



# Durham E-Theses

---

## *Algorithms and VLSI architectures for parametric additive synthesis*

Spanier, Jonathan Robert

### How to cite:

---

Spanier, Jonathan Robert (1999) *Algorithms and VLSI architectures for parametric additive synthesis*, Durham theses, Durham University. Available at Durham E-Theses Online: <http://etheses.dur.ac.uk/4536/>

### Use policy

---

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a [link](#) is made to the metadata record in Durham E-Theses
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the [full Durham E-Theses policy](#) for further details.

THE UNIVERSITY OF DURHAM

SCHOOL OF ENGINEERING

*Algorithms and VLSI  
Architectures for  
Parametric Additive  
Synthesis*

Jonathan Robert Spanier

19 JUL 2000

A THESIS SUBMITTED IN PARTIAL FUL-  
FILLMENT OF THE REQUIREMENTS OF THE  
COUNCIL OF THE UNIVERSITY OF DURHAM  
FOR THE DEGREE OF DOCTOR OF PHILOS-  
OPHY (Ph.D.).



MAY 1999

The copyright of this thesis rests  
with the author. No quotation  
from it should be published  
without the written consent of the  
author and information derived  
from it should be acknowledged.

Algorithms and VLSI Architectures for Parametric Additive Synthesis  
Jonathan Robert Spanier  
Ph.D. 1999

## Abstract

A parametric additive synthesis approach to sound synthesis is advantageous as it can model sounds in a large scale manner, unlike the classical sinusoidal additive based synthesis paradigms. It is known that a large body of naturally occurring sounds are resonant in character and thus fit the concept well. This thesis is concerned with the computational optimisation of a super class of formant synthesis which extends the sinusoidal parameters with a spread parameter known as bandwidth. Here a modified formant algorithm is introduced which can be traced back to work done at IRCAM, Paris.

When impulse driven, a filter based approach to modelling a formant limits the computational work-load. It is assumed that the filter's coefficients are fixed at initialisation, thus avoiding interpolation which can cause the filter to become chaotic. A filter which is more complex than a second order section is required.

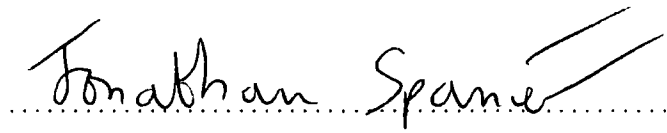
Temporal resolution of an impulse generator is achieved by using a two stage polyphase decimator which drives many filterbanks. Each filterbank describes one formant and is composed of sub-elements which allow variation of the formant's parameters. A resource manager is discussed to overcome the possibility of all sub-banks operating in unison. All filterbanks for one voice are connected in series to the impulse generator and their outputs are summed and scaled accordingly.

An explorative study of number systems for DSP algorithms and their architectures is investigated. I invented a new theoretical mechanism for multi-level logic based DSP. Its aims are to reduce the number of transistors and to increase their functionality.

A review of synthesis algorithms and VLSI architectures are discussed in a case study between a filter based bit-serial and a CORDIC based sinusoidal generator. They are both of similar size, but the latter is always guaranteed to be stable.

# Declaration

I hereby declare that this thesis is a record of work undertaken by myself, that it has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged.

A handwritten signature in cursive script, reading "Jonathan Spanier", written over a dotted line. The signature is followed by two diagonal strokes.

Jonathan Robert Spanier, May 1999

© Copyright 1999, Jonathan Robert Spanier

The copyright of this thesis rests with the author. No quotation from it should be published without the written consent of the copyright owner, and information derived from it should be acknowledged.

# Acknowledgments

I would like to take this opportunity to thank both my supervisors, Professor Alan Purvis and Dr. Simon Johnson for the encouragement and support they provided over the course of study.

The financial assistance provided by the University of Durham Studentship and the Royal Academy of Engineering International Travel Grant Scheme is gratefully acknowledged.

I would like to thank Mike Ellison of the University IT Computing Service for helping me to operate the Cadence Design Integration tools. My thanks go to members of the electronics workshop and to the departmental computing support; especially to Peter Friend, Ian Hutchinson, Peter Baxendale and Trevor Nancarrow for their advice and assistance in electronic and computing problems. Special thanks go to Matthew Jubb and John Glover for enlightening me in the joys of UNIX system administration and running Linux.

I am indebted to my fellow postgraduates, especially to Steve Robinson, Martin Bradley and the rest of the Telecoms Networks Research Group. I would like to thank my colleagues in the Digital and Optical Signal Processing Group and the VLSI Research Group for the informative discussions we had over the three years. I would like to acknowledge the  $\text{\LaTeX}$  community for providing such a useful and worthwhile typesetting package.

Special thanks also go to Austin Cassidy, Dr. Roger Woods and Dr. Saeed Vaseghi of The Queen's University of Belfast for providing UNIX support, portable computers and for all their encouragement.

Finally I would like to thank my parents for providing steadfast support throughout all my endeavours and to give special thanks to my mother, who tirelessly proof read this thesis despite not understanding any of the technical aspects.

---

# Contents

---

<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Abbreviations</b>	<b>viii</b>
<b>1 Background and Structure</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Aims of this Thesis . . . . .	2
1.3 Summary of Thesis . . . . .	3
<b>2 Introduction to Synthesis methods</b>	<b>5</b>
2.1 Subtractive Synthesis . . . . .	6
2.1.1 The Digital Controlled Oscillator . . . . .	7
2.1.2 The Digital Controlled Amplifier . . . . .	10
2.1.3 The Digital Controlled Filter . . . . .	11
2.2 Additive Synthesis . . . . .	12
2.2.1 Time-Multiplexed Wavetable Oscillators . . . . .	13
2.2.2 Inverse FFT-based Oscillators . . . . .	14
2.2.3 Multirate Additive Oscillators . . . . .	14
2.3 Frequency Modulation . . . . .	15
2.4 Data Reduction for Synthesis . . . . .	16
2.5 Waveguide Modelling . . . . .	17
2.6 Speech Modelling . . . . .	18
2.6.1 Time-Domain Modelling . . . . .	18
2.6.2 Spectral Modelling . . . . .	19
2.6.2.1 VOSIM . . . . .	20
2.6.2.2 Parallel/Cascade Formant Synthesis . . . . .	21
2.6.2.3 Forme d'Onde Formantique . . . . .	24
2.7 Summary . . . . .	25

<b>3</b>	<b>Number Systems for VLSI Processing Elements</b>	<b>27</b>
3.1	Introduction to Number Systems . . . . .	27
3.2	Conventional Radix Number System . . . . .	30
3.2.1	Weighted Number Systems . . . . .	30
3.2.2	Sign-Magnitude Representation . . . . .	31
3.2.3	Diminished-Radix Complement Representation . . . . .	31
3.2.4	Radix Complement Representation . . . . .	32
3.2.5	Properties of Sign-Magnitude, $r-1$ 's and $r$ 's Complement Numbers . . . . .	32
3.3	The Signed-Digit Number System . . . . .	33
3.3.1	Definition of Signed-Digit Numbers . . . . .	33
3.3.2	Conversion Between The Conventional Radix- $r$ Number and its SD Form . . . . .	34
3.3.2.1	Conversion from Conventional to SD Systems . . . .	34
3.3.2.2	Conversion from SD to Conventional Systems . . . .	35
3.4	The Residue Number System . . . . .	35
3.4.1	The Multiplicative Inverse . . . . .	36
3.4.1.1	Euler's Formula . . . . .	37
3.4.2	Residue to Decimal Conversion . . . . .	37
3.4.2.1	Mixed-Radix Method . . . . .	37
3.4.2.2	Chinese Remainder Theorem . . . . .	38
3.4.3	The Quadratic Residue Number System . . . . .	38
3.4.3.1	Conversion To Complex Quadratic Residue Number System . . . . .	38
3.4.3.2	Conversion Back To Conventional Number System .	39
3.5	Addition Elements . . . . .	40
3.5.1	Conventional Binary Addition and Subtraction . . . . .	40
3.5.2	Signed Digit Addition and Subtraction . . . . .	41
3.5.2.1	The Algorithm. . . . .	41
3.6	Array Multiplication Techniques . . . . .	44
3.6.1	Cellular Array Multipliers Conclusions . . . . .	46
3.7	Canonical Multiplier Recoding . . . . .	47
3.7.1	Introduction to Multiplication String Recoding Algorithms .	47
3.7.2	Canonical Signed-Digit Multiplier Recoding . . . . .	48
3.7.3	Canonical Recoding Algorithm . . . . .	48
3.7.4	The Booth's Multiplier Algorithm . . . . .	49
3.7.5	Efficiency of Multiplier Algorithms and Design Alternatives .	50
3.8	Summary . . . . .	51
<b>4</b>	<b>VLSI Technologies and Applications</b>	<b>53</b>
4.1	Novel VLSI Technologies . . . . .	54
4.2	Current-Mode Logic . . . . .	55
4.2.1	Current Source . . . . .	56
4.2.2	Current Mirror . . . . .	57
4.2.3	Threshold Detector . . . . .	58
4.2.4	Bidirectional current input circuit . . . . .	58

4.2.5	Module VLSI Scaling . . . . .	60
4.3	Negative Differential Resistance . . . . .	60
4.4	Voltage-Mode Multiple-Valued Logic . . . . .	62
4.4.1	Operational Amplifier Approach . . . . .	62
4.4.2	Neuron MOSFET Approach . . . . .	62
4.5	DAD Logic . . . . .	63
4.5.1	Mathematical Theory behind DAD Logic . . . . .	66
4.5.2	Harmonic Oscillator . . . . .	67
4.5.3	Bipolar Level System . . . . .	68
4.5.4	Value Determination through Approximation . . . . .	70
4.5.4.1	Analogue Eigenvalue Methods . . . . .	70
4.5.4.2	The Shooting Method . . . . .	71
4.5.4.3	The Pruess Method . . . . .	74
4.5.4.4	Important Issues regarding Eigenvalue Determination . . . . .	77
4.6	Summary . . . . .	77
<b>5</b>	<b>VLSI requirements for synthesis</b> . . . . .	<b>79</b>
5.1	DSP Systems Design . . . . .	79
5.1.1	Multiplier Constraints . . . . .	79
5.1.2	MAC Constraints . . . . .	80
5.1.3	ALU Constraints . . . . .	81
5.1.4	Data Address Generator . . . . .	81
5.1.5	Program Sequencer . . . . .	81
5.1.6	Memory . . . . .	82
5.1.7	Architecture Summary . . . . .	82
5.2	Datapath length criteria . . . . .	83
5.2.1	Overflow and underflow reduction . . . . .	83
5.2.2	Truncation, Rounding and Unbiased Rounding . . . . .	84
5.2.3	Saturation Arithmetic . . . . .	85
5.2.4	Other noise problems . . . . .	85
5.3	Filter based Sine Generator . . . . .	86
5.3.1	Filter Structure . . . . .	87
5.3.2	Multiplier Design . . . . .	88
5.3.3	Accumulator Design . . . . .	90
5.3.4	Filter Design . . . . .	90
5.4	The CORDIC Sine Generator . . . . .	91
5.4.1	Derivation for Circular Mode . . . . .	91
5.4.2	The CORDIC Algorithm . . . . .	92
5.4.3	CORDIC Sine Algorithm . . . . .	94
5.4.4	Design of the CORDIC sine Generator . . . . .	95
5.4.5	Sequencer Design . . . . .	98
5.4.6	ROM Design . . . . .	100
5.4.7	CORDIC register Design . . . . .	102
5.4.8	Adder Design . . . . .	103
5.4.9	Construction and Testing . . . . .	103
5.5	Comparisons between Sinusoidal Generators . . . . .	108



<b>6</b>	<b>Decaying Sinusoidal Additive Synthesis</b>	<b>110</b>
6.1	Analysis of a Second Order Filter Structure . . . . .	111
6.2	Analysis of the FOF technique . . . . .	112
6.2.1	Implementation Approaches to the FOF wavefunction . . . . .	114
6.2.2	Wavetable Based FOF Synthesis . . . . .	115
6.2.3	Filter Based FOF Synthesis . . . . .	117
6.2.4	Full Hanning Window FOF envelope . . . . .	120
6.2.5	FOF filter structure with oscillation . . . . .	122
6.2.6	Miscellaneous Wavefunction Designs . . . . .	124
6.3	Approaches to formant synthesis . . . . .	125
<b>7</b>	<b>Parametric Additive Synthesis Implementation</b>	<b>127</b>
7.1	Optimisation of the filter $t^2 e^{-\alpha t} \sin(\omega_0 t)$ . . . . .	128
7.1.1	Analysis of the Transversal Element of the Filter Topology . . . . .	128
7.1.2	Transversal Filter Structure . . . . .	129
7.1.3	IIR Filter Structure . . . . .	130
7.2	Source Excitation Driver . . . . .	131
7.2.1	Discrete Summation Formulas for Impulse Generation . . . . .	133
7.2.2	Impulse generation using Decimation . . . . .	134
7.2.2.1	The Decimator Structure . . . . .	134
7.2.2.2	Decimator Design . . . . .	135
7.2.2.3	Reduced Coefficient Storage Design . . . . .	137
7.3	Synthesiser Structure . . . . .	138
7.3.1	The importance of Phase in Audio . . . . .	138
7.3.2	Parallel Formant Model . . . . .	139
<b>8</b>	<b>Algorithm Scheduling and Miscellaneous Topics</b>	<b>143</b>
8.1	Algorithm Scheduling . . . . .	143
8.1.1	Controller Algorithm Overview . . . . .	144
8.1.2	Filterbank Scheduler Controller . . . . .	145
8.1.3	Number of Filters required . . . . .	146
8.2	Miscellaneous Topics . . . . .	147
8.2.1	Envelope Generation . . . . .	147
8.2.2	Subtractive Synthesis Extensions . . . . .	148
8.2.3	Synthesis by Rule . . . . .	149
8.3	User Interface Design . . . . .	150
8.4	Hardware/Software Segmentation . . . . .	151
<b>9</b>	<b>Conclusions and Further Work</b>	<b>153</b>
9.1	Conclusions . . . . .	153
9.2	Further Work . . . . .	156
<b>A</b>	<b>The Laplace to Z Transform Algorithm</b>	<b>158</b>

---

---

# List of Figures

---

2.1	Simplified Subtractive Voice Architecture . . . . .	7
2.2	Block Diagram of a Wavetable Oscillator . . . . .	8
2.3	Second Order Digital Filter . . . . .	11
2.4	LPC Synthesiser . . . . .	19
2.5	Typical VOSIM waveform . . . . .	20
2.6	Block Diagram of Serial-parallel formant synthesiser . . . . .	23
2.7	FOF synthesiser topology . . . . .	24
2.8	Properties of the FOF wavefunction . . . . .	25
3.1	Block Diagrams of ripple carry adders for conventional binary arithmetic . . . . .	41
3.2	Block Diagram of Part of the SD adder/subtractor . . . . .	43
3.3	The Structure of a 4x4 NMM . . . . .	45
3.4	The Structure of a 4x2 AMM . . . . .	45
4.1	Current Source Circuit Symbol and Schematic . . . . .	56
4.2	Current Mirror Circuit Symbols and Schematics . . . . .	57
4.3	Threshold Detector Circuit Symbol and Schematic . . . . .	58
4.4	Bidirectional Current Input Circuit Symbol and Schematic . . . . .	59
4.5	Bidirectional Current Input I-V characteristics . . . . .	59
4.6	Block Diagrams of Negative differential Resistance . . . . .	61
4.7	Schematic representation of a neuron MOS transistor . . . . .	63
4.8	Functional Diagram of DAD Logic Quantiser Operation . . . . .	65
4.9	Stationary state solutions for the Harmonic Oscillator Potential . . . . .	68
5.1	A Simplified Structure of a DSP Processor . . . . .	80
5.2	Block Diagram of the IIR Sinusoidal Generator . . . . .	87
5.3	Schematic Diagram of the 5 level Booth Multiplier Module . . . . .	89
5.4	Block Diagram of Sinusoidal Generator . . . . .	97
5.5	Block Diagram of the Barrel Shifter . . . . .	98
5.6	Schematic Diagram of a T flip-flop . . . . .	99
5.7	Schematic Diagram of first T flip-flop for mod counting . . . . .	99
5.8	Schematic Diagram of a modulo 21 counter . . . . .	100

---

5.9	Schematic Diagram of a Synchronous Reset Flip-Flop . . . . .	103
5.10	Comparison between Sine Function and CORDIC Simulation . . . . .	105
5.11	Error Difference between Sine Function and CORDIC Simulation . . . . .	106
5.12	Photomicrograph of the CORDIC Cosine Chip . . . . .	108
6.1	The FOF Wavefunction . . . . .	112
6.2	Power Spectrum of the FOF Wavefunction . . . . .	114
6.3	Simplified CSOUND FOF Implementation . . . . .	116
6.4	Time domain behaviour of FOF envelope generator . . . . .	118
6.5	Zero-Pole plot of the FOF envelope generator . . . . .	119
6.6	Heterodyned version of the FOF Algorithm . . . . .	120
6.7	Time domain and z-plane responses to Full Hanning FOF . . . . .	121
6.8	Time Domain response of the $t^2 e^{-\alpha t} \sin(\omega_0 t)$ wavefunction . . . . .	123
6.9	z-plane response to $t^2 e^{-\alpha t} \sin(\omega_0 t)$ wavefunction . . . . .	123
6.10	Magnitude and Phase Spectra for the $t^2 e^{-\alpha t} \sin(\omega_0 t)$ wavefunction . . . . .	124
7.1	Flowgraph of Direct Form FIR Filter Structure . . . . .	129
7.2	Second Order Coupled IIR filter Form . . . . .	131
7.3	Simple Impulse Generator Schematic . . . . .	131
7.4	Amplitude vs Frequency for the sinc function . . . . .	132
7.5	Reduced Dynamic Memory Storage Polyphase Filter . . . . .	136
7.6	Block Diagram of $t^2 e^{-\alpha t} \sin(\omega t)$ based formant synthesiser . . . . .	140
8.1	Block Diagram of the controller for idle/running status . . . . .	145
8.2	Block Diagram of Resource Filter Allocation . . . . .	146
8.3	Schematic of a Bipolar Envelope Generator . . . . .	149

---

## List of Tables

---

3.1	Multiples of the multiplicand to be added after scanning a triplet of multiplier bits in an overlapped pairwise scanning system . . . . .	48
3.2	The Booth Multiplier Radix-2 String Recoding Algorithm . . . . .	49
5.1	Input-Output Functions for CORDIC Modes . . . . .	93
5.2	CORDIC Sine Chip Input-Output Connections . . . . .	106
5.3	Current Consumption of CORDIC Sine Chip . . . . .	107
6.1	Comparisons between Formant Filter Topologies . . . . .	126
8.1	Decision Table for Bipolar Envelope Ramp Generator . . . . .	148

---

## List of Abbreviations

---

ADSR	Attack Decay Sustain Release
AES	Audio Engineering Society
ALU	Arithmetic Logic Unit
AM	Amplitude Modulation
ASIC	Application Specific Integrated Circuit
CCRMA	Center for Computer Research in Music and Acoustics
CMOS	Complementary Metal-Oxide Silicon
CORDIC	COordinate Rotation DIgital Computer
DAC	Digital Analogue Converter
DCA	Digital Controlled Amplifier
DCF	Digital Controlled Filter
DCO	Digital Controlled Oscillator
DFT	Discrete Fourier Transform
DIL	Dual In Line
DSP	Digital Signal Processor
ES2	European Silicon Structures
FFT	Fast Fourier Transform

---

FIFO	First In First Out
FIR	Finite Impulse Response
FM	Frequency Modulation
FOF	Forme d'Onde Formantique
FPGA	Field Programmable Gate Array
HP	Hewlett Packard
IEEE	The Institute of Electrical and Electronic Engineers
IIR	Infinite Impulse Response
IMEC	Inter-Universitair Micro-Elektronica Centrum
IRCAM	Institut de Recherche et Co-ordination Acoustique/Musique
LPC	Linear Predictive Coding
MAC	Multiply-Accumulate unit
MIDI	Musical Instruments Digital Interface
MOSFET	Metal-Oxide Silicon Field Effect Transistor
MVL	Multiple-Valued Logic
NDR	Negative Differential Resistance
RNS	Residue Number System
ROM	Read Only Memory
SD	Signed Digit
SHARC	Super Harvard ARchitecture Computer
SQUID	Superconducting QUantum Interference Device
STL	Simulation and Test Language

---

---

TTL	Transistor Transistor Logic
VHDL	VHSIC Hardware Description Language
VLSI	Very Large Scale Integration or Integrated circuits
VOSIM	VOcal SIMulation

---

---

# CHAPTER 1

## Background and Structure

---

### 1.1 Introduction

Since the dawn of time, mankind has used materials from the Earth to mould into instruments to hit, pluck, stroke or blow. Over the aeons, man has harnessed new developments in Science and Technology to transform or create new tones and new playing techniques. Initially, these came through material science, as in the tones created from different types of wood and varnish. Now, in the 20th century the ubiquitous silicon chip has taken the limelight.

Man uses VLSI technology either to make a facsimile of an instrument by modelling the instruments' properties or to create new ones which require new performance criteria to control them. Research into this field requires a multidisciplinary approach utilising the skills of musicians, physicists and computer scientists.

At the dawn of the millennium, musicians now have a plethora of different synthesis tools to generate unique sounds, but the commercialisation of these algorithms has been relegated to software implementation. Most manufactures have only implemented the easy algorithms, such as wavetable, subtractive and Frequency Modulation synthesis. They are becoming reluctant to spend any more man years and money on ASIC design, which will never repay their initial investment due to its small customer base. However, most manufactures have jumped onto the multimedia concept. This leaves musicians with a quandary on whether to buy similar hardware instruments or to go down the software extensibility route. The latter approach seems good value due to its upgradability, but this assumes that computational resources are fixed. In reality, algorithms consume valuable computation



thus slowing down the synthesis engine to near or below real time. It is difficult for musicians, who normally rely on their tactile and auditory responses, to respond in the same way to purely acoustically based sounds.

## 1.2 Aims of this Thesis

This thesis looks at the mapping of a parametric additive synthesis technique to a specialised hardware implementation, in order to discover an optimal implementable approach built around a recursive filter with the minimum number of arithmetic and storage operations. The parametric additive synthesis technique is simply a formant based model which reduces the number of parameters needed to model a sound, and is applicable because most instruments resonate. The alternative method would be a sinusoidal based additive synthesis which has an explosion of controllable parameters and would be difficult to implement.

The standard approach to digital CMOS design has been shown to be effective in reducing design times predominantly because digital is easier to work with than analogue. In this thesis, I explore different forms of digital structures by increasing the number of encodings per wire. To this end, current mode CMOS logic and negative differential resistance devices are discussed. My novel theoretical approach uses a constant number of transistors to generate many discrete levels. This is closer to the behaviour of negative differential devices and should consume less power.

A comparison between a bit-serial sinusoidal generator and a CORDIC parallel based sinusoidal generator is made, and they are found to be of comparable size. However, despite taking 21 clock cycles to generate a sinusoid, the CORDIC version is always stable. The IIR filter coefficients sets the frequency of oscillation and is sensitive to the pole quantisation of its transfer function. This form takes 25 clock cycles to perform one filter operation, with both adders and multipliers operating in parallel.

The standard FOF wavefunction approach is very greedy on memory and arithmetic resources. A filter based approach overcomes this problem. The filter structure chosen has 4 zeros in complex conjugate form on the real axis and 3 poles in complex conjugate pair form. This is radically different to any other FOF form.

Octavation can be performed by using two phase accumulators plus an adder. This forms an idealised impulse generator running at a high sampling rate. The impulses are decimated to a lower sampling rate, using an optimal two stage decimator. Using polyphase filters and FIR symmetry, a novel coefficient storage reduction scheme is proposed.

---

The filters are controlled by an allocation system which utilises decrementing counters and a linked list scheduler. The need for synthesis rules, envelope generation, subtractive synthesis modules and intuitive user interface will all be highlighted.

### 1.3 Summary of Thesis

The structure of this thesis is as follows :-

Chapter 2 presents a review of common audio synthesis techniques for music generation. It highlights the subtractive synthesis approach based on modular components progressing through to sinusoidal additive synthesis. The remainder of the chapter looks briefly at physical modelling and speech inspired synthesis techniques based on formant modelling.

Chapter 3 examines the multitude of approaches to number representation and the effect of these on numerical computation. Both of these are of paramount importance in VLSI DSP design and implementation as they enable designers to meet the performance criteria of the chosen algorithm. Conventional and redundant number representations are discussed and their merits in addition and multiplication are highlighted. Finally, array and Booth recoded multipliers are discussed.

Chapter 4 discusses different material structures for DSP, highlighting multivalued logic architectures using current mode logic and voltage-mode logic paradigms. Multivalued logic, based on negative differential resistance based devices, plus a new concept which I have invented, are also presented.

Chapter 5 expands on Chapter 3 and deals with hardware implementation requirements necessary for audio synthesis and discusses wordlength issues caused by addition and multiplication. These increases in wordlength can be minimised using truncation or rounding schemes. Problems caused by noise are also addressed. The rest of the chapter then applies this knowledge to a comparison of two sinusoidal oscillators implemented using either a bit-serial filter or a CORDIC based algorithm.

Chapter 6 examines a formant based synthesis technique by modelling the time-domain impulse response of a second order filter. This creates a damped sinusoid having no attack and the problem is overcome by looking at the FOF methodology. The method is then discussed in terms of implementational metrics. Implementation of this structure is then approximated by various filter topologies and some unusual waveform generators are highlighted.

Chapter 7 chooses from Chapter 6 one filter topology with an elegant structure and its implementation is discussed. The driver to this filter is designed and optimised.

---

Chapter 8 addresses scheduling of the filters and synthesis structures to enable easy use of the technique. Rules to enable realistic simulation of singing voices are discussed as are other approaches to enhance the sonic capabilities of the synthesis model. The chapter ends with details of how the algorithm would be partitioned between hardware and software.

Chapter 9 describes the conclusions reached in this work, and suggestions for further research in this topic are discussed.

Appendix A discusses the mathematical derivation of the Laplace to Z transform.

---

---

## CHAPTER 2

# Introduction to Synthesis methods

---

Audio synthesis techniques provide a valuable playground in which to explore the route from algorithm design to a full-custom VLSI architecture. This thesis will concentrate on the synthesis of musical tones. This field has benefited from the cross-fertilisation of ideas from engineers to musicians and evolved from original work at Bell Telephone Laboratories. The work at Bell Telephone Laboratories was originally in speech compression, but in the 1960's Max Mathews created the first digital software generating musical tones known as MUSIC IV [82, 114]. The first commercially available digital music synthesiser was introduced by Yamaha in the early eighties. This instrument was based on research by John Chowning [11] of Stanford University, which used the FM synthesis technique. Before the advent of the DX series of synthesisers by Yamaha, musicians were using analogue based subtractive synthesisers of Moog, ARP and others [137]. The digital approach provided repeatability of sounds because the technology was not dependent on temperature drifting of components as was found in the analogue systems. The FM algorithm also provided musicians with bright, clean sounding timbres that were unique. However, musicians now prefer the analogue systems because they provide warm sounding timbres caused by their imperfections. Musical instrument manufacturers began to explore newer techniques in order to give the musicians better control over the sounds.

All the early digital synthesisers relied upon custom VLSI architectures to provide

---

the computational power necessary to generate multiple voices in real-time and to make the instruments affordable. Nowadays, with new general purpose DSP chips designed for sustained computation, manufacturers choose the programmable (software) and hardware approach. We have gone full-circle, from off-line compiler based software synthesis to real-time synthesis engines like the new CSOUND, based around the Analog Devices floating point SHARC DSP chip [138]. Does this mean that a thesis on VLSI and synthesis algorithms is worthwhile ? I think so because software based engines require hardware which can support the computations which must be computed within a sample interval and so require powerful new VLSI designs. The other approach is the full-custom ASIC design which has a pay-off when the number of units sold is very large. However, the latter approach requires a longer gestation period. The ASIC approach is of value for sample-rate converters [2] and mobile phones. For the music market this approach is becoming less attractive. But optimising the algorithms will make software use the available hardware resources more effectively.

This chapter will review the current synthesis techniques available today and highlight their possible hardware configurations.

## 2.1 Subtractive Synthesis

In 1964 Robert Moog constructed a transistor-based analogue voltage controlled oscillator, and presented a paper on it at the sixteenth annual convention of the AES [82]. Over a period of rapid growth, engineers realised the need for other functions to emulate instruments and make new tones. New voltage controlled functions were provided in the form of Voltage Controlled Filters and Amplifiers. The subtractive synthesis technique uses filtering to change the harmonic structure of the input which is normally of a rich harmonic content, for example, the sawtooth waveform.

---

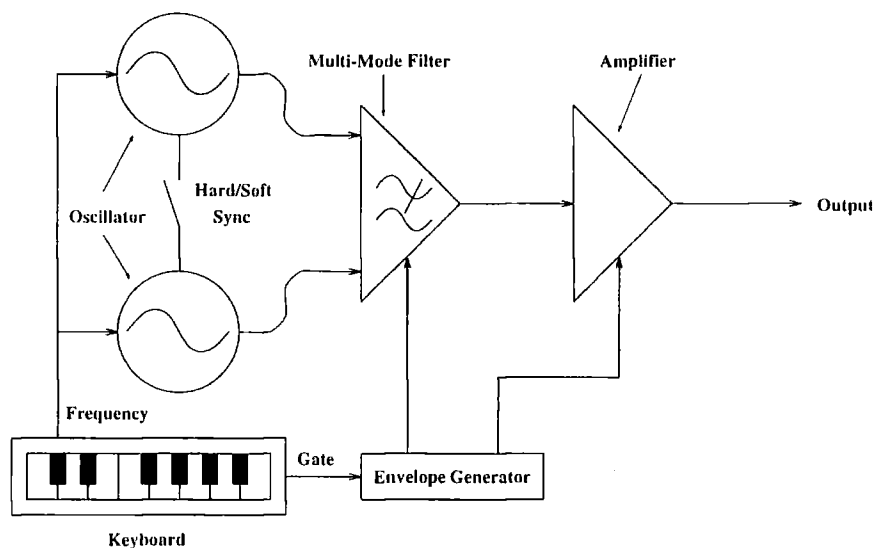


Figure 2.1: Simplified Subtractive Voice Architecture

An example of a typical single voice, dual oscillator subtractive synthesiser is shown in figure 2.1. The keyboard generates two control signals; the gate which is an on/off signal indicating the duration of the note and a signal proportional to the frequency of the note value. The gate signal drives the DCF and/or DCA via an envelope generator. The two oscillators (DCO) can either be controlled separately or chained together, thus providing a richer evolving sound. These oscillators in analogue synthesisers produce a range of waveforms, most notably, Sinusoidal, Triangular, Pulse and Sawtooth. In the digital equivalent, the musician has an infinite choice of waveform. Incidentally, modern day samplers are basically subtractive synthesisers with oscillators having user definable waveforms.

The next three sub sections will concentrate on digital hardware to implement the DCO, DCA and DCF modules in a subtractive synthesiser.

### 2.1.1 The Digital Controlled Oscillator

There are many different ways of generating an oscillator; either filter based [31, 53] or by wavetable lookup. This sub-section will discuss the wavetable based approach to oscillator generation. Essentially, wavetable based oscillators are counters being incremented by a variable representing frequency of oscillation. The output from the counter is used as an address for a ROM containing a discrete-time version of the analogue waveform. The counter can be thought of as a periodic ramp waveform, hence each ramp count is transformed by *looking up* in a table. The contents of the table (ROM) can be any shape the musician likes, not simply a sinusoid. The only

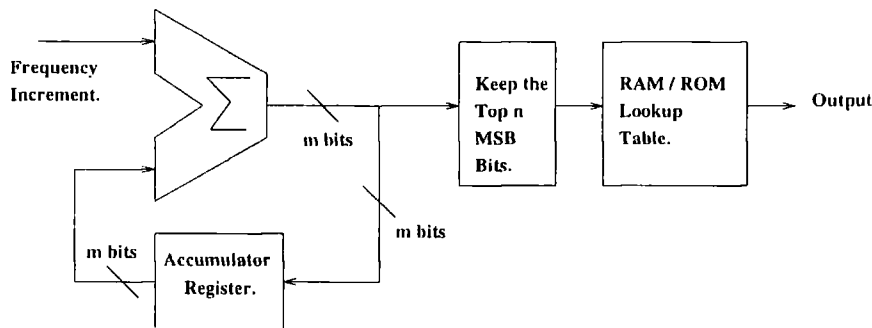


Figure 2.2: Block Diagram of a Wavetable Oscillator

proviso being that the waveform in ROM is single-cycled; so the counter provides the means of generating repetitive cycles of the waveform.

There are two types of wavetable lookup oscillators [10], one is the variable-rate form which drives the counter, ROM and the output conversion elements. This approach provides high-fidelity but is very expensive in terms of hardware. Due to the variable-rate nature, the approach cannot be multiplexed and is thus unsuitable for VLSI implementation. The other type is known as the fixed sample-rate oscillator which is also known as the phase-accumulator oscillator. A block diagram of this oscillator is shown in figure 2.2.

The phase-accumulator oscillator contains the phase angle increment and phase angle registers. The phase angle register holds the *instantaneous* phase of the waveform. Each clock interval, the phase angle contents are added by the phase angle increment and when the output of the adder overflows, the adder performs a modulo operation. This operation naturally occurs when the phase angle is greater than  $2^m - 1$ , and  $m$  is the number of bits in the adder. The clock signal is generated via a crystal for stability. This technique requires interpolation when the sample falls between two storage locations. Failure to correct for this non-integer sample will manifest itself as audio distortion. In figure 2.2 we have a truncating oscillator, the  $m - n$  bits can be thought of a fractional part, used if necessary for interpolation. In silicon, interpolating oscillators increase the chip real-estate and hence requires thought to whether the expense is justified. However, this oscillator is suitable for VLSI implementation because the adder can be time-division multiplexed, providing many oscillators. Multiple registers are required, but in VLSI the adder consumes more space. By providing a system clock which is  $k$  times the sampling rate and logic which can operate at this speed,  $k$  oscillators can be generated at the sample rate.

The frequency of the waveform,  $f$  given the phase angle increment,  $I$  is

$$f = f_s \frac{I}{L}$$

where  $f_s$  is the sampling rate and  $L$  is the table length. What is now required is the number of bits in the adder to provide sufficient resolution for musical applications. Using the method as presented in Snell's paper [127] the size of the adder is given by equation 2.1

$$b = 1 + 3.32 \log_{10} \left( \frac{F_{\max}}{F_{\min}} + 1 \right) \quad (2.1)$$

Where  $F_{\max}$  is the maximum desirable frequency of any sinusoidal component and  $F_{\min}$  is the minimum step or change in frequency. Due to work done by many scientists the values for these parameters are 20 kHz and 0.03 Hz respectively. Substituting into equation 2.1 gives an adder word length of 21 bits. This is the minimum required resolution necessary for musical purposes. If  $F_{\max}$  is a power of two, then the true frequency of the oscillator can be read directly from the phase angle increment without using any extra hardware/software, providing a direct human readable value.

An expression which relates  $b$  to the minimum interval the oscillator can be tuned to, requires knowledge of the non-linear equally tempered musical scale. The minimum interval is normally expressed in cents and a cent is 1/1200 of an octave. One octave is a doubling/halving by a factor of two. The minimum interval (or tuning accuracy) about a particular note frequency  $F$  is given below.

$$C_{\min} = 3986.31 \log_{10} \left( \frac{F_{\max}}{F(2^{b-1} - 1)} + 1 \right) \quad (2.2)$$

It can be shown that various values for  $F$ ,  $F_{\max}$  and  $b$  give non-linear values for  $C_{\min}$  by using equation 2.2. Therefore to achieve adequate resolution across the entire audio spectrum, the designer must choose a value for  $b$  which provides better than 1 cent tuning resolution at the *lowest* frequency the oscillator will run at.

As mentioned earlier, this type of wavetable lookup generates distortion if the sample value in the phase accumulator is not integer, i.e. a non zero fractional part. Thus the correct sample required is in between two memory locations. What is required is a mathematical expression that predicts the noise as a function of the size of table. An empirical result was discovered by Moore [94] and the Signal-to-Noise ratio was found to be  $6(k - 2)$  dB for a truncating oscillator and  $6(k - 1)$  dB for rounded (to nearest integer) oscillators. The  $k$  and  $k + 1$  being the bits of precision in the oscillator (not including the sign bit) for each result respectively. It is also assumed



that the table has  $2^k$  entries. So, if we do not want interpolated wavetables, the designer must choose a larger wavetable to reduce the noise generated by the lookup process. Moore also found that for sinusoidal interpolation there needs to be  $2(k-1)$  bits of precision (excluding the sign bit) in a table of  $2^k$  entries to provide a SNR of 64 dB. The interpolation formulae for sinusoidal interpolation can be as follows : take  $I$ , the integer part and  $f$ , the fractional part of the phase angle respectively.

$$\sin(I + f) = \text{sintable}[I] + f(\text{sintable}[I + 1] - \text{sintable}[I])$$

An alternative form using the approximation  $\sin(x) \approx x$  for small  $x$  expressed in radians is

$$\sin(I + f) \approx \text{sintable}[I] + f\text{costable}[I]$$

So far, the analysis has not taken into account the simplifications that can be made for symmetrical waveforms like sinusoids. Since a sine function is symmetrical about  $180^\circ$  and the first quarter cycle is symmetrical about  $90^\circ$  storage space can be reduced by a factor of 4. The savings in memory as the table length increases make this approach very attractive.

For the  $180^\circ$  approach, the designer will implement half the table size and the sign bit would control a two's complement inverter, to convert the positive table value to its negative. The designer would normally need to add one to this result to actually implement two's complement, but this can be assumed to be negligible.

In the  $90^\circ$  reduced table method, the most significant two bits are used to decide on whether to complement and/or scan the table. The most significant is the sign bit as usual. The next bit determines the negation of the address for the table and thus performs backward scanning. The remaining bits are the address for the ROM table lookup. If the penultimate most significant bit is non-zero, this will alter the address bits to the ROM by two's complementing them. However if the address bits are zero at the same time as this control bit is active (1), the phase angle is either  $90^\circ$  or  $270^\circ$  exactly and the complement would give an incorrect address for zero. This special case is overcome by substituting the maximum address instead, and is accomplished by *oring* the carry bit of the two's complementer with the address bits.

### 2.1.2 The Digital Controlled Amplifier

The DCA is nothing more than a multiplier. However, for the VLSI designer, the choice becomes tricky because the larger the wordlength the more area is used (in

parallel multiplier circuits). The observant reader might wonder if a floating point multiply would be beneficial. Unfortunately, for commercial uses, a multiplier of this type is excessively expensive in terms of design time and floor space. It is worth noting that floating point multipliers actually truncate the mantissa part, just like integer based multipliers, primarily to reduce the number of bits to wire. This can be shown clearly by the following analysis. Say, the musician wants to multiply a 16 bit number by an 8 bit amplitude envelope, the multiplier would need to have  $16 + 8 = 24$  bits output. However, if this multiplier is going to be used again, the designer would need to provide sufficient headroom, otherwise saturation and an incorrect answer will result.

2.1.3 The Digital Controlled Filter

The DCF provides gross modification of the harmonic structure of the input by either boosting or attenuating user specifiable frequencies. This module is the most expensive to implement in silicon because filtering in the discrete-time domain requires numerical operations and storage elements operating at the sample rate. In analogue synthesisers, the filter is relatively simple to implement because the arithmetic calculations are implemented directly using active and passive electronic components [139]. In the digital domain, the designer must choose on the arithmetic format for the computations and then construct the operators accordingly.

There are many different types of digital filters, where their topologies influence ease of implementation and accuracy. In this introductory chapter the second order state variable digital filter will be used as a case study.

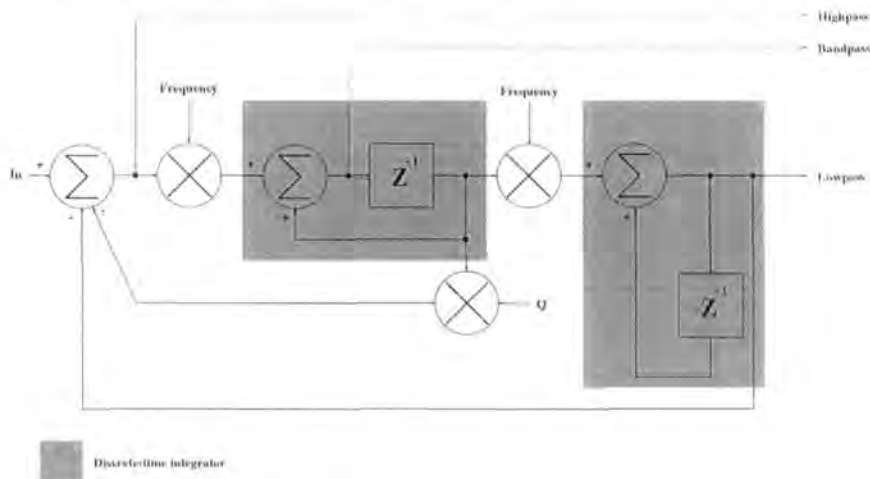


Figure 2.3: Second Order Digital Filter

As shown in figure 2.3 the digital filter is quite complicated as it has addition, multiplication and delay ( $z^{-1}$ ) elements. This filter is composed of three elements; a mixer/adder and two integrators in cascade. This filter has a  $-12$  dB/octave roll-off in each of the output modes (lowpass, highpass and bandpass). The frequency parameter is defined as  $2 \sin(\frac{\pi F}{f_s})$  and the Q (Quality factor) or bandwidth parameter is  $bw/f$ , where  $bw$  is the bandwidth and  $f$  is the desired frequency cut-off for the high and lowpass modes of the filter. This filter topology has the main advantage that the two variables, frequency and Q are independent. Note that the Q factor in the diagram is actually the reciprocal of the real Q factor.

As mentioned earlier in section 2.1.1 it is possible to implement an oscillator using filters. To implement a decaying oscillator, the first frequency parameter is multiplied by  $-1$  and the first adder is modified to have no input and the other inputs are transformed into add operators. To implement a sinusoidal oscillator, the user sets infinite Q; hence the Q parameter, as shown in the diagram, becomes zero. A final note, for constant ring time oscillation, set Q to zero, set zero input and replace the feedback connection of the first integrator by a factor being  $-\exp(\frac{1}{F_s D})$  and  $D$  is the time taken for the oscillation to decay to 37% of its value <sup>1</sup>.

In chapter 5 the complexities of digital filter implementations will be discussed in terms of the addition, multiplication and delay VLSI primitives. There are two main approaches to implementing these filters in ASICs; either by using bit-serial or bit-parallel techniques. The latter uses more floor space and therefore is more expensive to manufacture.

## 2.2 Additive Synthesis

Additive synthesis is a technique whereby complex waveforms can be decomposed into simpler waveforms ( normally sinusoidal ) and regenerated by adding the components together. Any waveform can be decomposed into a sum of simpler waveforms provided that the components belong to an orthogonal basis. In the simplest case the components are  $\sin(x)$  and  $\cos(x)$ . An orthogonal basis function means the  $\sin(x)$  is  $90^\circ$  out of phase compared to  $\cos(x)$ . Mathematically, a set of functions  $\{f_n(x)\}_{n=1}^{\infty}$  is said to be an *orthogonal system* with respect to the non-negative weight function  $w(x)$  on the interval  $[a, b]$  if

$$\int_a^b f_m(x) f_n(x) w(x) dx = 0, \quad \text{whenever } m \neq n$$

---

<sup>1</sup>The amplitude falls to 8.7 dB.

---

The technique was pioneered by the physicist Joseph Fourier during his research on heat flow and the results were published widely, especially in “Théorie Analytique de la Chaleur” (Analytical Theory of Heat) in 1822. The Fourier series is defined as follows :-

Let  $f$  be a piecewise continuous function on the interval  $[-T, T]$ . Then the Fourier series of  $f$  is the trigonometric series

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \left\{ a_n \cos\left(\frac{n\pi x}{T}\right) + b_n \sin\left(\frac{n\pi x}{T}\right) \right\} \quad (2.3)$$

The coefficients  $a_n$ 's and  $b_n$ 's are given by the following formulas

$$a_n = \frac{1}{T} \int_{-T}^T f(x) \cos\left(\frac{n\pi x}{T}\right) dx \quad n = 0, 1, 2, \dots \quad (2.4)$$

$$b_n = \frac{1}{T} \int_{-T}^T f(x) \sin\left(\frac{n\pi x}{T}\right) dx \quad n = 1, 2, 3, \dots \quad (2.5)$$

In equation 2.3, the continuous time form has been used. In the digital domain, the Fourier series becomes periodic due to the sampling process and integrals become summations. Also, calculations of the coefficients in equations 2.4, 2.5 are simplified by using complex arithmetic. All the DFT does is to correlate the data with sinusoids to see how much of a particular harmonic is contained within the signal. Once the harmonics are found, resynthesis of the signal can be performed by using equation 2.3.

A relative to sinusoidal additive synthesis uses what are known as wavelets. A wavelet is a time-domain and frequency-domain limited function. In [38], Guillemin and Kronland-Martinet discuss the relationships of acoustical signals in terms of wavelets. This modern area of additive synthesis has advantages in signal reconstruction (resynthesis) due to its resemblance with multirate DSP techniques. However, formant based synthesis techniques do have many similarities with this approach and to a musician these are much more intuitive to use.

There are three methods to implement sinusoid additive synthesis. They are time-division multiplexed fixed sample-rate wavetable oscillators, inverse FFT and multirate additive synthesis.

### 2.2.1 Time-Multiplexed Wavetable Oscillators

This approach is the logical extension of a fixed sample-rate, wavetable oscillator, whereby inserting registers at key locations allow the hardware to perform multiple oscillator lookups by utilising pipelining in the adder and ROM output stages. The

works by [10, 58, 86, 127] utilise this concept. All the oscillators run at the highest sample-rate and the maximum number of oscillators that they are capable of is about 256. The oscillators are then multiplied by an amplitude factor and summed together.

An extension to this scheme reduces the data requirements, by storing the sinusoid as a piecewise linear function with interpolation. To overcome certain limitations caused by the straight-line segment approximation, Gaussian noise is injected into the audio stream to mask the audible artefacts. The storage requirements are just 65 elements compared with 65536 in a normal lookup table oscillator without symmetry logic. A chip implementing 127 sinusoid oscillators using  $1\mu\text{m}$  CMOS technology in an area of  $27\text{mm}^2$  was created by workers at Sheffield University [47].

Another method will be discussed in chapter 5, which is based on the CORDIC iteration algorithm. This technique improves on the previous memory reduction algorithm by almost 70 %

### 2.2.2 Inverse FFT-based Oscillators

This technique utilises the Fast Fourier Transform to generate a complex waveform from the amplitude and phases of the individual harmonics. It is only feasible because there are algorithms which perform the DFT very quickly by noting symmetry caused by the discrete-time nature of the signals and the sample length.

The technique uses the overlap-add methodology because the FFT is a periodic function and the signal is time varying. The actual approach uses the short-time FFT. The number of oscillators the musician requires imparts directly on the computational cost and delay before hearing the signal. If the algorithm is based on a 1024 point radix-2 FFT, any frequencies which are not integer multiples of the data length will require interpolation.

This system can be built purely in software [34] and will eventually be commercialised.

### 2.2.3 Multirate Additive Oscillators

This approach partitions the oscillators in a tree which depends on the maximum oscillator frequency within a particular sub-band. The oscillators are based on the table lookup scheme *but* their sample clocks are not operating at a fixed sample-rate, rather they operate on factors of the output sampling-rate. This approach optimises resolution of the oscillators and efficiency by noting the hierarchical nature of music [104].

---

The oscillators in each sub-bank will have different frequency resolutions and the frequency control word in the phase accumulator will need some conversion algorithm for the benefit of the musician. The oscillators operating at the lowest frequency will be up-sampled in stages until they have been up-sampled to the audio sampling rate. At each sub-bank the oscillators running at that frequency are added and up-sampled up through the tree.

The system will require clever scheduling techniques and sophisticated user interfaces to operate [107, 108]. It has the potential of generating thousands of sinusoids using current ASIC design methodologies.

### 2.3 Frequency Modulation

FM synthesis uses standard wavetable hardware as described earlier in this thesis. Its application to musical tone synthesis can be traced back to John Chowning in 1973 [11]. He used the technique to generate audio spectra where both carrier and modulating frequencies are in the audio band. Thus the side-bands generated by the process appears in the audio band and these form the spectrum. This technique was sold to Yamaha and the first digital synthesiser, known as the DX7, was born.

