased, and the supervisor and equation modules transition into an *IDLE* state.

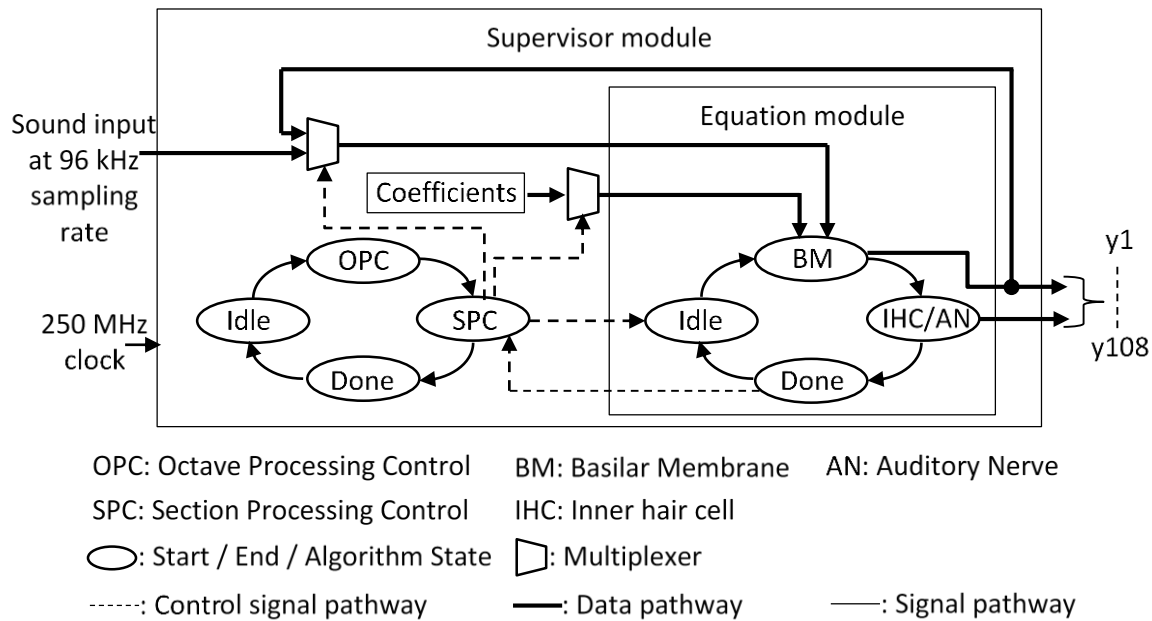


Figure 3-4: Architecture of the CAR-Lite model implemented on FPGA.

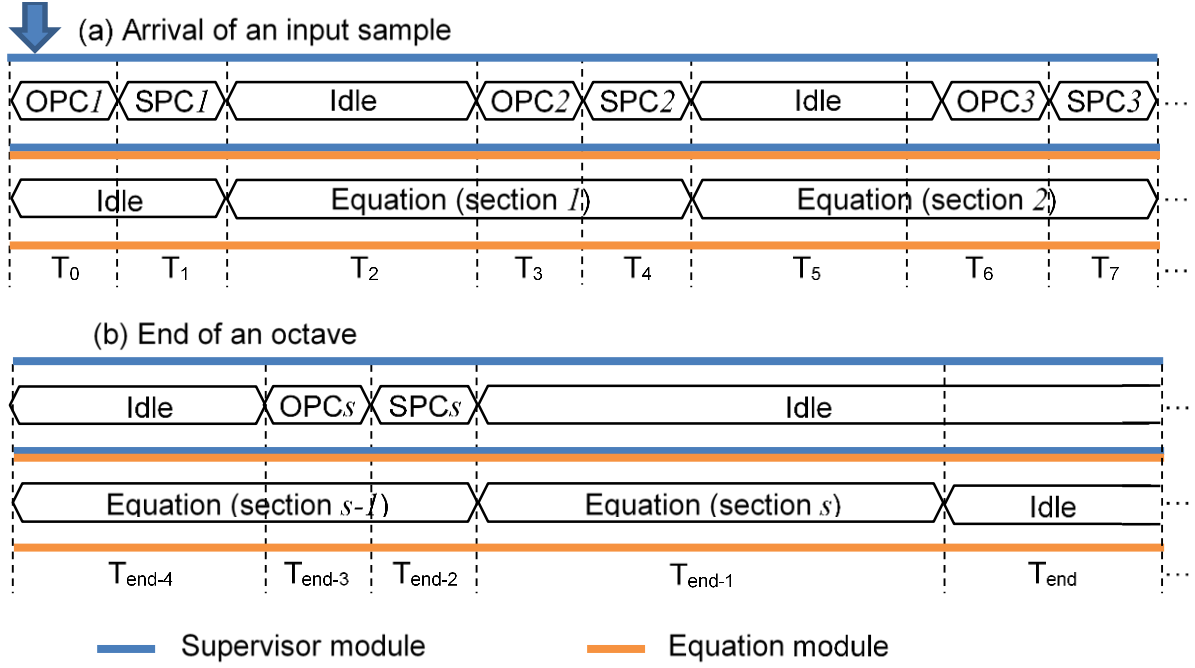


Figure 3-5: FPGA output vector waveforms of the supervisor and equation modules operating at (a) the arrival of an audio sample; and (b) the end of the processing of a cochlear octave, where 12 cochlear sections have been processed at a specific sampling rate.

3.2.4.1. Supervisor Module

The supervisor module oversees the running of the CAR-Lite model, whose equations are housed in the equation module. The module has a finite state machine (FSM) that controls which octave and which sections are processed at any time. It uses two states to achieve this: octave processing control (OPC) and section processing control (SPC). The former is explicitly used for controlling multiple sampling rates across octave groups, whereas the latter is used for selecting coefficients and data entry into the equation module.

The OPC state implements sampling rate reduction by a factor of 2 for every octave from o_2 to o_9 . These octave groups ignore every second input sample. A 9 bit register is used to administer the activation of octaves, where every 1 bit is allocated to an octave group, i.e. bit 1 to o_1 , bit 2 to octave o_2 , up to bit 9 to octave o_9 . Two conditions are required to activate an octave group as defined by the following Boolean control logic:

$$B(n, t) = B(n - 1, t) \cdot \overline{B(n, t - 2^{n-2})} \quad (3-12)$$

where B is a 9 bit register with the n^{th} bit corresponding to octave group, o_n . The first condition in equation (3-12) is that the preceding octave group o_{n-1} has already been activated, which ensures sequential octave group activation in the cascade. The second condition in equation (3-12) requires that o_n is inactive at previous discrete-time, $t - 2^{n-2}$. In the latter condition, the toggled bit generates the bypassing of every 2^{nd} input sample to o_n . These conditions apply to all octave groups except for bit 1 corresponding to o_1 . This bit is set at the arrival of an input sample. Sections in octave groups that are not processed at the present timestamp, t , simply output values from the previous timestamp, $t-1$. As an example (see Figure 3-6), to activate o_4 when the 9th input sample arrives, o_3 must be activated as

part of the first condition of equation (3-12). The second condition of equation (3-12) stipulates that o_4 must not be activated when the 5th input sample arrived. When these two conditions are true, then o_4 is activated.

Figure 3-6 displays the octave processing map for up to 19 input samples from system start-up. At the arrival of an input sample, the $xReady$ flag is set HIGH and is stored in bit 1 of register B . This enables the twelve corresponding cochlear sections from 1 to 12 in o_1 to be processed. Subsequently, the contents of B are left-shifted and bit 2 is set as HIGH to enable the AND (or multiply) operation associated with equation (3-12). But before this is done, the old value of bit 2 is read and inverted to bypass octave processing at every alternate discrete time. After the left-shift and AND operations, the final value of bit 2 determines whether cochlear sections 13 to 24, corresponding to o_2 , should run. This operation continues for the remaining 7 bits in B . Once completed, the contents of B are cleared to prepare for the next input sample.

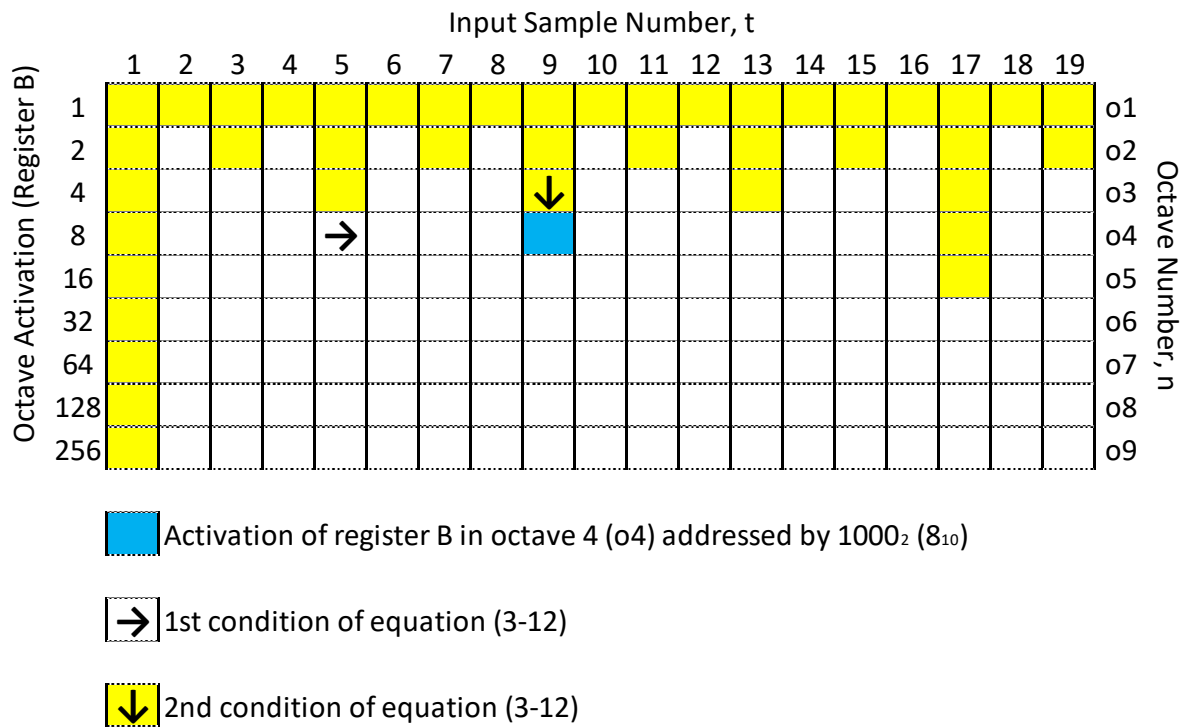


Figure 3-6: Octave processing map based on the contents of 9 bit register B from equation (3-12) for the first 19 input samples. Yellow boxes indicate octave activation and white boxes indicate no activation.

When a new octave is required to run, the OPC state explicitly updates the section index in the SPC state to be processed, which indicates the next cochlear section to be processed. Otherwise, the SPC tracks the section number to be processed by itself and only updates the OPC with the already processed cochlear section number. The SPC invokes the equation module with coefficients and input data for a cochlear section to be processed, and once, the equation module finishes, the SPC reads back the output data and stores them in memory.

3.2.4.2. Equation Module

An equation module comprises BM, BMd, IHC, and AN stages of the cochlea defined by equations (3-1), (3-7), (3-8), and (3-9), respectively. A finite state machine (FSM) is used to

achieve serial processing of these stages for a single cochlear section within the equation module. These equations are broken down into smaller parts comprising two operands. Each of these parts is processed within a discrete state of the FSM. A single cochlear section requires 20 states corresponding to 20 system clock cycles to complete processing. Individually, each BM, BMd, IHC, and AN stage in the equation module takes 17, 1, 1, and 1 system clock cycles respectively, to process ($17 + 1 + 1 + 1 = 20$ system clock cycles). The 20 clock cycles correspond to a latency of 80 ns ($= 20 \text{ cycles} / 250 \text{ MHz}$) per cochlear section based on a system clock rate of 250 MHz. For a total of 108 cochlear sections, the latency is 8.64 μs ($= 80 \text{ ns} \times 108 \text{ cochlear sections}$). This latency is less than 10.42 μs ($= 1 / 96 \text{ kHz}$) – the time interval between the arrivals of two successive audio samples at a rate of 96 kHz.

Due to their low number, all 48 ($= 4 \text{ coefficients per cochlear section} \times 12 \text{ cochlear section per cochlear octave}$) 8 bits coefficients, namely a , c , g , and h , are stored on the FPGA chip, instead of the off-chip memory.

3.2.4.3. Hardware Resource Utilisation

The logic utilisation in adaptive logic modules (ALM) on an Altera Cyclone V FPGA for the implementation was only 2.57% with 747 registers used. With such low utilisation, more complex IHC and AN models as well as other sound processing algorithms, such as loudness, pitch, and timbre auditory-based models can be appended to the CAR-Lite cochlear model on a single FPGA. Furthermore, running models on a single FPGA reduces implementation complexities and data transmission constraints inherent in the usage of multiple FPGAs.

Table 3-1 shows a comparison of FPGA utilisation between CAR-Lite and other selected cochlear models on FPGA. CAR [3], [4] and CAR-FAC [25], [27] models have single audio sampling rates at 48 kHz and 32 kHz, respectively. The CAR-Lite model uses the least ALM and registers as opposed to the other cited cochlear models. One exception is Gambin's model [28], which utilises fewer registers on an FPGA than the CAR-Lite model. However, the CAR-Lite accommodates more cochlear sections, which can create a higher resolution auditory image.

Model	FPGA	No. of filters	ALM Utilisation	Registers Utilisation
Gambin et al. [28]	Xilinx XC3S500E	24	-	352
Leong et al. [29]	Xilinx XCV1000-6	88	-	6,800
CAR [3], [4]	Xilinx Virtex-6	102	-	9,480
CAR-FAC [25], [27]	Altera Cyclone V	100	18%	-
Thakur et al. [5]	Altera Cyclone V	70	3%	3,899
This work (CAR-Lite)	Altera Cyclone V	108	2.57%	747

Table 3-1: Performance of cochlear filters on FPGA.

3.2.5. Software Floating-point vs. Hardware Fixed-point

The hardware implementation of the CAR-Lite model was verified by comparing the fixed-point Altera Quartus SystemVerilog simulations with benchmark results generated by the floating-point Matlab implementation. Figure 3-7 displays the gain difference between the software floating-point and hardware fixed-point BM, BMd, and IHC signals generated using the log chirp signal described in section 3.2.7. Gain differences begin minimally at high frequencies (low cochlear section number) and rise towards low frequencies (high cochlear

section number) for all three responses. This growth is due to the cascaded configuration, which enables differences to accumulate at every cochlear filter section. The difference increases nonlinearly for the BM response from section 80 and beyond (250 Hz and below). However, this effect is curbed for the BMd and IHC signals, resulting in a steady linear rise in its gain error responses. This slow growth is primarily due to the BM lateral section difference defined by equation (3-7), which results in the BM velocity, BMd. In addition, this effect along with quantisation errors between the floating- and fixed-point numbers accentuate the gain difference of the BMd signal and even more of the IHC signal at fixed intervals, giving rise to regular spikes occurring at every octave.

Despite the gain differences, the correlation coefficients between floating-point and fixed-point responses of BM, BMd, IHC, and AN signals are 0.99, 0.99, 0.99, and 0.98, respectively. These scores show the highest degree of similarity attainable from a fixed-point model with the signal and coefficient bit widths set at 16 bits and 8 bits respectively. The few mismatches present are due to quantisation errors in the fixed-point model.

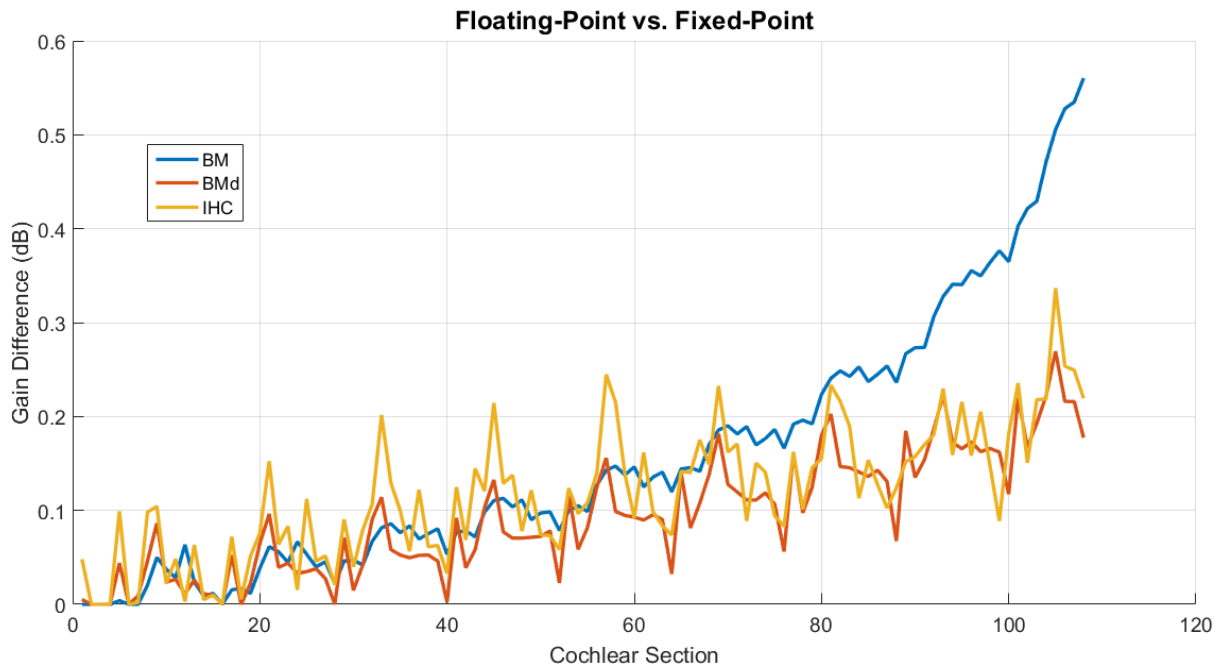


Figure 3-7: Gain difference between floating-point and fixed-point responses of the BM, BMd, and IHC signals.

3.2.6. CAR vs. CAR-Lite

This subsection shows that the use of multiple sampling rates results in lower bit widths of AR coefficients in CAR-Lite than with the conventional CAR model using a single sampling rate. Figure 3-8 displays the coefficients used in the CAR model [3], [4], which has a single sampling rate of 32 kHz. The low cochlear section number represents high frequency, and the high cochlear section number represents low frequency. The values of coefficients a and r converge to unity from low to high cochlear sections. Their values saturate within 5% of unity beyond section 55 (CF = 1,061 Hz). Using 8 bits to represent these numbers, both a and r are represented by the largest 8 bit number at 255, beyond section 87 (CF = 167 Hz). Similarly, coefficient c saturates to 5% full-scale above zero from section 96 (CF = 99 Hz) and beyond. Therefore, 8 bit numbers are insufficient to differentiate coefficients between neighbouring cochlear sections. These effects cause the gain response

of all the filters in the CAR model to have uneven peaks, especially in the low-frequency range, which covers cochlear section 60 (CF = 795 Hz) and above. Using 16 bits to represent the coefficients alleviates this problem and ensures a smooth peak gain response, as seen in [3], [4], [25]. Furthermore, a minimum bit width of 16 bits is necessary to represent the small changes in coefficients between successive cochlear sections, especially in the low and high-frequency regions in the CAR model.

Figure 3-9 displays the coefficients for each of the 12 sections (an octave) used in CAR-Lite. These coefficients do not have values close to either 0 or 1. The only exception is coefficients a and c , in section 1. However, these only occur for a single section instead of a group of sections as is the case in the CAR model. In section 2, although c has a value still within 5% of unity, the value of a has moved beyond 5% (full-scale) above 0. As a result, the coefficients are representable by a smaller bit width, i.e. 8 bits instead of 16 bits used for the CAR model coefficients. To process 108 cochlear sections on an FPGA regardless of the system clock rate and an audio sampling frequency of 96 kHz, the CAR-Lite model requires only 48 bytes (4-coefficients per cochlear section \times 12-cochlear sections \times 1-byte per coefficient) of coefficients storage as opposed to 864 bytes (4-coefficients per cochlear section \times 108-cochlear sections \times 2-bytes per coefficient) needed by the CAR model [3]. On an FPGA with a 250 MHz system clock rate and 96 kHz audio sampling rate, the CAR-Lite model can process up to 130 cochlear sections ($\approx (250 \text{ MHz} / 96 \text{ kHz}) / 20$ clock cycles per cochlear section), while the CAR model can only process up to 89 cochlear sections ($\approx (250 \text{ MHz} / 96 \text{ kHz}) / 29$ clock cycles per cochlear section).

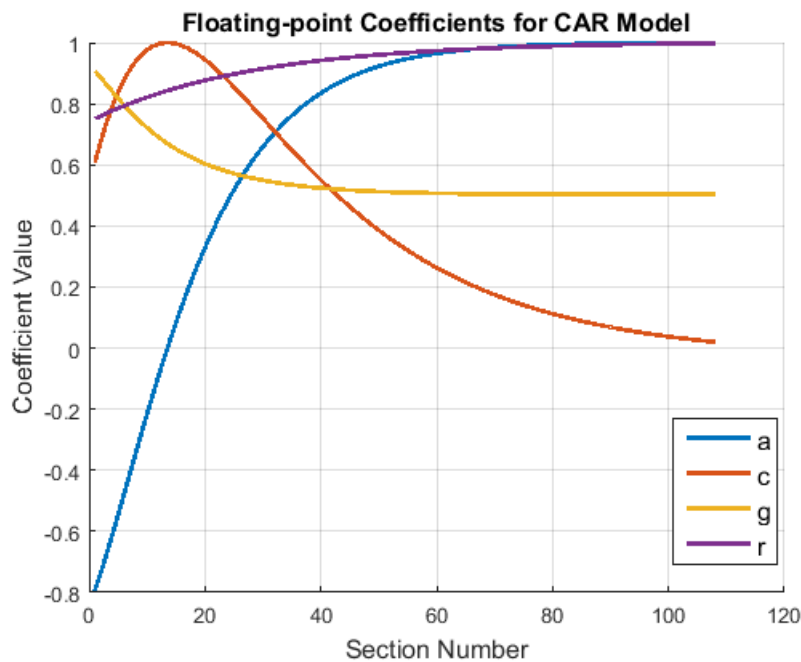


Figure 3-8: 16 bits AR coefficients used in the CAR [3], [4] and CAR-FAC [25], [27] models.

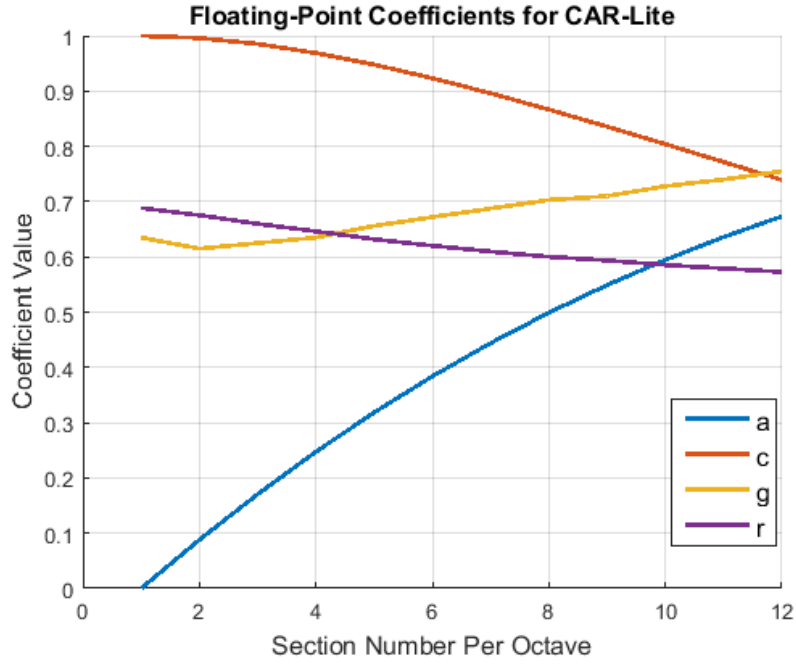


Figure 3-9: 8 bits AR coefficients used in each of the nine octaves in the CAR-Lite model [30].

3.2.7. Response to Log Chirp

A logarithmic sine tone sweep, also known as a log chirp, is used as the input signal to demonstrate the similarity of responses of the various stages of the CAR-Lite model between the software floating-point and hardware fixed-point implementations. The duration of the log chirp is 10.93s ranging from 10 Hz to 48 kHz. Figure 3-12 and Figure 3-13 illustrate the hardware fixed-point implementation of the BMd, IHC, and AN responses, respectively, while Figure 3-7 projects the difference between the software floating-point and hardware fixed-point implementations. Figure 3-10(a) and Figure 3-10(b) displays the gain and phase responses for the 108 CAR filter sections for the fixed-point BM response. The gain, $G(s)$, and phase, $P(s)$, are calculated as follow:

$$G(j\omega) = 20 \log_{10} \left| \frac{Y(j\omega)}{X(j\omega)} \right| \quad (3-13)$$

$$P(j\omega) = \tan^{-1} \left(\frac{\text{Im} \left| \frac{Y(j\omega)}{X(j\omega)} \right|}{\text{Re} \left| \frac{Y(j\omega)}{X(j\omega)} \right|} \right) \quad (3-14)$$

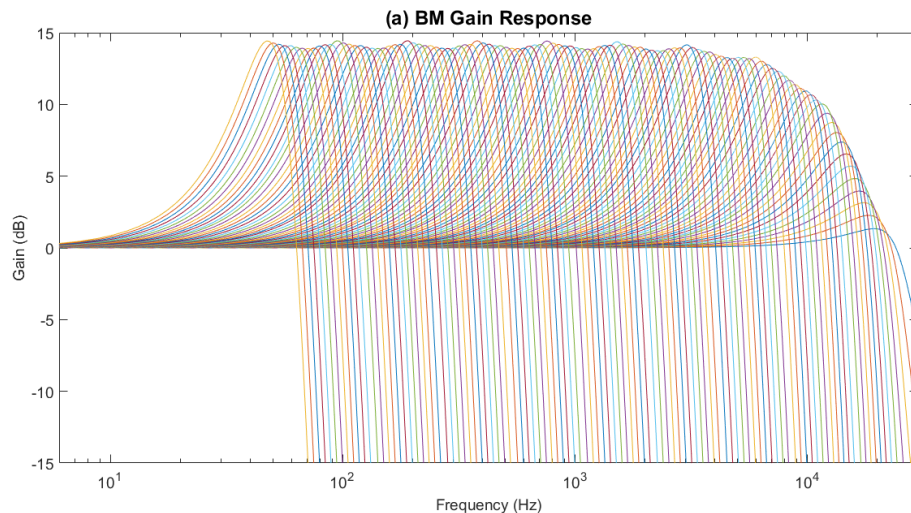
where $Y(j\omega)$ is the Fourier transform of the BM signal; $X(j\omega)$ is the Fourier transform of the input signal; Re is the real component and Im is the imaginary component of a complex signal.

Coefficient r in equations (3-1) and (3-4) is responsible for the approximately constant gain responses for all cochlear sections except for the ones in the first two octaves at high frequencies. Here, the cochlear sections have reduced gains that increase with decreasing centre frequencies from 19.43 kHz (section 1) to 5.996 kHz (section 24). One method of correcting these reduced gains is to amplify the gains in this region by using weighting networks [31] to ensure constant gain responses for all cochlear sections. This solution involves scaling the BM signals with a vector of gain-correcting factors, where each factor

corresponds to a cochlear section. However, due to the dependence of a BM signal from one cochlear section used as an input to the next cochlear section, the amplification of the gains at a low-section (high frequency) affects the gain at the next higher section (lower frequency). As a result, the gains across all the cochlear sections, which include DC gains across all channels, is scaled up. One solution to this issue is to amplify the BMd signal instead of the BM signal since a BMd signal generated from one cochlear section does not affect the BMd signal in the next higher cochlear section. An example of this is illustrated in Figure 3-11, where the gains of sections 1 to 24 are scaled up to the mean of the gains from sections 25 to 108, while the gains of sections 24 to 108 remain the same as the levels shown in Figure 3-10(c). The gain-correcting factors for sections 1 up to 24 are tuned to 4.63, 4.78, 4.42, 3.98, 3.55, 3.16, 2.84, 2.56, 2.33, 2.14, 1.98, 1.85, 1.72, 1.62, 1.55, 1.48, 1.42, 1.36, 1.32, 1.28, 1.25, 1.22, 1.19, and 1.17 respectively.

Another method of addressing the irregular gains from sections 1 to 24 is to ignore these reduced gains. This notion means that signals above 6 kHz do not have significant gain representation, such as those below 6 kHz. This solution is taken because the constant gains of the CAR-Lite model encapsulates the most sensitive region of human hearing between 2 kHz to 5 kHz as well as perceptible pitch information frequencies between 20 Hz to 5 kHz [32], [33]. The lack of sensitivity above 6 kHz, is captured by the decreasing gains of the CAR-Lite model and maintaining these signals with reduced gains decreases the emphasis of the frequencies above 6 kHz that may contain low-impact vocal information in speech and singing signals.

The effect of subjecting CAR filter output to lateral section difference leads to BMd gain responses in Figure 3-10(c). As described in subsection 3.2.2, this gain response has 6 dB/octave pre-CF low-frequency slopes, which is equivalent to BM velocity required to calculate IHC signals.



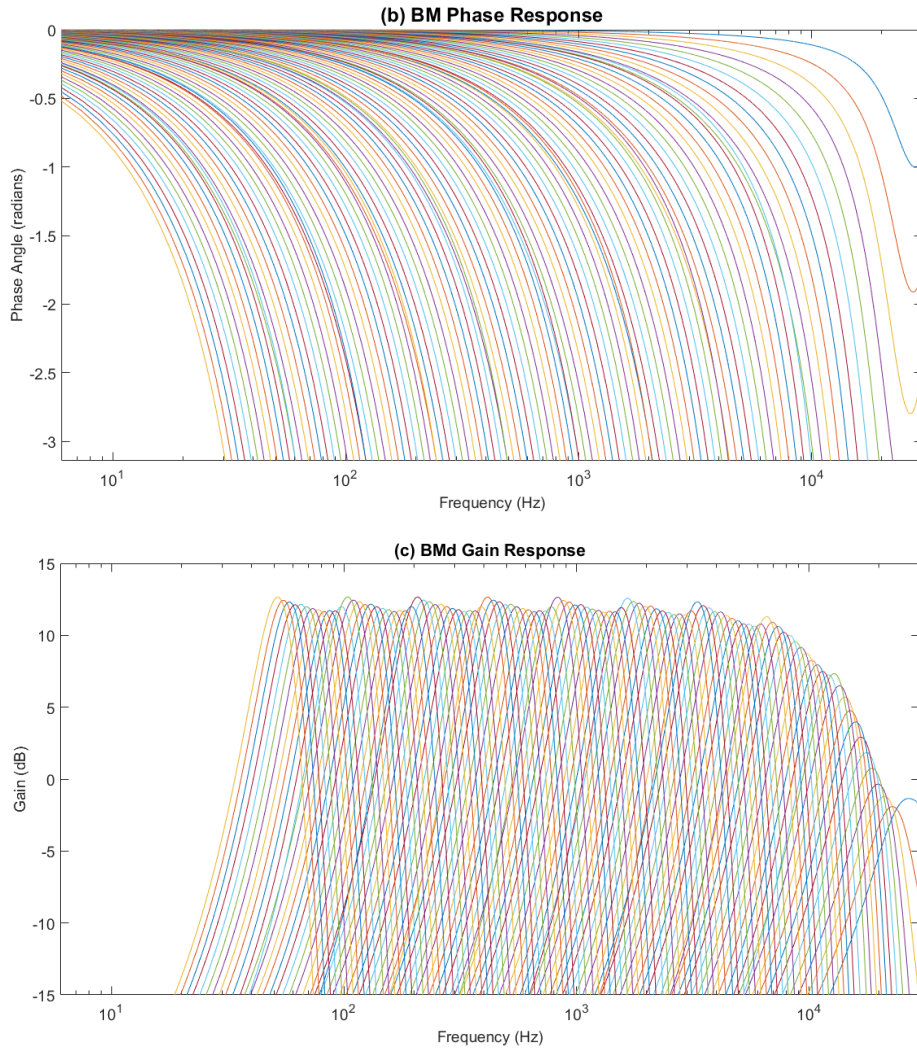


Figure 3-10: Fixed-point (a) gain response and (b) phase response of the BM signal (BM displacement) calculated from equation (3-1). Fixed-point (c) gain response of the BMd signal (BM velocity) calculated from equation (3-7). Each curve represents the response of a cochlear section, starting from section 108 (extreme left curve) to section 1 (extreme right curve). Note that the BMd response for sections 1 to 4 are below 0 dB due to the lateral cochlear section difference and the reduced gains of the ARs at the first two octaves (24 sections).

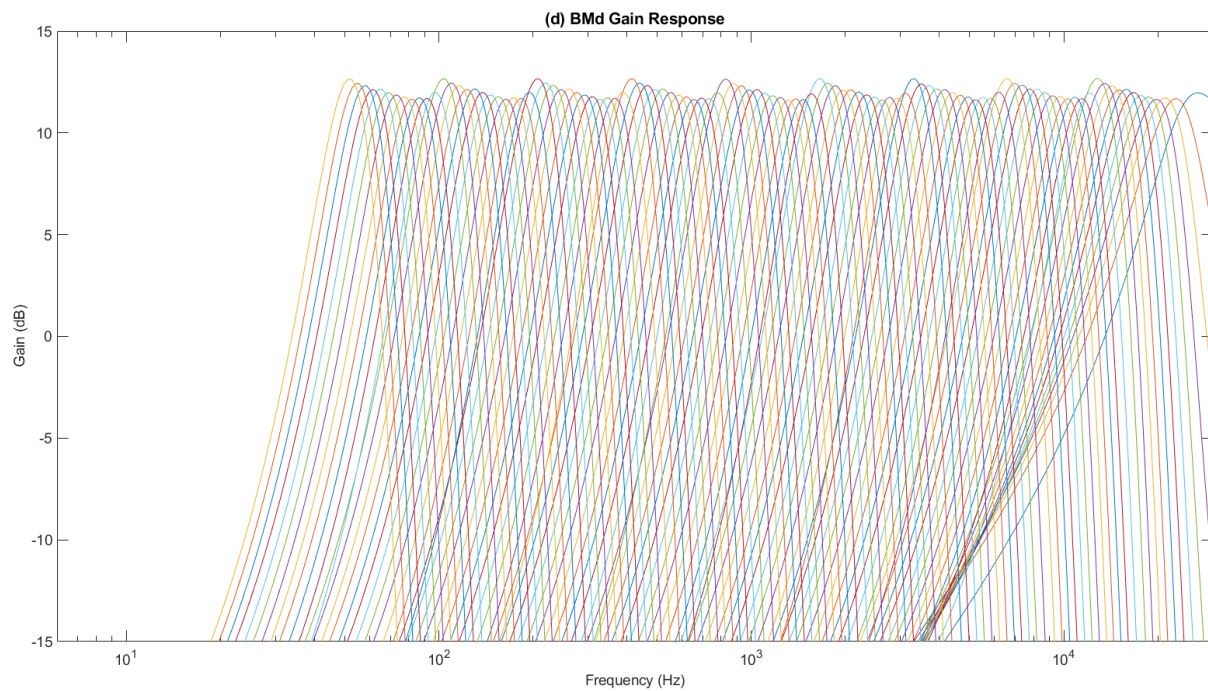
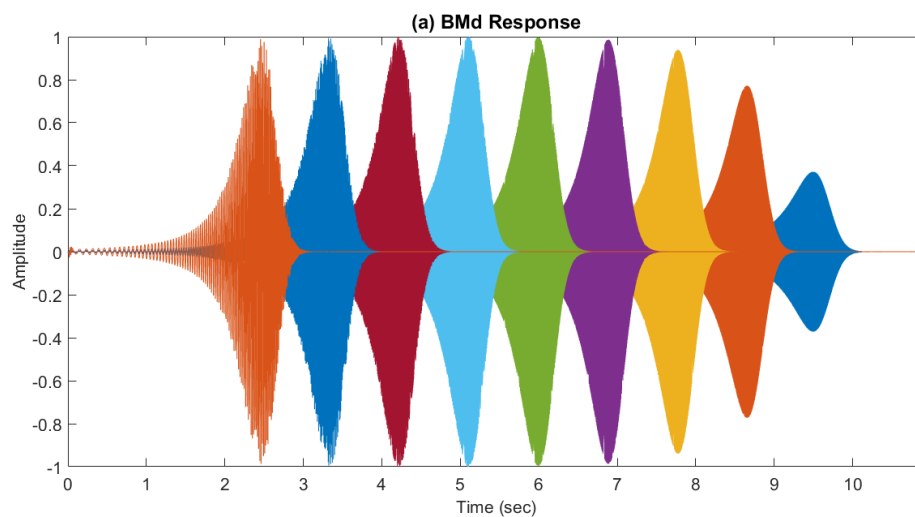


Figure 3-11: Fixed-point gain response of the BMD signal with the gain responses of sections 1 (extreme right) to 24 scaled up to the mean of the gain response levels from sections 25 to 108 (extreme left).



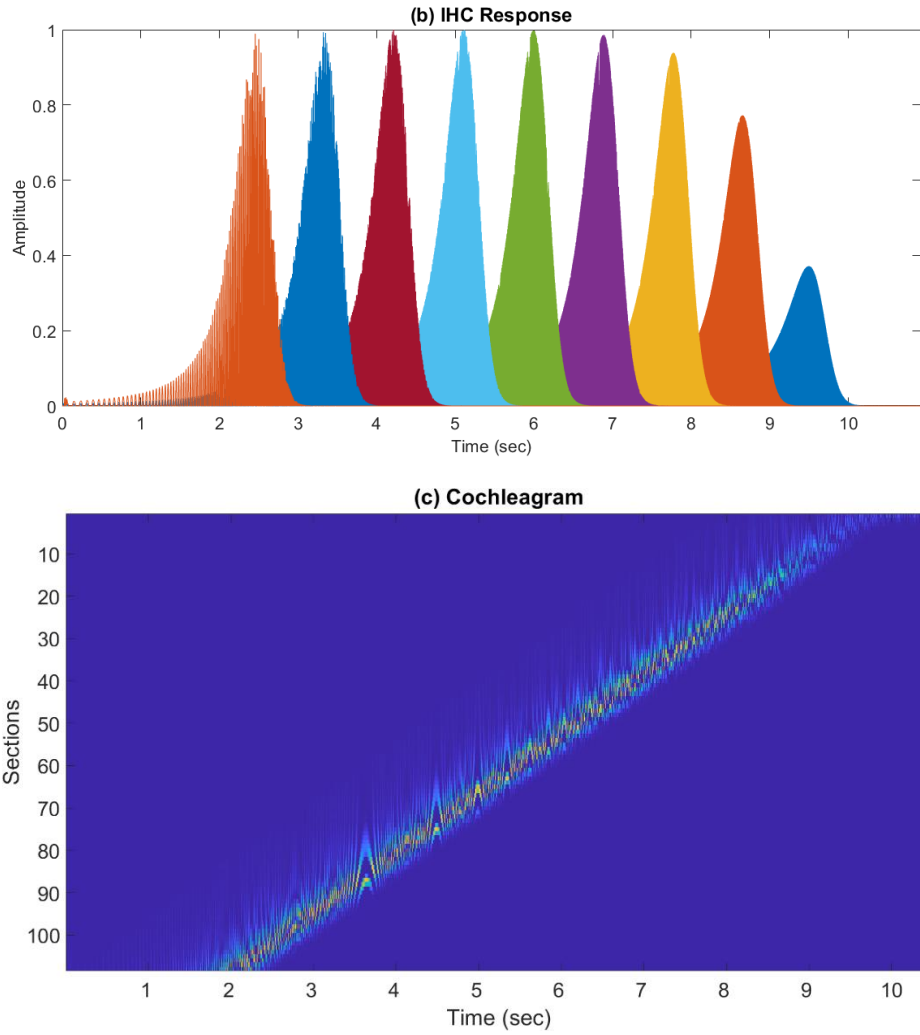


Figure 3-12: Normalised fixed-point responses of (a) the basilar membrane velocity, BMD; (b) the inner hair cell, IHC, based on log chirp input signal for cochlear sections 104 (extreme left signal), 92, 80, 68, 56, 44, 32, 20 and 8 (extreme right signal); (c) IHC for all 108 sections.

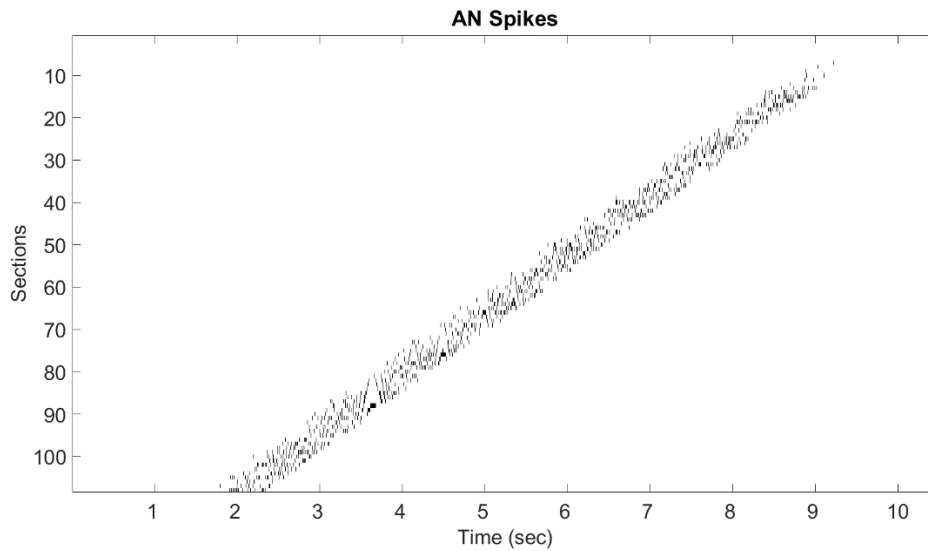


Figure 3-13: Fixed-point auditory nerve (AN) spike response based on the log chirp input signal.

3.2.8. Response to Music

Real-world sound signals vary unpredictably in intensity levels at various times, which affects loudness. One such signal is music, which is used to test the behaviour of the CAR-Lite model to generate responses at different intensity levels. Doing so provides information on the capability of the model to process real-world signals.

The 96 dB dynamic range of the CAR-Lite model is governed by the bit width of the input signal. Hence, input signals from 0 dB full-scale (dBFS) and below are accurately representable in CAR-Lite with 16 bit numbers. Beyond 0 dBFS, an input signal is amplified, which means that more than 16 bits are required to represent them accurately. However, due to the fixed bit width, the amplitudes of the input signal are saturated. Here, saturation refers to the inability of the peaks and troughs of an input signal to be represented at the correct levels. Instead, amplitude values larger than $2^{N-1} - 1$ are capped at $2^{N-1} - 1$ and amplitude values lower than -2^{N-1} are capped at -2^{N-1} , which in turn causes audible distortion.

To overcome the saturation of amplitudes, an automatic gain control (AGC) is used to vary the amplitude levels of an input signal. The AGC algorithm selected for use in this dissertation is detailed in appendix A. While the correction of the levels by the AGC removes the audible distortion, the timbre of the signal is altered, as discussed in section 7.5.2. The pitch components are also affected, but the fundamental frequency in the signal remains intact, as discussed in section 6.3.2. Despite altering the timbre and pitch components of a signal, the signal is intelligible.

The audio codec [40] on board the Altera V GX starter kit [41], acquires a sound signal via a microphone or another computing device connected to the starter kit. An AGC circuit in the audio codec conditions the input signal and suppresses saturation before transmitting the signal out of the codec and into the Cyclone V FPGA to be processed by the CAR-Lite model. To simulate such an operation, the AGC algorithm in appendix A conditions a sound signal before inputting it into the CAR-Lite model. Here, the input signal is a fixed-point representation of a musical signal of a piano note, A3. Figure 3-14 displays the musical signal at three intensity levels (-20 dBFS, 0 dBFS, and 20 dBFS) with the AGC algorithm disabled [Figure 3-14(a) – (c)] and enabled [Figure 3-14(d) – (f)].

Figure 3-15 displays the BM response of the input signals from Figure 3-14 calculated using fixed-point arithmetic in the CAR-Lite model. Here, the displayed BM signals are selected arbitrarily, i.e. the eighth cochlear section from each of the nine octaves: 8, 20, 32, 44, 56, 68, 80, 92, and 104. The increasing amplitudes of the musical signal from -20 dBFS to 20 dBFS are also reflected in the increasing amplitudes of the BM signals, especially when the AGC is disabled. When the AGC is enabled, the amplitudes across the three intensity levels are similar to one another, as observed in Figure 3-14(d-f) and Figure 3-15(d-f). This is because the AGC algorithm normalises the maximum amplitudes of the signals to +1 and -1 in floating-point representation, corresponding to 32,767 and -32,768 in 16 bits fixed-point representation.

The benefit of the AGC is evident for a 20 dBFS signal. When the AGC is disabled, the high amplitudes of the input signal and its BM representation are saturated at 32,767 and -32,768, as observed in Figure 3-14(c) and Figure 3-15(c), respectively. For the latter, this saturation is represented as red waveforms in Figure 3-16(a), ranging from cochlear sections

32 (CF = 4,005 Hz) to 80 (CF = 250 Hz). This saturation indicates that the CAR-Lite model is unable to properly represent the levels of an input signal at the BM stage and beyond. In contrast, the amplitude saturation is absent when the AGC is enabled, as shown in Figure 3-14(f), Figure 3-15(f), and Figure 3-16(b). Hence, the AGC enables the representation of amplitudes of the musical signal at various intensity levels, which then enables the CAR-Lite model to represent the signal at the BM stage and beyond accurately.

Let us assume that the bit width of the BM signal, b_{BM} , varies and is calculated using fixed-point arithmetic with maximum and minimum amplitude values of the BM signal from Figure 3-15, as follows:

$$b_{BM} = \lceil \log_2(\max(BM) - \min(BM)) \rceil \quad (3-15)$$

where $\lceil \dots \rceil$ is a ceiling operation. b_{BM} increases from 13 to 16 bits with increasing intensity from -20 dBFS to 20 dBFS when the AGC is disabled, as observed in the left half of Figure 3-17. In contrast, when the AGC is enabled, b_{BM} is constant at 16 bits for the same increasing intensity, as seen in the right half of Figure 3-17. This attribute indicates that the AGC enables the bit width of the BM signal to be maintained uniformly across intensity levels.

Overall, the AGC (described in appendix A) aids in improving the dynamic range (DR) of the CAR-Lite model, while maintaining a fixed bit width of an input signal across intensity levels. Consequently, the responses of the CAR-Lite model also have fixed bit widths and are unaffected by saturation issues across various intensity levels.

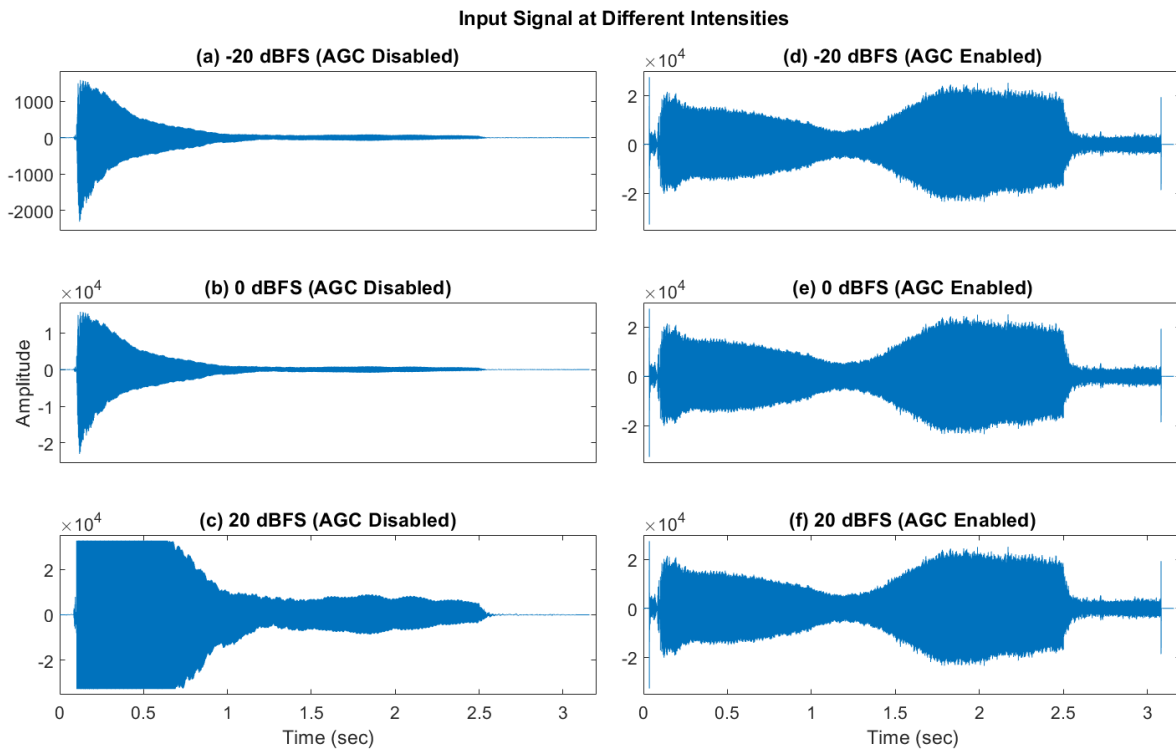


Figure 3-14: Musical signals (piano note A3) generated in fixed-point at various intensities from -20 dBFS to 20 dBFS. The left column depicts the signals directly input into the CAR-Lite model; the right column depicts the signals from the left column conditioned by an AGC algorithm before being input into the CAR-Lite model. Note that the 20 dBFS signal in (c) has its amplitude clipped at 32,767 $[= (2^{16-1}) - 1]$ and -32,768 $[= -2^{16-1}]$.

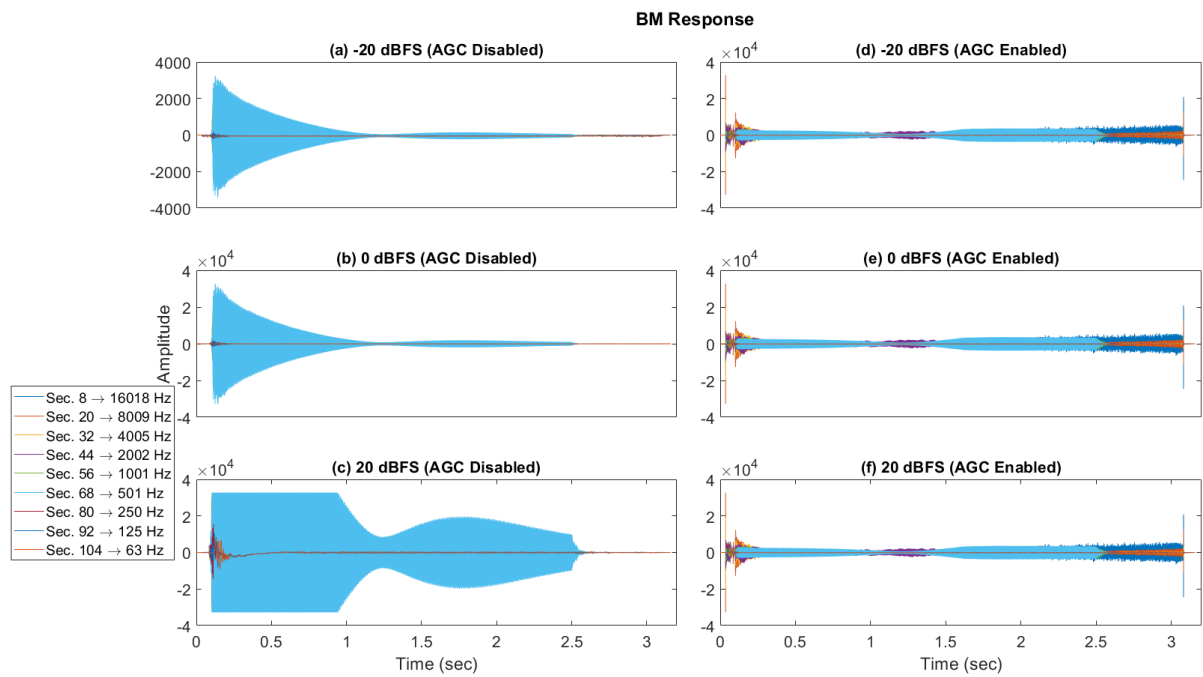


Figure 3-15: Fixed-point BM responses of the input signals from Figure 3-14. Responses in the left column are results of the input signals at three intensities with the AGC algorithm disabled, whereas the responses in the right column responses are input signals conditioned by the AGC algorithm before being processed by the CAR-Lite model.

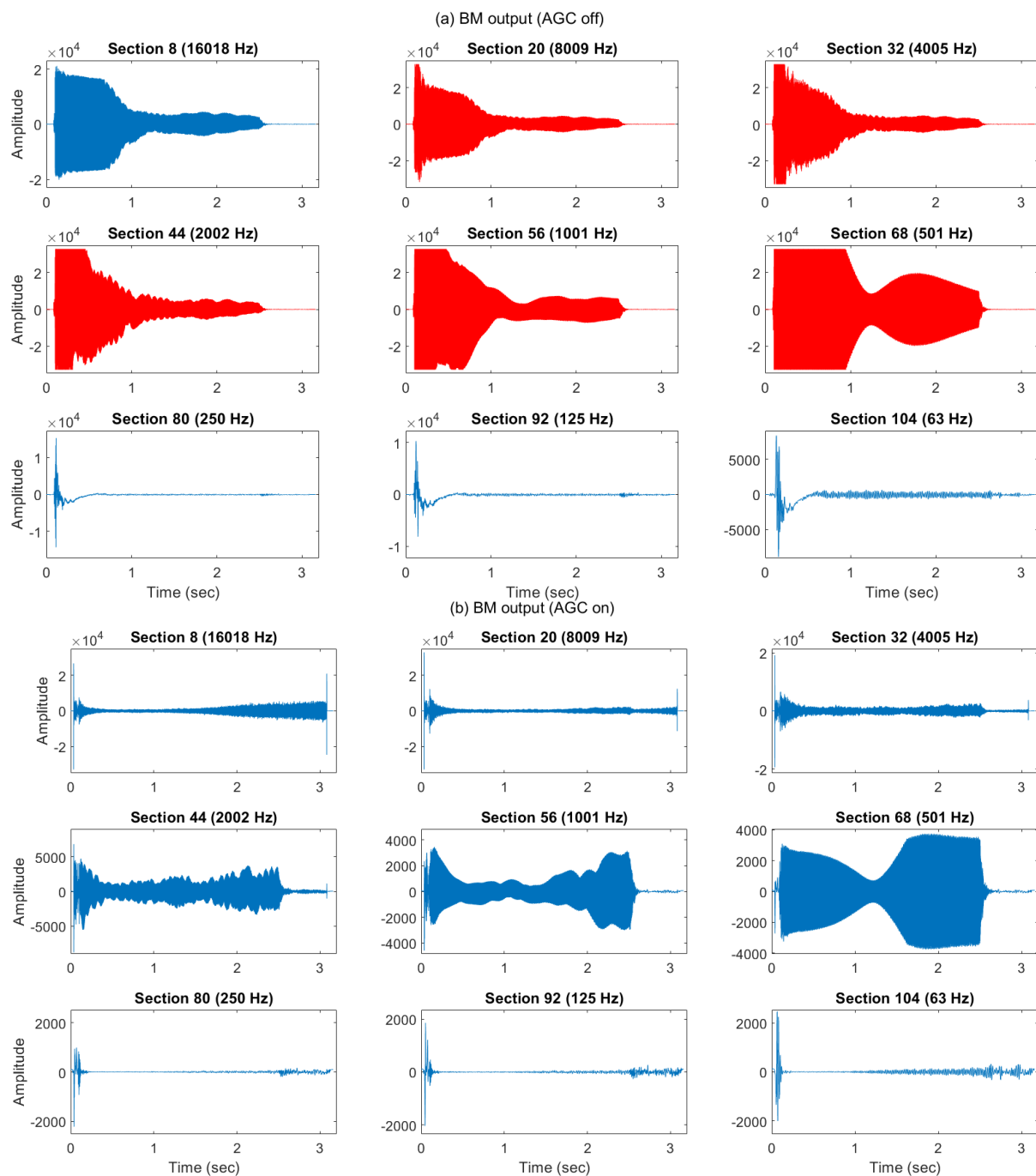


Figure 3-16: Fixed-point BM responses across cochlear sections for a 20 dBFS musical signal with (a) AGC disabled and (b) AGC enabled. Red waveforms have amplitudes clipped at 32,767 and -32,768 while blue waveforms have unclipped amplitudes.

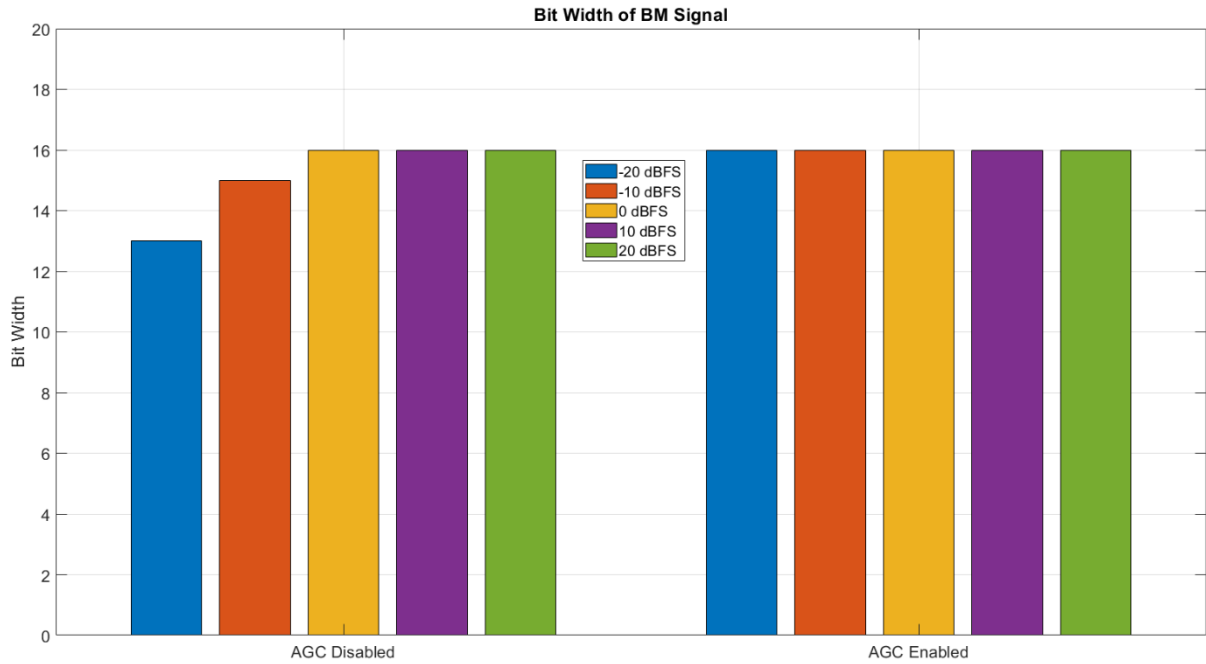


Figure 3-17: Bit widths of the BM signal generated from the input musical signal at multiple intensities from -20 dBFS to 20 dBFS with the AGC algorithm disabled and enabled.

3.3. Encoding Sound Intensity (SI)

The CAR-Lite model was designed to be a small-scale cochlear model for implementation on hardware. Although the CAR filters in the CAR-Lite model emulate basilar membrane (BM) characteristics, the auditory nerve (AN) segment does not adhere to biology. This attribute is due to the spike trains output from the AN segment of the model being generated from the zero-crossings (ZC) of the basilar membrane velocity (BMd) signals defined by equation (3-9), instead of inner hair cells (IHC) signals. As a result, this operation can be modelled with only a single bit comparator check on the most significant bit of a BMd output sample. However, the ZC algorithm is unable to capture sound intensity (SI), which is essential to describe perceptual loudness [42]. To represent sound intensity using spike trains, it is ideal to design the spiking algorithm based on how a mammalian cochlea characterises sound intensity at the AN stage.

In this section, the AN algorithm in the CAR-Lite model is modified to characterise sound intensity, leading to the formation of the CAR-Lite-SI model. The CAR-Lite-SI differs from the CAR-Lite model, particularly in the manner the auditory nerve (AN) segment encodes sound as an array of spike trains. The upcoming subsections present a new spiking algorithm that enables AN spike trains to capture sound intensity satisfactorily. The solution involves replacing the simple ZC spike generation algorithm with a sophisticated and biologically plausible algorithm, as is presented in subsections 3.3.1 and 3.3.2. Furthermore, the use of spike trains to represent sound intensity is essential to the design of simple algorithms, which can be implemented on hardware and operated at low power [43]. This algorithm projects intensity levels at multiple spiking rates through multiple neuron firing thresholds. Subsections 3.3.3 and 3.3.4 present an FPGA implementation of the CAR-Lite-SI model. The response of the model to input pure tones at various intensity levels is presented in subsections 3.3.5 and 3.3.6 as well as to real-world signals in subsection 3.3.7.

3.3.1. CAR-Lite-SI

The CAR-Lite-SI model is divided into two stages based on their respective sampling rates: the basilar membrane and inner hair cell (BM-IHC) stage and the spontaneous rate and auditory nerve (SR-AN) stage. The BM-IHC stage operates at multiple sampling rates as presented in the CAR-Lite model in section 3.2 and remains the same here, whereas the SR-AN stage housing the new AN algorithm, operates at a single sampling rate. The input to the second stage of the auditory nerve (AN) spike generation is the inner hair signal (IHC) signal, as opposed to the BMd signal in the CAR-Lite model. Since the IHC signal is positive, no sign bit is required to represent an IHC sample, resulting in less memory usage. Furthermore, the use of an IHC signal as a means to generate an AN signal agrees with biology.

At the second stage, action potentials, which are also known as spikes, are generated using leaky-integrate-and-fire (LIF) neurons. LIF neurons have been implemented on FPGA for a large parallel simulation of a neuromorphic cortex [44], which serves as an inspiration for the new AN spike generator algorithm. A LIF neuron contains a low-pass filter (LPF) stage to generate spikes. This LPF can be seen as a smoothing filter to remove high-frequency artefacts, which was absent in the CAR-Lite model. The spikes generated here are input to an optional complementary pseudorandom binary number generator (PBNG). The optional use of a PBNG is adopted from a biologically inspired rendition of AN spike generation for representing sound intensities [19], [45]. Figure 3-18 illustrates the signal pathway of the CAR-Lite-SI model. The next section describes the new SR-AN algorithm, and Figure 3-19 shows its signal pathway.

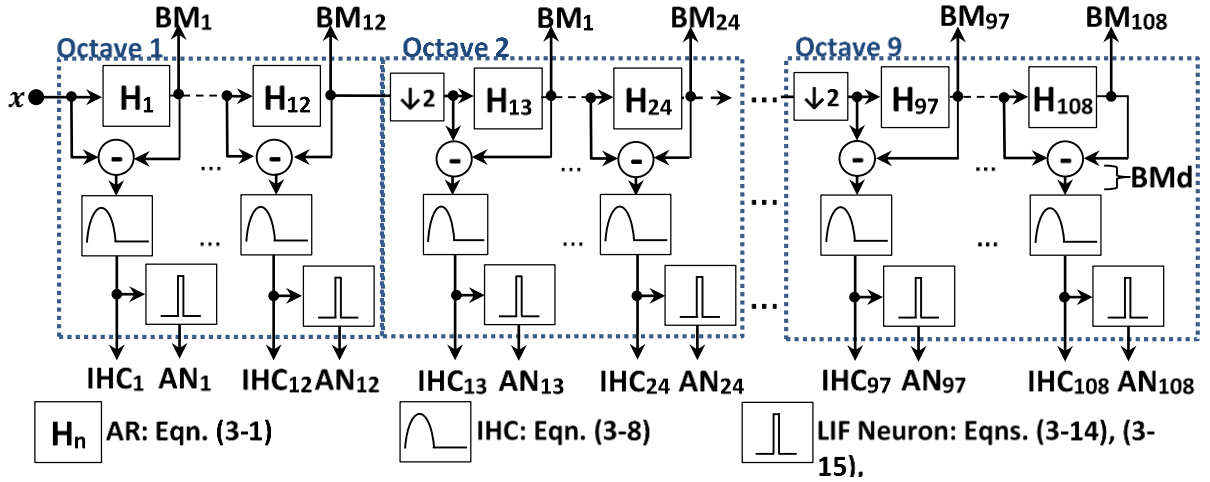


Figure 3-18: The CAR-Lite-SI with IHC signals as input to the new AN algorithm instead of the BMd signals used initially in section 3.2 in the CAR-Lite model.

3.3.2. The New Auditory Nerve Algorithm

In a mammalian cochlea, there are approximately 30,000 auditory nerve (AN) fibres [46]. About 10 to 30 fibres are connected to an IHC [47], and each fibre has a specific spiking threshold [48]. A fibre generates a spike due to the release of neurotransmitters from the IHC to the AN fibre via the synaptic cleft [49]. This release of neurotransmitters is based

on the build-up of presynaptic potassium in the IHC [45], which can be modelled as a moving average filter [50] by temporally integrating the IHC values [51]:

$$TI(s, t_{TI}) = \frac{1}{T} \sum_t^{t+T} IHC(s, t) \quad (3-16)$$

where t_{TI} is the subsampled time index for the temporally integrated (TI) IHC values; and T is an octave-specific accumulation cap, which is set to a multiple of 2. Thus, each octave is subsampled at a different factor, which brings the sampling rate across all octaves down to a common value. In addition, equation (3-16) functions as an anti-aliasing filter that can be implemented on FPGA with only addition and right-shift operations.

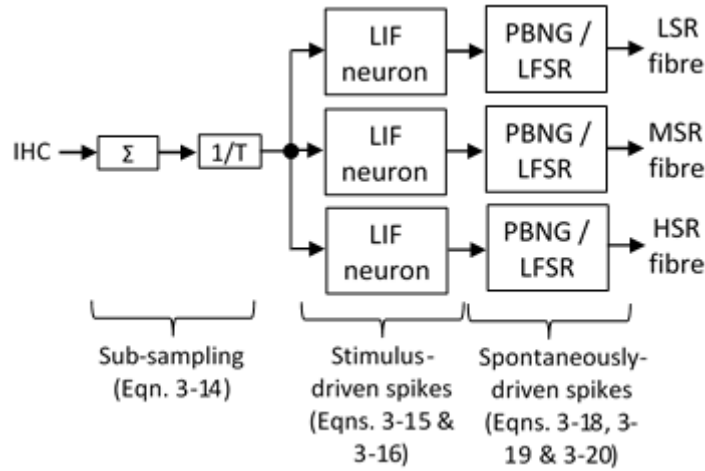


Figure 3-19: Spontaneous-rate (SR) – auditory nerve (AN) algorithm in place of the positive zero-crossing algorithm from section 3.2.

From TI , an AN spike can be generated in two conditions [19], [52]: (a) in the presence of a sound signal and (b) randomly, in the absence of a sound signal. In the former condition, a spike can be generated using a variation of the Fitzhugh-Nagumo neuron model [53], [54], which is also known as a leaky-integrate-and-fire (LIF) neuron model:

$$v(s, t_{TI}) = v(s, t_{TI} - 1) + c_{LIF}(TI(s, t_{TI}) - v(s, t_{TI} - 1)) \quad (3-17)$$

$$AN_{aSR}(s, t_{TI}) = \begin{cases} 1, & v(s, t_{TI}) > thresh_{aSR} \\ R_{aSR}, & v(s, t_{TI}) \leq thresh_{aSR} \\ 0, & AN(s, t_{TI} - t_{refrac}) \end{cases} \quad (3-18)$$

where v is the internal state voltage of a LIF neuron, which is reset to 0 after the corresponding AN associated with it fires a spike; $thresh_{aSR}$ is the spike threshold at which the LIF neuron fires a binary pulse of 1 for a specific fibre a (termed as H , M , or L - see next paragraph); R_{aSR} is the output of a PBNG with a uniform distribution between 0 and 1 representing spontaneity (optionally enabled); and c_{LIF} is the coefficient to determine how much of TI is accumulated to or leaked out from v and is computed using:

$$c_{LIF} = \frac{1}{f_{s-AN} \times \tau_{LIF}} \quad (3-19)$$

Here, f_{s-AN} is the sampling rate of the AN stage and τ_{LIF} is a time step constant. The minimum inter-spike interval durations of the spikes generated for all three conditions in equation (3-18) remain the same.

In the absence of a sound signal, the spiking activities of AN fibres do not cease. Instead, spikes are generated randomly at lower rates, due to the presence of presynaptic calcium [55], than when a sound signal is present. A possible outcome of incorporating random spike generation in an auditory model is the strengthening of a low-intensity sound signal represented in the AN fibres. Sparsely populated spikes in an AN fibre representing low information entropy can be enhanced by random spikes generated at low frequency. This notion results in a rise in phase-locked spikes that may improve the fidelity of a speech signal representation [56]. Randomly generated spikes may also improve timing precision of a stimulus, where the inter-spike-intervals (ISI) of a stimulus-driven spike train can be reinforced with a separate spike train generated from intrinsic neuron noise properties through coincidence matching [57]. Yet another contribution of random spikes is in general neural encoding in the auditory pathway, where combined stimulus-driven spikes and spontaneously-driven spikes containing attributes such as spike counts and spike train structure provides a statistical foundation [58] that may be beneficial to sound discrimination tasks involving source localisation, loudness, pitch, and timbre.

These AN fibres can be generally categorised into three primary groups based on their respective spontaneous rates (SR) of spike generation: high SR (HSR), medium SR (MSR), and low SR (LSR) [59], [60]. In HSR fibres, IHC presynaptic calcium builds up more than in LSR fibres. As a result, more spikes are generated in HSR fibres than in LSR fibres. This effect can be modelled in equation (3-18) by using a low threshold for HSR fibres and a high threshold for LSR fibres. Hence, for an ideal situation where there is no spontaneous spike generation for all three fibres, a sound with increasing intensity enables spike generation initially on the HSR fibre only, followed by HSR and MSR fibres, and finally across all three fibres.

In [55], the concentration of calcium is modelled as a cubic function, which is proportional to spike generation through the release of a finite number of circulating neurotransmitters between the IHC and AN fibres [49]. This feature is key to characterising varying spontaneous rates and the falling sensitivity of spike generation in AN fibres to a sound signal of constant intensity and frequency [61], called neural adaptation [62]. An alternative rendition is to bypass the modelling of calcium concentration and neurotransmitter release and generate spikes randomly using a Poisson process model [19], [63].

In the new AN spike generation algorithm in the CAR-Lite-SI model, linear methods are used for generating spikes randomly. The advantage of this implementation is that it is simple and implementable on hardware, which means that the spontaneous rates are fixed and capture deterministic dependencies that might otherwise be unpredictable if variable rates are used. Neural adaptation is also not implemented in this model as I aim to improve the AN algorithm in the CAR-Lite model to capture intensity as it appears in a sound signal across all cochlear sections. As a result, the number of spikes generated in this model is higher than those observed in animal studies as well as biologically plausible models strictly adhering to animal studies using sophisticated nonlinear equations [45]. In other words, the

spike rate of this model can operate at levels that might otherwise lead to a saturated spike rate in animal biology.

To exhibit the spontaneous rate effect across the three fibres, R_{aSR} from equation (3-18) is expanded as follow:

$$R_{HSR} = R_1 \wedge R_2 \quad (3-20)$$

$$R_{MSR} = R_3 \wedge R_4 \wedge R_5 \quad (3-21)$$

$$R_{LSR} = R_6 \wedge R_7 \wedge R_8 \wedge R_9 \quad (3-22)$$

where \wedge represents a 2-input logical AND operation; R_m is the content of bit- m of a 32 bit variable generated by a twisted generalised feedback shift register algorithm [64], which is a sophisticated implementation of the Berlekam-Massey algorithm to generate pseudorandom binary number sequence [65].

The number of bits used for the calculation of R_{aSR} in equations (3-20), (3-21), and (3-22) grow at one bit per equation in the order of HSR, MSR, and LSR. This growth indicates the increasing sparsity of spontaneous spikes across the three fibres, i.e. HSR fibres generate more spikes that are more densely populated than in MSR and LSR fibres. As a quantitative illustration of this effect, let us assume that the probability of R_m , $P(R_m)$ being either a *LOW* or *HIGH*, is 0.5. Then, the likelihood of R_{HSR} being a *HIGH* is 0.25, which is calculated as $P(R_1) \cdot P(R_2)$ from equation (3-20). The likelihood of R_{MSR} being a *HIGH* is 0.125 using $P(R_3) \cdot P(R_4) \cdot P(R_5)$, and for R_{LSR} being a *HIGH* is 0.0625 using $P(R_6) \cdot P(R_7) \cdot P(R_8) \cdot P(R_9)$. The decreasing probability factors correlate to sparser spike distributions across HSR, MSR, and LSR fibres, respectively.

Using this method, the inter-spike-interval generated by the spontaneous effect [second condition of equation (3-18)] is twice of that generated by the sound signal [first condition of equation (3-18)]. As a result, stimulus-driven spikes have twice the spike rate of the spontaneously-driven spikes. This attribute is advantageous as the stimulus-driven spikes can be clearly distinguished from spontaneously-driven spikes [66] in a sound classification and sound recognition tasks despite the contributions of spontaneous-driven spikes to stimulus-driven spikes in terms of information entropy improvement [56]. This specific form of algorithm is used solely for simulating spontaneous rate spikes in software. A hardware implementation of the PBNG is detailed in subsection 3.3.4.2, where an 8 bit linear feedback shift register (LFSR) is used to calculate R_{HSR} (3-20), R_{MSR} (3-21), and R_{LSR} (3-22).

3.3.3. Fixed-Point Implementation

The model was initially implemented using floating-point numbers to acquire a benchmark response in Matlab. Subsequently, the model was converted to fixed-point numbers in Matlab to simulate digital hardware (FPGA) responses. The fixed-point version is then implemented on FPGA in SystemVerilog. For the fixed-point implementation, each CAR coefficient is 8 bits and each LIF neuron coefficient and firing threshold, as well as every sample corresponding to the signals in the CAR-Lite-SI model, are set to 16 bits to match the floating-point responses. These bit widths are the minimum required to attain correlation coefficient scores, which quantifies the degree of similarity between floating-point and fixed-

point gains responses, as close as possible to positive unity (+1). Here, signal refers to input audio data, the BM output, the IHC output, and the LIF neuron internal state voltage. The AN signal is a single bit.

3.3.3.1. SR-AN Stage

Only the fixed-point implementation of the SR-AN stage is covered here as the BM-IHC stage is the same as the equation module described in subsection 3.2.4.2. The maximum rate of spike generation in the MAP model described in chapter 2 with default settings, is 1,000 spikes/s primarily in the HSR fibre at the onset of a stimulus. Based on this maximum rate, the sampling rate of the SR-AN stage, f_{s-AN} , is set at 3 kHz, which corresponds to the sampling rate of the sixth cochlear octave, o_6 . This sampling rate corresponds to a Nyquist frequency of 1.5 kHz and enables spike generation up to 1,500 spikes/s. Temporal integration via the moving average filter defined by equation (3-16) is applied only from octaves 1 to 5, which allows the down-sampling of IHC values from high multiple sampling rates to a low single rate of 3 kHz. In addition, the binary spikes generated from octaves 7 to 9 are up-sampled to 3 kHz. Using this approach, there is no need to manipulate the 16 bit LIF neuron voltages from equation (3-17) multiple times to generate the spike train. Instead, the single bit spikes themselves from equation (3-18) are dealt with, which reduces memory utilisation from 16 bits to 1 bit per SR-AN fibre from o_7 to o_9 . Table 3-2 shows the BM-IHC sampling rates across nine cochlear octaves and their respective sub-sampling factors in the SR-AN stage to achieve a sampling rate of 3 kHz.

Octave	f_s (kHz)	Down-sampling Factor	Up-sampling Factor
1	96	32	1
2	48	16	1
3	24	8	1
4	12	4	1
5	6	2	1
6	3	1	1
7	1.5	1	2
8	0.75	1	4
9	0.375	1	8

Table 3-2: Sub-sampling factors in the SR-AN stage operating at a sampling rate of 3 kHz.

AN fibres in a biological cochlea possess different spiking thresholds across sound intensities [55] and frequencies [67]. For the sake of simplicity, the spiking thresholds in this model are only set for the three discrete fibres regardless of the cochlear section. They are calculated as follow:

$$thresh_{LSR} = h_{LSR} \cdot th_{mn} \quad (3-23)$$

$$thresh_{MSR} = h_{MSR} \cdot thresh_{LSR} \quad (3-24)$$

$$thresh_{HSR} = h_{HSR} \cdot thresh_{LSR} \quad (3-25)$$

where h_{LSR} , h_{MSR} , and h_{HSR} are fibre-specific scalars; th_{mn} is the mean of the distribution of the maximum of a LIF neuron internal state voltage across all cochlear sections, s , for an input pure-tone signal, x , for 23 fixed frequency, f , at three arbitrary frequency ranges: 100

Hz to 1 kHz in 100 Hz steps; 1 kHz to 5 kHz in 500 Hz steps; and 5 kHz to 10 kHz in 1 kHz steps:

$$th_{mn} = \frac{1}{23} \sum_{f=100 \text{ Hz}}^{10 \text{ kHz}} \max_{f \in f(x)} v_f(s, t) \quad (3-26)$$

The internal voltage defined by equation (3-17) in this case is allowed to vary freely and is not reset to 0 V after the respective AN fibre fires. Using this approach, the LSR fibre threshold can be determined, along with the thresholds for MSR and HSR fibres using equations (3-24) and (3-25), respectively. The fibre specific scalars are then tuned to achieve a mean spike rate of 1,000 spikes/s at the HSR fibre. The mean spike rate is calculated using the total number of spikes per fibre over the time duration of an input signal. Table 3-3 shows the settings to achieve this.

Variables	Floating-point	Fixed-point
τ_{LIF}	1.33 ms	
c_{LIF}	0.25	16,384
h_{HSR}	0.01	
h_{MSR}	0.1	
h_{LSR}	2.329×10^{-2}	
th_{mn}	0.979	
$thresh_{HSR}$	2.28×10^{-4}	7
$thresh_{MSR}$	2.28×10^{-3}	75
$thresh_{LSR}$	2.28×10^{-2}	747
Absolute refractory period closest to 075 ms [19], [68]:	1.33 ms (1 sample skip)	

Table 3-3: Variables set to achieve a mean spike rate of 1,000 spikes/s at the AN HSR fibre.

3.3.4. FPGA Implementation

Figure 3-20 displays the architecture of the CAR-Lite-SI model implemented on an Altera Cyclone V FPGA (target FPGA chip: 5CGXFC5C6F27C7N). The FPGA system and audio codec clock rates are set at 250 MHz and 96 kHz, respectively. The model is characterised by three modules: a supervisor module, a BM-IHC module, and an SR-AN module. As observed from Figure 3-21, the operation of the supervisor and the BM-IHC modules in the CAR-Lite-SI model are identical to the supervisor and equation modules in the CAR-Lite model shown in Figure 3-5. The SPC and OPC states in the supervisor module of the CAR-Lite-SI model also control the invocation of the SR-AN module in addition to controlling the invocation of the BM-IHC module.

The role of the supervisor module is to pre-empt the next cochlear section to be processed and prepare its coefficients corresponding to the BM-IHC and SR-AN. It does so one cochlear section at a time, whereby it prepares coefficients for the BM-IHC module corresponding to section s and coefficients for the SR-AN module corresponding to section $s-1$. This operation is indicative of the SR-AN module cascaded after the BM-IHC module. As an illustration, in Figure 3-21, the supervisor prepares BM-IHC coefficients for section 2 and SR-AN coefficients for section 1 at T_3 and T_4 , while the BM-IHC module is processing section 1 and the SR-AN module is in *IDLE* state. The inactive SR-AN is awaiting samples to be output from the BM-IHC module via the supervisor module as well as coefficients from the supervisor module for section 1. It begins processing section 1 at T_5 . This scenario is repeated for all twelve cochlear sections corresponding to an octave. After the twelfth

section is processed, all three modules continue processing the next twelve sections serially in the next higher octave. When all the octaves are processed, all three modules transition to the *IDLE* state until the next audio sample arrives. Descriptions of the three modules are detailed in the next two subsections.

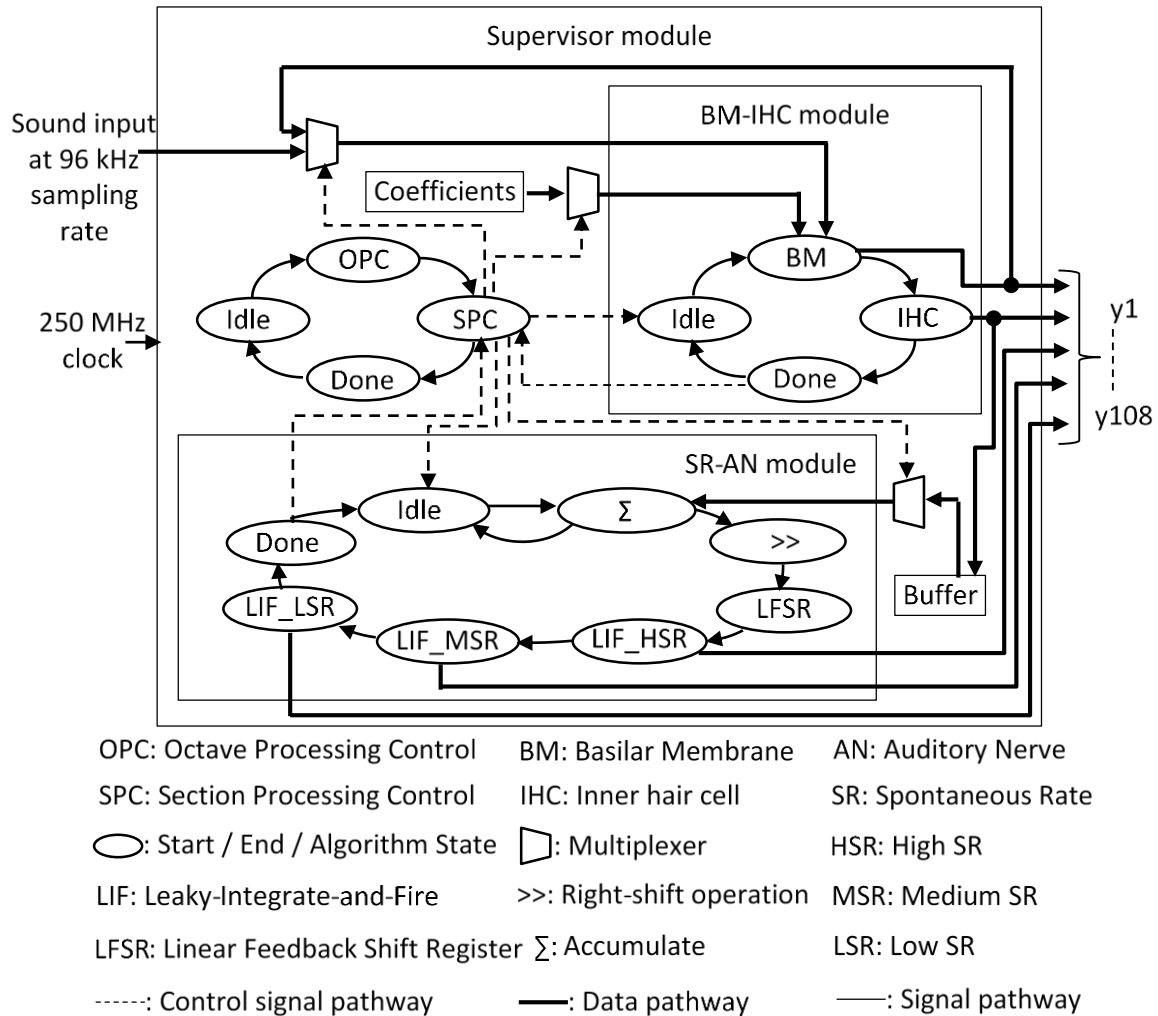


Figure 3-20: FPGA architecture of the CAR-Lite-SI model with the extended SR-AN module, where y represents a selected output signal from either the BM, BMd, IHC or AN stage.

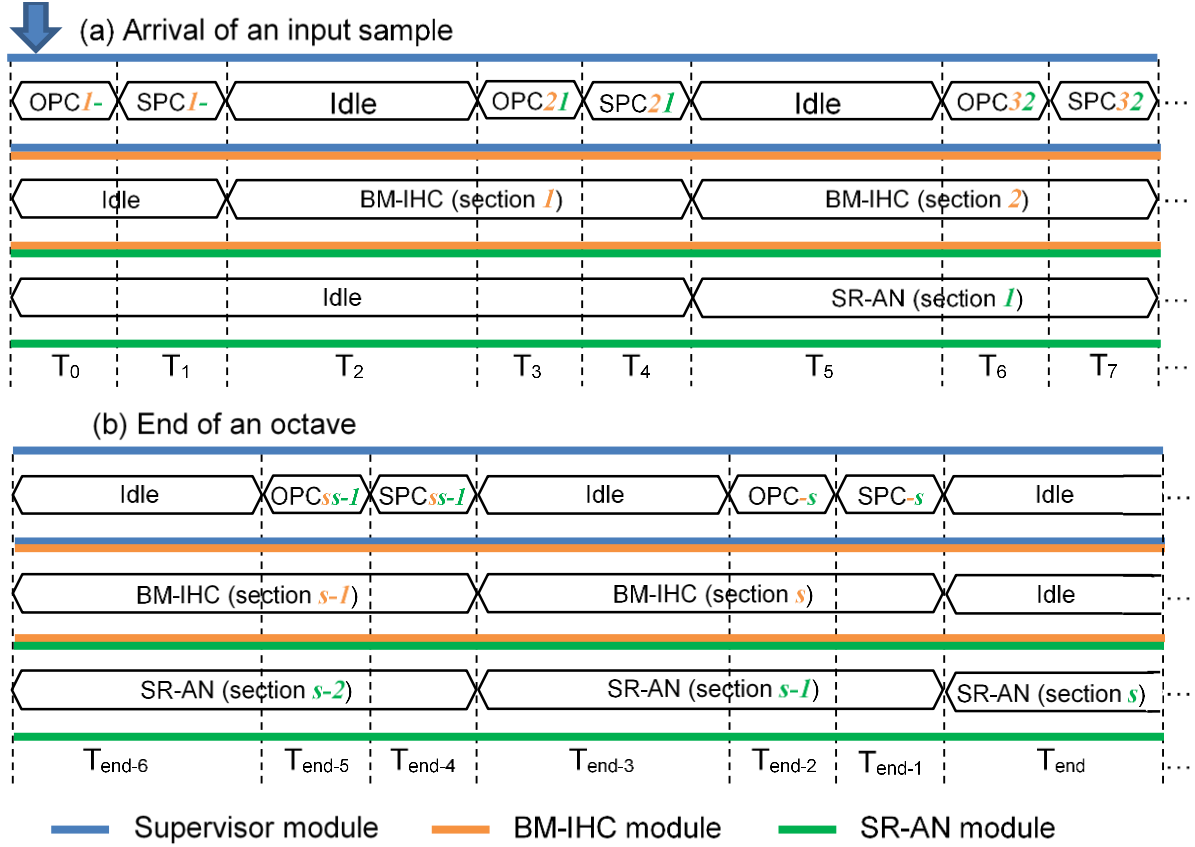


Figure 3-21: FPGA output vector waveform of the supervisor, BM-IHC, and SR-AN modules operating when (a) an audio sample arrives; and (b) at the end of the processing of a cochlear octave, where 12 cochlear sections have been processed at a specific sampling rate.

3.3.4.1. Supervisor and BM-IHC Modules

The implementation of the supervisor module is identical to the implementation described in subsection 3.2.4.1. The BM-IHC module is similar to the equation module described in subsection 3.2.4.2 with one exception. The auditory nerve (AN) spike generator using the positive zero-crossing of the BMD signal, defined by equation (3-9), is omitted. This omission brings the number of states in the finite state machine (FSM) of the BM-IHC module to 16 as opposed to 20 states in the equation module of the CAR-Lite model. The latency of the BM-IHC module is 64 ns (= 16 states / 250 MHz) per cochlear section. The BM, BMD, and IHC stage takes 14, 1, 1 clock cycles respectively, to process.

3.3.4.2. SR-AN Module

The structure of the SR-AN module is similar to the equation module in subsection 3.2.4.2 in that equations (3-16), (3-17), (3-18), (3-20), (3-21), and (3-22) are broken down into two operands per state, as part of an FSM. A total of 9 states are required to calculate the spiking activities of three AN fibres per cochlear section at 1 clock cycle per state, which amounts to a latency of 36 ns (= 9 states / 250 MHz). In the first state, IHC values are accumulated to generate T as many times as defined by an octave-specific cap. In the next state, the mean of the accumulated values is calculated by a right-shift operation as the octave-specific cap is a multiple of 2 as defined by equation (3-16).

In the next state, an 8 bit linear feedback shift register (LFSR) using Galois configuration [69], is updated to generate a pseudorandom sequence of binary numbers. The LFSR replaces the 32 bit software-based PBNG for generating spontaneous spiking activity on the FPGA. Its polynomial feedback is characterised as:

$$L(D) = D^8 + D^6 + D^5 + D^4 + 1 \quad (3-27)$$

where $L(D)$ is output of the LFSR defined by a polynomial equation; and D^m is the period of the exponent, m , indicating a tap-off from the m^{th} flip-flop to an XOR gate. Figure 3-22 illustrates the LFSR configuration.

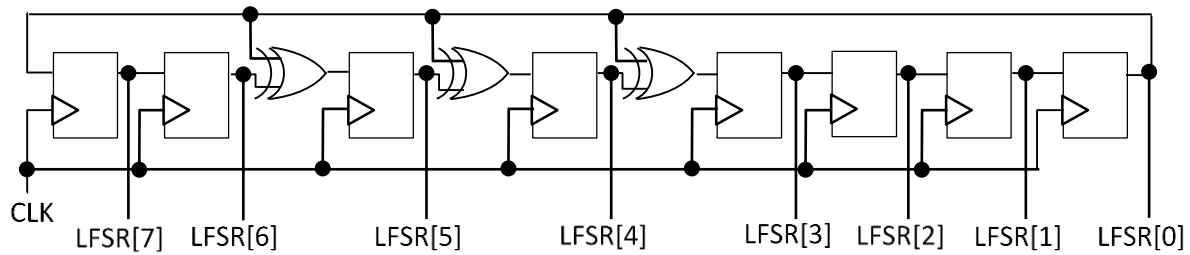


Figure 3-22: Linear feedback shift register (LFSR) implemented in fixed-point arithmetic and on an FPGA.

After the LFSR state, the three AN fibres types corresponding to a specific cochlear section, are calculated serially in the order of HSR, MSR, and LSR in three separate states (4 to 6), using equations (3-17) and (3-18). This serial computation is possible as the BM-IHC and the SR-AN modules are processed in a pipeline architecture and the latency of the SR-AN module is lower than BM-IHC. So, there is no need to calculate the spikes for the three AN fibres for each cochlear section in parallel. As a result, there is no risk of AN data corruption due to either buffer overrun or underrun. Within the same three serial states (4 to 6) of stimulus-driven spikes generation, the output of the LFSR calculated in the third state are then combined per equations (3-20), (3-21), and (3-22) to generate three spontaneous rate bits. In the next state, the three spontaneous rate bits are used for updating the three AN fibre output based on equation (3-18).

3.3.4.3. Hardware Resource Utilisation

The logic utilisation in adaptive logic modules (ALM) on an Altera Cyclone V FPGA for the fixed-point implementation is 15% with 8,420 registers used. Table 3-4 shows a comparison of the FPGA resource utilisation between the CAR-Lite and the CAR-Lite-SI models as well as with other analogue and digital models. The SR-AN algorithm described in subsection 3.3.2 is also implemented for the single-sampling rate CAR model [3] for the sake of comparison with the CAR-Lite-SI model. This modified CAR model is hereby known as the CAR-SI model. The CAR-Lite-SI model utilises lower ALMs and registers than the CAR-SI model by 2% and 2,161 registers, respectively. The CAR-SI also has the highest utilisation over all other models in Table 3-4. The exception is the single-sampling rate CAR-FAC model with OHC and automatic gain control, which has the highest overall ALM utilisation [27]. The ALM utilisation for all other digital models is significantly lower. Therefore, it can be concluded that the SR-AN algorithm increases FPGA resource utilisation significantly.

The CAR-Lite-SI model utilises logic significantly higher than the CAR-Lite model by five times in ALM utilisation and twelve times in the utilisation of registers. This attribute is due to the CAR-Lite-SI model having an extra module (SR-AN) than the CAR-Lite model. In addition, the increase of the AN output sections from 108 to 324 (3 AN fibres \times 108 cochlear sections) channels as well as a more sophisticated manner of generating spikes have led to the increased number of registers used for buffering as well as heightened inter-connectivity between the modules in the CAR-Lite-SI model. However, there is still ample space available on the FPGA for expanding CAR-Lite-SI further, which enables the implementation of more complex algorithm such as neural adaptation [70] in the SR-AN segment and nonlinearity in the BM response through outer hair cell (OHC) feedback and automatic gain control such as the one used in the CAR-FAC model [27]. Incidentally, the increase in logic utilisation due to the SR-AN algorithm is justifiable as the CAR-Lite-SI model can represent sound intensity levels in the spike-time domain as presented in subsection 3.3.7, which cannot be performed by the other models in Table 3-4 except for the CAR-SI model.

A power analysis tool from Altera, called PowerPlay with default settings enabled, is used to estimate the power required to run the CAR-Lite-SI model because the power consumption for the Altera Cyclone V GX Starter Kit cannot be measured directly. As shown in Table 3-4, the estimated power consumed by the CAR-Lite-SI model is higher than the CAR-Lite model. This characteristic is to be expected due to the higher logic utilisation of the former than the latter. However, the multi-sampling rate CAR-Lite-SI model still consumes less power than the single sampling rate CAR-SI model, which consumes the highest power as compared to all other presented models in Table 3-4. Another significant observation is that the FPGA-based cochlear models consume more power than analogue silicon cochlear models. However, analogue silicon cochleae may become unstable due to the various degrees of tolerances of analogue electronic components in the circuit.

Model	FPGA	No. of filters	ALM utilisation	Registers utilisation	Power (mW)
Sarpeshkar et al. [71]	-	16	-	-	0.3
Lazzaro et al. [72]	-	119	-	-	5
Hamilton et al. [73]	-	83	-	-	57
Shih-Chii et al. [74]	-	128	-	-	22
Wang et al. [75]	-	9	-	-	0.09
CAR-FAC [27]	Altera Cyclone V	100	18%	5,235	240
Thakur et al. [5]	Altera Cyclone V	70	3%	3,899	-
CAR-Lite [76]	Altera Cyclone V	108	2.57%	887	234
*CAR-SI [3]	Altera Cyclone V	108	17%	10,581	250
This work (CAR-Lite-SI)	Altera Cyclone V	108	15%	8,420	244

Table 3-4: Performance of cochlear filters on analogue (described as '-' under FPGA column) and digital hardware. *Single sampling rate cochlear model fitted with SR-AN algorithm from subsection 3.3.4.2.

