Imperial College London
Department of Department of Electrical and Electronic Engineering

VARIATION-AWARE HIGH-LEVEL DSP CIRCUIT DESIGN OPTIMISATION FRAMEWORK FOR FPGAs

Rui Policarpo Duarte

July 31, 2014

Supervised by Dr. Christos-Savvas Bouganis

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Department of Electrical and Electronic Engineering of
Imperial College London

Declaration of Originality

I herewith certify that all material in this dissertation which is not my own work has been properly acknowledged.

Rui Policarpo Duarte

Copyright Declaration

VARIATION-AWARE HIGH-LEVEL DSP CIRCUIT DESIGN OPTIMISATION FRAMEWORK FOR FPGAs

or: How I Learned to Stop Worrying and Love the Uncertainty

Abstract

The constant technology shrinking and the increasing demand for systems that operate under different power profiles with the maximum performance, have motivated the work in this thesis. Modern design tools that target FPGA devices take a conservative approach in the estimation of the maximum performance that can be achieved by a design when it is placed on a device, accounting for any variability in the fabrication process of the device. The work presented here takes a new view on the performance improvement of DSP designs by pushing them into the error-prone regime, as defined by the synthesis tools, and by investigating methodologies that reduce the impact of timing errors at the output of the system. In this work two novel error reduction techniques are proposed to address this problem. One is based on reduced-precision redundancy and the other on an error optimisation framework that uses information from a prior characterisation of the device. The first one is a generic architecture that is appended to existing arithmetic operators. The second defines the high-level parameters of the algorithm without using extra resources. Both of these methods allow to achieve graceful degradation whilst variation increases. A comparison of the new methods is laid against the existing methodologies, and conclusions drawn on the tradeoffs between their cost, in terms of resources and errors, and their benefits in terms of throughput. In some cases it is possible to double the performance of the design while still producing valid results.

Acknowledgements

The work in this thesis was carried out under the supervision of Dr. Christos-Savvas Bouganis. I hereby acknowledge his valuable advices and guidance throughout my presence at Imperial College London, and his help through the critical moments of my PhD.

This research was supported by Fundação para a Ciência e Tecnologia (Foundation for Science and Technology in Portugal) through PhD grant SFRH/BD/69587.

To my parents.

Contents

Contents **8**

List of Figures **13**

List of Tables **22**

List of Algorithms **25**

List of Abbreviations **26**

1. Introduction **29**

1.1. Motivation 29

1.2. Overview 34

1.3. Contributions 35

2. Background and Related Work **37**

2.1. Introduction 37

2.2. Synthesis of Arithmetic Circuits for FPGAs 38

2.2.1. Embedded Arithmetic Blocks 38

2.2.2. Dot-Product Operator 41

2.2.3. Linear Projection Algorithm 43

2.3. Sources of Variation 45
2.4. Error Analysis 47
2.5. Variation-Aware Methods for Throughput Increase in FPGAs 50
2.5.1. Variation-Aware Placement and Routing 50
2.5.2. Path-Delay Reduction 52
2.6. Error Recovery Methods 52
2.6.1. Razor 53
2.6.2. Reduced-Precision Redundancy 54
2.7. Probabilistic Computing 57
2.8. Resource Optimisation Through Bayesian Inference 58
2.8.1. Bayesian Factor Analysis Model 60
2.9. Summary 62

3. Performance of Arithmetic Units Under Variation 65
3.1. Introduction 65
3.2. Characterisation Framework 67
3.2.1. Introduction 67
3.2.2. Circuit Architecture 68
3.2.3. Characterisation Process 69
3.2.4. Software Support 72
3.3. Performance of Arithmetic Units Under Variation 73
3.3.1. Adder 74
3.3.2. Constant Coefficient Multiplier 79
3.3.3. LUT-Based Generic Multiplier 82
3.3.4. DSP-Based Multiplier 84
3.3.5. Voltage and Temperature Variation 84
3.3.6. Intra-die Process Variation 86

3.3.7. Inter-die Process Variation 88
3.3.8. Process Size Variation 89
3.4. Impact of Variation in Linear Projection Designs 102
3.5. Run-Time Investigation 103
3.6. Summary 106

4. Redundancy in Arithmetic Units 108
4.1. Introduction 108
4.2. Reduced-Precision Redundancy Framework 110
4.2.1. Architecture 111
4.2.2. Approximation Functions 116
4.2.3. Error Function Minimisation 124
4.3. ROM-XOR RPR Arithmetic Operators 125
4.3.1. Adder 127
4.3.2. Multiplier 131
4.3.3. Multiplier-Accumulator 135
4.4. Reduced-Precision Redundancy Evaluation 137
4.4.1. Circuit Architecture 138
4.4.2. Adder 140
4.4.3. Multiplier 141
4.4.4. Multiplier-Accumulator 147
4.4.5. Linear Projection Designs 147
4.5. Summary 149

5. Optimisation Framework for Acceleration of Linear Projection Designs 151
5.1. Introduction 151
5.2. Bayesian Formulation 155

5.3. Objective Function 156
5.4. Sampling From a Posterior Distribution 158
5.4.1. Prior Distribution 158
5.4.2. Error Model 159
5.4.3. Area Model 160
5.5. Design Optimisation 161
5.6. DSP Block Support 164
5.7. Optimisation Strategies for Linear Projections 166
5.7.1. Linear Projection Targeting a Maximum Reconstruction MSE 167
5.8. Optimisation Framework Evaluation 167
5.8.1. Characterisation and Training Samples 169
5.8.2. Circuit Architecture 174
5.8.3. Area and Error Models Evaluation 177
5.8.4. Optimisation of Linear Projections for Throughput, Area and Errors 178
5.8.5. Optimisation of Linear Projections for Throughput and Errors 183
5.8.6. Scalability & Run-Time Investigation 190
5.9. Summary 192

6. Conclusions and Future Work **195**
6.1. Conclusions 195
6.2. Future Work 197
6.2.1. Short-Term Goals 197
6.2.2. Long-Term Goals 199

A. Appendix **202**
A.1. Hardware Platforms 202
A.2. FPGA Core Voltage and Temperature Control 203

A.3. Source code 204

Bibliography **207**

List of Figures

1.1. Examples of linear projection on 5 images (A-E) with 2000 dimensions (50×40 pixels) performed on a Cyclone III FPGA. The top row shows the result of a back-projection, from their projection on the FPGA to a smaller space with 40 dimensions without timing errors. The middle and bottom rows show the same result for an over-clocked implementation, and an implementation with projection coefficients quantised with 3 bits, respectively. 31

1.2. Percentage of erroneous results at the output of an arithmetic unit *vs* its clock frequency. The error-free ($\Delta f1$) and error-prone ($\Delta f2$) regimes are depicted as well as the conservative operational limit imposed by the synthesis tool (f_A). 32

2.1. Details of DSP Blocks available on Cyclone III and IV FPGAs from Altera (from [1, 2]). 39

2.2. Details of DSP Blocks available on Cyclone V FPGA from Altera (from [3]). 40

2.3. Block diagram of the circuit to do the unrolled implementation of dot-product between two vectors. 42

2.4. Block diagram of the circuit to do the rolled implementation of dot-product between two vectors. 43
2.5. High-level block diagram of the circuit to implement a Z^6 to Z^3 linear projection algorithm. 44
2.6. Original data set with 6 images of the same subject under different illumination conditions. 45
2.7. Example of the linear projection algorithm applied to one of the images from the data set using different word lengths (x axis) for the projection coefficients and different numbers of projection vectors (y axis). The images shown are the back-projections in the original space. 46
2.8. Classes of variation maps with different delay patterns, created from the characterisation of 129 FPGAs (from [4]). 51
2.9. Razor architecture (from [5]). 54
2.10. Typical RPR architecture applied to a system under VoS. 55
2.11. Error-detection boundaries of a RPR system. 56
2.12. High-level block diagram of the Bayesian framework (from [6]). . . . 59

3.1. Architecture of the circuit for the characterisation of arithmetic units. 69
3.2. Floor plan of the characterisation circuit for 3 multipliers on a Cyclone III 3C16 device from Altera. 70
3.3. High-level flow of the characterisation framework. 71
3.4. Flow chart of the actions executed by the TCL script to control the characterisation circuit on the FPGA. 72
3.5. Memory organisation for the BRAM, in the characterisation circuit, holding the input stream data. 74

3.6. Error variance and mean error for a 16-bit unsigned adder on a Cyclone III FPGA operated at 1200 mV, 20°C tested over a range of clock frequencies. 75
3.7. Histogram of the magnitude of errors, in $\log_2$ scale at the output of a 16-bit unsigned adder at 700 MHz. 76
3.8. Error variance on a 16-bit unsigned adder, on a Cyclone III FPGA, at different clock frequencies (blue), and the total number of critical-paths with the corresponding critical clock frequency (green). 78
3.9. Number of critical-paths per output bit of a 16-bit unsigned adder, in blue. The top 120 are presented in red. 79
3.10. Statistics for errors of CCMs sorted by different metrics: CCM value (1st row), area (2nd row), error variance (3rd row), mean error (4th row) and Hamming distance (5th row). 81
3.11. Variance of constant coefficients for CCMs tested twice on a Cyclone III FPGA under different clock frequencies. 81
3.12. Variance of constant coefficients for a LUT-based generic multiplier under different clock frequencies. 82
3.13. The first 100 error values from a 8-bit LUT-based unsigned multiplier (left) and the distribution of all errors (right), for constant multiplicand 222 in 2 locations of a Cyclone III FPGA at 320 MHz. 83
3.14. Error variance results for DSP-based multipliers targeting low-power (1000 mV, 35 °C) and high-performance designs (1400 mV, 5 °C). . . 85
3.15. Results for DSP-based multipliers with voltage variation. 85
3.16. Results for DSP-based multipliers with temperature variation. 86
3.17. Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #1. 87

3.18. Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #2. 89
3.19. Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #3. 90
3.20. Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #4. 90
3.21. Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #5. 91
3.22. Distribution of errors for 3 DSP-based multipliers on 3 different locations in DE0 board #4. 92
3.23. Error variance for DSP-based multipliers on 3 different locations in DE0 board #4. 93
3.24. Mean error for DSP-based multipliers on 3 different locations in DE0 board #4. 94
3.25. Error variance for DSP-based multipliers on 3 different locations in board #5. 95
3.26. Mean error for DSP-based multipliers on 3 different locations in DE0 board #5. 96
3.27. Error variance for DSP-based multipliers on 3 different locations on a Cyclone IV FPGA, DE0 Nano board #2, tests 1 and 2. 97
3.28. Results for DSP-based Multipliers on a Cyclone IV using different designs. 98
3.29. Error variance of 3 DSP-based multipliers on a Cyclone IV FPGA (DE0 Nano board #1). 99
3.30. Mean error of 3 DSP-based multipliers on a Cyclone IV FPGA (DE0 Nano board #1). 100

3.31. Error variance of 3 over-clocked DSP-based multipliers on a Cyclone IV FPGA (DE0 Nano board #2). 100
3.32. Mean error of 3 over-clocked DSP-based multipliers on a Cyclone IV FPGA (DE0 Nano board #2). 101
3.33. Error variance of over-clocked DSP-based multipliers on Cyclone III, IV and V FPGAs. 102
3.34. Faces obtained from the reconstruction of the linear projection implemented with LUT-based multipliers operating at 230, 250, 270 and 300 MHz. 104
3.35. Faces obtained from the reconstruction of the linear projection implemented with DSP-based multipliers operating at 510, 530, 550, 570 and 590 MHz. 105

4.1. Typical RPR architecture applied to a system under VoS. 110
4.2. New RPR architecture applied to a generic combinatorial operator. . 111
4.3. Illustration of the maximum clock frequencies, and delays, for standard and RPR units and their operating regimes: error-free/expected result (green), error-prone/approximated result (orange) and incorrect result (red). 114
4.4. Values of the 2 MSBits for the expected result and approximation functions 1 and 2. 120
4.5. Values of the 2 MSBits for the expected result and approximation function 1 and 3 MSBits for approximation function 2. 121
4.6. Diagram of the circuit attached to the arithmetic operators to produce the approximations and signaling of which result to use. 126
4.7. Diagram of an adder circuit with the proposed RPR scheme. 127

4.8. Example of an approximation produced for a 8:9/5:2/5:2 ROM-XOR RPR adder. 129
4.9. Difference in the MSBits between the detection approximation and the expected result for an 8-bit adder. 130
4.10. Results produced by a ROM-XOR 8:9/5:2/5:2 RPR adder. 131
4.11. Diagram of a multiplier circuit with the proposed RPR scheme. . . . 132
4.12. Example of an approximation produced for a ROM-XOR 8:16/5:2/5:2 RPR multiplier. 133
4.13. Difference between the detection approximation and the expected MSBits at the output of the multiplier. 134
4.14. Results produced by a ROM-XOR 8:9/5:2/5:2 RPR multiplier. . . . 135
4.15. Diagram of a rolled multiply-accumulate circuit with the proposed RPR scheme. 136
4.16. Diagram of a rolled multiply-accumulate circuit using basic arithmetic units with the proposed RPR scheme. 136
4.17. Block diagram of the test circuit for RPR units under variation. . . 138
4.18. Floor plan of the test circuit (red) for an RPR multiplier (yellow). . 139
4.19. Errors in the supporting blocks of the test circuit for different clock frequencies on a DE0 board. 140
4.20. Error variance and mean error of an 16-bit adder without RPR . . . 142
4.21. Variance and mean error at the output of an 8x8 bit unsigned LUT-based multipliers at different clock frequencies. 143
4.22. Errors in results (top) and error histogram (bottom) at the output of a NO RPR 8-bit multiplier at 300 MHz. 144
4.23. Errors in results (top) and error histogram (bottom) at the output of a LUT-SUB RPR 8-bit multiplier at 300 MHz. 145

4.24. Errors in results (top) and error histogram (bottom) at the output of a ROM-XOR RPR 8-bit multiplier at 300 MHz. 145
4.25. Histogram of the bits present in the values with errors at the output of the three LUT-based 8-bit multipliers tested at 300 MHz. 146
4.26. Architecture of a circuit to implement the dot-product operator of a projection vector. 149
4.27. Reconstructed faces (A-E) in the original space without variation errors (expected), and computed from the projections collected from implementations of different multipliers architectures (NO RPR, LUT-SUB RPR, ROM-XOR RPR) at 270 MHz. 150

5.1. Design flow using the proposed optimisation framework. 153
5.2. Block diagram of the circuit to do the rolled implementation of dot-product between two vectors. 154
5.3. Error variance of an unsigned 8-bit LUT-based generic multiplier. . . 160
5.4. Prior distribution for $\alpha = 0$ and $\beta = [0.1, 1.0, 4.0]$ for an 8-bit unsigned multiplier at 340 MHz. 162
5.5. Illustration of the generation of linear projection designs with 3 projected dimensions and $Q = 3$. 164
5.6. Histogram showing the distribution of the test samples generated for the linear projection test case according to their value. 170
5.7. Reconstruction MSE and Confidence Intervals for different training set sizes. 173
5.8. Histogram of 1.9k characterisation samples used in the characterisation of the arithmetic units for the Z^6 to Z^3 linear projection. 174
5.9. Histogram of 100 samples used in the training of the Optimisation Framework for the Z^6 to Z^3 linear projection. 175

5.10. Block diagram of the circuit to test the dot-product implementation, used in linear projection case study. 176
5.11. Evaluation of the area model against the actual circuit area for linear projection designs using LUT-based multipliers. 178
5.12. *Predicted*, *simulated* and *actual* performance reconstruction MSE vs. area of the linear projection designs produced by the proposed optimisation framework using LUT-based multipliers. The target clock frequency is 310MHz. 179
5.13. Estimated circuit area *vs* model reconstruction PSNR at 510 MHz. . 181
5.14. Actual circuit area *vs* implementation reconstruction PSNR at 510 MHz. 182
5.15. Maximum clock frequencies *vs* word length for a Z^6 to Z^3 linear projection circuit designed by the KLT transform. 183
5.16. MSE for the reconstruction of the projected data in the original space. The design points of the KLT correspond to 3-9 bit coefficient word length. 184
5.17. Comparison of the performance of the KLT and OF designs for the particular case of 1400 mV and 5 °C. 185
5.18. Performance of the KLT linear projection application under different core voltages (900 mV,1000 mV,1100 mV,1200 mV). 186
5.19. Performance comparison between the KLT and OF designs for the particular case of 900 mV. 187
5.20. Performance of the KLT optimised designs for low-power, tested on a different Cyclone III FPGA. 188
5.21. Performance of the KLT linear projection application depending on the temperature of the device. 189
5.22. Performance comparison between the KLT and OF designs at 20 °C (right). 190

5.23. Performance of the KLT and OF designs at 35 °C 191
5.24. Performance of the KLT and OF designs at 50 °C. 192
5.25. Performance of the KLT and OF designs for a Z^{10} to Z^4 linear projection at 1200 mV/20 °C. 193
5.26. Diagram with the relation between the different research areas around the proposed optimisation framework. 194

A.1. System to control core voltage and temperature of the FPGA. 204
A.2. Photo of the DE0 and DE0 Nano boards equipped with a thermoelectric cooler and a water-cooled heat-sink to control the temperature on the surface of the FPGAs. 205

List of Tables

2.1. Technology sizes of FPGAs from different Cyclone families and their maximum clock frequencies, for the different configurations of their DSP blocks. 39
2.2. Summary of the different types of variation, their origin and their contribution for the delay increase in circuit paths. 47
2.3. Summary of the existing techniques for acceleration of computations, in order to mitigate timing errors, and their limitations. 64

3.1. Top 15 most critical-paths from the slow model at 1200 mV 85°C for a 16-bit adder. 77
3.2. Operating conditions for characterisation of the DSP-based multipliers. 84
3.3. Locations of the DSP-based multipliers for the characterisation tests of a Cyclone III FPGA. 87
3.4. Locations of the DSP-based multipliers for the characterisation tests of a Cyclone IV FPGA. 91
3.5. Impact of the different sources of variation in the performance of arithmetic units. 107

4.1. Results of 2x2-bit multiplications using the original and truncated operands. 114

4.2. Denominations of the labels in the nomenclature adopted for the proposed RPR scheme. 115
4.3. Results of 2x2-bit multiplications using the original and truncated operands. 119
4.4. Results produced by two different approximations for 2x2-bit multiplications using truncated operands. 119
4.5. Number of BRAMs (M9K/M10K) per input and output word lengths required to implement the approximation ROM in different Cyclone FPGAs. 123
4.6. Objective functions for errors at the output of the RPR unit. 124
4.7. Resources and approximation results for different implementations of a single approximation function for a ROM-XOR 8:9/iWL:oWL/iWL:oWL RPR adder. 129
4.8. Correctness of the MSBits for different coefficients of the approximation function for ROM-XOR 8:9/iWL:oWL/iWL:oWL RPR addition, using different linear approximations. 130
4.9. Correctness of the MSBits for different approximation coefficients for ROM-XOR 8:16/iWL:oWL/iWL:oWL RPR multiplication. 133
4.10. Correctness of the MSBits for different approximation coefficients in an ROM-XOR 8:16/iWL:oWL/iWL:oWL RPR multiplication. . . . 134
4.11. Maximum clock frequency reported by the synthesis tool of 16-bit adders. 140
4.12. Resources, in LEs, taken by the different adder implementations. . . 141
4.13. Maximum clock frequency, in MHz, for operation without errors. . . 143
4.14. Resources taken by the different multiplier implementations. 143
4.15. Resources, in LEs, taken by the different multiplier implementations. 144

4.16. Power consumed by the different multiplier implementations, and respective test circuits, at 1200 mV, 20°C, 100 MHz and 600 MHz. . . . 144
4.17. Maximum clock frequency for different arithmetic unit given by the synthesis tools. 147

5.1. Results for the Jackknife resampling method within a 95% confidence interval. 171
5.2. Results for the Bootstrap resampling method for different sets with sampled variances within the 95% confidence interval. 172
5.3. Mean and Variance of the different data sets 174

A.1. Hardware platforms used in this work, and their main features. . . . 202
A.2. FPGAs used in this work, and their main features. 203

List of Algorithms

1. Algorithm to search the approximation coefficients for the $\star$ RPR operator. 118
2. Algorithm to sample Q linear projection designs from a prior distribution. 165

List of Abbreviations

ALM Adaptive Logic Module

ASIC Application-Specific Integrated Circuit

BRAM Block RAM

CCM Constant Coefficient Multiplier

CF Characterisation Framework

CMOS Complementary Metal-Oxide-Semiconductor

CSD Canonical Signed Digit

CPU Central Processing Unit

DVS Dynamic Voltage Scaling

DSP Digital Signal Processing

DUT Design Under Test

EEG Electroencephalogram

EMG Electromyography

FIR Finite Impulse Response

FPGA Field-Programmable Gate Array

FSM Finite State Machine

IIR Infinite Impulse Response

JTAG Joint Test Action Group

KLT Karhunen-Loeve Transformation

LE Logic Element

LFSR Linear Feedback Shift Register

LSBit Least Significant Bit

LSB Least Significant Byte

LUT Look-Up Table

MAC Multiply-Accumulate

MSE Mean-Square Error

MSBit Most Significant Bit

MSB Most Significant Byte

OF Optimisation Framework

PBL Probabilistic Boolean Logic

PCA Principal Component Analysis

PID Proportional Integral and Derivative

PLL Phase-Locked Loop

PSNR Peak Signal-to-Noise Ratio

PSOC Probabilistic System-on-a-Chip

PVT Process-Voltage-Temperature

RAM Random-Access Memory

ROM Read-Only Memory

RPR Reduced-Precision Redundancy

RTL Register-Transfer Level

SAR Synthetic Aperture Radar

SEU Single Event Upset

TCL Tool Command Language

TCP Transmission Control Protocol

TEC Thermoelectric Cooler

TMR Triple-Modular Redundancy

USB Universal Serial Bus

VHDL Very High-Speed Hardware Description Language

VLSI Very Large Scale Integration

VoS Voltage Over-Scaling

1 Introduction

1.1. Motivation

The constant scaling of the fabrication process has led to devices with increased performance characteristics by supporting higher clock frequencies with less power consumption, but also to an increased variation in *intra-die* and *inter-die* performance characteristics. This process variation affects the characteristics of transistors on a device, changing physical dimensions, altering the timing threshold and finally affecting their overall performance. Besides variations introduced by the physical constraints, transistors are also affected by other parameters such as voltage and temperature. This leads to devices that exhibit uneven performance across its

area and, as the technology continues to scale down, the fabricated devices will be even more susceptible to such variations [7]. As such, modern devices are no longer limited by their process technology performance, but by the performance of their worst performing transistor. Considering that more than one device is manufactured with the same fabrication process, we face many devices with different performance limits. Nowadays, setting operating limits not only concerns the worst performing transistor on a device, it is the performance of the worst transistor made for all manufactured devices of a given family.

To ensure that the implemented designs operate without errors once placed on an Field-Programmable Gate Array (FPGA) device, the synthesis tools use conservative models to determine the maximum error-free clock frequency of circuits for a family of devices. As a consequence, there is a significant gap between the clock frequency that can be achieved, as dictated by the models used within the synthesis tools, and what actually can be achieved by the actual device where the circuit will be placed on.

Designs that demand real-time performance, e.g. face recognition [8], can benefit from the aforementioned gap, via over-clocking, but at the chance of incurring in violation of the timing of the paths in its circuit, thus producing erroneous results [9]. Figure 1.1 illustrates the linear projection application on 5 subjects, with errors in computations from over-clocking and quantisation, and its back-projection in the original space. The top row corresponds to the result from the FPGA without errors, below the maximum clock frequency reported by the synthesis tool (160 MHz). The middle row shows the result obtained from the device operating with errors in the datapath when over-clocked at 270 MHz. The bottom row shows the results for the implementation of the linear projection using coefficients quantised with the maximum word length that produces designs that can operate at 270 MHz without errors, 3 bits in this case. In real-life scenarios, in implementations where throughput

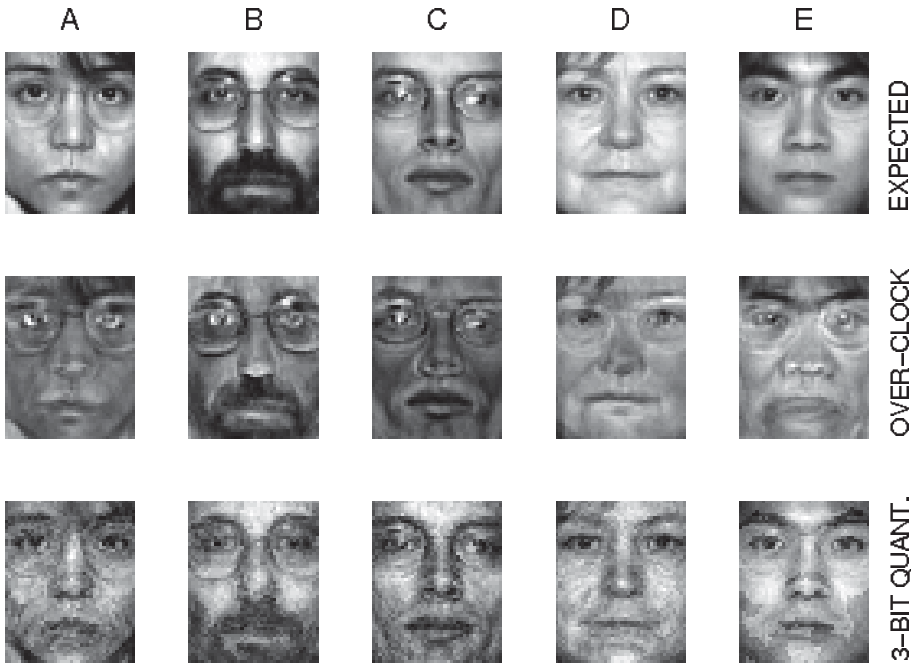

Figure 1.1.: Examples of linear projection on 5 images (A-E) with 2000 dimensions (50×40 pixels) performed on a Cyclone III FPGA. The top row shows the result of a back-projection, from their projection on the FPGA to a smaller space with 40 dimensions without timing errors. The middle and bottom rows show the same result for an over-clocked implementation, and an implementation with projection coefficients quantised with 3 bits, respectively.

is a hard constraint, the choice is often made between allowing errors to occur under variation of the operating conditions, regardless of the impact; or reduce the quality of the computations in order to operate without timing errors.

The focus of this thesis is to investigate mechanisms to increase the throughput of arithmetic units in Digital Signal Processing (DSP) designs on FPGAs under Process-Voltage-Temperature (PVT) variation without changing the algorithm being implemented, while investigating the tradeoff in throughput, circuit area and timing errors. Figure 1.2 illustrates the proposed concept when an arithmetic unit is operated under different clock frequencies. The maximum performance of the operator as reported by the synthesis tools (f_A) is illustrated along with the oper-

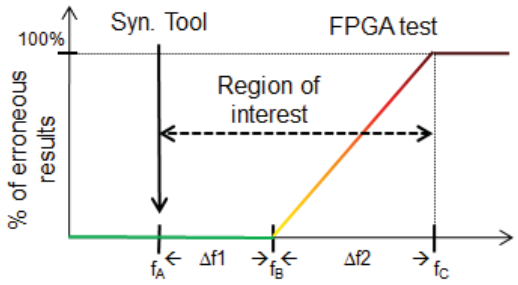

Figure 1.2.: Percentage of erroneous results at the output of an arithmetic unit *vs* its clock frequency. The error-free ($\Delta f1$) and error-prone ($\Delta f2$) regimes are depicted as well as the conservative operational limit imposed by the synthesis tool (f_A).

ational regions where the module can operate in a specific device under error-free ($\Delta f1$) and under error-prone regimes ($\Delta f2$). f_B and f_C represent the clock frequencies up to which the design operates on the FPGA without errors and with errors, respectively. Above f_C the arithmetic unit doesn't produce useful results.

Doing a characterisation of the arithmetic units prior to implementation, admits the possibility to determine the maximum clock frequency under specific operating conditions, thus avoiding timing errors. The drawback is the assurance of the same operating conditions throughout its lifetime, and accounting for aging.

On the other hand, applications that can tolerate some errors in their calculations can use it to maximise their operating clock frequency. Moreover, if the data in the problem being considered doesn't exercise the critical-paths in the design, or if it exercises them rarely, it grants the opportunity to increase the clock frequency beyond worst-case values.

When the throughput requirements of a design, that can't be further pipelined, require it to operate in an error-prone regime, redundancy is a candidate mitigation method because it offers an approximate result as an alternative to a wrong result. Moreover, being generic, it can be added to almost any DSP system [10, 11]. However, it requires extra resources and latency to be implemented. This extra latency

may not fit the implementation of algorithms that require all data to do be computed every clock cycle. Therefore, there is an opportunity in terms of research for methods and architectures to address redundancy within one clock cycle that needs to be investigated.

On the other hand, the extra resources may not be always available, therefore it is of interest to investigate the possibility to specify optimal parameters in an algorithm implementation to minimise timing errors. The idea is borrowed from a previous research work [6] where the high-level specification of the algorithm is created by a Bayesian framework considering the problem data and implementation resources simultaneously. Hence, an investigation of an extension of this concept to consider not only resources but also errors is envisioned.

As aforementioned, this research work considers DSP applications that can tolerate some errors in their calculations. In this direction, applications that retrieve data from sensors, or produce *fuzzy* results, can benefit from the methods discussed later in this thesis. Examples of candidate applications to operate under different operating conditions (i.e. high-performance and low-power) are: face recognition [12] and Synthetic Aperture Radar (SAR) [13] for real-time, or high-performance; and Electroencephalogram (EEG) [14] and Electromyography (EMG) [15] for low-power.

Often these applications are implemented on FPGAs because of the advantages they offer as low-power, high-performance and highly specialised embedded blocks. Moreover, FPGAs are well positioned to tackle the aforementioned research problems because of their reconfigurability capabilities which is essential for the characterisation process that no other competitive technologies offer.

1.2. Overview

Chapter 2 of this thesis starts with an overview on the specialised arithmetic units embedded in modern FPGA devices, and how they can be used to implement DSP applications. It also addresses the implementation of the linear projection algorithm, as an example of a candidate DSP application to be accelerated. Afterwards it gives a review on the sources of variation that can change the correct functioning of FPGAs when operating beyond the maximum limits, along with the state-of-the-art strategies to mitigate the uncertainty in the DSP design.

A framework for the characterisation of arithmetic units on FPGAs under variation is presented in Chapter 3. It then investigates the impact of different sources of variation on the most common arithmetic operators existing in the DSP designs implemented on FPGAs. Moreover, the impact of variation on the device is also assessed by comparing the output of a linear projection application against the expected results.

Chapter 4 details a novel Reduced-Precision Redundancy (RPR) framework for high-throughput without extra latency impact. This new RPR scheme is proposed as a mechanism to add resilience to arithmetic operators, in DSP designs that don't tolerate insertion of extra latency in their implementations. An evaluation on how basic arithmetic operators can use the proposed RPR framework is presented, along with the comparison between typical implementations. In addition, a linear projection application using the new RPR scheme is also evaluated.

Chapter 5 proposes an alternative method to add resilience to linear projection designs without adding extra resources to the arithmetic units, neither changing the implementation of the algorithm. Being the linear projection a DSP algorithm that tolerates some errors in its calculations, the proposed optimisation framework takes advantage of that fact to produce designs with deviations in their implementations

but that will perform, under variation, better than typical implementations. The optimisation framework proposed uses a prior characterisation of the arithmetic units on the FPGA under variation to create the linear projection designs. The effectiveness of the proposed optimisation framework is tested along with the typical implementation of the linear projection algorithm.

Chapter 6 concludes this thesis with comments and discussions on the benefits and tradeoffs resulting from the conducted research.

1.3. Contributions

The research work investigated in this thesis has produced results that originated several publications, accepted in peer-reviewed conditions, which are listed below:

- [16] R. P. Duarte and C.-S. Bouganis, "High-level linear projection circuit design optimization framework for FPGAs under over-clocking," in *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pp. 723–726, Aug 2012.

- [17] R. P. Duarte and C.-S. Bouganis, "A unified framework for over-clocking linear projections on FPGAs under PVT variation," in *Applied Reconfigurable Computing (ARC), 2014 10th International Symposium on*, pp. 49–60, 2014.

- [18] R. P. Duarte and C.-S. Bouganis, "Pushing the performance boundary of linear projection designs through device specific optimisations (abstract)," in *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, FPGA '14, (New York, NY, USA), pp. 245–245, ACM, 2014.

- [19] R. P. Duarte and C.-S. Bouganis, "Over-clocking of linear projection designs through device specific optimisations," in *21st Reconfigurable Architec-*

tures Workshop (RAW 2014), 2014.

2

Background and Related Work

2.1. Introduction

Design of DSP designs on FPGAs generally takes advantage of various features offered by this technology: customised circuit architecture and datapath, high-performance, low-power and small footprint, reconfigurability and specialised embedded blocks. These features have attracted designers to take advantage of them to optimise resources, performance and precision of the computations required for a given DSP application. However, FPGAs are silicon-based devices, hence they are subjected to physical limitations, namely degradation, process variation and variations in voltage and temperature.

This chapter revises the main aspects of synthesis of DSP designs on different FPGA families, and the linear projection algorithm as an the example of a DSP application to be implemented on FPGAs. It then proceeds to describe the investigation on the main sources of variability in silicon devices, as well as some of the design techniques proposed to exploit, or mitigate, the impact of variation in silicon devices while pursuing the maximisation of the performance of the system.

2.2. Synthesis of Arithmetic Circuits for FPGAs

A lot of research has been devoted to design and optimisation of arithmetic circuits for FPGAs. This section revises specific aspects of it, in terms of available technology and related work, that matter to the present research problem.

2.2.1. Embedded Arithmetic Blocks

FPGAs offer highly specialised embedded blocks that favour the implementation of DSP designs. These blocks implement frequently used basic arithmetic operators that would require many Logic Elements (LEs) in their implementation. Thus, they offer an increase in maximum clock frequency and savings in LEs. For this reason the complexity of the embedded blocks has increased, offering more functionality, with the new generations of FPGAs.

In the particular case of the devices considered for this research work, namely Cyclone III, IV and V [1, 2, 3], they all have embedded multipliers available. Cyclone V [3] offers DSP blocks with more functionality besides the multiplication. Details for the embedded multipliers available in Cyclone III and Cyclone IV are depicted in figure 2.1. It is organised in blocks of 9x9 multipliers that can be configured to produce 18x18 multiplications. The DSP blocks in Cyclone V are depicted in figure 2.2. Cyclone V offers extra resources to support 27x27 multiplication, 18x18

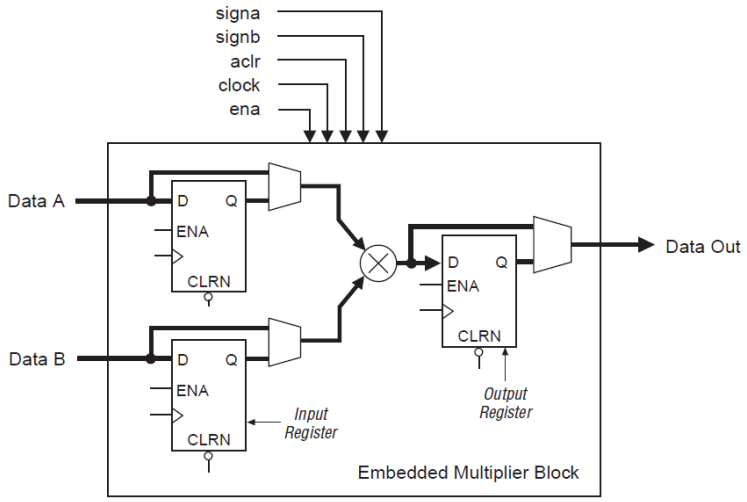

Figure 2.1.: Details of DSP Blocks available on Cyclone III and IV FPGAs from Altera (from [1, 2]).

Device	Size [nm]	Core [V]	Mult 9x9 [MHz]	Mult 18x18 [MHz]
Cyclone III	65	1.2	340	287
Cyclone IV	60	1.2	340	287
Cyclone V	28	1.1	340	287

Table 2.1.: Technology sizes of FPGAs from different Cyclone families and their maximum clock frequencies, for the different configurations of their DSP blocks.

multiply-accumulate and 18x19 complex multiplications (needs 2 DSP blocks). Its architecture is illustrated in figure 2.2. Even though these architectures are different, the underlying multiplier core is the same. When the DSP blocks are configured as 9x9 or 18x18 multipliers, only one multiplier is active, while the adders and the other multiplier remain inactive. The comparison for the multipliers for the different families is summarised in table 2.1. Despite of all devices have a different fabrication process, the manufacturer reports the same top clock frequency for all. For the sake of simplicity, in this work, for all device families the embedded multipliers will be referred to as DSP blocks.

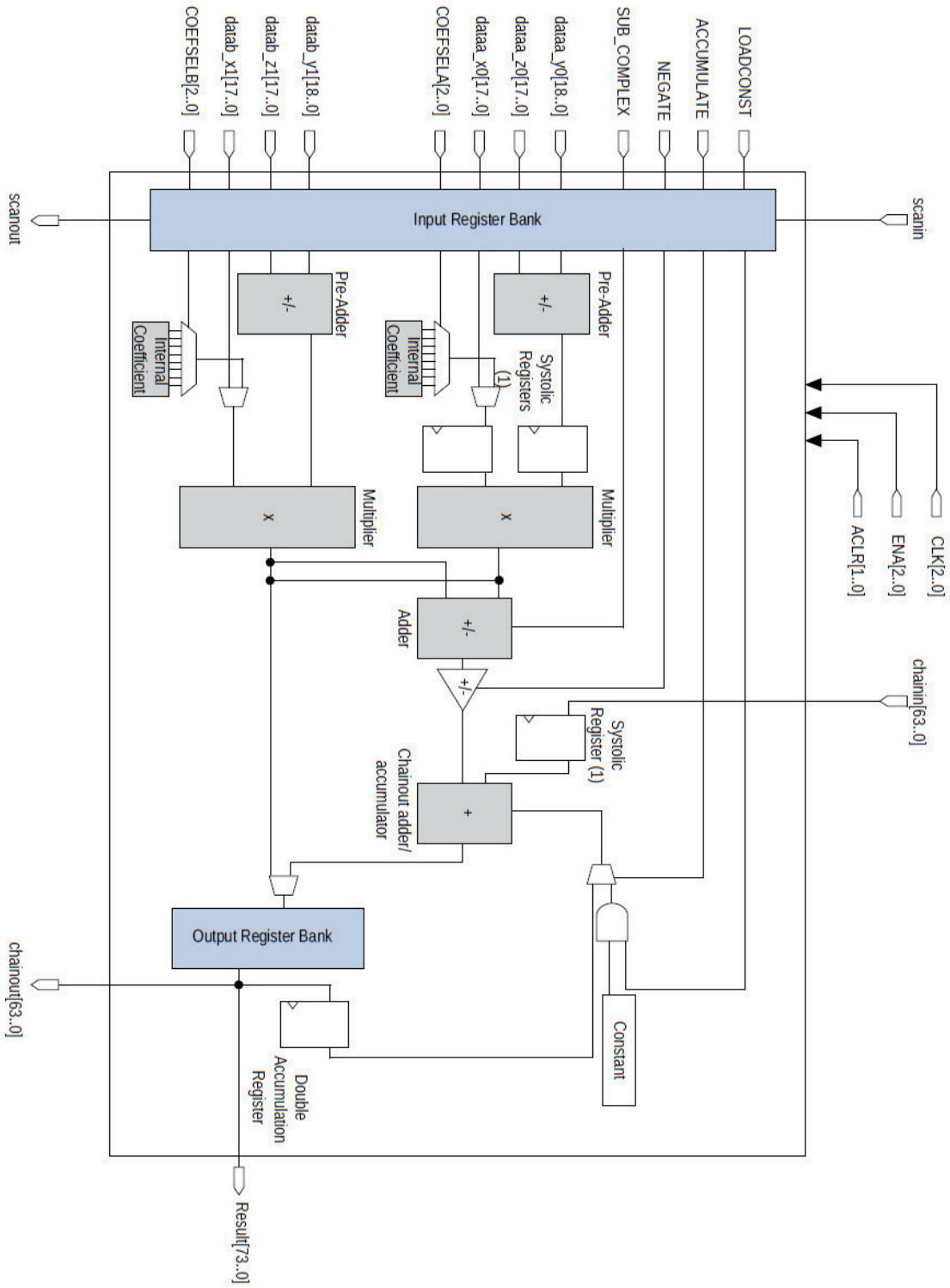

Figure 2.2.: Details of DSP Blocks available on Cyclone V FPGA from Altera (from [3]).

2.2.2. Dot-Product Operator

The dot-product, or inner-product, operator is one of the most widely used in DSP designs and one of the most frequently implemented on FPGAs. The dot-product of two vectors, A and B with n elements each, is represented as $A \cdot B$ and is defined as:

$$A \cdot B = \sum_{i=1}^{n} A_i B_i = A_1 B_1 + A_2 B_2 + \cdots + A_n B_n \tag{2.1}$$

This operation can be described as a sum of partial multiplications. Hence when it is implemented on a system capable of computing many operations in parallel, its scheduling can be defined according to the implementation constraints. From all the scheduling algorithms often used in digital systems [20], for the implementation of the dot-product, the unconstrained minimum-latency scheduling algorithm, also known as *as soon as possible*, and loop folding optimisation techniques were chosen. For clarity, they're referred to as unrolled and rolled designs, respectively.

Unrolled Design

The unrolled design is usually used for its increased performance, through parallelisation of its computations, without any regards for area consumption. In the case of the dot-product, the implementation performs all multiplications in parallel, in the first cycle, and then adds all partial results, in subsequent cycles. The unrolled implementation of the dot-product algorithm increases with the size, and word length, of the vectors to be processed. Hence, the maximum vector size supported depends on the size of the device. The circuit for the unrolled design is showed on figure 2.5. The squares with Z^{-1} identify the registers, and $\lambda_1..\lambda_N$ the different coefficients of the vector that multiply the values from the input vector X. For optimisation of area and performance the multiplications are usually assigned to Constant Coefficient Multipliers (CCMs) instead of generic multipliers.

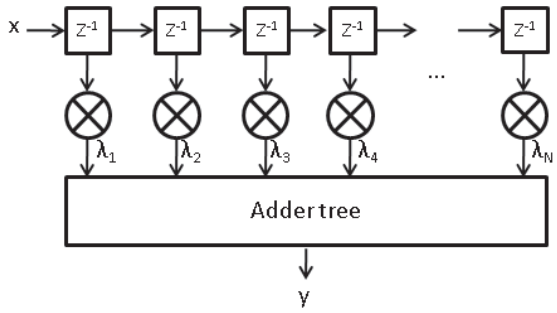

Figure 2.3.: Block diagram of the circuit to do the unrolled implementation of dot-product between two vectors.

Rolled Design

Rolled design is often chosen when hard area constraints are present, or when unrolling of the design is not possible due to the size of the vectors. In this design, the arithmetic units are reused to compute the partial results in each iteration. Thus, multiplications can no longer be implemented with CCMs. In addition, generic multipliers have the drawback of being slower than the CCMs, but the area required to implement doesn't scale with the problem size.

The circuit to implement a rolled design computes the algorithm iteratively, thus it has all of its operations assigned to the same operator in hardware, also known as Multiply-Accumulate. For this reason, the implementation resources of the rolled dot-product algorithm is constant for any vector size. Moreover, the size of the maximum vector supported in this implementation depends on the word length of the accumulator. The circuit for one vector of the rolled dot-product is showed on figure 2.4. In this case the same multiplier is used to produce multiplications for all $\Lambda_1..\Lambda_N$ values with the input vector X. The adder accumulates the N results from the multiplications. This circuit can have the multiplications assigned to Look-Up Table (LUT)-based generic multipliers or more advanced DSP-Blocks which, for some device families, can implement the complete operator.

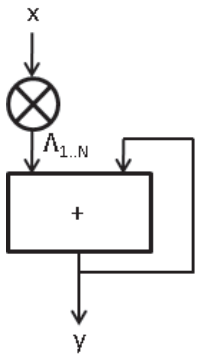

Figure 2.4.: Block diagram of the circuit to do the rolled implementation of dot-product between two vectors.

2.2.3. Linear Projection Algorithm

The linear projection, also known as Karhunen-Loeve Transformation (KLT), or Principal Component Analysis (PCA) [21], often applied in DSP applications such as image processing [22], and is formulated as follows. Given a set of N data $x^i \in R^P$, where $i \in [1, N]$, an orthogonal basis described by a matrix Λ with dimensions $P \times K$ can be estimated that projects these data to a lower dimensional space of K dimensions. The projected data points are related to the original data through the formula in (2.2), written in matrix notation, where $X = [x^1, x^2, ..., x^N]$ and $F = [f^1, f^2, ..., f^N]$, where $f^i \in R^K$ denote the factor coefficients, E the approximation error and Λ the orthogonal basis for the new space with dimensions $P \times K$.

$$F = \Lambda^T X + E \tag{2.2}$$

The original data can be recovered from the lower dimensional space via equation 2.3:

$$\hat{X} = \Lambda F \tag{2.3}$$

The objective of the transform is to find the Λ matrix such as the Mean-Square

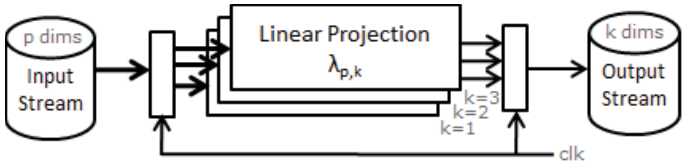

Figure 2.5.: High-level block diagram of the circuit to implement a Z^6 to Z^3 linear projection algorithm.

Error (MSE) of the approximation of the data, defined in 2.4, is minimised, as in equation 2.4:

$$MSE = ||X - \hat{X}|| \tag{2.4}$$

A standard technique is to evaluate the matrix Λ iteratively as described in equations 2.5 and 2.6, where λ_j denotes the j^{th} column of the Λ matrix.

$$\lambda_j = \underset{\|\lambda_j\|=1}{\arg\max}\ \mathrm{E}\{(\lambda_j^T X_{j-1})^2\} \tag{2.5}$$

$$X_j = X - \sum_{k=1}^{j-1} \lambda_k \lambda_k^T X \tag{2.6}$$

where $X = [x^1 x^2 ... x^N]$, $X_0 = X$, $\|\lambda_j\| = 1$ and $\mathrm{E}\{.\}$ refers to mean operator. This process is repeated for all columns of the Λ matrix. An in depth explanation of the algorithm, along with other dimension reduction techniques can be found in [23].

Equation 2.2 can be computed using the previously introduced dot-product, hence it can be implemented on an FPGA using any of the precedent multiply-accumulator architectures. Figure 2.5 shows the block diagram of a circuit to implement a Z^6 to Z^3 linear projection ($P = 6, K = 3$). The original data, in the higher space, is identified with *Input stream* and the factors as *Output stream*. The *Linear Projection* block is replicated K times and consists of an implementation of the dot-product operator, presented in section 2.2.2 of this thesis, with a vector of size P.