In FM, the instantaneous frequency of the carrier wave is varied by the modulating wave. The rate at which the carrier varies is the frequency of the modulating wave. The variation of the carrier, known as the peak frequency deviation, is proportional to the amplitude of the modulating wave. The equation describing the process, having a carrier frequency,  $c$ , modulation frequency,  $m$  and peak deviation,  $d$  is shown in equation 2.6

$$e = A \sin(2\pi ct + \frac{d}{m} \sin(2\pi mt)) \quad (2.6)$$

When  $I = \frac{d}{m}$  is zero, there is no modulation. When  $I$  is greater than zero, sidebands appear above and below the carrier and are separated by  $m$  Hz. The larger  $I$ , the modulation index, the more sidebands appear and the carrier amplitude reduces. The amplitudes of the carrier and sidebands are determined by Bessel functions of the first kind. These functions naturally arise from problems having circular symmetry.

The total bandwidth of the signal generated by FM is approximately  $2(d + m)$  Hz. The sidebands can be derived from the expansion of equation 2.6, where  $J_n(I)$  is the  $n$ 'th order Bessel function.

$$\begin{aligned}
e = & A\{J_0(I) \sin(2\pi ct) \\
& + J_1(I)[\sin(2\pi t(c+m)) - \sin(2\pi t(c-m))] \\
& + J_2(I)[\sin(2\pi t(c+2m)) + \sin(2\pi t(c-2m))] \\
& + J_3(I)[\sin(2\pi t(c+3m)) - \sin(2\pi t(c-3m))] \\
& + \dots \}
\end{aligned}$$

It can be seen that by varying few parameters, a rich time-varying spectrum can be produced. Chowning has been able to make quite realistic analogs of trumpets, violins and singing voices using this non-intuitive method.

FM synthesis can use non-sinusoidal carrier and/or modulation waves to produce a richer spectrum. The only caveat being the bandwidth must not exceed the audio bandwidth, otherwise aliasing will result.

## 2.4 Data Reduction for Synthesis

In this section, I will briefly discuss techniques used to reduce the number of parameters to describe timbres. This is of paramount importance in all synthesis engines because of finite computational resources. So it is important to know how much data is necessary to produce high quality audio synthesis of sound.

This aspect of sound synthesis is really beyond the scope of this thesis, as it is concerned with the analysis of sampled sound into its harmonic structure. These results are subjected to further analysis to reduce the parameters necessary so that on resynthesis a close facsimile to the original can be generated.

There are various approaches to this transformation and all are dependent on clustering of data into regions of similarity and the condensing of these areas into a reduced parameter space. Two important methods are known as Principal Component Analysis and Group Additive Synthesis. Principal Component Analysis is a statistical technique used to recast a correlated matrix of data into a set of orthonormal basis vectors subject to the minimisation of the sum of the mean-square differences between the original and reconstructed data sets. In [118], Sandell and Martens described this approach for additive synthesis of sound and reported that nearly identical resynthesis to original tones for cello, trombone and clarinet can be generated with a 40 - 70 % data reduction. In Group Additive Synthesis [101], a more complex analysis is involved which is NP-complete and thus intractable.

---

However, the data can be reduced by using methods derived from Computer Science research for amplitude envelopes. This scheme is still under active exploration and ongoing research.

In [46], different optimization methods were applied to piecewise-linear envelopes in order to reduce the data needed to describe timbres. Horner and Beauchamp used gradient search and genetic algorithm techniques and found that genetic algorithms were the best but were computationally expensive for real-time analysis/synthesis systems. The use of shared breakpoint times for harmonic amplitude envelopes enabled additive synthesis to be mapped onto wavetable interpolation schemes. However, the authors do not mention what synthesis engine should be used. In addition to this, work has been undertaken to apply a reduced data set to various synthesis models [45] for the Trumpet, Tenor voice and Chinese Pipa tones. But these require a great deal of human intuition to work efficiently. In my view this is because the data set was designed for one particular synthesis technique; therefore a model of the parameter relationships between these two methods would be required.

In this section, I have shown that some analysis would be required to provide synthesis techniques in order to model and use instrument timbres which already exist in the physical world. However, for new timbres and effects, the assumptions which can be applied to data reduction methods no longer apply. In this regime unusual audio textures can be produced by using reduced parameter spaces.

## 2.5 Waveguide Modelling

This method tries to model the physical world by solving partial differential equations using digital techniques. Thus digital waveguides are used. These are the discrete-time equivalents to transmission lines having two 'arms' providing the forward and backward propagating wave solutions to a pipe. Oscillators [124] can be made.

The primary aim of this technique is the concept that by modelling the real world the articulation of the digital simulations will provide realistic timbres and smooth parameter updates. However, this is normally offset by the man-machine user interface being different to the instrument being modelled and thus an interface model must be designed.

Currently, most waveguide based synthesis systems are monophonic because of the enormous memory and arithmetic operations required to perform the synthesis. A good introduction on this subject can be found in [126].

---



## 2.6 Speech Modelling

The most interesting of all the synthesis methods are the ones which are based on human speech emulation. The human voice is acknowledged as being the most versatile of all sound producing instruments and is capable of producing drum and wind instrument imitations, as well as ventriloquists and impersonators. The quest for speech synthesis has been man's goal for many centuries. Mankind's fascination for talking machines was first encountered in Ancient Greek mythology and more recently in Cervantes' "Don Quixote". The following quote sums up man's interest in speech synthesis :-

"The invention of a talking machine, and its operation in accordance with a well-considered plan, would be one of the boldest schemes to occur to the human intellect"

*Wolfgang Von Kempelen (1971)*

In this section time-domain (LPC) and spectrum (Formant synthesis including FOF) synthesis methods will be studied. Incidentally, waveguide modelling has been used in speech production [76], and is normally known as articulatory synthesis.

Instead of modelling the physical processes which produce speech, these approaches model the signal itself. This overcomes the limitations of modelling a system which is often non-linear and complex; thus simplifications and assumptions are made which result in poor emulation of the signal. However, these methods are abstract in construction, having no physically meaningful parameters and requiring complex rules to account for every perceptible nuance of the signal.

However, these engines are easy to construct and are found in many appliances including GSM phones, speech synthesisers and audio coding for broadcast. It is quite common to find that analysis of instrument's mechanics in terms of the audio signal is easier to investigate than their physical behaviour.

### 2.6.1 Time-Domain Modelling

This approach models the sample nature of the signal rather than the frequency components. In the literature there is some confusion as to what time-domain modelling refers to. I am assuming that this technique models the amplitude fluctuations as a function of time.

The aim of this technique is to model the signal to a linear all-pole IIR filter having the transfer function shown in equation 2.7.

---

$$H(z) = \frac{G}{1 + \sum_{k=1}^p a_p(k)z^{-k}} \quad (2.7)$$

Where  $p$  is the number of poles in the filter,  $G$  is the filter gain and  $\{a_p(k)\}$  are the parameters that determine the poles. Taking a short segment of the audio signal of about 20 ms and using linear prediction the coefficients  $a_p(k)$  can be determined. We model equation 2.7 by its inverse, which is a FIR filter. The analysis process works out the error between the FIR filter and the signal and minimises the sum of the squared errors and the residual signal which cannot be modelled by the filter is stored as the residual.

To reconstruct the signal, the parameters are inserted into an all-pole IIR filter and the residual signal is idealised by a noise source. Vowel sounds can be modelled by driving the filter with a periodic impulse train with a period equal to the pitch.

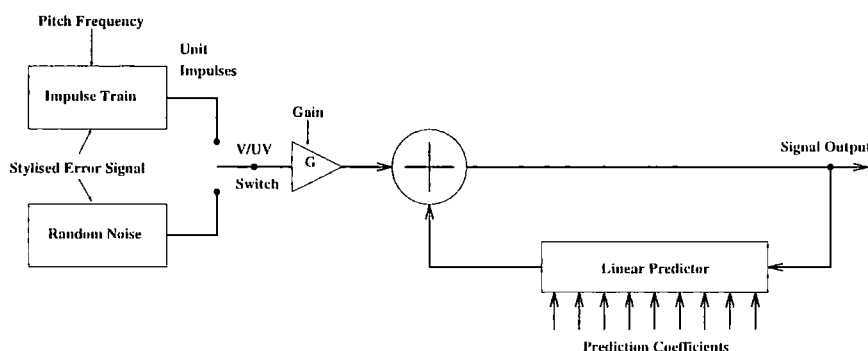


Figure 2.4: LPC Synthesiser

A schematic diagram of a LPC synthesiser is shown in figure 2.4 and by using lattice all-pole filters, the coefficients can be quantised to a few bits without loss of precision. Consequently, cheap and commercially viable systems can be made, e.g. Texas Instruments Speak and Spell. However, for music, the user requires greater precision. This difference can be traced back to speech coding for transmission over low bandwidth lines.

### 2.6.2 Spectral Modelling

In the previous sub-section the signal was approximated by modelling the time-varying signal in the time domain. Here, the signal is modelled via its frequency components. Most physical instruments, including the human voice, are composed of a cavity in which standing waves are set up. The air in these cavities thus resonates. In the speech and music fields these resonances are collectively known

as formants. This technique finds some known filter which can resonate and uses a parallel or cascade connection of them to generate a harmonically rich spectrum.

The VOSIM, parallel/cascade formant and FOF synthesisers will be discussed in the next sub-sections.

### 2.6.2.1 VOSIM

The VOSIM technique was designed to create a simulation of vocal sounds and hence was called VOcal SIMulation. It was created and developed by Werner Kaegi in 1973.

VOSIM uses a simple waveform controlled by three parameters and periodically repeated at the desired fundamental frequency. It consists of a series of  $\sin^2(x)$  pulses having a pulse-width of  $T$ , followed by a delay  $M$ . There is a decay parameter,  $b$  which is the factor by which each pulse height is reduced with respect to the preceeding pulse. The first pulse has unit amplitude. The last parameter is the number of pulses in each period. It can be clearly seen from diagram 2.5 that the period can be varied by altering  $M$ . A rigorous analysis of the waveform in both time and frequency domains can be found in [135].

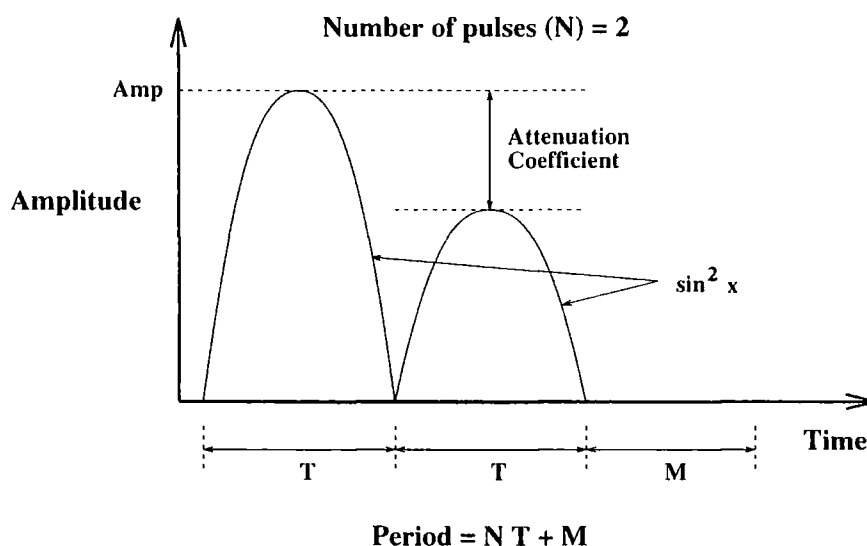


Figure 2.5: Typical VOSIM waveform

To generate format-like regions in the frequency domain, the musician will require two or more VOSIM oscillators. Each oscillator models a formant. The output of all the oscillators is summed to produce the desired effect. One major draw-back

to the technique is that it is only useful in producing tones with variable pitch but stationary spectra.

The beauty of this technique is manifest in the hardware. Each VOSIM oscillator can be computed using a slightly modified wavetable oscillator. The first pulse is generated by setting the amplitude factor to unity and scanning the table pointer forwards, thus reading the pulse at a rate dependent on the pulse-width parameter. Once the pulse has been read, the amplitude factor is multiplied by the decay parameter and the table pointer is reset and allowed to scan through the table as before. If all the pulses are generated, the wavetable oscillator will output zeros for  $M$  seconds.

### 2.6.2.2 Parallel/Cascade Formant Synthesis

The formant synthesiser uses three to five second order low pass filters, each having variable amplitude, bandwidth and frequency controls. These filters are either connected in serial (cascade) form or parallel form. They are driven like LPC with an idealised pulse train and/or noise source. The former configuration is suitable for non-nasal voiced sounds and the latter is superior for nasals, fricatives and stops. The cascade form models the vocal tract without nasal coupling and thus is a closer model to the human voice. The parallel form is an abstract model.

In the serial form, if a small number of resonant filters is to be used, then a correcting function is required. Ideally, this depends on the number of formants and their frequencies, but can be approximated by a fixed network [76]. The advantage of the serial approach is that the formant amplitudes are predictable from the knowledge of the formant frequencies and bandwidths.

However, the parallel form has greater flexibility and control as all three parameters are directly controllable. This creates two problems. Firstly, the user has yet another parameter to control; in a musical context this might be desirable to aid novel, non-physical based timbres. The second problem is potentially more serious. The phase cancellation at frequencies between two resonances may cause zeros in the frequency response when the parallel formant filters are summed together. This can be explained by considering the sum of two second-order bandpass filter functions in the continuous domain <sup>2</sup>.

$$\frac{s}{s^2 + \beta_1 s + \omega_1^2} + \frac{s}{s^2 + \beta_2 s + \omega_2^2} = \frac{s(2s^2 + s(\beta_1 + \beta_2) + \omega_1^2 + \omega_2^2)}{(s^2 + \beta_1 s + \omega_1^2)(s^2 + \beta_2 s + \omega_2^2)} \quad (2.8)$$

---

<sup>2</sup>The equation in the discrete time form is similar, except the parameters are trigonometric and exponential functions.

It can be seen that the numerator term of equation 2.8 will have complex conjugate roots between the two pole frequencies  $\omega_1$  and  $\omega_2$ . This will produce a deep minima, which can be removed by alternating the signs for each formant filter.

A general parallel/series hybrid formant synthesiser [69] is shown in figure 2.6.  $F_0$  is the pitch and AV, AVS, AH, AF, AB and A1 to A6 are amplitude factors. RGS, RNP and R1 to R6 are resonators and RGZ and RNZ are anti-resonators having variable bandwidth and frequency.

The synthesiser as shown in figure 2.6 can be simplified to a standard parallel structure and the formant filters can be factorised into common terms, this would reduce the arithmetic and storage costs.

---

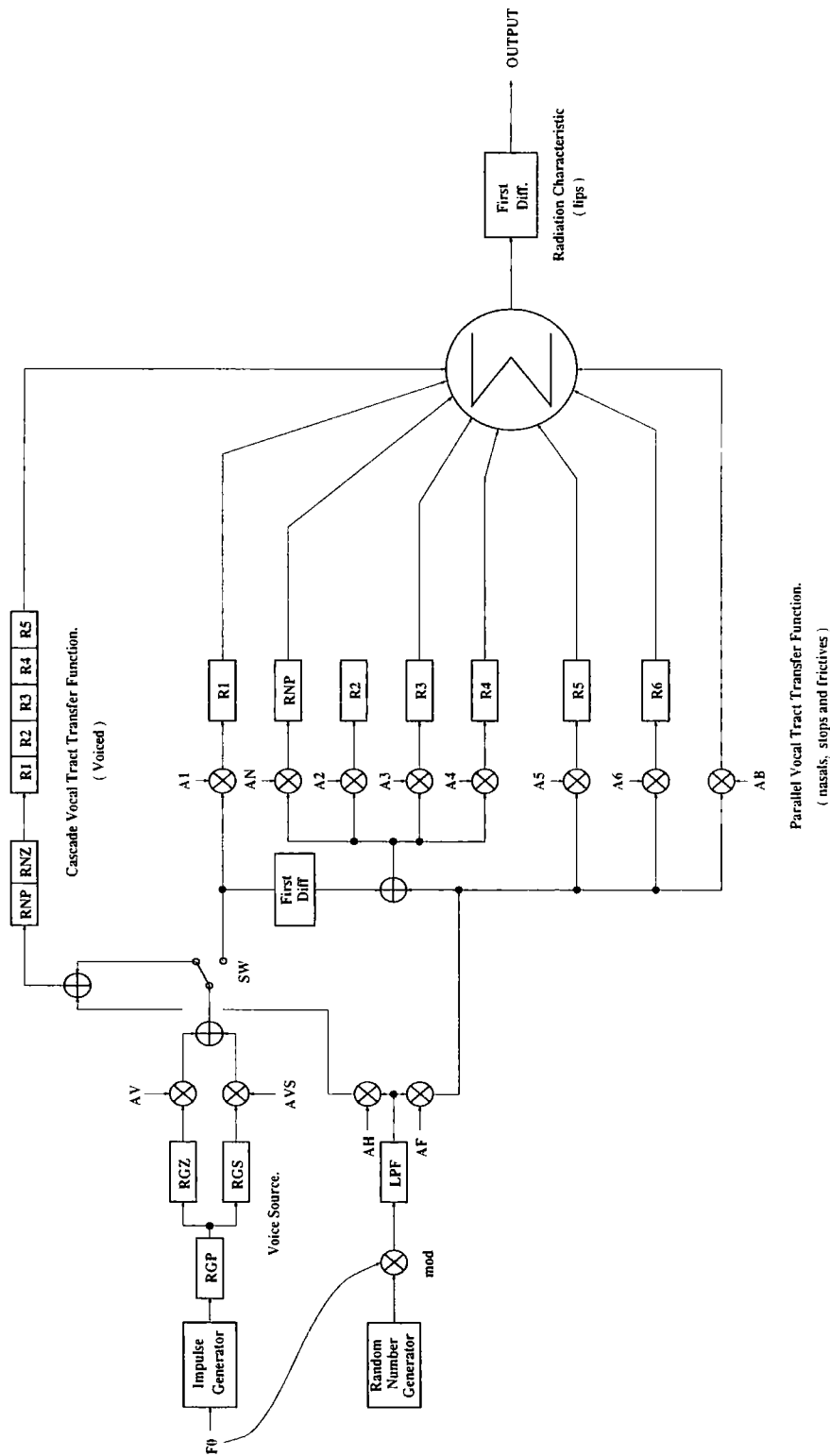


Figure 2.6: Block Diagram of Serial-parallel formant synthesiser. (After Klatt (1980) [69])

### 2.6.2.3 Forme d'Onde Formantique

FOF was developed by Xavier Rodet in 1978 and a control program known as CHANT allowed musicians to generate excellent renditions of singing voices, providing the musicians composed at IRCAM in Paris, France. Later in the development of the system, cymbals, drums and string based instruments were emulated.

The *Forme d'Onde Formantique* (FOF) algorithm is a source-filter synthesis model capable of emulating the singing voice [115]. The source is simply an impulse train and the filter is a modified decaying sinusoid (a formant). The technique is a special case of a parallel formant synthesiser because the amplitude of the signal is calculated in the time domain using exponential functions and sinusoidal wavetables. This method has been shown to require less numerical precision than an equivalent filter and is also computationally efficient [116] in integer arithmetic. Floating-point arithmetic was chosen for flexibility. There are two versions of FOF, the first being wavetable based as in CHANT and CSOUND and the other being second order filter based as in the Samson's Box version belonging to CCRMA, Stanford. The standard CHANT configuration for FOF is a parallel FOF bank connected to a common excitation signal as shown in figure 2.7

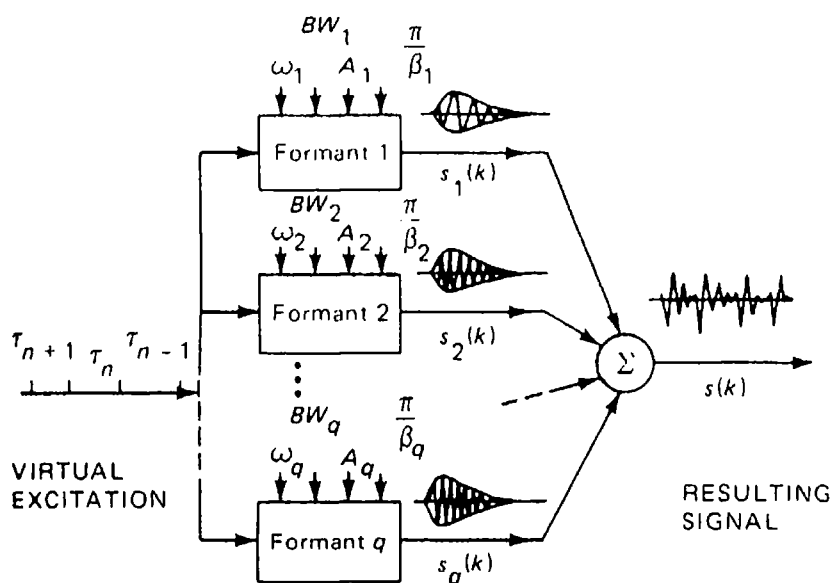


Figure 2.7: FOF synthesiser topology. (After G.Bennett & X. Rodet [6])

The FOF algorithm tries to model the formant structure in the time domain with an analytical expression. It is based on half a hanning window for the *attack* portion

of the waveform, as shown in equation 2.9.

$$\psi(t) = \begin{cases} \frac{1}{2}(1 - \cos(\beta t))e^{-\alpha t} \sin(\omega_0 t + \phi) & 0 \leq t \leq \frac{\pi}{\beta} \\ e^{-\alpha t} \sin(\omega_0 t + \phi) & t \geq \frac{\pi}{\beta} \end{cases} \quad (2.9)$$

$\alpha$  is related to the bandwidth of the *filter* and  $\omega_0$  is the angular formant frequency. By varying  $\beta$ , the *attack* portion of the waveform can be reduced or expanded with a corresponding effect in the frequency domain. This is shown diagrammatically in figure 2.8, where  $\omega_0 = 2.5\text{kHz}$  and  $\alpha/\pi = 80\text{Hz}$ .

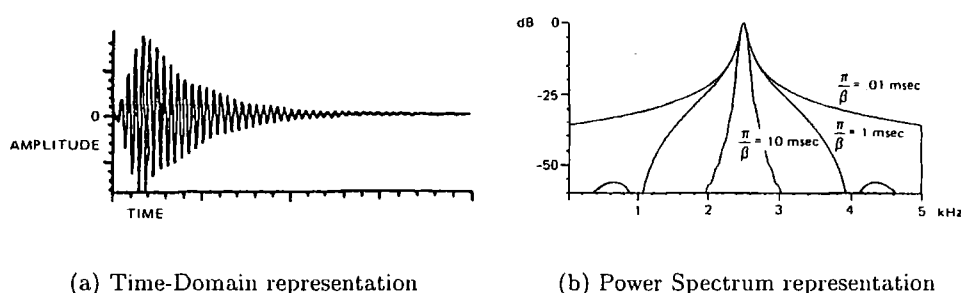


Figure 2.8: Properties of the FOF wavefunction. (After G. Bennett & X. Rodet [6])

Currently, this particular synthesis technique has escaped hardware implementation due to its immense computational cost.

## 2.7 Summary

In this chapter, various synthesis techniques have been described and hardware implementations were discussed where appropriate. In this thesis a spectral based parametric additive synthesis algorithm will be investigated. This algorithm is essentially a formant based technique belonging to the FOF family. The fundamental reasoning behind using this technique is the reduced parameters needed to describe the spectrum of the instrument. In additive, each sinusoid is placed in the audio field with amplitude and frequency. In FM, though the parameters required are small, the parameters behave in a non-linear, counter-intuitive manner. This makes it difficult for musicians weaned on analogue subtractive synthesisers to create useful timbres. The formant based systems have intuitive parameters, used in recording studios globally. The parameters are more than FM and, on an oscillator basis, more numerous than additive, but each formant oscillator is really a spread of frequency components. Having amplitude, formant frequency and bandwidth parameters the



musician can place the oscillator in the audio field and build up complex evolving timbres.

Another reason behind FOF-like formant synthesisers for VLSI implementation is the reduced parameter updates required from a host machine [61]. However, internal bandwidth inside ASIC chips is not normally a problem because the wiring is fixed during development. Interfacing a chip to the outside world is a major headache, because the number of pins on a chip determines the cost of the device and each pin carries, at most, a two-state signal and consumes a finite area.

---

## CHAPTER 3

# Number Systems for VLSI Processing Elements

---

In this chapter, different number systems will be investigated and a review of arithmetic units for VLSI will be presented. The majority of the work presented here will be based on integer arithmetic, because floating-point arithmetic can be fabricated from integer arithmetic parts. It is uncommon for ASICs to be designed with floating-point capability because floating-point arithmetic elements use huge amounts of floor space, thus increasing the chip's size and cost. Integer arithmetic is preferred due to its lower cost and higher speed. However, most off-the-shelf DSP and microprocessor chips have floating-point units because the designers know these devices are to be used in general, non-optimised applications. They also provide a simple mechanism to upgrade functionality via reprogramming of software.

### 3.1 Introduction to Number Systems

Humans use the signed decimal number system to perform arithmetic operations. Most computers use a binary number system to perform useful arithmetic computation. But can computers use a different number system which will result in faster and more efficient computation ? It seems possible that they can, as humans use of the decimal system arose from their use of fingers and thumbs, and digital computers use of binary logic arose from the on off switch. Arithmetic computation can be performed in any number system the chip designer desires. Some number systems, like

residue arithmetic, minimise the number of carries in addition and multiplication operations and consequently these operations can be performed quickly [75, 90].

There are six important number systems known; they are the conventional radix, the signed-digit, the residue, the rational, the logarithmic and the floating point. In this chapter, the first three will be discussed in greater detail. A summary of these number systems follows :-

**Conventional Radix System** A fixed radix arithmetic system with a radix  $r \geq 2$  and a digit set of

$$\{0, 1, \dots, r - 1\}$$

All the digits are positively weighted and each number is uniquely represented; e.g. decimal, which is base-10 and has a digit set of  $\{0, 1, \dots, 9\}$ .

**Signed-Digit System** In this system, both positive and negative weighted digits are allowed for each digit, and the digit set is

$$\{-\alpha, \dots, -1, 0, 1, \dots, \alpha\}$$

Where  $\alpha$  is a bounded positive integer, and will be discussed in section 3.3. This is a redundant number system in that the signed-digit representation of a number may not be unique.

**Residue System** This system has no weighting factor assigned to each digit of a residue number. The order of the digits is immaterial in determining the value of the number. It is possible for mixed radices to be assigned to different digits. In a residue number system  $X$  is an integer represented by an  $n$ -tuple equal to

$$X = \{r_1, r_2, \dots, r_n\}_m$$

with respect to another  $n$ -tuple being

$$m = \{m_1, m_2, \dots, m_n\}$$

Each  $r_i$  is called the residue of  $X$  modulo  $m_i$ , where all the  $n$  moduli  $\{m_i \mid i = 1, 2, \dots, n\}$  are pairwise relatively prime. All the  $n$  residue digits  $r_i$  for  $i = 1, 2, \dots, n$  can be **independently** processed. The system is therefore carry-free when performing addition and multiplication.

This system was invented to combat arithmetic errors in the design of reliable computing systems. It can consequently be made fault tolerant, since operations can be performed in parallel with no carries.

---

**Rational System** This system represents numeric quantities as fractions in terms of numerator and denominator integer pairs. The system always yields rational numbers and under arithmetic operations  $(+, -, \times, \div)$ , may be closed without resorting to infinite precision. i.e.,  $\frac{1}{3} = 0.3333 \dots$  The problem with this system is that the numerators and denominators can become large at the beginning of a moderately sized computation and it is still a theoretical concept.

**Logarithmic System** This system employs a real number  $\mu > 1$  as the **BASE**. The set of real numbers is defined by the following logarithmic space  $L_\mu$ .

$$L_\mu = \{X \mid |x| = \mu^i; i \text{ an integer}\} \cup \{0\}$$

This allows geometric rounding rather than arithmetic rounding which enhances the number accuracy. The system is built using integer arithmetic units and log-antilog functions.

**Floating Point System** In this system a real number is expressed as two numbers

$$f = (m, e) = m \times r^e$$

where  $m$  and  $e$  are each a signed, fixed-point number and  $r$  is a given radix. The label  $m$  is known as the **mantissa** of the number  $f$  and the label  $e$  is known as the **exponent** of the number  $f$ .

The mantissa is normally fractional in one of the conventional number systems and the exponent is a biased or unbiased  $r$ 's complement integer. The radix  $r$  is self-implied and consequently is not stored. The radix point of the mantissa can float around by adjusting the exponent, but in the normalised number representation, the point lies near the most significant mantissa digit.

Before the discussions on conventional, signed-digit and residue number formats, it should be noted that the choice of a number system for arithmetic operations is governed by three points.

- Efficiency of representation
- Facility for arithmetic design
- Reliability of operation

A designer could decide to use binary computers and implement decimal arithmetic using Binary-Coded-Decimal number representation.

---

### 3.2 Conventional Radix Number System

As mentioned in section 3.1, a radix-r number,  $X$ , can be represented in a digital computer by a **digital vector** of  $(n+k)$ -tuples

$$X = (x_{n-1}, \dots, x_0 \bullet x_{-1}, \dots, x_{-k})_r$$

where each component  $x_i$  for  $-k \leq i \leq n-1$  is called the  $i$ 'th **digit** of the vector  $X$ . The first  $n$  digits  $(x_{n-1}, \dots, x_1, x_0)$  form the *integer* portion of the number  $X$  and the remaining  $k$  negatively indexed digits  $(x_{-1}, x_{-2}, \dots, x_{-k})$  form the *fractional* portion of the number  $X$ . A **radix point** is used to divide these two portions and is not stored in the computer, but is implied to exist at a certain point !

Mixed radix numbers are those assuming different radix values in different digit positions. An example would be the time format hours, minutes, seconds; having the mixed radices of ( 24, 60, 60 ).

#### 3.2.1 Weighted Number Systems

There are various number systems within our familiar radix number system. These subclasses are based on the *weighted radix number* format. In this number format, each digital vector,  $X$  is associated with a *unique* value denoted by

$$X_\nu = \sum_{i=-k}^{n-1} x_i \cdot \omega_i$$

where each  $\omega_i$  is called a **weighting factor for the  $i$ 'th digit**. The  $n+k$  weighting factors form a *weighting vector* denoted by

$$W = (\omega_{n-1}, \dots, \omega_0, \omega_{-1}, \dots, \omega_{-k})$$

The value of the number  $X$  can be obtained by  $X \cdot W$ , the **dot product** of the two vectors  $X$  and  $W$ . The illustrative example below will be beneficial. The weight vector

$$W = (r^{n-1}, \dots, r^0, r^{-1}, \dots, r^{-k})$$

will lead to the *conventional positional representation* of a radix-r number  $X$ , with a value

$$\begin{aligned} X_\nu &= X \cdot W \\ &= \sum_{i=-k}^{n-1} X_i \cdot \omega_i \\ &= \sum_{i=-k}^{n-1} x_i \cdot r^i \end{aligned}$$


---

The higher the radix number,  $r$ , the more radix  $y$  digits are required to encode each radix- $r$  digit ( where  $y \neq r$  and  $y < r$  ) <sup>1</sup> Therefore if the ASIC designer wishes to encode decimal, then  $r = 10, k = \lceil \log_2 10 \rceil = 4$  bits.

The weighted number system can be further subdivided into three common signed formats, known as **Sign-Magnitude**, **Diminished-Radix Complement** and **Radix Complement** representations.

### 3.2.2 Sign-Magnitude Representation

This format denotes digits of  $\mathcal{A}$  as  $(a_{n-1}a_{n-2}\dots a_1a_0)_r$ , where  $a_{n-1}$  denotes the sign bit and is zero for positive integers and  $(r-1)$  for negative numbers. The representation for sign-magnitude numbers is

$$\mathcal{A} = ((r-1)m_{n-2}\dots m_1m_0)_r$$

where  $m_i$  for  $n-2 \geq i \geq 0$  are the true magnitude digits and the magnitude equals

$$\sum_{i=0}^{n-2} m_i \cdot r^i$$

This number system has two number representations for zero and this zero vector is called a **dirty zero**.

### 3.2.3 Diminished-Radix Complement Representation

This format represents positive numbers by

$$\mathcal{A} = (0m_{n-2}\dots m_1m_0)_r$$

and negative numbers by

$$\mathcal{A} = ((r-1)\bar{m}_{n-2}\dots \bar{m}_1\bar{m}_0)_r$$

where  $\bar{m}_i = (r-1) - m_i$  for  $n-2 \geq i \geq 0$ . Therefore,  $\bar{\mathcal{A}} = r^n - 1 - \mathcal{A}$ .

The diminished radix complement representation is also known as the  $(r-1)$ 's complement. This representation has a non-unique zero vector, like the sign-magnitude representation and is thus redundant.

---

<sup>1</sup>In general for binary computers, at least  $k$  bits are required to encode a radix- $r$  digit, where  $k = \lceil \log_2 r \rceil$  and  $\lceil x \rceil$  means the least integer that is not less than the real number  $x$ .

### 3.2.4 Radix Complement Representation

This format represents positive numbers by

$$\mathcal{A} = (0m_{n-2} \dots m_1 m_0)_r$$

and negative numbers by

$$\mathcal{A} = (((r-1)\overline{m}_{n-2} \dots \overline{m}_1 \overline{m}_0) + 1)_r$$

In this notation,  $\overline{\mathcal{A}} = r^n - \mathcal{A}$ . This representation is known as the  $r$ 's complement representation.

This representation has a unique zero value and the zero vector is known as a **clean zero**. The representation is thus non-redundant.

### 3.2.5 Properties of Sign-Magnitude, $r-1$ 's and $r$ 's Complement Numbers

In all these formats, all the digits are required to be filled according to the following rules :-

- Sign-Magnitude numbers are filled with leading zeros for positive and negative numbers.
- $(r-1)$ 's complement numbers are filled with the sign's value. This is known as **sign extension**.
- $r$ 's complement numbers are filled with the sign's value and this process is known as **sign extension**,

In all these formats, overflow can occur when the positive number exceeds the upper bound and underflow can occur when the negative number exceeds the lower bound.

The beauty of the fixed-radix point number formats discussed, is that every  $n$ -digit integer can be considered as a fraction multiplied by a constant factor  $r^n$ ; conversely, every  $k$ -digit fraction can be considered as an integer multiplied by a constant factor  $r^{-k}$ . This is very useful when the designer of an arithmetic unit wants to implement the unit in fractional notation and without the extra complexity of floating-point number representations.

---

### 3.3 The Signed-Digit Number System

The Signed Digit ( SD ) number representations allow redundancy to exist and are useful in designing high-speed arithmetic machines [4]. Each signed digit may need more than one bit to represent it <sup>2</sup>. Wider data buses and an increase in data storage are required, but it is worth using the speed improvement the technique provides.

#### 3.3.1 Definition of Signed-Digit Numbers

Given a radix  $r$ , each digit of an SD number can assume the following  $2\alpha + 1$  values.

$$\Sigma_r = \{-\alpha, \dots, -1, 0, 1, \dots, \alpha\}$$

where the maximum digit magnitude  $\alpha$  must be within the following region.

$$\left\lceil \frac{r-1}{2} \right\rceil \leq \alpha \leq r-1 \quad (3.1)$$

Notice that integers are assumed to satisfy the following inequalities  $\alpha \geq 1$  and  $r \geq 2$

To yield the **minimum redundancy** in the balanced digit set  $\Sigma_r$ , one can choose the following value for the maximum magnitude. <sup>3</sup>

$$\alpha = \left\lfloor \frac{r}{2} \right\rfloor$$

Sometimes it may be useful to allow  $\alpha = r_e/2$ , where  $r_e$  is **even** and  $\alpha = (r_o - 1)/2$  when  $r_o$  is **odd**. Thus these adjacent odd and even radix values may give the *same digit set*. An example of this digit set corresponding to this choice of  $\alpha$  is shown below in two different forms; But is the same set when  $r_o = r_e + 1$ .

$$\Sigma_{r_o} = \left\{ -\frac{r_o-1}{2}, \dots, -1, 0, 1, \dots, \frac{r_o-1}{2} \right\}$$

$$\Sigma_{r_e} = \left\{ -\frac{r_e}{2}, \dots, -1, 0, 1, \dots, \frac{r_e}{2} \right\}$$

The SD number digit set for a radix-2 number is  $\Sigma_2 = \{-1, 0, 1\}$

The algebraic value,  $\mathcal{Y}_\nu$ , of an SD number

$$\mathcal{Y} = (y_{n-1} \dots y_0 y_{-1} \dots y_{-k})_r \quad (3.2)$$

<sup>2</sup>The sign bit is included in each bit.

<sup>3</sup> $\lceil x \rceil$  is the largest integer that is less than or equal to the real number  $x$ .



can be evaluated by

$$\mathcal{Y}_\nu = \sum_{i=-k}^{n-1} y_i \cdot r^i$$

Notice that there is no **explicit sign** as  $\mathcal{Y}$  can be positive and negative, also the zero has a unique expression if, and only if,  $y_i = 0$  for all  $i$  in equation 3.2.

The negation  $-\mathcal{Y}$  of an SD number  $\mathcal{Y}$  is achieved by changing the sign of all non-zero digits in  $\mathcal{Y}$ . Since zero is unique  $-0 = 0$ . The negative non-zero digits are represented by  $\bar{x}$ , where  $x$  is the positive non-zero digit before negation.

The SD number system is used because it eliminates the carry propagation chains in addition and subtraction operations. In order to break the carry chain, enabling fast addition and subtraction operations, the lower bound on  $\alpha$  should be made tighter and is expressed in equation 3.3.

$$\left\lceil \frac{r+1}{2} \right\rceil \leq \alpha \leq r-1 \quad (3.3)$$

It is known that division can use the less strict bound as found in equation 3.1.

As mentioned previously, this number system allows non-unique representations of a particular value. The **weight** of an  $n$ -digit SD vector  $\mathcal{Y}$  with value  $\mathcal{Y}_\nu$  is the number of non-zero digits in the representation and is denoted by  $\omega(n, \mathcal{Y}_\nu)$ . In general, the weight of an  $n$ -digit SD vector is defined as

$$\omega(n, \mathcal{Y}_\nu) = \sum_{i=0}^{n-1} |y_i|$$

and  $|y_i| = 1$  if  $y_i \neq 0$ .

The SD vector with the **minimal weight** is called a **minimal SD representation** with respect to given values of  $n$  and  $\mathcal{Y}_\nu$ . Later on in this thesis, the reader will see the importance of the minimal set applied to multiplier design.

### 3.3.2 Conversion Between The Conventional Radix- $r$ Number and its SD Form

Let  $X = (x_{n-1}, \dots, x_1, x_0)_r$  be a conventional radix- $r$  number and  $\mathcal{Y} = (y_{n-1}, \dots, y_1, y_0)_r$  be the equivalent SD number.

#### 3.3.2.1 Conversion from Conventional to SD Systems

For every conventional digit  $x_i$ , the **interim difference** digit  $d_i$  is generated by

$$d_i = x_i - r \cdot b_{i+2}$$


---

and the **borrow digit**  $b_{i+1}$  is defined to be

$$b_{i+1} = \begin{cases} 0, & \text{if } x_i < \alpha \\ 1, & \text{if } x_i \geq \alpha \end{cases}$$

The  $i$ 'th SD digit,  $y_i$ , is then obtained by

$$y_i = d_i + b_i$$

This conversion process has no borrow propagation and each digit is independently generated. This means that the number conversion can work in parallel or from any digit position.

### 3.3.2.2 Conversion from SD to Conventional Systems

This reverse process is achieved by adding the two SD numbers  $\mathcal{Y}^+$  and  $\mathcal{Y}^-$  together. The  $\mathcal{Y}^+$  number is formed from the positive ( non-zero ) digits of the SD number  $\mathcal{Y}$  and the  $\mathcal{Y}^-$  number is formed from the negative ( non-zero ) digits. The example below should make this process clear.

$$\begin{aligned} \mathcal{Y} &= \mathcal{Y}^+ + \mathcal{Y}^- = (1\bar{4}5\bar{2})_{10} \\ \mathcal{X} &= +(1050)_{10} - (0402)_{10} = (0648)_{10} \end{aligned}$$

This conversion performs subtraction in the conventional number system and consequently is not carry-free.

## 3.4 The Residue Number System

The Residue Number System ( RNS ) is defined in terms of a set of moduli. If  $\mathcal{P}$  denotes the moduli set, then

$$\mathcal{P} = \{p_1, p_2, \dots, p_L\}$$

The  $p_i$ 's are all integers and are pairwise relatively prime. Any integer in the residue class  $\mathcal{Z}_M$ <sup>4</sup>, where

$$M = \prod_{i=1}^L p_i$$

has a unique L-tuple representation given by

$$\mathcal{X} \xrightarrow{\text{RNS}} (x_1, x_2, \dots, x_L)$$

---

<sup>4</sup>A ring of integers in Modulo M; ie  $\{0, 1, \dots, M-1\}$

---

and  $x_i = \mathcal{X} \bmod p_i$  and  $x_i$  is called the  $i$ 'th residue of  $\mathcal{X}$ .

For a signed number system, any integer in  $(-\frac{M}{2}, \frac{M}{2})$  has a RNS  $L$ -tuple representation given by

$$x_i = \begin{cases} \mathcal{X} \bmod p_i & \text{if } \mathcal{X} > 0 \\ (M - |\mathcal{X}|) \bmod p_i & \text{if } \mathcal{X} \leq 0 \end{cases}$$

and this is a symmetric system.

The residue number system assigns all digits to have **equal** importance, unlike conventional binary systems which are weighted. If two RNS numbers  $\mathcal{X}$  and  $\mathcal{Y}$  are operated using the following arithmetic operators  $(+, -, \times)$  and the result is stored in  $\mathcal{Z}$ , then

$$\mathcal{Z} = \mathcal{X} \circ \mathcal{Y} = ((x_1 \circ y_1) \bmod p_1, \dots, (x_L \circ y_L) \bmod p_L)$$

only if and only if  $\mathcal{Z}$  belongs to  $\mathcal{Z}_M$ .

Notice that the  $i$ 'th RNS digit,  $z_i$  is defined in terms of  $(x_i \circ y_i) \bmod p_i$  and this implies that the operators do not generate any information ( carry signals ) from any other RNS digit !! The multiplication operator results in an RNS number belonging to  $\mathcal{Z}_M^2$ , if the two RNS numbers operated on belong to  $\mathcal{Z}_M$ . This number system makes possible high speed concurrent arithmetic operations and consequently is very attractive to implement in VLSI [36].

The division operator is not closed in the RNS system and consequently can only be approximated via iteration.

### 3.4.1 The Multiplicative Inverse

The multiplicative inverse is crucial in number conversions involving the RNS system and therefore an understanding of its formulation is essential [132].

For  $0 \leq a \leq P$ , if there exists an integer  $a^{-1}$  such that  $a \cdot a^{-1} \bmod P = 1$ , then  $a^{-1}$  is the **multiplicative inverse** of  $a \bmod P$ . The multiplicative inverse of  $a^{-1} \bmod P$  exists and is unique if and only if the Greatest Common Divisor of  $a$  and  $P$  is 1.

Multiplicative inverses are derived by

1. Direct Search Method.
2. Trial and Error Method.
3. Using Euler's Formula.

The next section will discuss Euler's Formula.

---

### 3.4.1.1 Euler's Formula

The 'totient function', denoted by  $\phi(p)$ , when  $p$  is larger than 1, is the number of non-zero elements in  $\mathcal{Z}_p$  that are relatively prime to  $p$ . When  $p$  equals one, then the totient function  $\phi(p) = 1$ .

If  $p$  is a prime  $q$ , then all non-zero elements of  $\mathcal{Z}_q$  are relatively prime to  $q$ , and so  $\phi(q) = q - 1$  whenever  $q$  is a prime. If  $p$  is a power of a prime, then the only elements of  $\mathcal{Z}_p$  not relatively prime to  $q^m$  are the  $q^{m-1}$  multiples of  $q$ . Therefore  $\phi(q^m) = q^m - q^{m-1} = q^{m-1}(q - 1)$

Euler's theorem states that if the Greatest Common Divisor of  $a$  and  $q$  equals one then  $a^{\phi(q)} \bmod q = 1$  and if  $p$  is any positive integer, then it also follows that  $a^{-1} \bmod p$  can be written as

$$a^{-1} = a^{\phi(p)-1} \bmod p$$

### 3.4.2 Residue to Decimal Conversion

There are two methods to carry out this task, the **Mixed-Radix Method** and **The Chinese Remainder Theorem**. This thesis will discuss the mixed-radix method first.

#### 3.4.2.1 Mixed-Radix Method

Before delving into the method some new notation will be introduced as follows :-

$$\langle a \cdot b \rangle_p \equiv a \cdot b \bmod p$$

$$c^{-1}[p] \text{ is the multiplicative inverse of } c \bmod p$$

The Mixed-Radix Coefficients are generated by

$$\mathcal{X} = \sum_{i=0}^{L-1} a_i \prod_{k=0}^i p_k$$

provided  $0 \leq a_i < p_{(i+1)}$  for all  $i > 0$  and  $p_0 = 1$ . The coefficient  $a_i$  is computed using a recursive algorithm employing intermediate variables  $S_{ij}$  with  $x = S_{00} \equiv a_0$ ;  $x_l = S_{0(l-1)}$  for  $l = 2, 3, \dots, L$  and  $S_{ii} \equiv a_i$  for  $i = 0, 1, \dots, (L-1)$ . The recursive formula is

$$S_{ij} = \left\langle S_{(i-1)j} - S_{(i-1)(i-1)} \cdot \left\langle \frac{1}{p_i} \right\rangle_{p(j+1)} \right\rangle_{p(j+1)}$$

Where  $j = i, (i+1), \dots, (L-1)$  for each subscript  $i$ .

### 3.4.2.2 Chinese Remainder Theorem

The Chinese Remainder Theorem is

$$\mathcal{X} = \left\langle \sum_{i=1}^L s_i \cdot \langle s_i^{-1} x_i \rangle_{p_i} \right\rangle_M$$

and  $s_i = \frac{M}{p_i}$  and  $s^{-1} \equiv s^{-1}[p]$ .

### 3.4.3 The Quadratic Residue Number System

The Quadratic Residue Number System [22, 133] is an encoding system which will make complex multiplication equivalent to two real multiplies, and complex addition equivalent to two real adds.<sup>5</sup> This system relies on the mathematics of Gaussian Primes and the rules are as follows :-

1. All integer primes of the form  $p = 4k + 3$  are among the Gaussian primes.
2. Since all integer primes of the form  $p = 4k + 1$  can be expressed as a sum of two squares, they may be factored into two distinct Gaussian primes as given by  $p = a^2 + b^2 = (a + jb)(a - jb)$  and  $j = \sqrt{-1}$ .
3. The prime number 2 can be factored as  $(1 + j)(1 - j)$ , but only *one* of these factors is distinct.

#### 3.4.3.1 Conversion To Complex Quadratic Residue Number System

To convert a complex number into the equivalent residue format, the following procedure is adopted.

If, given a complex integer input of the form  $(m + nj)$ , then

$$x^+ = \langle m + nj \rangle_{(a+bj)}$$

So

$$\begin{aligned} x^+(a - bj) &= \langle (m + nj)(a - bj) \rangle_{(a+bj)(a-bj)} \\ x^+a - x^+bj &= \langle (am + bn) + (an - bm)j \rangle_p \end{aligned}$$

P is prime, so that the residues mod  $p$  forms a field and consequently the multiplicative inverses of  $a$  can be found. Therefore, multiplying by  $a^{-1}[p]$  implies

$$x^+ - x^+ba^{-1}[p]j = \langle (m + a^{-1}[p]n) + (n - ba^{-1}[p]m)j \rangle_p$$

---

<sup>5</sup>Normal complex addition requires two real adds and normal complex multiplication requires four real multipliers and two real adds.

and is satisfied by

$$x^+ = \langle m + a^{-1}[p]bn \rangle_p$$

and is a real integer. For a particular choice of  $p$ , the term  $\langle a^{-1}[p]b \rangle_p$  is unique and can be labelled  $K_p$ .

The other Gaussian prime factor of  $p$  can be shown to be

$$x^- = \langle m - a^{-1}[p]bn \rangle_p = \langle m - K_p n \rangle_p$$

Therefore  $x^+$  and  $x^-$  are **real integers** which define the complex number  $m + nj$  as long as

$$c_m \leq m < c_m + p - 1$$

and

$$c_n \leq n < c_n + p - 1$$

Where the constants  $c_m$  and  $c_n$  are fixed constants and set to  $-\frac{p-1}{2}$ , so the complex numbers are centred about the origin in both ordinates.