Furthermore, as the analogue models are hardwired, they are not scalable. In contrast, FPGA-based cochlear models are stable as they operate in the digital domain and do not have tolerance issues with their internal components. Additionally, the reconfigurable attribute of the FPGA allows cochlear models to be scalable. This characteristic includes

adding or omitting cochlear octaves and sections, which is achievable at a shorter time than for the analogue silicon cochleae.

3.3.5. Response to Pure Tones

A 1 kHz, 0.68 s pure tone signal with a 50 ms ramp-up and 50 ms ramp-down is used to generate six types of AN fibre responses from the fixed-point implementation of the CAR-Lite-SI model, as illustrated in Figure 3-24. The pure tone is chosen to demonstrate and distinguish between spikes generated directly by the sound signal (stimulus-driven spikes) and spikes generated by the spontaneous effect (spontaneously-driven spikes). The HSR responses in Figure 3-24 depict the input signal spread across more contiguous cochlear sections [58 cochlear sections from sections 11 (13,470 Hz) to 69 (472 Hz)] than the MSR response [43 cochlear sections from sections 24 (6,357 Hz) to 67 (530 Hz)]. The LSR response of the 1 kHz pure tone is contained within the smallest number of contiguous cochlear sections at 27, spanning from sections 38 (2,832 Hz) to 65 (595 Hz). In terms of spontaneous rate activities, the HSR fibres generate the most densely packed random spikes, and the LSR fibres generate the least, which are in agreement with biologically plausible models [19], [20]. The intensity information from the combined stimulus-driven and spontaneous-driven spikes are acquirable by the mean spike rate across the three fibres presented in Table 3-5.

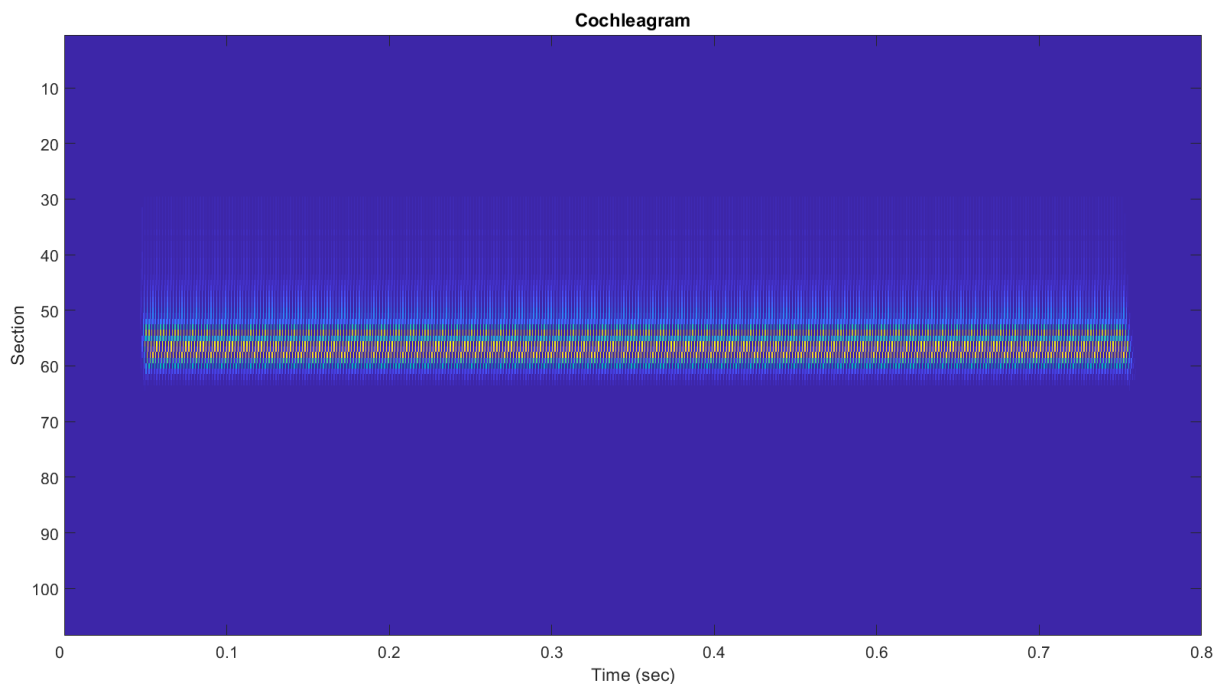


Figure 3-23: Time-frequency representation (cochleagram) of the IHC signal of a pure tone whose AN response is shown in Figure 3-24.

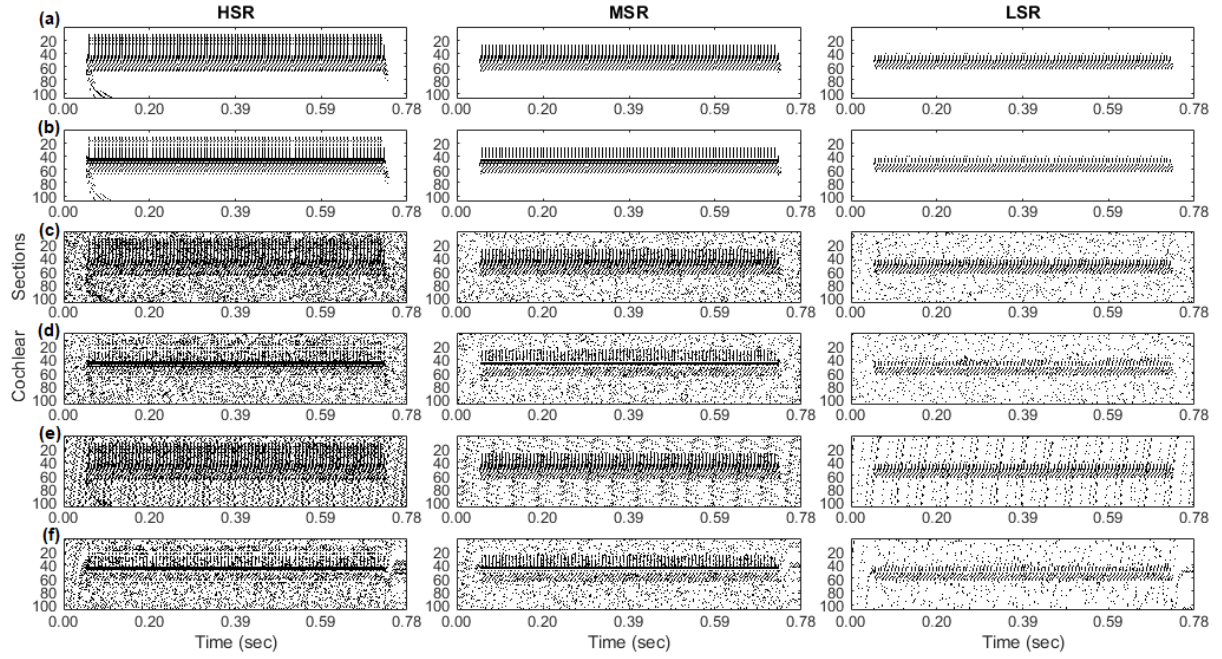


Figure 3-24: SR-AN spike response from a fixed-point implementation of the CAR-Lite-SI model of a 0.68 s, 1 kHz input pure tone with a 50 ms ramp-up and 50 ms ramp-down (IHC representation displayed in Figure 3-23) for the following conditions: (a) both spontaneity and refractoriness disabled; (b) spontaneity disabled and refractoriness enabled; (c) spontaneity enabled using PBNG and refractoriness disabled; (d) spontaneity enabled using PBNG and refractoriness enabled; (e) spontaneity enabled using LFSR and refractoriness disabled; and (f) spontaneity enabled using LFSR and refractoriness enabled.

When the AN refractory period is enabled (labelled as refractoriness in Table 3-5), the AN fibres produce sparser spike distributions, as observed for (b), (d), and (f) of Figure 3-24, and quantified as lower spike rates in Table 3-5, than when the AN refractoriness is disabled. This characteristic is mainly due to the bypassing of the computation of every second input sample to the SR-AN module to signify the inability of the AN to generate spikes immediately after firing. The absence of spontaneously-driven spikes (AN spontaneity disabled) coupled with the LIF neurons' inability to immediately fire (refractoriness enabled) leads to the lowest spike rate across all three fibre types. When both spontaneity and refractoriness is enabled, as seen in (d) and (f) of Table 3-5, the spike rate is close to the design requirements of a mean of 1,000 spikes/s on the HSR fibre.

Figure 3-24	Spontaneity		Refractoriness	Mean Spike Rate (spikes/s)		
	LFSR (FPGA)	PBNG (Software)		HSR	MSR	LSR
(a)	0	0	0	607	419	230
(b)	0	0	1	531	371	206
(c)	0	1	0	1,212	741	400
(d)	0	1	1	917	613	340
(e)	1	0	0	1,212	745	407
(f)	1	0	1	916	623	359

Table 3-5: Mean spike rate of each AN fibre with various spontaneous spiking conditions generated from the fixed-point implementation of the CAR-Lite-SI model.

3.3.6. Iso-Intensity Response

Figure 3-25 displays the AN spike rate response from the fixed-point implementation of the CAR-Lite-SI model to a frequency sweep of a 0.68 s pure tone input signal with its frequency ranging from 100 Hz to 1 kHz. For iso-intensity frequency responses [77] of the

AN, the input pure tones at the aforementioned varying frequencies are repeatedly streamed into the model at different input levels from 0 dB full-scale (FS) to -60 dBFS, in steps of -10 dBFS. At high-intensity levels of 0 dBFS, spike generation at all three fibres are mainly driven by the sound input frequency range. As sound level reduces, the spike rates of all the three fibre types drop expectedly regardless of the sound input frequency. Below -40 dBFS, the LSR fibres are no longer sensitive to the stimulus as illustrated in the negligible spike rates in Figure 3-25(c) when no spontaneously-driven spikes are present. When the spontaneously-driven spike generation algorithm is enabled, the spikes rates at these levels rise, as depicted in Figure 3-25(f) and Figure 3-25(i). At -60 dBFS, only the HSR fibres are driven by the sound input as observed in the active spike rates in Figure 3-25(a), and no spike rate activities in the MSR [Figure 3-25(b)] and LSR [Figure 3-25(c)] fibres when no spontaneously-driven spikes are present. When spontaneously-driven spikes are present, the MSR and LSR fibres are exclusively driven by spontaneous activity, as observed in Figure 3-25(e), (f), (h), and (i).

The spike rates exhibited in Table 3-5 and the iso-intensity curves in Figure 3-25 are higher than the iso-intensity curves recorded in animal studies that include squirrel monkey [78], guinea pig [61], and lake sturgeon [77]. This attribute is due to the absence of neural adaptation feature in the model as my intention is to characterise sound signal intensity variations as it occurs without any reduction to the sensitivity across all cochlear sections. As a result, the spike rates projected by the CAR-Lite-SI model remain constant as long as the sound signal is present. Furthermore, the iso-intensity curves from the animal studies are generally bell-shaped. In contrast, the iso-intensity curves in this model have a general outlook of an unsmoothed right-half of a bell (a steeper roll-off at the high frequency end than the low frequency end), which indicate that this model is generally more responsive to low-frequency sound signals than the responses found in animal studies [61], [77], [78]. This characteristic is an advantage as this enables the model to respond to low pitch speech and musical signals.

An alternative viewpoint is that the shape of these curves in the model resembles the gain response shape of a low-pass filter (LPF). An indication of this is that firstly, the spike rate above 1 kHz drops lower than those below the 1 kHz range. Secondly, the dips in the spike rate at high intensities from 0 dBFS to -30 dBFS for HSR fibres and from 0 dBFS to -10 dBFS for MSR fibres occur at 3 kHz, 6 kHz, 9 kHz, and 12 kHz. These dips in the iso-intensity curves are similar to the stopband ripple or known alternatively as the suppressed fluctuating high-frequency side-lobe of a moving average filter [50]. The cause of this is the temporal integration stage, which performs a moving average of IHC values and thus, functions as a low-pass filter with a cut-off frequency at 1.5 kHz. With regards to musical signals, the starting dip in the iso-intensity curve at 3 kHz corresponds to the eighth octave of the musical note, F#. This note and those at higher frequencies corresponding to the dips in the iso-intensity curves are considered to have high pitches that are outside the dominant region of pitch perception [79]–[81], so no measures are taken to alleviate these effects. For speech signals, the spectrum of interest is approximately from 200 Hz to 3 kHz [56], and so the dips in the iso-intensity curves have ideally minimal impact on speech signals as well.

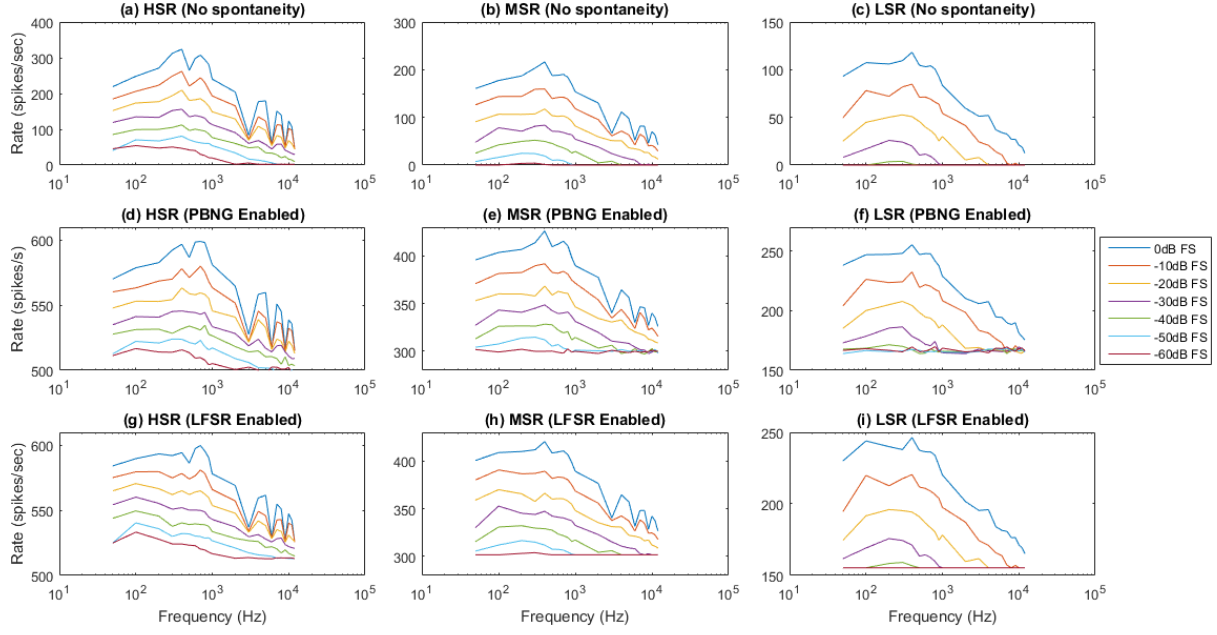


Figure 3-25: Iso-intensity responses of HSR, MSR, and LSR AN fibres for 108 cochlear sections generated with refractoriness enabled and (a) – (c) spontaneity disabled, (d) – (f) PBNG enabled and (g) – (i) LFSR enabled.

3.3.7. Noise Effect on Real-World Signals

Figure 3-26(a1)-(a6) displays a musical note signal, A4, from a piano [15], along with its AN responses. Figure 3-26(b1)-(b6) displays a speech signal with a male voice uttering “1-o-7” [82] and its corresponding AN responses. These two signals are up-sampled from 44.1 kHz to 96 kHz before being injected into the fixed-point hardware implementation of the CAR-Lite-SI model. At the AN stage, noisy spikes are introduced to these real-world signals either with the PBNG or the LFSR circuit when the spontaneity option is enabled (as seen in Table 3-6 and Table 3-7). Two methods are used to measure the effect of the noise on the signals. One involves calculating the signal-to-noise (SNR) ratio:

$$SNR = 20 \log_{10} \frac{spk_{AN}(s, n)}{spk_{SRAN}(s, n)} \quad (3-28)$$

where s is the cochlear section spanning to a maximum of S ($= 108$) sections; n is the sample number spanning up to the length of the sound signal N ; spk_{AN} is the AN response comprising stimulus-driven spikes and spontaneously-driven spikes (noisy spikes) generated either from the PBNG or LFSR circuit; spk_{SRAN} is the spontaneously-driven spikes generated from either the PBNG or LFSR circuit.

Alternatively, to understand how much information from the input sound signal is preserved in the AN response amid noise generated from either the PBNG or LFSR circuit, a spike coincidence ratio is used. It is calculated as the ratio of the number of spikes match between the AN response and the binarized TI response (subsamped low-pass filtered inner hair cell response), which is then normalized by the number of binary TI spikes:

$$C_{spk} = \frac{\sum_{s=1}^S \sum_{n=1}^N spk_{AN}(s, n) spk_{TI}(s, n)}{\sum_{s=1}^S \sum_{n=1}^N spk_{TI}(s, n)} \quad (3-29)$$

where spk_{TI} is the binarized TI response, which is assumed to be a 2D spike representation of the input signal unaffected by spontaneously-driven spikes (noisy spikes). It is defined as:

$$spk_{TI} = \begin{cases} 1, & TI(s,n) > 0 \\ 0, & TI(s,n) = 0 \end{cases} \quad (3-30)$$

The SNR profiles for the speech and musical signals using noises generated from the PBNG and the LFSR circuits across the three AN fibres are displayed in Figure 3-27. The real-world signals have three discrete intensity levels, represented as three distinct colour bars in the figure. The SNR is at its lowest for low-intensity signals and highest for high-intensity signals regardless of the input signal type used here. This attribute is expected as low-intensity signals result in a low number of stimulus-driven spikes generated at the AN stage and thus, spontaneously-driven (noisy) spikes affect the AN response significantly, which results in a low SNR. Conversely, high-intensity signals result in a higher number of stimulus-driven spikes generation, which lowers the influence of spontaneously-driven (noisy) spikes in the AN response. This characteristic indicates that an LSR fibre has a lower SNR than an MSR fibre, and an HSR fibre has a higher SNR than the MSR fibre. Moreover, as the intensity of the input signal increases as observed in Figure 3-27, the SNRs for all three fibres also increase due to the stimulus-driven spikes being represented more in these fibres than spontaneously-driven spikes.

The effect of refractoriness also generally reduces the SNRs for the three AN fibres because a spike generated immediately after a LIF neuron fires is likely a stimulus-driven spike and a delay in the LIF neuron firing due to refractoriness, suppresses this spike. There are instances where there are gaps of silence in an input sound signal, such as a speech signal. In this case, the stimulus-driven spikes generated during the gaps is suppressed. If the refractoriness is enabled, more noisy spikes are generated, especially during the gaps, which reduces the SNR.

Table 3-6 and Table 3-7 provides a comparison between spike coincidence ratios of the AN responses generated from the CAR-Lite and CAR-Lite-SI models for the speech and musical signals, respectively. AN responses of the fixed-point implementation of the CAR-Lite model are shown in Figure 3-26(a3) and (b3) corresponding to the 0 dBFS musical signal [Figure 3-26(a1)] and 0 dBFS speech signal [Figure 3-26(a2)] respectively. Their corresponding AN responses from the fixed-point implementation of the CAR-Lite-SI model are displayed in Figure 3-26(a4)-(a6) and (b4)-(b6), respectively. It is expectedly difficult to determine input sound intensity from the AN response of the CAR-Lite model because this response contains no intensity information and thus, is incapable of capturing sound intensity levels in the musical and speech signals. This attribute is evident in the low coincidence matching ratio, C_{spk} , under the “Zero-Crossings” column of Table 3-6 and Table 3-7. In contrast, the AN response of the CAR-Lite-SI model can represent sound intensity by the difference of spike rates across three AN fibres indicating three discrete levels. This characteristic is visually apparent in Figure 3-26(a4)-a(6) and (b4)-(b6) as well as quantitatively, with the higher C_{spk} values under the “LSR”, “MSR”, and “HSR” columns of Table 3-6 and Table 3-7.

Another observation from Table 3-6 and Table 3-7 is that added noise either via PBNG, or LFSR circuit increases C_{spk} . Stimulus-driven spikes are generated when the intensity of

the TI signal increases above the firing thresholds of the LIF neurons. When the intensity of the TI signal is low, and below firing thresholds, spikes are generated spontaneously according to equations (3-20), (3-21), and (3-22) based on the AN fibre type. Hence, if a spontaneously-driven spike coincides with a binarized TI sample, then this indicates a low-intensity signal representation in the spike trains of the AN. As a result, the coincidence matching ratio, C_{spk} , increases when noisy spikes in the form of spontaneously-driven spikes are introduced. Disabling the refractoriness further increases C_{spk} according to the explanation above that led to the rise in the SNR. In other words, the highest C_{spk} readings for all three fibre types occur when either PBNG or LFSR is enabled and refractoriness disabled. If the refractoriness is enabled, then C_{spk} readings for all three fibre types fall.

The coincidence matching ratio, C_{spk} , is higher for the CAR-Lite-SI model than the CAR-Lite model. It is larger when spontaneously-driven (noisy) spikes are introduced. The C_{spk} as well as the SNRs for all three AN fibres also increase when the intensity of the input sound signal increases. However, there is no indication that noisy spikes at the AN stage aids in boosting signal coherence [83]–[85]. A further investigation is warranted to understand whether induced noisy spikes aids in boosting signal coherence that may include feature extraction, pattern recognition, and classification tasks.

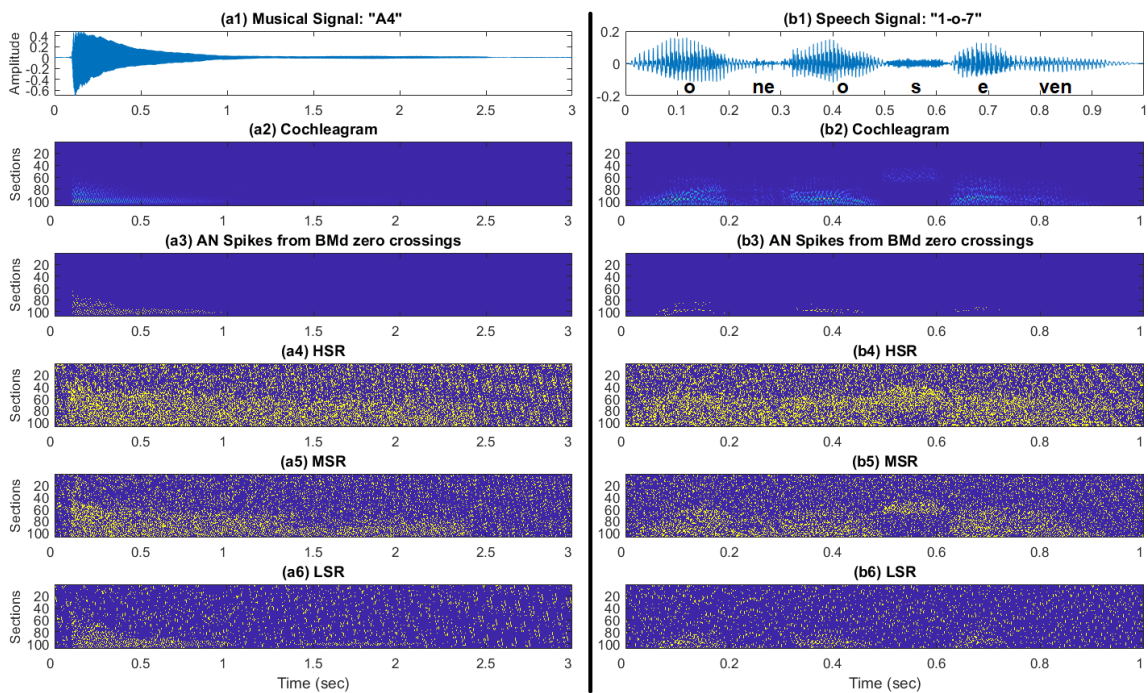


Figure 3-26: Sound intensity representation of real-world sound signals that includes (a1) musical forte note, A4, from a piano; and (b1) speech signal of a male voice uttering “1-o-7”, and their respective cochleagrams in (a2) and (b2), respectively. AN spikes generated from the fixed-point hardware implementation of the CAR-Lite model using (a3) the musical signal and (b3) the speech signal. AN spikes generated from the fixed-point hardware implementation of the CAR-Lite-SI model from the (a4) HSR, (a5) MSR, and (a6) LSR fibre responses of the musical signal; and (b4) HSR, (b5) MSR, and (b6) LSR responses of the speech signal. Note that the spontaneity is generated using LFSR and refractoriness is disabled.

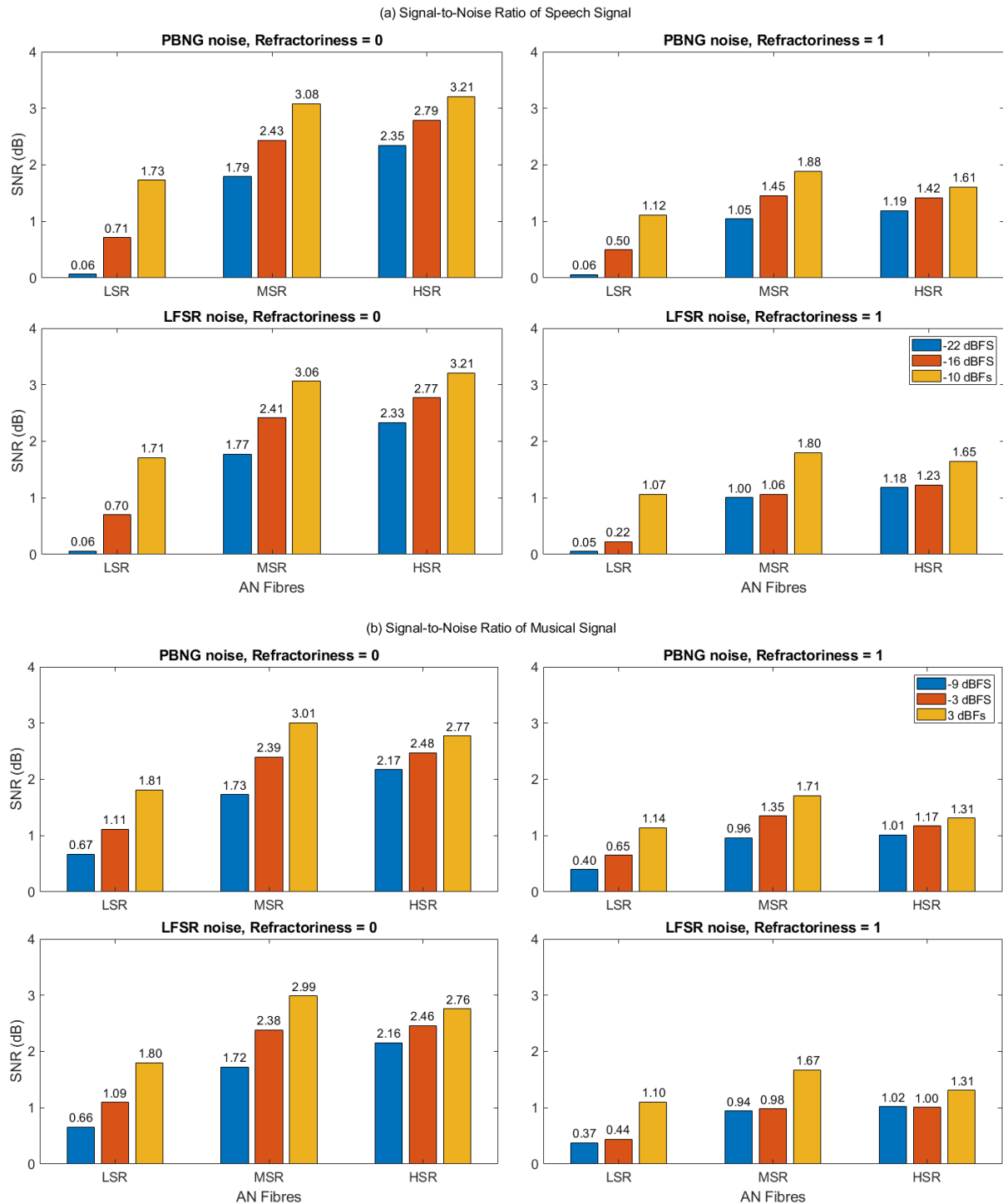


Figure 3-27: Signal-to-noise (SNR) ratio of the following real-world input signals with increasing intensity across the three AN fibres: HSR; MSR; and LSR under two noise (either PBNG or LFSR) and two refractoriness settings: (a) the speech signal of a male voice uttering “1-o-7”; and (b) the musical signal, A4 played from a piano.

Spontaneity		Refractoriness	Spike Coincidence Ratio			
LFSR (FPGA)	PBNG (Matlab)		Zero-crossings	LSR	MSR	HSR
0	0	0	0.008%	1.4%	6.6%	13%
0	0	1	0.008%	1.0%	4.6%	8.6%
0	1	0	0.008%	3.5%	10%	19%
0	1	1	0.008%	3.0%	7.6%	12%
1	0	0	0.008%	3.6%	10%	19%
1	0	1	0.008%	3.4%	8.2%	13%

Table 3-6: Coincidence matching ratio between the AN responses from the CAR-Lite model and the CAR-Lite-SI model, and the binarized TI response for the utterance of “1-o-7” under various spiking configurations.

Spontaneity		Refractoriness	Spike Coincidence Ratio			
LFSR (FPGA)	PBNG (Matlab)		Zero-crossings	LSR	MSR	HSR
0	0	0	0.002%	2.2%	17%	37%
0	0	1	0.002%	1.7%	11%	25%
0	1	0	0.002%	8.3%	27%	53%
0	1	1	0.002%	7.4%	20%	35%
1	0	0	0.002%	8.5%	22%	53%
1	0	1	0.002%	8.5%	22%	36%

Table 3-7: Coincidence matching ratio between the AN responses from the CAR-Lite model and the CAR-Lite-SI model, and the binarized TI response for the A4 piano note under various spiking configurations.

3.4. Chapter Summary and Conclusion

In the first half of this chapter, a small and simple cochlear model, known as CAR-Lite, comprising a cascade of asymmetric resonators (CAR), is presented. For an implementation of 108 cochlear sections, the model operates at nine different sampling rates, with an octave group of twelve sections operating at a specific sampling rate. It uses 16 bits signals and 8 bits coefficients, which is half of the bit width of coefficients used in the CAR model. Furthermore, the CAR-Lite model uses nine times fewer coefficients than the CAR model. On an Altera Cyclone V FPGA, the logic utilisation of the model is only 2.57%. Its dynamic range is 96 dB, which covers the range of speech, music and other perceivable audible stimuli. Stimuli at high intensity levels have saturated amplitudes resulting in information loss, which consequently yields saturated waveforms output from the model. However, using an automatic gain control (AGC) algorithm to vary the intensity levels of the stimuli enables the model to generate output responses without saturation.

In the second half of this chapter, the zero-crossing spiking algorithm in the CAR-Lite model is replaced with an abstract biologically plausible spiking algorithm capable of encoding sound at various intensity levels. Known as the CAR-Lite-SI model, the output of the CAR filters with high sampling rates are down-sampled, and the ones with low sampling rates are up-sampled to a common sampling rate before being converted to spikes using leaky-integrate-and-fire neurons. Spontaneous spike generation is also included in the model that increases the degree of similarity between the spike trains and the outputs of the CAR filters. This characteristic enables spike representation of real-world signals such as musical notes and speech to resemble closer to their output representation from the CAR filters. On an Altera Cyclone V FPGA, the logic utilisation is 15%, which allows room for expansion with the implementation of more sophisticated algorithms.

3.5. Bibliography

- [1] R. F. Lyon, “The Cascade of Asymmetric Resonators,” in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 293–298.
- [2] A. Saremi, R. Beutelmann, M. Dietz, G. Ashida, J. Kretzberg, and S. Verhulst, “A

- Comparative Study of Seven Human Cochlear Filter Models,” *J. Acoust. Soc. Am.*, vol. 140, no. 3, pp. 1618–1634, 2017, doi: 10.1121/1.4960486.
- [3] C. S. Thakur, T. J. Hamilton, J. Tapson, A. van Schaik, and R. F. Lyon, “FPGA Implementation of the CAR Model of the Cochlea,” in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1853–1856, doi: 10.1109/ISCAS.2014.6865519.
 - [4] C. S. Thakur, T. J. Hamilton, J. Tapson, A. van Schaik, and R. F. Lyon, “Live Demonstration: FPGA implementation of the CAR Model of the cochlea,” in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1853–1856, doi: 10.1109/ISCAS.2014.6865519.
 - [5] C. S. Thakur, R. M. Wang, S. Afshar, T. J. Hamilton, J. Tapson, S. A. Shamma, and A. van Schaik, “Sound Stream Segregation: A Neuromorphic Approach to Solve the ‘Cocktail Party Problem’ in Real-Time,” *Front. Neurosci.*, vol. 9, pp. 1–10, 2015, doi: 10.3389/fnins.2015.00309.
 - [6] J. B. Allen and M. M. Sondhi, “Cochlear macromechanics: Time domain solutions,” *J. Acoust. Soc. Am.*, vol. 66, no. 1, pp. 123–132, 1979, doi: 10.1121/1.383064.
 - [7] T. E. D. Painter and A. Spanias, “Perceptual Coding of Digital Audio,” *Proc. IEEE*, vol. 88, no. 4, pp. 451–513, 2000, doi: 10.1109/5.842996.
 - [8] P. Balazs, B. Laback, G. Eckel, and W. A. Deutsch, “Time – Frequency Sparsity by Removing Perceptually Irrelevant Components Using a Simple Model of Simultaneous Masking,” *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 18, no. 1, pp. 34–49, 2010, doi: 10.1109/TASL.2009.2023164.
 - [9] S. W. Smith, “The Discrete Fourier Transform,” in *The Scientist and Engineer’s Guide to Digital Signal Processing*, 1st ed., San Diego, CA, USA: California Technical Publishing, 1997.
 - [10] S. W. Smith, “The Fast Fourier Transform,” in *The Scientist and Engineer’s Guide to Digital Signal Processing*, 1st ed., San Diego, CA, USA: California Technical Publishing, 1997.
 - [11] R. F. Lyon, *Human and Machine Hearing: Extracting Meaning from Sound*. Cambridge University Press, 2017.
 - [12] R. F. Lyon, “Cascades of two-pole–two-zero asymmetric resonators are good models of peripheral auditory function,” *J. Acoust. Soc. Am.*, vol. 130, no. 6, p. 3893, 2011, doi: 10.1121/1.3658470.
 - [13] R. F. Lyon, “Waves in Distributed Systems,” in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 219–236.
 - [14] R. F. Lyon, “Resonators,” in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 145–168.
 - [15] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “RWC Music Database: Music Genre Database and Musical Instrument Sound Database,” in *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, 2003, pp. 229–230.
 - [16] X. Wang and B. E. Nelson, “Tradeoffs of designing floating-point division and square root on Virtex FPGAs,” in *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2003. FCCM 2003.*, 2003, doi:

10.1109/FPGA.2003.1227255.

- [17] P. Malík, "Natural logarithm and division floating-point high throughput co-processor implemented in FPGA," in *2016 IEEE Nordic Circuits and Systems Conference (NORCAS)*, 2016, pp. 1–6, doi: 10.1109/NORCHIP.2016.7792918.
- [18] C. J. Sumner, E. A. Lopez-Poveda, L. P. O'Mard, and R. Meddis, "A Revised Model of the Inner-Hair Cell and Auditory-Nerve Complex," *J. Acoust. Soc. Am.*, vol. 111, no. 5, pp. 2178–2188, 2002, doi: 10.1121/1.1453451.
- [19] L. H. Carney, "A model for the responses of low-frequency auditory-nerve fibers in cat," *J. Acoust. Soc. Am.*, vol. 93, no. 1, pp. 401–417, 1993, doi: 10.1121/1.405620.
- [20] X. Zhang, M. G. Heinz, I. C. Bruce, and L. H. Carney, "A phenomenological model for the responses of auditory-nerve fibers: I. Nonlinear tuning with compression and suppression," *J. Acoust. Soc. Am.*, vol. 109, no. 2, pp. 648–670, 2001, doi: 10.1121/1.1608963.
- [21] M. S. A. Zilany and I. C. Bruce, "Modeling auditory-nerve responses for high sound pressure levels in the normal and impaired auditory periphery," *J. Acoust. Soc. Am.*, vol. 120, no. 3, pp. 1446–1466, 2006, doi: 10.1121/1.2225512.
- [22] R. F. Lyon, "The Inner Hair Cell," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 320–330.
- [23] S. W. Smith, "Audio Processing," in *The Scientist and Engineer's Guide to Digital Signal Processing*, 1st ed., San Diego, CA, USA: California Technical Publishing, 1997, pp. 351–372.
- [24] M. Yitao and X. Li, "Music and Cochlear Implants," *J. Otol.*, vol. 8, no. 1, pp. 32–38, 2013, doi: 10.1016/S1672-2930(13)50004-3.
- [25] Y. Xu, C. S. Thakur, R. K. Singh, R. Wang, J. Tapson, and A. Van Schaik, "Electronic Cochlea : CAR-FAC Model on FPGA," in *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2016, pp. 564–567, doi: 10.1109/BioCAS.2016.7833857.
- [26] Mathworks, "corr2: 2-D correlation coefficient," 2017. <http://au.mathworks.com.au/help/images/ref/corr2.html> (accessed Oct. 15, 2017).
- [27] Y. Xu, C. S. Thakur, R. K. Singh, T. J. Hamilton, R. Wang, and A. van Schaik, "A FPGA Implementation of the CAR-FAC Cochlear Model," *Front. Neurosci.*, vol. 12, no. April, pp. 1–14, 2018, doi: 10.3389/fnins.2018.00198.
- [28] I. Gambin, I. Grech, O. Casha, E. Gatt, and J. Micallef, "Digital Cochlea Model Implementation Using Xilinx XC3S500E Spartan-3E FPGA," in *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*, 2010, pp. 946–949, doi: 10.1109/ICECS.2010.5724669.
- [29] M. P. Leong, C. T. Jin, and P. H. W. Leong, "Parameterized Module Generator for an FPGA-Based Electronic Cochlea," in *The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01)*, 2001, pp. 21–30.
- [30] R. K. Singh, Y. Xu, R. Wang, T. J. Hamilton, S. L. Denham, and A. van Schaik, "CAR-Lite: A Multi-Rate Cochlear Model on FPGA for Spike-based Sound Encoding," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 66, no. 5, pp. 1805–1817, 2019, doi: 10.1109/TCSI.2018.2868247.
- [31] S. A. Gelfand, "Acoustics and Sound Measurement," in *Essentials of Audiology*, 4th

- ed., New York, NY, US: Thieme Medical Publishers, 2016, pp. 1–29.
- [32] S. A. Gelfand, “Measurement Principles and the Nature of Hearing,” in *Essentials of Audiology*, 4th ed., New York, NY, US: Thieme Medical Publishers, 2016, pp. 70–90.
 - [33] C. J. Plack and A. J. Oxenham, “Overview: The Present and Future of Pitch,” in *Pitch: Neural Coding and Perception*, C. J. Plack, A. J. Oxenham, R. R. Fay, and A. N. Popper, Eds. New York, NY, USA: Springer, New York, NY, 2005, pp. 1–6.
 - [34] International Electrotechnical Commission, “723-03-11 - Dynamic Range,” 1997. <http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=723-03-11> (accessed Jun. 15, 2020).
 - [35] N. B. H. Croghan, K. H. Arehart, and J. M. Kates, “Quality and loudness judgments for music subjected to compression limiting,” *J. Acoust. Soc. Am.*, vol. 132, no. 2, pp. 1177–1188, 2012, doi: 10.1121/1.4730881.
 - [36] A. Bhatará, A. K. Tirovolas, L. M. Duan, B. Levy, and D. J. Levitin, “Perception of emotional expression in musical performance,” *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 37, no. 3, pp. 921–934, 2011, doi: 10.1037/a0021922.
 - [37] F.-G. Zeng, G. Grant, J. Niparko, J. Galvin, R. Shannon, J. Opie, and P. Segel, “Speech dynamic range and its effect on cochlear implant performance,” *J. Acoust. Soc. Am.*, vol. 111, no. 1, pp. 377–386, 2002, doi: 10.1121/1.1423926.
 - [38] G. Ballou, *Handbook for Sound Engineers*, 4th ed. MA, USA: Elsevier, 2008.
 - [39] M. Kirchberger and F. A. Russo, “Dynamic Range Across Music Genres and the Perception of Dynamic Compression in Hearing-Impaired Listeners,” *Trends Hear.*, vol. 20, no. February, pp. 1–16, 2016, doi: 10.1177/2331216516630549.
 - [40] Analog Devices, “SSM2603: Low Power Audio Codec (Rev. B).” Analog Devices, pp. 1–32, 2012.
 - [41] Terasic and Altera Corporation, “Cyclone V GX Starter Kit - User Manual.” Terasic, pp. 1–103, 2014.
 - [42] H. Fastl and E. Zwicker, “Loudness,” in *Psychoacoustics: Facts and Models*, 3rd ed., Springer-Verlag Berlin Heidelberg, 2007, pp. 203–238.
 - [43] T. Delbruck, “Introduction,” in *Event-Based Neuromorphic Systems*, 1st ed., S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, Eds. John Wiley & Sons Inc., 2015, pp. 1–6.
 - [44] R. M. Wang, C. S. Thakur, and A. van Schaik, “An FPGA-based Massively Parallel Neuromorphic Cortex Simulator,” *Front. Neurosci.*, vol. 12, no. April, pp. 1–18, 2018, doi: 10.3389/fnins.2018.00213.
 - [45] C. J. Sumner, E. A. Lopez-Poveda, L. P. O’Mard, and R. Meddis, “A Revised Model of the Inner-Hair Cell and Auditory-Nerve Complex,” *J. Acoust. Soc. Am.*, vol. 111, no. 5, pp. 2178–2188, 2002, doi: 10.1121/1.1453451.
 - [46] C. D. Geisler, “Coding of Acoustic Signals on the Auditory Nerve,” *IEEE Eng. Med. Biol. Mag.*, vol. 6, no. 2, pp. 22–28, 1987, doi: 10.1109/MEMB.1987.5006403.
 - [47] A. Merchan-Perez and M. C. Liberman, “Ultrastructural differences among afferent synapses on cochlear hair cells: Correlations with spontaneous discharge rate,” *J. Comp. Neurol.*, vol. 371, no. 2, pp. 208–221, 1996, doi: 10.1002/(SICI)1096-

- [48] C. M. Liberman, "Single-Neuron Labeling in the Cat Auditory Nerve," *Science* (80-.), vol. 216, no. 4551, pp. 1239–1241, 1982, [Online]. Available: <http://www.jstor.org/stable/1688751>.
- [49] R. Meddis, "Simulation of Mechanical to Neural Transduction in the Auditory Receptor," *J. Acoust. Soc. Am.*, vol. 79, no. 3, pp. 702–711, 1986, doi: 10.1121/1.393460.
- [50] S. W. Smith, "Moving Average Filters," in *The Scientist and Engineer's Guide to Digital Signal Processing*, 1st ed., San Diego, CA, USA: California Technical Publishing, 1997, pp. 277–284.
- [51] D. Eddins and D. Green, "Temporal integration and temporal resolution," in *Hearing*, B. C. J. Moore, Ed. Elsevier Inc., 1995, pp. 207–242.
- [52] R. Meddis, "Auditory-nerve first-spike latency and auditory absolute threshold: A computer model," *J. Acoust. Soc. Am.*, vol. 119, no. 1, pp. 406–417, 2006, doi: 10.1121/1.2139628.
- [53] R. FitzHugh, "Impulses and Physiological States in Theoretical Models of Nerve Membrane," *Biophys. J.*, vol. 1, no. 6, pp. 445–466, 1961, doi: 10.1016/S0006-3495(61)86902-6.
- [54] J. Nagumo, S. Arimoto, and S. Yoshizawa, "An Active Pulse Transmission Line Simulating Nerve Axon," *Proc. IRE*, vol. 50, no. 10, pp. 2061–2070, Oct. 1962, doi: 10.1109/JRPROC.1962.288235.
- [55] P. Heil and H. Neubauer, "Temporal integration of sound pressure determines thresholds of auditory-nerve fibers.," *J. Neurosci.*, vol. 21, no. 18, pp. 7404–15, 2001, doi: 10.1523/JNEUROSCI.1111-01.2001 [pii].
- [56] H. Mino, "The Effects of Spontaneous Random Activity on Information Transmission in an Auditory Brain Stem Neuron Model," *Entropy*, vol. 16, no. 12, pp. 6654–6666, 2014, doi: 10.3390/e16126654.
- [57] E. Schneidman, B. Freedman, and I. Segev, "Ion Channel Stochasticity May Be Critical in Determining the Reliability and Precision of Spike Timing," *Neural Comput.*, vol. 10, no. 7, pp. 1679–1703, 1998, doi: 10.1162/089976698300017089.
- [58] B. J. Richmond, "Stochasticity, spikes and decoding: Sufficiency and utility of order statistics," *Biol. Cybern.*, vol. 100, no. 6, pp. 447–457, 2009, doi: 10.1007/s00422-009-0321-x.
- [59] M. C. Liberman, "Auditory-nerve response from cats raised in a low-noise chamber," *J. Acoust. Soc. Am.*, vol. 63, no. 2, pp. 442–455, 1978, doi: 10.1121/1.381736.
- [60] C. J. Sumner, L. P. O'Mard, E. A. Lopez-Poveda, and R. Meddis, "A nonlinear filter-bank model of the guinea-pig cochlear nerve: Rate responses," *J. Acoust. Soc. Am.*, vol. 113, no. 6, p. 3264, 2003, doi: 10.1121/1.1568946.
- [61] M. Müller and D. Robertson, "Relationship between tone burst discharge pattern and spontaneous firing rate of auditory nerve fibres in the guinea pig," *Hear. Res.*, vol. 57, no. 1, pp. 63–70, 1991, doi: 10.1016/0378-5955(91)90075-K.
- [62] S. Anstis and S. Saida, "Adaptation to Auditory Streaming of Frequency-Modulated Tones," *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 11, no. 3, pp. 257–271, 1985,

doi: 10.1037/0096-1523.11.3.257.

- [63] X. Zhang, M. G. Heinz, I. C. Bruce, and L. H. Carney, "A Phenomenological Model for the Responses of Auditory-Nerve Fibers : I . Nonlinear Tuning," *J. Acoust. Soc. Am.*, vol. 109, no. 2, pp. 648–670, 2001, doi: 10.1121/1.1336503.
- [64] M. Matsumoto and Y. Kurita, "Twisted GFSR generators," *ACM Trans. Model. Comput. Simul.*, vol. 2, no. 3, pp. 179–194, 1992, doi: 10.1145/146382.146383.
- [65] J. L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Trans. Inf. Theory*, vol. 15, no. 1, pp. 122–127, 1969, doi: 10.1109/TIT.1969.1054260.
- [66] R. Meddis and E. A. Lopez-Poveda, "Auditory Periphery: From Pinna to Auditory Nerve," in *Computational Models of the Auditory System*, R. Meddis, E. A. Lopez-Poveda, R. R. Fay, and A. N. Popper, Eds. New York, Dordrecht, Heidelberg, London: Springer, 2010, pp. 7–38.
- [67] P. Heil and A. J. Peterson, "Basic response properties of auditory nerve fibers: a review," *Cell Tissue Res.*, vol. 361, no. 1, pp. 129–158, 2015, doi: 10.1007/s00441-015-2177-9.
- [68] R. Meddis, "Auditory-nerve first-spike latency and auditory absolute threshold: A computer model," *J. Acoust. Soc. Am.*, vol. 119, no. 1, pp. 406–417, 2006, doi: 10.1121/1.2139628.
- [69] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Linear Feedback Shift Register," in *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge, UK: Cambridge University Press, 2007, pp. 380–385.
- [70] C. J. Sumner, E. A. Lopez-Poveda, L. P. O'Mard, and R. Meddis, "Adaptation in a Revised Inner-Hair Cell Model," *J. Acoust. Soc. Am.*, vol. 113, no. 2, pp. 893–901, 2003, doi: 10.1121/1.1515777.
- [71] R. Sarpeshkar, M. W. Baker, C. D. Salthouse, J.-J. Sit, L. Turicchia, and S. M. Zhak, "An Analog Bionic Ear Processor with Zero-Crossing Detection," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, 2005, pp. 2004–2005, doi: 10.1109/ISSCC.2005.1493877.
- [72] J. Lazzaro, J. Wawrzynek, and A. Kramer, "Systems Technologies for Silicon Auditory Models," *IEEE Micro*, vol. 14, no. 3, pp. 7–15, 1994, doi: 10.1109/40.285219.
- [73] T. J. Hamilton, C. Jin, A. van Schaik, and J. Tapson, "An Active 2-D Silicon Cochlea," *IEEE Trans. Biomed. Circuits Syst.*, vol. 2, no. 1, pp. 30–43, 2008, doi: 10.1109/TBCAS.2008.921602.
- [74] S. C. Liu, A. Van Schaik, B. A. Minch, and T. Delbruck, "Asynchronous Binaural Spatial Audition Sensor With 2x64x4 Channel Output," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 4, pp. 453–464, 2014, doi: 10.1109/TBCAS.2013.2281834.
- [75] S. Wang, T. J. Koickal, A. Hamilton, R. Cheung, and L. S. Smith, "A Bio-Realistic Analog CMOS Cochlea Filter With High Tunability and Ultra-Steep Roll-Off," *IEEE Trans. Biomed. Circuits Syst.*, vol. 9, no. 3, pp. 297–311, 2015, doi: 10.1109/TBCAS.2014.2328321.
- [76] R. K. Singh, Y. Xu, R. Wang, T. J. Hamilton, S. L. Denham, and A. van Schaik, "CAR-Lite: A Multi-Rate Cochlea Model on FPGA," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5, doi: 10.1109/ISCAS.2018.8351394.

- [77] M. Meyer, R. R. Fay, and A. N. Popper, "Frequency tuning and intensity coding of sound in the auditory periphery of the lake sturgeon, *Acipenser fulvescens*," *J. Exp. Biol.*, vol. 213, no. 9, pp. 1567–1578, 2010, doi: 10.1242/jeb.031757.
- [78] J. E. Rose, J. E. Hind, D. J. Anderson, and J. F. Brugge, "Some effects of stimulus intensity on response of auditory nerve fibers in the squirrel monkey.," *J. Neurophysiol.*, vol. 34, no. 4, pp. 685–699, 1971, doi: 10.1152/jn.1971.34.4.685.
- [79] D. Pressnitzer, R. D. Patterson, and K. Krumbholz, "The lower limit of melodic pitch," *J. Acoust. Soc. Am.*, vol. 109, no. 5, pp. 2074–2084, 2001, doi: 10.1121/1.1359797.
- [80] D. Y. Chung and F. B. Colavita, "Periodicity pitch perception and its upper frequency limit in cats," *Percept. Psychophys.*, vol. 20, no. 6, pp. 433–437, 1976, doi: 10.3758/BF03208278.
- [81] O. Macherey and R. P. Carlyon, "Re-examining the upper limit of temporal pitch," *J. Acoust. Soc. Am.*, vol. 136, no. 6, pp. 3186–3199, 2014, doi: 10.1121/1.4900917.
- [82] R. Meddis, "Matlab Auditory Periphery (MAP) Model Technical Description." Essex, pp. 1–32, 2011.
- [83] F. G. Zeng, Q.-J. Fu, and R. Morse, "Human hearing enhanced by noise," *Brain Res.*, vol. 869, no. 1–2, pp. 251–255, 2000, doi: 10.1016/S0006-8993(00)02475-6.
- [84] R. P. Morse and E. F. Evans, "Additive noise can enhance temporal coding in a computational model of analogue cochlear implant stimulation," *Hear. Res.*, vol. 133, no. 1–2, pp. 107–119, 1999, doi: 10.1016/S0378-5955(99)00062-3.
- [85] S. E. Behnam and F. G. Zeng, "Noise improves suprathreshold discrimination in cochlear-implant listeners," *Hear. Res.*, vol. 186, no. 1–2, pp. 91–93, 2003, doi: 10.1016/S0378-5955(03)00307-1.

4. Auditory Pitch Model: Autocorrelogram Generation

(In memory of Ray Meddis)

This chapter presents a novel algorithm for calculating an autocorrelogram, which contains pitch information. The next section presents the motivation behind this novel algorithm. After that, details of the conventional algorithm and the novel algorithm for calculating an autocorrelogram are presented. The novel algorithm is applied to an auditory pitch model, and the resulting autocorrelograms are compared to autocorrelograms generated from the conventional algorithm to understand the degree of similarity between them. Two FPGA implementations of the model are presented – one with the conventional algorithm and the other with the novel algorithm. Their results are compared to indicate the benefits of the hardware implementation of operating the new algorithm over the conventional one. Additionally, the hardware responses of the model fitted with the novel algorithm are used to explain several auditory pitch phenomena.

4.1. Motivation

The workings of the bipolar cells in the vision pathway inspire the novelty of generating autocorrelogram for the pitch perception model presented in this chapter. The bipolar cells transmit neural images from the retina to the cortical regions of vision through ON and OFF events when there is a change in the field of view [1]. This binary event served as an inspiration for the development of event-based silicon retina such as the dynamic vision sensor (DVS) camera [2] and the asynchronous time-based image sensor (ATIS) camera [3], [4].

Out of the three auditory pitch models reviewed in chapter 2, Meddis physiological model of virtual pitch has been implemented on a field-programmable gate array (FPGA), which is a configurable digital electronics hardware platform – Lim et al. [5] and Jones et al. [6] used binary ON and OFF events to represent chopper neuron firing responses in the ventral cochlear nucleus (VCN) in FPGA. From the VCN module, the spikes are sent to an inferior colliculus (ICC) module housing coincidence neurons that sum up the number of spikes (ON events) and compare the summed value to a threshold using a comparator circuit. A coincidence neuron behaves identically in spiking operation to the chopper neurons and outputs an ON event via a spike when the sum of input spikes value crosses a threshold. Otherwise, the coincidence neuron outputs an OFF event (no spike). This binary property can be applied to the auditory nerve (AN) signal since the distribution of the time intervals between successive output spikes in the auditory nerve contributes to the pitch periodicity of the input sound signal [7].

Alternatively, an analogue very-large-scale-integrated (VLSI) chip implementation of the VCN-ICC model has been implemented by van Schaik and Meddis [8], [9]. van Schaik improved his VLSI design by incorporating a simple 2-input logical-AND gate to model coincidence detector neuron, using ON and OFF responses to extract periodicities between two successive spikes [10]. Using the binary ON and OFF events such as those used by Lim et al., Jones et al. and van Schaik et al. as well the utilisation of 2-input logical-AND gates from van Schaik et al., I will show in this chapter that the autocorrelograms generated from the model reviewed in chapter 2 are capable of using fewer computational resources on FPGA than the conventional means of computing these autocorrelograms. For estimating pitch information from these autocorrelograms, the reader is advised to refer to chapter 6.

In the next section, the algorithm is presented. The algorithm is then applied to CAR-Lite-ACF model presented in section 4.4. Within this section, FPGA implementations are presented with conventional mathematical operations as well as with the novel algorithm for comparison.

4.2. Algorithm Characteristics

The critical ingredient in calculating an autocorrelogram using either the conventional method or the novel method is the type of input signal. As presented in chapter 2, auditory pitch models using the autocorrelation function (ACF) model rely on low-pass filtered inner hair cell (IHC) signals from a cochlear model as their input. Hence, the input signal is made up of real-valued numbers. From a computational perspective, each point in the correlation matrix calculation requires the usage of an arithmetic logic unit (ALU) found in a central processing unit (CPU). The total number of points in an autocorrelogram, (the output correlation matrix), is dependent on the number of cochlear sections as well as the temporal window lag size.

Let us consider the dimensions of the cochlear model (CAR-Lite) described in chapter 3 for use with an ACF model to be described in a later section of this chapter. For this cochlear model, there are 108 sections. Assuming a temporal lag size of 2,048, an autocorrelogram generated from the output of these 108 sections contains 221,184 ($108 \times 2,048$) values, as observed in Figure 4-2. From a computational perspective, the multiplier of the ALU has to be invoked as many as 221,184 times at the onset of every 2,048 input sample. In real-time operation, this may block the processing of other functions that rely on the multiplier.

An alternative is to use more CPU cores containing multiple ALUs at higher clock speeds that can operate in parallel. However, this would come at a high cost of computational redundancy and increased power consumption.

Furthermore, CPU usage for autocorrelogram generation in real-time requires an understanding of the operating system (OS). The manner the OS divides software applications into multiple computational segments called threads to be processed on available CPU in a round-robin, and priority-based format also requires understanding, i.e. high-priority threads are serviced more frequently than low-priority threads [12]. This notion increases the complexity of implementing real-time autocorrelogram generation. Alternative platforms such as embedded systems consisting of affordable small to medium-sized microcontrollers usually do not possess sufficient CPUs, clock speed as well as high-speed data transmission medium to handle the calculation mentioned in the preceding paragraph. Although large and costlier microcontrollers may be capable of handling the calculations, CPU accesses are done through a real-time OS, which adds to development complexity and leads to higher power consumption than small- and medium-sized microcontrollers [13].

To reduce computational resources for generating an autocorrelogram, I propose to use binary spike trains generated from the auditory nerve as input to the ACF model instead of real-valued inner hair cell (IHC) signal. The binary spikes generated are pulses, and each binary spike has two states: ON and OFF. Hence, with binary spikes, only a single bit is required to represent an input signal to generate an autocorrelogram as opposed to an n bit real-valued IHC signal. This approach obviates the need for a conventional CPU as there is only be a single bit active. A resource-efficient method to calculate an autocorrelogram from binary spikes is to develop a customised CPU made up of a small number of logic gates with

a straightforward configuration. This form of processing can be implemented on an FPGA. A disadvantage with an autocorrelogram generated from binary spikes is that low amplitude sound information is lost, which may correspond to information from other sound sources. Here, low amplitude sound refers to magnitudes of sound lower than the salient or noticeable sound that is usually captured in an autocorrelogram generated using real-valued IHC signal. Despite the loss of information, the autocorrelogram still captures sufficient salient pitch information, as observed in this chapter as well as chapter 6.

An autocorrelogram generated from an ACF model [14], [15] uses multiply-and-accumulate (MAC) operations iteratively:

$$r_g(s, t, \tau) = \sum_{\tau=0}^{N-\tau} g(s, t) \cdot g(s, t - \tau) \quad (4-1)$$

where τ is a lag corresponding to the time-delayed shift of g representing an IHC sample output from a cochlear section, s , at time, t ; N is an upper limit where τ is capped.

In equation (4-1), an input sample, g , is a base-10 number; the dot operator represents multiplication. For m bit g values, the product between two g values results in a $2m$ bit number. This number is then normalised to m bits by dividing the $2m$ bit number by 2^m . After that, the bit-width of an output sample in r_g is calculated as $\log_2(N \cdot 2^m)$. As an example, if the values of g are 16 bit numbers, then each multiplication results in a 32 bit number. This 32 bit output sample is normalised to a 16 bit output sample, by dividing the 32 bit number with 2^{16} , as illustrated in Figure 4-1(a). Then, summing up M and N number of 16 bit values would result in $\log_2(N \cdot 2^{16})$ bits to represent an output sample in r_g . For N with a length of 2,048, an output sample in r_g is representable by at least 27 bits.

Using binary spike trains as inputs, equation (4-1) can be modified to use only logical-AND and accumulate operations (AAC) in the following respective forms:

$$r_b(s, t, \tau) = \sum_{\tau=0}^{N-\tau} b(s, t) \wedge b(s, t - \tau) \quad (4-2)$$

where b is a single bit binary number; \wedge represents a logical-AND operation. Each AND operation results in a single bit output as opposed to the $2m$ bit number required using equation (4-1) as illustrated in Figure 4-1(b). Moreover, no normalisation is required. The bit-width of an output sample in r_b , is now $\log_2(N \cdot 2)$ instead of $\log_2(N \cdot 2^m)$. Note that the former equation is calculated from the latter equation with $m = 1$ representing 1 bit. Hence, for N with a length of 2,048, an output sample in r_b is 12 bits. This exercise reduces memory utilisation based on a point value on an autocorrelogram by as much as 2.2 times, i.e. 27 bits / 12 bits. Despite the memory utilisation reduction, the use of low bit widths indicates a loss in precision, which, in turn, means the possible loss of pitch information.

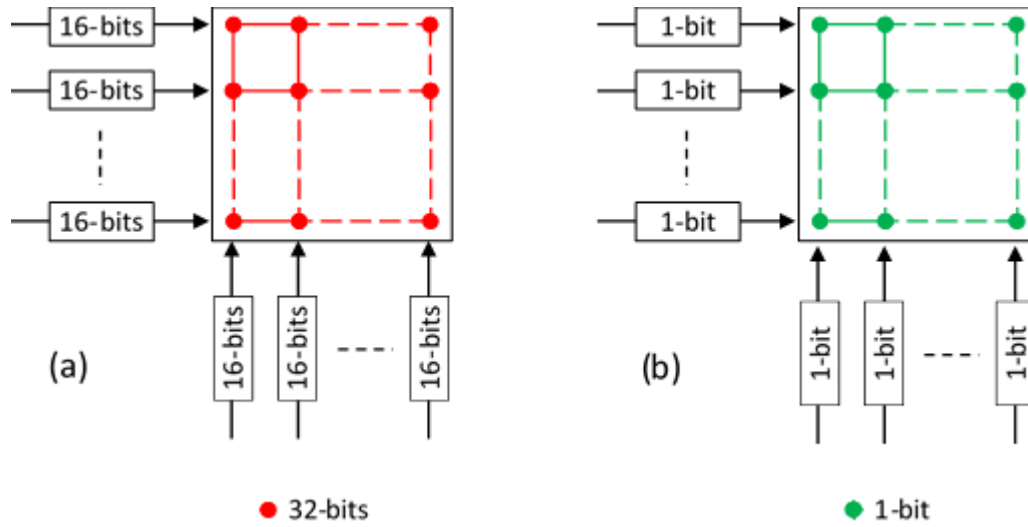


Figure 4-1: Bit width differences of elements in coincidence matrices calculated from (a) multiplication operation (red dot) using 16 bits IHC input and (b) logical-AND operation (green dot) using 1 bit AN input.

On FPGA (specifically the Altera Cyclone V 5CGXFC5C6F27C7N chip), the number of computational resources such as multipliers and combinational logic circuits (known alternatively as adaptive logic modules or ALMs) is limited. In terms of quantity (Qty), multipliers (Qty: 450) are a scarcer commodity than the ALMs (Qty: 29,080) [16]. There are three types of multipliers on board the FPGA: firm, soft, and combined. Firm multipliers use dedicated DSP circuit blocks numbering 150 in total, whereas soft multipliers use look-up tables (LUT) in configurable ALMs numbering 300 in total. Combination multipliers use a mix of DSP and ALM circuits numbering 450 in total. Even with 450 multipliers in total, it is insufficient to accommodate the parallel calculation of all the points in an autocorrelogram matrix comprising 221,184 multiplications. Instead, the multipliers would be time-multiplexed - the use of the multipliers is broken down over time to service all the multiplication operations, thereby reducing the need for a high number of parallel multiplier circuits. In contrast, the use of equation (4-2) to generate an autocorrelogram matrix enables logical-AND gates utilisation without using any computationally expensive and sophisticated multiplier circuits.

The advantages of the AAC operations algorithm defined by equation (4-2) over the conventional MAC operations algorithm defined by equation (4-1) are the computation, power and resource costs. The cost of the first three CPU processors mentioned in Table 4-1(a) increases with increasing CPU cores. The number of CPU cores indicates the number of multiplier circuits used as part of an ALU circuit. So, to accommodate the computations of the proposed auditory pitch model, a large-sized CPU core may be needed. Alternatively, a medium-sized CPU is usable by operating the CPU chip at a high global clock speed along with a real-time operating system to slice the multiply-accumulate (MAC) operations into smaller manageable operations that can be serviced by a CPU quickly over multiple global clock pulses instead of one. This task is known as a pipeline operation [13]. Nonetheless, the cost of the MAC-based CPU chips outweighs the cost of logical-AND chips observed in Table 4-1(b) as well as FPGA chips projected in Table 4-1(c) that possess reconfigurable digital logical elements and digital signal processors (DSP) equipped for both AAC-based and MAC-based operations. Hence, it is certainly cost-effective to utilise the novel AAC-based algorithm for real-time implementation over the conventional MAC-based algorithm.

(a) Processor	Clock Speed (Hz)	Number of MAC operations	Power Consumption (W)	Number of DSP CPU Cores	Chip Size (mm)	Cost (US \$)
TMS320C5545 [17]	150×10^6	400×10^6	150×10^{-3}	1	7×7	2.76
ADAU1467 [18]	295×10^6	1.2×10^9	69×10^{-3}	1	12×12	8.78
66AK2H14 [19]	1.2×10^9	307.2×10^9	10.2 to 16	8	40×40	661.26
(b) Logical-AND	Clock Speed (Hz)	Number of MAC operations	Power Consumption (W)	Number of AND gates	Chip Size (mm)	Cost (US \$)
SN74AUP1G08 [20]	-	-	3×10^{-6}	1	2.5×2.3	0.08
SN74AUC08 [21]	-	-	27×10^{-6}	4	4×4	0.26
(c) FPGA	Series	Clock Speed (Hz)	Number of Logic Elements	Number of Adaptive Logic Modules	Number of I/O pins	Cost (US \$)
5CEBA2U15C8N [22]	Cyclone V	550	25×10^3	9,434	176	39.53
5CGXFC7D7F31C8N [23]	Cyclone V	550	149×10^3	56,480	480	249.95

Table 4-1: Survey of computational resources for using (a) MAC-based computational processors, (b) logical-AND chips, and (c) a combination of MAC-based and AAC-based FPGA chips.

4.3. General Model Characteristics

The pitch model described in section 4.4 uses the output signals from the CAR-Lite model described in chapter 3 as their input. Based on the mathematical operation used in the pitch model, the input signal used is either the inner hair cell (IHC) or the auditory nerve (AN) signal. The IHC signal is described in section 3.2, while the AN signal is generated using leaky-integrate-and-fire (LIF) neuron, similar to the one described in subsection 3.3.2:

$$v_{LIF}(s, t) = v_{LIF}(s, t - 1) + c_{LIF}(IHC(s, t) - v_{LIF}(s, t - 1)) \quad (4-3)$$

$$AN(s, t) = \begin{cases} 1, & v_{LIF}(s, t) > thresh \\ 0, & v_{LIF}(s, t) \leq thresh \end{cases} \quad (4-4)$$

where v_{LIF} is an internal state voltage, which is reset to 0 after the AN associated with it fires a spike; c_{LIF} is a low-pass coefficient; $thresh$ is the firing threshold of a LIF neuron for section s , at time t .

In the CAR-Lite-SI model described in subsection 3.3.2 of chapter 3, the AN signal is resampled to a standard sampling rate of 3 kHz to ensure spike rates remained under 1,000 spikes per second. In the case of autocorrelogram generation for calculating pitch in this dissertation, this AN resampling is removed, thus, preserving the multiple sampling rates used by the different octaves in CAR-Lite, which enables an autocorrelogram to capture a broad pitch range. Additionally, spontaneous spike generation in the form of pseudorandom binary noise, and LIF neuron refractoriness are not implemented for the model in this chapter as the objective is to know the minimum computing mechanism required to generate an autocorrelogram on an FPGA.

When the CAR-Lite-ACF model uses the IHC signal as its input, its respective autocorrelogram output is calculated using MAC operations. When the model uses the AN signal as its input, its autocorrelogram output is calculated using AAC operations. The MAC-based model is designed first, in floating-point arithmetic in Matlab. This is regarded as the software implementation and is used as a benchmark for the design of the AAC-based

model, specifically in the determination of the firing threshold voltages of the leaky-integrate-and-fire (LIF) neurons. From the floating-point model, the fixed-point version is implemented in Matlab, which is regarded as the hardware implementation as it is designed to be implementable on an FPGA. From here, they are implemented on an Altera Cyclone V GX starter kit with a 5CGXFC5C6F27C7 FPGA chip via SystemVerilog using the Altera Quartus software application. SystemVerilog is used over Verilog and VHDL because it is a versatile hardware descriptive language that combines the properties of Verilog in terms of enabling quick C-style writeups of the model as well as the properties of VHDL in terms of early error detection during model design [24].

As far as the bit widths are concerned for the hardware model, each element in a MAC-based coincidence matrix (the matrix that holds real-valued numbers calculated from the product of two real-values before summation) is set at 16 bits – the same bit width as the inner hair cell (IHC) signal. For the AAC-based coincidence matrix (the matrix that holds only binary numbers, where each binary number is the result of the logical-AND operation between two binary numbers representing binary spikes), this bit width is set at 1 bit per matrix element. To ensure sufficient bits to represent the result of accumulation, each element in either MAC-generated or AAC-generated autocorrelogram is 32 bits wide. All other bit widths and variable settings of the CAR-Lite-ACF model used in section 4.4 are summarised in Table 4-2.

Index	Description	Variable	Setting
1	Sampling rate, f_s , of cochlear octave o_n , i.e. 9 sampling rates across 9 octaves.	$f_s - o_1$ to $f_s - o_9$	96 kHz to 375 Hz (in division of 2 starting from 96 kHz).
2	108 centre frequency, f_c , for 108 logarithmically-spaced cochlear sections.	f_c	50 Hz (section 108) to 24 kHz (section 1).
3	CAR filter coefficient bit width.	a, c, h, r	8 bits per coefficient.
4	Basilar membrane (BM) signal bit width (CAR filter output).	BM	16 bits per sample.
5	BM velocity signal bit width.	BMd	16 bits per sample.
6	Inner hair cell (IHC) bit width.	IHC	16 bits per sample.
7	Point sample bit width on a MAC-based coincidence matrix – each sample is the output of a multiplication operation only.	C_{ij}	16 bits per sample.
8	Intermediate variable bit width holding multiplication results between two 16 bit numbers.	y_{int}	32 bits
9	Point sample bit width on an autocorrelogram from both MAC- and AAC-based implementations.	G_{ij} or r_g	32 bits per sample.
10	LIF neuron voltage coefficient for AAC-based CAR-Lite-ACF model.	c_{LIF}	$f_c(I)/f_s - o_1 = 24 \text{ kHz} / 96 \text{ kHz} = 0.25$
11	LIF neuron voltage coefficient bit width.	c_{LIF}	16 bits
12	LIF neuron voltage firing threshold (CAR-Lite-ACF).	$v_{LIF-fire}$	0.27 V
13	LIF neuron firing threshold bit width.	$v_{LIF-fire}$	16 bits
14	Auditory nerve (AN) or LIF neuron output signal bit width.	AN	1 bit
15	Autocorrelation lag window size.	T	2,048 samples covering up to 50 Hz.
16	FPGA system clock speed.	$f_{FPGA-sys}$	250 MHz
17	FPGA audio codec speed.	$f_{FPGA-aud}$	96 kHz

Table 4-2: Settings of the CAR-Lite-ACF model implemented on software (via Matlab) and hardware (FPGA via SystemVerilog).

4.4. CAR-Lite-ACF Model

This model determines periodicity pitch with temporal cues output from an auditory model. It does this in three stages, using a cochlear model, cochlear section-by-section autocorrelation for periodicity detection, and cross-section integration of the resulting autocorrelation output to generate temporal periodicity profiles, also known as the summary autocorrelation function (ACF) [14]. Figure 4-2 illustrates a diagram of the model operating using AAC operations. For MAC operations, the input to the autocorrelation stage is the half-wave rectified inner hair cell (IHC) signal instead of the auditory nerve (AN) spikes.

The multiple sampling rates of the CAR-Lite stage are maintained here. Ideally, this indicates that the number of output samples reduces by a factor of 2 per octave. Thus, the lengths of the output signals for every octave differ. However, to analyse the cochleagram across all octaves, the output signals are padded with previously computed output samples at the reduction factor of the octaves. This action enables the lengths of all octaves to be the same, as was demonstrated in chapter 3. Hence, the highest sampling rate of octave 1 is the global sampling rate across all octaves, and the multiple sampling rates are deemed as the sampling rates local to the octaves. In this case, the input and output of the ACF stage are also governed by this global sampling rate.

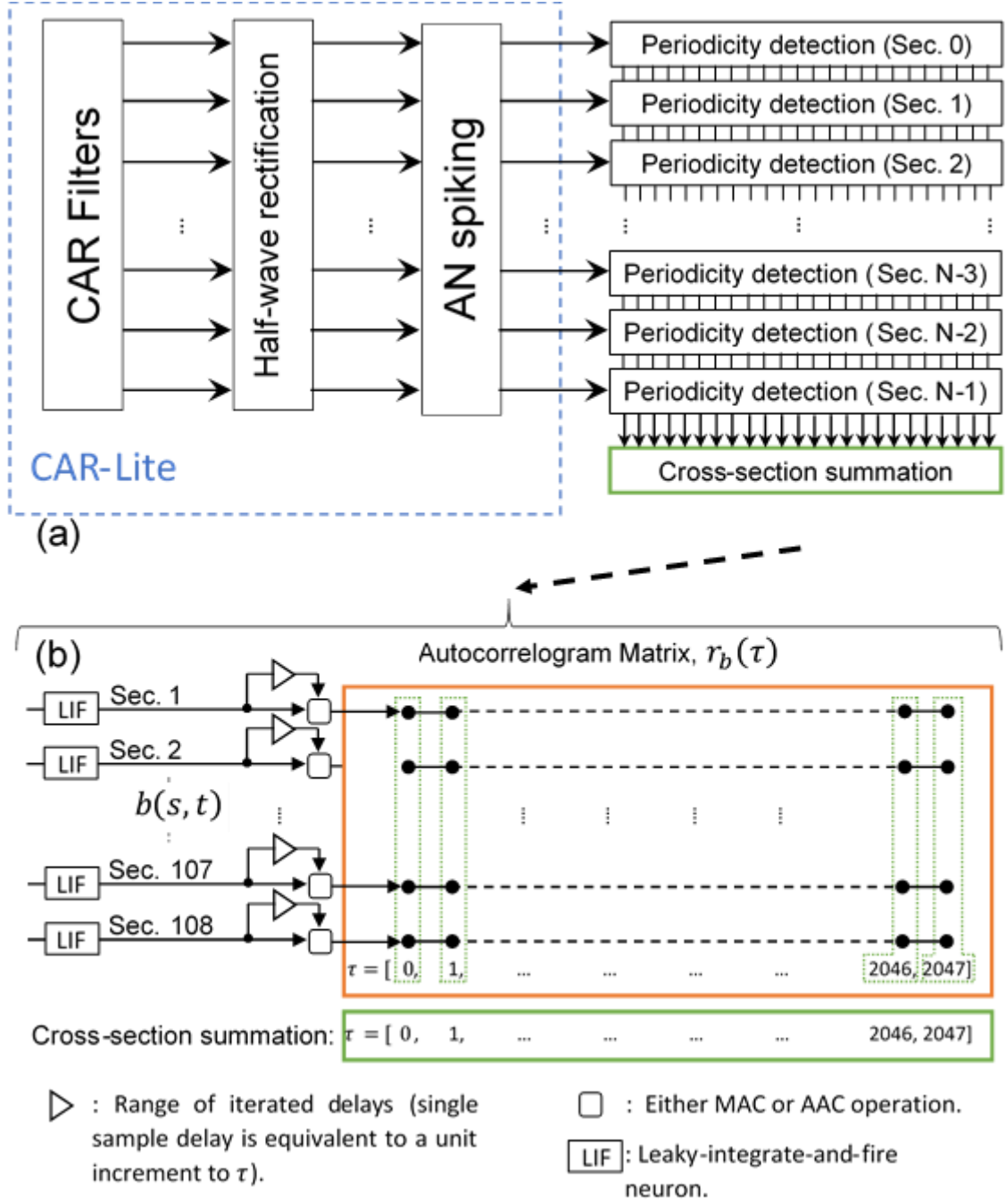


Figure 4-2: (a) The CAR-Lite-ACF model for pitch extraction from temporal cues in the sound signal. (b) A view of the periodicity detection algorithm, resulting in a 2D autocorrelogram matrix (orange) generated using either MAC or AAC operation and the cross-section summation (green), resulting in a 1D temporal profile containing pitch information. The LIF neurons binarize the inner hair cell (IHC) inputs from the CAR-Lite cochlear model.

For the AAC-based model, the firing threshold of the leaky-integrate-and-fire (LIF) neuron emulating the auditory nerve stage corresponds to the maximum of a correlation coefficient [calculated from equation (4-9)] between a benchmark floating-point MAC-based autocorrelogram and fixed-point AAC-based autocorrelograms. The latter autocorrelograms are calculated from a linear distribution of firing thresholds from 0.23 V to 0.31 V in 0.01 V intervals. This range is selected as it yields high correlation coefficients above 0.96. The selected firing threshold is 0.27 V from the correlation coefficient in Figure 4-3 with a complex tone used as the input signal. This firing threshold is used to ensure that all AAC-

based autocorrelograms in this dissertation resemble MAC-based autocorrelograms as closely as possible. The same stimulus is used to indicate the similarity in responses of the floating-point and fixed-point implementations, as illustrated in subsection 4.4.2. Subsection 4.4.3 showcases the capability of the hardware AAC-based model to explain missing fundamental frequency phenomenon, and subsection 4.4.4 uses the same model to explain the phenomenon of harmonics phase change effect on pitch.

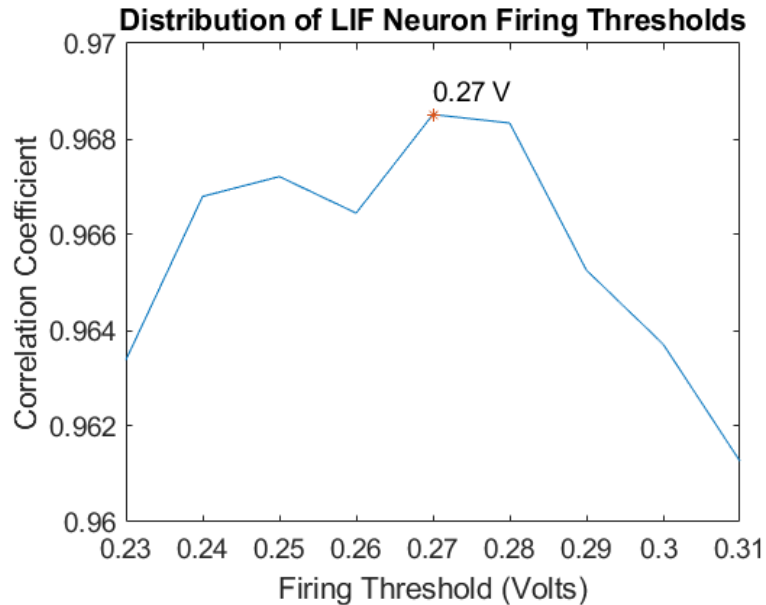


Figure 4-3: LIF neuron firing threshold determined with high confidence from the maximum correlation coefficient calculated in the comparison between MAC-based and AAC-based autocorrelograms.

4.4.1. FPGA Implementation

The Altera Cyclone V FPGA is used for the implementation of the CAR-Lite-ACF model in this subsection. The model settings are presented in Table 4-2 with the FPGA system clock and audio clock rates set at 250 MHz and 96 kHz, respectively.

This section is further segmented into two subsections. Subsection 4.4.1.1 describes the CAR-Lite-ACF model implemented on FPGA using MAC operations. Subsection 4.4.1.2 describes the same model implemented on FPGA using AAC operations.

4.4.1.1. CAR-Lite-ACF model using MAC operations

Figure 4-4 illustrates the CAR-Lite-ACF model implemented with multiply-accumulate (MAC) operations. There are three modules implemented in SystemVerilog: supervisor, BM-IHC, and ACF modules. The supervisor module manages the overall operation, while the BM-IHC module generates the 2D time-frequency signal as input to the ACF module for generating an autocorrelogram. In this case, as the ACF module here relies specifically on the inner hair cell (IHC) signal as its input, the calculation of the auditory nerve signal is omitted from the BM-IHC module.

The supervisor module monitors the acquisition of input audio data sample from an audio buffer and oversees the data transfer to and from the BM-IHC and ACF modules. The BM-IHC module hosts the equations that characterise the CAR-Lite cochlear model, while the ACF module contains the autocorrelation function (ACF) equations using MAC operations. The latency of the BM-IHC module is 64 ns (16 states \times 1/250 MHz)

corresponding to one cochlear section and 6.91 μ s for 108 cochlear sections. The latency of the ACF module is 8.21 μ s ($\{2 \text{ iterative states} \times 2,048 \text{ delay samples} / 2 \text{ parallel multiply, right-shift, and accumulate (MRSAC) operations} + 4 \text{ non-iterative states}\} \times 1/250 \text{ MHz}$) for calculating 2,048 samples in the delay vector per cochlear section. For 108 cochlear sections, the ACF module is instantiated 108 times, and so, the latency of this parallel invocation is equivalent to invoking the ACF module once. Overall, the latencies of the BM-IHC and ACF modules are each below 2,604 system clock cycles (based on 250 MHz) or 10.41 μ s, which is the duration between the arrival of two sequential input samples (based on 96 kHz audio sampling rate).

The ACF module is instantiated 108 times, which means that the circuit defining the characteristics of the ACF operation is cloned 108 times – one cochlear section output sample to one ACF module. In other words, when the BM-IHC module outputs an IHC output sample of a specific cochlear section, its dedicated ACF module is activated to run as seen in Figure 4-5. An alternative solution is to time-multiplex a single ACF module to process all the cochlear sections in a pipelined manner. However, this manner of processing is not ideal as the ACF module is required to perform 221,184 (108 cochlear sections \times 2,048 lag samples) multiply, right-shift, and accumulate (MRSAC) operations for every input sample acquired from the audio codec. Since the FPGA system clock is 250 MHz, there are only 2,604 clock cycles available between the arrivals of two adjacent input sound samples. This number is translatable as the maximum number of clock cycles available to finish 221,184 MRSAC operations. So, pipelining the MRSAC operations clearly cannot be accommodated unless either the system clock rate increases or each cochlear section is addressed by its own dedicated ACF module, i.e. ACF modules operating in parallel. Doing the former would mean unpredictable real-time operation due to timing latency irregularities, while the latter would provide a more meaningful result with 2,048 MRSAC operations for each cochlear section that can be processed within 2,604 clock cycles.

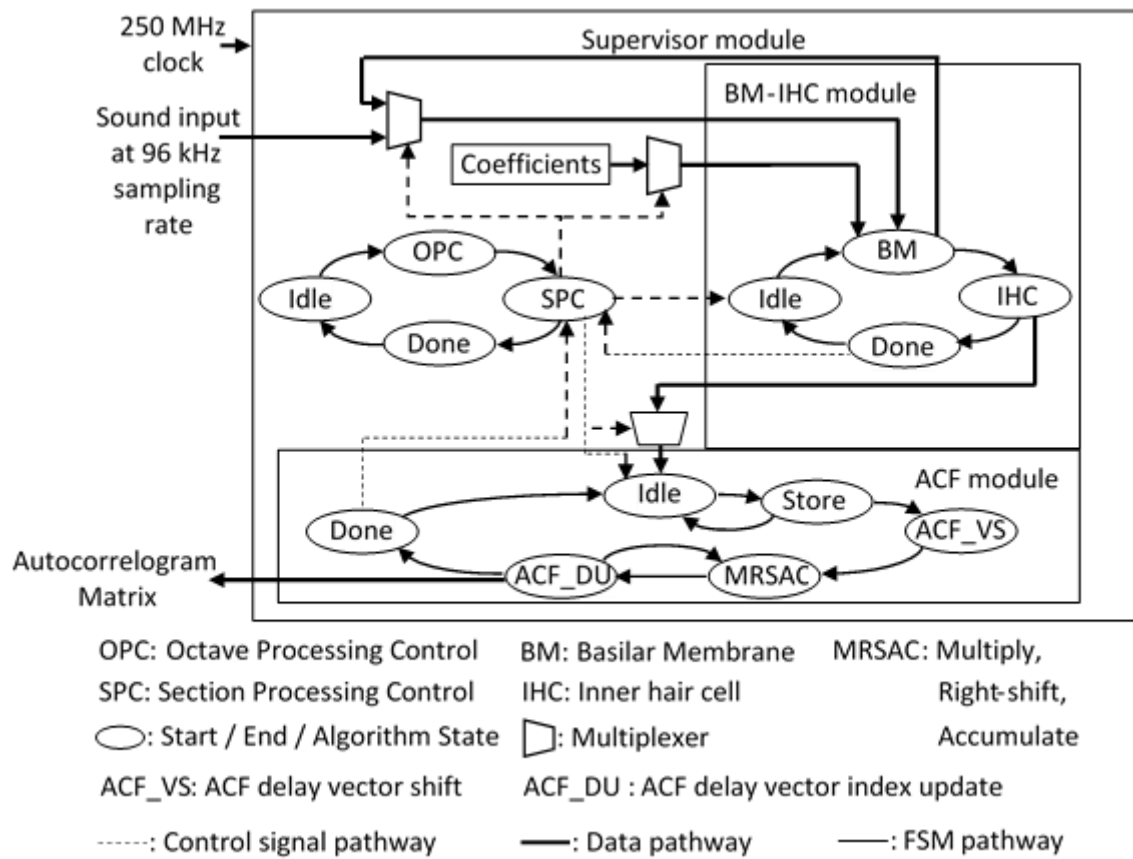


Figure 4-4: Architecture of the CAR-Lite-ACF model with fixed-point arithmetic implemented on an FPGA with MAC operations on inner hair cell (IHC) input signals. FSM: Finite state machine.

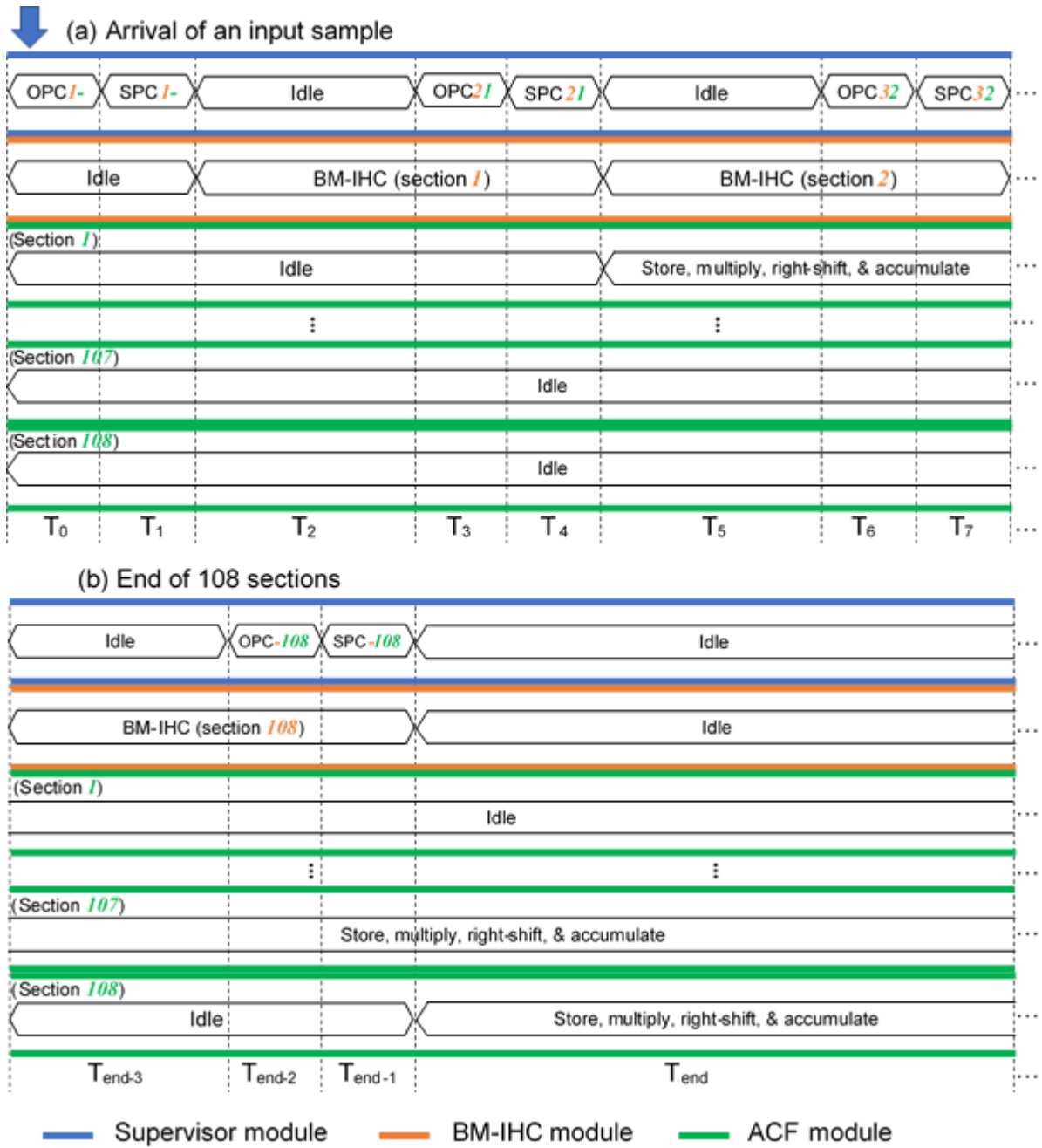


Figure 4-5: FPGA output vector waveforms of the supervisor, BM-IHC, and ACF modules operating at (a) the arrival of an audio sample; (b) the activation of each of 108 parallel ACF module at the arrival of an IHC sample from a specific cochlear section corresponding to the audio sample e.g. when the BM-IHC outputs an IHC sample for cochlear section 1, the dedicated ACF module for cochlear section 1 is activated; the same operation applies for the other 107 cochlear sections.

To generate a 108-section-by-2,048-sample autocorrelogram, up to 221,184 samples should be available before processing commences. However, waiting on 221,184 samples to arrive before commencing processing would mean that the digital signal processors (DSPs) assigned to calculate the autocorrelogram would be inactive until all the input samples are acquired. A more efficient approach is to distribute the computation evenly when an input sample arrives instead of waiting for a block of samples. With the block-based operation, even after the entire 221,184 input samples are received, computational resources have to be allocated either by the increase in FPGA system clock rate or a higher than usual number

of parallel DSPs to ensure that the autocorrelogram is available at the shortest possible time. Otherwise, a noticeable lag is observable when a sound signal is played and when its corresponding autocorrelogram is projected. Processing one sample at a time means that this lag is minimised to become unnoticeable [13].

The ACF module utilises a five-state finite state machine (FSM) to perform per sample computation comprising *Idle*, *ACF_VS*, *MRSAC*, *ACF_DU*, and *Done*. At the *Idle* state, when an input sample arrives, it is stored in x_n , where n signifies the cochlear section number. The next state is the vector shift (*ACF_VS*) state, where the 2,048 samples in the lag window vector are shifted down. Hence, all the values in 2,048 memory locations move one address downward, i.e. the oldest sample at memory address 2,047 is displaced by the second oldest value from memory address 2,046; input sample at address 2,045 moves to address 2,046, etc. The newly acquired input sample is stored at memory address 0. Following the vector shift is the *MRSAC* state, where the multiply, right-shift, and accumulate operations are nested to allow DSP to calculate all three operations within a single clock cycle. Here, multiplication is done for the input sample, $x(t)$, and $x(t)$ delayed by d samples from the lag window vector, $x_{-}(d)$. Two parallel *MRSAC* operations are implemented at this state to ensure that the latency of calculating 2,048 samples in the delay vector remains below 2,604 clock cycles (10.41 μ s). Following a set of *MRSAC* operation, d is incremented to point to the next delayed sample in the lag window in the delay-index update (*ACF_DU*) state. After this, the FSM enters the *MRSAC* state again to calculate the next set of lag calculation. When the *MRSAC* state processes all 2,048 of the delay samples, the FSM concludes by traversing to the *Done* state, where an octave-based counter is incremented.

As the cochlear octaves operate at different sampling rates, some input sound samples are dropped for cochlear octaves beyond octave 1 in the BM-IHC module. This characteristic means that the old input samples to the ACF modules for these octaves are retained when the input sound sample at their corresponding cochlear section is bypassed. An octave-based counter is used to track this retention of input samples to the ACF module in the *Store* state. After a lag window of 2,048 samples has been processed, the octave-based counter is incremented and checked against an octave-based threshold. A lower count than the octave-based threshold means that there are no input samples to be expected, and the same input sample is used to calculate the next ACF output vector. From the *Store* state, the FSM progresses to the *ACF_VS* state, where the contents of the lag window vector are shifted downward, and the topmost element in the vector is updated by the same input sample that has been retained. For cochlear sections in the first octave, the input samples are the new ones passed down by the BM-IHC module via the supervisor module.

Figure 4-6 illustrates an example of the ACF calculation for an eight-sample delay window vector across three cochlear sections: 0, 12, and 24 adhering to equation (4-1). These three cochlear sections are the first section from octaves 1 to 3. Hence, their sampling rates are 1, $\frac{1}{2}$, and $\frac{1}{4}$ of the sampling rate of octave 1. In other words, the new input samples for octaves 2 and 3 are available at the third, and fifth clock cycles of the sampling rate of octave 1. This characteristic is indicated by the delay window vector represented by the coloured columns in groups of 1, 2, and 4 that are labelled as x_a , x_b , and x_c , respectively in Figure 4-6. Every input sample received is stored at the top of delay window vectors, i.e. xa_0 , xb_0 , and xc_0 corresponding to inputs $x0$, $x1$, and $x2$. But before this takes place, the contents of the vectors are shifted downward by one sample. The contents of the input sample x_n is multiplied with every element of the delay vector (indicated

by the blue horizontal lines), and the corresponding results are accumulated over time (indicated by the red vertical lines) to project the autocorrelation output vector, y . Note that a 16 bit right-shift operation is implemented immediately after a multiplication operation to maintain an output bit width of 16 bits.

The SystemVerilog simulation of the example in Figure 4-6 is shown in Figure 4-7. The input samples x_0 , x_{12} , and x_{24} are shown along with the eight-column outputs of y_{0_0} , y_{12_0} , and y_{24_0} that are colour-coded identically to the columns of Figure 4-6. The ACF module is instantiated three times, which indicates parallel processing to generate the three output vectors simultaneously. The results of the parallel processing is observable from 20 ns to 140 ns, where the output samples of y_{0_0} , y_{12_0} , and y_{24_0} become available simultaneously.