Figure 2.6 demonstrates the linear projection algorithm, using an image from a

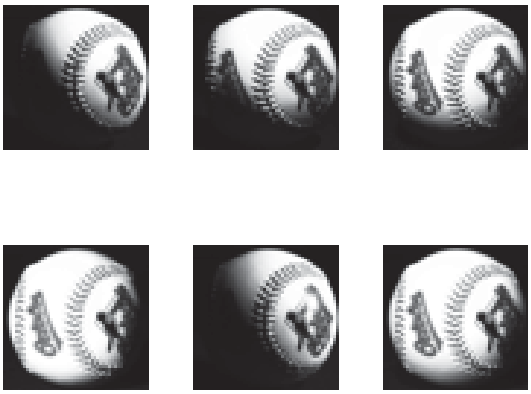

Figure 2.6.: Original data set with 6 images of the same subject under different illumination conditions.

set of 6 images of a ball under different illumination conditions. The original image is in Z^{2500} space and it is projected into Z^{64}, Z^{16} and Z^{2} spaces, using projection coefficients with word lengths ranging from 6 to 9 bits. Figure 2.7 shows the result of the reconstruction of the projection in the original space. The top row shows the reconstruction results for linear projection of a ball image from 2500 to 64 dimensions. The middle and bottom row show the results for 16 and 2 projection dimensions, respectively. The columns show the results for different word lengths of the projection coefficients from 6 bits (on the left column) to 9 bits (on the right column). It is observable, in this example, that for the same word length the images show similar results, even in cases with reduced number of projection dimensions, i.e. 2. It is also noticeable that reduction in the word length of the projection coefficients degrades the quality of the results for any number of projected dimensions.

2.3. Sources of Variation

Variation in the sizes of the physical structures of transistors affects their electrical characteristics across the device, namely threshold voltage, carrier mobility, impedances and current leakages. This type of variation, also known as process

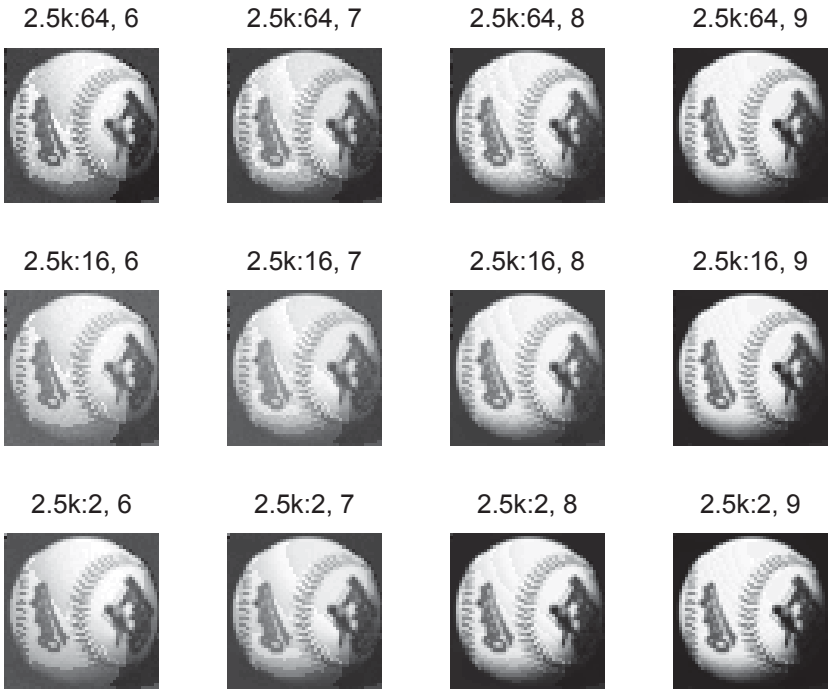

Figure 2.7.: Example of the linear projection algorithm applied to one of the images from the data set using different word lengths (x axis) for the projection coefficients and different numbers of projection vectors (y axis). The images shown are the back-projections in the original space.

Type of Variation	Origin	Path-Delay Increases
Process variation	Fabrication	Process Size reduction
Voltage	Low-power circuits	Power decrease
Temperature	Environment	Temperature increase[1]
Jitter	Fabrication	Placement and routing
Degradation	Aging	Time

Table 2.2.: Summary of the different types of variation, their origin and their contribution for the delay increase in circuit paths.

variation, can be observed on FPGAs through a characterisation of the device by measuring the delay for each basic element on the device. Contributions from [24, 25] show the measurements made for many devices from the same family and how the aging affects their characteristics. Moreover, silicon devices are sensitive to changes in supply voltage, temperature, signal cross talk and jitter [26]. Furthermore, designs targeting implementation on FPGAs endure extra variation from placement and routing.

The preceding variation sources affect the delay of paths in a circuit, thus leading to degradation of the maximum clock frequency for error-free operation. Table 2.2 summarises the most noticeable types of variation and how they influence the delay of the paths in the circuit. From the types of variation shown, process variation and voltage are the ones that have more impact on the performance of devices. The main difference is that process variation is uncontrollable while voltage can be controlled. Further information on the timing models available for Altera FPGAs is described in [27].

2.4. Error Analysis

Throughout this work, error is related to the uncertainty of the outputs of a DSP circuit, derived from erroneous, or non-deterministic, operation of the circuit which

[1] In fabrication processes below 65 nm the opposite effect can be observed as a consequence of temperature inversion. Source: `http://tech.tdzire.com/what-is-temperature-inversion/`

provides results different than the expected ones.

Given the nature of DSP systems, errors derived from violation of a circuit's timing constraints tend to be random due to variations in placement & routing, thermal, voltage and clock. Random errors are measured as the absolute difference between the actual and the expected results, and are characterised through different statistical metrics such as: mean absolute error, variance, standard deviation, root mean square error, and maximum error. All of which are derived from the absolute error observed since the focus in this work is methods to increase the resilience of generic arithmetic blocks.

Given a set of N data points $X_i \in R^P$, where $i \in [1..N]$, expected at the output of a DSP system, $\hat{X}_i$ is the actual data retrieved from the FPGA. The uncertainty can be described using different error metrics, which are defined as follows:

Absolute Error

$$E(\hat{X}) = \hat{X} - X \tag{2.7}$$

Relative Error

$$RE(\hat{X}) = \frac{\hat{X} - X}{X} \tag{2.8}$$

Mean Absolute Error

$$MAE(\hat{X}) = \mathrm{E}[\hat{X} - X] \tag{2.9}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \hat{X}_i - X_i \tag{2.10}$$

Mean Squared Error

$$MSE(\hat{X}) = \mathrm{E}[(\hat{X} - X)^2] \tag{2.11}$$

$$= \frac{1}{N}\sum_{i=1}^{N}(\hat{X}_i - X_i)^2 \tag{2.12}$$

Error Variance

$$Var(\hat{X} - X) = \mathrm{E}[(\hat{X} - X)^2] - (\mathrm{E}[(\hat{X} - X)])^2 \tag{2.13}$$

$$= \frac{1}{N}\sum_{i=1}^{N}(\hat{X}_i - \mu(\hat{X} - X))^2 \tag{2.14}$$

Error Standard Deviation

$$SD(\hat{X} - X) = \sqrt{\mathrm{E}[(\hat{X} - X)^2] - (\mathrm{E}[(\hat{X} - X)])^2} \tag{2.15}$$

$$= \sqrt{\frac{1}{N}\sum_{i=1}^{N}((\hat{X}_i - X_i) - \mu(\hat{X} - X))^2} \tag{2.16}$$

Another common measurement for DSP applications is the Peak Signal-to-Noise Ratio (PSNR) [28], and, given the MSE and the word length (WL), it is computed as follows:

$$power = 1 - \frac{1}{2^{WL}} \tag{2.17}$$

$$PSNR = 10\log_{10}\left(\frac{power^2}{MSE}\right) \tag{2.18}$$

2.5. Variation-Aware Methods for Throughput Increase in FPGAs

Several methods have been proposed to minimise, or recover from, timing errors caused by the aforesaid sources of variation. The different methods operate on different levels of the design, namely placement and routing, Register-Transfer Level (RTL) and algorithm level. This section is dedicated to the different methods usually used, stating their basic functioning, the main benefits and drawbacks.

2.5.1. Variation-Aware Placement and Routing

Variation-aware placement and routing [4, 29] makes use of a model created from a characterisation of the fabric where the design is going to be deployed. The main benefit of this method is to instruct the placement and routing tool [30] to assign the critical-paths to the fastest elements on the device, thus reducing the delay of the most critical-paths, increasing the overall clock frequency of the design. The drawback is the necessity to characterise the design before deploying the design, which is currently only available on FPGAs [31, 32]. Figure 2.8, from [4], illustrates the classification of the variation maps obtained from 129 FPGAs into 16 classes. For all classes, the slowest regions are in red and the fastest ones in blue. It is possible to observe that each class has regions with different delay patterns.

A contribution towards achieving graceful degradation in Finite Impulse Response (FIR) filter designs has been proposed in [33], thus extending the work from [34]. Here, a FIR filter design is optimised by reduction of the common subexpressions in canonical-sign-digit representation. Moreover, it takes advantage of process variation by assigning the least significant filter coefficients to locations exhibiting greater delay. Being slack directly related to the maximum clock frequency, [35] proposes a method to redistribute it taking into account the variability of a device in order

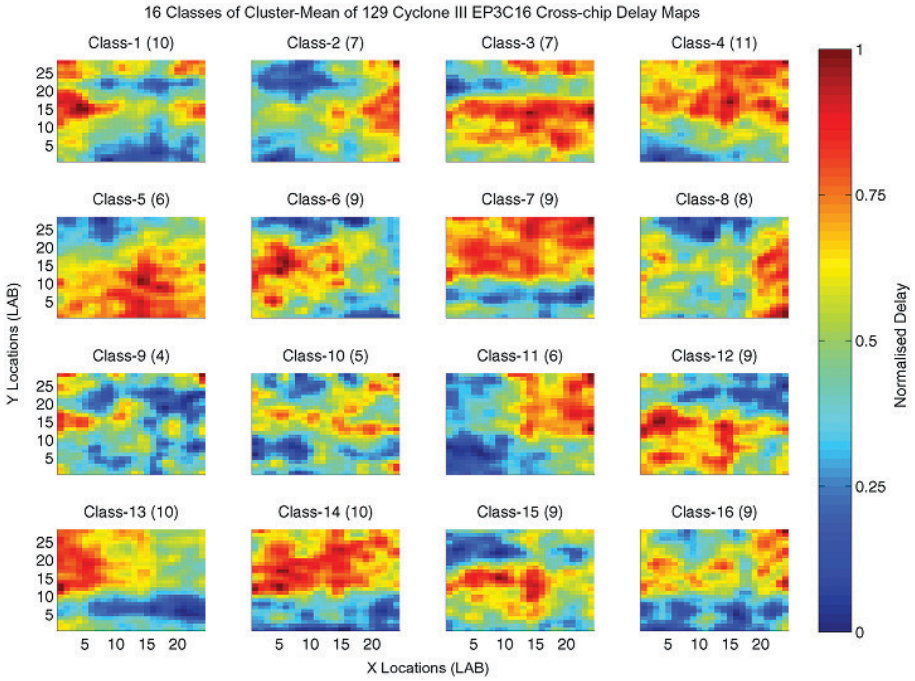

Figure 2.8.: Classes of variation maps with different delay patterns, created from the characterisation of 129 FPGAs (from [4]).

to achieve graceful degradation while targeting low-power designs. [36] addresses the problem of adjusting the timing models taking into account process variation for high-level specification. On a lower-level specification, [29] proposes a method to optimise placement on FPGAs as way to increase performance by 19.3% via chipwise placement to increase the performance of the critical-paths.

Upon identification of errors, [37] uses a "brute force" approach by replacing the faulty unit, by reconfiguring the circuit to use a different one. On a similar note, [38] proposes wear-levelling as a methodology to improve the performance, and thus the reliability, of FPGAs through periodic reconfiguration of the design. The technique places the circuit on different parts of the device to avoid operating in regions that have been subjected to stress due to long time operation. This methodology was able to reduce by 40% the increase in delay due to degradation.

Given that the DSP engineer often oversees low-level details, it may be useful to

pass information about device variation to an higher level in the design. Towards this direction, [39] presents a survey on techniques to mitigate process variation for statistical high-level synthesis specification.

2.5.2. Path-Delay Reduction

In order to increase the clock frequency of a datapath, on the RTL level, typically the DSP designer either reduces its word length or introduces additional pipeline levels. Word length optimisation while offering reductions in area and delay, it also reduces the precision of the results produced. Extensive work has been published on the aforementioned topics, offering techniques to implement and optimise DSP designs [40, 41, 42, 43, 44, 45, 46, 47]. An example of such techniques is one [48] that achieves better results than rounding the word length of filter coefficient using integer linear programming. On a different direction [49] proposes a method to find the most appropriate method for dynamic range estimation of linear time invariant systems as an alternative to interval arithmetic [50] and affine arithmetic [51]. Towards a more automated approach, [52] addresses the impact of word length optimisation in the noise analysis, from a high-level perspective. Nonetheless, pipelining, while preserving the word length and the quality of the results, adds extra latency to the datapath, which can make the implementation require a different algorithm, consuming more resources, or even make it unfeasible by not being able to meet the algorithm's specification [53, 42].

2.6. Error Recovery Methods

Since the early days of computing, engineers have been concerned with faults and errors from different natures [54, 55, 56]. Most of the mechanisms and methodologies proposed to mitigate them rely on extra circuitry and processing time. Usually a

compromise is achieved in terms of the minimum requirements by the application to work, amount of resources and the time to produce results. [57] proposed adaptive computing as a mode of operation where useful computations would be carried out even in the presence of errors by trading off throughput or accuracy. Through time this idea has been developed and recently [58] has proposed a method to adapt the circuit's voltage according to the level of errors, thus trading off power for accuracy. However, some circuits don't admit errors of any kind in their computations. Hence, circuits that require a deterministic output in their calculations have to rely on other methodologies to recover from errors [59], such as Razor [5], which is presented below. Other alternatives for fault tolerance techniques, their benefits and their limitations have been presented in [59] and [60].

2.6.1. Razor

Razor [5] has been proposed as an architecture to recover from timing errors in a datapath. The architecture is presented in figure 2.9. This circuit uses a shadow register to compare the output of the system with a value with a register after a small delay. Whenever the value in the shadow register differs from the value at the output of the main flip-flop, an error flag is activated, to indicate the controlling entity that an error has occurred. In this case, the input multiplexer switches from the output of the previous logic stage to the value stored in the shadow register, thus inserting the correct value in the datapath. The penalty for errors is a stall in the datapath, signaled to the control unit via the output *error*. It should be noted that all the produced results are error-free as long as the timing constraints for the shadow register and the multiplexer are met. This work was extended in [61] to detect and recover from errors due to PVT variation and reduce the soft error rate.

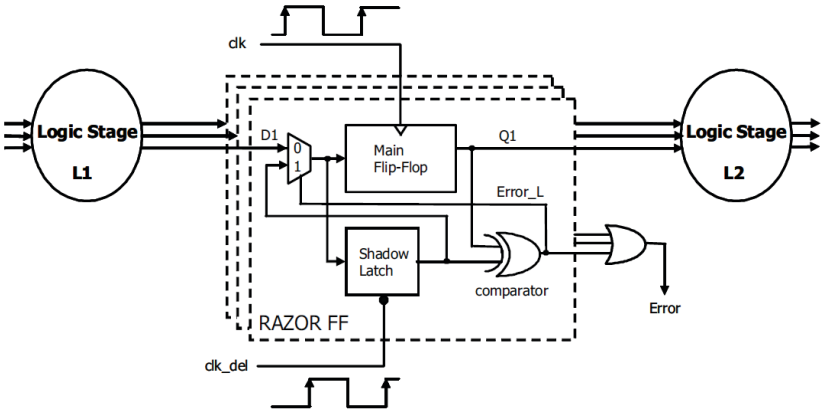

Figure 2.9.: Razor architecture (from [5]).

2.6.2. Reduced-Precision Redundancy

RPR was originally proposed in [11] as a mechanism to contain errors in designs under voltage over-scaling, for low-power, based on the assumption that DSP design can tolerate some errors in their calculations, trading off precision for power [62, 63].

It relies on a smaller version of the original system being computed in parallel using truncated operands. It compares both outputs and verifies if the magnitude of the difference is below a user defined limit (T). The method selects the original output if this result is below the specified threshold, and the approximated result otherwise.

RPR has also been proposed has a method to accelerate DSP circuits [64]. Other works [65, 66, 67] have applied RPR, and small variants, has error mitigation schemes, and have compared them to other error recovering schemes, such as Triple-Modular Redundancy (TMR) [68, 69], in terms of the tradeoff between errors and resources.

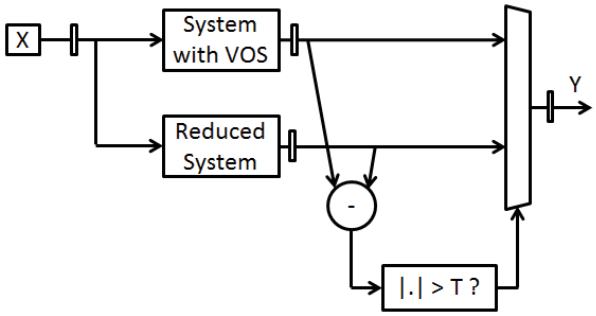

Figure 2.10.: Typical RPR architecture applied to a system under VoS.

Architecture

Figure 2.10 shows the RPR architecture proposed in [11]. It comprises the system under Voltage Over-Scaling (VoS) and a reduced version of that system. The magnitude of the difference between the outputs of both blocks is compared against a pre-defined threshold. In case it is smaller than the threshold value, the multiplexer selects the output from the original system, otherwise it selects the output from the reduced version of the system.

The range of clock frequencies supported by RPR is limited by the delay in the reduced datapath. To reduce the penalty in performance, [70] proposed a combinatorial circuit connected to the Most Significant Bits (MSBits) to replace the threshold comparator. Moreover, another limitation of RPR is the fact that it generates approximate results until the critical path of the approximation, from redundancy, is violated. Beyond the timing constraints imposed by the approximation, the unit isn't able to compute a correct approximation. Thus, in order to increase the throughput, smaller approximations have to be considered even though it means that less accurate approximations are to be passed to the output in case of an error.

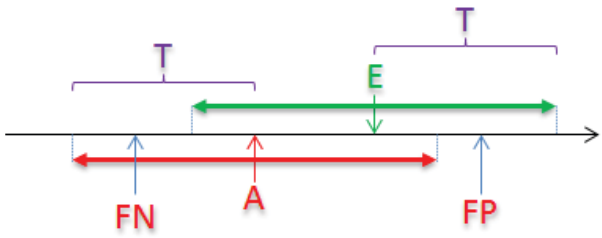

Figure 2.11.: Error-detection boundaries of a RPR system.

Error Analysis

Figure 2.11 shows the error detection boundaries of the reduced system. In this figure A is the approximate results from the reduced precision block; E is the expected (correct) result from the full precision block and T is the modulo of the maximum admissible error. Due to the reduced precision of the operands, the approximation will exhibit a negative bias when compared to the expected result. This is reflected in the detection of errors. To compensate for the negative bias, [11] proposed incorporating an extra circuit, known as Least Significant Bit (LSBit) estimator, inside the reduced block. This blocks assigns a set of LSBits with a value that is closer to the expected value.

As a consequence, in this RPR scheme four possible outputs can happen:

1. No error. The output of the system is the expected result: $Y = E$;
2. False Positive (FP). The output is within the admissible error but has been detected as an error: $Y \in]A + T, E + T]$;
3. False Negative (FN). The output is beyond the admissible error but has not been corrected: $Y \in [A - T, E - T[$;
4. Output within the admissible error: $Y \in [E - T, A + T]$.

Thus, this can generate 3 types of results:

1. expected value;
2. deviated value;
3. approximate value;

From these results, only the second requires attention when estimating the operation of the system when generating deviated values. In all other cases, the result is either the correct result or the approximation.

2.7. Probabilistic Computing

For a long time there has been an interest in producing certain results from uncertain entities [56]. Later a Probabilistic Automata [71] has been formulated has a general case of the deterministic automata. More recently that concern has been restated, establishing a comparison between routing channels in circuits as communication channels prone to errors from the environment noise [72].

When addressing the probability of an arithmetic unit producing always a correct result, there are three main categories of applications:

- don't tolerate errors - e.g. aircrafts, finance;
- tolerate some errors - e.g. multimedia [73], sensors;
- benefit from errors - e.g. quantum physics [74], inference [75, 76].

[77, 78, 79] propose systems that perform speculative computations, based on Complementary Metal-Oxide-Semiconductor (CMOS) devices with probabilistic behaviour, while producing approximate results and benefiting from power-savings or acceleration of computations. Another contribution from [80, 81] proposes a system based on a reliable core and many error-prone cores to accelerate computations. The main limitation in the aforementioned works is the limited scope of applications that

can support probabilistic behaviour in its implementations, as well as control on the level of probability in their operation.

2.8. Resource Optimisation Through Bayesian Inference

In the implementation of DSP designs truncating word length of its operators can often lead to suboptimal results. In this sense, [82] proposed a method to address word length optimisation from a high-level specification of the design. This method, through simulation, iteratively modifies the word length of a signal to find an optimal tradeoff between errors and area.

A method to optimise linear projection implementations through inference was first introduced in [83] and later extended in [84, 6]. Here, the problem is to find a basis matrix that produces the best approximation of the original data, minimising resources and reconstruction MSE of the projected data in the original space. One of the improvements, compared to other works is the avoidance of exhaustive search for solutions.

Being the factors F from a linear projection of data X and the basis matrix Λ, in equation 2.2 unknowns, searching for a possible solution for Λ and F is an ill-conditioned problem, for which solutions from heuristic methods are suboptimal. The framework [6] uses a Bayesian formulation of the factor analysis model instead of the KLT algorithm to find the elements of the Λ matrix that minimise the area cost, rather than a constant area cost consideration in the KLT algorithm. A high-level block diagram of the proposed framework is shown in figure 2.12. The framework receives the problem data, the area models and its parameters and iteratively computes the basis for all projection vectors. As a result it returns the basis matrix and the assignment for the embedded elements.

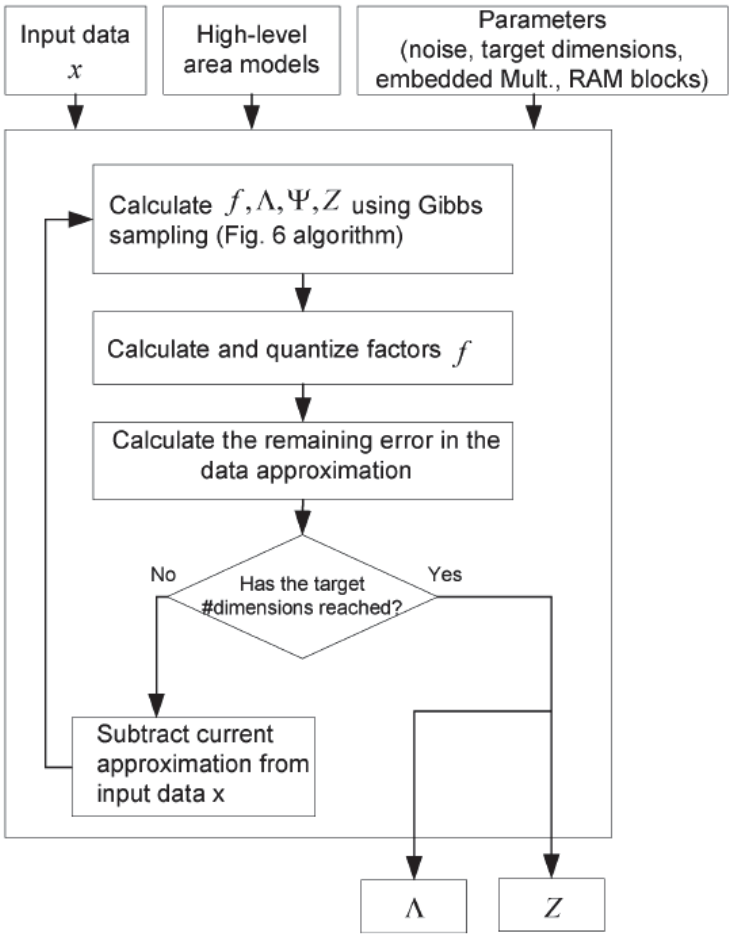

Figure 2.12.: High-level block diagram of the Bayesian framework (from [6]).

2.8.1. Bayesian Factor Analysis Model

Similarly to the KLT algorithm, the factor analysis model states that the original data in higher dimensional space is a linear combination of the factors, in a smaller sub-space plus an error term:

$$x^i = \Lambda f^i + \epsilon^i \tag{2.19}$$

where x corresponds to N instances of vectors P dimensions, and f to N instances of vectors with K dimensions. i belongs to $[1..N]$. Λ is the unobserved basis, or factor loading, matrix with size $P \times K$. This formulation assumes that the original data is centralised, therefore the mean is not considered.

The factor analysis models assume that the error terms ϵ^i are independent and multivariate normally distributed with zero mean and covariance matrix Ψ written as:

$$\epsilon^i \sim \mathcal{N}(0, \Psi) \tag{2.20}$$

It also assumes that the probability distribution of x for each observed case i has a multivariate normal density:

$$\begin{aligned} p(x^i|f^i, \Lambda, \Psi) &= \mathcal{N}(x^i|\Lambda f^i, \Psi) \\ &= (2\pi)^{-P/2}|\Psi|^{-1/2} \times \exp\left(-\frac{1}{2}{\epsilon^i}^T \Psi^{-1} \epsilon^i\right) \end{aligned} \tag{2.21}$$

where ϵ^i is the error in the approximation data: $\epsilon^i = x^i - \Lambda f^i$. It can be written in matrix notation as:

$$\begin{aligned} p(X|F,\Lambda,\Psi) &= \mathcal{N}(X|\Lambda F,\Psi) \\ &= (2\pi)^{-P/2}|\Psi|^{-1/2} \times \exp\left(-\frac{1}{2}\operatorname{tr}\left[E^T\Psi^{-1}E\right]\right) \end{aligned} \tag{2.22}$$

where $E = X - \Lambda F$.

The factors, as a result of a linear projection, are assumed to be normally distributed with zero mean and covariance matrix Σ_F:

$$f^i \sim \mathcal{N}(\Sigma_F) \tag{2.23}$$

The posterior probability of the factors is given by equation 2.24.

$$p(f^i|x^i,\Lambda,\Psi) \propto p(f^i)p(x^i|f^i,\Lambda,\Psi) = \mathcal{N}(f^i|m_F^*,\Sigma_F^*) \tag{2.24}$$

in which the posterior mean and variance are:

$$\begin{aligned} \Sigma_F^* &= (\Sigma_F + \Lambda^T\Psi^{-1}\Lambda)^{-1} \\ m_F^* &= \Sigma_F^*\Lambda^T\Psi^{-1}\Lambda x^i \end{aligned} \tag{2.25}$$

The complete density is achieved by integrating F from (2.22) producing:

$$\begin{aligned} p(X|\Lambda,\Psi) &= \mathcal{N}(X|\Lambda\Sigma_F\Lambda^T + \Psi) \\ &= (2\pi)^{-N/2}|\Lambda\Sigma_F\Lambda^T + \Psi|^{-1/2} \\ &\times \exp\left(-\frac{1}{2}\operatorname{tr}\left[X^T(\Lambda\Sigma_F\Lambda^T + \Psi)^{-1}X\right]\right) \end{aligned} \tag{2.26}$$

The complete density of the data is given by a normal distribution with covariance $\Lambda\Sigma_F\Lambda^T + \Psi$.

The probability distribution for the Λ matrix is shown in equation 2.27, under the assumption that all λ_{pk} values are independent.

$$p(\Lambda) = \prod_{p=1}^{P} \prod_{k=1}^{K} p(\lambda_{pk}) \tag{2.27}$$

The prior probability distribution is related to the inverse of the area required to implement each value of λ_{pk} (2.28). Hence coefficients which require less resources are more likely to be sampled.

$$p(\lambda_{pk}) \propto (A(\lambda_{pk}))^{-1} \tag{2.28}$$

Therefore the posterior for each λ_{pk} is:

$$p(\lambda_{pk}|X, F, \Psi) \propto p(X|F, \lambda_{pk}, \Psi) \prod_{p=1}^{P} \prod_{k=1}^{K} p(\lambda_{pk}) \tag{2.29}$$

Gibbs sampling algorithm [85] is then employed to draw samples from the posterior distribution.

2.9. Summary

This chapter revisits some of the most important concepts present in the design of DSP designs for FPGAs focused on performance optimisation, and also methods to mitigate timing errors, or to achieve graceful degradation.

Highly efficient arithmetic blocks, i.e. DSP, present in modern FPGAs offer the possibility to implement DSP designs with greater performance while still benefiting from flexibility. Still, in some smaller devices they are a scarce resource, thus being required to implement part of the circuit with LEs. In this sense depending on the implementation strategy (i.e. throughput, area, power), designs may have different

requirements, thus different implementations, as it has been demonstrated for the dot-product operator, one of the most frequently used in DSP designs.

A method to increase the throughput of circuits in silicon devices, while assuming zero-latency penalty, is to over-clock them beyond the limits specified by the synthesis tools, at the expense of operating in error-prone regimes dominated by variation of the operating conditions. The main sources of variation include process, temperature, voltage and aging of the device.

Alternative methods to increase performance are variation-aware methods. They offer a limited increase in performance by exploiting the physical variations in the device. Further increases in performance are likely to produce timing errors, that are often manifested in the MSBits, therefore mitigation techniques are proposed to recover from, or minimise, errors, usually at the expense of time and resources.

Probabilistic computing has been proposed as an alternative to mitigation methods, without using extra resources, where the applications can tolerate errors and still produce approximate results, while benefiting from power savings, or performance increase.

A previous optimisation framework to efficiently map linear projections on heterogenous FPGAs suggested that a similar approach could reduce timing errors, by using different sources of information, such as errors from variation of the operating conditions, in the optimisation process.

Table 2.3 summarises the different techniques that can be applied to minimise, or minimise, timing errors from Table 2.2. For each technique it shows how it actuates and its limitations.

The research conducted in this thesis considers the problem of accelerating DSP designs under variation that are sensitive to deep pipelining and word length optimisation, which are the typical solutions. In order to have DSP designs, without changes in their algorithms, operating with higher throughput under variation, an

Technique	Effect	Limitations
Deep pipelining	Break the critical-path by inserting registers	Unsuitable for some streaming algorithms, or algorithms using recursion
Word length optimisation	Reduce the critical path by processing less bits from truncated operands	Penalty in the quality of results
Razor [5]	Check if the output matches the shadow register	Temporal redundancy is unsuitable for streaming applications
Reduced-Precision Redundancy [11]	Check if the error is within a threshold	Requires extra latency. Unsuitable for algorithms using recursion

Table 2.3.: Summary of the existing techniques for acceleration of computations, in order to mitigate timing errors, and their limitations.

adaptation of a methodology for area optimisation is investigated. Moreover, from all error mitigation techniques considered, RPR was the one that had the potential to be adapted to the aforementioned requirements. Hence, this thesis proposes modifications in its architecture to eliminate extra latency imposed by the existing implementations. In a nutshell, this research work investigates alternatives to reduce, or mitigate, timing errors in applications that tolerate some errors in its calculations, and proposes methods to close an existing gap in this research area.

3

Performance of Arithmetic Units Under Variation

3.1. Introduction

Evaluation and analysis of the maximum performance of a DSP system, and comparison against the limits set by the synthesis tools, has always been a subject under continuous investigation in order to maximise the throughput of such systems. Usually device manufacturers limit the top performance of the designs in their devices to account for degradation due to aging. Moreover, the increased variation in the performance of transistors, and consequent uneven performance across

a device offers venues to reap extra performance from it. Therefore, performance characterisation is an important process in the design and optimisation of digital systems, as it assesses their maximum operating conditions, and their behaviour when operating beyond those limits.

The operation of designs is also impacted by the operating conditions, external to the device, such as voltage and temperature. This is particularly critical when different design strategies are targeted, such as low-power or high-performance, which change the design's operating limits. Moreover, when operating in the error-prone regime, arithmetic units exhibit correct operation depending on the characteristics of the data they're processing. The present chapter addresses this subject by proposing an architecture to characterise arithmetic units, and elaborates a study on the performance of the most common arithmetic units present in DSP systems under variation of the operating conditions of the device.

Examples of platforms to test the critical-paths of circuits on FPGAs can be found using: ring-oscillators [31], slack measurement [86] and transition probability [32]. Most of these works are focused on exercising the critical-paths, instead of assessing the impact of timing errors in the application results. In addition, an arithmetic unit could be optimised to operate with data sets with specific properties (i.e. distribution, mean and variance) with less, or no, errors at a higher clock frequency than the worst-case would determine.

Since there's no contribution available which fulfills this requirement, this work proposes a new characterisation framework, based on prototypes from [16, 17, 19], for the characterisation of arithmetic units on FPGAs. FPGAs were chosen given their ability to be reconfigured, which concedes the possibility to characterise arithmetic units on the device and later reconfigure the device to implement a complete design using information from that characterisation stage.

This chapter is devoted to the characterisation of arithmetic units operating un-

der different operating conditions, producing statistics, and analysing how errors appear with the change of the same operating conditions. This chapter also covers the technical details of the characterisation framework and it shows results for the characterisation of some of the most used arithmetic units in DSP. Unless otherwise stated, throughout this research work the default operating conditions for any given device are 1200 mV supply voltage and 20°C on its surface. Details on the control of the operating conditions (i.e. voltage and temperature) can be found in appendix A.2 of this thesis.

3.2. Characterisation Framework

3.2.1. Introduction

The aim of the proposed characterisation framework is to capture the errors at the output of an arithmetic unit when placed on various locations on an FPGA, clock frequencies, placement and routing configurations, temperatures and voltages. Therefore, a characterisation of the design under test can be achieved and the collected information can be utilised by other design frameworks. Such an approach is possible only due to the reconfigurability that is offered by FPGA devices. As it was mentioned previously in earlier contributions [16, 17, 19], prototypes of frameworks for the characterisation of CCMs and generic multipliers were introduced. The present framework extends all previous contributions as it supports a wider range of designs under test and enhances its functionality.

A key element of the presented work is the performance characterisation circuit of an arithmetic unit under various word length supports, locations, clock frequencies, placement and routing configurations, and devices. The supporting modules are independent of the design under test, and thus the proposed framework can be utilised for any arithmetic components.

The characterisation framework incorporates hardware and software blocks. The hardware block is connected to the arithmetic unit under test and is responsible for the low-level control of the process. Moreover, it provides a constant stream of data to the unit, and collects the generated data from it. The software block offers an interface between the host computer and the characterisation circuit, and it is responsible for the configuration of the device, downloading of the input stimulus to the characterisation circuit, collection of the data from the circuit, setting of the clock frequency and initiation of the test.

3.2.2. Circuit Architecture

A schematic of the architecture of the characterisation circuit is shown in Figure 3.1. It is composed by the following modules: *Input stream* Block RAMs (BRAMs), the unit under test, the *Output stream* BRAMs, a Phase-Locked Loop (PLL) to set the clock frequency and the Finite State Machine (FSM) to control the circuit.

As the objective is to capture the performance of an arithmetic unit under various operating conditions beyond the limit reported by the synthesis tools for error-free operation (i.e. over-clocking), when it is actually placed on the FPGA device, two clock domains are supported. The PLL generates the necessary clock signals as follows: the *datapath clk* clock is used for the unit under test (i.e. generic multiplier), where the *FSM clk* drives the FSM, BRAMs, and other supporting modules.

The fact that BRAMs are used to store the input streams, rather than implementing a pseudo-random generator, such as an Linear Feedback Shift Register (LFSR), offers the possibility to test the circuit with specific input vectors. The down side is that data transfers to the device are time consuming.

This circuit implemented on a Cyclone III FPGA requires 1034 LEs and 13 BRAMs with 32 bits word length and 2048 addresses, and its FSM can be clocked up to 910 MHz while operating without errors, to characterise an 8-bit arithmetic unit.

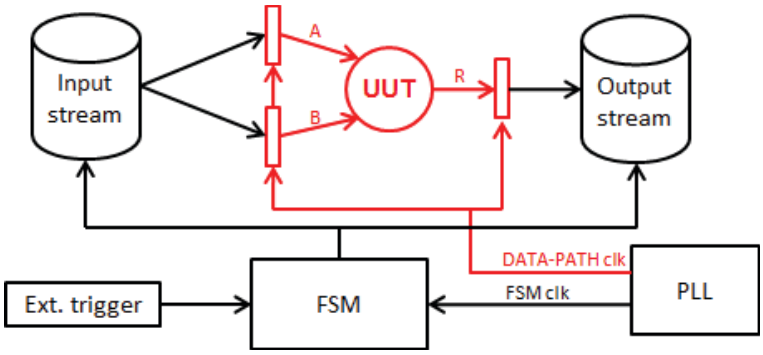

Figure 3.1.: Architecture of the circuit for the characterisation of arithmetic units.

The user is responsible to run the test below this limit. The characterisation circuit operates at high clock frequencies, therefore it required careful placement of the components in order to guarantee that the unit under test holds the critical-paths. The floor plan for a characterisation circuit, including the LUT-based generic multipliers under test is shown in figure 3.2. In this example the framework does the characterisation of 3 multipliers simultaneously. Their regions are identified as “mult0..2”. The other regions are designated for the supporting blocks and are identified with PLL and FSM. It should be noted that the characterisation framework has built-in checks on the correct functioning of the test circuit, namely assurance that the PLL is in locked state during the test execution, and also checks the data retrieved from the device for error conditions.

3.2.3. Characterisation Process

The whole characterisation process has been automated and the transfer of the data from and to the FPGA device takes place through the Joint Test Action Group (JTAG) interface. In essence, the configuration of the FPGA, the loading of the characterisation stimulus and characterisation results are transferred via a JTAG interface between the host and the FPGA. After the characterisation circuit bitstream is loaded into the FPGA, the host computer loads the characterisation

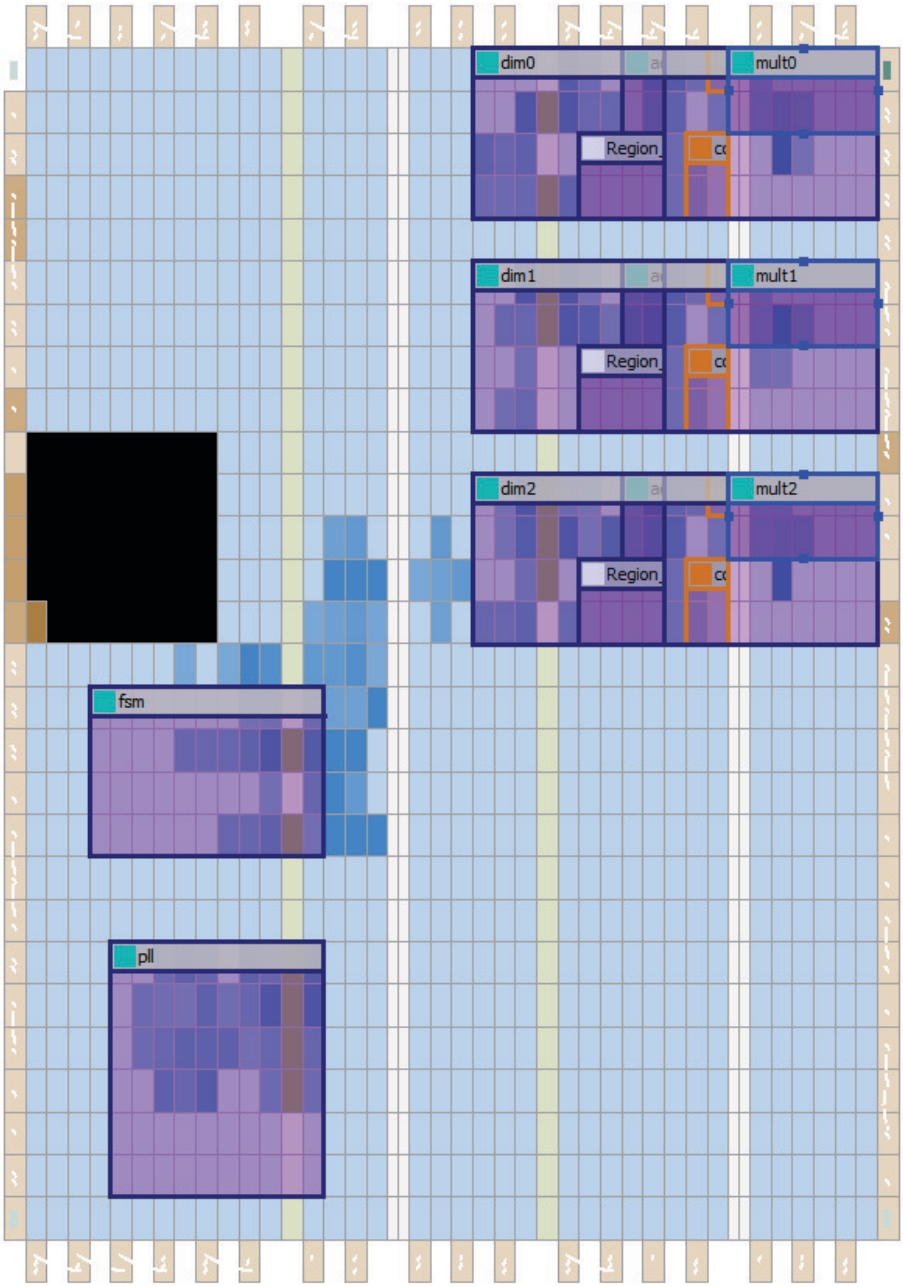

Figure 3.2.: Floor plan of the characterisation circuit for 3 multipliers on a Cyclone III 3C16 device from Altera.

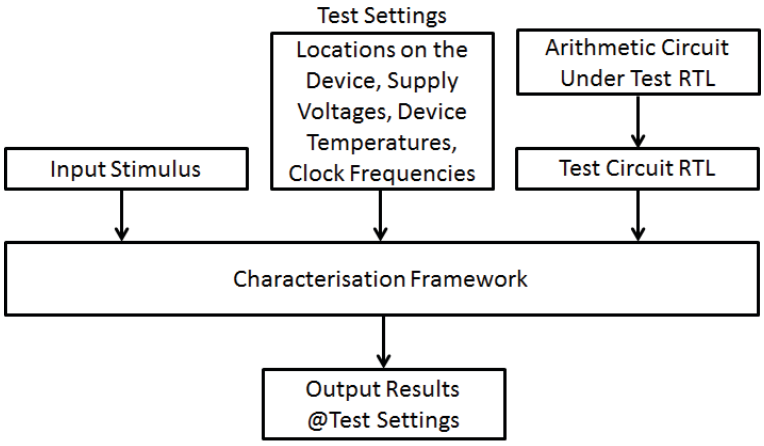

Figure 3.3.: High-level flow of the characterisation framework.

stimulus in the *Input stream* BRAMs. The FSM transfers the data from the input BRAMs into the registers at the input of the unit being characterised, performs the characterisation, and finally copies the data from the output registers of the multiplier into the *Output stream* BRAMs. The FSM stops after it reaches the end of the *Input stream*. Once the characterisation process completes, the results are retrieved by the host computer. This sequence of actions is repeated for a specified range of clock frequencies, supply voltages, locations and temperatures. Figure 3.3 shows the inputs and outputs of the characterisation framework. The inputs are the stimulus data, operating conditions and the RTL of the unit under test. The output of this framework is the actual output of the unit under test for the given operating conditions. Moreover, the framework can produce any statistics on the data observed.

It's important to state that special care has been given to the design of BRAMs interface and to the rest of the supportive modules to ensure that the critical path is always within the design under test. Thus, the frequency limit of the supportive modules is well within the region where the design under test generates erroneous results, for the framework to be able to capture the performance characteristics

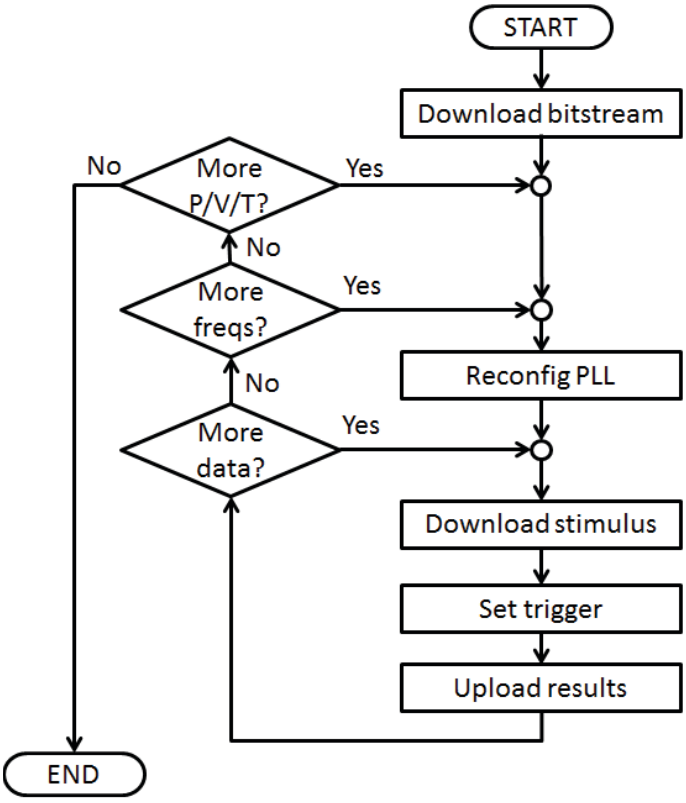

Figure 3.4.: Flow chart of the actions executed by the TCL script to control the characterisation circuit on the FPGA.

under these cases too.

3.2.4. Software Support

The interactions between the test circuit and the host computer are supported by a Tool Command Language (TCL) interpreter from the device vendor. For this particular application a script was written to handle the characterisation process. The flowchart in figure 3.4 illustrates the actions executed by the TCL script to interact with the characterisation circuit on the FPGA.

The characterisation process starts with the download of the bitstream with the characterisation circuit with the unit under test in place. In case the unit under

test requires extra configuration it is downloaded to the FPGA. After that the PLL is configured to produce the clock signals for the FSM and the datapath. Once the clock frequency is settled, the application sends the test stimulus to the FPGA, then it sends the trigger signal to initiate the test, and finally uploads the results. This process is repeated for all blocks of data, supply voltages, device temperatures, locations on the device and clock frequencies.

Actions involving data transfers, namely PLL reconfiguration, data stimulus and results, use Intel Hex files [87]. The contents of these files follow the organisation of the memories where they're used. In the case of the stimulus data, in order to sustain the high-throughput required by the unit under test, two values for each operand are retrieved from each memory address, and then pipelined into the *input stream*. Figure 3.5 depicts the data organisation for the input stream BRAM. The *Output stream* uses a similar organisation, instead it holds results with word lengths up to twice the word length of the *Input stream*, which is sufficient to cover the word length of the arithmetic operators considered. On the host computer side, these files are processed by Matlab scripts, created to make the translation between them and the workspace variables.

