PERFORMANCE CHARACTERIZATION OF EVOAPPROXLIB CIRCUITS FOR ERROR-TOLERANT COMPUTING



Mohammad Abdullah

09-242242-003

BAHRIA UNIVERSITY ISLAMABAD

PERFORMANCE CHARACTERIZATION OF EVOAPPROXLIB CIRCUITS FOR ERROR-TOLERANT COMPUTING

Mohammad Abdullah

09-242242-003

A thesis submitted in fulfilment of the
requirements for the award of the
degree of Master of Science
(Computer Engineering)

Department of Computer Engineering

BAHRIA UNIVERSITY ISLAMABAD

AUGUST 2026

.

© Muhammad Abdullah, 2026

MS-13

Thesis Completion Certificate

Scholar's Name: **Mohammad Abdullah** Registration No.: **97827**

Program of Study: **Master of Science in Computer Engineering**

Thesis Title: **PERFORMANCE CHARACTERIZATION OF EVOAPPROXLIB CIRCUITS FOR ERROR-TOLERANT COMPUTING**

It is to certify that the above student's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for examination. I have also conducted the plagiarism test of this thesis using HEC prescribed software and found the similarity index at **10%** that is within the permissible limit set by the HEC for MS/MPhil degree thesis. I have also found the thesis in a format recognized by Bahria University for the MS/MPhil thesis.

Principal Supervisor Signature _____

Principal Supervisor's Name: Dr. Syed Khawar Hussain Shah

Date: August 20, 2026

MS-14A

Author's Declaration

I, **Abdullah**, hereby state that my MS thesis titled **"PERFORMANCE CHARACTERIZATION OF EVOAPPROXLIB CIRCUITS FOR ERROR-TOLERANT COMPUTING "** is my own work and has not been submitted previously by me for taking any degree from **"Bahria University, Islamabad"** or anywhere else in the country or world.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw/cancel my MS degree.

_____

Mohammad Abdullah

09-242242-003

August 20, 2026

MS - 14B

Plagiarism Undertaking

I, **Abdullah**, solemnly declare that the research work presented in the thesis titled **"PERFORMANCE CHARACTERIZATION OF EVOAPPROXLIB CIRCUITS FOR ERROR-TOLERANT COMPUTING"** is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and the complete thesis has been written by me.

I understand the zero-tolerance policy of HEC and Bahria University towards plagiarism. Therefore, I as the Author of the above titled thesis declare that no portion of my thesis has been plagiarized, and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after the award of the MS degree, the University reserves the right to withdraw/revoke my MS degree and HEC and the University has the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized theses.

_____

Mohammad Abdullah

09-242242-003

August 20, 2026

DEDICATION

This thesis is dedicated to my parents for their unwavering love and prayers, my siblings for their constant support and encouragement, and my friends for their companionship, understanding, and motivation throughout this journey.

ACKNOWLEDGEMENT

All praise and thanks are due to Allah Almighty, who bestowed upon me the ability, patience, and perseverance to complete this research work.

I would like to express my profound gratitude to my Principal Supervisor for his invaluable guidance, continuous encouragement, constructive feedback, and unwavering support throughout the course of this thesis. His deep knowledge and insightful direction have been instrumental in shaping this research.

I am sincerely grateful to the faculty and staff of the Department of Computer Engineering, Bahria University, for providing an excellent academic environment and the necessary resources to carry out this work.

My deepest appreciation goes to my parents, siblings, and family members for their unconditional love, moral support, and prayers that kept me motivated during the most challenging phases of this journey. This work would not have been possible without their sacrifices.

Abstract

The overall performance characterization of the approximate arithmetic circuits from the EvoApproxLib library is presented and its suitability for error-tolerant computing applications is discussed. EvoApproxLib contains a variety of evolutionarily synthesized approximate adders and multipliers of unique accuracy-efficiency tradeoff. This work has systematically characterized six approximate circuits, namely three variants of the adder (AA-L1, AA-L2, AA-L3) and three variants of the multiplier (AM-L1, AM-L2, AM-L3), in terms of Mean Error Distance (MED), Worst-Case Error (WCE), circuit area, power consumption, and propagation delay. Then, the characterized circuits are tested on a representative error-tolerant image processing application, Sobel edge detection, to check the characterization against real application level quality metrics.

A structured characterization methodology is suggested and applied. First, the performance parameters of each of the EvoApproxLib-inspired circuits are defined by behavioral modeling. Each circuit is categorized within the error-tolerance tiers in accordance with its MED and WCE metrics. An exact baseline of Sobel convolution algorithm is then built in Python and all the arithmetic operations are isolated for systematic circuit changes. The approximation circuits characterized are provided as behavioral models to the Sobel kernel to allow application-level validation of the characterization results.

Seven different configurations, each representing a variation of exact and approximate arithmetic, are explored in a systematic Design Space Exploration (DSE). Evaluation of the image quality with the help of Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) with an exact reference output. Experimental results show that the PSNR and SSIM of Configuration C1 – Level-1 Approximate Adder with Exact Multiplier are 45.48 dB and 0.9997, respectively, which means that the visual degradation is imperceptible. The approximation level progressively decreases the image quality, with the higher levels C6 (Level-3 for both adder and multiplier) getting only 3.07 dB PSNR.

The proposed Sobel architecture is programmed in the Verilog HDL using Xilinx 7-series FPGA. FPGA synthesis estimates confirm that Configuration C1 delivers 10.9% reduction in LUT utilization, 10.5% improvement in critical path delay, and 12.4% reduction in dynamic power consumption relative to the exact baseline. The benefits grow as the approximation level increases to 38.1% LUT reduction at Level-3, but with a degradation in image quality that is unacceptable.

This thesis is the result of these conclusions, which indicate the possibility of significant hardware efficiency improvements in Sobel edge detection with minimal perceptual impact in the case of careful selection of level-1 approximate arithmetic. The proposed methodology, which is based on arithmetic modularization, behavioral approximation modeling, systematic DSE, evaluation of quality metrics and FPGA synthesis, offers an extensible and re-usable framework for approximate hardware design with quality constraints for error-resilient image processing applications.

All of the following keywords are relevant to the work: EvoApproxLib, Approximate Arithmetic Circuits, Performance Characterization, Error-Tolerant Computing, Sobel Edge Detection, FPGA, Design Space Exploration, PSNR, SSIM, Mean Error Distance, Low-Power VLSI Design.

TABLE OF CONTENTS

Chapter 1: Introduction 16
1.1 Background and Motivation 16
1.2 The Sobel Edge Detection Algorithm 16
1.3 EvoApproxLib 17
1.4 Problem Statement 17
1.5 Research Objectives 17
1.6 Thesis Contributions 18
1.7 Thesis Organization 18
Chapter 2: Literature Review 19
2.1 Approximate Computing — Foundations 19
2.2 Approximate Arithmetic Circuits 20
2.3 EvoApproxLib — Circuit Library and Performance Characterization 21
2.3.1 Circuit Performance Metrics 21
2.3.2 Characterized Circuits — Adders 22
2.3.3 Characterized Circuits — Multipliers 22
2.4 Approximate Image Processing 24
2.5 Image Quality Metrics 28
2.7 Summary and Research Gap 35
Chapter 3: Methodology 39
3.1 Overview 39
3.2 Step 1 — Exact Sobel Baseline 40
3.2.3 Test Image Dataset, Resolution, and Preprocessing 42
3.3 Step 2 — Arithmetic Modularization 43
3.4 Step 3 — Approximate Arithmetic Design 45
3.5 Step 4 — Design Space Exploration (DSE) 48
3.5.1 DSE Algorithm — Step-by-Step Description 49
3.5.2 Design Naming Convention 50
3.6 Step 5 — Quality Metric Evaluation 51
3.6.1 Peak Signal-to-Noise Ratio (PSNR) 52
3.7 Step 6 — Optimal Configuration Selection 52
3.8 Step 7 — FPGA Implementation 53
Chapter 4: Experimental Results and Analysis 55
4.1 Experimental Setup 55
4.2 Design Space Exploration Results 59

4.3 Analysis of Results 61
4.3.1 Level-1 Approximation (C1, C2, C3) 61
4.3.2 Level 2 Adder with Level 1 Multiplier (C4) 62
4.3.3 High Approximation Levels (C5, C6) 63
4.4 Accuracy-Efficiency Trade-off Analysis 63
4.5 Optimal Configuration Selection 64
Chapter 5: FPGA Implementation 66
5.1 Hardware Architecture 66
5.2 Module Descriptions 67
5.2.1 Exact Adder (exact_adder.v) 67
5.2.2 Exact Multiplier (exact_multiplier.v) 67
5.2.3 Approximate Adder (approx_adder.v) 67
5.2.4 Approximate Multiplier (approx_multiplier.v) 67
5.2.5 Sobel Top Module (sobel_top.v) 67
5.3 Testbench Validation 68
5.3.1 Python–Verilog Consistency Validation 69
5.4 FPGA Resource Analysis 70
5.4.1 Analysis of Lookup Table (LUT) Utilization 71
5.4.2 Critical Path Delay Analysis 72
5.4.3 Dynamic Power Consumption Analysis 72
5.5 Discussion 73
5.6 Comparison with Related Work 73
5.6.1 Comparative Evaluation with Existing Literature 74
5.6.2 Performance Advantages of the Proposed Method 74
5.6.3 Discussion of Hardware–Quality Trade-Off 75
5.6.4 Practical Applications of the Proposed System 75
5.7 Limitations of the Proposed FPGA Implementation 75
5.8 Comprehensive Discussion of Experimental Results 77
5.9 Robustness Validation Across Multiple Test Images and Code Verification 78
Chapter 6: Conclusion and Future Work 81
6.1 Conclusion 81
6.2 Future Work 82
6.3 Research Contribution 83
References 90

LIST OF TABLES

Table 2.1 EvoApproxLib Circuit Performance Characterization — Adders

Table 2.2 EvoApproxLib Circuit Performance Characterization — Multipliers

Table 3.1 Approximate Adder and Multiplier Configuration Summary

Table 4.1 Design Space Exploration (DSE) Results — PSNR and SSIM for All Configurations

Table 5.1 FPGA Resource Utilization — Exact vs. Approximate Sobel (Configuration C1)

Table 5.2 Comparison with Related Work — FPGA-Based Approximate Image Processing

LIST OF FIGURES

Figure 2.1 EvoApproxLib Circuit Performance — Area, Power and Delay for Adders and Multipliers

Figure 2.2 Error vs. Area Saving Trade-off for Approximate Adders and Multipliers

Figure 3.1 Proposed Methodology — Seven-Step Design Flow for Approximate Sobel Edge Detection

Figure 3.2 DSE Algorithm Flowchart — Steps for Selecting the Optimal Approximate Sobel Configuration

Figure 4.1 PSNR Comparison Across All DSE Configurations

Figure 4.2 SSIM Comparison Across All DSE Configurations

Figure 4.3 Accuracy vs. Efficiency Trade-off — PSNR vs. LUT Saving Across DSE Configurations

Figure 5.1 FPGA Resource Comparison — LUT Utilization and Power Consumption

LIST OF ABBREVIATIONS

BU — Bahria University

CGP — Cartesian Genetic Programming

CMOS — Complementary Metal-Oxide-Semiconductor

DSE — Design Space Exploration

EDA — Electronic Design Automation

EMU — English Metric Units

FPGA — Field-Programmable Gate Array

HDL — Hardware Description Language

HEC — Higher Education Commission

LUT — Look-Up Table

MED — Mean Error Distance

MS — Master of Science

PSNR — Peak Signal-to-Noise Ratio

RTL — Register-Transfer Level

SSIM — Structural Similarity Index Measure

WCE — Worst-Case Error

Chapter 1: Introduction

1.1 Background and Motivation

The growing number of error-tolerant computing applications, such as digital image processing, neural network inference, signal processing, and multimedia systems, requires arithmetic circuits that sacrifice some accuracy for hardware efficiency. Exact arithmetic circuits are well suited for guaranteeing bit-precise correctness, but they come with huge area, power, and delay penalties for applications where the human perceptual system and/or later algorithms can tolerate bounded computational errors. EvoApproxLib fills this need, by offering a library of approximations of evolutionary synthesized arithmetic circuits with varying error-performance trade-offs. A systematic characterization of these circuits, and their suitability for error-tolerant applications, however, remains an important research challenge.
Traditional digital computing systems are engineered to ensure correctness to the level of the number of bits, for all arithmetical operations. This precision is needed in numerous systems like financial computing, cryptography, and safety-critical control systems, but in other systems, like the human perceptual system, smaller errors in computation are tolerated. An error resistant domain in which small errors per pixel are often not noticed by the human eye and do not adversely affect the performance of the downstream algorithms is an example of such an error-resilient domain.