### 3.4.3.2 Conversion Back To Conventional Number System

Let us assume that we have a  $2r$  'digit' residue representation

$$(x_1^+, x_1^-, \dots, x_k^+, x_k^-, \dots, x_r^+, x_r^-)$$

Now forming

$$y'_k = \langle x_k^+ + x_k^- \rangle_{p_k}$$

and substituting for  $x_k^+$  and  $x_k^-$  as derived previously, results in

$$y'_k = \langle 2m \rangle_{p_k}$$

Finally

$$y_k = \langle \langle 2^{-1} \rangle_{p_k} \langle 2m \rangle_{p_k} \rangle_{p_k} = \langle m \rangle_{p_k}$$

and is the residue of the real part mod  $p_k$ .

The other term  $z'_k = \langle x_k^+ - x_k^- \rangle_{p_k}$  becomes

$$z_k = \langle n \rangle_{p_k}$$

and is the residue of the imaginary part mod  $p_k$ .

The final stage of the conversion is by passing  $y_k$  and  $z_k$  through the Mixed-Radix Conversion method or by using the Chinese Remainder Theorem as discussed earlier.

---

### 3.5 Addition Elements

In this section, adders and subtractors for conventional binary and signed-digit number representations will be discussed.

#### 3.5.1 Conventional Binary Addition and Subtraction

The simplest method of addition/subtraction in sign-complement addition is to follow the approach which is taught at school. So, addition is performed from the least significant digit with zero initial carry through to the most significant digit in a wave-like propagation. These adders are called ripple carry adders because the carry flows through from the least significant digit. Overflow can be detected by checking the carry bits on the penultimate two adder outputs and if they differ the result of the arithmetic operation is wrong. Subtraction is performed by adding the two numbers together, but radix-complementing the subtrahend. Addition is performed using a **full adder**, where the *sum* output is the digit addition with carry in binary and the *carry* output is the overflow from the sum. In digital form this is represented by

$$\begin{aligned} S_i &= A_i \oplus B_i \oplus C_i \\ C_{i+1} &= A_i \cdot B_i + B_i \cdot C_i + A_i \cdot C_i \end{aligned}$$

An add/subtract cell using an additional control line  $M$ , with  $M$  equal to one to subtract and zero for addition, can be represented by

$$\begin{aligned} S_i &= A_i \cdot (B_i \cdot M) \cdot C_i \\ C_{i+1} &= (A_i + C_i) \cdot (B_i \cdot M) + A_i \cdot C_i \end{aligned}$$

Figure 3.1 shows block diagrams of a ones-complement and twos-complement adder/subtractor ripple-carry adder design. The implementation of a sign-magnitude adder/subtractor requires a large number of XOR gates and a more complex sign and overflow circuit. All these circuits except, sign-magnitude coded adders based on the ripple-carry approach, take  $2n + 6$  time slots for an  $n$ -bit adder. This total time (or delay) is the time taken for the carry in the least-significant digit to travel through to the most-significant digit.

As the adder's data word increases, the ripple-carry method becomes inefficient. Therefore alternative adders have been developed which reduce the carry propagation and thus increase speed of computation. The carry-completion sensing adder is an asynchronous self-timed unit that aims to reduce the carry propagation length

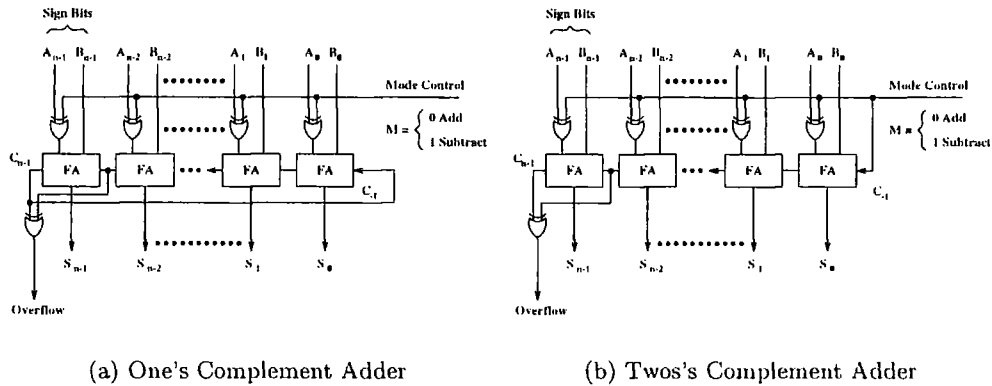


Figure 3.1: Block Diagrams of ripple carry adders for conventional binary arithmetic

to  $n/\log_2 n$ . Alternatives are conditional sum and carry-select adders. High speed addition can be accomplished with a carry lookahead technique. This approach computes the carry bit for each digit in parallel and sends this information to the sum part of a full adder. However, for large bit lengths, this approach requires an exponential increase in logic gates making chip routing and power dissipation more difficult.

### 3.5.2 Signed Digit Addition and Subtraction

Using the signed-digit number system, the designer can build a SD adder/subtractor with a carry propagation to one position to the left during the operation. The time required to perform the parallel addition is equal to the time required to add two adjacent digits. This is achieved by the following :-

- The radix  $r > 2$ .
- The algebraic value of zero must have a unique SD representation.
- There must be transformations between conventional sign-magnitude m-digit representation and SD m-digit representation for all algebraic values within the machine range.

#### 3.5.2.1 The Algorithm.

For parallel operation all digital positions  $n-1 \geq i \geq -k$  for  $(n+k)$ - digit SD numbers must satisfy :-



1. Let  $s_i$  be the  $i$ 'th **sum digit** of the resulting sum  
 $S = (s_{n-1} \cdots s_1 s_0 \bullet s_{-1} \cdots s_{-k})_r = \mathcal{Z} + \mathcal{Y}$  and  $t_i$  be the **transfer digit** from the  $(i-1)$ 'th digital position. Then  $s_i = f(z_i, y_i, t_i)$ .<sup>6</sup>
2. The transfer digit  $t_{i+1}$  to the  $(i+1)$ 'th digital position on the left is a function of the augend digit  $z_i$  and the addend digit  $y_i$ . ie.  $t_{i+1} = g(z_i, y_i)$ .

To achieve parallel subtraction of the subtrahend digit  $y_i$  from the minuend digit  $z_i$ , the operation

$$z_i - y_i \equiv z_i + \overline{y_i}$$

This operation is SD parallel addition of the minuend digit added to the **additive inverse** of the subtrahend digit.

The transfer digit,  $t_i$  can assume positive and negative values in SD addition and subtraction and is *never* propagated past the first adder position on the left.

The SD parallel adder/subtractor is a two step process on the digits  $z_i$ ,  $y_i$  and  $t_i$  and is the following :-

1. The outgoing transfer digit  $t_{i+1}$  and the interim sum digit  $\omega_i$  are generated by the addition of  $z_i$  to  $y_i$  and is

$$r \cdot t_{i+1} + \omega_i = z_i + y_i$$

2. The sum digit  $s_i$  is obtained by adding  $\omega_i$  to  $t_i$ , the transfer digit from the digital position  $i - 1$ , and is calculated by

$$s_i = \omega_i + t_i$$

This algorithm is shown graphically in figure 3.2 and for parallel operation

$$|s_i| \leq |z_i| \text{ and } |y_i|$$

and the unique zero representation criteria is satisfied by

$$|Z_i| \leq r - 1$$

SD subtraction requires the condition that for every  $y_i = a$ , there exists  $\overline{y_i} = -a$  such that

$$y_i + \overline{y_i} = a + (-a) = 0$$

---

<sup>6</sup>  $z_i$  and  $y_i$  are the  $i$ -th digits of the augend  $\mathcal{Z}$  and the addend  $\mathcal{Y}$  respectively.

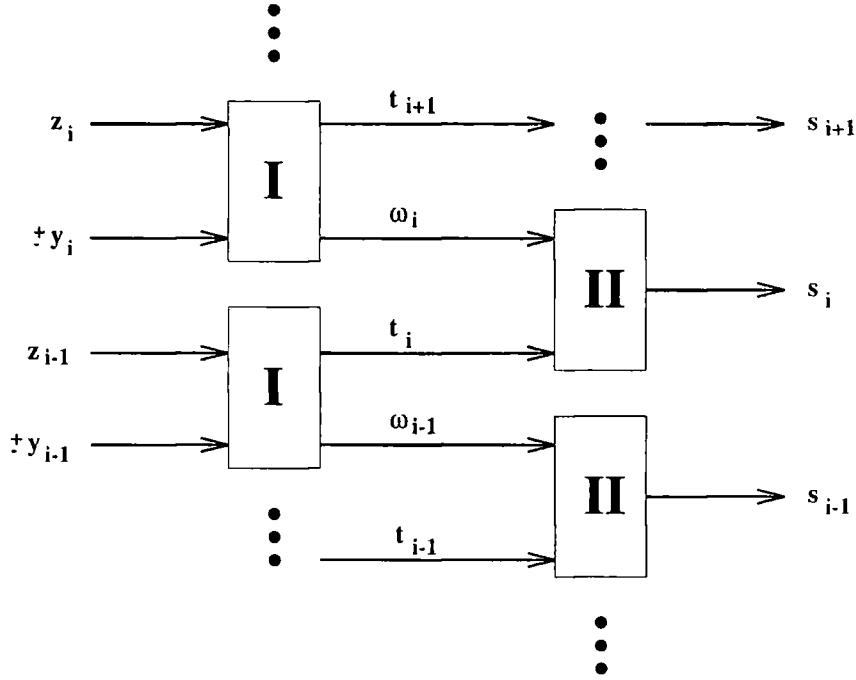


Figure 3.2: Block Diagram of Part of the SD adder/subtractor

The convertibility requirement between conventional and SD format is in the same form as in section 3.3.2.1, but the transfer digit ( the borrow difference ) can take negative values. The condition

$$|\omega_i| \leq r - 2$$

is obtained as the upper magnitude limit for the interim sum if the transfer digit  $t_i$  is restricted to -1, 0, 1. Therefore the minimum base to achieve the operation is three ( by substitution for the conventional digit in section 3.3 and  $x_i = 1$  ). A consequence of these conditions imply that the set of allowed values for each interim digit is

$$\omega_i \in \{-(r-2), \dots, -1, 0, 1, \dots, r-2\}$$

and this leads to two enlarged digit sets which are considered minimal when each digit assumes the smallest value  $\alpha = \frac{r+1}{2}$  for odd radix and  $\alpha = \frac{r}{2} + 1$  for even radix and maximal when  $\alpha = r - 1$ .

When  $\omega_i$  belongs to the sequence  $\{\omega_{\min}, \dots, -1, 0, 1, \dots, \omega_{\max}\}$  the transfer digit  $t_{i+1}$  is computed by

$$t_{i+1} = \begin{cases} 0, & \text{if } \omega_{\min} \leq z_i + y_i \leq \omega_{\max} \\ 1, & \text{if } z_i + y_i > \omega_{\max} \\ -1, & \text{if } z_i + y_i < \omega_{\min} \end{cases}$$

It is easy to perform addition and subtraction from least significant digit to most significant digit or vice versa.

The designer selects the required digit set for  $\omega_i$ , the interim sum digit to be the minimal set corresponding to the radix and the inputs to the adder/subtractor has a digit set which is minimal according to  $\alpha$  above.

Arithmetic right shifting an SD number and the transfer digit is generated when

$$|z_{-k}| \leq |z_i|_{\max} - 1$$

and is useful in multiple precision arithmetic operations. Overflow due to shifting left can be predicted by inspection of the most significant two digits before the shift operation. The total time delay of a fully-parallel SD adder/subtractor is determined by the add time of only one stage of the digit adder ( ie. stage I and stage II ) and is independent of word-length (  $m = n + k$  ).

3.6 Array Multiplication Techniques

Array multiplication algorithms are modelled on the familiar multiplication technique children use to perform multiplication. This method uses addition, multiplication and radix left shifting, starting from the units column. This is demonstrated in equation 3.6.

There are two basic types of modular array multipliers, which are known as the Non-additive Multiply Module ( NMM ) and the Additive Multiply Module ( AMM ). The NMM type computes the expression

$$Y = A \times B \tag{3.4}$$

and the AMM type computes the expression

$$Y = A \times B + C + D \tag{3.5}$$

The four by four NMM structure is displayed in figure 3.3 and implements equation 3.4 by the following worked equation.

				$a_3$	$a_2$	$a_1$	$a_0$	
			$\times)$	$b_3$	$b_2$	$b_1$	$b_0$	
				$a_3 \cdot b_0$	$a_2 \cdot b_0$	$a_1 \cdot b_0$	$a_0 \cdot b_0$	
				$a_3 \cdot b_1$	$a_2 \cdot b_1$	$a_1 \cdot b_1$	$a_0 \cdot b_1$	
				$a_3 \cdot b_2$	$a_2 \cdot b_2$	$a_1 \cdot b_2$	$a_0 \cdot b_2$	
				$a_3 \cdot b_3$	$a_2 \cdot b_3$	$a_1 \cdot b_3$	$a_0 \cdot b_3$	
				$P_7$	$P_6$	$P_5$	$P_4$	
				$P_3$	$P_2$	$P_1$	$P_0$	

$$\tag{3.6}$$

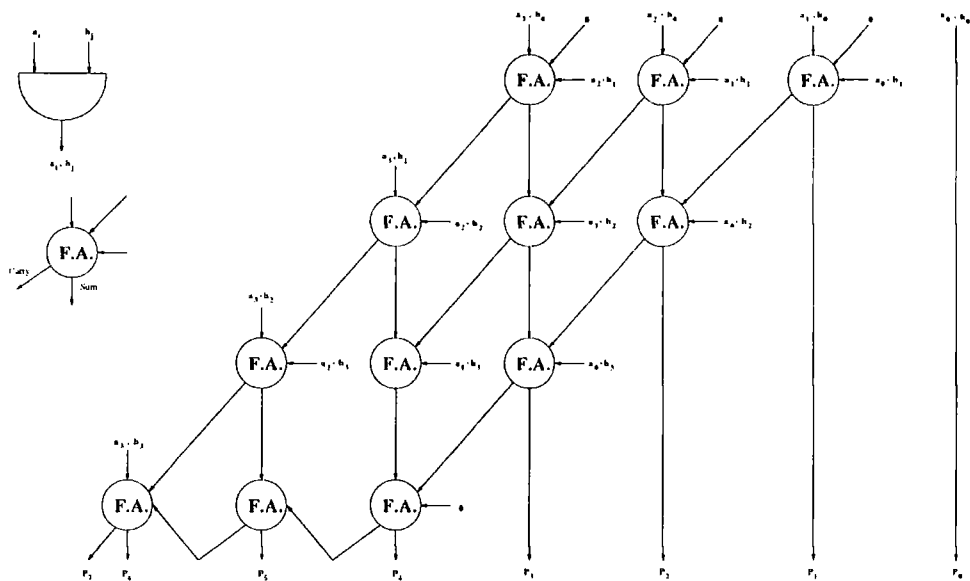


Figure 3.3: The Structure of a 4x4 NMM

The four by two AMM structure is displayed in figure 3.4 and implements the arithmetic calculation shown in equation 3.5 and worked out in the following equation.

		$a_3$	$a_2$	$a_1$	$a_0$	
$\times$	)			$b_1$	$b_0$	
		$a_3 \cdot b_0$	$a_2 \cdot b_0$	$a_1 \cdot b_0$	$a_0 \cdot b_0$	
		$a_3 \cdot b_1$	$a_2 \cdot b_1$	$a_1 \cdot b_1$	$a_0 \cdot b_1$	
		$c_3$	$c_2$	$c_1$	$c_0$	
$+$	)			$d_1$	$d_0$	
		$P_5$	$P_4$	$P_3$	$P_2$	$P_1$

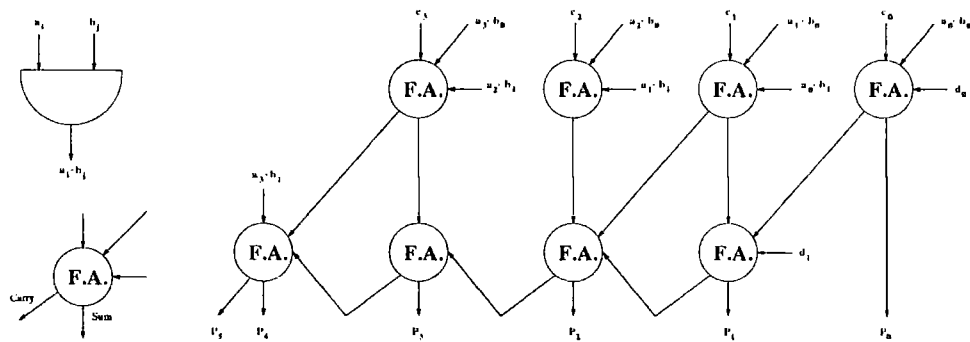


Figure 3.4: The Structure of a 4x2 AMM

The AMM array multiplier ( figure 3.4 ) is particularly attractive for DSP because it

performs a MAC operation. It uses less full adders than the 4x2 NMM and requires less AND gates to calculate the partial products. But compared to a square matrix ( eg.. 4x4, 8x8 .... ) the total delay time through the full adder network is two units slower than the NMM case [52]. The AMM will use more external wires than the NMM because it computes a more complicated function. The NMM has the disadvantage in requiring additional summing devices like multiple-operand Wallace tree adders; this however gives the structure a doubling in speed compared to the AMM case.

The  $k$ -input Wallace tree is a  $k$ -to- $\frac{k-1}{2}$  carry-save full adder and is used to create the sub-products simultaneously. A further requirement is to separate the operands into blocks usable in the 4x4 NMM. This is written as

$$P = A \times B = (A_H \cdot A_L) \times (B_H \cdot B_L) = A_H \times B_H + A_H \times B_L + A_L \times B_H + A_L \times B_L$$

for construction of an 8-by-8 array multiplier using four 4-by-4 NMM structures. The last term  $A_L \times B_L$  needs no further modifications, but the other parts require eight 3-input Wallace trees and a 12-bit Carry Propagate Adder.

To implement either type of array multiplier requires a large floor plan for VLSI, implying a large number of logic gates creating a fast multiplier but also a large power dissipation problem. Apart from the speed, the logic elements are repetitive and useful in VLSI design.

### 3.6.1 Cellular Array Multipliers Conclusions

There are many variants of cellular array multipliers based on the AMM and NMM type devices. Some also use novel adders like Wallace trees and carry propagate adders. There is also a non direct Universal Multiplication Array ( UMA ) device which provides multiplication in the standard binary formats ( unsigned, sign-magnitude, one's complement and two's complement numbers ) using support logic. The UMA is normally created using programmable AMM modules.

Designers have built specialised two's complement array multipliers using four types of adder. Each adder type has a certain number of negatively weighted inputs which are connected together using a certain mixture of adder types. The fastest adders are the Pezaris and Tri-Section, using all four types of adder and three types of adder respectively. The Baugh-Wooley uses one type of adder and is slightly slower than the others; it uses more combinational logic to provide the various product terms for the array.

As mentioned before, all these devices perform multiplication very quickly at the expense of a large network of logic gates with many connecting wires. This creates

a large amount of heat which has to be dissipated. Also valuable silicon is used up to perform one function. It would be better to use the minimal amount of silicon to provide many functions at a reasonable speed and at a low power dissipation ! The signed-digit number system, residue number systems and the CORDIC algorithm all show promise in this area.

3.7 Canonical Multiplier Recoding

3.7.1 Introduction to Multiplication String Recoding Algorithms

The standard binary multiplication algorithm is performed by adding the partial sums and arithmetic left shifting the operands. This is efficient only when the multiplier has more zeros than ones. If the multiplier has a string of ones then the standard algorithm will operate more slowly due to the increased number of ones. This problem can be solved by recoding the multiplier to search for consecutive ones and recode the data.

This recoding process can be shown by the following process; assuming there is a string of consecutive ones ( non zero elements ) in the multiplier.

Column Position	...	$i + k$	$i + k - 1$	$i + k - 2$	...	$i$	$i - 1$	...
Bit Content	...	0	1	1	...	1	0	...

Now the string property

$$2^{i+k} - 2^i = 2^{i+k-1} + 2^{i+k-2} + \dots + 2^{i+1} + 2^i$$

and the k consecutive ones can be replaced by the left hand side of the string property equation. Hence the recoded multiplier is the following :-

Column Position	...	$i + k$	$i + k - 1$	$i + k - 2$	...	$i$	$i - 1$	...
Bit Content	...	0	1	0	...	-1	0	...

Therefore execution time will be reduced, because the algorithm will shift across k-1 consecutive zeros and will perform one addition and one subtraction. The next stage of the algorithm is to work out how to search for strings of ones using either non-overlapped binary bit pairwise scanning or overlapped triplet bit pairwise scanning techniques. Table 3.1 shows the properties of triplet scanning.

This recoding method uses carry-save adder trees and a carry propagate adder.

Multiplier Bits				
The low-bit of the next higher pair $x_{i+2}$	The Present Pair $x_{i+1}$ $x_i$		Multiplicand Multiples to be added	Reasoning behind string property
0	0	0	0	No string of ones
0	0	1	+ 2A	End of string
0	1	0	+ 2A	Isolated one
0	1	1	+ 4A	End of string
1	0	0	- 4A	Beginning of string
1	0	1	- 2A	Beginning and ending
1	1	0	- 2A	Beginning of string
1	1	1	0	Centre of string

Table 3.1: Multiples of the multiplicand to be added after scanning a triplet of multiplier bits in an overlapped pairwise scanning system

### 3.7.2 Canonical Signed-Digit Multiplier Recoding

If the designer now replaces conventional multiplier digits with the high redundancy SD code, the addition operations are reduced and there is a greater average shift length across the zeros in the multiplier, leading to faster operation.

The requirements are a minimal SD vector which has the minimum weight for a prespecified  $\alpha$  as in section 3.3; this vector must be canonical. A minimal canonical SD vector means the SD vector  $\mathcal{D} = d_{n-1} \cdots d_1 d_0$  contains no adjacent non-zero digits, therefore

$$d_i \times d_{i-1} = 0 \text{ for } 1 \leq i \leq n-1$$

Research in this field has shown that a unique canonical SD vector  $\mathcal{D}$  for a number with a fixed value  $\alpha$  and a fixed length  $n$  exists provided the product of the two left most digits in  $\mathcal{D}$  does not equal one. This is expressed as

$$d_{n-1} \times d_{n-2} \neq 1$$

and is satisfied if an additional digit, which equals zero, is inserted at the left most end of the vector  $\mathcal{D}$ . Now the number is an  $n+1$  digit SD vector with a leading digit zero and the following algorithm is used to recode the conventional number system into the canonical SD format.

### 3.7.3 Canonical Recoding Algorithm

Given an  $(n+1)$ -digit binary vector  $\mathcal{B} = b_n b_{n-1} \cdots b_1 b_0$  with  $b_n = 0$  and  $b_i \in \{0, 1\}$  for  $0 \leq i \leq n-1$ . The  $(n+1)$ -digit canonical SD vector  $\mathcal{D}$  with  $d_n = 0$  and

$d_i = \{\bar{1}, 0, 1\}$  is manipulated so that both vectors represent the same numerical value.

$$\sum_{i=0}^n b_i \times 2^i = \sum_{i=0}^n d_i \times 2^i$$

- Start with the low order end of  $\mathcal{B}$  by setting the index  $i = 0$  and initial carry  $c_0 = 0$ .
- Generate the carry out using the standard rule in conventional binary arithmetic which is

$$c_{i+1} = b_{i+1} \cdot b_i + b_i \cdot c_i + b_{i+1} \cdot c_i$$

- Generate the  $i$ 'th digit  $d_i$  of the vector  $\mathcal{D}$  using

$$d_i = b_i + c_i - r \cdot c_{i+1}$$

- Increment index  $i$  and repeat if  $i \neq n$  otherwise algorithm has been completed.

The canonical SD vector has the minimum weight and all non-zero digits are separated by zeros. All that is required for multiplication is addition by  $A$  or  $-A$  when a non-zero digit appears; the zeros provide more shifting operations.

The algorithm can be easily extended to generating two or more signed-digits at a time. By using a higher radix, the multiples will increase and there will be a larger shift per cycle. eg..  $r = 4$  implies multiples of  $0, \pm A, \pm 2A$  and 2-digit shifting per cycle.

3.7.4 The Booth's Multiplier Algorithm

The Booth multiplier is based on string recoding and for conventional binary numbers in radix-2 the algorithm assigns the following inputs and outputs as in table 3.2.

Inputs		outputs	Remark on string property
$b_i$	$b_{i-1}$	$d_i$	
0	0	0	No string
0	1	1	End of string
1	0	$\bar{1}$	Beginning of string
1	1	0	Centre of string

Table 3.2: The Booth Multiplier Radix-2 String Recoding Algorithm



The output digit  $d_i$  for  $0 \leq i \leq n$  is obtained by the following discrimination rule :-

$$d_i = \begin{cases} 0, & \text{if } b_i = b_{i-1} \\ 1, & \text{if } b_i < b_{i-1} \\ \bar{1}, & \text{if } b_i > b_{i-1} \end{cases}$$

The procedure above requires the attachment of two zero valued dummy digits to each end of the  $n$ -digit binary vector. The algorithm generates no carries because each output digit is only a function of the two adjacent input bits. This algorithm can generate the output vector  $\mathcal{D}$  by examining adjacent input pairs of vector  $\mathcal{B}$  in parallel.

The recoding scheme can be expanded to higher radices by partitioning the recoded string into pairs or triplets. It is also important to remember that the recoded string can have adjacent non-zero digits having opposite signs.

After the string recoded vector is formed the following operations are performed.

- if  $b_i = b_{i-1}$ , shift right partial product
- if  $b_i < b_{i-1}$ , add A to partial product and then shift right
- if  $b_i > b_{i-1}$ , subtract A from partial product and then shift right

If the arithmetic left shifting operation exists, the multiply algorithm can proceed from the most significant bit. The subtraction operation is performed by two's complement addition. This algorithm is a boon to two's complement multiplication because the signs of the multiplier and multiplicand are automatically taken into account during the procedure.

### 3.7.5 Efficiency of Multiplier Algorithms and Design Alternatives

The measure of multiplier efficiency for these add-shift multiplier algorithms, are based on the average number of add-type operations and the average shift length between successive additions in the algorithm. The average shift length should increase and the average number of additions should decrease with respect to a simple add-shift multiplication algorithm.

The various multiplier design alternatives are summarised below :-

**Adder Bypassing** Shifting across strings of zeros in the multiplier, by using non-uniform shifts. This is ideal for asynchronous computers. The disadvantage is the increased control complexity.

**Reduced Addition Time** Parallel carry generation using Carry Lookahead Array (CLA) techniques or Carry-Save methods will reduce the addition time. SD arithmetic allows parallel addition to be performed very conveniently.

**Multiple Shifts** Provision must be made to shift more than one digit per iteration. This increases hardware, so the optimal choice is a tradeoff between speed and cost.

**Multiplier Recoding** The introduction of redundancy into multiplier designs through recoding, may not necessarily increase the circuit complexity, due to the simplicity in generating the multiplicand multiples. Recoding increases the average shift length.

**High-Radix Multiplication** The number of iterations required in a multiplication can be reduced by a factor of  $k = \log_r x$  and  $x$  is the number radix.

### 3.8 Summary

The simplest architectures are the array multipliers. These are very common in high speed applications because when pipelined they can become systolic arrays. Due to their inherent regularity, their great advantages are ease of design and the generation of area efficient layouts.

The multiplier recoding techniques can be applied to parallel and serial execution units, but come into their own when in serial form. The ground breaking paper of Lyon on serial booth encoded multipliers [79] was very useful in the VLSI DSP community as it enabled complex circuits to be designed. An example of these was a programmable filter bank [80] for cochlea simulation. A modified form with a routing matrix was built by Wawrzynek for musical synthesis applications [143, 144]. There are alternative forms of booth encoding which only have positive shifting for example the radix 4 serial multiplier [110].

There are many types of signed digit arithmetic units, which fall into regular array and serial architectures based on Avizienis work [4]. Work at Queen's University of Belfast on systolic arrays have produced IIR filters [150] which have lower latency as the digits can be extracted with the most significant digit first. A hybrid booth encoded multiply accumulator unit with an internal redundant binary format [51] can also be designed. Serial based multipliers [27] require a means of converting the digit's stream into two's complement form. This is carried out by the algorithm of Ercegovac and Lang in their paper [26].

---

The residue number system, which can be thought of as a highly redundant number system has been widely researched [36, 132]. It has been implemented in IIR [29] and FIR [59] DSP structures. It has also appeared in FFT processors [22] using gaussian residue arithmetic. There has even been a paper by Taylor and Huang on a floating point ALU [134]. However, its use with current digital number systems requires a large overhead either by using the Chinese Remainder Theorem or by using a mixed radix form changing between the two formats. Examples are [30] and more recently by [25, 42]. A mixed radix case [152] has been reported. The residue number system has been used successfully in a multi-valued logic based image processor [44] using current mode logic. Multipliers [62, 66] using this novel VLSI circuit design flow have also been constructed. In chapter 4 I will discuss this circuit technique in which logic can be held in more than two states. This number system has even found itself in optical computing [90] by using the Fredkin logic element [33].

This chapter has reviewed various ways of performing arithmetic computation using novel number systems. Some of these having no analogue to the human approach, have advantages in the design and implementation of VLSI architectures for DSP.

Since digital signal processing is concerned with describing the world in terms of integer representations, their transformation requires the use of arithmetic operations. There are many ways of representing numbers electronically and this chapter discussed the arithmetic operations these representations allow. Each representation and style help designers to build efficient computers. The most common number system in use today is the sign-magnitude and two's complement forms. However, by using redundant number systems inside the arithmetic blocks and by efficient conversion between the two formats, fast computation can be performed.

If we are willing to relinquish our beloved binary number system and discard digital CMOS design, an important new avenue in computation can be explored. This is discussed further in the next chapter. It remains to be seen whether alternative computation becomes predominant in the future. I think that these new forms will have a wonderful future !

The rest of this thesis concentrates on two's complement arithmetic applied to a musical synthesis algorithm. The design infrastructure enabling simulation to realisation necessitates this particular choice of method. Using two's complement arithmetic enables certain transformations to be achieved using a relatively small silicon area. This will be explored further in chapter 5.

---

---

## CHAPTER 4

# VLSI Technologies and Applications

---

This thesis has so far concentrated on arithmetic algorithms and possible structures. The signed digit number system has become more important in MAC designs due to the increased number redundancy. The residue arithmetic systems have been studied by a few researchers, but implementation is somewhat contrived because the designers still work in binary and its higher radices [132].

High radix arithmetic based processors have remained mainly an academic interest because of the increased hardware complexity. The cost of an arithmetic processor is proportional to the square of the word-length, and the speed is proportional to the logarithm of the word-length [52]. The multiply unit in processors contains  $\approx 10\%$  of the total semiconductor components and the cost of this element is  $\approx < 10\%$  of the total processor cost.

Novel VLSI technologies would allow more powerful arithmetic algorithms to be constructed in various number formats. However the designer would need to provide a backward compatibility path to standardised binary operated processors as they enable easy interfacing between the novel processor and current peripherals. Such compatibility is even more important when designers consider floating point IEEE compliant arithmetic processors.

### 4.1 Novel VLSI Technologies

The best architecture for an arithmetic processor is one which uses the minimum area of silicon and provides maximum speed. The complexity of the circuitry should be minimal. The complexity is defined by the number of components and their interconnections.

The current trend in digital electronics is to reduce the feature sizes of transistors to achieve faster operations. This is easily understandable and is usable up to the limits of lithographic techniques and chemical etching. Once the size of the devices become very small, quantum size effects become predominant and the transistor will not function. In this regime, new devices need to be invented. As transistor size is reduced, their voltage supply must also be reduced, and because of its quadratic dependancy, power consumption is decreased commensurably. However, there is a speed penalty which can be overcome by architectural transformations<sup>1</sup> and scaling the threshold voltage [71] of the transistor. As an interim measure these techniques are effective, but other problems like poor signal to noise ratios on chip could halt their progress.

I would like to highlight four novel technologies which show great promise. These are :-

**Novel Materials** This normally uses Gallium-Arsenide technology, allowing a corresponding increase in speed. Other fabrication technologies are Silicon-Germanium substrates, which the companies IBM and Analog Devices are now developing and marketing. Another technology is Molecular Electronics, but this is a very new technology and will take ten years or so before becoming commercially useful. Optical based computers do exist, but they take up a large amount of space ( the optical bench ) and cannot interface simply with silicon based systems. A related technology is known as plasma wave electronics [83]. Other examples include the field of spintronics [85] which utilise magnetic material to confine electrons in one of two spin states. The statistical average of all these confined electrons results in a memory device which requires no refreshing of the cells' contents. Their other advantage is that they can be scaled to smaller dimensions because they utilise quantum mechanical effects and thus increase package density.

**Different Digital Modules** I mention these because the Japanese have built a Josephson Junction SQUID based computer [39] working at 1GHz. Negative

---

<sup>1</sup>Which can also be applied to improve system stability [41].

---

Differential Resistance ( NDR ) based silicon circuits are also beginning to be used, as they can combine two or more operations in one transistor.

**Low Power Designs** This will take on more significance in the coming years as devices are squeezed onto smaller pieces of silicon. Smaller transistors makes the systems operate faster but it also increases power dissipation. In [145] low-power reversible computational elements was discussed.

**MVL Logic** This acronym stands for multiple-valued logic, whereby instead of binary encoding on wires, the designer chooses a logic which has more than two values.

The majority of digital circuits are based on voltage thresholds and positive feedback to the supply rails and thus provide noise immunity in digital design. An interesting low power, high speed technique for latch design can be achieved using current mode logic [151].

The rest of this chapter will concentrate on various multiple valued logic structures suitable for VLSI implementation. They are current-mode logic, negative differential resistance, voltage-mode logic and a new concept which I have created.

## 4.2 Current-Mode Logic

Current Mode Logic is based on the subset of Multiple-Valued Logic ( MVL ), which is concerned with architecture implementation in silicon by use of currents to carry information. The field of MVL is so wide, it is necessary to relabel a small subset which has the most benefit to VLSI designers. The IEEE society has a yearly international symposium on multiple-valued logic, from algebra to implementations of ternary computers.

Current Mode Logic is composed of four elements as follows :-

- Current Sources.
  - Current Mirrors. There are two types, nMOS and pMOS.
  - Threshold Detectors.
  - Bidirectional Current Input Devices.
-

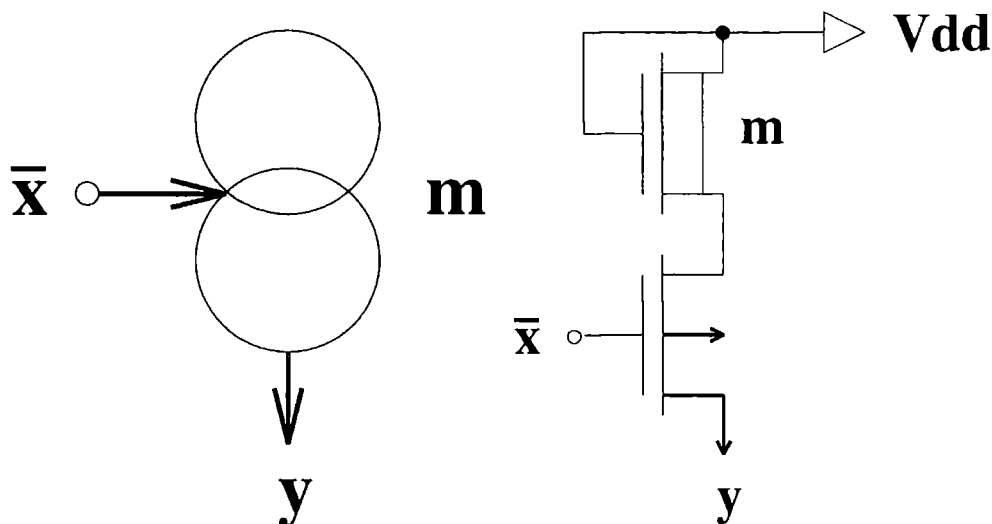


Figure 4.1: Current Source Circuit Symbol and Schematic

#### 4.2.1 Current Source

The current source as shown in figure 4.1 implements the function

$$\begin{cases} y = 0 & \text{if } \bar{x} = '1' \\ y = m & \text{if } \bar{x} = '0' \end{cases}$$

where  $\bar{x}$  is the input,  $y$  is the output and  $m$  is an integer.

The current source is implemented using a p-channel depletion-mode MOSFET for fast and stable operation. The saturation value of the drain current  $I_d$  is used as a constant current and is related by

$$I_d = K_d \cdot \frac{W}{L} \cdot V_T^2$$

where  $K_d$ ,  $V_T$ ,  $W$  and  $L$  are the transconductance, threshold voltage, channel width and channel length respectively. The unit current value is set by the dose control. The current source is insensitive to fluctuations in  $V_{DD}$ .

A voltage-switched current source can be implemented using a p-channel enhancement-mode MOSFET ( current pass transistor ) connected to a p-channel depletion-mode MOSFET ( see right hand circuit schematic of the threshold detector on page 58 ).

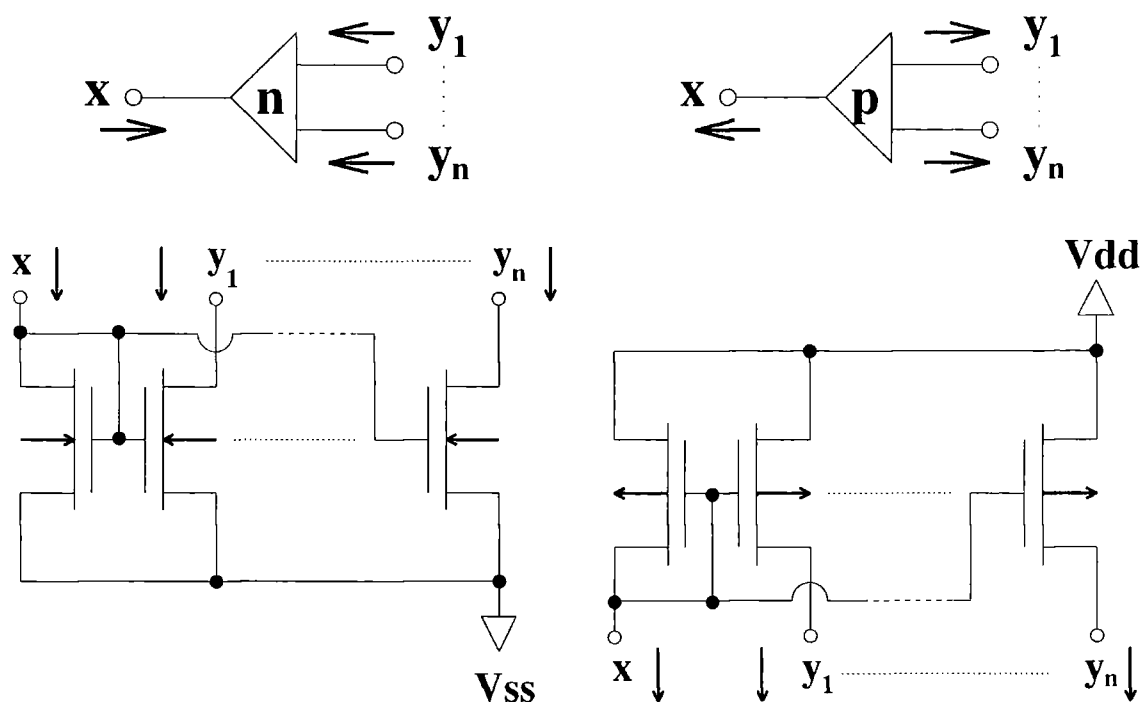


Figure 4.2: Current Mirror Circuit Symbols and Schematics

#### 4.2.2 Current Mirror

There are two types of current mirror circuits, nMOS and pMOS respectively and these are shown in figure 4.2 and implement the function

$$y_i = -a_i \cdot x \quad \text{for } i = 1 \cdots n$$

where  $x$  is the input,  $y_i$  is the output and  $a_i$  is a scale factor.

The current mirror is used to invert current direction, duplicate input current or scale the input current. The nMOS current mirror's input is operated in the saturation region and the input current  $I_{in}$  is given by

$$I_{in} = K \cdot (V_{in} - V_T)^2$$

where  $K$ ,  $V_T$  and  $V_{in}$  are the proportionality constant, the threshold voltage and the input voltage respectively. The output transistor operates in the saturation region because the gate is connected to the input of the current mirror, hence

$$I_{in} = I_{out}$$

A similar equation can be used for the pMOS current mirror.

For short channel length devices,  $I_{out}$  deviates from  $I_{in}$  due to the channel modulation effect. This could be cured by using a diffusion self-aligned MOSFET device.



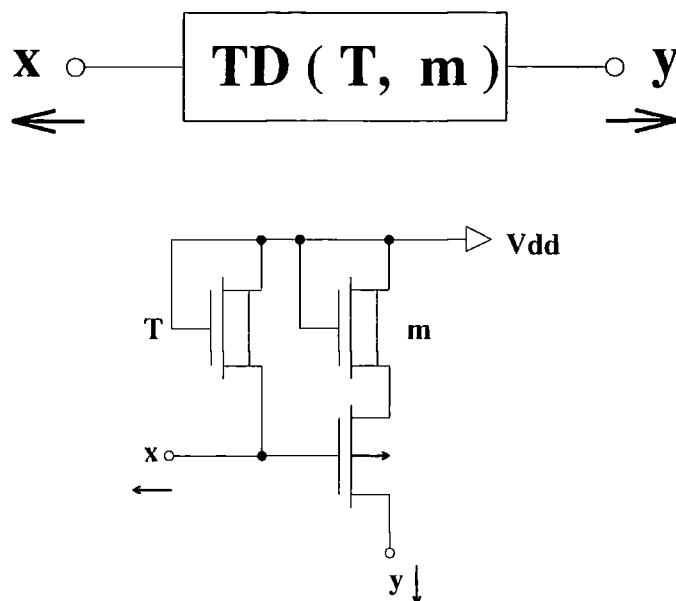


Figure 4.3: Threshold Detector Circuit Symbol and Schematic

### 4.2.3 Threshold Detector

The threshold detector is shown in figure 4.3 and implements the equation

$$\begin{cases} y = 0 & \text{if } x < T \\ y = m & \text{if } x \geq T \end{cases}$$

Where  $x$  is the input,  $y$  the output,  $T$  the threshold and  $m$  is an integer.

### 4.2.4 Bidirectional current input circuit

The Bidirectional Current Input device detects the polarity of a bidirectional current and is shown in figure 4.4. It uses current mirrors in addition to MOSFETS and a NOT gate. This device implements the function

$$\begin{cases} x_i^+ = x, & x_i^- = 0 & \text{if } x \geq 0 \\ x_i^+ = 0, & x_i^- = x & \text{if } x < 0 \end{cases}$$

Where  $x$  is the input and  $x_i^+$  and  $x_i^-$  are the outputs. The I-V transfer function of this device is shown in figure 4.5.

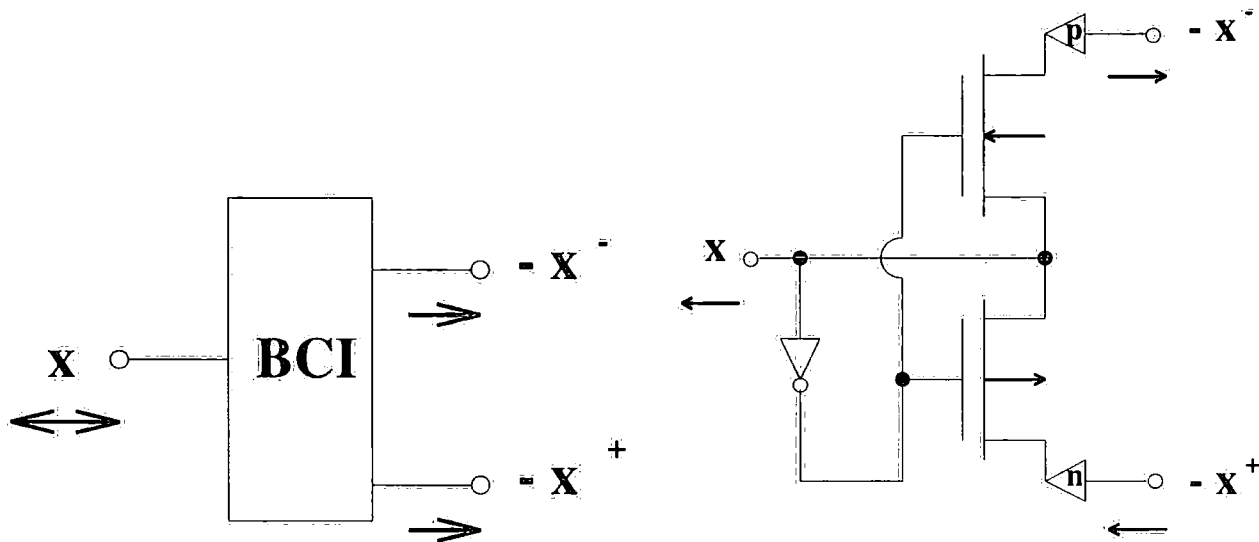


Figure 4.4: Bidirectional Current Input Circuit Symbol and Schematic

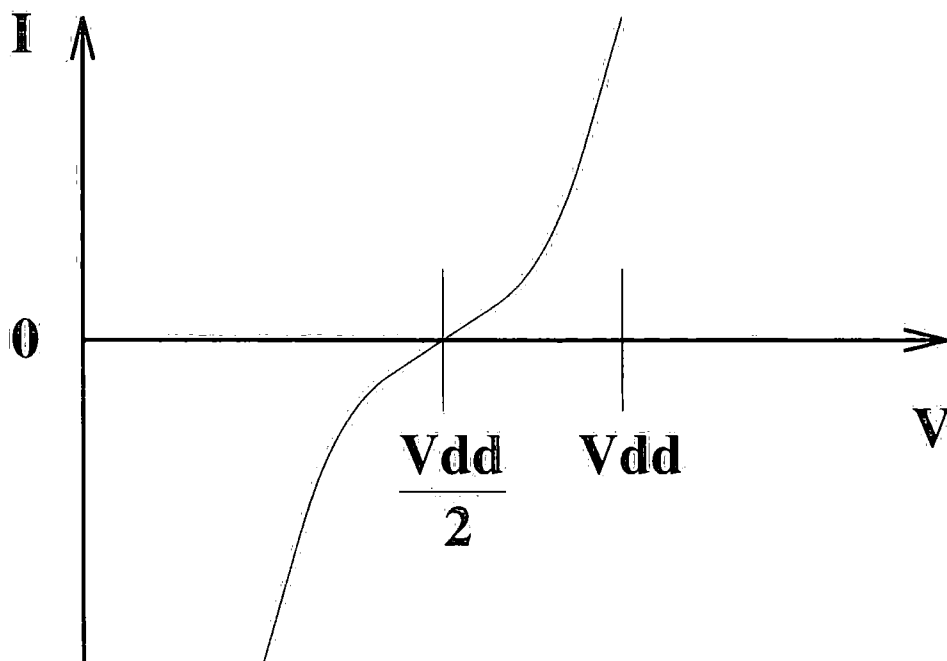


Figure 4.5: Bidirectional Current Input I-V characteristics

### 4.2.5 Module VLSI Scaling

The modules can be scaled easily and do not require more voltage space because they operate on currents. They require wider devices due to the number of current values used. The advantage with these devices is that the scaling is localised to parts of the chip area, whilst normal voltage driven devices ( not current based logic systems ) change the system wide voltage supply.

These modules have simple fan-in connections, using a wire, and complicated fan-out connections. This is the opposite of TTL NAND based logic. Generally they are also radix independent, but because of imperfect current mirrors, the radix is currently limited to four.

A good review of current mode CMOS multiple-valued logic circuits and theory can be found in [17, 55, 56]. An informative critique on CMOS current mode adder design can be found in [1]. Multipliers based on signed digit arithmetic can be implemented effectively with this paradigm [62, 66]. Residue arithmetic based devices have been built from signed-digit blocks and barrel shifters [44].

## 4.3 Negative Differential Resistance

Negative differential resistance occurs when part of the current versus voltage curve for a device has a negative slope, as shown in figure 4.6(a). A device which has negative resistance generates electrical power which violates energy conservation. However, devices and circuits can be designed which don't need external energy sources and are thus passive. These devices can be built systematically from two bipolar transistors and normal resistors as described in [12, 13, 14, 100] but are not suitable for VLSI implementation because at present they need large voltage ranges.