(a)	t								$\sum(x0(t) \cdot xa(t,d) / 2^{16}) = \underline{y0}$									
	$x0$	256	257	258	259	260	261	262	263									
	xa_0	256	257	258	259	260	261	262	263	1	2	3	4	5	6	7	8	$y0_0$
	xa_1	0	256	257	258	259	260	261	262	0	1	2	3	4	5	6	7	$y0_1$
	xa_2	0	0	256	257	258	259	260	261	0	0	1	2	3	4	5	6	$y0_2$
	xa_3	0	0	0	256	257	258	259	260	0	0	0	1	2	3	4	5	$y0_3$
	xa_4	0	0	0	0	256	257	258	259	0	0	0	0	1	2	3	4	$y0_4$
	xa_5	0	0	0	0	0	256	257	258	0	0	0	0	0	1	2	3	$y0_5$
	xa_6	0	0	0	0	0	0	256	257	0	0	0	0	0	0	1	2	$y0_6$
	xa_7	0	0	0	0	0	0	0	256	0	0	0	0	0	0	0	1	$y0_7$
- Multiply: $(x0(t) \cdot xa(t,d) / 2^{16})$; Accumulate: $\sum (x0(t) \cdot xa(t,d) / 2^{16})$.																		
(b)	t								$\sum(x12(t) \cdot xb(t,d) / 2^{16}) = \underline{y12}$									
	$x12$	512	512	513	513	514	514	515	515									
	xb_0	512	512	513	513	514	514	515	515	4	8	12	16	20	24	28	32	$y12_0$
	xb_1	0	512	512	513	513	514	514	515	0	4	8	12	16	20	24	28	$y12_1$
	xb_2	0	0	512	512	513	513	514	514	0	0	4	8	12	16	20	24	$y12_2$
	xb_3	0	0	0	512	512	513	513	514	0	0	0	4	8	12	16	20	$y12_3$
	xb_4	0	0	0	0	512	512	513	513	0	0	0	0	4	8	12	16	$y12_4$
	xb_5	0	0	0	0	0	512	512	513	0	0	0	0	0	4	8	12	$y12_5$
	xb_6	0	0	0	0	0	0	512	512	0	0	0	0	0	0	4	8	$y12_6$
	xb_7	0	0	0	0	0	0	0	512	0	0	0	0	0	0	0	4	$y12_7$
- Multiply: $(x12(t) \cdot xb(t,d) / 2^{16})$; Accumulate: $\sum (x12(t) \cdot xb(t,d) / 2^{16})$.																		
(c)	t								$\sum(x24(t) \cdot xc(t,d) / 2^{16}) = \underline{y24}$									
	$x24$	1024	1024	1024	1024	1025	1025	1025	1025									
	xc_0	1024	1024	1024	1024	1025	1025	1025	1025	16	32	48	64	80	96	112	128	$y24_0$
	xc_1	0	1024	1024	1024	1024	1025	1025	1025	0	16	32	48	64	80	96	112	$y24_1$
	xc_2	0	0	1024	1024	1024	1024	1025	1025	0	0	16	32	48	64	80	96	$y24_2$
	xc_3	0	0	0	1024	1024	1024	1024	1025	0	0	0	16	32	48	64	80	$y24_3$
	xc_4	0	0	0	0	1024	1024	1024	1024	0	0	0	0	16	32	48	64	$y24_4$
	xc_5	0	0	0	0	0	1024	1024	1024	0	0	0	0	0	16	32	48	$y24_5$
	xc_6	0	0	0	0	0	0	1024	1024	0	0	0	0	0	0	16	32	$y24_6$
	xc_7	0	0	0	0	0	0	0	1024	0	0	0	0	0	0	0	16	$y24_7$
- Multiply: $(x24(t) \cdot xc(t,d) / 2^{16})$; Accumulate: $\sum (x24(t) \cdot xc(t,d) / 2^{16})$.																		

Figure 4-6: Simulation of a MAC-based ACF algorithm with each table showing ACF results of an input signal, x_n , where n represents cochlear sections (a) 0, (b) 12, and (c) 24. xa_d , xb_d , and xc_d are the block memory allocated based on the ACF window lag size, d (set as 8 in this demonstration but as 2,048 in CAR-Lite-ACF). yn_d is the accumulated output at various discrete-time and for various lag sizes.

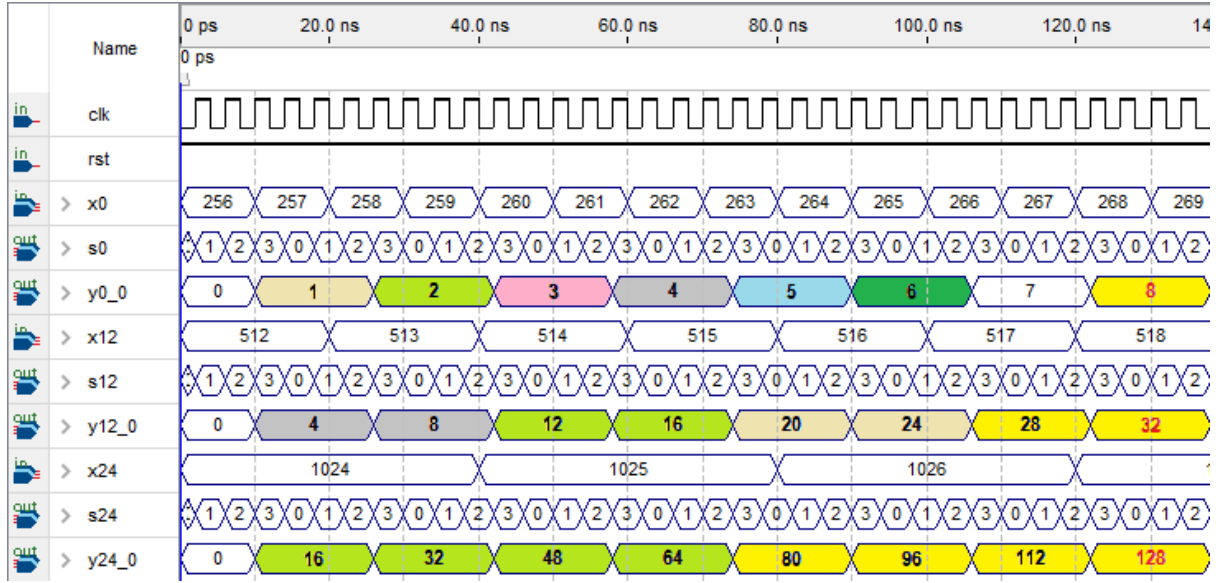


Figure 4-7: SystemVerilog simulation of a MAC-based multi-octave autocorrelation function (ACF) corresponding to the example illustrated in Figure 4-6 (row ending numbers in red font).

4.4.1.2. CAR-Lite-ACF model using AAC operations

The architecture of the CAR-Lite-ACF model implemented with AND-accumulate (AAC) operations is illustrated in Figure 4-8. It is similar to the MAC-based CAR-Lite-ACF model described in section 4.4.1.1 and has three modules: supervisor, BM-IHC-AN, and ACF. However, in this case, the ACF module has auditory nerve (AN) signals as its inputs that are generated in the BM-IHC-AN module with leaky-integrate-and-fire (LIF) neurons. The LIF neuron algorithm is the new addition to the BM-IHC module from the MAC implementation, and thus, its title for the AAC implementation is updated to reflect this: BM-IHC-AN.

The LIF neuron algorithm is similar to its implementation in the CAR-Lite-SI model in chapter 3 with three differences. Firstly, the refractory period is excluded from the LIF neuron spiking as it degrades the degree of similarity between AN spike and IHC signals, as demonstrated in subsection 3.3.7. Secondly, the multiple sampling rates from the CAR-Lite model is also maintained in the LIF neurons across all octaves, unlike the single sampling rate of 3 kHz in the CAR-Lite-SI model. This allows a large range of fundamental frequencies to be representable in an autocorrelogram, especially with regards to the musical notes used in chapter 6. Thirdly, only a single auditory nerve (AN) fibre firing threshold is implemented as a 16-bit value, as shown in Table 4-2, instead of three-fibre implementation as described by the CAR-Lite-SI model in chapter 3. The bit width of each AN signal corresponding to a cochlear section is 1-bit. So, for all 108 cochlear sections, only 108-bits are required as opposed to 1,728-bits (16-bits \times 108 cochlear sections) for the MAC implementation.

The supervisor module acquires AN signal from a cochlear section originating from the BM-IHC-AN module before invoking the ACF module for the specific cochlear section, as seen in Figure 4-9(b). For cochlear sections that are not processed due to the different sampling rates across the cochlear octaves, previously latched BM, BMd, IHC and AN data samples are reused.

The BM-IHC-AN module has latencies of 84 ns for 21 states corresponding to one cochlear section invocation and 9.07 μ s for 108 invocations corresponding to 108 cochlear

sections. Here the multiplication and right-shift operation in the ACF equation is replaced with the logical-AND operation. The latency of the ACF module is identical to the MAC-based operation described in section 4.4.1.1, which is at $8.21 \mu\text{s}$ ($\{2 \text{ iterative states} \times 2,048 \text{ delay samples} / 2 \text{ parallel AAC operations} + 4 \text{ non-iterative states}\} \times 1/250 \text{ MHz}$) for calculating 2,048 samples in the delay vector per cochlear section. The latencies for this model are below the safe duration threshold of $10.41 \mu\text{s}$ – the duration between the arrivals of two successive audio samples.

An example of the operation of the AAC-based ACF module is shown in Figure 4-10. Here, only logical-AND and accumulate operations are used instead of MRSAC operations based on equation (4-2). The colour-coded rows in Figure 4-10 are simulated in SystemVerilog, and their resultant FPGA vector waveforms are shown in Figure 4-11. In the latter, the colour-coded output vectors for the three rows are calculated in parallel by three separately invoked ACF modules. For demonstration purposes, the three input streams to the three parallel ACF modules have periods of 40 ns, 80 ns, and 160 ns corresponding to the reduction of sampling rates by a factor of 2 for the first three cochlear octaves.

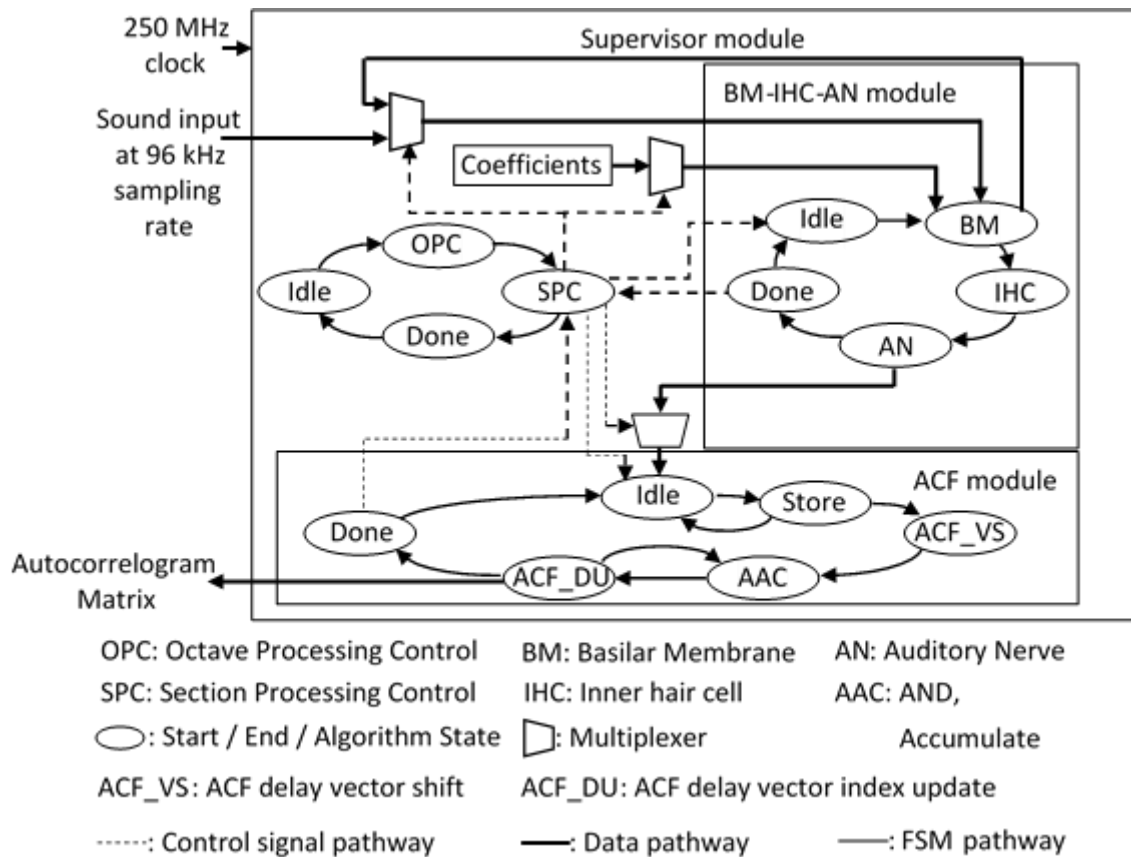


Figure 4-8: Architecture of the CAR-Lite-ACF model (fixed-point arithmetic) implemented on FPGA with AAC operations on the auditory nerve (AN) input signals. FSM: Finite state machine.

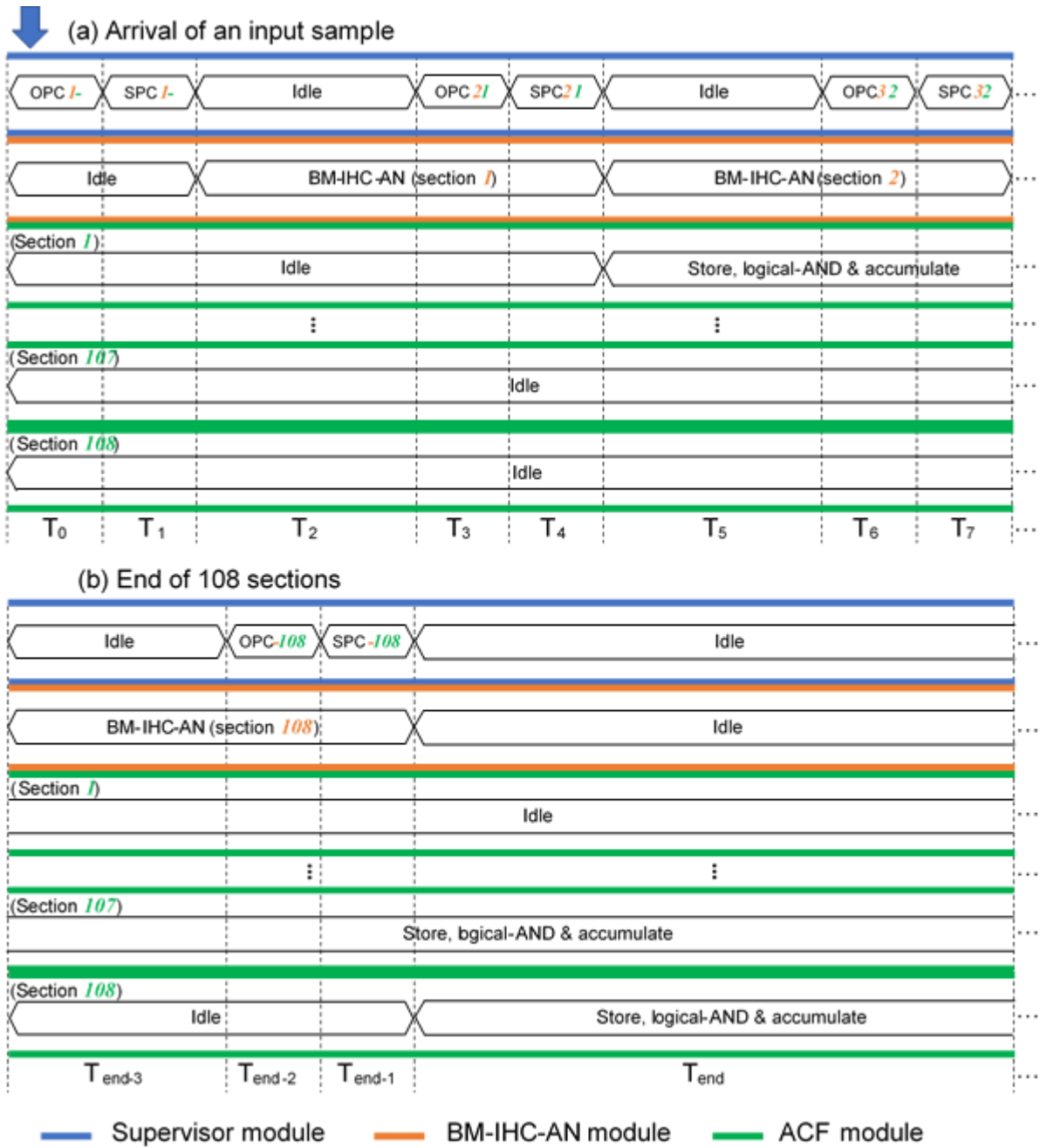


Figure 4-9: FPGA output vector waveforms of the supervisor, BM-IHC-AN, and ACF modules operating at (a) the arrival of an audio sample; (b) the activation of each of 108 parallel ACF modules at the arrival of one AN bit from a specific cochlear section corresponding to the audio sample e.g. when the BM-IHC-AN outputs one AN bit for cochlear section 1, the dedicated ACF module for cochlear section 1 is activated; the same operation applies for the other 107 cochlear sections.

(a)	t								$\sum(x0(t) \wedge xa(t,d)) = y0$								
$x0$	1	0	1	0	1	0	1	0									
xa_0	1	0	1	0	1	0	1	0	1	1	2	2	3	3	4	4	$y0_0$
xa_1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	$y0_1$
xa_2	0	0	1	0	1	0	1	0	0	0	1	1	2	2	3	3	$y0_2$
xa_3	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	$y0_3$
xa_4	0	0	0	0	1	0	1	0	0	0	0	0	1	1	2	2	$y0_4$
xa_5	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	$y0_5$
xa_6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	$y0_6$
xa_7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	$y0_7$
— Logical-AND: $(x0(t) \wedge xa(t,d))$; Accumulate: $\sum (x0(t) \wedge xa(t,d))$.																	
(b)	t								$\sum(x12(t) \wedge xb(t,d)) = y12$								
$x12$	1	1	0	0	1	1	0	0									
xb_0	1	1	0	0	1	1	0	0	1	2	2	2	3	4	4	4	$y12_0$
xb_1	0	1	1	0	0	1	1	0	0	1	1	1	1	2	2	2	$y12_1$
xb_2	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	$y12_2$
xb_3	0	0	0	1	1	0	0	1	0	0	0	0	1	1	1	1	$y12_3$
xb_4	0	0	0	0	1	1	0	0	0	0	0	0	1	2	2	2	$y12_4$
xb_5	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	$y12_5$
xb_6	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	$y12_6$
xb_7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	$y12_7$
— Logical-AND: $(x12(t) \wedge xb(t,d))$; Accumulate: $\sum (x12(t) \wedge xb(t,d))$.																	
(c)	t								$\sum(x24(t) \wedge xc(t,d)) = y24$								
$x24$	1	1	1	1	0	0	0	0									
xc_0	1	1	1	1	0	0	0	0	1	2	3	4	4	4	4	4	$y24_0$
xc_1	0	1	1	1	1	0	0	0	0	1	2	3	3	3	3	3	$y24_1$
xc_2	0	0	1	1	1	1	0	0	0	0	1	2	2	2	2	2	$y24_2$
xc_3	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	$y24_3$
xc_4	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	$y24_4$
xc_5	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	$y24_5$
xc_6	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	$y24_6$
xc_7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	$y24_7$
— Logical-AND: $(x24(t) \wedge xc(t,d))$; Accumulate: $\sum (x24(t) \wedge xc(t,d))$.																	

Figure 4-10: Simulation of an AAC-based ACF algorithm with each table showing ACF results of an input signal, xn , where n represents cochlear sections (a) 0, (b) 12, and (c) 24. xa_d , xb_d , and xc_d are the block memory allocated based on the ACF window lag size, d (set as 8 in this demonstration but as 2,048 in CAR-Lite-ACF). yn_d is the accumulated output at various discrete-time and for various lag sizes.

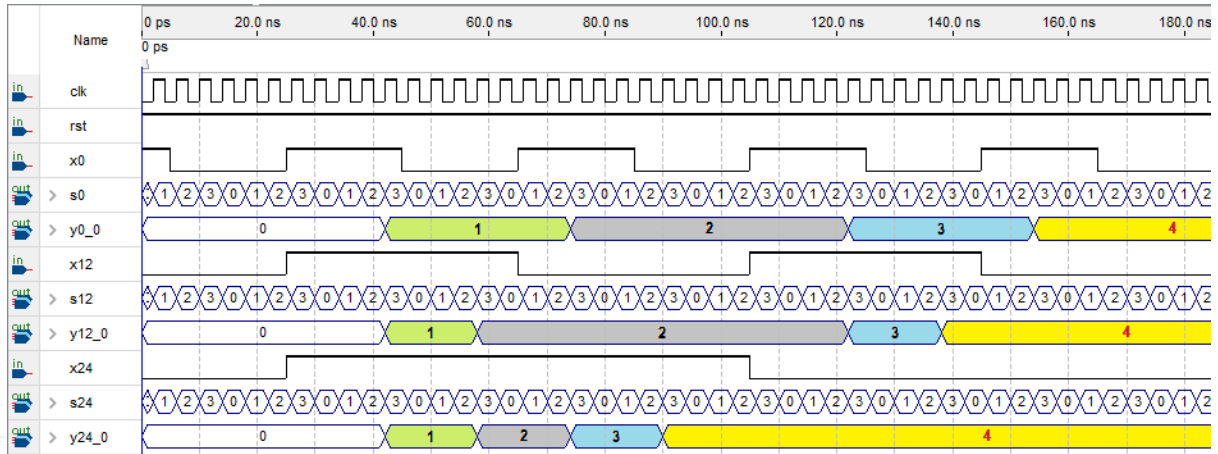


Figure 4-11: SystemVerilog simulation of the AAC-based multi-octave autocorrelation function (ACF) at 0 sample delay based on the example illustrated in Figure 4-10 (row ending numbers in red font).

4.4.1.3. Results

The results of the CAR-Lite-ACF model implemented on an Altera Cyclone V FPGA is shown in Table 4-3, which are extracted from the Altera Quartus compiler reports and the Altera PowerPlay reports for power measurements. The MAC-based implementation uses eight times more digital signal processors (DSPs) than the AAC-based implementation. The power consumption is also higher for the MAC-based implementation than the AAC-based implementation by 21 mW. However, the number of adaptive logic modules (ALM) and registers utilised by the AAC-based implementation is almost on par with the MAC-based implementation, though, it is slightly higher for the MAC-based implementation.

Each set of multiply, right-shift, and accumulate (MRSAC) and AAC operations in CAR-Lite-ACF is characterised in a nested equation housed within a single state of the finite state machine (FSM) in the ACF module. This implementation is due to the computations required based on the timing constraints of the arrivals of audio samples, i.e. 221,184 (108 cochlear sections \times 2,048 lag samples) calculations per input audio sample is required for CAR-Lite-ACF. The Quartus SystemVerilog compiler compensates the optimisation of DSP and logic utilisation for such nested equations by introducing more logic circuitry and registers.

In summary, for the CAR-Lite-ACF model, the AAC-based implementation uses less computational resources and consumes less power than the MAC-based implementation.

Model	Number of ALM Utilised (out of 29,080)	Number of Registers Utilised	Number of DSPs Utilised (out of 150)	Power (mW)
CAR-Lite-ACF (section 4.4.1.1: MAC-based)	1,392 (4.8%)	2,556	60 (40%)	265
CAR-Lite-ACF (section 4.4.1.2: AAC-based)	1,303 (4.5 %)	2,429	7 (4.7 %)	244

Table 4-3: Computational resources used on an Altera Cyclone V FPGA to implement the CAR-Lite-ACF model.

4.4.2. Response to Complex Tones

The complex tone comprises four sinusoidal tones at 100 Hz, 200 Hz, 500 Hz, and 1.5 kHz. Figure 4-12 displays the autocorrelogram matrices generated in floating-point and

fixed-point arithmetic using MAC and AAC operations. Placed below and to the right of each autocorrelogram in Figure 4-12 is a temporal profile and a spectral profile, respectively. The temporal profile waveform, y_{temp} , and spectral profile waveform, y_{spec} , are calculated as follow:

$$y_{temp} = \frac{1}{\max_s \sum_{s=1}^S y_{AC}(s, d)} \sum_{s=1}^S y_{AC}(s, d) \quad (4-5)$$

$$y_{spec} = \frac{1}{\max_d \sum_{d=1}^D y_{AC}(s, d)} \sum_{d=1}^D y_{AC}(s, d) \quad (4-6)$$

where y_{AC} is the 2D autocorrelogram; s is a cochlear section index; d is the lag sample number; S is the number of cochlear sections in the CAR-Lite cochlear model at 108; D is the maximum lag or delay at 2,048 samples. In other words, the temporal profile is the summation of rows of the autocorrelogram, and the spectral profile is the summation of the columns of the autocorrelogram, which are then normalised with the largest numbers in their respective summed vectors.

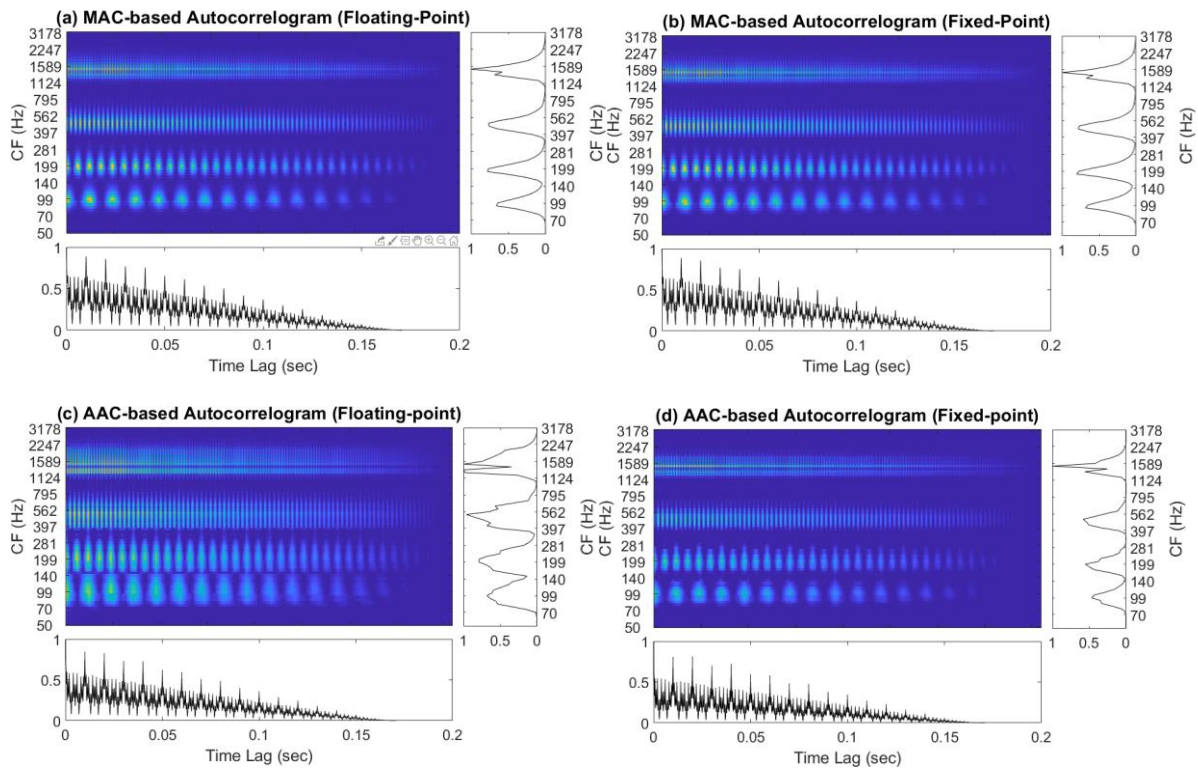


Figure 4-12: Autocorrelogram matrices (with blue background), temporal profiles (below an autocorrelogram), and spectral profile (right of an autocorrelogram) generated with (a) floating-point MAC operations, (b) fixed-point MAC operations, (c) floating-point AAC operations, and (d) fixed-point AAC operations. Note the abbreviation – CF: Centre frequency of a cochlear section.

In each of the four autocorrelograms in Figure 4-12, the four sinusoids of the complex tone are seen clearly as four light-blue coloured horizontal stripes. These stripes are also aligned with the four peaks in the spectral profile, which correspond to the frequencies of the four tones. However, this attribute is not apparent in the temporal profile. Figure 4-13 illustrates the temporal profile waveforms from Figure 4-12(a) and Figure 4-12(d) magnified

in the arbitrary region ranging from 29 ms to 41 ms. The duration between peaks provides information on the frequency components residing in the signal, which is calculated as follows:

$$f_i = 1 / ((p_j - p_{j-1}) / f_s) \quad (4-7)$$

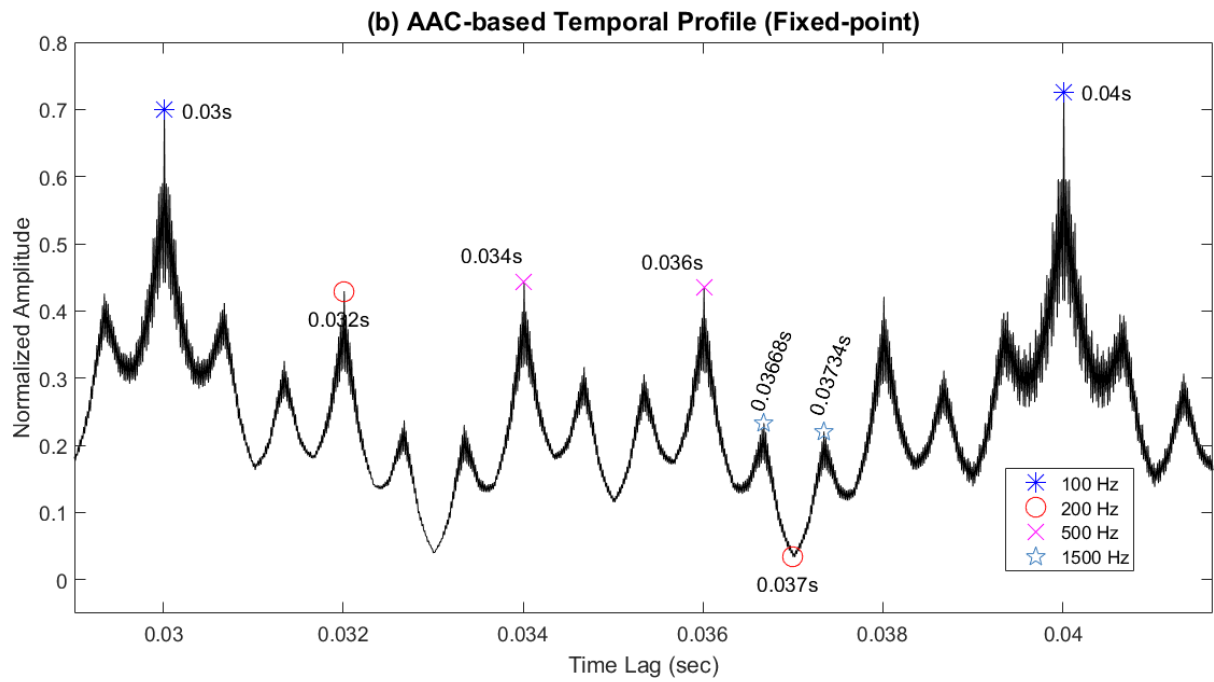
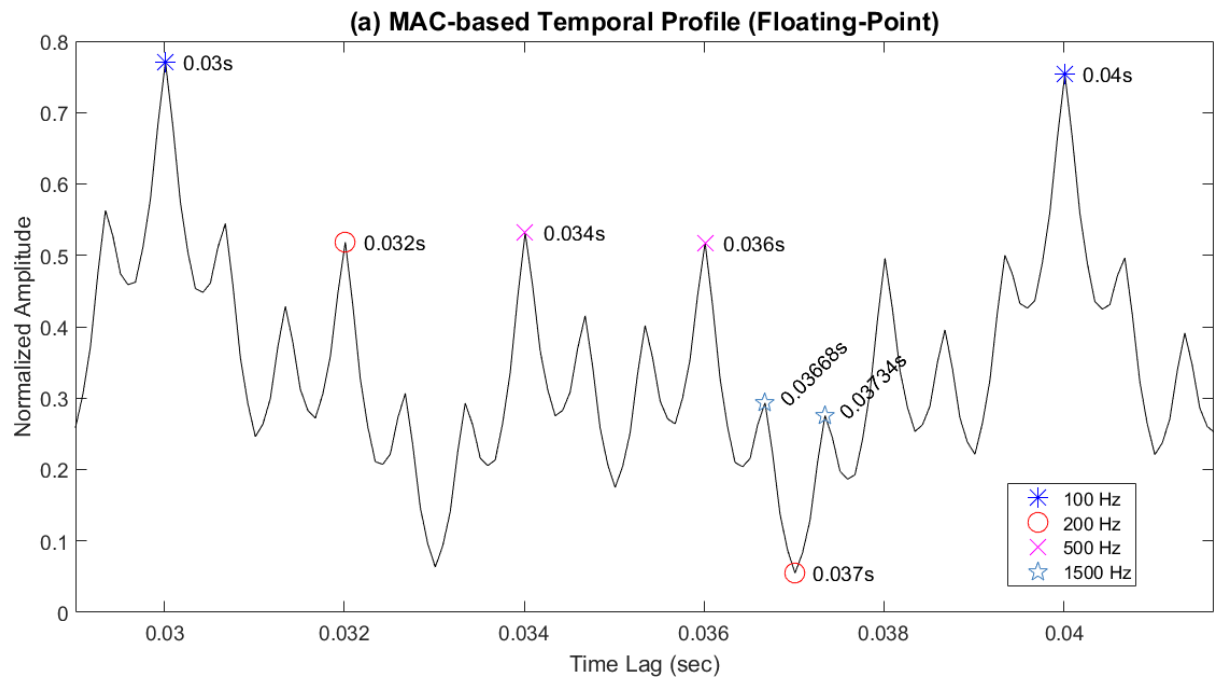
where the i th frequency component is calculated using the distance in samples between two peaks, p_j , and p_{j-1} over a sampling rate, f_s , of 96 kHz. In Figure 4-13, the lag on the x-axis is presented as time instead of samples. With this arrangement, equation (4-7) can still be used to calculate the frequency by setting the sampling rate, f_s , to 1 and replacing the sample numbers, p_j and p_{j-1} with timestamp information. The four frequency components are extractable, as is evident in Figure 4-13 using this technique together with manually marked timestamp information.

The temporal profile generated using AAC operations contains ‘noise’, which is observable in Figure 4-13(b) that is generated by the fixed-point arithmetic. This effect is due to the depth of the bit width reduction of the output signal from the CAR-Lite cochlear model to the ACF algorithm. The floating-point MAC-based signals are at least 64 bits wide and thus, have a smooth transition between two successive samples. In contrast, the 32 bits fixed-point AAC-based signals are generated from 1 bit AN binary spike-streams. Consequently, the streams of pulses continue to exist in the autocorrelogram and become apparent in the temporal profile. Here, they appear as ‘noise’ but have a resonance of 50 kHz following the pulse output of a LIF neuron. Hence, this ‘noise’ is regarded as a carrier signal, and it is the lower frequency modulated signal, which is of interest that carries the pitch information. In other words, pitch information is the demodulated signal output from the CAR-Lite-ACF model.

Two passes of a first-order infinite impulse response (IIR) low-pass filter (LPF) is applied to the signal with the following filter characteristic to suppress the high frequency ‘noise’:

$$y_{LPF}(d) = (1 - c_{LPF}) \cdot y_{LPF}(d - 1) + c_{LPF} \cdot y_{temp}(d) \quad (4-8)$$

where c_{LPF} is the coefficient set empirically at 0.2. The filtering effect results in a smoothed temporal profile signal, as displayed in Figure 4-13(c). Despite the phase shifts of the peaks by 60 μ s, the time duration between peaks remains the same, which results in the frequency characteristics of the four tones displayed as clearly as the MAC-generated temporal profile in Figure 4-13(a).



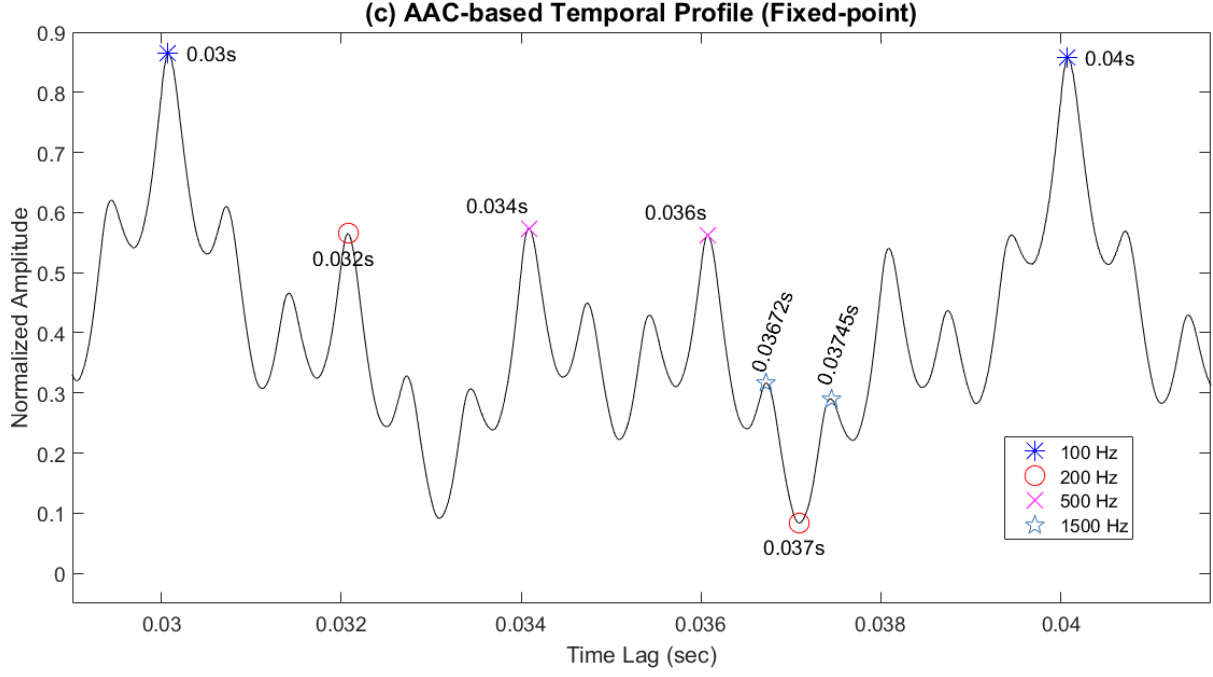


Figure 4-13: Magnified temporal profiles from Figure 4-12 between time lags of 29 ms and 41 ms generated from (a) floating-point MAC operations, (b) fixed-point AAC operations, and (c) a low-pass filtered version of the waveform in (b).

The degrees of similarity between the four autocorrelograms in Figure 4-12 and their corresponding temporal and spectral profiles are shown in Figure 4-14. A 2D correlation coefficient is used to quantify the degree of similarity, which is defined by:

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \quad (4-9)$$

where r is a correlation coefficient; m is the row number; n is the column number; A and B are the two 2D matrices to be compared; \bar{A} and \bar{B} are the mean of the 2D matrices.

The MAC-based floating-point and fixed-point generated autocorrelograms, as well as their two respective profiles, have the highest similarity with 2D correlation coefficient scores of 1. For the other autocorrelograms, the CCs range between 0.82 and 0.97 with a standard deviation of 0.083. The overall range of CC scores is higher for the temporal profiles than the autocorrelogram CC scores, ranging between 0.9 and 1, and even higher for the spectral profiles in the range of 0.97 and 1. The standard deviation of CC scores for the temporal profile is 0.045, and for the spectral profile, it is 0.014. Note that the temporal profiles used for comparison here are generated using AAC operations that have undergone two passes of low-pass filtering, as mentioned in the preceding paragraph. An essential comparison is between the floating-point MAC-based and fixed-point AAC-based autocorrelograms as the latter is used for FPGA implementation and a score of 0.97 shows that there is very little difference between the two. Also note that all the AAC-based autocorrelograms in Figure 4-14 use the LIF neuron firing threshold of 0.27 V extracted from Figure 4-3, which is based on the highest CC of 0.97 (under MAC_{flt} vs. AAC_{fix}).

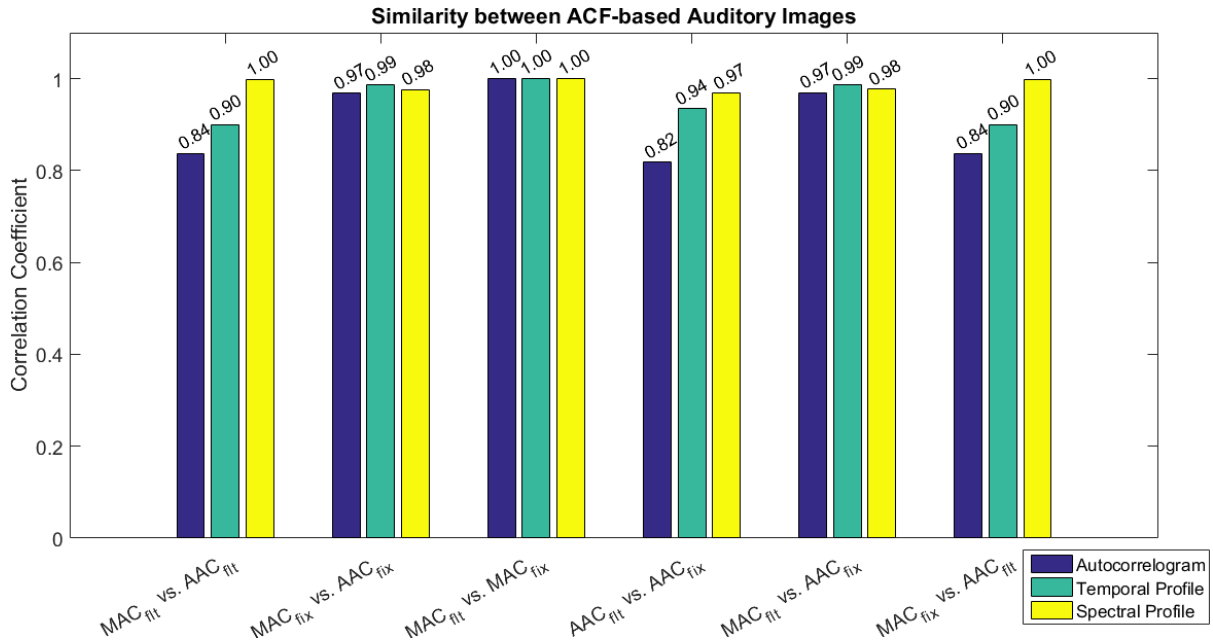


Figure 4-14: Degrees of similarity between the four autocorrelograms and their corresponding temporal and spectral profiles from Figure 4-12 using the 2D correlation coefficient (CC). Note the following abbreviation – fit: floating-point; fix: fixed-point.

4.4.3. Response to Missing Fundamental Frequency

An input signal containing 800 Hz, 1 kHz, and 1.2 kHz harmonics without its fundamental frequency, f_0 , of 200 Hz is used to test whether the missing f_0 information is acquirable from an autocorrelogram. Figure 4-15(a) displays the autocorrelogram representing the harmonic signal. The region below 800 Hz in the autocorrelogram depicts missing low pitch content in the input signal. This attribute indicates that the ratio of the distribution of pitch-related inter-spike intervals (ISI) across all cochlear sections are more dominant than non-pitch-related ISI [7].

Figure 4-15(b) displays a low-pass filtered temporal profile extracted from the autocorrelogram. Here, the inverse of the interval of the two peaks marked with blue dots results in 204 Hz, which is a close approximate to the missing 200 Hz f_0 in the harmonic signal. However, using the two highest peaks with the blue dot marked as “t0” and the red dot yields a wrong fundamental frequency. Hence, a real-world sound signal comprising harmonic and inharmonic contents does not yield an accurate fundamental frequency using the two highest peaks for calculation. This assessment is also reinforced by the low classification scores in chapter 6, using algorithm 1 using the two highest peaks. Conversely, a more sophisticated algorithm is capable of calculating the fundamental frequency from the temporal profile in Figure 4-15(b), as is also demonstrated in chapter 6. Nonetheless, with an appropriate pitch estimation algorithm, this exercise shows that a missing f_0 can be calculated with fixed-point arithmetic on hardware with AAC operations, which in turn, indicates low computational cost in generating an autocorrelogram.

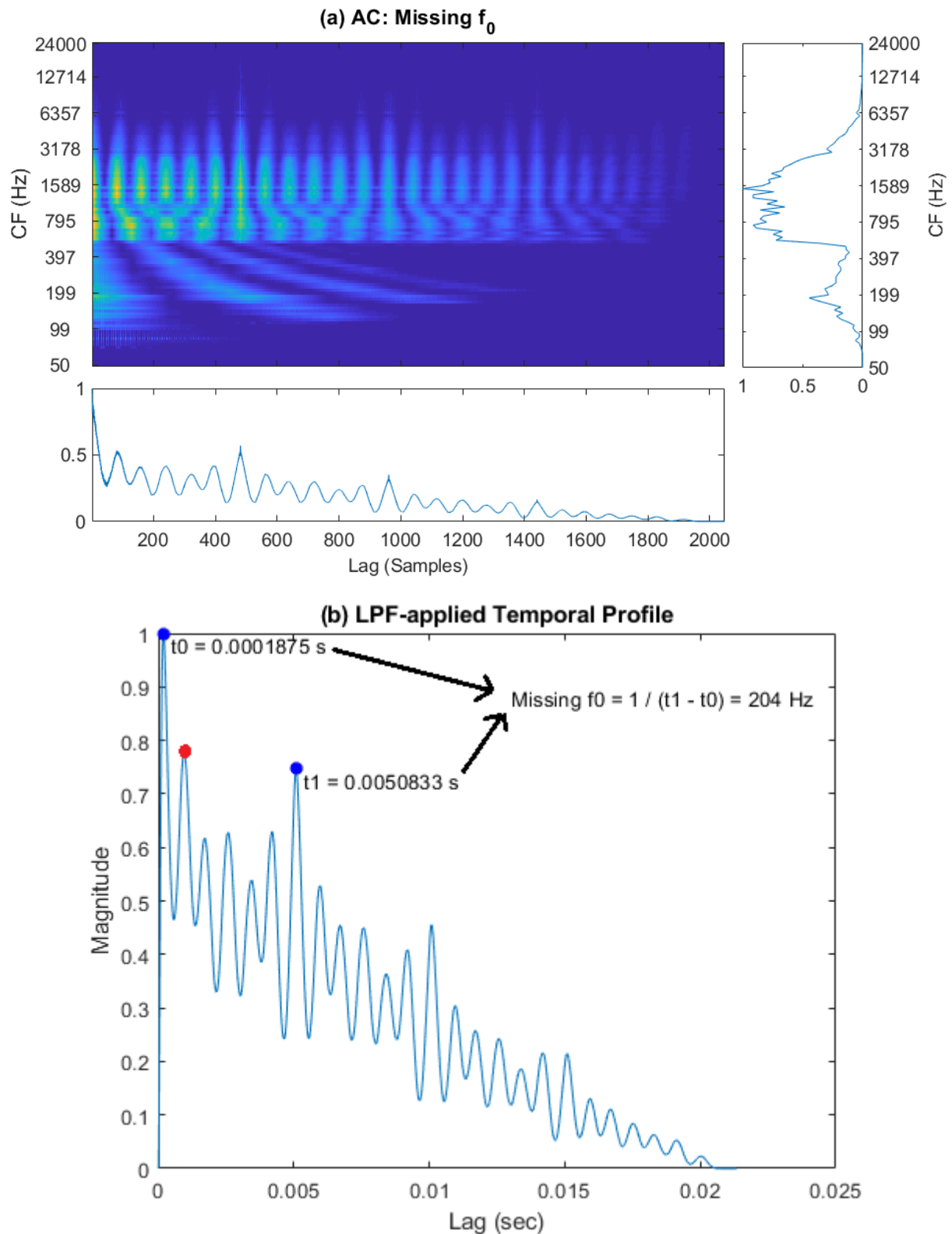


Figure 4-15: Harmonic signal comprising 800 Hz, 1 kHz, and 1.2 kHz without a fundamental frequency, f_0 , of 200 Hz represented on (a) an autocorrelogram calculated with fixed-point arithmetic as well as AAC operations and (b) a low-pass filtered temporal profile extracted from the autocorrelogram, displaying an approximate of the missing f_0 calculated from the two blue-dotted peaks in the temporal profile. Blue dots represent peaks used for calculating the missing fundamental frequency. The red dot represents a false positive peak that causes a wrong fundamental frequency calculated with respect to the first blue dot (t_0).

4.4.4. Response to Harmonics Phase Change

Chapter 2 introduces how pitch perceived is not affected by the phase change of the harmonics of a stimulus. Carlyon and Shackleton further articulated this phase insensitivity phenomenon on pitch perception over existing evidence. Using harmonics at alternating phase, they found that the corresponding pitch perceived shifted up by an octave [25]. For the alternating phase, each harmonic signal is generated alternately using either sine or cosine functions, as opposed to the sine phase signal, whose harmonics are generated only by sine functions. This pitch doubling effect is observable by the temporal envelope of the stimuli in Figure 4-16. The normalised summation of the first ten harmonics at 0° phase of a 100 Hz fundamental frequency (f_0) is shown in Figure 4-16(a). Here, the peak-to-peak (only high peaks) periodicity amounts to 100 Hz. When the phases are alternated, the peak-to-peak periodicity is halved.

Stimuli are generated at three discrete harmonic levels (LOW, MID, and HIGH), as illustrated in Table 4-4 to show the phase change effects on the temporal profile of an autocorrelogram (AC). Two stimuli ($f_0 = 150$ Hz and $f_0 = 300$ Hz) are generated in sine phase at the three harmonic levels. A third stimulus ($f_0 = 150$ Hz) is generated in an alternating phase at the three harmonic levels. Figure 4-17 illustrates the temporal profiles of these stimuli using hardware-based AAC operations and fixed-point arithmetic. The temporal profiles are conditioned by two passes of the low-pass filter defined by equation (4-8) with a cut-off at 400 Hz to remove high-frequency artefacts.

The peak-to-peak (high peaks only) periodicity of the profiles in Figure 4-17(a), (b), and (c) correspond to the stimuli f_0 of 150 Hz and for Figure 4-17(g), (h), and (i), they correspond to the stimuli f_0 of 300 Hz. The peak-to-peak periodicity remains at 150 Hz when the phases are alternated for the LOW and MID range harmonics, as shown in Figure 4-17(d), and (e). However, when HIGH range harmonics are introduced to the stimulus with an f_0 of 150 Hz, a new intermediate-level peak appears between two successive high peaks. Thus, the temporal profile is more similar to the 300 Hz profiles shown in Figure 4-17(g), (h), and (i) than the 150 Hz in Figure 4-17(a), (b), and (c). This characteristic leads to pitch being perceived at twice the f_0 , at 300 Hz, as was also observed by Carlyon and Shackleton [25] as well as Meddis [14].

The reason behind this behaviour is attributable to the theory of resolvability, as explained in chapter 2. LOW harmonics are known to be resolved in that each cochlear filter at this range, can output a pure tone due to its low bandwidth. Changing the phase of a pure tone does not affect its temporal envelope shape, and hence, its ACF is unaffected. In contrast, HIGH harmonics are unresolved as they are output by a high-frequency cochlear filter as a complex tone. This characteristic is due to the large bandwidth of each cochlear section at this range, resulting in an output signal that is a combination of pure tones. Changing the phases of harmonics changes the temporal envelope shape of a complex tone, which affects the ACF responses [26]. Therefore, an autocorrelogram generated with the CAR-Lite-ACF model using AAC operations is capable of capturing the effects of harmonics phase change. CAR-Lite-ACF can be regarded as a unitary model of pitch perception to resolved and unresolved harmonics presented in alternating phase [14].

Harmonic Level	Harmonic Range	$f_0 = 150$ Hz	$f_0 = 300$ Hz
LOW	1 st – 5 th	150 Hz – 750 Hz	300 Hz – 1.5 kHz
MID	8 th – 16 th	1.2 kHz – 2.4 kHz	2.4 kHz – 4.8 kHz
HIGH	*16 th – 48 th	2.4 kHz – 7.2 kHz	4.8 kHz – 14.4 kHz

Table 4-4: Three harmonic group levels used for generating stimuli to demonstrate phase insensitivity effect on pitch. * - denotes increment of 2 harmonic levels.

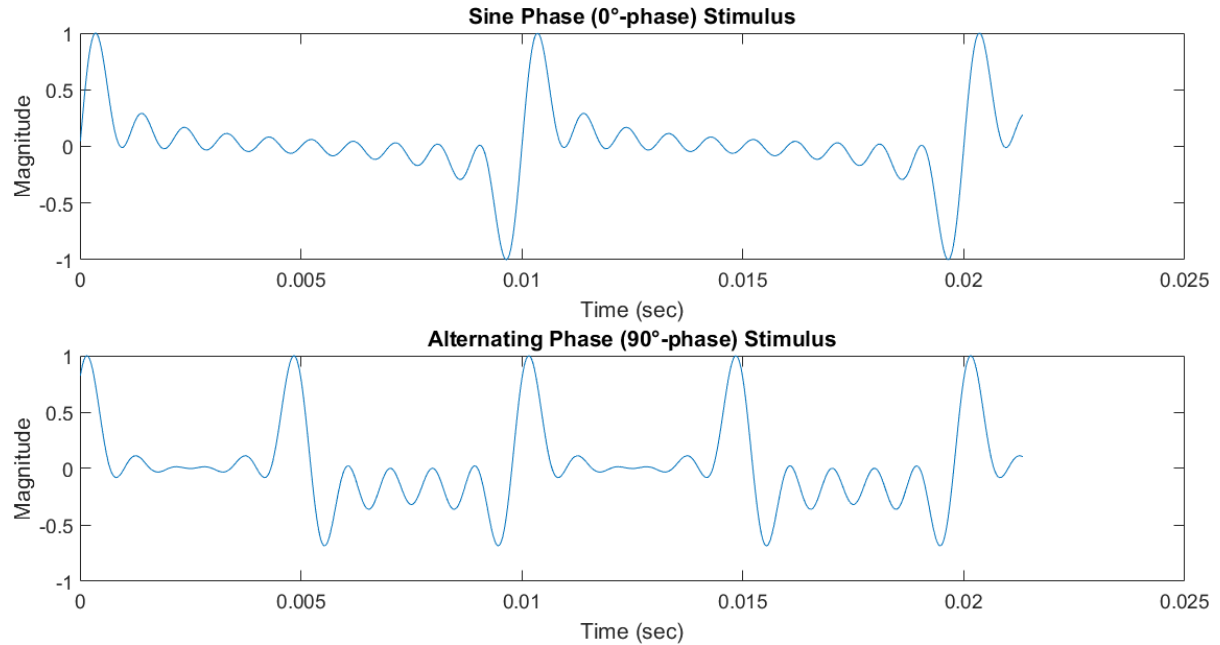


Figure 4-16: Stimulus with 1st to 10th harmonics for a fundamental frequency (f_0) of 100 Hz, summed in (a) sine (0°) phase, and (b) alternating (90°) phase.

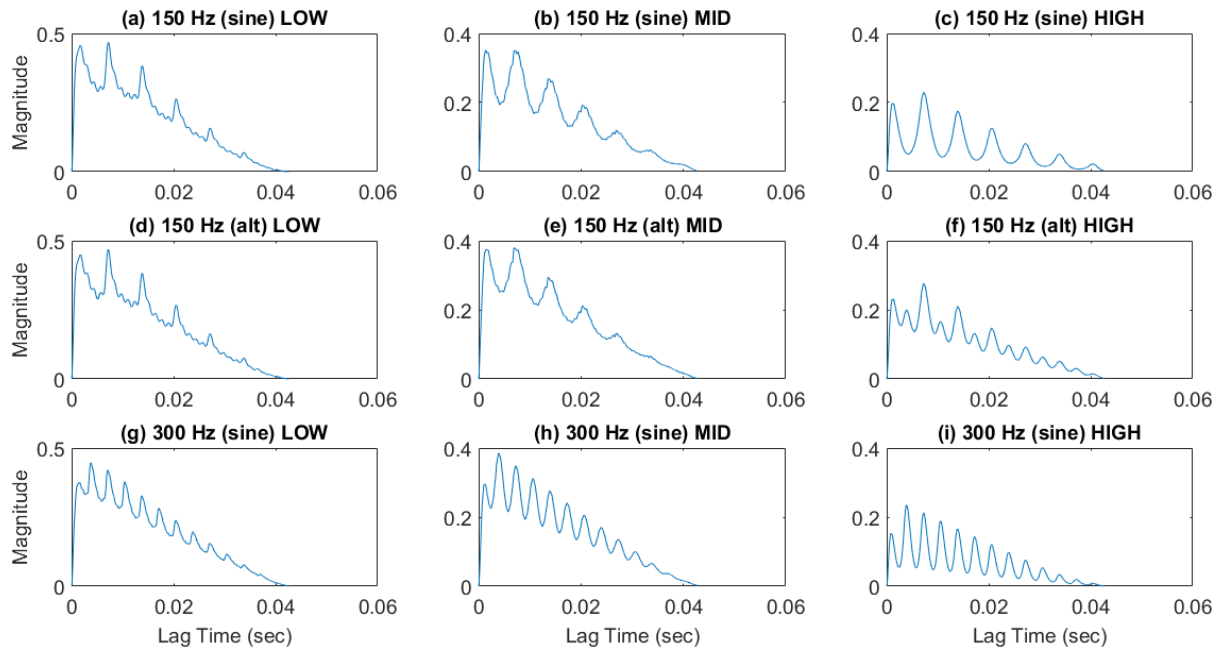


Figure 4-17: Temporal profiles of sine-phase and alternating-phase stimuli at three harmonic group levels as defined in Table 4-4 and generated from the CAR-Lite-ACF model using hardware-based AND-accumulate operations and fixed-point arithmetic.

4.5. Chapter Summary and Conclusion

This chapter presents a novel algorithm for generating an autocorrelogram for pitch detection. Conventional algorithm uses multiply and accumulate (MAC) operations on discrete real-values as its input, whereas the novel algorithm uses logical-AND and accumulate (AAC) operations on binary spike streams generated from leaky-integrate-and-fire (LIF) neurons. This novel algorithm is used for generating autocorrelograms. While low energy contributing to pitch information is removed from the autocorrelogram due to more significant quantisation errors introduced by AAC operations than MAC operations, salient pitch information such as the fundamental frequency of an input sound signal can be represented.

The firing thresholds are selected based on the highest correlation score on the autocorrelograms generated between the conventional and novel algorithms. This attribute results in a high degree of similarity between the autocorrelograms generated from both the algorithms. Additionally, the novel algorithm uses fewer computational resources such as logic circuit modules and digital signal processors on an FPGA. As a result, the power consumption of its implementation on FPGA is less than the conventional model, which makes the novel algorithm a more efficient real-time solution.

The autocorrelogram is insensitive to harmonic content phase change and is also capable of finding a missing fundamental frequency. Pitch estimation from software and hardware implementations of the autocorrelogram using the conventional and the novel algorithms on real-world musical signals at multiple intensity levels as well as with and without noise is described in chapter 6.

4.6. Bibliography

- [1] P. Sterling and J. B. Demb, "Retina," in *The Synaptic Organization of the Brain*, 5th ed., G. M. Shepherd, Ed. New York, USA: Oxford University Press, 2004, pp. 1–88.
- [2] T. Delbruck, "Algorithmic Processing of Event Streams," in *Event-Based Neuromorphic Systems*, S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, Eds. John Wiley & Sons Inc., 2015, pp. 365–380.
- [3] C. Posch, D. Matolin, R. Wohlgenannt, M. Hofstätter, P. Schön, M. Litzenberger, D. Bauer, and H. Garn, "Live demonstration: Asynchronous Time-based Image Sensor (ATIS) camera with full-custom AE processor," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, p. 1392, doi: 10.1109/ISCAS.2010.5537265.
- [4] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras with Spiking Output," *Proc. IEEE*, vol. 102, no. 10, pp. 1470–1484, 2014, doi: 10.1109/JPROC.2014.2346153.
- [5] S. C. Lim, A. R. Temple, S. Jones, and R. Meddis, "VHDL-based Design of Biologically Inspired Pitch Detection System," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, 1997, pp. 922–927, doi: 10.1109/ICNN.1997.616148.
- [6] S. Jones, R. Meddis, S. C. Lim, and A. R. Temple, "Toward a Digital Neuromorphic Pitch Extraction System," *IEEE Trans. Neural Networks*, vol. 11, no. 4, pp. 978–987, 2000, doi: 10.1109/72.857777.

- [7] P. A. Cariani and B. Delgutte, "Neural Correlates of the Pitch of Complex Tones. I. Pitch and Pitch Salience," *J. Neurophysiol.*, vol. 76, no. 3, pp. 1698–1716, 1996.
- [8] A. van Schaik and R. Meddis, "The Electronic Ear; Towards a Blueprint," in *Neurobiology Ionic Channels Neurons and the Brain*, L. Conti, Ed. Plenum Press, 1996, pp. 233–250.
- [9] A. van Schaik and R. Meddis, "Analog very large-scale integrated (VLSI) implementation of a model of amplitude-modulation sensitivity in the auditory brainstem," *J. Acoust. Soc. Am.*, vol. 105, no. 2 Pt 1, pp. 811–821, 1999, doi: 10.1121/1.426270.
- [10] A. van Schaik, "An Analogue VLSI Model of Periodicity Extraction in the Human Auditory System," in *Neural Information Processing, 1999. Proceedings. ICONIP '99. 6th*, 1999, pp. 107–112, doi: 10.1109/ICONIP.1999.843970.
- [11] G. E. Moore, "Cramming More Components onto Integrated Circuits (Reprint)," *Proc. IEEE*, vol. 86, no. 1, pp. 82–85, 1998, doi: 10.1109/JPROC.1998.658762.
- [12] R. K. Singh, "RTAP: Towards a Real-Time Auditory Periphery Simulation," in *International Conference on Future Computational Technologies*, 2015, pp. 52–57, doi: 10.17758/UR.U0315218.
- [13] S. Akhter and J. Roberts, *Multi-Core Programming: Increasing Performance through Software Multi-threading.*, 1st ed. Hillsboro: Intel Press, 2006.
- [14] R. Meddis and L. O'Mard, "A unitary model of pitch perception," *J. Acoust. Soc. Am.*, vol. 102, no. 3, pp. 1811–1820, 1997, doi: 10.1121/1.420088.
- [15] A. de Cheveigné, "Pitch Perception Models," in *Pitch: Neural Coding and Perception*, C. J. Plack, A. J. Oxenham, R. R. Fay, and A. N. Popper, Eds. New York, NY, USA: Springer Science+Business Media LLC, 2005, pp. 169–233.
- [16] Intel Corporation, "Cyclone V FPGA Features," 2016. <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/pt/cyclone-v-product-table.pdf>.
- [17] Texas Instruments, "TMS320C5545 Fixed-Point Digital Signal Processor," 2016. <http://www.ti.com/product/TMS320C5545> (accessed Dec. 09, 2018).
- [18] Analog Devices, "SigmaDSP Digital Audio Processor," 2018. <https://www.analog.com/en/products/adau1467.html#product-samplebuy> (accessed Dec. 09, 2018).
- [19] Texas Instruments, "66AK2Hxx Multicore DSP+ARM® KeyStone™ II System-on-Chip (SoC)," 2017. <http://www.ti.com/product/66AK2H14/samplebuy> (accessed Dec. 09, 2018).
- [20] Texas Instruments, "Low-Power Single 2-Input Positive-AND Gate SN74AUP1G08-Q1," 2012. <http://www.ti.com/product/SN74AUP1G08-Q1/samplebuy> (accessed Dec. 09, 2018).
- [21] Texas Instruments, "SN74AUC08 Quadruple 2-Input Positive-AND Gate," 2005. <http://www.ti.com/product/SN74AUC08/samplebuy> (accessed Dec. 09, 2018).
- [22] "Intel Cyclone V 5CEBA2U15C8N," 2018. <https://www.digikey.com/product-detail/en/intel/5CEBA2U15C8N/544-2749-ND/3880770> (accessed Dec. 09, 2018).

- [23] “Intel Cyclone V 5CGXFC7D7F31C8N,” 2018. <https://www.digikey.com/product-detail/en/intel/5CGXFC7D7F31C8N/544-2772-ND/3879506> (accessed Dec. 09, 2018).
- [24] R. Dekker, “What’s the Difference Between VHDL, Verilog, and SystemVerilog?,” *ElectronicDesign*, 2014. <https://www.electronicdesign.com/what-s-difference-between/what-s-difference-between-vhdl-verilog-and-systemverilog> (accessed Oct. 09, 2019).
- [25] R. P. Carlyon and T. M. Shackleton, “Comparing the fundamental frequencies of resolved and unresolved harmonics: Evidence for two pitch mechanisms?,” *J. Acoust. Soc. Am.*, vol. 95, no. 6, pp. 3541–3554, 1994, doi: 10.1121/1.409971.
- [26] R. D. Patterson, “A pulse ribbon model of monaural phase perception.,” *J. Acoust. Soc. Am.*, vol. 82, no. 5, pp. 1560–1586, 1987, doi: 10.1121/1.395146.

5. A Functional Primary Auditory Cortical Model

This chapter describes a functional model of a mammalian primary auditory cortex (A1). This model is based on the NSL model reviewed in chapter 2. Using this model as a reference, I present an implementable hardware model in this chapter, titled CAR-Lite-A1 model. The presentation includes a description of the model architecture as well as a filter considered from a survey of filters for use in the A1 segment of the model. This chapter also presents a hardware implementation of the model on FPGA as well as the comparison of the responses of the software and hardware models using several stimuli.

5.1. Motivation

The multiresolution spectro-temporal NSL model has three stages, as reviewed in chapter 2. The first stage includes the seed functions for the spectral and temporal domains. The seed function for the temporal domain is characterised by a gammatone function that is implemented as a bandpass filter on FPGA [1]. The seed function for the spectral domain is characterised by a Gabor-like function, which is the second derivative of a Gaussian function and is generally used as a filter to extract spatial response of a receptive field [2]. Its amplitude spectrum is similar to that of a bandpass filter. Both these seed functions contain exponential functions. For real-time operation in floating-point, exponential functions can be calculated using Schraudolph's fast exponential functions that can be approximated with only five 64 bits fixed-point mathematical operations and read back in a standard IEEE-754 floating-point format [3]. An example of the use of this fast exponential function is in the real-time implementation of Meddis's MAP model (described in chapter 2), called RTAP [4]–[6].

On FPGA, with fixed-point arithmetic, the implementation of an exponential function is attainable with an array of constants defining a basis exponential function stored in look-up tables. However, the bit width of the look-up tables must be significant to ensure that the precision of the output signal is maintained with respect to the limited bit widths required to represent input and intermediate signals as well as filter coefficients within the model. The use of the look-up tables increases memory usage as well as bit widths of the arithmetic logic units through the combined usage of a large number of DSPs. Furthermore, look-up tables also increase processing latencies due to frequent memory read and write accesses. In contrast, linear signal processing filters on FPGA use only a small number of coefficients, have low memory usage and enable low bit-width arithmetic operation. This characteristic was demonstrated in the implementation of the CAR-Lite cochlear model in chapter 3 that used 8 bits coefficients with 16 bits arithmetic operations.

In addition, the Gabor seed function mentioned above is used as a non-causal filter. It cannot principally be implemented in real-time. However, as the frequency axis of a 2D time-frequency cochleagram is constant due to a fixed number of cochlear sections, this filter can be implemented in real-time when all the samples of all the cochlear sections are available at a specific time. Implementations of a Gabor function on FPGA are possible [7]–[10], but these are approximations of a discrete Gabor transform that produces the basic "Mexican-hat" response. Intricate responses containing phase information calculations such as those related to the third stage of the NSL model require a large number of filter coefficients [11]–[14], which in turn, increases the utilisation of the limited memory on an FPGA. This effect also predictably, increases FPGA power consumption with the use of a more significant number of logic circuits to accommodate these additional operations.

In consideration with the factors presented so far, and to keep computational resources small, the spectro-temporal seed functions are replaced with a bandpass filter type that uses a small number of coefficients. Furthermore, the phase calculations of the third stage are omitted from the FPGA implementation. While phase information is essential in defining the 4D output response of the NSL model as well as synthesising the NSL model output audio close to the original input audio signal from the model's 4D response, they are less significant than amplitude variations in a sound signal in portraying timbre information [15]. Hence, this chapter introduces the CAR-Lite-A1 model that extracts only the spectro-temporal envelope from a sound signal without considering the phase information. This model is designed to be implementable on an FPGA. In other words, the CAR-Lite-A1 model is a modified version of the NSL model, which processes sound signal without calculating phase information. To understand the impact of the modifications, this chapter presents the analyses of the responses of the CAR-Lite-A1 model over various artificial stimuli. Chapter 7 presents the responses of the CAR-Lite-A1 model in regards to real-world stimuli, where the results of musical instruments classification, similar to the work of Patil et al. [16], are presented.

5.2. CAR-Lite-A1 Model

Figure 5-1 illustrates the CAR-Lite-A1 model. It comprises the CAR-Lite cochlear model described in chapter 3, at the initial stage and a functional A1 model containing a multiresolution spectro-temporal model derived from the NSL model described in chapter 2, at the second stage. Finite impulse response (FIR) filters are not considered for the model. Instead, only infinite impulse response (IIR) filters are considered, as IIR filters are generally faster than FIR filters in the context of execution speed [17].

The temporal and spectral modulation filters in the CAR-Lite-A1 model are selected from a group of causal IIR filter configurations surveyed in subsection 5.2.2. In adherence to the Occam's Razor principle, the selected configuration has a low number of coefficients, and stable operation across modulation ranges for FPGA implementation. This ensures low memory utilisation, which in turn conserves power utilised on an FPGA. Section 5.3 describes the replacement of the Hilbert transform within the NSL model with an IIR filter in the CAR-Lite-A1 model. Here, the priority is given to the filter that is capable of generating 90° phase shift at the modulation ranges.

Section 5.4 presents the CAR-Lite-A1 configuration with the filters from sections 5.2 and 5.3. Section 5.5 describes its fixed-point implementation, and section 5.6 describes its FPGA implementation. Finally, section 5.7 presents the results of the model.

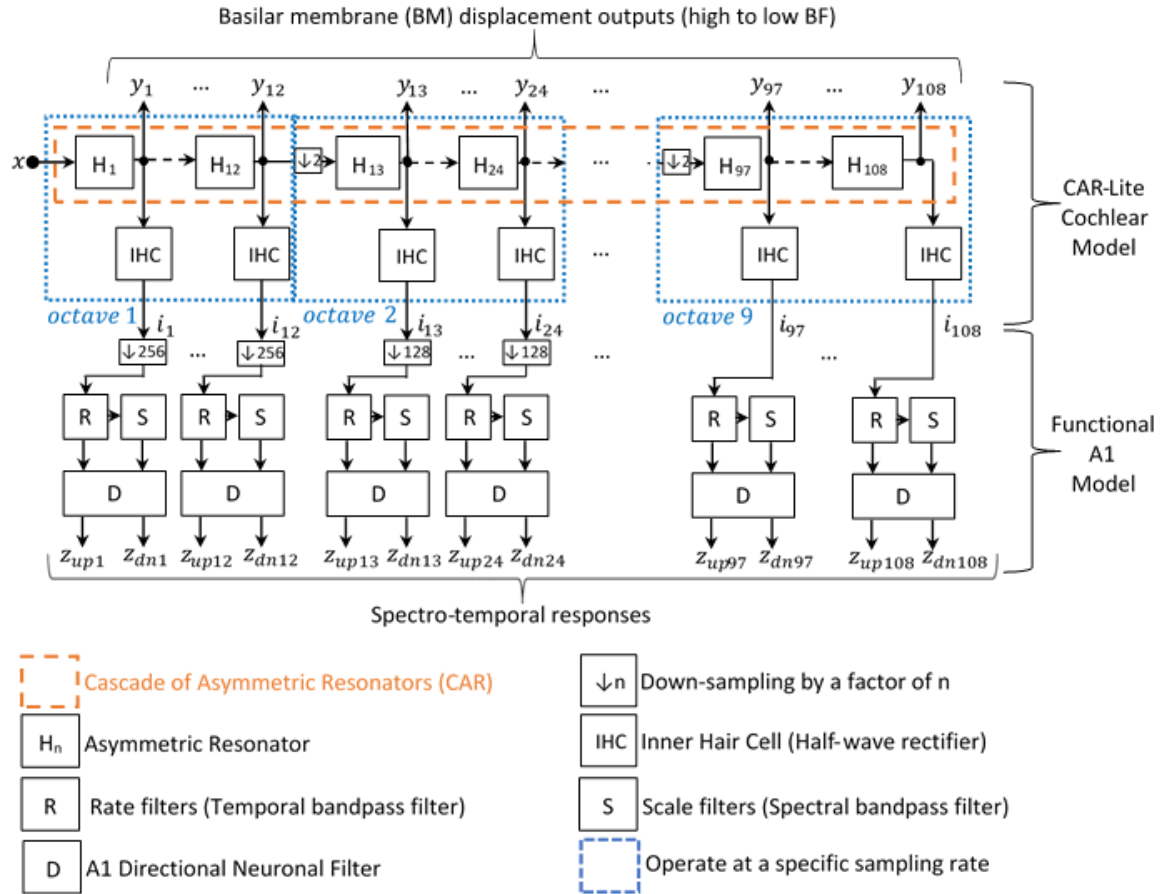


Figure 5-1: The CAR-Lite-A1 model comprising the CAR-Lite model described in chapter 3, and a multiresolution spectro-temporal model, known alternatively as a functional A1 model.

5.2.1. Input Sampling Rate

The input to the functional A1 segment of the CAR-Lite-A1 model, known as the A1 model, is an inner hair cell (IHC) signal from the CAR-Lite model. The first stage of the A1 model is a temporal bandpass filterbank, also known as a rate filterbank. The rate filterbank comprises multiple bandpass filters configured in parallel to detect envelopes from every cochlear section's IHC signal at multiple centre frequencies (or centre velocities) ranging from 2 Hz to 128 Hz, as specified fully in subsection 5.2.2. Due to the low modulation centre frequencies or also known as low envelope velocities, the IHC signal from every cochlear section is down-sampled to a standard sampling rate of 375 Hz, which is the sampling rate of cochlear octave 9. Based on the Nyquist sampling theorem, this is sufficient to accommodate the highest temporal modulation centre frequency of 128 Hz. Any energy more than 187.5 Hz is removed by an anti-aliasing filter defined by equation (5-1).

The motivation behind the down-sampling also lies with the inability of the selected filter specified in subsection 5.2.2.5 to operate at low frequencies with high sampling rates and low bit widths. This situation is realisable with a filter coefficient which is proportional to the ratio of the low modulation centre frequency to the sampling rate. The result is a small floating-point number that cannot be represented well by a fixed-point number with a small number of bits. Consequently, such filter coefficients are too close to zero and do not produce the desired filtering effect. For example, the ratio of a modulation centre frequency of 2 Hz and a sampling rate of 96 kHz (sampling rate of cochlear octave 1) results in

1/48,000. Consequently, a fixed-point number of more than 16 bits is required to represent this coefficient satisfactorily. An illustration of this effect is also described in subsection 3.2.6 of chapter 3 and agrees with [18].

The down-sampled IHC signal, IHC_{\downarrow} , is calculated using a mean function, identical to the calculation of the temporal integration variable, TI , in the CAR-Lite-SI model in chapter 3:

$$IHC_{\downarrow}(s, t_{\downarrow}) = \frac{1}{T} \sum_t^{t+T} IHC(s, t) \quad (5-1)$$

where the average IHC value of a cochlear section s at a subsampled time t_{\downarrow} , is calculated for every T samples. T is an octave specific constant set at a multiple of 2 to achieve a sampling rate of 375 Hz. Thus, the division can be implemented using a right-shift operation on FPGA. This form of averaging involves no multiplication and also functions as a low-pass filter as well as an anti-aliasing filter, which operates only with economical addition and right-shift operations.

The output of the temporal modulation filterbank is the input to a spectral modulation filterbank, also known as a scale filterbank. The scale filterbank extracts information on the density of energy across the cochlear sections. It does this with bandpass filters configured in parallel with spectral modulation centre frequencies (or centre densities) ranging from 0.25 cycles per octave (c/o) to 4 c/o, as specified fully in subsection 5.2.2. The selection of the sampling rate at 12 c/o corresponds to the number of cochlear sections per octave in the CAR-Lite model described in chapter 3. Based on the Nyquist sampling theorem, this sampling rate is capable of accommodating the highest spectral modulation centre density of 4 c/o.

5.2.2. Filter Configuration Survey for Spectro-Temporal Modulation Filters

In the NSL model reviewed in chapter 2, the rate filter is causal, whereas the scale filter is non-causal. A causal filter requires past and present input samples to generate an output sample, whereas a non-causal filter requires a past, present, and future input samples to generate an output sample. Hence, a causal filter can operate in real-time, because the generation of an output sample is dependent on a current input sample when it becomes available, in addition to stored past input samples. However, a non-causal filter cannot operate in real-time as the generation of an output sample requires future input samples as well as past and present input samples [19].

In the NSL model, the rate filterbank comprises causal filters, whereas the scale filterbank comprises non-causal filters. Hence, the real-time implementation of the rate filterbank is realisable, but this does not apply to the scale filterbank. Although the scale filterbank is implementable on FPGA with non-causal filters, this operation requires an entire vector of input samples ranging from the lowest to the highest frequency to be available, before the calculation of the output samples begins. This exercise does not indicate real-time operation as a non-causal scale filter module on FPGA remains inactive until an entire vector of input samples corresponding to all the cochlear sections is acquired. Thus, this adds to the delay in providing the output samples, which affects any causal filter that relies on these delayed output samples as inputs to operate. To alleviate this situation, it is a requirement to process every input sample as they become available for both the rate and scale filterbanks.

This notion requires the implementation of the scale filterbank as causal filters instead of non-causal filters.

The seed functions for the rate filter, h_t , and scale filter, h_s , in the NSL model are defined in chapter 2, and are repeated here for clarity:

$$h_t(t) = t^2 e^{-3.5t} \sin(2\pi t) \quad (5-2)$$

$$h_s(x) = (1 - x^2) e^{-x^2/2} \quad (5-3)$$

where t is time and x is frequency. Both h_t and h_s use nonlinear functions: the temporal seed function, h_t , uses an exponential, a sinusoidal and a squared function, while the spectral seed function, h_s , has an exponential and a squared function. Although nonlinear functions are implementable on FPGA [7], [9], their nonlinear characteristics are approximated using an array of numerical values that are stored in look-up tables on FPGA [20], [21] – a more extensive array generates an approximation of the nonlinear function with higher accuracy as opposed to a small array. Furthermore, the number of bits required to represent these numbers are significant, i.e., 64 bits [22]. Hence, these look-up tables generally require significant memory space for operation. In contrast, a linearly-weighted infinite impulse response (IIR) filter does not require such additional look-up tables. It only requires memory for its coefficients used based on the filter order, e.g., a second-order filter might require memory for only four coefficients depending on the filter configuration.

Furthermore, since the rate and scale filters have bandpass characteristics in the temporal and spectral domains respectively [23], each of them can be implemented by a linear and time-invariant IIR bandpass filter (BPF) instead of the original filters with nonlinear seed functions. This option would allow the IIR BPF to be implemented as a common filter for processing the rate and scale filters filter instead of two separate ones in the NSL model. Alternatively, the temporal domain filter can be implemented with a three-pole pair IIR filter governed by a gammatone function [24], and the spectral domain filter can be implemented as a multi-pass forward and backward IIR smoothing filter such as the one implemented in the automatic gain control (AGC) of the CAR-FAC model [25].

The common filter option mentioned in the preceding paragraph is selected for implementation. In other words, only a single filter is implemented in a software function, and, during processing, this function is invoked twice as many times with one invocation running the IIR filter as a temporal filter and the other invocation running the IIR filter as a spectral filter. Given the limited computational resource of an FPGA, this exercise is appealing as less non-volatile memory is required for storage of the CAR-Lite-A1 model code than if two separate seed functions, such as that of the NSL model, are used. This implementation is per the Occam's Razor principle, whereby a simple design approach is taken that aids in the conservation of computing resources, which in turn maintains low power utilisation.

Several IIR BPF configurations are implementable on an FPGA. Subsections 5.2.2.1 to 5.2.2.4 consider four such filters. Subsection 5.2.2.5 presents a selection of an appropriate IIR filter out of the four configurations for FPGA implementation. Finally, in subsection 5.2.2.6, the stability of the selected filter is presented for different centre velocities for the rate filterbank and centre densities for the scale filterbank. Note that only second-order

bandpass configurations are reviewed in the subsections below – the minimum order of degree for a bandpass filter. The exception is the LPF-HPF cascade configuration described in subsection 5.2.2.3 that uses first-order filters.

In the following subsections, the centre velocities, f_{rt} , of a rate filterbank and centre densities, f_{sc} , of a scale filterbank are defined in accordance with the settings of the NSL model used for musical instrument classification [16] as follows: $f_{rt} = [2.00, 2.83, 4.00, 5.66, 8.00, 11.31, 16.00, 22.63, 32.00, 45.25, 64.00, 90.51, \text{ and } 128.00]$ Hertz (Hz); $f_{sc} = [0.25, 0.35, 0.50, 0.71, 1.00, 1.41, 2.00, 2.83, 4.00]$ cycles per octave (c/o). The rate filter operates at a sampling rate of 375 Hz, and for the scale filter, the sampling rate is 12 c/o.

5.2.2.1. Standard Biquadratic

The standard bandpass filter (BPF) is derived from a biquadratic transfer function comprising the ratio of two quadratic equations. The transfer function has an infinite impulse response (IIR), which is a second-order recursive linearly-weighted time-invariant filter with two poles and two zeros [26]. It is considered recursive as the output of the filter is determined by the input signal as well as the delayed version of the input and output signals [27]. Its transfer function is defined as:

$$H(z) = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{a_0 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}} \quad (5-4)$$

where b_n are coefficients of a feedforward path; a_n are coefficients of a feedback path; z is a delay element, wherein z^{-n} is an n^{th} -order delay.

From the transfer function of equation (5-4), two major signal path configurations can be derived based on the placement order of the poles and zeros directly factorised from the transfer function. When the configuration has zeros-poles connected between the input and output pathways respectively (zeros connected to input pathway and poles connected to output), it is known as direct-form-1 [28]. When it has poles-zeros connected between the input and output pathways, it is known as direct-form-2 [29]. The direct-form-1 is immune to internal numerical overflow from multiplication and addition operations in its configuration. However, the direct-form-2 has no such immunity and is prone to numerical overflow. As a result, the direct-form-1 is capable of generating stable responses more consistently than direct-form-2 [29]. Despite its stability, the direct-form-1 is slower in generating an output signal as it has more delay components than direct-form-2 [28]–[30]. To ensure stable operations, only the direct-form-1 configuration is considered.

The direct-form-1 configuration is defined by the following difference equation [31]:

$$y(t) = \frac{b_0}{a_0} \cdot x(t) + \frac{b_1}{a_0} \cdot x(t-1) + \frac{b_2}{a_0} \cdot x(t-2) - \frac{a_1}{a_0} \cdot y(t-1) - \frac{a_2}{a_0} \cdot y(t-2) \quad (5-5)$$

where x is the input and y is the output. It comprises two serially cascaded finite impulse response (FIR) filters, i.e. one FIR segment deals with the input signal convolved with weights b_n , and the other deals with the output signal convolved with weights, a_n . It has five internal variables, whereby each internal variable corresponds to a term in the equation that requires storage at runtime. Since FIR filters are always stable, this also enables the direct-form-1 configuration to be more stable than the direct-form -2 configuration [29].

To obtain a bandpass filter response from equation (5-5), coefficients a and b , are defined according to [32] and are depicted in Table 5-1(a) under the “Standard Filter” column. Figure 5-2 displays the gain and phase responses of the standard direct-form-1 BPF configured as a rate filterbank using f_{rt} and as a scale filterbank using f_{sc} .

Coefficient	(a) 2 nd -order IIR Standard Filter	(b) 2 nd -order IIR Peaking Equaliser
b_0	α	$1 + \alpha \cdot A$
b_1	0	$-2 \cdot \cos(\omega_0)$
b_2	$-\alpha$	$1 - \alpha \cdot A$
a_0	$1 + \alpha$	$1 + \alpha/A$
a_1	$-2 \cdot \cos(\omega_0)$	$-2 \cdot \cos(\omega_0)$
a_2	$1 - \alpha$	$1 - \alpha/A$
Intermediate Parameters		
ω_0	$2 \cdot \pi \cdot f_0/f_s$	
α	$\sin(\omega_0)/(2 \cdot Q)$	
A	$10^{(dBgain/40)}$	
Q	1	
$dBgain$	5	
f_0	Centre velocities of rate filters, f_r , and centre densities of scale filters, f_s .	

Table 5-1: Coefficients used in (a) a 2nd-order IIR standard bandpass filter, and (b) a 2nd-order IIR peaking equaliser filter.

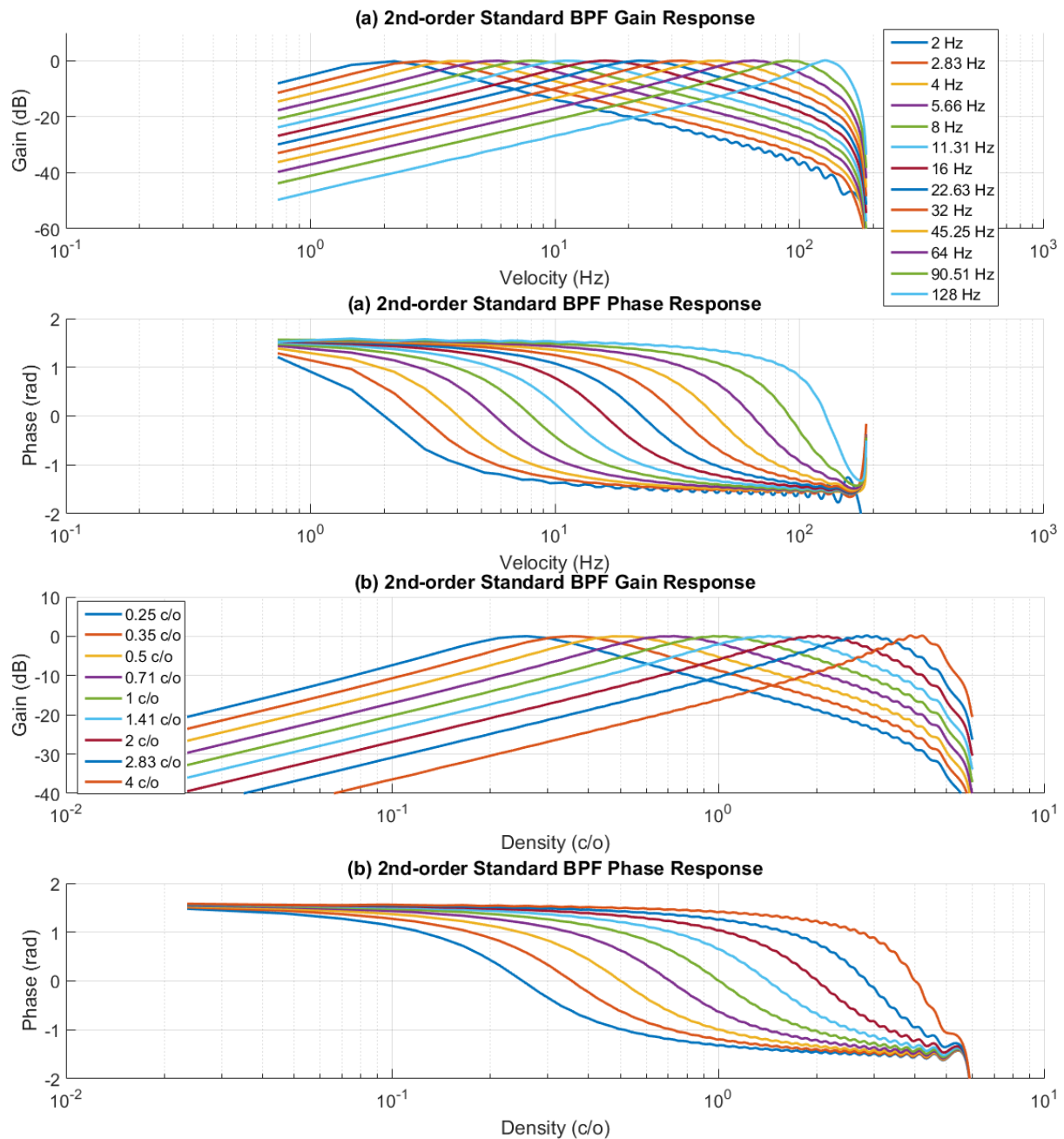


Figure 5-2: Gain and phase responses of (a) rate and (b) scale filterbanks using the 2nd-order IIR standard bandpass filter (BPF) direct-form-1 configuration.

5.2.2.2. Peaking Equaliser

Another derivative of the direct-form-1 IIR filter configuration from subsection 5.2.2.1, is a peaking equaliser filter, which uses the same difference equation from (5-5) but the coefficients are from Table 5-1(b). This configuration provides a gain response similar in shape to the aforementioned standard bandpass filter. However, one significant difference between the two is that the peaking equaliser filter has positive gains at centre velocities and centre densities as displayed in Figure 5-3. As a result, the peaking equaliser filters are known as boost filters. They are widely used in sound recording mixers for volume control at specific frequencies [33].

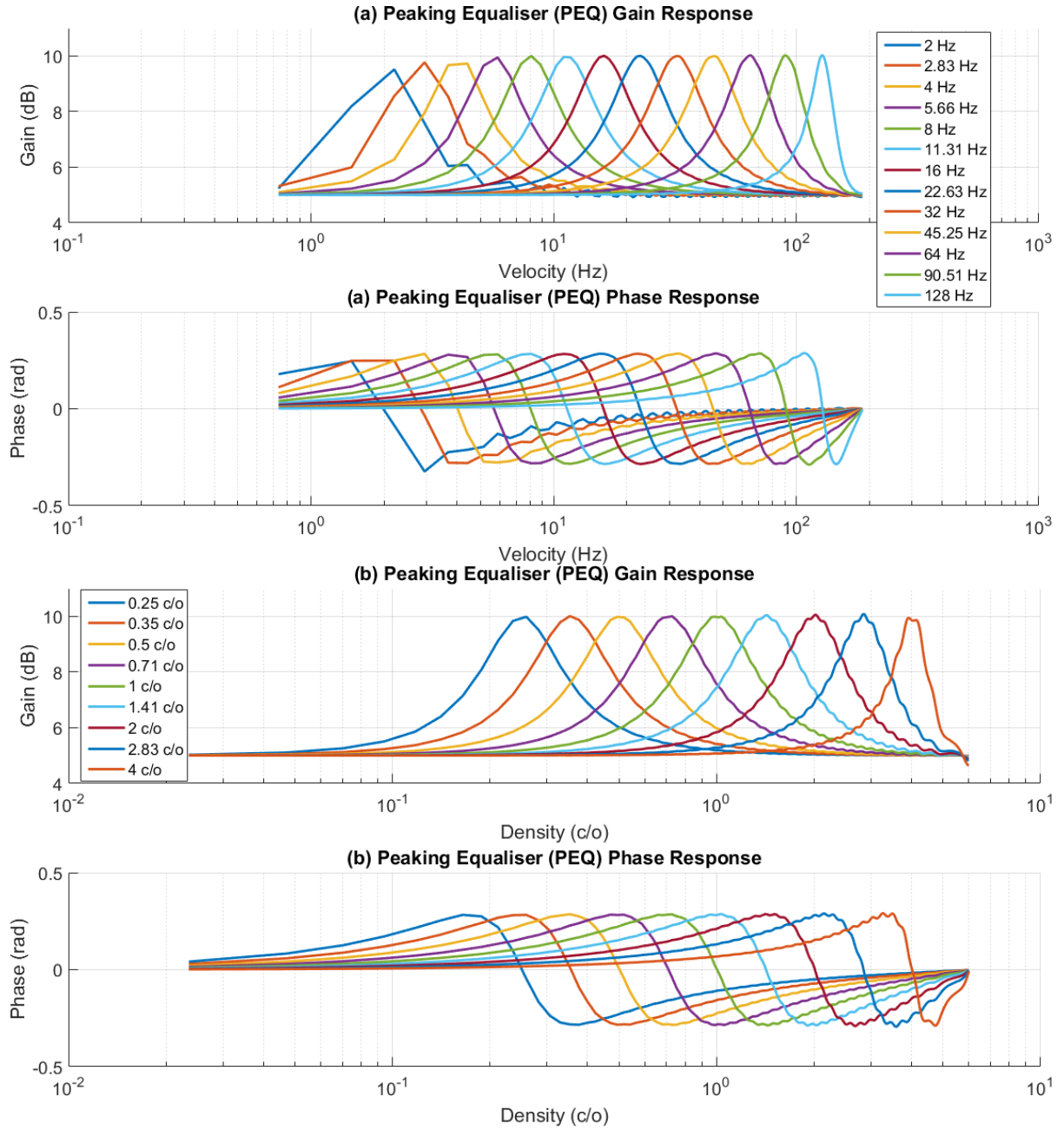


Figure 5-3: Gain and phase responses of (a) rate and (b) scale filters using a 2nd-order IIR peaking equaliser filter.

5.2.2.3. Low-pass Filter and High-pass Filter (LPF-HPF) Cascade

Another bandpass filter (BPF) configuration to be considered for hardware implementation of rate and scale filters is a serial cascade of a low-pass filter (LPF) and a high-pass filter (HPF) [34]. Based on the Fourier transform property, the convolution of the two filter kernels in the time domain is a multiplication of their responses in the frequency domain [35]. This property enables the summation of their filter orders, i.e. a first-order LPF cascaded with a first-order HPF results in a second-order BPF. Two passes of this second-order BPF results in a fourth-order BPF. This fourth-order BPF configuration has been used in the implementation of a single rate filter appended to a cochlear model on FPGA by Thakur et al. [1]. The second-order and fourth-order BPF cascade configuration has the following first-order responses:

$$y_l(t) = y_l(t-1) + c_l \cdot (x(t) - y_l(t-1)) \quad (5-6)$$

$$y_h(t) = c_h \cdot (y_h(t-1) + y_l(t) - y_l(t-1)) \quad (5-7)$$

where y_l is the LPF output and y_h is the HPF output at discrete time t . The coefficients for LPF and HPF, c_l and c_h , are calculated as follows:

$$c_l = 2 \cdot \pi \cdot \frac{f_0}{f_s} \quad (5-8)$$

$$c_h = \frac{1}{1 + 2 \cdot \pi \cdot \frac{f_0}{f_s}} \quad (5-9)$$

where f_s is the sampling rate; f_0 is either the centre velocity, f_{rt} , of a rate filter or centre density, f_{sc} , of a scale filter.

Figure 5-4 displays the gain and phase responses of the second-order cascaded configuration for the 13 centre velocities used in the rate filterbank and the 9 centre densities used in the scale filterbank. For the rate filters, the gain responses are similar to the second-order standard bandpass filter configurations except for low centre velocity at 2 Hz, which exhibits a gain response looking more like a low-pass filter response than a bandpass response. At centre velocities above 45 Hz and centre densities above 1 c/o for the scale filter, the filter exhibits a high-pass filter response rather than a bandpass response. Furthermore, the filter gain responses at the high end at 90.5 Hz and 2.83 c/o have linear gain responses, where the passband and stopband become indistinguishable. In addition, the filter responses for the highest centre velocity of 128 Hz and centre density of 4 c/o are not displayed as they have substantial gains at 507 dB and 334 dB respectively as compared to all other centre velocities and densities, which possess gains within the range of -3 dB to -10 dB (in decreasing centre velocities and centre densities respectively).