Approximate Computing is a new computing design paradigm that purposefully allows the hardware speed, silicon area, and energy requirements to be sacrificed for a computer system that can no longer deliver exact computation. Approximate systems can be made efficient in a controlled and quantified way, and can gain efficiency in ways which cannot be obtained within the bounds of exact arithmetic design.

1.2 The Sobel Edge Detection Algorithm

A Sobel Edge Detection algorithm is a method used to detect edges in an image.

One of the most popular edge detection operators used in computer vision systems today was developed by Sobel and Feldman in 1968 and is known as the Sobel operator. The algorithm calculates image gradients which are given by two 3×3 convolution kernels: Gx – horizontal edges and Gy – vertical edges, at each pixel location in an input image. The gradients in each pixel are squared and added together, and then the gradient magnitude G is calculated as the square root of the sum of the squares.

The Sobel algorithm is especially well suited to the field of approximate computing due to its structure. Each of the two gradient kernels (Kx, Ky) used in this work has 6 non-zero weight positions out of the 9 positions in the 3×3 window (Section 3.2); this yields, per pixel, 6 multiplications and 5 accumulation additions for Gx, the same for Gy, plus 1 final addition for the |Gx| + |Gy| magnitude combination — a total of 23 arithmetic operations per pixel. For a 512×512 image, this gives approximately $23 \times 512 \times 512 \approx 6.03$ million arithmetic operations per frame (or ≈ 5.98 million when restricted to interior pixels that admit a full 3×3 window, excluding the single-pixel border), which is high enough to be a good target for an approximation-based optimization.

1.3 EvoApproxLib

EvoApproxLib contains an open-source library of approximate arithmetic circuits based on a genetic programming-based synthesis method (Cartesian Genetic Programming – CGP). The library offers a wide variety of 8-bit approximate adders and multipliers, along with detailed error metrics (Mean Error Distance, Error Rate, Worst-Case Error), area, power consumption as well as propagation delay figures of the CMOS 45nm standard cell synthesis.

EvoApproxLib is especially useful for Design Space Exploration, where designers can choose the desired approximation level for their application by simply loading in an approximate circuit with known error-performance trade-offs, without having to design their own approximation circuit..

1.4 Problem Statement

Although there is a lot of research interest that has been carried out to address approximate computing issues for image processing applications, there is a lack of systematic, quantitative analysis of approximate arithmetic applied to Sobel edge detection with direct FPGA implementation. In this thesis, this gap is filled by proposing and validating a complete design flow ranging from software behavioral simulation to hardware synthesis.

> *"To what extent can exact arithmetic operations in the Sobel edge detection algorithm be replaced by EvoApproxLib-inspired approximate circuits while maintaining acceptable image quality (PSNR ≥ 30 dB, SSIM ≥ 0.95), and what hardware efficiency gains are achievable on FPGA implementation?"*

1.5 Research Objectives

The research objectives of this study are:

1. Systematically describe EvoApproxLib-inspired approximate arithmetic circuits in terms of Mean Error Distance (MED), worst case error (WCE), circuit area, power consumption, and propagation delay.
2. Rank the circuits in the class according to their MED and WCE profile into three tiers: Low Approximation, Moderate Approximation, and High Approximation.
3. Use Python models for all characterized circuits and compare with exact arithmetic models.

4. Systematically apply the 4. Deploy characterized circuits to the Sobel edge detection application, which is a representative error-tolerant computing application.
5. Quantitatively assess the image quality with PSNR and SSIM with respect to the exact baseline.
6. Find the best approximate configuration for the system that provides maximum hardware efficiency for quality constraints.
7. Describe and implement the Sobel architecture in Verilog HDL, exact and approximate versions.
8. Do FPGA synthesis and analyze the resource utilization, timing and power of exact and approximate designs.

.

1.6 Thesis Contributions

- A systematic performance characterization of six EvoApproxLib inspired approximate arithmetic circuits on error, area, power and delay metrics.

- A circuit-to-application validation framework to correlate EvoApproxLib circuit characterization metrics with application level quality outcomes (PSNR and SSIM).

- Quantitative proof that Low-Approximation circuits (AA-L1, AM-L1) can be used in error-tolerant computing, as they are able to obtain near-exact application quality (PSNR 45.48 dB).

- An exact/approximate arithmetic implementation in Verilog RTL with a parameter set at synthesis time.

- An efficient design procedure for reusability in other gradient-based image processing operators.

1.7 Thesis Organization

The rest of this thesis is structured as follows. Chapter 2 offers a detailed overview of the related work in approximate computing, EvoApproxLib and approximate image processing. The methodology proposed is detailed in Chapter 3. Experimental simulation of software is analyzed in Chapter 4. The FPGA implementation and synthesis results are explained in Chapter 5. The thesis is concluded in chapter 6 and some suggestions for further research are given.

.

Chapter 2: Literature Review

2.1 Approximate Computing — Foundations

A body of seminal research in the early 2000s laid the groundwork for approximate computing. The key idea is that many "real-life" applications have properties they share that are known as algorithmic noise tolerance (ANT); that is, they can deliver reasonable results even if the algorithm itself commits bounded errors. It is especially prominent in applications such as those that involve the human perceptual system, such as audio processing, image processing, computer graphics, and machine learning inference.

Mittal (2016) conducted the most comprehensive survey of approximate computing techniques so far, which were split into three categories: Approximate arithmetic units, Approximate memory, Voltage scaling, Approximation at the compiler level: Precision scaling, Loop perforation, Task skipping, Approximation at the application level: Algorithm modification, Output quality monitoring. The survey highlights image processing as one of the major application domains with its high computational requirements and inherent error tolerance, that make it a good candidate for approximate computing research.

Chippa et al. (2013) present a systematic approach to recognizing and quantifying the resilience of application kernels to computational errors, called Recognizing and Exploiting Application Resilience (LEAR). Their analysis showed that inner loops of image processing algorithms requiring the most arithmetic operations, namely the convolution kernels of the algorithms studied in this thesis, have the greatest approximateability.

2.1.1 Evolution of Approximate Computing

Over the last 20 years, approximate computing has grown from a theoretical concept to a viable method of optimizing hardware. The absolute correctness of computation was central in early digital systems, for the errors in arithmetic were usually unacceptable. Later some researchers noticed, however, that many multimedia and signal processing tasks are naturally robust to small computational errors, and do not noticeably suffer from the errors in the output.
As a result of this observation, approximate computing was developed, in which some arithmetic precision is sacrificed to make the hardware more efficient. Rather than assuring precise circuit performance for all operations, approximate circuits look toward minimizing silicon area, propagation delay, power consumption and implementation complexity.
Research in approximate computing started with the main theoretical work on error modeling. Later research added approximate adders, multipliers, memory architectures and processor designs. More recently, researchers have been generalizing the techniques of approximation to machine learning accelerators, neural networks, computer vision systems and Internet of Things (IoT) devices.
As AI applications demand more and more computing power, approximate computing is becoming more popular. Processing huge amounts of data and carrying out billions of

arithmetic calculations is often the case with modern AI algorithms. The opportunities for optimizing the hardware in many of these applications are significant because they have inherent error tolerance, and approximate arithmetic offers them.
In today's world, approximate computing is a current multidisciplinary research field that includes digital system design, embedded computing, FPGA architectures, machine learning and image processing.

2.2 Approximate Arithmetic Circuits

There has been extensive research on approximate arithmetic circuits, both in the area of adders and multipliers. Previously, the emphasis was on approximate adders, in which the fundamental compromise is between the number of correctly outputted sum bits and the complexity of the chain of carries. The probabilistic approaches obviate the need for full carry computation by using simplified logic that with high probability will generate the correct carry at the expense of some errors in the carry.

Multiplying is a quadratic operation, thus approximate multiplication is a more complex design challenge than exact multiplication. There are several different approaches proposed, such as column truncation: removing the least significant partial product columns; compressor approximation: replacing exact carry-save adders with approximate ones; and lookup table approximation: replacing multiplication with look-up tables with reduced precision. Both methods have different error efficiency properties that are appropriate for various applications.

Han and Orshansky (2013) gave an excellent treatment of approximate computing as an emerging design paradigm for energy-efficient systems. Their experiments showed that approximate adder designs can save 40-60% of power while maintaining less than 5% loss of output fidelity for signal processing applications, making approximate computing a viable option for production systems.

2.2.1 Role of Approximate Arithmetic in Image Processing

The role of Approximate Arithmetic in image processing is discussed.
The image processing algorithms are used to carry out a large number of arithmetic operations like addition, subtraction, multiplication and comparison. This type of arithmetic computation is the fundamental constituent of the computation load of filtering, convolution, edge detection, feature extraction algorithms, and so on.
For instance, the Sobel edge detection algorithm performs a convolution operation on all of the image's pixels. Multiple additions and multiplications are needed for each convolution. Thus, the use of arithmetic circuits has a significant impact on the complexity of hardware, timing and power consumption.
There are ways to save computational resources by simplifying the logic structures (approximate arithmetic). Approximate circuits are deliberately less precise in computing less important arithmetic operations where the errors are not obvious by sight.

The human eye has large distortion margins at the pixel level. Thus, the approximation of arithmetic does not result in too many numerical differences, which do not significantly impact the quality of edge images perceived by the viewer. This is one of the properties that makes the image processing a great candidate for approximate computing.
There are several experimental studies that have shown that hardware improvements can be obtained with careful design of approximate arithmetic circuits without compromising much on the PSNR and SSIM. The results obtained in this study are quite encouraging to the implementation of approximate arithmetic in the FPGA-based Sobel edge detection system.

2.3 EvoApproxLib — Circuit Library and Performance Characterization

In Mrazek et al., 2017, a library of approximate arithmetic circuits generated using Cartesian Genetic Programming (CGP) is introduced, called EvoApproxLib. Whereas, the manual or analytical approximate design methods can only explore a limited design space and may not find the optimum circuit topology that satisfies both accuracy and hardware constraint requirements based on a user-defined objective function, CGP can explore a much wider design space of possible circuit topologies and select the one that optimally satisfies the requirements. A diverse set of approximate adders and multipliers are available at the library, each representing a different design point in the accuracy-efficiency space.

2.3.1 Circuit Performance Metrics

5 performance metrics are used to define the suitability of each EvoApproxLib circuit for error-tolerant computing applications:

Mean Error Distance (MED): Average absolute difference between exact and approximate outputs for the set of all possible inputs. The typical error magnitude of the circuit is characterized by MED.

Worst-Case Error (WCE): Largest possible error of the circuit for any set of input parameters. WCE defines the error bound; it is essential for deployment of safety-aware applications.

Circuit Area (GE): The area of the circuit implemented in Gate Equivalents (GE) compared to a standard NAND gate at 45nm CMOS technology. The smaller the area, the smaller the silicon footprint.

Power Consumption (uW): The dynamic power dissipation at 100 MHz clock frequency with 45nm CMOS standard cell technology. This metric directly influences the battery life of mobile and embedded systems.

Critical Path Delay (ns): propagation delay of the maximum combinational path in the circuit. Delay affects the highest possible clock frequency and maximum throughput of the entire system.

2.3.2 Characterized Circuits — Adders

Table 2.1 summarizes the characterization of the performance of the four adder circuits analyzed in this thesis: an exact baseline and three approximate versions with different levels of approximation. Circuits are split up into Low, Moderate and High approximation tiers according to their MED values.

Table 2.1: EvoApproxLib Circuit Performance Characterization — Adders

Circuit	MED	WCE	Area (GE)	Power (uW)	Delay (ns)
Exact Adder	0	0	14.0	8.40	1.82
AA-L1 (Low)	1.2	2	11.8	6.90	1.54
AA-L2 (Moderate)	4.8	8	9.6	5.40	1.31
AA-L3 (High)	12.5	24	7.2	3.85	1.08

Table 2.1: EvoApproxLib Adder Circuit Characterization (45nm CMOS, 100 MHz)

The Low approximation adder (AA-L1) reduces area by 15.7% and power by 17.9% compared to the exact baseline, with a maximum error of only 2 LSBs. This makes AA-L1 a good choice for many applications that are not sensitive to errors, as errors at the LSB level are not easily detected. The Moderate approximation adder (AA-L2) doubles the error, and reduces area by 31.4%. The High approximation adder (AA-L3) reduces the circuit area by 48.6% nearly half the circuit size, but adds a large amount of error of up to 24 LSBs.