---

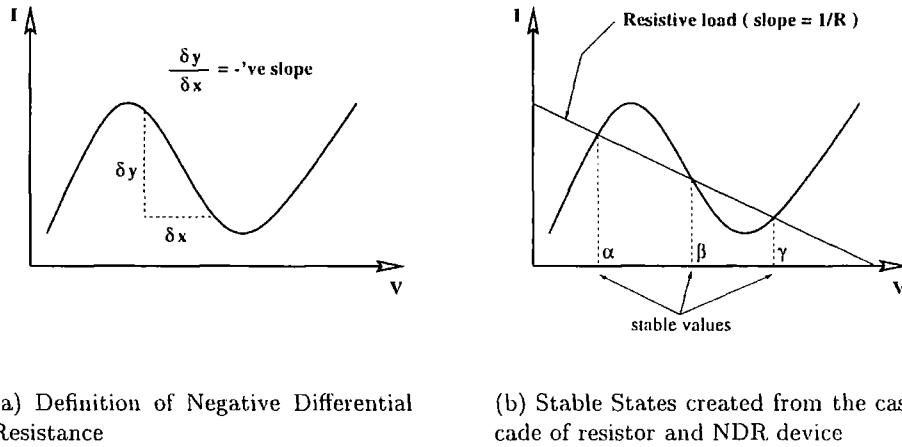


Figure 4.6: Block Diagrams of Negative differential Resistance

Alternate circuits supporting negative impedance are gyrators, which are built from operational amplifiers, resistors and capacitors. Gyrators are used in analog filter design to simulate inductors.

Devices which implement negative differential resistance can be manufactured using GaAs substrates and utilising the quantum mechanical phenomenon known as resonant tunnelling.

By connecting a resistor in cascade with a circuit or device which implements negative differential resistance, a circuit with multiple states can be established. This is shown diagrammatically in figure 4.6(b) and appears as the stable voltage points  $\alpha$ ,  $\beta$  and  $\gamma$ . If an external signal was applied across the nonlinear circuit element, three distinct values could be stored and either a ternary memory device or an xor function could thus be performed. Hence more functionality arising from fewer devices.

By design, physicists and engineers can create resonant tunnelling devices having multiple regions of negative differential resistance, which can be used for digital storage [123, 147] and analog frequency multiplication [8, 119]. They can also be used as building blocks for very fast 32 bit multiplication [91] using a logic design approach [92]. They are smaller than an optical based multiplier [75], but are currently quite power hungry and require GaAs Molecular Beam Epitaxy methods which are still a specialised niche market.

## 4.4 Voltage-Mode Multiple-Valued Logic

In this chapter, we have seen methodologies which generate many logic values by simulation of exotic non-linear behaviour, by current scaling, or by actual device physics. In this section, multithreshold logic circuits which are operated in voltages are discussed. There is a close correspondance with this approach and normal familiar binary logic systems.

This section is split into two. The first subsection will describe a voltage threshold mechanism built from operational amplifiers and the second will describe a functional transistor called  $\nu$ MOS.

### 4.4.1 Operational Amplifier Approach

In binary logic, designers allocate certain voltage ranges for high and low states and assume that the switch between these two stable states occurs over a narrow voltage range and at high speed. This should be familiar to designers of both CMOS and TTL logic families. These families have nice S-curved current-voltage characteristics for their inverters. The ranges provide a mechanism to protect current drain from previous and other logic elements in a design. The steepness of the I-V curve provides the logic level restoration necessary for noise immunity in digital circuits.

To provide multilevels we must guarantee adequate voltage ranges for each stable state, as voltage levels for each state can vary. A good rule of thumb is to separate each voltage state with  $\pm \frac{1}{2}$  its mid-value, so that the actual encoded voltage level is spread equidistantly over the supply voltage. This can be achieved by using operational amplifiers set up in parallel form, implementing an idealised inverter curve for output vs input voltages plus biasing. A comparator arrangement could be used if the number of circuits connected was very small, but this provides no method of renormalisation of the voltage levels.

The op-amp approach is useful in getting some practical experience in multi-valued logic but it is slow [109]. An approach using CMOS and Bipolar transistors in a multi-level neural network implementing Analogue to Digital conversion can be found in [153]

### 4.4.2 Neuron MOSFET Approach

A new functional device, called the Neuron MOS transistor ( $\nu$ MOS) because of its capability of simulating neurons, has been found to be very useful in implementing multi-level Boolean logic [121].

---

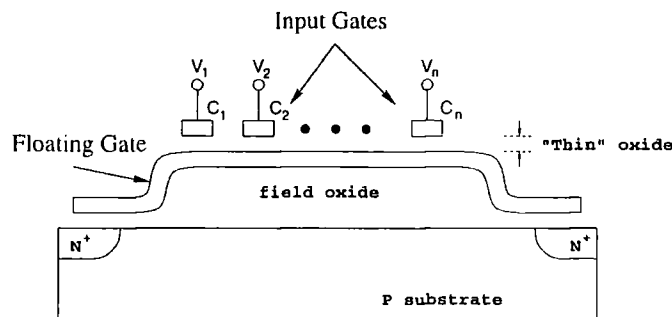


Figure 4.7: Schematic representation of a neuron MOS transistor

In diagram 4.7 the transistor is a normal MOS transistor with a floating gate to store charge, its main difference being its number of input gates. This structure is very similar to the charge coupled device, but instead of shunting charge from one region ( $N^+$ ) to the other using two phase clocks, it is used like a normal transistor. The transistor operates internally in a multi-voltage regime but its input gates operate in normal binary. It is a voltage operated device and the voltage thresholds are controlled by the capacitance of the input gate to the floating gate ( $C_n$ ) and the capacitance from the floating gate to the source-drain channel. Thus each input gate can have a different threshold voltage.

The transistor has been found to operate at low power over wide frequencies but is 10-20% slower than CMOS. The transistor has larger gate capacitance and provides better optimisation of drive current [122]. In [120] important device characteristics are measured and a new multiplier design manufactured [146] which has been simulated up to 500 MHz with nearly constant power consumption of  $100\mu\text{W}$  over 0-200 MHz.

The device has a large number of gate inputs and this could cause problems in large chips because of routing them over large distances. The other problem is getting the oxide layers produced smoothly with the standard CMOS process.

This transistor is like the floating gate transistors used in electrically erasable read only memory chips (FLASH memory). Using Fowler-Nordheim tunnelling [73] and clever incremental charge injection, a 256 level per cell analogue storage device has been manufactured for low cost audio applications [7].

## 4.5 DAD Logic

In this section, I will introduce a new concept in digital design which should be suitable for implementation in the standard CMOS logic technology.

The major driving force of this concept, is its use of components to the full; unlike CMOS which shunts current between two thresholds, my concept uses the behaviour of the transistor to reduce component count in the same way as analogue electronics perform their functions. In CMOS digital circuits, the transistor is operating in the saturation regime to perform the switching function. In analogue electronics the designer can use the sub-threshold region to perform functions which can consume low power [89] by utilising the transistor's nonlinearity which arises from its physical structure. These analogue components often out perform the equivalent function implemented using the digital technique in terms of power, speed and area. This is despite the individual transistors being larger. What is more surprising is that the analogue multiplier based on Gilbert's translinear loop was invented in 1968 and using the bipolar technology could operate at 1 GHz with four transistors [37]. It has taken thirty years for digital technology to approach this speed, whilst analogue design has improved commensurably by the improvements in fabrication and device scaling. Another reason for using analogue CMOS technology is that digital techniques at sub-micron technologies resemble that of analogue because of the need to amplify useful signals in a torrent of noise. It looks like our approximation of a switch is becoming more difficult to work with, because of the crosstalk generated in the wiring and the reduced voltage thresholds required.

However, if designers worked entirely in analogue we would be plagued by noise and signal recovery would not be possible. DAD Logic is my approach to solve this conundrum. **Digital-Analogue Duality Logic** uses analogue building blocks operating over a continuum of voltages and currents to generate two or more discrete logic levels. Notice that there are similarities between analogue and digital methodologies and the correspondence principle in Quantum Mechanics. In actual fact, my work is directly inspired by quantum mechanical theory.

My idea is to use DAD Logic to create discrete equidistant levels and to confine analogue values to these levels. This acts like the hysteresis in digital circuits acting on small drifts in the input voltages to pull the output to one of two values. Noise immunity is achieved providing the logic can "lock" to the relevant level in the presence of noise. Another name for this process is a quantiser, whose function is shown diagrammatically in figure 4.8.

---

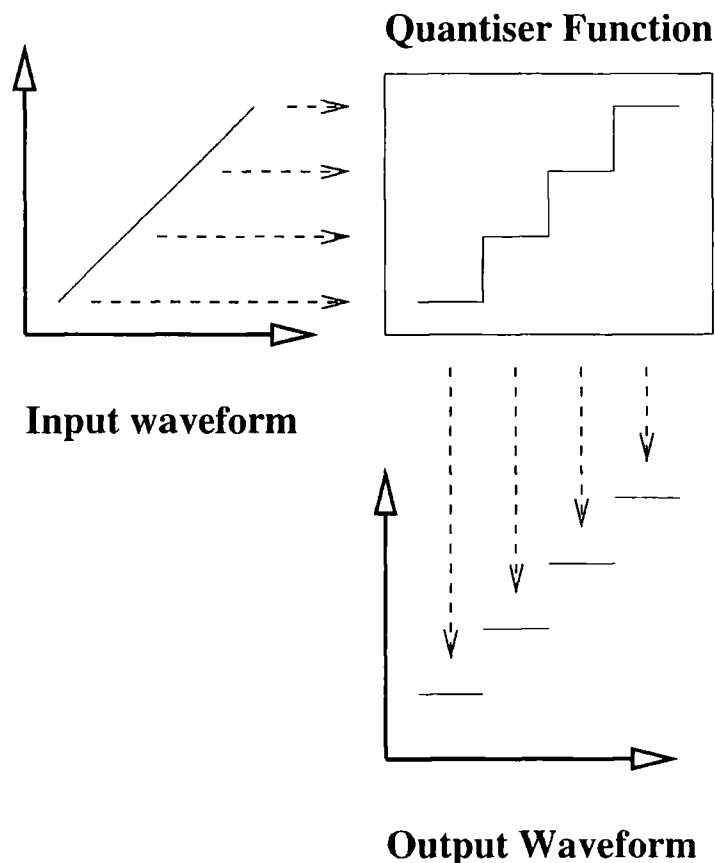


Figure 4.8: Functional Diagram of DAD Logic Quantiser Operation

Figure 4.8 shows a linear ramp waveform being applied to the quantiser function and the output waveform is discontinuous with equidistant values. Notice that the quantiser function resembles a staircase and input values which vary over the vertical parts of the step function become horizontal at the output. The approach shown above resembles work done on a multilevel neural network for Analogue to Digital conversion [153] and also on an implementation using resonant tunneling diodes [74]. The first reference implements each step by additional circuitry with variable offsets, whilst the latter implements it with a single device but in a different technology. My approach is to sense the input value and home in on the nearest discrete value; consequently this device is scalable to many levels with **no** increase in circuitry. The real beauty and power of this approach is that it allows designers to use analogue circuits to perform the same function as do complex digital algorithms. Some examples of complex digital algorithms are those required by exponential, division, multiplication and natural logarithm operators.

In the forthcoming sections, I will delve into the internal structure of the DAD Logic



functional block.

#### 4.5.1 Mathematical Theory behind DAD Logic

DAD Logic requires a mathematical model to converge a range of values to discrete ones. Luckily mathematicians have already discovered the tools to do this: differential equations. The solutions of these equations are known as characteristic values (eigenvalues) and characteristic functions (eigenfunctions). In most electronic engineering applications, we normally use initial value differential equations which have many solutions. The differential equations suitable for DAD Logic are the ones which have conditions imposed at the ends of a medium. A classic example is a vibrating string tied at its ends having frequencies which are related by integer multiples of the lowest frequency (the fundamental). These equations are known as boundary value differential equations.

The special class of functions needed for DAD Logic are the ones which have two boundary values (or limits); the equations governing this behaviour are known as Sturm-Liouville Theory <sup>2</sup>.

A second order linear differential operator  $\hat{L}$  defined on the interval  $a \leq x \leq b$  is said to be in self-adjoint form if

$$\hat{L} = \frac{d}{dx} \left( p(x) \frac{d}{dx} \right) + q(x)$$

Where  $p(x)$  is any function with continuous first order derivatives such that  $p(x) > 0$  or  $p(x) < 0$ , for all  $x$  in the interval  $a < x < b$  and  $q(x)$  is an arbitrary continuous function on the interval  $a \leq x \leq b$ .

An equation of the form

$$\hat{L}y = \lambda y$$

is called an eigenvalue equation of the differential operator  $\hat{L}$ . This equation has a trivial solution when  $y = 0$  and “useful” solutions when  $y$  is not equal to zero. The latter solutions only exist for certain values of  $\lambda$ , which are called eigenvalues of  $\hat{L}$ . To each such eigenvalue there may be one or more solutions  $y_\lambda$ , which are called eigenfunctions corresponding to the eigenvalue  $\lambda$ .

The generalised Sturm-Liouville problem, or system, is a second order homogeneous linear differential equation of the form

$$\frac{d}{dx} \left[ p(x) \frac{dy}{dx} \right] + (q(x) - \lambda w(x))y = 0$$

---

<sup>2</sup>Named after Jacques Charles Francois Sturm (1803-1855), a Swiss mathematician who worked in France with his friend Joseph Liouville (1809-1882), a French mathematician.

---

defined on the interval  $a \leq x \leq b$ , along with a pair of homogeneous boundary conditions such that the eigenfunctions corresponding to distinct eigenvalues for the adjoint operator are orthogonal. The parameter  $w(x)$  is called the weight function which is assumed positive for all  $x$  in the interval.

The most important property which will be utilised is that the eigenvalues are real and can be arranged in an ascending order of magnitude separated by an index of the eigenvalue. This index is the number of zeros on the open interval  $a < x < b$  for the particular eigenfunction. The other important construct which will be of use to find an eigenvalue given an estimated one is the Differential Green's Identity.

Suppose that in a regular Sturm-Liouville Problem on an interval  $a \leq x \leq b$  the coefficient functions  $p(x)$ ,  $q(x)$  and  $w(x)$  depend smoothly on some parameter  $v$ . Here we take this to mean that  $\frac{\partial p}{\partial v}$ ,  $\frac{\partial q}{\partial v}$  and  $\frac{\partial w}{\partial v}$  are continuous in both  $x$  and  $v$ . Let  $u$  be for each  $v$  a solution of the parametrised Sturm-Liouville differential equation and let  $u$  depend smoothly on  $x$  and  $v$ ; and likewise for the other solution to the equation  $pu'$ . Then

$$\frac{\partial \lambda}{\partial v} \int_c^d u^2 w dx = \left[ pu' \frac{\partial u}{\partial v} - u \frac{\partial(pu')}{\partial v} \right]_c^d + \int_c^d \left\{ \frac{\partial q}{\partial v} u^2 - \lambda \frac{\partial w}{\partial v} u^2 + \frac{\partial p}{\partial v} (u')^2 \right\} dx \quad (4.1)$$

The limits  $c$  and  $d$  are valid across the interval  $a \leq x \leq b$  and are more likely to be the interval limits themselves.

After this brief mathematical tour, it is time to look at some useful differential equations which have unique eigenvalues which are suitable for DAD Logic.

#### 4.5.2 Harmonic Oscillator

My first attempt at finding a suitable differential equation having equi-spaced states (levels) was the one dimensional quantum harmonic oscillator. This is the Schrödinger equation of a particle trapped in a parabolic well with the constant factors set to unity.

$$\frac{\partial^2}{\partial x^2} \psi(x) + (E - x^2)\psi(x) = 0 \quad (4.2)$$

Subject to the boundary conditions  $-\infty \leq x \leq +\infty$  and at these limits the wavefunction should be zero. Thus equation 4.2 can be transformed to the following equation which has power series solutions.

$$\frac{\partial^2}{\partial \xi^2} \mathcal{H}(\xi) - 2\xi \frac{\partial}{\partial \xi} \mathcal{H}(\xi) + (\epsilon - 1)\mathcal{H}(\xi) = 0 \quad (4.3)$$

The solutions for equation 4.3 are known as Hermite polynomials and have eigenvalues of  $\epsilon = E = (2n + 1)$  and  $n = 0, 1, 2, \dots$ . Consequently, from a simple equation, equally spaced levels result and are shown graphically in figure 4.9. This figure shows the solutions to equation 4.2 with  $\phi(x)$  and  $|\phi(x)|^2$  plotted against  $x$  and the polynomials plotted relative to their respective energy levels or eigenvalues. Note that the smallest eigenvalue is  $\frac{1}{2}$  and not zero.

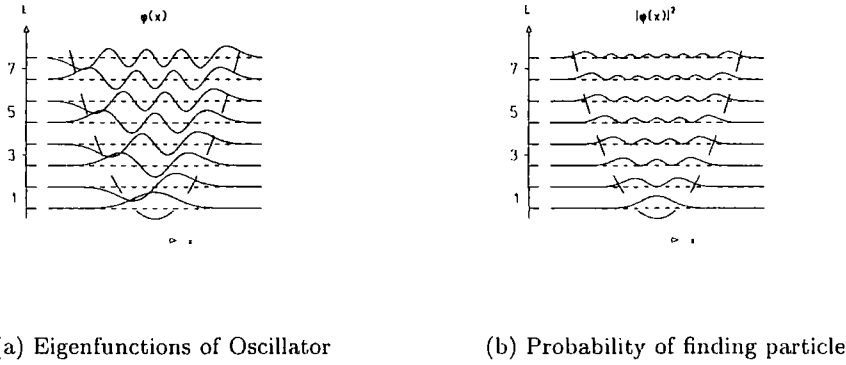


Figure 4.9: Stationary state solutions for the Harmonic Oscillator Potential

In the following section, I will introduce another differential equation system which has the ability of generating equidistant levels with sign. This type of system would be very useful to implement in signed number arithmetic transforms using DAD Logic.

#### 4.5.3 Bipolar Level System

Bipolar levels require a differential equation whose solutions are periodic. Thus trigonometric functions must be involved somewhere. If we express the three dimensional Schrödinger equation as spherical coordinates having a central force potential, the following expression results

$$-\frac{1}{r^2} \left[ \frac{\partial}{\partial r} \left( r^2 \frac{\partial \Psi}{\partial r} \right) + \frac{1}{r^2 \sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial \Psi}{\partial \theta} \right) + \frac{1}{r^2 \sin^2 \theta} \frac{\partial^2 \Psi}{\partial \phi^2} \right] + k^2 f(r) \Psi = E \Psi$$

This equation can be separated into three variables  $R(r)\Theta(\theta)\Phi(\phi)$ . The radial part, which includes the central potential and the energy, is independent from the polar

dependent variables and is unimportant for DAD Logic. Manipulating the equation for the polar parts, we obtain two differential equations being

$$\frac{1}{\Phi} \frac{d^2 \Phi}{d\phi^2} = -\alpha^2$$

and

$$\frac{1}{\Theta} \left[ \sin \theta \frac{d}{d\theta} \left( \sin \theta \frac{d\Theta}{d\theta} \right) + \beta \sin^2 \theta \Theta \right] = \alpha^2$$

Note the two separation constants ( $\alpha^2$  and  $\beta$ ). It turns out that the last two equations appear in quantum mechanics as angular momentum operators. These equations can be expressed as eigenvalue equations as follows :-

$$L^2 Y(\theta, \phi) = \beta \hbar^2 Y(\theta, \phi)$$

and

$$L_z Y(\theta, \phi) = -i\hbar \frac{\partial Y}{\partial \phi} = m Y(\theta, \phi)$$

Here  $\beta$  is defined to be  $l(l+1)$  and the  $L_z$  operator is independent of  $\theta$ . Also  $l$  is an integer belonging to  $l = 0, 1, 2, \dots$  and  $m$  is an integer too with these ranges:  $m = 0, \pm 1, \pm 2, \dots$ . One last point is that  $\theta$  must be periodic with a period of  $2\pi$  so that  $Y$  is single valued. What we have stumbled across is the Legendre and Associated Legendre equations and their polynomial solutions.

$$(1-x^2) \frac{d^2}{dx^2} P_n^m(x) - 2x \frac{d}{dx} P_n^m(x) + \left[ n(n+1) - \frac{m^2}{1-x^2} \right] P_n^m(x) = 0 \quad (4.4)$$

The Associated Legendre differential equation is defined as shown in equation 4.4 subject to the boundary condition  $-1 \leq x \leq 1$ . The polynomials are represented by  $P_n^m(x)$  and the equation becomes the Legendre equation if  $m = 0$ . The value of  $m$  is confined in the range  $-n \leq m \leq n$ . In polar coordinates, spherical harmonics arise which contain associated Legendre polynomials as a function of  $\cos \theta$ .

On occasion, DAD Logic may require bipolar integers without a zero value, but does such a condition exist? It so happens that a fourth dimensional Laplace's equation will generate the required equation [98], or, by reformulating angular momentum in terms of rotations, the  $L^2$  operator becomes the following.

$$\left\{ \frac{\partial}{\partial \beta^2} + \cot \beta \frac{\partial}{\partial \beta} + \frac{1}{\sin^2 \beta} \left( \frac{\partial^2}{\partial \alpha^2} + \frac{\partial^2}{\partial \gamma^2} - 2 \cos \beta \frac{\partial^2}{\partial \alpha \partial \gamma} \right) + l(l+1) \right\} \mathcal{D}_{mk}^{(l)}(\alpha \beta \gamma) = 0 \quad (4.5)$$

Equation 4.5 is subject to  $0 \leq \alpha < 2\pi$ ,  $0 \leq \beta < \pi$  and  $0 \leq \gamma < 2\pi$  and instead of associated Legendre polynomials, hypergeometric polynomials now arise subject to a differential equation [24] in  $\beta$ .

$$\left\{ \frac{d^2}{d\beta^2} + \cot \beta \frac{d}{d\beta} - \frac{m^2 + k^2 - 2mk \cos \beta}{\sin^2 \beta} + l(l+1) \right\} d_{mk}^{(l)}(\beta) = 0 \quad (4.6)$$

The result of equation 4.6 is that  $l$  now can take integer and half integer values and  $m$  (and  $k$ ) has  $2l+1$  values ranging from  $-l$  to  $l$ . The eigenfunctions of the equation now describe a symmetric top with  $m$  being the component of angular momentum along the body fixed symmetry axis and  $k$  is the component along an axis fixed in space. For DAD Logic it is useful to allow  $m = k$  and thus reduce the number of parameters to  $l$  and  $m$ . Here  $l$  selects the required range and  $m$  is the actual quantised values.

Thus bipolar equidistant values are obtained with the option of including zero, dependent on a parameter which can take integer or half integral values.

#### 4.5.4 Value Determination through Approximation

In this part, I will investigate some methods which I believe to show promise in the following problem. Given an estimate of an eigenvalue, converge to the actual (closest) eigenvalue as quickly and accurately as possible. This is the most crucial aspect of DAD Logic and unfortunately it is the part which still needs the most research.

Here, we will look at analogue methods using finite element techniques and then move onto two approaches which allow convergence to an eigenvalue. These last two approaches are based on programmed digital computation.

##### 4.5.4.1 Analogue Eigenvalue Methods

As mentioned in the last section, differential equations are modeled using finite difference techniques. So what are they? Imagine a horizontal line and a physical phenomena which smoothly varies along it. Discretise this line into equal subsections having a width  $\delta$ . Label any three nearest subsections 2, 0 and 1 going from left to right. Now label the transient or steady-state values of the dependent variable (the physical phenomena) and call it  $y$ . At each of the adjacent nodes, the value is  $y_0$ ,  $y_1$  and  $y_2$ . The first derivative (gradient) of this dependent variable with respect to  $x$  is approximated by taking the difference between the value of  $y$  at two adjacent nodes and dividing by  $\delta$ , (the node spacing). The approximation at the mid-points

---

between 2 and 0 or 0 and 1 respectively and are equal to

$$\left(\frac{dy}{dx}\right)_{x=1-0} = \frac{y_1 - y_0}{\delta}$$

$$\left(\frac{dy}{dx}\right)_{x=0-2} = \frac{y_0 - y_2}{\delta}$$

The first of these equations is known as the **forward difference approximation** and the latter is known as the **backward difference approximation**. Now, the first derivative at node 0 is obtained by averaging the first derivatives of the midpoints between 1 and 0 and then between 0 and 2 and is as follows

$$\left(\frac{dy}{dx}\right)_{x=0} = \frac{(dy/dx)_{x=1-0} + (dy/dx)_{x=0-2}}{2} = \frac{y_1 - y_2}{2\delta}$$

The second derivative of  $y$  with respect to  $x$  is defined as the rate of change of the first derivative and is the difference between the first derivatives at the midpoints and dividing by the grid spacing. So

$$\left(\frac{d^2y}{dx^2}\right)_{x=0} = \frac{(dy/dx)_{x=1-0} - (dy/dx)_{x=0-2}}{\delta} = \frac{y_1 + y_2 - 2y_0}{\delta^2}$$

This approach can be extended to many dimensions, which is required for partial differential equations.

Now, analogue methods use differential operator discretisation and set value by using electrical circuit elements. These elements are the components which are between the nodes in the approximation. This approach has been demonstrated effectively by [9, 70] but requires manual adjustment of the component values which is incredibly slow. Also, the network is large and requires proper termination for equations which have boundaries set to infinity. However, they can perform multiple variable operation simultaneously which occurs with partial differential equations. Other useful works on analogue computation are [57, 64, 65, 136, 148].

#### 4.5.4.2 The Shooting Method

Before plunging into this method, additional theory is required. Firstly, any high order linear differential equation can be reduced to many lower order versions. This is very useful because Sturm-Liouville theory predicts that eigenvalue determination requires only a first order differential equation, its eigenfunctions are obtained using quadrature and the other first order differential equation.

---

The Prüfer transformation transfers cartesian based second order ordinary differential equations into polar form. The resulting equations in  $r$  and  $\theta$  are as follows :-

$$\frac{d\theta}{dx} = \frac{1}{p} \cos^2(\theta) + (\lambda w - q) \sin^2(\theta) \quad (4.7)$$

$$\frac{1}{r} \frac{dr}{dx} = \left\{ \frac{1}{p} - (\lambda w - q) \right\} \sin(\theta) \cos(\theta) \quad (4.8)$$

The regular boundary conditions at  $a$  and  $b$  are transformed to the conditions  $\theta(a) = \alpha$  and  $\theta(b) = \beta$ . Here  $\alpha = \arctan(\frac{a_2}{a_1})$  and  $\beta = \arctan(\frac{b_2}{b_1})$ . The  $a_n$  and  $b_n$  arise because second order equations have two solutions and thus two coefficients exist at each of the boundary conditions.

So why transform the equation into Prüfer form ? In this form, there is only one solution for any choice of  $\alpha$  and  $\beta$ . Therefore, we can choose these coefficients in such a way as to make the solution of the differential equation the desired eigenvalue. Another useful point is that  $\alpha$  and  $\beta$  are determined upto a multiple of  $\pi$ . An appropriate choice of this multiple of  $\pi$  determines the number of times around the origin the eigenfunction travels, and becomes the number of zeros in the solution.

The principle of a shooting method for finding an eigenvalue of a regular Sturm-Liouville problem is quite simple. The differential equation is solved as an initial value problem in some form, here Prüfer, over the range  $a \leq x \leq b$  (in pre-transformed form) for a succession of trial values of  $\lambda$ ; these are automatically adjusted till the boundary conditions at both ends are satisfied at once. At this point, the eigenvalue is found. Generally, we may want to shoot from the two ends to a point somewhere in the middle. To do this requires the definitions of both a left hand and a right hand boundary conditions. The left hand ( $U_l(x, \lambda)$  at  $x = a$  is

$$p \frac{du}{dx} = a_1, u = a_2$$

For the right hand boundary, replace  $a$  with  $b$ . At the end point, a function  $D(\lambda)$  is defined and is known as a **miss-distance function** and describes the extent to which the boundary condition fails to be satisfied at that point. The eigenvalues are the zeros of this function. A natural choice for the miss-distance function is the Wronskian determinant and is

$$p \frac{dU_l(c, \lambda)}{dx} U_r(c, \lambda) - \frac{dU_r(c, \lambda)}{dx} U_l(c, \lambda) \quad (4.9)$$

Unfortunately, the miss-distance equation 4.9 is always an oscillating function and causes problems with root finding algorithms. However, these problems are solved

elegantly with Prüfer equations 4.7 and 4.8. In this case the boundary conditions become  $\theta(a, \lambda) = \alpha$  and  $\theta(b, \lambda) = \beta + k\pi$ .

The following algorithms below summarise the shooting method for eigenvalue determination. First I give the miss-distance computational algorithm.

---

**Algorithm 1** Miss-distance computation

---

Input:     Coefficient functions  
            Initial value of lambda  
            An accuracy specification

Output:    Miss-distance  $D(\lambda)$   
            Information on its accuracy

Method:

- (1) Choose an appropriate point,  $c$ , to shoot too. This may be done on the basis of a preliminary analysis and can be subsequently adjusted.
- (2) Set up the boundary values of  $\theta(a)$  and  $\theta(b)$
- (3) Integrate  $\theta'$  equation to compute theta left hand solution ( $\theta_L$ ) and theta right hand solution ( $\theta_R$ )
- (4) Setup the miss-distance to be left hand solution - right hand solution at point  $c$  ( $D(\lambda) = \theta_L(c) - \theta_R(c)$ )

---

Next is the eigenvalue determination process.

---



---

**Algorithm 2** Root finding for eigenvalue determination

---

Input:     Index  $k$  of the required eigenvalue  $\lambda$   
           Initial guess(es) for  $\lambda$

Output:     $\lambda$   
           error estimation information

Method:

(1) solve  $D(\lambda) = k\pi$  with an appropriate root finder

---

Note that the  $k$ 'th eigenvalue  $\lambda_k$  is the unique value such that the miss-distance equation is

$$D(\lambda_k) = k\pi$$

for  $k = 0, 1, 2, \dots$

The problem with this approach is how to handle **stiffness** which afflicts the  $\theta$  equation in parts of the range when  $\lambda w - q$  is a lot less than zero and negative. This gives arise to exponentially growing eigenfunctions which can easily create numerical overflow or in the analogue domain, saturation.

In the next subsection, the Pruess method is presented and is constructed in a similar way to the finite element approach discussed in section 4.5.4.1.

#### 4.5.4.3 The Pruess Method

The Pruess method attempts to solve a problem by reducing it to a simpler problem which is an approximation of the former problem. This is achieved by using piecewise constant mid-point approximation on the Sturm-Liouville coefficients  $p$ ,  $q$  and  $w$ . Thus the coefficients are split up into sub intervals and its value is taken to be the mid-point value of the sub interval. This is essentially a finite element approximation and it reminds me of zero order hold, which appears as a sampling process model in digital signal processing.

Let  $P$ ,  $Q$  and  $W$  have values  $p_i$ ,  $q_i$  and  $w_i$  respectively in the  $i$ 'th interval between  $x_{i+1}$  and  $x_i$  over  $i = 1, \dots, n$ . A solution to this approximation of the Sturm-Liouville problem is of the form

$$U(x) = c_i F_i(x) + d_i G_i(x)$$


---

Where  $F_i, G_i$  are independent solutions of

$$-\frac{d^2U}{dx^2} = k_i U$$

and  $k_i$  is a constant equal to

$$k_i = \frac{\Lambda w_i - q_i}{p_i}$$

The explicit form of a solution is the following, provided  $w_i = \sqrt{|k_i|}$

$$U(x) = U(x_{i-1}) \Psi_i(x - x_{i-1}) + \left( P \frac{dU}{dx} \right) (x_{i-1}) \Phi_i(x - x_{i-1}) / p_i$$

Where

$$\Psi_i(s) = \begin{cases} \cos(w_i s) & k_i > 0 \\ 1 & k_i = 0 \\ \cosh(w_i s) & k_i < 0 \end{cases}$$

$$\Phi_i(s) = \begin{cases} \frac{\sin(w_i s)}{w_i} & k_i > 0 \\ s & k_i = 0 \\ \frac{\sinh(w_i s)}{w_i} & k_i < 0 \end{cases}$$

Let  $U_i = U(x_i)$ ,  $PU'_i = PU'(x_i)$  and  $x = x_i$ , then

$$\begin{pmatrix} PU'_i \\ U_i \end{pmatrix} = T_i \begin{pmatrix} PU'_{i-1} \\ U_{i-1} \end{pmatrix} \quad i = 1, \dots, n$$

The transfer matrix is defined as

$$T_i = \begin{pmatrix} P_i \Phi'_i & P_i \Psi'_i \\ \Phi_i & \Psi_i \end{pmatrix} (h_i)$$

The value of  $h_i = x_i - x_{i-1}$  and  $T_i^{-1}$  is equivalent to replacing  $h$  by  $-h$  in  $T_i$  which is needed for the algorithm below.

**Algorithm 3** Simple Pruess Algorithm

1. Choose a meshpoint  $x_m$ , where  $0 \leq m \leq N$  as a matching point,  $c$
2. Set up initial values  $\begin{pmatrix} PU'_0 \\ U_0 \end{pmatrix}$  satisfying the boundary condition at  $a$ , and  
initial values  $\begin{pmatrix} PU'_N \\ U_N \end{pmatrix}$  satisfying the boundary condition at  $b$
3. For a trial value of  $\lambda$

- (a) For  $i = 1, \dots, m$  do

$$\begin{pmatrix} PU'_i \\ U_i \end{pmatrix} = T_i \begin{pmatrix} PU'_{i-1} \\ U_{i-1} \end{pmatrix}$$

- (b) Store the resulting values  $PU'_L(c), U_L(c)$

- (c) For  $i = N, \dots, m+1$  do

$$\begin{pmatrix} PU'_{i-1} \\ U_{i-1} \end{pmatrix} = T_i^{-1} \begin{pmatrix} PU'_i \\ U_i \end{pmatrix}$$

- (d) Store the resulting values  $PU'_R(c), U_R(c)$

- (e) Form a miss distance  $D(\lambda)$  by comparing  $PU'_L(c), U_L(c)$  with  $PU'_R(c), U_R(c)$

4. Adjust  $\lambda$  to solve the equation  $D(\lambda) = 0$

---

The adjustment of  $\lambda$  requires a zero count procedure. This is used to count the number of oscillations in the solution and *home in on* the required eigenvalue. If the algorithm was recast via the Prüfer transformation, the counting procedure is intertwined in the  $\theta$  variable.

To select a relevant mesh, the differential Green's identity as mentioned in equation 4.1 is used as an error control. It is necessary to use the standard idea of equidistribution to choose a mesh which satisfies the absolute value of this identity. The first part of the right hand side of equation 4.1 is equal to zero in this case. The integral on the left hand side is assumed to equal one, this implies a normalised eigenfunction.

---

#### 4.5.4.4 Important Issues regarding Eigenvalue Determination

The shooting and Pruess methods mentioned earlier are missing important details. This was done on purpose because they are generally designed to run on a stored program digital computer where everything is quantised. The details of these methods, including FORTRAN source code, can be found in the excellent book [111] "Numerical Solution of Sturm-Liouville Problems".

I can see that using these methods would make DAD Logic a reality. However, reducing the algorithms to allow implementation in analogue VLSI elements has not yet been investigated. It would be beneficial if these algorithms assumed the boundary conditions were fixed and modifications occurred inside these ranges. However, this could only work for a restricted range; or an infinite range if a transformation were possible. The current codes work by changing the boundary conditions in order to find the eigenvalue.

It would be wonderful if these methods could work in finite time, and convergence was always the same regardless of the eigenvalue. If a mesh selection is required for analog VLSI, there is a good chance that the circuitry could perform a constraint more efficiently than a program, by utilising the physical behaviour of the circuit elements themselves.

## 4.6 Summary

In this chapter, I have highlighted novel VLSI technologies and architectures which are or may be of benefit to applications using digital signal processing. Gallium-Arsenide based technology, Josephson Junction SQUIDS and low power designs were briefly discussed. The rest of the chapter concerned different architectures having the common goal of encoding multiple values on a single wire, unlike digital which has only two values.

Current-mode logic primitives were mentioned and then negative differential resistance based methods were discussed. These included both technological and simulated approaches. Then voltage based logic was discussed in terms of technological (Neuron MOSFET) and Operational Amplifier based.

The remainder of the chapter introduced my new idea which I have called DAD Logic. This is based on eigenvalue solutions to some second order differential equations which have two point boundary conditions. Methods of solving these equations were discussed using analogue and two programmed based approaches. The last two methods have been shown to be very accurate in numerical simulation of a whole

---

range of problems [111] and it was my idea that this could be harnessed to the needs of DAD Logic. However, at the time of writing this thesis, an analogue VLSI form of eigenvalue determination remains elusive. Consequently DAD Logic, which has enormous potential, will remain a theoretical idea at present.

---

---

## CHAPTER 5

# VLSI requirements for synthesis

---

In this chapter, the basic structures for performing DSP operations will be discussed. A CORDIC based sinusoidal generator implemented in the ES2 0.7 $\mu$ m CMOS process will be used as a case study. In addition, a bit-serial radix-2 two's complement recursive sinusoidal generator will be used for comparison.

### 5.1 DSP Systems Design

A Digital Signal Processor is a specialised microprocessor optimised for intensive arithmetic operations and all instructions take a single clock cycle. The DSP system consists of an Arithmetic Logic Unit ( ALU ), a Multiply-Accumulate unit ( MAC ), a program sequencer, data address generators and architectures optimised for fast memory access for data and instructions [43].

A simplified block diagram of a DSP is shown in figure 5.1, which includes the PMA ( Program Memory Address ), DMA ( Data Memory Address ), PMD ( Program Memory Data ) and DMD ( Data Memory Data ) buses.

#### 5.1.1 Multiplier Constraints

The multiplier ( Enclosed in the MAC ) must operate at a speed comparable to the RAM access time; this normally implies a full parallel multiplier. The common number formats are two's complement, unsigned, fractional or integer and mixed-mode operation. There are tradeoffs between fixed-point arithmetic and floating point arithmetic and extended or double precision capabilities.

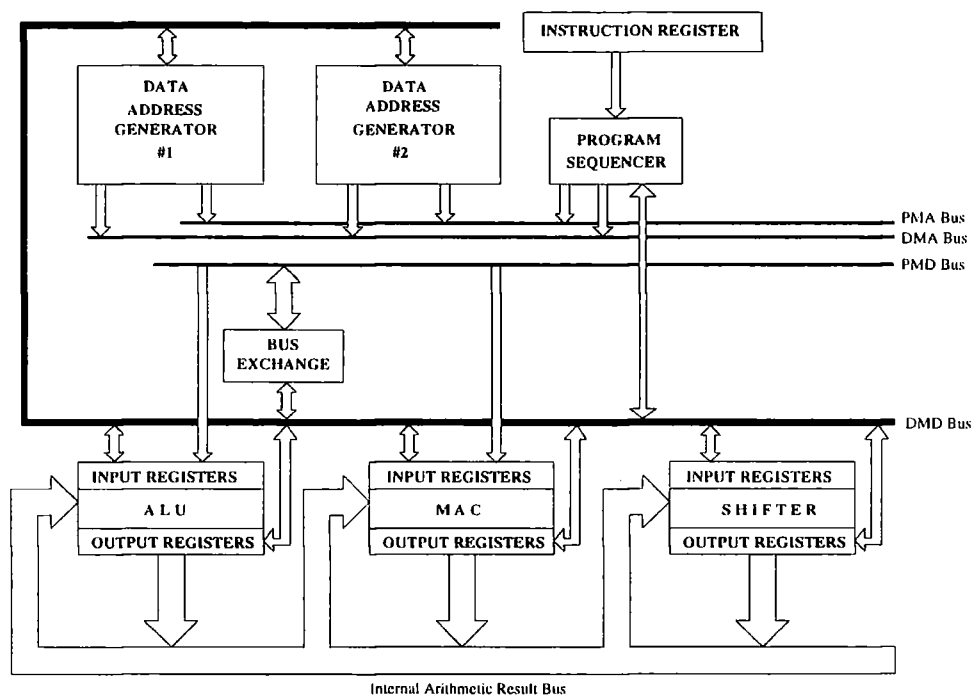


Figure 5.1: A Simplified Structure of a DSP Processor

The multiplier should complete the operation in a single cycle and its Input-Output registers should be configured for pipelined operation ( Latched registers ) or minimum latency operation ( known as transparent ). A pipelined processor has the capability of maximising hardware usage but with a fixed initial delay or latency. This initial delay is the time taken for the first datum to pass through all the processing elements. By using registers in between the functional elements, each element will be operating in parallel. Therefore, once the pipeline is full, the output data will appear once every clock cycle.

### 5.1.2 MAC Constraints

This part of the DSP is very important and must operate in a single cycle. The Accumulator should have adequate guard/extension bits to ease the possibility of overflow during the arithmetic operations. The MAC should detect overflow before it occurs and allow saturation arithmetic. It should support unbiased rounding and have internal feedback paths to lessen the overhead on loops.

The unit can either have optimal throughput ( pipelined processor ) or have transparent throughput ( un-pipelined processor ).

### 5.1.3 ALU Constraints

The Arithmetic Logic Unit should allow adequate accuracy and chain-ability for high precision computation. This normally implies a bit-slice design. When computing in double/extended precision the ALU should operate with the minimum lost time. It would be useful if the ALU provides division and/or multiply capability.

The arithmetic functions which an ALU should have are :-

ADD, SUBTRACT, ADD WITH CARRY, SUBTRACT WITH CARRY and ABSOLUTE VALUE.

The logic functions an ALU should have are :-

AND, OR, EXOR, BARREL SHIFTER and LOGICAL NEGATION.

The status flags the ALU would generate include the following :-

ZERO, EQUAL, LESS THAN, GREATER THAN and CARRY.

The ALU must provide efficient data movement and computation. This is achieved by either single-cycle data moves with arithmetic operation, or two operands loadable into the ALU in the same cycle.

The ALU should be able to access a large number of registers configured as a dual-set for context switching ( for data storage during interrupts ) or as a register file for fast access and storage of intermediate results. The ALU instruction set should include conditional operations. i.e. add and shift left if flag is set. The ALU should have adequate feedback pathways to provide an output to input link for an accumulator or provide I/O links to a shifter and ALU. This module should be designed to provide pipeline or transparent flow onto the input registers.

### 5.1.4 Data Address Generator

This module provides data accessing and must have the ability to address in linear, circular ( modulo ) and bit-reverse addressing. The bit-reverse option is required for most FFT algorithms. The generator should provide simple logic functions and masking; this is useful for interpolation and look-up tables. An optional extra would be a shifter to aid the FFT twiddle factor generation.

### 5.1.5 Program Sequencer

This module is not pipelined because a pipelined program sequencer will cause program failure. The main elements of this module are :-

- Program Counter.



- Conditional Logic Tester.
- Stack Counter for subroutine/interrupt return addresses, counter values and jump addresses.
- Loop Counters.
- Zero Overhead Conditional Branch Capability.

### 5.1.6 Memory

The memory should be dual-ported and the DSP device should have separate program and data address and data buses ( The Harvard Architecture ), accessed via the Data Address Generators. The memory mechanism would benefit by cache memory to allow most recent instructions to be stored on chip, providing a speed improvement; and support registers to be used as a scratch pad thus reducing access to external memory.

The memory control unit could have FIFO modules as a means of interfacing between slower memory and peripherals or between faster processors.

### 5.1.7 Architecture Summary

A DSP processor requires a controller, interface logic for input and output streams, storage and arithmetic processing elements. The minimum arithmetic elements are a MAC, though a division element is useful in Durbin recursion, matrix inversion and ray-tracing. However, for ASIC design, the designer maps the particular algorithm to the minimum hardware. So division is neglected due to finite space on a silicon die, but can be implemented using Newton-Raphson iteration.

Provided the algorithm computes operations regularly in time, having an almost systolic nature, the designer can implement it using arithmetic logic elements and a very simple controller. It is also convenient to assume that the completed device would perform better than an equivalent implementation on general purpose DSPs. Alternatively, a DSP core with restricted functionality could be built and the algorithm coded in software stored on chip. This is known as microcode. However, this thesis is concerned with full-custom ASIC designs.

Useful design criteria and analysis can be found in [81] for general purpose and custom DSP design. This book goes into some detail in multi-processor design and arithmetic processor design using flowgraphs and clock retiming. Unfortunately there are errors in diagram labelling and missing references.

---

## 5.2 Datapath length criteria

Before the comparative study between CORDIC and a serial based recursive oscillator, analysis of the size of the data is required. Two's complement arithmetic will be used throughout this investigation.

In chip design, the designer must ensure that arithmetic operations do not overflow and produce the wrong answer in the arithmetic sense. In floating point arithmetic, the designer is restricted due to the available bits allocated to the exponent and mantissa. Hence some sort of arithmetic truncation or rounding is required. Word-length increases for addition and multiplication are as follows :-

1. For  $N$  Additions, the designer allows  $\log_r N$  extra bits in the accumulator. This will guarantee no overflow.
2. For multiplication of two  $N$  bit numbers produces a  $2N$  bit result. The output could be rounded or truncated, but computation of integrals or sum of products would cause numerical errors.

Computing integrals with numbers of nearly full-scale without overflow can be determined by the following :-

- All the numbers have the same large value, so no cancellation occurs. Therefore the integral can be up to  $N$  times as large as any number, requiring  $\log_r N$  bits extra.
- The inputs are of white noise. Therefore the integral process would go up by  $\sqrt{(N)}$  and hence requires  $\log_r \sqrt{N}$  bits extra.
- The input is assumed to be sinusoidal of 0.99 full-scale. There would be no overflow, but there would be a slow drift caused by the build-up of truncation or rounding errors.

### 5.2.1 Overflow and underflow reduction

The designer can decide to allow intermediate products to be stored in an accumulator register having more precision than the input and output registers. This reduces the chance of overflow in additions. Double precision is the term given to an accumulator output which has two times the number of bits as the input and/or output. Extended precision is a compromise, having less bits than double precision, but allowing accumulation of  $r^g$  additions. Hence for two  $b$  bit numbers, extended

---

precision has  $b + g$  bit storage. Therefore, the least significant part of the multiplication is neglected. This is advantageous because the accumulator's word-length is reduced requiring less transistors to design it, and DSP operations are noise tolerant. This can be seen in equation form by writing a  $b$ -bit binary number as  $[HL]$  and rewriting the multiplication in double precision format  $(\frac{b}{2})$ -bit binary number

$$\begin{aligned}[HL]_1 \times [HL]_2 &= (2^{\frac{b}{2}}H_1 + L_1) \times (2^{\frac{b}{2}}H_2 + L_2) \\ &= 2^b H_1 \cdot H_2 + 2^{\frac{b}{2}}(H_1 \cdot L_2 + H_2 \cdot L_1) + L_1 \cdot L_2\end{aligned}$$

Note that  $2^i$  is a  $i$ -bit left-shift and the partial product  $L_1 \cdot L_2$  being the least significant part can be ignored. This trick can also be used with extended precision numbers to implement FIR filters and store the output as an extended precision number.

### 5.2.2 Truncation, Rounding and Unbiased Rounding

There are three methods of *quantising* an arithmetic operation to smaller word lengths. They are truncation, rounding and unbiased rounding. In this analysis, we assume that the number representations is fractional in value.

**Truncation:** The least significant bits are neglected. This systematically underestimates all values.

**Rounding:** If the least significant bits are less than half full-scale, the output becomes the lowest value. However, if more than half-scale, the output is rounded upwards.

**Unbiased rounding:** When a number falls at the half-way point, round the number upwards or downwards with equal probability.

One mechanism is to round towards an even number and relies on the random distribution of even and odd numbers. This results in a skew in the even-odd distribution, but an averaging of the round-off errors.

Normal biasing can be thought of as rounding towards  $+\infty$ . Incidentally, in floating point (IEEE 754) rounding is unbiased with the options of directed rounding towards  $+\infty$ , downwards towards  $-\infty$  and towards 0. Unbiased rounding has superior performance compared to truncation because the process chooses the nearest number to the unrounded result. This gives a zero mean, symmetrical error distribution. Truncation always throws away information and underestimates the result, consequently the error distribution is skewed negatively and has a non-zero mean.

### 5.2.3 Saturation Arithmetic

Saturation arithmetic is a digital equivalent to analogue based clipping and is advantageous because, if not invoked, word wrap would occur, producing horrible glitches in the sampled data stream. Word wrap occurs naturally in binary arithmetic when overflow occurs. These glitches can cause alias components in the spectrum which would be difficult to remove. This mechanism is useful on outputs of MACs to ensure no wraparound. However if the designer knows the maximum lengths *a priori*, this extra element is not required.

### 5.2.4 Other noise problems

In addition to arithmetic truncation and rounding, errors occur depending on how the input and output is quantised. Another problem is that the error caused by IIR filters is worse than an equivalent FIR filter because of feedback. Also IIR filters have a dead-band in the feedback loop which can cause instability by oscillating at low amplitude. This is known as *limit cycles* and is caused by rounding of near-unity coefficients to unity. However, a sinusoidal oscillator with constant amplitude can be constructed around a fixed-point IIR filter limit cycle.