3.3. Performance of Arithmetic Units Under Variation

In this work, Cyclone III, IV and V devices from Altera were used. The developed framework is very lean regarding its requirements and can be executed to any available board/device with some small changes. The purpose is to push the performance limits, for a linear projection design, through over-clocking. The tests conducted targeted adders and multipliers as they often are the computational bottleneck in DSP designs. All units considered in the characterisation use fixed-point unsigned representation. In the characterisation of the multipliers one of the inputs

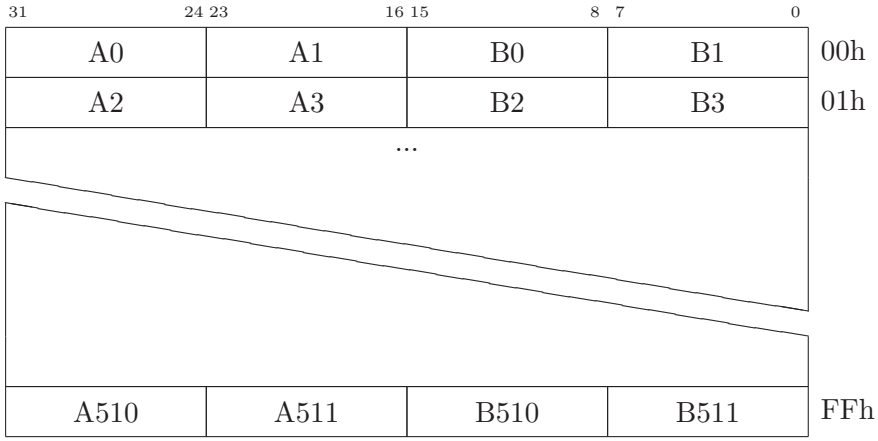

Figure 3.5.: Memory organisation for the BRAM, in the characterisation circuit, holding the input stream data.

was enumerated through all possible values, (i.e. all possible values for a given word length), where the other input was stimulated by a pseudo-random stream of input data following a uniform distribution. The results at the output of the multipliers were compared against the expected results. The temperature of the device was controlled through the use of a cooling element on top of the FPGA. The multipliers were synthesised multiple times at multiple locations in the device, with different placements and routings, to marginalise the performance of the generic multiplier over many locations on the device. The characterisation process was repeated for many boards from the same device family and different process sizes in order to evaluate the impact of inter and intra-die process variation. On this account the DSP-based multipliers were used as their placement and routing is constant for all locations in all devices, regardless of the synthesis options.

3.3.1. Adder

Adders are the arithmetic building blocks of any computational system, therefore they're in the interest of the performance characterisation work. The motivation

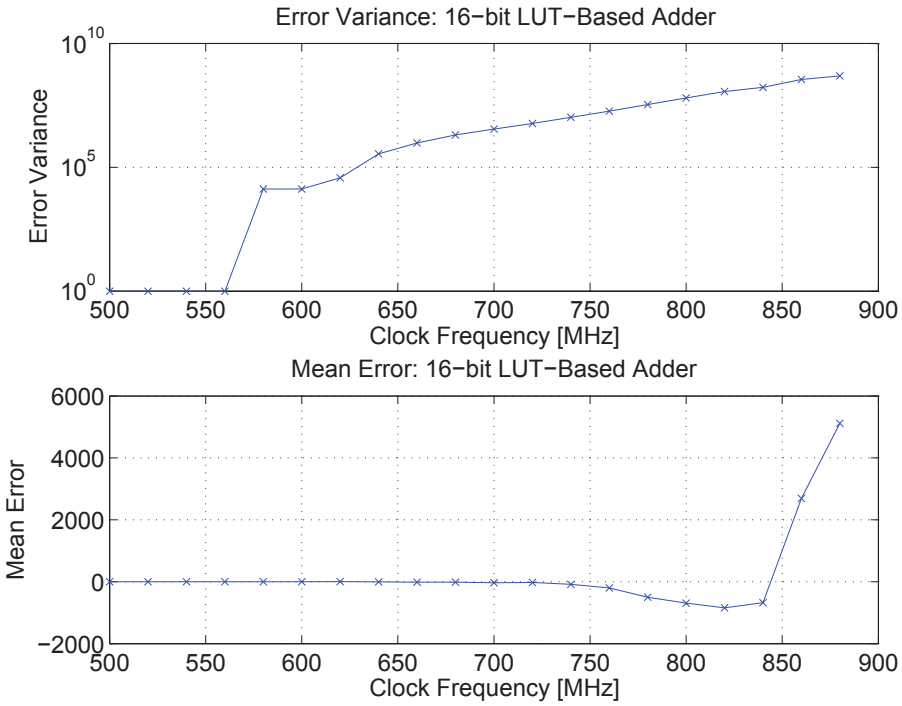

Figure 3.6.: Error variance and mean error for a 16-bit unsigned adder on a Cyclone III FPGA operated at 1200 mV, 20°C tested over a range of clock frequencies.

for this characterisation is to envisage a model of the performance of the unit when operating above the limits specified by the synthesis tool. In this study, the 16-bit unsigned adder was chosen to be tested with 20k pseudo-random uniform samples.

Figure 3.6 depicts the error variance and mean error for the aforementioned 16-bit adder at clock frequencies way above the limit specified by the synthesis tool, 411.18 MHz. It shows that up to 560 MHz, approximately 150 MHz above the limit from the synthesis tool, the adder provides correct results.

Figure 3.7 shows the histogram of the errors observed on the adder at 700 MHz. The magnitude of the errors observed is represented in $\log_2$ scale as it is more suitable to identify which bits contributed to the errors.

An investigation on the list of the 15 most critical-paths of the adder reveals,

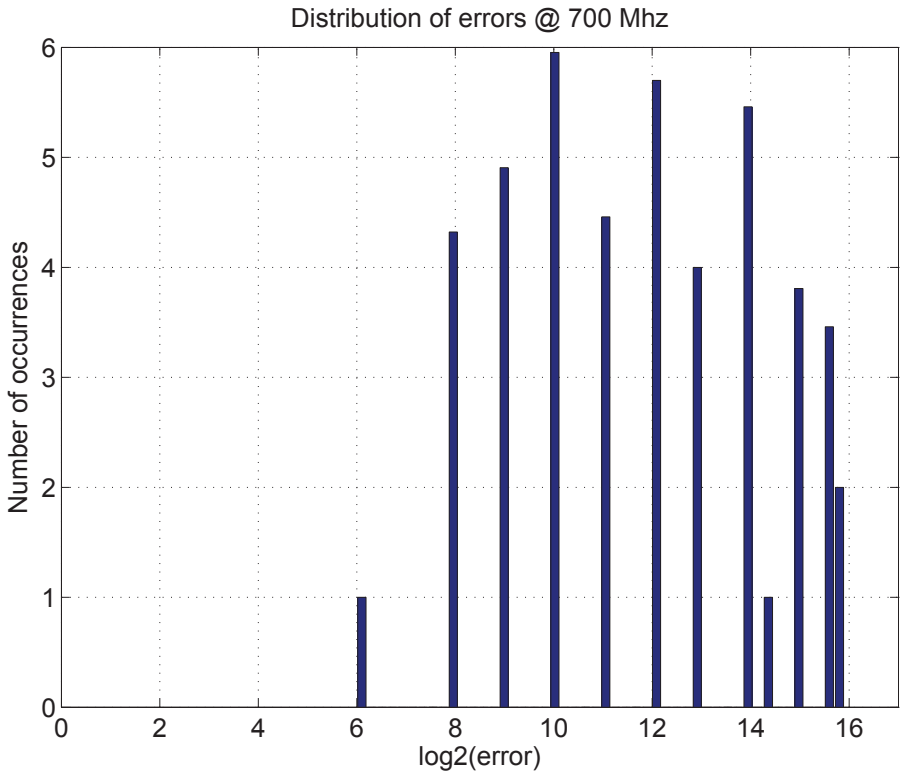

Figure 3.7.: Histogram of the magnitude of errors, in $\log_2$ scale at the output of a 16-bit unsigned adder at 700 MHz.

Slack	From	To	Relationship	Clock Skew	Data Delay
0.069	inB[0]	out[15]	2.5	-0.064	2.362
0.100	inA[0]	out[15]	2.5	-0.064	2.331
0.112	inA[1]	out[14]	2.5	-0.064	2.319
0.114	inB[1]	out[14]	2.5	-0.064	2.317
0.118	inA[1]	out[15]	2.5	-0.064	2.313
0.120	inB[1]	out[15]	2.5	-0.064	2.311
0.173	inB[0]	out[14]	2.5	-0.064	2.258
0.185	inB[0]	out[13]	2.5	-0.064	2.246
0.185	inA[2]	out[15]	2.5	-0.064	2.246
0.192	inA[0]	out[14]	2.5	-0.064	2.239
0.206	inA[3]	out[14]	2.5	-0.064	2.225
0.207	inB[3]	out[14]	2.5	-0.064	2.224
0.212	inA[3]	out[15]	2.5	-0.064	2.219
0.213	inB[3]	out[15]	2.5	-0.064	2.218
0.216	inA[0]	out[13]	2.5	-0.064	2.215

Table 3.1.: Top 15 most critical-paths from the slow model at 1200 mV 85°C for a 16-bit adder.

as expected, the MSBit occupy the top of the list, with the exception of the carry bit (bit 16). It is also observable that these paths have very similar delay, 0.147 ns, meaning that they have a similar critical operating clock frequency. In this example, the gap in maximum clock frequency for the top 15 critical-paths is less than 30 MHz.

Assuming that the timing models from the synthesis tool are shifted in clock frequency, it is possible to relate the test results and the path delays inside the adder. Figure 3.8 illustrates the aforementioned by representing in blue the error variance of the 16-bit adder for different clock frequencies, and in green, the total number of critical-paths with the corresponding delay. The initial value is set to the first clock frequency, 550 MHz, with error variance greater than zero. In more details, the tool specifies a maximum clock frequency of 411 MHz, while the characterisation test has shown that the maximum clock frequency to operate without errors is 560 MHz, thus there's a gap of 149 MHz in clock frequency.

In the particular case of a test at 700 MHz, it corresponds to a clock frequency of

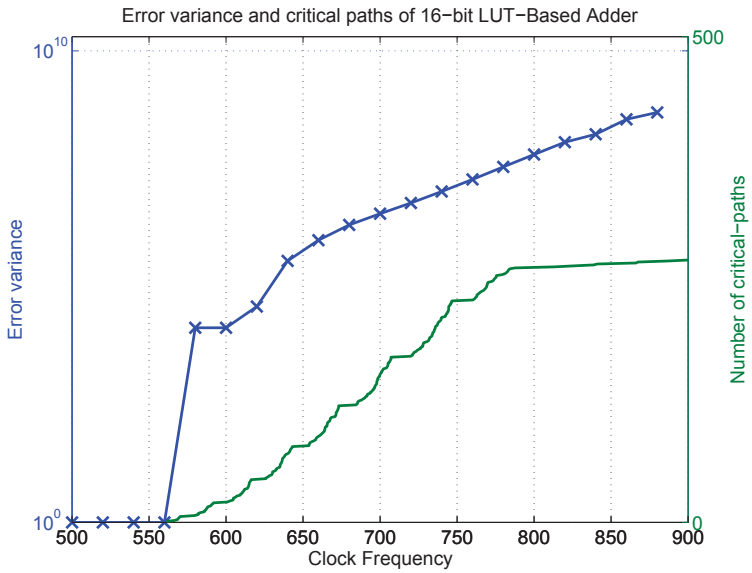

Figure 3.8.: Error variance on a 16-bit unsigned adder, on a Cyclone III FPGA, at different clock frequencies (blue), and the total number of critical-paths with the corresponding critical clock frequency (green).

551 MHz in the timing model. For this clock frequency, the model holds 153 paths with delay greater than the clock period. Figure 3.9 shows the number of critical-paths per output bit of a 16-bit unsigned adder, in blue. This figure also shows, in red, the top 150 critical-paths. It should be noted that a large increase in the error variance is observed within a small number of critical-paths, after which it evolves at a slower rate.

Taking the above into account, it's not possible to establish a relation between the operating conditions, the errors and the architecture of the 16-bit adder. Moreover, as soon as this arithmetic unit leaves the error-free regime, it produces errors with error variance nearly half of the maximum observed in the highest clock frequency considered. Hence, it can be concluded that the architecture of this unit isn't suitable to achieve graceful degradation. Future work in this matter could prove to be beneficial, such as studying and characterising other adder architectures.

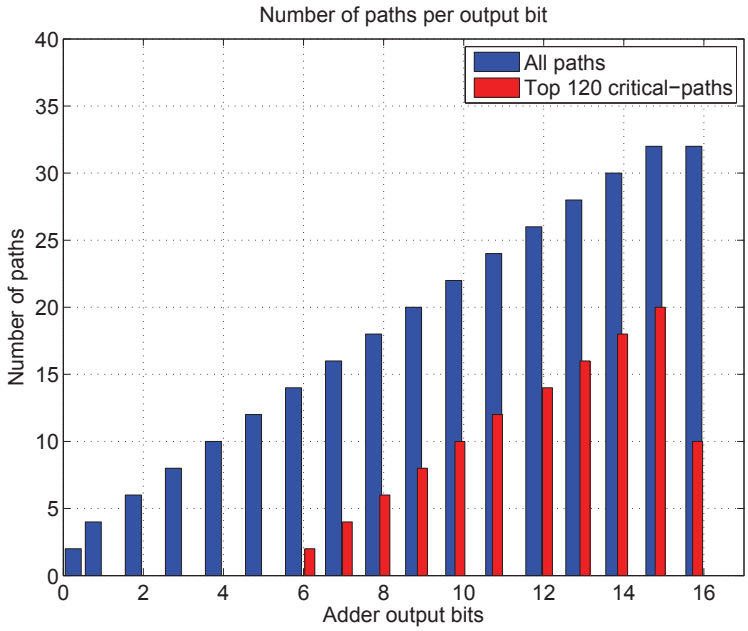

Figure 3.9.: Number of critical-paths per output bit of a 16-bit unsigned adder, in blue. The top 120 are presented in red.

3.3.2. Constant Coefficient Multiplier

CCMs are often used in DSP designs as they offer reduced area and high-throughput in operations were the constant multiplicands are defined in the specification stage of the circuit, and don't need to change over time. In this section the characterisation of the CCMs was performed for all constant coefficients for 8-bit unsigned CCMs on the FPGA. In this characterisation test the temperature of the device was left uncontrolled and the supply voltage was set to 1300 mV. Each constant coefficient was tested with a pseudo-random sequence of 6k 8-bit unsigned samples with a uniform distribution. The motivation to use such vector length and word length is to make the evaluation similar to many DSP applications [88]. The motivation to use unsigned representation is to prevent penalisation of the negative values under two's complement representation.

Figure 3.10 shows the result for the characterisation of all possible 8-bit unsigned

CCMs at 510 MHz. Each column presents a different metric, namely: constant coefficient value, circuit area, error variance, mean error and the absolute error Hamming distance. In the top row, the values in all columns are sorted by the value of the constant coefficient. In the following rows the metrics sorted by: area, error variance, mean error and Hamming distance.

Taking all into account it's not possible to conclude that there's a direct relation between the error metrics and any other metric. Nevertheless, it's observed that constant coefficients with smaller area, and error Hamming distance, present smaller error variances, and almost all constant coefficients with greater area presented greater error variances. This assertion is upheld by the plots on the 2nd and 5th rows, values sorted by area and Hamming distance, where the values with small area exhibit no error variance, neither mean error. On the other hand, in the 3rd row, where values are sorted by the error variance, the values with the greater error variance also show the largest area.

Figure 3.11 shows the magnitude of error variance for the different CCMs tested twice on the same Cyclone III FPGA. The colour in each cell is related to the magnitude of the error variance. It is observable that the coefficients with the smallest Hamming distance only show error variance at the top clock frequencies. It is also observable that some constant coefficients exhibit different levels of error variance for the same clock frequencies, even though there are no changes in the multiplier circuit, i.e. around coefficient 100 and 120 at 540 MHz. This observation can be justified as a manifestation of variation in temperature, jitter, cross-talk, self-heating, noise, or a combination of any of these. The scale in the figure is normalised to the maximum error variance observed.

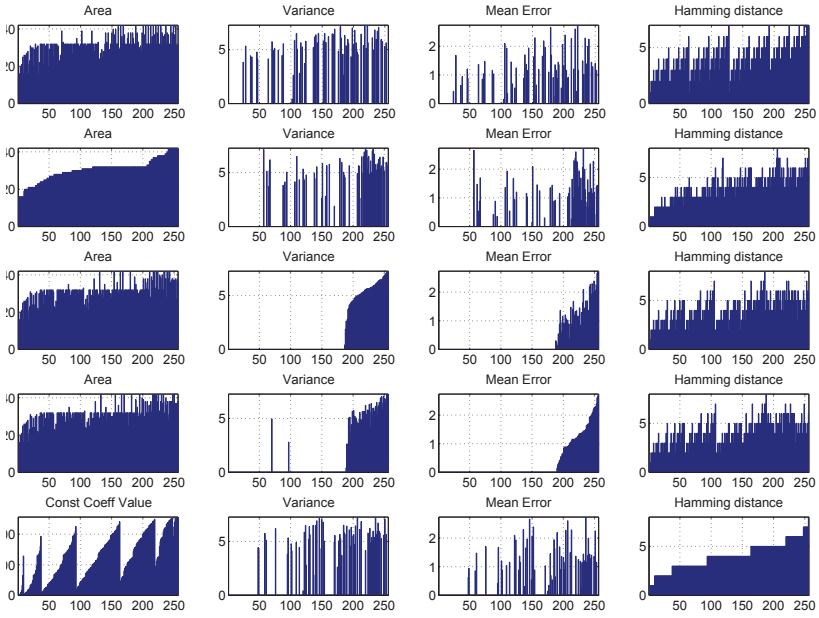

Figure 3.10.: Statistics for errors of CCMs sorted by different metrics: CCM value (1st row), area (2nd row), error variance (3rd row), mean error (4th row) and Hamming distance (5th row).

Figure 3.11.: Variance of constant coefficients for CCMs tested twice on a Cyclone III FPGA under different clock frequencies.

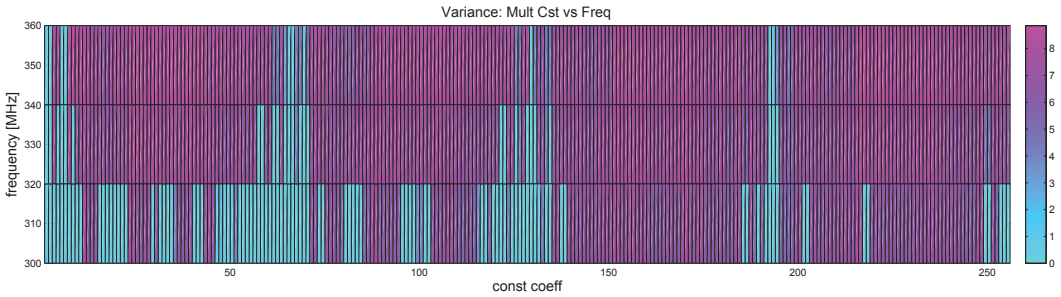

Figure 3.12.: Variance of constant coefficients for a LUT-based generic multiplier under different clock frequencies.

3.3.3. LUT-Based Generic Multiplier

The characterisation of LUT-based generic multipliers was performed under controlled operating conditions (i.e. temperature and voltage). Figure 3.12 shows the results for the characterisation of an 8-bit unsigned multiplier at different clock frequencies, in one location on the FPGA, for all possible constant coefficients, with 29.4k uniform pseudo-random samples. The number of samples is approximately half of all possible combinations for the input data. The data is represented in $\log_{10}$ scale of the error variance.

When compared to the CCM, the generic multiplier presents lower maximum clock frequency to operate without errors, a consequence of an increased number of paths. Moreover, small increases in the clock frequency makes more coefficients to present error variance, whereas the CCMs allow a greater increase in the clock frequency (apx. 60 MHz *vs* 100 MHz).

Figure 3.13 shows details for the error at the output of the multiplier, for the first 100 samples, when placed in two different locations on the device, identified with *loc 1* and *loc 2*. The target clock frequency is set at 320 MHz and one of the operands of the generic multipliers is fixed to a value of 222. The absolute error histograms, on the right, for the two locations are depicted in the same figure for the complete test. It can be concluded that the same multiplier placed in two different locations

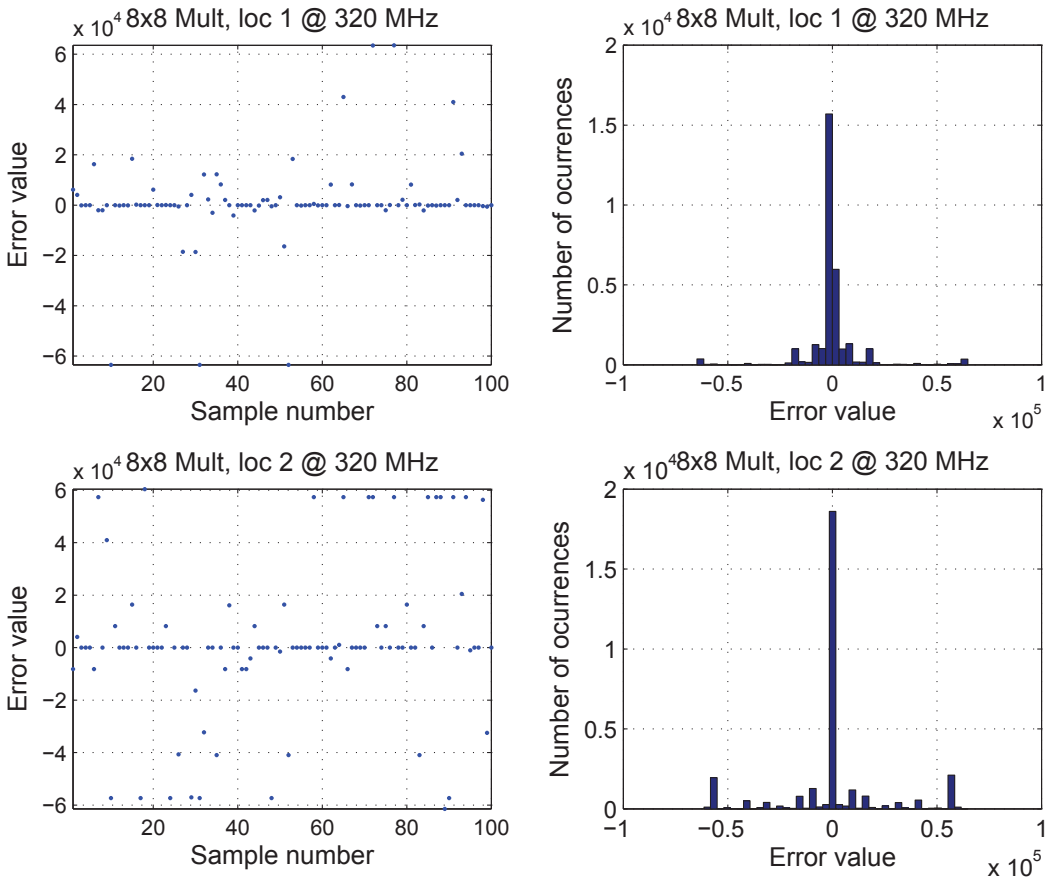

Figure 3.13.: The first 100 error values from a 8-bit LUT-based unsigned multiplier (left) and the distribution of all errors (right), for constant multiplicand 222 in 2 locations of a Cyclone III FPGA at 320 MHz.

produces different error patterns. It is observed that the multiplier on the bottom row had more errors in the MSBits (around $\pm 0.6 \times 10^5$) than the one on the top row. This can be justified by variations in placement & routing and process variation since the operating conditions are the same, and constant, for both multipliers during the test.

Scenario	Voltage	Temperature
Low-power	1000 mV	35 °C
High-performance	1400 mV	5 °C
Voltage sweep	1000-1400 mV	20 °C
Temperature sweep	1200 mV	5-50 °C
Process variation	1200 mV	20 °C

Table 3.2.: Operating conditions for characterisation of the DSP-based multipliers.

3.3.4. DSP-Based Multiplier

The embedded, or DSP-based, multipliers were tested on different sets of operating conditions, or scenarios, as summarised in table 3.2. These scenarios try to resemble some of the most common implementation objectives in DSP circuit design, e.g. low-power and high-performance.

Figure 3.14 presents the results for the error variance of low-power and high-performance scenarios, respectively. From the figure it is observable that the maximum clock frequency varies between 330 and 680 MHz thus exposing a gap of 350 MHz in their throughput.

3.3.5. Voltage and Temperature Variation

Voltage and temperature are the external sources of variation that can change more abruptly and may be difficult to control, e.g. battery powered and outdoor applications. Therefore, it's of great interest to evaluate the performance of arithmetic units under such variations. DSP blocks were used to minimise variations from placement & routing.

Figures 3.15 and 3.16 show the evolution of timing errors with voltage and temperature increase, respectively. It is distinguishable that voltage impacts the most in terms of maximum throughput of the DSP-based multipliers, approximately 80 MHz per 100 mV, whereas temperature variation, between 5 and 50°C, impacted the maximum throughput by no more than 10 MHz.

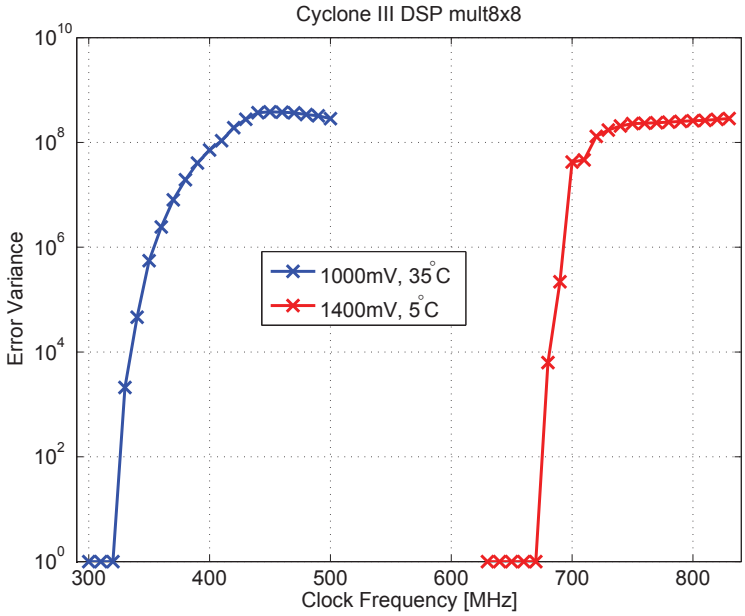

Figure 3.14.: Error variance results for DSP-based multipliers targeting low-power (1000 mV, 35 °C) and high-performance designs (1400 mV, 5 °C).

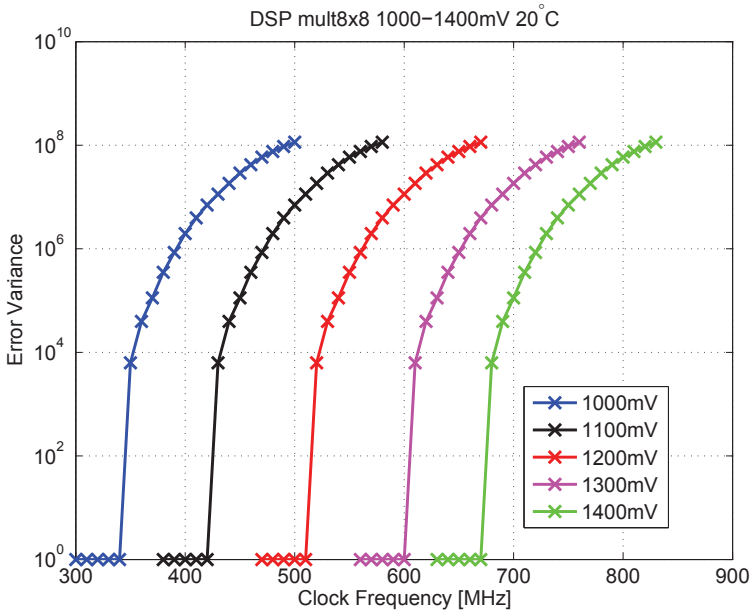

Figure 3.15.: Results for DSP-based multipliers with voltage variation.

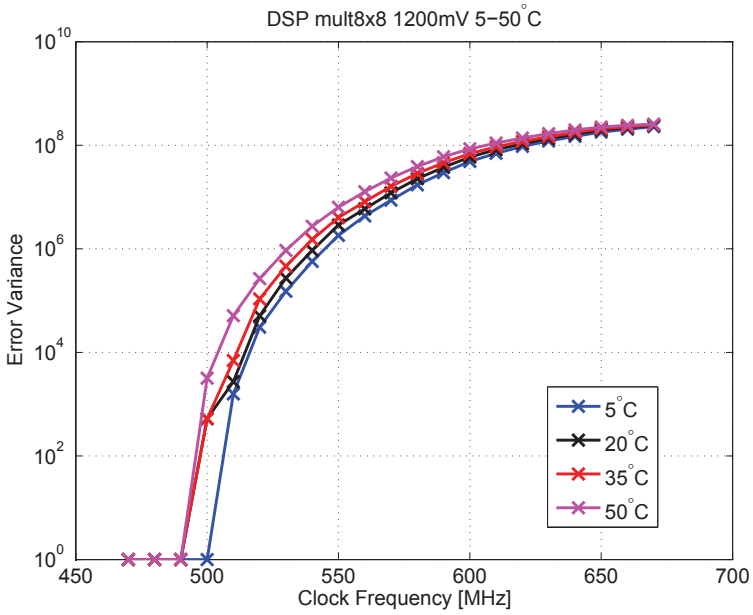

Figure 3.16.: Results for DSP-based multipliers with temperature variation.

3.3.6. Intra-die Process Variation

It's an established fact that devices are uneven, therefore this evaluation tries to model, in terms of mean error and variance, their impact. Unlike the LUT-based generic multipliers, the DSP-based multipliers have a fixed structure, embedded on the device, along with the other reconfigurable blocks, therefore it's the most suitable element to use in the assessment of the influence of process variation in the results.

The investigation on the impact of process variation is shown in figure 3.17. For three locations tested, it is observed up to 10 MHz penalty in maximum performance to operate without timing errors. In this test all multipliers were tested simultaneously using the same data. Table 3.3 indicates the coordinates of the multipliers on the device.

	X	Y
Loc 1	18	10
Loc 2	18	17
Loc 3	18	24

Table 3.3.: Locations of the DSP-based multipliers for the characterisation tests of a Cyclone III FPGA.

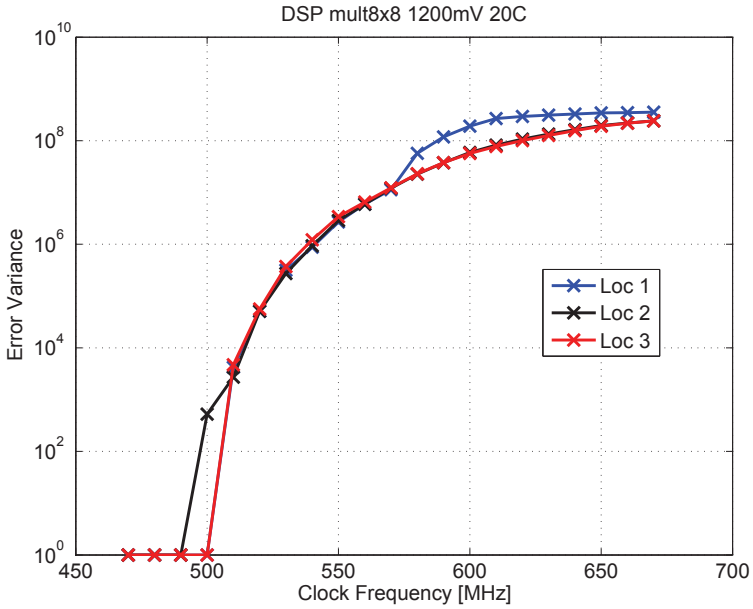

Figure 3.17.: Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #1.

3.3.7. Inter-die Process Variation

To assess the impact of process variation across different devices from the same family, the same test was repeated with the same synthesised circuit on other devices, from the same family, subjected to the same operating conditions.

An expected trend, in the absolute error statistics, is to have constant coefficients with values corresponding to power of 2 performing with less errors than the other constant coefficients. It is also expected to identify differences when the test is performed on different boards, due to process variations.

Figures 3.18-3.21 show the results for the same test repeated for different DE0 boards. Even though all graphs follow the same trend, it is observable that DE0 board #2 has different maximum clock frequencies to have the multipliers operating without errors. Figure 3.22 has the error distributions for the 3 locations, where the multipliers are implemented on the DE0 board #3, for different clock frequencies. It is obvious that all 3 multipliers exhibit different error patterns for the same test performed simultaneously using the same data on the same device. Hence, it is possible to conclude that this is related to the variations in the fabrication process.

Figures 3.23 and 3.24 show the error variance, and mean error, for different clock frequencies on a Cyclone III FPGA. Compared to the LUT-based generic multiplier, the DSP-based multiplier exhibits a smaller frequency band in the region between no coefficients with error and all coefficients with error. This is justified by the fact that delay due to routing increases the delay between paths with similar delay, whereas in the DSP-based all paths have a very small delay, thus making them with very similar delay. On the other hand, variation between DSP-based multipliers is smaller than between LUT-based multipliers. Consequently, small increases in clock frequency makes more paths to have their timing constraints violated. Compared to the results from another board, in figures 3.25 and 3.26, it is verified that some of the different coefficients have different errors at different frequencies and locations

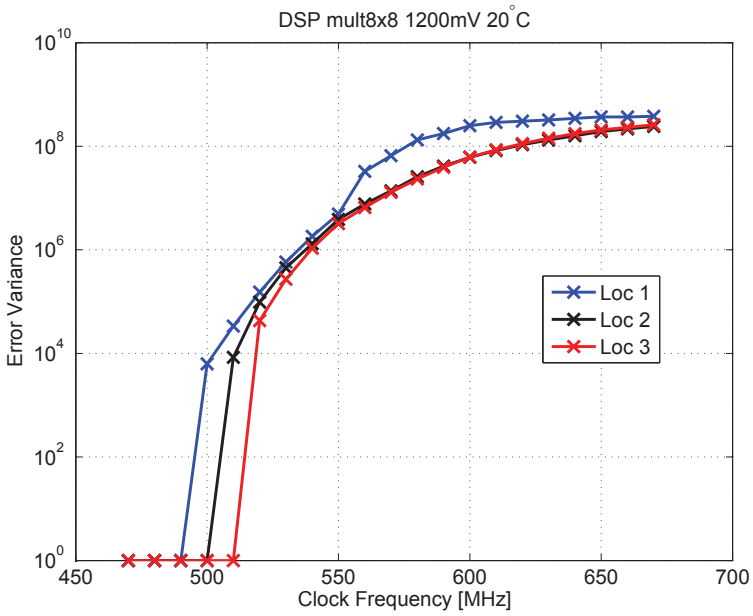

Figure 3.18.: Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #2.

on the device, even though it uses the same test circuit.

3.3.8. Process Size Variation

Regarding the results obtained from the characterisation of the devices, there's a question on whether such error modelling scales with the fabrication process. This section presents result for the same characterisation tests for two new device families, Cyclone IV and Cyclone V from Altera.

Figure 3.27 shows the characterisation test for two sets of locations, targeting a Cyclone IV device on a DE0 Nano board. It should be noted that location 2 remains constant for both tests, whereas the other locations change. Table 3.4 holds all the coordinates for both tests. From these plots it is possible to draw conclusions on the effects of the process technology (60 nm *vs* 65 nm) for the same architecture, as reported by the device manufacturer, and on the relation between the Hamming

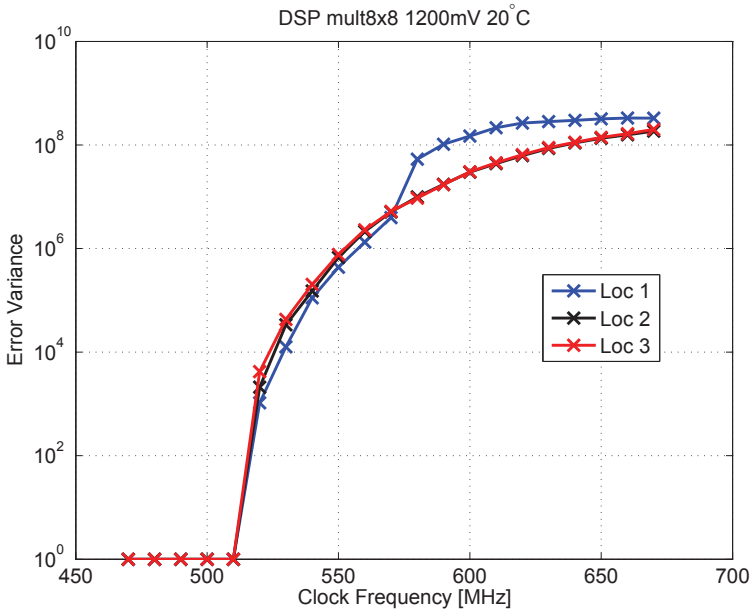

Figure 3.19.: Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #3.

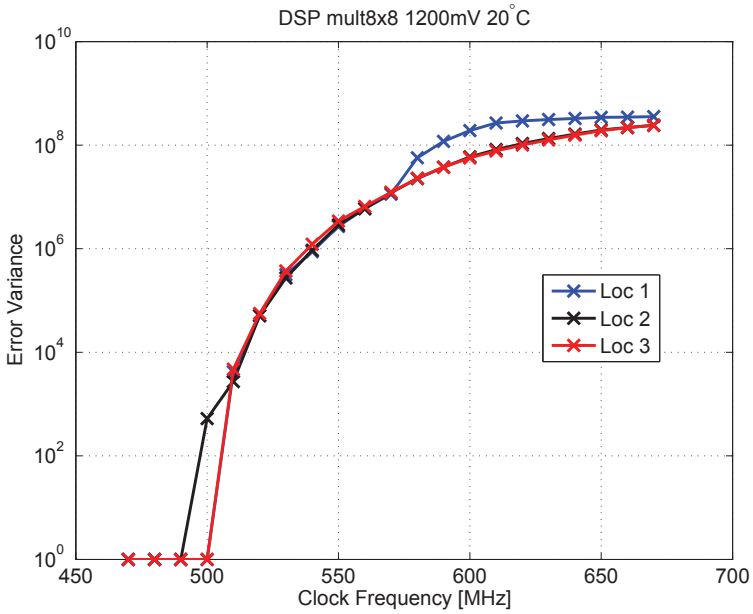

Figure 3.20.: Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #4.

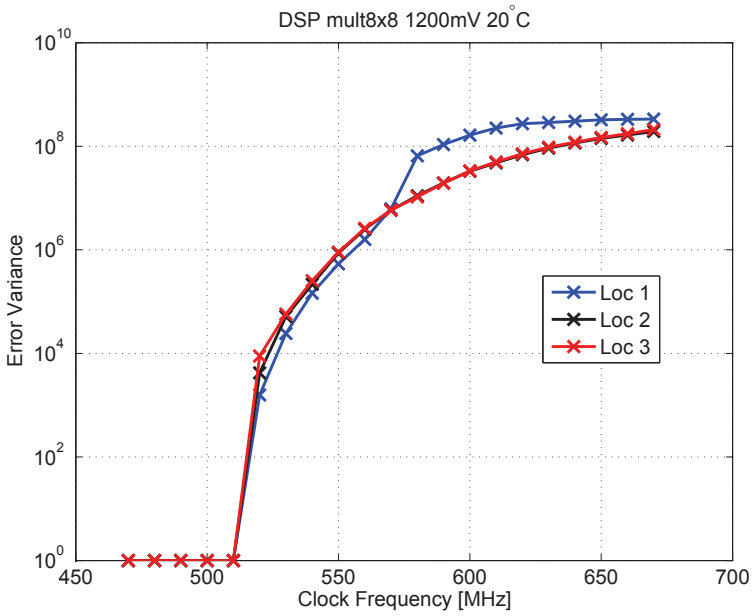

Figure 3.21.: Error variance from the characterisation of DSP-based multipliers on 3 different locations in DE0 board #5.

distance and the results when operating under variation. The maximum clock frequency, in the newer process, is increased by approximately 70 MHz, corresponding to a reduction in process size by 5 nm. In figure 3.28 there are the histograms for the errors on both tests on the same location. It is possible to conclude that there was no observed effect in the errors in one location of the device due to having other adjacent blocks operating.

Figures 3.29 and 3.30 show the error variance for different clock frequencies, in

(X,Y)	Test 1	Test 2
Loc 1	(42,16)	
Loc 2	(42,17)	(42,17)
Loc 3	(42,18)	
Loc 4		(42,14)
Loc 5		(42,20)

Table 3.4.: Locations of the DSP-based multipliers for the characterisation tests of a Cyclone IV FPGA.

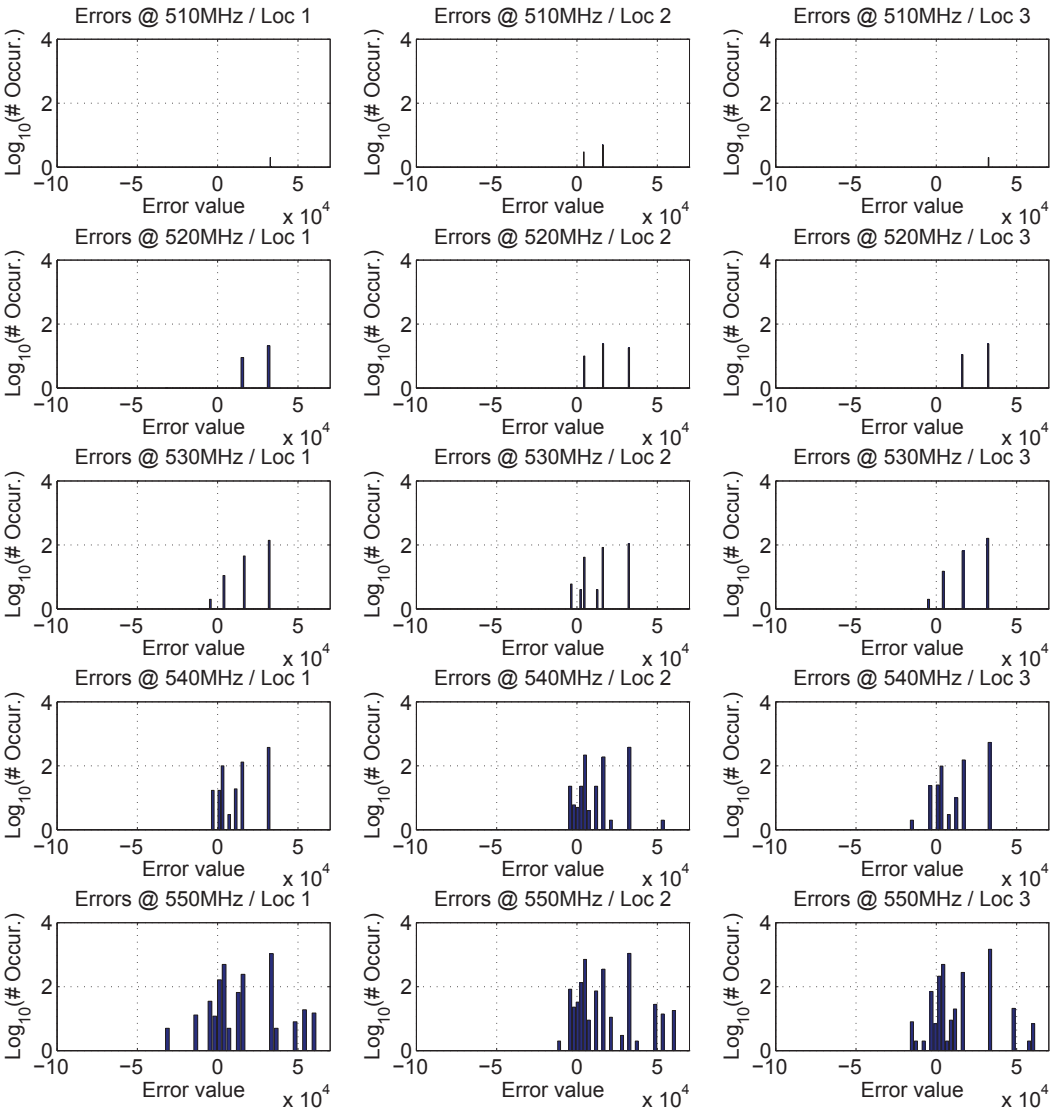

Figure 3.22.: Distribution of errors for 3 DSP-based multipliers on 3 different locations in DE0 board #4.

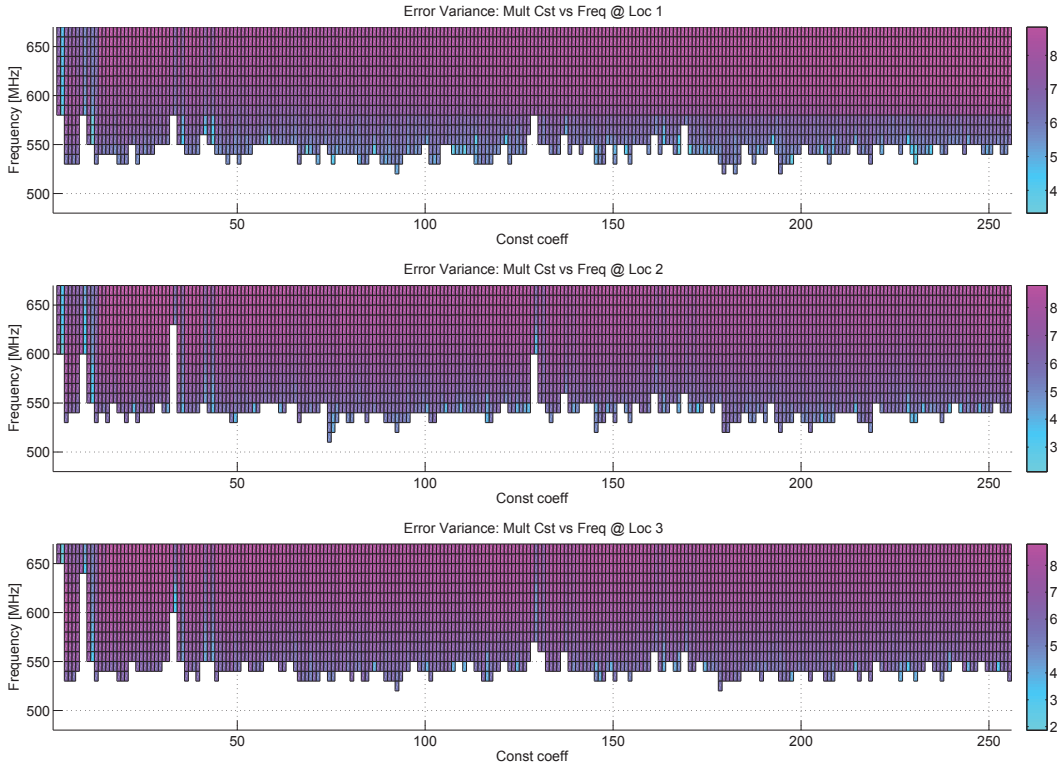

Figure 3.23.: Error variance for DSP-based multipliers on 3 different locations in DE0 board #4.