In [1], a fourth-order rate filter is used at only a single centre velocity at 4 Hz, where two second-order cascaded LPF-HPF are connected in series. With this specific configuration, the rate filters from 2 Hz to 64 Hz centre velocities and the scale filters from 0.25 c/o to 2 c/o centre densities have increasing gains as observed in Figure 5-5 instead of constant gains for the second-order cascaded configuration as shown in Figure 5-4. Moreover, the gains of the fourth-order BPF are lower than the gains of the second-order BPF, which would require a more significant bit widths for the former than the latter. In addition, in Figure 5-5, the fourth-order BPF cascade responses for two centre velocities at 90.5 Hz, 128 Hz, and two centre densities at 2.83 c/o, and 4 c/o, are not displayed as they have gains more significant than the gains delivered at lower centre velocities (2 Hz to 64 Hz) and lower centre densities (0.25 c/o to 2 c/o), respectively. For scale filters with centre densities at 90.51 Hz and 2.83 c/o, the gains are 296 dB and 28 dB, respectively. For rate filters with centre velocities at 128 Hz and 4 c/o, their gains are beyond 3,000 dB. Due to these factors, the fourth-order BPF cascade configuration used by Thakur et al. [1] is not considered in the characterisation of the rate and scale filterbanks.

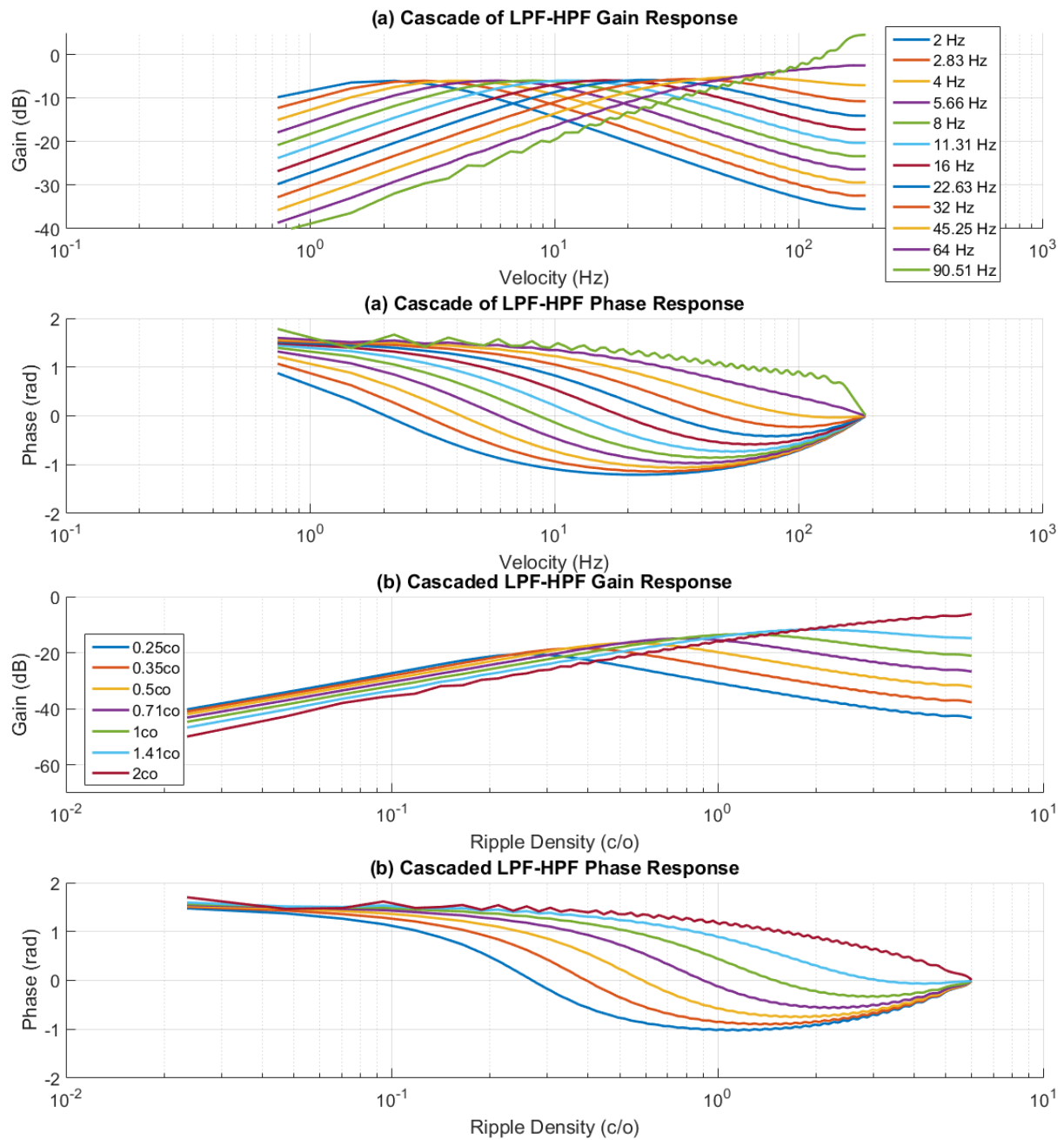


Figure 5-4: Gain and phase responses of (a) rate and (b) scale filterbanks using a 2nd-order cascade BPF comprising 1st-order low-pass filter (LPF) and a 1st-order high-pass filter (HPF).

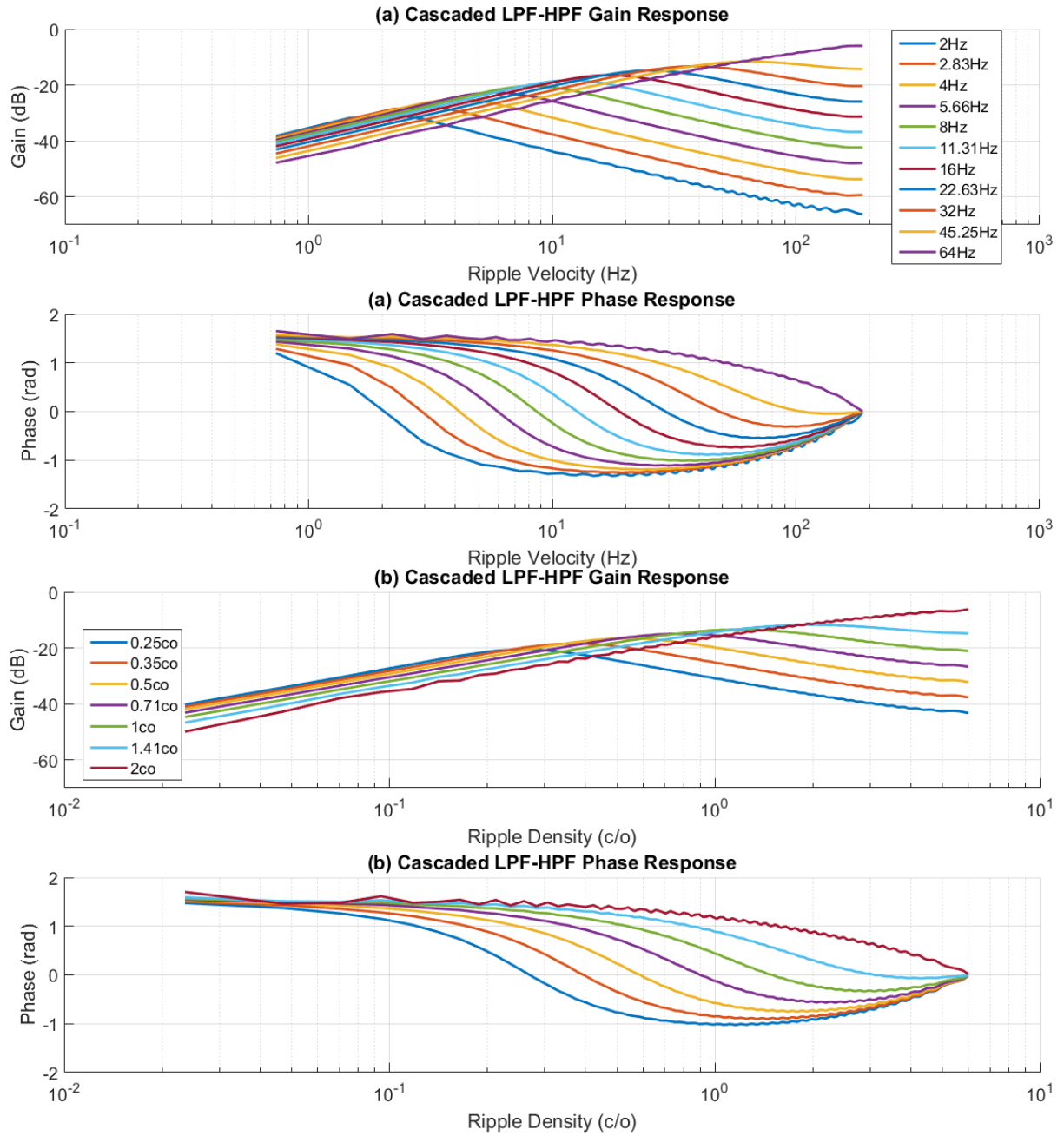


Figure 5-5: Gain and phase responses of (a) rate and (b) scale filterbanks using a 4th-order cascade BPF comprising two serial cascades of 2nd-order cascade BPF as used by Thakur et al. [1].

5.2.2.4. Coupled-Form (Asymmetric Resonator)

The last configuration for consideration is a coupled-form [36] of a second-order asymmetric resonator, which will be alternatively known in this chapter as an asymmetric resonator (AR). It has been used in the cascaded configuration of the CAR-Lite model, depicting basilar membrane (BM) characteristics for extracting resonances from a sound signal as described in chapter 3. For the sake of clarity, its transfer function is repeated here:

$$H(z) = \frac{y}{x} = g \left(\frac{z^2 + (-2a + hc)rz + r^2}{z^2 - 2arz + r^2} \right) \quad (5-10)$$

where x is the input; y is the output; g is the overall unity gain at DC; a is the real component, and c is the imaginary component of a complex number structure representation of the coupled-form configured filter; h controls the distance of zeros from the frequency of the poles and is set to the real components of the ringing frequency, c .

Figure 5-6 displays the coupled-form filter configuration connections. Its output is defined by:

$$y = g(x + hW_1) \quad (5-11)$$

where W_1 is one of two internal variables; the other being W_0 . They are defined as:

$$W_1 = r(aW_1 + cW_0) \quad (5-12)$$

$$W_0 = x + r(aW_0 - cW_1) \quad (5-13)$$

The direct-form-1 configuration suffers from the shift of its poles due to round-off errors in a fixed-point format, especially at high filter orders [28]. In other words, a serial cascade of either the second-order standard bandpass filters from subsection 5.2.2.1 or peaking equaliser filters from subsection 5.2.2.2, designed with poles exhibiting stable operations in floating-point might be unstable when it operates in fixed-point arithmetic. This unstable operation occurs especially at low resonant frequencies [37] as is the case for centre velocities and centre densities that are used for the rate and scale filters, respectively. As a consequence, the coupled-form configuration was designed to alleviate this quantisation effect at low resonant frequencies [38].

Figure 5-7 illustrates the gain and phase responses of the rate filterbank for all 13 centre velocities and Figure 5-8 illustrates the same responses of the scale filterbank for all 9 centre densities from the coupled-form configuration. From these responses, one conclusion is that the coupled-form configuration is actually a low-pass filter (LPF) as it has DC gain at low frequencies as observed in the 0 dB gains for all centre velocities in Figure 5-7 and all centre densities in Figure 5-8. This configuration allows low frequencies to pass through the filter. It also has a low damping factor enabling positive gains, which projects bandpass responses in the figures. As a result, numerical values at bandpass frequencies are scaled higher than low-pass frequencies.

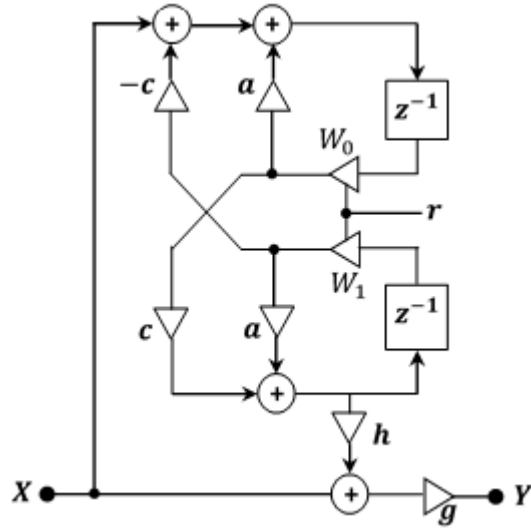


Figure 5-6: Coupled-form configuration of a 2nd-order asymmetric resonator. This configuration is also used to build the filters in the CAR-Lite model described in chapter 3. Adapted from Lyon [39].

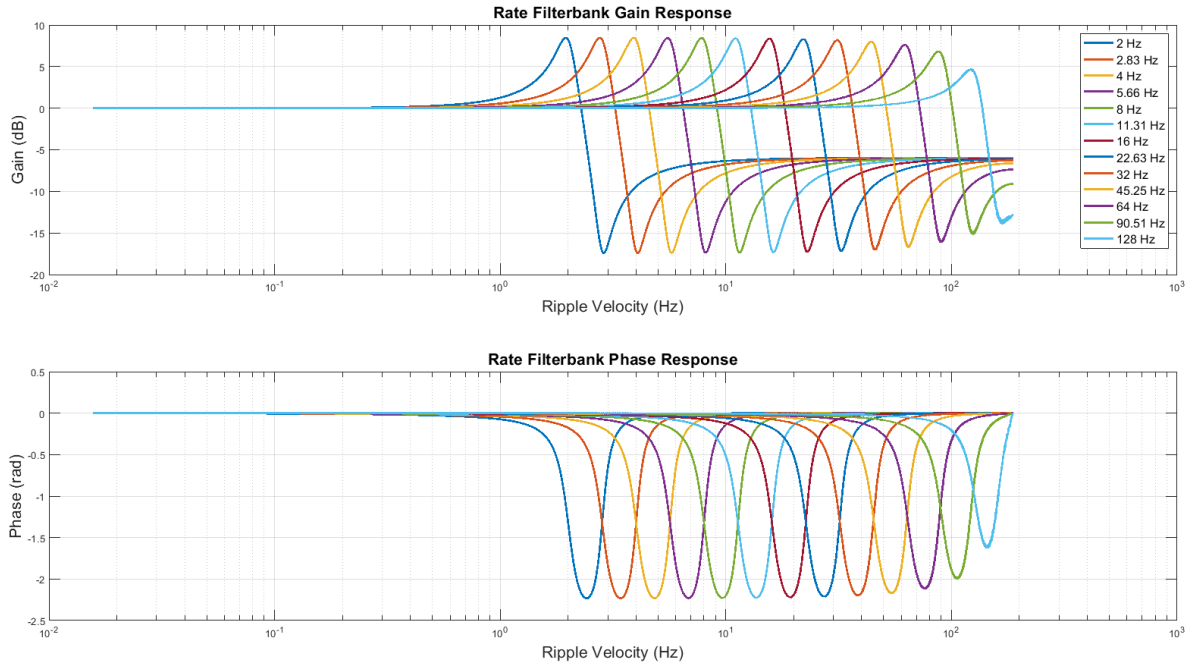


Figure 5-7: Frequency and phase responses of a coupled-form configured asymmetric resonator tuned to centre velocities of temporal modulations ranging from 2 Hz to 128 Hz.

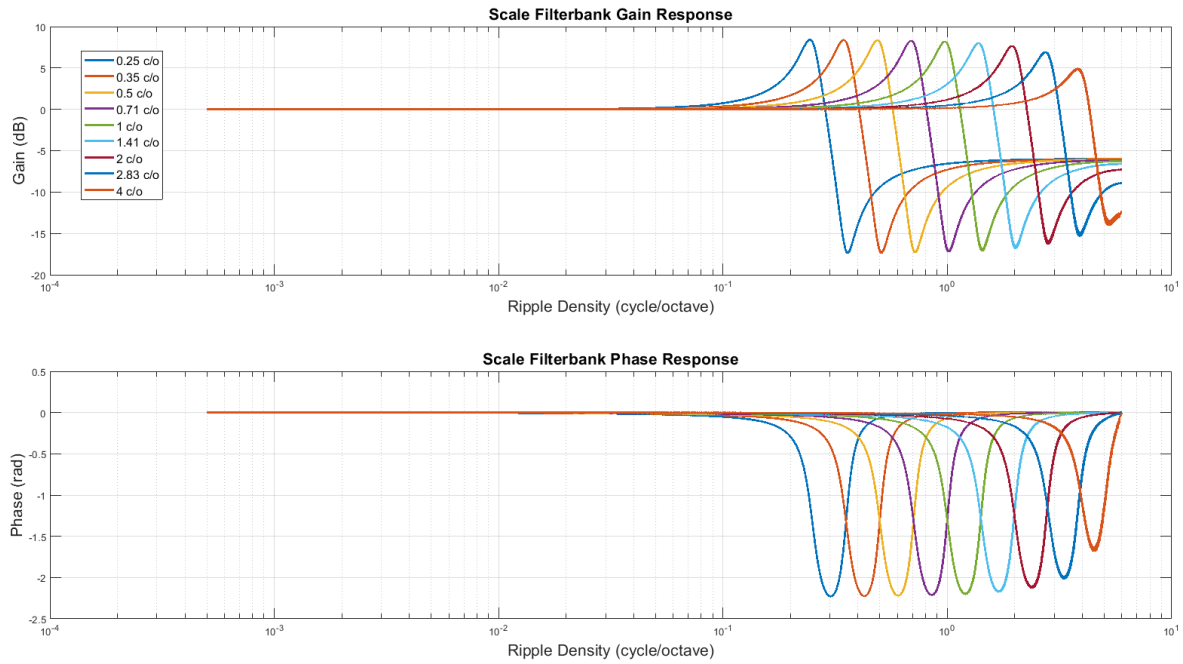


Figure 5-8: Frequency and phase responses of a coupled-form configured IIR filter tuned to centre densities of spectral modulations ranging from 0.25 c/o to 4 c/o.

5.2.2.5. Configuration Selection

Table 5-2 projects the coefficients for each of the four filter configurations surveyed above based on storage requirements ranging from the lowest to the highest. The table is further divided into two halves, whereby the first half shows coefficients requiring non-volatile (permanent) storage and the second half shows coefficients requiring volatile (runtime) storage. The LPF-HPF cascade requires the least amount of storage. However, as they have highly irregular gain responses and non-uniform bandpass shapes across all centre velocities and densities, the cascaded LPF-HPF configuration is not considered for implementations of the rate and scale filterbanks.

As part of the direct-form-1 configuration, the standard bandpass and peaking equaliser filters are capable of operating over the entire range of centre velocities and centre densities. However, the storage requirements of their respective coefficients are high in comparison with the two other configurations. Specifically, in terms of volatile storage requirements, coefficients of the peaking equaliser require the most storage, followed by coefficients of the standard BPF. The significant storage requirement for the two filter types is due to the reliance of the generation of an output sample on the current input sample, and two past weighted input samples as well as the two past weighted output samples in time respectively as defined by equation (5-1). This notion corresponds to five internal variables that require storage per output sample for one filter during runtime. In contrast, a second-order IIR filter with coupled-form configuration is only reliant on two internal variables to generate an output sample for the same range of centre velocities and centre densities. As a result, the coupled-form configuration is selected to implement the rate and scale filterbanks.

(a) Non-volatile Storage Requirements					
2 nd -order IIR filter configuration	Number of coefficients (per cochlear section)	Number of coefficients across 108 cochlear sections	Number of coefficients for 13 centre velocities (Rate filter bank)	Number of coefficients for 9 centre densities (scale filter bank)	Total number of coefficients requiring storage
LPF-HPF Cascade	2	2	26	18	44
Coupled-Form Resonator	4	4	52	36	88
Bandpass	5	5	65	45	110
Peaking Equaliser	5	5	65	45	110
(b) Volatile Storage Requirements					
LPF-HPF Cascade	2	216	2,808	1,944	4,752
Coupled-Form Resonator	2	216	2,808	1,944	4,752
Bandpass	5	540	7,020	4,860	11,880
Peaking Equaliser	6	648	8,424	5,832	14,256

Table 5-2: The four filter configurations reviewed in subsection 5.2.2 and their corresponding number of coefficients required for (a) non-volatile (permanent) storage and (b) volatile (runtime) storage.

5.2.2.6. Seed Function of the Selected Filter Configuration

In this subsection, the seed function of the selected second-order coupled-form asymmetric resonator implemented in the CAR-Lite-A1 model is compared with the temporal seed function, h_t , and spectral seed function, h_s , from the NSL model. The seed functions from the NSL model are described in chapter 2 and reiterated in subsection 5.3.1 in this chapter. The seed function of the selected asymmetric resonator can be found by applying the inverse z-transform [40] to $H(z)$ from equation (5-10). Doing so leads to the following temporal and spectral seed functions differentiated accordingly by t (time) and x (log frequency) variables:

$$h_{nt}(t) = a\delta(t) + (1 - a) \frac{\omega_r}{1 - \zeta^2} e^{-\gamma t} \sin(\omega_r t) \quad (5-14)$$

$$h_{ns}(x) = a\delta(x) + (1 - a) \frac{\omega_r}{1 - \zeta^2} e^{-\gamma x} \sin(\omega_r x) \quad (5-15)$$

where $\delta(\cdot)$ is a Kronecker delta function scaled by a ; ω_r is the ringing resonance corresponding to either the centre velocity or centre density; ζ is the damping factor.

The uniqueness of the two-pole-two-zero configuration is that each of the seed functions above has two terms separated by a sum operation, which can be illustrated as two parallel branches fed to a summing junction [41]. The second terms of h_{nt} and h_{ns} from above, containing the exponential and sine functions, are akin to the temporal seed function, h_t , defined by equation (5-22) in the NSL model. h_t is also known as a gammatone filter made of a sine tone multiplied by a gamma distribution, and it is used for building auditory filterbanks [42]. Setting $a = 0$, $\gamma = 1.5$, $\zeta = 0.1$, and $\omega_r = 2\pi$, enables h_{nt} response displayed in Figure 5-9(c) to closely resemble h_t response displayed in Figure 5-9(a). An identical response between h_t and h_{nt} is not sought as h_t contains an additional parameter, t^2 , which h_{nt} does not possess.

Moreover, introducing t^2 to h_{nt} requires a change in the transfer function of the latter that would require an extra module on the FPGA to be designed solely for h_t . Ideally, not introducing any new parameters to h_{nt} enables only one module to be used on the FPGA that can be used by both h_t and h_s as they would both have the same transfer function. Overall, h_{nt} is considered appropriately as a modified gammatone filter that has close characteristics as h_t and hence, is used in the CAR-Lite-A1 model in place of h_t .

Figure 5-9(b) displays the nonlinear Gabor seed function used in the NSL model. This filter can be approximated using a recursive linear IIR filter. To do this, Young et al. used a sixth-order transfer function comprising a serial cascade of a forward pass (causal) third-order and a backward pass (anti-causal) third-order IIR filter [12]. In the temporal domain, a forward pass filter is known as a causal filter, where the output of the filter is determined by the present, t , and past, $t - k$, input samples. In the spatial domain, the output of the forward pass filter is dependent on the input samples from the current, n , and previous neighbouring, $n - k$, space being analysed.

In contrast, a backward pass filter, which is known as an anti-causal filter in the temporal domain, generates an output sample from the present, t , and future, $t + k$, input samples. In the spatial domain, the output of the backward pass filter is dependent on the samples from the current, n , and upcoming neighbouring, $n + k$, space to be analysed. The combination of the forward and backward pass filters is feasible considering the upcoming input samples of the latter are already known and stored in memory when the forward pass filter is processed. An example of this implementation is the AGC filter [25] in the CAR-FAC model.

David et al. extended Young's design and incorporated the forward pass (causal) and backward pass (anti-causal) filters into a parallel cascade in addition to the serial cascade and reduced the filter order to a minimum of two, using only six coefficients instead of ten from Young's implementation [13]. Doing so means that the output of the Gabor filter can be approximated with a single pass (delay) using the parallel cascade configuration, similar to the coupled-form configuration. However, the second-order IIR Gabor filter requires non-volatile storage for up to six coefficients as opposed to four for the coupled-form configured asymmetric resonator per centre density. Due to its higher memory requirements, the IIR Gabor filter is not considered for implementation in the CAR-Lite-A1 model. Instead, the second-order coupled-form configured asymmetric resonator is selected.

The Gabor-like function, h_s , used in the NSL model has a bandpass-like response resembling a "Mexican-hat" as illustrated in Figure 5-9(b). To ensure that h_{ns} has a response closely resembling h_s , the parameters in equation (5-15) are set as follow: $a = 0$, $\gamma = 0.1$, $\zeta =$

0.1, and $\omega_r = 2\pi$. Doing so results in h_{ns} having a bandpass-like effect but with two unevenly suppressed sideband (a nonlinear sideband after octave 2 and a linear sideband below octave -2) as observed in Figure 5-9(d) instead of two even ones (a nonlinear sideband at octave 1.5 and another nonlinear one at octave -1.5) for h_s in Figure 5-9(c). This effect is insignificant as the sidebands represent neuronal inhibition that is meant to suppress the energy at regions outside the area of interest – in this case at 1 c/o centre density. So, at octaves below -1.5, the magnitudes of the energy are lower than the energy beyond 1.5 octaves, which are both deemed as insignificant areas outside the region of interest. Therefore, their respective sideband shapes are inconsequential. Another difference between h_s and h_{ns} responses is that the bandwidth of the excitatory neuronal region (areas above 0) is higher for h_{ns} at 3 octaves than h_s at 2 octaves. This difference indicates that h_{ns} is sensitive to energy spread across a larger number of cochlear sections than h_s by 1 octave, which is an allowable compromise at the expense of maintaining low computational cost.

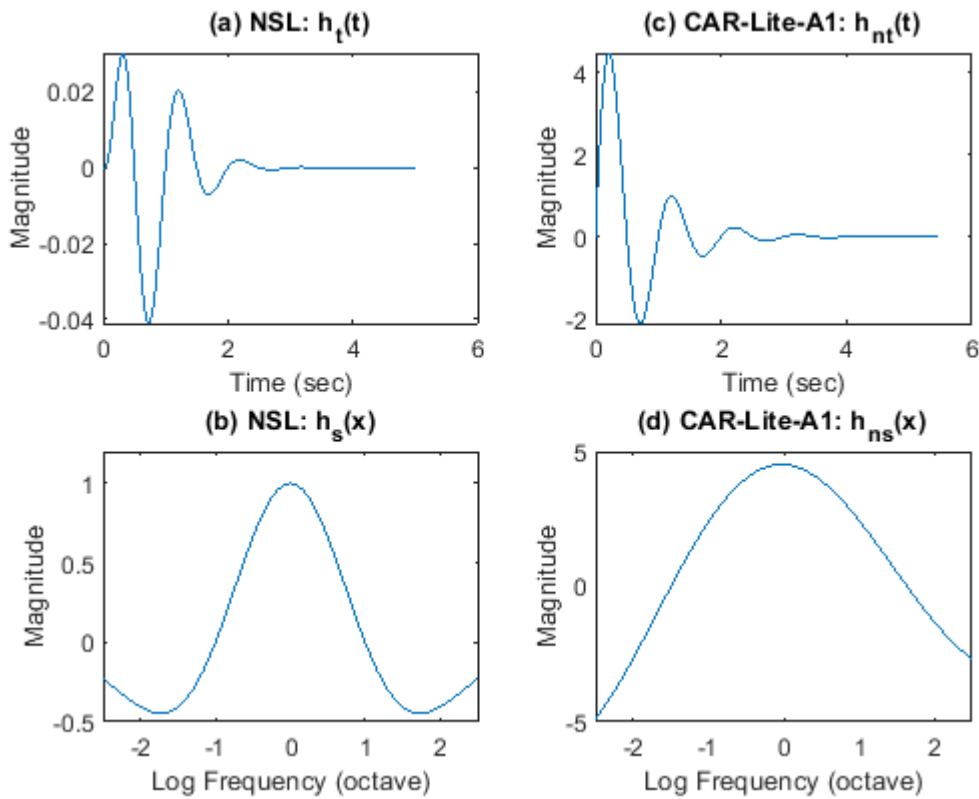


Figure 5-9: Seed functions used in the NSL model with (a) a causal gammatone function defined by equation (5-22) for a 1 Hz rate filter; (b) a non-causal Gabor-like function defined by equation (5-23) for a 1 Hz scale filter; Seed function used in the CAR-Lite-A1 with (c) a causal coupled-form configured asymmetric resonator defined by equation (5-14) for a 1 Hz rate filter; (d) a causal coupled-form configured asymmetric resonator defined by equation (5-15) for a 1 c/o scale filter.

5.2.2.7. Selected Filter Configuration Stability

Filter stability is paramount to indicate the presence of spectral and temporal modulation resonances accurately. Unstable filter operations produce an increase in amplitudes over a short period resulting in an inaccurate representation of modulation resonances. To ensure the stability of the AR, the poles of its transfer function should be contained within the unit circle of a pole-zero (PZ) map. The locations of its poles corresponding to centre velocities of a rate filterbank ranging from 2 Hz to 128 Hz are displayed in Figure 5-10, while its poles

corresponding to centre densities of a scale filterbank ranging 0.25 c/o to 4 c/o are displayed in Figure 5-11. All the pole pairs for the 13 centre velocities of the rate filterbank and 9 centre densities for the scale filterbank are within the unit circle. Having so indicates the stabilities of the rate and scale filterbanks with the aforementioned spectro-temporal modulation ranges.

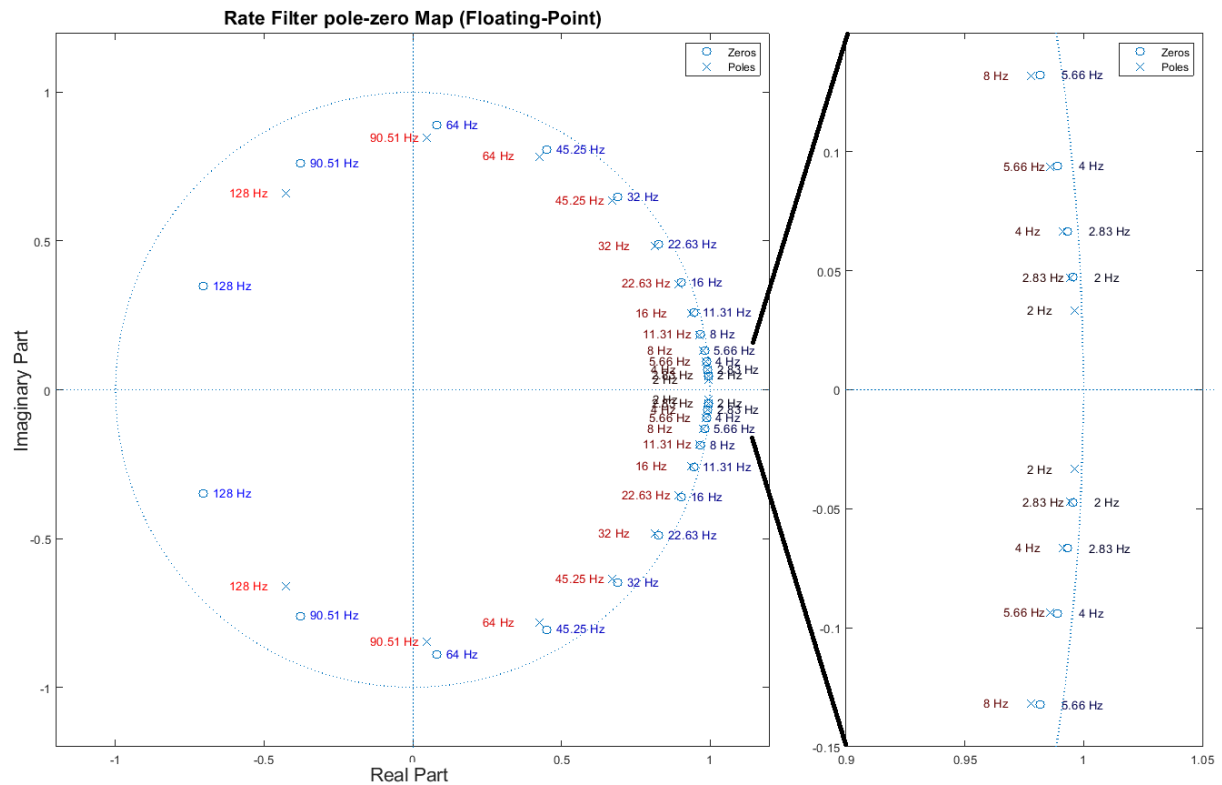


Figure 5-10: Pole-zero map of a 2nd-order AR tuned to 13 centre velocities of temporal modulations ranging from 2 Hz to 128 Hz.

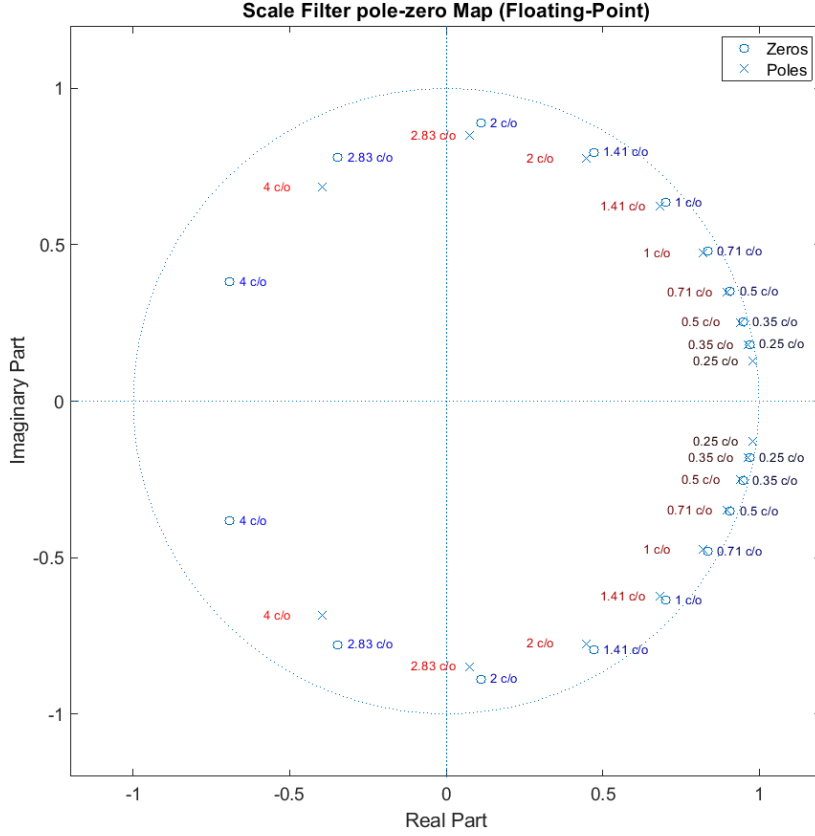


Figure 5-11: Pole-zero map of a 2nd-order AR tuned to 9 centre densities of spectral modulations ranging from 0.25 c/o to 4 c/o.

5.3. Spectro-Temporal Modulation Directionality

The populations of neurons in a mammalian A1 are sensitive to changes across spectro-temporal envelopes. In other words, due to the differential sensitivity of neuronal groups in the A1, these neuronal groups representing receptive fields can detect the movement of spectral-temporal peaks [43]. The changes in the movement are known as directionality of the modulated signal (also known as signal envelope). In the next subsection, this directionality characterised in the NSL model is presented again for clarity (initially presented in chapter 2), and the following sub-section describes the implementation of this directionality in the CAR-Lite-A1 model.

5.3.1. NSL Model

In the NSL model [23], the directionality of a spectro-temporal receptive field (STRF) is defined by:

$$STRF_{\downarrow} = \text{Re}\{h_{IRT}(t_{\downarrow}; \omega, \theta) \cdot h_{IRS}(s; \Omega, \phi)\} \quad (5-16)$$

$$STRF_{\uparrow} = \text{Re}\{h_{IRT}^*(t_{\downarrow}; \omega, \theta) \cdot h_{IRS}(s; \Omega, \phi)\} \quad (5-17)$$

where $STRF_{\downarrow}$ is the spectro-temporal response in the downward direction indicating decreasing temporal envelope velocities and/or spectral envelope densities respectively; $STRF_{\uparrow}$ is the spectro-temporal response in the upward direction indicating increasing temporal envelope velocities and/or spectral envelope densities respectively; h_{IRT} is the impulse response function of the rate filter; h_{IRS} is the impulse response function of the scale

filter; $*$ denotes a complex conjugate; ω is the temporal envelope velocity in time, t_{\downarrow} (\downarrow represents down-sampled t in the cochlear stage); Ω is the spectral envelope density across cochlear section, s ; θ and Φ are characteristic phases of the rate and scale filters respectively. The impulse response functions of the two filters can be represented in a complex form [23]:

$$h_{IRT}(t_{\downarrow}; \omega, \theta) = h_{irt}(t_{\downarrow}; \omega, \theta) + j\hat{h}_{irt}(t_{\downarrow}; \omega, \theta) \quad (5-18)$$

$$h_{IRS}(x; \Omega, \phi) = h_{irs}(s; \Omega, \phi) + j\hat{h}_{irs}(s; \Omega, \phi) \quad (5-19)$$

where $\hat{h}(\cdot)$ is a Hilbert transform or 90° phase shifted version of either a temporal function, h_{irt} , or a spectral function, h_{irs} . These functions are defined as:

$$h_{irt}(t_{\downarrow}; \omega, \theta) = h_t(t_{\downarrow}; \omega) \cos \theta + \hat{h}_t(t_{\downarrow}; \omega) \sin \theta \quad (5-20)$$

$$h_{irs}(x; \Omega, \phi) = h_s(x; \Omega) \cos \phi + \hat{h}_s(x; \Omega) \sin \phi \quad (5-21)$$

where h_t is a gammatone seed function responding to resonances in time, t_{\downarrow} ; h_s is a Gabor-like Gaussian seed function responding to resonances in frequency, x :

$$h_t(t_{\downarrow}) = t_{\downarrow}^2 e^{-3.5t} \sin(2\pi t) \quad (5-22)$$

$$h_s(x) = (1 - x^2) e^{-x^2/2} \quad (5-23)$$

5.3.2. CAR-Lite-A1 Model

Aside from the change in the temporal and spectral seed functions, from h_t and h_s to h_{nt} and h_{ns} respectively, another consideration is the inclusion of the calculation of their respective characteristic phases, θ , and, Φ . This calculation involves applying an arc-tangent on the division between the output values of Hilbert transformed seed function, \hat{h} , and seed function, h . Here, h refers to either h_t or h_s . In the NSL model, this phase information is required to resynthesize the original sound input stimulus from the spectro-temporal envelope information extracted from the rate and scale filterbanks. This exercise is done to evaluate the fidelity of these extracted cues to the original input stimulus. However, as my objective is to capture timbre cues from a hardware model, input stimulus re-synthesis is not required.

Furthermore, timbre is representable by envelope information and independently by phase information [15], [44], [45]. In other words, averaging envelope responses over time can represent timbre and so, the calculation of phase information is omitted from the CAR-Lite-A1 model. Doing so removes the need for implementing a lookup table for arc-tangent approximation as well as the need for a computationally intensive division operation on FPGA. This omission means that the temporal function, h_{irt} , and spectral function, h_{irs} , need not be implemented and the complex-form impulse responses, h_{IRT} , and, h_{IRS} , can be formed directly with h_{nt} and h_{ns} . As a result, the downward and upward STRF functions are defined as:

$$STRF_{\downarrow}(s, t_{\downarrow}; \omega, \Omega) = (h_{nt} + j\hat{h}_{nt})(h_{ns} + j\hat{h}_{ns}) \quad (5-24)$$

$$STRF_{\uparrow}(s, t_{\downarrow}; \omega, \Omega) = (h_{nt} - j\hat{h}_{nt})(h_{ns} + j\hat{h}_{ns}) \quad (5-25)$$

Expanding the two equations result in:

$$STRF_{\downarrow} = h_{nt} \cdot h_{ns} + jh_{nt} \cdot \hat{h}_{ns} + j\hat{h}_{nt} \cdot h_{ns} - \hat{h}_{nt} \cdot \hat{h}_{ns} \quad (5-26)$$

$$STRF_{\uparrow} = h_{nt} \cdot h_{ns} + jh_{nt} \cdot \hat{h}_{ns} - j\hat{h}_{nt} \cdot h_{ns} + \hat{h}_{nt} \cdot \hat{h}_{ns} \quad (5-27)$$

The terms in equations (5-26) and (5-27) can then be separated and characterised based on the real (Re) and imaginary (Im) components that are contained within them:

$$Re\{STRF_{\downarrow}\} = h_{nt} \cdot h_{ns} - \hat{h}_{nt} \cdot \hat{h}_{ns} \quad (5-28)$$

$$Re\{STRF_{\uparrow}\} = h_{nt} \cdot h_{ns} + \hat{h}_{nt} \cdot \hat{h}_{ns} \quad (5-29)$$

$$Im\{STRF_{\downarrow}\} = h_{nt} \cdot \hat{h}_{ns} + \hat{h}_{nt} \cdot h_{ns} \quad (5-30)$$

$$Im\{STRF_{\uparrow}\} = h_{nt} \cdot \hat{h}_{ns} - \hat{h}_{nt} \cdot h_{ns} \quad (5-31)$$

In the NSL model, the imaginary component is omitted. However, for the CAR-Lite-A1 model, the imaginary component is retained for the calculation of neuron directionality as they can be represented by real-valued signals as is presented in subsection 5.3.2.1.

The downward and upward spectro-temporal modulation envelope drifts, r_{\downarrow} and r_{\uparrow} , can be characterised by convolving a down-sampled 2D time-frequency matrix of inner hair cell (IHC) values [using equation (5-1)] with either the real or imaginary components of the 4D STRF filterbanks as follow:

$$Re\{r_{\downarrow}\} = IHC_{\downarrow}(s, t_{\downarrow}) \otimes Re\{STRF_{\downarrow}(s, t_{\downarrow}; \omega, \Omega)\} \quad (5-32)$$

$$Re\{r_{\uparrow}\} = IHC_{\downarrow}(s, t_{\downarrow}) \otimes Re\{STRF_{\uparrow}(s, t_{\downarrow}; \omega, \Omega)\} \quad (5-33)$$

$$Im\{r_{\downarrow}\} = IHC_{\downarrow}(s, t_{\downarrow}) \otimes Im\{STRF_{\downarrow}(s, t_{\downarrow}; \omega, \Omega)\} \quad (5-34)$$

$$Im\{r_{\uparrow}\} = IHC_{\downarrow}(s, t_{\downarrow}) \otimes Im\{STRF_{\uparrow}(s, t_{\downarrow}; \omega, \Omega)\} \quad (5-35)$$

where \otimes denotes convolution.

In the next subsection, a representation of the Hilbert transform with IIR filters is presented to generate A1 neuron directionality response, for FPGA implementation along with the AR representing rate and scale filters.

5.3.2.1. Characterising the Hilbert Transform

A sinusoid contains two frequency components: a positive frequency component and a negative frequency component. They are a sum of equal but opposite circular motion and can be projected on a frequency spectrum. Moreover, they can be analytically derived from Euler's identity [46]:

$$A \cos(\theta) = \frac{A(e^{j\theta} + e^{-j\theta})}{2} \quad (5-36)$$

$$A \sin(\theta) = \frac{A(e^{j\theta} - e^{-j\theta})}{2j} \quad (5-37)$$

where A is the amplitude; $j\theta$ is a positive frequency component; $-j\theta$ is a negative frequency component; θ is defined by frequency, ω , and phase, ϕ , of a sinusoid or more specifically by the equation, $\theta = \omega t + \phi$.

In demodulation filtering to extract the envelope of an input signal, filtering the negative frequency, $Ae^{j(-\omega t + \phi)}$, does not result in any information loss in the sinusoid. Hence, the positive frequency, $Ae^{j(\omega t + \phi)}$ of a real sinusoid, $A \cos(\omega t + \phi)$, is sufficient in representing the envelope of a signal. Furthermore, the NSL uses complex signal filtering to represent frequency directionality responses of neurons. In theory, this filtering involves the use of an imaginary coefficient, which is practically attainable by the 90° phase-shift of the real sinusoid. This results in a practical representation of a complex signal, known as an analytic signal, where no negative frequency is present [47]:

$$Ae^{j(\omega t + \phi)} = A \cos(\omega t + \phi) + jA \sin(\omega t + \phi) \quad (5-38)$$

Since the cosine and sine terms are real terms, the magnitude of the positive frequency is doubled.

To generate an analytic signal representation of a signal containing multiple sinusoids at multiple frequencies, the signal is combined with the 90° phase shift of itself based on equation (5-38) for all frequencies. This phase shift is attainable with the Hilbert transform of the signal. In other words, the Hilbert transform of a signal shifts the phases of every Fourier frequency component of an input signal by 90° [48]. In the time domain, the Hilbert transform, $\mathcal{H}[\dots]$, is defined as the convolution of an input signal, $x(t)$ with $1/(\pi t)$:

$$\mathcal{H}[x(t)] = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x(\tau)}{t - \tau} d\tau \quad (5-39)$$

The Fourier transform, $\mathcal{F}\{\dots\}$, of the Hilbert transform of $x(t)$ is:

$$\mathcal{F}\{\mathcal{H}[x(t)]\} = j\omega X(\omega) \quad (5-40)$$

where $j\omega$ is an imaginary term of a frequency, ω , which also denote a 90° phase shift of the frequency.

The use of analytic signals is essential in detecting amplitude envelopes in a signal [47]. In the NSL model, the output signals are analytic signals that are a result of only the real component of the STRF equations defined by (5-16) and (5-17) convolved with the 2D IHC input signal. In other words, the STRF equations of (5-18) and (5-19) primarily contain the impulse response of a rate filter, h_{irt} , an impulse response of a scale filter, h_{irs} , as well as

their respective Hilbert-transformed 90° phase-shifted signal, \hat{h}_{irt} and \hat{h}_{irs} . Here, the h_{irt} and h_{irs} are considered two separate real components of an analytic signal and \hat{h}_{irt} and \hat{h}_{irs} are their respective imaginary components. Analytic signals also define the outputs of the CAR-Lite-A1 model defined by equations (5-32) to (5-35). However, the outputs of both the real and imaginary components are used in the CAR-Lite-A1 instead of only the real component in the NSL model.

From a signal processing perspective, the Hilbert transform can be realised as a non-causal, time-invariant filter. In other words, it can be applied to an input signal but can only be initiated when the entire signal to be analysed is acquired. However, due to its non-causality, it cannot be implemented in hardware for real-time applications. Alternatively, the Hilbert transform can be applied to segmented parts of a temporal signal as they become available. In terms of its application, the Hilbert transform has been used in the construction of synthesised sounds for specific musical instruments [49]. It can also be used in image [50] and video [51] processing.

For hardware implementation, the alternative to Hilbert transform is to use a causal all-pass filter, which allows all frequency components of an input signal to pass through the filter with equal gain [52]. The 90° phase shift occurs specifically at its cut-off frequency at -3dB. The phase shifts of all other frequency components occur relative to the single frequency component at the -3-dB point, which means that the quadrature (90°) phase difference between an input and an output signal occurs only at a single central velocity point for a rate filter and a single centre density point for a scale filter as opposed to all Fourier components in a Hilbert transform. This attribute complements the respective real component that is a bandpass filtered signal at the same specific point of interest. Furthermore, the utilisation of an all-pass filter can generate an analytic output signal in place of a complex output signal with a Hilbert transform in the NSL model. Therefore, the all-pass filter is used in the CAR-Lite-A1 model to generate directional neuron responses based on equations (5-32) to (5-35).

The all-pass filter is configurable with a quadrature mirror Hilbert transformer (QMHT) [53], which uses a low-pass filter (LPF) and a high-pass filter (HPF) complementary pair in parallel as illustrated in Figure 5-12(a). Using first-order IIR filters, the phases at the cut-off frequency are at -45° and 45° for the LPF and HPF respectively, which results in an absolute differential phase of 90° at -3 dB. Here, the LPF branch is regarded as the real part, and the HPF is regarded as the imaginary branch of an analytic signal.

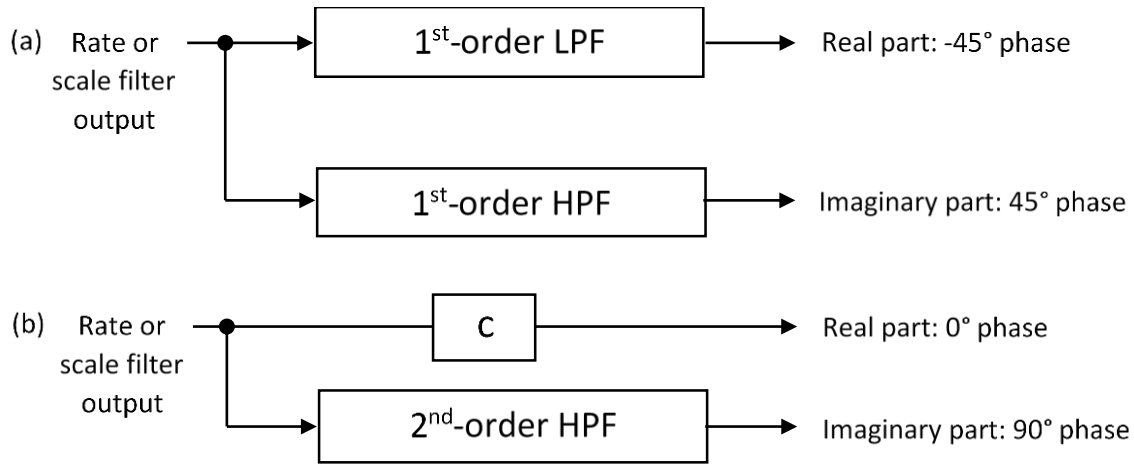


Figure 5-12: Quadrature mirror Hilbert transformer (QMHT) configuration to achieve a 90° absolute phase difference in an analytic signal using a) complementary parallel pair of 1st-order LPF-HPF; b) 2nd-order HPF in the imaginary branch and the real branch scaled by a constant, c, to manually regulate its amplitude with respect to the signal in the imaginary branch.

For low centre velocities such as a 2 Hz rate filter, the absolute differential phase generated from a 1st-order QMHT is approximately 90° as observed in Figure 5-13. However, the phases are not uniform at 90° for different centre velocities – at 32 Hz, the phase falls to approximately 60° as projected in Figure 5-14. At higher velocities, the phase difference becomes significantly smaller, especially at centre velocities from 32 Hz to 128 Hz for the rate filterbank, as well as centre densities from 2 c/o to 4 c/o for the scale filterbank. Therefore, a 1st-order LPF-HPF parallel configuration is unable to generate the desired quadrature phase for the entire range of centre velocities and centre densities.

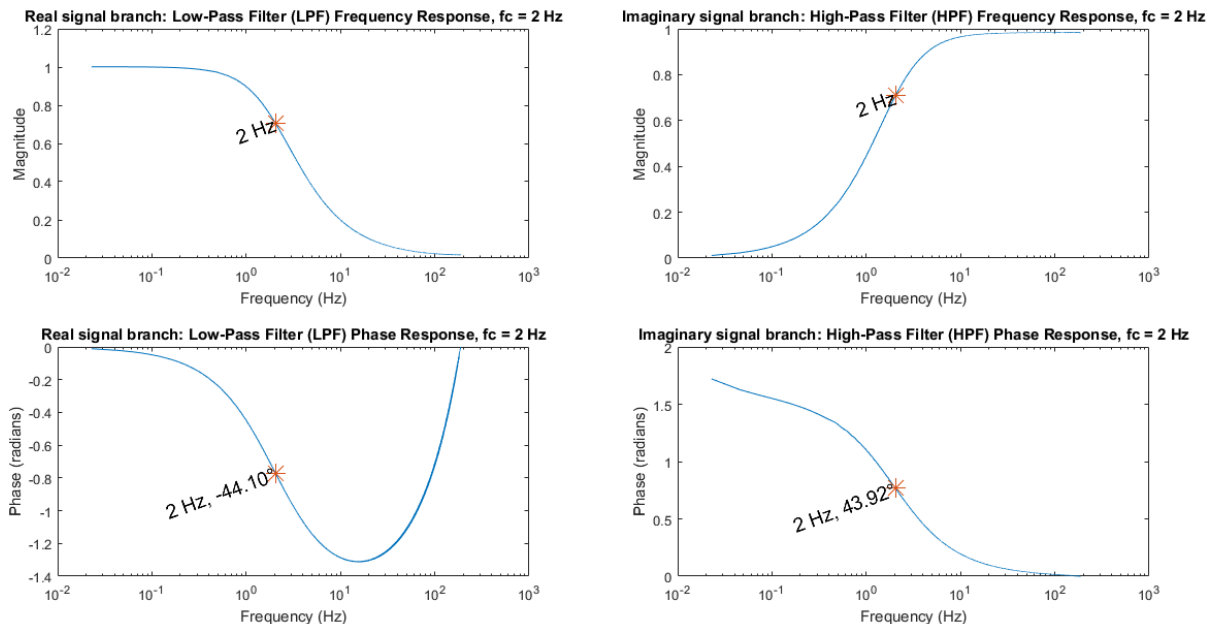


Figure 5-13: Gain and phase responses of a 1st-order IIR low-pass filter (LPF) and a high-pass filter (HPF) configured as a quadrature mirror Hilbert transformer (QMHT), generating an absolute differential phase of approximately 90° ($\approx |-44.10^\circ - 43.92^\circ|$) at 2 Hz centre velocity of a rate filter.

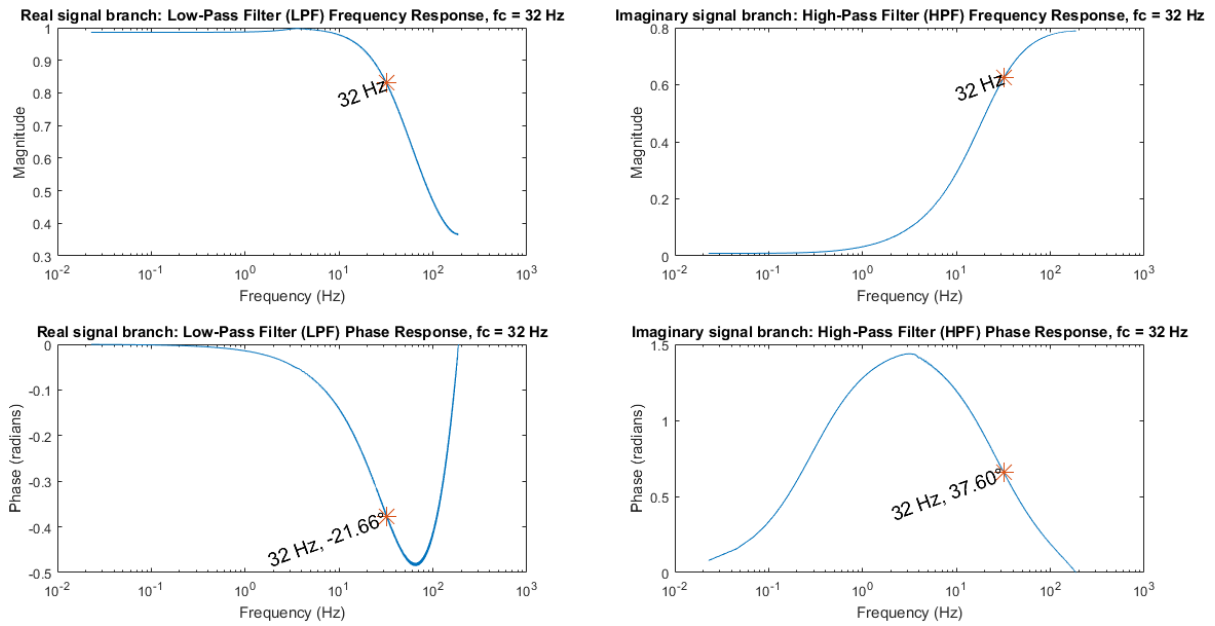


Figure 5-14: Gain and phase responses of a 1st-order IIR LPF and an HPF configured as a QMHT, generating an absolute differential phase of approximately 60° ($\approx |-21.66^\circ - 37.60^\circ|$) at 32 Hz centre velocity of a rate filter.

Using either a single second-order HPF or a single LPF results in 90° phase shifts at cut-off frequencies for all centre velocities and all centre densities of the rate and scale filterbanks respectively. Hence, a modified QMHT configuration can be attained as illustrated in Figure 5-12(b) by replacing the 1st-order IIR HPF with a second-order standard IIR biquadratic HPF on the imaginary branch and by replacing the 1st-order IIR LPF with a scalar constant to manually regulate the amplitude of the output signal on the real branch. Similarly, the reverse situation can be applied as well where the HPF is replaced with a scalar constant on the imaginary branch, and the 1st-order LPF is replaced with a second-order LPF on the real branch. Using either of these two configurations results in an absolute differential phase of approximately 90° at the cut-off frequencies for all centre velocities and centre densities. An example of this quadrature (90°) phase is shown for the rate filter with a 128 Hz centre velocity in Figure 5-15 and the scale filter with a 4 c/o centre density in Figure 5-16. The former is generated using a modified QMHT configured with only a second-order HPF at the imaginary branch and a scalar constant at the real branch, while the latter is generated with a modified QMHT configured with only a second-order LPF at the real branch and a scalar constant on the imaginary branch. For the CAR-Lite-A1 model, the former design is used henceforth, to represent the QMHT, to generate the upward and downward A1 neuron directional response.

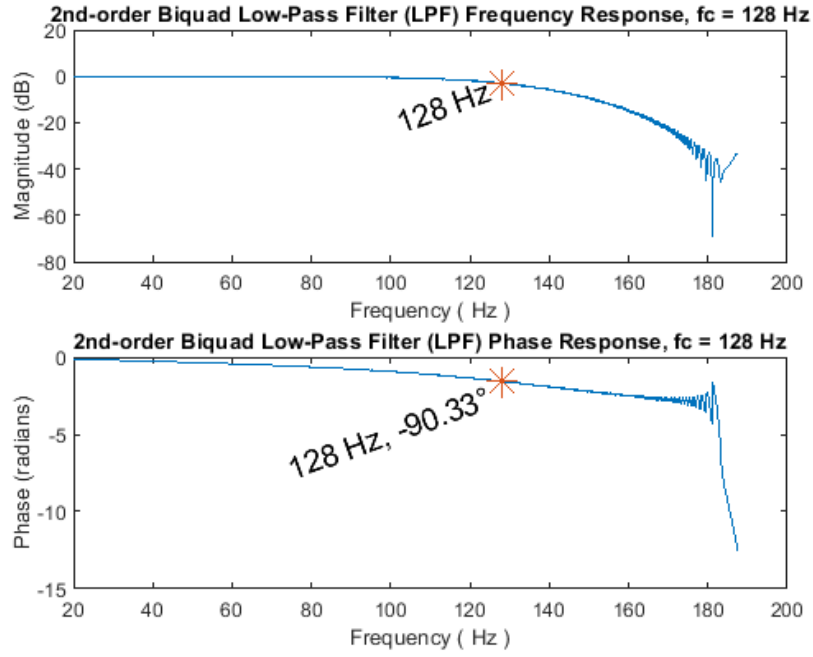


Figure 5-15: Gain and phase responses of a modified QMHT using a 2nd-order IIR biquadratic LPF on the imaginary branch and a scalar constant of 0.7071 on the real branch, generating an absolute differential phase of approximately 90° (precisely 89.08°) at 128 Hz centre velocity of a rate filter.

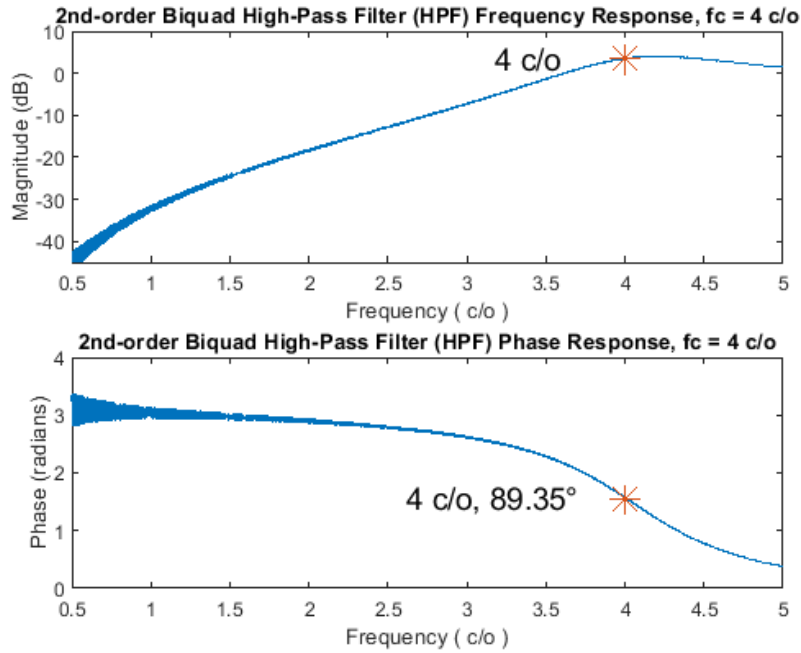


Figure 5-16: Gain and phase responses of a modified QMHT using a 2nd-order IIR biquadratic HPF on the real branch and a scalar constant of 0.7071 on the imaginary branch, generating an absolute differential phase of approximately 90° (precisely 90.33°) at 4 c/o centre density of a scale filter.

5.4. Circuit

Figure 5-17 presents a cutaway of the CAR-Lite-A1 model circuit from Figure 5-1 for one cochlear section. A single cochlear output (shown in a blue-dashed box in Figure 5-17) is fanned out to 13 parallel branches corresponding to 13 centre velocities of the rate filterbank, depicted in the red-dashed box in Figure 5-17. Each rate filter output is fanned out

to 9 parallel branches corresponding to 9 centre densities of the scale filterbank, represented in a green-dashed box in Figure 5-17. The outputs of the rate and scale filterbanks feed into a neuron directional filterbank, which generates pairs of 2D time-frequency upward and downward responses for all mixed combinations between one (of 13) centre velocity and one (of 9) centre density. A total of 234 (13 centre velocities \times 9 centre densities \times 2 directions) 2D time-frequency images are generated using either the real component or the imaginary component of the analytic signal defined by equations (5-32) to (5-35). Figure 5-18 separately illustrates the circuits of the real and imaginary components of the analytic signal for any combination of one rate filter centre velocity, and one scale filter centre density.

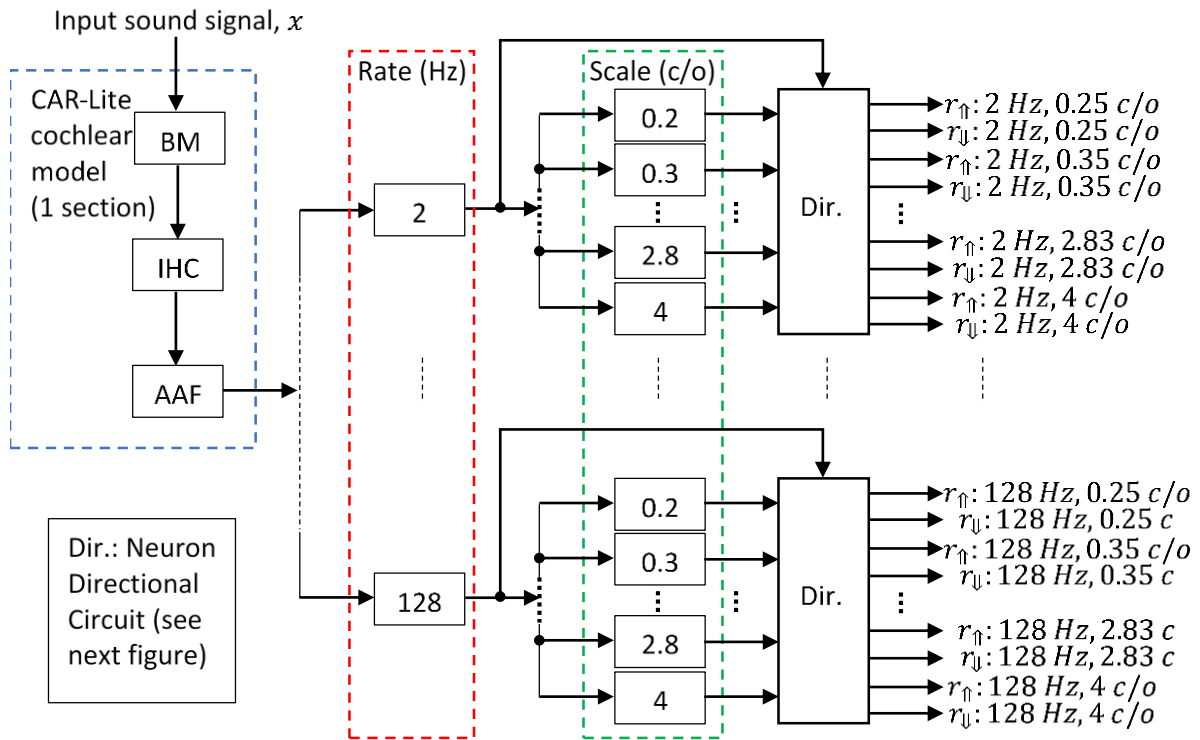


Figure 5-17: CAR-Lite-A1 model circuit showing one cochlear section (blue-dashed box) connected to a rate filterbank (red-dashed box), a scale filterbank (green-dashed box), and a neuron directional filterbank (depicted as “Dir.”). The output of “Dir.” generates responses from either the real (part of an analytic signal) circuit [Figure 5-18(a)] or the imaginary (part of an analytic signal) circuit [Figure 5-18(b)].

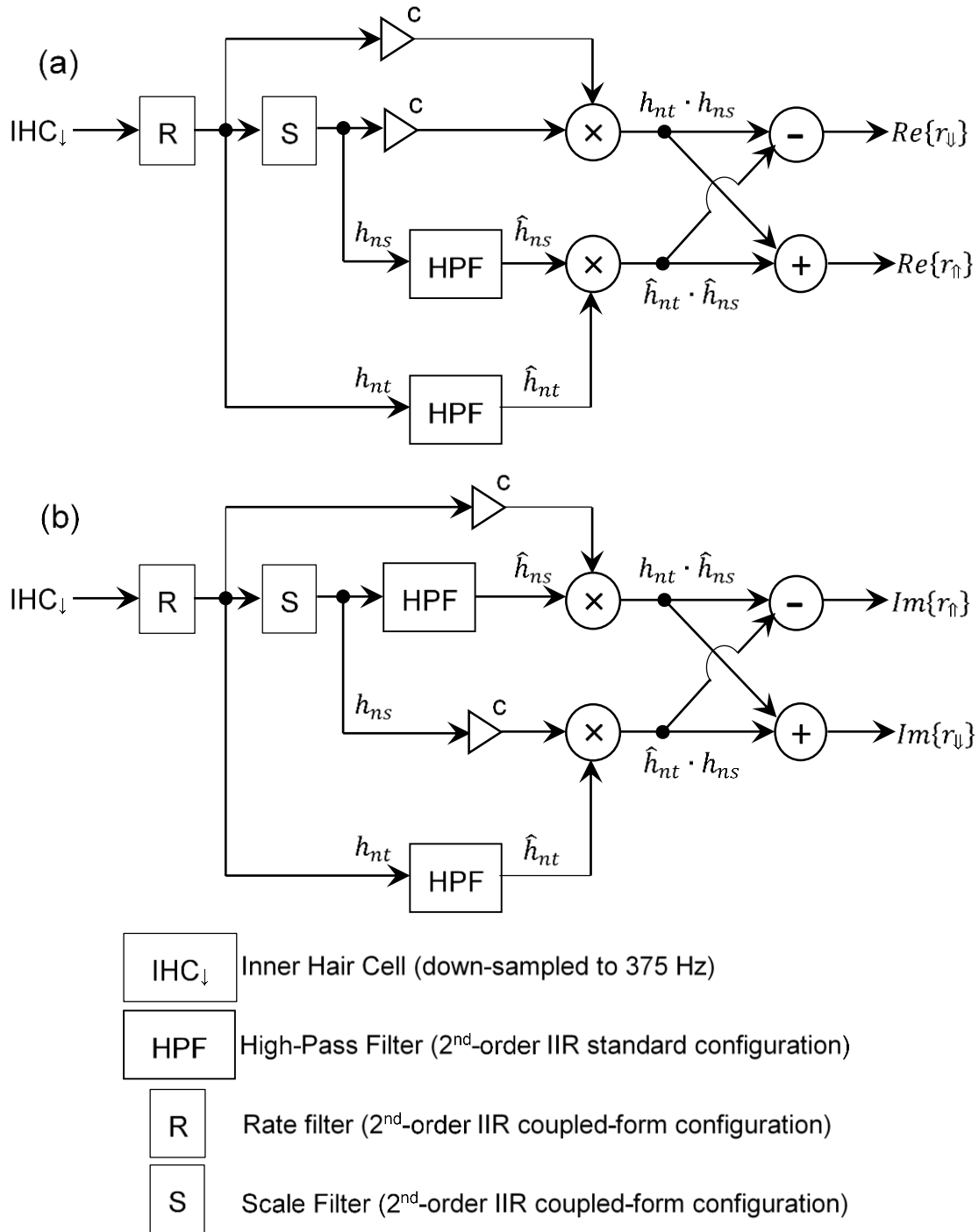


Figure 5-18: Cascade of rate and scale filters and quadrature mirror Hilbert transformers (QMHT) to generate A1 neuronal directionality for (a) the real and (b) imaginary components of the analytic signal defined by equations (5-32) to (5-35).

5.5. Fixed-Point Implementation

The floating-point implementation of the CAR-Lite-A1 model is translated to a fixed-point implementation to accommodate the model on an FPGA. Doing so reduces the bit widths and therefore the memory utilised for all coefficients as well as input and output samples represented in 64 bits floating-point format to varying sizes below 32 bits in fixed-point. Each sample of the input and output signals for every stage in the model are 16 bits wide in fixed-

point. All the coefficients of the 108 ARs of the CAR-Lite cochlear model are represented with 8 bits in fixed-point. Since the coefficients of the AR for the rate and scale filterbanks are both positive and negative, they are set to 9 bits in fixed-point. The extra bit at the most significant bit (MSB) accommodates the sign of the coefficients. 16 bits coefficients characterise the high-pass filter (HPF) used in the QMHT module.

5.5.1. Filter Stability

The stability of the filters must be studied to affirm the set bit widths of the coefficients of the AR and HPF in the A1 stage of the CAR-Lite-A1 model. The poles of the 9 bits coefficients of the AR for rate and scale filterbanks are all contained within the unit circle of the pole-zero maps as observed in Figure 5-19 and hence, the fixed-point implementation of these two filterbanks are stable. However, the placement of zeros of the ARs for 7 of the 13 centre velocities of the rate filters from 2 Hz to 11.31 Hz and a single centre density of 0.35 c/o from the scale filterbank, are outside the unit circle, which indicates that the inverse of the AR transfer functions of the centre velocities and centre density mentioned above is unstable [40]. In other words, for an invertible filter, the input signal can be recovered by applying the inverse of the rational transfer function of the filter to the output signal. However, for the aforementioned non-invertible filters, the input signal cannot be recovered by the same process.

In the NSL model, the rate and scale filters are invertible, which means that the input signal can be reconstructed. The degree of similarity between the reconstructed and the original input indicates a measure of fidelity and the capability of the filters to manipulate timbre cues [23], [54]. In contrast, the fixed-point implementation of the CAR-Lite-A1 model is unable to reconstruct an input signal due to the non-invertibility characteristic of the filters. Nonetheless, in the design of the CAR-Lite-A1 model, the stability of the ARs is prioritised over their invertibility. Instead of using reconstructed signals, an alternative measure of the fidelity of input signals is used, as demonstrated by the classification of musical instruments in chapter 7. Therefore, no modifications are necessary to alleviate these non-invertible characteristics of the ARs.

The fixed-point implementation of the HPFs with 16 bit coefficients in the QMHT rate and QMHT scale filterbanks are all stable as they have poles within the unit circle, as observed in Figure 5-20. For coefficients with bit widths lower than 10 bits, the HPFs become unstable as their poles are outside the unit circle, especially for low centre velocities of 2 Hz and 2.83 Hz of the QMHT rate filterbank as observed in Figure 5-21. From 10 bits to 15 bits, poles reside on the unit circle resulting in undamped response, which indicates resonances that do not decay. 16 bits coefficients are selected to ensure all the poles in the filterbank are within the unit circle resulting in the decay of resonances over time. With 16 bits coefficients, the zeros of all 13 centre velocities and 9 centre densities of the HPFs in the QMHT rate and QMHT scale filterbanks are outside the unit circle, which means they are all non-invertible. As mentioned in the preceding paragraph, this is an acceptable characteristic as there is no need to recover the input signal from the output signal.

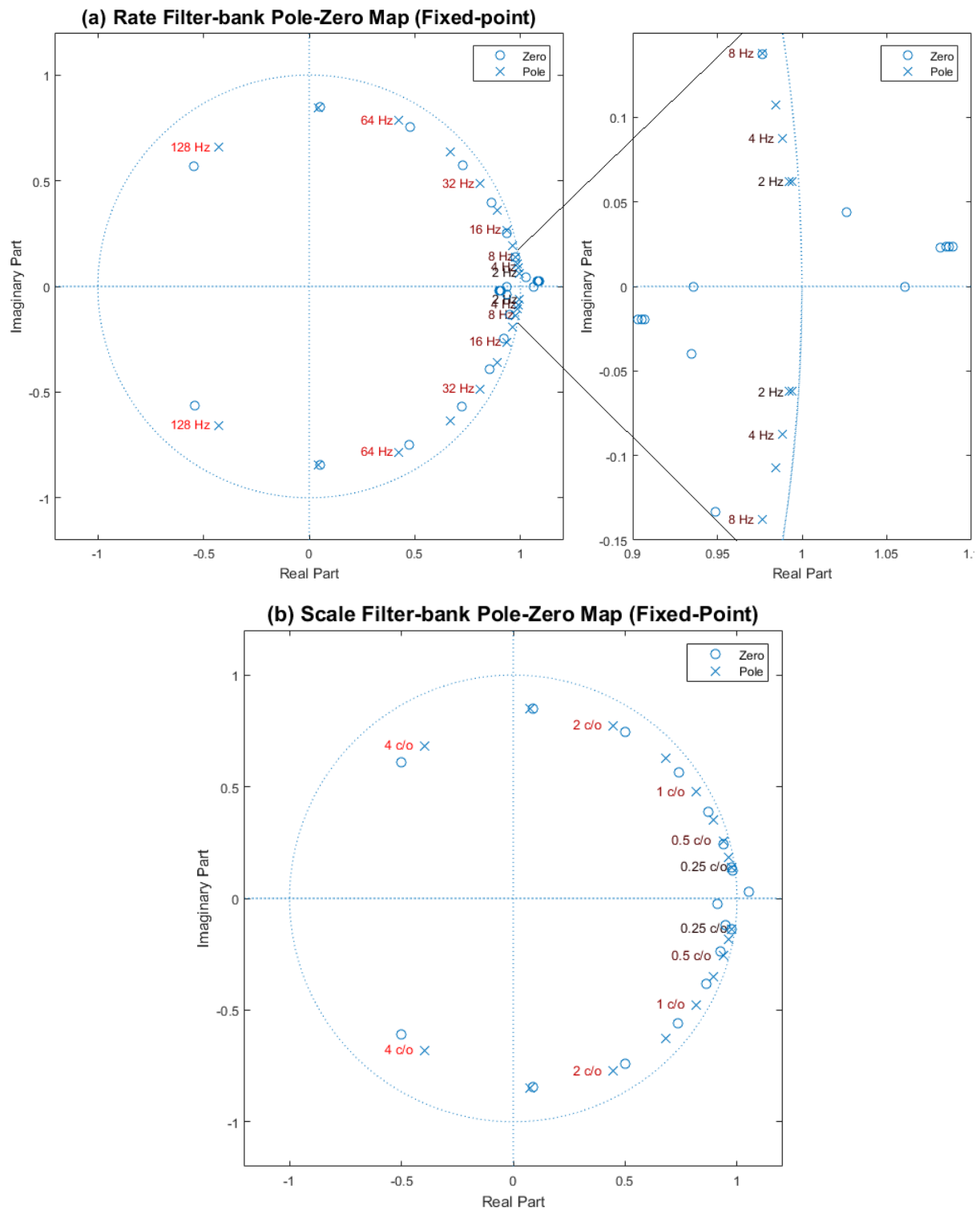
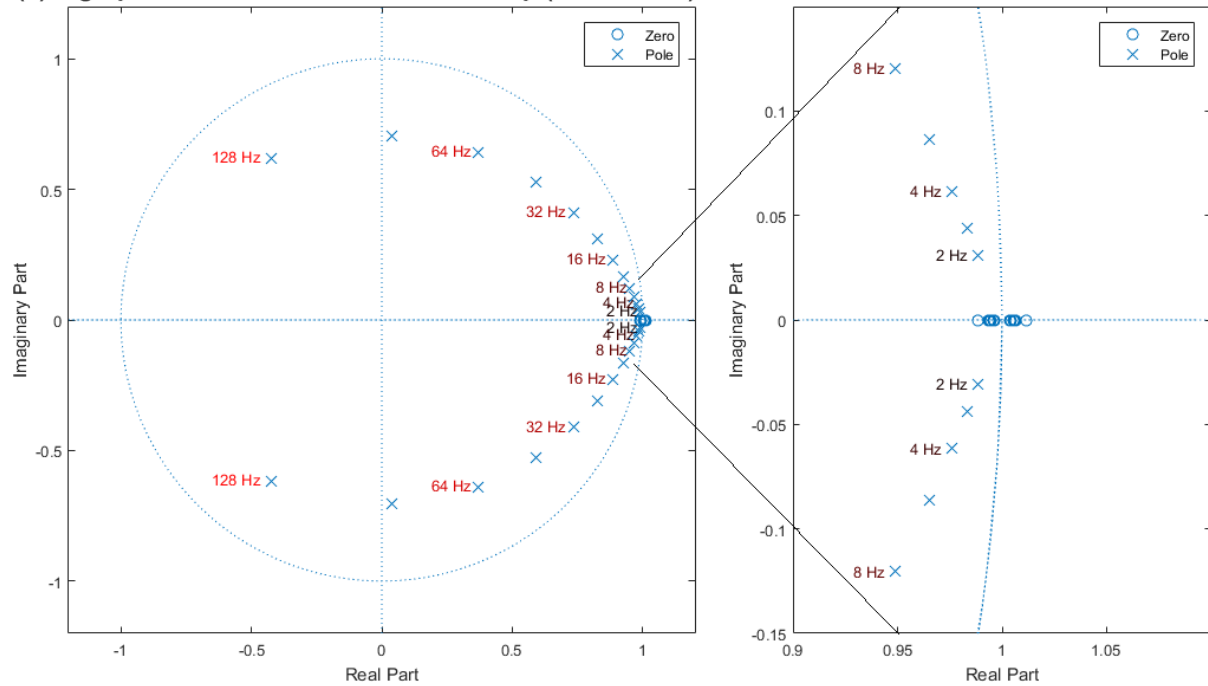


Figure 5-19: Pole-zero (PZ) map of a fixed-point implementation of a 2nd-order AR with 9 bits coefficients used in (a) the rate filterbank and (b) the scale filterbank.

(a) High-pass Rate Filterbank Pole-Zero Map (Fixed-Point)



(b) High-pass Scale Filterbank Pole-Zero Map (Fixed-Point)

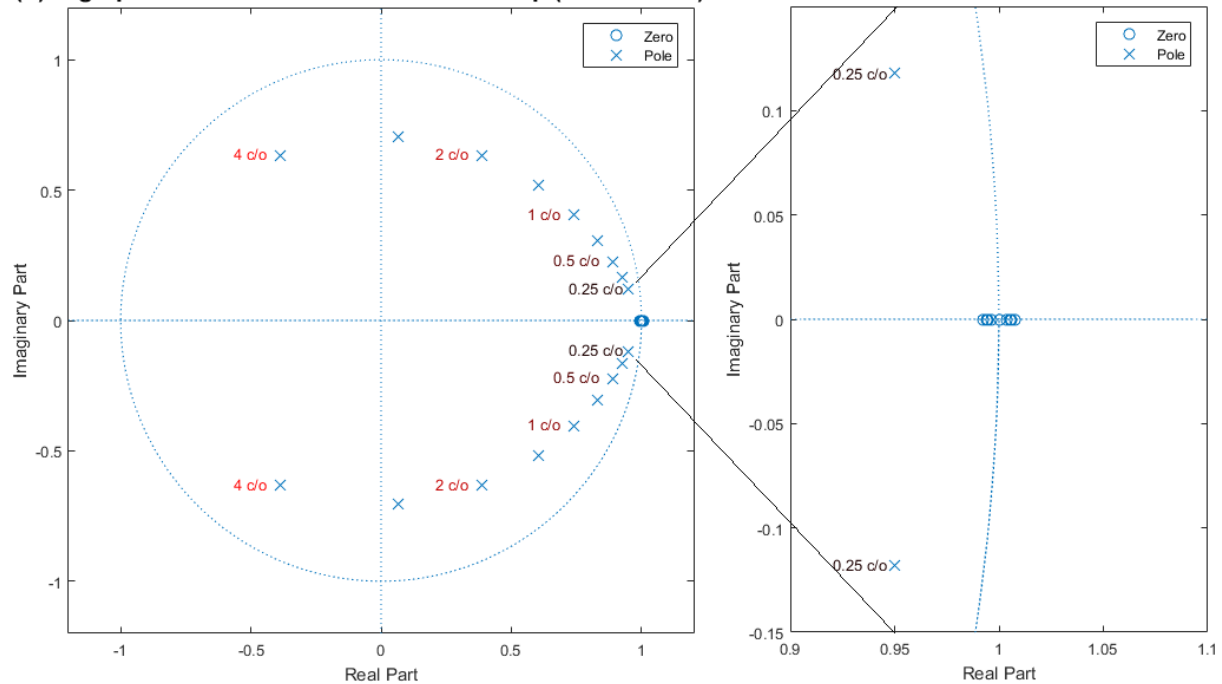


Figure 5-20: Pole-zero (PZ) map of a fixed-point implementation of a 2nd-order high-pass filter (HPF) with 16 bits coefficients used in (a) the QMHT rate filterbank and (b) the QMHT scale filterbank.

High-pass Rate Filterbank Pole-Zero Map (Fixed-Point)

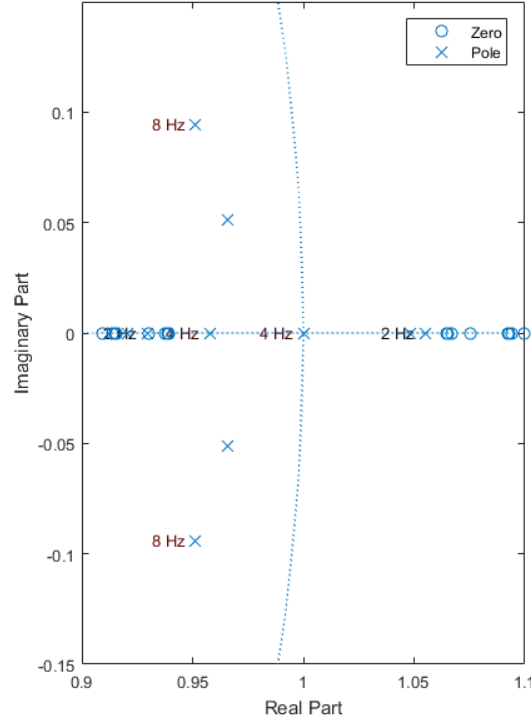


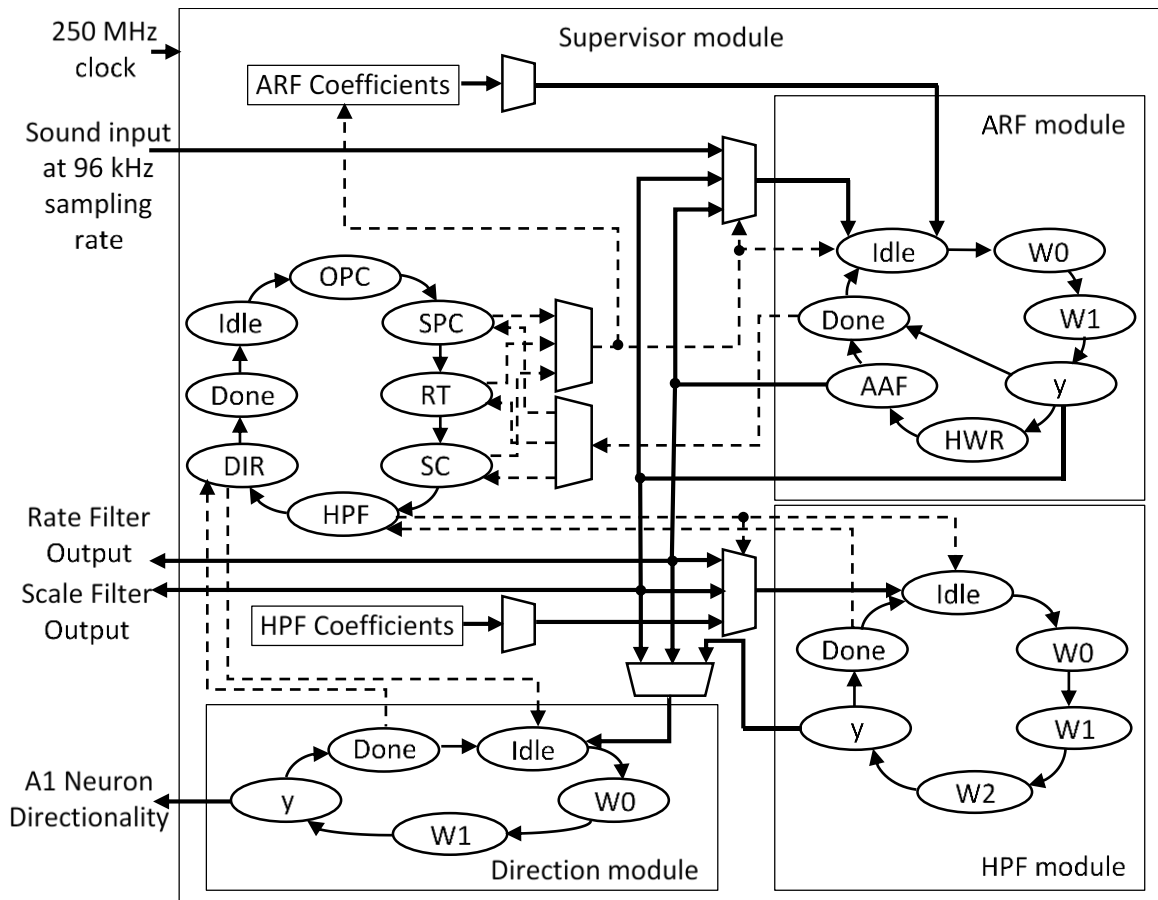
Figure 5-21: Magnified region of the unit circle (dotted blue vertical arc) in the pole-zero (PZ) map of a fixed-point implementation of a 2nd-order high-pass filter (HPF) with 8 bits coefficients used in the QMHT rate filterbank.

5.6. FPGA Implementation

This section presents the implementation of the CAR-Lite-A1 model on an FPGA. An Altera Cyclone V starter kit with a 5CGXFC5C6F27C7N FPGA chip is used with a global clock rate of 250 MHz and a 96 kHz audio sampling rate. The fixed-point implementation, which is interchangeably known as the hardware model of the CAR-Lite-A1 model, is implemented via SystemVerilog on Altera Quartus development software application. Instead of implementing a rate filterbank with 13 centre velocities and a scale filterbank with 9 centre densities on FPGA, only a single 4 Hz rate filter and a single 1 c/o scale filter along with neuron directionality associated with these filters are implemented. This implementation is done for comparing FPGA computational resource utilisation between the CAR-Lite-A1 model and other auditory models fitted with envelope extraction capabilities based on a single set of envelope extraction filter.

Figure 5-22 displays the architecture of the CAR-Lite-A1 model on FPGA with four core modules. These modules operate using time-division multiplexing, whereby each invoked module operates modularly by transmitting its output samples to the next module in the processing queue. Once a module finishes processing an input sample, it continues to process the next input sample when it becomes available. If an input sample is unavailable, the module remains inactive. Hence, with time-division multiplexing, the invoked modules ideally work in parallel when necessary as well as modularly as observed in Figure 5-23. The supervisor module foresees and regulates the time-division multiplexing operations using a combination of octave processing control (OPC) and section processing control (SPC) as explained in detail in chapter 3. In other words, the supervisor module is a top module that decides which octave and which section to process in the CAR-Lite cochlear model, its output down-sampling as well as the invocation of the A1 segment comprising the rate,

scale, and neuron directional filters respectively. The next subsection provides details of the operation of the modules.



OPC: Octave Processing Control RT: Rate filter control Wn: Filter internal variables
 SPC: Section Processing Control SC: Scale filter control n = [0,1,2]
 ○: Start / End / Algorithm State y: Filter output HPF: High-pass filter control
 ▢: Multiplexer / Demultiplexer AAF: Anti-Aliasing Filter DIR: A1 Directional Filter Control
 -----: Control signal pathway —: Data pathway —: Signal pathway

Figure 5-22: Architecture of the CAR-Lite-A1 model implemented on an FPGA with fixed-point arithmetic.

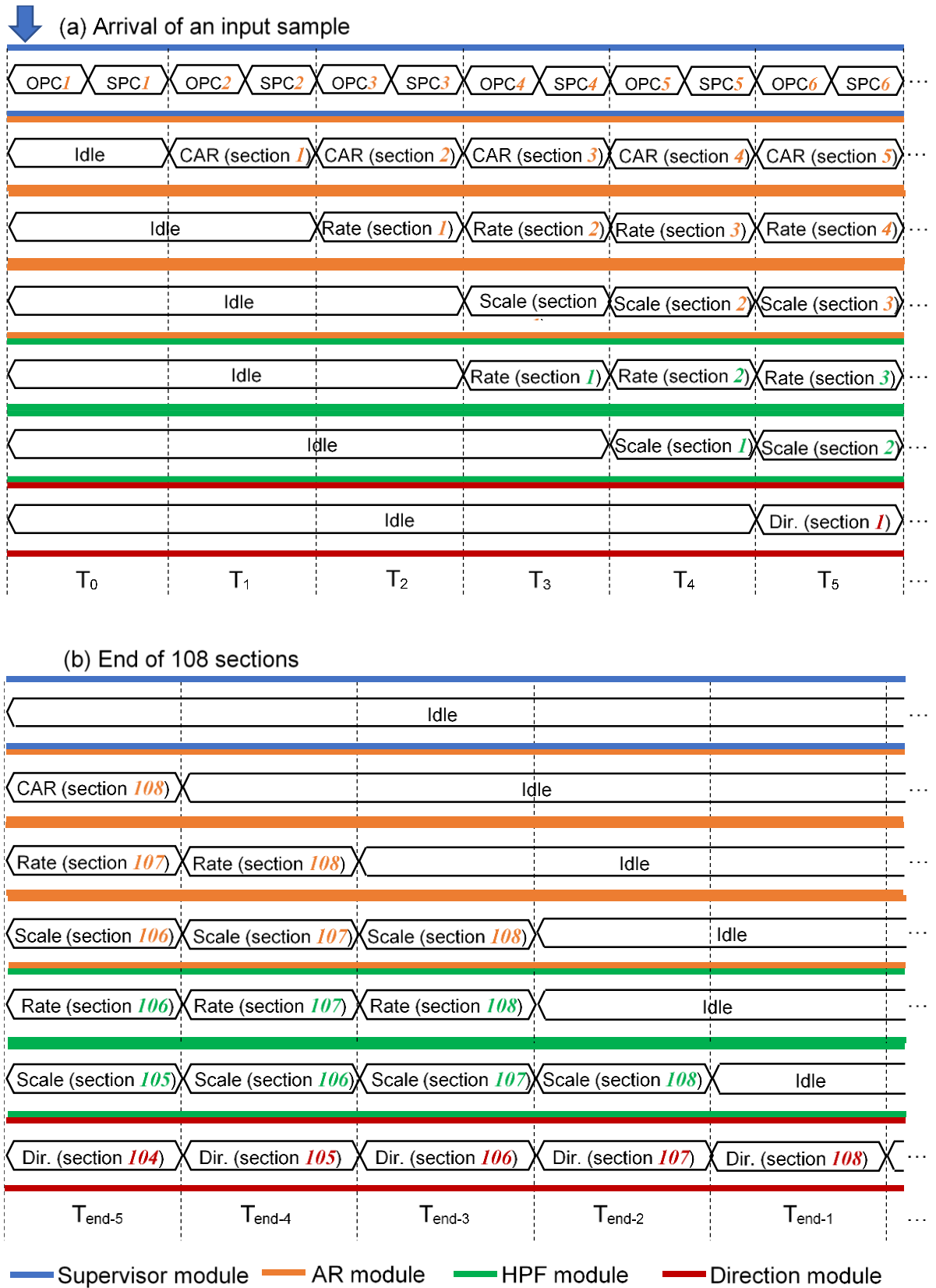


Figure 5-23: FPGA vector waveform of the AR, HPF, and Direction (Dir.) modules as part of the CAR-Lite-A1 model operating at (a) the arrival of an input sample; (b) at the end of 108 cochlear sections. Time duration T_n is based on 22 clock cycles (latency of 9.5 μ sec) from the AR module operating all 22 states – the maximum number of states corresponding to the AR module out of all four modules.

5.6.1. Operation of Modules

The four core modules in the FPGA implementation of the CAR-Lite-A1 model are *Supervisor*, *AR*, *HPF*, and *Direction*. The signal processing operations of the four modules obey the properties of convolution, including commutation, association, and distribution [55]. The *AR* module hosts the equations of a 2nd-order AR. At startup, the *AR* module is instantiated three times with each cloned module characterising a CAR filter of the basilar membrane (BM), a rate filter and a scale filter. The *supervisor* module determines the module to run. Along with the equations of the CAR filter, the *AR* module also hosts basilar membrane velocity (BMd) and inner hair cell (IHC) equations. The BMd and IHC equations are processed only in the instantiated *AR* module invoked as the CAR filter. The BMd and IHC computations are bypassed for the *AR* module instantiated as rate and scale filters.

From the instantiated *AR* module operating as a CAR filter, its corresponding IHC output sample is used as input to the A1 model via the supervisor module. As this CAR filter-based *AR* module continues to process the next cochlear section, its output IHC sample is down-sampled to 375 Hz, forming IHC_{\downarrow} as defined by equation (5-1) that is output to the *supervisor* module. The *supervisor* module invokes the second instantiated *AR* module that operates as the rate filter, with IHC_{\downarrow} as its input. The coefficients of this AR module are fixed to one (of 13) centre velocity at 4 Hz. If the other 12 centre velocities are utilised, the AR has to be cloned 12 more times, with the coefficients for each AR module clone set to one of 12 centre velocities. Following the conclusion of its processing, the rate filter-based *AR* module transmits the rate filter output sample, y_{rate} , to the *supervisor* module, before temporarily suspending its operation. It awakens when a subsequent IHC_{\downarrow} sample, corresponding to the next section is available.

At this point, the *supervisor* module invokes the following two modules simultaneously and transmits y_{rate} as their inputs: the third instantiated *AR* module dedicated to operate as a scale filter, and the first (of two) instantiated *HPF* module to perform a 90° phase shift on y_{rate} , to generate y_{Hrate} for use in the quadrature mirror Hilbert transformer (QMHT) operation depicted as \hat{h}_{nt} in equations (5-28) to (5-31). Both modules operate identical to the rate filter-based *AR* module, i.e. once they are processed, their outputs [y_{scale} for scale filter output and y_{Hrate} for high-pass filtered (HPF) rate filter output] are transmitted to the *supervisor* module. The two modules are suspended and await a new input (y_{rate}) from the next section.