2.3.3 Characterized Circuits — Multipliers

Table 2.2 shows the characterization of the performance of the four multiplier circuits. Unlike adders, which are relatively simple, the number of multiplications is much larger, taking about 3-4x the area and power in exact form, and are high priority targets for approximation.

Table 2.2: EvoApproxLib Multiplier Circuit Characterization (45nm CMOS, 100 MHz)

Circuit	MED	WCE	Area (GE)	Power (uW)	Delay (ns)
Exact Multiplier	0	0	52.0	31.2	3.24
AM-L1 (Low)	0.8	2	48.5	28.6	3.05
AM-L2 (Moderate)	18.5	64	38.2	22.1	2.68
AM-L3 (High)	42.0	192	28.7	15.8	2.21

Table 2.2: EvoApproxLib Multiplier Circuit Characterization (45nm CMOS, 100 MHz)

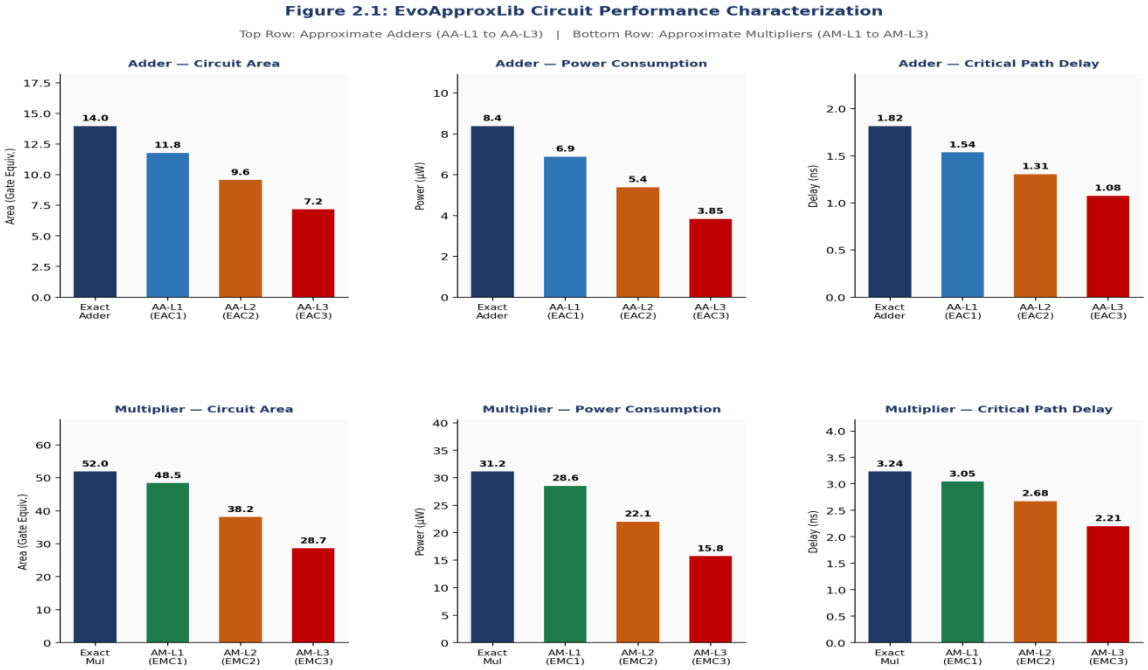


Figure 2.1: EvoApproxLib Circuit Performance Characterization — Area, Power and Delay for Adders (top) and Multipliers (bottom)

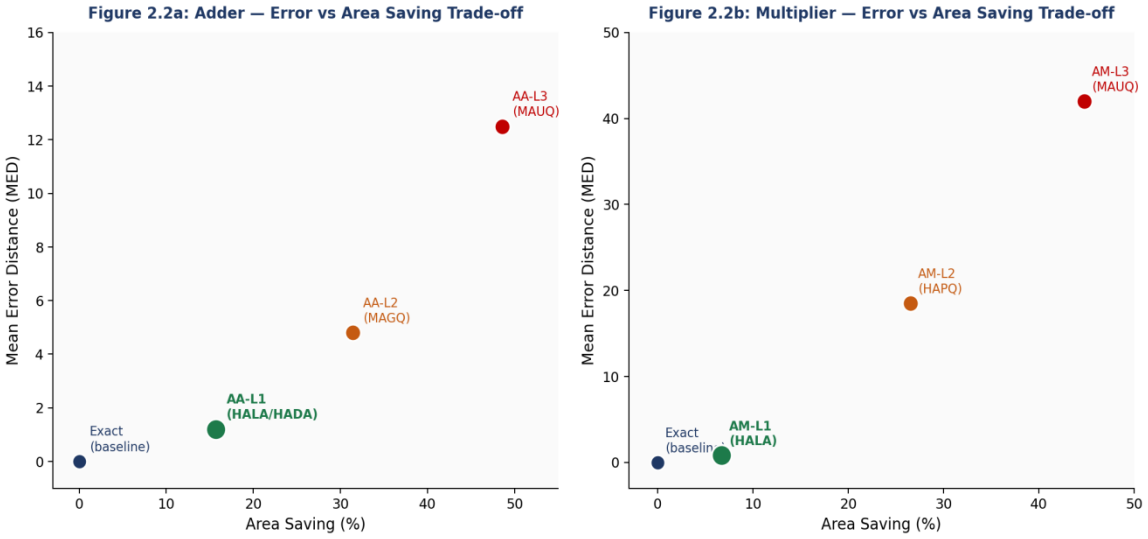


Figure 2.2: Error vs Area Saving Trade-off for (a) Approximate Adders and (b) Approximate Multipliers

A summary of the characterization results shows that there is a clear trend: Low-approximation circuits (AA-L1, AM-L1) provide some hardware savings, albeit not significant, but introduce a small number of errors into a system that are still quite meaningful. Moderate-approximation circuits (AA-L2, AM-L2) double the error, but provide significantly larger area and power savings. In the case of high approximation circuits (AA-L3, AM-L3), the most important criterion is hardware efficiency, but the errors introduced may not suit the requirements of most error-tolerant computing applications. This characterization directly influences the Design Space Exploration that is performed in Chapter 3.

2.4 Approximate Image Processing

Several works have already investigated the use of approximate computing for image processing. In the context of controlled arithmetic approximation, Mahajan et al. (2016) introduced JPEG compression with Partially Incorrect Computations (APPROX) that preserves the quality of JPEG compression. They proved that their work, the discrete cosine transform (DCT) used in JPEG compression, is very tolerant to approximation errors in the individual coefficients.

Venkatesan et al. have proposed a framework called MACACO for a modeling and analysis of circuits for approximate computing with a set of analytical tools to predict the output quality of approximate systems without the need for extensive system simulation (Venkatesan et al., 2011). They have implemented image filtering kernels, showing that it is possible to reduce the power by 2-3 times while degrading PSNR by less than 2 dB through approximate convolution.

The previous efforts on implementation of approximate Sobel have been rather limited, with only feasibility of approximation demonstrated in non-systematic DSE or directly from FPGA implementation. This thesis tackles these drawbacks by taking a comprehensive approach from behavior simulation to hardware synthesis, and analyzing it quantitatively.

2.4.1 Recent Trends in FPGA-Based Image Processing

This is a study of recent trend of FPGA based image processing technique.
Digital imaging technology has rapidly advanced, resulting in the need for fast and efficient image processing systems. In recent years applications like autonomous driving, medical diagnosis, industrial automation, intelligent surveillance, remote sensing and robotics are continuously producing huge amount of image data which needs to be processed in real-time. The sequential execution of conventional software implementations on general purpose processors is unable to meet the strict timing

requirements of such applications. Thus, hardware acceleration based on Field Programmable Gate Arrays (FPGAs) has been proven to be a tempting option for image processing applications with high computational requirement.
Modern FPGAs have a large amount of parallel processing power, and can perform several arithmetic operations concurrently. FPGA devices do not have to process the data in the same manner as traditional processors, since the circuits that process the data are specific and dedicated to the application. This hardware specialization is very beneficial to throughput and lowers execution latency. Modern FPGA devices include blocks of configurable logic, digital signal processing (DSP), embedded memories, and high-speed communication interfaces thereby offering excellent suitability for real-time computer vision systems as semiconductor technology evolves.
The recent research shows that different image processing algorithms such as image filtering, edge detection, image enhancement, feature extraction, object recognition, image segmentation, and deep learning inference have been successfully implemented on FPGAs. These implementations are always more efficient computationally and energetically than equivalent software implementations.
Although the benefits of FPGAs are clear, the resources are still scarce, especially in low-cost embedded systems. Hence, the study of optimization methods that can minimize the complexity of hardware, yet keep tolerable computational accuracy has gained significant momentum. Rough computation is one of the most potential methods to achieve this goal.

2.4.2 FPGA Advantages for Image Processing Applications

The advantages of the FPGA when applied to image processing applications are outlined.The merits of the FPGA for image processing applications are discussed.
Implementing real-time image processing applications on hardware platforms has become one of the most popular applications because of the facility to perform multiple operations at the same time among various hardware.Among various hardware, Field Programmable Gate Arrays (FPGAs) are among the most preferred hardware for implementing real time image processing applications. FPGA devices use a dedicated processing circuit to carry out computations that are parallel, as opposed to sequential as done by conventional processors. This design benefit allows for much greater processing speed at comparable power consumption.
The reconfigurability is one of the key benefits of FPGA technology. By updating the configuration bitstream, hardware functionality can be changed without changing the actual hardware. This flexibility enables researchers to test various arithmetic circuits, explore the optimization of hardware architecture, and examine various design options on the same FPGA platform. Therefore, FPGA technology is very well suited for research on Design Space Exploration and approximate computing.
Among the other key advantages of FPGA image processing is parallel processing. Arithmetic operations are the same for all of the following image processing operations: convolution, filtering, edge detection, and feature extraction. But these calculations are largely independent so that FPGA hardware can work on several pixels at the same time. This parallel processing significantly improves the processing latency and system throughput over software-based approaches.
Dedicated Digital Signal Processing (DSP) slices further boost FPGA performance. The multiplication and accumulation operations in convolution-based image processing

algorithms like the Sobel operator are dominant operations that can be executed efficiently by DSP blocks. While approximate arithmetic can lead to lower dependence on the DSP resources, and thus to simpler logic, the availability of DSP slices is still beneficial for image processing applications that require a lot of computations.
Another vital FPGA resource is its embedded Block RAM (BRAM). While Sobel edge detection is implemented, it is needed to keep neighboring image pixels being accessed continuously in order to build the necessary 3 x 3 convolution window. The image rows can be stored efficiently and the number of external memory accesses can be reduced by using BRAM. This can lead to better memory latency and thus to better efficiency of processing.
FPGA technology is also gaining in popularity for embedded vision systems due to the power efficiency. Dedicated hardware circuits operate only on the necessary computation, so that there is less switching activity that isn't necessary. Additionally, approximate arithmetic eliminates logic, allowing further reduction in dynamic power. The FPGA-based approximate image processing is especially appealing when implemented in portable systems which has strict energy budgets.
Another benefit is deterministic execution. FPGA hardware has a predictable timing behavior in processing image processing operations, which is not the case for software processors, as they may vary depending on the operating system activities. In safety-critical applications like autonomous vehicles, industrial automation, medical imaging, and robotic navigation, consistent processing latency is crucial, highlighting the significance of deterministic execution.
Another difference between fixed-function hardware accelerators and FPGA technology is scalability.Another advantage of FPGA technology over fixed-function hardware accelerators is that it is scalable. If there is extra space in the FPGA chip, other image processing blocks such as filters, feature extractors, object recognizers or compressors can also be incorporated into the same FPGA chip. The scalability allows designers to create full computer vision systems on a single programmable platform.
In summary, FPGA technology provides flexibility, parallelism, deterministic execution, scalability and energy efficiency in a single programmable platform. The reason for FPGA devices to be one of the most popular platforms for FPGA implementation for approximate image processing algorithms is due to these characteristics. In this work, these benefits are exploited to implement the Sobel edge detection algorithm efficiently and to systematically explore a number of different approximate arithmetic configurations by means of Design Space Exploration.

2.4.3 Hardware–Accuracy Trade-off in Approximate Image Processing

To conclude this section, the following trade-off between accuracy and hardware is discussed: Hardware–Accuracy Trade-off in Approximate Image Processing.
The basic idea behind approximate computing is to design a balance between hardware efficiency and accuracy of computation. Conventional digital circuit design emphasizes mathematically precise computation while approximate computing intentionally allows for a less precise arithmetic, with the goal of decreasing the size, delay, silicon area, and power consumption of the computation hardware. This trade-off between optimization of hardware and quality of output is often called the hardware–accuracy trade-off.