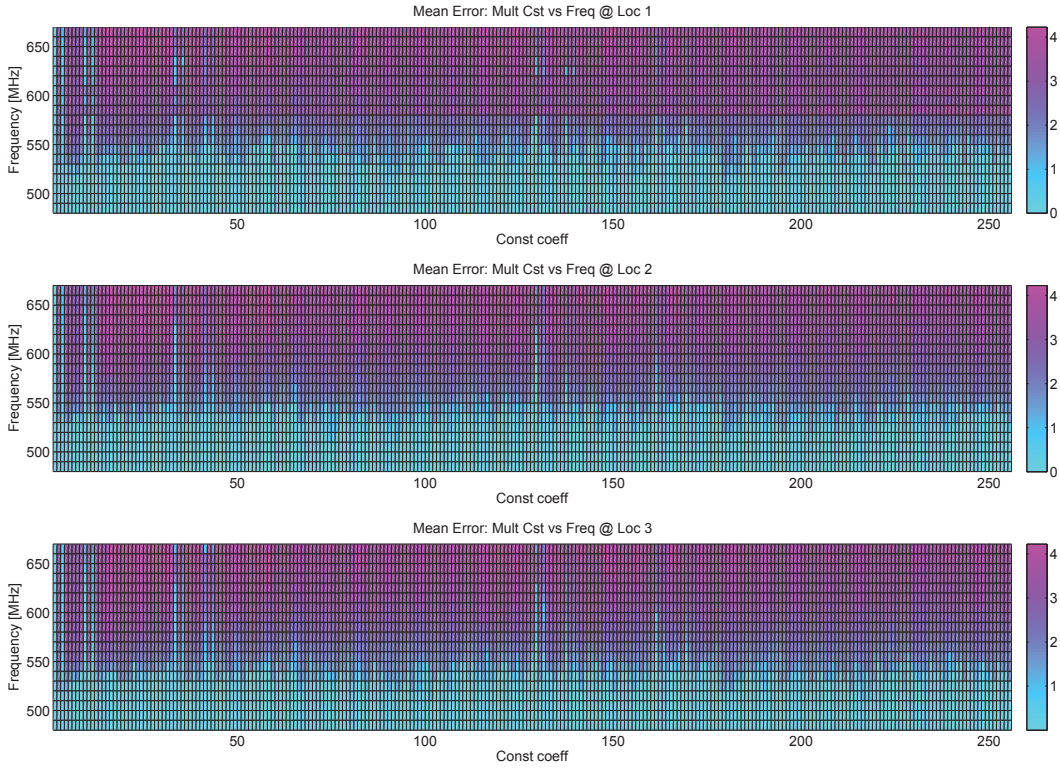

Figure 3.24.: Mean error for DSP-based multipliers on 3 different locations in DE0 board #4.

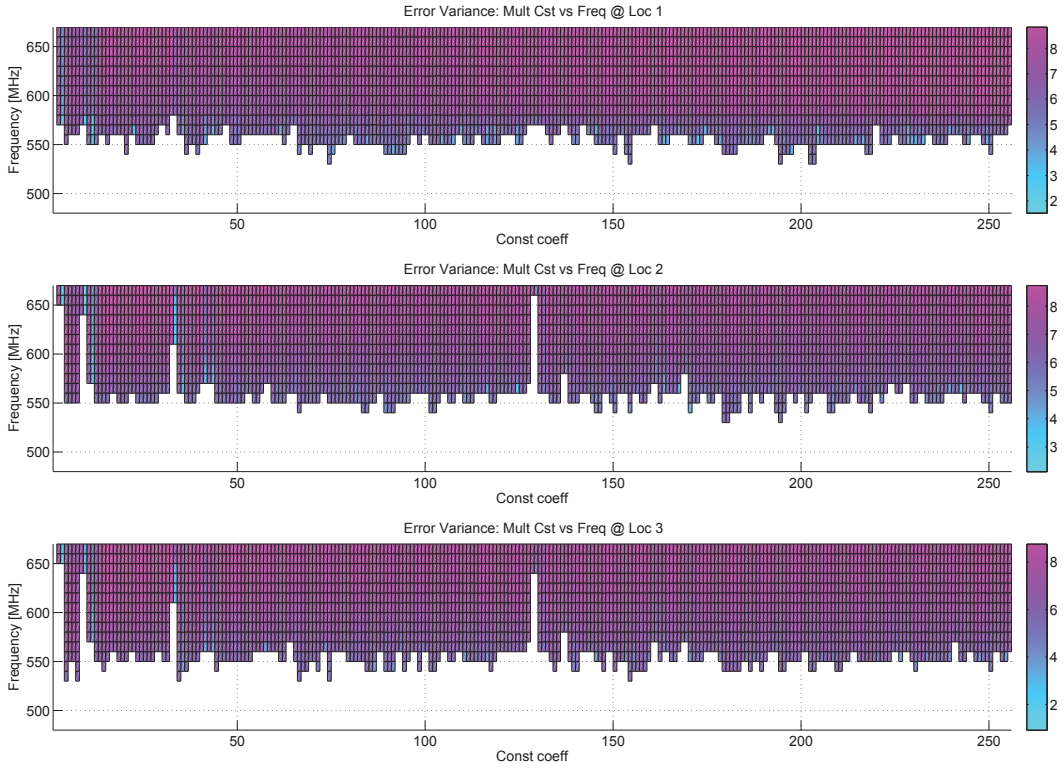

Figure 3.25.: Error variance for DSP-based multipliers on 3 different locations in board #5.

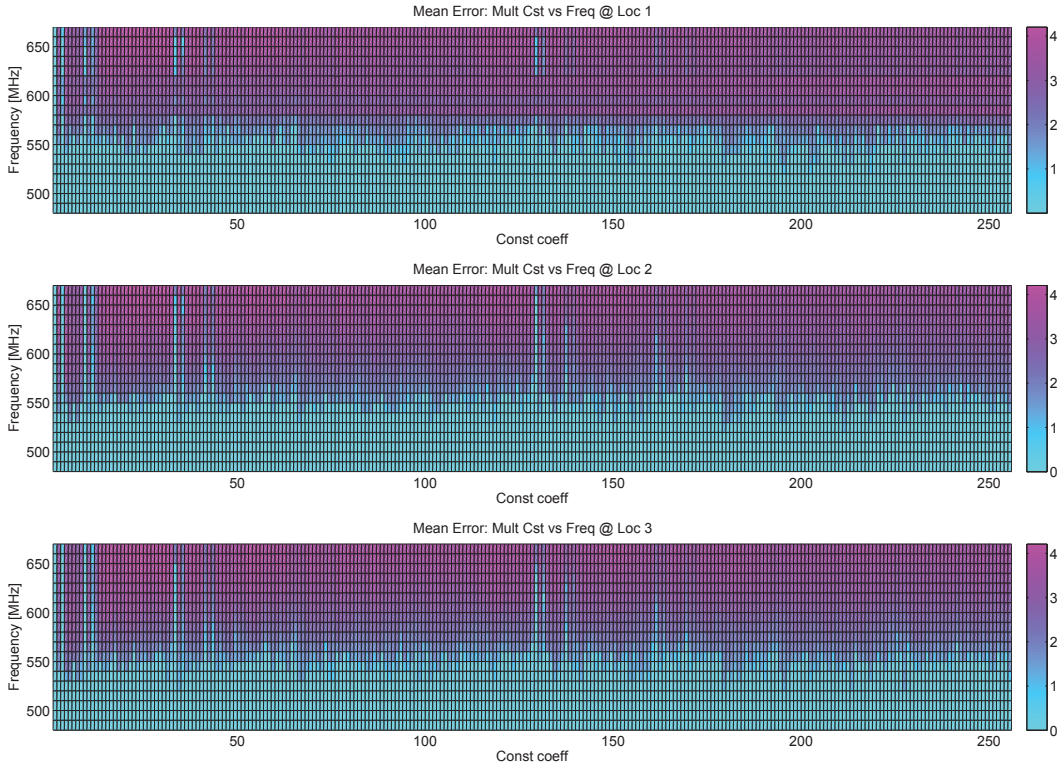

Figure 3.26.: Mean error for DSP-based multipliers on 3 different locations in DE0 board #5.

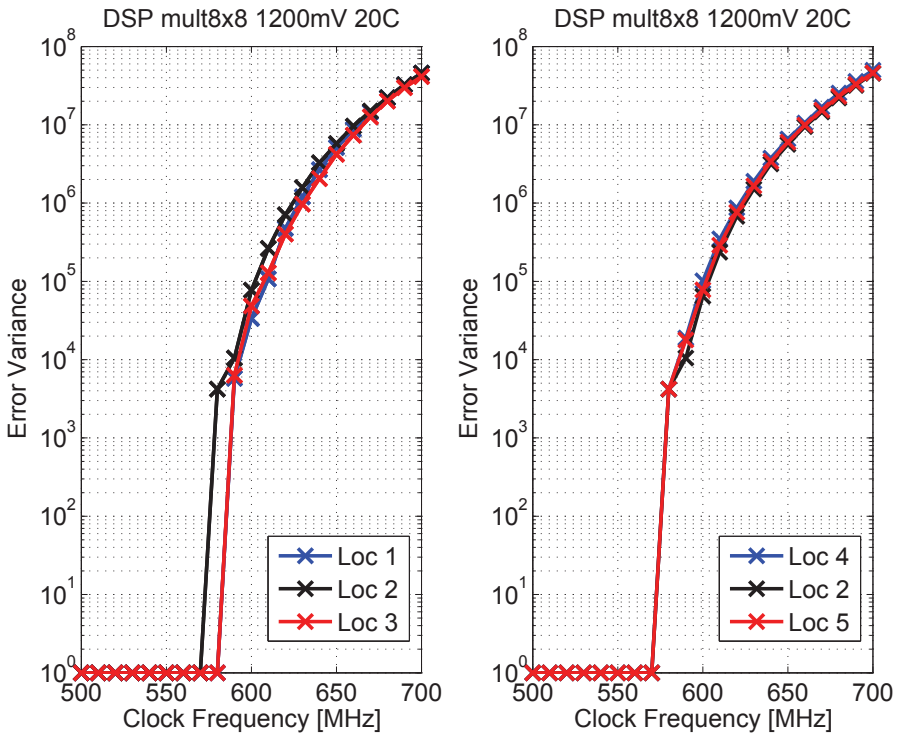

Figure 3.27.: Error variance for DSP-based multipliers on 3 different locations on a Cyclone IV FPGA, DE0 Nano board #2, tests 1 and 2.

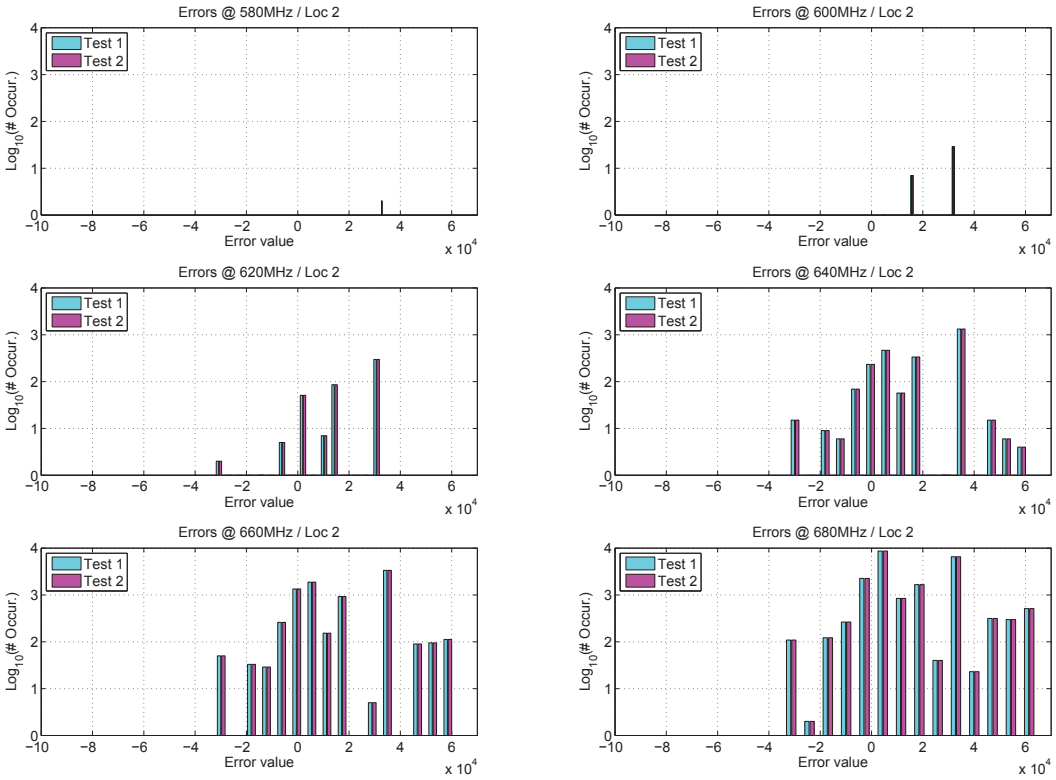

Figure 3.28.: Results for DSP-based Multipliers on a Cyclone IV using different designs.

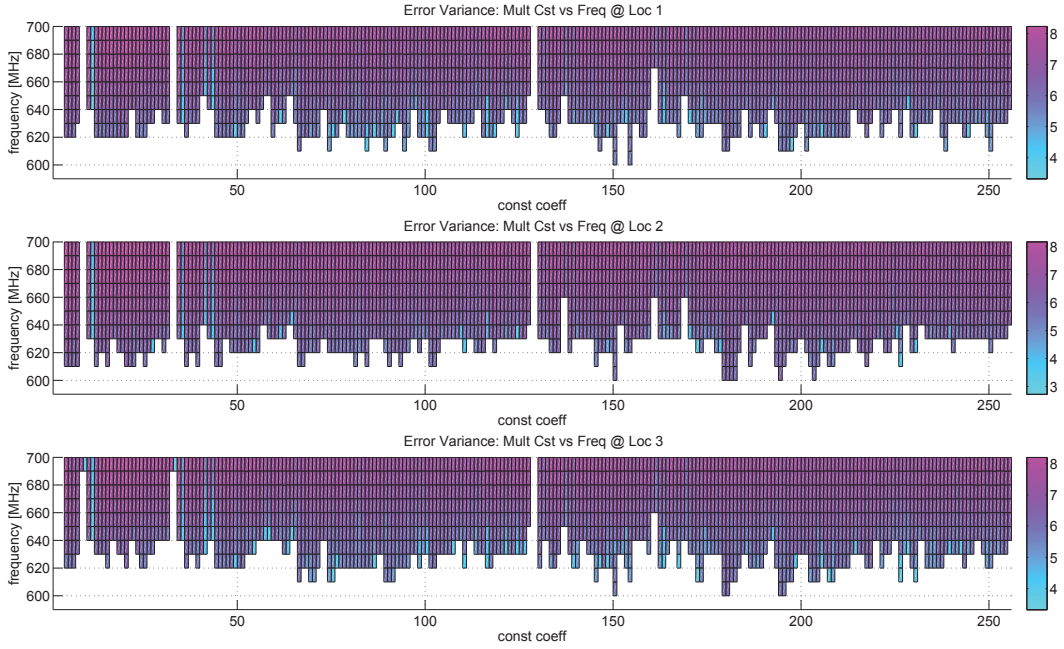

Figure 3.29.: Error variance of 3 DSP-based multipliers on a Cyclone IV FPGA (DE0 Nano board #1).

two different tests on the same board, using different locations for multipliers 1 and 3, while location 2 remains constant. While the error variance patterns change for locations 1 and 3, it remains constant for the 2nd, as illustrated previously with the error histograms. Figure 3.31 holds the same error variance but for a different board, and 3.32 the observed mean error. From all characterisations for the different multipliers, on different families of FPGAs, it is evident that the constant coefficients have different error variances. Thus, choosing their values carefully can create designs with small deviations from the standard implementation of an algorithm, but with great benefits in terms of error minimisation when compared to over-clocking typical design methodologies.

Recently a smaller fabrication process was introduced and a new family of FPGAs was introduced, Cyclone V [3]. This FPGA uses a 29 nm process size. This FPGA has some architectural differences from the previous Cyclone families, namely the

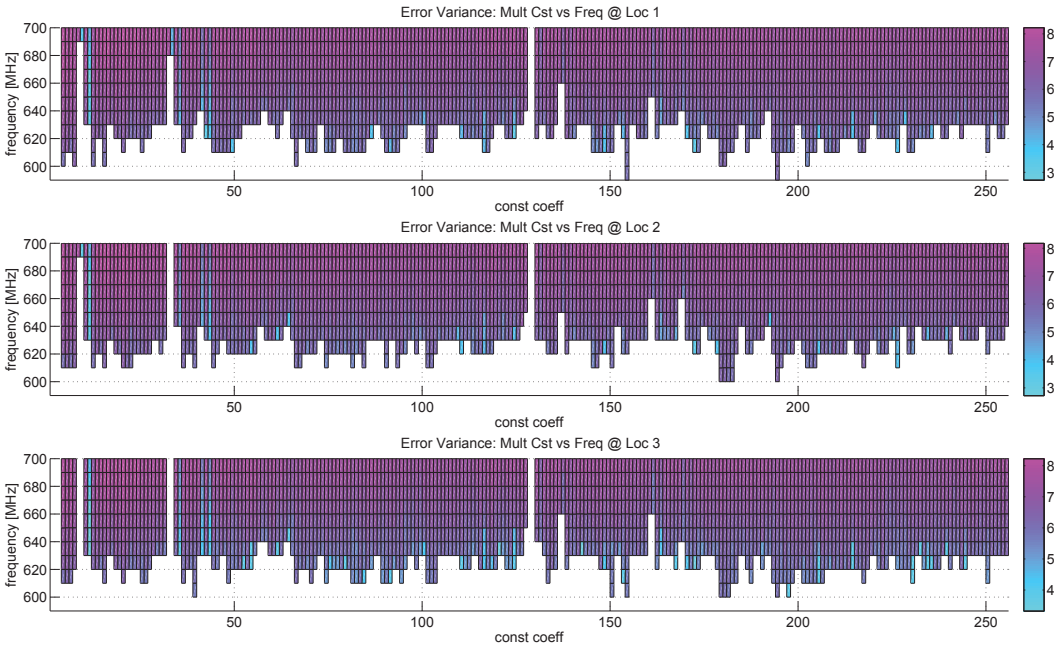

Figure 3.30.: Mean error of 3 DSP-based multipliers on a Cyclone IV FPGA (DE0 Nano board #1).

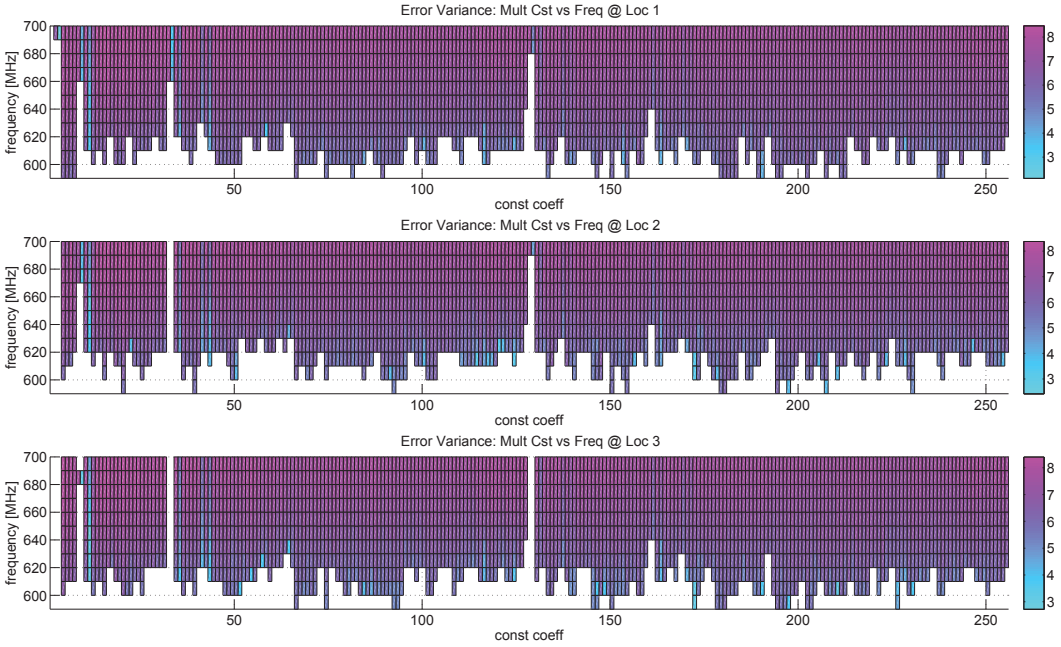

Figure 3.31.: Error variance of 3 over-clocked DSP-based multipliers on a Cyclone IV FPGA (DE0 Nano board #2).

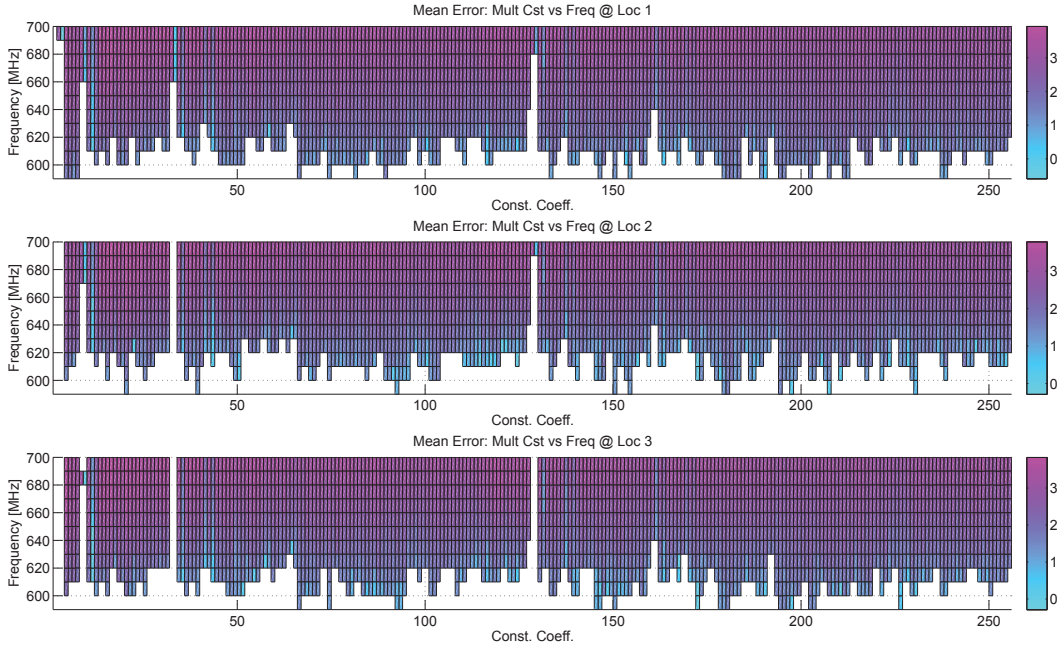

Figure 3.32.: Mean error of 3 over-clocked DSP-based multipliers on a Cyclone IV FPGA (DE0 Nano board #2).

use of Adaptive Logic Modules (ALMs) instead of LEs, and complex DSP blocks instead of embedded multipliers.

Figure 3.33 shows the results, in terms of error variance, for the same characterisation test, on a DSP Block of a Cyclone V FPGA fitted on a BeMicro CV board from Arrow, along with results for Cyclone III and IV for comparison. In this test, Cyclone V voltage was set to 1100 mV and temperature was kept at 23°C. Temperature was monitored using a DVM345DI multimeter from Velleman. In comparison with previous results, the evolution of error variance with the clock frequency increase resembles the results for high-performance from Figure 3.14, with a fast increase in error. As a result, there's a gap of 30 MHz from no errors to maximum errors in the DSP block.

In this case, the maximum clock frequency for error-free operation of the Cyclone V FPGA is 550 MHz, which is 20 MHz below the maximum found for the Cyclone IV

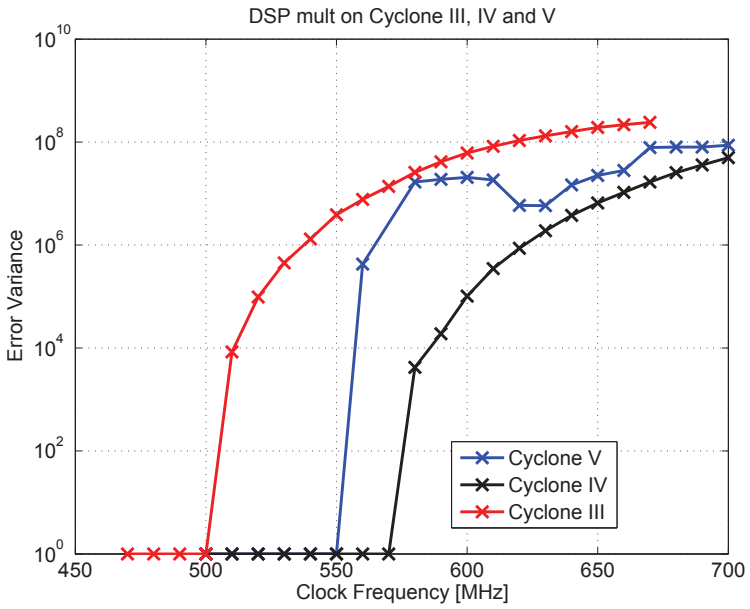

Figure 3.33.: Error variance of over-clocked DSP-based multipliers on Cyclone III, IV and V FPGAs.

FPGA. Even though the expectation is to obtain increased performance in smaller devices, changes in architecture of the DSP blocks, supply voltage and speed grade can contribute to a loss in performance.

3.4. Impact of Variation in Linear Projection Designs

Succeeding the characterisation test for individual arithmetic units, a DSP application was tested to demonstrate the impact of variation of the operating conditions in the final results. The application chosen is the linear projection. This algorithm has been presented in the background section of this thesis, and it is well known that it can tolerate some errors in its calculations.

The present test considers a face-recognition application performing projection of images with 2k dimensions (50x40 pixels) to a smaller space with 40 dimensions, on a DE0 board from Terasic [89], fitted with a Cyclone III 3C16 device from

Altera [1]. The test was carried out using generic multipliers implemented with LUTs and DSP blocks. During the test the FPGA was supplied with 1200 mV and kept at 20 °Celsius, through the usage of an external control system which is described in the appendix of this thesis. Given the number of dimensions involved in this example, the implementation follows the folded architecture implementation of the dot-product operator. Figures 3.34 and 3.35 show the back-projections, in the original space, for the projection of 5 images on an FPGA using LUT-based and DSP-based multipliers, respectively, at different clock frequencies. In this particular example the quality of the results is presented as back-projection of the projected data, in the circuit, into the original space and measured in dB, which is obtained from the PSNR introduced in equation 2.18. The first row shows the expected results without any errors. In the rows below, it is observable that with the increase in the over-clocking frequencies the faces produced by the circuit are similar to the original ones, even though some distortions are perceptible. It is observable that the DSP-based multiplier have its results degrading more gracefully than the LUT-based ones. For the same level of noise introduced, the DSPs-based multipliers have a gap in performance of 60 MHz whereas the LUT-based multipliers have less than 40 MHz. This is may be due to the fact that, up to a certain clock frequency, not all paths have their timing constraints being infringed. When most of the paths are experiencing timing errors, from extreme over-clocking, the magnitude of the errors renders the system unusable as the faces are no longer recognisable.

3.5. Run-Time Investigation

The main contribution to the run-time of the characterisation framework is the data transfers via JTAG. The execution times, in seconds, have been observed and been approximated to the following equation as a way to estimate the framework's

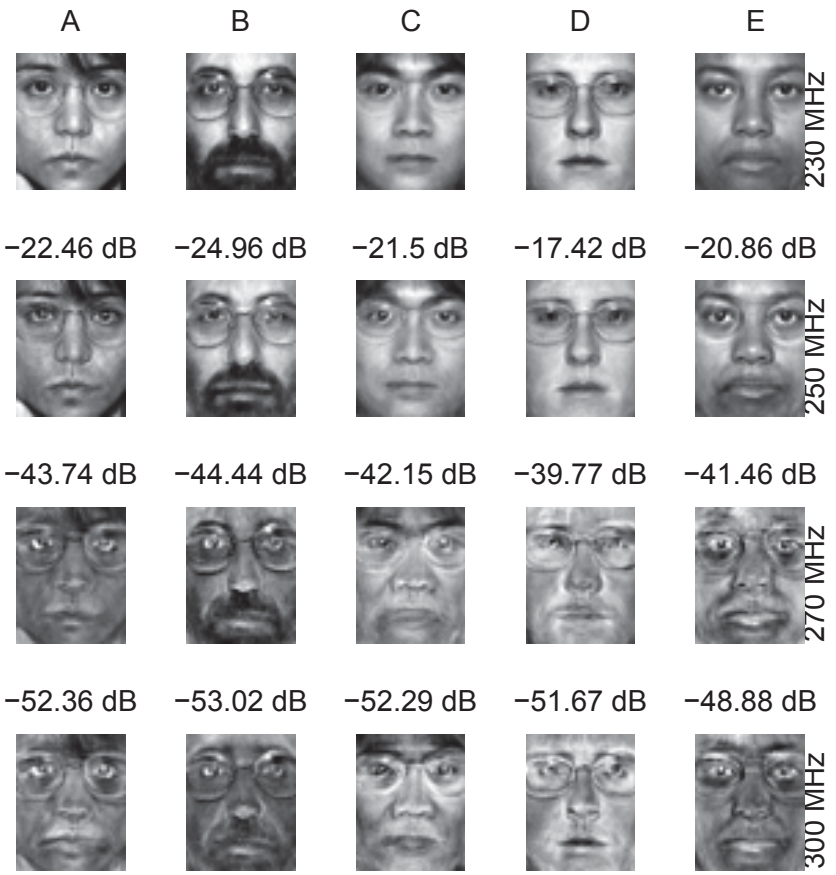

Figure 3.34.: Faces obtained from the reconstruction of the linear projection implemented with LUT-based multipliers operating at 230, 250, 270 and 300 MHz.

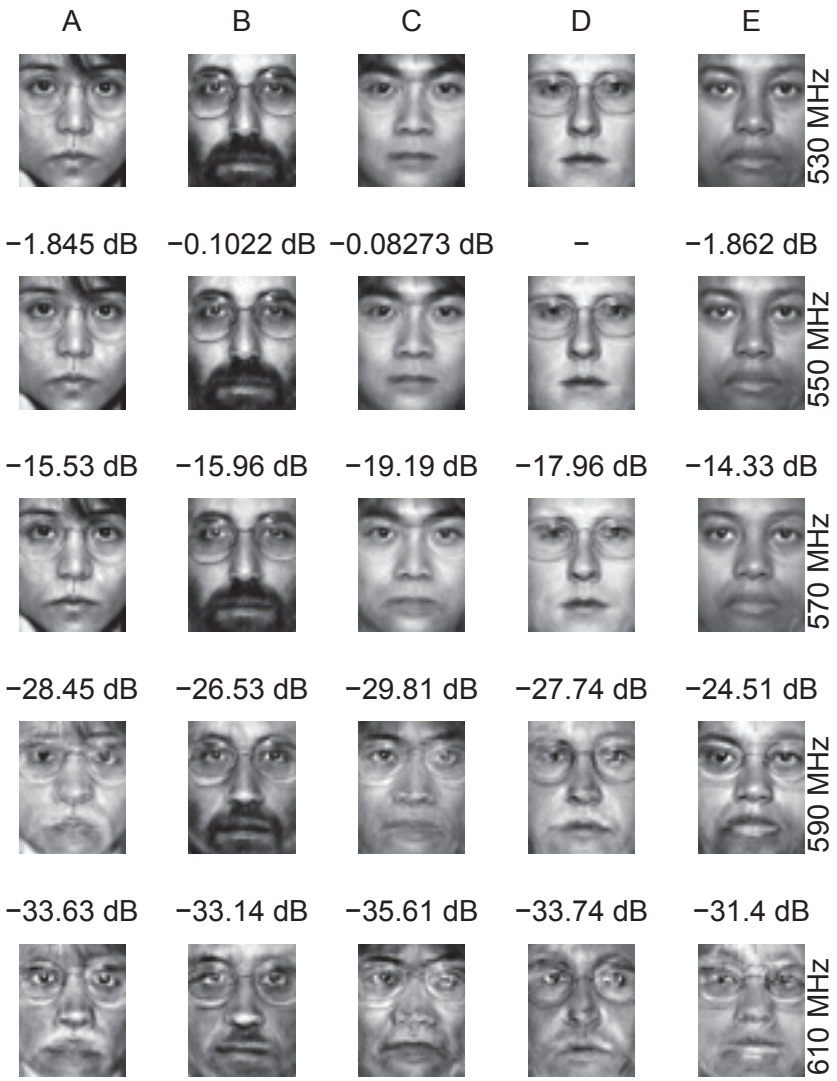

Figure 3.35.: Faces obtained from the reconstruction of the linear projection implemented with DSP-based multipliers operating at 510, 530, 550, 570 and 590 MHz.

run-time for tests on three arithmetic units simultaneously:

$$T = 1.7143n \tag{3.1}$$

The run-time (T) is measured in seconds, and depends on the number of 2k samples vectors tested (n) on the arithmetic unit. 2k samples is the capacity of the memories in the design.

3.6. Summary

In this chapter a framework has been proposed to characterise the performance of arithmetic units. In this case LUT and DSP-based generic multipliers were tested under various settings (i.e. clock frequency, location on the device, supply voltages and device temperatures). The characterisation framework was used to do the characterisation of common arithmetic units in DSP designs. Results show how the arithmetic units performed when subjected to variation of their operating conditions (voltage, temperature, location, placement & routing). Since arithmetic units often are the performance limiting elements in a DSP design, characterising them under certain operating conditions allows to maximise the operating clock frequency without incurring in timing errors. Moreover, since they don't exhibit all the same results, this can be used to further increase the performance by using the ones that exhibit less errors.

In addition, this framework also produces statistics on the results for the characterisation of the units under test, using different operating conditions. Table 3.5 summarises the impact of the different sources of variation in the most common arithmetic operators in a DSP design. Besides characterising the arithmetic units, the framework was adapted to assess the impact of variation in the results of a linear projection. This framework can be extended to other arithmetic units and devices,

Variation Source	Impact in performance
Process	Small
Voltage	Large
Temperature	Small
Placement & Routing	Medium

Table 3.5.: Impact of the different sources of variation in the performance of arithmetic units.

being the main limiting factor the placement & routing of the auxiliary blocks for the test circuit, as it needs to have its critical paths as short as possible in order to operate at higher clock frequencies than the unit under test.

The obtained results, for different operating conditions, demonstrate that as the clock frequency increases, more erroneous data appear at the output of the multipliers, demonstrating that the presence of errors is cumulative as the clock frequency increases, which is as expected. One interesting finding was the fact that the scaling down in the fabrication process has led to designs that degrade faster with the increase in clock frequency.

Notwithstanding, it was observed that placement of the arithmetic units under characterisation in various locations of the device, produced different error patterns in their outputs. However, this effect is not only attributed to the performance difference derived from the fabrication process of the FPGA device, but also from the variations in routing created by the synthesis tool.

4

Redundancy in Arithmetic Units

4.1. Introduction

Redundancy has been proposed as a method to incorporate resilience into arithmetic units, or sub-systems, on a datapath sensitive to errors. The objective is to provide the arithmetic units with resilience from variation errors, due to infringement of timing constraints on their critical-paths, when the design is under variation of its operating conditions.

RPR is based on the result, of a truncated version of the unit considered, computed in parallel with the original one. When the absolute difference, between the outputs of the original and the reduced units, is above a threshold specified by the user, then

the output of the reduced unit is placed at the output of the RPR unit instead of the value from the original unit. Since there's always a result present at the output of the RPR unit, it is advantageous to use it in architectures that don't tolerate stalls in their datapath. The background section of this thesis provides an explanation on how RPR works and offers an overview on the most significant contributions.

Even though this methodology allows to keep the throughput constant, as it has been proposed, it requires the insertion of, at least, an extra clock cycle in the datapath, a consequence of the register placed between the output of the original and approximation units and the correction circuit. This can be problematic, or even prohibitive, in circumstances where the algorithm doesn't allow further pipelining due to the cost in extra resources, or penalty in the quality of the results [90].

Despite of the attention it has received, there haven't been proposals to adapt the RPR methodology to arithmetic units which produce results without requiring extra clock cycles. The research presented in this chapter aims to close this gap by proposing a novel framework that allows to add resilience to arithmetic units by limiting the magnitude of the errors at their output.

An investigation of existing RPR schemes [11, 70, 67] shows that the majority adopts the architecture depicted in figure 4.1. This architecture could be modified to produce the result within a single clock cycle, if the registers at the output of the original and reduced units were to be removed. Although this modification would allow to use RPR within one clock cycle, the delay of the critical-paths in the new RPR unit would be longer than the delay of the critical-paths in the original arithmetic unit. In other words, the delay of the critical-paths in the new RPR unit would be the delay of the original arithmetic unit plus the delays of the subtracter, comparator and multiplexer, which come after the output of the original arithmetic unit considered.

Fundamentally, the delay of the redundant circuitry limits the maximum clock

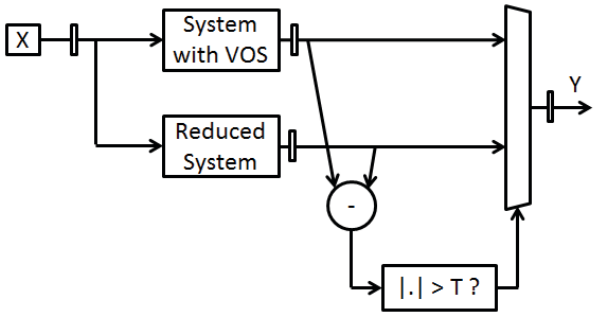

Figure 4.1.: Typical RPR architecture applied to a system under VoS.

frequency to produce correct results. In such scenario, even in case the circuit is clocked at clock frequencies below the maximum clock frequency of the original arithmetic unit, the value at the output of the RPR block could be different from expected as a consequence of violation of the new critical-paths. Moreover, the maximum clock frequency in the RPR unit is set by the delay of the most critical-paths between the input of the reduced unit and the output of the multiplexer.

The proposed framework intends to lift up this limitation by addressing the role of the components contributing to the delay of the new critical-paths, hence reducing them to a minimum. In order to achieve this, a new architecture was imagined and the design choices automated through the framework. The remaining of this chapter is devoted to explain the proposed RPR framework, how is the new architecture applied to different arithmetic operators and presenting the comparison in performance evaluation when compared to the typical RPR adapted to single clock cycle operation.

4.2. Reduced-Precision Redundancy Framework

The proposed RPR framework operates similarly to existing RPR schemes, as it creates a circuit that encapsulates the original unit, without modifying it, and replaces it in the original circuit. But it distances itself from previous works as it enjoys fea-

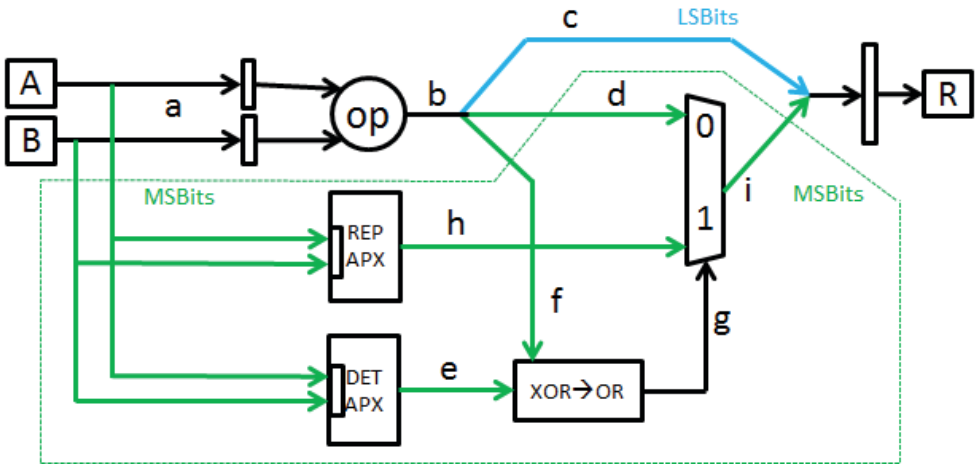

Figure 4.2.: New RPR architecture applied to a generic combinatorial operator.

tures that are specific for throughput and latency critical applications, and relies on a new RPR architecture for zero latency penalty. Also, it benefits from specialised blocks embedded in modern FPGAs as well as their reconfigurable capabilities.

In terms of operation, the user specifies the design budget, in terms of FPGA resources, and the RPR framework provides the solutions for possible implementations of RPR units.

4.2.1. Architecture

In contemplation of the aforementioned constraints a new architecture has been devised. The new architecture borrows the idea of replacing the MSBits at the output of the arithmetic unit with an approximation but it proposes new methods to detect variation errors and to produce approximations. Usually, RPR schemes target the MSBits of arithmetic operators, i.e. adders and multipliers in binary or two's complement representations, as they usually hold the paths with the longest delay [70].

Figure 4.2 shows the proposed architecture applied to a generic combinatorial unit

op (original). *A*, *B* and *R* are the inputs and output of the RPR unit, respectively. Identifier *a* refers to the input arguments and *b* to the result of the original operator. The remaining identifiers refer to the paths added by the RPR. Besides the original unit (*op*), the architecture includes a block to provide approximations used in error detection (*DET APX*) and another for correction (*REP APX*). The inputs of the approximation blocks are the truncated inputs *A* and *B*. Both inputs for the approximation blocks and original operator derive from the output of the previous block in the system. It also includes a combinatorial block, responsible for the indication of a mismatch in the MSBits, named *XOR→OR*, and a multiplexer to select which value pass to the output.

The output of the detection approximation (*DET APX*) is identified with *e*. This signal holds the MSBits to be used in the comparison with the MSBits at output of the original unit (signal *f*). The output of the replacement approximation (*REP APX*) is identified with *h*. This signal holds the MSBits to be used in the replacement, or correction, of the MSBits at output of the multiplexer (signal *i*).

The MSBits were chosen to be compared, and replaced, because usually they are the signals with the critical path in the original unit. The LSBits of the original result (signal *c*) aren't affected, hence they are the same as at the output of the original unit. MSBits are compared via a bit-wise XOR followed by an OR of all resulting bits (signal *g*). Whenever the MSBits match, the output of this block is zero, and one if at least one bit differs. In the first case, at the output of the multiplexer (*i*) will be the MSBits from the original unit (signal *d*). Otherwise, the multiplexer's output will be passed to the output of the replacement detection table (signal *h*). Moreover, this new architecture is focused on minimising variation errors due to encroachment of the timing constraints on the most critical-paths of a datapath. Thus, it *shows* the approximation result at the output of the multiplexer *while* the MSBits at the output of the original unit *don't match* the MSBits from

the detection approximation.

One of the main novelties in this architecture is the use of look-up tables, to produce the approximations. They are identified as Read-Only Memory (ROM), and they are used to hold the results for the approximation functions, instead of the truncated implementation of the operator. Another novelty is a bit-wise comparison ($XOR \rightarrow OR$) instead of a subtraction followed by a comparator to detect the presence of variation errors. This leads to significant savings in delay between the input ports and the output ports of the unit. Forasmuch, as the gap in delay between the output of the approximation (signal e) and the original result (signal b) increases, it allows to push the clock frequency even further, as illustrated in figure 4.3. This figure shows the maximum clock frequencies for the standard unit and the proposed RPR unit and their operating regimes: error-free/expected result (green), error-prone/approximated result (orange) and incorrect result (red). It also shows the delays that contribute to the maximum clock frequencies of RPR units. Standard units can operate without errors at higher clock frequencies than the RPR unit but once they reach their limit, results at their output will be incorrect and unpredictable. On the other hand, the clock frequency of the RPR unit will be impaired due to the redundant circuitry. Hence, the top frequency the RPR unit can operate, producing approximate results, is given by the delay of the elements in the approximation's critical-path.

The new architecture uses two approximations in parallel. The approximation for detection ($DET\ APX$) of errors is treated separately from the approximation to replace ($REP\ APX$) results detected as a mismatch. This is derived from the fact that information is missing from an approximation computed from truncated input arguments. So, a bit-wise comparison between the expected result, from the original unit, and approximation unit will identify mismatches. This is not desirable as it will mark expected results a mismatch and have the MSBits replaced with the values

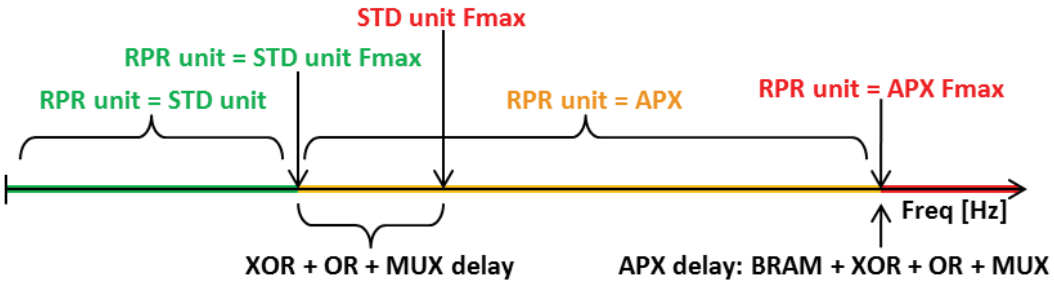

Figure 4.3.: Illustration of the maximum clock frequencies, and delays, for standard and RPR units and their operating regimes: error-free/expected result (green), error-prone/approximated result (orange) and incorrect result (red).

A	B	Result	MSB
1001	1111	10000111	1
1000	1100	01100000	0

Table 4.1.: Results of 2x2-bit multiplications using the original and truncated operands.

from the approximation. As illustrated in table 4.1 for a multiplication of 9 and 15. The MSBit from 2 MSBits of each operand differs from the expected MSBit.

On this account, in detection it is desirable to use as many bits as possible to obtain the least number of false positive mismatches. On the other hand, if the original unit is producing incorrect MSBits it is likely that some of the LSBits will also be incorrect. Hence, it is advantageous to replace more MSBits than the ones considered in the detection block. Nevertheless, if the MSBits are to be replaced by an approximation, then the impact of the lack of information will correspond to the approximation error at the output of the RPR unit, similarly to the existing RPR schemes.

Given the aforementioned, the proposed RPR framework not only allows to specify different word lengths for the inputs and outputs of the approximation ROMs, but it also allows to use different approximation functions for detection and replacement. The approximation functions in the ROMs depend on the arithmetic operator used,

Field	Description (word length of)
Ori iWL	input in the original unit
Ori oWL	output in the original unit
Det iWL	input in the detection ROM
Det oWL	output in the detection ROM
Rep iWL	input in the replacement ROM
Rep oWL	output in the replacement ROM

Table 4.2.: Denominations of the labels in the nomenclature adopted for the proposed RPR scheme.

and they are automatically created by the proposed RPR framework to minimise the impact of false positives and approximation errors.

On the ground of possible implementations using the new RPR architecture a nomenclature is proposed to help identifying the word lengths of key elements: original unit, detection ROM and replacement ROM. The syntax is as follows, with the description for each field in table 4.2:

Ori iWL : Ori oWL / Det iWL : Det oWL / Rep iWL : Rep oWL

Furthermore, to distinguish the different RPR schemes from each other, the following prefix is added to the above nomenclature, which is adopted throughout this work:

- LUT-SUB - existing RPR; approximation computed from a truncated arithmetic unit implemented using LUTs/LEs; an error is detected from the difference between the output of the original unit and the approximation;
- ROM-XOR - proposed RPR; detection approximation is retrieved from a ROM, and an error is detected from bitwise comparison with the MSBits from the original unit, and then replaced with an approximation, using the replacement, approximation from another ROM, in case of mismatch.