The process of sampling a continuous time analog signal to its digital form generates *quantization noise* because the signal is rounded or truncated to fit the digital word-length and the error is half the quantization step size. The actual error range for most input signals is assumed to be white noise with zero mean and a variance  $\sigma_e^2$  equal to

$$\sigma_e^2 = \frac{2^{-2b}}{12}$$

where  $b$  is the word-length in bits of a binary  $b$ -bit A/D converter. This variance is used as a benchmark against which all errors due to computation and coefficient are compared. Therefore, as soon as the total output error grows beyond this limit, information is lost because of the build up of noise.

The number of bits allocated to the digital signal relates to the signal-to-noise ratio by approximately  $6b$  dB. Also coefficient quantization is paramount in IIR filters, especially for the poles of the filter because they determine the filter's stability and frequency response.

Arithmetic errors caused by addition and overflow requires careful scaling between accumulation and multiplication steps. An additional error is caused by the filter's own topology. The total noise generated by a filter having  $N + 1$  numerator elements and  $M$  denominator elements and the denominator of the overall transfer function is  $D(e^{j\omega t_s})$  is shown in equation 5.1.

$$\sigma^2 = (N + M + 1) \left( \frac{2^{-2b}}{12} \right) \frac{1}{2\pi} \int_0^{2\pi} \frac{dw}{D^2(e^{j\omega t_s})} \quad (5.1)$$

Using equation 5.1, the designer can estimate the number of bits required to achieve accurate computation of filter and numerical algorithms. Floating-point arithmetic has a large dynamic range so scaling is not necessary. However, normalisation before addition, and multiplication are susceptible to noise accumulation.

### 5.3 Filter based Sine Generator

A sinusoidal generator may be constructed from a discrete-time filter implementation, having a sinusoidal impulse response. The filter topology chosen for this investigation is a simple second order filter with the transfer function shown in equation 5.2.

$$H(z) = \frac{z^2}{z^2 - 2 \cos(\omega_0 T_s) z + 1} \quad (5.2)$$

The poles of the filter are located at  $e^{\pm j\omega_0 T_s}$ , implemented as a complex conjugate pair. Equation 5.2 can be implemented using the canonical direct form two filter structure, having only feedback coefficients. The discrete time recurrence equation for this filter has one multiplier, whose purpose is to set the frequency of the sinusoid, as shown in figure 5.2.

This filter has limitations which are the following. Firstly, the impulse response is delayed by two samples; this can be cured by placing a double zero at zero radius in the z-plane. Secondly, the sinusoidal waveform requires a scaling constant which is pole dependent. By adding a zero at  $z = -1$  and another zero at  $z = 1$  as described in [125], automatically adjusts the scaling constant giving unity amplitude across all frequencies. The increased complexity is in the addition of another addition. Another variation would be to use the second order waveguide oscillator [124]. The last limitation is caused by the poles of the filter which account for noise accumulation as in equation 5.1 and stability. The poles are quantised in either fixed or floating point formats and when plotted as a distribution, the designer can see the spread of frequencies and radii [154]. For the filter design proposed, the frequency resolution caused by the pole coefficient quantization is poor for frequencies close to the origin, but improves as the frequency approaches half the Nyquist rate. Therefore to get adequate resolution requires longer coefficients and different topologies [87]. The filter chosen could have been replaced by the Gold-Rader coupled form [113]. This requires four multiplications in place of two, but

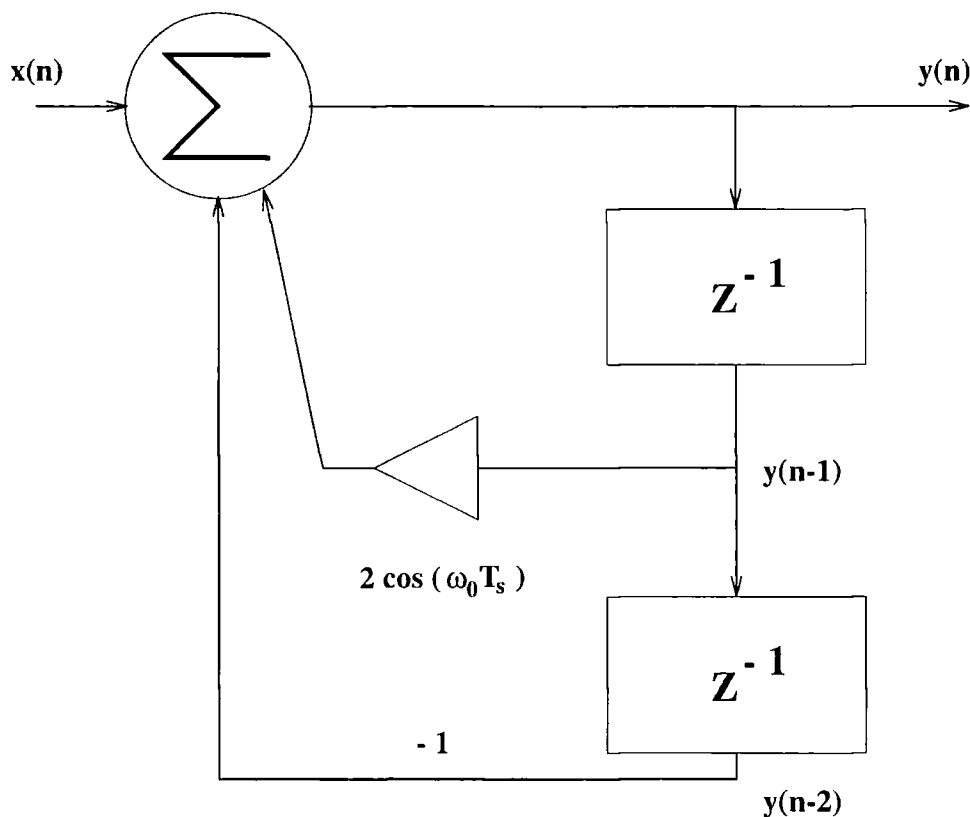


Figure 5.2: Block Diagram of the IIR Sinusoidal Generator

places the complex-conjugate poles on a uniform grid in the  $z$ -plane when quantising coefficients to fixed point arithmetic, thus giving better frequency precision.

### 5.3.1 Filter Structure

The filter structure requires multiplication, addition and delay elements and is implemented in two's complement integer arithmetic. Due to prohibitive cost in silicon area for array multiplication, see section 3.6 on page 44, a bit-serial approach was chosen.

The bit serial approach uses single wires to convey information as opposed to the simultaneous transmission of words on parallel wires (buses). This approach reduces routing requirements in chip design and communication between chips. By using pipelining, as most bit-serial systems naturally aspire to, a regular silicon structure results. Another advantage is that more functionality can be designed onto the chip for less space than an equivalent parallel architecture. The drawback is that the computation proceeds serially with a fixed latency. Most bit-serial systems [20] use least significant bit first transmission, which is unfortunate in rapid

processing as required by Digital Signal Processing. Using most significant bit first techniques [26, 27] coupled with redundant number systems allows computational speed to increase [150].

### 5.3.2 Multiplier Design

The multiplier can be implemented using  $N$  modules cascaded together to generate the partial products. However a subtraction stage and a complementer are required in addition to  $N - 1$  two's complement modules [79]. It is possible to reduce the latency of the pipeline by using recoding of the multiplicand as in Booth's recoding on page 49. The latency reduction arises from the reduction of addition operations in the partial product formulation.

A two's complement number represented as

$$y_k y_{k-1} \cdots y_2 y_1$$

has a value equal to equation 5.3, where  $n = k$  results in  $Y$  being in the range  $-1 \leq y \leq 1$  or if  $n = k - 1$ , then  $Y$  is bound by  $-2 \leq y \leq 2$ .

$$Y = \sum_{i=1}^{k-1} y_i 2^{i-n} - y_k 2^{k-n} \quad (5.3)$$

Multiplication would be  $P = X \cdot Y$ , where  $X$  and  $Y$  are the same length and range and are represented by equation 5.3. The result would be equal to equation 5.4.

$$\begin{aligned} P &= (-y_k 2^{k-n} + \sum_{i=1}^{k-1} y_i 2^{i-n}) \cdot (-x_k 2^{k-n} + \sum_{j=1}^{k-1} x_j 2^{j-n}) \\ &= y_k x_k 2^{2k-2n} + \sum_{i=1}^{k-1} \sum_{j=1}^{k-1} y_i x_j 2^{i+j-2n} \\ &\quad - \sum_{i=1}^{k-1} x_k y_i 2^{i+k-2n} - \sum_{j=1}^{k-1} y_k x_j 2^{j+k-2n} \end{aligned} \quad (5.4)$$

The partial products in equation 5.4 can also be expressed as in equation 3.6 on page 44, where each column represents multiplication by two. Thus for two  $N$ -bit number multiplications, there are  $N$  columns by  $N$  rows representing the  $2N$ -bit product. If this was implemented, the output would be delayed by  $N$  clock cycles. Using a radix-2 Booth recoder, see section 3.7.4 creates  $N$  regular elements unlike the non-booth recoded bit-serial approach. A radix-2, signed-digit, 5-level Booth encoder recodes triplets of the multiplicand in an overlapped manner, representing

signed digit numbers of -2, -1, 0, 1 and 2. However, the multiplicand has an even number of bits, so the first triplet has an extra bit inserted  $y_0$  having a value equal to zero. Thus the multiplicand is recoded using the following expression :-

$$Y = \sum_{i=0}^{k/2-1} (y_{2i} + y_{2i+1} - 2 y_{2i+2}) 2^{2i-k+2}$$

Now, the number of partial products decreases by a factor of two, because each module adds or subtracts by 0, X or 2X. As with the 3-level Booth recoder approach, each module is regular and supports two's complement arithmetic with no extra logic. It is possible to extend the recoding approach to higher levels, but then the hardware must be able to generate  $\pm 3X$ . However in [110] this problem is solved for a nonredundant radix-4 case. The bit-serial approach computes the multiplication of  $P$  by operating on each bit serially within the module. So,

$$P = XY = \sum_{i=0}^{k/2-1} (y_{2i} + y_{2i+1} - 2 y_{2i+2}) X 2^{2i-k+2}$$

This can be shown to equal equation 5.4 by performing multiplication as in equation 3.6 and as remembered in school mathematics classes. A simplified schematic diagram of the 5 level Booth multiplier module is shown in figure 5.3.

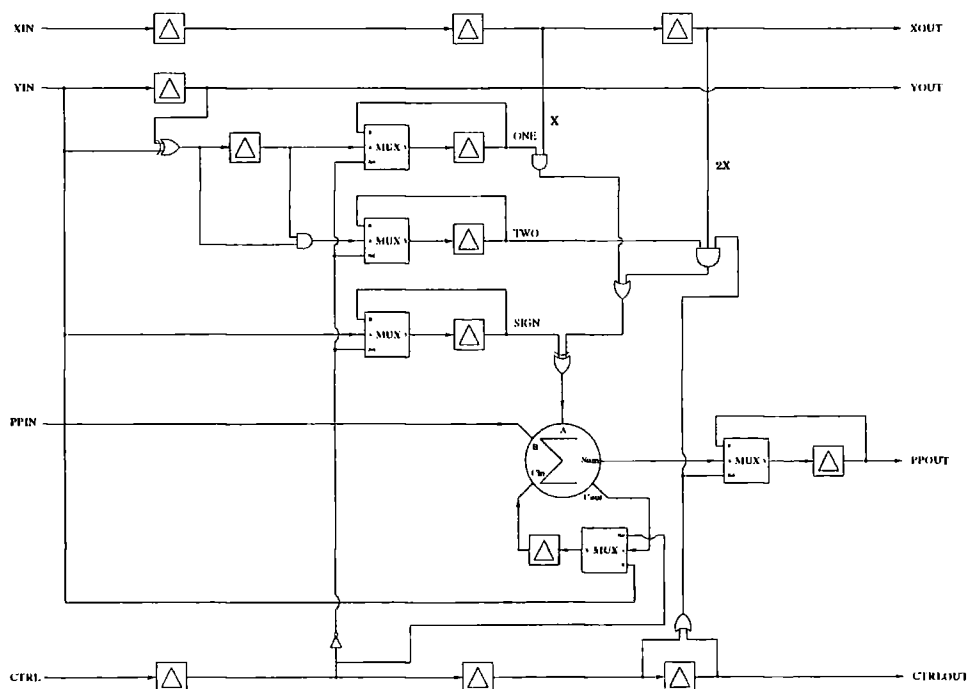


Figure 5.3: Schematic Diagram of the 5 level Booth Multiplier Module



To implement multiplication in a serial manner, the first module has  $y_0$  hardwired to equal zero. This will change both ONE and TWO outputs of the recoder as shown in the diagram 5.3. The inputs to these gated recoded signals will need minor modification. If the designer connected a cascade of these modules together, this design would implement multiplication with truncation at the final PPOUT output pin. The structure can be made to implement rounding to the nearest integer by inserting a one signal on the PPIN input of the first module at the correct time, otherwise PPIN should be set to zero [20]. The least significant N-bits of the product appear at the output of the adders every two clock ticks and can be retrieved using an additional multiplexor being fed from the serial adder.

### 5.3.3 Accumulator Design

The adder is simply a two's complement, least significant bit first, bit-serial adder module connected to the output of the multiplier.

### 5.3.4 Filter Design

The multiplier coefficients have a word-length of 16 bits, whilst the data path is 24 bits wide. The number of elements for the Booth multiplier would then be eight modules wide. Each multiplier has a latency of  $3N/2$  bit periods to compute the result. In this design, the delay elements are assumed to be contained within the Booth multiplier core and thus no external storage is necessary. As each multiplier in this design corresponds to the delay element, the  $-1$  factor in figure 5.2 becomes an extra multiplier. The Booth multiplier can be configured for coefficients to be updated synchronously with the data input, or they can be latched and made constant. The latter form will be used in this design. All numbers are assumed to be two's complement in the range of  $+2$  to  $-2$ . The  $y(n)$  input to the first delay element is truncated to 16 bits with sign extension for the multipliers. The least significant 16 bits of the output is sent serially to a bit-serial DAC. The  $x(n)$  input is a positive impulse of unity triggered by a bit line going high for almost one clock word ( 22 bits ) and whilst high outputs ones into the bit stream otherwise outputs zeros.

Assuming the core is predominantly built of adders and multipliers gives a total floor space used by the core of 8.25 mm x 2.2 mm using 8906 transistors. The standard cells used in this estimation were from the ES2 design kit for Cadence [28].

---

## 5.4 The CORDIC Sine Generator

The COordinate Rotation DIgital Computer (CORDIC) is a unified algorithm for computing trigonometric, square roots and multiply-accumulate functions. This algorithm was developed by Jack E. Volder in 1959 [140]. Further refinements to the basic algorithm were researched by [40, 141], allowing computation of hyperbolic functions and obtaining analytical rules to determine algorithm convergence. This is necessary for computation of periodic functions like the trigonometric functions.

The CORDIC system has two modes, known as **Rotation** and **Vectoring**. In the rotation mode (circular), the user inputs the Cartesian coordinate components plus an angle of rotation, and the system rotates the Cartesian coordinates through the angle given. In the vectoring mode (circular), the coordinate components are given and the corresponding magnitude and angle are computed. This structure is geometric by nature and performs rectangular  $\Leftrightarrow$  polar conversions. The algorithm achieves the conversion by using iteration in a divide-and-conquer strategy.

### 5.4.1 Derivation for Circular Mode

The rotation matrix to transform a point in x-y Cartesian coordinates through an angle  $\theta$  in an anti-clockwise direction is described by equation 5.5.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (5.5)$$

Now, rectangular to polar conversion can be performed by the inverse, which is expressed by equation 5.6 and 5.7.

$$\theta = \tan^{-1} \left( \frac{y}{x} \right) \quad (5.6)$$

$$r = \sqrt{x^2 + y^2} \quad (5.7)$$

Now substitute  $\theta = \tan^{-1} r^{-k}$  and using the following trigonometric identities

$$\sin \theta = \frac{\tan \theta}{\sqrt{1 + \tan^2 \theta}}$$

and

$$\cos \theta = \frac{1}{\sqrt{1 + \tan^2 \theta}}$$

Consequently we obtain

$$\sin[\tan^{-1} r^{-k}] = \frac{r^{-k}}{\sqrt{1 + (r^{-k})^2}}$$

and

$$\cos[\tan^{-1} r^{-k}] = \frac{1}{\sqrt{1 + (r^{-k})^2}}$$

and substituting into the equations 5.5 for transforming the vector (x,y), it can be seen that multiplication is a simple radix shift right. It should be noted that rotating an angle through  $-\tan^{-1} r^{-k}$  is achieved by changing the sign of the  $r^{-k}$  term.

An arbitrary angle  $\theta$  can be built up as follows, noting that  $\epsilon_k = +1$  or  $-1$ . This parameter can be shown to be related to the sign of  $y$  in the rotation mode and related to the sign of  $\theta$  in the vectoring mode.

$$\theta = \epsilon_{-1} 90^\circ + \sum_{k=0}^{\infty} \epsilon_k \tan^{-1} r^{-k} = \sum_{k=-1}^{\infty} \epsilon_k \theta_k \quad (5.8)$$

The magnification after  $n$  iterations is equal to

$$\prod_{k=1}^n \sqrt{1 + r^{-2k}} \quad (5.9)$$

Unfortunately, equation 5.9 results in a factor which must be normalised. Since this is a constant, the reciprocal can be input to the  $x$  or  $y$  input on initialisation. This factor can also be forced to unity by the addition of another layer of arithmetic operations.

#### 5.4.2 The CORDIC Algorithm

The generalised CORDIC algorithm can be expressed by the matrix equation 5.10.

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \\ Z_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & m \delta_i r^{-i} & 0 \\ -\delta_i r^{-i} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \delta_i \alpha_i \end{bmatrix} \quad (5.10)$$

$\alpha_i$  is equal to  $\lim_{m \rightarrow \pm 1 \text{ or } 0} \frac{1}{\sqrt{m}} \arctan(\sqrt{m} r^{-i})$  and  $i \geq 0$ . The variable  $\delta_i$  is equal to  $-\text{sign}(Z_{i+1})$  for rotating mode and  $\text{sign}(X_{i+1}) \cdot \text{sign}(Y_{i+1})$  for vectoring mode<sup>1</sup>. Where  $\text{sign}(x)$  is equal to 1 when  $x \geq 0$  and  $-1$  otherwise. The parameter  $m$  determines the coordinate system to be used and is equal to  $-1$  for hyperbolic, 0 for linear and 1 for circular coordinate systems. Incidentally, in linear mode  $\alpha_i$  becomes just  $r^{-i}$  and can be implemented using a shifter. In this mode there is no magnification and the system will thus perform accurate multiply and accumulate or division and accumulate functions to machine precision. As the algorithm stands, in

Rotation Mode	Vectoring Mode
<b><math>m = +1</math> Circular</b>	
$x \rightarrow K_1(x \cos z - y \sin z)$ $y \rightarrow K_1(y \cos z + x \sin z)$ $z \rightarrow 0$	$x \rightarrow K_1 \sqrt{x^2 + y^2}$ $y \rightarrow 0$ $z \rightarrow z + \tan^{-1} \left( \frac{y}{x} \right)$
<b><math>m = 0</math> Linear</b>	
$x \rightarrow x$ $y \rightarrow y + x \cdot z$ $z \rightarrow 0$	$x \rightarrow x$ $y \rightarrow 0$ $z \rightarrow z + \left( \frac{y}{x} \right)$
<b><math>m = -1</math> Hyperbolic</b>	
$x \rightarrow K_{-1}(x \cosh z + y \sinh z)$ $y \rightarrow K_{-1}(y \cosh z + x \sinh z)$ $z \rightarrow 0$	$x \rightarrow K_{-1} \sqrt{x^2 - y^2}$ $y \rightarrow 0$ $z \rightarrow z + \tanh^{-1} \left( \frac{y}{x} \right)$

Table 5.1: Input-Output Functions for CORDIC Modes

hyperbolic mode convergence fails and is solved by repeating iterations as necessary, see [141] for more details.

Table 5.1 gives a summary of the operations available using the CORDIC iteration algorithm. The rotation mode requires inputs  $x$ ,  $y$  and the angle  $\theta$  ( $z$ ) and the vectoring mode requires the inputs  $x$  and  $y$  with the  $\theta$  parameter set to zero. The magnification factor  $k$  with subscript 1 or  $-1$  equals  $\sum_{i=0}^{n-1} \sqrt{1 + m r^{-i}}$ .

Thus using the rotation mode of CORDIC with the circular coordinate system, a sinusoidal oscillator can be devised. However, the beauty of this technique is that it can generate complex transcendental functions from nothing more than addition and arithmetic shifts. It has also become a very useful building block in general DSP and lattice filtering [49]. There are other techniques available for generating transcendental functions like BKS [5] but they are not as efficient as CORDIC in terms of memory storage and computational effort, except possibly in a redundant number system.

The algorithm can only resolve the smallest angle  $\alpha_{n-1}$  of zero in  $n$  steps and requires an internal word-length of  $L + \log_r L$  bits for the desired  $L$  bits output. The extra  $\log_r L$  bits are required because the arithmetic operations truncate in the

<sup>1</sup>In circular vectoring mode  $\delta_i = \text{sign}(Y_{i+1})$  only.

conventional weighted number system. An analytical investigation into CORDIC convergence limits can be found in [50]. By using redundant arithmetic such as signed-digit, the scaling factor can be made constant [131] and operations can be implemented in a bit-serial manner. As mentioned before, the scale factor is removed by incorporating extra shifts and addition in between some parts of the usual CORDIC algorithm.

Finally, before the actual design of the sine generator, it is necessary to understand the limits of convergence for CORDIC [48]. In linear mode convergence is bounded by unity, in hyperbolic mode the angle is bounded to 1.1182 radians, and in circular mode it is bounded to 1.7433 ( $99^\circ$ ). As this section is concerned with sine and cosine generation, I will only mention how to increase the range for this particular mode of operation. If the angle ( $z$ ) is driven to zero, as in the rotation mode, then the binary digits can represent  $\pm 180^\circ$  and the designer can utilise the wrap-around property of two's complement arithmetic and number representation as described by Daggett [18]. The alternative route is to add a range reduction process to the algorithm as an initialisation pass to map any angle greater than  $99^\circ$  into that range [40]. This approach is most amenable to the CORDIC algorithm requiring a shift of  $90^\circ$  and then a shift of  $45^\circ$ . It should be noted that the derivation of the circular mode using the vector rotation matrix equation 5.5 has these modifications built in.

### 5.4.3 CORDIC Sine Algorithm

Using the CORDIC rotation algorithm in circular mode, sine and cosine functions can be generated in quadrature. The CORDIC structure allows the trigonometric functions to be generated without multiplication by utilising addition and shifting primitives.

This algorithm requires a clock input. The shifter will be implemented using a barrel shifter to allow division by powers of two. This algorithm requires a ROM to store in binary code the  $\arctan(2^{-i})$  where  $i$  is an integer from -1 to 16. To generate  $\sin(\frac{2\pi f}{f_s})$  requires 18 iterations and an internal clock of  $18 * 44.1\text{KHz} = 775.2\text{ KHz}$ . The 44.1KHz is the sampling rate for audio applications. This assumes that the  $\sin(x)$  at a certain period of time has a word-length of 16 bits. Consequently the CORDIC  $X$  and  $Y$  variables have an internal resolution of 20 bits ( $n + \log_2(n)$ , where  $n$  is the output accuracy in bits. ) The clock speed is quite slow, but the CORDIC algorithm can be made to compute thousands of sinusoid oscillators by increasing the clock and by pipelining the circuit. The iteration algorithm is illustrated in matrix form in equation 5.11 noting that  $\delta_i = -\text{sign}(Z_{i+1})$  and the

iteration counter,  $i$  is bounded in the range of  $-1 \leq i \leq 19$  with  $\lambda$  equal to zero when  $i$  is equal to  $-1$  and  $\lambda$  is equal to one <sup>2</sup> otherwise. One final point is the shifter  $2^{-i}$  is set to one when  $i = -1$ .

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \\ Z_{i+1} \end{bmatrix} = \begin{bmatrix} \lambda & \delta_i 2^{-i} & 0 \\ -\delta_i 2^{-i} & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \delta_i \arctan(2^{-i}) \end{bmatrix} \quad (5.11)$$

To generate the sine of an angle, the algorithm requires  $Z_i$  to be loaded with the two's complement value of the angle at iteration time  $i = -1$ . After 21 cycles, the result appears on  $X_{i+1}$  and  $Y_{i+1}$  ( $i = 19$ ). In this application, the cosine output is more valuable, since at power up this output will be zero and the audio stream will be glitch-less. Remembering the scale factor equation 5.9 with  $k$  starting from zero results in the cosine to be bounded by a number bigger than unity. To rescale the output to be bounded by unity requires the input  $X_{-1}$  to be assigned the reciprocal of the magnification factor, which is 0.6072529351 instead of unity. The other variable  $Y_{-1}$  is set to zero.

So far, the design has concentrated around the CORDIC algorithm. This part of the system would only generate a single sine of an angle. Using a phase accumulator in two's complement arithmetic configured as a ramp generator, as in table lookup oscillators, allows the CORDIC algorithm to implement a sinusoidal function with respect to time. The phase accumulator is simply an adder and a feedback storage element which generates a ramp function of the form shown in equation 5.12. Note that  $A_x$  is the output word at time slot  $x$ ,  $\delta$  is the increment and  $n$  is the word-length of the adder.

$$A_{n+1} = (A_n + \delta) \bmod 2^n \quad (5.12)$$

#### 5.4.4 Design of the CORDIC sine Generator

Using equations 2.1 and 2.2 with  $F_{\max}$  to be 20 kHz and  $F_{\min}$  to be 0.002 Hz as a reasonable estimate, gives a phase accumulator word-length of 24 bits. But only the top 20 bits are used by the CORDIC algorithm. The phase accumulator internal word-length was designed to have 20 bits, to directly interface to the CORDIC iteration algorithm, which has an internal 20 bit datapath for all variables,  $X$ ,  $Y$  and  $Z$ . The output from the cosine path of the CORDIC processor is truncated to the most significant 16 bits.

---

<sup>2</sup>Actually the reciprocal of the magnification factor.

The chip packaging has a direct relationship to cost, because the more pins required imply a larger silicon die size. In this design, a 40 pin DIL package was chosen. There are 16 pins for the output cosine, two diagnostic pins which are outputs from internal iteration signalling, and a pin indicating the sign of  $Z_{i+1}$ . Apart from power pins and clock the phase accumulator is loaded in blocks of 8 bits using a load and two chip select pins. Finally, there is a reset to zero pin to initialise the chip. The input pins drive TTL non-inverting with internal pull-up resistors of 125 k $\Omega$  and the output pins are driven by CMOS non-inverting 4 mA source/sink pads.

A schematic diagram of the CORDIC iteration algorithm can be seen in figure 5.4. The  $X$  input is either zero or the reciprocal of the scaling factor under the control of  $\lambda$ . The selecting of zero or non-zero constant can be achieved by using an *AND* gate with  $\lambda$  as one input and the constant as the other. Using DeMorgan's rule,  $A \cdot B = \overline{\overline{A} + \overline{B}}$  and  $\overline{A}$  can come from the *NOT* output of a D Flip-Flops and the signal  $B$  ( $\lambda$ ) can be inverted and stored in a ROM. This reduces the transistor count per bit from 6 to 4. A ROM table is required to store the angles used by the iteration, being of value  $90^\circ$ ,  $45^\circ$  and  $\arctan(2^{-i})$  for  $i > 0$  with 20 bits of precision and coded in Q19 format.

---

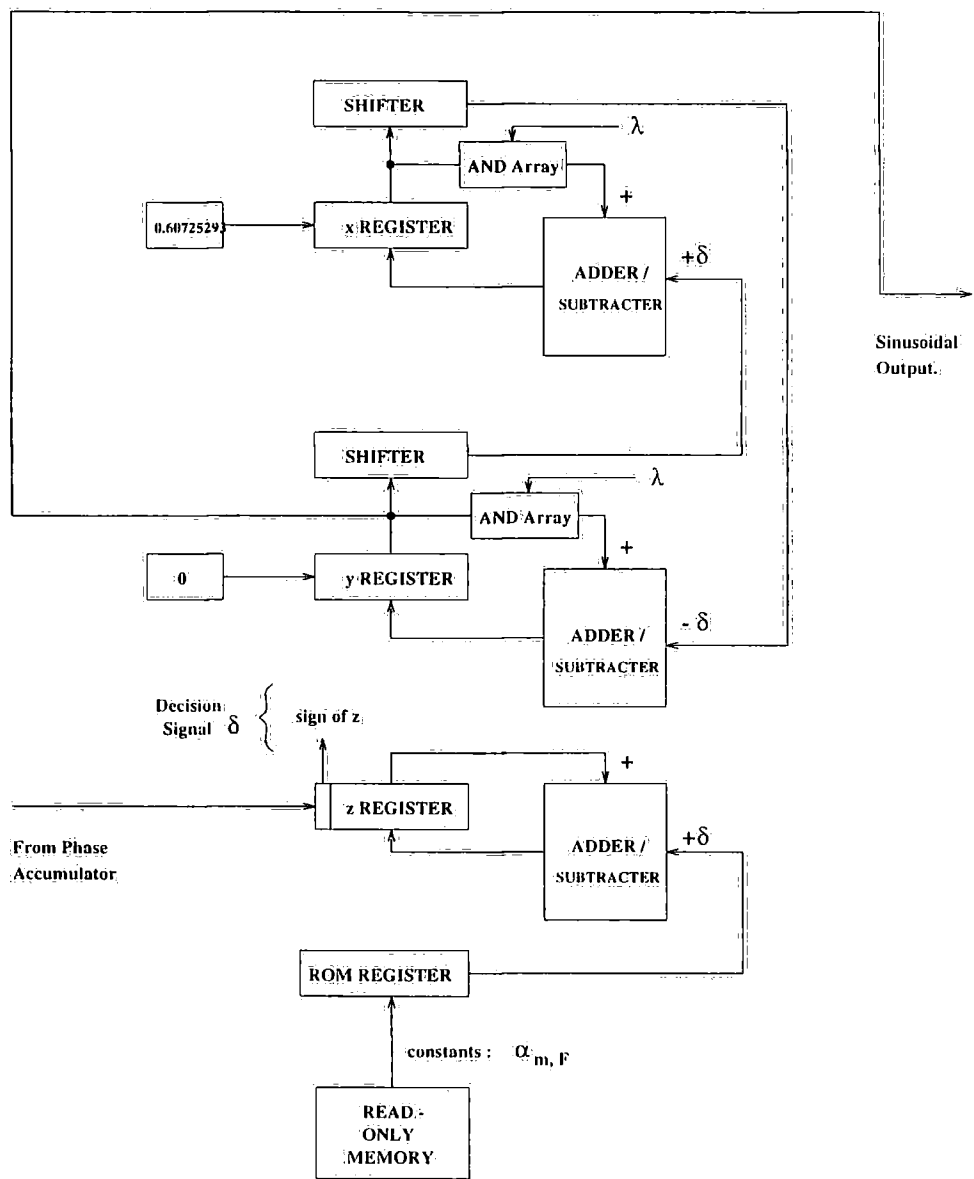


Figure 5.4: Block Diagram of Sinusoidal Generator



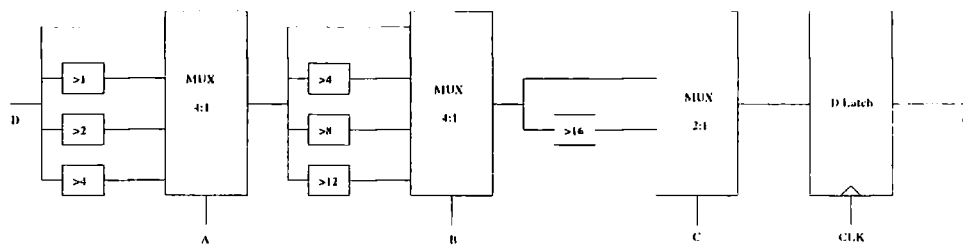


Figure 5.5: Block Diagram of the Barrel Shifter

The implementation of the shifter,  $2^{-i}$  would normally be implemented using an  $n \times n$  crosspoint switch with each switch (transistor) connected diagonally to carry out the shift. An example CORDIC implementation using this with redundancy caused by the arithmetic right shifting can be seen in [40] and is labelled a scaler. My approach is to split the shift up into smaller shifts by using a series of multiplexors connected in a tree topology as shown in diagram 5.5. This is known as a Barrel Shifter and is ideal for use in automated datapath generation. The inputs A, B and C are 2, 2 and 1 bit lines long and the signal for a particular shift is best served by storing the values in a ROM. Now all that is required is a sequencer which drives the ROM inputs according to  $i$ . This is implemented as a modulo counter.

The input interface is a simple 2-to-4 de-multiplexor with load signal driving four 8-bit registers. The load signal is used as a clock for the registers. The last bit sequence to the multiplexor (all ones) updates the input register in the phase accumulator, so changes in setting the input angle word do not effect the sinusoidal output. The output of the chip ready for DAC input is composed of edge-triggering D-flip-flops with asynchronous reset to zero.

#### 5.4.5 Sequencer Design

Initially a mod 19 counter was designed as the sequencer unit. This required 5 D flip-flops and 10 4x4 Karnaugh maps and a state-machine design was attempted. Unfortunately, the state equations gave 18 NAND gates ranging from 2 to 4 inputs. This was not very efficient for VLSI design, but by using toggle flip-flops the number of external gates dropped. The T flip-flop is based around a D flip-flop and has an enable (EN) and synchronous reset (SYNCR) inputs as in figure 5.6. The synchronous reset allows the flip-flop to reset only at the clock edge and is active low (i.e. zero resets flip-flop). The circuit can perform a toggle operation every clock cycle by setting enable to route  $\overline{Q}$  to the D input of the flip-flop. This performs a divide by two counter.

To count modulo 64, the least significant flip-flop has EN tied to unity and all the

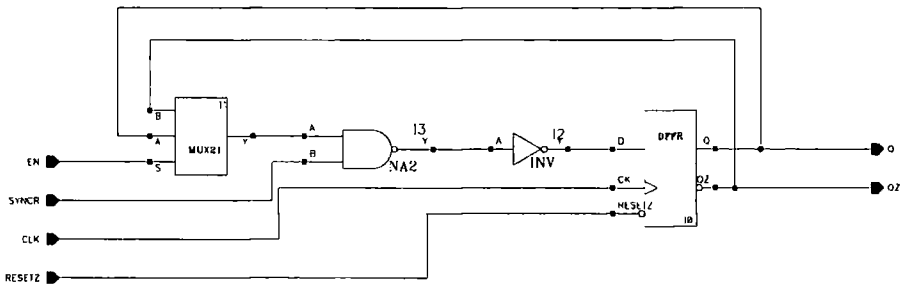


Figure 5.6: Schematic Diagram of a T flip-flop

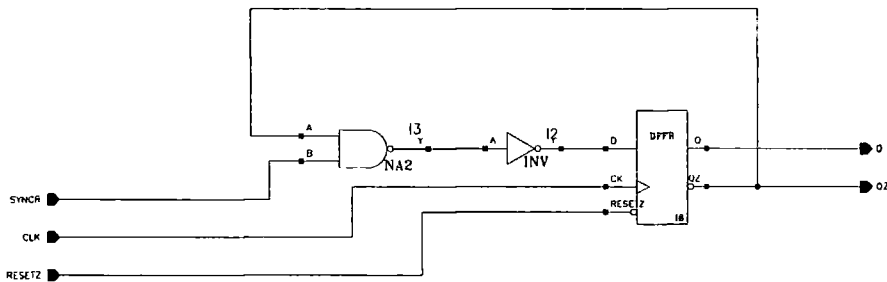


Figure 5.7: Schematic Diagram of first T flip-flop for mod counting

remaining four flip-flops have two input AND gates with enable and Q connections for each flip-flop daisy chained to the enable of the next flip-flop along. The most significant flip-flop has no AND gate across it. Thus a modulo ripple counter has been made. The first flip-flop always has enable set to one and can be implemented as shown in figure 5.7.

To convert this counter to a modulo 19 counter, the designer requires the monitoring of state 18 which in binary is 10010. This is used to reset the counter for the next cycle. Since the counter always starts from zero, it never counts beyond 18 so the sequence becomes 1XX1X with X equal to don't care states. Therefore the global synchronous reset input is driven by a NAND gate connected to the  $Q_2$  and  $Q_5$  outputs of the flip-flops. It is possible to remove the synchronous resets on the T flip-flops three and four as this line is not necessary. However, flip-flop one will

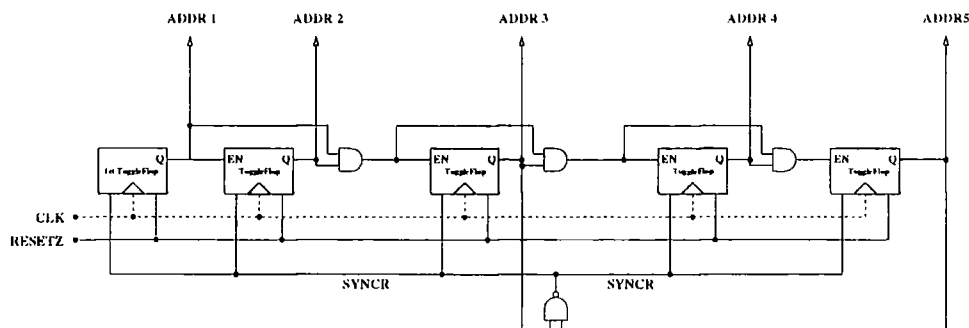


Figure 5.8: Schematic Diagram of a modulo 21 counter

change state. This extension was not implemented as it had no major advantage in VLSI. The circuit will work but the checks for 1, 11, 111 and 1111 implemented with the AND gates are the limiting factor because flip-flop five has 3 ganged gates resulting in a ripple delay. If the clock was sufficiently fast, a false count would be possible. This can be overcome by using two, three and four input AND gates.

The actual counting sequence is modulo 21, thus the counter must be reset after the next cycle at count 20 or 10100 in binary. Therefore the NAND gate inputs required are connected to  $Q_2$  and  $Q_4$  outputs as in figure 5.8.

#### 5.4.6 ROM Design

The ROM array was initially designed to store the binary representation of the angles used by the CORDIC algorithm. However, since the shifter uses 5 lines and there are two extra control signals, a ROM also became the most convenient way to sequence the control aspects of the design. This is achieved by controlling the ROM address using the modulo 21 counter. The angles are generated by the following C++ source fragment on page 101. The output of the C++ code is a table of count (ROM address) and ROM contents in hexadecimal. These elements are recoded into two columns, the address in decimal and the contents in binary, and are passed to the ES2 Megacell generator which is part of the ES2 design kit for Cadence. The complete ROM contents in human readable form can be found on page 102. The signals SSL, NSTART ( $\bar{\lambda}$ ) and INIT are internal control signals and `select<0:4>` controls the barrel shifter.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define PI      3.141592654
#define MASK    0xffff
#define SHIFT   12
#define MAX     21

void main(void)
{
    double temp;          // workspace variable
    int count;            // counter cycling from 0 to 21
    long table[MAX];      // storage for arctan table

    temp = pow(2.0,19.0) - 1.0; // Uses Q19 format
    temp = temp / PI;          // modifies Q19 format for angles
                                // within 2 PI

    table[0] = (long)floor(PI/2*temp); // 90 degrees

    for(count=1; count <MAX; count ++){
        table[count] = (long)floor(atan(pow(2.0,-(double)(count-1)))
                                * temp);
    }