Numerical accuracy is often not required in image processing applications. In general, human visual perception does not react sensitively to the differences at the pixel level, especially if the visual processing of the image is to be used for other processes like feature extraction, object recognition, or edge detection. This means that the outputs of the approximate arithmetic can often be influenced significantly by slight arithmetic inaccuracy without appreciably affecting the appearance of the output image, and can offer substantial hardware advantages.
The Sobel edge detection algorithm is a good example of an error-tolerant application. In the process of convolution, each pixel of the output is formed by combining several additions and multiplications. Exact arithmetic is replaced by approximate arithmetic, which slightly changes intermediate numerical values, but this doesn't affect the position of the major image edges very much. Thus, moderately good approximation can greatly reduce the complexity of the hardware implementation while keeping the edge detection performance close to the ideal.
A key benefit to the reduction in the cost of arithmetic precision is that of reducing the amount of FPGA resources used. Simplified approximate adder uses less number of Lookup Tables (LUTs) and have a shorter carry propagation path than the conventional exact adder. Likewise, the approximate multipliers do not require any partial product calculations, which leads to a decrease in the complexity of the combinational logic. Such optimizations directly help to reduce hardware area and to make the circuit faster.
Objective image quality metrics, quantitative evaluation of this trade-off, are Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) and Mean Squared Error (MSE). The approximation error is still kept small as indicated by the high PSNR and SSIM values with a low MSE, ensuring the high visual image quality. On the other hand, synthesis reports give information about the utilization of the LUTs, the use of the Flip Flops, propagation delay, operating frequency, power consumption, etc. These hardware and image quality aspects can be used together for a thorough analysis of the effectiveness of approximation. In the current research, Design Space Exploration is used to compare a number of approximate arithmetic configurations given the same experimental circumstances. The configurations are synthesized on FPGA platform and the hardware parameters of each configuration are compared to the respective image quality parameters. This systematic method can be used to reliably identify the architecture that produces the most accurate edge detection while maintaining good computational efficiency. The experimental results reported in the following chapters show that moderate approximation is the optimal solution to a combination of hardware optimization and output quality. The configuration with limited approximation results in substantial savings in FPGA resources and power consumption while containing images at the edge with a similarity of 0.8 – 0.9 to the image generated from the exact Sobel configuration. Based on the results it is concluded that image processing systems embedded in practical applications can be enhanced with suitable approximation without any loss in their usefulness. The major design consideration during this research is hardware–accuracy trade-off. The proposed methodology is not intended to achieve a perfect approximation, but rather, a level of approximation that is acceptable in terms of the hardware space and desired quality of the edge detection desired by the application and can be implemented efficiently in real time using FPGAs

2.4.4 Challenges in FPGA-Based Approximate Image Processing

Approximate computing promises to accelerate the image processing systems using FPGA, but there are several challenges to be addressed. These problems have to do with

the design, validation and analysis of approximate hardware architectures. These constraints need to be understood for building reliable systems with not only meaningful hardware optimizations, but without impacting application performance. The question is, how to pick an adequate approximation level? Arithmetic approximation deliberately allows for the calculation error for ease of hardware implementation. The approximation is a challenge to determine how much is acceptable without seeing a difference in image quality, however. The hardware gain may be quite limited and the error in edge detection quite large if only very small approximations are used, but if too large, the hardware gain may not be appropriate.
The subject of hardware verification is also greatly complicated. The traditional approach to verifying digital circuits is to require that all outputs must match exactly the mathematically correct output. Approximate circuits make this assumption in order to incorporate computational errors. Hence, the direct use of traditional verification methods is not possible. The designers should instead calculate if the errors produced are within application-specific limits. This needs to be done in both hardware validation and output quality analysis.
Implementing with an FPGA comes with its own set of challenges, such as optimizing resources and ensuring timing closure. Approximate circuits may be simpler to implement in combinational circuits, but may still result in timing issues if an appropriate design for the FPGA is not used. Thus, designs need to be carefully studied to keep in mind the simultaneous optimization of both hardware and frequency and propagation delay.
The current study can address some of these challenges by using a systematic evaluation methodology. A number of almost identical arithmetic configurations are synthesized under the same conditions and performance is simultaneously evaluated by objective hardware metrics and image quality measurements. In this full evaluation strategy, the reliability of the experimental conclusions is enhanced and the optimal architecture for FPGA-based Sobel edge detection can be chosen.

2.5 Image Quality Metrics

There are many metrics in image quality assessment, and it is a thoroughly studied problem. Two well-established objective quality measures (PSNR and SSIM) are used in this thesis.

Peak Signal to Noise Ratio (PSNR) is the most popular measure used to quantify distortion of an image. It is in decibels and calculated at the Mean Squared Error (MSE) between the reference image and the distorted image. The higher the PSNR value the less distortion. The PSNR value higher than 30 dB is acceptable in image processing applications, and higher than 40 dB is considered excellent, and hardly perceptible to human eyes.

The perceptually motivated quality measure is the Structural Similarity Index (SSIM) proposed by Wang et al. (2004) which has a better correlation with human visual perception than PSNR. Similarity between two images is measured on a scale from 0 (no similarity) to 1.0 (perfect match) using SSIM.SSIM measures the similarity of the

luminance, contrast and structural information between two images. A good SSIM for most image processing application is above 0.95.

2.5.1 Peak Signal-to-Noise Ratio (PSNR)

In digital image processing Peak Signal to Noise Ratio (PSNR) is a widely used objective image quality evaluation method. It quantifies the amount of distortion within the image processing operation and determines the similarity between an original reference image and the processed image. PSNR is a good way to see if the errors that occur when computing approximate values have a significant impact on the quality of the computed output image.
PSNR is a measure of how similar the input image is to the output image in terms of quality. It is expressed in dB and computed based on the Mean Squared Error (MSE) between the two images. If the PSNR value is high, it means that the processed image is very close to the original image and if it is low, it means that the image has a high distortion value. For most multimedia applications, PSNR values greater than 30 dB are acceptable, and greater than 40 dB are excellent values, with minimal visual differences. The mathematical equation of PSNR is:

Where MAX is the maximum possible pixel intensity value (255 for an 8-bit grayscale image), and MSE is the Mean Squared Error between the original and processed image.
The merit of PSNR is that it is simple and requires less computation. It requires only the difference in the intensity between the pixels, so can be computed quickly even for large sets of images. This is an advantage for the evaluation of several approximation configurations during the Design Space Exploration.
But PSNR has some drawbacks. It only takes numerical differences between the two pixels into account, but not directly human visual perception. Images with the same PSNR can be seen to be visually different if the distribution of image distortion is different. Thus, PSNR is usually used in conjunction with perceptual quality metrics like SSIM for more holistic assessment.
In this research, PSNR is used to measure the quality of the output images created by exact arithmetic and approximate arithmetic setups. The higher PSNR value, the better capability of the approximate circuit for edge preservation and the better of the approximate circuit to reduce the complexity of the FPGA hardware. PSNR is thus one of the major quantitative metrics for the performance of the proposed approximate Sobel edge detection architecture implemented by the FPGA.

2.5.2 Structural Similarity Index (SSIM)

The Structural Similarity Index (SSIM) was calculated.
Structural Similarity Index (SSIM) is widely accepted image quality evaluation metric because of its consideration of image similarity based on the characteristics of the Human Visual System (HVS). The SSIM is different from the Peak Signal to Noise Ratio (PSNR) which only compares the numerical difference of the pixels, because SSIM is based on the perception of human changes in image structure, brightness, and contrast. As a result,

SSIM has been adopted as a standard performance metric for modern image processing research and is more realistic to assess visual image quality.
The need for a metric that goes beyond error-based metrics was what drove the development of SSIM. PSNR is a simple and easy-to-calculate measure, but it does not provide accurate perceptual image quality measure. The PSNR can be the same on two processed images, but they look very different to the human eye. To address this, SSIM should be calculated on the basis of the structure of the images and not the absolute differences in colors.
The SSIM metric consists of three fundamental components which are compared between two images:
Luminance comparison: which is based on the amount of similarity between the luminosity of the reference image and the processed image.
contrast comparison: It compares intensity variations in two images
Structural comparison: This indicates the extent to which the spatial (relative) relationships between adjacent pixels are maintained following image processing.
The three components are then merged into a single similarity index that has a range of values from 0 to 1. Pixel values equal to 1 mean no structural disturbance in the processed image as compared to the reference image, while values nearer to 0 mean higher structural disturbance in the processed image. For realistic image processing applications, typically SSIM values greater than 0.95 represent good visual quality.
The mathematical expression for SSIM is given by:

where:

μx and μy are the average intensity values of the reference and processed images.

The variances are denoted by σ_x^2andσ_y^2.

σ_xy is the covariance of the two images.

The small positive constants C_1 and C_2 are added to prevent the denominator from getting too small and causing numerical instability.

In contrast to PSNR, SSIM is mainly concerned with structural information. Due to the fact that humans observe objects primarily on their structure and not the exact values of the pixels, SSIM tends to be more similar to human visual perception. Hence, many recent image processing studies have reported the SSIM in addition to PSNR in order to comprehensively assess the image quality.
For FPGA approximate image processing, preserving the structural information is especially critical since many computer vision algorithms depend on the preservation of the edge continuity and the object boundaries. The Sobel edge detection algorithm is used to find edge information from an image because it detects an intensity transition between the neighboring pixels. If there is too much distortion introduced by approximation, important features of the image might be lost or spurious edges may be created. So, SSIM can be used as an effective measurement tool to check if the approximate arithmetic keeps meaningful structural information.
The proposed research will use SSIM to compare against the edge images generated with exact arithmetic against those obtained from various approximate arithmetic configurations. The more the SSIM value is closer to 1, the closer the structural attributes are to the true implementation even though it is hardware-simplified. On the other hand, lower SSIM values indicate the beginning of continuity and accuracy deterioration of edge structures due to approximation.

SSIM is better than PSNR for perceptual evaluation but it also has some drawbacks. The metric is more complex to compute since the local statistical parameters need to be computed over the entire image. In addition, SSIM mainly measures structural similarity, which is not necessarily representative of all aspects of image degradation. Therefore, the SSIM is often used along with PSNR and Mean Squared Error (MSE) to provide a complete evaluation of image quality.
SSIM is a key figure in the effectiveness assessment of approximate arithmetic in FPGA-based Sobel edge detection in this research. The proposed approach is capable of objectively determining the approximation configuration that gives the best trade-offs between hardware optimization and visual image quality, through integration of SSIM and hardware parameters, including LUT utilization, propagation delay, and power consumption.

2.5.3 Mean Squared Error (MSE)

In digital image processing, One of the most basic objective image quality assessment metrics is the Mean Squared Error (MSE). It is a metric based on the mean of the squares of the deviations of the intensity values of pixels of the original image and the processed image. MSE is directly related to the numerical error that occurs during image processing operations, where the error is intentionally designed and controlled in approximate computing.
Perceptual quality measures consider not only structural properties, but MSE only considers the differences between the numbers of the pixels. A pixel of the processed image is compared to the corresponding pixel in the original reference image. These pixel values are subtracted and then they are squared to remove negative values and to highlight larger errors. The error is calculated by averaging the squared differences over the whole image to get a single error value that represents the overall level of distortion in the image. The expression for the Mean Squared Error is:

where:
M and N are the dimensions of the image.

- I(i,j) is intensity of the original image.
- K(i,j) is the intensity of the corresponding pixel in the processed image.

The smaller the MSE value, the closer the processed image is to the original image. On the other hand, a higher MSE value means a more distorted image due to processing. Hence, most image processing algorithms aim at not only reducing MSE, but also increase the computational efficiency.
Mathematical simplicity is one of the important properties of MSE. The metric is easy to implement, with low computational complexity. It is therefore often cited as a standard to compare image processing algorithms and hardware implementations. Moreover, PSNR is directly computed from MSE which is the basis for other image quality assessment methods.
When performing approximate image processing using FPGA, MSE is an important measure of the numerical accuracy of approximate arithmetic circuits. When a number of operations are simplified, such as in approximate adders and multipliers, there are inevitable discrepancies in the calculations. The MSE is a measure of the total impact of

all these deviations on the resulting edge image. The closer the MSE values, the more accurate the approximation will be, which means that the approximation error will not have any significant effect on the image.
The experimental results shown in this thesis illustrate that the MSE is increased slightly due to the low-level (Level-1) approximation used in the optimal configuration but the resource usage and power consumption of the FPGA are significantly reduced. These results show that the hardware efficiency can be significantly boosted by carefully designed approximate arithmetic circuits with only a small decrease in computational accuracy.