As an example of this notation, an RPR 8-bit multiplier, with a detection ROM with 5 input bits and 2 output bits, replacement ROM with 4 input bits and 3

output bits is represented as: ROM-XOR 8:16/5:2/4:3 multiplier. Additionally, it is possible to consider other combinations for the RPR architecture, if the design constraints impose them, i.e. ROM-SUB or LUT-XOR, but they aren't covered in this work as their critical-paths will exhibit more delay than the ROM-XOR, thus expecting less benefits from its adoption. Moreover, this architecture scales to other arithmetic operations, or a set of operations, to be applied to RPR while being computed within a clock cycle. For such cases the nomenclature can be extended by separating the word lengths, for the input and output ports, with comas, e.g. ROM-XOR 9,8:17/6,5:2/5,4:3 multiplier.

4.2.2. Approximation Functions

At the heart of the proposed RPR framework is the computation of the approximation functions that are stored in the ROMs. These approximation functions try to accurately represent the MSBits of the original unit to minimise the approximation errors, and are distinct from other approximations used to optimise synthesis of arithmetic functions such as Taylor and MacLaurin series. Previous contributions on RPR, had the approximate, or redundant, result computed from a truncated version of the arithmetic unit in parallel. Here, the new approximation functions are evaluated by the RPR framework offline. As a result, the proposed RPR framework is able to support different types of functions from the truncated operators:

- truncated approximation;
- linear approximation;
- other approximations derived from different objective functions, e.g. mode values of the MSBits from the expected results and values that minimise error variance.

The linear approximation of a generic binary arithmetic operator ($\star$), with truncated operands, can have one or more unknown coefficients (i.e. k_A, k_B, l, m_A and m_B with $k_A, k_B, l, m_A, m_B \in \mathbb{Z}$).

$$A = \sum_{i=N-iWL}^{N-1} a_i.2^i, \quad B = \sum_{i=N-iWL}^{N-1} b_i.2^i \tag{4.1}$$

$$X = (A \cdot k_A + m_A) \star (B \cdot k_B + m_B) + l \tag{4.2}$$

In this direction, the framework exhaustively searches the function's domain for the coefficients in the linear approximation function which results in the minimisation of the objective function, between the expected and the approximation MSBits. The truncated approximation is a particular case of the linear approximation with its coefficients being the neutral element of the operations involved. Algorithm 1 illustrates this while considering $k_A = k_B$ and $m_A = m_B$. The minimum and maximum values for k, l and m are arbitrarily chosen by the user. The complexity of this algorithm depends on the number of approximation coefficient values (C) and is defined as $O(C.N^2)$, where $C = (k_{max} - k_{min})(l_{max} - l_{min})(m_{max} - m_{min})$ and O the order of complexity.

The usage of a ROM to store the approximations concedes the opportunity to implement any function while the cost, in terms of hardware resources, is constant regardless of the function being implemented. An example of an approximation function evaluated in this scenario resembles the mode of the expected MSBits, hence minimising the number of errors between expected and approximation values. More details about the approximations considered for the different arithmetic operations are presented below.

Given that the approximations don't hold all the information required to detect all expected MSBits correctly, some false positive error detections will occur.

Algorithm 1: Algorithm to search the approximation coefficients for the $\star$ RPR operator.

$N, iWL, oWL, k_{min}, k_{max}, l_{min}, l_{max}, m_{min}, m_{max} \leftarrow input$
$A_{min} \leftarrow 0$, $A_{max} \leftarrow 2^{N-1}$
$B_{min} \leftarrow 0$, $B_{max} \leftarrow 2^{N-1}$
$num\_msb\_errors \leftarrow 0$, $max\_num\_errors \leftarrow 2^{2N}$
for $m = m_{min}$ **to** m_{max} **do**
  for $l = l_{min}$ **to** l_{max} **do**
    for $k = k_{min}$ **to** k_{max} **do**
      for $A = A_{min}$ **to** A_{max} **do**
        for $B = B_{min}$ **to** B_{max} **do**
          $A_{trunc} \leftarrow \text{get\_msb}(A, iWL)$
          $B_{trunc} \leftarrow \text{get\_msb}(B, iWL)$
          $E(A, B) \leftarrow A \star B$ {Compute the expected result out of the original unit}
          $X(A, B) \leftarrow (A_{trunc} \cdot k + m) \star (B_{trunc} \cdot k + m) + l$ {Compute the approximate result from the truncated operands}
          $E_{msb}(A, B) \leftarrow \text{get\_msb}(E(A, B), oWL)$
          $X_{msb}(A, B) \leftarrow \text{get\_msb}(X(A, B), oWL)$
          $msb\_error(A, B) \leftarrow E_{msb}(A, B) - X_{msb}(A, B)$ {Example of the objective function to obtain the best match in the approximation MSbs}
          if $msb\_error(A, B) \neq 0$ **then**
            $num\_msb\_errors \leftarrow num\_msb\_errors + 1$
          end if
        end for
      end for
      if $num\_msb\_errors < max\_num\_errors$ **then**
        $max\_num\_errors \leftarrow num\_msb\_errors$
        $best\_k \leftarrow k$
        $best\_l \leftarrow l$
        $best\_m \leftarrow m$
      end if
    end for
  end for
end for
return $best\_k$, $best\_l$, $best\_m$, max_num_errors

A	B	Result	MSB
1001	1111	10000111	1
1000	1100	01100000	0

Table 4.3.: Results of 2x2-bit multiplications using the original and truncated operands.

l	m	Apx Result	Apx MSBs
l = 0	m = 0	01100000	0
l = 15	m = 1	10000100	1

Table 4.4.: Results produced by two different approximations for 2x2-bit multiplications using truncated operands.

Consequently, certain correct values are likely to be replaced with values for other approximations. To minimise this undesirable artifact, it is considered using another approximation function, to be stored in the replacement ROM, different from the one in the detection ROM. The incentive for this is the fact that it is possible to produce approximations with an absolute error smaller than the error from the detection approximation, either by using a different approximation function, different word lengths, or both.

This is exemplified by computing the Most Significant Bytes (MSBs) for two different approximations using truncated operators and comparing them against the expected result. The example considers a ROM with 2-bit input operands, and one MSB at the output. Table 4.3 presents the results of 2x2-bit multiplications using the original and truncated operands, and its MSBs. Table 4.4 presents the results and the MSBs for two approximations, using different approximation coefficients in equation 4.2. This example shows that the MSB from a truncated approximation is deviated from the expected result. By allowing a wrong approximation to be replaced with a value from another approximation would produce the expected, and correct, result.

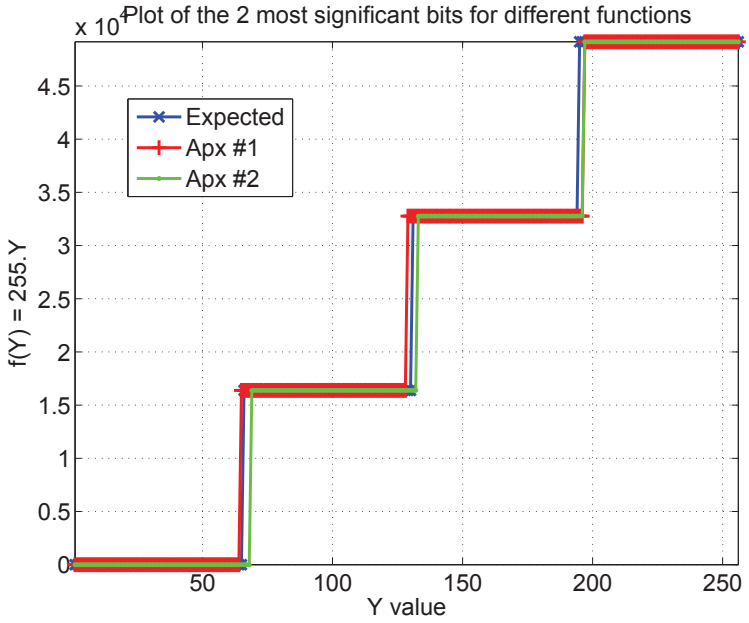

Figure 4.4.: Values of the 2 MSBits for the expected result and approximation functions 1 and 2.

Figure 4.4 illustrates the case for a multiplication of a constant 8-bit unsigned value with a vector $(255 \times Y)$, where Y is a vector with a sequence of 8-bit unsigned values. The blue line is the expected, and correct, result for the 2 MSBits. The green line is the approximation function 1 (APX #1) and the red line the approximation 2 (APX #2). It is observable that, in many cases, when the APX #1 (red) is different than the expected (blue), if those points in APX #1 were to be replaced with the APX #2 points, that would make the MSBits correct. Notwithstanding, other values would be replaced with values different from the expected because of the approximations errors in the replacement ROM.

A process to minimise the value of the objective function, in the case of using the linear approximation, is to tradeoff accuracy for resolution. For this reason, for the same ROM size, it is preferable to have many input bits with few output bits to detect as many correct MSBits as possible, and a few input bits with many output

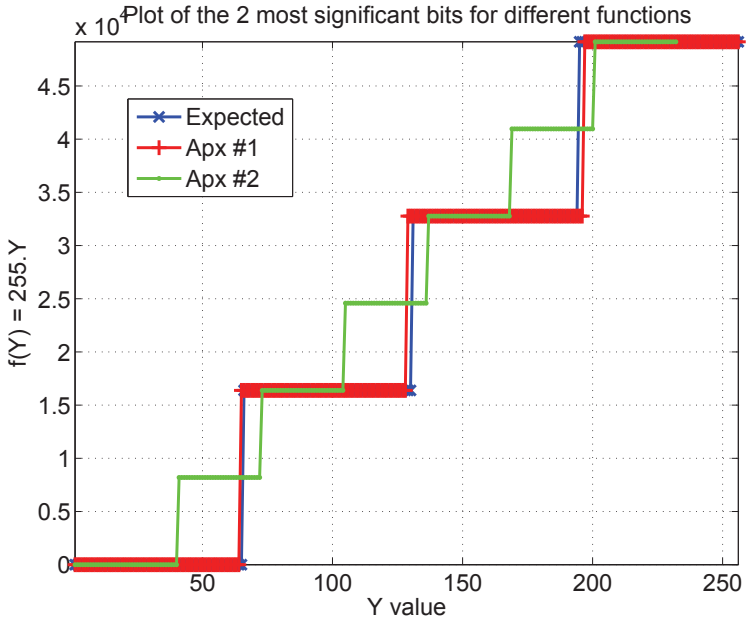

Figure 4.5.: Values of the 2 MSBits for the expected result and approximation function 1 and 3 MSBits for approximation function 2.

bits to replace with the smallest error possible. As a result, the approximations will produce more errors but with smaller magnitude. Figure 4.5 exemplifies this by using two approximation functions targeting the two MSBits in a ROM-XOR 8:16/6:2/5:3 RPR multiplier. It is discernible that there are more mismatches, but the discrepancy between the expected and the approximation is smaller than in the previous example.

Even though there are results incorrectly detected as wrong, at the output of the ROM-XOR RPR units, it is anticipated that they are ineligible when compared to the timing errors, in the MSBits, of the original units.

Ideally, it would be enticing to have all possible values in the ROM, instead of truncating the operand's word length, but the resources needed would cause such implementation unfeasible.

The resources needed to store an approximation function in ROM depend exclusively on the word length of the inputs and the output considered for the approximation, regardless of the function being used. Conversely, the framework determines which word lengths are possible to use given a limited resource budget.

The number of BRAMs required to implement the ROM used in the approximation for different numbers of input and output bits is given by the following equation 4.3:

$$NUM\_BRAMs = \left\lceil \frac{2^{(iWL_A + iWL_B)} oWL}{BRAM\_SIZE} \right\rceil \tag{4.3}$$

In this expression iWL_A, iWL_B and oWL are the word lengths of the inputs and output of the BRAM while *BRAM_SIZE* refers to the number of bits available per BRAM on the target FPGA.

Table 4.5 shows the number of BRAMs for different word lengths for various Cyclone devices. This table was constructed considering the BRAM sizes available across the different Cyclone families, disregarding memory configuration limitations and assuming complete occupation of the BRAMs. It is important to state that different device families have different BRAM sizes: Cyclone III and IV have 9k bits and Cyclone V has 10k bits. It's pertinent to mention that the proposed RPR framework supports different word lengths for each input of the approximation function (asymmetric operators).

Moreover, in the particular case of using only one ROM, to hold both detection and replacement approximations, the result would be the same as using always the LSBits from the original unit and replace the MSBits with the approximation. This would simplify the design by eliminating the multiplexer, and a ROM, in the design, thus increasing the maximum clock frequency of the unit. Nevertheless, such scheme is not presented in this work even though the proposed framework could easily be adapted to generate the approximations and evaluate the error/area tradeoff.

			Cyclone	
			III, IV	V
Input WL	Output WL	ROM bits	M9K	M10K
8, 8	1	65536	8	7
8, 8	2	131072	15	14
8, 8	3	196608	22	20
8, 8	4	262144	30	27
7, 7	1	16384	2	2
7, 7	2	32768	4	4
7, 7	3	49152	6	5
7, 7	4	65536	8	7
6, 6	1	4096	1	1
6, 6	2	8192	1	1
6, 6	3	12288	2	2
6, 6	4	16384	2	2
5, 5	1	1024	1	1
5, 5	2	2048	1	1
5, 5	3	3072	1	1
5, 5	4	4096	1	1
5, 5	7	7168	1	1
4, 4	16	4096	1	1
5, 7	2	8192	1	1
4, 8	1	4096	1	1

Table 4.5.: Number of BRAMs (M9K/M10K) per input and output word lengths required to implement the approximation ROM in different Cyclone FPGAs.

Scope	Objective Function
Best MSBit match	$max\{i\} : \forall i \in oWL, ori(i) = apx(i)$
Min. error variance	$min\left\{var\left(\sum_{i=oWL} ori(i).2^i - \sum_{i=oWL} apx(i).2^i\right)\right\}$
Min. mean error	$min\left\{\frac{1}{N}\sum_{N}\left(\sum_{i=oWL} ori(i).2^i - \sum_{i=oWL} apx(i).2^i\right)\right\}$
iWL: word length of the input from the original unit and the approximation oWL: word length of the output from the original unit and the approximation ori: result from the original unit apx: result from the approximation function N: number of values tested	

Table 4.6.: Objective functions for errors at the output of the RPR unit.

4.2.3. Error Function Minimisation

Up until now, as the RPR has been presented, it only concerns the minimisation of the errors by producing approximations that faithfully resemble the MSBits from the original arithmetic units. However, in some cases where resource constraints or complexity of the arithmetic units doesn't allow to compute all the expected results, it may be desirable to specify different objective functions for the generation of the approximations by the framework.

At present, the framework supports the following objective functions, derived from different error metrics between the expected and the approximation results. Table 4.6 summarises some of these functions for 3 application scopes.

The first case can be achieved either by exhaustively trying different values for the approximation coefficients, of the input bits and linear approximation coefficients, or from the mode of the expected MSBits, whereas the others are achieved only by exhaustive search. In all cases, the framework investigates sets of linear approximation coefficients that minimise each objective function through exhaustive search.

As a result of using approximations different conditions can occur at the output of the multiplier:

1. No error: the value at the output of the multiplier is the expected result and it's passed to the output of the multiplexer;

2. False Positive: the value at the output of the multiplier is the expected result but it's detected as wrong and the approximation is passed to the output of the multiplexer;

3. False Negative: the bits at the output of the multiplier that are compared with the ones from the approximation are the same, but the rest of the result is wrong. In this case the error is smaller than the MSBits protected, except if this methodology is instead applied to Most-Prone-to-Error bits [1].

4. Error: the value at output of the multiplier isn't the expected result and it's detected as wrong and the correct approximation is passed to the output of the multiplexer.

Nevertheless, it should be remembered that these approximations are considered in the error-free regime. Whenever they're applied to the error-prone regime they'll have different importance. Hence, it's not guaranteed that the best error-free approximation will produce the best results. Producing function approximations with specific behaviour on error-prone regimes implies additional research on the simulation and test of the RPR units, which is left for future work.

4.3. ROM-XOR RPR Arithmetic Operators

The proposed RPR framework has been introduced for generic arithmetic units, and it doesn't depend on the actual problem. The specifics on how different basic arithmetic units are supported by the proposed RPR framework are explained in this

[1] Most-Prone-to-Error bits are defined as the bits that are more likely to fail, and they may not be the MSBits. They are determined from characterisation of the arithmetic unit with a specific data set. Arithmetic units with different architecture, and input data with different distributions, are likely to exhibit different levels of error across their output bits.

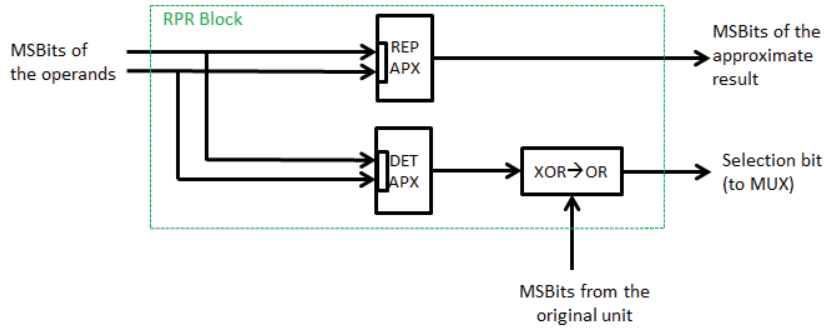

Figure 4.6.: Diagram of the circuit attached to the arithmetic operators to produce the approximations and signaling of which result to use.

section. The proposed framework is demonstrated for the application of redundancy on arithmetic units usually used in DSP designs, namely adders, multipliers and multiplier-accumulators. Furthermore, the demonstration is conducted adopting the objective function that minimises the mismatches in the MSBits between the expected and the actual results.

Although there are many possible synthesis implementation variants for each unit, this framework is transparent to them and assumes the default synthesis implementation from the vendor tool. Moreover, the proposed framework can be extended to different arithmetic operators and scaled to problems of different sizes. Figure 4.6 shows the details of the block that is attached to the arithmetic operators to provide the approximation, and the indication to use either the result from the original arithmetic unit or the approximate result. The inputs of the block are the **MSBits!**s (**MSBits!**s) from the truncated operands, and the output of the original arithmetic unit. The outputs are the bit, that goes to the multiplexer in figure 4.2, and the approximate result.

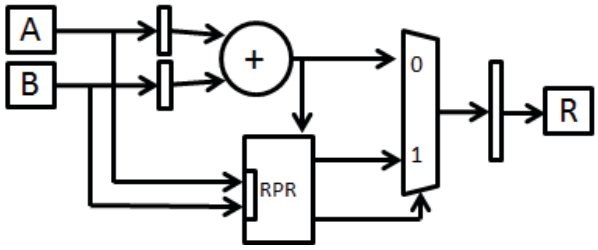

Figure 4.7.: Diagram of an adder circuit with the proposed RPR scheme.

4.3.1. Adder

The adder is one of the building blocks of arithmetic circuits. Even though it's a simple block, for large word lengths, carry propagation becomes the bottleneck in terms of throughput.

The diagram showed in figure 4.7 applies the proposed RPR on the adder operator. The block identified with RPR is a simplification of the detection and replacement ROMs, plus the XOR and OR circuitry. For this operator the linear approximation function considered is as follows:

$$apx(a, b, k) = a + b + k \tag{4.4}$$

where a and b are the truncated input arguments and k is a bias to compensate for the truncation of the input operands. The proposed RPR framework searches for the value of k which minimises the objective function (table 4.6) between the expected and the approximation MSBits.

The search for the k is presented as a particular case of the algorithm 1 and is as follows. A value is assigned to k, from an arbitrary set, and all possible values for the operands in the original adder, and in the approximation function, are tested to evaluate the error from the approximation function. The process is then repeated for all possible values of k. The selected value for k for the implementation is the one that minimises the objective function.

To illustrate the application of the RPR framework, an 8-bit unsigned adder is considered. The result of an approximation, following (4.4), for a 5-bit inputs is presented in figure 4.8. In this figure it is noticeable a staircase-like shape, typical of quantisation introduced due to truncated operands. The k values that minimise the objective function for the approximation errors are summarised in table 4.7. The table shows the word length of the inputs (iWL) and output (oWL), total number of bits to be saved in ROM, the k values that minimise the objective function, the number of approximation errors, and the percentage of sets of correct MSBits in the approximate results. It's possible to verify that the results are as expected: more input bits generate more correct results, as well as less output bits. Moreover, even for larger ROMs there's no approximation that generates all output bits as expected. Figure 4.9 shows the discrepancy in the values of the MSBits for a particular case of a ROM-XOR 8:9/5:2/5:2 RPR adder.

To improve upon these results, a second approximation function is used as a generator of the bits replaced in the RPR adder, while the other approximation is used to detect errors, as explained previously. In addition to searching for an approximation constant k, the proposed framework optimises two approximation functions, with two approximation coefficients k_{det} and k_{rep}, simultaneously.

Table 4.8 shows the linear approximation coefficients to produce the same MSBits as the original unit for different ROM sizes. Comparing with the single approximation, it is evident that there's an increase in the ROM size, however the output of the RPR 8-bit adder is the same as the 8-bit original adder, when operating in the error-free regime. Figure 4.10 shows the output of a ROM-XOR 8:9/5:2/5:2 adder without any approximation errors.

iWL	oWL	ROM bits	k	Apx. Errors	% Correct
5,5	1	1024	0:7	896	98.63
5,5	2	2048	0:7	1792	97.26
5,5	3	3072	0:7	3584	94.53
6,6	1	4096	0:3	384	99.41
6,6	2	8192	0:3	768	98.82
6,6	3	12288	0:3	1536	97.65
7,7	1	16384	0:1	128	99.80
7,7	2	32768	0:1	256	99.60
7,7	3	49152	0:1	512	99.21

Table 4.7.: Resources and approximation results for different implementations of a single approximation function for a ROM-XOR 8:9/iWL:oWL/iWL:oWL RPR adder.

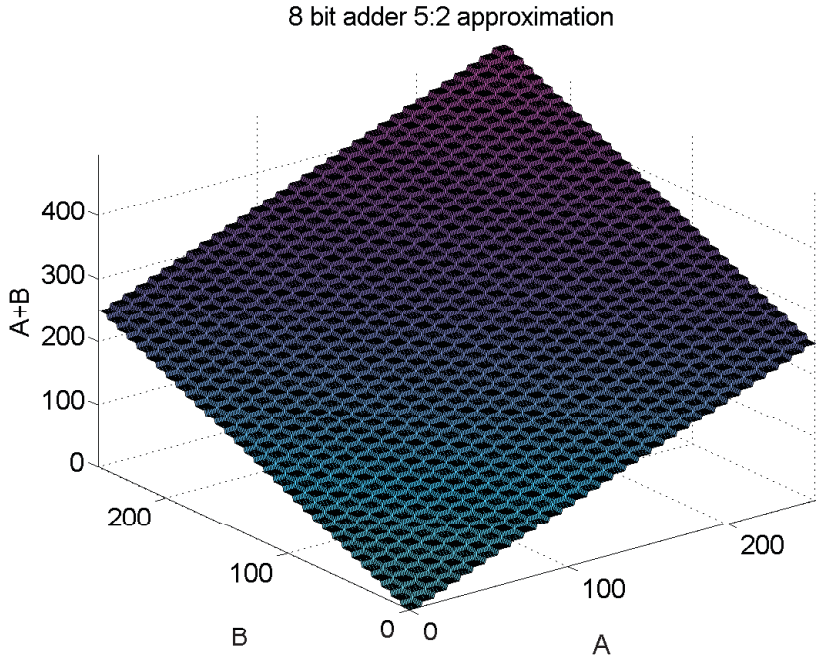

Figure 4.8.: Example of an approximation produced for a 8:9/5:2/5:2 ROM-XOR RPR adder.

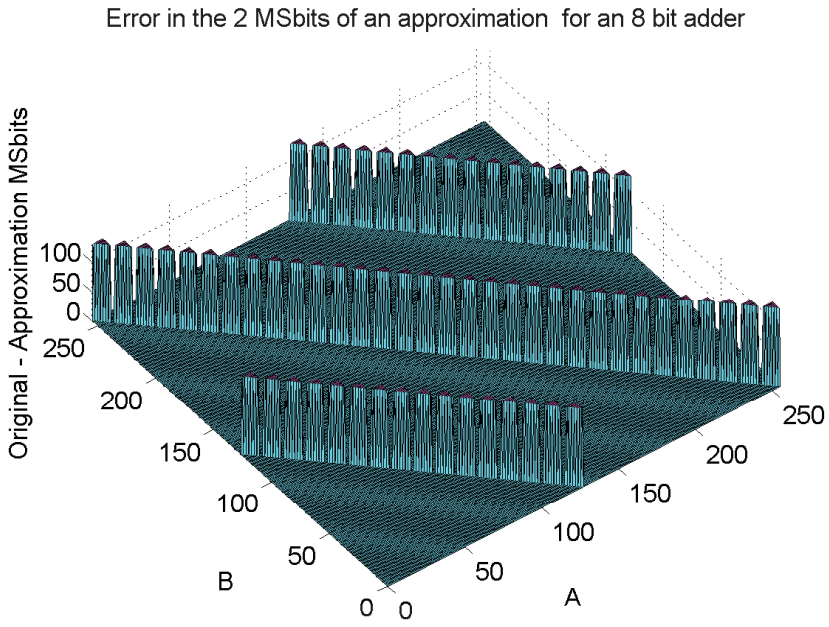

Figure 4.9.: Difference in the MSBits between the detection approximation and the expected result for an 8-bit adder.

iWL	oWL	ROM bits	k_{det}	k_{rep}	Apx. Errors	% Correct
5,5	1	2048	8	0	0	100
5,5	2	4096	8	0	0	100
5,5	3	6144	8	0	0	100
6,6	1	8192	4	0	0	100
6,6	2	16384	4	0	0	100
6,6	3	24576	4	0	0	100
7,7	1	32768	2	0	0	100
7,7	2	65536	2	0	0	100
7,7	3	98304	2	0	0	100

Table 4.8.: Correctness of the MSBits for different coefficients of the approximation function for ROM-XOR 8:9/iWL:oWL/iWL:oWL RPR addition, using different linear approximations.

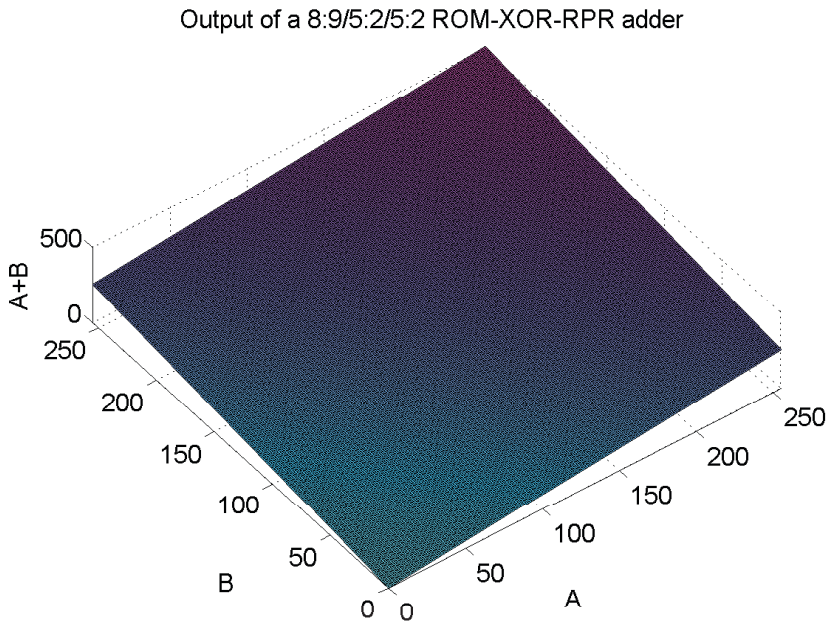

Figure 4.10.: Results produced by a ROM-XOR 8:9/5:2/5:2 RPR adder.

4.3.2. Multiplier

Like addition, multiplication is widely used and present nearly in all DSP systems. Synthesis of the circuit to implement this unit has more, and longer, paths than the adder, and often holds the critical-paths of DSP designs. The only difference in the architecture, when compared to the RPR adder, is the original arithmetic unit. For this arithmetic operation, a new approximation function is studied to minimise the error function.

The implementation of multiplication using the proposed RPR follows the architecture presented earlier and is illustrated in figure 4.11. The proposed RPR architecture doesn't make a distinction between different multiplier architectures, therefore it can be implemented in technologies alternative to FPGAs.

Equation 4.5 is the function chosen to approximate the multiplication.

$$apx(a,b) = (a+m)(b+m) + l \tag{4.5}$$

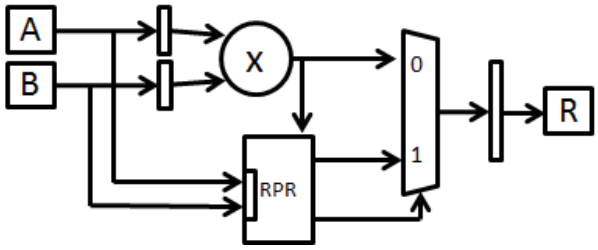

Figure 4.11.: Diagram of a multiplier circuit with the proposed RPR scheme.

In this case a and b are the truncated inputs, and m and l are the approximation function coefficients. These coefficients exist to overcome the distortions imposed by the approximation computed from truncated input values, such as negative bias.

Table 4.9 summarises the values of l and m to use in the detection approximation function for the different resource utilisations. Figure 4.12 shows the values of the approximation function for all possible 5-bit input values. Like in the RPR adder, the approximation exhibits a staircase like shape, typical of quantisation introduced due to truncated operands. Figure 4.13 shows the discrepancy in the values of the MSBits for a particular case of a ROM-XOR 8:16/5:2/5:2 RPR multiplier. As expected, it is observable from table 4.9 that the greater the number of input bits, and the least number of output bits, used in the approximation the greater is the number of correct approximations.

Using two distinct approximations, for detection and correction, allows to set the correct MSBits whenever the detection detects them as wrong. Table 4.10 shows the linear approximation coefficients to produce the same MSBits as the original unit for different resource budgets. Figure 4.14 shows the output of a ROM-XOR 8:16/5:2/5:2 RPR multiplier without approximation errors, when compared to the expected result from the original multiplier.

iWL	oWL	m_{det}	l_{det}	Apx. Errors	% Correct
5,5	1	0	1216	338	99.48
5,5	2	3	103	1080	98.35
5,5	3	3	103	2613	96.01
6,6	1	0	512	182	99.72
6,6	2	1	151	528	99.19
6,6	3	1	151	1279	98.04
7,7	1	0	128	74	99.88
7,7	2	0	128	240	99.63
7,7	3	1	-151	591	99.09

Table 4.9.: Correctness of the MSBits for different approximation coefficients for ROM-XOR 8:16/iWL:oWL/iWL:oWL RPR multiplication.

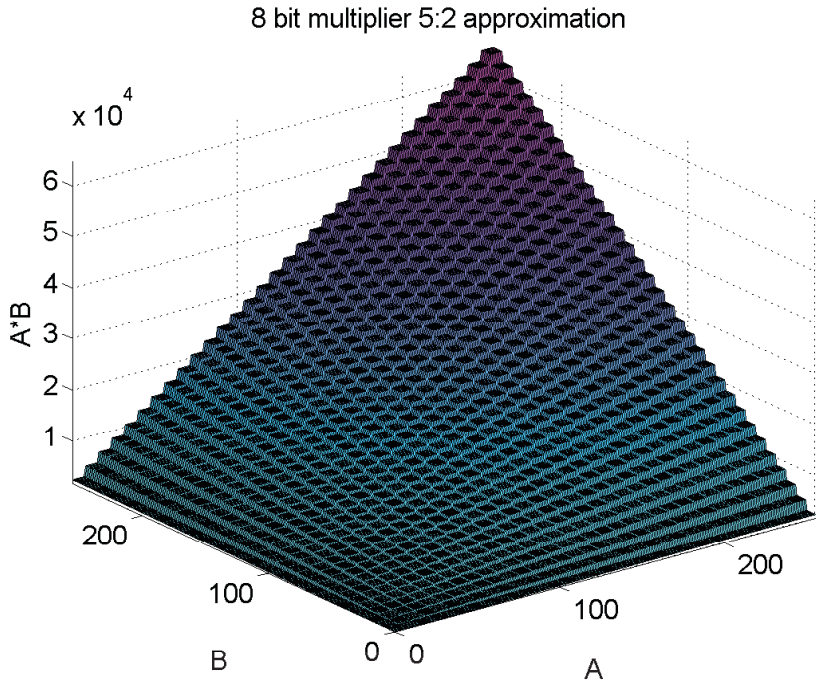

Figure 4.12.: Example of an approximation produced for a ROM-XOR 8:16/5:2/5:2 RPR multiplier.

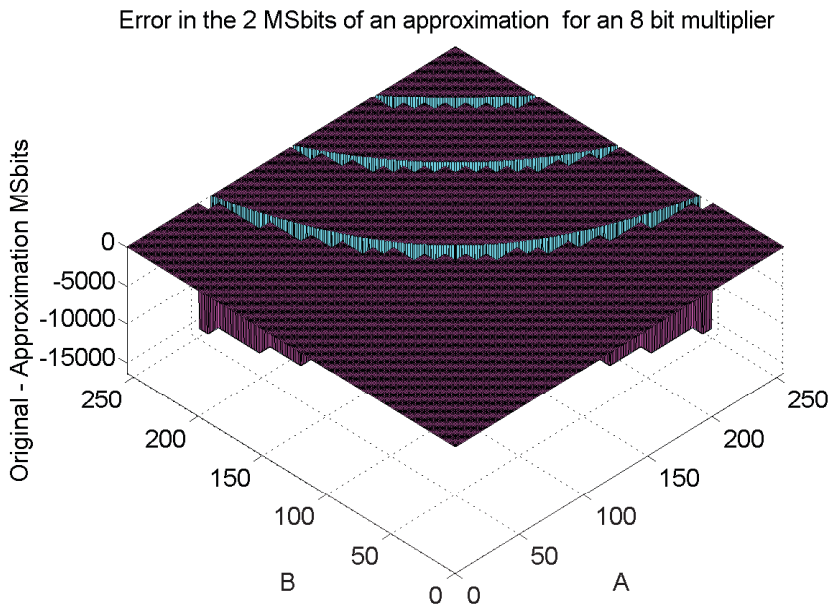

Figure 4.13.: Difference between the detection approximation and the expected MSBits at the output of the multiplier.

iWL	oWL	m_{det}	l_{det}	m_{rep}	l_{rep}	Apx. Errors	% Correct
5,5	1	1	2015	0	0	0	100
5,5	2	3	1767	0	0	0	100
5,5	3	5	887	0	0	0	100
6,6	1	1	723	0	0	0	100
6,6	2	1	887	0	0	0	100
6,6	3	1	887	0	0	0	100
7,7	1	0	356	0	0	0	100
7,7	2	0	444	0	0	0	100
7,7	3	0	448	0	0	0	100

Table 4.10.: Correctness of the MSBits for different approximation coefficients in an ROM-XOR 8:16/iWL:oWL/iWL:oWL RPR multiplication.

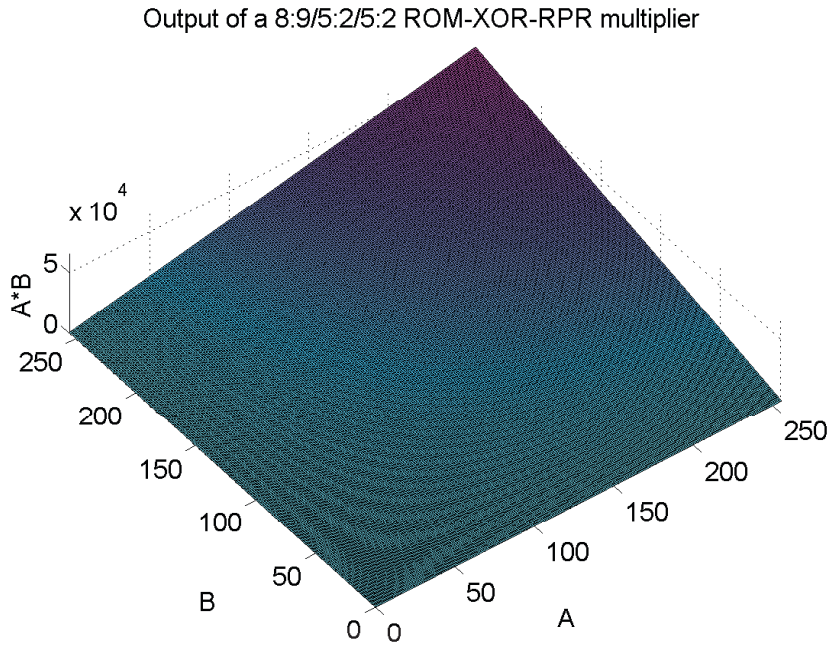

Figure 4.14.: Results produced by a ROM-XOR 8:9/5:2/5:2 RPR multiplier.

4.3.3. Multiplier-Accumulator

Implementing a multiplier-accumulator using the proposed RPR scheme involves adding the RPR circuitry to the operator without changing its functionality, that is to compute new values in each clock cycle. As this operator exhibits extra complexity, when compared to the previous operators, the approximation functions have to hold a new intermediate result in regard for the preservation of the circuit's functionality. Figure 4.15 shows the block diagram of a folded multiply-accumulate unit with RPR. The RPR block is attached to the datapath in a similar pattern as the previous units, but uses a new approximation function, shown in equation 4.6:

$$apx(a, b, m, l) = (a + m)(b + m) + r + l \tag{4.6}$$

In this approximation function, a, b, and r are the truncated input operands, and m and l the approximation coefficients which are determined by the RPR framework

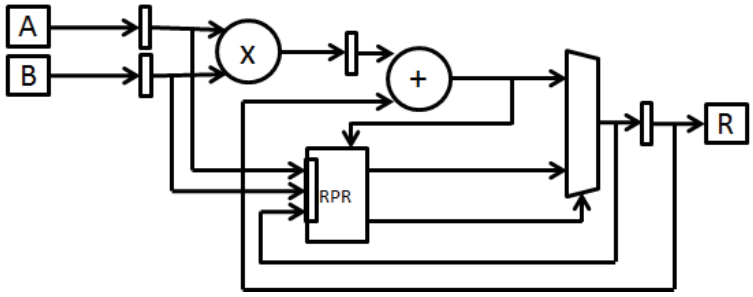

Figure 4.15.: Diagram of a rolled multiply-accumulate circuit with the proposed RPR scheme.

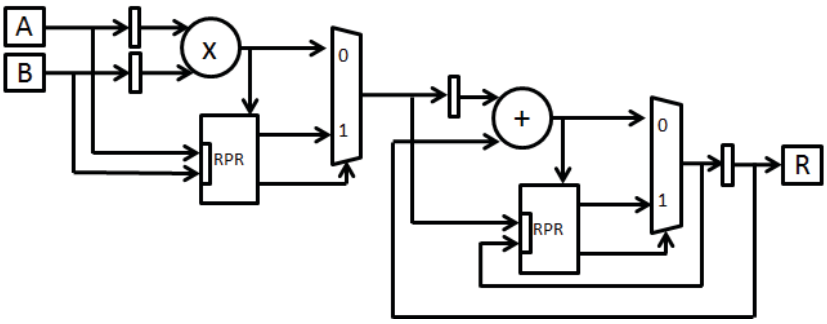

Figure 4.16.: Diagram of a rolled multiply-accumulate circuit using basic arithmetic units with the proposed RPR scheme.

to minimise the error function.

Alternatively, the multiply-accumulate unit with RPR can be constructed at the expense of the previous arithmetic RPR units, as shown in figure 4.16. In this case, each operator has their own RPR block attached. Moreover, when there's a significant gap in the delay of the critical-paths in the adder and in the multiplier, or when it's foreseen that one of these units will never experience timing violations in their paths, it may be preferable to apply the RPR block to the most prone to error unit, hence saving circuit resources. This way the functionality of the circuit is preserved while re-utilising existing units, thus avoiding to compute new approximations, on the assumption that the input data follows the same distribution.

4.4. Reduced-Precision Redundancy Evaluation

To evaluate the proposed RPR framework its performance is compared against existing implementations of stand-alone arithmetic units, and on a DSP application, with and without RPR. Other design techniques for arithmetic circuits such as carry look-ahead adders and Karatsuba multipliers offer reduced/fast carry chains. Circuits with fast carry chains achieve less propagation delay than the standard ripple-carry chain, which benefits from dedicated carry lines inside the LEs on the FPGA. It is foreseen that the savings in delay aren't sufficient to run these arithmetic units with the same clock rates as the other arithmetic units considered in this evaluation, without producing timing errors (their MSBits are still the most critical-paths in the circuit). Given the aforementioned, or because they require extra latency, they haven't been considered.

The targeted application is the linear projection, which has been introduced in the background chapter of this thesis. The stimulus used by the evaluation is a pseudo-random vector with 20k samples, for the stand-alone arithmetic units, and a set of 50x40 pixels grayscale images. In the implementation, data is represented using 9-bit sign-magnitude representation.

The aforementioned performance for each circuit is compared in terms of clock frequency, circuit resources, errors and power consumption. It is expected to observe that the arithmetic units using RPR can perform with less error (absolute error, mean and variance) than the other units when operating beyond the error-free region.

For precise measurements the operating conditions of the device were kept constant, at 20 degrees Celsius, through the usage of a cooling element on top of the FPGA device and an external power supply. All tests were carried on a Cyclone III FPGA from Altera [1], on a DE0 board from Terasic [89]. More details about the experimental setup can be found in appendix A.2 of this thesis.

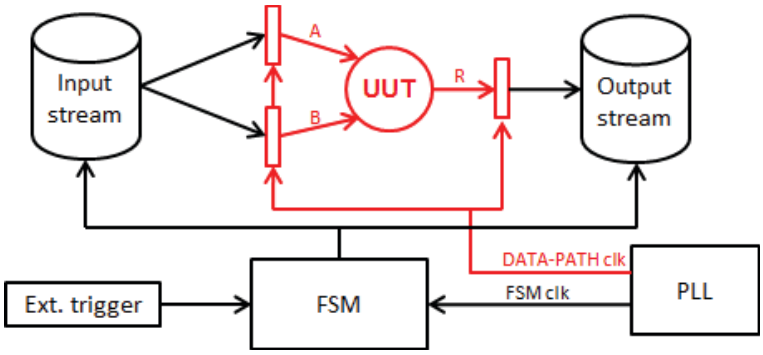

Figure 4.17.: Block diagram of the test circuit for RPR units under variation.

4.4.1. Circuit Architecture

To facilitate the implementation, and minimise the influence of external circuitry in the evaluation of the design, all designs were placed inside a test circuit. In addition, this test circuit promotes the reduction of placement and routing variation in the auxiliary blocks of the design. Hence, it guarantees that the delays of the most critical-paths reside inside the units under test.

The test circuit is presented in figure 4.17. It controls the test execution (FSM), and provides the unit under test with a constant stream of data. The datapath of the circuit, which holds the unit under test (UUT), is highlighted in red. This part of the circuit is clocked at twice the clock frequency of the FSM. Furthermore, this test circuit can support the characterisation of many units in parallel.

Example of the floor plan for the test circuit of an RPR multiplier implemented on a Cyclone III FPGA is presented in figure 4.18. The different blocks are highlighted, being the unit under test identified with *red_mult0*. The maximum clock frequency for the supporting blocks of this circuit, according to the synthesis tool is 700 MHz, notwithstanding, actual tests reveal that it supports clock frequencies up to 930 MHz, as illustrated in figure 4.19.

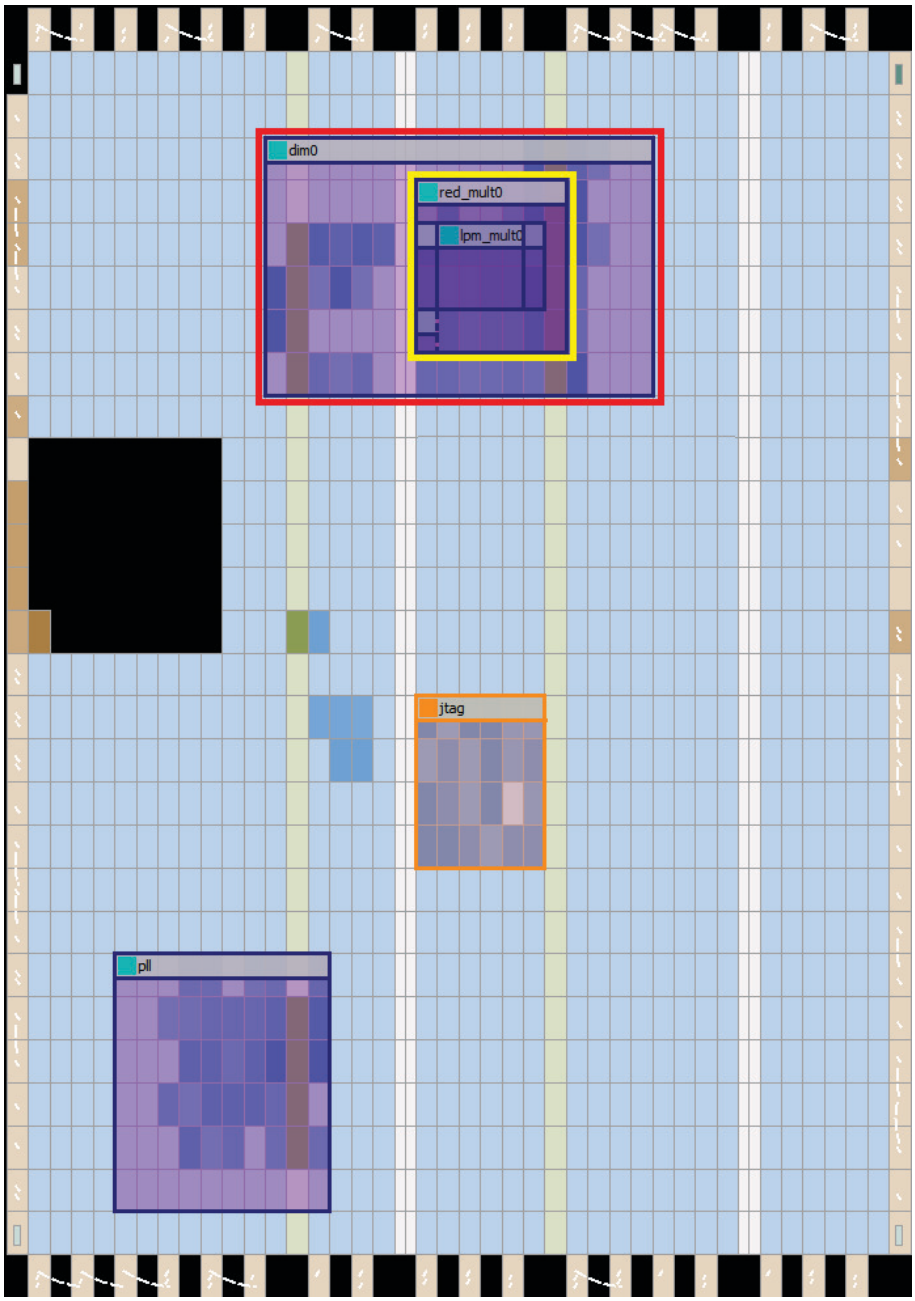

Figure 4.18.: Floor plan of the test circuit (red) for an RPR multiplier (yellow).