    fprintf(stdout,"\n\tROM Address\tROM value\n\n");
    for(count=0;count<MAX;count++){
        fprintf(stdout,"\t%d\t\t%08x\n",count,table[count] >> SHIFT);
    }
}
```



Index	SSL	INIT	select<0:4> & NSTART	romtable<19:0>	rom (hex)	shift
0	1	1	- 00000	- 00000000000000000000	00000	I
1	0	1	- 00000	- 01000000000000000000	04000	0
2	0	0	- 00000	- 00011111111111111111	01fff	0
3	0	0	- 10000	- 00010010111001000000	12e40	1
4	0	0	- 01000	- 00001001111110110011	09fb3	2
5	0	0	- 11000	- 00000101000100010001	05111	3
6	0	0	- 00100	- 00000010100010110000	028b0	4
7	0	0	- 10100	- 00000001010001011101	0145d	5
8	0	0	- 01100	- 00000000101000101111	00a2f	6
9	0	0	- 11100	- 00000000010100010111	00517	7
10	0	0	- 00010	- 00000000001010001011	0028b	8
11	0	0	- 10010	- 00000000000101000101	00145	9
12	0	0	- 01010	- 00000000000010100010	000a2	10
13	0	0	- 11010	- 00000000000001010001	00051	11
14	0	0	- 00110	- 00000000000000101000	00028	12
15	0	0	- 10110	- 00000000000000010100	00014	13
16	0	0	- 01110	- 00000000000000001010	0000a	14
17	0	0	- 11110	- 00000000000000000101	00005	15
18	0	0	- 00111	- 00000000000000000010	00002	16
19	0	0	- 11111	- 00000000000000000000	00000	17
20	0	0	- 11111	- 00000000000000000000	00000	18

The generator created a ROM with 21 words, having 27 bits per word configured in 108 columns by 8 rows. The output data from the ROM is buffered. Using 0.7 $\mu$ m ES2 design rules, the dimensions of the ROM are 902.50 x 374.40 ( $\mu$ m x  $\mu$ m) with a total area of 0.34 square millimetres. Unfortunately, the ROM data address space is rounded up to the nearest power of two.

5.4.7 CORDIC register Design

The registers, *X* and *Y* are implemented using synchronous flip flops because they are necessary for initialisation of the CORDIC iteration. The *Y* register is composed of synchronous reset flip-flops whilst the *X* register has both reset and set flip-flops

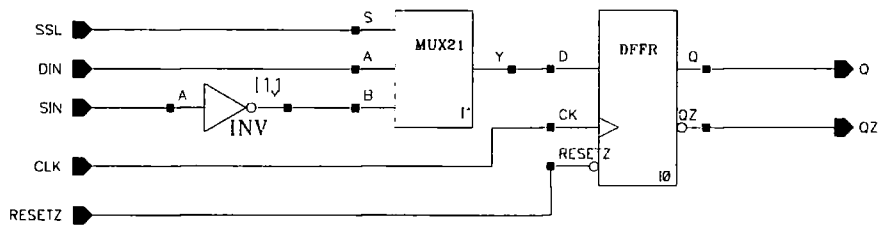


Figure 5.9: Schematic Diagram of a Synchronous Reset Flip-Flop

which describe the binary constant for normalisation. A diagram of the synchronous reset flip flop is shown in figure 5.9 and to implement the synchronous reset, pins SIN and SSL are tied together. To implement a synchronous set latch, the inverter is removed.

The Z register is implemented using a normal latch/flip-flop but has a multiplexor on its input to access the angle generated from the ramp generator and the output of a 20-bit adder/subtractor.

5.4.8 Adder Design

The adder/subtractor is simply a two's complement full adder as shown schematically in figure 3.1 on page 41.

5.4.9 Construction and Testing

The chip was designed using the 0.7 $\mu$ m ES2 two metal layer CMOS design rules for Cadence and implemented in a 40 pin DIL package. The chip die size was 3.5 mm x 3.5 mm with a core size of 2.5 mm x 2.5mm. The entire design was entered using hierarchical schematic data entry. The initial design was to have used ES2's datapath compiler, but unfortunately it would not work with the macro generator and had to be ignored. Schematic entry proved to be troublesome, due to the placement of the datapath bus. An example of this occurred with the barrel shifter design. This was detected by comparison with a C source program and a simulation of the shifter. The comparison indicated incorrect behaviour for shifting greater than three bits. I looked closely at the schematic to make sure all the wires were

labelled properly, and found some missed interconnections. A Verilog or VHDL high level description language should have been used to save valuable design time.

The chip pad layout was drawn using the notes from the guide [28] for simulation and routing.

Test vectors for a C++ simulation and design were developed. I compared the C++ simulation with the Verilog simulator contained within Cadence. I implemented the excitation file for the design using Cadence's propriety language called STL ( Simulation and Test Language ) and the output was displayed in both tabular and graphical form.

To test the chip, I used test vectors based on a phase increment value of  $30^\circ$ . Over time, the phase accumulator added this increment to its output. This created a periodic ramp function which drives the CORDIC angle input. The phase increment was coded as the 24-bit number 0x155550 in hexadecimal. This number was applied to the chip in three 8-bit portions and then a control command was issued to pass the full 24 bits to the phase accumulator. The comparison between the expected value of the sine of the angle and the corresponding output of the chip simulation can be shown clearly in figure 5.10. The sine function has been interpolated as a smooth line between all x ordinates for visual clarity. The chip simulation appears at integral sample values which are normalised to the output sample timestep. This rate is 20 times slower than the input clock rate used to drive the whole chip.

---

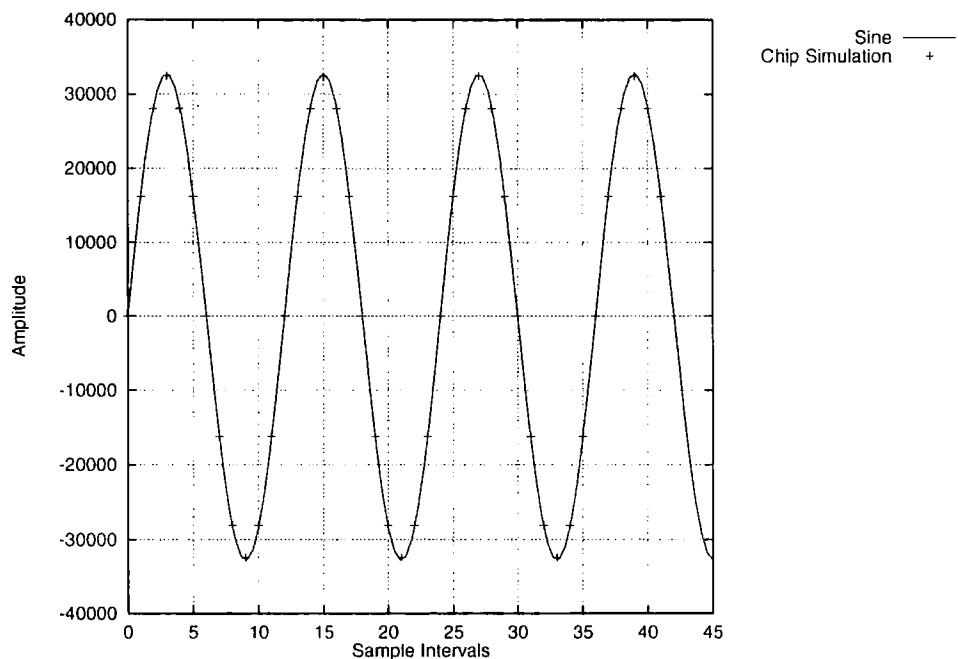


Figure 5.10: Comparison between Sine Function and CORDIC Simulation

The error difference between the sine function and the CORDIC simulation appears as the least significant bit of a 16 bit two's complement number. This error varies with the phase value and can be seen in figure 5.11. The error points appear as dots in the mid-points of the boxes, which are plotted here for clarity. The error difference was as expected, because the CORDIC algorithm is iterative in nature. A constant error across all inputs would not occur due to two's complement rounding errors. The error is small enough to be neglected in this application as the noise floor of the audio output circuitry would be much larger.



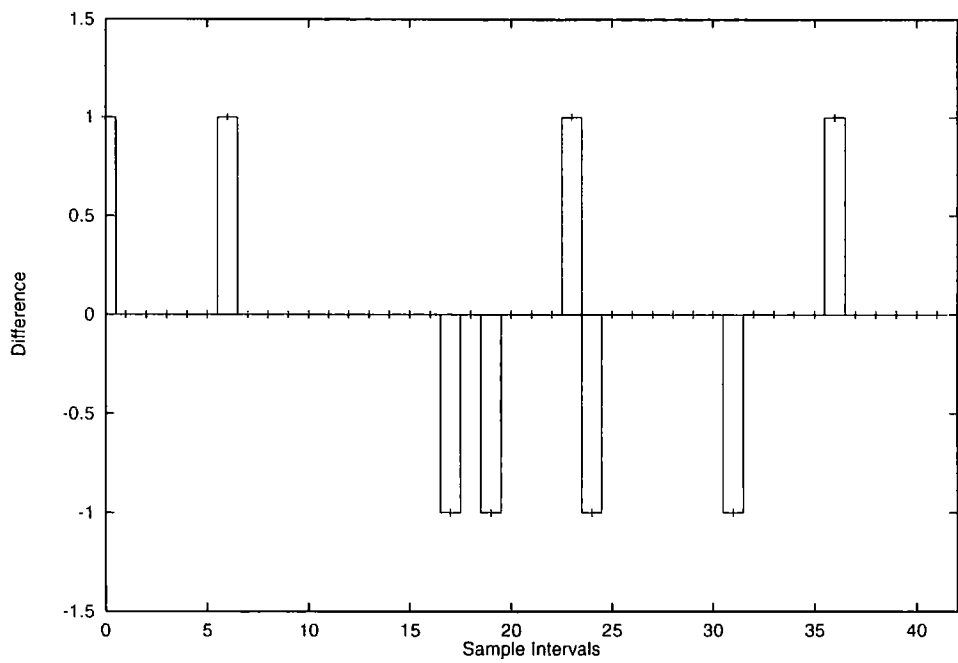


Figure 5.11: Error Difference between Sine Function and CORDIC Simulation

The chip was then routed via the floorplanning environment and initially failed for power line routing. This was overcome by manual/interactive routing [28, 149]. After final testing the design was sent electronically to Rutherford Appleton Laboratories in Oxfordshire to be checked. The design was then sent to IMEC in Belgium for manufacture under the EUROCHIP project. The chip contained approximately 7000 transistors. It was then tested using Durham University’s chip testing rig using the HP 8180A data generator and HP 8182A data analyser. The chip has the following inputs and outputs as shown in table 5.2.

Name	Function	I/O
CLK	Clock	input
INP<7:0>	Angle Increment	input
RAMS<1:0>	Mux Select	input
LOAD	Angle Transfer	input
RESETZ	Initilisation	input
DC<15:0>	Sine Data	output
INIT	Latch Load	output
NSTART	ROM NSTART Signal	output
SIGN	CORDIC Sign Signal	output

Table 5.2: CORDIC Sine Chip Input-Output Connections

Frequency (Hz)	Current Consumption (mA)
Quiescent	0.4
2	0.7
20000000	22
50000000	54

Table 5.3: Current Consumption of CORDIC Sine Chip

The RESETZ line of the chip was driven by the 8180A and was set as a return to zero signal. This caused the chip always to reinitialise. Thus non return-to-zero coding was used. The output levels of the generator was set to 0V and 5V (true TTL levels). The first test was to check total current consumption of the chip at various frequencies by grounding the inputs and setting the 8180A data generator into cycle mode. Table 5.3 shows the current consumption of the chip for various clock rates; the current figure was obtained from the analogue meter on the test rig. This table clearly shows the linear relationship between clock frequency and current consumption.

I then set the 8182A data analyser to be clocked via an external input, the INIT pin of the chip. Thus the analyser only saved the output signals when there was a transition on INIT. I transferred the Verilog simulation test vectors to the 8180A data generator and configured the outputs (inputs to the chip). Initially the output cycled as expected from Verilog simulation. However, looking at the analyser log of activity, I saw that the cycle locked up into a wrong value. To investigate what was happening, I connected an Oscilloscope to an output pin and noticed an asymmetric waveform which was confusing the scope's trigger. The oscilloscope confirmed that the output pads of the chip were capable of driving up to 10 MHz. Beyond this frequency the voltage drop was too severe for CMOS or TTL to operate.

The device was found to work intermittently. Initially I thought this problem was internal to the chip. However the chip worked when one of the Oscilloscope's leads was connected to ground and the other to one of the input pins on the test rig. After further exploration between the oscilloscope and test rig, I found a poor ground connection.

A scanned photomicrograph of the completed die can be seen in figure 5.12: the block in the top left hand corner represents the Megacell ROM core.

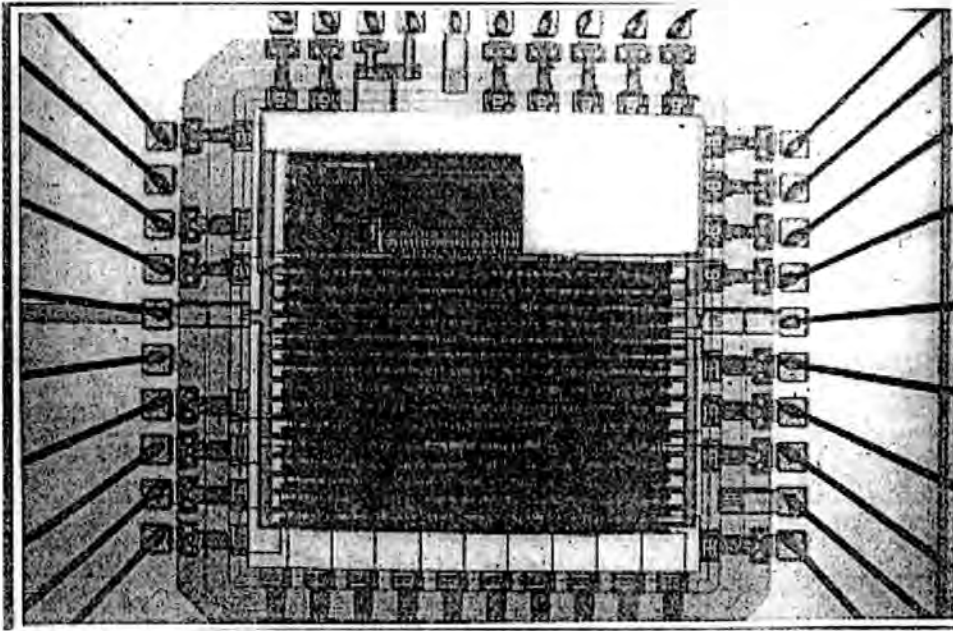


Figure 5.12: Photomicrograph of the CORDIC Cosine Chip

## 5.5 Comparisons between Sinusoidal Generators

Both the CORDIC and bit-serial IIR filter designs are of comparable size, however it must be stressed that the filter design would take-up more space and the number of transistors and size is a conservative estimate. The bit serial Booth multiplier and adder arrangement has a major disadvantage in that the poles of the filter determine the sinusoidal frequency, whilst the CORDIC approach has no poles and is always stable. The filter form has a latency on each multiplier of 24 clock periods which is comparable to the iteration time of 21 clock periods for the CORDIC processor.

The CORDIC and filter oscillators would need serious modification to enable multiple sinusoids to be generated. The CORDIC structure would need RAM and a sub counter to allow 21 sinusoids to be generated in a conveyor belt (pipelined) fashion. It would also need a better output stage, possibly an adder to add the sinusoids together, but the amplitudes of the sinusoids are constant. The filter structure would need a RAM store for the delay elements and substantial rerouting of the design. But the arithmetic modules can be made to compute many sinusoids if they have adequate storage and each filter generated sinusoid is computed and stored in a multiplexed manner. This approach originally designed for a cochlea simulation was done using  $4\text{ }\mu\text{m}$  nMOS design rules [80], having a sample rate of 500 kHz for each filter with a clock of 12 MHz. The filter approach automatically can provide

sinusoidals of varying amplitudes, but like the CORDIC design would need some output modifications unless the samples were sent out in a multiplexed manner and analogue circuitry provided the accumulation.

By increasing the maximum clock frequency and pipelining or multiplexing the computational elements with a slight increase in silicon area, these designs could generate sinusoids approaching a million a second. A software based real-time sinusoidal oscillator implemented on one INMOS 20 MHz T800 chip generates only 8 oscillators using the IIR filter approach [54]. Compared to software based systems running on DSP or microprocessors, these VLSI hardware designs outperform software based systems easily and with far lower clock speeds [105].

An interesting approach to sinusoidal tone generation [78] is to use oversampling in addition to sigma-delta modulation, in order to convert an  $N \times N$  bit multiplier to a  $N \times 1$  bit multiplier to be used for IIR [60] and FIR filter structures. The modulator is a second order type having a quantiser element which is implemented as a sign bit extractor. This structure has 4 additions and two delay elements, but on closer inspection two adders are  $N$  by 1 bit types and hence can be replaced by multiplexors. This topology is ideal for on-chip self-test applications where spare silicon floorspace is at a premium. A 16 tone oscillator implemented as two 8 tone oscillators operating at a clock rate of 5.2 MHz with an effective sampling rate for each oscillator of 655 KHz was proposed [77]. A four tone version using 84 % of a Xilinx XC4010 FPGA was then built.

In [117], Roza proposed an oversampling scheme using sigma-delta modulators suitable for video based applications, whose aim was to reduce power dissipation and clock skew in the datapaths. This oversampled sigma-delta approach is ideally suited to DSP based systems which require a direct interface to the analogue world. Sigma-delta converters are used before a one-bit DAC in order to re-quantise an  $N$ -bit signal and to reshape the noise spectra beyond the Nyquist rate. This improves the signal-to-noise ratio in band and is cheaper to implement than a multi-bit DAC.

---

---

## CHAPTER 6

# Decaying Sinusoidal Additive Synthesis

---

In this chapter, a parametric additive synthesis technique for musical tone generation will be presented. The classical approach approximates the waveform in the time-domain by a series of sinusoids, as in equation 2.3. This method is known as additive synthesis using Fourier analysis. However, in reality, musical instruments' timbre have bursty characteristics. Most instruments are difficult to model using Fourier techniques because sharp transitions require many high frequency harmonics (partials). Another failing of classical additive synthesis results from the assumption that the waveform is periodic for all eternity. Clearly, this approach is absurd and requires non-intuitive approaches to pretend that the waveform is periodic. Hence the implementation of short-time Fourier analysis, overlap-add and overlap-save techniques.

The alternative approach, as discussed in this thesis, is to assume that the basic element of the instruments' waveform is a wave packet. This approach is analogous to quantum mechanics; both cases the waveform is periodic and has an envelope. The envelope makes the waveform discontinuous in time, but Gabor proved that gaussian modulated sinusoids behave like sinusoids [35] and an additive synthesis technique could be implemented. In my section on additive synthesis it was stated that any waveform can be decomposed by a sum of simpler waveforms provided these simple waveforms were orthogonal to each other.

I have chosen to use a wave packet which has a sharp attack and an exponential

decay. This elemental waveform can be modelled as the impulse response of a damped second order filter. This waveform is very useful because it has the ability to model resonances (formants) in physical systems. Therefore, the proposed synthesis technique is inspired by classical instruments; e.g. violins, trumpets and the versatile human voice. The spectra generated by real musical instrument's can be modelled by parallel and/or series connection of second order filter sections. The rest of this chapter will be concerned with the investigation of these filters, their limitations and alternative solutions.

### 6.1 Analysis of a Second Order Filter Structure

It is well known that the z-transform of an exponentially decaying sinusoid, expressed as  $e^{-\alpha t} \sin(\omega t + \phi)$  is the following :-

$$\frac{\sin(\phi)z^2 + \sin(\omega - \phi)e^{-\alpha}z}{z^2 - 2e^{-\alpha} \cos(\omega)z + e^{-2\alpha}}$$

By dividing numerator and denominator by  $z^2$ , this filter structure is similar in form to the transfer function below.

$$\frac{1 + dz^{-1}}{1 + az^{-1} + bz^{-2}}$$

Notice the structure has two unity and three variable coefficients. When expressed as a linear difference equation, there are three multiplications and four additions. The multiplications arise from the coefficients  $a, b$  and  $d$ . The numerator part of the transfer function, when  $d = 1$ , generates zeros in the spectrum at 0 and Nyquist frequencies. This corresponds to a bandpass filter in the digital domain. Therefore resonances (formants) can be modelled using bandpass filters with frequency, amplitude, phase and bandwidth parameters.

However, for vocal synthesis [115], the waveform has a noticeable attack rather than an abrupt attack as generated by a second order filter. The approach chosen by Xavier Rodet [115] was to modulate the impulse response of a second order filter by an asymmetric Hanning window. This window was chosen because of its useful frequency placement properties and time domain behaviour similar to the natural glottal pulse in human speech [76]. The human ear is sensitive to phase below 500 Hz [93] and consequently a time-reversed glottal pulse, as approximated by integrating a short pulse twice would be differentiated relative to the human glottal pulse. The wavefunction chosen generates static zeros in the z-domain, whilst the real human impulse expands and shrinks with the pitch period. However, both methods are of equal merit provided the glottal spectrum has no fixed real-frequency zeros.

## 6.2 Analysis of the FOF technique

The *Forme d'Onde Formantique* (FOF) wavefunction is composed of three sections. The first section is the second order bandpass filter response function, the second is half a Hanning window and the last is a unity constant. The wavefunction is as described in equation 2.9 and is reproduced below.

$$\psi(t) = \begin{cases} \frac{1}{2}(1 - \cos(\beta t))e^{-\alpha t} \sin(\omega_0 t + \phi) & 0 \leq t \leq \frac{\pi}{\beta} \\ e^{-\alpha t} \sin(\omega_0 t + \phi) & t \geq \frac{\pi}{\beta} \end{cases}$$

This continuous time form has  $\frac{\pi}{\beta}$  equal to the attack time<sup>1</sup> and  $\alpha$  proportional to the bandwidth of the filter. The initial phase ( $\phi$ ) and angular formant frequency ( $\omega_0$ ) are also defined. This wavefunction is triggered by an idealised impulse train with a period inversely proportional to the pitch. Figure 6.1 shows one wavefunction grain initiated at zero time with a bandwidth of 40 Hz and attack duration and oscillation frequency adjusted to be visually clear.

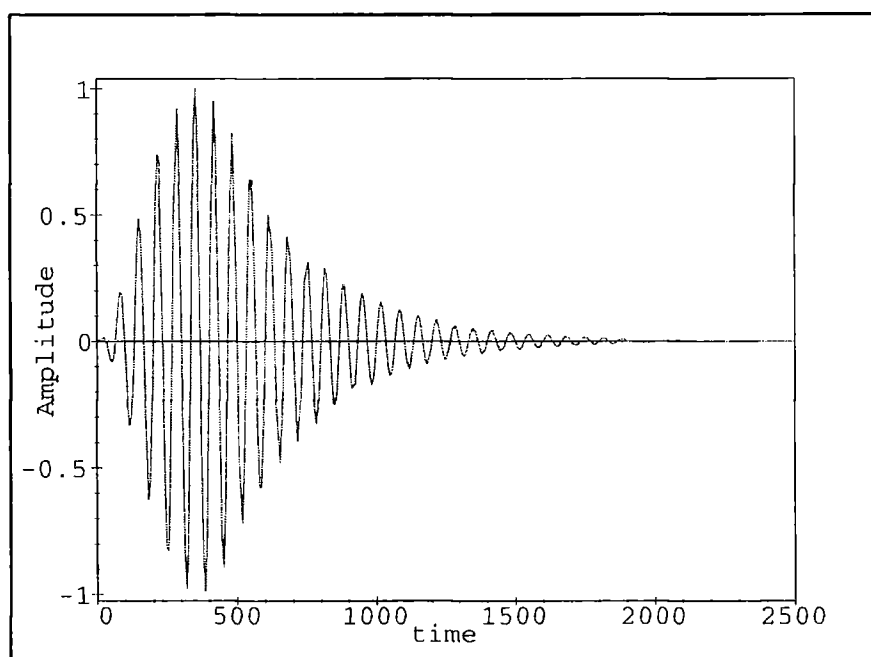


Figure 6.1: The FOF Wavefunction

The spectral properties of the wavefunction can be illustrated by taking the Laplace transform of the equation 2.9 in a piece-wise linear fashion. Here, it is necessary to

<sup>1</sup>Beta is known as the skirtwidth in the frequency domain and evaluates to the - 40 dB point.

assume that the wavefunction's amplitude falls to zero at infinite time. Only the envelope is operated on; the oscillatory part is neglected. The Laplace transform is found to be

$$\frac{1}{2} \frac{\beta^2 (e^{-\frac{\pi(\alpha+s)}{\beta}} + 1)}{((s + \alpha)^2 + \beta^2)(s + \alpha)} \quad (6.1)$$

The power spectrum can be obtained by substituting  $s = j\omega$  and multiplying by the complex conjugate. Then the answer is the square root of the expression and is shown in equation 6.2, with  $\mathcal{W} = e^{-\frac{\alpha\pi}{\beta}}$ .

$$|\Psi(\omega)| = \frac{\beta^2}{2} \frac{\sqrt{\mathcal{W}^2 + 2\mathcal{W} \cos(\frac{\pi\omega}{\beta}) + 1}}{\sqrt{\alpha^2 + \omega^2} \sqrt{(\alpha^2 + \omega^2)^2 + \beta^2(\beta^2 + 2\alpha^2 - 2\omega^2)}} \quad (6.2)$$

It can be shown that around the centre frequency of the formant, the power spectrum approximates to

$$|\Psi(\omega)| \approx \frac{1}{2} \frac{\sqrt{\mathcal{W}^2 + 2\mathcal{W} + 1}}{\sqrt{\alpha^2 + \omega^2}} = \frac{\mathcal{W} + 1}{2} \frac{1}{\sqrt{\alpha^2 + \omega^2}}$$

Therefore the shape of the FOF wavefunction about the formant frequency is of the form

$$\frac{K}{\sqrt{\alpha^2 + (\omega_c - \omega)^2}}$$

This corresponds to the second order bandpass filter structure.



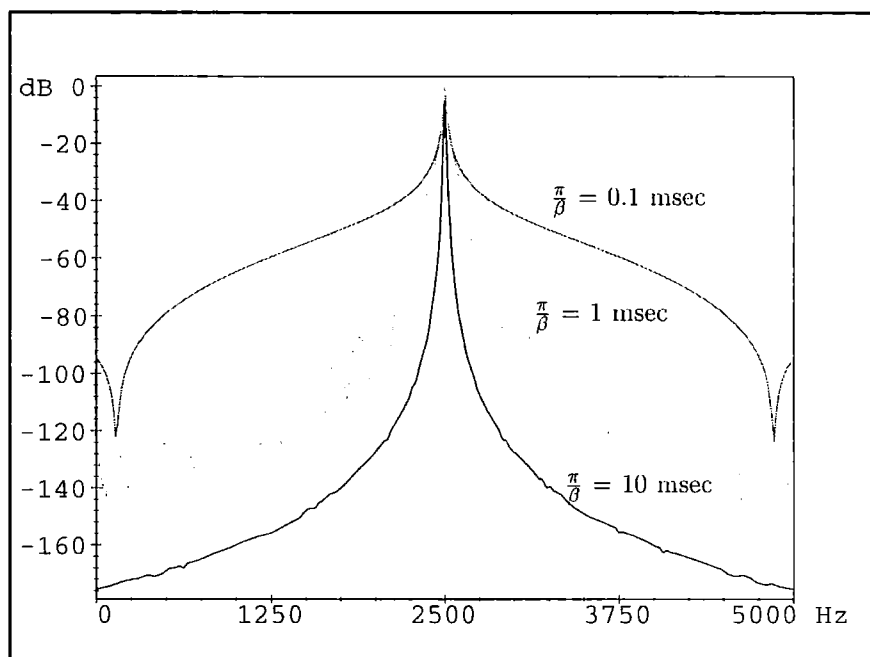


Figure 6.2: Power Spectrum of the FOF Wavefunction

A graphical plot of the power spectrum of the FOF wavefunction equation 2.9 was obtained by varying the angular frequency,  $\omega$  and plotting the magnitude in decibels of  $\Psi$ , using equation 6.2. This is shown in figure 6.2 with  $\alpha = 80 \cdot \pi$  for various attack durations ( $\frac{\pi}{\beta}$ ). Note that the bandwidth is 80 Hz.

Further work by researchers [21] applied the oscillation into the Laplace transform of the wavefunction by substituting  $s = s - j\omega_0$  into equation 6.1. To get the power spectrum they substituted the new  $s$  parameter by  $j\omega$  and followed the steps as above. The researchers then implemented the wavefunction in the time-domain using recursion and found that the arithmetic cost was 11 multiplications and 8 additions per sample. They generalised the skirtwidth parameter  $\frac{\pi}{\beta}$ . However, I believe this generalisation is ultimately deemed unimportant because the purpose of the skirtwidth parameter is to tidy the formant shape in the frequency domain and is normally constant for the timbre to be synthesised.

### 6.2.1 Implementation Approaches to the FOF wavefunction

In this section, I will investigate the two different approaches to implementing the FOF wavefunction. They are broadly cast as time domain, involving table lookup,

or filter based composed of defining the problem in a recursive manner. Both methods can result in efficient construction of the waveform, however they both have disadvantages.

### 6.2.2 Wavetable Based FOF Synthesis

This approach is the most common implementation of FOF and appears in CHANT<sup>2</sup> and CSOUND implementations. As CHANT source code is not commonly available, the discussion on this approach will involve the unit generator concept of CSOUND.

CSOUND is a program which allows musicians to create instruments by wiring control and generator building blocks together, in a manner reminiscent of the old analogue synthesisers with patch cords to route sound circuitry together. The system uses two files, one contains the instrument definition whilst the other tells the program when to play them. This concept is like the conductor and the orchestra. The program has been designed to compute the different flows in music optimally. This is achieved by separating the low frequency control from the regular and faster audio stream. This enables the use of vector based arithmetic to speed up computation, although implementation is on each unit generator due to the design of a fixed scheduler.

Now, we focus our attention on the FOF unit generator. The analysis will be simplified as CSOUND has recently been upgraded and the unit generator has been completely re-written.

---

<sup>2</sup>New version of CHANT includes FOF wavetable and filter approaches.

---

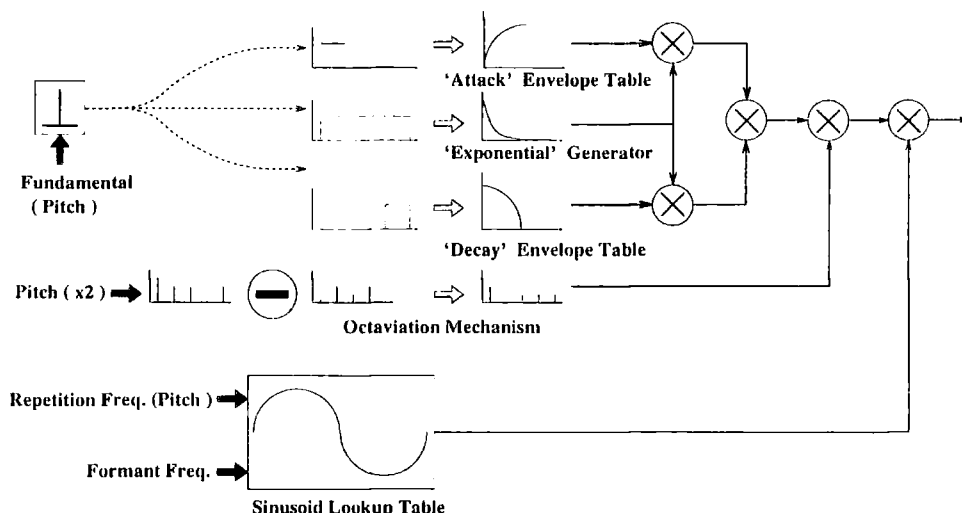


Figure 6.3: Simplified CSOUND FOF Implementation

In figure 6.3, the major implementation aspects of a wave-table based FOF wavefunction synthesiser are presented. When a new wavefunction is triggered, the attack envelope and exponential decay are adjusted to the nearest sample interval and act as if operating at a much higher sampling rate. After a period of time, related to the skirtwidth parameter, only the exponential decay is computed and finally the decay envelope is triggered. All these are multiplied together and the formant frequency is applied. The formant frequency is driven by its frequency and the pitch rate, aligning the sinusoid correctly with respect to the impulse trigger. The oscillator's phase adjustment is paramount to the perceived audio quality because it affects the frequency spectrum when multiple formants are summed together. This removes spectral nulls in the spectra which have unpleasant audio artifacts [116]. Finally, octavation [15] allows the musician to metamorphose the sound by altering the excitation frequency and amplitude in powers of two, thus picking out even and odd partials. This can be made gradually to fade out alternate excitations, eventually halving the rate at which new excitations are produced and thus halving the pitch. This effect can best be described as making a vocal imitation sound deeper and reverberant. This is shown schematically as two impulse trains, one twice the frequency of the other, being subtracted to create a compound impulse train.

The wave-table approach requires storage allocation for each new excited wavefunction and these resources are only relinquished when the wavefunction has decayed to zero at a certain time in the future. The inspired reader will also realise that each new wavefunction will consume more computational requirements, thus slowing down the synthesis. Each wavefunction is summed internally and for real-time

operation and all wavefunctions must be computed within the sampling period. Clearly, for high fundamental frequency, this approach would require many wavefunctions to be computed and thus be a burden on the finite resources available. The insane solution is to have a computational device having unbounded storage and computational resources, an infinitely powerful computer !

In [116], Rodet et al. waxed lyrical about the importance of floating point arithmetic in the control and synthesis of vocally inspired sounds. However, for VLSI implementation, floating point arithmetic units consume enormous amounts of silicon and possibly power consumption. In my opinion it is better to optimise the design to use cheap arithmetic processing elements using integer based arithmetic so that the control can be partitioned to run on a host processor.

The problem I discovered with the wavetable based approach as applied to VLSI implementation, is the unbounded nature of the computation and storage requirements. A single chip silicon based engine requires all computation to be computed within a sampling period and storage should be on chip. The wavetable method would require some computational fix to prune out wavefunctions, thus distorting the synthesis quality. If this approach is not followed, the system would only be optimistically capable of monophonic synthesis. The aim of this thesis is to create real-time polyphonic voice synthesis.

### 6.2.3 Filter Based FOF Synthesis

The filter-based approach is more amenable to silicon because time-overlapping of the impulse response to each source pulse input is the result of the autoregressive structure of the filter. Current filter approaches, most notably the second order filter approach as implemented in Samson's Box, require complex methodologies to solve the zeros in the spectrum resulting from the parallel summation of formant filters. This problem is common to many speech parallel based formant synthesisers, see equation 2.8 on page 21.

To overcome this difficulty I decided that a more complex filter having properties similar to the FOF wavefunction is required. This section investigates a hybrid architecture consisting of a filter generating the asymmetric envelope of the FOF wavefunction being multiplied (heterodyned) by a sine wave. The sine wave would be generated by a wave table approach optimised for silicon. The CORDIC based sine generator is ideal for this synthesis method. This generator was discussed earlier in this thesis and can be found in section 5.4 starting from page 91.

Using a Laplace to Z transformation method of section A for equation 6.1, the

---

envelope can be recoded into a structure suitable for digital implementation. The z-transform of this envelope can be shown to be equal to equation 6.3. A slight change of the wavefunction definition in equation 2.9 is to substitute  $\beta$  to equal  $\frac{\pi}{\beta}$ . This makes the  $\beta$  parameter equal to an integer sample value, directly corresponding to the attack time.

$$\Psi(z) = \frac{(e^{-\alpha\beta}z^{-\beta} + 1)}{2} \frac{z}{(z - e^{-\alpha T_s})} \frac{((1 - \cos(\frac{\pi T_s}{\beta}))e^{-2\alpha T_s} + (1 - \cos(\frac{\pi T_s}{\beta}))e^{-\alpha T_s}z)}{(z^2 - 2e^{-\alpha T_s} \cos(\frac{\pi T_s}{\beta})z + e^{-2\alpha T_s})} \quad (6.3)$$

It was found that implementation of equation 6.3 in a parallel structure, when modelled using MATLAB, created visible ripples in the time domain response. Implementing the equation as a cascade of a four non-zero tap FIR filter and two IIR filters of orders one and two respectively, gave the correct response. Figure 6.4 shows the FOF envelope with varying bandwidths and attack durations. These envelopes have also been normalised.

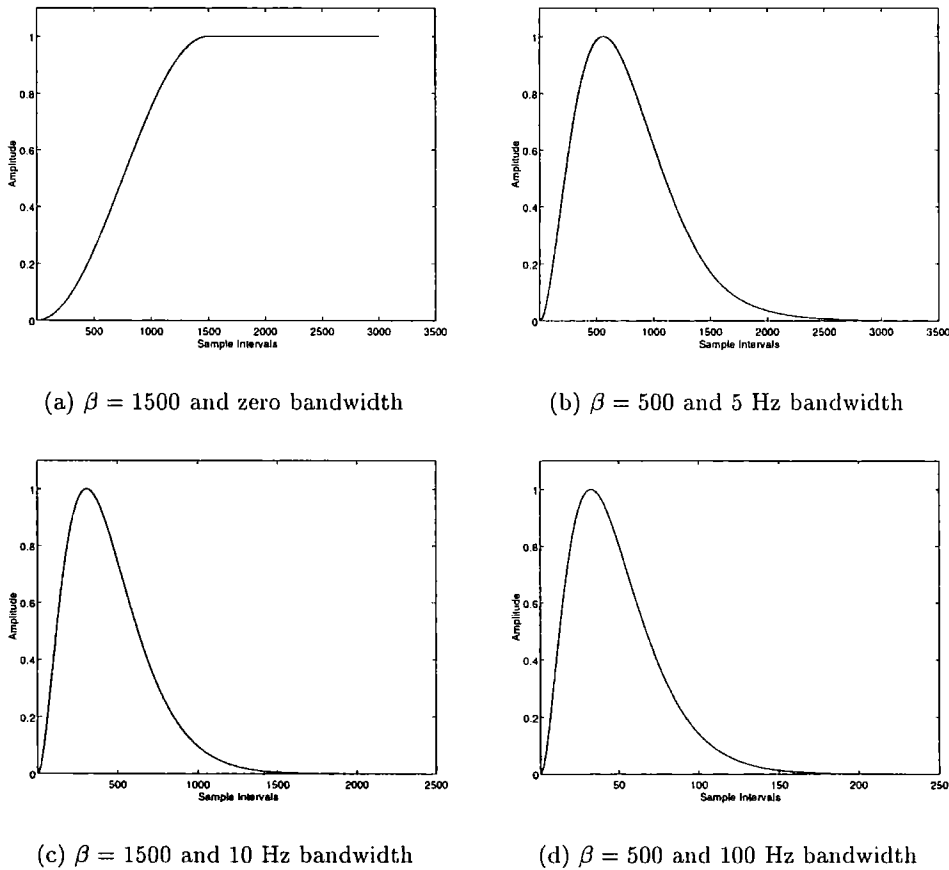


Figure 6.4: Time domain behaviour of FOF envelope generator

Assuming unit amplitude impulse excitation of the filter (equation 6.3), the maximum amplitude occurs when  $t = \frac{2}{\pi} \arctan(\frac{\pi}{\alpha\beta})\beta$ . The  $\beta$  parameter is assumed to be quantised to sample number. Therefore, 1 ms equals 44.10 samples, when the sampling rate  $f_s$  equals 44.1 kHz. Also,  $\alpha = \pi$  bandwidth /  $f_s$  and letting the bandwidth be represented by  $\Delta$ , gives the scaling factor of

$$\frac{f_s^2 e^{-2\Delta \frac{\arctan(\frac{f_s}{\Delta\beta})\beta}{f_s}}}{\Delta^2 \beta^2 + f_s^2} \quad (6.4)$$

Clearly as bandwidth and/or attack duration approaches zero, the factor approaches unity.

The pole-zero plot for the time domain responses of figure 6.4 are shown in figure 6.5.

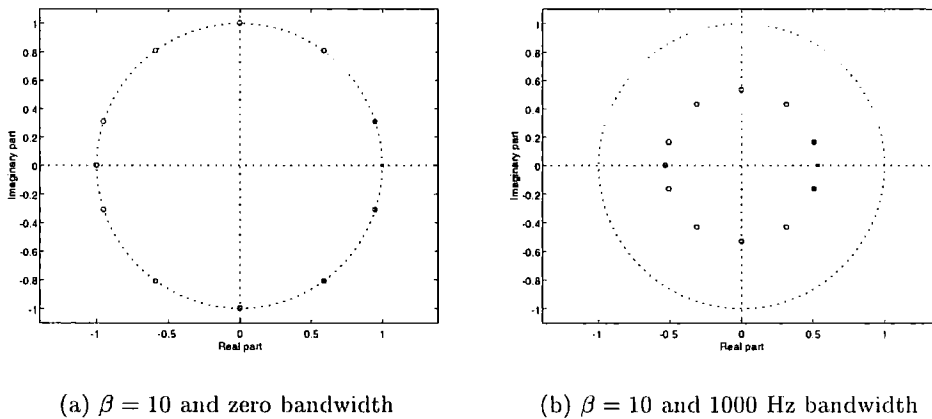


Figure 6.5: Zero-Pole plot of the FOF envelope generator

Notice that there are three poles and  $\beta + 1$  zeros and, ideally, pole-zero cancellation occurs around the three poles. This has the effect of requiring long integer wordlengths for stability, optimum performance and resolution for the bandwidth. However, the structure seems to be compact and usable. If the musician assumes the skirt-width/attack time to be constant and quantised to the sampling rate of 0.03 seconds, then 1323 samples are required. The total delay line size of the filter would be  $3 * \beta$ , and equals approximately 4 kilobytes. This memory requirement for each filter is excessive and could be reduced by using multi-rate techniques on each filter. This approach might make the synthesis technique unrealisable. Alternatively, if musicians use this synthesis algorithm and fix the attack time parameter, the provision of resources to implement such a parameter becomes unnecessary.

The complete filter-oscillator hybrid FOF formant generator [128] is shown in figure 6.6. Notice both oscillator and filter are initiated by the trigger (impulse generator) and the outputs of these are multiplied together. Heterodyning is normally applied to sinusoids and creates sidebands in the frequency domain. Lowpass filters are required to remove the sidebands generated by the sum of the two input sinusoids. In our case this final filter is not required. In the literature [76], Linggard mentioned a speech synthesiser built by Lawrence in 1953. This approach multiplied the envelope by a carrier sinusoidal generator and the resulting product was further multiplied by a sinusoidal generator operating at the carrier frequency plus the required formant frequency. The output of each formant-envelope product was lowpass filtered to remove the high frequency sidebands caused by heterodyning, and the resultant was summed to create a voice. An alternative, similar to my approach, is known as the ICL synthesiser implemented in 1976 by Underwood and Martin.

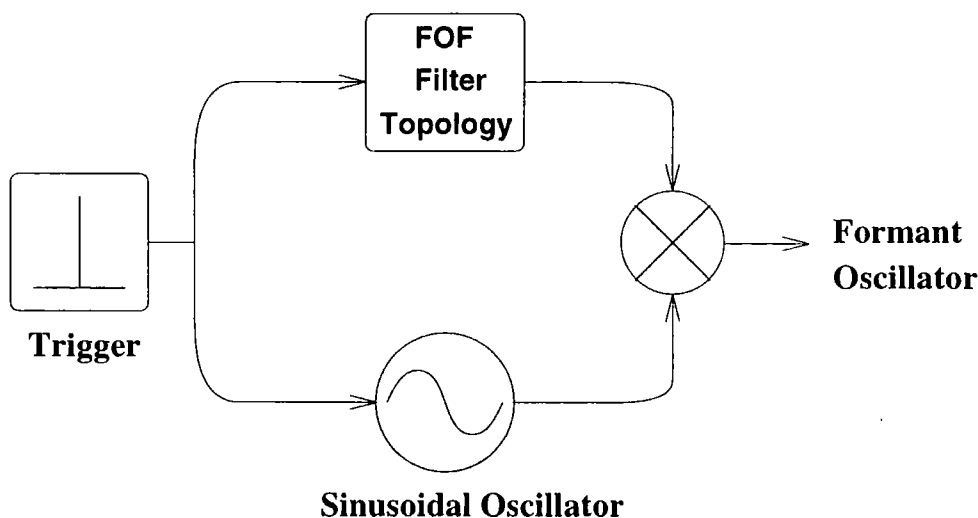


Figure 6.6: Heterodyned version of the FOF Algorithm

#### 6.2.4 Full Hanning Window FOF envelope

The next approach I experimented with was when the exponential decay is multiplied by a Hanning window, so  $\psi(t)$  equals the following :-

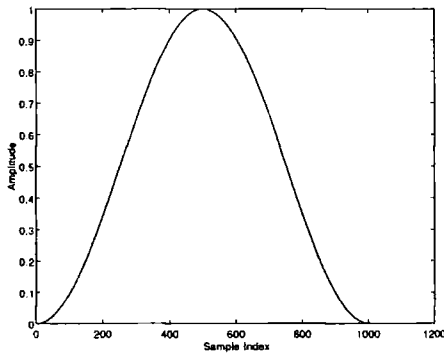
$$\psi(t) = \begin{cases} \frac{1}{2}(1 - \cos(\frac{\pi t}{\beta}))e^{-\alpha t} & 0 \leq t \leq 2\beta \\ 0 & \text{elsewhere} \end{cases} \quad (6.5)$$

The z-transform of equation 6.5 was originally derived through empirical means and can be solved as an extension to the z-transform of a Hanning window (without

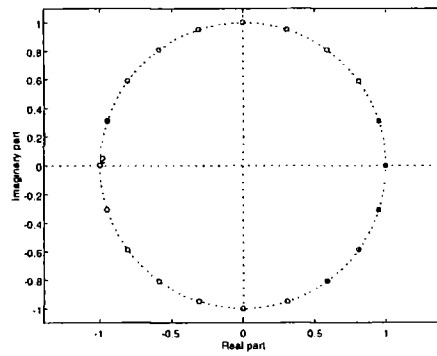
exponential decay). The empirical derived equation was found to be as described by equation 6.6.

$$H(z) = \frac{e^{-\alpha T_s} (1 - \cos(\frac{\pi T_s}{\beta})) (z^{-1} + e^{-\alpha T_s} z^{-2} - e^{-2\alpha \beta T_s} z^{-2\beta-1} - e^{-2\alpha \beta T_s} e^{-\alpha T_s} z^{-2\beta-2})}{2(1 - e^{-\alpha T_s} z^{-1})(1 - 2e^{-\alpha T_s} \cos(\frac{\pi}{\beta}) z^{-1} + e^{-2\alpha T_s} z^{-2})} \quad (6.6)$$

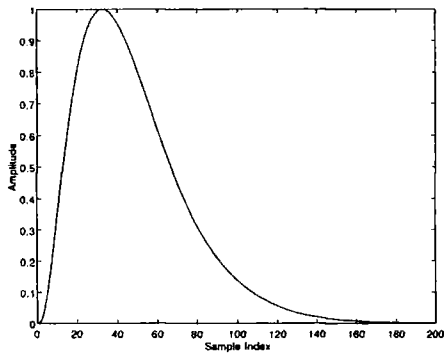
Figure 6.7 shows two envelope responses and zero-pole plots. The reader should note that again the time domain responses have been scaled using equation 6.4.



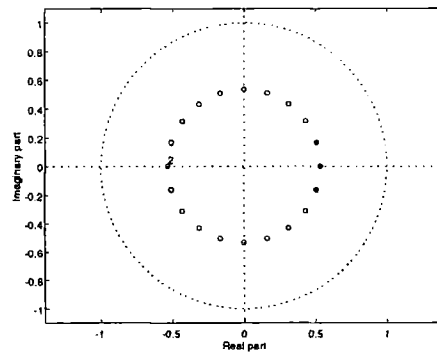
(a)  $\beta = 500$  and zero bandwidth



(b)  $\beta = 10$  and zero bandwidth



(c)  $\beta = 500$  and 100 Hz bandwidth



(d)  $\beta = 10$  and 1000 Hz bandwidth

Figure 6.7: Time domain and z-plane responses to Full Hanning FOF

As before with the FOF filter version, this structure will use a large amount of storage to function. Despite this, there are only 4 non-zero elements in the numerator (FIR filter). One major disadvantage occurs when  $\beta$  is very small, which results in a glitch. So the optimum setting of  $\beta$  is sufficiently long to allow the exponential to decay.



### 6.2.5 FOF filter structure with oscillation

In this section, I will propose a filter which removes the need for an attack parameter. This approach is used because the attack parameter is normally a constant for most synthesis algorithms involving FOF. The wavefunction chosen was the

$$t^2 e^{-\alpha t} \sin(\omega_0 t)$$

because it has an attack caused by the  $t^2$  element which eventually falls to zero because the exponential decay predominates as time progresses. Initially a colleague [103] proposed a version which used too many delay elements. I commented on this in his paper in 1996 [128]; but subsequently managed to reduce the number of delay elements by directly computing the z-transform using the Laplace to z technique (section A).

Equation 6.7 is the z-transform of  $t^2 e^{-\alpha t} \sin(\omega_0 t)$ . Note there is an additional scaling factor of  $\sin(\frac{2\pi f}{f_s})$  which should be added to the z domain transfer function.

$$H(z) = \frac{e^{-\alpha} z^{-1} + 2 \cos(\frac{2\pi f}{f_s}) e^{-2\alpha} z^{-2} - 6 e^{-3\alpha} z^{-3} + 2 \cos(\frac{2\pi f}{f_s}) e^{-4\alpha} z^{-4} + e^{-5\alpha} z^{-5}}{(1 - 2 e^{-\alpha} \cos(\frac{2\pi f}{f_s}) z^{-1} + e^{-2\alpha} z^{-2})^3} \quad (6.7)$$

This filter topology requires a scaling factor which is crudely equal to

$$\frac{4e^{-2}}{\alpha^2}$$

using the starting equation of  $t^2 e^{-\alpha t}$ . The time domain response to this filter is shown in figure 6.8.

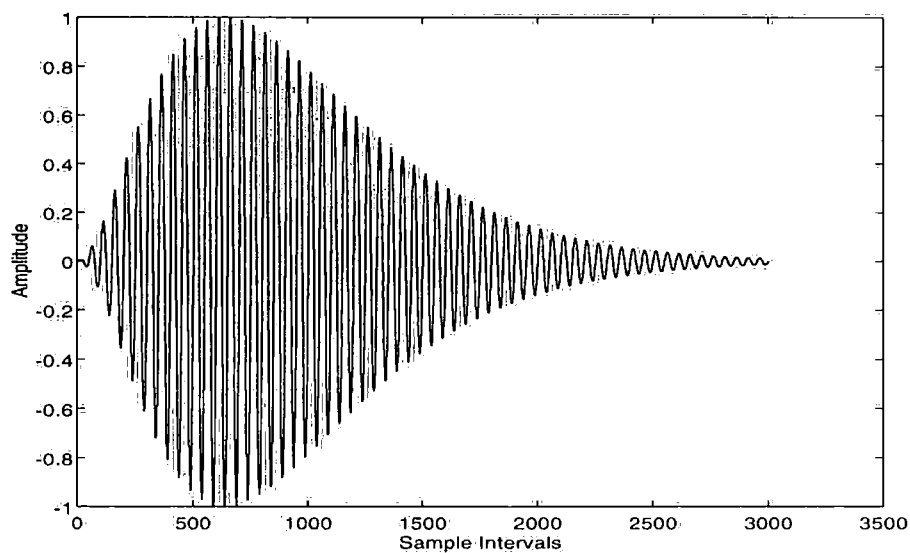


Figure 6.8: Time Domain response of the  $t^2 e^{-\alpha t} \sin(\omega_0 t)$  wavefunction

The zero-pole diagram for a frequency of 480 Hz and a bandwidth of 250 Hz is shown in figure 6.9.

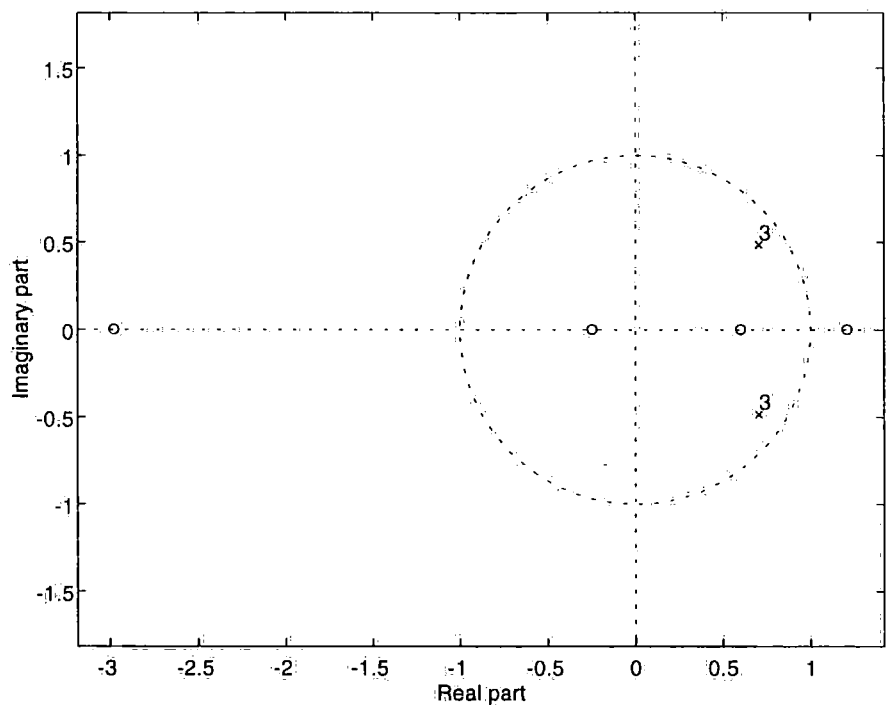


Figure 6.9: z-plane response to  $t^2 e^{-\alpha t} \sin(\omega_0 t)$  wavefunction

In the z-plane plot of figure 6.9 I would like to point out that there are four zeros. Each pair is almost the reflection about the unit circle of each other. Notice that the zeros outside the unit circle may cause problems when integer arithmetic is used in the filter implementation. This may be solved by factorising the zeros.

The frequency and phase spectrums of this wavefunction with narrow and wide bandwidths and constant frequency of oscillation are shown in figure 6.10.

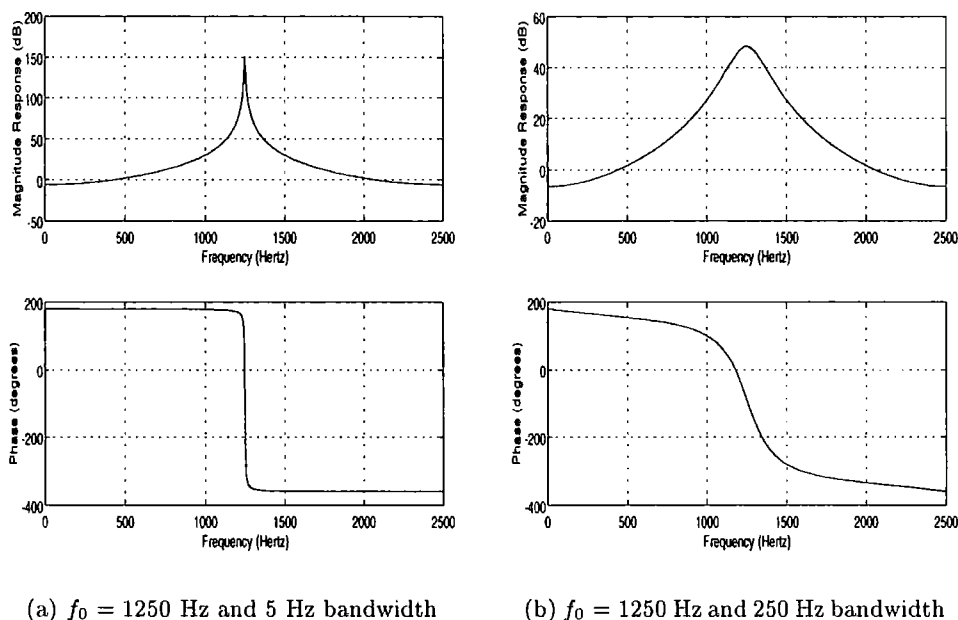


Figure 6.10: Magnitude and Phase Spectra for the  $t^2 e^{-\alpha t} \sin(\omega_0 t)$  wavefunction

The responses shown above are not normalised and the magnitude responses are very similar to the normal second order bandpass filter sections. The filter topology is approximately three times the computational cost of a second order filter, but this extra cost results in a  $t^2$  shaped attack.

### 6.2.6 Miscellaneous Wavefunction Designs

In the penultimate section of this chapter, I will describe some structures which can generate FOF style wavefunctions.

1. The standard FOF wavefunction as described by equation 2.9 can be expanded into the following form.

$$\psi(t) = \frac{1}{2} e^{-\alpha t} \sin(\omega_0 t)$$

$$- \frac{1}{4} e^{-\alpha t} \sin(\omega_0 t \pm \frac{\pi \tau}{\beta})$$

Where  $\tau$  varies from 0 to  $\beta$  and then remains constant. It is easy to prove that when  $\tau$  is equal to  $\beta$ , the equation above becomes the familiar decaying sinusoid. If  $\tau$  is allowed to decrease to zero from  $\beta$  at some later time, then the FOF as implemented in CSOUND results. I explored this approach but it was not accurate enough. It was possible to hear the difference.

2. In the literature [115], Rodet mentioned an envelope of the form  $e^{-\alpha t \frac{(t-c)^{-2}}{t}}$ , where  $c$  is a constant. Consequently I proposed the following envelope :-

$$k \frac{1}{e^{\alpha t} - 1}$$

This envelope is the max plank thermal distribution for black body radiation and has a peak at  $\alpha = 2.821$ . Those with knowledge of Quantum Mechanics will realise that the equation is a closed form expression for an infinite series. Consequently, both waveforms are not implementable as a filter.

3. Miller Puckette has a patent [112] on a similar structure using multiplications and sinusoidal modulation to generate the following

$$X(t) = \sum_{k=-\infty}^{\infty} e^{j(k\omega_0 + \omega_s)t} e^{-\frac{|k\omega_0 + \omega_s - \omega_c|}{\delta}}$$

All these implementations are unsuitable for filter implementation due to infinite series or other problems. They can be implemented using look-up tables, additions and multiplications and are ideal for implementation on general purpose computers or DSPs.

### 6.3 Approaches to formant synthesis

This chapter has concentrated on the various implementations of formants in the time domain. They can either be implemented in the time domain, using lookup tables or as the impulse response of a set of filters.

In this thesis, I proposed the time squared exponential decaying sinusoid wavefunction as a replacement for FOF because it can be implemented as a filter. This allows an elegant and uncluttered implementation without multiplication and generation of the sinusoid. However, a filter implemented in integer arithmetic has a reduced number of allowable bandwidths and frequencies caused by the quantisation of the

---

coefficients. This is thought to be of minor consequence because the human audio system prefers rich evolving textures rather than precise static sounds.

This chapter will conclude with a table showing the relative cost of the FOF and full Hanning FOF envelopes and the time squared decaying sinusoid filter implementations. All filters in this analysis are implemented in a cascaded direct form structure and any gain factors are ignored. In this analysis, the actual coefficient values are assumed to be automatically generated by a host processor with sufficient computational power.

Structure	Multiplications	Adds/Subtracts	Delays	Sections
FOF	5	5	$\beta + 5$	4
Hanning FOF	7	7	$2\beta + 5$	3
Time Squared	11	10	11	4

Table 6.1: Comparisons between Formant Filter Topologies

Both the FOF and Hanning window FOF filters are comparable in computation and storage requirements, but they need either a second order sinusoid generator or a lookup table to shift the envelope around the formant frequency. The disadvantage with these envelope based filters is that the  $\beta$  or window parameter is quantised in units of the sample period. It is possible to create all-pass filters to generate non-integer digital time delays [72], but the computational cost becomes excessive in this application. The surprise result from looking at table 6.1 is that the Time Squared filter uses the most arithmetic resources but the least storage requirements.

All these filter topologies do not have an envelope to tidy up the exponential decay. This is unlike the CSOUND and CHANT implementations and is deemed unnecessary because of increased computational and storage costs. A compromise solution would be to truncate the response after three time constants, since the decay would be within 0.1 % of its final value. This would only happen if the impulse generator's pulse rate were sufficiently slow, so that no overlapping of responses could occur within the filter bank. A counter would be required to orchestrate this procedure.

In the next chapter, the time squared sinusoidal decaying oscillator will be extended to include the source implementation of this source-filter topology.

---

## CHAPTER 7

# Parametric Additive Synthesis Implementation

---

In the previous chapter the reader was guided through the different implementations of damped sinusoidal oscillators. By providing damping, a spread of harmonics appears in the spectrum, instead of a single harmonic (with zero bandwidth). As most sound generating devices have built-in damping, it seems reasonable to try to model this using wavefunctions having this property. The advantage, when compared with Fourier-based Additive Synthesis using sinusoids, is the reduced number of oscillators required to model the spectra. Consequently sinusoidal based synthesis techniques require advanced algorithms, either FFT based [34] or multirate [104, 106, 108] in order to be cost effective because of the immense computational burden of generating many thousands of sinusoids. In comparison the alternative, fixed sample rate approaches [10, 58, 86, 127] can only produce a few hundred. Formant based systems compute decaying sinusoids and thus the number of oscillators is reduced to a minimum of one per gross formant region of the spectrum to be modelled.

This chapter extends my work done on implementing a decaying sinusoid oscillator as a filter which has a fixed attack envelope. The first section will discuss implementable details of the  $t^2 e^{-\alpha t} \sin(\omega_0 t)$  filter. The remaining sections will deal with the excitation generator that drives the filters and also the coalition of the separate filters for audio output. The structures to be presented will be for the implementation of a single formant. For modelling of formant based instruments, e.g. violin, guitar and the human voice, a parallel collection of formants is needed.

## 7.1 Optimisation of the filter $t^2 e^{-\alpha t} \sin(\omega_0 t)$

The filter structure as shown by the z transfer function of equation 6.7 consists of a transversal (FIR) and three second order IIR filters implemented in cascade configuration because the parallel form has poor zero sensitivity and scaling problems [53].