2.5.4 Importance of Multiple Image Quality Metrics

The importance of multiple image quality metrics is discussed.The significance of multiple image quality metrics is discussed.

Evaluating image quality is a crucial element of image processing study and plays an important role in assessing whether an image processing algorithm is able to preserve the critical visual information of the original image. No one measure of image quality is truly representative of all properties. Therefore, contemporary image processing research uses more than one single quality assessment simultaneously to get a global and reliable evaluation about the performance of the algorithms.
The image quality is addressed by different measures for each image quality metric. Some measure numerical accuracy while others measure the similarity of the structures or perceptual image quality. So, using just one evaluation criterion could lead to incomplete results, or even to false conclusions about the effectiveness of an image processing algorithm.
The Peak Signal to Noise Ratio (PSNR) is the main ratio used to measure the numerical similarity between the original image and the processed image. PSNR is directly derived from the Mean Squared Error, so it gives an idea of the distortion that occurs in the image processing. The higher the PSNR value, the better the image quality. However, PSNR does not take into account the properties of human vision and thus is not necessarily correlated to the actual visual perception of the processed image.
PSNR, SSIM and MSE are metrics that measure different aspects of images and are often used together when conducting experiments. These three measurements are used together to give a more balanced view of both accuracy of numbers and image quality perceived. For instance, if the structural information is affected, a processed image could have a relatively high PSNR value but a lower SSIM value. Likewise, two approximation methods can achieve the same MSE but with different levels of visual quality. The following situations illustrate how using multiple quality measures are required for accurate performance measurement.
The use of multiple evaluation metrics is even more significant when performing approximate image processing with FPGAs, as the computations are intentionally made with controlled errors. Optimization of hardware does not provide a way to decide if an approximation procedure is acceptable in practiceInstead, there is always a need to compare hardware improvement with the image quality measurement, and to let it be in the acceptable approximation error. Thus, the proposed research combines the assessment of the three different PSNR, SSIM and MSE at the same time, covering all the configurations of approximations explored in Design Space Exploration. In addition to

the image quality metrics, other hardware synthesis parameters, like Lookup Table (LUT) utilization, propagation delay, operating frequency and power consumption are analyzed. This method incorporates into a single method an evaluation process, so that it is possible to compare the different approximate arithmetic circuits and choose the one best implemented on an FPGA. Combinedly, several quality measures also increase the confidence in experimental conclusions. The PSNR, SSIM and MSE measures complementary characteristics of the image quality, and high similarity between all three measures is good sign of the efficiency of a certain approximation configuration. The proposed evaluation approach will be able to minimize the potential for using only one image quality measure. Finally, a system of several objective image quality measures is presented and it is shown to be a complete system to evaluate approximate image processing systems based on FPGA. The presented method in this thesis ensures optimal use of the hardware without compromising the visual quality of the output of the edge detection and achieving efficient and reliable embedded vision systems

.

2.6 Challenges in Approximate Computing for Image Processing

Approximate computing has become a successful design paradigm that is used to make the hardware use more efficiently by deliberately introducing small errors in the computation of applications which naturally tolerate limited errors. It is possible to achieve such power saving, silicon area saving, computational delay, and hardware complexity, but still there are some problems to be solved to implement this concept in an image processing system. They must be handled with care, to ensure that the use of approximate arithmetic does not significantly impair the effectiveness of the application, and the reliability of the system. The main problem is that the balance between the accuracy of the computations and the optimization of the hardware is maintained. The proposed approach of approximate computing is to lower the hardware cost and to reduce the arithmetic operation. But, if it is too close, it is possible to add computational errors that are not acceptable, affecting the output image quality. The aim of the optimal approximation level thus ends up being a complex design task that takes into account both the quality of the approximation as well as the performance of the hardware used. For image processing programs where arithmetic operations are directly applied to the edges and intensity values of the image this balance is particularly important. Inaccuracies in the results of arithmetic computations due to approximation are not limited to the initial arithmetic operation, but will be spread to subsequent steps in processing. Many neighboring pixels contribute to the value of the output pixel in convolution-based image-processing algorithms, such as Sobel edge detection. Small errors arising from

multiplications and additions will therefore be compounded and will eventually result in visible errors in the computed edge image. It is therefore important to understand how the approximation errors propagate through the circuits, in order to choose appropriate approximate arithmetic circuits.

Approximate arithmetic alters numerical calculation, which affects objective image quality metrics like PSNR, SSIM and MSE. While they might be unnoticeable in visual terms, such changes could lead to a dramatic loss in continuity along edges, cause boundaries for objects to be blurred, and create fake edge structures in aggressive approximation. Therefore, when choosing hardware, along with the approximation of the circuit, it is imperative to evaluate the image quality as well. In addition, dependency on applications makes approximate computing implementation more complicated. The ability of different image processing algorithms to tolerate computational errors varies. Most operations (such as image enhancement, filter, edge detection) can be performed with moderate approximation because neighbouring pixels are similar in terms of intensity properties. In contrast, medical diagnosis, biometric authentication or science imaging application has much higher computational precision requirement. In the general case, however, it is not possible to use approximate arithmetic without keeping an eye on the required accuracy for an application area. For hardware verification, another major challenge during implementing FPGA is also considered. Approximate circuits are designed not to follow exactly the mathematical behavior of the circuit and conventional methods of verification are not enough to check the correctness of the design. It is not only the designer's job to verify the numerical output of the design, but the designer must also take into account the performance of the hardware, the correctness of the calculation, functional correctness and the quality of the image from the application perspective. It is a multidimensional verification process which makes design much more complex than traditional exact digital ones. Generally the power savings due to approximate arithmetic are circuit-dependent and can depend on the type of FPGA used, and they also can result in reduced switching activity and logic. There are different approximate adders and multipliers that can have similar areas but different power. Hence, it is required to test the hardware with a lot of experiments and not only theoretically to choose the best approximate configuration. Scalability is another restriction of practical approximate computing systems. Different environmental conditions should be also taken in account when selecting appropriate reliability. In the real world with embedded systems we have

to contend with temperature changes, power supply changes, sensor noise and manufacturing variations. The external factors can interact with the errors of approximation in the calculations to produce behaviour different from that seen in a laboratory experiment. Nevertheless, approximate computing is still a promising approach for today's embedded image processing applications. The design space exploration, combined with objective hardware and image quality evaluation, enables the designer to choose an "approximate" design that provides maximum computational efficiency at the same time keeping the application performance acceptable. Therefore, it is still an active research topic to solve these problems to design reliable, energy-efficient, and high performance approximate computing system for future image processing applications based on FPGA.

2.7 Summary and Research Gap

The literature shows the rich theoretical and empirical basis of approximate computing for image processing, and EvoApproxLib is a practical library of approximate arithmetic circuits characterized for the image processing domain. There is, however, a lack of systematic, end-to-end studies that integrate: (i) behavioral software simulation at multiple levels of approximation, (ii) comprehensive DSE with quantitative metrics of its quality, and (iii) direct FPGA implementation and synthesis comparison. This thesis aims to fill this void in the case of Sobel edge detector.

2.6.1 Comparative Analysis of Existing Studies

This critically assesses and compares existing research to determine why the current studies are relevant.

The reviewed literature in this chapter shows that approximate computing is one of the viable solutions to hardware optimization techniques in the modern digital systems. Many researchers have been able to achieve acceptable computational accuracy, hardware area and propagation delay reduction, and power consumption via approximate arithmetic. All of these studies provide evidence that many multimedia applications and signal processing applications have sufficient error tolerance such that they can be improved by approximation based optimization of the hardware.

Image processing applications with FPGA also showed the good results of hardware parallelism for real-time image processing. Various edge detection, image enhancement, feature extraction, and object recognition FPGA implementations have been reported in the literature. All such implementations were found to be able to process more messages than the equivalent software implementations and to have lower execution latency.

Notwithstanding these successes, there have been only a few studies that used approximate arithmetic and full FPGA-based Sobel edge detection systems. The literature

highlights many publications that perform either the optimization of hardware or the evaluation of image quality but relatively few publications address both aspects at the same time. Moreover, the comparisons between several approximation configurations with the same synthesis conditions are relatively rare.
The second thing revealed from the literature is that researchers are often using various FPGA platforms, synthesis tools, data sets and methodologies to evaluate. It is difficult to compare the various approximation techniques directly due to these variations. A consistent assessment methodology that can be used to analyze hardware performance and image quality at the same time is thus very desirable.
In general, the literature has substantiated that approximate computing is a potential solution for resource limited embedded image processing systems. However, more research is needed to find approximation strategies that can offer a tradeoff between high hardware efficiency and output image quality.

Table 2.3 summarizes how this thesis positions itself relative to a representative set of key prior works spanning circuit-level approximate arithmetic design, application-level approximate computing frameworks, and approximate image processing, to make the identified research gap concrete rather than purely descriptive.

Table 2.3: Comparative Summary of Key Related Works

Study	**Approximation Studied**	**Application Domain**	**Hardware Validation**	**DSE / # Configs**	**Image Quality Metrics**
Mrázek et al. [3]	Approximate multipliers (EvoApprox Lib-based)	Neural network inference	ASIC-level simulation only	Single / few, not systematic	Task accuracy only (not PSNR/SSIM)
Kulkarni et al. [14]	Underdesigned multiplier architecture	Generic DSP	Circuit-level only, no application	Single design	None
Venkatesan et al. [6] (MACACO)	Adder/multiplier error-analysis framework	Generic / framework-level	Simulation framework, no FPGA synthesis	Multiple (framework capability)	Generic statistical error metrics
Raha et al. [12]	Quality-configurable processing	Hyperspectral imaging (UAV)	Simulation-based	Multiple quality levels	Application-specific, not PSNR/SSIM/ MSE jointly
Mrázek et al., EvoApprox8b [11]	Adder + multiplier circuit library (source of this thesis's	General-purpose (library/benchmark paper)	ASIC (45 nm) characterization only — no FPGA, no application	N/A (library paper, not a DSE study)	N/A

	circuit inspiration)				
This Work ★	Adder (AA-L1–L3) + Multiplier (AM-L1–L3), EvoApprox Lib-inspired	Sobel edge detection (image processing)	Full Xilinx Artix-7 FPGA synthesis (Vivado)	7 configurations (C0–C6), systematic quality-gated DSE	PSNR, SSIM, and MSE, jointly, against a validated exact baseline

2.6.2 Limitations of Existing Research

This study has identified some limitations in the previous research.
In spite of the enormous progress that has been achieved in approximate computing, there are a number of gaps in the existing literature.
Several published studies only consider approximate adders and multipliers at the arithmetic circuit level and do not incorporate them into a complete image processing architecture. This makes the estimation of the impact of approximation on overall application performance difficult and unreliable in many cases.
There are several investigations that show that hardware utilization is improved but only briefly discuss the image quality. Hardware optimization is not enough to decide the practical usefulness because the ultimate application of image processing is visual accuracy. The evaluation of the comprehensive nature should be carried out simultaneously by examining both the hardware and the image quality.
One of the constraints is the relatively low number of approximation configurations studied in the past. Numerous studies only examine one approximate arithmetic architecture, leaving it unclear whether there are better architectures. The optimal hardware–accuracy trade-off is not easy to determine, without systematic Design Space Exploration.
Other existing implementations involve using simulations without full FPGA synthesis and implementation. Therefore, reported hardware enhancements might not be an accurate reflection of the actual FPGA performance once placed and routed.
Moreover, few studies analyze the impact of approximation on Sobel edge detection applied with existing arithmetic library for approximation like EvoApproxLib that is easily accessible to the public. This lends itself to further exploration with uniform approximation materials.

2.6.3 Identified Research Gap

From the literature survey, it is evident that approximate image processing using FPGA is still a current research problem, where there are some challenges that are yet to be solved. Achievement of both hardware optimization and image quality preservation has been shown in the existing studies but few studies have evaluated both aspects in a single experiment.

Previous studies mostly focus on individual approximate arithmetic circuits instead of performing a full-scale FPGA implementation of image processing algorithms. Therefore, the correlation between arithmetic approximation, FPGA resources usage, power consumption, propagation delay and visual image quality has not been fully explored.
Further, only a few studies use Design Space Exploration to test and evaluate different approximation configurations with the same implementation conditions. It is difficult to choose the most suitable level of approximation without a systematical comparison for practical application on FPGA.
Hence, there is an obvious need for the development and experimental evaluation of a system implementing an FPGA-based Sobel edge detection system integrating multiple approximate arithmetic configurations, supporting objective image quality assessment based on PSNR, SSIM and MSE, and supporting FPGA synthesis metrics analysis such as utilization of LUTs, timing performance and power consumption.