The output of the scale filter, y_{scale} , is transmitted to a second instantiated *HPF* module, representing a QMHT, which introduces a 90° phase shift to the scale filter output producing y_{Hscale} that represents \hat{h}_{ns} in equations (5-28) to (5-31). Once y_{Hscale} is generated, it is transmitted to the supervisor module. The supervisor module then transmits y_{rate} , y_{scale} , y_{Hrate} , and y_{Hscale} to the *Direction* module, where the downward and upward A1 neuron directional outputs are calculated using equations (5-32) to (5-35). Table 5-3 illustrates the latencies of the invoked modules on FPGA. Each projected latency corresponding to an invoked module is below the threshold latency of 10.41 μ sec (2,604 clock cycles) – the time between the arrivals of two sequential audio samples. This latency ensures that no buffer overrun occurs, which is necessary to avoid output data corruption during real-time operation.

FPGA Modules Invoked in Parallel			1 Cochlear Section		108 Cochlear Sections	
Module Number	Module	Filter Type	Number of States	Latency (nsec)	Number of States	Latency (μsec)
1	<i>Supervisor</i>	-	9	36	972	3.888
2	<i>AR</i>	Cochlea	22	88	2,376	9.504
3	<i>AR</i>	Rate	17	68	1,836	7.344
4	<i>AR</i>	Scale	17	68	1,836	7.344
5	<i>HPF</i>	90° phase-shifted Rate	16	64	1,728	6.912
6	<i>HPF</i>	90° phase-shifted Scale	16	64	1,728	6.912
7	<i>Direction</i>	A1 Neuron Directional	11	44	1,188	4.752

Table 5-3: Latencies of modules in the CAR-Lite-A1 model running on an Altera Cyclone V FPGA.

5.6.2. Hardware Resource Utilisation

Table 5-4 shows the computational resources required to run the CAR-Lite-A1 model on an Altera Cyclone V GX Starter Kit fitted with a 5CGXFC5C6F27C7N FPGA chip. Computational resources refer to the number of digital signal processors, adaptive logic modules (ALMs) and registers utilised on the FPGA. For comparison, the CAR-Rate model [1] is included in Table 5-4 as well. The CAR-Rate model uses a 70-section CAR segment of the CAR-FAC cochlear model described in chapter 2 and a 4 Hz rate filter configured using an LPF-HPF cascade configuration described in subsection 5.2.2.3. The 4 Hz rate filter is invoked 70 times, and for each invocation, its input is one of 70 cochlear section output sample. The model uses a pipeline architecture and time-division multiplexing for cycling between the modules to process all the 70 sections.

From Table 5-4, it is clear that the CAR-Lite-A1 model with 108 sections uses more computational resources than the CAR-Rate model, as the former uses 38 more cochlear sections than the latter. Furthermore, the CAR-Lite-A1 has more filter modules in its A1 circuit (5 filter modules – 2 *AR*, 2 *HPF*, and 1 *Direction*) as opposed to one rate filter in the CAR-Rate model per cochlear section output connection. Reducing the number of cochlear sections of CAR-Lite-A1 model from 108 to 72 (6 octaves instead of 9 octaves with 12 sections per octave) brings the readout of the computational resources to be slightly less than the CAR-Rate model. The power consumption is 4 mW higher for the 108-section CAR-Lite-A1 implementation than the 72-section CAR-Rate implementation but can be regarded as an insignificant increase given the more considerable hardware usage of the former over the latter.

Model	Number of Cochlear Filters	Number of ALM Utilised (out of 29,080)	Number of Registers Utilised	Number of DSPs Utilised (out of 150)	Power (mW)
CAR-Rate [1]	70	1,793 (6.17%)	3,899	10 (6.67%)	-
CAR-Lite-A1 (6 octaves only)	72	1,743 (5.99%)	3,425	10 (6.67%)	240
This work (CAR-Lite-A1)	108	2,763 (9.5%)	5,043	12 (8.00%)	244

Table 5-4: Computational resources used on an Altera Cyclone V FPGA to implement A1 models.

5.6.3. Software Floating-Point vs. Hardware Fixed-Point

Figure 5-24 displays the degree of similarity between the responses of the software-based floating-point and the hardware-based fixed-point implementations of the CAR-Lite-A1 model simulated in Matlab. A correlation coefficient is used to show this degree of similarity between two 2D matrices as defined with the following equation:

$$CC = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \quad (5-41)$$

where m is the row number; n is the column number; A and B are the two 2D matrices to be compared; \bar{A} and \bar{B} are the mean of the 2D matrices.

Five types of signals are used individually as inputs to the model, which are discussed in detail in subsections 5.7.1 and 5.7.2. They include a decreasing-frequency log chirp, an increasing-frequency log chirp, a frequency modulated signal, and two moving ripple signals – one with decreasing and the other with increasing temporal velocities and spectral densities. The output signals are 3D (time-frequency-rate) rate filter output signal, 4D (time-frequency-rate-scale) scale filter output signal, 4D upward and 4D downward neuron directional output signals. The averages and standard deviations (denoted as average \pm standard deviation) of the correlation coefficients (CC) between the software floating-point and hardware fixed-point implementations of the five types of output signals for the real and imaginary analytic circuits are 0.98 ± 0.02 and 0.98 ± 0.02 respectively, indicating high degrees of similarities between floating-point and fixed-point implementations.

A 4D (time-frequency-rate-scale) matrix output from the CAR-Lite-A1 model is converted into a 2D summary (rate-scale) matrix to analyse the five signal types. This conversion is done using two methods: sum and root-mean-square (RMS). The next section provides details for these methods. Figure 5-25 displays the degree of similarity between the floating-point and fixed-point implementations of 2D (rate-scale) neuron directional responses calculated using sum operations from equations (5-42) and (5-43) and RMS operations from equations (5-44) and (5-45), respectively. The averages and standard deviations of CC for the sum responses of the real and imaginary analytic circuits between the software floating-point and hardware fixed-point implementation are 0.94 ± 0.03 and 0.96 ± 0.03 respectively. For the RMS responses of the real and imaginary analytic circuits between the software floating-point and hardware fixed-point implementation, these values are at 0.97 ± 0.01 and 0.97 ± 0.02 respectively. These results indicate that when a 4D output from the model is transformed into a 2D summary matrix, a response from an implementable hardware model can closely match the response from a software model.

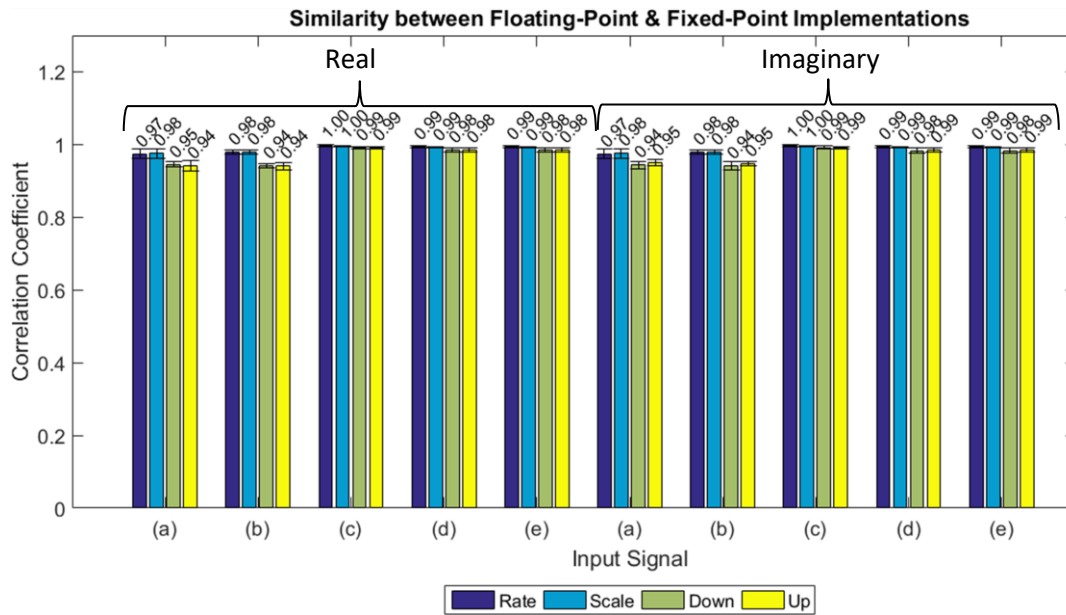


Figure 5-24: Correlation coefficients showing the similarity between the software floating-point and hardware fixed-point implementations of the CAR-Lite-A1 model 3D (rate) and 4D (scale, down, and up) responses for various input signals: log chirp with (a) decreasing frequency and (b) increasing frequency; (c) frequency-modulated signal; moving ripple with (d) decreasing and (e) increasing temporal velocity and spectral density respectively.

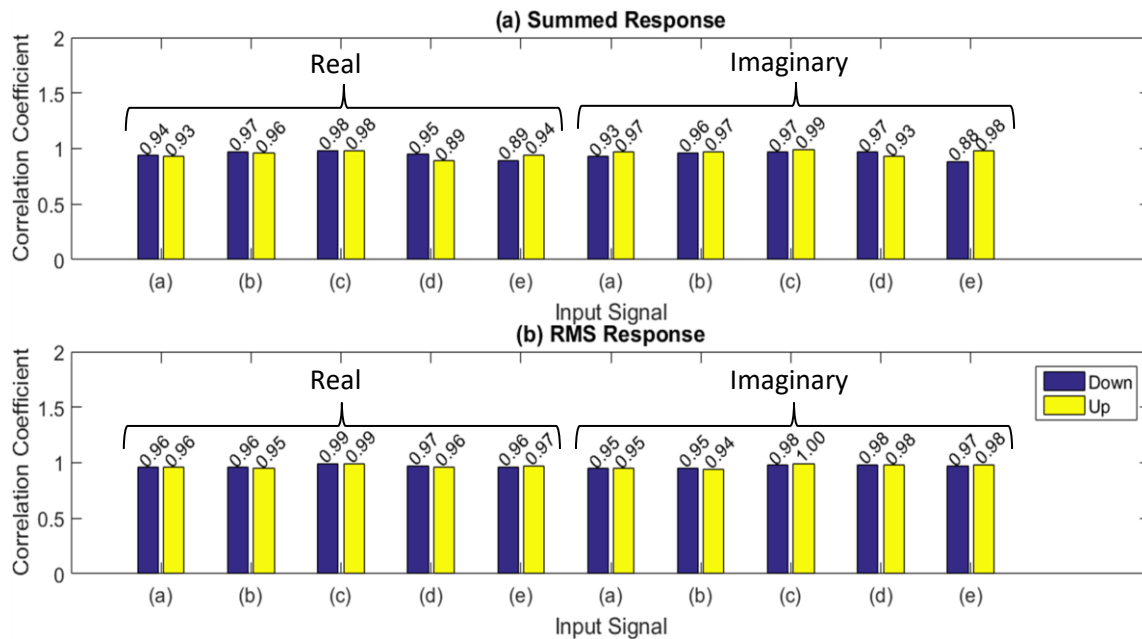


Figure 5-25: Correlation coefficients showing the similarity between the software floating-point and hardware fixed-point implementations of the 2D neuron directional rate-scale response from the CAR-Lite-A1 model for the same input signal described in Figure 5-24 using the following arithmetic operations to collapse at time-frequency axes (detailed in section 5.7), while maintaining rate-scale axes: (a) summed (b) RMS.

5.7. Model Responses

The upward and downward neuron directional outputs of the CAR-Lite-A1 model are each four-dimensional (4D) containing time, frequency, rate, and scale. Figure 5-26 displays two such 4D responses from the hardware fixed-point implementation specified in section 5.5, using a moving ripple input signal described in subsection 5.7.1 and Table 5-5. In this

figure, the two 4D responses are 4D matrices corresponding to the downward and upward neuron directional filter outputs. Additionally, each 4D matrix displayed contains only 3 arbitrarily selected rate filter settings (4 Hz, 16 Hz, and 128 Hz) out of a total 13, and 3 arbitrarily selected scale filter settings (0.25 c/o, 1 c/o, and 4 c/o) out of a total of 9. Hence, the size of each 4D matrix displayed here is 108 cochlear sections by 103,217 samples by three centre velocities (three rate filter settings mentioned earlier) by three centre densities (three scale filter settings mentioned earlier). A typical 4D output matrix size is $108 \times 103,217 \times 13 \times 9$, considering all 13 rate filter centre velocities and 9 scale filter centre densities for this specific input signal.

At a low rate (centre velocity) filter or low scale (centre density) filter setting, the output image has a blurry response. Conversely, at a high rate filter or high scale filter setting, the output image has a sharper response. This effect is consistent with image processing filtration outcomes. However, a piece of valuable information that is not observable from a 4D output matrix is a summary of power distribution across all rate and scale settings. To attain such an observation, it is beneficial to convert the 4D matrix to a conventional 2D matrix. Doing so would provide more unambiguous visual cues of power distribution over the two axes on only a single graph instead of multiple graphs laid out on additional two-dimensional axes (bringing it to 4D). The process of this conversion is presented in the next paragraph, onwards.

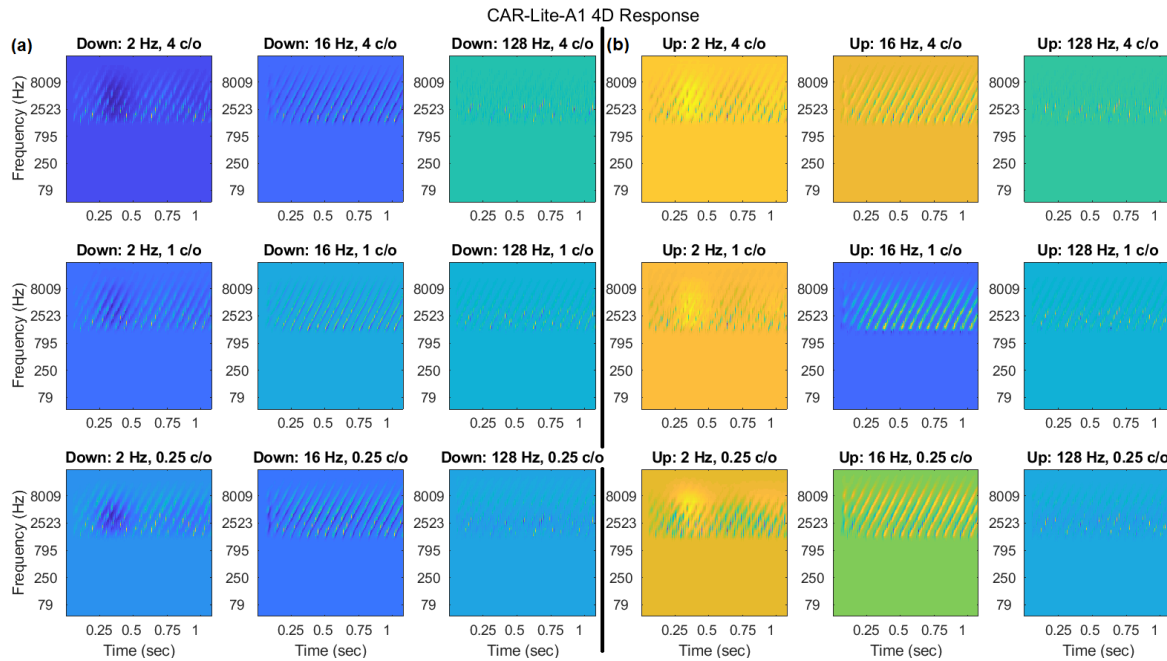


Figure 5-26: Hardware fixed-point representation of a 4D (a) downward and (b) upward neuron directional responses from the CAR-Lite-A1 model using upward drifting spectro-temporal envelopes in a moving ripple input signal configured from subsection 5.7.1 and Table 5-5. Note the input signal and its cochleagram representation are displayed in Figure 5-28(a) and (b), respectively.

According to Chi et al. [23] and Patil et al. [16], a 4D response of a functional A1 model can be converted to a summary 2D response for analysis. The contents of the response on the axes belonging to the 2D axes to be excluded are summed up, which enable the 2D summary matrix to be insensitive to translations to the excluded axes. As an example, to enable the upward and downward neuron directional 4D matrices to capture only temporal envelope and spectral envelope changes, containing the degree of modulation over time

(rate) as well as the degree of modulation across frequency (scale), high time resolution and high frequency resolution information should be excluded in the summary 2D responses. Similarly, in the CAR-Lite-A1 model, this attribute is achievable by summing the contents of the two 4D matrices over time and frequency using equations (5-42) and (5-43). Subsequently, the rate and scale dimensions are represented on the x- and y-axes respectively.

An alternative technique to dimensionality reduction is to apply RMS to the vector dimensions of the 4D matrix. The RMS of a time-varying vector of a dimension is its average power calculated as a constant over its finite vector of samples [56]. In electrical circuit theory, the RMS power of a load in a circuit is the average power when the circuit is supplied with either an alternating current (AC) or direct current (DC) [57]. The RMS calculation includes the following operations as determined by equations (5-44) and (5-45): multiply, sum, divide, square-root. Although these operations are implementable on an FPGA, the mathematical operations outnumber the mathematical operations from the summation technique defined by equations (5-42) and (5-43). Hence, the RMS method would use more computational resources on an FPGA than the summation method. As a result, the RMS method is regarded as a software solution for dimensionality reduction as it would potentially run on a computer and process the A1 neuron directional responses output from an FPGA. The summation method is regarded as a hardware solution for dimensionality reduction as its operation can be executed entirely on an FPGA.

The sum [equations (5-42) and (5-43)] and root-mean-square (RMS) [equations (5-44) and (5-45)] functions that are used in the collapse of the time and frequency dimensions are defined as:

$$r_{\uparrow}^s(\omega, \Omega) = \sum_{s=0}^S \sum_{t_{\downarrow}=0}^{T_{\downarrow}} r_{\uparrow}(s, t_{\downarrow}; \omega, \Omega) \quad (5-42)$$

$$r_{\downarrow}^s(-\omega, \Omega) = \sum_{s=0}^S \sum_{t_{\downarrow}=0}^{T_{\downarrow}} r_{\downarrow}(s, t_{\downarrow}; -\omega, \Omega) \quad (5-43)$$

$$r_{\uparrow}^r(\omega, \Omega) = \sqrt{\frac{1}{S} \sum_{s=0}^S \frac{1}{T_{\downarrow}} \sum_{t_{\downarrow}=0}^{T_{\downarrow}} r_{\uparrow}(s, t_{\downarrow}; \omega, \Omega)^2} \quad (5-44)$$

$$r_{\downarrow}^r(-\omega, \Omega) = \sqrt{\frac{1}{S} \sum_{s=0}^S \frac{1}{T_{\downarrow}} \sum_{t_{\downarrow}=0}^{T_{\downarrow}} r_{\downarrow}(s, t_{\downarrow}; -\omega, \Omega)^2} \quad (5-45)$$

where r_{\uparrow}^s and r_{\downarrow}^s are the 2D summary upward and downward neuron directional responses respectively calculated using a sum function; r_{\uparrow}^r and r_{\downarrow}^r are the 2D summary upward and downward neuron directional responses respectively calculated using an RMS function; T_{\downarrow} is the total number of samples in time; S is the total number of cochlear sections at 108. The downward directional response, r_{\downarrow}^s and r_{\downarrow}^r undergoes a sign change at all centre velocities of the rate filterbank. This effect flips the 2D matrices horizontally along the x-axis for

visualisation purpose, similar to time reversal in signal process theory [55]. Thus, the downward neuron directional response is presented in negative rates ($-\omega$) and upward neuron directional response is presented in positive rates (ω).

Three distinct input signals are used to exhibit the response of the CAR-Lite-A1 model: moving ripple, frequency-modulated (FM) and logarithmic frequency sweep (log chirp) signals. Table 5-5 provides the settings of the three input signals used for illustrating the results in the next two subsections.

Moving Ripple	FM	Log Chirp
Sampling rate, $f_s = 96$ kHz	Sampling rate, $f_s = 96$ kHz	Sampling rate, $f_s = 96$ kHz
$f_0 = 1.5$ kHz	$f_c = 1$ kHz	Ascending: $f_0 = 0.01$ Hz
Signal duration, $t_D = 1$ sec	Signal duration, $t_D = 1$ sec	Signal duration, $t_D = 1$ sec
Bandwidth, $B = 0.9$ octaves		
Frequency spacing, $\Delta f = 0.0625$ octave	$f_m = 16$ Hz	Ascending: $f_1 = 48$ kHz
Amplitude, $A = 0.9$	$m_i = 100\%$	Descending: $f_0 = 48$ kHz
Phase, $\varphi = 0^\circ$		
Rate, $\omega = 16$ Hz		Descending: $f_1 = 0.01$ Hz
Scale, $\Omega = 1$ c/o		

Table 5-5: Input signal settings.

5.7.1. Response to Moving Ripple

A moving ripple signal comprises pure tones distributed over a logarithmic frequency axis. In addition, the pure tones are modulated in time, varying either up or down the frequency-axis at a constant velocity. It is defined by:

$$S(x, t) = A \cdot \sin(2\pi \cdot (\omega \cdot t + \Omega \cdot x) + \varphi) \quad (5-46)$$

where A is the amplitude of the signal; ω is envelope velocity over time, t ; Ω is the density of frequency components; x is the position over the frequency-axis characterised by $\log_2(f/f_0)$ – f is a pure tone frequency and f_0 is the lowest pure tone frequency; φ is phase of the signal.

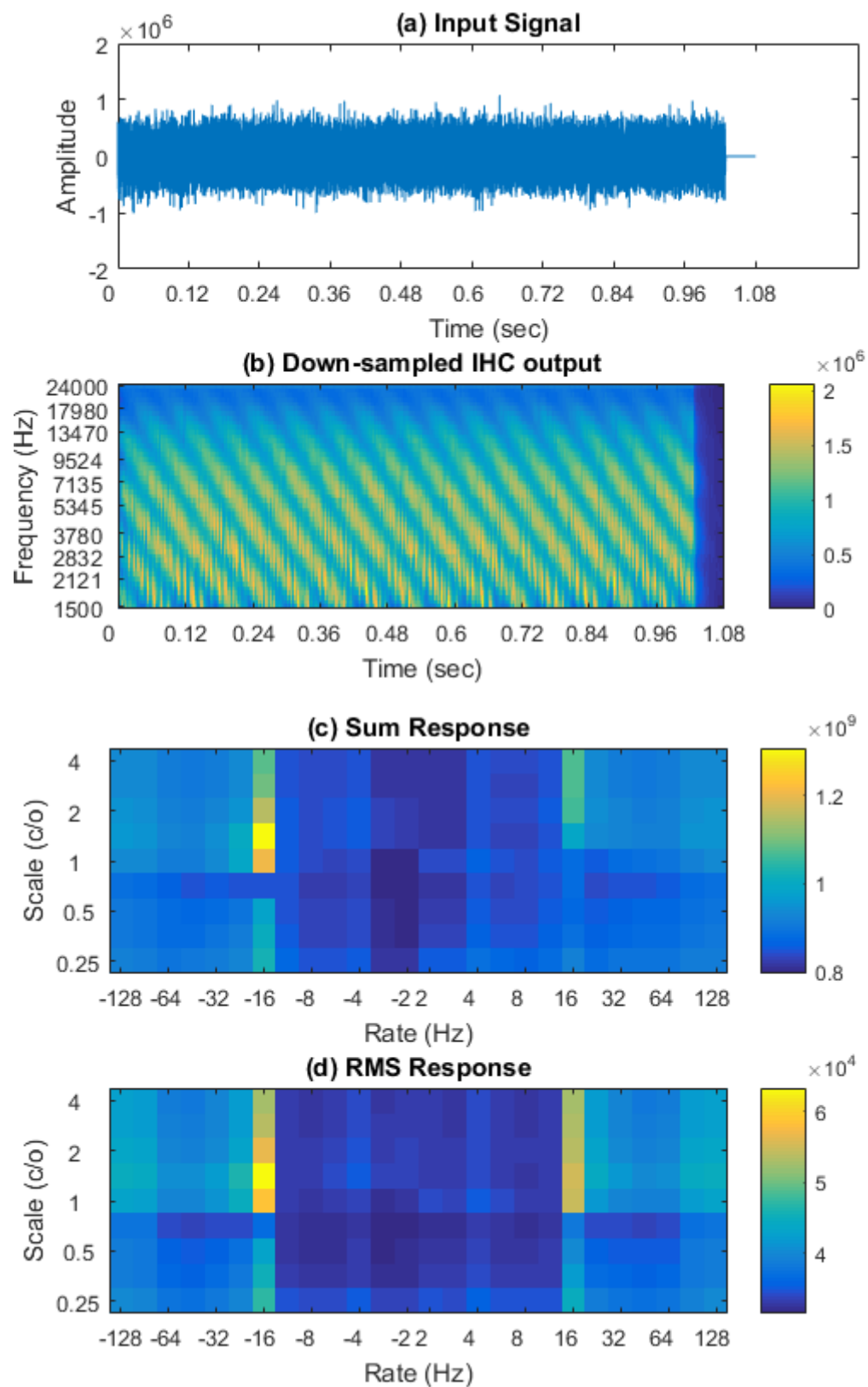
Figure 5-27(a) illustrates a moving ripple signal traversing in a downward direction using the settings presented in the first column of Table 5-5. Figure 5-28(b) illustrates a moving ripple signal traversing in an upward direction using the same settings from Table 5-5. The downward input signal is time-reversed (flipped along the time axis) to attain the upward signal. Figure 5-27(b) and Figure 5-28(b) displays the down-sampled IHC output from the CAR-Lite-A1 model of the two input signals. In the former, the downward moving ripple is represented by multiple parallel thick illuminated lines with negative gradients (lines from the upper left to the lower right), while in the latter, the upward moving ripple is represented by the same lines but with positive gradients (lines from the lower left to the upper right).

Parts (c) – (f) of Figure 5-27 and Figure 5-28 are formed with the combined neuron directional responses of the downward and upward filterbanks in the CAR-Lite-A1 model. The downward response is a 2D summary matrix output from the downward directional filterbank and displayed in negative rates on the x-axis (left-half of image), while the upward response is a 2D summary matrix output from the upward directional filterbank and displayed in positive rates on the x-axis (right-half of image). Parts (c) and (e) of Figure 5-27

and Figure 5-28 are 2D summary matrices calculated using sum operations defined by equations (5-42) and (5-43). Parts (d) and (f) of Figure 5-27 and Figure 5-28, are 2D summary matrices calculated using RMS functions defined by equations (5-44) and (5-45) respectively. Parts (c) and (d) of Figure 5-27 and Figure 5-28 are the responses of the real component of the analytic equation circuit from Figure 5-18(a), while parts (e) and (f) are the responses of the imaginary component of the analytic equation circuit from Figure 5-18(b).

The maximum energy (pixel activation) for the responses calculated with the imaginary component of the analytic system circuit for both upward and downward directions are located precisely at the rate-scale coordinates of (16 Hz, 1 c/o), which corresponds to the rate and scale setting of the moving ripple signal. In contrast, the maximum activations for the sum and RMS responses calculated with the real component of the analytic system circuit, regardless of directions, are located at (16 Hz, 1.414 c/o). These shifted responses are due to the multiplication of the 90° phase-shifted rate, and scale terms to each other, i.e. $\hat{h}_{nt} \cdot \hat{h}_{ns}$, which are then either added or subtracted from the non-phase shifted rate-scale terms, i.e. $h_{nt} \cdot h_{ns}$, in the real component equations defined in (5-28) and (5-29). Alternatively, the equations in the imaginary component defined in (5-30) and (5-31), have the 90° phase-shifted rate term multiplied to the non-phase shifted scale, i.e. $\hat{h}_{nt} \cdot h_{ns}$, and the 90° phase-shifted scale term multiplied to the rate term, i.e. $h_{nt} \cdot \hat{h}_{ns}$, instead of the two phase-shifted terms multiplied to each other. As a result, the imaginary component of the analytic system circuit projects maximum energy more precisely than the real component of the analytic system circuit.

For the downward drifting moving ripple in Figure 5-27, the sum and RMS responses for the real component of the analytic circuits show maximum pixel activation (value) in the negative half of the rate axis and no activation in the positive half. For the imaginary component analytic circuit using the same downward drifting, moving ripple input signal, the maximum pixel activation is also on the negative half for the sum response, but it is situated on the positive half for the RMS response. For an upward drifting moving ripple in Figure 5-28, both the sum and RMS responses for the real component of the analytic circuits show maximum pixel activation (value) in the positive half of the rate axis and no activation in the negative half. For the imaginary component analytic circuit, maximum pixel activation is also at the positive half of the rate axis for the sum response, while it is situated on the negative half for the RMS response. The inconsistency of the representation of the upward and downward drifting ripples between the sum and RMS responses is described in the next subsection (5.7.1.1).



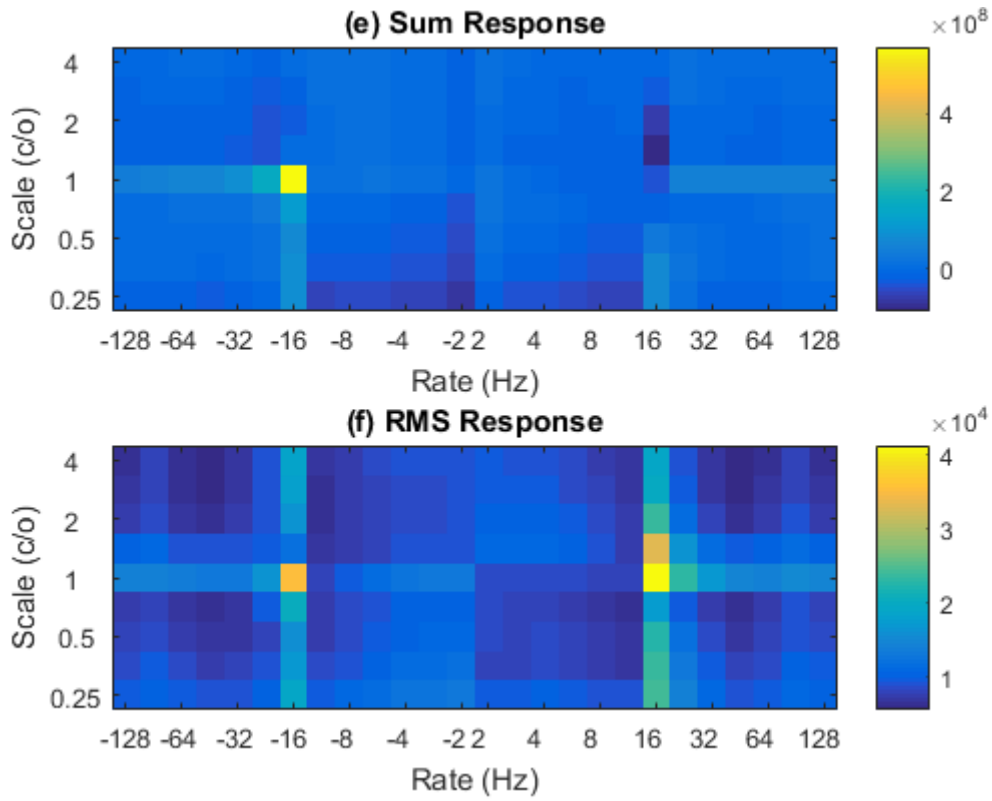
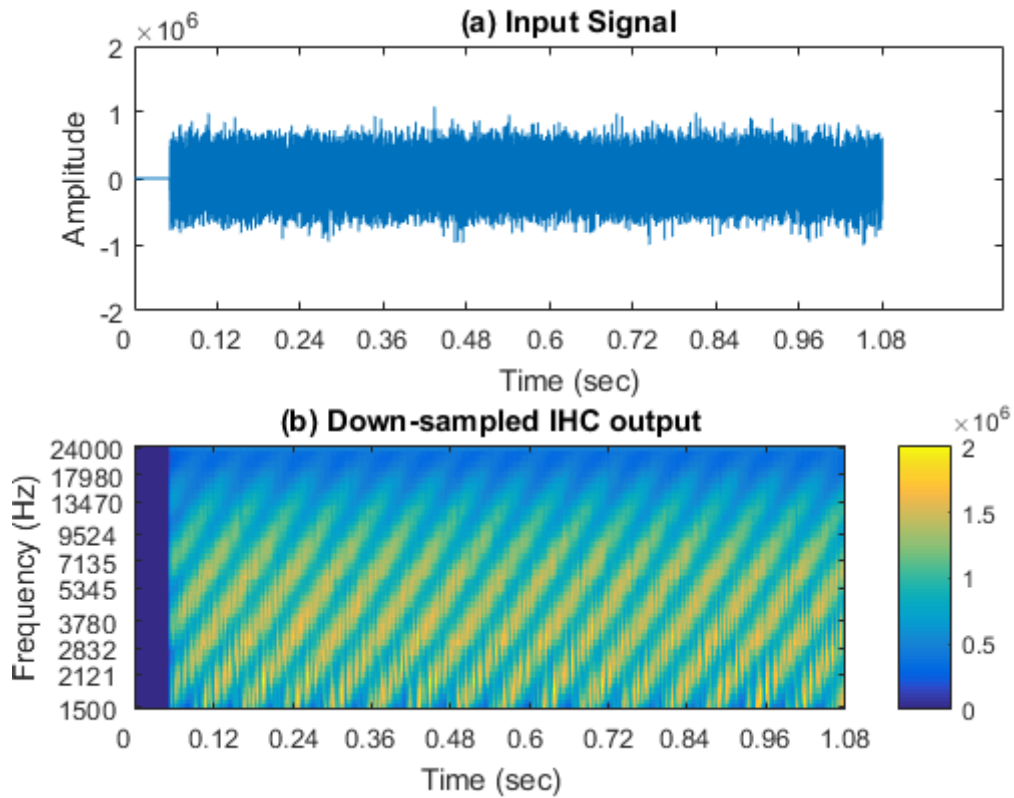


Figure 5-27: Fixed-point (hardware) representation of (a) a moving ripple signal with decreasing frequencies over time at (b) the down-sampled IHC stage, and (c) – (f) the responses of the A1 neuron directional filter stage of the CAR-Lite-A1 model. (c) and (d) are the sum and RMS responses of the real component of the analytic transfer function, whose circuit is depicted in Figure 5-18(a) respectively. (e) and (f) are the sum and imaginary responses of the imaginary component of the analytic transfer function, whose circuit is depicted in Figure 5-18(b).



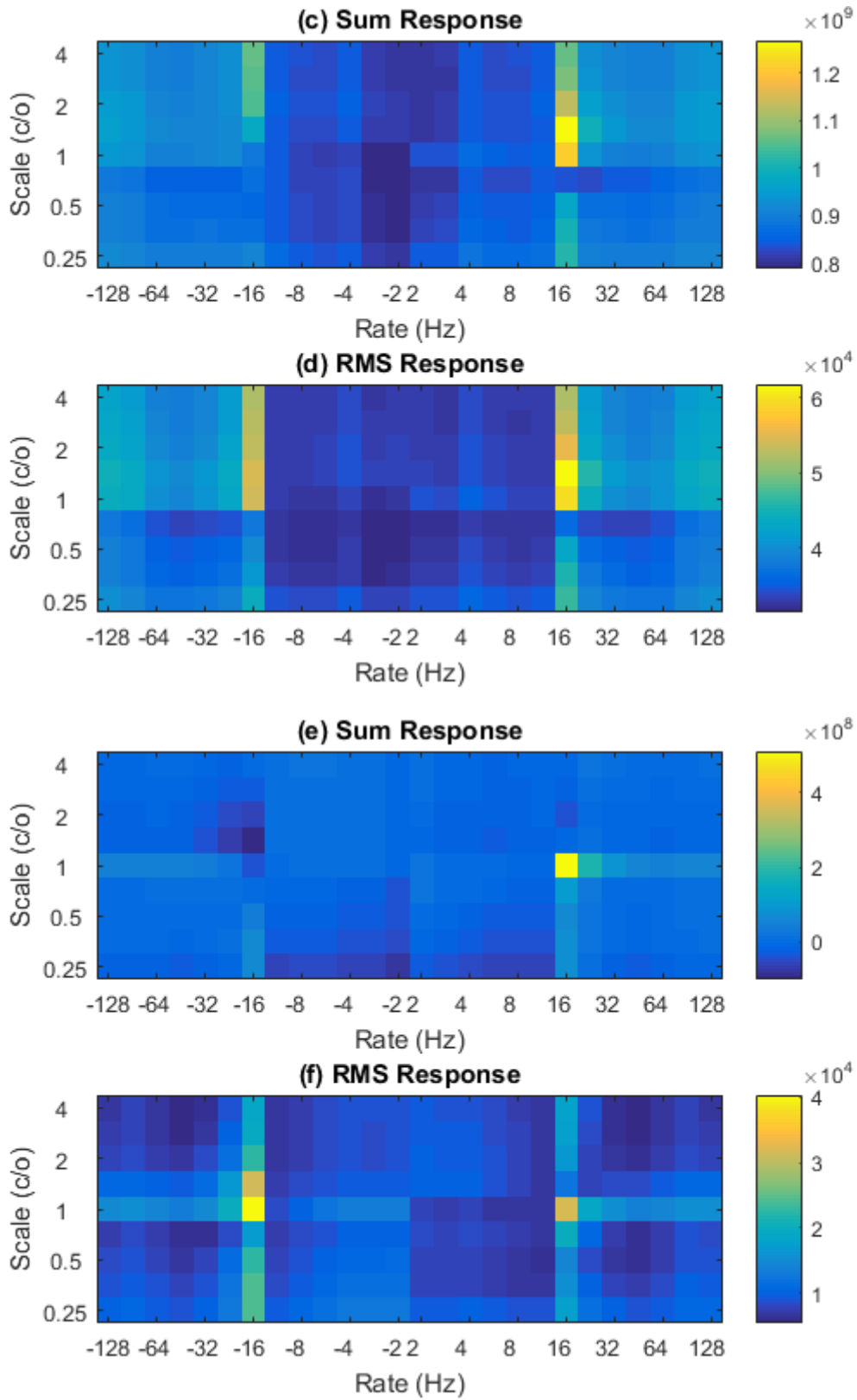


Figure 5-28: Fixed-point (hardware) representation of (a) a moving ripple signal with increasing frequencies over time and its equivalent representation at (b) the down-sampled IHC stage, and (c) – (f) the responses of the A1 neuron directional filter stage of the CAR-Lite-A1 model. (c) and (d) are the sum and RMS responses of the real component of the analytic transfer function depicted in Figure 5-18(a) respectively. (e) and (f) are the sum and RMS responses of the imaginary component of the analytic transfer function depicted in Figure 5-18(b).

5.7.1.1. Difference between Sum and RMS Profiles

The different RMS response of the imaginary component of the analytic circuit, mentioned in the preceding subsection, can be attributed to the multiplication of the 90° phase-shifted terms to the non-phase shifted terms, i.e. $\hat{h}_{nt} \cdot h_{ns}$ and $h_{nt} \cdot \hat{h}_{ns}$. The effect of these terms produces a more distributed energy across both the negative half and positive half of the rate axis, than the real component of the analytic circuit, which has more energy concentration to a specific half based on the direction of envelope drift. In other words, this energy distribution in the RMS response is due to the calculation of average power spread across the 2D summary rate-scale matrix, whereas the sum response puts the highest accumulative magnitudes from the 4D output matrix to the forefront of the 2D summary rate-scale matrix.

The sum response can also reproduce the effect observed in the RMS response of the imaginary component of the analytic circuit by merely including an absolute term before the summation in equations (5-42) and (5-43):

$$r_{\uparrow}^s(\omega, \Omega) = \sum_{s=0}^S \left| \sum_{t_{\downarrow}=0}^{T_{\downarrow}} |r_{\uparrow}(s, t_{\downarrow}; \omega, \Omega)| \right| \quad (5-47)$$

$$r_{\downarrow}^s(-\omega, \Omega) = \sum_{s=0}^S \left| \sum_{t_{\downarrow}=0}^{T_{\downarrow}} |r_{\downarrow}(s, t_{\downarrow}; -\omega, \Omega)| \right| \quad (5-48)$$

The responses of equations (5-47) and (5-48) is observable in Figure 5-29, which are similar to the RMS responses shown in Figure 5-27(f) and Figure 5-28(f). Ideally, the absolute term is explicitly calculated as the root-sum-square of a complex number, i.e. $|\pm x| = \sqrt{Re(\pm x)^2 + Im(\pm x)^2}$ [55]. If no imaginary numbers exist, as is the case for the output signals from the CAR-Lite-A1 model, then the equation is simplified to $|\pm x| = \sqrt{Re(\pm x)^2} = x$. More specifically, this operation converts any signed real-valued number to a real-valued positive number. By doing so, neuron directionality information might be lost at the expense of the average power information calculated with the RMS operation. In other words, neuron directionality information from the CAR-Lite-A1 model is influenced mainly by the sum and subtract operations defined in equations (5-28), (5-29), (5-30), and (5-31), which affects the signs of the output values. However, RMS operations produce only positive output values regardless of the signs of their input samples, thereby potentially removing vital neuron directionality information.

Note that in the NSL model [23], Chi et al. showed that the downward response is displayed on the positive rate axis and the upward response on the negative rate axis. This effect is in contrast with the CAR-Lite-A1 model responses displayed in Figure 5-27 and Figure 5-28, as well as Figure 5-30 and Figure 5-31 in the next subsection (5.7.2), i.e. the downward response is displayed on the negative rate axis, and the upward response is displayed on the positive rate axis. The sign difference of the rate dimension stems from the structure of the models: The NSL model is a sophisticated model using three stages of causal, non-causal, and nonlinear signal processing filters, whereas the CAR-Lite-A1 model is an abstract of the NSL model made of two stages of causal, linear and analytic signal

processing filters. Although the signs of the rate axis can be swapped to correct this difference, the analytic signal responses from the A1 neuron directional display of the CAR-Lite-A1 model is maintained here to differentiate with the signal responses from the same stage of the NSL model.

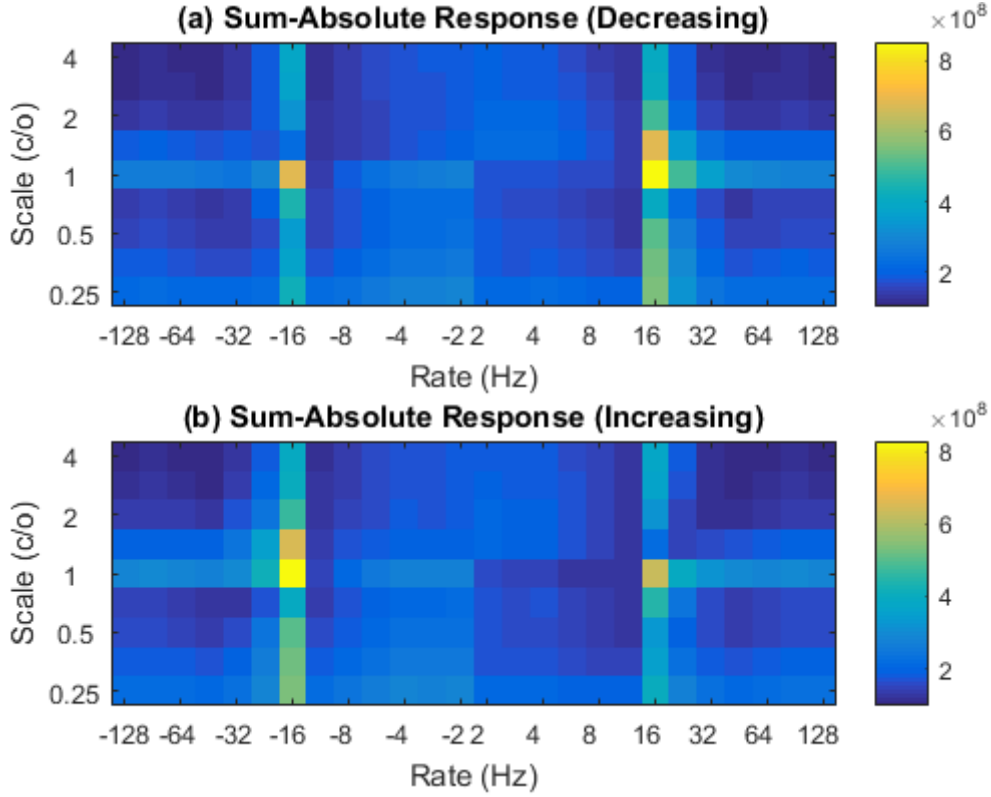


Figure 5-29: Fixed-point representation of the imaginary A1 neuron directional responses using sum-absolute operations to indicate similarity to the RMS operations responses [Figure 5-27(f) and Figure 5-28(f)] of (a) a moving ripple with decreasing frequencies over time and (b) a moving ripple with increasing frequencies over time.

5.7.2. Response to Frequency-Modulated and Log Chirp Signals

In-vivo physiological recordings of mice auditory cortex using directional frequency sweeps indicate the presence of ON-OFF receptive fields [58]. This ON-OFF characteristic should be ideally manifested from the A1 neuron directional response used in the CAR-Lite-A1 model, which means that a sound signal with an increasing envelope frequency should activate (ON) the positive half only, while the negative half is deactivated (OFF). A sound signal with decreasing envelope frequency activates (ON) the negative half while deactivating (OFF) the positive half.

In section 5.7.1, the mutually exclusive responses described in the preceding paragraph was observed using moving ripples input signal. To test the consistency of the directional neuron response of the CAR-Lite-A1 model, two different input signals are used here: a frequency-modulated (FM) signal and a logarithmically-spaced frequency sweep (log chirp) signal. The FM signal uses a short frequency range of drifting envelope, while the log chirp uses a long frequency range of drifting envelope. The FM signal, x_{FM} , and log chirp signal, x_{LC} , are defined as follow:

$$x_{FM} = \cos(2\pi f_c t + m_i \cos(2\pi f_m t)) \quad (5-49)$$

$$x_{LC} = \sin \left(2\pi f_0 \left(\frac{(f_1/f_0)^{t/t_D} - 1}{\ln((f_1/f_0)^{1/t_D})} \right) \right) \quad (5-50)$$

where f_c is the carrier signal frequency; f_m is the modulation frequency or signal envelope frequency; m_i is the modulation index in percent; f_0 is the start frequency; f_1 is the end frequency; t_D is the duration of the signal or the final element of a time vector, t . The settings of these variables are projected in Table 5-5. Figure 5-30 and Figure 5-31 display the input signals, down-sampled IHC responses and neuron directional responses to the FM and log chirp input signals respectively.

From Figure 5-30 and Figure 5-31, the sum and RMS responses of the real and imaginary analytic circuits to ascending and descending frequency sweeps, are in agreement with the ON-OFF characteristics described in the first paragraph of this subsection – for ascending frequency sound signals, maximum values corresponding to the most salient or highest energy activations are found at the positive half of the rate axis. In contrast, for descending frequency sound signals, maximum values are found at the negative half of the rate axis. The exceptions are the RMS responses of the imaginary analytic circuits, whose responses are indistinguishable for both increasing and decreasing frequency sweeps. This effect is due to its more evenly distributed energy as opposed to the other type of responses as mentioned in subsection 5.7.1.1.

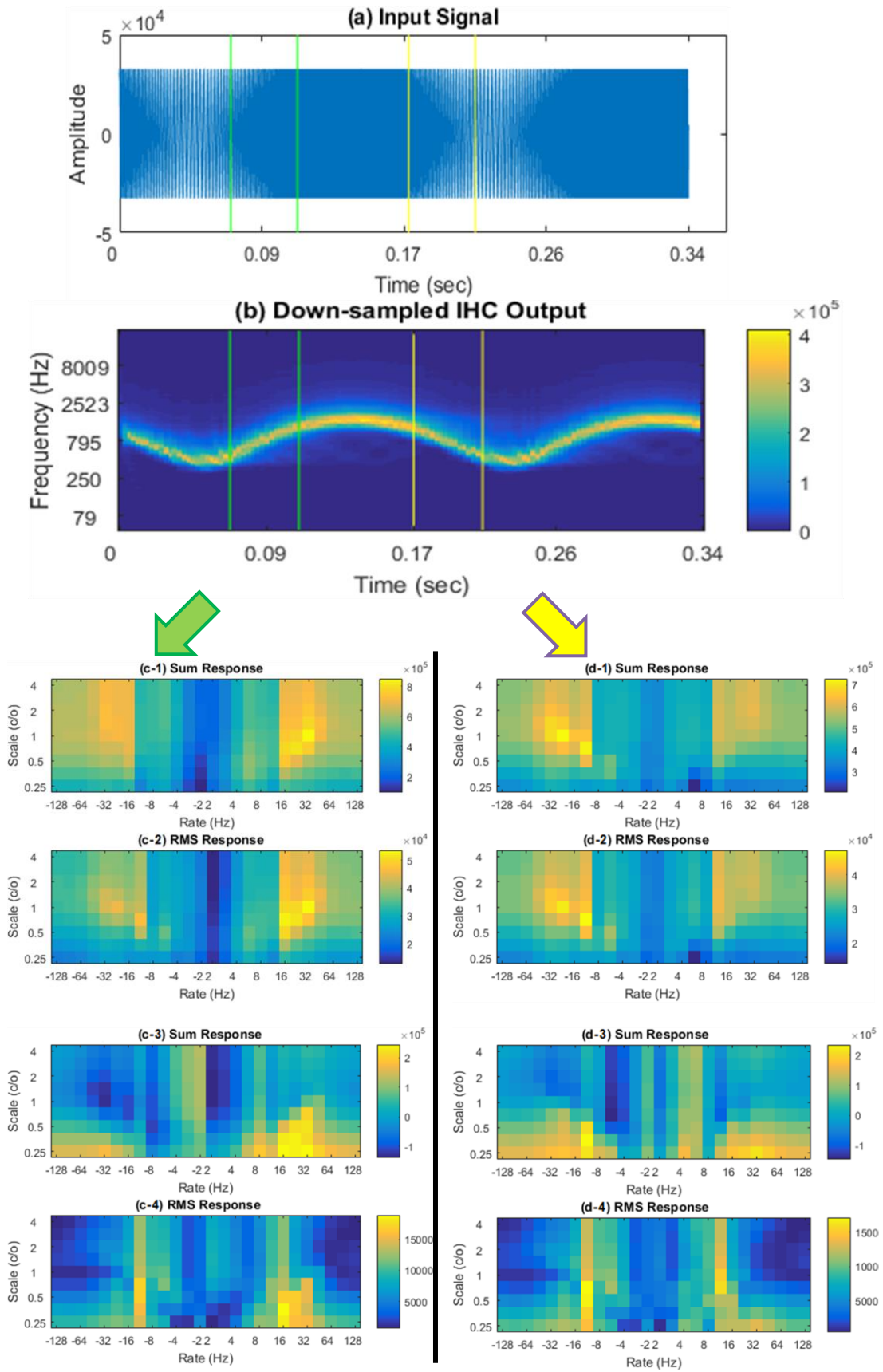


Figure 5-30: Fixed-point representation of (a) an FM signal configured with settings from the second column of Table 5-5 and its equivalent representation from (b) the down-sampled IHC stage. Summary A1 neuron directional filter responses recorded from the CAR-Lite-A1 model for: the increasing frequency segment (in green

vertical lines) of the FM signal calculated using (c-1) sum and (c-2) RMS operations with a Real circuit as well as (c-3) sum and (c-4) RMS operations using an Imaginary circuit; the decreasing frequency segment (in yellow vertical lines) of the FM signal calculated using (d-1) sum and (d-2) RMS operations with a Real circuit as well as (d-3) sum and (d-4) RMS operations with an Imaginary circuit. Here, Real circuit refers to the real component of the analytic signal circuit depicted in Figure 5-18(a) and Imaginary circuit refers to the imaginary component of the analytic signal circuit depicted in Figure 5-18(b).

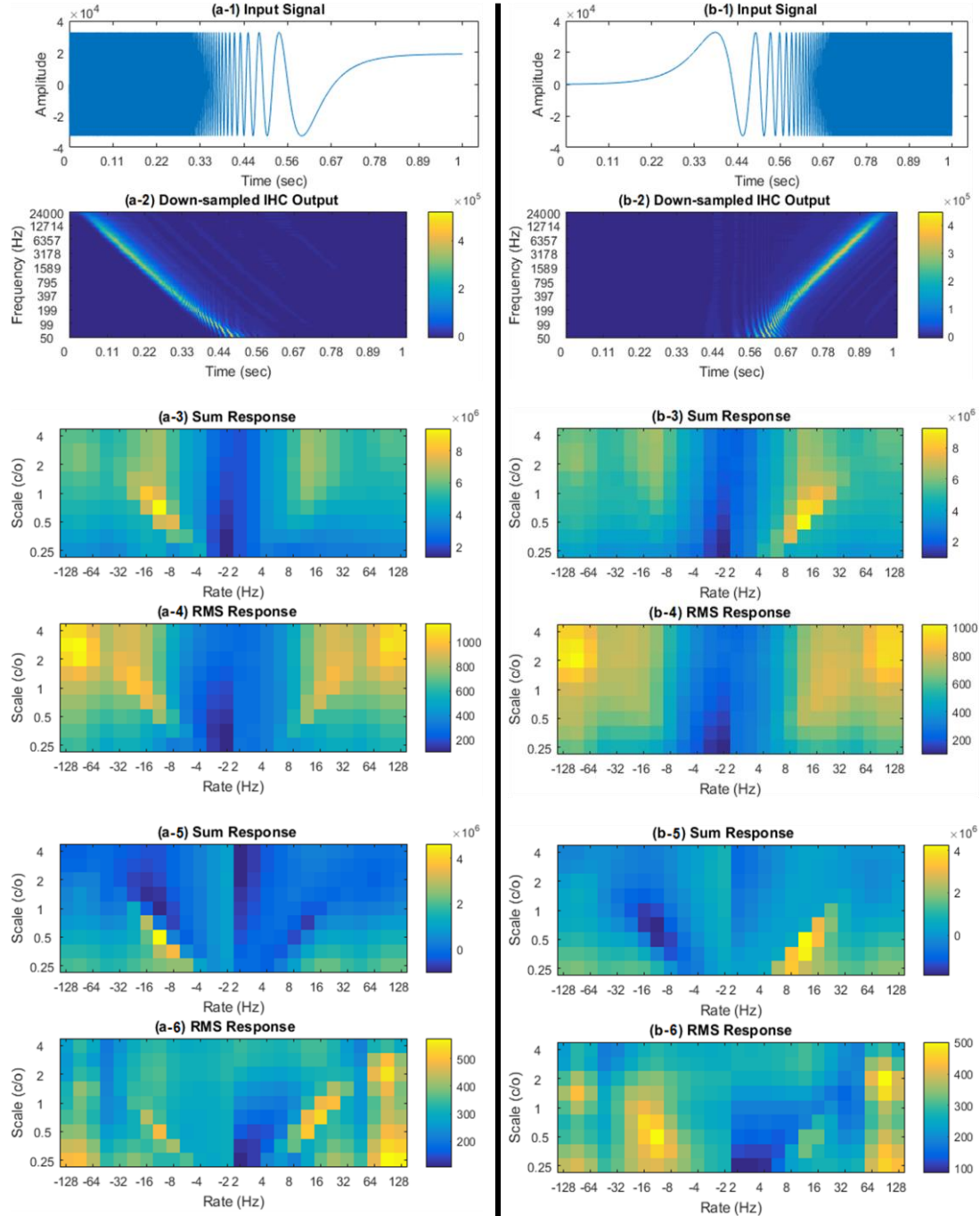


Figure 5-31: Fixed-point representation of a log chirp signal with (a) decreasing frequency and (b) increasing frequency configured with settings from the third column of Table 5-5 along with their corresponding representation from the down-sampled IHC stage [(a-2) – decreasing and (b-2) – increasing]. Parts (a-3) – (a-6) are the A1 neuron directional filter responses of the CAR-Lite-A1 model for the log chirp signal with decreasing frequency, while parts (b-3) – (b-6) are the responses for the log chirp with increasing frequency. Parts (a-3), and (b-3) are the sum responses, while (a-4), and (b-4), are the RMS responses of the real component of the analytic signal circuit depicted in Figure 5-18(a) respectively. Parts (a-5), and, (b-5), are the sum responses, while (a-6)

and (b-6) are the RMS responses of the imaginary component of the analytic signal circuit depicted in Figure 5-18(b).

5.8. Chapter Summary and Conclusion

This chapter proposes a new functional primary auditory cortical model named CAR-Lite-A1. This model uses the CAR-Lite model, described in the first half of chapter 3, as its frontend connected to a functional primary auditory cortical model (A1 segment of CAR-Lite-A1). The filters in the A1 segment of the CAR-Lite-A1 model are designed with causal, linear, time-invariant filters with a focus on hardware (FPGA) implementation. These filters are used for extracting temporal and spectral envelopes as well as the directionality of the frequency changes in an input sound signal, resulting in a 4D response. Phase calculation is omitted from the model to maintain a small model size on a single FPGA as well as to lessen the burden on computational resources utilised. Studies are performed to ensure the stability of the filters to be implemented on FPGA. Re-synthesis of a sound signal from the output of the CAR-Lite-A1 model is not possible with the newly designed filters as the inverse of the transfer functions of these filters yield unstable operations.

The 4D output response of the CAR-Lite-A1 model is condensed into a 2D summary profile for a compact 2D representation of timbre. The sizes of the 2D summary profile matrix is fixed regardless of the length of input signals, which is advantageous for timbre classification.

On FPGA, the resources used by the CAR-Lite-A1 model are compared with a model that only extracts temporal envelope (CAR-Rate). Furthermore, the CAR-Lite-A1 model is capable of extracting envelope information from common signals used in other auditory cortical experiments such as moving ripple, frequency-modulated, and log chirp signals. Chapter 7 presents the performance of the responses of the CAR-Lite-A1 model with real-world signals, specifically in regard to musical instruments classification.

5.9. Bibliography

- [1] C. S. Thakur, R. M. Wang, S. Afshar, T. J. Hamilton, J. Tapson, S. A. Shamma, and A. van Schaik, "Sound Stream Segregation: A Neuromorphic Approach to Solve the 'Cocktail Party Problem' in Real-Time," *Front. Neurosci.*, vol. 9, pp. 1–10, 2015, doi: 10.3389/fnins.2015.00309.
- [2] J. P. Jones and L. A. Palmer, "An Evaluation of the Two-Dimensional Gabor Filter Model of Simple Receptive Fields in Cat Striate Cortex," *J. Neurophysiol.*, vol. 58, no. 6, pp. 1233–1258, 1987.
- [3] N. N. Schraudolph, "A Fast, Compact Approximation of the Exponential Function," *Neural Comput.*, vol. 11, no. 4, pp. 853–862, 1999, doi: 10.1162/089976699300016467.
- [4] R. K. Singh, "A Real-time Implementation of the Primary Auditory Neuron Activities," University of Western Sydney, 2012.
- [5] R. K. Singh, "RTAP: Real-Time Auditory Periphery." Singh, Ram Kuber, Sydney, 2014, [Online]. Available: <https://github.com/rtap-dev/RTAP>.
- [6] R. K. Singh, "A Real-Time Implementation of a Dual Resonance Nonlinear Filterbank," in *International Conference on Engineering and Natural Sciences*, 2015, pp. 36–41, doi: <http://iierdl.org/proceeding.php?pid=24>.

- [7] Y. Cheol, P. Cho, S. Bae, Y. Jin, K. M. Irick, and V. Narayanan, "Exploring Gabor Filter Implementations For Visual Cortex Modeling on FPGA," in *2011 21st International Conference on Field Programmable Logic and Applications*, 2011, pp. 311–316, doi: 10.1109/FPL.2011.63.
- [8] J. A. Hernandez and B. Romero, "Multiscale Image Representation Based on Gabor Transform Using Reconfigurable FPGA," in *2003 IEEE International Computer Architectures for Machine Perception*, 2003, pp. 256–262, doi: 10.1109/CAMP.2003.1598171.
- [9] Y. C. P. Cho, N. Chadramoorthy, K. M. Irick, and V. Narayanan, "Multiresolution Gabor Feature Extraction for Real Time Applications," in *2012 Workshop on Signal Processing Systems*, 2012, pp. 55–60, doi: 10.1109/SiPS.2012.56.
- [10] G. Giun, C. Lee, Z. Huang, C. Chen, and C. Chen, "Implementation of Gabor Feature Extraction Algorithm for Electrocardiogram on FPGA," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 798–801, doi: 10.1109/ISCAS.2015.7168754.
- [11] S. Qiu, F. Zhou, and P. E. Crandall, "Discrete Gabor transforms with complexity $O(N \log N)$," *Signal Processing*, vol. 77, no. 2, pp. 159–170, 1999, doi: 10.1016/S0165-1684(99)00030-4.
- [12] I. T. Young, L. J. van Vliet, and M. van Ginkel, "Recursive Gabor Filtering," *IEEE Trans. Signal Process.*, vol. 50, no. 11, pp. 2798–2805, 2002, doi: 10.1109/TSP.2002.804095.
- [13] E. David, P. Ungureanu, and M. Ansorge, "A Fast Recursive Implementation of Gabor Filters," in *International Symposium on Signals, Circuits and Systems, 2005 (ISSCS 2005)*, 2005, pp. 581–584, doi: 10.1109/ISSCS.2005.1511307.
- [14] C. Lin, L. Tao, and H. K. Kwan, "Parallel-Computing-Based Implementation of Fast Algorithms for Discrete Gabor Transform," *IET Signal Process.*, vol. 9, no. 7, pp. 546–552, 2015, doi: 10.1049/iet-spr.2014.0300.
- [15] R. Plomp and H. J. M. Steeneken, "Effect of Phase on the Timbre of Complex Tones," *J. Acoust. Soc. Am.*, vol. 46, no. 2B, pp. 409–421, 1969, doi: 10.1121/1.1911705.
- [16] K. Patil, D. Pressnitzer, S. Shamma, and M. Elhilali, "Music in Our Ears: The Biological Bases of Musical Timbre Perception," *PLoS Comput. Biol.*, vol. 8, no. 11, pp. 1–16, 2012, doi: 10.1371/journal.pcbi.1002759.
- [17] S. W. Smith, "Filter Comparison," in *The Scientist and Engineer's Guide to Digital Signal Processing*, 1st ed., San Diego, CA, USA: California Technical Publishing, 1997, pp. 343–350.
- [18] M. Adeli, J. Rouat, S. Wood, S. Molotchnikoff, and E. Plourde, "A Flexible Bio-Inspired Hierarchical Model for Analyzing Musical Timbre," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 24, no. 5, pp. 875–889, 2016, doi: 10.1109/TASLP.2016.2530405.
- [19] M. I. T. Opencourseware, "Determining a System's Causality from its Frequency Response." Massachusetts Institute of Technology, pp. 1–2, 2009.
- [20] J.-A. Piñeiro, M. D. Ercegovic, and J. D. Bruguera, "Algorithm and Architecture for Logarithm, Exponential, and Powering Computation," *IEEE Trans. Comput.*, vol. 53, no. 9, pp. 1085–1096, 2004, doi: 10.1109/TC.2004.53.

- [21] D. Wang, M. D. Ercegovic, and Y. Xiao, "Complex Function Approximation Using Two-Dimensional Interpolation," *IEEE Trans. Comput.*, vol. 63, no. 12, pp. 2948–2960, 2014, doi: 10.1109/TC.2013.181.
- [22] E. Jamro, K. Wiatr, and M. Wielgosz, "FPGA Implementation of 64-Bit Exponential Function for HPC," in *2007 International Conference on Field Programmable Logic and Applications*, 2007, pp. 718–721, doi: 10.1109/FPL.2007.4380753.
- [23] T. Chi, P. Ru, and S. A. Shamma, "Multiresolution Spectrotemporal Analysis of Complex Sounds," *J. Acoust. Soc. Am.*, vol. 118, no. 2, pp. 887–906, 2005, doi: 10.1121/1.1945807.
- [24] R. Patterson, I. Nimmo-smith, J. Holdsworth, and P. Rice, "An efficient auditory filterbank based on the gammatone function." Speech-Group meeting of Institute of Acoustics on Auditory Modelling, RSRE, Malvern, pp. 1–33, 1987, [Online]. Available: https://www.researchgate.net/publication/245316556_An_efficient_auditory_filterbank_based_on_the_gammatone_function.
- [25] R. F. Lyon, "The AGC Loop Filter," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 331–344.
- [26] H. Li, "Configure the Coefficients for Digital Biquad Filters in TLV320AIC3xxx Family," 2010. [Online]. Available: <http://www.ti.com/lit/an/slaa447/slaa447.pdf>.
- [27] I. Grout, "Introduction to Digital Signal Processing," in *Digital Systems Design with FPGAs and CPLDs*, Newnes, 2008, pp. 475–536.
- [28] J. O. Smith III, "Direct-Form I," *Introduction to Digital Filters with Audio Applications*, 2007. https://ccrma.stanford.edu/~jos/filters/Direct_Form_I.html (accessed Oct. 05, 2018).
- [29] J. O. Smith III, "Direct Form II," *Introduction to Digital Filters with Audio Applications*, 2007. https://ccrma.stanford.edu/~jos/filters/Direct_Form_II.html (accessed May 28, 2019).
- [30] C. L. Philips, J. M. Parr, and E. A. Riskin, "Continuous-Time Linear-Time Invariant Systems," in *Signals, Systems, and Transforms*, 4th ed., Upper Saddle River, NJ, USA: Prentice Hall, 2008, pp. 89–149.
- [31] T. Lizhe and J. Jean, "Infinite Impulse Response Filter Design," in *Digital Signal Processing - Fundamentals and Applications*, 3rd ed., Academic Press, 2019, pp. 315–419.
- [32] R. Bristow-Johnson, "Cookbook formulae for audio equalizer biquad filter coefficients." <http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt> (accessed Oct. 05, 2018).
- [33] J. Hodgson, "Mastering The Final Say," in *Understanding Records: A Field Guide to Recording Practice*, 1st ed., Bloomsbury Academic & Professional, 2014, pp. 189–230.
- [34] S. W. Smith, "Introduction to Digital Filters," in *The Scientist and Engineer's Guide to Digital Signal Processing*, 1st ed., San Diego, CA, USA: California Technical Publishing, 1997, pp. 261–276.
- [35] S. W. Smith, "Applications of the DFT," in *The Scientist and Engineer's Guide to Digital Signal Processing*, 1st ed., San Diego, CA, USA: California Technical Publishing, 1997, pp. 169–184.

- [36] C. M. Rader and B. Gold, "Effects of Parameter Quantization on the Poles of a Digital Filter," *Proc. IEEE*, vol. 55, no. 5, pp. 688–689, 1967, doi: 10.1109/PROC.1967.5634.
- [37] R. Lyons, "Coupled-Form 2nd-Order IIR Resonators: A Contradiction Resolved," 2012. <https://www.dsprelated.com/showarticle/183.php> (accessed Oct. 08, 2018).
- [38] F. Harris and W. Lowdermilk, "Implementing Recursive Filters with Large Ratio of Sample Rate to Bandwidth," in *2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*, 1983, pp. 1149–1153, doi: 10.1109/ACSSC.2007.4487403.
- [39] R. F. Lyon, "The Cascade of Asymmetric Resonators," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 293–298.
- [40] C. L. Philips, J. M. Parr, and E. A. Riskin, "The z-Transform," in *Signals, Systems, and Transforms*, 4th ed., Upper Saddle River, NJ, USA: Prentice Hall, 2008, pp. 546–598.
- [41] R. F. Lyon, "Resonators," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 145–168.
- [42] M. Slaney, "An Efficient Implementation of the Patterson-Holdsworth Auditory Filter Bank," 1993.
- [43] D. A. Depireux, J. Z. Simon, D. J. Klein, and S. A. Shamma, "Spectro-Temporal Response Field Characterization with Dynamic Ripples in Ferret Primary Auditory Cortex," *J. Neurophysiol.*, vol. 85, no. 3, pp. 1220–1234, 2001.
- [44] R. Plomp, "The perception of timbre of steady-state complex tones," *J. Acoust. Soc. Am.*, vol. 86, no. S1, p. S57, 1989, doi: 10.1121/1.2027565.
- [45] T. Nagai, S. Mori, H. Ono, and S. Saito, "Effect of phase on the timbre and detection of timbre of complex tones," *J. Acoust. Soc. Am.*, vol. 58, no. S1, p. S82, 1975, doi: 10.1121/1.2002342.
- [46] J. O. Smith III, "Positive and Negative Frequencies," *Mathematics of the Discrete Fourier Transform (DFT), with Audio Applications --- Second Edition*, 2007. https://ccrma.stanford.edu/~jos/mdft/Positive_Negative_Frequencies.html (accessed Sep. 12, 2019).
- [47] J. O. Smith III, "Analytic Signals and Hilbert Transform Filters," *Mathematics of the Discrete Fourier Transform (DFT), with Audio Applications --- Second Edition*, 2007. https://ccrma.stanford.edu/~jos/r320/Analytic_Signals_Hilbert_Transform.html (accessed Sep. 12, 2019).
- [48] P. Singh, "Studies on Generalized Fourier Representations and Phase Transforms," *arXiv*, pp. 1–15, 2018.
- [49] J. H. Justice, "Analytic signal processing in music computation," *IEEE Trans. Acoust.*, vol. 27, no. 6, pp. 670–684, 1979, doi: 10.1109/TASSP.1979.1163321.
- [50] J. A. Davis, D. E. Mcnamara, and D. M. Cottrell, "Image processing with the radial Hilbert transform: theory and experiments," *Opt. Lett.*, vol. 25, no. 2, pp. 99–101, 2000, doi: 10.1364/OL.25.000099.
- [51] X. Jin and S. Goto, "Hilbert Transform-Based Workload Prediction and Dynamic Frequency Scaling for Power-Efficient Video Encoding," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 31, no. 5, pp. 649–661, 2012, doi:

10.1109/TCAD.2011.2180383.

- [52] P. A. Regalia, S. K. Mitra, and P. P. Vaidyanathan, "The Digital All-Pass Filter: A Versatile Processing Building Block," *Proc. IEEE*, vol. 76, no. 1, pp. 19–37, 1988, doi: 10.1109/5.3286.
- [53] C. Tsai, "Design and Realization of Quadrature Mirror Hilbert Transformers Using Even-Order Elliptic IIR Filters," in *2011 4th International Congress on Image and Signal Processing*, 2011, vol. 1, pp. 2271–2274.
- [54] T. Chi, Y. Gao, M. C. Guyton, P. Ru, and S. Shamma, "Spectro-temporal modulation transfer functions and speech intelligibility," *J. Acoust. Soc. Am.*, vol. 106, no. 5, pp. 2719–2732, 1999, doi: 10.1121/1.428100.
- [55] C. L. Philips, J. M. Parr, and E. A. Riskin, *Signals, Systems, and Transforms*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2008.
- [56] L. L. Beranek and T. J. Mellow, *Acoustics: Sound Fields and Transducers*. Academic Press, 2012.
- [57] H. Zumbahlen, "Other Linear Circuits," in *Linear Circuit Design Handbook*, Elsevier Newnes, 2008, pp. 84–191.
- [58] J. Sollini, G. A. Chapuis, C. Clopath, and P. Chadderton, "ON-OFF receptive fields in auditory cortex diverge during development and contribute to directional sweep selectivity," *Nat. Commun.*, vol. 9, no. 1, pp. 1–12, 2018, doi: 10.1038/s41467-018-04548-3.

6. Pitch Estimation and Classification of Musical Notes

In this chapter, the auditory models described from chapters 3 to 4 are tested with real-world musical signals. The CAR-Lite cochlear model described in chapter 3 is the first module to be tested. The musical signals are monophonic, which means that the input signal contains a musical note from a single musical instrument. These musical signals are sourced from the Real World Computing (RWC) musical instrument database [1]. The fundamental frequencies of the selected musical notes are verified with the PitchPerfect Musical Instrument Tuner software application.

The output of the CAR-Lite cochlear algorithm is transmitted to an algorithm that generates an autocorrelogram containing pitch information. Together, these two algorithms result in the formation of an auditory pitch model, CAR-Lite-ACF, described in chapter 4. Pitch estimation algorithms described in this chapter approximate fundamental frequencies from the autocorrelograms. A classifier is then used to determine whether the estimated fundamental frequencies are related to the ground truths of musical pitch. Here, musical pitch refers specifically to the fundamental frequencies of musical notes. The distinction between the pitch estimation and classifier algorithms allows dedicated functional hardware blocks to be implemented for dedicated processing if required.

The proposed classification algorithms presented in this chapter are implementable on hardware, i.e. a field-programmable gate array (FPGA). The Occam's Razor principle is evoked to ensure that the design of the pitch extraction and classification algorithms remain simple for FPGA implementation while maintaining low computational resource usage. As such, this chapter is divided into four parts to investigate the classification accuracies of the hardware-implementable algorithms.

Section 6.1 details the settings of the CAR-Lite-ACF model to prepare the monophonic musical signal as input to the classifier, as well as the determination of the ground truth information to validate the fundamental frequency, f_0 , of an input signal. Section 6.2 describes algorithms for estimating f_0 from an autocorrelogram generated from the CAR-Lite-ACF model. Section 6.3 presents the results of the classification, while section 6.4 provides the chapter summary and conclusion.

6.1. Pathway to Pitch Estimation: Model Settings and Ground Truth

According to Bigand and Tillmann [2], “musical notes define the smallest building block of Western tonal music”. Plack and Oxenham further remarked [3], “If a sound does not produce a sensation of pitch, it cannot be used to produce a melody.” In other words, musical melody is perceivable through musical pitch [4]. The fundamental frequency of a sound stimulus is one dimension of musical pitch, and is acquirable from an autocorrelogram (AC).

In chapter 4, two methods are presented to calculate an AC matrix. One method involves conventional multiply-accumulate (MAC) operations, and another involves the computationally less expensive logical-AND-accumulate (AAC) operations. However, its capability of representing pitch information from real-world sound signals is unknown. Therefore, AC matrices generated by AAC algorithms are tested in this section to see how well they can represent musical pitch in the form of fundamental frequencies from

monophonic musical notes. This exercise paves the way for the AAC-generated AC matrices to be tested with more complex polyphonic musical signals as well as other complex sound stimuli in the future.

AAC-generated AC matrices are produced from binary spikes calculated from leaky-integrate-and-fire (LIF) neurons representing the auditory nerve (AN) stage of the CAR-Lite cochlear model. These spikes are generated when the internal voltages of the LIF neurons cross a firing threshold. This firing threshold, as well as all other variables in the CAR-Lite-ACF model, are identical to the ones used in chapter 4, i.e. the firing threshold is set at 0.27 V based on 0 dBFS intensity level. This threshold is based on the highest correlation coefficients of the comparison of the AC generated between multiply-accumulate (MAC) and logical-AND-accumulate (AAC) operations – the MAC generated matrices are used as a benchmark in the comparison.

For the exercise in this chapter, the fundamental frequencies extracted from AAC-generated AC are compared with the fundamental frequencies extracted from MAC-generated AC. The CAR-Lite-ACF model that produces the MAC-generated AC matrices (calculated from the half-wave rectified signals output from the CAR-Lite model representing the inner hair cell stage) have the same system parameter settings as the ones described in chapter 4. This comparison provides a quantitative performance measurement of the novel AAC-generated AC in relation to the conventional MAC-generated AC.

The algorithms for acquiring and classifying musical pitch are primarily aimed at the estimation of the fundamental frequency, f_0 , of a monophonic musical signal. The pitch estimation algorithm that extracts f_0 information from the AC matrix is detailed in section 6.2. Subsection 6.2.5 describes the algorithm for classifying musical notes. These algorithms are designed to be implementable on an FPGA. Their designs are based explicitly on selected piano musical notes from the fourth octave and are used for estimating f_0 from musical notes ranging from the second octave to the seventh octave, selected from various musical instruments. The fourth octave range contains the largest number of common musical notes that can be produced across a wide range of musical instruments and is thus, used in many musical pieces. The musical instrument, piano, has the largest range of musical notes that also covers the diverse ranges of notes from many other instruments. Hence, the musical notes from the piano are selected to design the pitch estimation algorithms highlighted in upcoming subsections.

The estimated f_0 from AC matrices are compared with a ground truth vector, f_{GT} , which contains the f_0 corresponding to musical notes. These ground truth frequencies are calculated as follow:

$$f_{GT}(i) = 2^{(i-49)/12} \times 440 \quad (6-1)$$

where i is an index of notes ranging from 1 (A0 – note A at octave 0) to 108 (G#9 – note G sharp at octave 9). Section 6.3 presents the results of classifying musical notes from the AC matrices. The full details of the musical notes and musical instruments used in this exercise as well as their classification accuracy scores are covered in appendices B and C.

6.2. Pitch (f_0) Estimation from Autocorrelogram (AC)

The estimation of an f_0 from a 2D 108×2048 autocorrelogram (AC) calculated from the CAR-Lite-ACF model is mainly dependent on which part of a monophonic musical signal the input signal is extracted from. This section presents the study of which point in a musical note signal provides an ideal representation of the f_0 . A musical note is selected for this study, as mentioned in the next paragraph. Once this location is known, this method of extracting the input signal is applied to other monophonic notes across octaves and other musical instruments for the classification of musical notes described in section 6.3.

A preliminary study is conducted here to determine the ideal location in a musical note to estimate the f_0 . An A4 piano note is arbitrarily selected as the monophonic reference signal. The first step involves selecting multiple starting points in the entirety of the A4 signal. This number is set arbitrarily at fourteen. The locations are also arbitrarily selected, as displayed in Figure 6-1. There are more starting points placed close to the maximum of the A4 signal to understand the impact of the magnitude levels on f_0 estimation. From each of the fourteen starting points, fourteen input vectors, each containing 2,048 samples, are extracted to generate fourteen ACs. Figure 6-2 displays all fourteen ACs cross-correlated to one another.

The region of interest in this similarity matrix is the area with the highest correlation coefficients (CC) that indicates the highest degree of similarity, which happens to be from windows 3 to 6 and from windows 10 to 13. Windows 3 to 6 is close to the peak positive (maximum) point in the sound signal (denoted by the red asterisk in Figure 6-1), but windows 10 to 13 have no prominent feature to enable signal extraction. Hence, the maximum point is selected as the starting point of the input signal to generate an AC. An example of an AC generated by this method is shown in Figure 6-3(a) using the same A4 note signal. As described in chapter 4, summing all the rows of the AC in Figure 6-3(a) results in a temporal profile displayed in Figure 6-3(c). From the temporal profile, an f_0 of 442 Hz ($\approx 96 \text{ kHz}/217$) is estimated, which is close to the ground truth f_0 of A4 note at 440 Hz ($\approx 2^{(0-49)/12} \times 440$). Subsection 6.2.1 further elaborates on this technique.

Subsections 6.2.1 to 6.2.3 present three characteristics for estimating f_0 , which lead to the formation of six f_0 estimation algorithms, as summarised in subsection 6.2.4. The classifier algorithm is presented in subsection 6.2.5. Subsection 6.2.6 presents an FPGA implementation of the three characteristic algorithms as well as the classification algorithm.

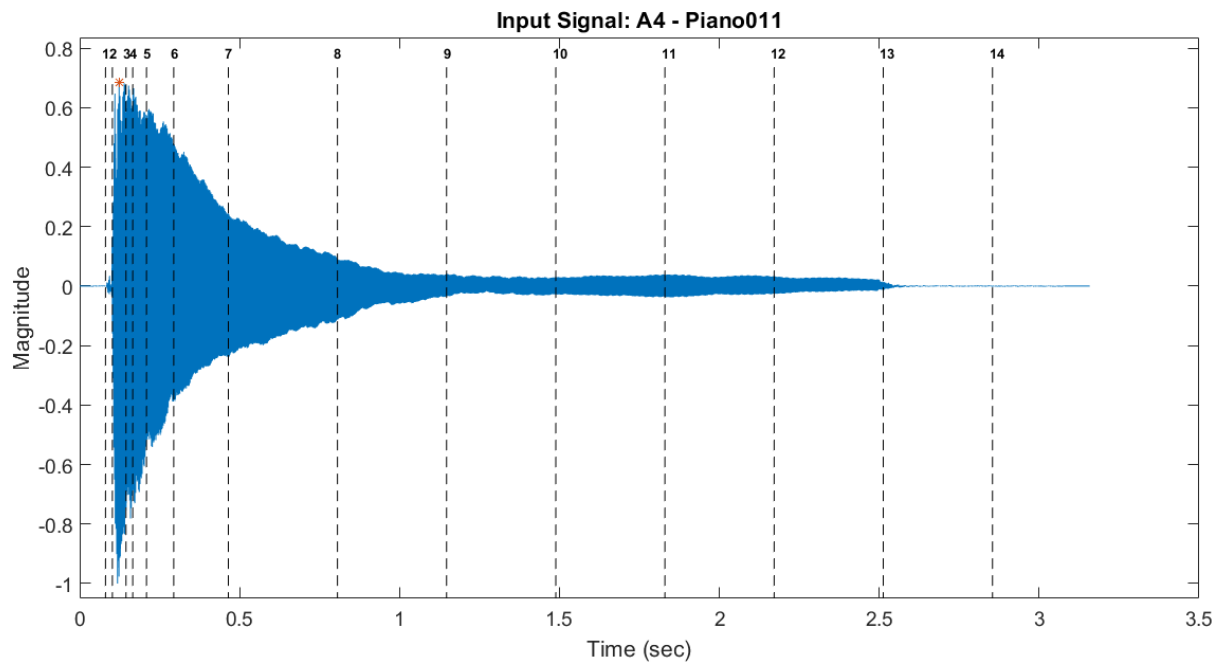


Figure 6-1: Fourteen arbitrarily selected discrete starting points from the A4 musical signal played on the piano for ACF windowed analyses. The red asterisk, "*", indicates the starting point of the input signal ultimately selected for generating an AC.

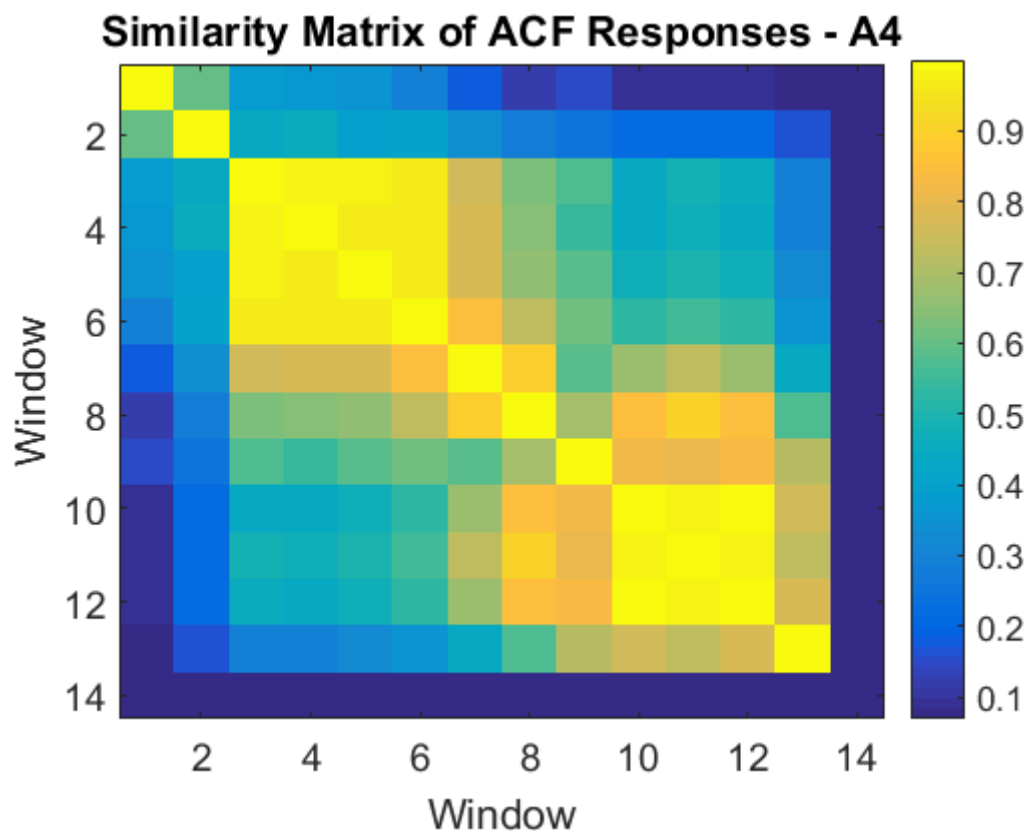


Figure 6-2: A similarity matrix displaying the correlation coefficients between fourteen autocorrelograms (ACs) indicating the degree of similarity between the ACs. The ACs are generated from fourteen input signals selected from the fourteen arbitrarily selected starting points scattered throughout the monophonic musical signal of piano note, A4, in Figure 6-1.

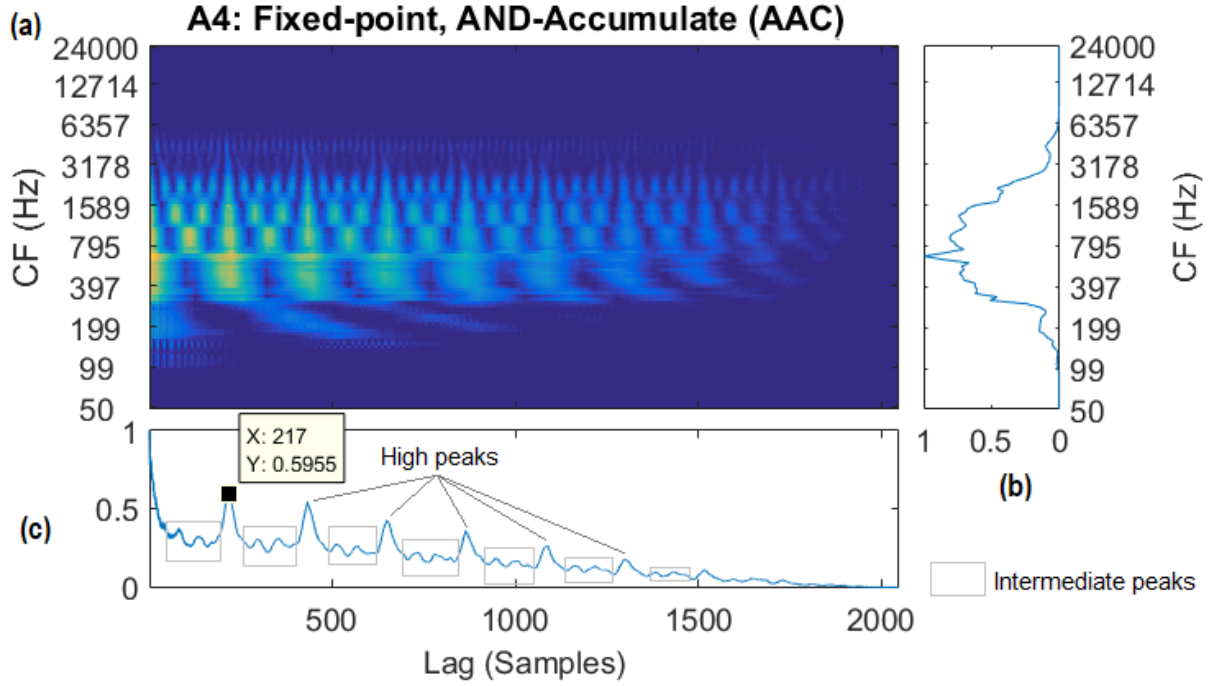


Figure 6-3: (a) Autocorrelogram (AC) response calculated from the maximum magnitude point of an A4 piano note. Calculation is in fixed-point arithmetic from leaky-integrate-and-fire (LIF) neurons using AND-accumulate (AAC) operations representing auditory nerve (AN) signals originating from the CAR-Lite-ACF model. The AC can be represented as (b) a spectral profile, and (c) a temporal profile.

6.2.1. Peak-Picking

A fundamental frequency, f_0 , can be calculated from the peaks in the temporal profile of an AC. The peaks themselves can be divided into two types: high-level and intermediate-level, as seen in the labels in Figure 6-3(c). All algorithms for calculating f_0 from an autocorrelogram generated from the CAR-Lite-ACF model involve only the high peaks. The upcoming paragraphs describe the detection of these high peaks.

The automated process of finding peaks in the temporal profile of the AC involves a peak-picking algorithm [5] implemented in an iterative loop spanning the length of a temporal profile. The algorithm has two parts: (1) the detection of all peaks in the temporal profile of an AC, and (2) differentiate high peaks from ordinary peaks. The first part involves in the formation of a vector, n_p , containing sample numbers, n , of detected peaks. For every iteration, it uses the current sample and past two samples to determine a peak. This iterative three-sample peak detection is a simple form of peak detection that is implementable on hardware:

$$n_p = \begin{cases} n - 1, & m_{n-2} < m_{n-1} \geq m_n \\ 0, & \text{otherwise} \end{cases} \quad (6-2)$$

where m is a magnitude in the temporal profile at sample $n - k$; k can be any number between 0 to 2 to select any one of three adjacent samples and p is a peak index.

The second part of the peak-picking algorithm involves the formation of a new vector, n_{hp} , containing the samples of only the high peaks extracted from all the peaks found in the temporal profile in n_p . Like equation (6-2), this part uses the current peak and two past detected peaks for every iteration within the same loop that houses equation (6-2):

$$n_{hp} = \begin{cases} n_{p-1}, & \Delta_{m-} < \Delta_{m0} > \Delta_{m+} \\ & \text{or } \Delta_{m-} < \Delta_{m0} < \Delta_{m+} \\ \text{unassigned}, & \text{otherwise} \end{cases} \quad (6-3)$$

$$\Delta_{m-} = m(n_{hp-1}) - m(n_{p-2}) \quad (6-4)$$

$$\Delta_{m0} = m(n_{hp-1}) - m(n_{p-1}) \quad (6-5)$$

$$\Delta_{m+} = m(n_{hp-1}) - m(n_p) \quad (6-6)$$

where Δ_k is the difference in the magnitude of a last known high peak, $m(n_{hp-1})$, and a detected peak at $m(n_{p-k})$, in the temporal profile. Two conditions are formulated in equation (6-3) to define a high peak as observed in the temporal profile of the A4 piano note. The first condition checks if the middle peak is the highest out of the three adjacent peaks. The second condition checks if the first peak is the highest out of all three peaks with decreasing magnitudes. If any of these two conditions are met, the sample number (index) of the detected peak is recorded in the n_{hp} vector. When this algorithm is first initiated, the first peak detected is set as a high peak by default (see Figure 6-4 for reference).

The fundamental frequency, f_0 , of a perceived pitch [6] is defined by two adjacent high peaks in the n_{hp} vector. The high peak to high peak period, T_{pp} , and its corresponding frequency, f_{pp} , between two adjacent high peaks in the temporal profile [7] are calculated as:

$$T_{pp} = \frac{n_{hp} - n_{hp-1}}{f_s} \quad (6-7)$$

$$f_{pp} = \frac{1}{T_{pp}} \quad (6-8)$$

where f_s is the sampling rate.

Figure 6-4 displays the extracted high peaks using the abovementioned peak-picking method from a temporal profile of a piano note, A4. The temporal profile in Figure 6-4(a) is the profile from Figure 6-3(c) (generated using AAC operations) that has been further conditioned by a 2nd-order low-pass filter to remove high-frequency artefacts, as suggested in chapter 4. Applying equations (6-7) and (6-8) to all adjacent pairs of high peaks result in the f_{pp} vector. The mean and the median of the f_{pp} vector are found to be 446 Hz and 444 Hz, respectively. Using pitch matching classifier from subsection 6.2.5, the closest ground truth musical frequency, f_{GT} , defined by equation (6-1), which corresponds to these values, is 440 Hz. Incidentally, this is the fundamental frequency of the A4 note.

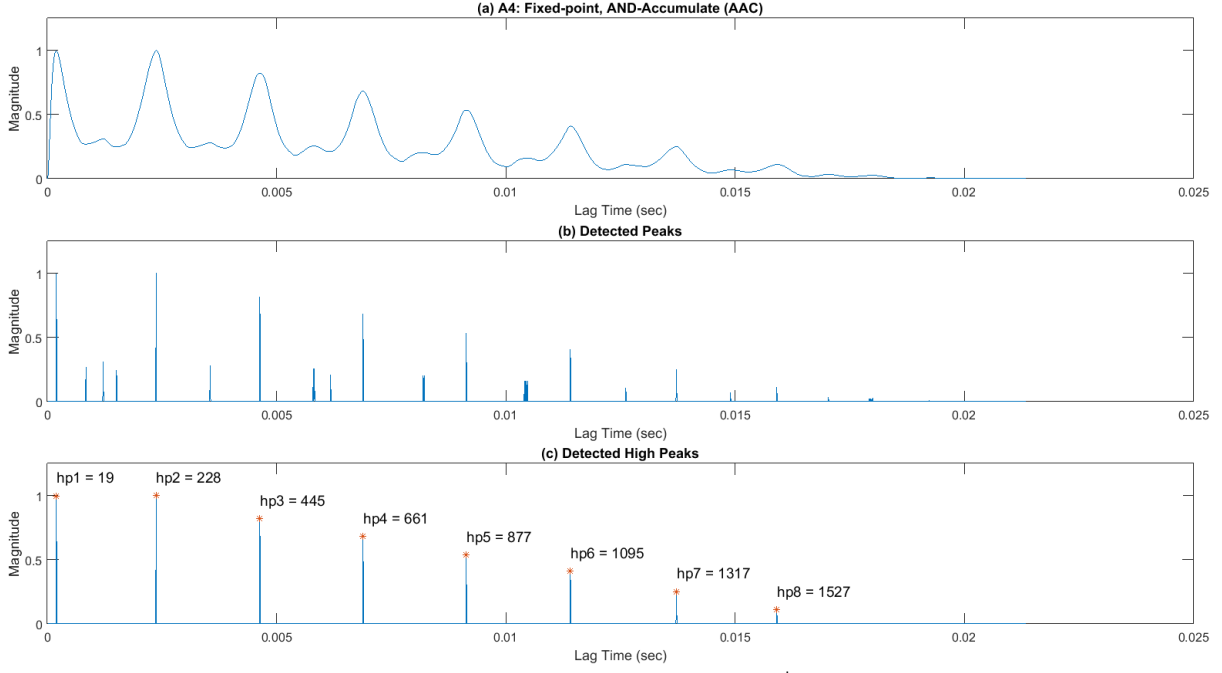


Figure 6-4: Automatic peak detection of an A4 piano note: (a) from a 2nd-order low-pass filtered temporal profile computed using AND-accumulate (AAC) operations in fixed-point arithmetic; (b) display of all detected peaks from the temporal profile using equation (6-2); (c) display of detected high peaks, hp_i , from the temporal profile using equation (6-3). The numbers on the top of the high peaks are sample numbers that can be used to calculate the f_{pp} using equations (6-7) and (6-8), which corresponds to the estimated f_0 of the A4 note.

6.2.2. Weighting High Peaks

If the distance between any two detected high peaks in a temporal profile results in T_{pp} , which in turn leads to frequency, f_{pp} , then only one pair of successive peak magnitudes can be ideally selected to achieve this. The preferred pair would be hp_1 and hp_2 , at the start of the temporal profile since it is the quickest to find out of all other high peaks. They are also likely to be the most reliable as the high peaks here are distinguishable from adjacent intermediate peaks, i.e. magnitude difference between the first high peak, hp_1 and any intermediate peaks before the second high peak, hp_2 , is significant.

There are instances, where anomalous peaks with intermediate magnitudes are detected that result in a falsely estimated f_{pp} . One example of this is observable from the temporal profile of an E4 piano note in Figure 6-5(a). The peak-picking algorithm from equations (6-2) to (6-6) has detected an additional erroneous peak, hp_2 . Calculating f_{pp} from this erroneous point, using either hp_1 or hp_3 , results in an f_{pp} that does not match the ground truth of 330 Hz (f_{GT} of E4). A three-step solution is taken to resolve this issue.