### 7.1.1 Analysis of the Transversal Element of the Filter Topology

The numerator (FIR part) of the transfer function gives rise to four zeros which are configured in pairs related by the reflection around the unit circle and which lie on the real axis. In tests using MATLAB, the numerator as described by equation 6.7 does not precisely give the correct zeros as mentioned above. It seemed reasonable to assume that these errors occurred due to the conversion between symbolic and numerical form. This statement was proved to be correct by investigating the different output numerical formats of MATLAB (normal and rational).

For each pair of zeros, the following factors arise.

$$N(z) = z^2 + \xi z + 1 \quad (7.1)$$

If  $|\xi| > 2$ , two real zeros satisfy symmetry with  $z_1 = r$  and  $z_2 = \frac{1}{r}$  and equation 7.1 can be expressed as follows :-

$$N(z) = z^2 + \left(r + \frac{1}{r}\right) z + 1$$

If  $|\xi| < 2$ , then two complex conjugate pairs result and  $N(z)$  becomes :-

$$N(z) = z^2 + 2 \cos(x) z + 1$$

if  $|\xi| = 2$ , then there are two zeros either at  $z = 1$  or  $z = -1$ .

For four zeros, an expression for  $N(z)$  becomes

$$N(z) = z^4 + \xi z^3 + \gamma z^2 + \xi z + 1$$

If  $\xi < 4(\gamma - 2)$  and  $\gamma > 2$ , the expression can be factored as shown below.

$$\begin{aligned} N(z) &= z^4 \\ &+ \left[ \frac{2r^2 + 1}{r} \cos(x) \right] z^3 \end{aligned}$$


---

$$\begin{aligned}
& + \left[ r^2 + \frac{1}{r^2} + 4 \cos^2(x) \right] z^2 \\
& + \left[ \frac{2r^2 + 1}{r} \cos(x) \right] z \\
& + 1
\end{aligned}$$

This equation can be factorised whenever a zero or more lie on the real axis or on the unit circle.

### 7.1.2 Transversal Filter Structure

The FIR part of the formant oscillator as mentioned in section 7.1.1, can be factorised according to the real coefficients in  $N(z)$ . As these zeros are always on the real axis, their values are always the reciprocals of each other. This is because the filter can never be allowed to have zero bandwidth because it would then become unstable, growing in amplitude without bounds. In tests, the bandwidth will always be a small fraction of the sampling rate, so the leftmost zero will appear within  $z = -10$ . However, at the time of writing, the designer has not been able to factorise and simplify the FIR structure. The hoped benefits would be reduced computation of the coefficients on the host processor.

A FIR filter can be implemented in direct form by investigating the figure 7.1, which is in flow graph form. The  $z^{-1}$  arms represents delays, the  $+$  below nodes represents add/subtract elements and the  $h(0 \dots 5)$  represent the multiplications.

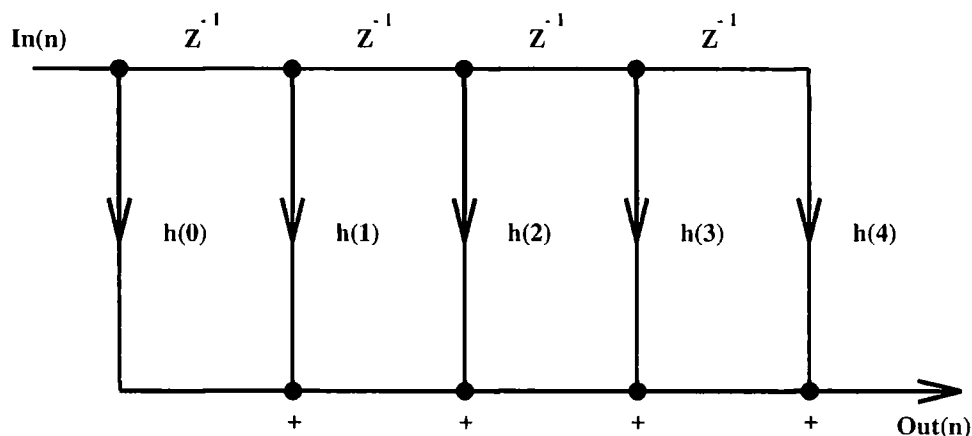


Figure 7.1: Flowgraph of Direct Form FIR Filter Structure

The filter topology chosen was the direct form as shown in figure 7.1. However, alternative forms are possible [102], by expressing the filter as a cascade of second



order sections or in fourth order form with zeros at  $re^{\pm j\theta}$  and  $\frac{1}{r}e^{\pm j\theta}$ . The latter approach is sensitive to quantisation and maintains the linear phase properties, but with an increase in multiplication cost.

### 7.1.3 IIR Filter Structure

The denominator of equation 6.7 is represented by three identical IIR filters. These filters have the direct form similar to figure 5.2 but the  $-1$  gain factor on the  $y(n-2)$  output becomes  $-e^{-2\alpha}$  and the other gain factor has an additional factor multiplied to it and equal to  $e^{-\alpha}$ . An undesirable problem occurs when the oscillation frequency is very small; the three complex conjugate poles become two pairs reflected about the unit circle and two real poles centred around  $z = 1$ .

There has been a large amount of information published on filter topologies [16, 19, 99, 154] and their analysis. The topology of the filter imparts important benefits to the designer, such as coefficient sensitivity and noise immunity. In [87] various filter forms were analysed for the equalisation of digital audio. The direct canonical filters were found to have poor resolution of poles near the origin when the coefficients are quantised. The structure had poles densely packed at high frequency but had the advantage of the minimum number of delays. Ladder and lattice filter structures have good coefficient sensitivity and roundoff noise performance and are the digital equivalent to their analog counterparts. In [84], Massie comments on the decoupling of bandwidth and tuning coefficients. These filters are found in LPC synthesisers and in a musical instrument built by E-Mu. They are normally cascaded to represent formant regions and are essentially a series formant synthesiser. In a further paper [23] a method of controlling bandwidth and frequency was proposed so that the frequencies would be logarithmically swept. This allowed musically useful applications of the filter to sound transformation.

Work done by [3, 31] on digital sinusoidal oscillators might be adaptable to musical purposes, if the structures could be implemented with decay (bandwidth) properties. This would be of benefit in producing ultra low frequency formant regions.

The approach I took was to use the Gold-Rader coupled form [113] because the wavefunction requires a wide range of frequencies and bandwidths for musical generation and because this filter structure quantises the poles on a uniform mesh around the unit circle. The work done by Kingsbury [67, 68] generalised the coupled form approach for both poles and zeros near the real axis and could be useful if the FIR part of the filter topology were to be included with some of the IIR filters.

The figure 7.2 shows a schematic diagram of the coupled form filter with a constant

gain of  $R \sin(\theta)$  and two complex conjugate poles.

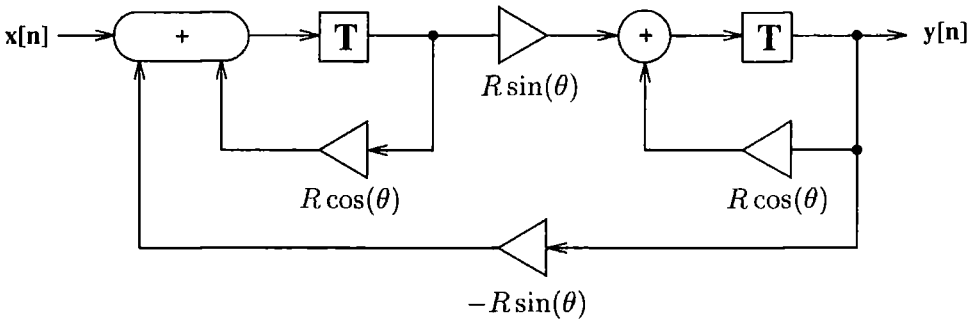


Figure 7.2: Second Order Coupled IIR filter Form

7.2 Source Excitation Driver

In this section I will discuss the design and implementation of the impulse generator. This generator then drives the filter bank, as described earlier in this chapter. There are many ways of generating impulses (pulse trains) and in this thesis two designs will be presented. The first one is based on discrete summation formulae to approximate a pulse train with resolution finer than the sample rate. The latter one uses a high speed impulse generator and multi-rate techniques to achieve the same effect.

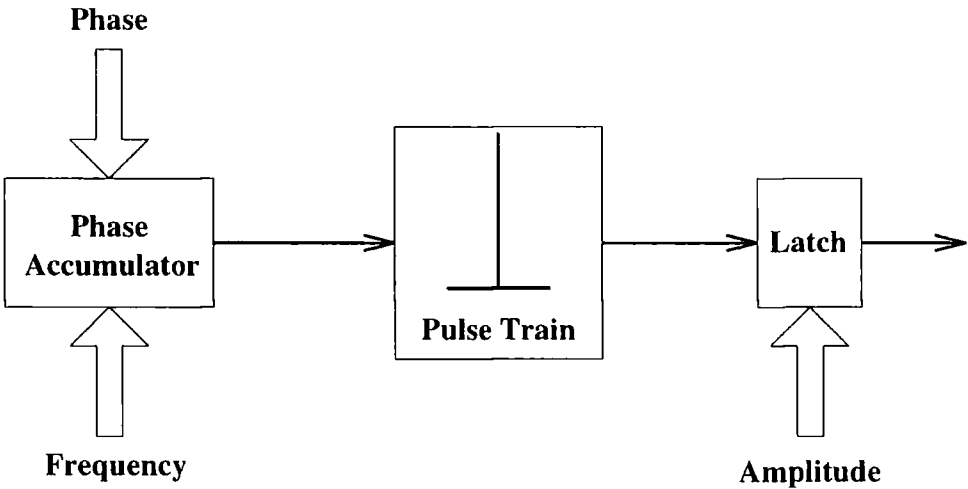


Figure 7.3: Simple Impulse Generator Schematic

The obvious method of implementing a discrete-time version of an impulse train is to approximate it by a unit-sample pulse-train. This can be easily implemented by using a phase accumulator. The most significant bit of the accumulator is designed

to flip at the repetition rate. This line then becomes a trigger to drive a multiplexor and a latch. The multiplexor selects zero or some amplitude  $A$  in the chosen wordlength and number format. This can be clearly seen in figure 7.3 However, with  $f$  being the repetition frequency, the period in samples would be  $F_s/f$  and the expression must be an integer. If the expression is not exactly an integer, the pulse rate will jitter around the nearest sample interval. The human pitch perception is so sensitive that the ear can detect this jitter, which corresponds to a tenth of a percent of the sampling rate. How can the designer overcome this dilemma ?

The Fourier transform of a pulse (strictly speaking a Dirac delta function), is the sinc function

$$T_s \frac{\sin(\frac{\pi f}{f_s})}{\frac{\pi f}{f_s}}$$

Where the sinc function is defined to be unity when  $f$  is zero. The function has many zeros related by  $\pm \frac{2\pi n}{T_s}$ , with  $n \geq 1$ , as shown in the normalised amplitude against normalised frequency diagram 7.4 below.

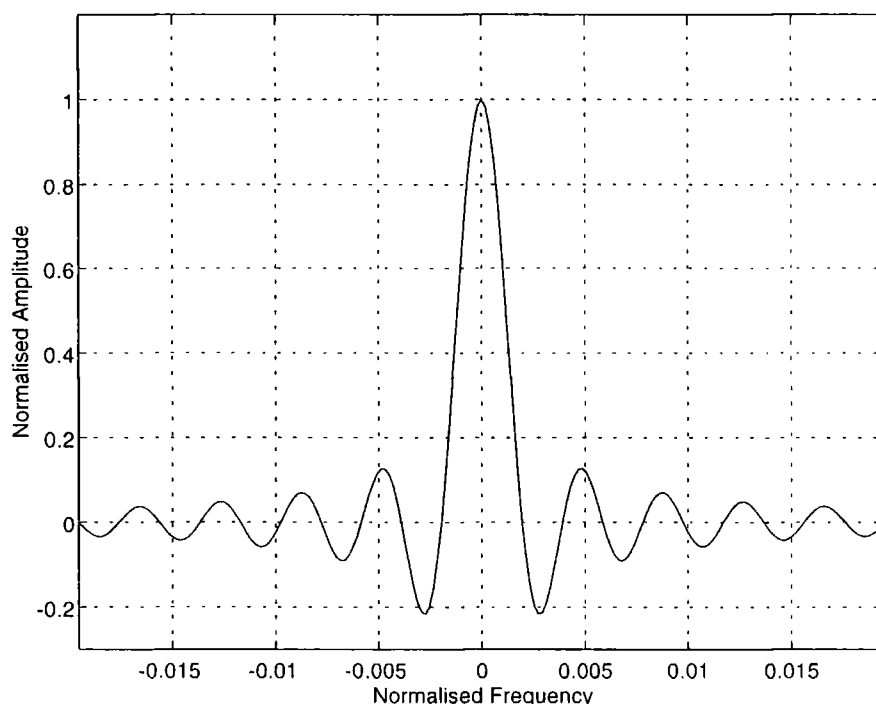


Figure 7.4: Amplitude vs Frequency for the sinc function

The figure 7.4 should really be a magnitude spectrum and the diagram clearly shows the zero crossings. The figure shows a great deal of energy above the first

zero causing aliasing of all higher harmonics. When a pulse train is generated, the impulses appear as lines within the sinc spectra. The lines are separated by  $\frac{2\pi}{T}$ , where  $T$  is the period between pulses. To allow equal excitation of any filter set to any formant frequency, the first zero, at  $\frac{2\pi}{T_s}$  must be well above the highest formant frequency of the system.

Therefore, to allow pulses with finer than the sampling period resolution, the pulses must be resampled at a higher sampling rate. One approach is to describe the effect of a higher sampling rate pulse on a lower sampling rate. This is essentially an impulse which is low pass filtered. Low pass filtering has a sinc impulse response function and can be band-limited to avoid aliasing in the frequency domain. The low pass filter acts as an interpolator and is the digital analog of a plane wave incident on a circular aperture in optics.

### 7.2.1 Discrete Summation Formulas for Impulse Generation

In [129], Stilson and Smith presented a sum of sinc functions represented by

$$y(n) = \sum_{l=-\infty}^{\infty} \frac{\sin(\pi(n + l \cdot P))}{\pi(n + l \cdot P)}$$

where  $P = \frac{f_s}{f}$  and is *not* an integer and by using discrete summation formulae  $y(n)$  becomes

$$y(n) = \frac{\sin(\frac{\pi n M}{P})}{P \sin(\frac{\pi n}{P})} \quad (7.2)$$

where  $M$  is the number of harmonics and is always odd because the impulse train has one harmonic at DC and an even number of non-zero harmonics. Thus

$$M = 2 \left\lfloor \frac{P}{2} \right\rfloor + 1$$

Hence  $M$  is the largest odd integer not exceeding the period  $P$  in samples.

Formant generation can be achieved by using discrete summation formulas as proposed by [95] hinges on the closed form for summing sinusoids as shown in equation 7.3 with  $a < 1$ .

$$\sum_{k=0}^N a^k \sin(\theta + k\beta) = \frac{\sin(\theta) - a \sin(\theta - \beta) - a^{N+1} [\sin\{\theta + (N+1)\beta\} - a \sin(\theta + N\beta)]}{1 - 2a \cos(\beta) + a^2} \quad (7.3)$$

Notice that the right hand side of equation 7.3 has variable amplitude controlled by  $a$ . The normalisation factor for expression 7.3 is

$$\sqrt{\frac{1 - a^2}{1 - a^{2N+2}}}$$

The normalisation factor becomes erroneous when any harmonics from other summation formulae fall upon the computed formulas, overlapping results giving an incorrect amplitude factor. I suggest that the reader consults the relevant article [95] to obtain the solution. The summation of many sinusoids of zero phase relative to each other will result in the approximation of an ideal unit impulse.

Both the approaches mentioned in this section compute functions directly in the time domain and thus require an expensive (for integer) division circuit. Relating these algorithms to the spectral based parametric additive synthesis technique proposed in this thesis results in a rather expensive scheme to implement octavation, requiring multiplication. As mentioned earlier, multiplication is currently more expensive to implement in silicon than addition as it is performed from multiple addition circuits in VLSI.

In the next section, a multi-rate technique will be proposed which can be made very computationally efficient.

### 7.2.2 Impulse generation using Decimation

In section 7.2, the algorithm requires accurate control of the impulse rate. In CSOUND the FOF algorithm requires accurate control of the impulse generator such that if the impulse falls within a sample period, the wavefunction is advanced by the time difference between the impulse firing and the next sample interval. Failure to take this into account would result in poor quality synthesis. Therefore the impulses are generated at a high sampling rate and the output is resampled at a lower rate, which is the audio sampling rate.

The impulse generator uses the phase accumulator with a single-bit output. This bit latches a register which holds the amplitude of the impulse if the bit is one, otherwise it outputs zero, as shown in figure 7.3. Using this simple scheme, octavation and simple filter scaling can be achieved. The output is then passed through a decimator.

#### 7.2.2.1 The Decimator Structure

A decimator is conceptually a device which drops every  $M - 1$  samples from the high sample rate. Essentially it acts as a switch operating at the low sampling rate

---

and provides a decrease in sampling rate by a factor of  $M$ . However, the output of the switch requires band-limiting because the information is passable at the high sample rate, but when resampled at the lower rate would cause aliasing.

By analysing the FOF unit generator in CSOUND, the effective sample rate required to provide adequate temporal resolution was found to be the following.

$$\text{effective } SR = \frac{2^{24} \cdot \text{fundamental}}{\text{fundphs} \cdot f_s}$$

The equation arising from CSOUND has the value  $2^{24}$  in the denominator. This value is the maximum length of a 24 bit phase accumulator as implemented in CSOUND. Consequently fundphs varies from 0 to  $2^{24} - 1$  inclusive. It turns out that the worst case sampling rate is when fundphs is equal to 1.

Where fundphs = 1 and fundamental = 7KHz results in an effective sample rate of 2.7MHz. This rate corresponds to a time scale of 360ns. The decimation factor is approximately  $2^6$  times the original sample rate of 44.1KHz. The smallest increment generated by the impulse generator using a 24 bit phase accumulator is found to be equal to 0.168Hz. By approximating the interpolation factor to be  $2^6$  results in a maximum excitation frequency of 7418.86Hz.

#### 7.2.2.2 Decimator Design

The first approach is to implement a decimator which removes every 64 samples and the resulting rate output drives a FIR lowpass filter. Using the Remez-exchange algorithm and multi-rate design equations in [53] and allowing 0.01 dB ripple in the passband and - 72 dB rejection in the stop band produces a finite impulse response filter with a length of 4953. This filter is not implementable with the Remez-exchange algorithm and so a multiple stage decimator approach was investigated. It can be shown that one big filter and decimator can be split into a cascade of smaller, cheaper and easier to design sections. These sub-sections have the property of conforming to the original specification. It was discovered that the optimum multi-stage implementation using source code available on disk [53], was in two stages.

The first stage decimator has a decimation factor of 16 and has 81 coefficients, the second stage has a decimation factor of 4 and requires 329 coefficients. The cascade of both filters will meet the required specifications. The actual filter coefficients were generated using the signal processing toolkit of MATLAB.

By using powers of two for decimation ratios, a compact and efficient decimator can be designed [32]. However, the current implementation has excessive latency caused

by the FIR filter coefficients and excess computation caused by dropping every  $M-1$  samples from the stream. It is possible to coalesce both filter and commutator to form an efficient structure known as a polyphase filter. Essentially the polyphase filter reschedules the data flow to only compute the  $M$ 'th samples which appear at the low sampling rate output. Therefore, the filter becomes a time varying filter because the coefficients are changed on a sample by sample basis. Thus the original filter becomes a series of parallel filters, each one computing once per sample. The minor disadvantage of this approach is that the filter length must be a multiple of the decimation factor.

Since the filter is decomposed into  $D$  sub-filters, each of length  $K$ ,  $K$  can be determined by the following formula, with  $D$  as the decimation factor and  $M$  is now the length of the original FIR filter.

$$K = \frac{M}{D} \quad (7.4)$$

The filters designed earlier have new lengths of 96 and 332 respectively, by using equation 7.4. This gives a sample delay of 6 and 83, so a total delay of 89 samples, instead of 428 samples. However, the structure still requires 428 coefficients to be stored. It should be noted that the filter coefficients are symmetrical, so using suitable address logic, the coefficient storage can be reduced by a half.

An example of a reduced sample delay polyphase filter with time-varying coefficients is shown diagrammatically in 7.5 with  $M = 12$ ,  $D = 4$  and  $K = 3$ . The little arrows shaped in a loop determine the coefficients which will be used for the multiplication operations on a sample-by-sample basis.

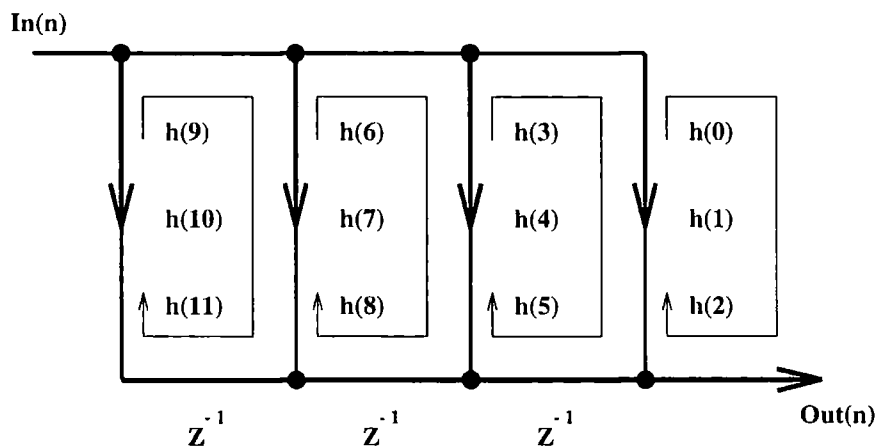


Figure 7.5: Reduced Dynamic Memory Storage Polyphase Filter

The filter topology shown in figure 7.5 in flow graph form is the standard FIR filter (see figure 7.1) having undergone certain transformations as described in [32].

It would be possible to implement the decimator using Infinite Impulse Response filters, but apart from the design complexity required to meet the specification, they are not linear-phase. I believe that the polyphase approach allows the FIR filter to compete well with an IIR filter structure in terms of storage and number of arithmetic operations per second.

### 7.2.2.3 Reduced Coefficient Storage Design

As mentioned in the previous section on decimator design, the coefficient storage required for each polyphase filter can be reduced due to the coefficients being symmetrical. Taking the coefficients of a normal FIR and mapping them with the polyphase filter-bank equations, it is possible to determine the coefficient symmetries.

$$p_k(n) = h(k + n \cdot D) \begin{cases} k = 0, 1, \dots, D - 1 \\ n = 0, 1, \dots, K - 1 \\ K = \frac{M}{D} \end{cases} \quad (7.5)$$

Using the equations 7.5 shown above, it can clearly be seen how many filter banks are required for the decimation process and the relationship between the polyphase filter's coefficients and the original filter's coefficients  $h(n)$ . For the  $D = 16$  and  $M = 96$  case we get the following impulse response structure for each sub-filter :-

15	31	47	32	16	0
0	16	32	47	31	15
1	17	33	46	30	14
2	18	34	45	29	13
3	19	35	44	28	12
⋮	⋮	⋮	⋮	⋮	⋮
13	29	45	34	18	2
14	30	46	33	17	1

The numbers above represent the coefficient index relative to the original FIR filter. There are only seven banks of six coefficients which are unique and in the table the coefficients and delays are read from left to right starting at a delay of  $z^{-6}$ . The symmetry of coefficients can thus be read right to left (forwards) or left to



right (backwards) depending on the sample count modulo sixteen. This can be implemented using a RAM address selector, the most significant bits determining the bank and the least significant bits controlling the coefficient storage element. The remaining indices which have no elements stored in them are neglected. Thus a table lookup implemented in radix 2 can be used and is thus efficient in binary based VLSI.

This novel approach to coefficient storage reduction, which I formulated, has not been reported in the literature.

### 7.3 Synthesiser Structure

In this final section, the entire formant based additive synthesiser is presented. But first there will be an interlude section on the importance of phase in audio systems.

#### 7.3.1 The importance of Phase in Audio

The ear is phase sensitive at low frequencies ( $< 500$  Hz) as described in [93]. So the formant wavefunction presented in this thesis would be detectable to a minority of people as the shape is approximately the time reversed of the real glottal pulse shape. However, this perceived disadvantage can be made into an advantage, especially for creating new timbres.

Work done by Avery and Julius [142] created a framework of generating FIR (linear phase) filters using IIR filters and truncating the IIR filters' response. They achieved this by using a pole-zero cancellation scheme. To get linear phase they created time reversed truncated IIR filters which allowed magnitude squared filter design. However, these filters reduced the arithmetic requirements of an equivalent filter, but storage remained the same. Also the need to make unstable time-reversed filters stable was required. The purpose of this technique was to generate efficient FIR filters for audio analysis and equalisation problems. In these problems, the system designer does not want unnecessary coloration to occur during analysis or equalisation and thus linear phase is of paramount importance.

In synthesis of waveforms, as is the case in this thesis, phase is important for creating filters to generate particular impulse responses. However, these filters can not have linear phase because the wavefunctions generated are asymmetrical. Adding many formant regions together may require a structure which supports linear phase accumulation. In relation to FOF wavetable based synthesis, the skirt width parameter has a very minor use despite current thinking. The addition of many wavefunctions

---

together uses an initial phase fix which is very counter-intuitive and abstract in construction.

### 7.3.2 Parallel Formant Model

It is well known that instability is caused when filter coefficients are altered whilst the filter is still active. To remedy this problem, I implemented a minimum of 4 filter/oscillator combinations per FOF oscillator. Each of these combinations represents an equivalent FOF wavefunction, as found in CSOUND.

The full synthesiser structure is shown in figure 7.6 and uses two impulse generators (one creates pulses twice as fast as the other) and an adder cell (subtractor) to implement octavation. As in all formant based synthesisers, the filter-banks are added together in alternate signs in order to avoid nulls in the entire spectrum. Each filter has separate amplitude/gain controls, which are not shown in the diagram. Finally, the output needs to be stored in floating point, or it requires some sort of dynamic compression [88] to restrict the dynamic range. This mirrors the use of compressors for voice and audio to 'fit' onto analogue tapes.

---

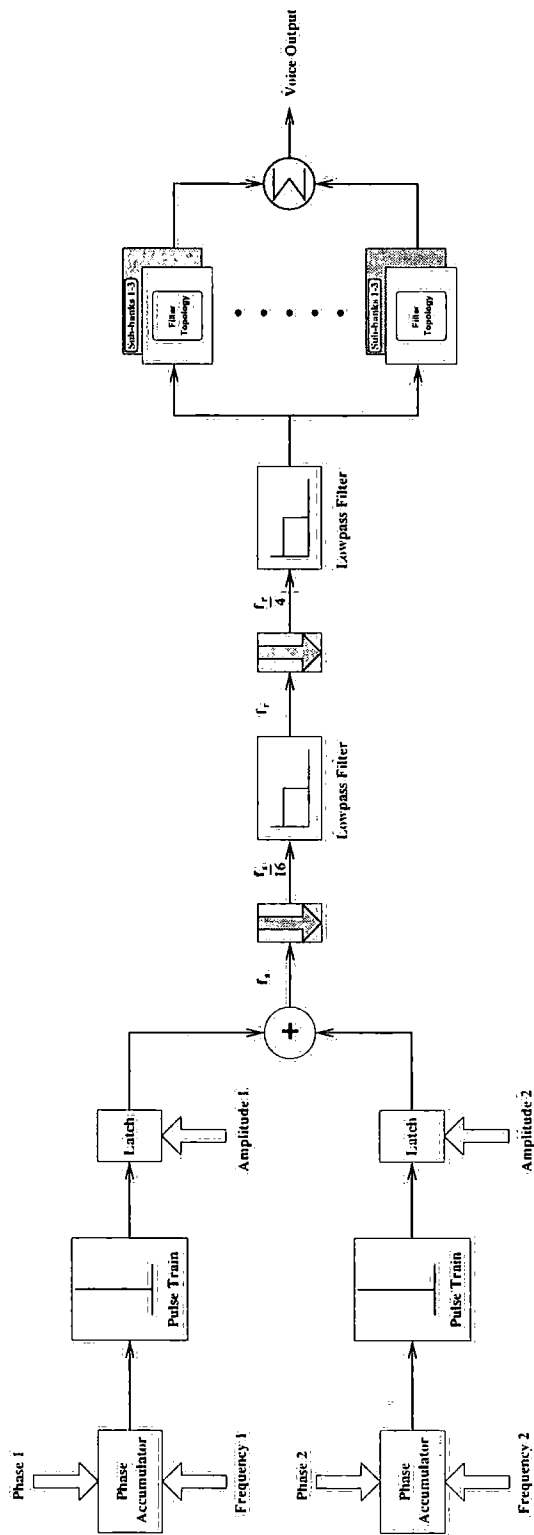


Figure 7.6: Block Diagram of  $t^2 e^{-\alpha t} \sin(\omega t)$  based formant synthesiser

My novel and elegant approach to octavation is ideally suited to VLSI and is here described. When implementing octavation the two impulse generators require the same frequency values but the phase values must differ by a factor of two. Over time the amplitude 2 value in figure 7.6 is increased in a two's complement sense. In this way the scaled impulse alternates between full value and progressively smaller values when the impulse streams coming out of the impulse generators coincide and diverge at the output of the adder. When the impulse train appears to be operating at half the original frequency, the process can be repeated by resetting the phase values to the new lower rate with the same factor of two relationship. If octavation is not required, amplitude 2 value is set to zero and phase 2 and frequency 2 values are ignored.

The two stage decimator as shown in figure 7.6 rejects every 16 samples. The output is sent through the first lowpass filter. The 4th sample of this data stream then passes through the second lowpass filter. In reality the decimator uses a polyphase filter design as described earlier, see figure 7.5. Using my reduced storage scheme based on the symmetry of the filter's response, the number of storage elements can be further reduced by another factor of two. The increase in control circuitry is only marginal, but I have produced a great reduction in memory storage requirements. My method saves memory usage without changing the latency of the polyphase filter structure.

Instead of updating the filter coefficients on a sample-by-sample basis, I chose to follow in the footsteps of Rodet [116] and freeze the coefficients on initialisation and disallow any attempt to alter them whilst the filter was computing the wavefunction. This approach requires a counter to signal the system when a wavefunction is complete. If another impulse arrives to the filter whilst it is generating a wavefunction, updating the coefficients must be halted.

The system thus requires multiple filter-banks so coefficient updates can occur and provide variation in timbre. However, too many per formant would be a burden on VLSI resources and too few would reduce the flexibility of the synthesis technique. I decided that four filter banks per formant was the minimum resource necessary both to keep VLSI costs down and to maximise the flexibility of the synthesis technique. This number was calculated from the number of overlaps required in CSOUND for a fundamental frequency (pitch) of around 440 Hz.

Each filter would be composed of three IIR filters in a Gold-Rader coupled form structure and a FIR filter combined together in series. The updating mechanism for a bank of four filters to describe one formant region will be discussed in the

---

next chapter. This structure, coupled with the update mechanism, has never been reported in the literature for a synthesis engine.

In speech synthesis, the number of controllable formants is five, so the minimum number of formants for this synthesis technique should be the same. However, an extra two would provide the musician with extra flexibility. The total number of filter-banks then required would then be 28. In a polyphonic voiced instrument each voice would then require 28 filter-banks all connected to a common impulse generator and decimator.

This approach wisely constrains the computational cost to a precise figure. If an algorithm requires increasing computation and storage requirements and operates in finite time, real-time audio synthesis would eventually become impossible. My approach solves this problem but unfortunately requires approximately six times the number of filters compared with old parallel formant synthesisers. Currently, VLSI design is nowhere near the level of integration required for this type of model based synthesis technique.

---

---

## CHAPTER 8

# Algorithm Scheduling and Miscellaneous Topics

---

In this chapter, I will discuss the “missing” sections of the proposed algorithm. This chapter consists of four subsections, the first being algorithm scheduling of the filterbanks. The middle section will deal with the parts of the algorithm which are simple to implement, such as ADSR amplitude envelopes. A description of the user interface design will follow and I will end with hardware/software segmentation.

### 8.1 Algorithm Scheduling

The  $t^2 \exp(-\alpha t) \sin(\omega_0 t)$  generator as shown earlier in this thesis can be composed of a cascade of 1st and 2nd order filter sections, whose coefficients control formant frequency and bandwidth of the particular formant. The work by Rodet [116] on CHANT mentioned that a filter based structure using 2nd order bandpass filters was constructed to operate on the Samson’s Box at CCRMA. This implementation requires the filter coefficients to be linearly interpolated to alter the bandwidth and formant frequency and may cause problems in filters due to instability in the poles of the structure. The approach I [128] took uses multiple filters driven by a common excitation source plus structures to control each filter’s execution. This approach fails when the excitation rate is large and the available filters are small <sup>1</sup>. The chosen solution is to stop the updating of the new coefficients and to utilise the autoregressive nature of the filters to perform overlapping waveforms automatically.

---

<sup>1</sup>This will in practice always be true.

On the grounds that the signal would already be evolving, this simple mechanism avoids the problem which plagues wavetable based FOF synthesis, by allowing operations to be performed in finite time and with constant computational resources. Two problems arise with this approach. The first being the increased computational overhead required; this is solved by implementing the algorithm in silicon. The second problem is the control of each filter's state. This then becomes a scheduling problem.

### 8.1.1 Controller Algorithm Overview

Each filterbank crudely represents one formant of a particular range of frequencies and bandwidths. This approach seems very expensive in terms of resources, but, provided the designer chooses the number of filters in each filterbank as a compromise between physical resources and musical freedom, it will result in a useable and implementable system. For the sake of brevity, say the number of sub filters in each filterbank is four. The algorithm as it stands has no information on how to control these filters.

Clearly, some form of memory is required to keep track of each sub-filter's status. The status flag indicates the filter's availability, being in one of two states; **Idle** and **Running**.

Initially all sub-filters are in the idle state, and as soon as one filter receives an excitation impulse from the impulse generator, the coefficients are loaded and the filter starts running. How do we know when the filter has decayed to zero, assuming no new excitation impulse arrives? The observant reader will remember that all the practical filter structures investigated were ones which had no truncation in the impulse response, and thus relied on the bandwidth parameter to decay to zero in three time constants. It is possible to track the data stream output from the filter and decide when it is zero even though the output is oscillatory. However, this approach adds more expensive arithmetical operations and a simpler and overestimating approach does exist which involves a decrementing counter. When the filter is triggered, the counter is initialised with a time value related to three time constants at the chosen sampling rate. Then every time a new sample period occurs, the counter is decremented by one. When the counter reaches zero, the sample is computed normally and the filter is zero initialised and the idle flag is set as shown in figure 8.1. Counters are easy to design in silicon and are extremely compact.

---

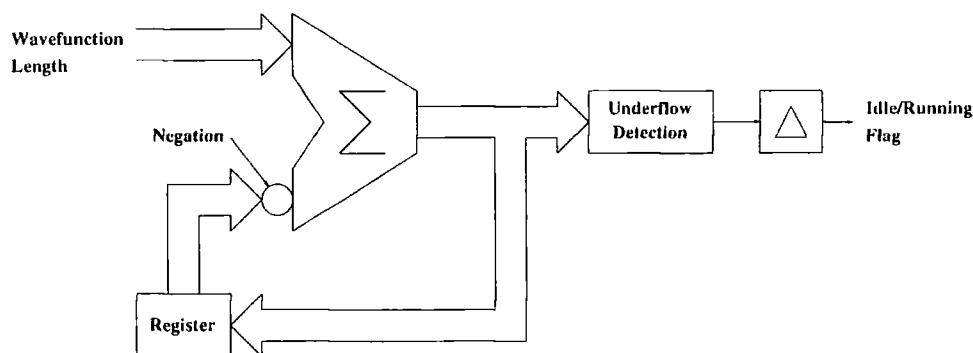


Figure 8.1: Block Diagram of the controller for idle/running status

### 8.1.2 Filterbank Scheduler Controller

Each filter's status can thus be expressed as a single bit being either zero (idle) or one (running). Thus for one filterbank a nibble of memory and four counters are required. It was initially thought that the nibble's bits could be made to perform a cyclic shift and each filter would be triggered by a new excitation, regardless of whether new coefficients were required. Thus the nibble represented the newest filter running. However, if the bandwidths of the filters vary widely, the linear sequence will become wasteful of resources because some filters will remain idle for a long time. Also this approach limits the choice of scheduling algorithm. A better approach is to use a linked list structure and a search method to find idle filters (resource slots).

Linked lists have been used in at least two musical applications. The first being a sound studio workstation for Lucasfilm [96] and more recently in a multirate additive synthesis engine [108]. The latter approach used linked lists to group sinusoids together, whilst the former allowed data of a transient nature to bypass a queuing system and be updated between samples.

In the present application, as the number of elements in the linked list is very small, the implementation requires deletion and insertion of elements at the head of the list. The list acts as an input queue but elements are flushed asynchronously by the decrementing counter. This is different from both the Lucasfilm Audio Processor and the Multirate Additive Synthesis Engine. This new approach allows the system always to know which filter element is most recently triggered by accessing the top element of the list. This removes the need to search the list. Normally linked lists are used when data of unknown length needs to be stored as occurs in random access floppy disks. In this application, there are a finite number of resource slots which cannot be exceeded. Hence the requirement of a decrementing counter which



is activated whenever a new filter needs to be initiated. However, if all filters are allocated, then this counter should trigger an interrupt to stop the filter being initiated. If this interrupt is triggered, the current filter's triggered counter is reset and any frequency and bandwidth parameters which have changed during the course of the musical note event is ignored.

A further refinement to the linked list algorithm would be to map the elements of the list to an index which redirects the DSP engine to an area of memory holding the filter parameters. Thus the list would appear more amenable to silicon as its functionality would be reduced and the elements would simply be indices. A simplified filterbank resource allocator is shown in figure 8.2.

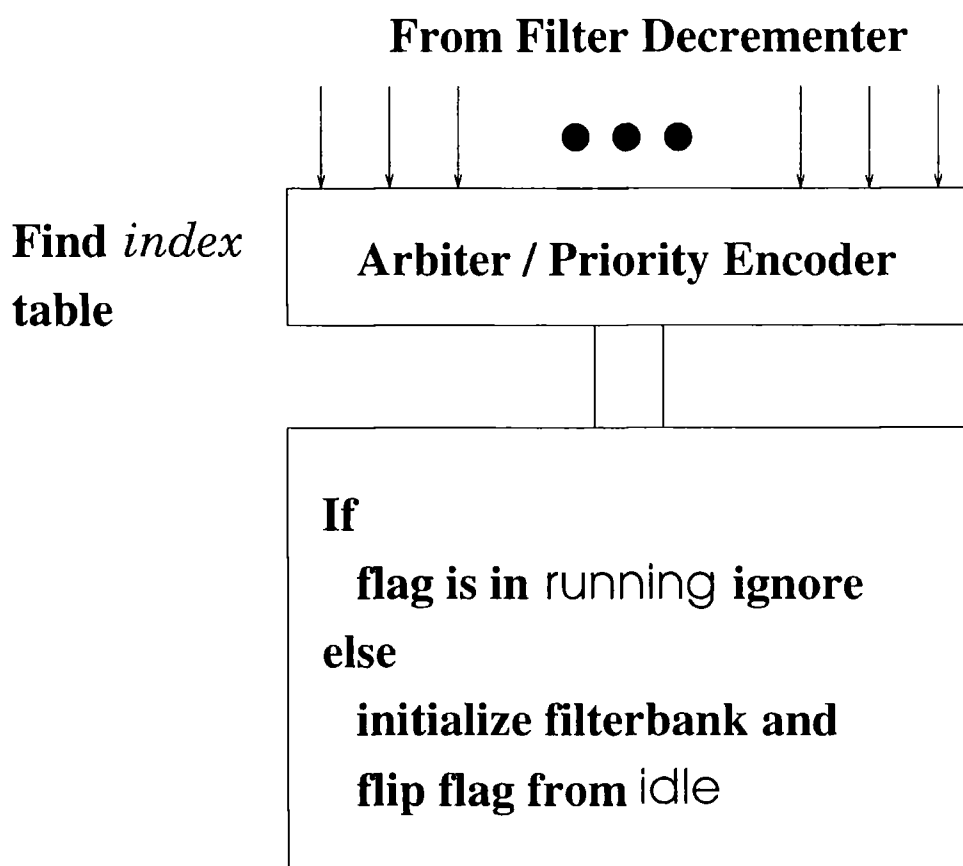


Figure 8.2: Block Diagram of Resource Filter Allocation

### 8.1.3 Number of Filters required

In the original CSOUND FOF unit generator, the subroutine prints error messages if the number of overlaps exceeds the number input in the score and orchestra files.

The number of overlaps is equal to

$$(\beta + \gamma) \cdot f_p$$

where  $\beta$  is the time at which the final decay starts,  $\gamma$  is the length of this final decay and  $f_p$  is the fundamental excitation frequency. Thus the equation represents the total grain lifetime in seconds multiplied by a rate, giving number of overlaps. Assuming the maximum excitation frequency of 7 kHz and  $\beta$  and  $\gamma$  set to 0.01 and 0.007 seconds respectively for singing voice emulation, the number of overlaps is 140. Most parameters are constant for a lot longer than the maximum excitation frequency. Setting the excitation frequency to equal old excitation frequency divided by parameter update rate, provides a worst case estimate for the number of overlaps and hence number of filters required. The value is found to be 14. A power of two for the number of filters is preferable and 8 was chosen as adequate because it provides parameter variation without causing excess overhead in filter construction.

## 8.2 Miscellaneous Topics

In this sub-section, the overlooked aspects of the synthesis method will be addressed.

The two simplest elements to implement are impulse jitter and formant gain. Formant gain is achieved by multiplication, whilst the jitter is overcome by operating the phase accumulator at high speed and downsampling the result.

### 8.2.1 Envelope Generation

Envelopes are time varying parameters and in musical synthesisers they are implemented as piece-wise linear line segments. The structure is composed around an adder, a multiplexor and additional logic arranged in such a way as to allow breakpoints to be performed [97]. This method has three inputs; current value, increment and final value. It has one output and the logic has four inputs (including the output) to control the mapping of the final value to the output or to the adder's output.

The logic required to implement bipolar envelopes is shown in table 8.1, where **X** is either zero or one. Also, 1 in all columns except the **Action** column represents a negative number and 0 represents a positive number. When **Action** is one, output is the **Final** value, otherwise the **New** output is selected. This can be implemented in silicon or on a general purpose DSP, after the truth table is converted into sum-of-products form. After computation, the current value input becomes the envelope output as shown in figure 8.3. The generator is also capable of threshold, maximum

Final - New	Current	Increment	Final	New	Action
0	0	0	0	0	0
0	0	0	0	1	1
1	0	0	0	X	1
X	0	0	1	X	1
0	0	1	0	X	1
1	0	1	0	0	0
1	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	1	1
1	0	1	1	X	0
0	1	0	0	X	0
1	1	0	0	0	1
1	1	0	0	1	0
0	1	0	1	0	1
0	1	0	1	1	0
1	1	0	1	X	1
X	1	1	0	X	1
0	1	1	1	X	1
1	1	1	1	0	1

Table 8.1: Decision Table for Bipolar Envelope Ramp Generator. (After [97])

and minimum functions. In this implementation, the envelope generation is of paramount importance, but updating the envelope in piece-wise linear fashion must be made by setting the increment value before the final breakpoint value. Using a Boolean truth table to logic minimisation process <sup>2</sup> the **Action** control line may be controlled via the following Boolean expression :-

$$A = \overline{F} \cdot (\overline{N} + \overline{C}) + C \cdot \overline{N} + \overline{I} \cdot F + I \cdot N$$

where the symbols are defined as follows :-

$$\begin{array}{lll} F = \text{Final} & N = \text{New} & C = \text{Current} \\ I = \text{Increment} & A = \text{Action} & \end{array}$$

8.2.2 Subtractive Synthesis Extensions

In this sub-section, I will briefly highlight the three most useful subtractive synthesis techniques; ADSR envelopes, digital filtering and low frequency oscillators, which all provide more acoustical movement to the sound.

---

<sup>2</sup>Using the Quine-McCluskey method.

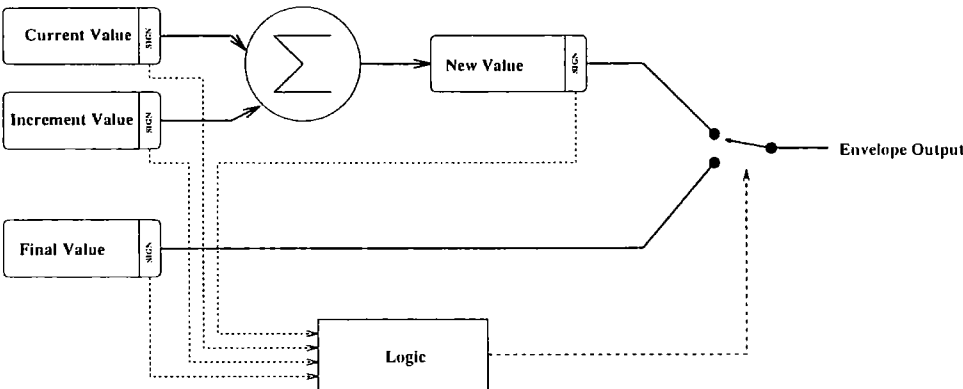


Figure 8.3: Schematic of a Bipolar Envelope Generator. (After [97])

The ADSR envelopes are directly related to the previous subsection and are an extension to the break-point envelope generator. This allows Attack, Decay, Sustain and Release (ADSR) style envelopes to control either the global amplitude or the individual formant amplitude. This envelope can be implemented as interrupts from the bipolar envelope generator to trigger new increment and final values which will be loaded into it.

Digital filtering of the sum of all the formant oscillators provides better emulation of the characteristics of the singing voice [6] by using a static filter with low cut-off frequency and a small bandwidth in order to impose frequency asymmetry to the formant oscillators. Dynamic effects can be imposed by driving a digital filter with an envelope or a keyboard scaling function. However, in this case, because the filter's parameters are time-varying, the designer needs a filter which provides independent coefficient modification. Therefore, varying frequency or bandwidth parameters does not affect the filter's stability. This type of filter is known as a state variable filter. A second order variant is shown in figure 2.3.

Low frequency oscillators provide periodic movement to timbres and can be implemented as a table lookup oscillator. They, like envelopes, can be used as a modulation source within any part of the synthesis technique, requiring extra additions and multiplications. Vibrato, for example, is the output amplitude being multiplied by an LFO, which is similar to AM radio.

### 8.2.3 Synthesis by Rule

In parametric formant additive synthesis systems, because the user is confronted with adding formants in parallel, certain computational rules need to be invoked.

The major rules concern the scaling factors of the filters and the methods used in order to ensure that dynamic range is preserved. This scaling can be thought of as a compander, that is a dynamic range compressor, linked to its inverse, the expander. This scaling technique is used naturally when there is too much dynamic range available for the chosen storage medium, as is used in record producing studios. However, in a studio situation this technique can be abused for creative gain, by making certain sounds more implosive.

Other rules concern performance criteria, especially the relation of formants relative to the perceived pitch. This happens in singing synthesis and is a frequency tracking technique. In natural (non synthesis) singing, the singer will use articulation to control tonal clarity and volume and therefore the harmonic structure of the sound will become purer as the pitch increases. These structures impart naturalness to the timbre. The singer can also impart chaos onto the tone producing vibrato effects. All these elements need to be either modelled by the synthesis technique or implanted into the synthesis model as a form of rules [6, 130], thus capturing the essence of the singing voice.

By providing a rule system, the composer can explore other musical timbres which have no natural analogue. In the next section in this chapter, the user interface aspects of the synthesis technique will be discussed.

### 8.3 User Interface Design

Any musical instrument to be used by a human musician needs to be ergonomically designed. The designer is faced with many conflicting parameters concerning on the one hand, implementation of the synthesis method, and on the other hand, making the instrument easy to use. The FOF synthesis technique used the LISP language to provide a hierarchical knowledge base to further enhance the sonic possibilities of CHANT which is known as FORMES [116]. However, in a realisable and commercial setting, if musicians are forced to learn another language in order to achieve greater flexibility; progress in using the synthesis system for generating music will be impeded. Musicians have spent a long time getting to know the intricacies of their chosen instrument and anything to provide rapid acceptance of a new instrument must be paramount in the minds of musical instrument manufacturers. No wonder the majority of electronic instruments utilise the MIDI specification, which was based on the Piano. It is also no surprise that implementing a keyboard is easy compared to translating vibrating strings into an electronic message as used in MIDI.

---

Therefore the designer must provide some tactile feedback from the instrument so that the musician can create music in real-time.

For this synthesis system, the design requires some “templates” which act as models for the particular instruments the musician might wish to emulate; e.g. violin, singing voices or drums. When Yamaha brought out their VL1, the first monophonic waveguide based synthesiser to implement a set of physical models of wind instruments, the template approach was used. However, maybe an instrument should have greater depths which would mean musicians could get to know the possibilities over time, be they of performance or acoustical value. This happened in the Synergy, an additive based 32 oscillator wave table synthesiser which provided a floating split feature, allowing two different tones to move across the keyboard rather than remaining static [63]. To date, no other instrument has provided this feature. Due to the rise of sampling, instruments have multiple split points which are statically assigned across the keyboard.