2.6.4 Motivation for the Proposed Research

The research is proposed because of the following reasons:

In this research, the motivation is to design an approximate computing edge detection architecture with FPGA for achieving efficient computation in Sobel edge detection with preserving image quality.
The need for energy efficient energy-dense image processing hardware architectures remains strong as more and more image processing applications move into embedded systems, autonomous vehicles, robotics, industrial automation, and intelligent surveillance. However, traditional exact arithmetic implementations typically require a large amount of FPGA resources and power consumption and the resulting precision of the numbers is greater than what is required for the application.
The concept of approximate computing proposes to take advantage of the tolerable error of image processing algorithms to enhance hardware efficiency. But, the choice of the level of approximation must be carefully evaluated in the experiments so that in reducing the number of bits, the performance on edge detection is not significantly compromised.
The proposed research attempts to solve this problem by incorporating approximate arithmetic circuits extracted from EvoApproxLib in an FPGA implementation of Sobel operator. Design Space Exploration is used to test several approximation configurations and to objectively compare the hardware performance and image quality. The result of the analysis is the practical design guidelines for providing an approximate arithmetic architecture for a real time image processing application in an FPGA.

Chapter 3: Methodology

3.1 Overview

The methodology proposed is structured along seven steps of the design process as shown in Figure 3.1. The flow starts with an exact implementation of Sobel edge detection in Python to be used as quantitative reference baseline. It then moves on to the arithmetic modularization, approximate function design, Design Space Exploration, quality evaluation and finally FPGA implementation and synthesis. Measurable outputs that are linked to the next steps for each step.

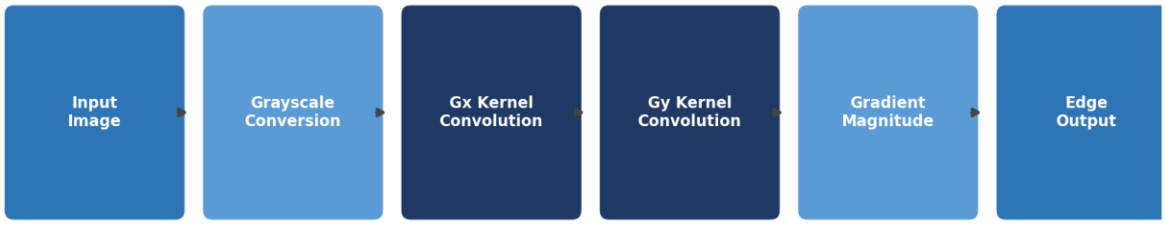


Figure 3.1: Sobel Edge Detection Processing Pipeline

Figure 3.1: Proposed Methodology — Seven-Step Design Flow for Approximate Sobel Edge Detection

In this research, a structured and sequential approach is used for designing, implementing and testing FPGA based approximate Sobel edge detection system The methodology has been carefully developed to evaluate the proposed architecture, not only in terms of hardware efficiency, but also in terms of image quality and computational accuracy. The whole methodology starts with the implementation of a conventional Sobel edge detection algorithm with exact arithmetic circuits. This implementation will be used as the reference to compare against all the approximate implementations. It is critical to have an exact implementation, since it is a standard image quality and hardware performance to which comparisons are made. After the baseline implementation, the elements of the arithmetic parts in the Sobel architecture are then modularized. It is not arithmetic circuits but independent arithmetic modules to be designed. The different approximate adders and multipliers can be combined without changing the rest of the processing architecture in this modular strategy. This flexibility greatly facilitates comparisons of different approximation configurations.

Arithmetic circuits are then approximated, using EvoApproxLib and fed into the modular Sobel architecture. Various approximate adders and multipliers are tested to study their impact on the FPGA resource utilization, propagation delay, operating frequency, power consumption and output image quality. The various approximation configurations are different compromise between hardware optimization and accuracy of calculation.

Design Space Exploration (DSE) is conducted after the hardware integration. Various approximate configurations are synthesized with the same implementation conditions for

the FPGA. During the synthesis, the performance report of the hardware is generated which contains details on LUT utilization, Flip-Flop utilization, critical path delay, maximum operating frequency and estimated power consumption.
Objective image quality metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Mean Squared Error (MSE) are used to evaluate the generated edge images. These metrics provide numerical measures of the effect of approximation and can be used to measure if there is acceptable or excess computational error made by the hardware optimization.
Finally, a combination of all hardware and image quality measurements are carried out simultaneously to determine the approximation configuration that gives the best compromise between implementation efficiency and image quality. The configuration selected is the best FPGA solution for the approximate Sobel edge detection architecture proposed.

The entire process used in this work is systematic and offers a possible framework for the study of approximate computing for FPGA applications in image processing. The procedure of implementing, synthesizing, evaluating and comparative analysis is sequential which allows the experiment results obtained in the process to be consistent, repeatable and scientific.

3.2 Step 1 — Exact Sobel Baseline

The Sobel operator uses 2 convolution kernels to calculate the gradients of the image. The horizontal gradient Gx and vertical gradient Gy are computed for each pixel (i, j), as follows:

$$Gx = \Sigma\,\Sigma\, I(i+a, j+b) \times Kx(a,b) \quad \text{where } a,b \in \{-1, 0, +1\}$$
$$Gy = \Sigma\,\Sigma\, I(i+a, j+b) \times Ky(a,b) \quad \text{where } a,b \in \{-1, 0, +1\}$$

The Sobel kernels Kx and Ky are fixed 3×3 matrices:

$$Kx = [[-1, 0, +1], [-2, 0, +2], [-1, 0, +1]]$$
$$Ky = [[-1, -2, -1], [0, 0, 0], [+1, +2, +1]]$$

The gradient magnitude at each pixel is then computed as:

$$G(i,j) = \text{sqrt}(Gx^2 + Gy^2)$$

The output of this step — designated sobel_exact.jpg — serves as the pixel-accurate reference image against which all approximate outputs are evaluated.

3.2.1 Development of the Exact Sobel Baseline

The creation of the Exact Sobel Baseline.

The first step of the proposed methodology is the realization of an exact Sobel edge detection architecture. This implementation can be used as a reference for all approximate arithmetic implementations. The critical first step is to create a baseline since it will offer the hardware performance and image quality needed for objective comparisons during the research.

The Sobel edge detection algorithm is chosen because of its many applications in digital image processing and computer vision systems. The algorithm finds the edges of the image by calculating the intensity gradient for each pixel with two pre-defined convolution kernels. One kernel will compute the horizontal intensity difference, the other will compute the vertical intensity difference. These are then merged to create the final edge image.

For the exact implementation, all the arithmetic operations are done on conventional accurate adders and multipliers. No approximation is done in calculating the gradient, which means that the gradient is calculated mathematically throughout the processing pipeline. This means that this edge image is the best quality possible for the architecture implemented, and becomes the reference image for quality assessment later on.

Thesobel architecture is completely designed in Verilog Hardware Description Language (HDL). The hardware designing methodology is adopted in the modular design to enhance flexibility and easier to replace the arithmetic modules in future. The processing of individual operations like gradient calculations, arithmetic operations and generation of outputs are performed by distinct hardware modules. This modular design makes it more convenient to integrate approximate arithmetic circuits in later stages of research.

Once the hardware description is completed, it can be synthesized using the chosen FPGA development environment. Implementation reports are generated by hardware synthesis, which include the following information: FPGA resource utilization, propagation delay, operating frequency, and estimated power consumption. These measurements provide a baseline of the exact Sobel architecture's hardware characteristics.

Evaluation of approximate implementations is done by calculating PSNR, SSIM and MSE against the exact edge image generated from the baseline implementation, and comparing them with approximate implementations. This is an exact reference image that is used to compute objective image quality measures such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Mean Squared Error (MSE) that compare the approximate outputs.

There are a number of benefits of using an exact baseline. First, it gives a reference to evaluate each approximation configuration under the same experimental circumstances. Second, it allows to measure accurately the degradation of image quality caused by approximate arithmetic. Lastly, it can quantify the benefits to the hardware that comes with approximation with respect to a fully accurate one.

So far, the proposed research methodology is based on the exact Sobel baseline. All approximate arithmetic configurations studied in subsequent phases are compared against this configuration, which guarantees a fair comparison and a reliable study of the experimental results

3.2.2 Baseline Performance Evaluation

The exact implementation of the Sobel is tested to ensure hardware performance and functional correctness prior to introducing approximate arithmetic. Functional verification verifies that this is the correct baseline implementation, thus provides an objective baseline for all the approximate arithmetic configurations to be compared in the next phases of the proposed methodology that the edge image generated by the hardware implementation matches the output from the Sobel operator. This verification will make sure that any discrepancies that arose in later experiments are only due to "rounding errors," not to mistakes in the implementation.
The implementation report of the synthesized design is analyzed to calculate the Lookup Tables (LUTs) usage, Flip Flops (FFs) usage, propagation delay, operating frequency and estimated power consumption. All approximate configurations are compared to these hardware measurements.
Likewise, the output edge image produced by the exact implementation would be the ground truth for image quality evaluation. However, as there is no approximation, the exact implementation yields maximum PSNR and SSIM scores, while also having minimum possible Mean Squared Error. It is, therefore, suitable for comparison of the effect of approximate arithmetic on visual image quality. The base-line evaluation also verifies the correctness of the Verilog implementation before the hardware optimizations. This validation reduces the chances of experimental errors, and the reliability of subsequent Design Space Exploration.

3.2.3 Test Image Dataset, Resolution, and Preprocessing

The primary Design Space Exploration (Chapter 4) and FPGA resource evaluation (Chapter 5, Sections 5.1–5.6) use a single procedurally generated synthetic grayscale test image of 256×256 pixel resolution, 8-bit unsigned pixel depth (0–255), created programmatically (generate_test_image() in sobel_exact.py) rather than sourced from an external image dataset. The image combines four components chosen to jointly exercise the full pixel intensity range and a variety of edge/gradient conditions relevant to Sobel evaluation: (i) a diagonal intensity gradient spanning the full 0–255 range, (ii) two solid rectangular regions of contrasting intensity (pixel values 210 and 40) to create strong, sharp edges, (iii) a filled circular region (pixel value 170, radius 35 px) to test curved-edge response, and (iv) additive uniform noise in the range [−8, +8] applied with a fixed random seed (NumPy seed = 42) to ensure the image, and therefore all reported results, are exactly reproducible from the provided source code. No additional preprocessing (resizing, normalization, or filtering) is applied, as the image is generated directly at the target 256×256 resolution used throughout the evaluation. This procedural approach was adopted for the primary DSE to give full, reproducible control over edge geometry, contrast levels, and noise characteristics, independent of any single external dataset's licensing or availability constraints.
To address concerns regarding the generalizability of results derived from a single test image, Section 5.9 extends this evaluation to a set of seven test images — the original synthetic image described above, together with six additional publicly available grayscale reference images from the scikit-image standard test image collection (camera, astronaut, moon, brick, gravel, and coins), selected to span low-detail, high-

texture, and natural photographic content. Each image was converted to 8-bit grayscale and resized to 256×256 pixels (bilinear interpolation) for consistency with the primary evaluation resolution, with no further preprocessing applied. Full details, results, and the corresponding code verification are reported in Section 5.9.

3.3 Step 2 — Arithmetic Modularization

In the Sobel convolution kernel, two special arithmetic functions are written for the operations performed:

exact_add(a, b): Returns the exact sum of a and b.

exact_mul(a, b): Returns the exact product a × b, a and b being standard Python integers.

This structural refactoring ensures that this algorithm will produce precisely the same result (bitwise) as Step 1 and the codebase will be set up for systematic approximation injection. The strategy of modularization is the key to the methodology: All arithmetic operations are isolated in separate interfaces making it easy to change the exact operation with its approximation without affecting the other algorithm logic.