The first part of the solution is to sort the magnitudes of the high peaks in a decreasing order. Doing so also rearranges their respective sample indices. This action puts the high peaks at the start of the sorted vector, followed by the intermediate peaks. The advantage of this arrangement is that the resultant f_{pp} calculations between the high peaks can be done iteratively at fixed increments of 1 instead of variable increments if they are not sorted. The search for the number of high peaks in a group is set arbitrarily to nine. In other words, only the top-9 high peaks are regarded as reliable in the calculation of f_{pp} and thus, are retained. All other peak indices outside the detected high peaks indices are set to 0.

The second part of the solution consists of comparing eight f_{pp} values calculated from these nine peaks with the elements of the ground truth musical frequency vector, f_{GT} . The f_{GT} elements closest to f_{pp} values are recorded in a separate vector, f_{ppm} . Their corresponding indices are recorded in k_{ppm} :

$$f_{ppm}(i) = \begin{cases} f_{GT}(j-1), & \text{if } \Delta f_1 < \Delta f_2 \\ f_{GT}(j), & \text{if } \Delta f_1 > \Delta f_2 \\ f_{GT}(j), & \text{otherwise} \end{cases} \quad (6-9)$$

$$k_{ppm}(i) = \begin{cases} j-1, & \text{if } \Delta f_1 < \Delta f_2 \\ j, & \text{if } \Delta f_1 > \Delta f_2 \\ j, & \text{otherwise} \end{cases} \quad (6-10)$$

$$\Delta f_1 = f_{pp}(i) - f_{GT}(j-1) \quad (6-11)$$

$$\Delta f_2 = f_{GT}(j) - f_{pp}(i) \quad (6-12)$$

where the i^{th} peak-to-peak frequency and index from the sorted vector containing decreasing peak magnitudes, is closest to the j^{th} f_{GT} .

The third part of the solution is to assign weights to the top-8 detected musical peak-to-peak frequency, f_{ppm} . The weights are required to ensure that if all eight f_{ppm} elements are unique, then the most influential element in the f_{ppm} vector is used for representing f_0 . Otherwise, the most commonly occurring element in the f_{ppm} vector is the estimated f_0 . Hence, the first element in f_{ppm} corresponding to the first two detected high peaks in the temporal profile is assigned the highest weight of 8 points. The weight reduces by 1 for the next element in the f_{ppm} vector. So, the eighth f_{ppm} element has a weight of 1. A winner-take-all (WTA) algorithm determines the f_{ppm} element with the highest score as the estimated f_0 . The score, s_{fppm} , for the i^{th} f_{ppm} is calculated using:

$$s_{fppm}(i) = \sum_k i \cdot w_{fpp}(i) \quad (6-13)$$

where i is the index of an element in f_{ppm} ; w_{fpp} is a position-based weight of an f_{ppm} element.

One advantage of this exercise is that if a false positive f_{ppm} element is detected with a significant weight, and an alternate f_{ppm} element is detected several times with lower rank, the combined high s_{fppm} score may elevate the alternate f_{ppm} element to a higher priority. An illustration of this effect is observable from the detected high peaks of the E4 piano note in Figure 6-5(b), where the first f_{ppm} element of 349 Hz is a false positive ($s_{fppm} = 2 \times 7 = 14$) but is overridden by the combined score ($s_{fppm} = 3 \times 6 + 4 \times 5 = 38$) of the two lower

weighted f_{ppm} elements at 330 Hz. In this case, the higher-scoring f_{ppm} element corresponding to 330 Hz matches the f_{GT} of E4 (330 Hz).

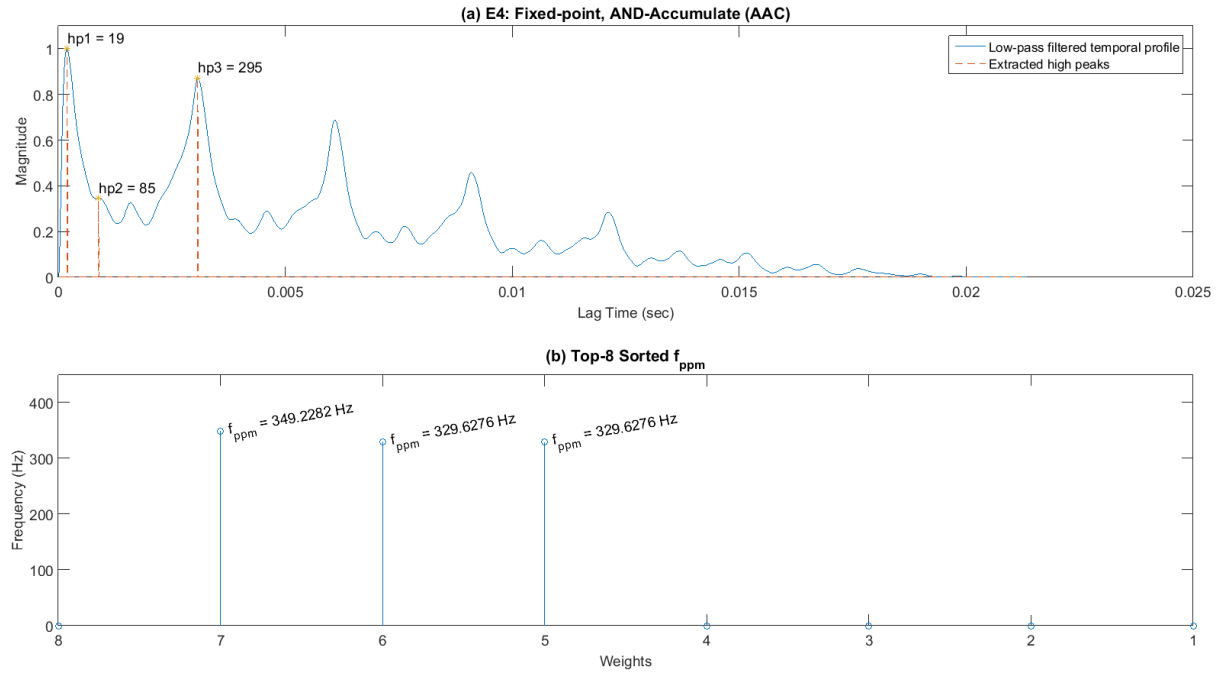


Figure 6-5: (a) High peaks detected from the temporal profile of piano note, E4 ($f_{GT} = 330$ Hz), and (b) top-8 weighted peak-to-peak musical frequency, f_{ppm} , sorted based on decreasing magnitude of detected high peaks.

6.2.3. Threshold-Bound Search

In the context of the threshold-bound algorithm presented in this subsection, the following definitions are used: An inharmonic frequency is calculated between any two adjacent high peaks. It is not equal to $f_{ppU}(1)$ [a unique frequency element from the f_{ppm} vector described in the next paragraph] as well as being not an integer multiple of $f_{ppU}(1)$. In contrast, a harmonic frequency is calculated from any two adjacent high peaks and is also not equal to $f_{ppU}(1)$ but is an integer multiple of $f_{ppU}(1)$.

One issue with the weight-based winner-take-all (WTA) approach described in section 6.2.2 is that the number of inharmonic frequency elements, $f_{ppm}(> 1)$ – [calculated from high peaks beyond the first two], can outnumber the principal frequency element, $f_{ppm}(1)$, calculated from the first two high peaks. This is observable in Figure 6-6(b), where the combined score is larger for the 208 Hz $((5 \times 4) + (3 \times 6) = 38)$ frequency element than the 311 Hz $(2 \times 7 = 14)$ element. This algorithm results in a wrongly estimated f_0 of 208 Hz instead of one that matches the ground truth frequency at 311 Hz. To alleviate this issue, a new vector, f_{ppU} , is formulated, which only holds unique elements of f_{ppm} , i.e. only a single frequency element is stored in f_{ppU} even if repeated frequency elements are encountered in f_{ppm} . Another vector, k_{ppU} , holds the sample indices, k , corresponding to the unique frequency elements in f_{ppU} :

$$k_{ppU}(i) = \begin{cases} k(j), & \text{if } f_{ppm}(i) \neq f_{ppm}(j) \\ -1, & \text{if } f_{ppm}(i) = f_{ppm}(j) \end{cases} \quad (6-14)$$

where -1 indicates repeated elements from the vector that are omitted.

From the unique frequency elements, two conditions are defined to filter out inharmonic f_{ppU} elements. The first condition tests for the presence of harmonics related to the principal frequency element, $f_{ppU}(1)$, which has the most significant weight, w_{fpp} , indicating the highest priority and probable frequency related to the ground truth frequency, f_{GT} . This test is characterised using a modulo operation to see if the unique frequency elements are an integer multiple of $f_{ppU}(1)$:

$$f_{mod} = f_{ppU}(i) \bmod f_{ppU}(1) \quad (6-15)$$

A zero in f_{mod} , indicates the presence of a harmonic component of $f_{ppU}(1)$, whereas a non-zero k_{mod} indicates the presence of an inharmonic component.

In the RWC database [1], all twelve piano notes (in the 011PFNOF.wav file) in the fourth octave contain inharmonic components with respect to $f_{ppU}(1)$. Each of more than half of the twelve notes contains at least two frequency components that are separated by a semitone, which is the minimum interval between two consecutive notes in a Western musical scale. Figure 6-5(b) illustrates an example of such a behaviour from the E4 note, where the difference between the 350 Hz (f_0 of F4) component is one semitone apart from the 330 Hz (f_0 of E4) component – musically, E4 and F4 are one semitone apart. In other words, although there is a difference in these two frequency components, both of them are still essential in defining the f_0 of the input signal.

The second condition involves calculating the distance between the principal frequency element, $f_{ppU}(1)$, and an inharmonic component. The distance plays a significant part in either the retention or omission of the inharmonic component for consideration in the f_0 estimation as an inharmonic component. In other words, if an inharmonic component is near $f_{ppU}(1)$, it is included in the estimation of f_0 ; otherwise, it is not included:

$$|f_{ppU}(i) - f_{ppU}(1)| > th_{semitone} \quad (6-16)$$

where $th_{semitone}$ is a threshold of one semitone. Hence, if any f_{ppU} element [excluding $f_{ppU}(1)$], whether it is a harmonic of $f_{ppU}(1)$ or not, is one semitone apart from $f_{ppU}(1)$, it is retained in the f_0 estimation. If it is more than one semitone apart, it is omitted from the f_0 estimation. Here, the f_0 estimation uses the weights method described in subsection 6.2.2. Applying this threshold-bound algorithm to the D#4 piano note results in the omission of the 208 Hz elements, as displayed in Figure 6-6(c). This omission is due to the 208 Hz (f_0 of G#3) element being seven semitones apart from the 311 Hz (f_0 of D#4) element. As a result, only the 311 Hz, which is considered as a high priority element in terms of its magnitude, is retained.

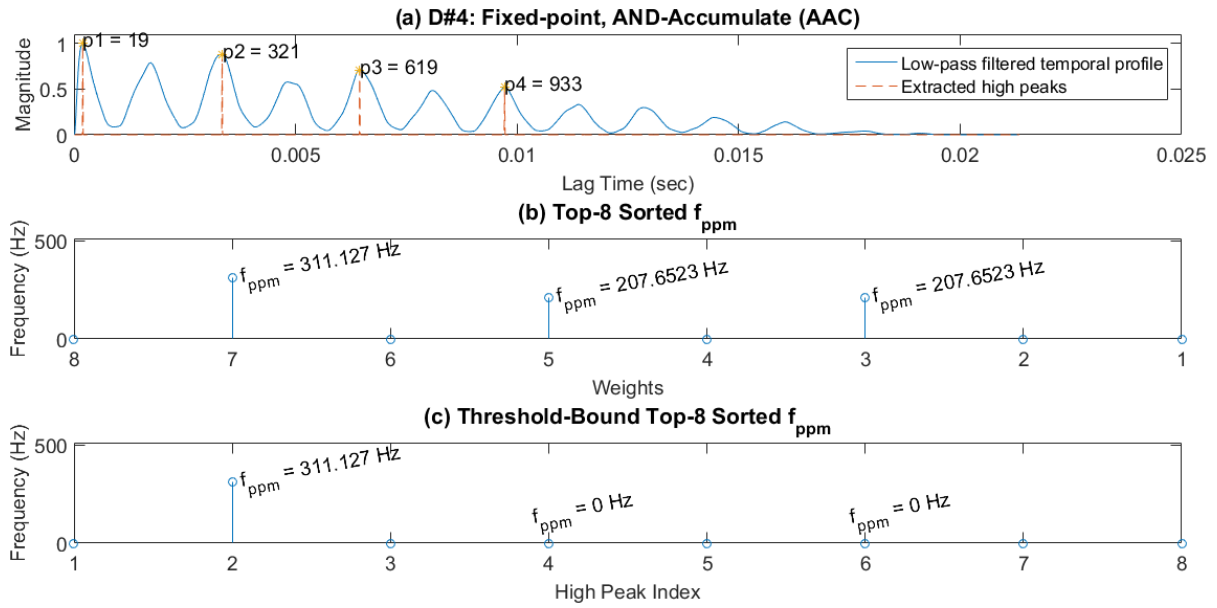


Figure 6-6: (a) High peaks detected from the temporal profile of piano note, D#4 ($f_{GT} = 311 \text{ Hz}$); (b) top-8 weighted peak-to-peak musical frequency, f_{ppm} , sorted based on decreasing magnitude of the detected high peaks; (c) a threshold-bound omission of 208 Hz elements, which is larger than 1 semitone from the $f_{ppU}(1)$, of 311 Hz calculated from the first two detected high peaks.

6.2.4. Summary of Algorithms

Algorithm	Description
1	f_0 estimated from the first two high peaks only using peak-picking algorithm from subsection 6.2.1.
2	f_0 estimated from the most commonly occurring frequency component calculated from all the peaks encountered throughout the temporal profile, using the peak-picking algorithm from subsection 6.2.1.
3	f_0 estimated from the most commonly occurring frequency calculated from only the high peaks encountered throughout the temporal profile – non-weighted winner-take-all (WTA) algorithm described in subsection 6.2.2.
4	f_0 estimated from the weighted winner-take-all (WTA) algorithm applied to frequency components calculated from only the high peaks, as described in subsection 6.2.2.
5	f_0 estimated from the threshold-bound algorithm applied to frequency components calculated from only the high peaks, as described in subsection 6.2.3.
6	f_0 estimated from the conditional combination of algorithms 4 and 5. If there are eight different frequency elements encountered from all the nine high peaks (top-9 peaks) encountered, then execute algorithm 4; else if less than eight different frequency elements are calculated (less than nine high peaks encountered), execute algorithm 5.

Table 6-1: Summary of algorithms for estimating f_0 from the temporal profile of an autocorrelogram (AC) for use in classifying musical notes, whose results are presented in section 6.3.

Table 6-1 summarises the algorithms for estimating an f_0 described in subsections 6.2.1, 6.2.2, and 6.2.3. The f_0 calculated from these algorithms are used for the classification of musical notes, whose results are presented in section 6.3.

6.2.5. Classifier Algorithm

This subsection describes the operation of a pitch matching algorithm for classifying musical notes implemented in Matlab. The subsequent paragraph elaborates on the contents of the classification algorithm, which is implemented in both Matlab and SystemVerilog. All ACs corresponding to the musical notes from various musical instruments used as inputs to the classifier are generated offline and permanently stored. When the classification algorithm starts, it loads all the ACs in working memory and their respective temporal profiles are calculated before entering an iterative loop. In the iterative loop, each of the six f_0 estimation algorithms described in Table 6-1 are run on the temporal profiles.

After an f_0 is estimated, it is checked against a vector of ground truth musical notes frequency, f_{GT} . When an estimated f_0 equals an f_{GT} for a specific musical note, it is considered a successful match. Otherwise, it is considered a failed match. This pitch matching algorithm is a simple derivative of the k-nearest neighbour method [8] of classifying musical notes. This process is done iteratively for each estimated f_0 . The results are consolidated outside the iterative loop, and the classification accuracy scores are shown based on the following criteria: the algorithms used for generating the ACs using either multiply-accumulate (MAC) or logical-AND-accumulate (AAC) operations together with either floating-point (software) or fixed-point (hardware) arithmetic.

6.2.6. FPGA Implementation

This subsection describes the FPGA implementation of the three AC f_0 estimation algorithms described from subsections 6.2.1 to 6.2.3 using SystemVerilog. It also describes the pitch matching classifier described in subsection 6.2.5 for classifying the estimated f_0 . The frequency calculations in these algorithms are replaced with periodicity calculations, which alleviates the need to use computationally expensive division operations and rely simply on addition and subtraction operations. The FPGA used here is an Altera Cyclone V GX starter kit with a 5CGXFC5C6F27C7 FPGA chip operating at a system clock rate of 100 MHz. A single top module hosts a finite state machine with 14 states, which are processed serially. The combined latency of all 14 states is 140 ns, where each state requires 1 clock cycle (10 ns) to process. However, the algorithms are designed to run iteratively in multiple cycles for the duration of a given input signal. Hence, the combined latency varies based on the input signal attributes, such as the length of the input signal and the number of peaks present in the temporal profile of the AC.

The FSM starts with state 0, where all the constants are initialised. States 1 to 3 detect peaks from the input temporal profile signal based on the peak-picking algorithm in subsection 6.2.1 and record their respective sample indices, y_p . States 4 to 7 sort the detected peaks in the order of decreasing magnitude and store them in y_{xhp} , as described in subsection 6.2.2. Their corresponding sample indices, y_{hp} , are also recorded. States 8 to 11 calculate the peak-to-peak distance between the sorted sample numbers, y_{dhp} , and search for the distances that is in agreement with the distance of the two largest magnitudes based on the description in subsection 6.2.3. The number of equal distances is counted and compared with one another. The count for the equal distance between the two largest magnitude is given priority over other distances. If this count, y_{WTA} , is the largest, then the distance is known as the estimated fundamental period, $y_{TO_estimate}$, corresponding to a fundamental frequency. Otherwise, the highest count is chosen. The classifier is

implemented in states 12 and 13 that predicts a musical fundamental period, $y_{TO_predicted}$, close to $y_{TO_estimate}$.

As an illustration of the FPGA implementation of the AC f_0 estimation and classification algorithms, a 48-sample input temporal profile signal is generated based on the properties of the D#4 piano note illustrated in Figure 6-6. Figure 6-7 displays the input signal and the three red arrows depict the three common peak-to-peak distances that yield a fundamental period of 10 samples. Figure 6-8 depict the corresponding FPGA output vector waveform of the AC f_0 estimation and classification.

$y_{detected}$ in Figure 6-8(a) illustrates nine pulses indicating nine detected peaks from the input signal. y_{p1} to y_{p9} in Figure 6-8(b) depicts the sample numbers of these nine peaks. These peaks are then sorted in decreasing magnitude order. y_{xhp1} to y_{xhp5} depict the top-five largest of these magnitudes, and y_{hp1} to y_{hp5} are their corresponding sample numbers. y_{dhp1} and y_{dhp2} display the differences between the sample numbers of the neighbouring high peaks. A y_{WTA} of 3 and $y_{TO_estimated}$ of 10 indicate that a peak-to-peak distance of 10 samples occurs most frequently (3 times) in the input signal as opposed to any other peak-to-peak distances. This result corresponds to the three distances illustrated by three arrowed red lines having a distance of 10 samples in Figure 6-7. Given ten known trained classes of known fundamental periodicity ranging from 6 to 15 samples, the predicted fundamental period, $y_{TO_predicted}$, is correctly depicted at 10 samples.

Table 6-2 presents the FPGA computational resources used for implementing the AC f_0 estimation and classification algorithms. A temporal profile with 2,048 samples would increase the resources utilised. However, the scaled-down 48-sample input signal was selected to showcase that the algorithms are implementable and operable on hardware. The AC f_0 algorithms require no digital signal processors (DSPs), as no multiplication operations exist in the algorithms. Despite the low logic utilisation by the algorithms, the power consumed is at 239 mW.

FPGA	Number of ALM Utilised (out of 29,080)	Number of Registers Utilised	Number of DSPs Utilised (out of 150)	Power (mW)
Altera Cyclone V	1,879 (6.5%)	1,810	0 (0%)	239

Table 6-2: Computational resources used by an Altera Cyclone V FPGA to implement the AC f_0 estimation and classification algorithms.

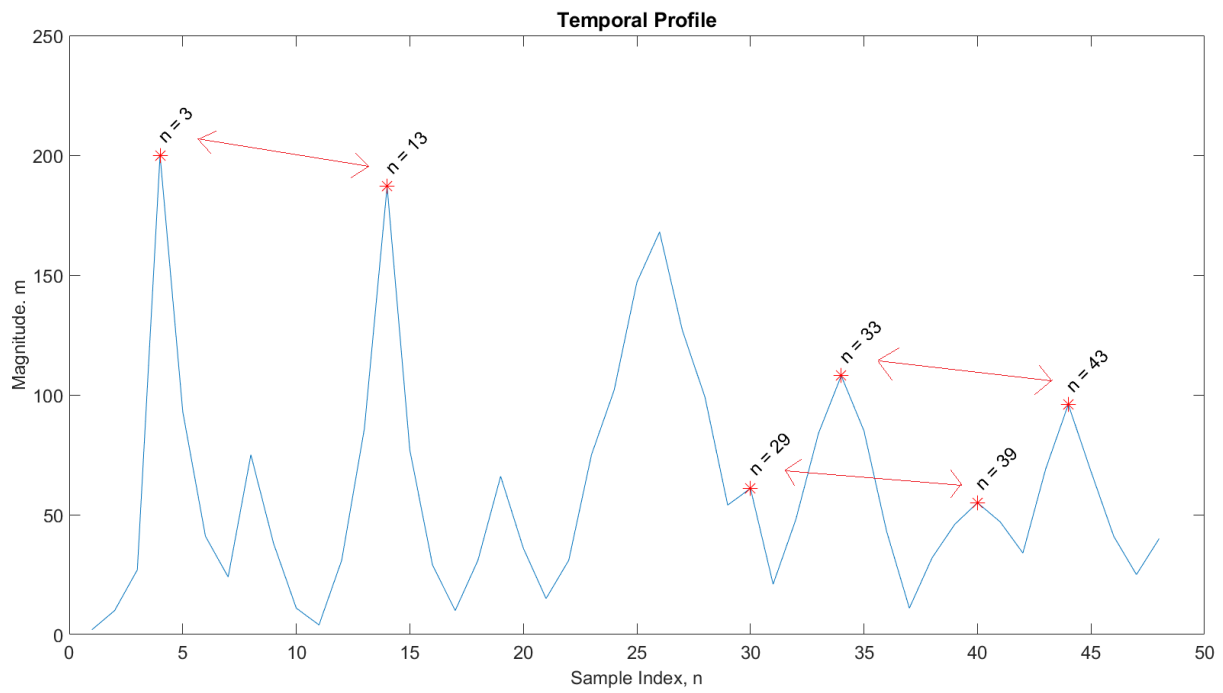


Figure 6-7: A 48-sample temporal profile generated based on properties of the D#4 piano note illustrated in Figure 6-6. The red arrows show the common peak-to-peak distances that yield the estimated f_0 .

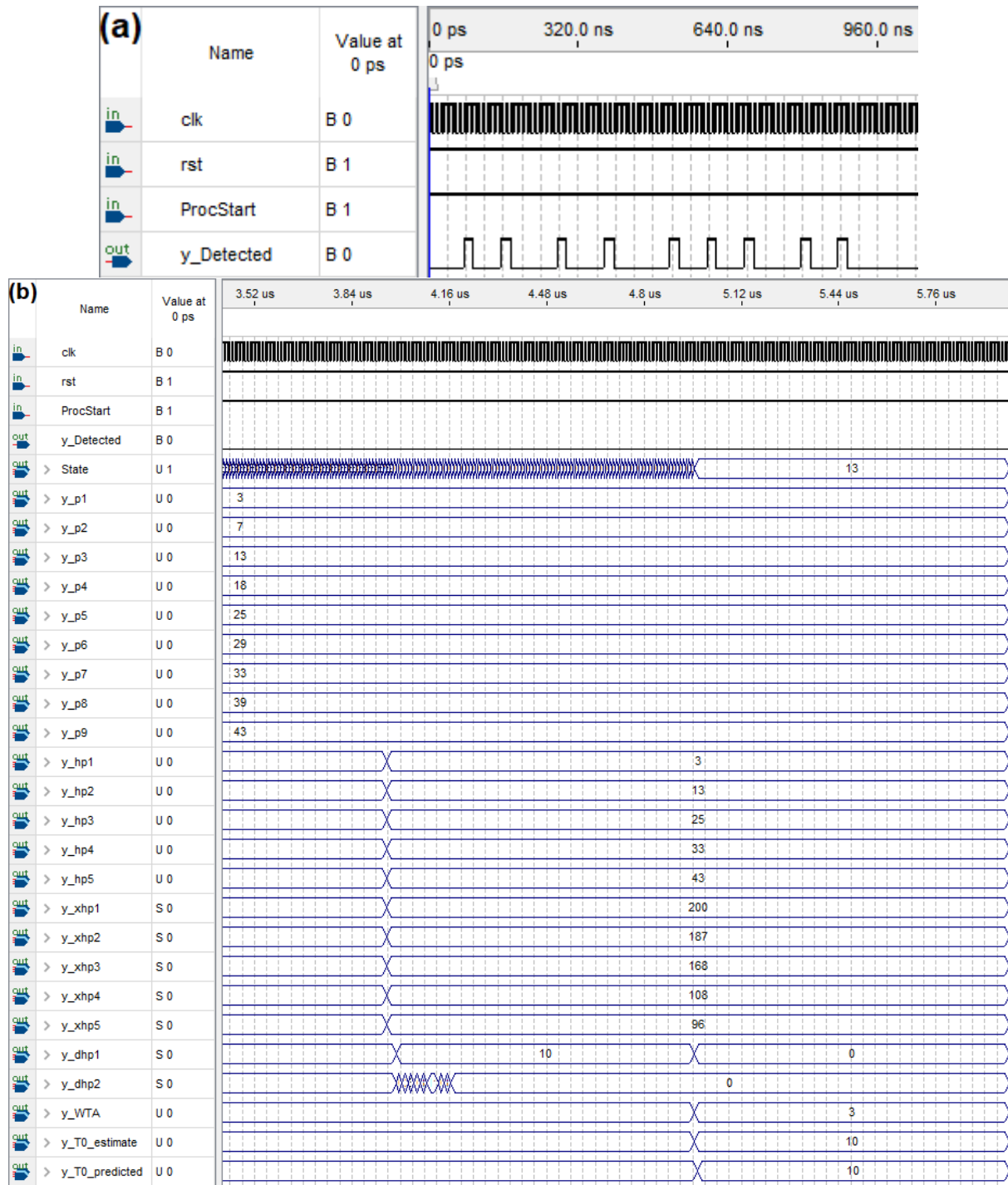


Figure 6-8: Output vector waveform of an FPGA implementation of the AC f_0 estimation and classification algorithms described from subsections 6.2.1 to 6.2.5: (a) The 9 pulses depict the 9 peaks detected within the first 960 ns. (b) Classification results depict the correctly predicted musical T_0 of 10 samples with the estimated T_0 of 10 samples used as an input into the classifier. The estimated T_0 is determined based on the most commonly occurring peak-to-peak distance, which is depicted quantitatively from the counter, y_WTA .

6.3. Results and Evaluation

The following three subsections present the results of the classification of monophonic musical notes. Subsection 6.3.1 presents the classification results using the pitch estimation algorithms from section 6.2 on the autocorrelogram responses from the CAR-Lite-ACF model. The classification results of the responses from the model are differentiated based on

the computational method of generating the output autocorrelogram: either floating-point (software) and fixed-point (hardware) arithmetic calculated with either multiply-accumulate (MAC) or logical-AND-accumulate (AAC) operations. Subsection 6.3.2 presents the classification results based on the varying intensity and signal-to-noise ratio (SNR) levels of musical notes as well as the automatic gain control (AGC) algorithm presented in appendix A applied to the notes. Subsection 6.3.3 presents the difference in sizes of the output autocorrelograms used in the classification exercise in subsection 6.3.1. Subsection 6.3.4 compares the results of the classification presented in subsection 6.3.1 with work done by other authors.

6.3.1. Accuracy of Pitch Estimation (0 dBFS Intensity Level)

The results of the classification of musical notes displayed below are broken into three octave-group ranges: low (octaves 2 and 3), as illustrated in Figure 6-9; middle (octaves 4 and 5), as illustrated in Figure 6-10; high (octaves 6 and 7), as illustrated in Figure 6-11. Each octave comprises 12 musical notes in the following order based on increasing frequency: C, C#, D, D#, E, F, F#, G, G#, A, A#, B. Figure 6-12 displays the overall performance based on the mean results across all three groups. The result for each octave group is calculated as a mean of the results across various musical instruments listed in the three respective tables as follows: mean results in Figure 6-9 calculated from Table B-1; Figure 6-10 from Table B-2; Figure 6-11 from Table B-3. An accuracy score in each of the three tables results from a specific algorithm listed in Table 6-1 for a temporal profile from an autocorrelogram (AC) output matrix. It is further segregated based on how each output matrix is generated, i.e. either by multiply-accumulate (MAC) operations or by AND-accumulate (AAC) operations in either floating-point (software) or fixed-point (hardware) arithmetic.

The musical instruments are selected based on the availability of musical notes for the specific octave groups in the RWC database [1]. Ten octaves (12 notes per octave) from ten musical instruments are selected from the RWC database representing low-range octave groups (octaves 2 and 3) that dominantly play musical notes in the bass (low frequency) range, as seen in Table B-1. Musical notes from most musical instruments in the RWC database reside in the mid-range octave groups (octaves 4 and 5). Given the abundance of musical notes in this range, eleven octaves from ten random musical instruments are selected, as projected in Table B-2. The high-range octave groups (octaves 6 and 7) has the least number of musical instruments available to generate musical notes, and so only three musical instruments are selected for this range, as observed in the low number of musical instruments listed in Table B-3.

The algorithm with the weakest performance (lowest accuracy score) across the three-octave groups is algorithm 2, as observed in Figure 6-12. This poor performance is expected as the f_0 here is estimated from all the peaks found in the temporal profile, which include high and intermediate peaks. In contrast, f_0 estimated from the first two high peaks alone from algorithm 1 yields approximately two times higher accuracy scores. The next higher-performing score is from algorithm 4 – weighted high peaks, and a winner-take-all (WTA) algorithm, whose mean accuracy score is 27% higher than that of algorithm 1. This performance is followed by algorithm 5 – the threshold-bound algorithm, whose score is 5% higher than algorithm 4. The non-weighted high peaks WTA algorithm (algorithm 3) and the

hybrid combined algorithms 4 and 5, known as algorithm 6, have equal performance, both outranking the accuracy score of algorithm 5 by 10%.

Since the algorithms for estimating an f_0 are designed based on notes from octave 4 of a piano, the highest results observed in Figure 6-12 are expectedly from the mid-range, precisely due to the algorithms 3 and 6. Furthermore, the accuracy scores do not vary much across musical instruments as well as between octaves 4 and 5, as observed in Table B-2. In contrast, there is a significant difference in accuracy scores based on octave 6 notes (high-range) generated by a flute and a piano, as highlighted in Table B-3 for all algorithms. In the same table, another significant difference in accuracy scores is observable between octaves 6 and 7. A similar observation is made for the low-range octave group from Table B-1, where there is a significant performance difference between string instruments (piano and guitars) and wind instruments (trombone and tenor saxophone) as well as between octaves 2 and 3. These contrasts show the lack of robustness of the algorithms to predict f_0 across octaves and musical notes. Hence, the f_0 estimation algorithms have to be designed around various musical instruments besides the piano to enhance their robustness of accurately approximating f_0 from these instruments. These observations also suggest that the timbre and pitch relation across octaves represented in autocorrelograms affect the performance of these algorithms to predict an f_0 , as was reported in [9].

Figure 6-13 illustrates the capability of the temporal profile calculated from the autocorrelogram (AC) to represent low pitch (C2 note) and high pitch (C7 note) musical piano notes. A low pitch note has a low fundamental frequency, whereas a high pitch note has a high fundamental frequency. For a low pitch note, the peak-to-peak distance on the AC is irregular, as observed in Figure 6-13(a). This observation indicates the presence of inharmonic frequencies, which are frequencies that are multiple integers of other frequencies besides f_0 . Thus, the peak-to-peak f_0 estimation becomes highly obscured as they usually contain information on the inharmonic frequency. Instead, the estimation algorithms have to be designed to consider peaks not directly adjacent to a single peak. For example, the f_{GT} for C2 of 65 Hz corresponds to a peak with an insignificant magnitude immediately after the 0.015 s mark in Figure 6-13(a) with respect to the first detected peak at p1. Since the f_0 estimation algorithms described in this chapter analyse only peak-to-peak information, this results in an inaccurate f_0 estimation. In contrast, the peak-to-peak distance on the AC is regular for a high pitch note, as observed in Figure 6-13(b). This observation indicates that the f_0 is not affected by inharmonic frequencies for this specific high pitch monophonic signal and that it can be estimated accurately by using peak-to-peak algorithms described in this chapter.

Despite the significant performance differences across octaves and musical instruments, the mean accuracy score difference across the computational response per algorithm (MAC floating-point vs MAC fixed-point vs AAC floating-point vs AAC fixed-point) is generally on par with one another with small differences across them for all the octave groups, as observed in Figure 6-12. This observation is mainly applicable to algorithms 2, 3, 5, and 6 on responses calculated from the hardware-based AAC operations. The only exception is for algorithm 4, where the hardware-based AAC operations have the lowest score of the four computation types with the difference between the closest computation (hardware-based MAC operation) being only 3%. Overall, the results indicate that the hardware-based AAC operations have a highly similar performance with the MAC operations

in terms of representing monophonic musical pitch information in addition to consuming lesser hardware resources than the latter, as presented in chapter 4.

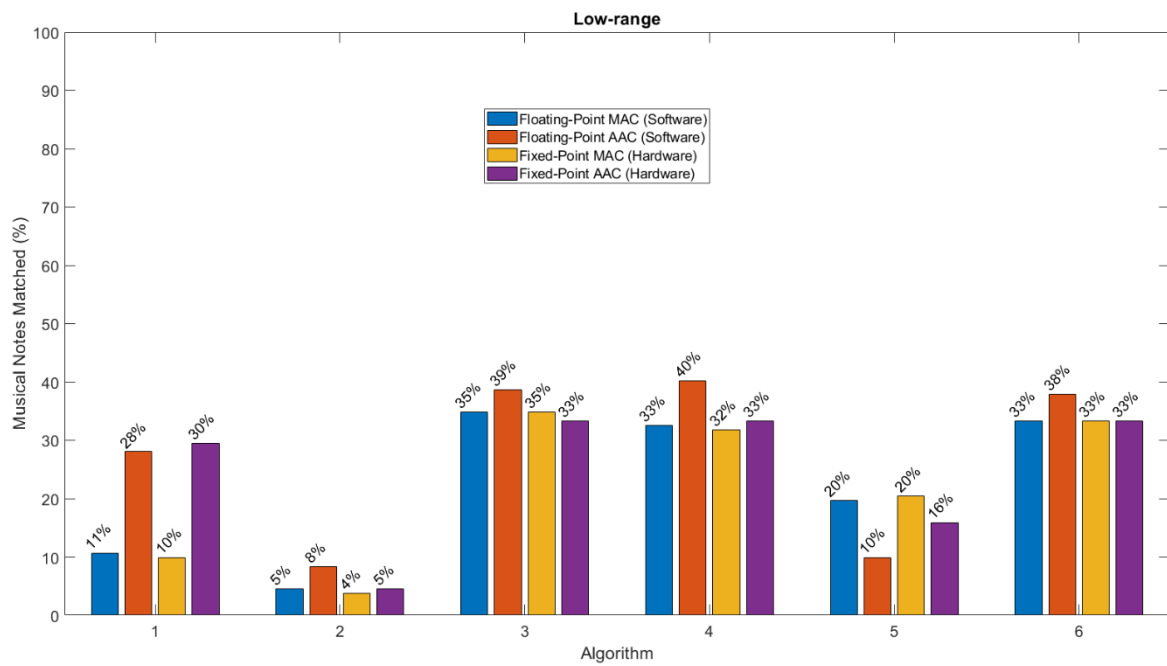


Figure 6-9: Classification of musical notes average scores for low-range octave groups 2 and 3.

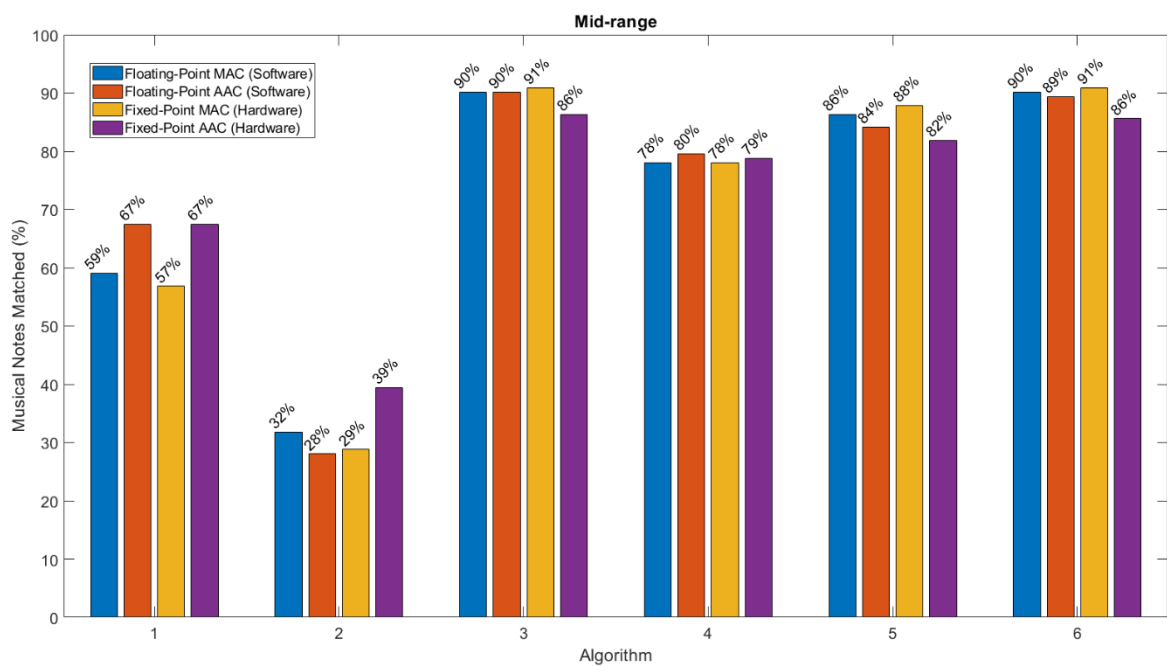


Figure 6-10: Classification of musical notes average scores for middle-range octave groups 4 and 5.

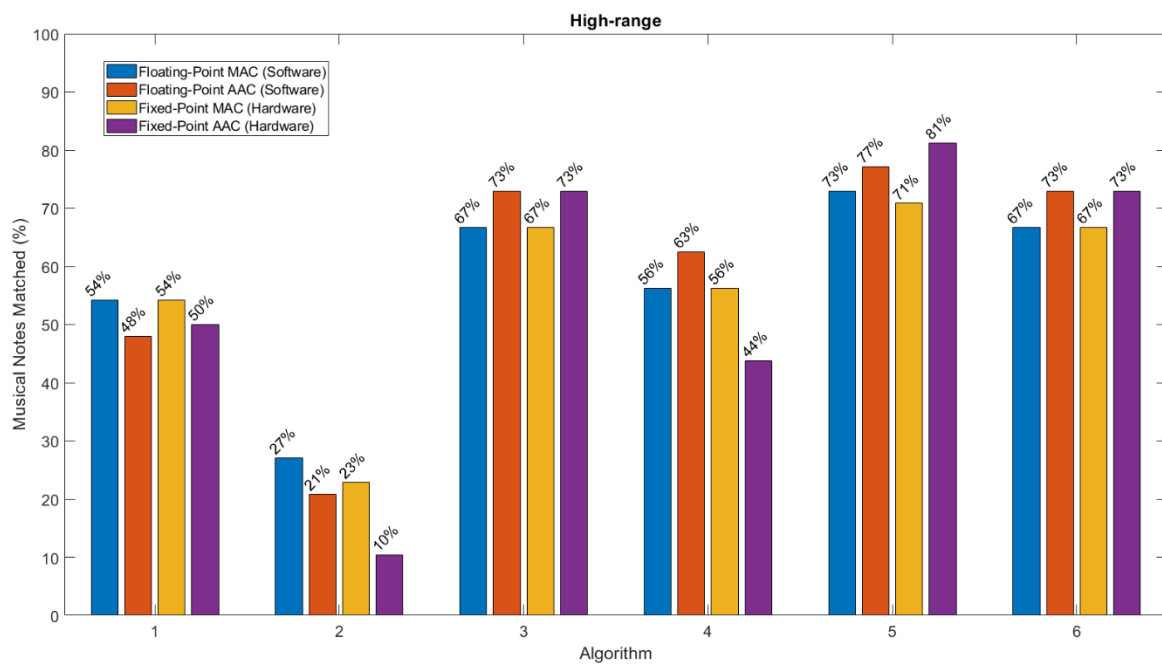


Figure 6-11: Classification of musical notes average scores for high-range octave groups 6 and 7.

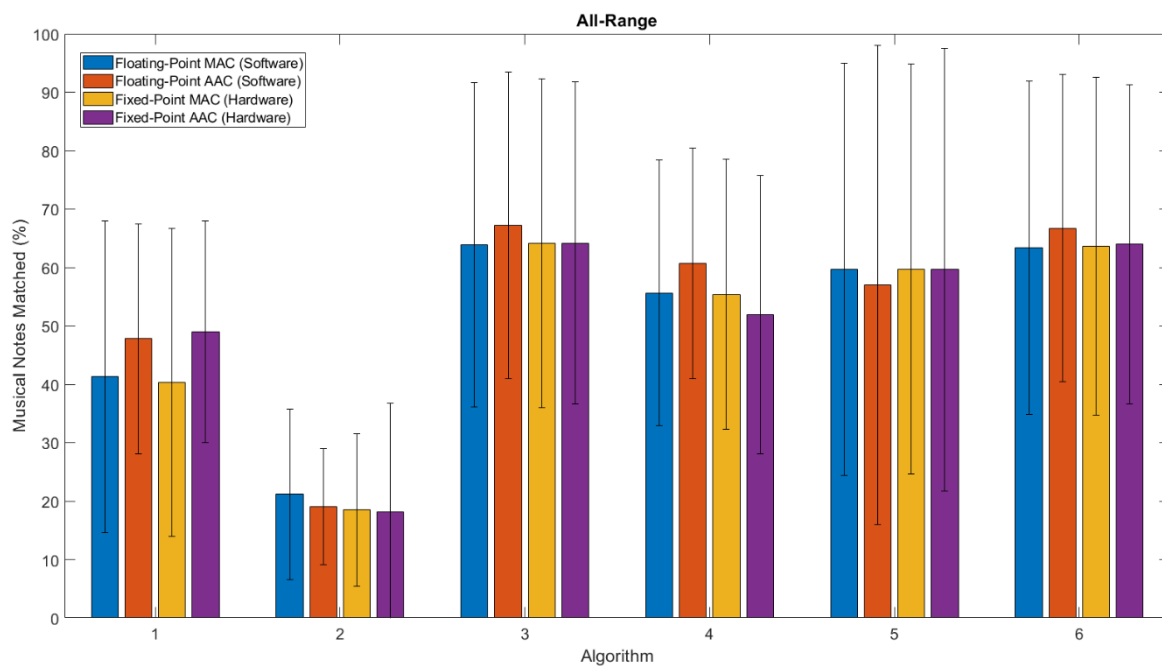


Figure 6-12: Classification of musical notes average scores across all three-octave groups with the vertical lines on the bars showing standard deviation.

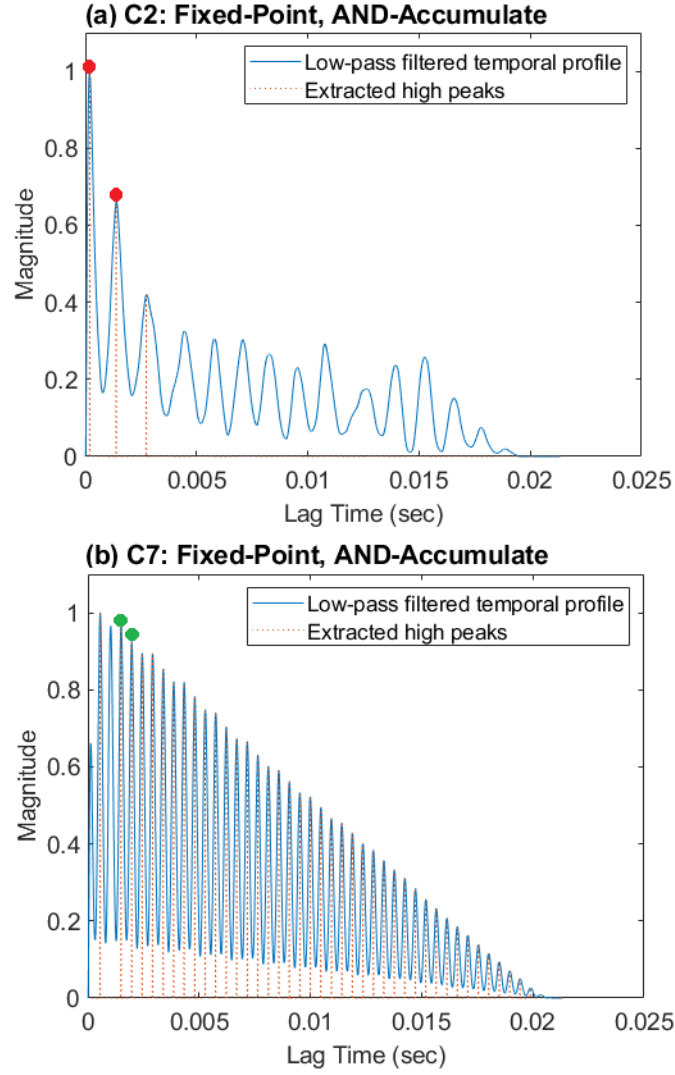


Figure 6-13: A comparison of low pitch (C2) and high pitch (C7) musical notes represented on a temporal profile: (a) note C2; (b) note C7. Note: Red dots indicate wrong f_0 estimation, while green dots indicate correct f_0 estimation.

6.3.2. Varying Intensity and Noise Levels

As real-world sound signals vary in intensity levels and are affected by noise, it is necessary to understand how much varying the intensity and signal-to-noise ratio (SNR) levels affect f_0 estimation and classification accuracy. This section presents the results of such effects. Additionally, the influence of an automatic gain control (AGC) algorithm to condition the amplitudes of the input signals is also presented. Computation of the autocorrelograms is segmented based on combinations of two calculation types: either MAC or AAC and either floating-point or fixed-point. Twelve piano notes are selected: A4 – G#4.

Section 3.2.8 presented the limitation of the dynamic range (DR) of the CAR-Lite model when the intensity levels of the input signals vary. This limitation is observed as the saturation of amplitude levels of the BM signal when the input signal is saturated at intensity levels above 0 dB full-scale (FS). An AGC algorithm is utilised to condition an input signal so that its amplitude levels are maintained between $2^{N-1} - 1$ and -2^{N-1} for an N bits fixed-point implementation. For a 16 bits BM signal, this range is from 32,767 to -32,768. Details of the AGC algorithm are presented in appendix A. This algorithm is not implemented on FPGA

because the Cyclone V FPGA starter kit [10] used for the implementation of the CAR-Lite-ACF model has an audio codec that comprises an AGC circuit [11].

Figure 6-14 display the waveforms of a C4 piano note at an intensity level of 0 dBFS. The left column displays the waveforms when the AGC is disabled while the right column displays the waveforms when the AGC is enabled. The varying gain conditioning applied by the AGC normalises the energy across frequencies, as observed in the broader area of energy present in the autocorrelogram in Figure 6-14(d). The temporal profile generated from this autocorrelogram introduces more intermediate peaks, as observed in Figure 6-14(f), which consequently reduces the accuracy scores of the pitch estimation algorithms (3, 5, and 6) with the three highest performance described in Figure 6-12. Their inability to estimate the correct f_0 is evident in the lower classification accuracy scores in Figure 6-16(a) as opposed to when AGC is disabled – shown in the scores of algorithms 3, 5, and 6 under “AGC enabled”.

Applying the second-order low-pass filter (LPF) to the temporal profile in Figure 6-14(f) reduces the magnitudes of all the peaks, including the largest peak close to time lag 0 resulting in the temporal profile in Figure 6-14(h). Note that the position of this peak varies but is close to 0. In the latter temporal profile, both intermediate and high peaks are still present, unlike the temporal profile of Figure 6-14(g) generated with the AGC disabled. However, with the suppression of the largest peak close to time lag 0 after applying the LPF, the time lag of the new largest peak from the time lag precisely at 0 corresponds to the pitch of the note. Hence, a new algorithm (# 7) is devised to estimate pitch frequency by simply finding the time lag of the maximum peak and taking its inverse, as illustrated in Figure 6-14(h). This algorithm is applicable only when the AGC is enabled, as it has a poor performance when the AGC is disabled. These results are observable in Figure 6-16, as shown in the contrasting scores of algorithm 7 between “AGC off” and “AGC on”.

Figure 6-15 presents the classification accuracy across varying intensity levels ranging from -20 dBFS to 20 dBFS in increments of 10 dBFS for every SNR level. Each bar score in Figure 6-15 is the average of the accuracy scores of the four pitch estimation algorithms (3, 5, 6, and 7) in Figure 6-16. The SNR levels range from 20 dB to -20 dB in increments of 20 dB, where the input signal is mixed with white Gaussian noise. For every SNR level, three accuracy scores are recorded from each specific pitch estimation algorithm at a specific intensity level, as presented in appendix C. Each bar score in Figure 6-15(b), (c), and (d) is an average all the scores across the four pitch estimations algorithms under a specific intensity level.

When the AGC is disabled, the accuracy scores in Figure 6-15 generally remain uniform at levels from -20 dBFS to 0 dBFS. At 10 dBFS, they improve for both MAC-based and AAC-based temporal profiles, but at 20 dBFS, the accuracy only improves for AAC-based temporal profiles while the performance remains the same for MAC-based temporal profiles. Input signals larger than 0 dBFS are amplified but have saturated amplitudes due to a limited DR as opposed to signals below 0 dBFS that have no saturation issues. These two combined effects of amplification and saturation improve the performance of the pitch estimation algorithms to detect peaks in the temporal profile adhering to the ground truth f_0 regardless of the SNR levels. Since this attribute is more prominent for the temporal profiles calculated with AAC operations, the AAC-based temporal profiles represent pitch information more reliably than MAC-based temporal profiles, especially for signals at high intensity levels

with limited DR. Noise introduced to the AAC-based temporal profiles via quantisation errors between consecutive amplitude levels is a likely cause for this improvement.

When the AGC is enabled, the accuracy scores under the four computational settings in Figure 6-15 remain the same throughout the intensity levels. This is because the AGC algorithm normalises the amplitudes of the input signal to the full dynamic range of the CAR-Lite model at 96 dB regardless of the intensity levels, which ensures uniform performance across levels. The accuracy scores are generally lower than the scores when the AGC is disabled, but this is attributable to the lack of robustness of the pitch estimation algorithms – the algorithms (3, 5, and 6) were designed using signals that are not conditioned by the AGC with the exception of algorithm 7. Also, the MAC-based temporal profiles have higher accuracy scores than the AAC-based temporal profiles. This performance contrast is due to the sophistication of the AGC algorithm that varies the gain of an input signal across several bandwidths of frequencies, which results in the weakening of salient foreground harmonic components while strengthening background subharmonic components, as observed in Figure 6-14(d) in comparison to Figure 6-14(b). Since the AAC computation of temporal profile captures salient features, the lack of such salient features in an input signal reduces the performance of the pitch estimation algorithm when estimating f_0 from AAC-based temporal profiles.

Figure 6-17(a) displays how much the accuracy scores vary across the intensity levels. Here, each bar score is calculated from the mean of the standard deviations calculated from the four settings under the four algorithms across the four SNR levels. A low bar score shows performances of the settings are close to the average scores of the settings, and a high bar score indicates more fluctuations in the performance. Generally, the accuracy scores are more varying when the AGC is enabled than when it is disabled, which indicate that the performance of the pitch estimation and classification is more consistent when the AGC is disabled than when the AGC is enabled. The exception is at 20 dBFS, where the scores are more varying when the AGC is disabled than when it is enabled. This large varying factor is due to the significant contrast in accuracy scores, especially with the high scores at 20 dBFS noiseless and saturated input signals and the low scores at -20 dB SNR.

Figure 6-16 presents the pitch estimation performance for each of the four computational settings under the four algorithms averaged across intensity levels from -20 dBFS to 20 dBFS for four SNR levels. Each bar is the average of the scores from all five intensity levels under each of the four pitch estimation algorithms. The result indicates how algorithms perform with a white Gaussian noise that may be part of real-world sound signals. The accuracy scores of algorithms 3, 5, and 6 are higher when the AGC is disabled than when the AGC is enabled. This is the case only when there is no noise and at 20 dB SNR. At higher noise levels below 20 dB SNR, the accuracy scores of the three algorithms are further reduced but are similar to one another regardless of the AGC. In contrast, the accuracy scores of algorithm 7 are expectedly higher when the AGC is enabled than when the AGC is disabled because the algorithm is designed for use specifically with the AGC enabled. Overall, the performance of the four algorithms degrades as the noise levels increase.

Figure 6-17(b) displays the mean of the standard deviation of accuracy scores across the four SNR levels calculated from the four computational settings under the four algorithms (3, 5, 6, and 7) and five intensity levels (-20 dBFS to 20 dBFS in increments of 10 dBFS).

When the AGC is disabled, the accuracy scores of the algorithms is most varied at 0 dB SNR, which shows the impact of the white Gaussian noise on maximising the contrast in performance across the settings. The variations in accuracy scores remain small for other noise levels. When the AGC is enabled, the variations in accuracy scores and the accuracy scores themselves reduce with decreasing SNR. So, assuming that the signals used in this section are akin to real-world signals, the pitch estimation algorithms presented in this chapter do not perform well when they are exposed to real-world signals with increasing noise.

Generally, varying intensity levels affects the perceived loudness of a sound signal. Although loudness, together with pitch affects timbre cue such as brightness [12], loudness does not affect the pitch of a sound signal, especially its fundamental frequency, f_0 [13]. Hence, it can also be concluded that varying intensity levels of a sound signal does not affect its f_0 . This conclusion is validated by the results of the classification of musical notes across varying intensity levels in this subsection. Furthermore, the introduction of increasing noise levels to the musical signals degrades the performance of the classification due to distortion of the wavelengths of the musical signals and consequently their f_0 representation.

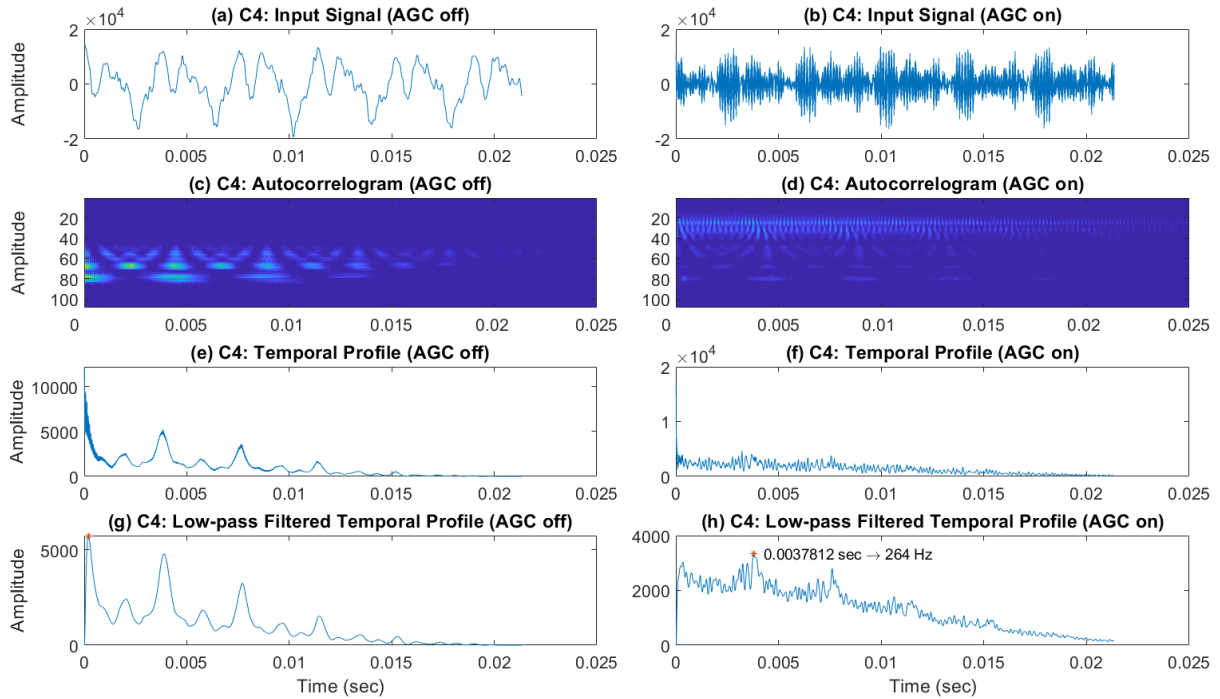


Figure 6-14: Fixed-point response of the CAR-Lite-ACF model calculated with the AAC algorithm of a C4 piano note (0 dBFS) as input. The left column displays output response with the note signal input directly into the model, and the right column shows note signal conditioned by an automatic gain control (AGC) algorithm. * indicates maximum amplitude found in (g) and (h). Note that the ground truth for C4 is $f_0 = 262$ Hz.

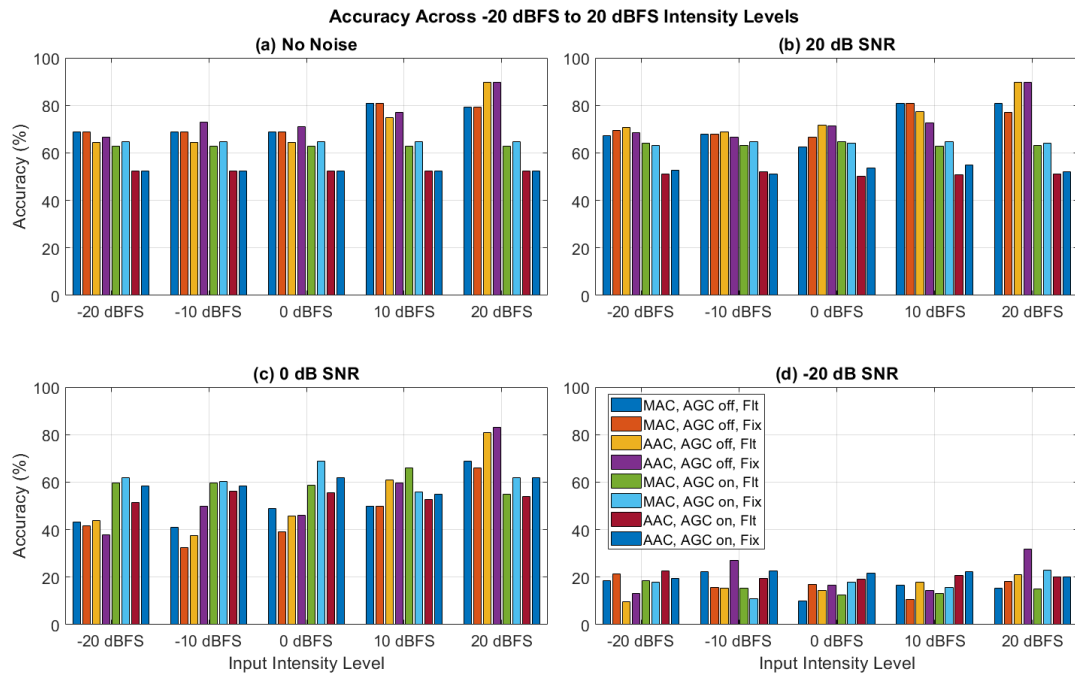


Figure 6-15: Musical pitch estimation and classification accuracy results across multiple intensity levels for four signal-to-noise ratios (SNR). Each score under every SNR level is an average of three scores (see appendix C) as well as the average of scores from the four algorithms (3, 5, 6, and 7).

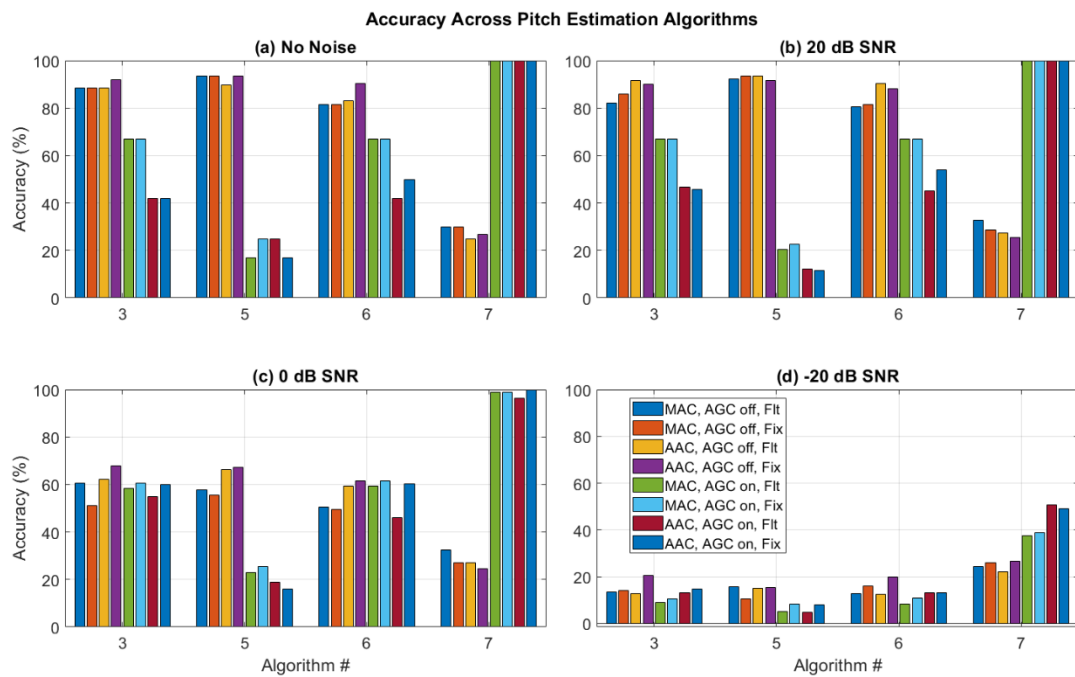


Figure 6-16: Musical pitch estimation and classification accuracy scores across algorithms for varying signal-to-noise ratios (SNR). Each score under every SNR level is an average of three scores, as well as across intensity levels (see appendix C for details).

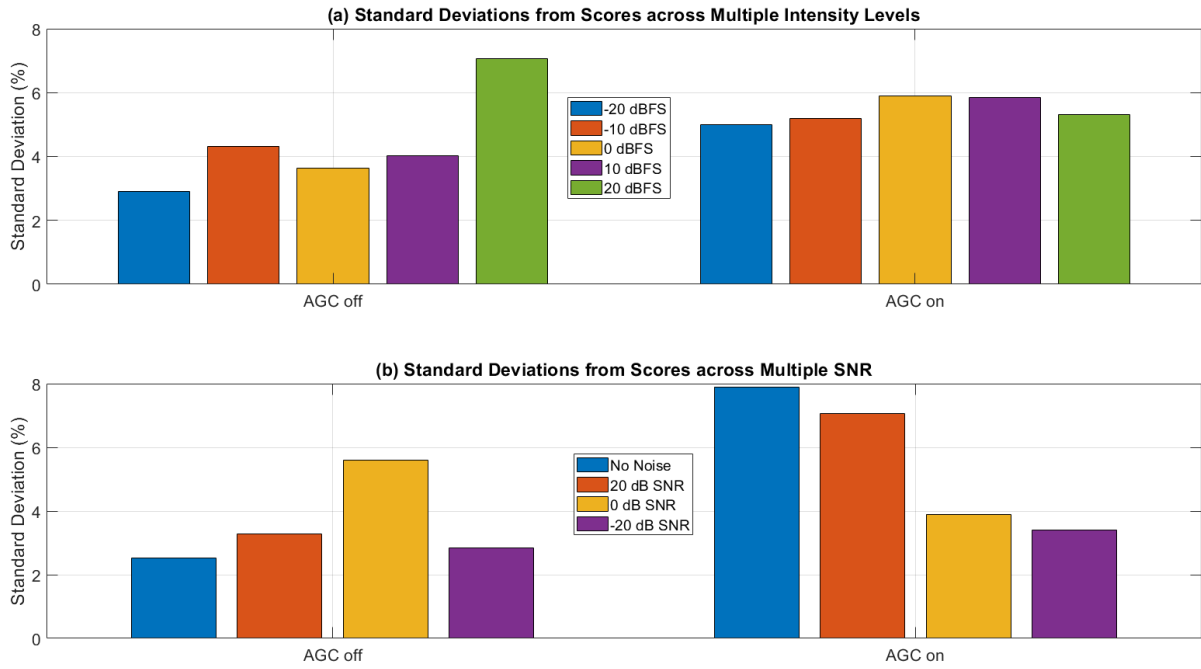


Figure 6-17: Mean of the standard deviation of accuracy scores from Figure 6-15 and Figure 6-16 shown across (a) intensity levels and (b) SNR levels. Mean values here are calculated from the accuracy scores of four computational settings under algorithms 3, 5, 6, and, 7, and (a) across SNR, and (b) across intensity levels.

6.3.3. Autocorrelogram File Sizes

The storage of 2D autocorrelograms for pitch estimations of music notes is crucial for the identification of patterns of musical notes. A similar application is the use of over millions of 2D spectrograms for representing musical tracks in an extensive database for automatic music identification [14]. The storage size of these spectrograms is dependent on how they are computed, i.e. using either floating-point or fixed-point arithmetic, which consequently indicates how much electronic memory is required. Similarly, the same notion can be used on the autocorrelograms used in this chapter for pitch estimation. Thus, this section presents the difference in storage sizes of autocorrelograms computed in floating-point and fixed-point arithmetic.

Figure 6-18 illustrates the file sizes of the autocorrelograms (AC) at 0 dBFS used as inputs to the musical notes pitch estimation algorithm described in subsection 6.3.1. ACs generated in software-based floating-point with multiply-accumulate (MAC) operations have the largest file sizes due to each sample in an AC matrix having formed with a double-precision number as opposed to a single-precision number for each sample in an AC generated in hardware-based fixed-point arithmetic. As a result, floating-point generated ACs with MAC operations require more than 1.5 times storage space than fixed-point generated ACs with MAC operations.

For the logical-AND-accumulated (AAC) generated ACs, each sample is an accumulation of single bits as opposed to double and single precision numbers for floating-point MAC and fixed-point MAC ACs respectively. This difference results in AAC-generated ACs having even lower storage than the MAC-generated ACs, i.e. a fixed-point AAC-generated AC requires 24 times less storage space than a floating-point MAC-generated AC.

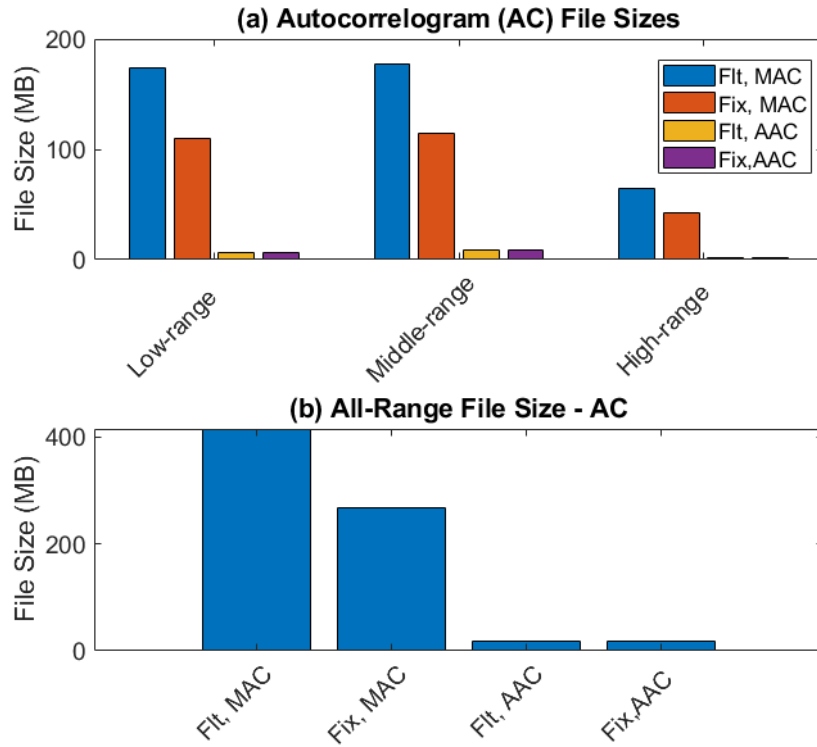


Figure 6-18: (a) Autocorrelogram (AC) file sizes distributed across 0 dBFS three-octave group ranges for four different computation types; (b) Total AC file sizes for all three octave groups differentiated by the four computation types. Fit: Floating-point; Fix: Fixed-point; MAC: Multiply-accumulate; AAC: Logical AND-Accumulate.

6.3.4. Comparison with Other Models: Accuracy

This subsection presents the comparison of the classification of monophonic musical notes with two other studies at only 0 dBFS. The reason behind this scarcity in comparison is that studies on pitch detection revolve mainly around speech [15] and polyphonic music [16]. With the RWC database, many studies are also undertaken with polyphonic music database [17]–[19] rather than the monophonic music database. Therefore, no comparison is performed for performances based on multiple intensity and SNR levels. The classification results from algorithm 7 (AGC enabled) is also not considered.

Table 6-3 displays the results of the classification of musical notes extracted from the temporal profile of autocorrelograms (AC) generated from subsection 6.3.1 at 0 dBFS intensity level with two other publications. Cerezuela-Escudero et al. [20] used a binaural 64-channel biologically-inspired model to generate spike trains that encode pitch information from a monophonic musical note. The spike trains are input to a two-layer neural network comprising a convolutional spiking layer and a winner-take-all (WTA) layer. The neural network is trained with four notes: F3, F4, F5, and F6. Fifty different musical notes played from an electric piano, are used to test the classifier to match the same four notes used for training the classifier, which resulted in a 97.5% classification accuracy.

Avci et al. [21] used time and frequency features extracted from monophonic musical signals, specifically from the viola and violin as inputs to the classification algorithms. Four classifiers are used for classifying the musical notes: (1) linear distinction analysis; (2) k-nearest neighbour (3) support vector machine; (4) random forests. Each classifier is trained with four notes (G3, D4, A4, and E5) from a viola and a violin. The classifiers are then tested

with 512 different notes from each of the two instruments to match with the four notes used for training the classifier. The classification results are then segregated based on the following extracted features: temporal, spectral and a combination of temporal and spectral features. The highest classification accuracy was reported for the combined temporal and spectral feature using the random forests classifier at 79.6%.

Since Cerezuela-Escudero et al. [20] used four notes across four octaves (3 to 6) as inputs, the AC pitch estimation results from octaves 3 to 6 for the piano are extracted from subsection 6.3.1. Here, an octave result refers to the classification of twelve notes per octave. Classification accuracies for viola and violin are also extracted from subsection 6.3.1 and included in the accuracy scores presented in Table 6-3, as Avci et al. [21] used musical notes from these two musical instruments as inputs. Hence, a total of 84 notes are used for testing the classifier described in this chapter. The AC responses are generated using AAC operations from the fixed-point (hardware) implementation of the CAR-Lite-ACF model. Algorithm 6 from Table 6-1 is used for estimating f_0 from the AC responses. The estimated f_0 from the ACs are used independently as inputs to the classification algorithm described in subsection 6.2.5, whose accuracy scores are averaged and presented in Table 6-3.

The highest classification accuracy occurs for [20] at 97.5% while the classification accuracy of the work presented in this chapter is the next highest at 89%. The key to the classification exercise in this chapter is the pitch estimation algorithm. These algorithms are designed based on three notes (A4, E4, and D#4) from the fourth octave of the piano. It estimates the fundamental frequency of a note and compares it with a ground truth frequency vector that is generated with equation (6-2). Hence, only a vector of 84 numbers representing the f_0 of 84 classes of notes is used as training data in the classifier instead of four classes of notes from [20] and [21], whereby the length of each note signal is significantly larger than 84 numbers. Moreover, with a larger set of trained data, the classifier in this chapter is capable of classifying ten times more classes of notes than in [20] and [21] combined as observed in Table 6-3.

Overall, although the f_0 estimation algorithms used in this chapter have limited musical octave and musical instrument estimation ranges, they are still capable of operating beyond the three notes range as well as the musical instrument range (piano) that their designs are based on – which [20] and [21] do not showcase.

Index	Author	Trained Notes Range	Number of Tested Notes	Instrument	Accuracy
1	Cerezuela-Escudero et al. [20]	F3, F4, F5, and F6	50	Electric Piano	97.5%
2	Avci et al. [21]	G3, D4, A4, and E5	512	Viola and Violin	79.6%
3	This work – AC	C3-B6 (P), C3-B3 (V1), C5-B5 (V1), C5-B5 (V2)	84	Piano (P), Viola (V1) and Violin (V2)	89%

Table 6-3: Comparison of the results of classifying musical notes between the work in this chapter and other works.

6.4. Summary and Conclusion

This chapter presents algorithms for estimating fundamental frequency, f_0 , from the temporal profile of an autocorrelogram (AC). The AC is generated from the CAR-Lite-ACF model, which is described in chapter 4. The f_0 estimation algorithms are designed using three piano notes from the fourth octave: A, E, and D#. The inputs to the f_0 estimation algorithms are monophonic musical signals (single note from a single instrument) from several musical instruments across six octaves (2 to 7) with twelve notes per octave. A pitch matching classifier is used to match the estimated f_0 with a ground truth vector of musical notes frequency for classification. Both the f_0 estimation and pitch matching classification algorithms are implementable on FPGA to indicate that hardware-based classification of musical notes is achievable.

The AC has a high accuracy for representing f_0 only at mid-range octaves (4 and 5). The AC results also vary across different musical instruments but are generally inadequate for low-range (octaves 2 and 3), suggesting that a more robust algorithm should account for musical pitch as well as general cross-octave features. The results are consistent regardless of the computation types used to generate the input signals for classification, i.e. either multiply-accumulate (MAC), AND-accumulate (AAC), floating-point (software) arithmetic or fixed-point (hardware) arithmetic.

Using musical signals from the fourth octave, the performances of the pitch estimation algorithms do not vary much across intensity levels. However, when an automatic gain control (AGC) algorithm is used for varying the gain of the input musical signals, there are more considerable variations across the performances of the algorithms than when the AGC is disabled. Furthermore, noise levels degrade the performances of these algorithms. These results indicate that the pitch estimation algorithms are not sufficiently robust in extracting pitch information when an AGC algorithm is used and when noise is added to the input signal.

In conclusion, the hardware-based fixed-point implementation of the CAR-Lite-ACF model is capable of generating responses for representing musical pitch and consume lower storage space (also presented in chapter 4) than its software-based floating-point counterpart. However, their robustness must be improved to accommodate complex real-world signals. One method to improve robustness is to use a running autocorrelation function [7] to ensure that peaks in the temporal summary profile do not decay for increasing lags, plausibly yielding a more reliable and simplified fundamental frequency estimation algorithm than the ones presented in this chapter.

6.5. Bibliography

- [1] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Music Genre Database and Musical Instrument Sound Database," in *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, 2003, pp. 229–230.
- [2] E. Bigand and B. Tillmann, "Effect of Context on the Perception of Pitch Structures," in *Pitch: Neural Coding and Perception*, C. J. Plack, A. J. Oxenham, R. R. Fay, and A. N. Popper, Eds. New York, NY, USA: Springer, New York, NY, 2005, pp. 306–351.
- [3] C. J. Plack and A. J. Oxenham, "The Psychophysics of Pitch," in *Pitch: Neural Coding*

- and Perception, C. J. Plack, A. J. Oxenham, R. R. Fay, and A. N. Popper, Eds. New York, NY, USA: Springer, New York, NY, 2005, pp. 7–55.
- [4] C. J. Plack and A. J. Oxenham, “Overview: The Present and Future of Pitch,” in *Pitch: Neural Coding and Perception*, C. J. Plack, A. J. Oxenham, R. R. Fay, and A. N. Popper, Eds. New York, NY, USA: Springer, New York, NY, 2005, pp. 1–6.
 - [5] J. Gao and D. Xu, “Noise-robust Pitch Detection Algorithm Based on AMDF with Clustering Analysis Picking Peaks,” in *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, 2016, pp. 1144–1148, doi: 10.1109/ITNEC.2016.7560544.
 - [6] R. Meddis and M. J. Hewitt, “Virtual pitch and phase sensitivity of a computer model of the auditory periphery. I: Pitch identification,” *J. Acoust. Soc. Am.*, vol. 89, no. 6, pp. 2866–2882, 1991, doi: 10.1121/1.400725.
 - [7] R. F. Lyon, “The Auditory Image,” in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 355–378.
 - [8] A. Klapuri and M. Davy, *Signal Processing Methods for Music Transcription*. New York: Springer Science+Business Media LLC, 2006.
 - [9] J. Marozeau, A. de Cheveigné, S. Mcadams, and S. Winsberg, “The dependency of timbre on fundamental frequency,” *J. Acoust. Soc. Am.*, vol. 114, no. 5, pp. 2946–2957, 2003, doi: 10.1121/1.1618239.
 - [10] Terasic and Altera Corporation, “Cyclone V GX Starter Kit - User Manual.” Terasic, pp. 1–103, 2014.
 - [11] Analog Devices, “SSM2603: Low Power Audio Codec (Rev. B).” Analog Devices, pp. 1–32, 2012.
 - [12] L. E. Marks, “On Cross-Modal Similarity: The Perceptual Structure of Pitch, Loudness, and Brightness,” *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 15, no. 3, pp. 586–602, 1989, doi: 10.1037/0096-1523.15.3.586.
 - [13] G. Kochanski, E. Grabe, J. Coleman, and B. Rosner, “Loudness predicts prominence: Fundamental frequency lends little,” *J. Acoust. Soc. Am.*, vol. 118, no. 2, pp. 1038–1054, 2005, doi: 10.1121/1.1923349.
 - [14] A. L. Wang and K. H. Street, “An Industrial-Strength Audio Search Algorithm,” in *2003 ISMIR International Symposium on Music Information Retrieval*, 2003.
 - [15] B. S. Lee, “Noise robust pitch tracking by subband autocorrelation classification,” Columbia University, 2012.
 - [16] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, “Automatic Music Transcription: Challenges and Future Directions,” *J. Intell. Inf. Syst.*, vol. 41, no. 3, pp. 407–434, Jul. 2013, doi: 10.1007/s10844-013-0258-3.
 - [17] V. Emiya, R. Badeau, and B. David, “Multipitch estimation of quasi-harmonic sounds in colored noise,” in *Proceedings of the 10th International Conference on Digital Audio Effects, DAFx 2007*, 2007, pp. 93–98.
 - [18] E. Benetos and S. Dixon, “A Shift-Invariant Latent Variable Model for Automatic Music Transcription,” *Comput. Music J.*, vol. 36, no. 4, pp. 81–94, 2012, doi: 10.1162/COMJ_a_00146.

- [19] L. Song, M. Li, and Y. Yan, "Melody extraction for vocal polyphonic music based on bayesian framework," in *Proceedings - 2014 10th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2014*, 2014, pp. 570–573, doi: 10.1109/IIH-MSP.2014.147.
- [20] E. Cerezuela-Escudero, A. Jimenez-Fernandez, R. Paz-Vicente, M. Dominguez-Morales, A. Linares-Barranco, and G. Jimenez-Moreno, "Musical notes classification with Neuromorphic Auditory System using FPGA and a Convolutional Spiking Network," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–7, doi: 10.1109/IJCNN.2015.7280619.
- [21] K. Avci, M. Arican, and K. Polat, "Machine Learning Based Classification of Violin and Viola Instrument Sounds for the Same Notes," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, 2018, pp. 4–7, doi: 10.1109/SIU.2018.8404422.
- [22] D. P. W. Ellis, "tf_agc - Time-frequency automatic gain control," 2010. https://labrosa.ee.columbia.edu/matlab/tf_agc/ (accessed May 10, 2020).
- [23] M. Slaney, "Auditory Toolbox Version 2," 1998. [Online]. Available: <https://engineering.purdue.edu/~malcolm/interval/1998-010/AuditoryToolboxTechReport.pdf>.

7. Classification of Musical Instruments

In this chapter, the auditory model described in chapter 5 (CAR-Lite-A1) is tested with real-world musical signals. The CAR-Lite cochlear model described in chapter 3 is used with monophonic input signals, which means that the input signal represents a musical note from a single musical instrument. These musical signals are sourced from the Real World Computing (RWC) musical instruments database [1].

The output of the CAR-Lite cochlear model is transmitted to a spectro-temporal envelope extraction algorithm. This combined model is inspired by a mammalian primary auditory cortex (A1) model and is used for the classification of musical instruments described in this chapter.

Two musical instruments classification algorithms are presented in this chapter. One of them is implementable on digital hardware such as a field-programmable gate array (FPGA). Again, the Occam's Razor principle is adhered to ensure that the design of the classification algorithms remain simple for FPGA implementation while maintaining low computational resource utilisation.

Besides the focus on an FPGA-implementable classification algorithm, the emphasis is also on the types of input signals used and their various combinations. Hence, the two proposed classification algorithms are used to test various possible input combinations to determine which combination generates the highest classification accuracy score.

This chapter is divided into six parts to investigate the classification accuracies of the hardware-implementable algorithms. Section 7.1 details the manner timbre can be extracted from a sound signal. Section 7.2 describes the selection of musical notes and instruments. Section 7.3 describes the various representations of the input signals from the CAR-Lite-A1 model, which are used as inputs to the classification algorithm. Section 7.4 details the classification algorithm and subsection 7.4.4 specifically details FPGA implementation. Section 7.5 presents the classification results. Finally, section 7.6 summarises and concludes this chapter.

7.1. Timbre Representation

Timbre describes the quality of a sound stimulus [2] regardless of its pitch [3] and loudness [4]. The quality of a sound is the shape of temporal and spectral features [5] that are unique to a sound source, which enables it to be an ideal cue for source recognition [6] and classification [7]. The temporal and spectral features of timbre are describable by several parameters. Essential temporal parameters include the time of the initial transient of a sound stimulus or attack time [8] and the temporal envelope of a sound stimulus [9]. Essential spectral parameters include the centre of gravity of a sound stimulus spectral energy, also known as spectral centroid, stimulus power spectrum around the spectral centroid, also known as spectral spread [10], and the variation of the stimulus spectrum over time or spectral flux [11]. According to Caclin et al. [12], the attack time and the spectral centroid parameters are the most widely reported timbre cues. This notion is reinforced by Kong et al. [13].

Another manner timbre is encoded is through spectro-temporal receptive fields (STRF) found in the mammalian primary auditory cortex, which capture spectro-temporal envelopes

from a sound stimulus [14]. Using mathematical STRF transfer functions to represent spectro-temporal envelopes, Patil et al. demonstrated a significant musical timbre classification success at approximately 98% accuracy conforming to psychoacoustical results [15]. Here, musical timbre is confined to monophonic musical signals from various 11 classes of musical instruments. Since the CAR-Lite-A1 model described in chapter 5 is an abstract of the model used by Patil et al., this chapter is dedicated to musical timbre classification using the responses from the CAR-Lite-A1 model. The objective is to understand how well the responses of the CAR-Lite-A1 model perform in the classification of monophonic signals from musical instruments. From this exercise, more difficult classification tasks can be designed, such as segmenting and classifying polyphonic musical signals.

7.2. Musical Notes Selection

A total of 12 classes of musical instruments are used from the RWC database [1] for the classification of musical instruments. Each instrument class is divided into two subclasses based on its respective manufacturer. Four notes with normal playing styles are selected from each instrument. Three of the four notes, A3, D4, and G#4, selected are based on [15] while the fourth note A4, is randomly selected. If any one of these notes is unavailable, then the next higher or lower note is selected if available. Similarly, if the notes in octaves 3 and 4 are unavailable, then the set of notes are selected from the next lower or higher octave if available. An example of this particular selection is for the harmonica (A4, G4, A5, and C5) and the flute (A4, D4, G#4, and A5).

Each of the four notes is further divided into three loudness levels: high, middle, and low. Hence, a total of 24 (3 loudness levels \times 4 musical notes \times 2 musical instruments per class) notes are used from each musical instrument class, which brings the combined total musical notes to 288 from 12 different musical instruments. Table 7-1 displays the musical instruments and their respective musical notes used for musical instruments classification. A further deliberation of musical notes selection based on pitch and loudness are presented in subsections 7.2.1 and 7.2.2, respectively.

7.2.1. Timbre Invariance

In psychoacoustic experiments, timbre invariance is the ability of a human listener to judge whether two different musical notes emanate from the same musical source, i.e. either a musical instrument or a singer [16]. Handel and Erickson found that non-musicians were able to successfully identify musical wind instruments if a series of musical notes emanating from the instruments are at most, one octave apart [16]. If the notes were more than an octave apart, the subjects were unable to identify the instruments correctly. Marozeau et al. extended the experiments to include musical stringed instruments and found similar findings to Handel and Erickson [10]. Hence, for non-musicians, the timbre invariance bandwidth is one octave. Steele and Williams also had the same findings regarding non-musician using musical wind instruments but found that the timbre invariance bandwidth was higher for musicians, up to 2.5 octaves [17].

To characterise timbre invariance described in the aforementioned psychoacoustic experiments, the response of a computational model must not be affected by changes in pitch in the range of 1 to 2.5 octaves. Calhoun and Schreiner reported that the response of ripple transfer function characterising the functional primary auditory cortex (A1) remains a

constant shape regardless of the change in the fundamental frequency, f_0 , or the changes in the number of frequency components [18]. In other words, the ripple transfer functions are capable of representing timbre invariance, similar to those in the NSL model. Indeed, Patil et al. demonstrated a high classification accuracy of musical instruments (11 classes) with responses from such ripple transfer functions in the NSL model [15]. He used three musical notes distributed within two octaves, per musical instrument, to perform the classification task. The usage of three notes distributed in two octaves is within the timbre invariance maximum bandwidth limit of 2.5 octaves mentioned in the preceding paragraph.

For the classification of musical instruments using the CAR-Lite-A1 model responses as input to a classifier described within this subsection, four notes are used per musical instrument distributed over two octaves, i.e., within the timbre invariance maximum bandwidth (of 2.5 octaves). The selected notes include two notes that are an octave apart (e.g. start note, A3, and A4 – note A from the third and fourth octaves) and two additional randomly selected notes within two octaves above the start note [e.g. D4 and G#4 – notes D and G# (known as G-sharp) from the fourth octave]. Table 7-1 details the notes used from 12 musical instrument classes for the classification exercise.

7.2.2. Dynamics and Articulation

The previous subsection (7.2.1) presented the relationship between timbre and pitch. In this subsection, the relationship between timbre and loudness is explored as well as the playing style of a musical instrument.

Melara and Marks found crosstalk between timbre and loudness, which led to bright sounds perceived by human listeners to be louder than dull sounds [19]. Here, bright sounds are sound that contains more high-frequency components than dull sounds. The change in the loudness levels in a series of musical notes is known as the dynamics of a musical piece [20]. According to Nakamura [21] as well as Fabiani and Friberg [22], the perception of dynamics does not correspond to a fixed sound level but is attributed more to the playing style of a specific musical instrument. In other words, it is largely dependent on a musician's style of playing a musical instrument based on a musical context that the musician wishes to convey to a listener. However, from the perspective of gauging the capability of software and hardware models in classifying musical instruments involving monophonic musical signals, the dynamics of each note is fixed at three discrete sound levels. They include forte (high) [23], mezzo (medium) [24], and piano (low) [25].

The playing style, also known as articulation, is the manner of separation of single notes or a group of notes usually indicated by the composer as markings in a musical score [26]. Musical notes with several articulation styles are used across all musical instruments from Table 7-1. However, they can be classified generally as normal [labelled in Table 7-1 for 12 musical instrument classes as normal (NO), hard mallet / normal (HN), soft mallet / normal (SN), apoyando / finger (AF), al aire / finger (RF), apoyando / pick (AP), al aire / pick (RP)] and staccato [labelled in Table 7-1 for 12 instrument classes as staccato (ST), tonguing (TO), and spiccato (SP)]. A normal articulation is when the resonance of a musical note is allowed to either gradually decay to silence or resonate to a duration equivalent to the point of transition from gradual decay to silence. For a wind instrument, this duration is subject to a musician's judgement to allow the instrument to resonate as long as possible. In contrast,

a staccato articulation is when a musical note ends before its normal articulation duration [26].















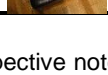
Index	Instrument	Articulation	Manufacturer	Subclass 1	Subclass 2	Notes Used
1	Piano	NO and ST	Yamaha			A3, D4, G#4, A4
2			Steinway & Sons			
3	Vibraphone	HN and SN	Musser			A3, D4, G#4, A4
4			Saito			
5	Marimba	HN and SN	Musser			A3, D4, G#4, A4
6			Yamaha			
7	Harmonica	NO and TO	Hohner - Blues Harp			A4, G4, A5, C5
8			Hohner - Chromatic			
9	Classical Guitar	AF and RF	Stafford			A3-S3, D4-S2, G#4-S1, A4-S1
10			Kohno Masaru			
11	Acoustic Guitar	AP and RP	Ovation			A3-S3, D4-S2, G#4-S1, A4-S1
12			Yamaha			
13	Viola	NO and SP	Andre Dugad			A3-S3, D4-S2, G#4-S2, A4-S1
14			Iida Yu			
15	Trombone	NO and ST	Conn			A3, D4, G#4, A4
16			S.E. Shires			
17	Soprano Sax	NO and ST	Yamaha			A3, D4, G#4, A4
18			Yanagisawa			
19	Bassoon	NO and ST	Heckel			A3, D4, G#4, A4
20			Puchner			
21	Clarinet	NO and ST	Buffet Crampon			A3, D4, G#4, A4
22			Selmer			
23	Flute	NO and ST	Sankyo			A4, D4, G#4, A5
24			Louis Lot			

Table 7-1: Twelve classes of musical instruments and their respective notes from the RWC database [1] used in the classification of musical instruments using the responses of the CAR-Lite-A1 model. Each instrument class is further divided into two subclasses based on two separate manufacturers, with each subclass containing the same notes played from the instrument, e.g. 1 set of notes (A3, D4, G#4, and A4) recorded from a Yamaha piano and another set of the same notes recorded from a Steinway piano for classification. Note that 'S' in the "Notes Used" column represents string number for musical notes generated by musical stringed instruments. Also note the following abbreviation – NO: Normal; ST: Staccato; HN: Hard mallet / Normal; SN: Soft mallet / Normal; TO: Tonguing; AF: Apoyando / Finger; RF: Al aire / Finger; AP: Apoyando / Pick; RP: Al aire / Pick; SP: Spiccato.

7.3. Feature Representation

This subsection presents how features of musical notes from various musical instruments are extracted for the classification of musical instruments. Subsection 7.3.1 addresses the usage of monophonic musical notes regardless of their respective varying lengths (time durations). This notion then results in the use of summary profiles, as presented in subsection 7.3.2.

7.3.1. Temporal Invariance

The musical notes extracted from the RWC database have varying time durations. A 4D CAR-Lite-A1 model output response of a musical note has the following dimensions based on the same settings as chapter 5: $s_f \times n \times s_s \times s_r$. Here, s_f is the number of cochlear sections, fixed at 108; s_s is the number of scale filters, fixed at 9; s_r is the number of rate filters, fixed at 13; n is the number of samples corresponding to the time duration of a

musical note. In image classification, the size of the images is fixed [27]. With this principle, it is ideal to keep dimension sizes constant for the classification task in this exercise. One method is to truncate the sample size, n , to a small fixed sample size. This act involves finding the ‘attack’ or onset segment of a musical note up to either its ‘sustain’ segment, where the amplitude of the input signal remains constant or its ‘decay’ segment, where the amplitude of the input signal decays, whichever exists and is encountered first. The truncation size is dependent on the selected musical note from the database with the smallest time duration. The disadvantage of this method is that a small sample size would result in the loss of information critical to timbre characteristics of the instrument.

Instead of truncating the sample size down, an alternative method is to use all the samples in the signal and generate a summarised representation, similar to summary autocorrelation profile used by Meddis [28], and summary rate-scale profile used by Chi et al. [29]. Although high temporal resolution information is replaced with low temporal resolution information in this method, the latter captures the envelope summary of an entire musical note (all the samples corresponding to the time duration of the note) instead of a truncated segment of the note. The reduced number of dimensions also ensures a reduction in computational resources such as memory as well as processing time. Another advantage is that a summarised representation provides intuitive visual cues that may not be captured with a large high-resolution 4D matrix.

7.3.2. Summary Profiles

The features used for the classification of musical instruments are represented by four types of input signals corresponding to recordings from four stages of the CAR-Lite-A1 model. They include a 1D summary spectral profile of the inner hair cell (IHC) stage denoted as $A1x$; a 2D summary scale filter output ($A1y$); a 2D summary upward (Up) and a 2D summary downward ($Down$) neuron directional response. Figure 7-1 illustrates the signals for an A3 musical note (normal style and forte dynamics) played on a piano. The IHC summary spectral profile consists of the summed columns of a 2D IHC output matrix. As there are 108 cochlear sections, an IHC summary spectral profile is a 1×108 vector. The remaining three signals are each a 2D matrix transformed from a 4D matrix. It involves collapsing the time and frequency dimension while retaining 13 temporal velocity and 9 spectral density components. In other words, fine spectro-temporal information is removed, while coarse spectro-temporal envelope information is retained.

As an illustration, a 4D $108 \times 375 \times 9 \times 13$ matrix is transformed into a 2D 9×13 matrix. Each element in the transformed 9×13 matrix is a summary of a 108×375 matrix corresponding to a specific rate (centre velocity of a rate filter) and scale (centre density of a scale filter) term. There is a total of 13 rate filters and 9 scale filters. Two methods are used to achieve this summarised transformation as detailed in chapter 5: finding the a) RMS and b) the sum of every 108×375 example matrix. The result of either the RMS or sum operation of one 108×375 example matrix leads to a single number. Combining this number from each of 13 rate filters and each of 9 scale filters, leads to the generation of a 9×13 matrix. For each instrument, six 9×13 matrices are generated in addition to the IHC spectral profile ($A1x$), i.e. three ($A1y$, Up , and $Down$) from the RMS operation and three ($A1y$, Up , and $Down$) from the sum operation.