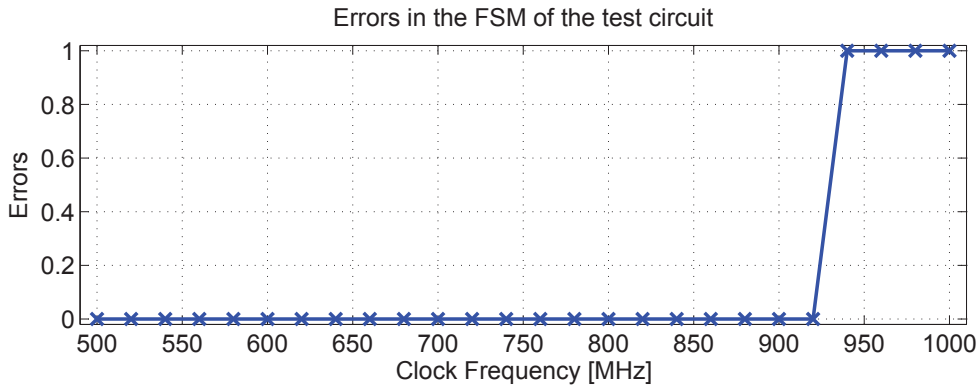

Figure 4.19.: Errors in the supporting blocks of the test circuit for different clock frequencies on a DE0 board.

16-bit Adder	Synthesis Tool
NO RPR	411 MHz
ROM-XOR RPR	248 MHz

Table 4.11.: Maximum clock frequency reported by the synthesis tool of 16-bit adders.

4.4.2. Adder

Implementation of adders on FPGAs is facilitated by their architecture which offers dedicated carry propagation lines, thus avoiding routing through the LEs. Because of its simplicity and reduced number of LEs it is the fastest arithmetic operator possible to implement on an FPGA.

Implementing an adder with the proposed ROM-XOR RPR adds extra complexity to the circuit impacting its maximum clock frequency. Hence, unviable to compete against a typical adder in terms of maximum clock frequency. The maximum clock frequency results for each implementation of a 16-bit adder are in table 4.11.

The limitation in the maximum clock frequency of the ROM-XOR RPR 16-bit adder is due to the inclusion of BRAMs in the datapath. The total delay of the critical-path in the RPR design is 4 ns, which is greater than the delay of a 16-bit adder, that is 2.5 ns. Another limitation is the number of extra resources required

Adder	NO RPR	ROM-XOR RPR	Extra LEs
16-bit	18	175	89.71 %
24-bit	26	207	87.43 %
32-bit	33	235	85.95 %
36-bit	37	252	85.31 %
48-bit	51	299	82.94 %
64-bit	68	363	81.26 %

Table 4.12.: Resources, in LEs, taken by the different adder implementations.

to implement RPR. Table 4.12 summarises the resources required by the NO RPR and ROM-XOR RPR adders using different word lengths. On average, introducing redundancy requires extra 85% in LEs, besides the BRAMs. Figure 4.20 shows the results on the errors for the RPR adder of the typical adder without RPR.

4.4.3. Multiplier

LUT-Based Multiplier

Three types of 8-bit unsigned multipliers were implemented for performance comparison: NO RPR, LUT-SUB RPR and ROM-XOR RPR. Figure 4.21 shows the variance and mean of the error between the expected value and the value read from the board for each multiplier implementation. As anticipated, the maximum throughput of the ROM-XOR RPR scheme is close to the multiplier without any redundancy, and it performs better than any other design under extreme over-clocking, e.g. 300 MHz. Only above 340 MHz the ROM-XOR RPR multiplier achieves the same level of error variance as the other multipliers. Notwithstanding, the mean error of the RPR multiplier remains close to zero. Tables 4.13, 4.14 and 4.16 show the maximum clock frequencies, the resources occupied and the power consumed by the different 8x8 LUT-based multiplier implementations. In more detail, 4.15 shows the resources, apart from the extra BRAMs, required for NO RPR and ROM-XOR RPR multiplier implementations, and its relative increase. To add ROM-XOR RPR

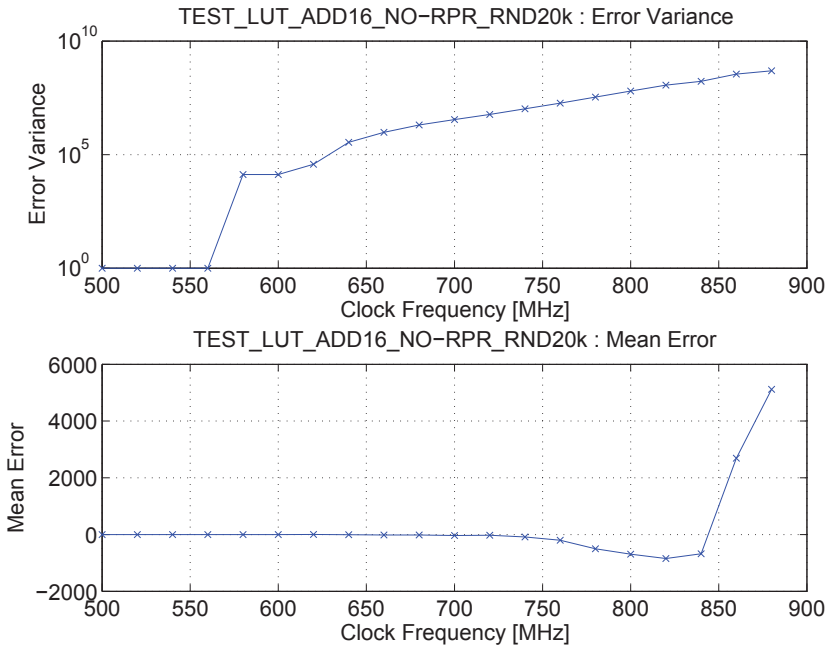

Figure 4.20.: Error variance and mean error of an 16-bit adder without RPR

to a multiplier requires extra 111 LEs on average. For small multipliers this extra LEs can be prohibitive to implement, but as the word length increases, the resilience in the multiplier requires relatively few extra resources. Although not presented, the same tend is expected for extra power consumption.

Figures 4.21, 4.22, 4.23 and 4.24 show the errors, and their distribution, for the same test vector and clock frequency on the three multipliers. For the same clock frequency it is observable, in figure 4.25, that the multiplier without redundancy produces less errors but most of them have a large magnitude. The LUT-SUB RPR multiplier, because of the long delay on its critical-paths, it produces many large errors. On the other hand, the ROM-XOR RPR produces more errors than the first one but they are smaller in value. This can be advantageous for applications that can tolerate small errors in computations masked as sensor noise allowing the application to produce valid, but slightly deviated, results.

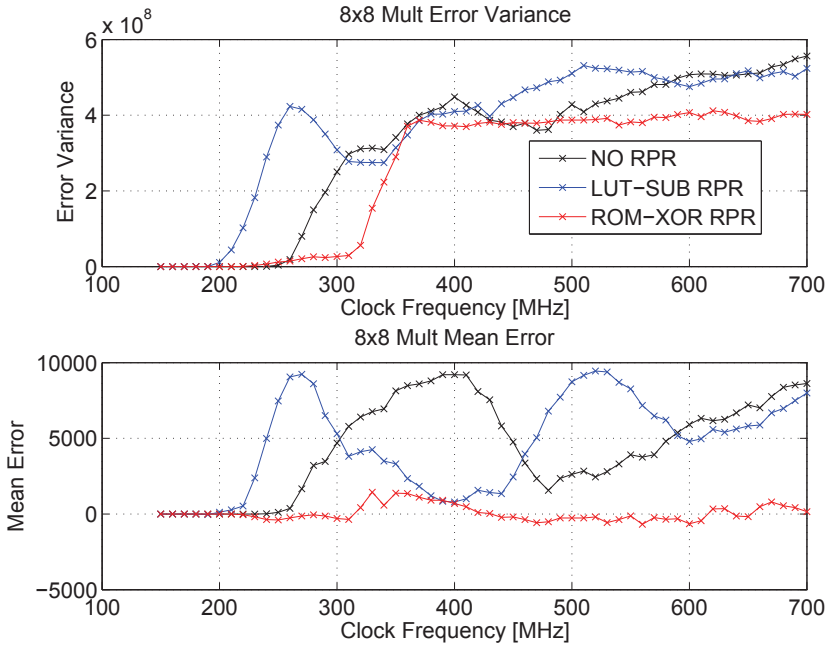

Figure 4.21.: Variance and mean error at the output of an 8x8 bit unsigned LUT-based multipliers at different clock frequencies.

8-bit Multiplier	Synth. Tool [MHz]	FPGA Test [MHz]
NO RPR	175	230
LUT-SUB RPR	133	170
ROM-XOR RPR	149	210

Table 4.13.: Maximum clock frequency, in MHz, for operation without errors.

8-bit Multiplier	LEs	BRAMs
NO RPR	101	0
LUT-SUB RPR	158	0
ROM-XOR RPR	206	2

Table 4.14.: Resources taken by the different multiplier implementations.

Multiplier	NO RPR	ROM-XOR RPR	Extra LEs
8-bit	101	206	103.96 %
16-bit	345	452	31.01 %
24-bit	721	836	15.95 %
32-bit	1213	1321	8.90 %
36-bit	1513	1627	7.53 %
48-bit	2605	2722	4.49 %

Table 4.15.: Resources, in LEs, taken by the different multiplier implementations.

8-bit Multiplier	System [mW]	Multiplier [mW]
Test Circuit	46.56 / 83.6	-
NO RPR	52.44 / 97.2	5.88 / 13.6
ROM-XOR RPR	57.48 / 105.7	10.92 / 22.1

Table 4.16.: Power consumed by the different multiplier implementations, and respective test circuits, at 1200 mV, 20°C, 100 MHz and 600 MHz.

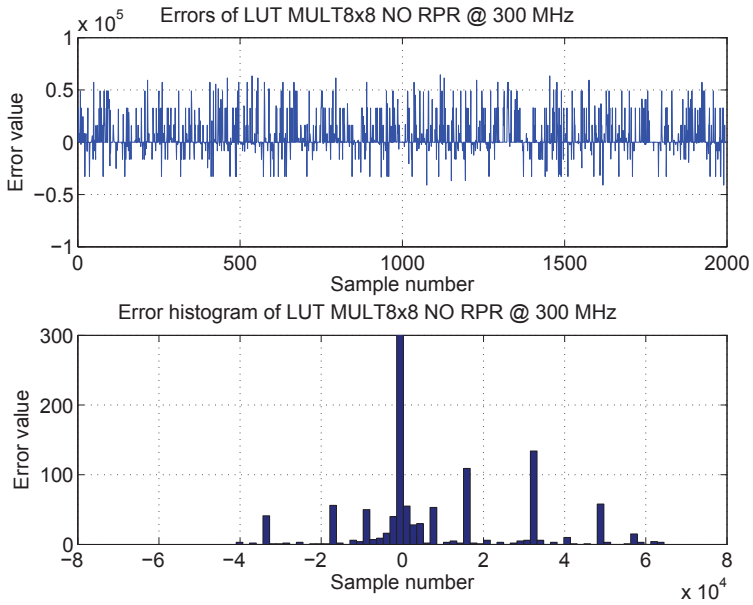

Figure 4.22.: Errors in results (top) and error histogram (bottom) at the output of a NO RPR 8-bit multiplier at 300 MHz.

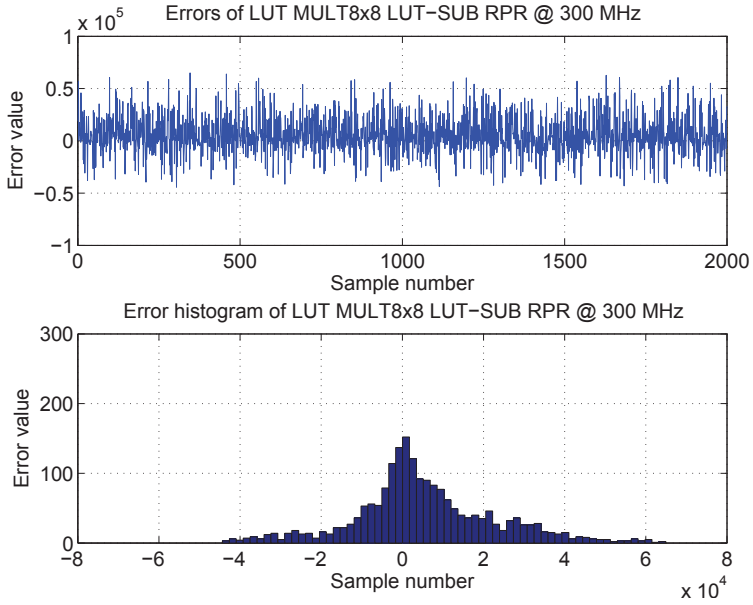

Figure 4.23.: Errors in results (top) and error histogram (bottom) at the output of a LUT-SUB RPR 8-bit multiplier at 300 MHz.

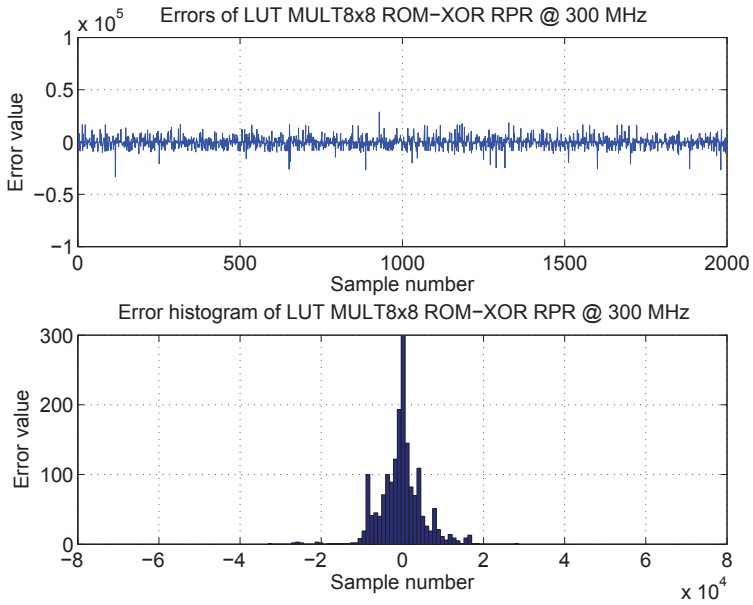

Figure 4.24.: Errors in results (top) and error histogram (bottom) at the output of a ROM-XOR RPR 8-bit multiplier at 300 MHz.

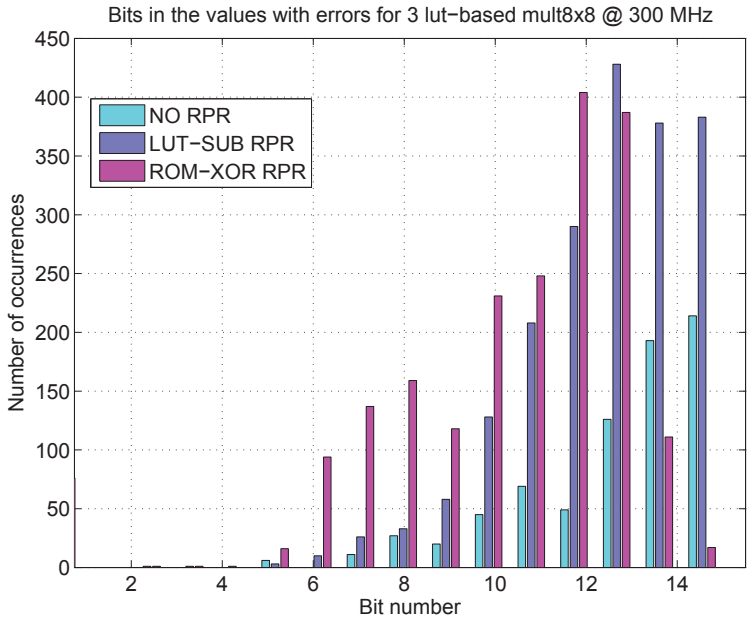

Figure 4.25.: Histogram of the bits present in the values with errors at the output of the three LUT-based 8-bit multipliers tested at 300 MHz.

DSP-Based Multiplier

DSP blocks, or embedded multipliers, are present in all modern FPGAs; therefore it's essential to investigate if the proposed methodology can be applied to them and how. The framework is transparent to the type of multipliers used. It is important though, that the arithmetic units can operate faster than the critical-path in the redundant circuitry.

Forasmuch as the new architecture for the ROM-XOR RPR has been proposed, there's a limitation in using the output registers in the DSP block, as the new architecture firstly makes a decision in which value to use and then registers it. This represents an increase in the delay of the critical path, through the DSP block, from 2.72 ns (i.e. using its internal registers) to 4.4 ns (i.e. using external registers). Nevertheless, one must not forget that a condition that enables the adoption of the RPR within one clock cycle is the fact that the delay of the critical-path in the

Arithmetic Unit	Synth. Tool [MHz]
NO RPR 16-bit Adder	411
ROM-XOR RPR 16-bit Adder	248
ROM-XOR RPR 8-bit Mult	149

Table 4.17.: Maximum clock frequency for different arithmetic unit given by the synthesis tools.

arithmetic unit must be greater than the critical-path of the redundant circuit. It just so happens that the delay of the critical path of the redundant circuitry requires at least 4 ns, thus enabling the adoption of the ROM-XOR RPR if the limitation, of not using the output registers, is to be permitted.

4.4.4. Multiplier-Accumulator

The synthesis of a Multiply-Accumulate (MAC) unit reveals that the delay of the critical-path of the 16-bit adder is smaller, by less than half the delay of the critical-path of the 8x8-bit LUT-based RPR multiplier, as summarised in table 4.17. This gap in maximum clock frequency between the two RPR units, suggests that the version of the MAC unit using discrete adder and multiplier can be achieved with redundancy added only to the multiplier. This imposes that the maximum increase in frequency this MAC implementation supports can't exceed the maximum clock frequency of the adder for error-free operation. Consequently, the previous RPR multiplier can be reutilised to implement the MAC unit with RPR.

4.4.5. Linear Projection Designs

Linear projection is an algorithm widely used in many DSP applications with near real-time requirements, and often implemented in portable systems to compress data or extract features from sensors, e.g. image compression, SAR. For this reason, the new RPR is implemented and compared with other implementations. The application does image compression from a Z^{2000} to Z^{40} space. It uses images of faces from

the CODID CVAP Object Detection Image Database [91]. The results below present a comparison between the ROM-XOR 8:16/5:3/5:3 RPR, the LUT-SUB RPR with an approximation computed from the 5 MSBit and a threshold equivalent to the 2 MSBit, and the implementation of the linear projection without any redundancy. These configurations were arbitrarily chosen having in mind the computation of the approximations in similar conditions.

The core of the linear projection is the MAC operation, which has been introduced previously. Figure 4.26 shows the architecture of the circuit to compute the dot-product of a projection vector, which is a folded MAC. The circuit in the figure is replicated for all projected dimensions considered. From all possible implementations possible, this was chosen because it is the one which allows to reduce the area footprint and support larger problems sizes without area constraints. In the circuit implementation, all inputs are encoded using 9-bit sign-magnitude representation. The output of the multiplier is 16 bits unsigned. The word length of the output depends on the number of accumulation stages (multiplier output word length plus $\log_2(num\_stages)$).

Figure shows the results for the 3 implementations at 270 MHz. The top row shows the expected result, without variation errors. The following row correspond to the results for: NO RPR, LUT-SUB RPR and ROM-XOR RPR. On top of each image there's the PSNR from the reconstruction of the projected data, on the FPGA, into the original space in software. It's evident that the ROM-XOR RPR creates the linear projections which better recreate the original images and produce the least reconstruction PSNR for all images.

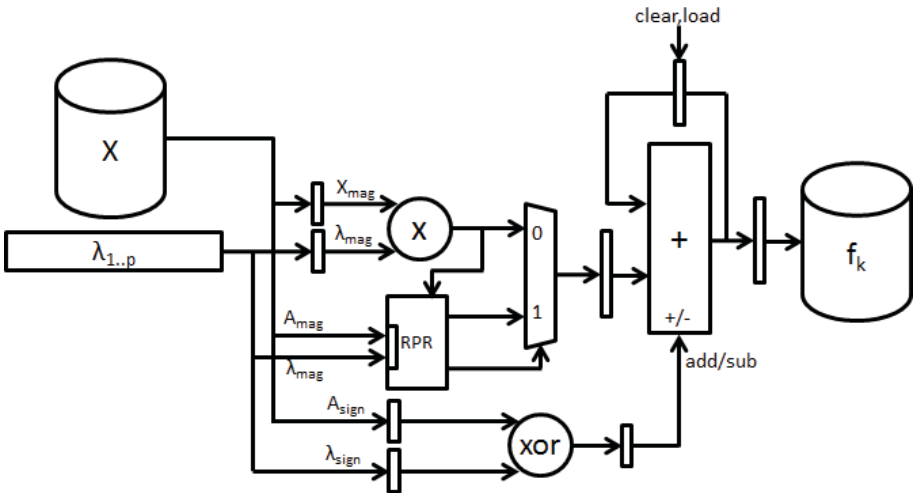

Figure 4.26.: Architecture of a circuit to implement the dot-product operator of a projection vector.

4.5. Summary

This contribution proposes a framework for a novel RPR architecture which is able to perform in systems with high-throughput and low-latency. Results show that units smaller word lengths don't benefit from it, being the method more suitable for longer word lengths as the cost in extra resources becomes relatively small. Notwithstanding, the benefits in recovering from errors overshadow the implementation cost, since there's no comparable methodology available. From these results, from the multiplier performance perspective, it's conclusive that a little penalty in the maximum performance for the error-free regime is later retributed when operating in the error-prone regime. The main drawback is the usage of extra LEs and 2 BRAMs besides the standard multiplier implementation.

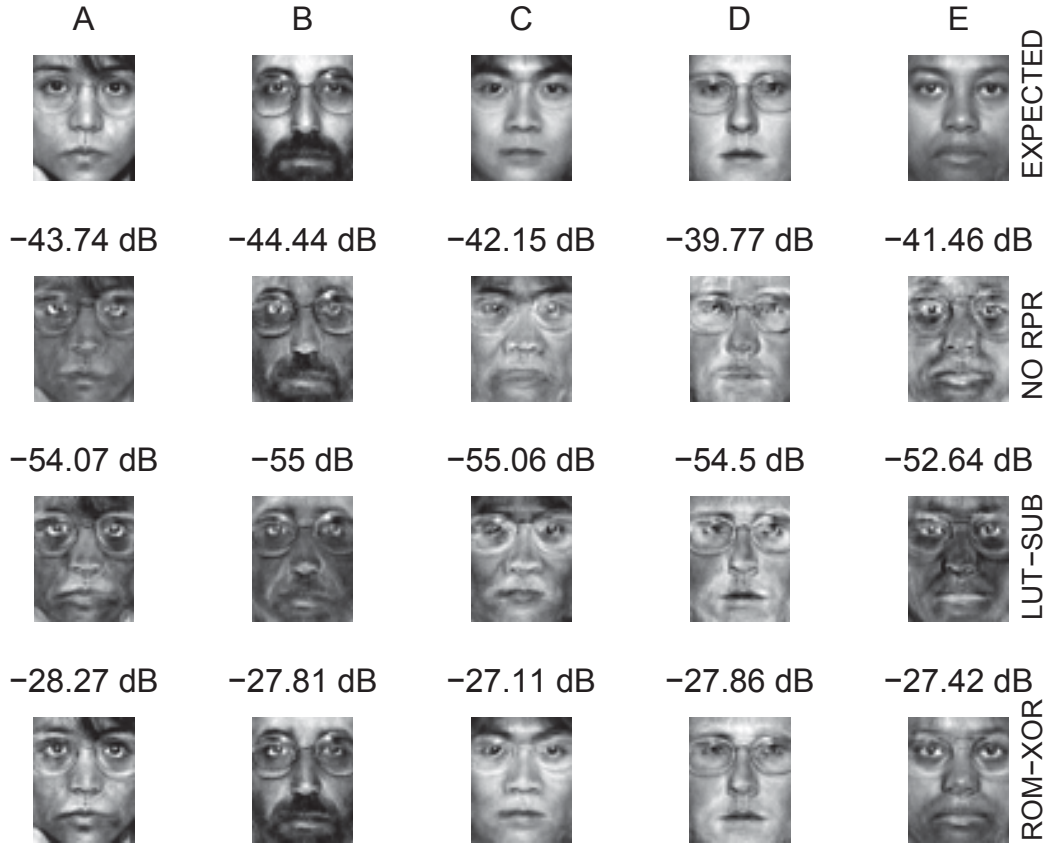

Figure 4.27.: Reconstructed faces (A-E) in the original space without variation errors (expected), and computed from the projections collected from implementations of different multipliers architectures (NO RPR, LUT-SUB RPR, ROM-XOR RPR) at 270 MHz.

5

Optimisation Framework for Acceleration of Linear Projection Designs

5.1. Introduction

In digital systems, when they infringe the timing constraints of their most critical-paths, they incur timing errors. To avoid, or reduce, these errors many methods and techniques have been proposed [5, 11, 67]. Unfortunately, most of the resilience schemes proposed penalise the performance and increase resources of the design.

Depending on the application considered, they may not even show any real benefit when compared to not applying any error mitigation methodology, as demonstrated by the results obtained from the characterisation framework earlier introduced in this thesis. The proposed Optimisation Framework (OF) encompasses the aforementioned methodology as it produces high-level specifications of linear projection designs without introducing supplementary RTL for the purpose of reducing errors and increasing resilience.

Variation errors depend on many sources, some of them uncontrollable, namely temperature, hence depicting the probabilistic behaviour of the circuit. The proposed OF optimises and mitigates this probabilistic behaviour, by trying to expose the impact of variability of the fabric into high-level algorithmic specifications. The work presented here aims to address this variation, establishing a per device optimisation concept, allowing the design to exploit extra performance capabilities in the arithmetic operators on a specific FPGA device. The aforesaid gains can't be attained by existing design methodologies.

The key enabler to the success of the OF is the device characterisation step, performed by the Characterisation Framework (CF), described in chapter 3 of this thesis, resulting in the collection of information that is utilised in the high-level specification of the design in the targeted system, leading to a final performance closer to what is achievable by the targeted FPGA device rather than what is reported by the conservative models of the synthesis tools.

Figure 5.1 depicts the design flow of the proposed OF. The first step is to characterise the arithmetic units and create the models for errors when these units are operated under certain conditions (i.e. clock frequency, core voltage, temperature and location on the device). The obtained performance information (i.e. errors that were observed at the output of the arithmetic units) as well as information regarding their area are injected into a Bayesian formulation of the problem in order to obtain

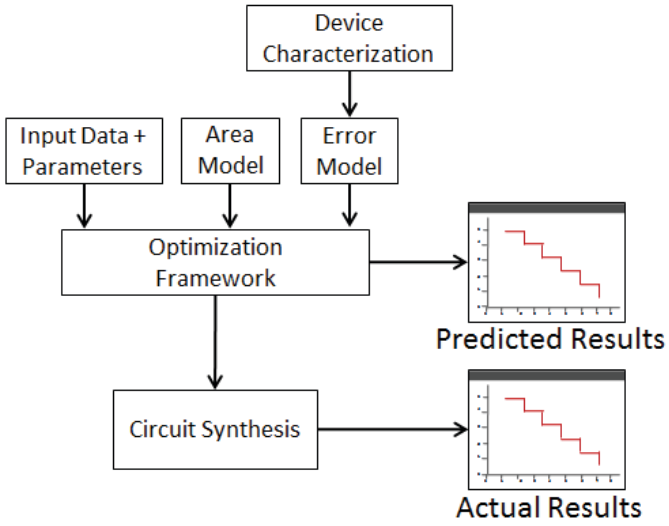

Figure 5.1.: Design flow using the proposed optimisation framework.

a high-level specification of the design to implement. This formulation is capable of producing designs that avoid, or minimise, variation errors hence outperforming standard implementations in terms of tradeoff between performance and errors. In the final step, the designs are synthesised and implemented on an FPGA, to evaluate the actual results.

Moreover, motivated by the fact that many DSP algorithms, including the linear projection, aren't critical to errors in many parts of their designs, the work presented here allows to operate arithmetic operators beyond their error-free limits, and considers also the region of operation where errors appear in the datapath. This allows the exploitation of the tradeoff between performance and error-prone calculations, thus pushing even further the achieved performance of the system.

The proposed OF for linear projection is explained in detail in this chapter. Earlier prototypes of the OF have been published in [16, 17, 19]. The linear projection algorithm has been introduced in section 2.2.3 of this thesis, and is here recapitulated. The projected data points are related to the original data through the formula in

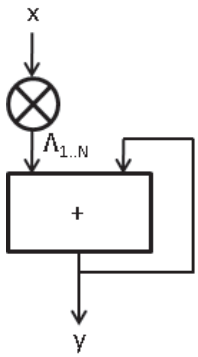

Figure 5.2.: Block diagram of the circuit to do the rolled implementation of dot-product between two vectors.

(5.1), written in matrix notation, where $X = [x^1, x^2, ..., x^N]$ and $F = [f^1, f^2, ..., f^N]$, where $f^i \in R^K$ denote the factor coefficients, E the approximation error and Λ the orthogonal basis for the new space with dimensions $P \times K$.

$$F = \Lambda^T X + E \tag{5.1}$$

A rolled architecture of the dot-product algorithm is used to implement the linear projection in hardware, as presented in figure 5.2.

In the process of optimisation of linear projections, high-level design decisions happen in the coefficients of the projection matrix, and for most of the possible implementations of the algorithm, multipliers hold the most critical paths of the datapath. Therefore, in this case the OF is devoted to the optimisation of multiplications implemented in LUT-based and DSP block multipliers.

Furthermore, the objective is to select the word length of these multipliers used for the implementation of each dot-product in (2.2) along with values of the coefficients of the Λ matrix that define the lower dimension space. The key characteristic of this work is that the obtained circuit will be over-clocked to frequencies beyond those reported by the synthesis tool in order to push further its overall performance.

Expected good results for the obtained circuits is an achieved PSNR, of the reconstruction, greater than the one for the reference implementation, under the same operating conditions. On the other hand, designs that present a PSNR smaller than the reference implementation are considered bad results. The proposed OF supports different problem size and can be adapted to other algorithms and extended to support other arithmetic operators.

5.2. Bayesian Formulation

The Bayesian formulation considers the subspace estimation and hardware implementation simultaneously, allowing the OF to efficiently explore the possibilities of custom design offered by FPGAs, generating DSP designs targeting maximisation of performance and minimisation of errors. It borrows the idea from a Bayesian Factor Analysis Model for optimisation of resource allocation in heterogeneous devices which is introduced in section 2.8 of this thesis and is described in detail in [83, 84, 6]. The proposed OF not only optimises the resource utilisation: it is also extended to account for the stochastic behaviour of multipliers under variation.

For the linear projection application, the OF samples the values for the Λ matrix factors and their word length, and estimates the basis matrix Λ, the noise covariance Ψ, and the F factors using Gibbs sampling algorithm [85] from the posterior distribution of the variables.

$$\hat{X} = \Lambda F \tag{5.2}$$

Since neither Λ and F are known, solving (5.2), to minimise the reconstruction MSE in the original space, given by

$$\text{MSE} = \frac{\sum\sum(\Lambda F - X)^2}{PN} \tag{5.3}$$

is an ill-conditioning problem, without trivial or direct solution. The main advantage of the OF is the ability to obtain F, Λ and Ψ according to their prior and posterior probabilities, without having to adopt an heuristic method to explore the solution space. P and N are the size of the projection space and the number of cases, respectively.

The OF treats the Λ matrix as a random matrix and generates a probability density function for it, which is used to sample its values from. The OF allows to insert prior information in the sampling of the Λ matrix, changing the posterior distribution to accommodate the impact of uncertainty on it. The prior distribution of the Λ matrix, $p(\Lambda, f, l, v, T)$, is the product of the probabilities of the individual elements at a clock frequency f, location on the FPGA l, core voltage v, and the temperature of the device T,

$$p(\Lambda, f, l, v, T) = \prod_{q=1}^{P} \prod_{k=1}^{K} \left[p(\lambda_{pk}, f, l, v, T) \right]. \tag{5.4}$$

This prior distribution expresses the uncertainty of each Λ matrix and has to meet the following properties:

$$p(\lambda_{pk}, f, l, v, T) \geq 0, \forall \lambda_{pk}, \tag{5.5}$$

$$\sum_{\lambda_{pk}} p(\lambda_{pk}, f, l, v, T) = 1. \tag{5.6}$$

Any given coefficient λ_{pk} has a probability of being sampled greater or equal than zero, and the cumulative probability of all possible λ_{pk} values is 1.

5.3. Objective Function

Even though the OF samples projections according to a prior distribution it creates candidate designs that are suboptimal. Given that, it orders the generated designs

in terms of resources and errors and then extracts the designs which present the optimal points on a Pareto curve.

The purpose of the objective function is to map both reconstruction errors and variation errors into a single quantity to be minimised by the OF. Let's denote with $\hat{X}$ the result of the reconstruction of the projected data in matrix format. Then, the objective function U is defined as in (5.7), where both reconstruction errors and variation errors are captured. The matrix formulation is defined as:

$$U = \mathrm{Tr}\left(\mathrm{E}\left[\left(X-\hat{X}\right)^{T}\left(X-\hat{X}\right)\right]\right) \tag{5.7}$$

E denotes the expectation and Tr the trace operator. It expresses the reconstructed data as a function of the Λ matrix, and the variation error with ε such as $\hat{X} = \Lambda(F+\varepsilon)$. By imposing ε to have zero mean, which is achieved by subtracting a constant in the circuit, and using the fact that the Λ matrix is orthogonal and orthonormal, the objective function is expressed as:

$$U = \mathrm{Tr}\left(\mathrm{E}\left[(X-\Lambda F)^{T}(X-\Lambda F)\right]\right) + \mathrm{Tr}\left(\mathrm{E}\left[\varepsilon^{T}\varepsilon\right]\right) \tag{5.8}$$

$$= \mathrm{Tr}\left(\mathrm{E}\left[(X-\Lambda F)^{T}(X-\Lambda F)\right]\right) + \sum_{j}\mathrm{var}(\varepsilon_j) \tag{5.9}$$

Here j denotes the columns of the Λ matrix. By assuming that the errors at the output of the multipliers are uncorrelated, then the first term in the final expression relates to the approximation of the original data from the linear projection, without any variation errors, where the second term relates to the error variance from the characterisation of the arithmetic units under PVT variation. Thus, the errors due to dimensionality reduction and variation are captured by one objective function without any need to formulate a problem using a multi-objective function.

5.4. Sampling From a Posterior Distribution

The proposed OF utilises information, from the CF, regarding the performance of the multipliers for a given device and their respective resource utilisation, by suitably constructing a prior distribution function for the coefficients of the Λ matrix.

5.4.1. Prior Distribution

As previously stated, the OF samples values for the coefficients of the Λ matrix. Therefore, the prior distribution is used to model the distribution of the values acquired from the sampling process by penalising Λ matrix instances with large area and variation errors by assigning low probabilities to them, favouring matrices without variation errors and consuming the least area. The prior distribution is expressed as a function of the error variance at the output of multipliers on the FPGA when under variation and is given by equation 5.10.

$$p\left(\lambda_{pk}, f, l, v, T\right) = g\left(A\left(\lambda_{pk}\right), E\left(\lambda_{pk}, f, l, v, T\right)\right) \tag{5.10}$$

$$g\left(A\left(\lambda_{pk}\right), E\left(\lambda_{pk}, f, l, v, T\right)\right) = c\left[A\left(\lambda_{pk}\right)^{-1}\left(E\left(\lambda_{pk}, f, l, v, T\right)+1\right)^{-1}\right] \tag{5.11}$$

c and is a constant used to ensure that

$$\sum_{\lambda_{pk}, f, l, v, T} g(A(\lambda_{pk}), E(\lambda_{pk}, f, l, v, T)) = 1. \tag{5.12}$$

$A(\lambda_{pk})$ is the area to implement the multiplier and $E(\lambda_{pk}, f, l, v, T)$ on the error variance observed in the multiplier characterisation for a given λ_{pk} value tested at f clock frequency, location l, core voltage v and temperature T.

5.4.2. Error Model

The performance characterisation, to create the error model, of the arithmetic operators is performed on the ones that exhibit paths with the longest delay in the datapath of the DSP design considered. In case of the linear projection, that corresponds to the multipliers.

The error model uses the information about the uncertainty of multipliers, with specific word lengths, at specific clock frequencies, placement coordinates, core voltages and temperatures.

The method used to collect error information relied on the CF which stimulates the test circuit with a specific test vector and then compares the output of the arithmetic units against the expected result. This was repeated for all operating conditions (clock frequency f, location l, voltage v and temperature T) on the target device. This error model uses the variance of the variation errors observed, as it is a statistical metric that measures how distant the outputs of the circuit are from the expected values. In addition, a metric such as the mean isn't suitable as variation errors can cancel out, thus producing a mean close to zero, even though there are errors in all observations.

$$\mathrm{E}(\lambda_{pk}, f, l, v, T) = var(\lambda_{pk}, f, l, v, T) \tag{5.13}$$

At present only CCM, generic and DSP-based multipliers are characterised as usually they are the arithmetic units, in DSP circuits, where the critical-paths reside. Nonetheless, this work can be extended to support other arithmetic operators required by other applications being optimised.

Figure 5.3 illustrates the error variance for all multiplicands of an 8-bit LUT-based generic multiplier at different clock frequencies stimulated with a pseudo-random test vector. The x axis represents the constant multiplicands and the y axis the

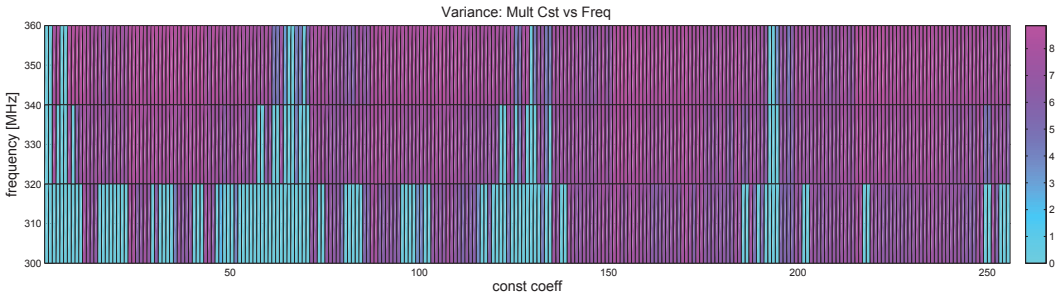

Figure 5.3.: Error variance of an unsigned 8-bit LUT-based generic multiplier.

clock frequencies. For each coordinate the colour of the cell represents the error variance observed during the test. Examining the results shows that although a multiplier produces errors for some constant multiplicands, others don't produce errors. Moreover, the presence of errors is cumulative with the frequency increase, meaning that multiplicands with errors at lower over-clocking frequencies also have errors at higher frequencies.

5.4.3. Area Model

The OF uses an area model to estimate the area of the generated designs without having to synthesise them. The design area comprises the area of the dot-product (multipliers and adders) employed in each design. Since the multipliers have been synthesised for the characterisation step, their actual value is derived from the synthesis reports.

In the case of using CCMs, the area depends on the value of the multiplicands considered, whereas the area of a generic multiplier depends on the word length of the multiplicands, regardless their value. In case of using DSP block multipliers the area of the design is constant regardless the word length of λ_{pk}. For all multipliers only the word length of λ_{pk} changes, and the word length of the input data remains constant.

The linear projection circuit, which corresponds to equation 2.2, is often imple-

mented using CCMs, followed by an adder tree, or by MAC units using generic multipliers. Therefore, the area of the adders is also accounted in the circuit area model.

Hyper-Parameters

To change the influence of area and errors in the prior distribution, *Hyper-Parameters* were introduced. These *Hyper-Parameters* have impact on the performance of the designs generated given that λ_{pk} values requiring more resources and prone to error may be excluded by assigning them a very low prior probability. Equation 5.11 is now modified to hold the *Hyper-Parameters* α and β:

$$g\left(A\left(\lambda_{pk}\right), E\left(\lambda_{pk}, f, p, v, T\right)\right) = c\left[A\left(\lambda_{pk}\right)^{-\alpha}\left(E\left(\lambda_{pk}, f, p, v, T\right)+1\right)^{-\beta}\right] \quad (5.14)$$

The current implementation of the OF uses empirical values for the *Hyper-Parameters* and doesn't perform any informed selection on them.

In figure 5.4 there's an example of prior distributions for $\alpha = 0$ and 3 different values of β using the same multiplier characterisation. It illustrates the impact of the *Hyper-Parameter* in the prior distribution, giving more or less emphasis on the errors due to variation. The figure shows that for $\beta = 0.1$ all λ_{pk} values have almost the same probability of being sampled, whereas for $\beta = 4.0$, λ_{pk} values with high variation errors have a very low probability of being selected.

5.5. Design Optimisation

The OF generates a number of designs that minimise the selected objective function U for a given FPGA area due to its sampling process. The resulting designs are the ones that fall on a Pareto curve of area *vs* MSE of the reconstruction of the projected data in the original space, for a given clock frequency, FPGA location,

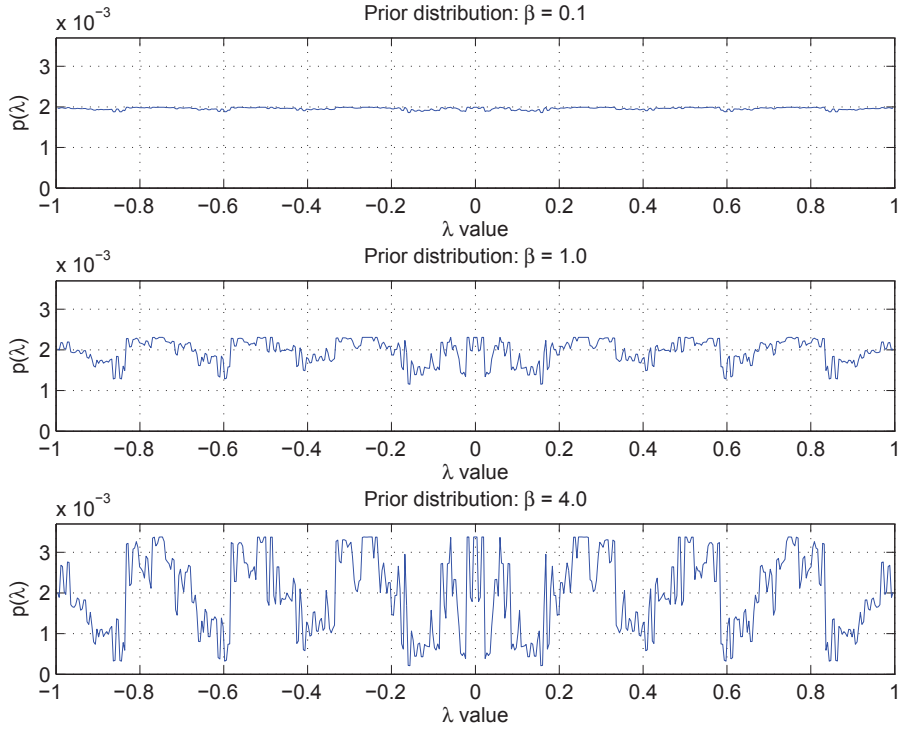

Figure 5.4.: Prior distribution for $\alpha = 0$ and $\beta = [0.1, 1.0, 4.0]$ for an 8-bit unsigned multiplier at 340 MHz.

core voltage and temperature. The OF estimates each dimension (i.e. column) of the Λ matrix in a sequential manner. The user supplies the targeted dimensions K, the targeted operating conditions (clock frequency f, location coordinates l, core voltage v and temperature T), α and β *Hyper-Parameters*.

The OF iteratively samples candidate projections using different word lengths, for each projection vector. This means that accounting for all possible combinations, the OF would have to generate wl^K designs, being wl the number of word lengths considered and K the number of projection vectors. As a result of the time required to account for all possible combinations, an internal optimisation parameter has been added to the OF. It is the number of designs to be passed from one iteration, or projection vector, and is known as Q. At the end of each iteration, the candidate designs are sorted by their expected MSE, given by the objective function, and placed in Q bins. The designs which exhibit the least MSE are extracted and passed to the next iteration or used to implement the linear projection design.

Algorithm 2 shows the pseudo-code to select Q designs. This algorithm doesn't guarantee the best Q designs possible out of the OF, but it is effective in reducing the number of candidates through an automated selection. To obtain Q designs the OF processes the first projection for the input data using different word lengths. The reconstruction MSE for the projections is calculated and organised in Q equally spaced bins, which are bounded by the minimum and maximum values prevenient from these projections. The optimal projections are the ones with minimum error for the same area. From each bin, the projection with the least MSE is selected. This results in Q projections which are used in following projected dimension. In the following iteration, for the new projection dimension, the process aforementioned is repeated considering the error from the previous projection as the data to be projected. Figure 5.5 illustrates the iterations for the generation of linear projection designs. The dashed horizontal lines represent the boundaries between bins, and the

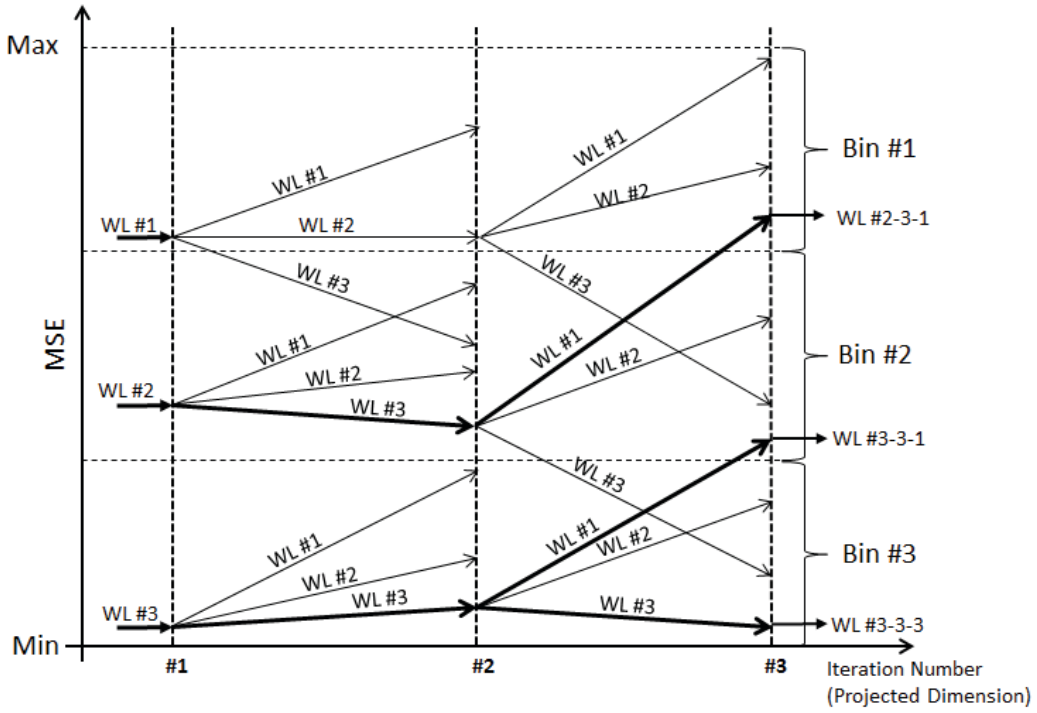

Figure 5.5.: Illustration of the generation of linear projection designs with 3 projected dimensions and $Q = 3$.

dashed vertical lines the iterations for each projected dimension.