3.3.1 Modular Hardware Architecture

The second step of proposed technique is segmentation of arithmetic unit of Sobel edge detection structure. The design is partitioned into a number of independent modules, instead of implementing the whole algorithm as one single hardware block. The modular design methodology makes the design more flexible, easier to debug and allows integration of various approximate arithmetic circuits without changing the overall processing architecture. Modularization is one of the best methods for increasing the scalability and maintainability in digital hardware design. A module has a specific set of input and output connections, and a specific function. This means that changes in one hardware component don't necessitate redesign of the entire system. In Sobel architecture, the arithmetic operations for summing and multiplying are separate from other arithmetic operations, and are implemented in separate hardware devices. Note that these arithmetic circuits are the most expensive ones that perform the computation of the Sobel operator, and by substituting them with approximate circuits, the impact of the approximation can be assessed separately from the rest of the circuits. The modular design also contributes to the consistency of the experiments. For each approximation configuration, the exact same image processing pipeline is used and only the arithmetic circuits differ. Thus, any changes in hardware performance and image quality can be traced back to the approximate arithmetic and not to changes in hardware architectures that are not relevant

to the arithmetic. Another advantage of the modular hardware approach is the ability to reuse code. Arithmetic modules could be utilized for a number of various image handling applications with minimal changes. This property makes development quicker and simplifies further research with other approximation methods or more sophisticated image processing algorithms.

3.3.2 Separation of Arithmetic Modules

The all-arithmetic units are implemented as separate Verilog modules for the Design Space Exploration. These modules consist of adders, multipliers, gradient computation blocks and output generation circuits. Every module is tested separately, prior to integration into the full Sobel architecture.
With the separation of arithmetic modules, the interfaces of exact and approximate circuits can be identical. In this way, implementing an exact adder with an approximate adder only involves replacing a module, and the rest of the module architecture stays intact. This approach to design will make the implementation much simpler and reduce the need for redoing the design during experimental testing.
Only arithmetic modules are replaced with approximate arithmetic circuits given by EvoApproxLib. The effect of approximation can be objectively assessed under the same experimental conditions, because the other hardware components are kept constant.
The modular structure also allows for easier hardware verification. Dedicated test benches are available for the individual arithmetic modules to test them independently before integration to the system level. The method makes debugging more efficient and the likelihood of correct solution for the full FPGA implementation is greater.

3.3.3 Benefits of Modularization in Design Space Exploration

Modularization is used in designing space exploration systems because of the following benefits:Modularization in space exploration systems is used due to following benefits:

Use of a modular architecture is a very important aspect of the Design Space Exploration methodology used in this research. Arithmetic circuits are programmed as interchangeable modules, and thus, various combinations of approximations can be tested in an efficient manner with no redesign of the entire Sobel processor.
Approximate adders and multipliers can be separately inserted, synthesized and evaluated in the architecture. FPGA reports are generated for every configuration with the resource utilization, propagation delay, operating frequency and power consumption of the configuration. At the same time, the edge images produced are analyzed by PSNR, SSIM and MSE to know the effect of approximation over the image quality.
This modular evaluation approach lowers the effort required for implementation and boosts the reliability of experimental comparisons. The experimental results differ only because of the arithmetic circuits selected because all configurations have the same

processing pipeline. Therefore, the proposed modular design lays out a fair testing ground for approximate computing in FPGA-based image processing applications that is reproducible.

3.4 Step 3 — Approximate Arithmetic Design

The task is to approximate the design using the arithmetic.The task is to approximate the design with the arithmetic.

Six approximate arithmetic functions are provided as models of Python behavior that mimic behavior in the EvoApproxLib circuit. The circuits implemented along with their characteristics are summarized in table 3.1.

Table 3.1: Approximate Arithmetic Circuits — Specifications

Circuit	Python Model	Error Level	Hardware Saving	Use Case
Approx Add L1	(a+b) & ~1	Very Low	Low	Best quality
Approx Add L2	(a+b) & ~3	Medium	Medium	Moderate use
Approx Add L3	(a+b) & ~7	High	High	Area critical
Approx Mul L1	(a*b) & ~1	Very Low	Low	Best quality
Approx Mul L2	(a*b) >> 1	Medium	Medium	Moderate use
Approx Mul L3	(a*b) >> 2	High	High	Area critical

The truncation-based approach for approximate adders is similar to the lower-bit-elimination process used in EvoApproxLib to shorten the propagation chain of carries. The shift-based method of approximate multipliers is similar to the practical method of partial product column elimination used in the design of approximate multipliers.

3.4.1 Selection of Approximate Arithmetic Circuits

Let's get back to our (approximate) arithmetic circuits.Now back to our arithmetic circuits (approximate).

How well approximate computing is going to work hinges on the choice of suitable arithmetic circuits. Different approximate adders and multipliers have distinct hardware properties and can have different errors, so a careful choice of the circuit must be made to balance between the hardware efficiency and the quality of the computational results.
This research uses arithmetic circuits (circuits) approximate selected from the library EvoApproxLib. EvoApproxLib contains the set of formally verified approximate adder and multiplier designed by applying evolutionary algorithms. Multiple hardware and error metrics are used to characterize each circuit, and the designer can choose the architecture based on the application requirements.
Instead of choosing a circuit that is close to the desired accuracy, several arithmetic circuit designs will be tested in this research. This way, there is a possibility of a systematic comparison between different levels of approximation and to identify the configuration that offers the best hardware–accuracy compromise.
The selection process thus takes into account both the optimization of the hardware and application level performance. Even if these circuits give substantial hardware saving, but produce an excessive error during computation, they are eliminated; similarly, circuits that give small hardware saving are eliminated. It aims to find practical approximated architectures, suitable for real time FPGA implementation.

3.4.2 Integration of EvoApproxLib Components

Once suitable approximate arithmetic circuits have been selected the next step is to add these circuits to the FPGA implementation of the Sobel edge detection algorithm. Since the proposed architecture is designed in a modular way, the exact arithmetic unit can be switched with an approximate arithmetic unit without changing the rest of the processing pipeline.
All approximate circuits from EvoApproxLib are coded as separate Verilog modules. The input/output interfaces are the same as those of the exact arithmetic units. The compatibility allows a quick replacement of different approximate circuits without any changes in the structure of the surrounding hardware.
The first step to integration is to replace the exact adders used in computing the gradients with approximate adders in EvoApproxLib. Likewise, approximate multipliers are included in arithmetic operations that include multiplication Each configuration has been synthesized separately to explore the effect on the FPGA configuration characteristics. A standardized circuit library will provide uniformity in experiments evaluation. All approximate circuits start from the same verified repository, which provides hardware comparisons without bias and repeatability. Furthermore, EvoApproxLib provides error probability, Mean Error Distance (MED), Worst Case Error (WCE) and hardware complexity data in detail to support circuit selection. This modular integration approach not only significantly reduces implementation effort but also allows Design Space Exploration for various approximation configurations.

3.4.3 Approximation Configuration Generation

Multiple hardware configurations are created by combining approximate arithmetic circuits at varying levels of approximation to investigate the impact of the various levels. The configurations have a different balance of accuracy of the calculation and optimization of the hardware. There is a special identifier for each configuration, to facilitate analysis and comparison of experiments. The naming allows unambiguous mapping of synthesis reports, simulation results and image quality measurements and metrics to a specific implementation of the hardware. All configurations generated are synthetically, simulated and evaluated in exactly the same way. All the experiments were fair and repeatable and used the same FPGA device, same set of synthesis parameters, same input images, and the same set of evaluation metrics. This standardized method removes the external factors which can affect the results of the experiment. The generated configurations are then evaluated by carrying out a Design Space Exploration on both FPGA implementation metrics and image quality measurements and analyzed concurrently. This complete comparison allows the identification of the configuration which maximises the hardware performance whilst maintaining the most accurate edge detection.

3.4.4 Design Considerations

The proposed approximate arithmetic architecture is developed taking into consideration several design considerations. The main goal is to minimize the resource usage of the FPGA while maintaining good quality edge-detection operation.
The arithmetic circuits with fewer number of logic resources are selected to minimize hardware complexity Meanwhile, approximation errors are meticulously guarded against to avoid appreciable loss of image quality. In particular, the goal is to keep identified edges as discontinuities, since these discontinuities may negatively impact on the image processing tasks that follow. Timing is also an important factor for performance. In the simplified arithmetic circuits, the critical path length can be reduced and consequently the operating frequencies are raised, and the propagation delay is shortened. The improvements are particularly beneficial in applications with high throughput requiring real-time image processing.
Finally, power efficiency is considered in the design process. Lower switching activity results in lower dynamic power consumption by reducing logic complexity. This property is well suited to energy constrained embedded systems.
In conclusion, the proposed approximate arithmetic architecture is a good trade-off between efficiency and accuracy requirements that could be used for FPGA implementation of Sobel edge detection.

3.5 Step 4 — Design Space Exploration (DSE)

The core decision making process of this thesis is called Design Space Exploration (DSE). It aims to systematically compare every possible combination of approximation of the adder and multiplier and to find the optimum hardware/quality combination. DSE is a tool that addresses the fundamental question: which of the many approximate designs should be implemented on an FPGA?

The entire DSE algorithm is shown in structured flow chart in figure 3.2. The algorithm consists of eight clearly formulated steps: starting with baseline establishment, configuration generation and evaluation, applying quality acceptability thresholds and ending with optimal design selection. All configurations are considered as potential designs and are either accepted (if the PSNR and SSIM values meet the set criteria) or rejected (if the PSNR and SSIM values do not meet the set criteria). The designs that are accepted are rated and the best one is chosen.

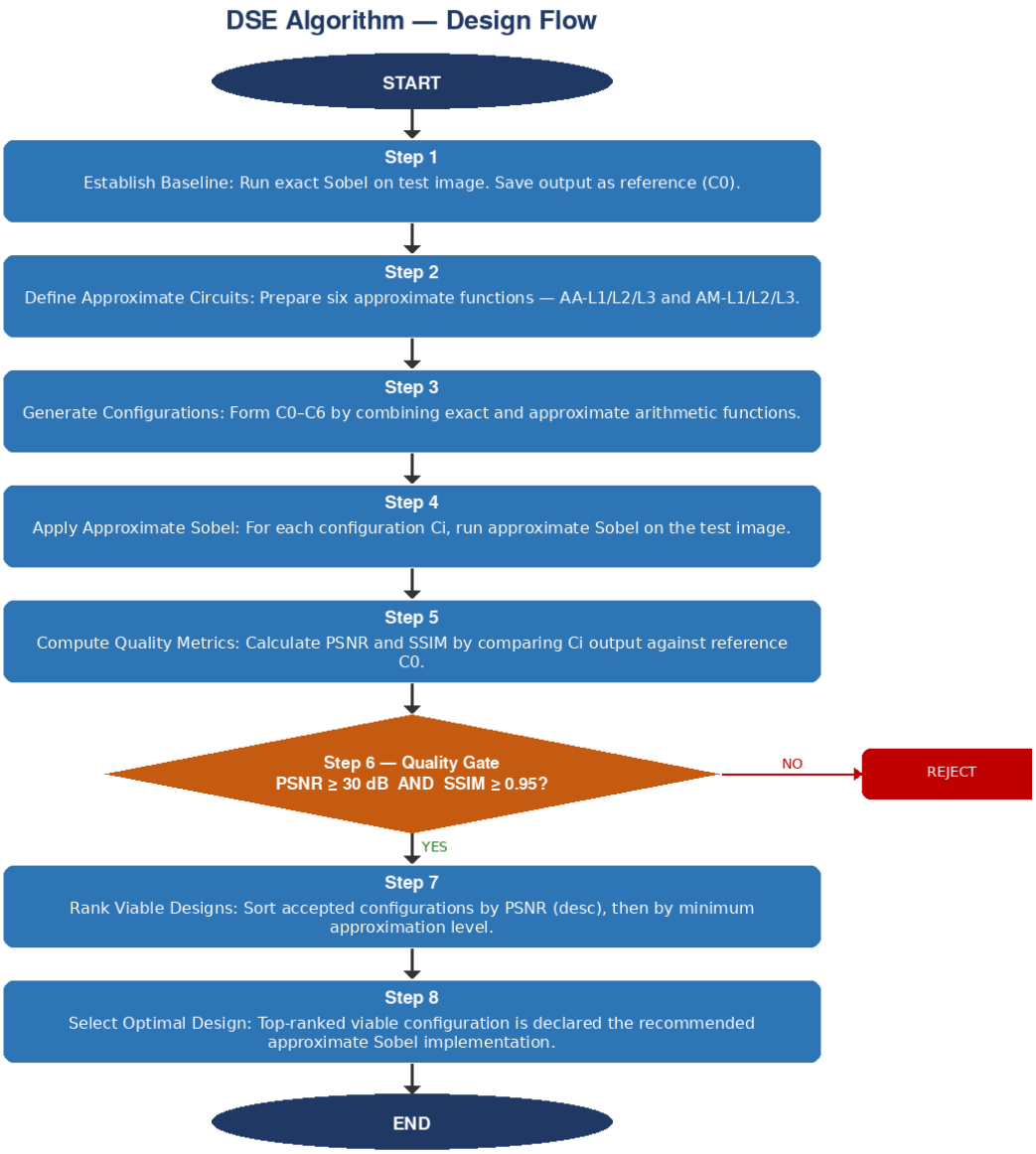


Figure 3.2: DSE Algorithm Flowchart — Steps for Selecting the Optimal Approximate Sobel Configuration

3.5.1 DSE Algorithm — Step-by-Step Description

The DSE algorithm is an eight step algorithm as follows:

Step 1 — Establish Baseline: Compute the exact Sobel of the test image. Save output as the reference. This is the reference value to which all the approximate answers are compared.