For the type of instrument under discussion, the designer should provide a simplistic synthesis model for quick editing of recognisable sounds and an advanced layer beneath this. The latter layer could be thought of as a language such as FORMES or CSOUND, providing flexibility should a musician wish to explore the synthesis method. The instrument would have a control surface providing a keyboard and knobs to interact using the template programming paradigm. If the musician wanted to use the advanced programming mode, he would have to program the instrument outside its physical entity by using the MIDI interface or other network system. It is well known that most musicians do not program synthesisers, they play them and use tones created by third parties. Hence the programming system should be thought of as an expansion possibility for which the manufacture could charge a premium. These programmable layers would be interfaced to the synthesis engine either by a direct link through shared memory to the engine or via a software layer which would implement extra functionality, e.g. LFOs and rule bases.

## 8.4 Hardware/Software Segmentation

This thesis has shown that the filter, impulse and envelope generators are suitable for implementation in silicon. The impulse and envelope generators are simple in terms of hardware and hence would use minimal resources on a die. Consequently they could be multiplexed and could be designed automatically to perform their job with a low data rate from the host. This information would be in the form of piecewise linear breakpoints (values) and rates (gradients). The filter itself is numerically

---

intensive and is best suited to hardware because of its regular high speed structure.

The polyphase decimator is a bit of a chimera, in that it could be implemented in hardware or software depending on the spare resources available on the silicon die.

The rest of the design, including controller, user interface and subtractive filtering would be implemented in software on a host computer. The filterbank controller, because of its non arithmetic components, could easily be designed to run on the host in software. The host would need to be reasonably fast in logic computation because of the filterbank controller's fast response. However, this would make a more expensive host to co-processor interface due to the fast data transfers across the bus.

This would provide the flexibility necessary for the interaction between man and machine.

---

---

## CHAPTER 9

# Conclusions and Further Work

---

### 9.1 Conclusions

This thesis has investigated the implementational issues of a novel formant synthesis technique loosely based on the FOF paradigm. Previous incarnations have been unsuitable for real-time implementation owing to the way the algorithm evolved under a software environment. In order to understand the difficulties involved in this research direction, a review of arithmetic algorithms and architectures was performed. In addition, the major synthesis algorithms were reviewed, noting that the nearest competitor is sinusoidal additive synthesis.

The current trend in the field is to implement algorithms in software, relying on chip manufacturers and processes to increase clock speeds and minimise transistor sizes on a die. This increases both the number of operations and also the speed of which a silicon chip can perform. This has led to the proliferation of sinusoidal additive synthesis techniques using inverse fast fourier transforms. Alternatively a multi-rate approach to additive synthesis can be used, borrowing heavily on current telecommunication theory using quadrature mirror filters and perfect reconstruction. However, to generate rich sound still requires an enormous amounts of data to describe each individual sinusoid in phase, frequency and amplitude at every instant.

In formant based synthesis, the source can be modelled by an idealistic pulse shape and the filter describes a decaying sinusoid. Instead of a single impulse at a specific height and frequency, as in sinusoidal synthesis, a peak was obtained in the frequency



domain, which has a spread of frequencies caused by the decaying sinusoid. Using this representation, it is possible to model many sounds with less parameters than in sinusoidal based additive synthesis. In most primitive parametric based additive synthesis engines using formants, the structure is either a cascade or a parallel arrangement of filters. It is well known that the human voice can be modelled accurately in this way.

The synthesis algorithm proposed extends the simplistic parallel formant structure to new heights and complexities, but keeps the parameters the same as in the simplistic case. After reviewing many filter topologies, the time squared decaying sinusoid filter was chosen for its fixed number of operations and delays over all bandwidths and frequencies. Because of the pole sensitivities caused by IIR filters, a Gold-Rader coupled form second order filter and a small FIR filter were used. Scaling of these filters was addressed and it was assumed that the controlling processor would select the optimum values.

The driving function was optimised for silicon by using a phase accumulator, and the single bit output controlled a latch having a variable amplitude or zero. This then passed through a polyphase decimator to preserve timing information at a lower rate. This is analogous to light passing through an aperture and spreading outwards in space. The decimator is symmetric, so a structure to reduce the physical coefficients by half was found. A single decimator could not be produced due to the large number of coefficients and so a two stage version was found to be optimal. The filters were then run all in parallel at a slow rate.

In order to control the filters, a system to monitor their progress was required. Counters were necessary to deallocate them after a certain length of time. This is very similar to note priority and note stealing algorithms in synthesisers, but the technique has been extended to a basic part of the synthesis itself. This added burden was necessary because once the coefficients were loaded into the filter they became frozen until the filter's response decayed to zero. This mirrors the wave-function mentality as found in FOF. When all filters are used up in a particular formant, the impulse driver excites the last filter and resets the counter until an available filter appears. The filter coefficients in this case stay the same.

The design of the technique requires the partitioning of DSP intensive operations from a controller. High level user interface requirements were addressed with the addition of low speed functions to dynamically vary the spectra. These should operate in the controller or in the main processor of the synthesis engine.

It had been hoped that the addition of parallel formant structures would avoid

---

spectral zeros. Methods to overcome this problem proved difficult. The simplest approach is to alternate the filterbanks' signs. Alternatively the output of the decimated impulse response could be delayed according to an index, so that each filter would have a slightly time-skewed driver signal. One factor which seemed to be negligible was the initial attack envelope of the waveform. This is the opposite of the work highlighted in [115, 116].

The key conclusions obtained were the reduction of bandwidth outside the DSP engine and the high bandwidth inside, thus allowing a reduction in parameter updates. The design chosen traded flexibility for realizability in order to provide a system suitable for VLSI implementation. This design models the large scale spectral properties of a sound rather than its fundamental components. This follows on logically from the fact that many sounds can be described as audio waves constrained in a box, resulting in formants. The only commercial synthesiser which uses formants to model vowel voices is the subtractive based sampler known as the E-mu Morpheus which has a 14 pole interpolating coefficient digital filter structure per voice, but it is not easily controlled.

A comparison between a bit-serial sinusoidal generator and a CORDIC parallel version found them to be of comparable size and speed. The CORDIC version takes 21 clock cycles to generate a sample representing the sine of an angle. The bit-serial IIR filter version takes 24 clock cycles for multiplication and an additional clock cycle for addition to complete a multiplication and add operation. Two sets of delay and arithmetic operators are performed in parallel, and so the IIR filter is marginally slower than the CORDIC variant.

My new theoretical approach to digital design, known as DAD Logic, uses an active feedback mechanism to map a continuum of voltages to discrete voltages. This replaces the hysteresis mechanism for binary digital circuits. The resulting scheme appears as a multi-voltage quantiser and allows standard analogue circuit design practice to be used, which reduces the number of transistors necessary in a design. Instead of coding multiplication over many digits and wires, as in digital design, the number of wires is dramatically reduced and the physical behaviour of the transistors are utilised to perform the multiplication. Thus the reduced number of wires and transistors allow higher speed with lower power consumption. To the outside world, the chip appears to be entirely digital, in that it only accepts binary digital input.

---

## 9.2 Further Work

Further work can be directed in a number of directions. As it stands, there have been no direct implementations of the system in silicon or software. In order to make the technology useful ergonomic constraints would have to be deployed. This would result in an intuitive user interface which musicians could use with minimal training.

An interesting line of attack would be to look at the discrete summation formulas [95, 129] for formant generation and investigate the possibilities of implementation of a FOF wavefunction as a square root filter. The summing of formant regions and the normalisation factors necessary to obtain a perfect multi-formant model of the sound also needs investigation. Hopefully this could also be applied to filter implementation. In theory and in practice filters in cascade form of a particular time domain model performed better than the equivalent parallel form, even when floating point arithmetic was used. Does the parallel form require manual fine tuning to overcome numerical imprecision ?

Rapid prototyping of algorithm to ASIC using high-level languages such as Silage and CAD design tools such as Hyper and CATHEDRAL need to be used to a greater degree. Also high-level simulation of algorithms in Matlab and/or Ptolemy are essential, as coding of algorithms is very time consuming. The need for some high level optimisation process like Javalina [41] to symbolically analyse and control filter topologies would enable designers to investigate their performance and to reduce the number of multipliers necessary. Noise and sensitivity requirements could also be controlled. To my knowledge HYPER and IRIS do not allow the topology represented by a signal flow diagram to be rearranged.

In order to build this synthesis system, ready built Intellectual Property components would be required. Combining these components with VHDL would reduce design time significantly. With such methods and using a system design paradigm algorithms which had been created in software could be transferred to ASIC implementation very easily.

In chapter 4 the need for different low level technologies and architectures was investigated. This is especially important with portable applications requiring low power consumption. If signals on wires carried more than a binary representation, then designers could make the arithmetic elements more versatile, and the space savings could be utilised to provide more functional units. Low power could be achieved by using parallel execution units whilst still operating at modest clock rates. Asynchronous design removes the global clock and replaces it with handshakes controlling

---

dataflow locally. It can operate as fast as clocked logic and was erroneously thought to have a low power benefit. DAD Logic, which is inherently analogue based, would automatically operate in an asynchronous manner should the need arise.

---

---

## CHAPTER A

# The Laplace to Z Transform Algorithm

---

This algorithm uses the property that the product of two signals is the *convolution* in the s domain (Laplace).

The ideal sampling function,  $\delta_{T_s}(t)$ , is represented as a train of unit amplitude impulses and is expressed as follows :-

$$\delta_{T_s}(t) = \sum_{n=0}^{\infty} \delta(t - nT_s)$$

Taking the Laplace transform of  $\delta_{T_s}$  is

$$\mathcal{L}[\delta_{T_s}(t)] = \Delta_{T_s}(p) = \sum_{n=0}^{\infty} e^{-nT_s p}$$

The result above when expressed in closed form becomes

$$\Delta_{T_s}(s - p) = \frac{1}{1 - e^{-(s-p)T_s}} \quad (\text{A.1})$$

Let the Laplace transform of the input signal,  $x(t)$ , be  $\mathcal{X}(p)$  and computing the Laplace transform of the product of  $x(t)$  and the sampling function,  $\delta_{T_s}(t)$ , the following expression results :-

$$\begin{aligned}\mathcal{L}[x(t) \cdot \delta_{T_s}(t)] &= \mathcal{Y}(s) \\ &= \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} \mathcal{X}(p) \Delta_{T_s}(s-p) dp \quad (\text{A.2})\end{aligned}$$

Substituting equation A.1 into equation A.2 results in the following expression :-

$$\mathcal{Y}(s) = \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} \mathcal{X}(p) \frac{1}{1 - e^{-(s-p)T_s}} dp \quad (\text{A.3})$$

Now, from the z transform,  $e^{-sT_s} = z^{-1}$ , so equation A.3 can be rewritten to equation A.4 below. Note that any Laplace transform of  $x(t)$  which has  $e^{-\alpha s T_s}$  can be substituted for  $z^{-\alpha}$ .

$$\mathcal{Y}(z) = \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} \frac{\mathcal{X}(p)}{1 - e^{pT_s} z^{-1}} dp \quad (\text{A.4})$$

The integral in equation A.4 can be evaluated using Cauchy's residue theorem. The theorem states that if  $\mathcal{P}(s)$  is analytic within and on a closed contour  $\mathcal{C}$ , except possibly at a finite number of singularities within  $\mathcal{C}$ , then

$$\frac{1}{2\pi j} \oint_{\mathcal{C}} \mathcal{P}(s) ds = \sum_r k_r$$

where  $k_r$  are the residues of  $\mathcal{P}(s)$  at the singularities.

Applying the theorem yields the following algorithm A.5.

$$\mathcal{Y}(z) = \sum_{\text{poles of } \mathcal{X}(p)} \text{residues of } \left[ \frac{\mathcal{X}(p)}{1 - e^{pT_s} z^{-1}} \right] \quad (\text{A.5})$$

The residue for a pole of order  $m$  at  $p = x$  can be evaluated using the following expression

$$\frac{1}{(m-1)!} \lim_{p \rightarrow x} \left\{ \frac{d^{m-1}}{dp^{m-1}} \left[ (p-x)^m \mathcal{X}(p) \frac{z}{z - e^{pT_s}} \right] \right\}$$

---

## References

---

- [1] Abo, A. and Mehta, S. "CMOS Current Mode Adders." May 1993. From URL [kabuki.eecs.berkeley.edu/~abo/](http://kabuki.eecs.berkeley.edu/~abo/).
- [2] Adams, R. and Kwan, T. "A Stereo Asynchronous Digital Sample-Rate Converter for Digital Audio." *IEEE Journal of Solid-State Circuits*, 29(4), pp. 481–488, April 1994.
- [3] Al-lbrahim, M. and Al-Khateeb, A. "Digital Sinusoidal Oscillator with Low and Uniform Frequency Spacing." *IEE Proceedings - Circuits, Devices and Systems*, 144(3), pp. 185–189, June 1997.
- [4] Avizienis, A. "Signed-Digit Number Representations for Fast Parallel Arithmetic." *IEEE Transactions on Electronic Computers*, EC-10, pp. 389–400, September 1961.
- [5] Bajard, J.-C., Kla, S. and Muller, J.-M. "BKM : A new Hardware Algorithm for Complex Elementary Functions." *IEEE Transactions on Computers*, 43(8), pp. 955–963, August 1994.
- [6] Bennett, G. and Rodet, X. "Synthesis of the Singing Voice." In Mathews, M. and Pierce, J., eds., *Current Directions in Computer Music Research*, pp. 19–44. MIT Press, Cambridge, Massachusetts, 1989.
- [7] Brennan, J. "Multilevel ASICs Boost Audio Recording Applications." *IEEE Circuits and Systems Magazine*, pp. 18–21, May 1996.
- [8] Capasso, F., Sen, S., Beltram, F., Lunardi, L., Vengunlekar, A., Smith, P., Shah, N., Malik, R. and Cho, A. "Quantum Functional Devices : Resonant Tunneling Transistors, Circuits with Reduced Complexity, and Multiple-Valued Logic." *IEEE Transactions on Electron Devices*, 36(10), pp. 2065–2082, October 1989.

- 
- [9] Carter, G. and Kron, G. "A.C. Network Analyzer Study of the Schrödinger Equation." *Physical Review*, 67(1/2), pp. 44-49, Jan 1/15 1945.
  - [10] Chamberlin, H. *Musical Applications of Microprocessors*. Hatden Books, A division of Howard W. Sams & Company, USA, second edition, 1985. ISBN 0-8104-5768-7.
  - [11] Chowning, J. M. "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation." In Roads, C. and Strawn, J., eds., *Foundations of Computer Music*, pp. 1-29. MIT Press, Cambridge, Massachusetts, 1985. Reprinted from the *Journal of the Audio Engineering Society* 21(7), 1973.
  - [12] Chua, L. O. and Deng, A.-C. "Negative Resistance Devices: Part II." *International Journal of Circuit Theory and Applications*, 12(4), pp. 337-373, 1984.
  - [13] Chua, L. O., Yu, J. and Yu, Y. "Negative Resistance Devices." *International Journal of Circuit Theory and Applications*, 11(2), pp. 161-186, 1983.
  - [14] Chua, L. O., Yu, J. and Yu, Y. "Bipolar-JFET-MOSFET Negative Resistance Devices." *IEEE Transactions on circuits and Systems*, CAS-32(1), pp. 46-61, January 1985.
  - [15] Clarke, J., Manning, P., Berry, R. and Purvis, A. "VOCEL : New Implementations of the FOF Synthesis Method." In *Proceedings of the International Computers Music Conference*, pp. 333-348. Cologne, W-Germany, 1988.
  - [16] Crochiere, R. E. and Oppenheim, A. V. "Analysis of Linear Digital Networks." *Proceedings of the IEEE*, 63(4), pp. 581-595, April 1975.
  - [17] Current, K. "Current-Mode CMOS Multiple-Valued Logic Circuits." *IEEE Journal of Solid-State Circuits*, 29(2), pp. 95-107, February 1994.
  - [18] Daggett, D. "Decimal-binary conversions in CORDIC." *IRE Transactions On Electronic Computers*, EC-8(3), pp. 335-339, September 1959.
  - [19] Dattorro, J. "The Implementation of Recursive Digital Filters for High-Fidelity Audio." *Journal of the Audio Engineering Society*, 36(11), pp. 851-878, November 1988.
  - [20] Denyer, P. and Renshaw, D. *VLSI Signal Processing: a Bit-Serial Approach*. Addison-Wesley Publishing Company, 1985.
-



- [21] Depalle, P., Matignon, D. and Stroppa, M. "Source-filter formulation and analytic control of the skirtwidth of CHANT formant-wave-functions." In *Proceedings of the International Computer Music Conference*, pp. 372-373. San Jose, California, USA, October 1992.
  - [22] Despain, A., Peterson, A., Rothaus, O. and Wold, E. "Fast fourier transform processors using gaussian residue arithmetic." *Journal of Parallel And Distributed Computing*, 2(3), pp. 219-237, 1985.
  - [23] Ding, Y. and Rossum, D. "Filter Morphing of Parametric Equalizers and Shelving Filters for Audio Signal Processing." *Journal of the Audio Engineering Society*, 43(10), pp. 821-826, October 1995.
  - [24] Edmonds, A. *Angular Momentum in Quantum Mechanics*. Princeton Landmarks in Mathematics and Physics series. Princeton University Press, 2nd Edition, 4th printing, 1996.
  - [25] Elleithy, K. and Bayoumi, M. "Fast and Flexible Architectures for RNS Arithmetic Decoding." *IEEE Transactions on Circuits and Systems - II : Analog and Digital Signal Processing*, 39(4), pp. 226-235, April 1992.
  - [26] Ercegovac, M. D. and Lang, T. "On-the-Fly Conversion of Redundant into Conventional Representations." *IEEE Transactions on Computers*, C-36(7), pp. 895-897, July 1987.
  - [27] Ercegovac, M. D. and Lang, T. "Fast Multiplication Without Carry-Propagate Addition." *IEEE Transactions on Computers*, 39(11), pp. 1385-1390, November 1990.
  - [28] ES2. *ES2 Cadence Design Kit User Guide*. European Silicon Structures, France, May 1994.
  - [29] Etzel, M. and Jenkins, W. "The design of specialized residue classes for efficient recursive digital filter realization." *IEEE Transactions On Acoustics, Speech And Signal Processing*, ASSP-30(3), pp. 370-380, 1982.
  - [30] F.J.Taylor and Ramnarayanan, A. "An efficient residue-to-decimal converter." *IEEE Transactions On Circuits And Systems*, CAS-28(12), pp. 1164-1169, 1981.
  - [31] Fliege, N. and Wintermantel, J. "Complex Digital Oscillators and FSK Modulators." *IEEE Transactions on Signal Processing*, 40(2), pp. 333-342, February 1992.
-

- 
- [32] Fliege, N. J. *Multirate Digital Signal Processing : Multirate Systems, Filter Banks, Wavelets*. John Wiley & Sons, 1994. ISBN 0-471-93976-5.
- [33] Fredkin, E. and Toffoli, T. "Conservative logic." *International Journal of Theoretical Physics*, 21(3/4), pp. 219–253, 1982.
- [34] Freed, A., Rodet, X. and Depalle, P. "Synthesis and Control of Hundreds of Sinusoidal Partial on a Desktop Computer without Custom Hardware." In *Proceedings of the International Computer Music Conference*, pp. 98–101. Tokyo, Japan, September 10-15 1993.
- [35] Gabor, D. "Theory of Communication." *Journal of the Institute of Electrical Engineers*, 93(Part III), pp. 429–457, 1946.
- [36] Garner, H. "The residue number system." *IRE Transactions On Electronic Computers*, EC-8(6), pp. 140–147, 1959.
- [37] Gilbert, B. "A Precise Four-Quadrant Multiplier with Subnanosecond Response." *IEEE Journal of Solid-State Circuits*, SC-3(4), pp. 365–373, December 1968.
- [38] Guillemain, P. and Kronland-Martinet, R. "Characterization of acoustic signals through continuous linear time-frequency representations." *Proceedings of the IEEE*, 84(4), pp. 561–585, 1996.
- [39] Hatano, Y., Yano, S., Mori, H., Yamada, H. and Hirano, M. "A 1-GIPS Josephson Data Processor." *IEEE Journal of Solid-State Circuits*, 26(6), pp. 880–883, June 1991.
- [40] Haviland, G. and Tuszynski, A. "CORDIC Arithmetic Processor Chip." *IEEE Transactions on Computers*, C-29(2), pp. 68–79, February 1980.
- [41] Hebel, K. J. "Javelina: An Environment for Digital Signal Processing Software Development." *Computer Music Journal*, 13(2), pp. 39–47, Summer 1989.
- [42] Hiasat, A. "New designs for a sign detector and a residue binary convertor." *IEE Proceedings-G Circuits Devices And Systems*, 140(4), pp. 247–252, 1993.
- [43] Higgins, R. *Digital Signal Processing in VLSI*. Prentice-Hall Inc., 1990.
- [44] Honda, M., Kameyama, M. and Higuchi, T. "Residue arithmetic based multiple-valued VLSI image processor." In *IEEE 22nd International Symposium On Multiple-Valued Logic*, pp. 330–336. IEEE Press, 1992.
-

- 
- [45] Horner, A. "A comparison of Wavetable and FM Parameter Spaces." *Computer Music Journal*, 4(21), pp. 55-85, Winter 1997.
  - [46] Horner, A. and Beauchamp, J. "Piecewise-Linear Approximation of Additive Synthesis Envelopes: A Comparison of Various Methods." *Computer Music Journal*, 20(2), pp. 72-95, Summer 1996.
  - [47] Houghton, A. D., Fisher, A. J. and Malet, T. F. "An ASIC for Digital Additive Sine-wave Synthesis." *Computer Music Journal*, 19(3), pp. 26-31, Fall 1995.
  - [48] Hu, X., Harber, R. G. and Bass, S. C. "Expanding the Range of Convergence of the CORDIC Algorithm." *IEEE Transactions on Computers*, 40(1), pp. 13-21, January 1991.
  - [49] Hu, Y. "CORDIC-Based VLSI Architectures for Digital Signal Processing." *IEEE Signal Processing Magazine*, pp. 16-35, July 1992.
  - [50] Hu, Y. H. "The Quantization Effects of the CORDIC Algorithm." *IEEE Transactions on Signal Processing*, 40(4), pp. 834-844, April 1992.
  - [51] Huang, X., Liu, W.-J. and Wei, B. W. "A High-Performance CMOS Redundant Binary Multiplication-and-Accumulation ( MAC ) Unit." *IEEE Transactions On Circuits And Systems - 1: Fundamental Theory And Applications*, 41(1), pp. 33-39, January 1994.
  - [52] Hwang, K. *Computer Arithmetic - Principles, Architecture And Design*. John Wiley & Sons, Inc., 1979.
  - [53] Ifeachor, E. and Jervis, B. *Digital Signal Processing : A Practical Approach*. Addison-Wesley Publishing Company, 1993.
  - [54] Itagaki, T., Purvis, A. and Manning, P. "Real-time Synthesis on a Multi-processor Network." In *Proceedings of the International Computer Music Conference*, pp. 382-385. Aarhus, Denmark, September 1994.
  - [55] Jain, A., Bolton, R. and Abd-El-Barr, M. "CMOS Multiple-Valued Logic Design - part I : Circuit Implementation." *IEEE Transactions on Circuits and Systems - I : Fundamental Theory and Applications*, 40(8), pp. 503-514, August 1993.
  - [56] Jain, A., Bolton, R. and Abd-El-Barr, M. "CMOS Multiple-Valued Logic Design - part II : Function Realization." *IEEE Transactions on Circuits and Systems - I : Fundamental Theory and Applications*, 40(8), pp. 515-523, August 1993.
-

- 
- [57] James, M., Smith, G. and Wolford, J. *Analog and Digital Computer Methods IN ENGINEERING ANALYSIS*. International Textbook Company, Scranton, Pennsylvania, 1964.
- [58] Jansen, C. "Sine Circuitu : 10,000 high quality sine waves without detours." In *Proceedings of the International Computer Music Conference*, pp. 222–225. Montreal, Canada, 1991.
- [59] Jenkins, W. and Leon, B. "The use of residue number systems in the design of finite impulse response digital filters." *IEEE Transactions On Circuits And Systems*, CAS-24(4), pp. 191–201, 1977.
- [60] Johns, D. and Lewis, D. "IIR Filtering on Sigma-Delta Modulated Signals." *Electronics Letters*, 27(4), pp. 307–308, 14th February 1991.
- [61] Kahrs, M. "Notes on Very-Large-Scale Integration and the Design of Real-Time Digital Sound Processors." In Roads, C., ed., *The Music Machine : Selected Readings from Computer Music Journal*, pp. 623–631. MIT Press, Cambridge, Massachusetts, 1989. Reprinted from *Computer Music Journal*, vol. 5, No. 2, Summer 1981.
- [62] Kameyama, M., Kawahito, S. and Higuchi, T. "A multiplier chip with multiple-valued bidirectional current-mode logic circuits." *IEEE Computer Magazine*, 21, pp. 43–56, April 1988.
- [63] Kaplan, S. J. "Developing a Commercial Digital Sound Synthesizer." In Roads, C., ed., *The Music Machine : Selected Readings from Computer Music Journal*, pp. 611–622. MIT Press, Cambridge, Massachusetts, 1989. Reprinted from *Computer Music Journal*, vol. 5, No. 3, Fall 1981.
- [64] Karplus, W. J. *Analog Simulation - SOLUTION OF FIELD PROBLEMS*. McGraw-Hill Series in Information Processing and Computers. McGraw-Hill Company, Inc., 1958.
- [65] Karplus, W. J. and Soroka, W. W. *ANALOG METHODS - Computation and Simulation*. McGraw-Hill Series in Engineering Sciences. McGraw-Hill Company, Inc., 2nd edition, 1959.
- [66] Kawahito, S., Kameyama, M., Higuchi, T. and Yamada, H. "A high speed compact multiplier based on multiple-valued bi-directional current-mode circuits." In *IEEE 17Th International Symposium On Multiple-Valued Logic*, pp. 172–180. IEEE Press, 1987.
-

- 
- [67] Kingsbury, N. "Second-Order Recursive Digital Filter Element for Poles near the Unit Circle and the Real Z Axis." *Electronics Letters*, 8(6), pp. 155–156, 23rd March 1972.
- [68] Kingsbury, N. "Digital-filter 2nd-Order Element with Low Quantising Noise for Poles and Zeros at Low Frequencies." *Electronics Letters*, 9(12), pp. 271–273, 14th June 1973.
- [69] Klatt, D. "Software for a cascade/parallel formant synthesizer." *Journal of the Acoustic Society of America*, 67(3), pp. 971–995, March 1980.
- [70] Kron, G. "Electric Circuit Models of the Schrödinger Equation." *Physical Review*, 67(1/2), pp. 39–43, Jan 1/15 1945.
- [71] Kuroda, T., Suzuki, K. et al. "Variable Supply-Voltage Scheme for Low-Power High-Speed CMOS Digital Design." *IEEE Journal of Solid-State Circuits*, 33(3), pp. 454–462, March 1998.
- [72] Laakso, T. I., Välimäki, V., Karjalainen, M. and Laine, U. K. "Splitting the Unit Delay - Tools for fractional delay filter design." *IEEE Signal Processing Magazine*, 13(1), pp. 30–60, January 1996.
- [73] Lenzlinger, M. and Snow, E. "Fowler-Nordheim Tunneling into Thermally Grown SiO<sub>2</sub>." *Journal of Applied Physics*, 40(1), pp. 278–283, January 1969.
- [74] Levy, H. J. and McGill, T. "A Feedforward Artificial Neural Network Based on Quantum Effect Vector-Matrix Multipliers." *IEEE Transactions on Neural Networks*, 4(3), pp. 427–433, May 1993.
- [75] Li, Y., Eichmann, G., Dorsinville, R. and Alfano, R. "Demonstration of a picosecond optical-phase-conjugation-based residue-arithmetic computation." *Optics Letters*, 13(2), pp. 178–180, 1988.
- [76] Linggard, R. *Electronic Synthesis of Speech*. Cambridge University Press, 1985.
- [77] Lu, A. and Roberts, G. "An Analog Multi-Tone Signal Generator for Built-In-Self-Test Applications." In *The International Test Conference*, pp. 650–659. 1994.
- [78] Lu, A. K., Roberts, G. W. and Johns, D. A. "A High-Quality Analog Oscillator Using Oversampling D/A Conversion Techniques." *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing*, 41(7), pp. 437–444, July 1994.
-

- 
- [79] Lyon, R. "Two's Complement Pipeline Multipliers." *IEEE Transactions on Communications*, COM-24, pp. 418-425, April 1976.
- [80] Lyon, R. F. "Filters : An Integrated Digital Filter Subsystem." In Denyer, P. and Renshaw, D., eds., *VLSI Signal Processing: a Bit-Serial Approach*, pp. 253-262. Addison-Wesley Publishing Company, 1985.
- [81] Madisetti, V. K. *Digital Signal Processors AN INTRODUCTION TO RAPID PROTOTYPING AND DESIGN SYNTHESIS*. Butterworth-Heinemann, Boston, 1995. ISBN 0-7506-9406-8.
- [82] Manning, P. *Electronic and Computer Music*. Clarendon Press, Oxford, 1985.
- [83] Marks, P. "Making waves." *New Scientist*, 157(2124), pp. 34-37, 7th March 1998.
- [84] Massie, D. C. "An Engineering Study of the Four-Multiply Normalized Ladder Filter." *Jorunal of the Audio Engineering Society*, 41(7/8), pp. 564-582, July 1993.
- [85] Matthews, R. "Take a spin ..." *New Scientist Magazine*, 157(2123), pp. 24-28, 28th February 1998.
- [86] Mauchly, J. and Charpentier, A. "Practical Considerations in the Design of Music Systems using VLSI." In *The Proceedings of the AES 5th International Conference on Music and Digital Technology*, pp. 28-36. The Audio Engineering Society, May 1987.
- [87] McNally, G. "Digital Audio : Recursive Digital Filtering for High Quality Audio Signals." Report No. 1981/10, The BBC Research Department, 1981.
- [88] McNally, G. "Dynamic Range Control of Digital Audio Signals." *Journal of the Audio Engineering Society*, 32(5), pp. 316-327, May 1994.
- [89] Mead, C. *Analog VLSI And Neural Systems*. Addison-Wesley Publishing Company, 1989.
- [90] Mirsalehi, M., Shamir, J. and Caulfield, H. "Residue arithmetic processing utilizing optical fredkin gate arrays." *Applied Optics*, 28(18), pp. 3940-3946, 1987.
- [91] Mohan, S., Mazumder, P. and Haddad, G. "Subnanosecond 32 bit multiplier using negative differential resistance devices." *Electronics Letters*, 27(21), pp. 1929-1931, 1991.
-

- 
- [92] Mohan, S., Mazumder, P., Haddad, G., Mains, R. and Sun, J. "Logic design based on negative differential resistance characteristics of quantum electronic devices." *IEE Proceedings-G*, 140(6), pp. 383-391, 1993.
- [93] Moore, B. C. *An Introduction to the Psychology of Hearing*. Academic Press Ltd., London, second edition, 1991. ISBN 0-12-505624-9.
- [94] Moore, F. "Table Lookup Noise for Sinusoidal Digital Oscillators." In Roads, C. and Strawn, J., eds., *Foundations of Computer Music*, pp. 326-334. MIT Press, Cambridge, Massachusetts, 1985. Appeared in *Computer Music Journal* 1(2):26-29, 1977.
- [95] Moorer, J. A. "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulas." *Journal of the Audio Engineering Society*, 24(9), pp. 717-727, November 1976.
- [96] Moorer, J. A. "The Lucasfilm Audio Signal Processor." In Roads, C., ed., *The Music Machine : Selected Readings from Computer Music Journal*, pp. 599-609. MIT Press, Cambridge, Massachusetts, 1989. Reprinted from *Computer Music Journal*, vol. 6, No. 3, Fall 1982.
- [97] Moorer, J. A., Chauveau, A., Roads, C., Eastty, P. and Lawson, J. "The 4C Machine." In Roads, C. and Strawn, J., eds., *Foundations of Computer Music*, pp. 261-280. MIT Press, Cambridge, Massachusetts, 1991. Reprinted from *Computer Music Journal*, vol. 3, No. 3, 1979, pg 16-24.
- [98] Moshinsky, M. and Smirnov, Y. *The Harmonic Oscillator in Modern Physics*. Harwood Academic Publishers, 1996.
- [99] Mullis, C. T. and Roberts, R. A. "Synthesis of Minimum Roundoff Noise Fixed Point Digital Filters." *IEEE Transactions on circuits and systems*, CAS-23(9), pp. 551-562, September 1976.
- [100] Nielsen, R. O. and Willson, JR, A. N. "A Fundamental Result Concerning the Topology of Transistor Circuits with Multiple Equilibria." *Proceedings of the IEEE*, 68(2), pp. 196-208, February 1980.
- [101] Oates, S. and Eaglestone, B. "Analytical Methods for Group Additive Synthesis." *Computer Music Journal*, 21(2), pp. 21-39, Summer 1997.
- [102] Oppenheim, A. V. and Schaffer, R. W. *Digital Signal Processing*. Prentice/Hall International, Inc., 1975.
-

- 
- [103] Phillips, D. November 1995. Matlab FOF script and E-Mail correspondence.
- [104] Phillips, D., Purvis, A. and Johnson, S. "A Multirate Optimisation for Real-Time Additive Synthesis." In *Proceedings of the International Computer Music Conference*, pp. 364–367. Aarhus, Denmark, September 1994.
- [105] Phillips, D., Purvis, A. and Johnson, S. "An Efficient Algorithm and Architecture for Real-Time Additive Synthesis of Musical Tones." In *Euromicro Conference*, pp. 1–8. Como, Italy, September 1995.
- [106] Phillips, D., Purvis, A. and Johnson, S. "Multirate Additive Synthesis." In *Proceedings of the International Computer Music Conference*, pp. 496–499. Hong Kong, August 1996.
- [107] Phillips, D., Purvis, A. and Johnson, S. "On an efficient VLSI Architecture for the multirate additive synthesis of musical tones." *Journal of Systems Architecture*, 43(1-5), pp. 337–340, 1997.
- [108] Phillips, D. K. *Algorithms and Architectures for the Multirate Additive Synthesis of Musical Tones*. Ph.D. thesis, School of Engineering, University of Durham, December 1996.
- [109] Prieto, A., Pelayo, F. and Lloris, A. "Multithreshold logic circuits implemented with operational amplifiers." *International Journal of Electronics*, 58(3), pp. 395–406, 1985.
- [110] Primlani, K. K. and Meador, J. L. "A Nonredundant-Radix-4 Serial Multiplier." *IEEE Journal of Solid-State Circuits*, 24(6), pp. 1729–1736, December 1989.
- [111] Pryce, J. D. *Numerical Solution of Sturm-Liouville Problems*. Monographs on Numerical Analysis. Clarendon Press, 1993.
- [112] Puckette, M. "Process and device for musical and vocal dynamic sound synthesis by non-linear distortion and amplitude modulation." June 4th 1996. USA Patent No. 5,524,173.
- [113] Rader, C. and Gold, B. "Effects of Parametric Quantization on the Poles of a Digital Filter." *Proceedings of the IEEE*, pp. 688–689, May 1967.
- [114] Roads, C., Strawn, J., Abbott, C., Gordon, J. and Greenspun, P. *The Computer Music Tutorial*. MIT Press, Cambridge, Massachusetts, 1996.
-



- 
- [115] Rodet, X. "Time Domain formant-wave-function Synthesis." *Computer Music Journal*, 8(3), pp. 9–14, 1984.
- [116] Rodet, X., Potard, Y. and Barrière, J.-B. "The CHANT Project : From the Synthesis of the singing voice to synthesis in general." *Computer Music Journal*, 8(3), pp. 15–31, 1984.
- [117] Roza, E. "Recursive Bitstream Conversion: the Reverse Mode." *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing*, 41(5), pp. 329–336, May 1994.
- [118] Sandell, G. J. and Martens, W. L. "Perceptual Evaluation of Principal-Component-Based Synthesis of Musical Timbres." *Journal of the Audio Engineering Society*, 43(12), pp. 1013–1028, December 1995.
- [119] Sen, S., Capasso, F., Cho, A. and Sivco, D. "Resonant Tunneling Device with multiple Negative Differential Resistance : Digital and Signal Processing Applications with Reduced Circuit Complexity." *IEEE Transactions on Electron Devices*, ED-34(10), pp. 2185–2191, October 1987.
- [120] Shibata, T. and Ohmi, T. "A functional MOS Transistor Featuring Gate-Level Weighted Sum and Threshold Operations." *IEEE Transactions on Electron Devices*, 39(6), pp. 1444–1455, June 1992.
- [121] Shibata, T. and Ohmi, T. "Neuron MOS Binary-Logic Integrated Circuits – Part I: Design Fundamentals and Soft-Hardware-Logic Circuit Implementation." *IEEE Transactions on Electron Devices*, 40(3), pp. 570–576, March 1993.
- [122] Shibata, T. and Ohmi, T. "Neuron MOS Binary-Logic Integrated Circuits – Part II: Simplifying Techniques of Circuit Configuration and their Practical Applications." *IEEE Transactions on Electron Devices*, 40(5), pp. 974–979, May 1993.
- [123] Shieh, M.-H. and Lin, H. C. "A Multiple-Dimensional Multiple-State SRAM Cell Using Resonant Tunneling Diodes." *IEEE Journal of Solid-State Circuits*, 29(5), pp. 623–630, May 1994.
- [124] Smith, J. and Cook, P. "The second-order Digital Waveguide Oscillator." In *Proceedings of the International Computer Music Conference*, pp. 150–153. 1992.
-

- 
- [125] Smith, J. O. and Angell, J. B. "A Constant-Gain Digital Resonator Tuned by a Single Coefficient." *Computer Music Journal*, 6(4), pp. 36-40, Winter 1982.
- [126] Smith, III, J. O. "Physical Modeling using Digital Waveguides." *Computer Music Journal*, 16(4), pp. 74-91, Winter 1992.
- [127] Snell, J. "Design of a Digital Oscillator That Will Generate upto 256 Low-Distortion Sine Waves in Real Time." In .Roads, C. and Strawn, J., eds., *Foundations of Computer Music*, pp. 289-325. MIT Press, Cambridge, Massachusetts, 1985. Appeared in *Computer Music Journal* 1(2): 4-25, 1977.
- [128] Spanier, J., Johnson, S. and Purvis, A. "Optimisations of the FOF Algorithm for VLSI Implementation." In *Proceedings of the International Computer Music Conference*, pp. 493-495. Hong Kong, August 1996.
- [129] Stilson, T. and Smith, J. "Alias-Free Digital Synthesis of Classic Analog Waveforms." In *Proceedings of the International Computer Music Conference*, pp. 332-335. Hong Kong, August 1996.
- [130] Sundberg, J. "Synthesis of Singing by Rule." In Mathews, M. and Pierce, J., eds., *Current Directions in Computer Music Research*, pp. 45-55. MIT Press, Cambridge, Massachusetts, 1989.
- [131] Takagi, N., Asada, T. and Yajima, S. "Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation." *IEEE Transactions on Computers*, 40(9), pp. 989-995, September 1991.
- [132] Taylor, F. "Residue arithmetic: a tutorial with examples." *IEEE Computer Magazine*, 17(5), pp. 50-62, 1984.
- [133] Taylor, F. "On the complex residue arithmetic system ( CRNS )." *IEEE Transactions On Acoustics, Speech And Signal Processing*, ASSP-34(6), pp. 1675-1677, 1986.
- [134] Taylor, F. and Huang, C. "A floating-point residue Arithmetic Unit." *Journal of the Franklin Institute*, 311(1), pp. 33-53, January 1981.
- [135] Tempelaars, S. "The VOSIM Signal Spectrum." *Interface*, 6, pp. 81-96, 1977.
- [136] Tomovic, R. and Karplus, W. J. *HIGH SPEED ANALOG COMPUTERS*. John Wiley & Sons, Inc., 1962.
-

- 
- [137] Vail, M. *Vintage Synthesizers : Groundbreaking Instruments and Pioneering Designers of Electronic Music Synthesizers*. Miller Freeman Books, San Francisco, 1993. ISBN 0-87930-275-5. (pbk).
- [138] Vercoe, B. "Extended Csound." In *Proceedings of the International Computer Music Conference*, pp. 141–142. Hong Kong, August 1996.
- [139] Vittoz, E. "Analog VLSI Signal Processing : Why, Where and How ?" *Journal of VLSI Signal Processing*, 8(1), pp. 27–44, 1994.
- [140] Volder, J. "The CORDIC Trigonometric Computing Technique." *IRE Transactions on Electronic Computers*, EC-8(3), pp. 330–334, September 1959.
- [141] Walther, J. "A unified algorithm for elementary functions." In *Proceedings Of The Spring Joint Computer Conference - AFIPS*, pp. 379–385. Atlantic City, N.J., U.S.A, May 1971.
- [142] Wang, A. L.-C. and Smith, III, J. O. "On fast FIR filters implemented as tail-canceling IIR filters." TR Stan-M-90, CCRMA, Dept. of Music, Stanford University, Stanford, CA 94305-8180, November 1994. Obtained from <ftp://ccrma-ftp.stanford.edu>.
- [143] Wawrzynek, J. "VLSI Models for Sound Synthesis." In Mathews, M. and Pierce, J., eds., *Current Directions in Computer Music Research*, pp. 113–148. MIT Press, Cambridge, Massachusetts, 1989.
- [144] Wawrzynek, J. and Mead, C. "A VLSI Architecture for Sound Synthesis." In Denyer, P. and Renshaw, D., eds., *VLSI Signal Processing: a Bit-Serial Approach*, pp. 277–297. Addison-Wesley Publishing Company, 1985.
- [145] Wayner, P. "Silicon in Reverse." *Byte Magazine*, 19(8), pp. 67–74, August 1994.
- [146] Weber, W., Prange, S. J., Thewes, R., Wohlrab, E. and Luck, A. "On the Application of the Neuron MOS Transistor Principle for Modern VLSI Design." *IEEE Transactions on Electron Devices*, 43(10), pp. 1700–1708, October 1996.
- [147] Wei, S.-J. and Lin, H. C. "Multivalued SRAM Cell Using Resonant Tunneling Diodes." *IEEE Journal of Solid-State Circuits*, 27(2), pp. 212–216, February 1992.
- [148] Wilkins, B. *Analogue and Iterative methods in Computation, Simulation and Control*. Chapman and Hall Ltd., 1970.
-

- [149] Willoughby, M. *Design Framework II : Automatic Place and Route*. Rutherford Appleton Laboratory, May 1995.
- [150] Woods, R., Floyd, G., Wood, K., Evans, R. and McCanny, J. "Programmable high-performance IIR filter chip." *IEE Proceedings - Circuits and Systems*, 142(3), pp. 179–185, June 1995.
- [151] Yamashina, M. and Yamada, H. "An MOS Current Mode Logic (MCML) Circuit for Low-Power GHz Processors." *NEC Research and Development*, 36(1), pp. 54–63, January 1995.
- [152] Yassine, H. and Moore, W. "Improved Mixed-Radix Conversion for Residue Number System Architectures." *IEE Proceedings - G*, 138(1), pp. 120–124, February 1991.
- [153] Yuh, J.-D. and Newcomb, R. W. "A Multilevel Neural Network for A/D Conversion." *IEEE Transactions on Neural Networks*, 4(3), pp. 470–483, May 1993.
- [154] Zölzer, U. "Roundoff Error Analysis of Digital Filters." *Journal of the Audio Engineering Society*, 42(4), pp. 232–244, April 1994.