5.6. DSP Block Support

When DSP blocks are used, which are embedded in the FPGA's architecture, they occupy an entire block regardless of the word length and the values of the projection coefficients. Thus, they are assigned constant resources. This is similar to assign *Hyper-Parameter* α the value zero in the prior distribution, as no change in resources to implement the multiplier is observed. In this case the prior distribution accounts only for the contribution of variation errors.

To sum up, the proposed OF doesn't make a distinction between different multiplier architectures. If different types of multipliers are to be considered in the

Algorithm 2: Algorithm to sample Q linear projection designs from a prior distribution.

Require: $Q \geq 1 \wedge K \geq 1 \wedge \beta > 0 \wedge f, v, T > 0$
Ensure: Q linear projection designs
  $X \leftarrow input$ {original data N cases}
  for $d = 1$ **to** K **do**
    Create new empty $Candidate\_Projs$ list
    for $wl = wl_{min}$ **to** wl_{max} **do**
      $prior \leftarrow generate\_prior(wl, \beta, f, p, v, T)$
      $\lambda_{d,wl} \leftarrow sample\_projection(X, prior, wl)$
      $area_{\lambda_d,wl} \leftarrow estimate\_area(\lambda_d, wl)$
      $uncertainty(\lambda_d, f, p, v, T) \leftarrow characterisation\_var(\lambda_d, f, p, v, T)$
      $F \leftarrow (\lambda_d^T \lambda_d)^{-1} \lambda_d^T X$
      $error \leftarrow X - \sum_{j=1}^{d} \lambda_{j,wl} F$
      $MSE_{d,wl} \leftarrow \left(\sum \sum error^2 / PN\right) + uncertainty(\lambda_d, f, p, v, T)$
      $Proj \leftarrow (\lambda_{d,wl}, area_{d,wl}, MSE_{d,wl})$
      Add $Proj$ to $Candidate\_Projs$ list
    end for
    $CurArea = 0$, $CurMSE = Max(MSE(Candidate\_Projs))$
    while $Candidate\_Projs$ list not empty **do**
      Sort $Candidate\_Projs$ by Area
      Extract $Candidate\_Projs$ at head of the list
      if $Area(Candidate\_Projs) > CurArea$ **then**
        if $MSE(Candidate\_Projs) > CurMSE$ **then**
          Add $Candidate\_Projs$ to $Pareto_Points$ list
          $CurArea = Area(Candidate\_Projs)$
          $CurMSE = MSE(Candidate\_Projs)$
        end if
      end if
    end while
    Create Q bins $= [MSE_{min} : (MSE_{max} - MSE_{min})/Q : MSE_{max}]$
    for $q = 1$ **to** Q **do**
      Extract the 1st $Pareto\_Point$:
      $MSE <= MSE_{min} + (q-1)(MSE_{max} - MSE_{min})/Q$
    end for
  end for
  Create Q designs from the extracted Q projections
  return Q designs

optimisation of the design then the OF offers the flexibility to support different implementation requirements at the same time, stretching the methodology proposed in [83] to consider errors due to variation.

Since the routing inside DSP-Blocks doesn't change, the errors due to variation across different units on the same device, or on different devices, will reflect the impact of process variation and jitter, thus eliminating variation due to placement and routing inside the arithmetic unit.

5.7. Optimisation Strategies for Linear Projections

Linear projection designs are frequently implemented with tight requirements in terms of throughput, maximum circuit area, and minimum acceptable reconstructed quality, which can lead to the adoption of different design strategies. To account for them, the OF is able produce design using two different strategies to originate designs that:

- achieve a reconstruction error below a maximum value tolerated error in their reconstruction,
- produce the least reconstruction error for a specific number of projected dimensions.

So far the methodology described imposes the number of projected dimensions, or the amount of data generated at the output of the linear projection system. To account for the alternative strategy minor changes were introduced in the OF and are discussed below.

5.7.1. Linear Projection Targeting a Maximum Reconstruction MSE

When an linear projection has to be implemented with specific quality requirements, and resources are not a constraint, the OF creates a linear projection design that meets that. Even though it may suggest that a linear projection can scale uncontrollably, is has been showed that the Bayesian Framework, in some cases, is able to create optimal linear projection designs that achieve the target reconstruction MSE with less projections than with the application of the reference methodology (PCA) [6].

In this scenario, the design generation algorithm used will add projection vectors to the design until the required reconstruction MSE is achieved. In more details, this design generation algorithm, for all possible word lengths, samples projection vectors and adds the one with the least reconstruction MSE to the design. By adding projection dimensions, the algorithm iteratively adds more information to the projection, thus reducing the reconstruction error. It stops when the generated design achieves a reconstruction MSE lower than the target MSE.

To support this functionality, the design optimisation algorithm differs in the *loop* which iterates through all the K projection dimensions (d) to become a *loop* which returns whenever the maximum reconstruction MSE is met.

In case the target reconstruction MSE is set below the optimal achievable by the OF, or if the errors due to variation are too high, the number of projection vectors required is likely to increase.

5.8. Optimisation Framework Evaluation

A case study is conducted to evaluate the performance of the proposed optimisation framework, targeting different strategies for linear projection designs with high-

performance and low-latency. Since there are applications that are constrained by their sources to process new data at every clock cycle, and deep pipelining isn't possible, it is of great importance to examine how the proposed OF copes with those requirements.

The case study presented concerns of a small problem to demonstrate the applicability of the proposed methodology. The incentive to present a small problem relies on the fact that the regularity of the architecture to implement a linear projection design makes its performance independent of the dimensions of the problem. Moreover, given the novelty of the methodology proposed, it is intentional to keep the problem as simple as possible to assess the impact of the proposed methodology.

To evaluate the performance of the proposed optimisation framework in the generation of circuit designs for dimensionality reduction problems based on linear projection, a reference design is implemented that relies on the KLT transformation, considering different word lengths, modeling the existing approach to the above problem. The generated designs are evaluated in terms of reconstruction MSE/PSNR and required hardware resources for different operating conditions (clock frequency, core voltage, temperature and location on the FPGA).

The performance evaluation of the proposed work happens in three domains, namely the *predicted*, *simulated* and *actual*. The *predicted* performance is the performance expected by the designs generated by the proposed optimisation framework using the described error model. The *actual* is the performance observed on the optimised design when it is implemented on an FPGA device.

The *simulated* performance is the performance of a design using the same data sets as the ones that are utilised for the *actual* performance evaluation of the design in the device, but employing the information from the characterisation. This intermediate result is not fundamental, but it provides an insight of the quality of the error model due to operation under variation. Deviations between this performance

characterisation and the *actual* performance of the designs on the device still exist due to placement and routing variation and jitter.

It is worth mentioning that aiming to abide with the impact of placement and routing variability, designs generated by the proposed optimisation framework were synthesised using different pseudo-random seeds in the synthesis tool. Moreover, to abide with the impact of variability due to jitter, synthesised circuits were tested many times. All designs were implemented on Cyclone III EP3C16 FPGAs from Altera [1], attached to DE0 boards from Terasic [89].

5.8.1. Characterisation and Training Samples

To test the application of the OF in linear projection various sets of samples were generated to characterise the arithmetic units, in order to create the candidate designs and test the designs on an FPGA, but all of them were obeying the same model.

Test Samples

This set contains examples of synthetic data being passed to the design in the interest of producing linear projection results. Usually the number of samples used is determined by the application. Since there are no constraints imposed by the example chosen, 29.4k samples were used to test the designs since it is approximately half of the total number of combinations for input data. The data was generated using the Matlab source code `generate_data.m` in the Appendix A.3. Figure 5.6 shows the distribution of the samples generated for a linear projection application.

Characterisation Samples

The characterisation process tests arithmetic units on a device under variation and produces statistics, such as variance of error, which is incorporated in the error

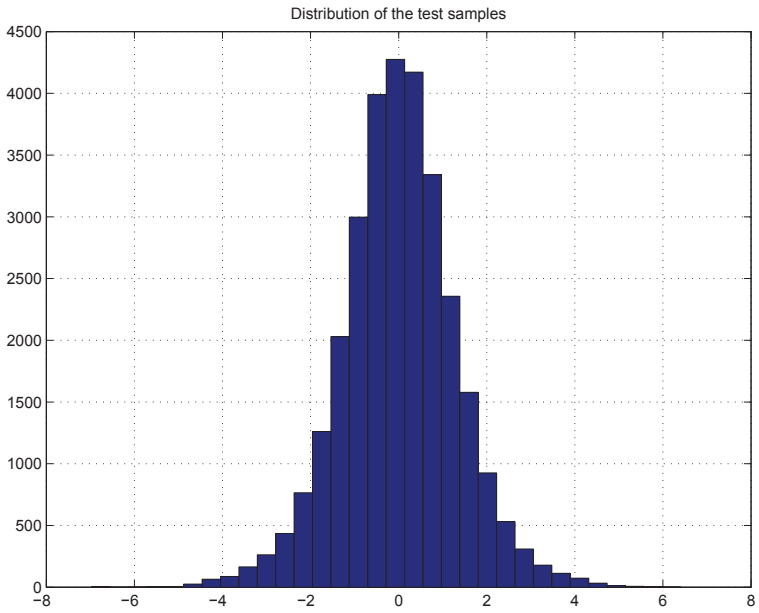

Figure 5.6.: Histogram showing the distribution of the test samples generated for the linear projection test case according to their value.

model used by the OF during the optimisation process. Moreover, the characterisation process needs to reflect the impact of the problem data, while using a smaller data set to reduce the characterisation time. Hence, it is important to create a characterisation data set with the similar variance and probability distribution using the minimum number of samples in order to achieve this.

The determination of the minimum number of samples is achieved through resampling methods, namely Jackknife [92, 93] and Bootstrap [94]. These methods estimate the variance of data subsets, which are used to compare contra the variance of the complete data set.

In effect, the complete data set corresponds to 29.4k samples, with the same mean and variance as the test data set, to be used in the characterisation.

The characterisation evaluates this data set assuming one of the multiplicands as a constant coefficient on a 8-bit unsigned generic multiplier under variation. The resampling methods were applied to all constant coefficients individually. From

R	S	Num Samples	% Coefs. in CI
5	1425	7125	100
5	1900	9500	100
5	2280	11400	100
5	3675	18375	100
5	4900	24500	100
5	5880	29400	100
6	1425	8550	97.26
6	1900	11400	97.64
6	3675	22050	97.26
6	4900	29400	97.64
8	1425	11400	81.96
8	3675	29400	81.96

Table 5.1.: Results for the Jackknife resampling method within a 95% confidence interval.

the complete data set, R subsets with different sizes S were created to be tested for validity under a 95% confidence interval. It is envisioned that the greater the number of samples, the greater is the number of coefficients exhibiting a variance within the 95% confidence interval. Moreover, greater number of subsets requires more samples to meet the variance of the characterisation data set.

Jackknife The Jackknife method was initially proposed in [92], and later extended in [93]. It determines statistical measurements, i.e. mean and variance, for subsets of a sampled set without considering one sample at a time, and then produces as a result their average. The subsets were created from a set derived from the complete data set obtained from the characterisation of the device. Each subset is extracted from the complete data set without repetition.

The results for the tests performed are summarised in table 5.1. The greater the number of samples for the subsets, the greater the number of coefficients having their variance within the confidence interval. Therefore, these results are in line with the predicted results.

R	S	Num Samples	% Coefs. in CI
5	1425	7125	100
6	1425	8550	97.26
8	1425	11400	81.96
5	1900	9500	100
6	1900	11400	97.64
5	2280	11400	100

Table 5.2.: Results for the Bootstrap resampling method for different sets with sampled variances within the 95% confidence interval.

Bootstrap Similar to Jackknife, the Bootstrap differs on how the subsets are created. This method samples, with replacement, from the set and then creates a subset and produces the variance for it. This process is repeated many times (usually $>$ 1k) to estimate the shape of distribution of the variance. Table 5.2 summarises the results for different tests.

Training Samples

As aforementioned, the OF samples many possible designs in order to be able to extract the optimal ones. In large application this can lead to long execution times ($>$ 100h). To accelerate this process the linear projection designs are sampled using a training set, which is smaller than the data set in the problem. The requirement is that the training set has to keep the same statistical properties as the problem set in order to produce Λ matrices representative of the problem set.

The work presented in [95] examines the two arguments and show results for a known data set. This study suggests that the best results happen when large numbers of samples and high ratios, between test and training samples, are present.

In this case study, in order to define the number of samples, a linear projection matrix (Λ) was created to describe the linear projection of 5k samples, using the KLT algorithm. The MSE for different sizes of samples was then computed using random input samples. In order to be able to model the expected variation of the

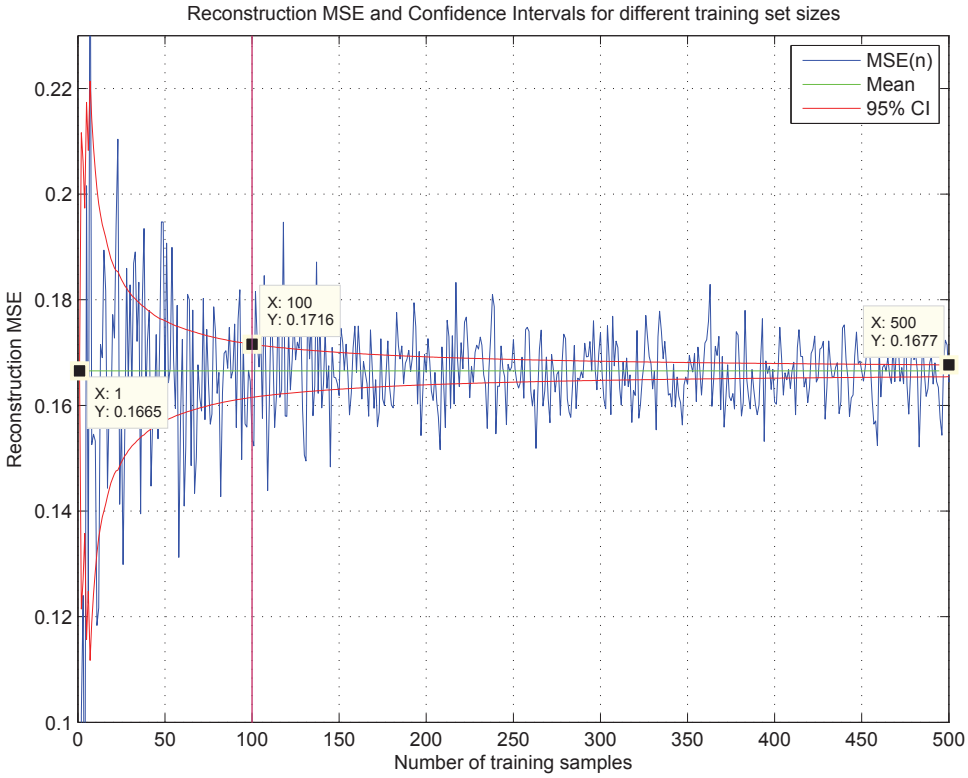

Figure 5.7.: Reconstruction MSE and Confidence Intervals for different training set sizes.

reconstruction MSE for each sample size, a 95% confidence interval was computed.

Figure 5.7 shows the reconstruction MSE and the 95% confidence interval for the different training set sizes. Also, three values of the confidence interval have been identified: average value, achievable with infinite number of samples, and the values for 100 and 500 samples. When compared the limits of the confidence interval for 100 samples, with the limits of the confidence interval for 500 samples it shows an increase of 2.32% whereas it represents an increase of 3.06% when compared to the mean value (5k samples).

Figures 5.8 and 5.9 show the distribution of 1.9k and 100 samples respectively, generated for a Z^6 to Z^3 linear projection application. Table 5.3 summarises the mean value and variance of each data set. It is observable from this data that the

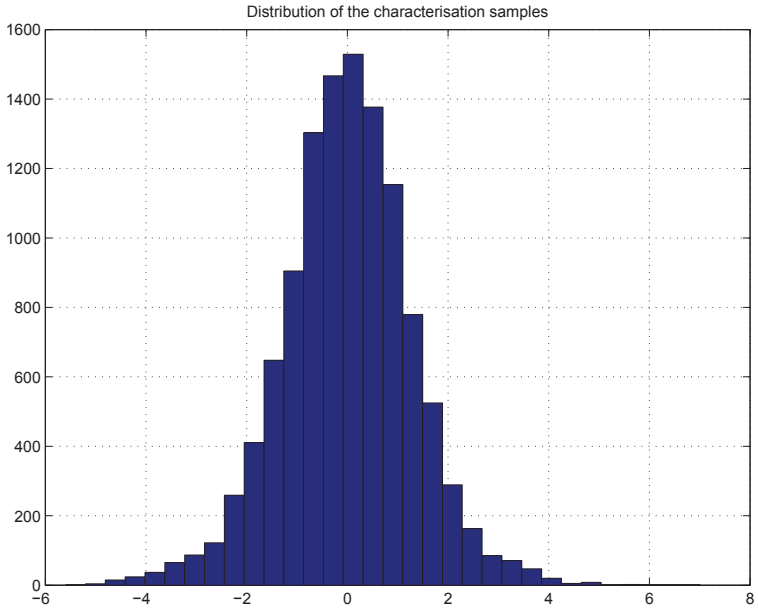

Figure 5.8.: Histogram of 1.9k characterisation samples used in the characterisation of the arithmetic units for the Z^6 to Z^3 linear projection.

Data Set	Mean	Variance
5k	2.246×10^{-3}	1.665
1.9k	-4.246×10^{-4}	1.653
100	2.708×10^{-2}	1.778

Table 5.3.: Mean and Variance of the different data sets

distribution, and its mean value and variance is close for all data sets.

5.8.2. Circuit Architecture

The linear projection application considered in the case study relies on the implementation of a dot-product. The block diagram of the dot-product circuit implementation.was presented in figure 5.2. Figure 5.10 shows the block diagram of the test circuit implementation. This circuit was designed for maximisation of throughput and allowed relaxation in terms of area requirements. The datapath of the Design Under Test (DUT) is replaced with a dot-product implementation: 2.5 or 2.4.

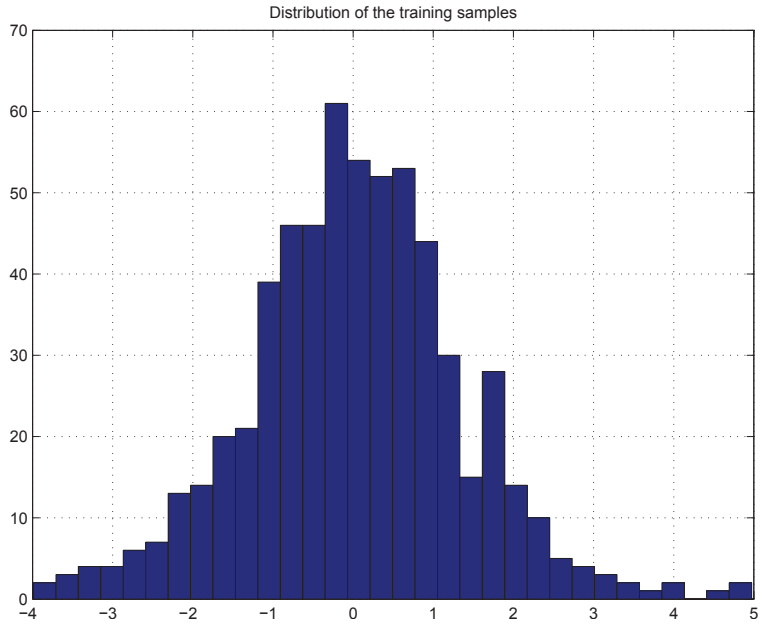

Figure 5.9.: Histogram of 100 samples used in the training of the Optimisation Framework for the Z^6 to Z^3 linear projection.

q

The circuit is composed of the following blocks:

1. Input and output memory blocks to emulate the input and output streams in the circuit;
2. datapath being tested under variation, e.g. dot-product;
3. FSM to control the datapath;
4. Clock generator for datapath and FSM.

Similarly to the circuit in the characterisation framework, presented in chapter 3 of this thesis, only the datapath is sensitive to variation as it runs at twice the clock rate of the rest of the circuit. Data is written/read to/from memories via JTAG interface. The same interface is also used to trigger the execution of the test. FSM (ext ctrl).

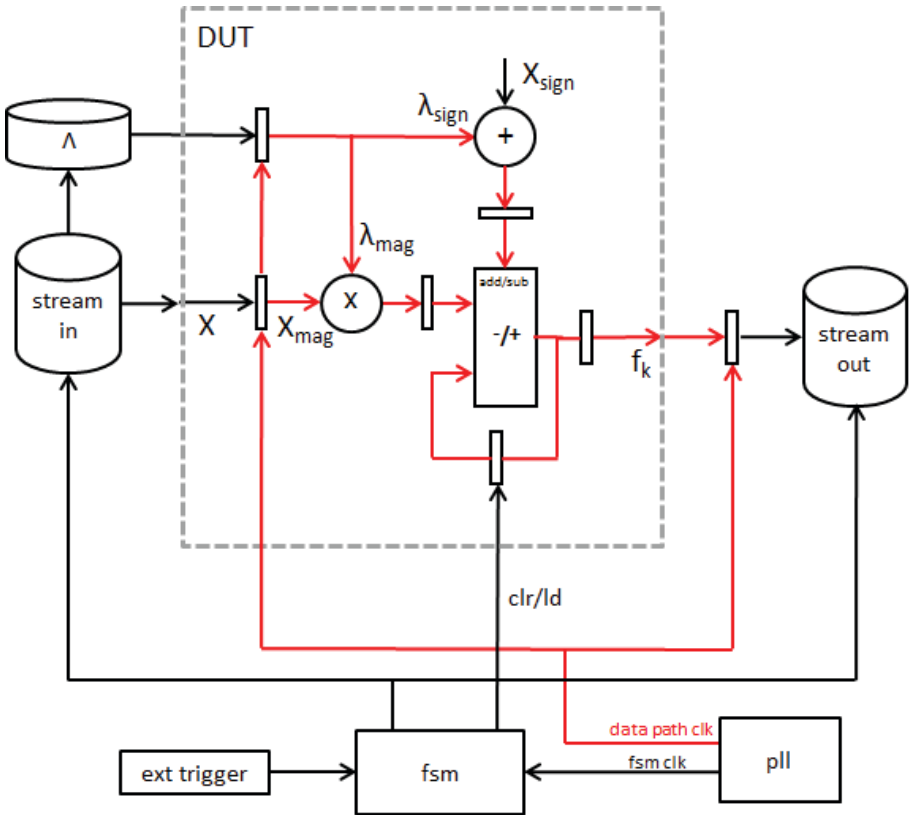

Figure 5.10.: Block diagram of the circuit to test the dot-product implementation, used in linear projection case study.

5.8.3. Area and Error Models Evaluation

Area Model

The area model is created to give a quick, but accurate, estimate on the resources taken by a candidate design without having to actually synthesise it. Area is measured in terms of number of LEs required to implement the circuit. This estimate is based on the resources reported by the synthesis tool for the multipliers in the characterisation circuit, however small deviations are expected due to further optimisations performed by the synthesis tool. The area estimate ($\hat{A}_{LinProj}$) for each dot-product follows a linear approximation of the form

$$\hat{A}_{LinProj} = r.A_{Mult}(wl, coeff) + s \tag{5.15}$$

Where A_{Mult} is the area required to implement a multiplier of a given word length, and with a specific value, r and s are the approximation parameters and they change for different types of multipliers, such as CCM and generic multipliers.

Figure 5.11 shows an example of the relation between the predicted and the actual resources occupied by LUT-based multipliers on the FPGA. The figure also shows that most of the data points fall inside the 95% confidence interval for the area estimation.

Error Model

Figure 5.12 depicts design points under *predicted*, *simulated* and *actual* performances. It should be noted that all area results refer to the actual area utilised by the design.

The obtained results show that the errors in the model (or *predicted*) is pessimistic in regards for the actual results. This is due to the fact that the characterisation of the device used a longer data sequence, which produced more timing errors, whereas

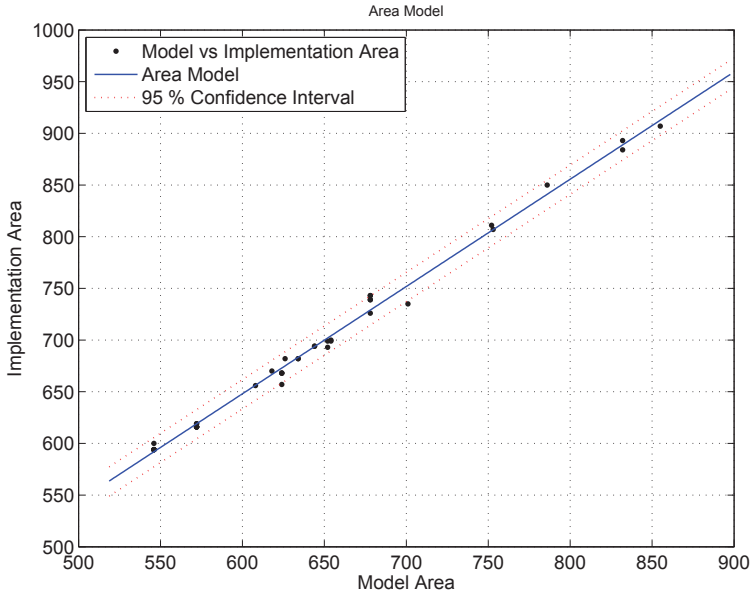

Figure 5.11.: Evaluation of the area model against the actual circuit area for linear projection designs using LUT-based multipliers.

the actual test didn't generate (so many) errors.

The *simulated* results are closer to the actual results as they result from the characterisation of the multipliers using the actual test data instead of the pseudo-random.

It would be ideal to have *predicted*, *simulated* and *actual* data points overlapping, but the nature of the problem (i.e. size of the application, distribution of the data, placement and routing variation, along with other sources of variation) doesn't allow to achieve that.

5.8.4. Optimisation of Linear Projections for Throughput, Area and Errors

This test case considers the cases where the linear projection design is implemented using LUTs. In terms of design options it covers the use of different implementations for the dot-product algorithm, using CCMs or generic multipliers. Both cases are

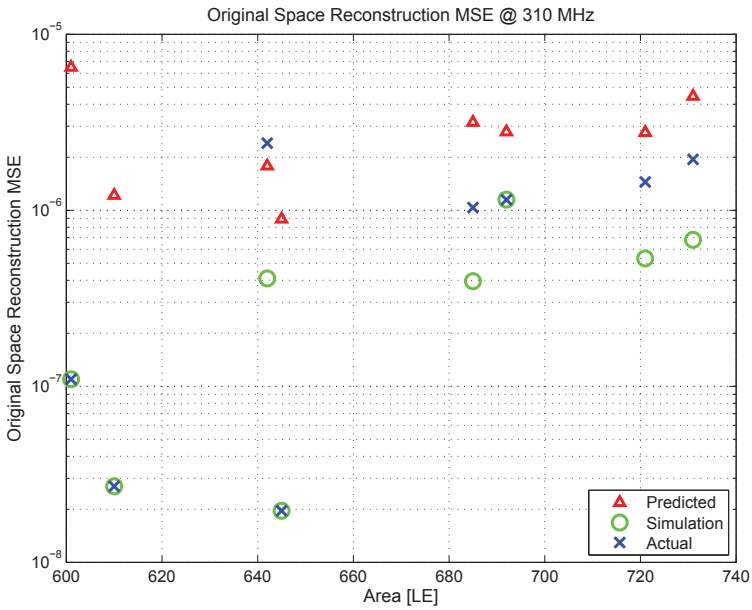

Figure 5.12.: *Predicted*, *simulated* and *actual* performance reconstruction MSE vs. area of the linear projection designs produced by the proposed optimisation framework using LUT-based multipliers. The target clock frequency is 310MHz.

sensitive to the amount of resources required, therefore the optimisation tries to maximise throughput while minimising area and errors. The clock frequencies were chosen so that arithmetic units produce some errors in order to evaluate the proposed optimisation framework.

Constant Coefficient Multipliers

CCMs enjoy minimal resource usage, and they have few critical paths becoming the multiplier choice for DSP applications that demand high- performance. The tradeoff, when compared to the generic multiplier, is that absence of generality.

In this case the performance of the proposed optimisation framework, considering area and errors, (*OFae*) is compared against the standard KLT implementation (*KLT*) and optimisation for area only (*OFa*). Figures 5.13 and 5.14 show the results for the model and the implementation, respectively. The results are presented as Pareto curves and they show the optimal sets of designs, in terms of area and reconstruction PSNR at 510 MHz, which is 2.32 times faster than the maximum specified by the synthesis tool.

The model results show the estimated hardware resources *vs* the estimated reconstruction PSNR in the original space. The optimisation framework considers the information about the CCMs operating at 510 MHz, but it assumes the arithmetic units don't have errors in their calculations. The results are good as they show that *OFae* designs are always better than the *KLT* designs. Compared to designs optimised only for area (*OFa*), the designs optimised for area and errors (*OFae*) perform better, or at least the same.

The implementation results show the actual hardware resources required *vs* the results from the board when operated under over-clocking at 510 MHz. For a PSNR above 31 dB the *KLT* design requires 39% more hardware resources than the *OFae* design. Below the limit of 2000 LEs, the *OFae* design increased the PSNR by 4.35 dB

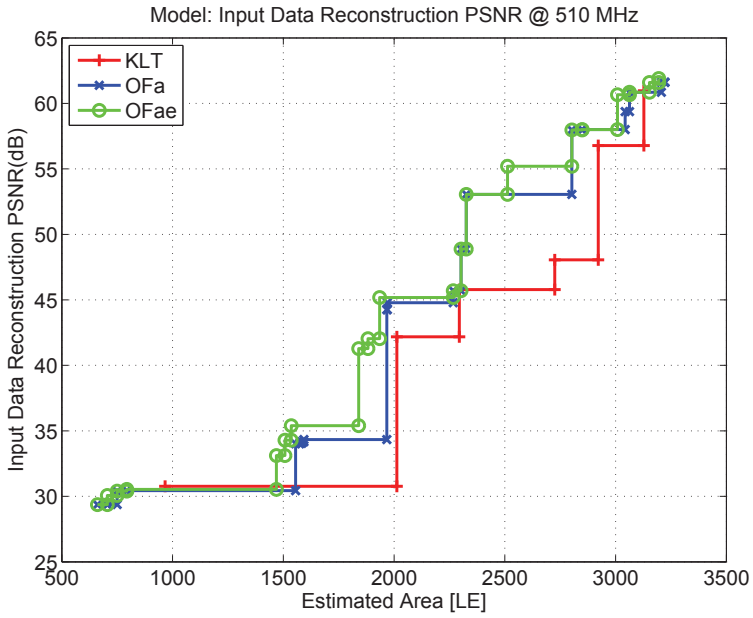

Figure 5.13.: Estimated circuit area *vs* model reconstruction PSNR at 510 MHz.

and more 8.79 dB PSNR than *OFa* and *KLT* designs, respectively. The deviations between the *predicted* and the *actual* happen because the adder tree wasn't modelled, and their errors haven't been considered in the optimisation process. Nevertheless, *OFae* provides the design with the best reconstruction PSNR in both cases.

The above study demonstrates that is possible to optimise a linear projection design for area, reconstruction data PSNR and resilience to operation under over-clocking simultaneously, by inserting information regarding the area and performance of the arithmetic units. It is also demonstrated, that for a target PSNR it is possible to reduce resource usage and increase the clock frequency by 290 MHz.

Generic Multipliers

Performance Limits The performance limitations of the generated linear projections designs following the existing methodology of applying the KLT transform and then mapping the design on an FPGA are depicted in Figure 5.15. The design space

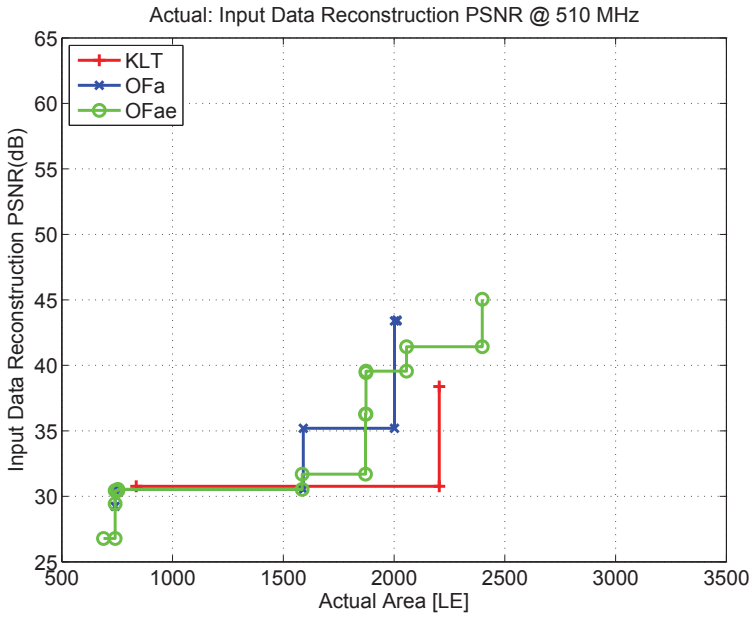

Figure 5.14.: Actual circuit area *vs* implementation reconstruction PSNR at 510 MHz.

exploration is performed along the word length employed by the generic multipliers, which corresponds to the quantisation step of the design process. The figure shows the maximum clock frequency reported by the synthesis tool (Tool Fmax - green), the datapath maximum frequency when the design is placed to the targeted FPGA without observing any errors in the calculations in the datapath (datapath Fmax - yellow), and the range of the frequencies where the design starts generating errors due to over-clocking (FSM Fmax - red).

Targeting a Minimum MSE for Given Number of Dimensions In this case study, the target clock frequency is set to 310 MHz, which is 1.85 times higher than the possible frequency reported by the synthesis tool for a KLT design employing 9 bits coefficient word length. At this clock frequency some KLT-based designs (i.e. the ones with large area footprint) will operate with errors in their datapath (Figure 5.15).

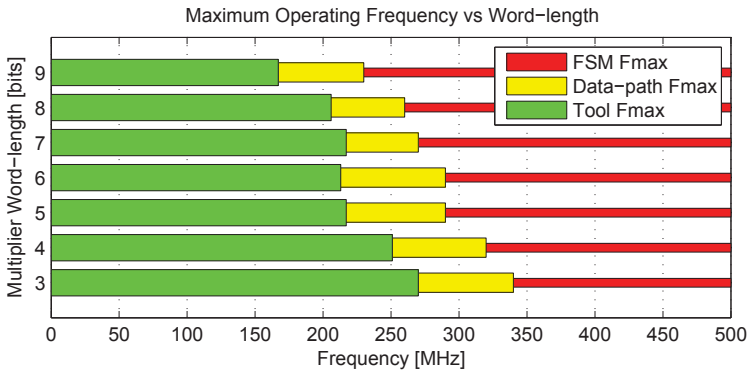

Figure 5.15.: Maximum clock frequencies *vs* word length for a Z^6 to Z^3 linear projection circuit designed by the KLT transform.

Figure 5.16 shows the actual performance of the designs produced by the proposed optimisation framework and the KLT approach, along with their predicted performances. The predicted performance for the KLT-based designs is based on the extension of the existing methodology and the adoption of the objective function, equation 5.7. However, no optimisation with respect to over-clocking characterisation has been performed in the KLT-based designs. The results show that the proposed optimisation framework produces designs that behave close to the predicted results under over-clocking, as well as they produce around an order of magnitude on average lower reconstruction error for the same area when compared to the KLT. All in all, the results are in line with the expected increased performance of the optimised designs of the KLT-based ones, and in most cases the optimised designs perform close to the predicted by the model.

5.8.5. Optimisation of Linear Projections for Throughput and Errors

To better demonstrate the impact of variation for each design strategy only one setting has been changed, even though the framework supports variation of many operating conditions simultaneously. The results for the reference implementation

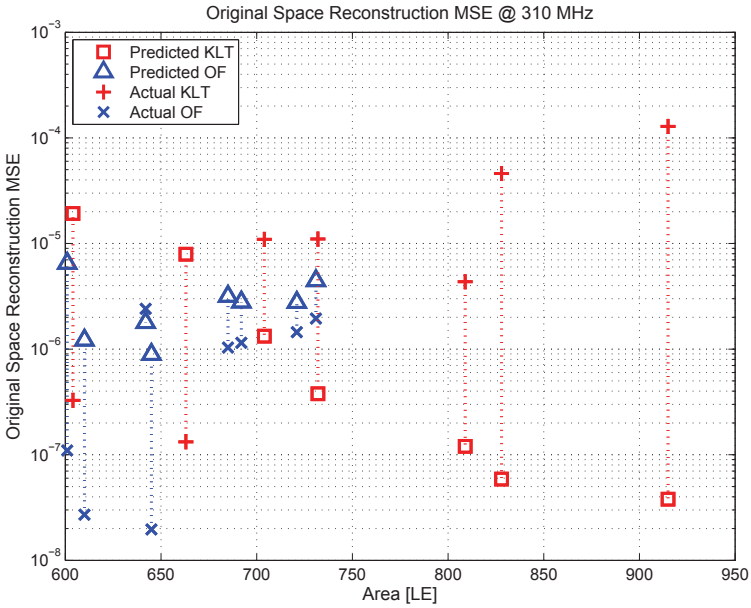

Figure 5.16.: MSE for the reconstruction of the projected data in the original space. The design points of the KLT correspond to 3-9 bit coefficient word length.

without information about the characterisation of the device are identified with *KLT*, whereas the results for the proposed framework are identified with *NEW*. They are compared in terms of PSNR of the reconstructed data in the original space.

Regarding the test circuit it was verified that after synthesis, the tool reported a resource usage of 126 logic cells and 3 9×9 embedded multipliers, and a maximum clock frequency of 342 MHz. Examining the timing report revealed that the critical paths belong to the embedded multiplier and the delay for the remaining components in the datapath, i.e. accumulator, and the FSM, are out of reach for the selected over-clocking frequencies.

Targeting Maximum Performance Optimising a linear projection design aiming for the maximum performance implies an increased FPGA core voltage and active cooling of the device. During the characterisation of the multipliers and test

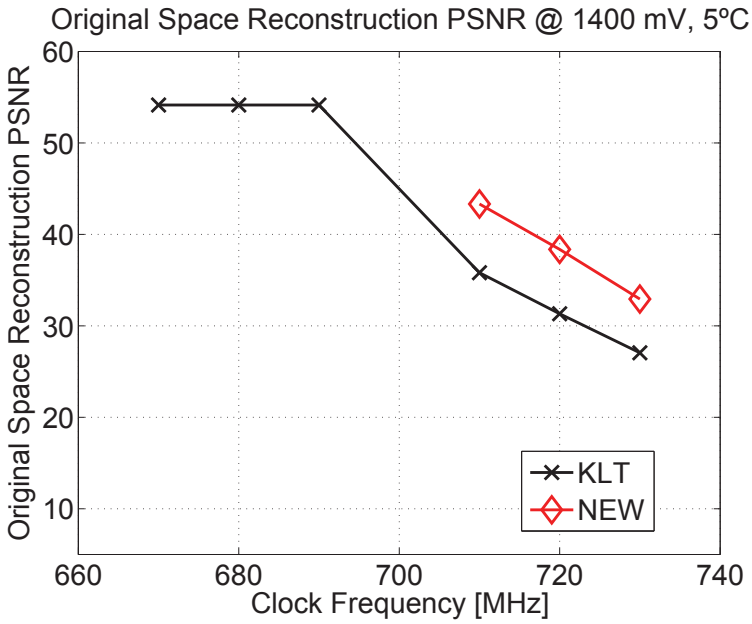

Figure 5.17.: Comparison of the performance of the KLT and OF designs for the particular case of 1400 mV and 5 °C.

of the designs generated, the device was kept at 5 °C and supplied with 1400 mV, instead of the 1200 mV specified by the manufacturer.

With a clock frequency twice as much as the maximum specified by the synthesis tool for the normal working conditions, the designs generated by the proposed framework exhibited a reconstruction PSNR up to 15 dB better than the KLT designs for the same working conditions, as can be observed in Figure 5.17. On the other hand, if a target PSNR of 30 dB is to be met, then the designs generated by the framework can operate up to 20 MHz higher than the KLT designs.

Targeting Low Voltage Linear projection circuits operating under limited power budgets, or battery operated, tend to operate using the least core voltage possible and be without any active cooling components. Figure 5.18 shows the results for the KLT designs when operating at 35 °C with different FPGA core voltages. This design strategy considered 900 mV as the minimum core voltage for the FPGA.

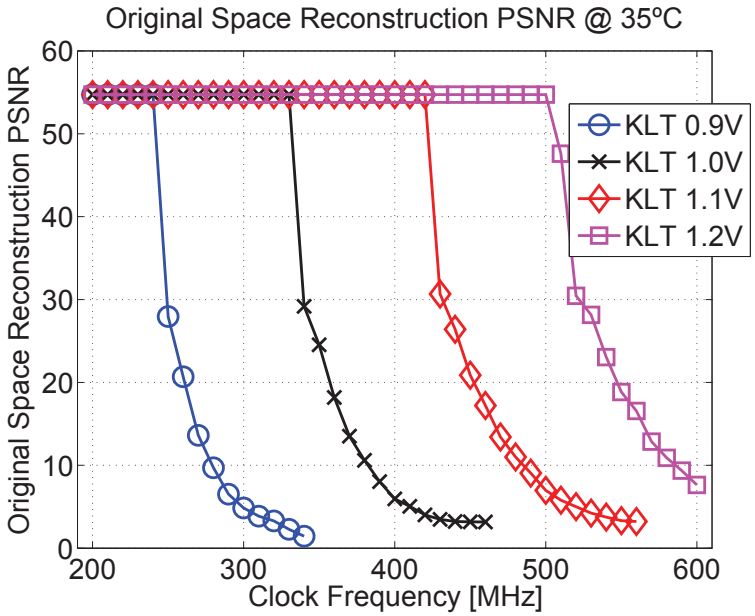

Figure 5.18.: Performance of the KLT linear projection application under different core voltages (900 mV,1000 mV,1100 mV,1200 mV).

Figure 5.19 shows that the designs created by the framework achieve a better PSNR up to 10 dB for the same clock frequency, or for similar PSNR, a clock frequency up to 10 MHz higher than the reference designs.

Targeting Process Variation Previously it was shown, through characterisation, that different embedded multipliers perform differently as a consequence of process variation. To demonstrate that the optimisation performed addresses the impact of variability of the device, in the design process, a previously optimised, and synthesised, design was placed on a different DE0 board and tested. Figure 5.20 demonstrates the aforementioned as NEW (design optimised using the characterisation from a different DE0 board) performs similar to the KLT implementation, which has no information about the targeted device. Notwithstanding, all of these designs (KLT and NEW) perform with increased reconstruction PSNR in the new board (figure 5.20) than the DE0 board originally used in the optimisation (figure

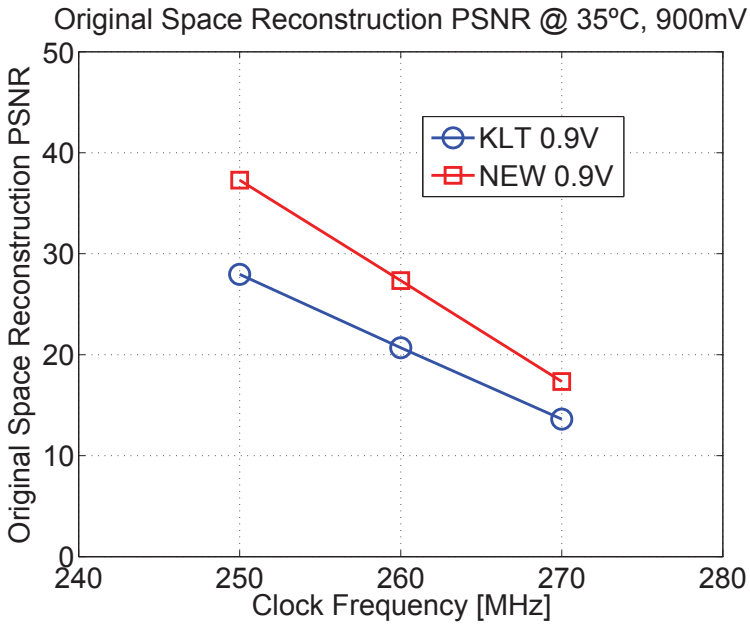

Figure 5.19.: Performance comparison between the KLT and OF designs for the particular case of 900 mV.

5.19). This is due to fact that the new board, on average, has embedded multipliers with smaller delays, a consequence of inter-die variation.

Targeting Device Temperature Tolerance It is well established that temperature affects the performance of silicon devices. Implementing linear projection designs without any active cooling components, and operating them in environments prone to large temperature variation can compromise their correct functioning. Usually, if an implementation of a linear projection has to consider a wide range of temperatures, then it will have to cope with the worst performance of them.

To go beyond with the optimisation methodology, it was considered a scenario where a single design could offer better performance than the reference designs for a range of temperatures, instead of a design for the worst-case temperature. Seeing that the errors increase with the temperature, optimising a design for the worst-case temperature can restrict the coefficients available to implement the linear projection

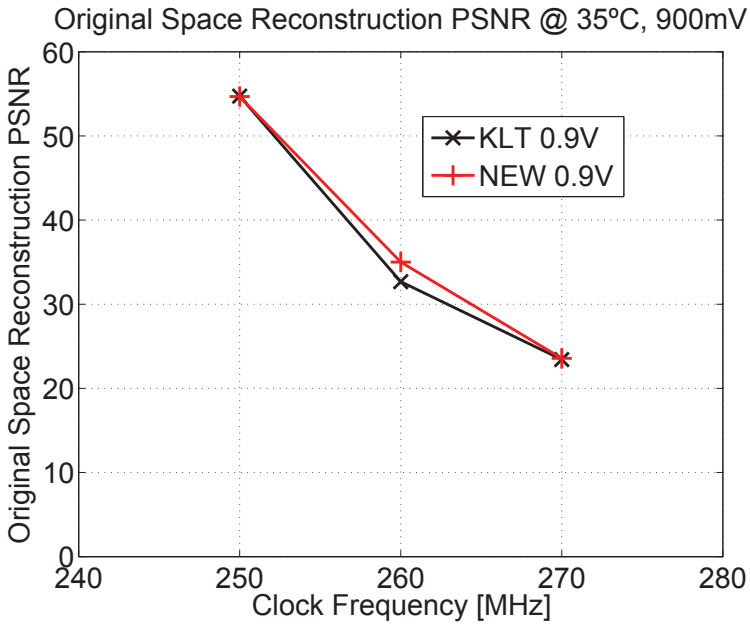

Figure 5.20.: Performance of the KLT optimised designs for low-power, tested on a different Cyclone III FPGA.

design, hence placing a ceiling on the best reconstruction MSE that a particular design could achieve, even without errors in its datapath.