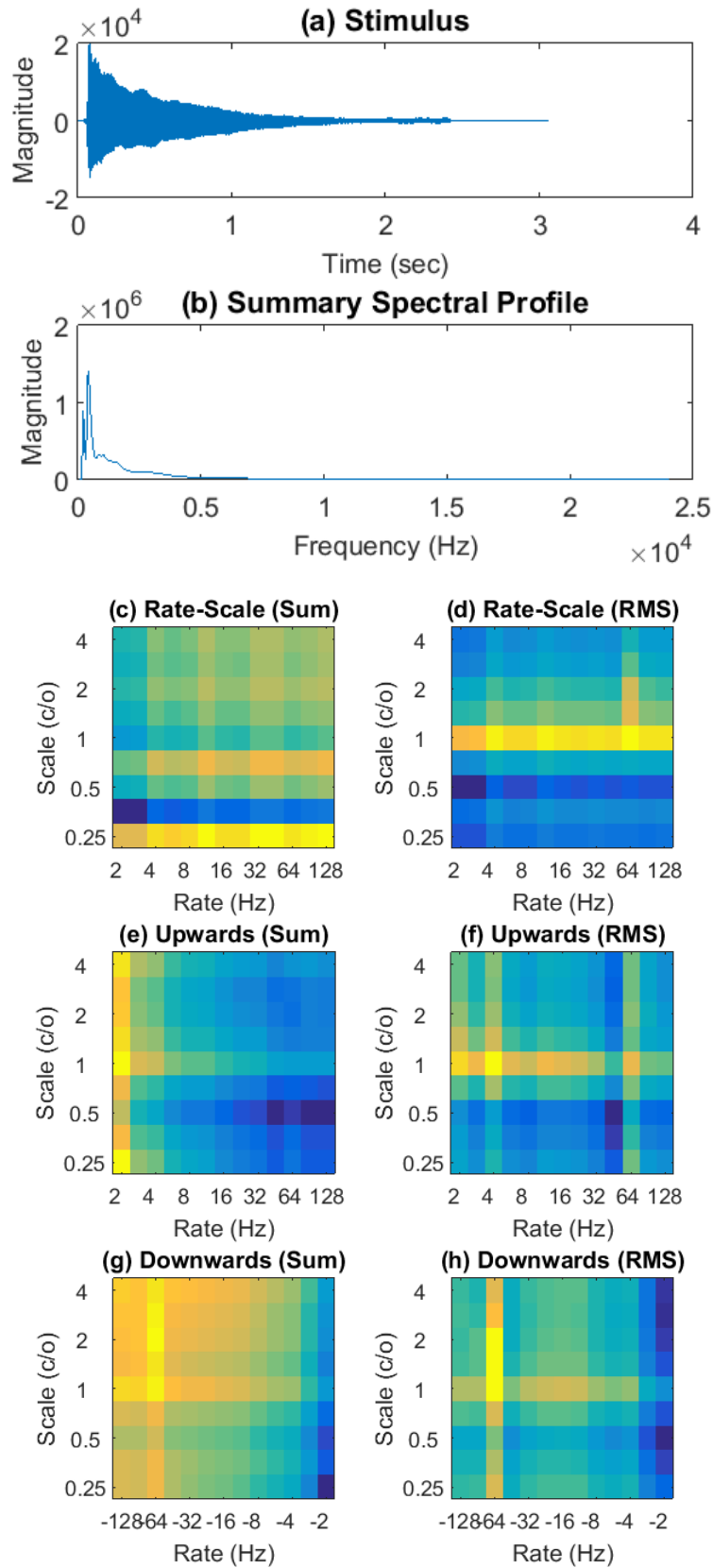


Figure 7-1: Fixed-point response of (a) an A3 note played from a piano recorded from the following stages of the CAR-Lite-A1 model: (b) Summary spectral profile of the 2D IHC matrix; summary rate-scale cascaded filter output calculated using (c) sum and (d) RMS operations; summary upward neuronal response calculated using (e) sum and (f) RMS operations; summary downward neuronal response calculated using (g) sum and (h) RMS operations.

7.4. Input Similarity and Classification Algorithms

The simplest method of finding the similarity between two images (2D matrices) is to calculate the 2D correlation between them. This equation results in the generation of a single floating-point value that defines the similarity between the two images. However, the 2D correlation equation comprises several nonlinear mathematical operations, which are obstacles in the implementation of the equation on an FPGA as is presented in subsection 7.4.1. Subsection 7.4.2 presents the timbre distance algorithm as an alternative to the 2D correlation. This algorithm does not contain any nonlinear mathematical operations and thus, is implementable on an FPGA.

Subsection 7.4.3 introduces two classification algorithms: a linear classifier using the 2D correlation coefficient algorithm from subsection 7.4.1 and a k-nearest neighbour (KNN) classifier using the timbre distance algorithm from subsection 7.4.2. Subsection 7.4.4 presents the FPGA implementation of the timbre distance algorithm presented in subsection 7.4.2 as well as the KNN classifier in subsection 7.4.3. Finally, subsection 7.4.5 describes the application of the two classification algorithms on the musical notes from various musical instruments. Here, each note is represented by the summary profiles described in subsection 7.3.2.

7.4.1. 2D Correlation Coefficient (CC)

The basic algorithm used in the linear classifier is a 2D correlation coefficient equation. This algorithm has been used by Dau et al. to find similarities between a predictive model response (described in chapter 2 under the section of functional primary auditory cortical model review) and the performance responses of subjects in psychoacoustic experiments [30]. The 2D correlation coefficient equation is defined as:

$$CC = \frac{\sum_m \sum_n (A_{m,n} - \bar{A})(B_{m,n} - \bar{B})}{\sqrt{\left(\sum_m \sum_n (A_{m,n} - \bar{A})^2\right) \left(\sum_m \sum_n (B_{m,n} - \bar{B})^2\right)}} \quad (7-1)$$

where CC is a correlation coefficient showing the degree of similarity between two $m \times n$ sized matrices, A and B ; \bar{A} and \bar{B} are the averages of A and B respectively. A CC score of 1 indicates identical matrices; 0 shows no resemblance; -1 shows two images that are inverted, e.g. if matrix A contains all positive elements, then $B = -A$ or if matrix B has same the elements as matrix A with the elements in the former swapped horizontally and vertically.

As an example of the similarity indication with the 2D correlation coefficient (CC), let us consider finding the CC, between the following four 2D matrices:

$$a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}; b = \begin{bmatrix} 11 & 12 \\ 13 & 14 \end{bmatrix}; c = \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}; d = \begin{bmatrix} 9 & 2 \\ 6 & 9 \end{bmatrix}$$

The CC between matrices a and b is 1.0, indicating the difference between the elements in the two matrices is the same despite all the elements in b being different from a . However, all the elements in b are shifted from a by 10, i.e., each component in a is nonlinearly scaled. In such a case, b represents a larger intensity signal than a but both are still considered identical to each other. In other words, a 2D summary representation of a high intensity

(forte) musical signal is highly similar to a 2D summary representation of a low intensity (piano) musical signal, and therefore, the CC between them is close to 1 (see chapter 5 for a description of how 2D summary profiles are generated). The CC between b and c is 0.4, indicating low similarity. The CC between c and d is -0.6, indicating non-similarity as well as image inversion.

Despite containing square root and division operations, the implementation of the 2D correlation coefficient defined by equation (7-1) on FPGA is possible. This practice involves the use of numerical methods such as the Newton-Raphson method [31] to implement the square-root and division operations. The numerical methods aim to minimise the errors between an input variable and a function output. The error for a square-root operation is calculated as:

$$e_{sqr} = x - (y_{sqr} \cdot y_{sqr}) \quad (7-2)$$

where y_{sqr} is the square of an input variable, x . The error for a division operation is calculated as:

$$e_{div} = x_{num} - (x_{den} \cdot y_{div}) \quad (7-3)$$

where y_{div} is the resultant division output value, and x_{num} is the numerator input value and x_{den} is the denominator input value. The errors in both equations are calculated iteratively. The outputs y_{sqr} and y_{div} are increased in increment of 1 per iteration until the errors, e_{sqr} and e_{div} fall below a threshold error set at 1. The final values of y_{sqr} and y_{div} are ideally the approximation of the square root and division operations, respectively. Since this implementation is targeted for an FPGA, the input values are expected to be in fixed-point integers.

Figure 7-2 depicts the degree of similarity calculated using various two input combinations of matrices a to d . CC is a vector of correlation coefficients calculated using equation (7-1). y_{div} is a vector of 16 bit variables representing the fixed-point approximations of the CC using equations (7-2) and (7-3). e_{sqr} and e_{div} are two vectors containing 16 bit numbers calculated from equations (7-2) and (7-3) and normalised by their respective maximum vector values.

The mean of each input matrix (\bar{A} , and \bar{B}) is calculated using the average of the all the values from each matrix. This algorithm is implementable on FPGA with a right-shift operation on the sum of all the values. As the CC values range between -1 to 1, the output of y_{div} does not vary as the fixed-point numbers are rounded to the nearest whole number. In this case, the rounded number is 1. Hence, the fixed-point CC does not correspond to the conventionally calculated CC. An alternative method of calculating the CC is by using the two error values, e_{sqr} and e_{div} . This characteristic is observable in Figure 7-2, where their responses follow the responses of the CC, even though they are offset vertically.

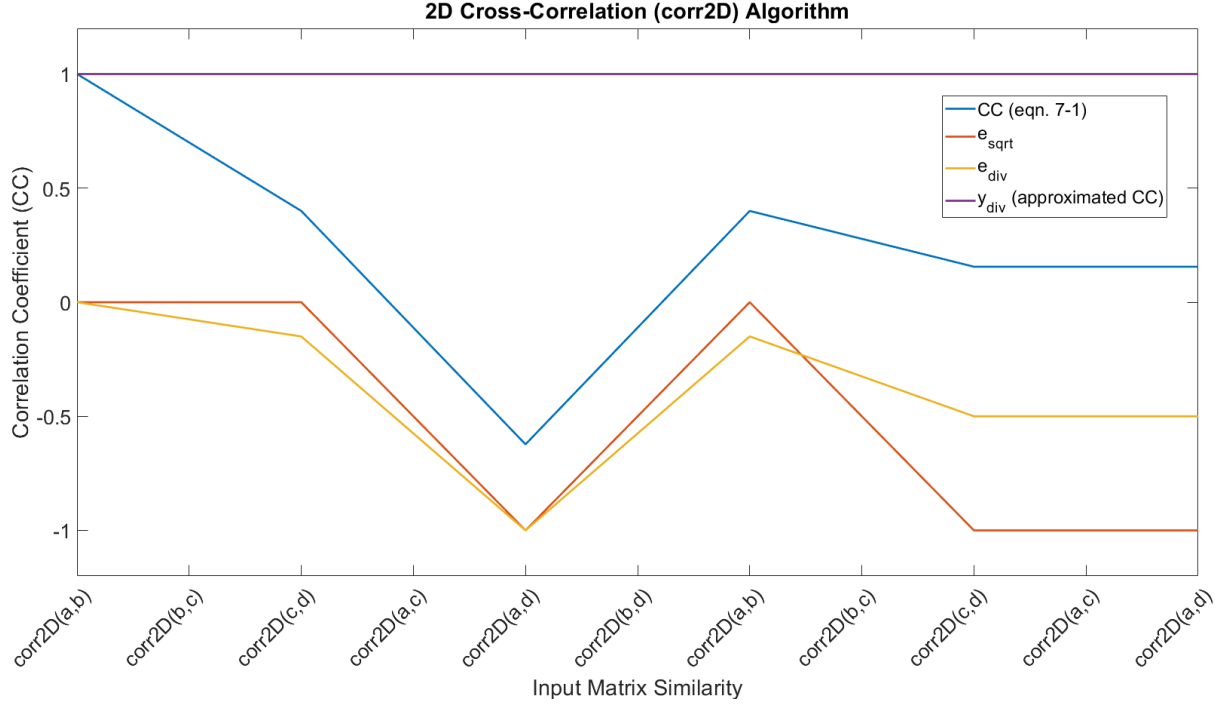


Figure 7-2: A comparison of correlation coefficients (CCs) calculated from equation (7-1), y_{div} , e_{sqrt} , and e_{div} that are based on various two input combinations between matrices a to d . Note that the lines for e_{sqrt} and e_{div} are normalised to their respective maximum values.

7.4.2. Timbre Distance (TD)

The 2D correlation coefficient defined by equation (7-1) uses several mathematical operations such as average, square-root, and division, which require numerical methods for evaluation. Although the numerical methods in subsection 7.4.1 are implementable on FPGA, the latencies of the numerical methods vary and are mostly dependent on the bit width of the sample values in the input matrices. Hence, numbers with large bit widths require a longer time to process than numbers with short bit widths. Furthermore, there is a likelihood that the approximated correlation coefficient (CC) is affected as observed by y_{div} in Figure 7-2, whose approximated CC values are all 1 for various 2D input correlations – different from the actual CC depicted in the same graph. This effect, in turn, results in inaccurate hardware-based classification scores.

An alternative to the 2D-correlation algorithm in subsection 7.4.1 is to calculate the timbre distance. This method is inspired by the calculation of the Euclidean distance used by Meddis and O'Mard to find the distance between peaks in a temporal profile of an autocorrelogram to acquire pitch information, i.e. fundamental frequency [32]. The first of three steps involves calculating the differences between the samples in the two 2D input matrices [from equation (7-1)]:

$$\Delta_{m,n} = |A_{m,n} - B_{m,n}| \quad (7-4)$$

Summing all the error samples in $\Delta_{m,n}$ yields:

$$s_e = \sum_m \sum_n \Delta_{m,n} \quad (7-5)$$

where s_e is the sum of error samples which defines the similarity between the two input images, i.e. a small s_e shows a larger degree of similarity, whereas a large s_e shows a smaller degree of similarity.

As an example, calculating s_e for matrix a and b (defined in subsection 7.4.1) and for matrix b and c matrices yield two values: 40 and 40. These numbers indicate that matrices a and c are not similar to matrix b but are most likely identical to each other. However, this conclusion is inaccurate if the contents of the two matrices are observed from subsection 7.4.1. This characteristic is reinforced in the same subsection, where the CC between a and b is 1.00, and for matrices b and c , it is 0.40. In other words, despite matrix b having samples with larger magnitudes than those from matrix a , the distance (error) between the neighbouring magnitudes of samples in a and b are the same. Hence, an alternative approach is required, as presented next.

Finding the sums of horizontal differences of neighbouring samples in a , b , and c yield s_{ha} , s_{hb} and s_{hc} , respectively:

$$\begin{aligned} s_{ha} &= \sum \left[\begin{vmatrix} 1 & -2 \\ 3 & -4 \end{vmatrix} \right] = \sum \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 2 \\ s_{hb} &= \sum \left[\begin{vmatrix} 11 & -12 \\ 13 & -14 \end{vmatrix} \right] = \sum \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 2 \\ s_{hc} &= \sum \left[\begin{vmatrix} 1 & -3 \\ 4 & -2 \end{vmatrix} \right] = \sum \begin{bmatrix} 2 \\ 2 \end{bmatrix} = 4 \end{aligned}$$

Since s_{ha} and s_{hb} are equal at 2, these results coincide with the CC between a and b being 1.00, indicating similarity. Conversely, s_{ha} and s_{hc} are different at 2 and 4, respectively, which correspond to the low degree of similarity of 0.40 between matrix a and c .

So, the sum of magnitude errors between neighbouring horizontal samples is applicable directly to $\Delta_{m,n}$ calculated from equation (7-4), which results in the timbre distance concentrated in the horizontal direction, TD_h , between two 2D sound images. Hence, the second of the three steps is the calculation of the directional timbre distance variable:

$$TD_h = \sum_m \sum_n |\Delta_{m,n} - \Delta_{m,n+1}| \quad (7-6)$$

Applying this formula across vertically neighbouring samples yields:

$$TD_v = \sum_m \sum_n |\Delta_{m,n} - \Delta_{m+1,n}| \quad (7-7)$$

The final step involves summing both TD_h and TD_v yielding the overall timbre distance, TD between the two input matrices representing the two sound signals:

$$TD = TD_h + TD_v \quad (7-8)$$

As an example, let us find the timbre distance between matrices a and b (a vs b), a and c (a vs c) as well as a and d (a vs d). The first step is to use equation (7-4) to find the difference between each of the three pairs of matrices mentioned before:

$$\Delta_{m,n}(a \text{ vs } b) = \left| \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 11 & 12 \\ 13 & 14 \end{bmatrix} \right| = \begin{bmatrix} 10 & 10 \\ 10 & 10 \end{bmatrix}$$

$$\Delta_{m,n}(a \text{ vs } c) = \left| \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix} \right| = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\Delta_{m,n}(a \text{ vs } d) = \left| \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 9 & 2 \\ 6 & 9 \end{bmatrix} \right| = \begin{bmatrix} 8 & 0 \\ 3 & 5 \end{bmatrix}$$

The second step involves applying equations (7-6) and (7-7) to find the horizontal (TD_h) and vertical (TD_v) timbre distances between the magnitudes of samples from the three matrices from above:

$$TD_h(a \text{ vs } b) = \sum_m \sum_n \left[\begin{bmatrix} |10 - 10| \\ |10 - 10| \end{bmatrix} \right] = \sum_n \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

$$TD_v(a \text{ vs } b) = \sum_m \sum_n [|10 - 10| \quad |10 - 10|] = \sum_m [0 \quad 0] = 0$$

$$TD_h(a \text{ vs } c) = \sum_m \sum_n \left[\begin{bmatrix} |0 - 1| \\ |1 - 2| \end{bmatrix} \right] = \sum_n \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 2$$

$$TD_v(a \text{ vs } c) = \sum_m \sum_n [|0 - 1| \quad |1 - 2|] = \sum_m [1 \quad 1] = 2$$

$$TD_h(a \text{ vs } d) = \sum_m \sum_n \left[\begin{bmatrix} |8 - 0| \\ |3 - 5| \end{bmatrix} \right] = \sum_n \begin{bmatrix} 8 \\ 2 \end{bmatrix} = 10$$

$$TD_v(a \text{ vs } d) = \sum_m \sum_n [|8 - 3| \quad |0 - 5|] = \sum_m [5 \quad 5] = 10$$

Finally, the combined timbre distances (TD) between TD_h and TD_v are calculated:

$$TD(a \text{ vs } b) = TD_h(a \text{ vs } b) + TD_v(a \text{ vs } b) = 0 + 0 = \mathbf{0}$$

$$TD(a \text{ vs } c) = TD_h(a \text{ vs } c) + TD_v(a \text{ vs } c) = 2 + 2 = \mathbf{4}$$

$$TD(a \text{ vs } d) = TD_h(a \text{ vs } d) + TD_v(a \text{ vs } d) = 10 + 10 = \mathbf{20}$$

So, if two input matrices are similar as is the case between a and b , the timbre distance between them is close to 0 (bold number highlighted in yellow). A 2D correlation between them results in a CC of 1. If the two input matrices are not similar, the timbre distance between them is non-zero (bold numbers highlighted in blue and green), while the 2D correlation between the input matrix pairs result in CCs close to 0. The TD is larger between a and d (TD = 20) than a and c (TD = 4). The CCs between a and d is at 0.16, which is lesser than the CC between a and c at 0.40. Thus, the larger the timbre distance, the lower the CC, which indicates a higher degree of dissimilarity between the two input matrices.

7.4.3. Classification

This subsection presents two classification algorithms used for the classification of musical instruments. The first involves a linear classifier using the 2D correlation coefficient described in subsection 7.4.1. The second involves a k-nearest neighbour (KNN) classifier using timbre distance described in subsection 7.4.2.

The main difference between the linear and the KNN classifiers is the method a prediction is made for a test signal. For an input musical (test) signal corresponding to an unknown musical instrument, this characteristic refers to the prediction of a musical instrument from a known (trained) set of signals of musical instruments most similar to the input signal. This similarity corresponds to the maximum of a vector of correlation coefficients (CC) for the 2D correlation coefficient algorithm as part of a linear classifier:

$$CC_{max} = \max_{i \in N} CC(i) \quad (7-9)$$

where i is the index of a single CC out of a total of N CCs. For example, the comparison between matrix a and matrices b [$CC(1) = 1.00$], c [$CC(2) = 0.40$], and d [$CC(3) = 0.16$] in subsection 7.4.1 results in a maximum CC of 1.00. This result indicates that matrix b is the most similar to matrix a .

Conversely, the similarity in a KNN classifier is the minimum value from a timbre distance (TD) vector:

$$TD_{min} = \min_{i \in N} TD(i) \quad (7-10)$$

As an example, comparing the TD between matrix a and matrices b [$TD(1) = 0$], c [$TD(2) = 4$], and d [$TD(3) = 20$] results in a minimum TD of 0. This result also indicates that matrix b is the most similar to matrix a .

7.4.4. FPGA Implementation

This subsection describes the FPGA implementation of the timbre distance algorithm as well as the k-nearest neighbour (KNN) classification algorithm described in subsections 7.4.2 and 7.4.3, respectively. The Altera Cyclone V starter kit with a 5CGXFC5C6F27C7 FPGA chip operating at a system clock rate of 100 MHz is utilised for implementing a finite state machine (FSM) with nine states in a single top module using SystemVerilog. The minimum latency for processing all nine states serially is 90 ns. However, as several states within the FSM are iteratively processed, the overall latency is dependent on the size of the input matrices.

State 0 initialises all constants. State 1 processes equation (7-4) to calculate y_d corresponding to the difference between two input matrices, $\Delta_{m,n}$. States 2, 3, and 4 process equations (7-6) and (7-7) to calculate y_TD_h and y_TD_v corresponding to the directional timbre distances, TD_h and TD_v , respectively. The input numbers to the summation algorithm to generate y_TD_h and y_TD_v are stored in y_Horz_Dir and y_Vert_Dir , respectively. State 5 processes equation (7-8) to calculate y_TD corresponding to the combined timbre distance, TD . State 6 updates the state indices and decides the next state to process. If there are more one input matrix in the queue, the FSM returns to state 1 to process them.

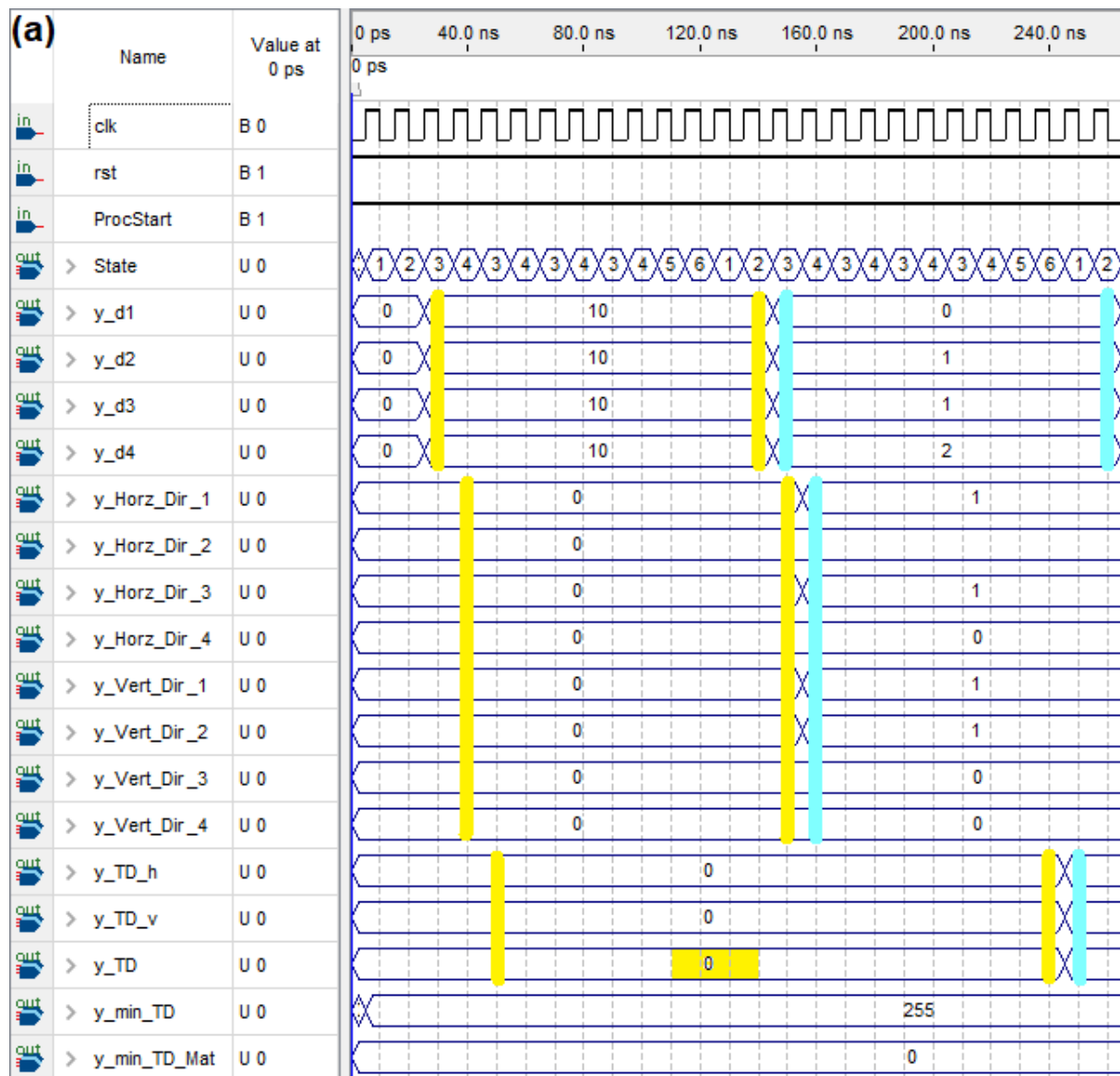
Otherwise, it goes to state 7, where the KNN classification algorithm is processed. Here, the minimum of all combined timbre distances (TD), y_min_TD , is determined in accordance with equation (7-10). The output of the classifier is made available in state 8.

Figure 7-3 illustrates the FPGA implementation of the application of the KNN classifier to the example presented in the latter half of 7.4.2, i.e. finding the minimum timbre distance between the following input matrices: a vs b , a vs c , and a vs d . The numbers bounded between two yellow vertical lines or highlighted in yellow correspond to the results of the algorithms applied to input matrices a and b . The ones in blue and green show the results of a vs c and a vs d , respectively. y_d1 to y_d4 projects $\Delta_{m,n}$ corresponding to the difference between the samples in the two input matrices, where y_d1 and y_d2 are the numbers in the top row of $\Delta_{m,n}$ and y_d3 and y_d4 are numbers in the bottom row. From y_d1 to y_d4 , $y_Horz_Dir_1$ to $y_Horz_Dir_4$ as well as y_Vert_Dir1 to y_Vert_Dir4 are calculated. Subsequently, the directional and combined timbre distances, y_TD_h , y_TD_v , and y_TD are calculated. y_TD is used as input to the classifier using the three combined timbre distance values, 0, 4, and 20 related to a vs b , a vs c , and a vs d , respectively. Here, a is the test data, and b , c and d are the trained data. y_min_TD illustrates the minimum timbre distance of 0, which corresponds to matrix b as depicted by $y_min_TD_Mat$ as 1. In other words, since matrix b has the lowest timbre distance to matrix a , it is the most similar to a as opposed to c and d . Note that the sample indices of matrices, a , b , c , and d are represented on FPGA as 0, 1, 2, and 3.

Table 7-2 presents the computational resources required by an Altera Cyclone V FPGA to process the timbre distance and KNN classification algorithms. The number of ALM and registers utilised for the algorithm here is lower than the resources used for the AC f_0 estimation algorithms in chapter 6, due to two reasons. Firstly, the number of samples used in the input signals here is smaller than the number of samples used in the input signals of AC f_0 estimation algorithms. Since the algorithms described in this subsection is scalable, the computational resources increase if the number of input samples grows. Secondly, the AC f_0 estimation algorithms are each a culmination of three (subsections 6.2.1, 6.2.2, and 6.2.3) algorithms, respectively, in comparison to a single algorithm (timbre distance) here. As a result, the power consumed by an FPGA is smaller for the timbre distance and KNN classification algorithms described here than the AC f_0 estimation algorithms. Finally, no digital signal processors (DSPs) are used as expected, as the timbre distance and KNN classification algorithms do not contain any multiplication operation.

FPGA	Number of ALM Utilised (out of 29,080)	Number of Registers Utilised	Number of DSPs Utilised (out of 150)	Power (mW)
Altera Cyclone V	332 (1.1%)	655	0 (0%)	239

Table 7-2: Computational resources used by an Altera Cyclone V to implement the timbre distance and KNN classification algorithms.



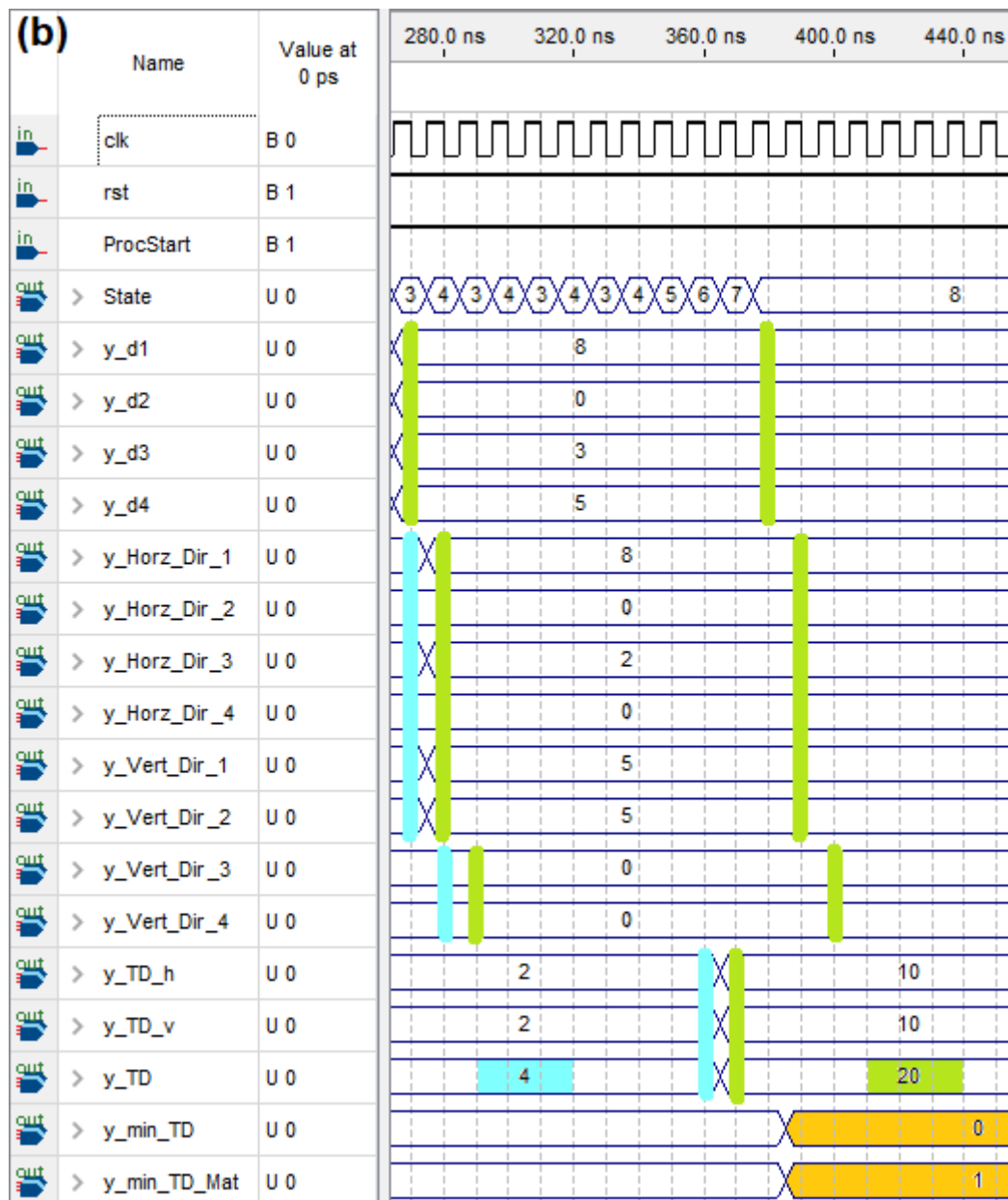


Figure 7-3: Output vector waveform of an FPGA implementation of the timbre distance and the KNN classifier algorithms presented in subsections 7.4.2 and 7.4.3, respectively. Part (a) displays the results of **a vs b**, and the partial results of **a vs c**, while part (b) is a continuation of (a) showing the remaining results of **a vs c** as well as **a vs d**. Classification results are in orange highlight.

7.4.5. Application to Musical Signals

This subsection describes how the musical signals are used in the classification of the musical instruments. As there are equal number of trained and test signal sets, each test signal under a musical instrument class comprising the four notes and three loudness levels are compared to the trained signals for all musical instrument classes. The trained signal with the highest match score is the instrument class that matches the test signal instrument class. This match is considered successful if the labels associated with the trained and test classes match. Otherwise, it is a considered classification failure. An example of this classification is presented the following paragraphs.

Figure 7-4 displays the layout of the musical classification algorithm with four musical instrument classes as an example. Its output is a 4×4 matrix of P variables. Here, P represents either CC from a linear classifier or TD from a KNN classifier. In the classification task of musical instruments presented in section 7.5, twelve instrument classes are used to generate a 12×12 output matrix. The inputs, x_{iak} and x_{jbk} , correspond to either one of the four input signals [$A1x$ ($k = 1$), $A1y$ ($k = 2$), Up ($k = 3$), or $Down$ ($k = 4$)] addressed by index, k . To calculate the 4×4 output matrix, the four musical instrument classes are divided into two groups based on its manufacturer, whereby a is the trained group and b is the test group. The input signal of instruments in groups a (input signal, x_{ia}) and b (input signal, x_{jb}) are individually addressed by indices i and j respectively – for illustration in Figure 7-4, i and j range from 1 to 4. Each musical instrument class has four notes at three dynamic levels, and so there are a total of twelve notes per instrument, which means that a $4 \times 4 \times 12$ output matrix is generated and averaging the P values across the twelve notes result in a 4×4 matrix, $P_{iak,jbk}$, as seen in Figure 7-4. At this stage, $P_{iak,jbk}$ corresponding to one of the four signal types can be combined with the other three signal types (or more) by further averaging, to form $P_{ia,jb}$.

In the case of a linear classifier, all P elements from Figure 7-4 are replaced with CC values. The index of the maximum CC , P_m , is extracted from every column of $CC_{ia,jb}$ based on equation (7-9):

$$P_m = \max_{jb \in I} \left(\sum_{k=1}^K \frac{1}{D} \sum_{d=1}^D \frac{1}{N} \sum_{n=1}^N CC_{iak,jbk,n,d} \right) \quad (7-11)$$

where I is the total number of musical instrument classes with i addressing instrument labels from group a and j addressing instrument labels from group b ; K is the total number of input signals used with each input addressed by k ; D is the total number of dynamic levels with d addressing individual dynamic level for groups a and b ; N is the total number of musical notes per instrument with n addressing individual note per instrument for groups a and b .

In the case of a KNN classifier, all P elements from Figure 7-4 are replaced with TD values. The index of the minimum TD , P_m , is extracted from every column of $TD_{ia,jb}$ based on equation (7-10), which is described as the minimum distance between a trained signal from group a and a test signal from group b :

$$P_m = \min_{jb \in I} \left(\sum_{k=1}^K \frac{1}{D} \sum_{d=1}^D \frac{1}{N} \sum_{n=1}^N TD_{iak,jbk,n,d} \right) \quad (7-12)$$

The row index, l_{ia} corresponding to the ground truth label of a trained signal class, is compared with the column index label, l_{jb} , corresponding to the test signal label with the highest matching score, P_m . Since the trained and test set classes are aligned (piano-piano, guitar-guitar, etc.), their respective labels ideally match. However, the labels of the test signal classes are defined exclusively by P_m , which may not match with the same label as

the trained signal after classification. The correct classification is defined by in a binary vector, u_b :

$$u_b = \begin{cases} 1, & l_{ia} = l_{jb} \\ 0, & l_{ia} \neq l_{jb} \end{cases} \quad (7-13)$$

The classification accuracy, y , is then calculated by summing the binary vector:

$$y = \frac{1}{I} \sum_{i=1}^I u_b(i) \quad (7-14)$$

where I is the number of classes of musical instrument.

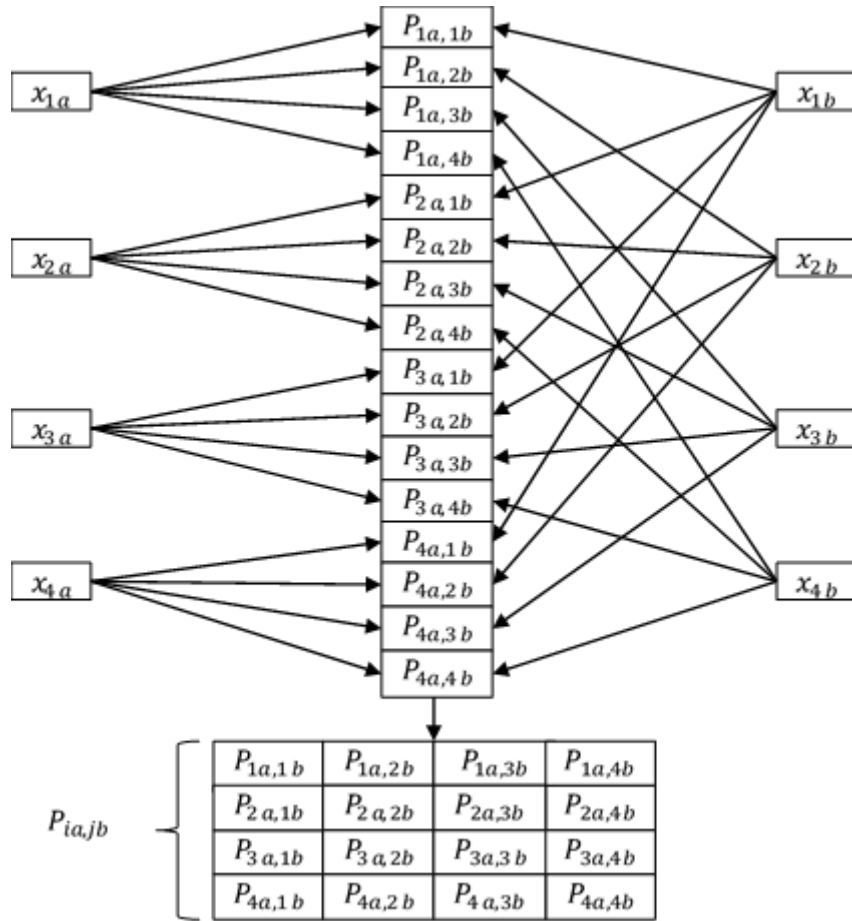


Figure 7-4: Algorithm for classifying musical instruments adapted from Dau et al. [30]. Here, x_{iak} is the training set signals and x_{ibk} are the test set input signals for four musical instrument classes. P represents either CC from the linear classifier or TD from the KNN classifier.

7.5. Results and Evaluation

This subsection presents the results of the classification of musical instruments in three parts. Subsection 7.5.1 presents classification results based on the combinations of various responses recorded from the CAR-Lite-A1 model. Subsection 7.5.2 presents classification accuracies based on varying input signal intensity and signal-to-noise (SNR) levels.

Subsection 7.5.3 presents the difference in file sizes of the four types of input signals calculated in floating-point (software) and fixed-point (hardware) arithmetic used in the classification. Subsection 7.5.4 compares the highest classification result of the responses from the CAR-Lite-A1 model presented in subsection 7.5.1 with five other models.

7.5.1. Accuracy Comparison of 0 dBFS Input Signals

The input signals recorded from the CAR-Lite-A1 model and described in Table 7-1 for classification are available in two formats: floating-point and fixed-point. The former is known as the software implementation response based on the original design of the CAR-Lite-A1 model, whereas the latter is a hardware (FPGA) implementation response of the same model. Chapter 5 describes the configurations of the model corresponding to the two responses. Each response is further divided into two types of circuit as part of an analytic signal representation described in chapter 5: real, Re , and imaginary, Im . Two additional circuits are introduced for classification based on the additive and multiplicative combinations of Re and Im circuits: $Re + Im$, and $Re \times Im$. The use of these different combinations provides a broader and more diverse range of input signal combinations than the two responses Re and Im . These signals are classified using the linear and KNN classifiers described in section 7.4 and with the input signals at 0 dBFS intensity level (no amplitude clipping).

Figure 7-5 displays a pair of confusion matrices following the classification of the hardware-based fixed-point signals of $A1x$, $A1y$, and Up from a Re circuit, whose results are averaged (described in subsection 7.4). The FPGA implementable KNN classifier is used on these signals. Note that the summary profiles for $A1y$ and Up are calculated using sum operation (the alternative is RMS as described in chapter 5). Figure 7-5(a) displays the raw timbre distances (TD) and Figure 7-5(b) displays ternary-state representation of the Figure 7-5(a), where only the minimum TD from every row of Figure 7-5(a) is displayed. The minimum TD falling on the diagonal of the confusion matrix depicts accurate classification, and if it falls outside the diagonal, it depicts inaccurate classification. The classification accuracy of Figure 7-5 is 91.67% - the highest accuracy encountered among 240 classification accuracies for various input signal combinations (TD averaged) that includes 120 accuracies of software floating-point responses in Table D-3 and 120 accuracies of hardware fixed-point responses in Table D-4.

The highest KNN classification accuracy score corresponding to the hardware-based fixed-point RMS-calculated input summary profiles is 75%. This score is from the input signal combinations of Up , and $Down$ responses using the $Re + Im$ circuit. For the sum-calculated summary profiles, the same score of 75% is found for four input signal combinations – one for floating-point and one for fixed-point responses from Im and $Re + Im$ circuits respectively. In contrast, the highest KNN classification accuracy score corresponding to the floating-point RMS-calculated input summary profiles is lower than fixed-point signals mentioned above at 67% for 42 input signal combinations. The highest classification accuracy score for the sum-calculated floating-point summary profiles is 91.67% for Re circuit with $A1y$, Up , and $Down$ combined input signals. The same score is found for 22 input combinations under sum-calculated fixed-point summary profiles – 7 Re , 7 Im , 7 $Re + Im$, 1 $Re \times Im$. Refer to Table D-3 and Table D-4 for the specific combinations of input signals for the attainment of the classification accuracy scores mentioned in this paragraph.

Table 7-4 presents a summary of the KNN classification results of musical instruments. Out of 120 input signal configurations, the fixed-point responses outscore floating-point responses by 14 (see row 2 in Table 7-4) – 35 fixed-point scores are higher than floating-point scores, while only 21 floating-point scores are higher than fixed-point scores. Of the 35 fixed-point responses, 29 summary profiles are calculated using sum operations, and 6 are calculated using RMS operations. Out of the 21 lower fixed-point scores, 5 summary profiles are calculated using RMS operations, and 16 are calculated using sum operations. The remaining 64 input signal configurations have identical scores between fixed-point and floating-point responses.

The highest linear classification accuracy score is 100% from two hardware-based fixed-point input sum-calculated summary profiles – *A1x* and *Down* combination using a $Re + Imag$ circuit as well as *A1x*, *A1y*, *Up*, and *Down* combination using a $Re \times Imag$ circuit. The highest linear classification score is lower for software-based floating-point input sum-calculated summary profiles at 83.33% – *A1x* and *Up* using a *Im* circuit. The highest linear classification score is the same at 83.33% for ten floating-point RMS-calculated summary profiles but is found for 18 fixed-point input sum-calculated summary profiles. The highest linear classification accuracy is 91.67% for a single fixed-point RMS-calculated input summary profile – *A1x* and *Down* combination using an *Im* circuit. The same score is found for ten fixed-point sum-calculated input summary profiles.

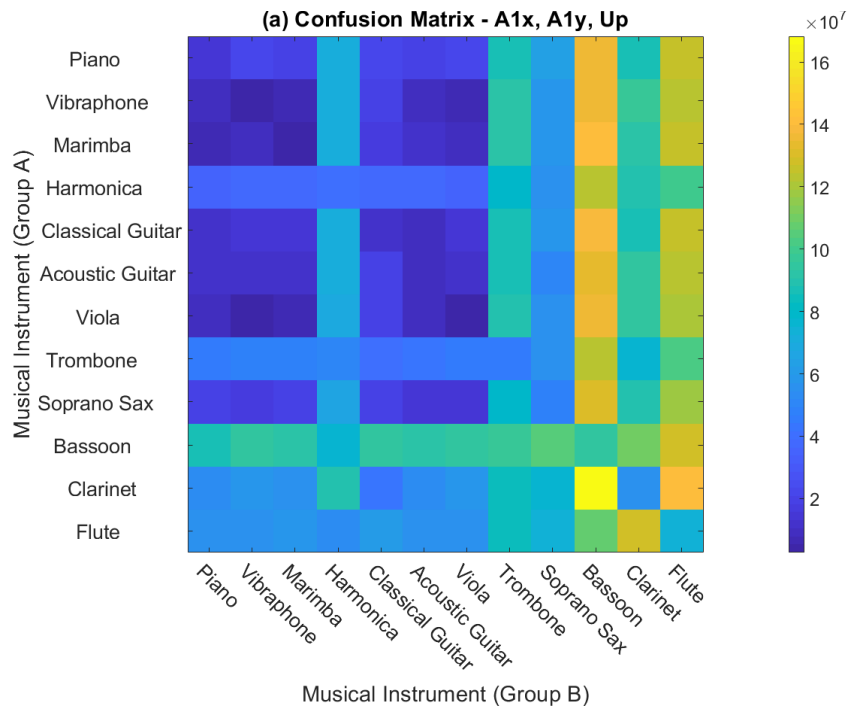
Table 7-3 presents a summary of the linear classification results. Out of 120 input signal configurations, 59 fixed-point scores are higher than floating-point scores. Of the 59 fixed-point responses, 47 summary profiles are calculated using sum operations, and 12 are calculated using RMS operations. Of the remaining 61 input signal configurations, only 19 fixed-point scores are lower than floating-point scores, and 42 have identical scores. Out of the 19 lower fixed-point scores, 17 summary profiles are calculated using RMS operations, and 2 are calculated using sum operations.

The software-based linear classifier outperforms the hardware-based KNN classifier, as observed in Table 7-3 and Table 7-4. The former has a higher maximum, mean, and median classification accuracy scores than the latter. The linear classifier has approximately 1.6- ($\approx 47 / 29$) and 2-times ($\approx 12 / 6$) more fixed-point input combinations with higher classification accuracies than floating-point input combinations in comparison with the KNN classifier. However, the KNN classifier has approximately 1.5 times more input combinations with fixed-point and floating-point responses having equal classification accuracies than the linear classifier. These results indicate that the 2D correlation algorithm in the linear classifier is more capable of quantifying small non-similar effects between two 2D matrices than the timbre distance algorithm in the KNN classifier. Conversely, the KNN classifier is more capable of quantitatively projecting salient (prominent magnitude) differences between two 2D matrices that are more ambiguous than the linear classifier, resulting in lower accuracy scores.

The fixed-point response produces a more accurate classification of the musical instruments than the floating-point response with input combinations of the former having more than 3- and 1.5-times higher accuracy scores than the latter for linear classification (59-to-19 accuracy score ratio) and KNN classification (35-to-21 accuracy score ratio), respectively. One possible explanation is that large magnitudes represented in a fixed-point response create more substantial magnitude differences between neighbouring elements in

a summary profile than in a floating-point response. This notion means that the fixed-point system overlooks small-amplitude variations that are captured by the floating-point system due to quantisation error. In other words, a higher precision floating-point representation may capture minor differences across pixels that affect correlation coefficient (CC), and timbre distance (TD) scores more significantly than a lower precision fixed-point representation that overlooks such insignificant differences, but still manages to successfully classify musical instruments based on salient timbre cues in the input signal. This notion suggests that the fixed-point responses accentuates salient timbre features while ignoring non-salient timbre features more capably than the floating-point responses.

The results also indicate that sum-calculated summary profiles are capable of attaining up to 2- and 1.6-times higher classification accuracy scores than RMS-calculated summary profiles in fixed-point arithmetic using linear classification (30-to-15 accuracy score ratio) and KNN classification (31-to-19 accuracy score ratio), respectively. In other words, these results suggest that a hardware implementation of the CAR-Lite-A1 model represents timbre as capably as a software implementation of the same model. Conversely, the RMS-calculated summary profiles have a lower average classification accuracy score and standard deviations than sum-calculated summary profiles for both floating-point and fixed-point arithmetic. These results indicate that RMS-calculated summary profiles represent timbre more consistently for various input combinations of monophonic signals. In contrast, sum-calculated summary profiles represent timbre well for only a selected number of input combinations of monophonic signals. Hence, the performance of the hardware model is higher than the software model for only a small number of input signals, whereas the software model has more consistent performance levels than the hardware model.



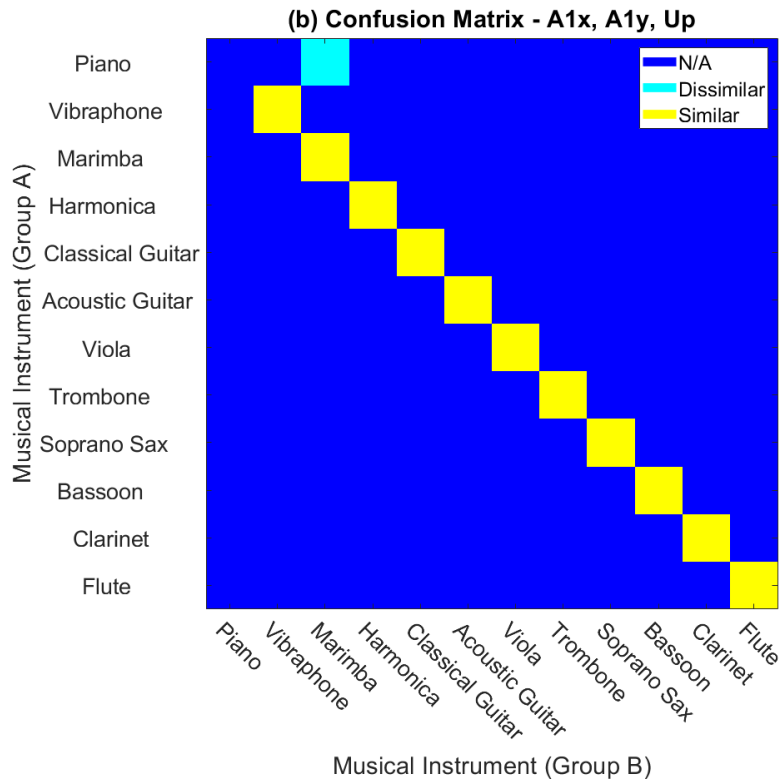


Figure 7-5: Confusion matrix from musical instruments classification using FPGA-implementable timbre distance and KNN classifier of a fixed-point representation of summary profiles of $A1y$ and Up signals using sum operation as well as $A1x$ signal from the CAR-Lite-A1 model. (a) Confusion matrix with raw timbre distance values following classification and (b) ternary-state confusion matrix with minimum timbre distances extracted from (a).

	Floating-Point (Flt)	Fixed-Point (Fix)	Sum (Flt)	Sum (Fix)	RMS (Flt)	RMS (Fix)
Input configurations:						
1) Quantity (Qty):	120	120	60	60	60	60
2) Flt vs. Fix (Qty with higher accuracies):	19	59	-	-	-	-
3) Flt vs. Fix (Qty with equal accuracies):	42		-	-	-	-
4) Flt vs. Fix (Sum qty with higher accuracies):	2	47	-	-	-	-
5) Flt vs. Fix (RMS qty with higher accuracies):	17	12	-	-	-	-
6) Flt vs. Fix (% with higher accuracies):	16%	49%	-	-	-	-
7) Flt vs. Fix (Qty with equal accuracies):	35%		-	-	-	-
8) Flt vs. Fix (Sum % with higher accuracies):	2%	39%	-	-	-	-
9) Flt vs. Fix (RMS % with higher accuracies):	14%	10%	-	-	-	-
10) Sum vs. RMS (Qty with higher accuracies):	-	-	5	30	34	15
11) Sum vs. RMS (% with higher accuracies):	-	-	8%	50%	57%	25%
Classification Accuracies						
Maximum:	83%	100%	83%	100%	83%	92%
Mean:	69%	74%	66%	77%	73%	71%
Median:	75%	75%	67%	79%	75%	75%
Standard Deviation:	9.79%	11.45%	10%	12%	8%	10%

Table 7-3: Summary of musical instruments classification results with a linear classifier. The maximum, mean, median, and standard deviation under each column correspond to the different input signal configurations and the arithmetic type used to calculate the input signals, i.e. either floating-point or fixed-point.

	Floating-Point (Flt)	Fixed-Point (Fix)	Sum (Flt)	Sum (Fix)	RMS (Flt)	RMS (Fix)
Input configurations:						
1) Quantity (Qty):	120	120	60	60	60	60
2) Flt vs. Fix (Qty with higher accuracies):	21	35	-	-	-	-
3) Flt vs. Fix (Qty with equal accuracies):	64		-	-	-	-
4) Flt vs. Fix (Sum qty with higher accuracies):	16	29	-	-	-	-
5) Flt vs. Fix (RMS qty with higher accuracies):	5	6	-	-	-	-
6) Flt vs. Fix (% with higher accuracies):	18%	29%	-	-	-	-
7) Flt vs. Fix (Qty with equal accuracies):	53%		-	-	-	-
8) Flt vs. Fix (Sum % with higher accuracies):	13%	24%	-	-	-	-
9) Flt vs. Fix (RMS % with higher accuracies):	4%	5%	-	-	-	-
10) Sum vs. RMS (Qty with higher accuracies):	-	-	35	31	11	19
11) Sum vs. RMS (% with higher accuracies):	-	-	58%	52%	18%	32%
Classification Accuracies						
Maximum:	92%	92%	92%	92%	67%	75%
Mean:	66%	66%	73%	72%	60%	60%
Median:	67%	67%	83%	75%	67%	67%
Standard Deviation:	14%	16%	13%	19%	11%	10%

Table 7-4: Summary of musical instruments classification results with an FPGA implementable KNN classifier. The maximum, mean, median, and standard deviation under each column correspond to the different input signal configurations and the arithmetic type used to calculate the input signals, i.e. either floating-point or fixed-point.

7.5.2. Varying Intensity and Noise Levels

Real-world sound signals vary in intensity levels and are affected by noise. So, this subsection presents the results of classification of musical instruments based on musical signals with varying intensity and noise levels. The effect of an automatic gain control (AGC) algorithm to condition the amplitudes of the musical signals is also presented. The intensity

levels range from -20 dB full-scale (FS) to 20 dBFS in increments of 10 dBFS. The added noise is white Gaussian noise with the signal-to-noise ratio (SNR) ranging from -20 dB to 20 dB in increments of 20 dB.

Input signals from three musical instruments are selected based on how the notes are generated, as denoted in brackets: piano (string), vibraphone (percussion), and flute (wind). The musical notes used are the same as those from the exercise described in subsection 7.5.1 at three loudness levels: forte (high), mezzo (medium), and piano (low). The output responses of the notes are selected based on the highest accuracy scores from subsection 7.5.1, namely $A1x$, $A1y$ and Up . The musical signals are further categorised in the order of the computing arithmetic utilised for calculating the output response from the CAR-Lite-A1 model: floating-point (Flt), fixed-point output (Fix).

Accuracy results are also presented based on the use of an AGC algorithm on the musical signals before they are input to the CAR-Lite-A1 model. Details of the AGC algorithm are presented in appendix A. An example of the use of the AGC is presented in Figure 7-6 on an A3 signal from a piano at two intensity levels: 0 dBFS and 20 dBFS. At 0 dBFS [Figure 7-6(ia)], the input signal is well represented because its amplitudes are within a range of $2^{N-1} - 1$ and -2^{N-1} . This range is 32,767 and -32,768 for an input signal bit width of 16 bits ($N = 16$) yielding a dynamic range of 96 dB, as specified in section 3.2.8. However, at 20 dBFS, the amplification of the input signal results in the saturation of its amplitudes. So, amplitudes beyond 32,767 and below -32,768 are clipped at 32,767 and -32,768, respectively, as shown in Figure 7-6(ib), which consequently generates audible distortion.

The AGC algorithm minimises amplitude saturation by varying the input signal to ensure that its amplitudes are not clipped, as observed in Figure 7-6(id). However, the application of the AGC to an input signal changes the temporal and spectral contents of the signal, as observed in Figure 7-6. Consequently, these changes affect the timbre of the signal. If the AGC is applied to a group of signals from a musical instrument, the timbre changes uniformly throughout all the signals. Therefore, the group of input signals is considered as a separate class as opposed to the same signals not conditioned by the AGC. In other words, the classification of musical instruments has to be further segregated based on whether the AGC is enabled.

The AGC algorithm is not implemented on FPGA because an AGC circuit has already been implemented as part of the audio codec circuit [33] on board the Cyclone V FPGA starter kit [34]. However, the AGC algorithm described in appendix A is more akin to intensity level conditioning in a cochlear model such as the AGC used in the CAR-FAC model [35] and hence, provides an alternative perspective to a conventional hardware AGC.

Figure 7-7 displays the accuracy results of varying the intensity levels of the musical signals presented to the CAR-Lite-A1 model. The accuracy of each level is the average of the output responses across varying SNR levels (three recordings per SNR level), which include output responses unaffected by noise. The full results are shown in appendix E. Without noise, and AGC disabled, accuracy scores of floating-point (Flt) and fixed-point (Fix) responses at all intensity levels are at 100% using the 2D correlation coefficient (CC) algorithm in the linear classifier and 67% using the timbre distance (TD) algorithm in the classifier. This is regardless of the clipping of the input signals above 0 dBFS and up to 20

dBFS, as observed in Figure 7-6(ib), which indicates that the responses of the CAR-Lite-A1 model can be represented capably even though their respective notes' amplitudes are saturated.

When the AGC is enabled, the envelope of the musical signal is altered from the original, as observed in Figure 7-6(ic) and (id) and (ii). Although the *A1y* and *Up* profiles are different with and without the AGC activation, their timbre cues remain categorically similar when either AGC is enabled or disabled regardless of the change in intensity. The accuracy results are at 100% from both the linear and KNN classifiers, which are improvements over the scores when the AGC is disabled, as mentioned above, especially for the KNN classifier. This attribute indicates that the musical timbral features are retained despite envelope alterations with the introduction of the AGC.

Figure 7-8 displays the accuracy scores of varying the noise levels of the musical signal. Each noise level has three sets of recordings for every intensity level, which are then averaged. Multiple recordings are done here as white Gaussian noise introduces randomness to the contents of the output signals. When noise is introduced at 20 dB SNR, there are instances where the accuracy is maintained, as observed in Figure 7-8(id) and (iid), or improved, as seen in Figure 7-8(ib) and (iib). This attribute indicates that the presence of mild noise may maintain and even improve the capability of the classifiers especially, the KNN classifier to distinguish between musical instruments regardless of whether the AGC is enabled and irrespective of the summary output responses calculated using either sum or RMS operations. However, this is only the case for 50% of the recordings at 20 dB SNR. A more general observation is that the accuracy reduces as the SNR reduces especially below 20 dB SNR, which indicates that a more prominent noise level commonly obscures the classifiers from distinguishing musical instruments than when there is milder noise.

Figure 7-9(a) displays the average of standard deviations of accuracy scores across the intensity levels from Figure 7-7, which indicate how much the scores vary at each intensity level. Below 0 dBFS, the scores calculated using the TD algorithm in the KNN classifier fluctuate the most. Above 0 dBFS, the scores calculated using the CC algorithm in the linear classifier fluctuate the most. These fluctuations occur when the AGC is disabled. In contrast, when the AGC is enabled, the fluctuations reduce. This attribute indicates that the AGC algorithm aids in maintaining the accuracy scores, but these scores are generally lower than the scores when the AGC is disabled, especially for the sum-based response as observed in Figure 7-7.

Figure 7-9(b) displays the average of the standard deviations of accuracy scores across the SNR levels from Figure 7-8, indicating how much the scores vary at each SNR level. At no noise levels, the scores expectedly do not vary at all. At 20 dB SNR and below, the fluctuation of the scores increases with decreasing SNR levels. In other words, as noise level grows, the variations of accuracy scores grow. Furthermore, the contrast between the linear classifier scores are more significant across floating-point and fixed-point responses as well as summed and RMS responses at every SNR level, especially when the AGC is disabled. This attribute indicates that the scores calculated from the CC algorithm in the linear classifier when the AGC is disabled fluctuate more than any other settings.

Generally, varying intensity levels changes the perceived loudness of a sound signal. However, perceived loudness does not affect timbre [19], [36]. Hence, intensity levels should not affect timbre. In this subsection, it was found that the results of musical instruments classification are not affected significantly by intensity level changes, thereby adhering to the conclusions of [19], [36]. Additionally, increasing noise levels added to the sound expectedly reduced the performance of the classification indicating the degradation of timbral cues with increasing noise levels.

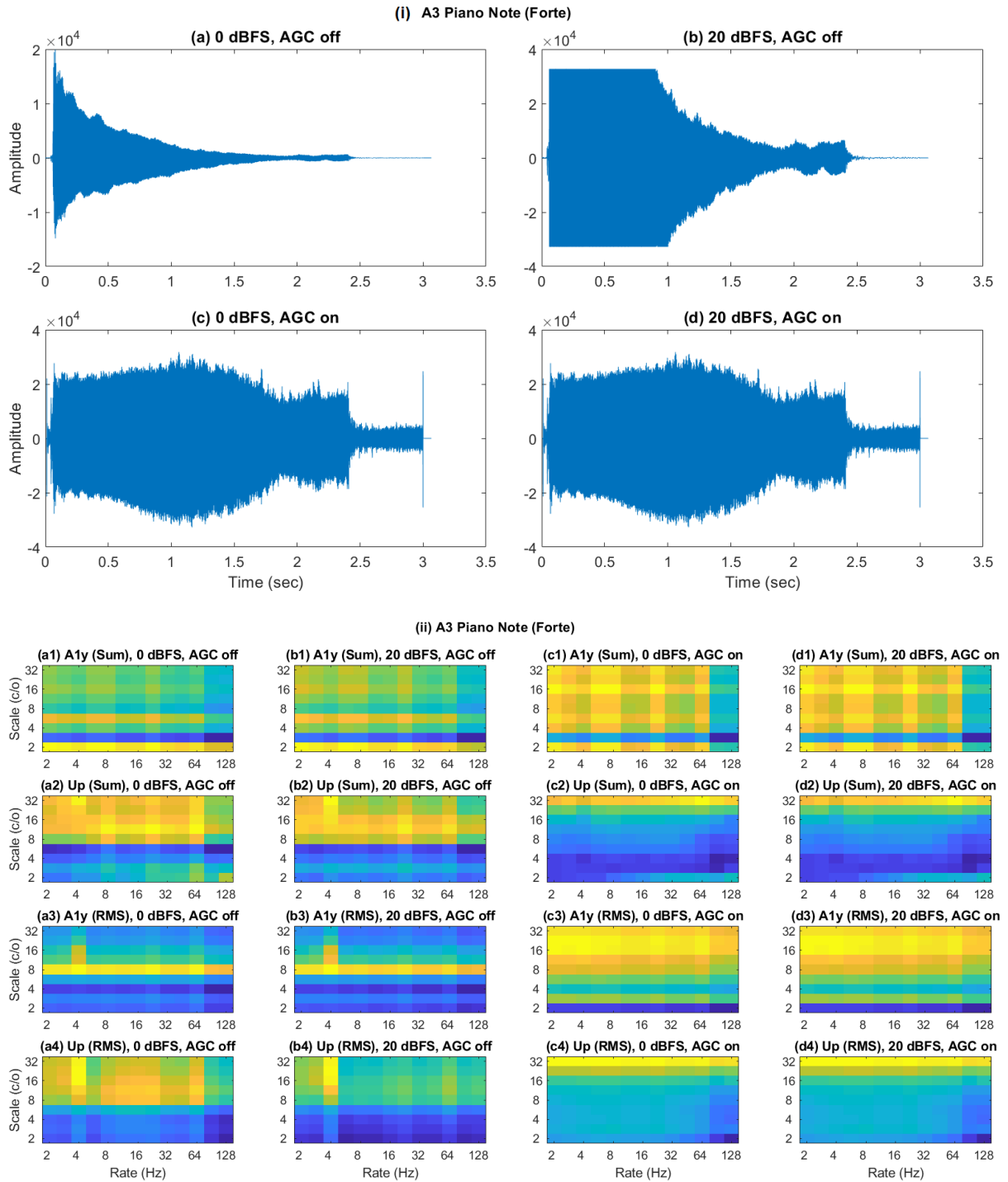


Figure 7-6: (i) Fixed-point representation of the A3 piano note signal at two intensity levels, 0 dBFS and 20 dBFS, with the AGC algorithm, enabled and disabled and (ii) their corresponding $A1y$ and Up profiles showing differences with and without the AGC enabled but are categorically similar despite different intensity levels.

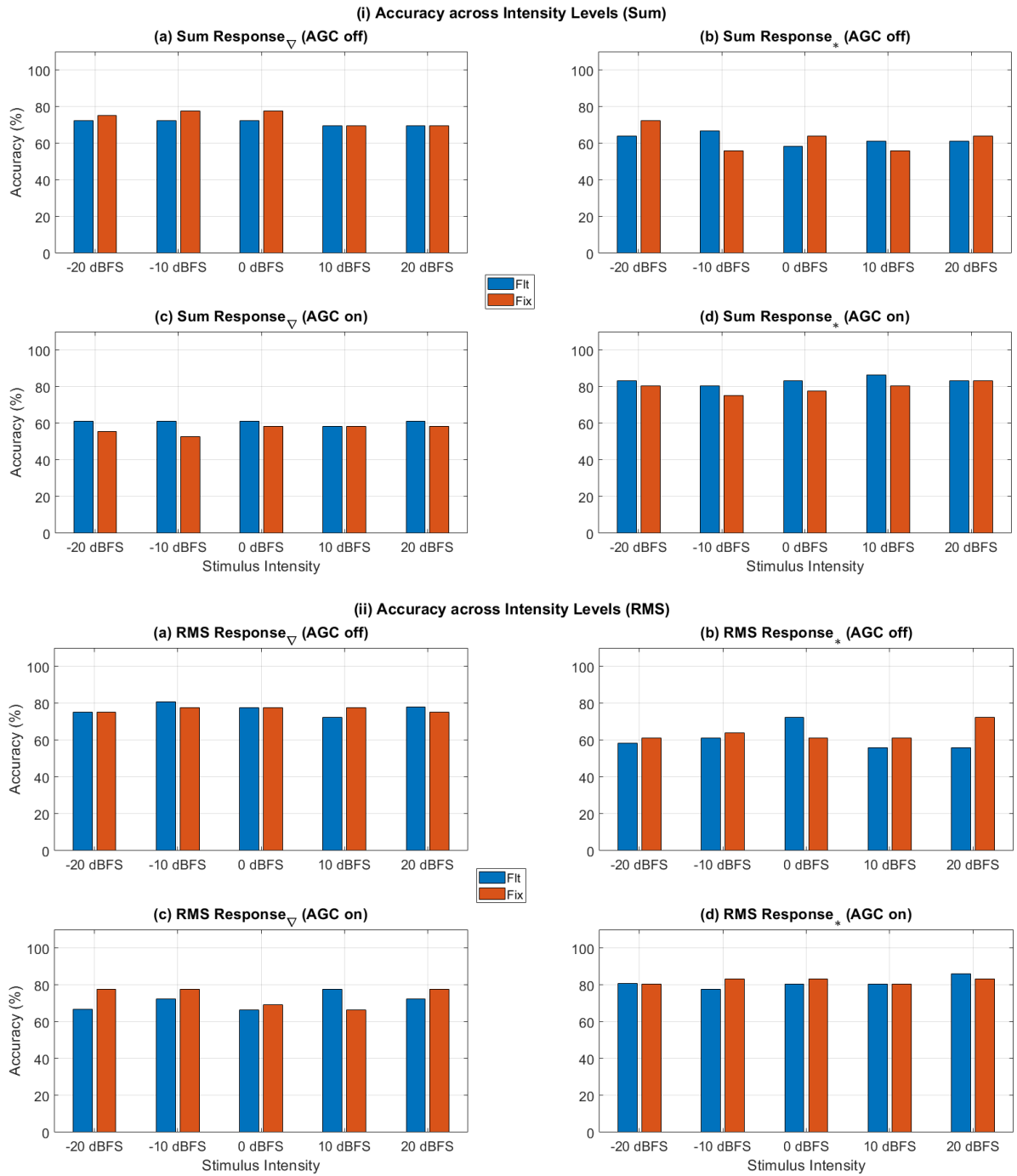


Figure 7-7: Accuracy scores based on varying intensity levels of musical signals for their (i) summed and (ii) RMS output responses from the CAR-Lite-A1 model that are averaged across varying SNR levels. Vertical bars represent standard deviations. Note: ∇ represents linear classifier; $*$ represents KNN classifier; FIt is the classification scores of the responses of input signals calculated with floating-point arithmetic; Fix is the classification of the responses of input signals calculated with fixed-point arithmetic.

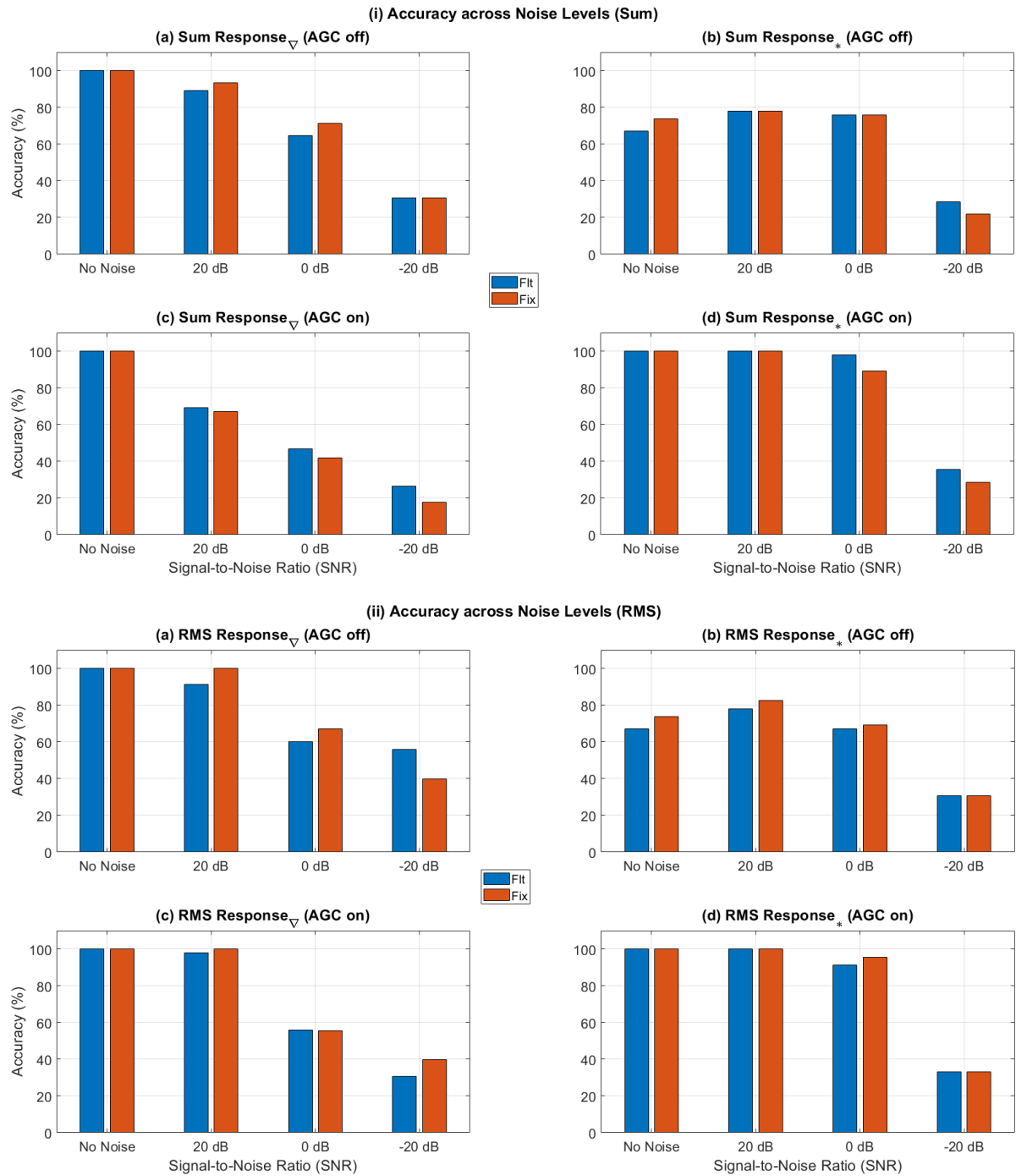


Figure 7-8: Accuracy results based on varying signal-to-noise (SNR) levels of musical signals for their (i) summed and (ii) RMS output responses from the CAR-Lite-A1 model that are averaged across varying intensity levels. Vertical bars represent standard deviations. Note: ∇ represents linear classifier; $*$ represents KNN classifier; FIt is the classification scores of the responses of input signals calculated with floating-point arithmetic; Fix is the classification of the responses of input signals calculated with fixed-point arithmetic.

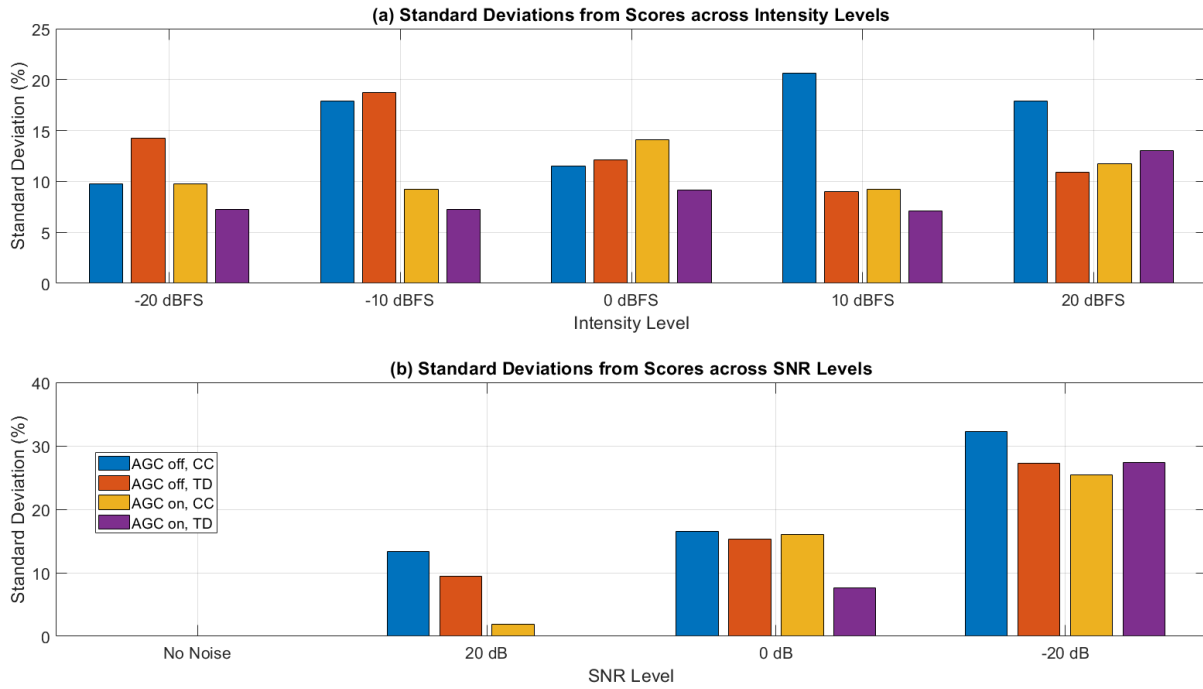


Figure 7-9: Mean of the standard deviation of accuracy scores from Figure 7-7 and Figure 7-8 shown across (a) intensity levels and (b) SNR levels. CC: 2D correlation coefficient algorithm in the linear classifier; TD: Timbre distance algorithm in the KNN classifier.

7.5.3. In-model Comparison: 4D and 2D Matrix File Sizes

Commercial automatic music identification requires the use of over millions of 2D spectrograms representing musical tracks in an extensive database, which consequently requires large electronic memory for storage [37]. In section 6.3.3, it was reported that 2D autocorrelograms computed in fixed-point arithmetic has lower storage size than autocorrelograms computed in floating-point arithmetic. In this section, the storage size difference is explored for 4D matrices and 2D summary matrices generated using floating-point and fixed-point arithmetic.

Figure 7-10 displays the file sizes of 4D and summary 2D matrices ($A1y$, Up , and $Down$) generated from the CAR-Lite-A1 model as well as the IHC spectral profile ($A1x$) at 0 dBFS without noise. A significant contrast in file sizes is observable in (a) and (b) of Figure 7-10 – the 4D fixed-point response requires 15 times less storage space [946.4 Gigabytes (GB) / 62.56 GB] than the 4D floating-point response. This file size contrast is due to the former using double-precision numbers, as opposed to single-precision numbers used by the latter.

The fixed-point sum-calculated 2D summary response requires two times less storage space [15.19 Megabytes (MB) / 6.58 MB] than its floating-point equivalent. However, the file sizes of the RMS-calculated 2D summary responses are similar (7.93 MB / 7.712 MB) for both floating-point and fixed-point arithmetic. This is because the RMS operation is calculated in floating-point arithmetic regardless of the input signal format. Another observation is the significant storage space difference between the 4D and 2D responses: the 2D floating-point response for both sum-calculated and RMS-calculated summary profiles require at least 40,000 times less [946.4 GB / (7.93 MB + 15.19 MB)] storage space than the 4D floating-point response and the 2D fixed-point response for both sum-calculated and RMS-calculated summary profiles require 4,400 times less [62.56 GB / (7.71 MB + 6.58 MB)] storage space than the 4D fixed-point response.

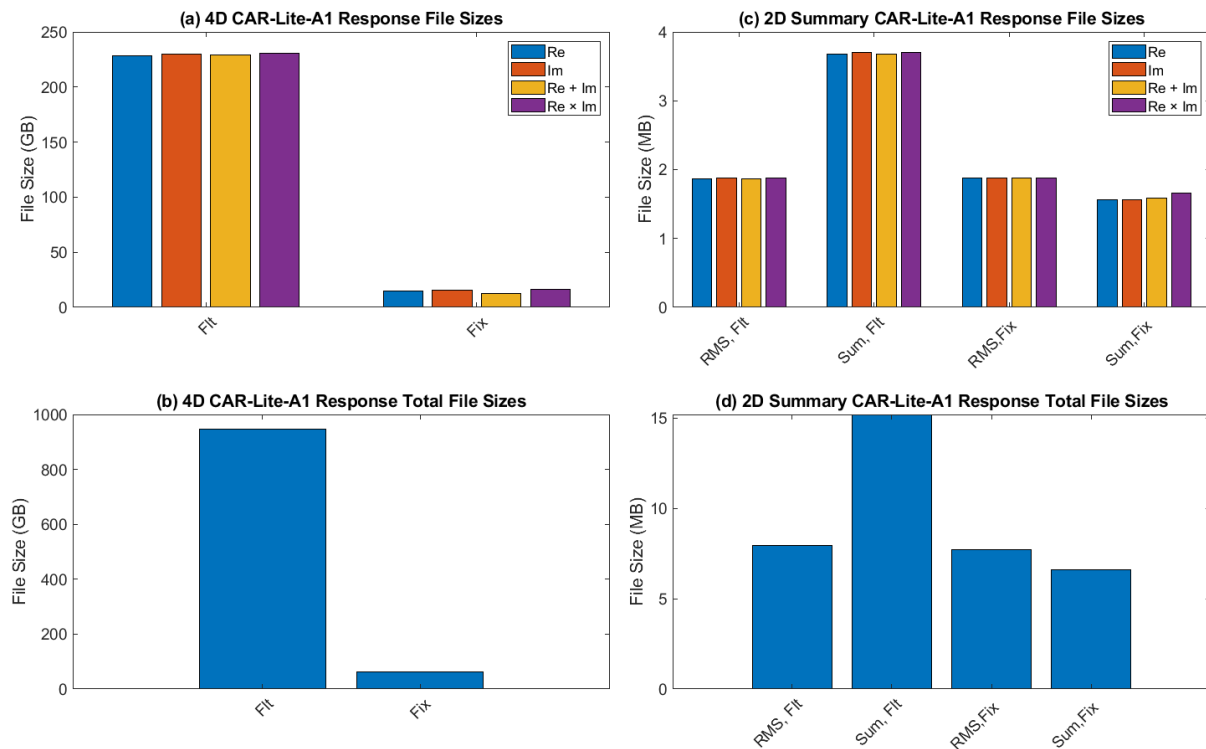


Figure 7-10: File sizes of the CAR-Lite-A1 model output from the inner hair cell (IHC), scale filter, upward and downward neuron directional matrices calculated using one of the four following circuit configurations: Real (Re), Imaginary (Im), Re + Im, and Re \times Im. File sizes displayed are for (a) 4D output matrices across all four circuits; (b) 4D output matrices accumulated from all four circuits; (c) summary 2D output matrices across all circuits and (d) summary 2D output matrices accumulated from all four circuits.

7.5.4. Comparison with Other Models: Accuracy

Table 7-5 displays the highest classification accuracy scores of musical instruments from five publications as well as the work presented from subsection 7.5.1. The comparison is only for studies at 0 dBFS, as the majority of the musical timbre studies revolve around single intensity levels for monophonic musical signals as outlined in Table 7-5 as well as of polyphonic musical signals [38]–[40], which are excluded from the table.

The highest classification accuracy score of 100% from the linear classifier used in the first half of this work is ranked the highest out of the seven scores presented. Alternatively, the highest classification accuracy score of 91.7% from the k-nearest neighbour (KNN) classifier used in the second half of this work is ranked fourth, outranking the linear classifier used by Barbedo and Tzanetakis [41], single hidden layer artificial neural network (ANN) with back-propagation used by Kostek [42], and support vector machine (SVM) configured with nonlinear radial basis function used by Essid et al. [43]. These three outranked works use conventional spectral-only and temporal-only algorithms to extract features as detailed in Table 7-6 for classification. In contrast, the higher accuracy scores from this work as well as the works of Patil et al. [15] and Burred et al. [44] are extracted using spectro-temporal algorithms.

As mentioned, in subsection 7.5.1, the software-based linear classifier using the 2D correlation coefficient algorithm is capable of quantitatively capturing small magnitude differences between two classes. Thus, it is sufficiently robust to attain 100% classification accuracy for two separate input combinations. Alternatively, the timbre distance algorithm in the hardware-based KNN classifier is a simple algorithm that is designed to be

implementable on an FPGA. The simple design of the KNN classifier means that it is not as robust as the linear classifier. This characteristic is evident as the former has a lower classification accuracy of 91.7% over the latter.

In the case of Burred et al., the high classification accuracy score is achieved by applying principal component analysis to rank the errors of detectable spectro-temporal envelopes. This high accuracy score is attributable to a small number of musical instrument classes used in the classification task - it is not known how much the accuracy score is affected if the number of musical instrument classes is increased. In the case of Patil et al., a nonlinear support vector machine (SVM) with Gaussian kernel, similar to Essid et al., was used to achieve a very high score. Alternatively, even with a linear SVM, Patil et al. achieved an accuracy score of 96.2%, which is close to the maximum accuracy score of 98.7% he achieved using a nonlinear SVM, as presented in Table 7-5.

The classification accuracy errors encountered in this work, specifically by the hardware-based KNN classifier, stem from several factors. Firstly, the mechanics of the KNN classification is simple as it is designed to be implementable on FPGA. So, the matching of two musical instrument classes is a straightforward measure of a timbre distance between them. Conversely, Patil et al. used a more computationally intensive and mathematically-involved classifier than the KNN classifier, known as a support vector machine (SVM) to classify musical instruments – the nonlinear kernels in the SVM projects data to a higher dimensional plane, known alternatively as a hyperplane, so that the data is linearly separable. In contrast, the KNN classifier does not project data on to a higher plane, which plausibly cause lower classification accuracy.

Secondly, data dimensionality reduction (from 4D to 2D) used by Patil et al. to achieve the 98.7% accuracy involved the use of tensor singular value decomposition that had parameters tuned to match psychoacoustical timbre experiments, which allowed distinct musical timbre features to be extracted. Conversely, dimensionality reduction in this work was achieved by sum operations that are implementable without high computational cost on FPGA, and RMS operations that are implementable in software to work in real-time. These algorithms cannot be optimised to extract distinct features corresponding to the same psychoacoustic experiments used by Patil et al. As a result, the lack of distinct feature selectivity is another cause for the errors in classification accuracy.

Thirdly, the number of instrument classes used by the KNN classifier in this work is slightly higher than the work of Patil et al. and equal to Kostek. Incidentally, Kostek achieved the lowest accuracy of 71% out of the six other works presented in Table 7-5. Only the work of Barbedo and Tzanetakis has a higher number of instrument classes at 25 but yields a lower classification accuracy score than this work. Hence, a general observation from Table 7-5 is that as the number of instrument classes increases, there is a likelihood of a reduction in classification accuracy owing to the variances in timbre cue. However, the effect of increasing instrument classes may be overlooked if a sophisticated classification algorithm and feature extraction methods are considered.

In closing, the common theme in the top-four classification accuracy scores from Table 7-5 is the use of spectro-temporal features for classification as summarised in Table 7-6. Hence, in all the seven cases presented, it can be concluded that in addition to the classification algorithms and their respective linearity, the feature extraction methods and the

number of instrument classes are crucial in achieving high accuracy scores as is the case primarily when spectro-temporal envelope features are used.

Index	Author	Number of Instrument Classes	Classifier	Accuracy
1	Patil et al. [15]	11	SVM	98.7%
2	Burred et al. [44]	5	PCA	94.9%
3	Essid et al. [43]	10	SVM	87.0%
4	Barbedo and Tzanetakis [41]	25	Linear	80.3%
5	Kostek [42]	12	ANN	71.3%
6	This work	12	Linear	100%
7	This work	12	KNN	91.7%

Table 7-5: Comparison of the classification of musical instruments between this work and other works with the highest classification accuracy score displayed from each work.

Index	Author	Features
1	Patil et al. [15]	Multiple spectro-temporal modulation profiles.
2	Burred et al. [44]	Spectro-temporal envelope descriptors, fundamental frequency (f_0) correlated descriptors, and f_0 -invariant descriptors.
3	Essid et al. [43]	Autocorrelation coefficients, zero crossing rates, mel-frequency cepstral coefficients, spectral centroid, spectral widths, spectral asymmetry, spectral skewness, spectral flatness, and octave band signal intensities.
4	Barbedo and Tzanetakis [41]	Bandwidth, crest factor, onset duration, slope of amplitude decay, amplitude roughness, amplitude envelope variation, centroid, note spread, skewness, kurtosis, amplitude modulation (AM) frequency, AM amplitude, and frequency trajectory.
5	Kostek [42]	MPEG7 feature vectors and wavelet-based feature vectors.
6, and 7	This work	Summary inner hair cell spectral profile, summary spectro-temporal modulation (rate-scale filter output) profile, and summary spectro-temporal modulation directional (upward and downward) neuron profiles.

Table 7-6: Features used for the classification of musical instruments from the works mentioned in Table 7-5.

7.6. Summary and Conclusion

This chapter presents the classification of musical instruments using the responses of the CAR-Lite-A1 model described in chapter 5 as inputs to the classifiers. Four musical notes at three discrete loudness levels are selected from 12 classes of musical instruments. Each class is divided into two subclasses, which denote the same musical instrument type built by two different manufacturers. For each musical note, four types of responses from the CAR-Lite-A1 model are used for musical instruments classification: a summary inner hair cell spectral profile, a summary rate-scale filter output profile, a summary upward neuron directional profile and a summary downward neuron directional profile. The summary profiles are generated using either a sum or RMS operation. The responses of the CAR-Lite-A1 model are further segregated based on floating-point (software) arithmetic and fixed-point (hardware) arithmetic as well as the four analytic circuit combinations: ‘real’, ‘imaginary’,

'real+imaginary', and 'real×imaginary', which offers a broader range of input signal combinations to formulate a classification accuracy than if only the 'real' component is used.

Two classification algorithms are used for the classification. One is a linear classifier that uses a 2D correlation coefficient equation whose correlation coefficient defines the similarity between two 2D input matrices. The other is a KNN classifier that uses a simpler algorithm known as timbre distance, whose design is based on the principle Occam Razor to match two 2D input matrices, which is implementable on an FPGA. The linear classifier generates the highest musical instruments classification accuracy score at 100%, while the maximum accuracy score for the KNN classifier stood closely at 92%. Both results are from sum-calculated input summary profiles using hardware-based fixed-point arithmetic. The accuracy scores are approximately 2.3 times higher for input summary profiles calculated with hardware-based fixed-point arithmetic than software-based floating-point arithmetic. The accuracy scores are also relatively uniform across multiple intensities of the input musical signals. Furthermore, with the introduction of an automatic gain control (AGC), the scores also remain relatively uniform regardless of the intensities. The introduction of noise predictably reduces the accuracy scores – more noise reduces performance.

In conclusion, the hardware-based fixed-point auditory model of the CAR-Lite-A1 model presented in chapter 5 is capable of generating responses for representing musical timbre, and consume lower storage space than its software-based floating-point counterpart, as observed from the results presented in this chapter. However, to gauge its robustness, the model has to be fully implemented on FPGA and tested with a more diverse range of sound signals.

7.7. Bibliography

- [1] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Music Genre Database and Musical Instrument Sound Database," in *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, 2003, pp. 229–230.
- [2] D. L. Wessel, "Timbre Space as a Musical Control Structure," *Comput. Music J.*, vol. 3, no. 2, pp. 45–52, 1979, [Online]. Available: <http://www.jstor.org/stable/3680283>.
- [3] C. L. Krumhansl and P. Iverson, "Perceptual Interactions Between Musical Pitch and Timbre," *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 18, no. 3, pp. 739–751, 1992, doi: 10.1037/0096-1523.18.3.739.
- [4] S. Samson, R. J. Zatorre, and J. O. Ramsay, "Multidimensional Scaling of Synthetic Musical Timbre: Perception of Spectral and Temporal Characteristics," *Can. J. Exp. Psychol.*, vol. 51(4), no. Dec., pp. 307–315, 1997, doi: 10.1037/1196-1961.51.4.307.
- [5] G. Yu and J.-J. Slotine, "Audio Classification from Time-Frequency Texture," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 1677–1680, doi: 10.1109/ICASSP.2009.4959924.
- [6] T. R. Agus, C. Suied, S. J. Thorpe, and D. Pressnitzer, "Fast recognition of musical sounds based on timbre," *J. Acoust. Soc. Am.*, vol. 131, no. 5, pp. 4124–4133, 2012, doi: 10.1121/1.3701865.
- [7] M. a Loureiro, H. B. De Paula, and H. C. Yehia, "Timbre Classification Of A Single Musical Instrument," in *ISMIR 2004*, 2004, pp. 546–549, doi: 10.1.1.106.416.

- [8] J. W. Gordon, "The perceptual attack time of musical tones," *J. Acoust. Soc. Am.*, vol. 82, no. 1, pp. 88–105, 1987, doi: 10.1121/1.395441.
- [9] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams, "The Timbre Toolbox: Extracting Audio Descriptors from Musical Signals," *J. Acoust. Soc. Am.*, vol. 130, no. 5, p. 2902, 2011, doi: 10.1121/1.3642604.
- [10] J. Marozeau, A. de Cheveigné, S. Mcadams, and S. Winsberg, "The dependency of timbre on fundamental frequency," *J. Acoust. Soc. Am.*, vol. 114, no. 5, pp. 2946–2957, 2003, doi: 10.1121/1.1618239.
- [11] S. McAdams, S. Winsberg, S. Donnadieu, G. De Soete, and J. Krimphoff, "Perceptual Scaling of Synthesized Musical Timbres: Common Dimensions, Specificities, and Latent Subject Classes," *Psychol. Res.*, vol. 58, no. 3, pp. 177–192, 1995, doi: 10.1007/BF00419633.
- [12] A. Caclin, E. Brattico, M. Tervaniemi, R. Näätänen, D. Morlet, M.-H. Giard, and S. McAdams, "Separate Neural Processing of Timbre Dimensions in Auditory Sensory Memory," *J. Cogn. Neurosci.*, vol. 18, no. 12, pp. 1959–1972, 2006, doi: 10.1162/jocn.2006.18.12.1959.
- [13] Y. Kong, A. Mullangi, J. Marozeau, and M. Epstein, "Temporal and Spectral Cues for Musical Timbre Perception in Electric Hearing," *J. Speech, Lang. Hear. Res.*, vol. 54, no. June, pp. 981–995, 2011, doi: 10.1044/1092-4388(2010/10-0196)b.
- [14] S. Shamma, "Encoding Sound Timbre in the Auditory System," *IETE J. Res.*, vol. 49, no. 2, pp. 145–156, 2003, doi: 10.1080/03772063.2003.11416333.
- [15] K. Patil, D. Pressnitzer, S. Shamma, and M. Elhilali, "Music in Our Ears: The Biological Bases of Musical Timbre Perception," *PLoS Comput. Biol.*, vol. 8, no. 11, pp. 1–16, 2012, doi: 10.1371/journal.pcbi.1002759.
- [16] S. Handel and M. L. Erickson, "A Rule of Thumb: The Bandwidth for Timbre Invariance Is One Octave," *Music Percept. An Interdiscip. J.*, vol. 19, no. 1, pp. 121–126, 2001, doi: 10.1525/mp.2001.19.1.121.
- [17] K. M. Steele and A. K. Williams, "Is the Bandwidth for Timbre Invariance only One Octave?," *Music Percept. An Interdiscip. J.*, vol. 23, no. 3, pp. 215–220, 2006, doi: 10.1525/mp.2006.23.3.215.
- [18] B. M. Calhoun and C. E. Schreiner, "Spectral envelope coding in cat primary auditory cortex: linear and non-linear effects of stimulus characteristics," *Eur. J. Neurosci.*, vol. 10, no. 3, pp. 926–940, 1998, doi: 10.1046/j.1460-9568.1998.00102.x.
- [19] R. D. Melara and L. E. Marks, "Interaction among auditory dimensions: Timbre, pitch, and loudness," *Percept. Psychophys.*, vol. 48, no. 2, pp. 169–178, 1990, doi: 10.3758/BF03207084.
- [20] M. Thieme, "Dynamics," *Grove Music Online*, 2019. .
- [21] T. Nakamura, "The communication of dynamics between musicians and listeners through," *Percept. Psychophys.*, vol. 41, no. 6, pp. 525–533, 1987, doi: 10.3758/BF03210487.
- [22] M. Fabiani and A. Friberg, "Influence of pitch, loudness, and timbre on the perception of instrument dynamics," *J. Acoust. Soc. Am.*, vol. 130, no. 4, pp. EL193–EL199, 2011, doi: 10.1121/1.3633687.