The new idea is focused on sampling linear projection designs using the information from the characterisation of the device at specific temperatures along with its probability to operate under those temperatures. To accomplish this, it was investigated the weighted average of the characterisation errors for a range of operating temperatures in the generation of linear projection designs. As follows, the prior distribution from equation (5.16) is now:

$$g\left(E\left(\lambda_{pk}, f, p, v, T\right)\right) = \sum_{i} \gamma_i c \left[1 + E\left(\lambda_{pk}, f, p, v, T_i\right)\right]^{-\beta} \tag{5.16}$$

Here i iterates over all contributing temperatures, and $\sum_i \gamma_i = 1$. The different weights represent the probability of the design to operate at those conditions. This particular test case used temperatures 20, 35 and 50 °C and $\gamma_{20} = 0.3$, $\gamma_{35} = 0.5$

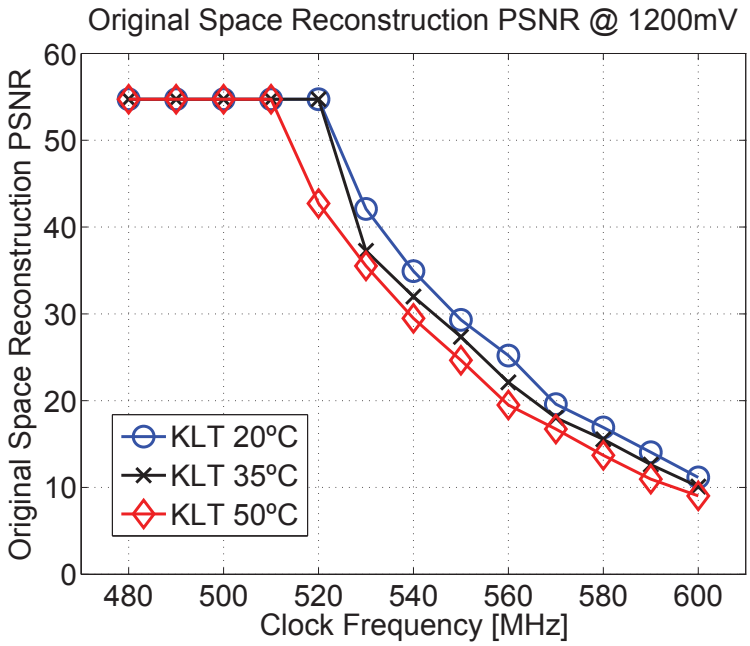

Figure 5.21.: Performance of the KLT linear projection application depending on the temperature of the device.

and $\gamma_{50} = 0.2$. In practice, the proposed framework generates circuit designs per clock frequency, covering all the temperatures within the expected range. They are identified with OF WAVG in the results. Designs optimised for a single temperature are identified with NEW 20/35/50°C.

Figure 5.21 shows how the performance of the reference linear projection circuit varies with the temperature of the device, with a supply voltage of 1200 mV. Figure 5.22 shows in detail the comparison between the reference and the optimised designs for a set of temperatures. Figures 5.23 and 5.24 hold the results for 35 and 50 °C, respectively.

The figures show that the designs generated by the framework always outperform the KLT designs for all temperatures. Furthermore, at 530 MHz the PSNR is more than 10 dB better than the KLT design, and at 530 MHz the performance of the NEW designs at 35 °C is better than the KLT design at 20 °C. The NEW designs

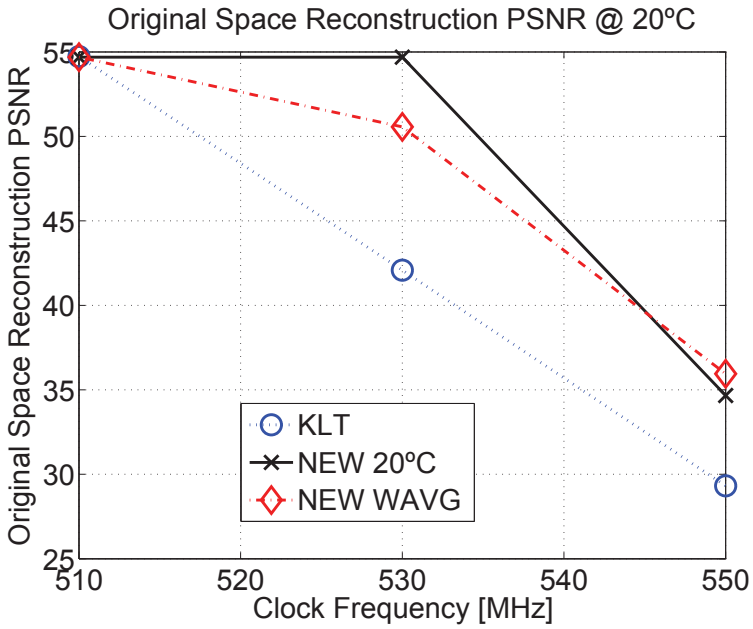

Figure 5.22.: Performance comparison between the KLT and OF designs at 20 °C (right).

perform significantly better than the NEW WAVG ones since they are optimised for a particular temperature whereas the NEW WAVG cover a range of operating conditions. Nonetheless, at 530 MHz, the NEW WAVG designs performed slightly better than the NEW 35°C since 50% of the information for the NEW WAVG is the same as in the NEW 35°C design.

5.8.6. Scalability & Run-Time Investigation

The proposed optimisation framework can support other linear projection problems with different spaces and input data characteristics. Figure 5.25 shows the optimisation scaled to a Z^{10} to Z^4 linear projection. In this figure, it's observable that the optimised designs perform better than the reference ones. This optimisation process involved 100 samples in the generation of the designs, and they were tested with 500 samples.

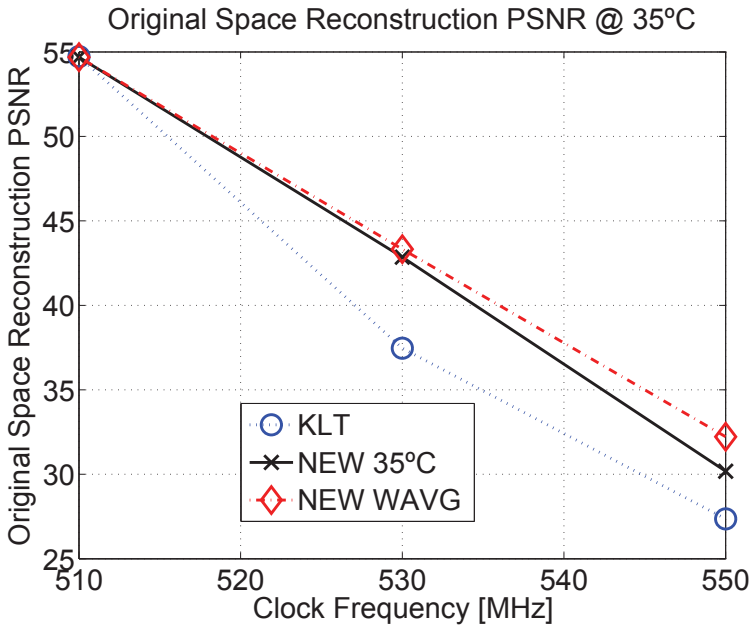

Figure 5.23.: Performance of the KLT and OF designs at 35 °C .

The run-time requirements of the proposed optimisation framework has also been investigated, when the proposed framework is executed in a Intel Core-i7 processor. A model was derived from the approximation of various observed run-times. It provides an estimate (i.e. in seconds) for the optimisation process under difference settings.

$$Time = (1 + Q(K-1)) \sum_{1}^{\#HP} \sum_{1}^{\#Freqs} \sum_{1}^{\#wl} R(wl) \tag{5.17}$$

$$R(wl) = 0.4266 \times exp^{(0.6427 \times wl)} \tag{5.18}$$

Equation (5.18) models the time to sample one projection vector of a given word length, where (5.17) models the required time to sample a complete set of designs for a given number of clock frequencies ($\#Freqs$), projected dimensions (K), maintained designs (Q), values of *Hyper-Parameter* β ($\#HP$), and word lengths (wl). The results for both equations are in seconds. As an example, the execution of the

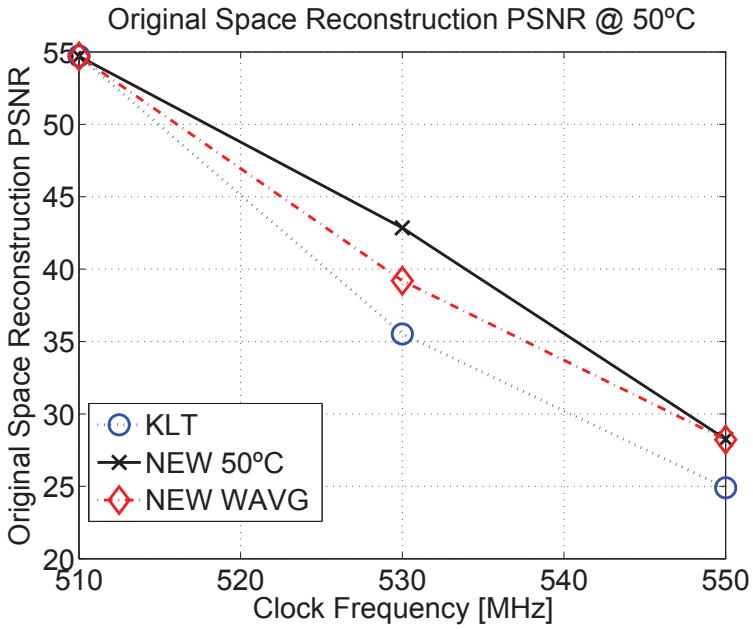

Figure 5.24.: Performance of the KLT and OF designs at 50 °C.

proposed optimisation framework using ($\#Freqs = 1$, $K = 3$, $Q = 5$, $\#HP = 2$, $wl = [3..9]$), the processing time is 1 hour and 44 minutes, which is considered acceptable for design an optimisation of digital circuits.

5.9. Summary

This work proposes a novel approach for acceleration of Linear Projections by introducing the idea of device specific performance characterisation to address the impact of variation. As such, it makes use of a framework to characterise the performance of multipliers on a specific FPGA device under variation of the operating conditions. Moreover, a novel approach is introduced for the utilisation of such information for the design and optimisation of a Linear Projection circuit design for performance and resilience simultaneously. The work shows that high-performance improvements can be achieved when considering such device oriented optimisations, specific to FPGA

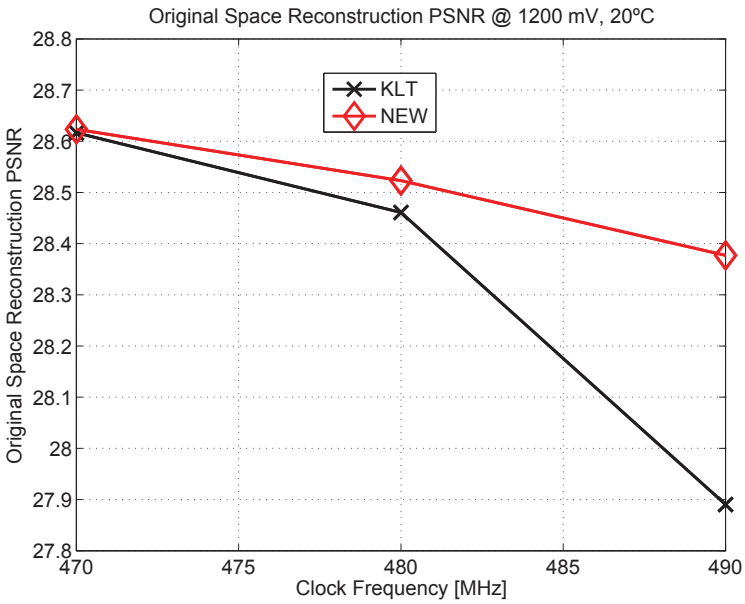

Figure 5.25.: Performance of the KLT and OF designs for a Z^{10} to Z^4 linear projection at 1200 mV/20 °C.

devices due to their reconfigurability properties, that are not possible through the available synthesis tools. On the other hand, the main limitations of this optimisation framework are the run-time, accuracy in the predicted performance, and the requirement of an error model per device.

To extend this work to different applications it requires the problem to be specified through an objective function, which is to be minimised given the problem data and characterisation of the device.

Figure 5.26 shows how other research areas can relate with the proposed optimisation framework, namely probabilistic and adaptive computing, fault tolerance, and design optimisation techniques.

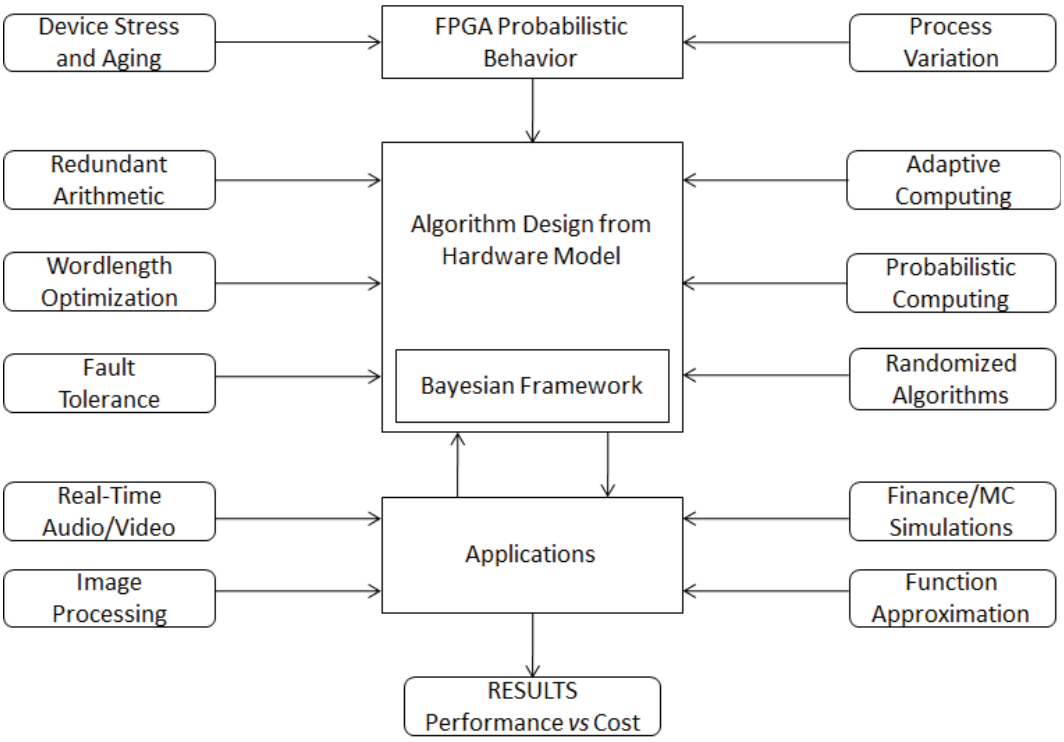

Figure 5.26.: Diagram with the relation between the different research areas around the proposed optimisation framework.

6

Conclusions and Future Work

6.1. Conclusions

The constant scaling in the fabrication process has led to devices exhibiting an increase in their process variation. As a consequence, synthesis tools are becoming more conservative, thus lowering the maximum clock frequency that a design could achieve. Hence, when the maximum throughput offered by traditional design techniques isn't enough, over-clocking the design is a method to increase it. However, this makes the design susceptible to produce errors as a consequence of variation in the operating conditions of the device. This thesis investigated methods to assess the impact of errors on arithmetic units and applications, on devices under

variation, as well as methods to mitigate those errors.

The proposed characterisation framework tried to model arithmetic units, and the linear projection circuit, on FPGA devices under variation. Initially it was thought as a method to characterise the same units under extreme over-clocking. It has demonstrated to be an important alternative to existing characterisation methods, as it creates statistics on the actual error values, rather than the bits. This way it is possible to understand the values applications are going to process under those operating conditions. The framework also has shown that some units can be employed to accelerate designs with no, or little, errors, achieving graceful degradation.

Reduced-Precision Redundancy is a generic method to mitigate errors in the datapaths where the arithmetic units always generate errors and there's no mechanism to mitigate, or recover, from those errors. In this sense, the new RPR scheme fulfills the need for a generic method that can provide resilience to a datapath without introducing extra latency, neither having to change the implementation of the algorithm. The (non-trivial) solution imagined proved to work by accelerating the units in the datapath while controlling the errors. Despite the fact that the novel architecture requires twice the LEs and 2 ROMs, tests have showed that the quality of the results obtained at the output of the RPR unit can't be achieved by other mitigation methods for the same operating conditions. Moreover, even though only timing errors were considered in this work, it can be utilised to mitigate permanent faults.

However, there are scenarios where resources are scarce and hence it's not feasible to add extra resources to mitigate errors. On this account the optimisation framework uses information from the previous characterisation to create an error model. From that model it generates linear projection designs, through an inference method, that can produce results with less errors, when compared to traditional implementations operating under the same conditions. This method is suitable to

be adopted in FPGAs, due to its reconfigurability properties, as it allows to have a prior characterisation and a later implementation on the same device. It was also identified that when accounting for timing errors, throughput, errors and area, the designs generated by the OF using DSP-based multipliers were the ones offering the best tradeoff.

6.2. Future Work

Following the work presented in this thesis, there are several short-term research directions that could be used to enhance, or extend, it. Other questions, related to the utility of this research work in the future, and other research avenues that can arise from this work, are discussed in the long-term goals.

6.2.1. Short-Term Goals

Characterisation Framework

As it is demonstrated in the work presented in this thesis, characterisation of the arithmetic units is an essential step towards the optimisation of DSPs designs. The main limitation of the current implementation is the run-time, hence it would be beneficial to accelerate this process. In more detail, the duration of the characterisation process is unobservable when compared to the time devoted to the communication between the FPGA board and the host computer, therefore replacing the control interface would allow a faster characterisation. In terms of data used in the characterisation, it would be interesting to investigate the impact of different data sets in the results in order to create generic models for different classes of DSP applications (e.g. faces, motion sensors, radar). Data from different sources has different properties, as consequence the same arithmetic units exhibit different responses for the same operating conditions. This is something that doesn't favour the adoption

of a unified error model. Regarding the data produced from the characterisation, in applications where often the timing errors reside in bits other than the MSBits, determining the Most-Prone-To-Error-Bits can open new research avenues to protect the bits that impact the application the most, in devices with limited resources.

Optimisation Framework

Results demonstrated that the OF offers the best tradeoff in terms of throughput, timing errors and area is achieved by the OF using DSP blocks. Since this is a limited resource for some applications, future work would be applying the OF to assign the most prone to error coefficients to the DSP blocks while assigning the other to LEs and BRAMs. Hence, optimisation for heterogeneous resources and errors would be achieved at the same time.

On the other hand, having showed the benefit of the framework, it is of great interest to extend the framework to support other problems such as digital filters. [96] and [97] have proposed Bayesian formulations for FIR filters which can be adapted to the optimisation framework proposed.

In terms of execution bottleneck, Gibbs sampling algorithm consumes most of the time in the optimisation process. Since it's an iterative method it could be worth looking at alternative methods to produce samples, e.g. Tabu search [98], or to enhance the optimisation framework using Bayesian Programming [99].

Reduced-Precision Redundancy

This thesis proposes a novel adaptation of the RPR using a different architecture and methods to generate approximations to correct errors. Regarding this work, the approximation function generation is an issue that deserves further attention, along with modeling of the output of an RPR unit under variation.

In more detail, investigation of an optimisation for the selection of the parameters

within a given resource budget is expected to reduce the number of design decisions made by the user, thus automating the process. So far only the error-free regime has guaranteed correctness at the output. Hence, there's opportunity to investigate a model for the output of the RPR unit when operated under variation.

Moreover, there are opportunities for more detailed studies on:

- support of other, more complex, arithmetic operators;
- optimisation of the process to searching for the function approximation coefficients to minimise the error functions, e.g. optimise the search of all useful combinations [100], and using genetic algorithms [49];
- add information about characterisation of the arithmetic units to protect the most-prone-to-error-bits instead of the MSBits.

In this direction, [101] has proposed a methodology to generate approximate circuits from a "golden RTL" and a quality constraint that defines the amount of error introduced. Small approximate circuits can be an alternative to ROMs when BRAMs aren't available on implement RPR on the device.

Since both RPR and the optimisation framework achieve better results than any other methods, it would be interesting to make a comparison between them considering designs from the optimisation framework using RPR units, and compare them with the other implementations.

6.2.2. Long-Term Goals

Reliability of digital systems has enjoyed from many contributions, but it's still far from complete, therefore it's foreseen that this is a research topic which still needs to be paid attention to in the future. On the specific topic of optimisation of designs to operate under PVT variation, the current trend indicates that process variability will increase, thus increasing the impact of contributions to mitigate

it. On this account, the increase of intra and inter-die variation, will push further the adoption of a per device optimisation. Characterisation offers opportunity for implementations with *personalised* circuits which can be advantageous for achieving the maximum performance offered by a fabrication process. Additionally, an interesting by-product is the promotion of security in circuit designs by turning the replication of the circuit unfeasible (i.e. Physical Unclonable Functions). Being the time of the characterisation process, and the cross-platform support for, a limitation in terms of its mass adoption. It could be surpassed if the device vendors were to acquire this information during the fabrication process (binning) and register it on user-accessible storage on the device.

The other existing limitation concerns the prediction of errors. This can be achieved in two different classes of approaches: characterise-model-optimise, hardening of circuits and create different architectures, designed for specific error patterns. The first class is related to the work presented in this thesis, where arithmetic units were characterised and selected according to their performance under variation. The second class promotes the hardening of the basic elements, or building-blocks, at the expense of many resources, and time, to guarantee its correct operation. To best of the authors knowledge, there's no contribution that resembles the third class, being the best approximation to it redundant, or online, arithmetic, which *naturally* protects the MSBits but at the expense of large area requirements, lower operating clock frequencies and extra latency.

As it was observed in this work, one of the difficulties is to model the errors as they depend on the data of the problem being considered, the process variation, the placement and routing and the operating conditions. It's anticipated a demand for more accurate simulation models to make the bridge between architectures for arithmetic units and unreliable silicon devices. Regarding this matter, in terms of implementation platform, FPGAs enjoy from combined properties that other

technologies can't compete against: reconfigurability and custom design.

A

Appendix

A.1. Hardware Platforms

This work was carried out on various hardware platforms using various FPGAs from Altera. Tables A.1 and A.2 compile the information for each board used, and its respective FPGA.

Board	FPGA Family	FPGA Device	Vendor
DE0	Cyclone III	3C16	Terasic
DE0 Nano	Cyclone IV	4CE22	Terasic
BeMicro CV	Cyclone V	5CEA2	Arrow

Table A.1.: Hardware platforms used in this work, and their main features.

Device	Process	LEs	18x18 Mult.	BRAMs
3C16	65 nm	15.4k	56	56
4CE22	60 nm	22.3k	66	66
5CEFA2	28 nm	25.0k	50	176

Table A.2.: FPGAs used in this work, and their main features.

A.2. FPGA Core Voltage and Temperature Control

To control and keep the operating conditions of the FPGA constant during the experiments, the core voltage and the temperature on the top of the FPGA were set externally to the design under test on the FPGA.

On the FPGA, the core voltage is set by a digitally controlled power supply (PSU) from TTI [102]. The temperature is controlled by an Proportional Integral and Derivative (PID) controller, running on an Arduino [103], using a Thermoelectric Cooler (TEC) as the active cooling/heating element. The temperature was calibrated with a commercial thermometer from Lascar Electronics [104]. Both power supply and temperature controller are controlled via independent serial ports on the host computer.

The operating conditions are set by a client, by sending commands to a Python server that runs on the host computer. This computer establishes the connection between the serial ports and the test script running in Matlab, via Transmission Control Protocol (TCP) sockets.

Figure A.1 depicts the elements of this system and its interfaces with the host computer. A Universal Serial Bus (USB) port is used to control the power supply (PSU) connected to the power pins on the target FPGA board. The other USB port is connected to the Arduino used to control the temperature via a TEC. The temperature sensors for the temperature controller are placed between the top of the FPGA and the TEC. On the other side of the TEC there's a water-cooled heat-sink to dissipate the excess thermal energy. The FPGA board is placed inside a sealed

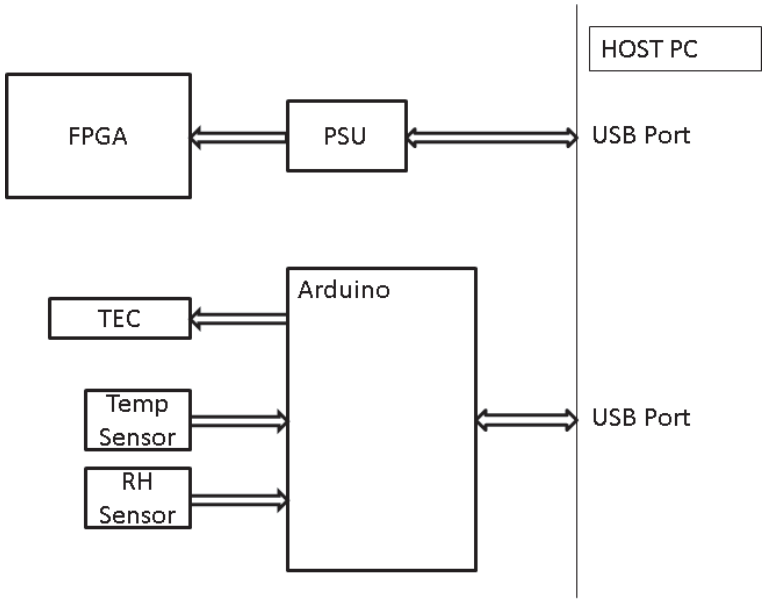

Figure A.1.: System to control core voltage and temperature of the FPGA.

container along with a dehumidifier in order to avoid formation of condensation, which could damage the circuit. A relative humidity (RH) sensor is connected to the Arduino, and it is used to monitor the humidity inside a plastic container. In case the relative humidity increases above a threshold, set below the dew point for a pair of device and ambient temperatures, the software automatically stops the test to avoid formation of condensation on the FPGA board. The photographs of the DE0 and DE0 Nano boards equipped with the temperature control setup is depicted in figure A.2.

A.3. Source code

generate_data.m

```
1 function [Cs, C, W, Psi, Y, X, E] = generate_data(D, N)
2 %function [Cs, C, W, Psi, Y, X, E] = generate_data(D, N)
3 % The function generates data according to a specific model for test
```

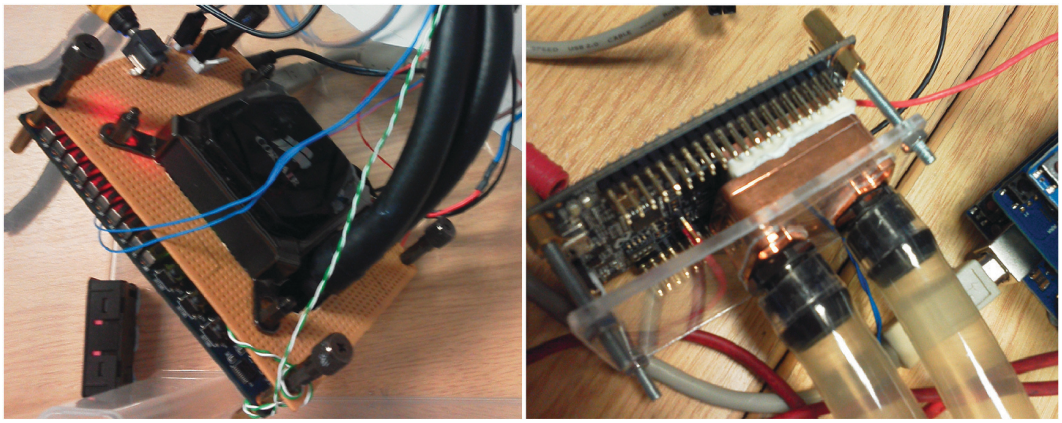

Figure A.2.: Photo of the DE0 and DE0 Nano boards equipped with a thermoelectric cooler and a water-cooled heat-sink to control the temperature on the surface of the FPGAs.

```
4  % purposes. The model under investigation is:
5  %   y = W*x + e
6  % where W is the projection matrix and e is the error that follows
7  % a normal distribution with zero mean.
8  % The input D(1) is the number of dimensions of the origina space y
9  % and the input D(2) is the number of dimensions in the smaller space
10 % The function returns:
11 % Cs - the sample covariance matrix of the data
12 % C - the theoretical covariance matrix
13 % W - the projection matrix
14 % Psi - the covariance matrix of the noise (theoretical)
15 % Y - High Dim data
16 % X - Reduced Dim data
17 % E - error data
18 %
19 % Assumptions
20 % 1. The covariance matrix for the noise is simple. It is diagonal with the
21 %    same variance.
22 % e.g generate_data([OriDim RedDim], NumCases)
23
```

```
24 settings
25
26 % Check the input
27 if D(1)<=D(2)
28     error('The first entry of the D matrix should be larger than the second entry.');
29 end
30
31 % Generate the W matrix
32 W = randn(D);
33
34 % Generate the Psi matrix for the noise. Scale the variance by a number
35 % noise_scale = 1E-3;
36 Psi = diag(ones(D(1),1) * rand(1) * sets.NOISE_SCALE);
37
38 % Final optimum covariance matrix is given by:
39 C = W*W' + Psi;
40
41 % Generate samples and the sample covariance matrix
42 % I need this step in order to assess the quality of the reconstruction
43 X = zeros(D(2),N);
44 E = zeros(D(1),N);
45 MU = zeros(D(1),1);
46 for i=1:N
47     X(:,i) = (mvnrnd(zeros(D(2),1), eye(D(2))))';
48     E(:,i) = (mvnrnd(MU, Psi))';
49 end
50 M = mean(X,2);
51 X = X-repmat(M,1,N);     % descentra os dados = remove bias
52 Y = W*X + E ;
53 Cs = (1/N) * Y*Y';  % sample covariance matrix
54
55 %eof
```

Bibliography

[1] Altera, "Cyclone III device handbook." Online. `http://www.altera.co.uk/literature/hb/cyc3/cyclone3_handbook.pdf`.

[2] Altera, "Cyclone IV device handbook." Online. `http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf`.

[3] Altera, "Cyclone V device handbook." Online. `http://www.altera.com/literature/hb/cyclone-v/cyclone5_handbook.pdf`.

[4] Z. Guan, J. Wong, S. Chaudhuri, G. Constantinides, and P. Cheung, "A two-stage variation-aware placement method for fpgas exploiting variation maps classification," in *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pp. 519–522, Aug 2012.

[5] C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," 2003.

[6] C.-S. Bouganis, I. Pournara, and P. Cheung, "Exploration of heterogeneous FPGAs for mapping linear projection designs," vol. 18, no. 3, pp. 436–449, 2010.

[7] P. Sedcole and P. Y. K. Cheung, "Parametric yield modeling and simulations of FPGA circuits considering within-die delay variations," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 1, pp. 10:1–10:28, June 2008.

[8] J. M. Gilbert and W. Yang, "A real-time face recognition system using custom vlsi hardware," 1993.

[9] M. de Kruijf and K. Sankaralingam, "Exploring the synergy of emerging workloads and silicon reliability trends," in *SELSE*, 2009.

[10] N. Shanbhag, K. Soumyanath, and S. Martin, "Reliable low-power design in the presence of deep submicron noise," in *Low Power Electronics and Design, 2000. ISLPED '00. Proceedings of the 2000 International Symposium on*, pp. 295 – 302, 2000.

[11] B. Shim and N. Shanbhag, "Reduced precision redundancy for low-power digital filtering," in *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on*, vol. 1, pp. 148–152 vol.1, 2001.

[12] H. Ngo, R. Gottumukkal, and V. Asari, "A flexible and efficient hardware architecture for real-time face recognition based on eigenface," in *VLSI, 2005. Proceedings. IEEE Computer Society Annual Symposium on*, pp. 280–281, May 2005.

[13] J. Nascimento and J. Bioucas Dias, "Vertex component analysis: a fast algorithm to unmix hyperspectral data," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 43, pp. 898–910, April 2005.

[14] L. Ke and R. Li, "Classification of eeg signals by multi-scale filtering and pca," in *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, vol. 1, pp. 362–366, Nov 2009.

[15] J.-U. Chu, I. Moon, and M. seong Mun, "A real-time EMG pattern recognition system based on linear-nonlinear feature projection for a multifunction myoelectric hand," *Biomedical Engineering, IEEE Transactions on*, vol. 53, no. 11, pp. 2232–2239, 2006.

[16] R. Duarte and C. Bouganis, "High-level linear projection circuit design optimization framework for FPGAs under over-clocking," in *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pp. 723–726, Aug 2012.

[17] R. P. Duarte and C.-S. Bouganis, "A unified framework for over-clocking linear projections on FPGAs under PVT variation," in *Applied Reconfigurable Computing (ARC), 2014 10th International Symposium on*, pp. 49–60, 2014.

[18] R. P. Duarte and C.-S. Bouganis, "Pushing the performance boundary of linear projection designs through device specific optimisations (abstract only)," in *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, FPGA '14, (New York, NY, USA), pp. 245–245, ACM, 2014.

[19] R. P. Duarte and C.-S. Bouganis, "Over-clocking of linear projection designs through device specific optimisations," in *21st Reconfigurable Architectures Workshop (RAW 2014)*, pp. 9–60, 2014.

[20] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1st ed., 1994.

[21] K. Pearson, "The problem of the random walk," *Nature*, vol. 72, p. 342, aug 1905.

[22] J. Taur and C. W. Tao, "Medical image compression using principal component analysis," in *Image Processing, 1996. Proceedings., International Conference on*, vol. 1, pp. 903–906 vol.2, 1996.

[23] I. Fodor, "A survey of dimension reduction techniques," tech. rep., 2002.

[24] E. A. Stott, J. S. Wong, N. P. Sedcole, and P. Y. K. Cheung, "Degradation in FPGAs: measurement and modelling," in *FPGA*, pp. 229–238, 2010.

[25] J. S. J. Wong, P. Sedcole, and P. Y. K. Cheung, "Self-measurement of combinatorial circuit delays in FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, pp. 10:1–10:22, June 2009.

[26] P. Sedcole, J. S. Wong, and P. Y. K. Cheung, "Characterisation of FPGA clock variability," in *Proc. IEEE Computer Society Annual Symp. VLSI ISVLSI '08*, pp. 322–328, 2008.

[27] Altera, "System design with advance FPGA timing models." Online, February 2014. `www.altera.com/literature/wp/wp-01213-advance-fpga-timing-models.pdf`.

[28] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 2366–2369, Aug 2010.

[29] L. Cheng, J. Xiong, L. He, and M. Hutton, "FPGA performance optimization via chipwise placement considering process variations," in *Proc. Int. Conf. Field Programmable Logic and Applications FPL '06*, pp. 1–6, 2006.

[30] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Field-Programmable Logic and Applications*, pp. 213–222, 1997.

[31] H. Yu, Q. Xu, and P.-W. Leong, "Fine-grained characterization of process variation in FPGAs," in *Field-Programmable Technology (FPT), 2010 International Conference on*, pp. 138–145, Dec 2010.

[32] J. S. J. Wong and P. Y. K. Cheung, "Timing measurement platform for arbitrary black-box circuits based on transition probability," 2013.

[33] N. Banerjee, J. H. Choi, and K. Roy, "A process variation aware low power synthesis methodology for fixed-point FIR filters," in *Proceedings of the 2007 international symposium on Low power electronics and design*, ISLPED '07, (New York, NY, USA), pp. 147–152, ACM, 2007.

[34] I.-C. Park and H.-J. Kang, "Digital filter synthesis based on minimal signed digit representation," in *Proceedings of the 38th annual Design Automation Conference*, DAC '01, (New York, NY, USA), pp. 468–473, ACM, 2001.

[35] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Slack redistribution for graceful degradation under voltage overscaling," in *Asia and South Pacific Design Automation Conference*, pp. 825–831, 2010.

[36] J. Jung and T. Kim, "Timing variation-aware high-level synthesis," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pp. 424–428, Nov 2007.

[37] V. Lakamraju and R. Tessier, "Tolerating operational faults in cluster-based FPGAs," in *8th international ACM/Sigda symposium on field programmable gatearrays*, pp. 187–194, 2000.

[38] E. Stott and P. Y. K. Cheung, "Improving FPGA reliability with wear-levelling," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pp. 323–328, Sept 2011.

[39] Y. Xie and Y. Chen, "Statistical high-level synthesis under process variability," *IEEE Design & Test of Computers*, vol. 26, no. 4, pp. 78–87, 2009.

[40] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," vol. 42, no. 9, pp. 569–577, 1995.

[41] L. Ciminiera and P. Montuschi, "Carry-save multiplication schemes without final addition," *Computers, IEEE Transactions on*, vol. 45, pp. 1050 –1055, sep 1996.

[42] K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Wiley, 1999.

[43] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," vol. 22, no. 10, pp. 1432–1442, 2003.

[44] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, *Synthesis and optimization of DSP algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 2004.

[45] N. Herve, D. Menard, and O. Sentieys, "Data wordlength optimization for FPGA synthesis," in *Proc. IEEE Workshop Signal Processing Systems Design and Implementation*, pp. 623–628, 2005.

[46] J. Deschamps, G. Bioul, and G. Sutter, *Synthesis of arithmetic circuits: FPGA, ASIC, and embedded systems*. John Wiley, 2006.

[47] G. Caffarena, G. A. Constantinides, P. Y. K. Cheung, C. Carreras, and O. Nieto-Taladriz, "Optimal combined word-length allocation and architectural synthesis of digital signal processing circuits," vol. 53, no. 5, pp. 339–343, 2006.

[48] D. Kodek, "Design of optimal finite wordlength FIR digital filters using integer programming techniques," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, pp. 304 – 308, jun 1980.

[49] E. Darulova, V. Kuncak, R. Majumdar, and I. Saha, "Synthesis of fixed-point programs," in *EMSOFT*, pp. 1–10, 2013.

[50] R. E. Moore, "Automatic error analysis in digital computation," Technical Report Space Div. Report LMSD84821, Lockheed Missiles and Space Co., Sunnyvale, CA, USA, 1959.

[51] J. Cong, K. Gururaj, B. Liu, C. Liu, Z. Zhang, S. Zhou, and Y. Zou, "Evaluation of static analysis techniques for fixed-point precision optimization," in *Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on*, pp. 231–234, April 2009.

[52] A. Ahmadi and M. Zwolinski, "A symbolic noise analysis approach to word-length optimization in DSP hardware," in *Proc. Int. Symp. Integrated Circuits ISIC '07*, pp. 457–460, 2007.

[53] S. E. McQuillan and J. V. McCanny, "A systematic methodology for the design of high performance recursive digital filters," vol. 44, no. 8, pp. 971–982, 1995.

[54] C. E. Shannon, "A mathematical theory of communication," vol. 27, pp. 379–423, July 1948.

[55] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 26, no. 2, pp. 147–160, 1950.

[56] J. Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," 1956.

[57] M. A. Breuer, "Adaptive computers," *Information and Control*, vol. 11, no. 4, pp. 402–422, 1967.

[58] P. K. Krause and I. Polian, "Adaptive voltage over-scaling for resilient applications," in *Proc. Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, pp. 1–6, 2011.

[59] D. Roberts, T. Austin, D. Blauww, T. Mudge, and K. Flautner, "Error analysis for the support of robust voltage scaling," in *Proc. Sixth Int. Symp. Quality of Electronic Design ISQED 2005*, pp. 65–70, 2005.

[60] U. Sharma, "Fault tolerant techniques for reconfigurable platforms," in *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*, A2CWiC '10, (New York, NY, USA), pp. 60:1–60:4, ACM, 2010.

[61] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D. M. Bull, and D. T. Blaauw, "RazorII: In situ error detection and correction for PVT and SER tolerance," vol. 44, no. 1, pp. 32–48, 2009.

[62] R. Hegde and N. R. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *Proceedings of the 1999 International Symposium on Low Power Electronics and Design*, ISLPED '99, (New York, NY, USA), pp. 30–35, ACM, 1999.

[63] J. Huang, J. Lach, and G. Robins, "A methodology for energy-quality tradeoff using imprecise hardware," *DAC*, 2012.

[64] Y.-K. Cheng and Y.-H. Huang, "Frequency-overscaling dsp circuit design with reduced-precision redundancy and subword detection processing," in *Communications, Circuits and Systems, 2009. ICCCAS 2009. International Conference on*, pp. 431–434, July 2009.

[65] K. S. Morgan, D. L. McMurtrey, B. H. Pratt, and M. J. Wirthlin, "A comparison of TMR with alternative fault-tolerant design techniques for FPGAs," vol. 54, no. 6, pp. 2065–2072, 2007.

[66] M. Sullivan, H. Loomis, and A. Ross, "Employment of reduced precision redundancy for fault tolerant FPGA applications," in *Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on*, pp. 283–286, 2009.

[67] B. Pratt, M. Fuller, and M. Wirthlin, "Reduced-precision redundancy on FPGAs," *Int. J. Reconfig. Comput.*, vol. 2011, pp. 3:3–3:3, Jan. 2011.

[68] C. Carmichel, "Triple module redundancy design techniques for Virtex FPGAs." Xilinx website, July 2006.

[69] R. Hentschke, F. Marques, F. Lima, L. Carro, A. Susin, and R. Reis, "Analyzing area and performance penalty of protecting different digital modules with Hamming code and triple modular redundancy," in *Proceedings of the 15th symposium on Integrated circuits and systems design*, (Washington, DC, USA), pp. 95–, IEEE Computer Society, 2002.

[70] B. Shim, S. Sridhara, and N. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, pp. 497 –510, may 2004.

[71] M. O. Rabin, "Probabilistic automata," *Information and Control 6*, 1963.

[72] G. De Micheli, "Robust system design with uncertain information," in *Formal Methods and Models for Co-Design, 2003. MEMOCODE '03. Proceedings. First ACM and IEEE International Conference on*, p. 283, june 2003.

[73] M. Breuer, "Multi-media applications and imprecise computation," in *Digital System Design, 2005. Proceedings. 8th Euromicro Conference on*, pp. 2–7, 2005.

[74] C. S. Calude, "Algorithmic randomness, quantum physics, and incompleteness," in *Proceedings of the Conference "Machines, Computations and Universality" (MCU2004), Lectures Notes in Comput. Sci. 3354*, pp. 1–17, Springer, 2004.

[75] V. Wong and M. Horowitz, "Soft error resilience of probabilistic inference applications," in *IN PROCEEDINGS OF THE WORKSHOP ON SYSTEM EFFECTS OF LOGIC SOFT ERRORS*, 2006.

[76] E. M. M. V. K. Tenenbaum, Joshua B.; Jonas, "Stochastic digital circuits for probabilistic inference," tech. rep., Massachusetts Institute of Technology, November 2008.

[77] J. Margarida, C. He, G. de Veciana, and S. Bijansky, "Defect tolerant probabilistic design paradigm for nanotechnologies," in *Proceedings of the 41st annual Design Automation Conference*, DAC '04, (New York, NY, USA), pp. 596–601, ACM, 2004.

[78] L. N. B. Chakrapani and K. V. Palem, "A probabilistic Boolean logic and its meaning," tech. rep., Rice University, Department of Computer Science, June 2008.

[79] L. N. Chakrapani, P. Korkmaz, B. E. S. Akgul, and K. V. Palem, "Probabilistic system-on-a-chip architectures," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, pp. 29:1–29:28, May 2008.

[80] J. Bau, R. Hankins, Q. Jacobson, S. Mitra, B. Saha, and A. A. Tabatabai, "Error resilient system architecture (ERSA) for probabilistic applications," In The 3rd Workshop on System Effects of Logic Soft Errors (SELSE), 2007.

[81] L. Leem, H. Cho, J. Bau, Q. A. Jacobson, and S. Mitra, "Ersa: Error resilient system architecture for probabilistic applications," in *Design, Automation, and Test in Europe*, pp. 1560–1565, 2010.

[82] K.-I. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," vol. 20, no. 8, pp. 921–930, 2001.

[83] C. S. Bouganis, I. Pournara, and P. Y. K. Cheung, "Efficient mapping of dimensionality reduction designs onto heterogeneous FPGAs," in *Proc. 15th Annual IEEE Symp. Field-Programmable Custom Computing Machines FCCM 2007*, pp. 141–150, 2007.

[84] C.-S. Bouganis, S.-B. Park, G. A. Constantinides, and P. Y. K. Cheung, "Synthesis and optimization of 2D filter designs for heterogeneous FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 1, pp. 24:1–24:28, January 2009.

[85] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-6, pp. 721 –741, nov. 1984.

[86] J. Levine, E. Stott, G. Constantinides, and P. Cheung, "Smi: Slack measurement insertion for online timing monitoring in fpgas," in *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pp. 1–4, Sept 2013.

[87] Intel, "Hexadecimal object file format specification." Online, January 1988. `http://microsym.com/editor/assets/intelhex.pdf`.

[88] B. Fisher, "Cvonline: Image databases." online, 2014.

[89] Terasic Technologies, "Terasic DE0 board user manual v. 1.3," 2009.

[90] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters. I. Pipelining using scattered look-ahead and decomposition," vol. 37, no. 7, pp. 1099–1117, 1989.

[91] S. Ekvall and D. Kragic, "Receptive field cooccurrence histograms for object detection," in *In Proc. IEEE/RSJ International Conference Intelligent Robots and Systems, IROS'05*, pp. 84–89, 2005.

[92] M. H. Quenouille, "Approximate tests of correlation in time-series," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 11, no. 1, pp. pp. 68–84, 1949.

[93] J. W. Tukey, "Bias and confidence in not quite large samples (abstract)," *The Annals of Mathematical Statistics*, vol. 29, pp. 614–623, 06 1958.

[94] B. Efron, "Bootstrap methods: Another look at the jackknife," *The Annals of Statistics*, vol. 7, pp. 1–26, 01 1979.

[95] J. W. Osborne and A. B. Costello, "Sample size and subject to item ratio in principal components analysis." Online, 2004. `http://PAREonline.net/getvn.asp?v=9&n=11`.

[96] C.-Y. Chan and P. M. Goggans, "Using Bayesian inference for linear phase log FIR filter design," *AIP Conference Proceedings*, vol. 1193, no. 1, pp. 329–335, 2009.

[97] C.-Y. Chan and P. M. Goggans, "Using bayesian inference for the design of {FIR} filters with signed power-of-two coefficients," *Signal Processing*, vol. 92, no. 12, pp. 2866 – 2873, 2012.

[98] F. Glover, "Tabu search: A tutorial." Interfaces, 20(4), 74-94., 1990.

[99] P. Bessiere, E. Mazer, J. M. Ahuactzin, and K. Mekhnacha, *Bayesian Programming*. Chapman & Hall/CRC, 1st ed., 2013.

[100] D. E. Knuth, *The Art of Computer Programming, Seminumerical Algorithms*, vol. 2. Addison-Wesley, 1981.

[101] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: Systematic logic synthesis of approximate circuits," in *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, (New York, NY, USA), pp. 796–801, ACM, 2012.

[102] Aim & Thurlby Thandar Instruments, "The new pl-p series - advanced bus programmable dc power supplies." http://www.tti-test.com/products-tti/pdf-brochure/psu-npl-series-8p.pdf.

[103] "Arduino." Online. http://arduino.cc/.

[104] Lascar Electronics, "EL-USB-TC K, J, and T-type thermocouple temperature USB data logger." Online. `http://www.lascarelectronics.com/temperaturedatalogger.php?datalogger=364`.