- [23] OnMusic Dictionary, “forte,” 2015. <http://dictionary.onmusic.org/terms/1487-forte> (accessed Feb. 06, 2019).
- [24] OnMusic Dictionary, “mezzo,” 2015. <http://dictionary.onmusic.org/terms/2162-mezzo> (accessed Feb. 06, 2019).
- [25] OnMusic Dictionary, “piano,” 2015. <http://dictionary.onmusic.org/terms/2594-piano> (accessed Feb. 06, 2019).
- [26] Y. Feng, Y. Zhuang, and Y. Pan, “Music Information Retrieval by Detecting Mood via Computational Media Aesthetics,” in *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*, 2003, doi: 10.1109/WI.2003.1241199.
- [27] M. D. McDonnell, M. D. Tissera, T. Vladusich, A. Van Schaik, J. Tapson, and F. Schwenker, “Fast, Simple and Accurate Handwritten Digit Classification by Training Shallow Neural Network Classifiers with the ‘Extreme Learning Machine’ Algorithm,” *PLoS One*, vol. 10, no. 8, pp. 1–20, 2015, doi: 10.1371/journal.pone.0134254.
- [28] R. Meddis and M. J. Hewitt, “Virtual pitch and phase sensitivity of a computer model of the auditory periphery. I: Pitch identification,” *J. Acoust. Soc. Am.*, vol. 89, no. 6, pp. 2866–2882, 1991, doi: 10.1121/1.400725.
- [29] T. Chi, P. Ru, and S. A. Shamma, “Multiresolution Spectrotemporal Analysis of Complex Sounds,” *J. Acoust. Soc. Am.*, vol. 118, no. 2, pp. 887–906, 2005, doi: 10.1121/1.1945807.
- [30] T. Dau, B. Kollmeier, and A. Kohlrausch, “Modeling auditory processing of amplitude modulation. II. Spectral and temporal integration,” *J. Acoust. Soc. Am.*, vol. 102, no. 5, pp. 2906–2919, 1997, doi: 10.1121/1.420345.
- [31] A. Hasnat, T. Bhattacharyya, A. Dey, S. Halder, and D. Bhattacharjee, “A Fast FPGA Based Architecture for Computation of Square Root and Inverse Square Root,” in *2017 Devices for Integrated Circuit (DevIC)*, 2017, no. 1, pp. 383–387, doi: 10.1109/DEVIC.2017.8073975.
- [32] R. Meddis and L. O’Mard, “A unitary model of pitch perception,” *J. Acoust. Soc. Am.*, vol. 102, no. 3, pp. 1811–1820, 1997, doi: 10.1121/1.420088.
- [33] Analog Devices, “SSM2603: Low Power Audio Codec (Rev. B).” Analog Devices, pp. 1–32, 2012.
- [34] Terasic and Altera Corporation, “Cyclone V GX Starter Kit - User Manual.” Terasic, pp. 1–103, 2014.
- [35] R. F. Lyon, “The AGC Loop Filter,” in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 331–344.
- [36] R. D. Melara and L. E. Marks, “Hard and soft interacting dimensions: Differential effects of dual context on classification,” *Percept. Psychophys.*, vol. 47, no. 4, pp. 307–325, 1990, doi: 10.3758/BF03210870.
- [37] A. L. Wang and K. H. Street, “An Industrial-Strength Audio Search Algorithm,” in *2003 ISMIR International Symposium on Music Information Retrieval*, 2003.
- [38] J.-J. Aucouturier, F. Pachet, and M. Sandler, “‘The Way It Sounds’: Timbre Models for Analysis and Retrieval of Music Signals,” *IEEE Trans. Multimed.*, vol. 7, no. 6, pp. 1028–1035, 2005, doi: 10.1109/TMM.2005.858380.

- [39] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Instrument Identification in Polyphonic Music: Feature Weighting with Mixed Sounds, Pitch-Dependent Timbre Modeling, and Use of Musical Context," *Proc. 6th Int. Soc. Music Inf. Retr. Conf.*, pp. 558–563, 2005.
- [40] T. Heittola, A. Klapuri, and T. Virtanen, "MUSICAL INSTRUMENT RECOGNITION IN POLYPHONIC AUDIO USING SOURCE-FILTER MODEL FOR SOUND SEPARATION," in *International Society for Music Information Retrieval Conference*, 2009, [Online]. Available: <http://www.cs.tut.fi/sgn/arg/klap/ismir09-heittola.pdf>.
- [41] J. G. A. Barbedo and G. Tzanetakis, "Musical Instrument Classification Using Individual Partial," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 19, no. 1, pp. 111–122, 2011, doi: 10.1109/TASL.2010.2045186.
- [42] B. Kostek, "Musical Instrument Classification and Duet Analysis Employing Music Information," *Proc. IEEE*, vol. 92, no. 4, pp. 712–729, 2004, doi: 10.1109/JPROC.2004.825903.
- [43] S. Essid, G. Richard, and B. David, "Musical Instrument Recognition by Pairwise Classification Strategies," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 14, no. 4, pp. 1401–1412, 2006, doi: 10.1109/TSA.2005.860842.
- [44] J. Burred, A. Robel, and T. Sikora, "Dynamic Spectral Envelope Modeling for Timbre Analysis of Musical Instrument Sounds," *Audio, Speech, Lang. ...*, vol. 18, no. 3, pp. 663–674, 2010, doi: 10.1109/TASL.2009.2036300.
- [45] D. P. W. Ellis, "tf_agc - Time-frequency automatic gain control," 2010. https://labrosa.ee.columbia.edu/matlab/tf_agc/ (accessed May 10, 2020).
- [46] M. Slaney, "Auditory Toolbox Version 2," 1998. [Online]. Available: <https://engineering.purdue.edu/~malcolm/interval/1998-010/AuditoryToolboxTechReport.pdf>.

8. Summary, Conclusion, and Future Work

This chapter is divided into three sections. Section 8.1 summarises and concludes this dissertation. Section 8.2 suggests ideas for continuing on the work presented in this dissertation. Section 8.3 presents publications resulting from the work presented.

8.1. Summary and Conclusion

In this dissertation, several neuromorphic auditory models are presented. These models are designed in software in floating-point arithmetic before being implemented on an FPGA in fixed-point arithmetic. These models are designed to keep computational cost low when running on hardware and so, they are all designed to characterise optimal operations of mammalian auditory models. For all the models presented, the computational resources used on an FPGA are also presented, which include logic modules utilisation, registers utilisation as well as power consumption.

The first model presented is a simple cochlear model described in the first half of chapter 3, titled CAR-Lite. Using multiple sampling rates, the CAR-Lite is capable of using nine times fewer coefficients than a model that uses a single sampling rate, resulting in smaller non-volatile storage of variables for the former than the latter. This low storage reduces the use of memory chips, yielding low power consumption and manufacturing cost and also reduces the silicon area of a circuit that is appealing for mobile devices. The response of the CAR-Lite model is also investigated with a musical input signal at multiple intensity levels where it was found that signals with amplitudes larger than 0 dBFS are saturated to the largest smallest number representable for a 16 bit signal. To correct this issue, an automatic gain control (AGC) algorithm is used for smoothly varying the levels to ensure that the input signal is fully representable within the dynamic range of the model before they are injected into the model.

The second half of chapter 3 discusses the inability of the CAR-Lite model to represent spike trains at multiple intensity levels, which leads to the formation of the CAR-Lite-SI. This model involves the modification and the extension of the CAR-Lite model to include a biologically inspired spiking algorithm with multiple neuronal firing thresholds as a demonstration of its capability to represent sound intensity (SI), which the CAR-Lite model is unable to achieve.

In chapter 4, the CAR-Lite model is used with an algorithm for generating an autocorrelogram (AC) to represent auditory pitch. Known as CAR-Lite-ACF, the model generates an AC using the autocorrelation function, which involves the correlation of signals from individual cochlear sections with delayed version of themselves. A novel method of approximating the autocorrelation function is presented using logical AND-accumulate (AAC) operations instead of the conventional multiply-accumulate (MAC) operations. This novel algorithm is similar to the biologically inspired spiking algorithm from the CAR-Lite-SI model except that it utilises a single neuron firing threshold instead of multiple firing thresholds. Using this novel method, salient information corresponding to high energy above the firing threshold is retained in regard to the fundamental frequency represented in the input signal. Conversely, low energy that may contribute to general pitch information in regard to the harmonics of the fundamental frequency is removed from the ACs, as the spiking algorithm contributes to more significant quantisation error introduced here from the AAC operations

over the conventional MAC operations. Moreover, the use of spike streams as input to the autocorrelation function instead of a smoothed input signal introduces noises in ACs that require additional computation for smoothing. However, the CAR-Lite-ACF model equipped with AAC operations utilises less computational resources on FPGA than the model equipped with MAC operations in addition to representing fundamental frequencies.

In chapter 5, the CAR-Lite-A1 model is presented, where the CAR-Lite model is used with a functional primary auditory cortical (A1) model, primarily designed for representing musical timbre – sound unique to a specific musical instrument. The A1 segment of the model comprises a rate filterbank and a scale filterbank for extracting spectro-temporal envelopes respectively from an input sound signal, to generate a 4D output. The two filterbanks are designed with the same filters as the ones used in the CAR-Lite cochlear model due to their low coefficient utilisation. The model also has a spectro-temporal envelope directional filterbank to detect the rise and fall in the amplitudes of the spectral and temporal envelopes for use as timbral features. However, the directional filterbank does not calculate phase information of the rate and scale filterbanks, which are disregarded to maintain a low computational cost. The 4D response of the directional filterbank comprising time, frequency, rate (temporal envelopes), and scale (spectral envelopes) is transformed into a 2D rate-scale summary profile for a compact timbre representation. This transformation is beneficial in musical timbre classification, as a 2D summary profile representing a musical signal has a fixed matrix size, despite the varying time durations (lengths) of musical signals from various musical instruments.

In chapter 6, a description of the classification of musical notes is presented using autocorrelograms (ACs) generated from real-world musical signals. The pitch information extracted from an AC is a fundamental frequency, which is estimated using algorithms designed with musical notes from the fourth octave of a piano. Additionally, the implementations of these algorithms grow in complexity to increasing pitch estimation accuracies. The estimated fundamental frequencies are compared with the ground truth of musical note frequencies. It is shown that both the fundamental frequency estimation and classification algorithms are implementable on an FPGA (hardware). The pitch estimation algorithm is found to produce the highest accuracies at 100% specifically at the fourth and fifth musical octaves (24 classes of notes per instrument) for six musical instruments (piano, viola, violin, trumpet, and clarinet) generated using fixed-point arithmetic (hardware model). However, their performances reduce for musical notes from other octaves indicating the lack of robustness in the pitch estimation algorithms in estimating fundamental frequencies outside the fourth and fifth octaves.

Across intensity levels from -20 dB full-scale (FS) to 0 dBFS for musical signals from the fourth octave of a piano, the performances of the pitch estimation algorithms are similar. The performances improve slightly for saturated musical signals above 0 dBFS, indicating that saturated amplitude levels do not affect the representation of fundamental frequency in a sound signal. However, with an AGC algorithm enabled to vary the gain of the input signals, the performances reduce. This is because the selected AGC algorithm attempts to normalise the energy by enhancing the amplitude levels of high-frequency components while reducing those of other frequencies, which renders the pitch estimation algorithms from incorrectly extracting the fundamental frequencies. A separate pitch algorithm is designed to work specifically with the AGC enabled producing 100% accuracies but it in turn has low accuracies (below 30%) when the AGC is disabled. When noise is introduced, and its levels

are increased from 20 dB signal-to-noise-ratio (SNR) to -20 dB SNR, the performances for all pitch estimation algorithms reduce, as noise expectedly introduces more random peaks and obscures the prominent peaks in the ACs pertaining to fundamental frequencies.

In chapter 7, a description of the classification of musical instruments is presented using the responses of the CAR-Lite-A1 model. Summary 2D profiles are used instead of the 4D response generated directly from the model to conserve classifier runtime memory and processing time. A timbre distance (TD) algorithm is designed to be part of the classifier algorithm, which is implementable on an FPGA. A classification accuracy comparison between floating-point (software) and fixed-point (hardware) implementations are made. It is found that the highest accuracy for the fixed-point (hardware) implementation is 92% using a k-nearest neighbour (KNN) classifier with the TD algorithm for 12 classes of musical instruments.

Across intensity levels from -20 dBFS to 20 dBFS, the accuracies of the hardware-based classification mentioned above do not vary significantly. They remain approximately at 60% for three of the 12 classes of musical instruments. Hence, the saturated musical signals above 0 dBFS do not significantly affect timbre. However, the timbre of the signals changes when the AGC algorithm is enabled because timbral cues are dependent on time and frequency information, which are altered by the AGC. However, these changes occur uniformly for all the signals from the three classes of musical instruments, so they are still distinguishable when compared to one another. In this case, not only is the accuracy maintained, but it increases approximately to 80% across the intensity levels suggesting that the normalisation of time and frequency information provided by the AGC enhances unique timbral cues. When noise is added from 20 dB SNR to -20 dB SNR, the performances expectedly reduce due to the introduction of random amplitudes that obscure timbral cues.

For the classification exercises of both musical notes and instruments, it is found that the fixed-point generated responses used significantly less storage than floating-point generated responses for the two models, i.e. AC: 24 times less; 2D summary A1 profiles: 2 times less. This attribute is crucial for real-world applications, which rely on algorithms with low computational resources and limited power supply.

Overall, the results of the auditory hardware-based models presented in this dissertation, which are designed with basic operations derived from more sophisticated auditory software models are capable of capturing primary auditory responses. However, these results do not mean that an auditory hardware-based model is superior to its software-based counterpart. Instead, they should pave the way for further tests and improvements in robustness. In other words, although the hardware-based models in this dissertation show promising results, they have to be tested rigorously with a diverse range of input signals including complex real-world input signals in addition to those presented in this dissertation.

8.2. Future Work

This scholarly journey has allowed me to explore work from a diverse range of disciplines. My focus on musical signals stems from a perspective of learning music intuitively through machine hearing models as a complementary learning platform. As such, the work in this dissertation is only the start of my journey of learning and improving the presented models. Future funding will undoubtedly aid in the development of these hardware-based models and will provide opportunities to test their robustness with complex

signals such as polyphonic musical signals. As the characteristics of these models evolve through future research and development, these models can be applied to other fields beyond musical signal processing, such as speech and environmental sound signal processing, mechanical vibration analysis, multimedia and communications technology and even biomedical applications.

For any aspiring enthusiast or scholar, I present the following areas in my work presented in this dissertation that can be improved:

- 1) Design a feedback pathway to the multi-sampling rate CAR-Lite cochlear model to adjust the poles and zeros of the asymmetric resonator in the model as an input signal is streamed. This feedback represents the effect of the outer hair cell (OHC) response, which influences the BM response. Hence, for loud sound levels, this feedback model reduces BM response. One such feedback model is the fast-acting compression (FAC) used in the CAR-FAC model [1].
- 2) Randomise the firing thresholds of the leaky-integrate-and fire (LIF) neurons in the CAR-Lite-SI model instead of fixing them. An alternative is to implement spike rate variation through adaptive threshold adjustment at the auditory nerve stage of the CAR-Lite-SI cochlear model in place of fixed spike firing rates. This adjustment leads to a sparser spike representation that adheres to biology as well as possibly reduces power consumption on hardware during runtime.
- 3) Investigate the performance of sound identification tasks with and without the added binary spike-based noise in the spontaneous rate fibres of the auditory nerve stage of the CAR-Lite-SI model described in section 3.3 in chapter 3.
- 4) Classification of musical notes using the spike-based responses of the CAR-Lite-ACF models calculated from logical-AND-accumulate (AAC) operations using spiking neural networks [2].
- 5) Implement multiple fundamental frequencies, f_0 , estimation suitable for polyphonic musical signals [3] instead of the single f_0 estimation used in this dissertation for monophonic musical signals.
- 6) Implement phase angle calculations for the temporal modulation (rate) filter and spectral modulation (scale) filter that are hardware implementable. These phase angles can be used for calculating neuronal directionality of varying degrees rather than the binary neuronal directionalities used in the CAR-Lite-A1 model presented in this dissertation.
- 7) Implement a spike-based model of the CAR-Lite-A1 adhering to known spiking attributes of the mammalian primary auditory cortex (A1) [4].
- 8) Compare the performance of the CAR-Lite-ACF models with a biologically plausible model such as Jones's neuromorphic pitch extraction system [5] (derived from Meddis's virtual pitch physiological model presented in chapter 2 [6]) using f_0 estimation from monophonic musical signals.
- 9) Extract harmonic and inharmonic pitch information from the autocorrelogram (AC) for musical pitch classification instead of merely using f_0 estimation algorithms on them.
- 10) Reduce the number of parameters of the coupled-form asymmetric resonator in the rate and scale filterbanks in the CAR-Lite-A1 model by removing W_0 , W_1 , h , and g , and tune a and c to the respective centre velocities and centre densities.

- 11) Design filter stability by picking the closest pole location inside the unit circle instead of only checking for filter stability.
- 12) Use a backward and forward smoothing filter as an alternative for the scale filter such as the one implemented in the AGC filter [7] in the CAR-FAC model.
- 13) Use a lower sampling rate at 6 or 12 kHz for generating the autocorrelogram using the CAR-Lite-ACF model.
- 14) Use strobed temporal integration [8] to generate autocorrelogram, as demonstrated by Lyon [9] at higher fidelity than the one mentioned in this dissertation.
- 15) Use a coupled-form filter as the high-pass filters (HPF) in the quadrature mirror Hilbert transformer (QMHT) in the CAR-Lite-A1 model if the tuned filter variables are capable of generating a 90° phase shift.
- 16) Use the entire 4D responses from the CAR-Lite-A1 model for musical timbre classification instead of their summary profiles used in this dissertation.
- 17) Use classifiers such as support vector machine [10], extreme learning machine (ELM) [11] and deep convolution neural network [12] for the classifications of musical notes and instruments.
- 18) Use running autocorrelation function [9] to yield a more robust and simplified fundamental frequency estimation algorithm than the one presented in chapter 6.

8.3. Publications

R. K. Singh, Y. Xu, R. Wang, T. J. Hamilton, S. L. Denham, and A. van Schaik, "CAR-Lite: A Multi-Rate Cochlear Model on FPGA for Spike-based Sound Encoding," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 66, no. 5, pp. 1805–1817, 2019.

R. K. Singh, Y. Xu, R. Wang, T. J. Hamilton, S. L. Denham, and A. van Schaik, "CAR-Lite: A Multi-Rate Cochlea Model on FPGA," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

Y. Xu, S. Afshar, R. K. Singh, R. Wang, A. Van Schaik, and T. J. Hamilton, "A Binaural Sound Localization System using Deep Convolutional Neural Networks," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 2–6.

Y. Xu, C. S. Thakur, R. K. Singh, T. J. Hamilton, R. Wang, and A. van Schaik, "A FPGA Implementation of the CAR-FAC Cochlear Model," *Front. Neurosci.*, vol. 12, no. April, pp. 1–14, 2018.

Y. Xu, S. Afshar, R. K. Singh, R. Wang, T. J. Hamilton, and A. van Schaik, "A Machine Hearing System for Binaural Sound Localization Based on Instantaneous Correlation," in *IEEE International Symposium on Circuits and Systems*, 2018.

Y. Xu, C. S. Thakur, R. K. Singh, R. Wang, J. Tapson, and A. van Schaik, "Electronic Cochlea: CAR-FAC Model on FPGA," in *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2016, pp. 564–567.

8.4. Bibliography

- [1] R. F. Lyon, "The CARFAC Digital Cochlear Model," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 293–298.

- [2] E. Cerezuela-Escudero, A. Jimenez-Fernandez, R. Paz-Vicente, M. Dominguez-Morales, A. Linares-Barranco, and G. Jimenez-Moreno, "Musical Notes Classification with Neuromorphic Auditory System Using FPGA and a Convolutional Spiking Network," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–7, doi: 10.1109/IJCNN.2015.7280619.
- [3] A. Klapuri, "Auditory Model-Based Methods for Multiple Fundamental Frequency Estimation," in *Signal Processing Methods for Music Transcription*, A. Klapuri and M. Davy, Eds. New York, USA: Springer US, 2006, pp. 229–265.
- [4] M. Elhilali, J. B. Fritz, D. J. Klein, J. Z. Simon, and S. A. Shamma, "Dynamics of Precise Spike Timing in Primary Auditory Cortex," *J. Neurosci.*, vol. 24, no. 5, pp. 1159–1172, 2004, doi: 10.1523/JNEUROSCI.3825-03.2004.
- [5] S. Jones, R. Meddis, S. C. Lim, and A. R. Temple, "Toward a Digital Neuromorphic Pitch Extraction System," *IEEE Trans. Neural Networks*, vol. 11, no. 4, pp. 978–987, 2000, doi: 10.1109/72.857777.
- [6] R. Meddis and L. P. O'Mard, "Virtual Pitch in a Computational Physiological Model," *J. Acoust. Soc. Am.*, vol. 120, no. 6, pp. 3861–3869, 2006, doi: 10.1121/1.2372595.
- [7] R. F. Lyon, "The AGC Loop Filter," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 331–344.
- [8] R. D. Patterson and J. Holdsworth, "A functional model of neural activity patterns and auditory images," *Adv. Speech, Hear. Lang. Process.*, vol. 3, pp. 547–563, 1996.
- [9] R. F. Lyon, "The Auditory Image," in *Human and Machine Hearing: Extracting Meaning from Sound*, Cambridge University Press, 2017, pp. 355–378.
- [10] K. Patil, D. Pressnitzer, S. Shamma, and M. Elhilali, "Music in Our Ears: The Biological Bases of Musical Timbre Perception," *PLoS Comput. Biol.*, vol. 8, no. 11, pp. 1–16, 2012, doi: 10.1371/journal.pcbi.1002759.
- [11] M. D. McDonnell, M. D. Tissera, T. Vladusich, A. Van Schaik, J. Tapson, and F. Schwenker, "Fast, Simple and Accurate Handwritten Digit Classification by Training Shallow Neural Network Classifiers with the 'Extreme Learning Machine' Algorithm," *PLoS One*, vol. 10, no. 8, pp. 1–20, 2015, doi: 10.1371/journal.pone.0134254.
- [12] Y. Xu, S. Afshar, R. K. Singh, R. Wang, A. Van Schaik, and T. J. Hamilton, "A Binaural Sound Localization System using Deep Convolutional Neural Networks," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 2–6, doi: 10.1109/ISCAS.2019.8702345.

A. Automatic Gain Control (AGC)

An automatic gain control (AGC) algorithm is used for smoothly varying the gain of a sound signal. The advantage of the AGC algorithm described below is that it corrects the amplitudes of a sound signal, whereby a low-intensity sound signal is amplified, and a high-intensity sound signal is reduced. In this case, the AGC accepts only a floating-point sound signal. The modified sound signal is then converted to a fixed-point format before it is input to the CAR-Lite model.

The AGC algorithm selected for use in this dissertation is designed by Ellis [1], which emulates the functionality of the outer hair cells in a cochlea to control the mechanical motions of the basilar membrane (BM). It does this by computing the short-time Fourier transform (STFT) of a sound signal and smooths that energy using time and frequency scales before normalising the scaled energy.

The algorithm starts by taking the STFT of the input signal, x :

$$X_m(f) = \sum_{n=1}^N x(n) \cdot g(n - mR) \cdot e^{-j2\pi f n} \quad (\text{A-1})$$

where n is the sample number; N is the size of the input signal; $X_m(f)$ is the discrete Fourier transform of a window of the input signal centred at mR and at a specific frequency, f ; m is the sliding analysis window number; $g(n)$ is the analysis window function of size M – here M is set at a default of 4,096; and R is the hop size set at a default of 2,048.

The next step is to smoothen the STFT response by multiplying it with weights, w :

$$A(f) = w \cdot |X_m(f)| \quad (\text{A-2})$$

where $A(f)$ is the smoothened response of the STFT; w is a weighted value calculated using the proximity of a nonlinear frequency, $f_{Mel}(n)$, to a linear centre frequency, f_{FFT} , within a frequency bin:

$$w = \max(0, \min(l, h)) \quad (\text{A-3})$$

where l is the proximity of $f_{Mel}(n)$ to f_{FFT} , where $f_{Mel}(n) < f_{FFT}$ and $n = 1$ for a group of 3 f_{Mel} ; h is the proximity of f_{FFT} to $f_{Mel}(n)$, where $f_{Mel}(n) > f_{FFT}$, and $n = 3$. They are calculated as:

$$l = \frac{f_{FFT} - f_{Mel}(1)}{f_{Mel}(2) - f_{Mel}(1)} \quad (\text{A-4})$$

$$h = \frac{f_{Mel}(3) - f_{FFT}}{f_{Mel}(3) - f_{Mel}(2)} \quad (\text{A-5})$$

Here, f_{FFT} is a vector comprising 20 linearly scaled frequencies calculated with fast Fourier transform; the FFT bins, f_{Mel} is a nonlinear frequency vector computed with a logarithmic transformation of f_{FFT} [2]. f_{Mel} is known as a melody scale, where the perceived pitch is distributed linearly [3]. For every element of f_{FFT} , groups of 3 f_{Mel} elements are iteratively extracted to calculate l and h . At the end of the iteration, weights are calculated as per equation (A-3).

The next part is to normalise X_m with a factor, E . The first of the two steps of acquiring E is to calculate the smooth attack and decay of the signal, which requires the calculation of its weights as:

$$a_i = \max_{i \in f_{sc}} \left(a_i \cdot e^{-R/(f_s \cdot t_{sc})}, A(i) \right) \quad (\text{A-6})$$

where f_s is the sampling rate of the input signal set at 96,000 Hz; t_{sc} is the time scale set at 0.5. The second step involves the calculation of E :

$$E = \text{diag} \left(\frac{1}{s + s'} \cdot w^T \cdot a \right) \quad (\text{A-7})$$

where diag represents matrix diagonal; s is the weight summed across the frequency scales and s' is the binary activation of s :

$$s = \sum_{i=1}^{f_{sc}} w \quad (\text{A-8})$$

$$s' = \begin{cases} 1 & \text{if } s_{ij} = 0 \\ 0 & \text{if } s_{ij} \neq 0 \end{cases} \quad (\text{A-9})$$

After normalisation, the signal, X_m , is transformed from the Fourier domain to the time domain with an inverse STFT involving the calculation of an inverse fast Fourier transform of a discrete Fourier transform of the signal:

$$x'(n) = \text{Re} \left(\sum_{m=1}^M \int_{-1/2}^{1/2} \frac{X_m(f)}{E(f)} \cdot e^{j2\pi f n} df \right) \quad (\text{A-10})$$

where x' is the real-valued input signal processed with the AGC algorithm; and M is the total number of sliding analysis windows. To ensure, the signal remains between an amplitude of -1 to 1, the signal is normalised as follows:

$$x'' = \frac{x'}{\max(|x'|)} \quad (\text{A-11})$$

Figure A-1 illustrates the effect of the AGC algorithm with a 20 dBFS speech signal [1]. Figure A-1(b) displays the output of the AGC algorithm in floating-point, while Figure A-1(c) displays the same output represented in 16 bits fixed-point. The AGC reduces the amplitudes of the input signal to be smaller than the original signal with the floating-point signal bounded within ± 1 , and the fixed-point signal bounded within +32767 and -32768. In other words, the AGC corrects the amplitude levels of an input signal to be bounded within a specific range, which is beneficial for maintaining a fixed bit width regardless of changes in the intensity levels of the signal.

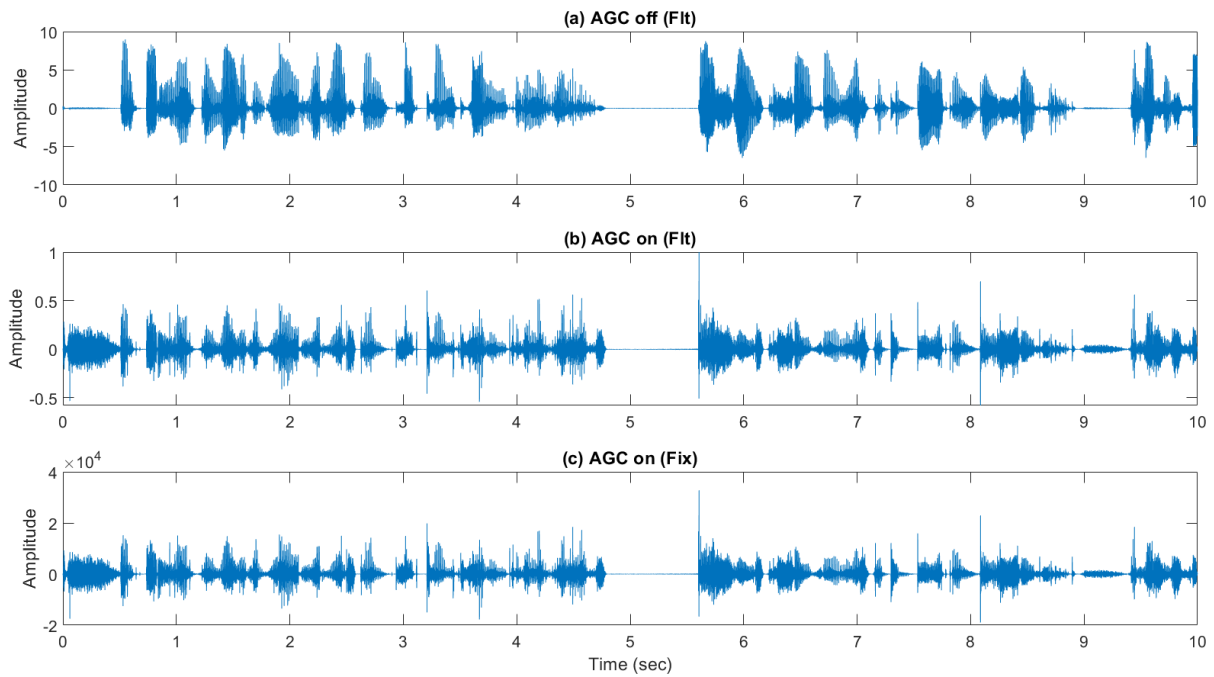


Figure A-1: (a) 20 dBFS floating-point representation of a speech signal [1] input to the AGC algorithm. (b) The output of the AGC (b) in floating-point and (c) in 16 bits fixed-point.

Bibliography

- [1] D. P. W. Ellis, "tf_agc - Time-frequency automatic gain control," 2010. https://labrosa.ee.columbia.edu/matlab/tf_agc/ (accessed May 10, 2020).
- [2] M. Slaney, "Auditory Toolbox Version 2," 1998. [Online]. Available: <https://engineering.purdue.edu/~malcolm/interval/1998-010/AuditoryToolboxTechReport.pdf>.
- [3] S. S. Stevens, J. Volkman, and E. B. Newman, "A Scale for the Measurement of the Psychological Magnitude Pitch," *J. Acoust. Soc. Am.*, vol. 8, no. 3, pp. 185–190, 1937, doi: 10.1121/1.1915893.

B. Classification of Musical Notes (0 dBFS, No Noise)

Index	Instrument	Note Range	Computation	Algorithm					
				1	2	3	4	5	6
1	Piano	C2-B2	Fixed-Point AAC	0%	0%	0%	0%	0%	0%
			Fixed-Point MAC	0%	0%	0%	0%	0%	0%
			Floating-Point AAC	0%	0%	0%	0%	0%	0%
			Floating-Point MAC	0%	0%	0%	0%	0%	0%
		C3-B3	Fixed-Point AAC	8%	8%	58%	58%	33%	58%
			Fixed-Point MAC	33%	8%	33%	42%	0%	33%
			Floating-Point AAC	8%	8%	58%	58%	42%	58%
			Floating-Point MAC	33%	0%	25%	33%	0%	25%
2	Classical Guitar	C3-B3	Fixed-Point AAC	8%	25%	25%	8%	33%	8%
			Fixed-Point MAC	75%	25%	25%	25%	8%	25%
			Floating-Point AAC	8%	17%	17%	8%	33%	8%
			Floating-Point MAC	67%	8%	8%	8%	25%	8%
3	Electric Guitar	C3-B3	Fixed-Point AAC	0%	0%	0%	0%	0%	0%
			Fixed-Point MAC	42%	25%	25%	33%	8%	25%
			Floating-Point AAC	0%	0%	8%	0%	0%	0%
			Floating-Point MAC	67%	8%	17%	17%	25%	17%
4	Viola	C3-B3	Fixed-Point AAC	17%	0%	75%	75%	25%	75%
			Fixed-Point MAC	42%	0%	75%	75%	0%	75%
			Floating-Point AAC	17%	0%	75%	75%	25%	75%
			Floating-Point MAC	42%	0%	67%	67%	17%	67%
5	Cello	C3-B3	Fixed-Point AAC	0%	8%	42%	42%	17%	42%
			Fixed-Point MAC	8%	8%	58%	58%	0%	58%
			Floating-Point AAC	0%	8%	42%	42%	8%	42%
			Floating-Point MAC	8%	0%	50%	50%	0%	50%
6	Trombone	C3-B3	Fixed-Point AAC	17%	8%	83%	83%	50%	83%
			Fixed-Point MAC	17%	0%	92%	92%	25%	92%
			Floating-Point AAC	17%	8%	92%	92%	58%	92%
			Floating-Point MAC	17%	0%	83%	83%	42%	83%
7	Tenor Saxophone	C3-B3	Fixed-Point AAC	67%	0%	67%	58%	58%	67%
			Fixed-Point MAC	75%	25%	83%	75%	67%	75%
			Floating-Point AAC	58%	0%	67%	50%	58%	67%
			Floating-Point MAC	83%	33%	75%	67%	67%	75%
8	Baritone Saxophone	C2-B2	Fixed-Point AAC	0%	0%	25%	25%	0%	25%
			Fixed-Point MAC	0%	0%	33%	33%	0%	33%
			Floating-Point AAC	0%	0%	25%	25%	0%	25%
			Floating-Point MAC	0%	0%	33%	33%	0%	33%
9	Electric Bass	C2-B2	Fixed-Point AAC	0%	0%	8%	8%	0%	8%
			Fixed-Point MAC	17%	0%	0%	8%	0%	0%
			Floating-Point AAC	0%	0%	0%	0%	0%	0%
			Floating-Point MAC	8%	0%	8%	8%	0%	8%
10	Contrabass	C2-B2	Fixed-Point AAC	0%	0%	0%	0%	0%	0%

			Fixed-Point MAC	0%	0%	0%	0%	0%	0%
			Floating-Point AAC	0%	0%	0%	0%	0%	0%
			Floating-Point MAC	0%	0%	0%	0%	0%	0%

Table B-1: Accuracy scores for classifying musical notes from various musical instruments having musical notes under low-range octave groups (octaves 2 and 3). The numbers denoted after the musical notes (under 'Note Range') are the octave numbers.

Index	Instrument	Note Range	Computation	Algorithm					
				1	2	3	4	5	6
1	Piano	C4-B4	Fixed-Point AAC	50%	17%	100%	58%	92%	100%
			Fixed-Point MAC	75%	42%	92%	50%	92%	83%
			Floating-Point AAC	42%	8%	100%	58%	92%	100%
			Floating-Point MAC	75%	50%	83%	50%	92%	75%
2	Piano	C5-B5	Fixed-Point AAC	42%	42%	50%	33%	92%	50%
			Fixed-Point MAC	67%	8%	67%	33%	100%	67%
			Floating-Point AAC	42%	33%	50%	33%	92%	50%
			Floating-Point MAC	58%	25%	67%	33%	100%	67%
3	Vibraphone	C5-B5	Fixed-Point AAC	33%	17%	83%	75%	67%	83%
			Fixed-Point MAC	42%	0%	92%	83%	83%	92%
			Floating-Point AAC	25%	0%	83%	75%	67%	83%
			Floating-Point MAC	33%	25%	92%	83%	75%	92%
4	Accordion	C5-B5	Fixed-Point AAC	83%	0%	92%	83%	58%	92%
			Fixed-Point MAC	92%	0%	92%	92%	58%	92%
			Floating-Point AAC	83%	8%	92%	83%	58%	92%
			Floating-Point MAC	83%	17%	92%	92%	58%	92%
5	Violin	C5-B5	Fixed-Point AAC	67%	33%	100%	100%	100%	100%
			Fixed-Point MAC	67%	33%	100%	100%	100%	100%
			Floating-Point AAC	67%	42%	100%	100%	100%	100%
			Floating-Point MAC	75%	42%	100%	100%	100%	100%
6	Viola	C5-B5	Fixed-Point AAC	75%	25%	100%	100%	100%	100%
			Fixed-Point MAC	75%	17%	100%	100%	92%	100%
			Floating-Point AAC	58%	25%	100%	100%	100%	100%
			Floating-Point MAC	67%	25%	100%	100%	92%	100%
7	Cello	C4-B4	Fixed-Point AAC	58%	17%	83%	75%	83%	83%
			Fixed-Point MAC	58%	8%	83%	58%	75%	83%
			Floating-Point AAC	75%	17%	83%	75%	83%	83%
			Floating-Point MAC	67%	25%	50%	42%	58%	50%
8	Trumpet	C5-B5	Fixed-Point AAC	83%	67%	100%	100%	100%	100%
			Fixed-Point MAC	83%	50%	100%	100%	100%	100%
			Floating-Point AAC	75%	58%	100%	100%	100%	100%
			Floating-Point MAC	92%	83%	100%	100%	100%	100%
9	Soprano Sax	C5-B5	Fixed-Point AAC	17%	25%	92%	42%	83%	92%
			Fixed-Point MAC	33%	25%	83%	75%	83%	83%
			Floating-Point AAC	17%	17%	92%	42%	83%	92%
			Floating-Point MAC	33%	17%	83%	83%	83%	83%

10	Oboe	C5-B5	Fixed-Point AAC	83%	67%	92%	100%	83%	92%
			Fixed-Point MAC	83%	50%	83%	83%	50%	83%
			Floating-Point AAC	83%	67%	100%	100%	100%	100%
			Floating-Point MAC	83%	67%	83%	83%	50%	83%
11	Clarinet	C5-B5	Fixed-Point AAC	58%	42%	100%	92%	92%	100%
			Fixed-Point MAC	67%	75%	100%	100%	92%	100%
			Floating-Point AAC	58%	42%	100%	92%	92%	100%
			Floating-Point MAC	75%	58%	100%	100%	92%	100%

Table B-2: Accuracy scores for classifying musical notes from various musical instruments having musical notes under middle-range octave groups (octaves 4 and 5). The numbers denoted after the musical notes (under 'Note Range') are the octave numbers.

Index	Instrument	Note Range	Computation	Algorithm					
				1	2	3	4	5	6
1	Piano	C6-B6	Fixed-Point AAC	58%	33%	83%	75%	92%	83%
			Fixed-Point MAC	58%	17%	92%	75%	92%	92%
			Floating-Point AAC	58%	33%	83%	75%	92%	83%
			Floating-Point MAC	58%	8%	92%	75%	92%	92%
		C7-B7	Fixed-Point AAC	25%	17%	17%	17%	33%	17%
			Fixed-Point MAC	8%	17%	25%	17%	42%	25%
			Floating-Point AAC	25%	0%	17%	17%	25%	17%
			Floating-Point MAC	17%	0%	25%	8%	58%	25%
2	Marimba	C6-B6	Fixed-Point AAC	67%	25%	92%	75%	83%	92%
			Fixed-Point MAC	58%	17%	92%	75%	92%	92%
			Floating-Point AAC	67%	17%	92%	75%	83%	92%
			Floating-Point MAC	58%	25%	92%	83%	92%	92%
3	Flute	C6-B6	Fixed-Point AAC	67%	33%	75%	58%	83%	75%
			Fixed-Point MAC	67%	33%	83%	83%	83%	83%
			Floating-Point AAC	67%	42%	75%	58%	83%	75%
			Floating-Point MAC	67%	8%	83%	8%	83%	83%

Table B-3: Accuracy scores for classifying musical notes from various musical instruments having musical notes under high-range octave groups (octaves 6 and 7). The numbers denoted after the musical notes (under 'Note Range') are the octave numbers.

C. Classification of Musical Notes based on Varying Intensity and SNR Levels

Accuracy: MAC, No Noise, Fix, AGC off						Accuracy: MAC, No Noise, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	75%	75%	75%	67%	92%	1	0%	0%	0%	0%	0%
2	42%	42%	42%	17%	17%	2	0%	0%	0%	0%	0%
3	92%	92%	92%	83%	83%	3	67%	67%	67%	67%	67%
4	50%	50%	50%	33%	50%	4	67%	67%	67%	67%	67%
5	92%	92%	92%	100%	92%	5	25%	25%	25%	25%	25%
6	83%	83%	83%	83%	75%	6	67%	67%	67%	67%	67%
7	8%	8%	8%	58%	67%	7	100%	100%	100%	100%	100%
Accuracy: AAC, No Noise, Flt, AGC off						Accuracy: AAC, No Noise, Flt, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	33%	33%	33%	58%	75%	1	0%	0%	0%	0%	0%
2	17%	17%	17%	8%	17%	2	0%	0%	0%	0%	0%
3	92%	92%	92%	75%	92%	3	42%	42%	42%	42%	42%
4	58%	58%	58%	25%	75%	4	50%	50%	50%	50%	50%
5	83%	83%	83%	100%	100%	5	25%	25%	25%	25%	25%
6	83%	83%	83%	75%	92%	6	42%	42%	42%	42%	42%
7	0%	0%	0%	50%	75%	7	100%	100%	100%	100%	100%
Accuracy: AAC, No Noise, Fix, AGC off						Accuracy: AAC, No Noise, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	33%	42%	33%	67%	75%	1	0%	0%	0%	0%	0%
2	17%	8%	8%	8%	17%	2	0%	0%	0%	0%	0%
3	92%	100%	100%	75%	92%	3	42%	42%	42%	42%	42%
4	67%	75%	67%	25%	75%	4	50%	50%	50%	50%	50%
5	83%	92%	92%	100%	100%	5	17%	17%	17%	17%	17%
6	92%	100%	92%	75%	92%	6	50%	50%	50%	50%	50%
7	0%	0%	0%	58%	75%	7	100%	100%	100%	100%	100%

Table C-1: Pitch estimation accuracy results for 12 musical notes (A4 – G#4) without white Gaussian noise.

Accuracy (Recording 1): MAC, 20 dB SNR, Flt, AGC off						Accuracy (Recording 1): AAC, 20 dB SNR, Flt, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	67%	75%	75%	50%	58%	1	42%	42%	17%	50%	75%
2	8%	33%	42%	17%	8%	2	8%	8%	8%	0%	17%
3	83%	83%	67%	83%	92%	3	100%	100%	92%	75%	92%
4	58%	50%	33%	25%	50%	4	50%	50%	75%	33%	75%
5	92%	83%	83%	100%	92%	5	92%	92%	92%	100%	100%
6	83%	83%	67%	83%	83%	6	100%	92%	83%	75%	92%

7	17%	17%	17%	58%	58%	7	0%	8%	8%	50%	75%
Accuracy (Recording 1): MAC, 20 dB SNR, Fix, AGC off						Accuracy (Recording 1): AAC, 20 dB SNR, Fix, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	58%	67%	67%	75%	75%	1	42%	25%	33%	42%	67%
2	8%	33%	8%	25%	17%	2	8%	0%	8%	17%	42%
3	92%	92%	83%	83%	83%	3	100%	92%	92%	75%	92%
4	75%	50%	67%	25%	58%	4	58%	75%	67%	33%	83%
5	92%	92%	92%	100%	92%	5	92%	92%	83%	100%	100%
6	92%	75%	83%	83%	75%	6	100%	92%	92%	75%	92%
7	8%	8%	8%	58%	58%	7	0%	0%	0%	42%	75%
Accuracy (Recording 2): MAC, 20 dB SNR, Flt, AGC off						Accuracy (Recording 2): AAC, 20 dB SNR, Flt, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	67%	67%	75%	50%	83%	1	58%	25%	50%	58%	67%
2	17%	25%	8%	0%	17%	2	8%	8%	8%	8%	17%
3	83%	83%	75%	83%	83%	3	92%	92%	100%	75%	92%
4	67%	58%	58%	42%	50%	4	67%	58%	83%	25%	75%
5	92%	92%	92%	100%	92%	5	83%	83%	92%	100%	100%
6	83%	83%	75%	83%	75%	6	92%	92%	100%	75%	92%
7	8%	8%	17%	58%	58%	7	0%	0%	8%	50%	75%
Accuracy (Recording 2): MAC, 20 dB SNR, Fix, AGC off						Accuracy (Recording 2): AAC, 20 dB SNR, Fix, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	67%	75%	67%	67%	83%	1	33%	50%	58%	33%	50%
2	33%	42%	42%	17%	17%	2	8%	8%	0%	8%	17%
3	92%	92%	83%	83%	83%	3	92%	92%	100%	75%	92%
4	50%	42%	42%	33%	50%	4	67%	67%	83%	42%	75%
5	92%	92%	92%	100%	92%	5	83%	83%	92%	92%	100%
6	83%	92%	75%	83%	75%	6	92%	83%	100%	67%	92%
7	8%	8%	17%	67%	67%	7	0%	0%	8%	50%	75%
Accuracy (Recording 3): MAC, 20 dB SNR, Flt, AGC off						Accuracy (Recording 3): AAC, 20 dB SNR, Flt, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	67%	67%	75%	58%	75%	1	33%	33%	25%	42%	75%
2	17%	17%	17%	25%	25%	2	8%	8%	8%	17%	17%
3	83%	92%	75%	83%	83%	3	100%	92%	100%	83%	92%
4	67%	67%	50%	33%	58%	4	75%	58%	58%	33%	75%
5	92%	92%	92%	100%	92%	5	92%	83%	92%	100%	100%
6	83%	92%	75%	83%	75%	6	100%	92%	92%	83%	92%
7	8%	8%	17%	58%	83%	7	0%	0%	0%	58%	75%
Accuracy (Recording 3): MAC, 20 dB SNR, Fix, AGC off						Accuracy (Recording 3): AAC, 20 dB SNR, Fix, AGC off					

AGC off						off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	75%	75%	75%	67%	67%	1	25%	33%	42%	42%	67%
2	33%	8%	17%	25%	33%	2	8%	8%	8%	8%	8%
3	92%	83%	83%	83%	83%	3	92%	92%	100%	75%	92%
4	42%	58%	67%	33%	50%	4	75%	67%	75%	33%	75%
5	92%	92%	92%	100%	92%	5	83%	83%	92%	100%	100%
6	83%	83%	83%	83%	75%	6	92%	83%	100%	75%	92%
7	8%	8%	8%	50%	50%	7	0%	8%	0%	50%	75%

Table C-2: Pitch estimation accuracy results for 12 musical notes (A4 – G#4) with 20 dB SNR (white Gaussian noise) and AGC disabled.

Accuracy (Recording 1): MAC, 20 dB SNR, Flt, AGC on						Accuracy (Recording 1): AAC, 20 dB SNR, Flt, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	67%	67%	67%	67%	67%	3	50%	50%	42%	50%	50%
4	67%	67%	67%	67%	67%	4	50%	58%	58%	58%	58%
5	17%	25%	25%	17%	25%	5	17%	17%	17%	8%	8%
6	67%	67%	67%	67%	67%	6	42%	50%	50%	42%	50%
7	100%	100%	100%	100%	100%	7	100%	100%	100%	100%	100%
Accuracy (Recording 1): MAC, 20 dB SNR, Fix, AGC on						Accuracy (Recording 1): AAC, 20 dB SNR, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	67%	67%	67%	67%	67%	3	50%	50%	50%	58%	42%
4	67%	67%	67%	67%	67%	4	58%	58%	58%	58%	50%
5	25%	25%	25%	25%	25%	5	17%	8%	17%	8%	17%
6	67%	67%	67%	67%	67%	6	58%	58%	58%	58%	50%
7	100%	100%	100%	100%	100%	7	100%	100%	100%	100%	100%
Accuracy (Recording 2): MAC, 20 dB SNR, Flt, AGC on						Accuracy (Recording 2): AAC, 20 dB SNR, Flt, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	67%	67%	67%	67%	67%	3	42%	50%	50%	50%	42%
4	67%	67%	67%	67%	67%	4	58%	50%	50%	58%	50%
5	25%	17%	25%	17%	17%	5	17%	8%	8%	17%	17%
6	67%	67%	67%	67%	67%	6	42%	50%	42%	50%	42%
7	100%	100%	100%	100%	100%	7	100%	100%	100%	100%	100%

Accuracy (Recording 2): MAC, 20 dB SNR, Fix, AGC on						Accuracy (Recording 2): AAC, 20 dB SNR, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	8%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	67%	67%	67%	67%	67%	3	42%	42%	42%	50%	50%
4	67%	67%	58%	67%	67%	4	50%	50%	50%	58%	58%
5	17%	17%	17%	25%	17%	5	17%	8%	17%	8%	8%
6	67%	67%	67%	67%	67%	6	50%	50%	50%	58%	58%
7	100%	100%	100%	100%	100%	7	100%	100%	100%	100%	100%
Accuracy (Recording 3): MAC, 20 dB SNR, Fit, AGC on						Accuracy (Recording 3): AAC, 20 dB SNR, Fit, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	67%	67%	67%	67%	67%	3	42%	50%	42%	42%	50%
4	67%	67%	67%	67%	67%	4	42%	33%	42%	50%	42%
5	25%	17%	25%	17%	17%	5	17%	8%	8%	8%	8%
6	67%	67%	67%	67%	67%	6	42%	42%	42%	42%	50%
7	100%	100%	100%	100%	100%	7	100%	100%	100%	100%	100%
Accuracy (Recording 3): MAC, 20 dB SNR, Fix, AGC on						Accuracy (Recording 3): AAC, 20 dB SNR, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	67%	67%	58%	67%	67%	3	42%	42%	42%	50%	42%
4	67%	58%	67%	67%	67%	4	50%	50%	50%	58%	50%
5	17%	33%	25%	25%	25%	5	8%	8%	17%	8%	8%
6	67%	67%	67%	67%	67%	6	50%	50%	50%	58%	50%
7	100%	100%	100%	100%	100%	7	100%	100%	100%	100%	100%

Table C-3: Pitch estimation accuracy results for 12 musical notes (A4 – G#4) with 20 dB SNR (white Gaussian noise) and AGC enabled.

Accuracy (Recording 1): MAC, 0 dB SNR, Fit, AGC off						Accuracy (Recording 1): AAC, 0 dB SNR, Fit, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	8%	1	0%	8%	0%	0%	8%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	42%	50%	42%	50%	83%	3	58%	50%	58%	42%	75%
4	17%	25%	25%	42%	67%	4	50%	33%	42%	25%	50%
5	50%	50%	50%	67%	75%	5	58%	42%	50%	67%	92%
6	42%	42%	42%	50%	83%	6	58%	42%	50%	42%	75%
7	25%	0%	25%	58%	58%	7	0%	0%	8%	42%	58%

Accuracy (Recording 1): MAC, 0 dB SNR, Fix, AGC off						Accuracy (Recording 1): AAC, 0 dB SNR, Fix, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	8%	1	0%	0%	8%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	58%	33%	58%	42%	58%	3	50%	75%	67%	67%	83%
4	42%	25%	50%	25%	50%	4	33%	58%	75%	33%	50%
5	50%	42%	33%	50%	75%	5	50%	58%	58%	42%	92%
6	50%	33%	58%	42%	58%	6	42%	67%	75%	50%	83%
7	25%	0%	17%	33%	75%	7	8%	0%	0%	50%	75%
Accuracy (Recording 2): MAC, 0 dB SNR, Flt, AGC off						Accuracy (Recording 2): AAC, 0 dB SNR, Flt, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	8%	8%	0%	0%	1	0%	0%	0%	8%	17%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	58%	58%	75%	42%	58%	3	67%	42%	83%	75%	92%
4	33%	50%	50%	17%	42%	4	50%	17%	83%	42%	58%
5	58%	50%	58%	58%	75%	5	67%	50%	75%	83%	100%
6	58%	58%	67%	33%	58%	6	50%	33%	83%	75%	92%
7	8%	17%	17%	58%	67%	7	8%	8%	17%	58%	67%
Accuracy (Recording 2): MAC, 0 dB SNR, Fix, AGC off						Accuracy (Recording 2): AAC, 0 dB SNR, Fix, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	17%	0%	8%	0%	1	0%	8%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	58%	58%	42%	50%	58%	3	67%	83%	67%	75%	83%
4	50%	58%	42%	42%	42%	4	50%	33%	58%	25%	58%
5	33%	42%	50%	58%	75%	5	58%	67%	33%	75%	92%
6	58%	50%	50%	50%	58%	6	50%	58%	67%	75%	83%
7	25%	0%	8%	42%	50%	7	0%	0%	0%	42%	75%
Accuracy (Recording 3): MAC, 0 dB SNR, Flt, AGC off						Accuracy (Recording 3): AAC, 0 dB SNR, Flt, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	8%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	58%	50%	58%	42%	67%	3	50%	67%	42%	58%	75%
4	33%	58%	33%	42%	42%	4	33%	50%	33%	42%	67%
5	58%	50%	33%	50%	83%	5	42%	50%	33%	92%	92%
6	50%	50%	33%	42%	50%	6	50%	67%	42%	58%	75%
7	8%	17%	8%	50%	67%	7	17%	0%	8%	42%	75%
Accuracy (Recording 3): MAC, 0 dB SNR, Fix, AGC off						Accuracy (Recording 3): AAC, 0 dB SNR, Fix, AGC off					

Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	8%	8%	0%	0%	0%	1	0%	0%	0%	0%	8%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	33%	42%	58%	58%	58%	3	42%	58%	58%	58%	83%
4	33%	25%	42%	50%	42%	4	17%	42%	50%	25%	67%
5	58%	50%	42%	83%	92%	5	58%	67%	75%	83%	100%
6	33%	33%	50%	58%	58%	6	25%	58%	50%	58%	83%
7	17%	8%	0%	33%	75%	7	0%	8%	0%	42%	67%

Table C-4: Pitch estimation accuracy results for 12 musical notes (A4 – G#4) with 0 dB SNR (white Gaussian noise) and AGC disabled.

Accuracy (Recording 1): MAC, 0 dB SNR, Flt, AGC on						Accuracy (Recording 1): AAC, 0 dB SNR, Flt, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	67%	58%	67%	67%	42%	3	58%	50%	58%	58%	58%
4	67%	58%	67%	83%	50%	4	75%	75%	75%	75%	67%
5	25%	25%	25%	33%	17%	5	8%	33%	17%	8%	8%
6	67%	58%	67%	83%	50%	6	42%	50%	42%	42%	42%
7	100%	100%	92%	100%	100%	7	100%	92%	100%	100%	100%
Accuracy (Recording 1): MAC, 0 dB SNR, Fix, AGC on						Accuracy (Recording 1): AAC, 0 dB SNR, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	50%	67%	58%	58%	50%	3	58%	58%	58%	58%	67%
4	50%	67%	67%	58%	58%	4	58%	58%	50%	58%	75%
5	33%	25%	33%	25%	17%	5	8%	8%	42%	8%	8%
6	58%	67%	67%	58%	50%	6	58%	58%	50%	58%	67%
7	100%	100%	100%	92%	100%	7	100%	100%	100%	100%	100%
Accuracy (Recording 2): MAC, 0 dB SNR, Flt, AGC on						Accuracy (Recording 2): AAC, 0 dB SNR, Flt, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	58%	58%	50%	67%	42%	3	50%	50%	67%	58%	58%
4	58%	67%	50%	67%	42%	4	50%	58%	50%	50%	50%
5	8%	25%	25%	25%	25%	5	8%	42%	8%	17%	33%
6	58%	67%	50%	67%	42%	6	50%	50%	42%	50%	42%
7	92%	100%	100%	100%	100%	7	100%	92%	92%	100%	92%
Accuracy (Recording 2): MAC, 0 dB SNR, Fix, AGC on						Accuracy (Recording 2): AAC, 0 dB SNR, Fix, AGC on					

Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	67%	58%	67%	58%	67%	3	58%	58%	67%	50%	67%
4	67%	58%	58%	58%	67%	4	58%	58%	67%	50%	75%
5	25%	8%	33%	17%	42%	5	8%	17%	25%	17%	8%
6	67%	58%	67%	58%	67%	6	58%	58%	67%	50%	67%
7	100%	100%	100%	92%	100%	7	100%	100%	100%	100%	100%
Accuracy (Recording 3): MAC, 0 dB SNR, Flt, AGC on						Accuracy (Recording 3): AAC, 0 dB SNR, Flt, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	67%	50%	58%	67%	58%	3	50%	58%	50%	50%	50%
4	58%	58%	58%	58%	58%	4	50%	58%	58%	50%	50%
5	17%	25%	17%	25%	25%	5	8%	8%	42%	17%	25%
6	58%	50%	58%	58%	58%	6	42%	50%	50%	42%	50%
7	100%	100%	100%	100%	100%	7	100%	100%	100%	92%	92%
Accuracy (Recording 3): MAC, 0 dB SNR, Fix, AGC on						Accuracy (Recording 3): AAC, 0 dB SNR, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	58%	58%	75%	50%	67%	3	58%	67%	58%	50%	67%
4	58%	58%	83%	50%	50%	4	67%	67%	50%	58%	67%
5	25%	25%	42%	8%	25%	5	25%	8%	25%	8%	17%
6	58%	58%	83%	50%	58%	6	67%	67%	50%	58%	67%
7	100%	100%	100%	100%	100%	7	100%	100%	100%	100%	100%

Table C-5: Pitch estimation accuracy results for 12 musical notes (A4 – G#4) with 0 dB SNR (white Gaussian noise) and AGC enabled.

Accuracy (Recording 1): MAC, -20 dB SNR, Flt, AGC off						Accuracy (Recording 1): AAC, -20 dB SNR, Flt, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	0%	33%	8%	17%	0%	3	17%	0%	8%	25%	8%
4	0%	33%	8%	17%	0%	4	8%	8%	17%	8%	8%
5	17%	17%	17%	25%	0%	5	17%	8%	8%	17%	25%
6	0%	33%	8%	17%	0%	6	8%	8%	8%	17%	8%
7	42%	25%	0%	17%	17%	7	8%	17%	33%	17%	42%
Accuracy (Recording 1): MAC, -20 dB SNR, Fix, AGC off						Accuracy (Recording 1): AAC, -20 dB SNR, Fix, AGC off					

Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	17%	8%	25%	0%	8%	3	0%	42%	42%	25%	33%
4	25%	8%	33%	0%	8%	4	0%	42%	33%	25%	33%
5	17%	17%	8%	0%	17%	5	0%	33%	17%	17%	17%
6	25%	8%	33%	0%	8%	6	0%	42%	33%	25%	33%
7	33%	17%	8%	25%	17%	7	8%	42%	25%	33%	42%
Accuracy (Recording 2): MAC, -20 dB SNR, Flt, AGC off						Accuracy (Recording 2): AAC, -20 dB SNR, Flt, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	8%	25%	8%	8%	17%	3	0%	8%	0%	8%	25%
4	0%	25%	0%	0%	17%	4	0%	17%	0%	8%	17%
5	25%	33%	17%	0%	8%	5	0%	17%	17%	8%	17%
6	8%	25%	0%	0%	25%	6	0%	17%	0%	8%	25%
7	33%	33%	17%	17%	25%	7	8%	33%	25%	8%	17%
Accuracy (Recording 2): MAC, -20 dB SNR, Fix, AGC off						Accuracy (Recording 2): AAC, -20 dB SNR, Fix, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	8%	17%	8%	8%	33%	3	17%	0%	17%	8%	25%
4	8%	17%	8%	8%	33%	4	8%	8%	17%	8%	33%
5	8%	8%	17%	8%	17%	5	8%	8%	17%	0%	33%
6	8%	17%	8%	8%	33%	6	8%	8%	17%	8%	25%
7	42%	17%	33%	33%	42%	7	33%	33%	8%	17%	25%
Accuracy (Recording 3): MAC, -20 dB SNR, Flt, AGC off						Accuracy (Recording 3): AAC, -20 dB SNR, Flt, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	25%	8%	0%	25%	17%	3	17%	25%	17%	17%	17%
4	25%	8%	0%	17%	25%	4	25%	17%	17%	33%	8%
5	17%	0%	8%	25%	25%	5	8%	8%	25%	25%	25%
6	25%	8%	0%	17%	25%	6	17%	17%	17%	33%	8%
7	25%	25%	33%	33%	25%	7	17%	25%	17%	33%	33%
Accuracy (Recording 3): MAC, -20 dB SNR, Fix, AGC off						Accuracy (Recording 3): AAC, -20 dB SNR, Fix, AGC off					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					

Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	25%	17%	25%	0%	8%	3	25%	33%	0%	0%	42%
4	25%	33%	25%	0%	8%	4	25%	25%	0%	8%	33%
5	25%	0%	0%	8%	8%	5	17%	25%	8%	8%	25%
6	25%	33%	25%	0%	8%	6	25%	25%	0%	8%	42%
7	25%	33%	17%	33%	17%	7	17%	33%	17%	25%	42%

Table C-6: Pitch estimation accuracy results for 12 musical notes (A4 – G#4) with -20 dB SNR (white Gaussian noise) and AGC disabled.

Accuracy (Recording 1): MAC, -20 dB SNR, Flt, AGC on						Accuracy (Recording 1): AAC, -20 dB SNR, Flt, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	17%	8%	0%	8%	17%	3	8%	17%	25%	25%	17%
4	17%	8%	0%	8%	17%	4	8%	8%	33%	25%	17%
5	17%	8%	0%	0%	8%	5	8%	8%	0%	8%	0%
6	17%	8%	0%	8%	17%	6	8%	17%	25%	25%	8%
7	42%	33%	33%	25%	58%	7	42%	67%	33%	50%	50%
Accuracy (Recording 1): MAC, -20 dB SNR, Fix, AGC on						Accuracy (Recording 1): AAC, -20 dB SNR, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	8%	0%	17%	8%	8%	3	17%	8%	8%	42%	0%
4	8%	0%	17%	8%	8%	4	17%	8%	8%	42%	0%
5	8%	0%	0%	8%	8%	5	17%	0%	0%	25%	0%
6	8%	0%	17%	8%	8%	6	17%	8%	8%	42%	0%
7	50%	25%	50%	42%	58%	7	25%	67%	58%	58%	33%
Accuracy (Recording 2): MAC, -20 dB SNR, Flt, AGC on						Accuracy (Recording 2): AAC, -20 dB SNR, Flt, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	17%	0%	8%	17%	0%	3	25%	8%	17%	0%	8%
4	17%	0%	8%	8%	0%	4	8%	17%	17%	33%	0%
5	8%	0%	0%	8%	0%	5	0%	0%	17%	8%	8%
6	17%	0%	8%	8%	0%	6	17%	8%	8%	0%	0%
7	50%	17%	50%	33%	25%	7	50%	67%	75%	67%	42%
Accuracy (Recording 2): MAC, -20 dB SNR, Fix, AGC on						Accuracy (Recording 2): AAC, -20 dB SNR, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	17%	0%	8%	17%	0%	3	25%	8%	17%	0%	8%
4	17%	0%	8%	8%	0%	4	8%	17%	17%	33%	0%
5	8%	0%	0%	8%	0%	5	0%	0%	17%	8%	8%
6	17%	0%	8%	8%	0%	6	17%	8%	8%	0%	0%
7	50%	17%	50%	33%	25%	7	50%	67%	75%	67%	42%

Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	8%	8%	8%	0%	17%	3	0%	17%	17%	8%	0%
4	8%	8%	8%	0%	17%	4	0%	17%	17%	17%	0%
5	8%	0%	0%	8%	8%	5	0%	8%	8%	17%	0%
6	8%	8%	8%	0%	17%	6	0%	17%	17%	8%	0%
7	50%	25%	25%	42%	58%	7	50%	67%	42%	17%	58%
Accuracy (Recording 3): MAC, -20 dB SNR, Flt, AGC on						Accuracy (Recording 3): AAC, -20 dB SNR, Flt, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	0%	33%	0%	0%	8%	3	25%	0%	0%	8%	17%
4	0%	33%	0%	0%	8%	4	25%	17%	0%	17%	25%
5	0%	0%	0%	8%	17%	5	8%	0%	8%	0%	0%
6	0%	33%	0%	0%	8%	6	33%	8%	0%	17%	25%
7	42%	42%	50%	42%	25%	7	50%	33%	25%	42%	67%
Accuracy (Recording 3): MAC, -20 dB SNR, Fix, AGC on						Accuracy (Recording 3): AAC, -20 dB SNR, Fix, AGC on					
Intensity (dBFS):	-20	-10	0	10	20	Intensity (dBFS):	-20	-10	0	10	20
Algorithm						Algorithm					
1	0%	0%	0%	0%	0%	1	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	2	0%	0%	0%	0%	0%
3	17%	17%	8%	17%	17%	3	17%	17%	17%	8%	25%
4	17%	17%	17%	17%	17%	4	17%	17%	17%	8%	25%
5	8%	17%	8%	17%	25%	5	0%	25%	8%	8%	0%
6	17%	17%	17%	17%	17%	6	17%	17%	17%	8%	25%
7	25%	17%	58%	25%	33%	7	50%	42%	58%	50%	58%

Table C-7: Pitch estimation accuracy results for 12 musical notes (A4 – G#4) with -20 dB SNR (white Gaussian noise) and AGC enabled.

D. Classification of Musical Instruments (0 dBFS, No Noise)

Index	Profile		Circuit				Response				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
1	✓		✓				✓				75.00%
2	✓		✓					✓			50.00%
3	✓		✓						✓		66.67%
4	✓		✓							✓	58.33%
5	✓		✓				✓	✓			58.33%
6	✓		✓				✓		✓		75.00%
7	✓		✓				✓			✓	75.00%
8	✓		✓					✓	✓		66.67%
9	✓		✓					✓		✓	58.33%
10	✓		✓						✓	✓	58.33%
11	✓		✓				✓		✓	✓	75.00%
12	✓		✓					✓	✓	✓	75.00%
13	✓		✓				✓	✓	✓		75.00%
14	✓		✓				✓	✓		✓	66.67%
15	✓		✓				✓	✓	✓	✓	75.00%
16	✓			✓			✓				75.00%
17	✓			✓				✓			50.00%
18	✓			✓					✓		41.67%
19	✓			✓						✓	50.00%
20	✓			✓			✓	✓			58.33%
21	✓			✓			✓		✓		83.33%
22	✓			✓			✓			✓	66.67%
23	✓			✓				✓	✓		66.67%
24	✓			✓				✓		✓	66.67%
25	✓			✓					✓	✓	41.67%
26	✓			✓			✓		✓	✓	66.67%
27	✓			✓				✓	✓	✓	75.00%
28	✓			✓			✓	✓	✓		75.00%
29	✓			✓			✓	✓		✓	66.67%
30	✓			✓			✓	✓	✓	✓	75.00%
31	✓				✓		✓				75.00%
32	✓				✓			✓			50.00%
33	✓				✓				✓		50.00%
34	✓				✓					✓	50.00%
35	✓				✓		✓	✓			58.33%
36	✓				✓		✓		✓		66.67%
37	✓				✓		✓			✓	58.33%
38	✓				✓			✓	✓		66.67%
39	✓				✓			✓		✓	75.00%

Index	Profile		Circuit				Response				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
40	✓				✓				✓	✓	50.00%
41	✓				✓		✓		✓	✓	50.00%
42	✓				✓			✓	✓	✓	58.33%
43	✓				✓		✓	✓	✓		66.67%
44	✓				✓		✓	✓		✓	75.00%
45	✓				✓		✓	✓	✓	✓	66.67%
46	✓					✓	✓				75.00%
47	✓					✓		✓			50.00%
48	✓					✓			✓		66.67%
49	✓					✓				✓	75.00%
50	✓					✓	✓	✓			58.33%
51	✓					✓	✓		✓		75.00%
52	✓					✓	✓			✓	75.00%
53	✓					✓		✓	✓		66.67%
54	✓					✓		✓		✓	75.00%
55	✓					✓			✓	✓	66.67%
56	✓					✓	✓		✓	✓	75.00%
57	✓					✓		✓	✓	✓	75.00%
58	✓					✓	✓	✓	✓		75.00%
59	✓					✓	✓	✓		✓	75.00%
60	✓					✓	✓	✓	✓	✓	75.00%
61		✓	✓				✓				75.00%
62		✓	✓					✓			58.33%
63		✓	✓						✓		58.33%
64		✓	✓							✓	58.33%
65		✓	✓				✓	✓			75.00%
66		✓	✓				✓		✓		75.00%
67		✓	✓				✓			✓	75.00%
68		✓	✓					✓	✓		75.00%
69		✓	✓					✓		✓	75.00%
70		✓	✓						✓	✓	58.33%
71		✓	✓				✓		✓	✓	75.00%
72		✓	✓					✓	✓	✓	75.00%
73		✓	✓				✓	✓	✓		75.00%
74		✓	✓				✓	✓		✓	75.00%
75		✓	✓				✓	✓	✓	✓	75.00%
76		✓		✓			✓				75.00%
77		✓		✓				✓			58.33%
78		✓		✓					✓		75.00%
79		✓		✓						✓	66.67%
80		✓		✓			✓	✓			75.00%
81		✓		✓			✓		✓		83.33%

Index	Profile		Circuit				Response				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
82		✓		✓			✓			✓	83.33%
83		✓		✓				✓	✓		66.67%
84		✓		✓				✓		✓	58.33%
85		✓		✓					✓	✓	75.00%
86		✓		✓			✓		✓	✓	75.00%
87		✓		✓				✓	✓	✓	66.67%
88		✓		✓			✓	✓	✓		75.00%
89		✓		✓			✓	✓		✓	75.00%
90		✓		✓			✓	✓	✓	✓	66.67%
91		✓			✓		✓				75.00%
92		✓			✓			✓			58.33%
93		✓			✓				✓		66.67%
94		✓			✓					✓	75.00%
95		✓			✓		✓	✓			75.00%
96		✓			✓		✓		✓		83.33%
97		✓			✓		✓			✓	83.33%
98		✓			✓			✓	✓		66.67%
99		✓			✓			✓		✓	58.33%
100		✓			✓				✓	✓	66.67%
101		✓			✓		✓		✓	✓	83.33%
102		✓			✓			✓	✓	✓	75.00%
103		✓			✓		✓	✓	✓		75.00%
104		✓			✓		✓	✓		✓	75.00%
105		✓			✓		✓	✓	✓	✓	83.33%
106		✓				✓	✓				75.00%
107		✓				✓		✓			58.33%
108		✓				✓			✓		75.00%
109		✓				✓				✓	75.00%
110		✓				✓	✓	✓			75.00%
111		✓				✓	✓		✓		83.33%
112		✓				✓	✓			✓	83.33%
113		✓				✓		✓	✓		75.00%
114		✓				✓		✓		✓	75.00%
115		✓				✓			✓	✓	75.00%
116		✓				✓	✓		✓	✓	83.33%
117		✓				✓		✓	✓	✓	75.00%
118		✓				✓	✓	✓	✓		83.33%
119		✓				✓	✓	✓		✓	83.33%
120		✓				✓	✓	✓	✓	✓	75.00%

Table D-1: Accuracy results for classifying musical instruments using 2D correlation coefficient equation in a linear classifier on floating-point responses of various input signal configurations recorded from the CAR-Lite-A1 model.

Index	Profile		Circuit				Responses				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
1	✓		✓				✓				75.00%
2	✓		✓					✓			58.33%
3	✓		✓						✓		91.67%
4	✓		✓							✓	58.33%
5	✓		✓				✓	✓			66.67%
6	✓		✓				✓		✓		83.33%
7	✓		✓				✓			✓	83.33%
8	✓		✓					✓	✓		91.67%
9	✓		✓					✓		✓	66.67%
10	✓		✓						✓	✓	75.00%
11	✓		✓				✓		✓	✓	83.33%
12	✓		✓					✓	✓	✓	83.33%
13	✓		✓				✓	✓	✓		91.67%
14	✓		✓				✓	✓		✓	83.33%
15	✓		✓				✓	✓	✓	✓	83.33%
16	✓			✓			✓				75.00%
17	✓			✓				✓			58.33%
18	✓			✓					✓		66.67%
19	✓			✓						✓	66.67%
20	✓			✓			✓	✓			66.67%
21	✓			✓			✓		✓		83.33%
22	✓			✓			✓			✓	83.33%
23	✓			✓				✓	✓		91.67%
24	✓			✓				✓		✓	83.33%
25	✓			✓					✓	✓	58.33%
26	✓			✓			✓		✓	✓	83.33%
27	✓			✓				✓	✓	✓	91.67%
28	✓			✓			✓	✓	✓		91.67%
29	✓			✓			✓	✓		✓	83.33%
30	✓			✓			✓	✓	✓	✓	91.67%
31	✓				✓		✓				75.00%
32	✓				✓			✓			58.33%
33	✓				✓				✓		66.67%
34	✓				✓					✓	58.33%
35	✓				✓		✓	✓			66.67%
36	✓				✓		✓		✓		83.33%
37	✓				✓		✓			✓	100.00%
38	✓				✓			✓	✓		75.00%
39	✓				✓			✓		✓	75.00%
40	✓				✓				✓	✓	58.33%
41	✓				✓		✓		✓	✓	91.67%
42	✓				✓			✓	✓	✓	75.00%

Index	Profile		Circuit				Responses				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
43	✓				✓		✓	✓	✓		83.33%
44	✓				✓		✓	✓		✓	75.00%
45	✓				✓		✓	✓	✓	✓	83.33%
46	✓					✓	✓				75.00%
47	✓					✓		✓			58.33%
48	✓					✓			✓		58.33%
49	✓					✓				✓	66.67%
50	✓					✓	✓	✓			66.67%
51	✓					✓	✓		✓		75.00%
52	✓					✓	✓			✓	75.00%
53	✓					✓		✓	✓		83.33%
54	✓					✓		✓		✓	83.33%
55	✓					✓			✓	✓	66.67%
56	✓					✓	✓		✓	✓	83.33%
57	✓					✓		✓	✓	✓	91.67%
58	✓					✓	✓	✓	✓		91.67%
59	✓					✓	✓	✓		✓	83.33%
60	✓					✓	✓	✓	✓	✓	100.00%
61		✓	✓				✓				75.00%
62		✓	✓					✓			75.00%
63		✓	✓						✓		50.00%
64		✓	✓							✓	50.00%
65		✓	✓				✓	✓			75.00%
66		✓	✓				✓		✓		66.67%
67		✓	✓				✓			✓	75.00%
68		✓	✓					✓	✓		50.00%
69		✓	✓					✓		✓	66.67%
70		✓	✓						✓	✓	50.00%
71		✓	✓				✓		✓	✓	58.33%
72		✓	✓					✓	✓	✓	41.67%
73		✓	✓				✓	✓	✓		75.00%
74		✓	✓				✓	✓		✓	75.00%
75		✓	✓				✓	✓	✓	✓	83.33%
76		✓		✓			✓				75.00%
77		✓		✓				✓			75.00%
78		✓		✓					✓		75.00%
79		✓		✓						✓	66.67%
80		✓		✓			✓	✓			75.00%
81		✓		✓			✓		✓		83.33%
82		✓		✓			✓			✓	91.67%
83		✓		✓				✓	✓		75.00%
84		✓		✓				✓		✓	66.67%

Index	Profile		Circuit				Responses				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
85		✓		✓					✓	✓	75.00%
86		✓		✓			✓		✓	✓	83.33%
87		✓		✓				✓	✓	✓	66.67%
88		✓		✓			✓	✓	✓		75.00%
89		✓		✓			✓	✓		✓	75.00%
90		✓		✓			✓	✓	✓	✓	75.00%
91		✓			✓		✓				75.00%
92		✓			✓			✓			75.00%
93		✓			✓				✓		66.67%
94		✓			✓					✓	50.00%
95		✓			✓		✓	✓			75.00%
96		✓			✓		✓		✓		75.00%
97		✓			✓		✓			✓	58.33%
98		✓			✓			✓	✓		66.67%
99		✓			✓			✓		✓	58.33%
100		✓			✓				✓	✓	75.00%
101		✓			✓		✓		✓	✓	75.00%
102		✓			✓			✓	✓	✓	58.33%
103		✓			✓		✓	✓	✓		75.00%
104		✓			✓		✓	✓		✓	75.00%
105		✓			✓		✓	✓	✓	✓	58.33%
106		✓				✓	✓				75.00%
107		✓				✓		✓			75.00%
108		✓				✓			✓		75.00%
109		✓				✓				✓	66.67%
110		✓				✓	✓	✓			75.00%
111		✓				✓	✓		✓		83.33%
112		✓				✓	✓			✓	83.33%
113		✓				✓		✓	✓		75.00%
114		✓				✓		✓		✓	66.67%
115		✓				✓			✓	✓	66.67%
116		✓				✓	✓		✓	✓	83.33%
117		✓				✓		✓	✓	✓	75.00%
118		✓				✓	✓	✓	✓		83.33%
119		✓				✓	✓	✓		✓	83.33%
120		✓				✓	✓	✓	✓	✓	83.33%

Table D-2: Accuracy results for classifying musical instruments using 2D correlation coefficient equation in a linear classifier on fixed-point responses of various signal input configurations recorded from the CAR-Lite-A1 model.

Index	Profile		Circuit				Response				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
1	✓		✓				✓				66.67%
2	✓		✓					✓			83.33%
3	✓		✓						✓		41.67%
4	✓		✓							✓	50.00%
5	✓		✓				✓	✓			83.33%
6	✓		✓				✓		✓		66.67%
7	✓		✓				✓			✓	66.67%
8	✓		✓					✓	✓		83.33%
9	✓		✓					✓		✓	83.33%
10	✓		✓						✓	✓	50.00%
11	✓		✓				✓		✓	✓	66.67%
12	✓		✓					✓	✓	✓	83.33%
13	✓		✓				✓	✓	✓		83.33%
14	✓		✓				✓	✓		✓	83.33%
15	✓		✓				✓	✓	✓	✓	83.33%
16	✓			✓			✓				66.67%
17	✓			✓				✓			83.33%
18	✓			✓					✓		50.00%
19	✓			✓						✓	50.00%
20	✓			✓			✓	✓			83.33%
21	✓			✓			✓		✓		83.33%
22	✓			✓			✓			✓	75.00%
23	✓			✓				✓	✓		83.33%
24	✓			✓				✓		✓	83.33%
25	✓			✓					✓	✓	50.00%
26	✓			✓			✓		✓	✓	66.67%
27	✓			✓				✓	✓	✓	83.33%
28	✓			✓			✓	✓	✓		83.33%
29	✓			✓			✓	✓		✓	83.33%
30	✓			✓			✓	✓	✓	✓	83.33%
31	✓				✓		✓				66.67%
32	✓				✓			✓			83.33%
33	✓				✓				✓		66.67%
34	✓				✓					✓	58.33%
35	✓				✓		✓	✓			83.33%
36	✓				✓		✓		✓		66.67%
37	✓				✓		✓			✓	58.33%
38	✓				✓			✓	✓		83.33%
39	✓				✓			✓		✓	83.33%
40	✓				✓				✓	✓	58.33%
41	✓				✓		✓		✓	✓	66.67%
42	✓				✓			✓	✓	✓	91.67%

Index	Profile		Circuit				Response				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
43	✓				✓		✓	✓	✓		83.33%
44	✓				✓		✓	✓		✓	83.33%
45	✓				✓		✓	✓	✓	✓	75.00%
46	✓					✓	✓				66.67%
47	✓					✓		✓			83.33%
48	✓					✓			✓		50.00%
49	✓					✓				✓	58.33%
50	✓					✓	✓	✓			83.33%
51	✓					✓	✓		✓		66.67%
52	✓					✓	✓			✓	66.67%
53	✓					✓		✓	✓		83.33%
54	✓					✓		✓		✓	83.33%
55	✓					✓			✓	✓	50.00%
56	✓					✓	✓		✓	✓	66.67%
57	✓					✓		✓	✓	✓	83.33%
58	✓					✓	✓	✓	✓		83.33%
59	✓					✓	✓	✓		✓	83.33%
60	✓					✓	✓	✓	✓	✓	83.33%
61		✓	✓				✓				66.67%
62		✓	✓					✓			41.67%
63		✓	✓						✓		66.67%
64		✓	✓							✓	66.67%
65		✓	✓				✓	✓			66.67%
66		✓	✓				✓		✓		66.67%
67		✓	✓				✓			✓	66.67%
68		✓	✓					✓	✓		41.67%
69		✓	✓					✓		✓	41.67%
70		✓	✓						✓	✓	66.67%
71		✓	✓				✓		✓	✓	66.67%
72		✓	✓					✓	✓	✓	41.67%
73		✓	✓				✓	✓	✓		66.67%
74		✓	✓				✓	✓		✓	66.67%
75		✓	✓				✓	✓	✓	✓	66.67%
76		✓		✓			✓				66.67%
77		✓		✓				✓			41.67%
78		✓		✓					✓		66.67%
79		✓		✓						✓	58.33%
80		✓		✓			✓	✓			66.67%
81		✓		✓			✓		✓		66.67%
82		✓		✓			✓			✓	66.67%
83		✓		✓				✓	✓		41.67%
84		✓		✓				✓		✓	41.67%

Index	Profile		Circuit				Response				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
85		✓		✓					✓	✓	66.67%
86		✓		✓			✓		✓	✓	66.67%
87		✓		✓				✓	✓	✓	41.67%
88		✓		✓			✓	✓	✓		66.67%
89		✓		✓			✓	✓		✓	66.67%
90		✓		✓			✓	✓	✓	✓	66.67%
91		✓			✓		✓				66.67%
92		✓			✓			✓			41.67%
93		✓			✓				✓		66.67%
94		✓			✓					✓	50.00%
95		✓			✓		✓	✓			66.67%
96		✓			✓		✓		✓		66.67%
97		✓			✓		✓			✓	66.67%
98		✓			✓			✓	✓		41.67%
99		✓			✓			✓		✓	41.67%
100		✓			✓				✓	✓	66.67%
101		✓			✓		✓		✓	✓	66.67%
102		✓			✓			✓	✓	✓	41.67%
103		✓			✓		✓	✓	✓		66.67%
104		✓			✓		✓	✓		✓	66.67%
105		✓			✓		✓	✓	✓	✓	66.67%
106		✓				✓	✓				66.67%
107		✓				✓		✓			41.67%
108		✓				✓			✓		66.67%
109		✓				✓				✓	66.67%
110		✓				✓	✓	✓			66.67%
111		✓				✓	✓		✓		66.67%
112		✓				✓	✓			✓	66.67%
113		✓				✓		✓	✓		41.67%
114		✓				✓		✓		✓	41.67%
115		✓				✓			✓	✓	66.67%
116		✓				✓	✓		✓	✓	66.67%
117		✓				✓		✓	✓	✓	41.67%
118		✓				✓	✓	✓	✓		66.67%
119		✓				✓	✓	✓		✓	66.67%
120		✓				✓	✓	✓	✓	✓	66.67%

Table D-3: Accuracy results for classifying musical instruments using timbre distance algorithm in a k-nearest neighbour (KNN) classifier on floating-point responses of various input signal configurations recorded from the CAR-Lite-A1 model.

Index	Profile		Circuit				Responses				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
1	✓		✓				✓				66.67%
2	✓		✓					✓			91.67%
3	✓		✓						✓		50.00%
4	✓		✓							✓	58.33%
5	✓		✓				✓	✓			83.33%
6	✓		✓				✓		✓		83.33%
7	✓		✓				✓			✓	83.33%
8	✓		✓					✓	✓		91.67%
9	✓		✓					✓		✓	91.67%
10	✓		✓						✓	✓	58.33%
11	✓		✓				✓		✓	✓	83.33%
12	✓		✓					✓	✓	✓	91.67%
13	✓		✓				✓	✓	✓		91.67%
14	✓		✓				✓	✓		✓	91.67%
15	✓		✓				✓	✓	✓	✓	91.67%
16	✓			✓			✓				66.67%
17	✓			✓				✓			91.67%
18	✓			✓					✓		41.67%
19	✓			✓						✓	58.33%
20	✓			✓			✓	✓			83.33%
21	✓			✓			✓		✓		75.00%
22	✓			✓			✓			✓	58.33%
23	✓			✓				✓	✓		91.67%
24	✓			✓				✓		✓	91.67%
25	✓			✓					✓	✓	50.00%
26	✓			✓			✓		✓	✓	58.33%
27	✓			✓				✓	✓	✓	91.67%
28	✓			✓			✓	✓	✓		91.67%
29	✓			✓			✓	✓		✓	91.67%
30	✓			✓			✓	✓	✓	✓	91.37%
31	✓				✓		✓				66.67%
32	✓				✓			✓			91.67%
33	✓				✓				✓		66.67%
34	✓				✓					✓	58.33%
35	✓				✓		✓	✓			83.33%
36	✓				✓		✓		✓		66.67%
37	✓				✓		✓			✓	75.00%
38	✓				✓			✓	✓		91.67%
39	✓				✓			✓		✓	91.67%
40	✓				✓				✓	✓	58.33%
41	✓				✓		✓		✓	✓	66.67%
42	✓				✓			✓	✓	✓	91.67%

Index	Profile		Circuit				Responses				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
43	✓				✓		✓	✓	✓		91.67%
44	✓				✓		✓	✓		✓	91.67%
45	✓				✓		✓	✓	✓	✓	91.67%
46	✓					✓	✓				66.67%
47	✓					✓		✓			91.67%
48	✓					✓			✓		41.67%
49	✓					✓				✓	41.67%
50	✓					✓	✓	✓			83.33%
51	✓					✓	✓		✓		41.67%
52	✓					✓	✓			✓	41.67%
53	✓					✓		✓	✓		58.33%
54	✓					✓		✓		✓	50.00%
55	✓					✓			✓	✓	41.67%
56	✓					✓	✓		✓	✓	41.67%
57	✓					✓		✓	✓	✓	41.67%
58	✓					✓	✓	✓	✓		58.33%
59	✓					✓	✓	✓		✓	50.00%
60	✓					✓	✓	✓	✓	✓	41.67%
61		✓	✓				✓				66.67%
62		✓	✓					✓			41.67%
63		✓	✓						✓		66.67%
64		✓	✓							✓	58.33%
65		✓	✓				✓	✓			66.67%
66		✓	✓				✓		✓		66.67%
67		✓	✓				✓			✓	66.67%
68		✓	✓					✓	✓		41.67%
69		✓	✓					✓		✓	41.67%
70		✓	✓						✓	✓	58.33%
71		✓	✓				✓		✓	✓	66.67%
72		✓	✓					✓	✓	✓	41.67%
73		✓	✓				✓	✓	✓		66.67%
74		✓	✓				✓	✓		✓	66.67%
75		✓	✓				✓	✓	✓	✓	66.67%
76		✓		✓			✓				66.67%
77		✓		✓				✓			41.67%
78		✓		✓					✓		66.67%
79		✓		✓						✓	66.67%
80		✓		✓			✓	✓			66.67%
81		✓		✓			✓		✓		66.67%
82		✓		✓			✓			✓	66.67%
83		✓		✓				✓	✓		41.67%
84		✓		✓				✓		✓	41.67%

Index	Profile		Circuit				Responses				Accuracy
	Sum	RMS	Real	Imag	Real + Imag	Real × Imag	A1x	A1y	Up	Down	
85		✓		✓					✓	✓	66.67%
86		✓		✓			✓		✓	✓	66.67%
87		✓		✓				✓	✓	✓	41.67%
88		✓		✓			✓	✓	✓		66.67%
89		✓		✓			✓	✓		✓	66.67%
90		✓		✓			✓	✓	✓	✓	66.67%
91		✓			✓		✓				66.67%
92		✓			✓			✓			41.67%
93		✓			✓				✓		66.67%
94		✓			✓					✓	58.33%
95		✓			✓		✓	✓			66.67%
96		✓			✓		✓		✓		66.67%
97		✓			✓		✓			✓	66.67%
98		✓			✓			✓	✓		41.67%
99		✓			✓			✓		✓	41.67%
100		✓			✓				✓	✓	75.00%
101		✓			✓		✓		✓	✓	66.67%
102		✓			✓			✓	✓	✓	41.67%
103		✓			✓		✓	✓	✓		66.67%
104		✓			✓		✓	✓		✓	66.67%
105		✓			✓		✓	✓	✓	✓	66.67%
106		✓				✓	✓				66.67%
107		✓				✓		✓			41.67%
108		✓				✓			✓		58.33%
109		✓				✓				✓	58.33%
110		✓				✓	✓	✓			66.67%
111		✓				✓	✓		✓		66.67%
112		✓				✓	✓			✓	66.67%
113		✓				✓		✓	✓		66.67%
114		✓				✓		✓		✓	58.33%
115		✓				✓			✓	✓	58.33%
116		✓				✓	✓		✓	✓	66.67%
117		✓				✓		✓	✓	✓	66.67%
118		✓				✓	✓	✓	✓		66.67%
119		✓				✓	✓	✓		✓	66.67%
120		✓				✓	✓	✓	✓	✓	66.67%

Table D-4: Accuracy results for classifying musical instruments using timbre distance algorithm in a k-nearest neighbour (KNN) classifier on fixed-point responses of various input signal configurations recorded from the CAR-Lite-A1 model.

E. Classification of Musical Instruments based on Varying Intensity and SNR Levels

	SUM									
	2D Correlation (Re)									
	No AGC					AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Fix:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Timbre Distance (Re)									
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	67%	67%	67%	67%	100%	100%	100%	100%	100%
Fix:	67%	67%	67%	67%	67%	100%	100%	100%	100%	100%

	RMS									
	2D Correlation (Re)									
	No AGC					AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Fix:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Timbre Distance (Re)									
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	67%	67%	67%	67%	100%	100%	100%	100%	100%
Fix:	67%	67%	67%	67%	100%	100%	100%	100%	100%	100%

Table E-1: Accuracy results for classifying musical instruments based on varying intensity levels of musical signals without noise. Flt: Floating-point; Fix: Fixed-point; Re: Real component of a complex signal output from the CAR-Lite-A1 model.

	SUM (Recording 1)									
	2D Correlation (Re)									
	No AGC					AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	67%	100%	100%	100%	67%	67%	67%	67%	67%
Fix:	100%	100%	100%	67%	67%	67%	67%	67%	67%	67%
	Timbre Distance (Re)									
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	100%	100%	67%	67%	100%	100%	100%	100%	100%
Fix:	67%	67%	67%	67%	100%	100%	100%	100%	100%	100%

	RMS (Recording 1)									
	2D Correlation (Re)									
	No AGC					AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	67%	100%	100%	100%	100%	100%	100%	100%	100%
Fix:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Timbre Distance (Re)									

Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	67%	100%	67%	67%	100%	100%	100%	100%	100%
Fix:	100%	67%	67%	67%	67%	100%	100%	100%	100%	100%

SUM (Recording 2)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	100%	100%	100%	100%	67%	67%	67%	67%	67%
Fix:	100%	100%	100%	100%	100%	67%	67%	67%	67%	67%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	100%	67%	67%	67%	100%	100%	100%	100%	100%
Fix:	100%	67%	100%	67%	100%	100%	100%	100%	100%	100%

RMS (Recording 2)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	100%	100%	100%	67%	100%	100%	100%	100%	100%
Fix:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	100%	67%	67%	67%	100%	100%	100%	100%	100%
Fix:	100%	67%	67%	100%	100%	100%	100%	100%	100%	100%

SUM (Recording 3)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	100%	67%	67%	67%	100%	67%	67%	67%	67%
Fix:	100%	100%	100%	100%	67%	67%	67%	67%	67%	67%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	67%	67%	67%	67%	100%	100%	100%	100%	100%
Fix:	67%	67%	67%	67%	100%	100%	100%	100%	100%	100%

RMS (Recording 3)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	100%	100%	67%	100%	67%	100%	100%	100%	100%
Fix:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Timbre Distance (Re)										

Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	67%	100%	67%	67%	100%	100%	100%	100%	100%
Fix:	67%	100%	67%	100%	100%	100%	100%	100%	100%	100%

Table E-2: Three sets of accuracy results for classifying musical instruments based on varying intensity levels of musical signals at 20 dB SNR. Flt: Floating-point; Fix: Fixed-point; Re: Real component of a complex signal output from the CAR-Lite-A1 model.

SUM (Recording 1)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	67%	67%	67%	67%	67%	67%	33%	33%	33%
Fix:	100%	100%	67%	67%	67%	67%	33%	67%	33%	33%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	100%	67%	67%	100%	100%	100%	100%	100%	100%
Fix:	100%	100%	100%	67%	67%	100%	100%	100%	100%	100%

RMS (Recording 1)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	67%	33%	67%	67%	67%	67%	33%	67%	67%
Fix:	67%	67%	67%	67%	67%	100%	33%	67%	33%	67%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	67%	100%	67%	33%	100%	100%	100%	100%	100%
Fix:	67%	100%	100%	67%	67%	100%	100%	100%	100%	100%

SUM (Recording 2)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	33%	67%	33%	33%	67%	67%	67%	33%	33%
Fix:	67%	67%	100%	33%	67%	33%	33%	33%	33%	67%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	100%	100%	67%	100%	67%	100%	100%	100%	100%	100%
Fix:	67%	67%	67%	67%	67%	100%	67%	67%	100%	100%

RMS (Recording 2)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	33%	67%	100%	67%	67%	33%	67%	67%	67%	33%

Fix:	67%	67%	67%	100%	33%	67%	100%	33%	33%	100%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	33%	67%	100%	67%	67%	100%	100%	100%	100%
Fix:	33%	67%	33%	67%	67%	100%	67%	100%	100%	100%

SUM (Recording 3)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	100%	67%	100%	67%	33%	33%	33%	33%	67%
Fix:	67%	67%	67%	67%	67%	33%	33%	33%	67%	33%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	33%	67%	67%	67%	100%	100%	100%	100%	67%
Fix:	100%	67%	67%	67%	67%	100%	67%	67%	67%	100%

RMS (Recording 3)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	100%	33%	33%	33%	67%	33%	33%	67%	67%
Fix:	33%	67%	100%	67%	67%	67%	33%	33%	33%	33%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	33%	100%	100%	33%	67%	67%	67%	67%	100%	100%
Fix:	100%	100%	67%	33%	67%	100%	100%	67%	100%	100%

Table E-3: Three sets of accuracy results for classifying musical instruments based on varying intensity levels of musical signals at 0 dB SNR. Flt: Floating-point; Fix: Fixed-point; Re: Real component of a complex signal output from the CAR-Lite-A1 model.

SUM (Recording 1)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	67%	67%	0%	67%	0%	0%	0%	33%	33%
Fix:	0%	33%	33%	33%	33%	0%	0%	0%	33%	0%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	0%	33%	0%	33%	0%	0%	0%	33%	67%	33%
Fix:	0%	0%	33%	33%	33%	33%	0%	67%	0%	33%

RMS (Recording 1)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20

Flt:	33%	33%	33%	67%	33%	33%	0%	0%	67%	0%
Fix:	33%	33%	0%	33%	33%	33%	33%	0%	33%	67%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	33%	33%	33%	0%	67%	0%	0%	67%	33%	0%
Fix:	33%	0%	67%	33%	67%	0%	33%	33%	0%	33%

SUM (Recording 2)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	0%	33%	33%	67%	33%	33%	67%	67%	33%	67%
Fix:	33%	33%	67%	33%	33%	33%	0%	33%	33%	67%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	33%	0%	67%	0%	33%	67%	33%	33%	67%	67%
Fix:	33%	0%	67%	33%	33%	0%	33%	33%	67%	33%

RMS (Recording 2)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	67%	67%	67%	0%	100%	0%	67%	0%	33%	67%
Fix:	67%	100%	67%	67%	0%	33%	67%	67%	0%	33%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	0%	0%	33%	33%	0%	67%	33%	33%	0%	33%
Fix:	0%	67%	33%	0%	33%	33%	33%	33%	33%	67%

SUM (Recording 3)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	33%	0%	0%	0%	0%	0%	0%	33%	33%	0%
Fix:	33%	33%	0%	33%	33%	0%	33%	33%	0%	0%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	33%	67%	0%	67%	67%	33%	33%	33%	0%	33%
Fix:	33%	33%	0%	0%	0%	33%	33%	0%	33%	33%

RMS (Recording 3)										
2D Correlation (Re)										
No AGC						AGC Enabled				
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20

Flt:	67%	67%	67%	67%	67%	33%	33%	67%	33%	33%
Fix:	33%	0%	33%	0%	100%	33%	67%	33%	67%	33%
Timbre Distance (Re)										
Intensity (dBFS):	-20	-10	0	10	20	-20	-10	0	10	20
Flt:	33%	67%	67%	33%	33%	67%	33%	0%	33%	100%
Fix:	33%	0%	33%	67%	0%	33%	67%	67%	33%	0%

Table E-4: Three sets of accuracy results for classifying musical instruments based on varying intensity levels of musical signals at -20 dB SNR. Flt: Floating-point; Fix: Fixed-point; Re: Real component of a complex signal output from the CAR-Lite-A1 model.