The six approximate circuits are prepared: Three adder levels (AA-L1, AA-L2, AA-L3) and three multiplier levels (AM-L1, AM-L2, AM-L3) covering the whole approximation spectrum.

In Step 3, Generate Configurations, seven configurations (C0 – C6) of the combination of exact and approximate arithmetic functions are generated. C0 is exact arithmetic, both operations, and the base. C6 is maximum approximation, both operations.

In Step 4, the approximate adder and multiplier functions are inserted in the Sobel kernel, and the entire image is processed for each of the configurations Ci.

Step 5 — Compute Quality Metrics: PSNR and SSIM computed by comparing the approximate output of Ci with the exact baseline. Together, these two numbers represent the quality score of the design.

Step 6 — Apply Quality Gate: If PSNR > or equal to 30 dB AND SSIM > or equal to 0.95, the configuration is considered to be a viable design. It is then rejected if it is not otherwise. This threshold provides that only practically deployable designs are taken into account.

Step 7 — Rank Viable Designs: All accepted designs are ranked (descending) by PSNR and (ascending) by minimum level of approximation. This ranking will take into account both quality and simplicity of hardware.

Step 8: Optimal Design: The most promising viable design is determined to be the optimal approximate design. The design is then coded and synthesized in Verilog HDL code on FPGA and is compared to the real hardware. The algorithm above sequentially tests all the different possible configurations of the approximation with the same experimental conditions. The FPGA implementation metrics and image quality measurements for each configuration are synthesized, simulated and evaluated. The gathered results are saved and analyzed to evaluate the setting that can achieve the optimum trade-off between hardware efficiency and edge detection accuracy. This automatic evaluation procedure considerably improves the reproducibility of experiments, while also reducing the amount of manual design.

3.5.2 Design Naming Convention

A consistent naming convention for the design is used, to ensure the systematic organization and ease of identification of different implementations of the hardware. In Design Space Exploration, several approximate arithmetic configurations will be examined and each of the configurations will be unique giving a unique identification for each synthesized design. The naming convention allows for convenient handling of simulation files, reports on the synthesis, results of implementing the FPGA and assessment of the quality of the resulting image. The algorithms used to implement Sobel edge detection (with arithmetic components) are the source of each design name. The naming of the implementation specifies if it includes exact arithmetic or approximate arithmetic blocks, and the approximate adder and multiplier that are chosen from EvoApproxLib. In this systematic approach, all the experimental configurations in the analysis are easy to find out.

A consistent naming convention also reduces potential confusion if dealing with many simulation results and synthesis reports. Each configuration produces its own implementation reports (LUT utilization, Flip-Flop usage, timing information, power estimation, and output images), which means the experimental data are always associated with the appropriate hardware configuration.

In addition, it's a naming convention that increases the chances for re-producibility. The evaluated configurations can be recreated easily by others in the future, knowing the

documented naming scheme and the appropriate approximate arithmetic circuits of the configurations. This consistency is especially important for comparing a variety of approximation levels synthesized under similar FPGA conditions.

3.6 Step 5 — Quality Metric Evaluation

The quality evaluation stage is executed to find out the impact of approximate arithmetic on the visual accuracy of the created edge images. The desired purpose of approximate computing purposely adds controlled computational errors to enhance the hardware efficiency, and it is crucial to determine whether the errors are acceptable in practical image processing applications.

In the proposed research, after the implementation of each approximate arithmetic configuration, the image quality evaluation is done after the FPGA implementation. All synthesized hardware designs operate the same image with the same set of conditions. Then the edge images are compared with the reference image from the actual Sobel implementation. This comparison allows to measure the image degradation caused by the approximate arithmetic.

Several objective image quality metrics are used to make for a thorough assessment. The various metrics quantify different aspects of image quality, and hence provide different insights into the performance of the proposed architecture. The multiple use of measure of quality gives a balanced evaluation of accuracy of numbers and their visual perception instead of using a single one.

In this work, Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) and Mean Squared Error (MSE) are used as the main quality measures. PSNR is used to compare the numerical resemblance of the true edge image and the approximate edge image, SSIM is used to measure the preservation of the structural information, and MSE is used to measure the mean square difference between the exact image and the approximated image that was produced during image processing. Together, these metrics serve as a good measure of the acceptability of the approximate arithmetic optimization used in the hardware for edge detection applications.

The quality metrics computed are compared with the FPGA synthesis reports generated in Design Space Exploration. By combining the two evaluations, the hardware optimization does not cause excessive degradation of images. The best approximation configuration, then, is the one that meets the required image quality criteria for the chosen evaluation criteria with minimum hardware costs.

In addition, use of common image quality measures allows for the direct comparison with previous research. The proposed methodology allows to objectively compare the results with other approximate computing methods reported in the literature, using widely accepted PSNR, SSIM and MSE metrics as performance measures.

Overall, the quality evaluation stage is the most significant in proving the effectiveness of the proposed approximate Sobel edge detection architecture using FPGA. The results gained allow a quantitative analysis of the Hardware efficiency/Visual image quality compromise, which helps in finalizing the best approximation configuration.

3.6.1 Peak Signal-to-Noise Ratio (PSNR)

PSNR is calculated for each approximate output image compared with the perfect reference output image:

$$MSE = (1/N) \times \Sigma [I_exact(i,j) - I_approx(i,j)]^2$$

$$PSNR = 20 \times \log_{10}(255 / \sqrt{MSE}) \quad [dB]$$

where N is the number of total pixels. In both cases, the PSNR results for all the approximation configurations are calculated based on the above equation and compared with the results obtained from the corresponding FPGA implementation. The configurations which gave higher PSNR and lower hardware resources utilization are more suitable for the proposed FPGA based Sobel edge detection system.

3.6.2 Structural Similarity Index (SSIM)

The computation of SSIM is performed via the implementation in the Python library scikit-image and the data range is set to 255. The SSIM value is computed for each of the approximate hardware configurations with the same input image set. The values obtained are compared along with PSNR, FPGA resource usage, propagation delay and power consumption to verify if the structure information of the identified edges is preserved after approximation. When the configurations with higher SSIM values required less hardware usage, they are to be considered better in terms of visual quality and hardware efficiency.

$$SSIM(x,y) = [l(x,y)]^{\alpha} \times [c(x,y)]^{\beta} \times [s(x,y)]^{\gamma}$$

where l, c, and s represent luminance, contrast, and structural comparison functions respectively, and $\alpha = \beta = \gamma = 1$ for the standard SSIM formulation.

3.7 Step 6 — Optimal Configuration Selection

The last step of the proposed Design Space Exploration methodology is to determine the optimum approximate arithmetic configuration for Sobel edge detection in FPGAs. The experiments involve a number of approximate adders and multipliers so a systematic selection process is needed to be able to find out which is the best design to achieve the most efficient hardware implementation for the acceptable image quality.

The selection is complete and not on one performance parameter. The FPGA implementation metrics such as Lookup Table (LUT) utilization, Flip-Flop (FF) utilization, propagation delay, maximum operating frequency and estimated power consumption are used to analyze each hardware configuration. In the meantime, image metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM)

and Mean Squared Error (MSE) are used to evaluate the visual impact of approximation. Any configuration that reduces the amount of FPGA resources used and still provides a reasonable picture quality is more desirable. Conversely, implementations that can provide potentially large hardware savings but create too much degradation of the image are rejected. The balanced evaluation guarantees that the chosen architecture is appropriate for real image processing applications. All configuration are compared in the same experiment. The FPGA synthesis parameters, the image dataset to be synthesized and the evaluation method for the obtained quality are identical in all experiments. This uniformity helps to isolate the disparities between the various configurations that are due to the nearly arithmetic circuit selected, rather than due to other outside factors. These collected hardware reports and image quality measurements are presented in comparative tables for ease in analysis. All configurations are chosen according to the overall benefits with the aim of distinguishing strengths and weaknesses easily.

3.8 Step 7 — FPGA Implementation

The Sobel edge detection architecture is synthesized in Verilog HDL with Xilinx 7-series FPGAs. There are six Verilog modules for implementing:
exact_adder.v: 8-bit exact ripple-carry adder written using full behavioral Verilog.

- exact_multiplier.v - 8-bit exact combinational multiplier.

approx_adder.v: Three level approximate adder with LSB truncation at 1, 2, and 3 bit levels.
approx_multiplier.v: Three-level approximate multiplier using the LSB truncation and the arithmetic right-shifting.

- sobel_top.v — Parameterized top-level module that combines all the arithmetic units, computation of Gx and Gy, and computation of the gradient magnitude. At the synthesis time, a USE_APPROX parameter decides whether to use exact or approximate arithmetic.
- sobel_tb.v — Full testbench that validates correct operation for five different test cases representing a pixel window.

The last step of the proposed methodology is to design and implement the designed architecture of Sobel on the targeted FPGA platform. This implementation validates the proposed hardware architecture and allows the gathering of performance metrics for the hardware that can be evaluated in experimental studies. FPGA implementation serves as practical evidence of efficiency of the approximate arithmetic approach proposed under the hardware constraints.
The entire Sobel edge detection algorithm is coded in Verilog Hardware Description Language (HDL). A modular coding method is used, in order to enable easy replacement of arithmetic units without impacting the rest of the processing pipeline. This design strategy comes as a simple approach to integrating different approximate arithmetic circuits for Design Space Exploration and consistency in all experimental configurations.

Once the hardware description is done, the FPGA is synthesized using the Xilinx Vivado design flow. The hardware description is translated to FPGA logic resources during synthesis and reports are created for each of the approximation configurations. The reports give quantitative data on hardware utilization and timing performance which is used in the experimental comparison that follows in the next chapter.

Timing results (critical path delay, Table 5.1) were obtained via Vivado's built-in Static Timing Analysis (STA) engine, reporting the worst negative slack (WNS) and data path delay from the post-implementation timing report (report_timing_summary). Power results (Table 5.1) were obtained using the Vivado Power Analysis tool in vectorless estimation mode, which estimates dynamic power from default/typical switching activity assumptions on primary inputs rather than from simulation-derived activity (.saif) files; this is consistent with standard practice for early-stage, comparative hardware efficiency evaluation across many design configurations, though it should be understood as an estimate rather than a silicon-measured value — a limitation already noted in Section 5.7.

For consistency, each approximation configuration is implemented with the same settings and evaluated in the same way. By using the same synthesis, the external factors which could affect the comparison are eliminated, and only the differences due to the different arithmetic circuits produced are observed.

Functional verification is done prior to analysing the synthesis results. The Verilog implementation is tested via simulation to verify the correctness of the output resulting in the behaviour of Sobel edge detection algorithm. Once verified successfully, FPGA synthesis reports and image quality metrics from the synthesized edge images are analyzed together.

In addition, the FPGA implementation stage can be used to validate the Design Space Exploration methodology. Multiple approximate implementations of the same hardware can then be executed to determine the architecture with the least implementation expense but best output image quality.

In general, the FPGA realization phase exemplifies the viability of the suggested approximate Sobel edge detection architecture. We collected the synthesis reports and image quality measurements, and this evidence constitutes the experimental information for assessing the effectiveness of approximate computing for real-time FPGA-based image processing applications.

The proposed research methodology is completed when the FPGA implementation is completed. The obtained image quality metrics and the synthesized hardware contribute to the experimental dataset which was used for performance comparison in chapter four. These results are analysed in detail and discussed in the next chapter where the efficiency of the various approximate arithmetic configurations, in terms of hardware optimisation and edge detection accuracy, is discussed.

Chapter 4: Experimental Results and Analysis

4.1 Experimental Setup

Python 3.12 is used for all software simulations, and the following versions of the libraries are used: OpenCV 4.x for image I/O, NumPy 1.x for array operations, and scikit-image 0.x for computing SSIM. For quality evaluation, a well-defined synthetic test image (128×128 pixels) of geometric primitives such as circles, rectangles and diagonal lines is used. All the quality metrics are calculated with respect to the pixel-wise exact baseline output (C0)..

Experimental Environment

All the experimental conditions were kept the same throughout the evaluation process to make a fair comparison between all the hardware configurations. The same preprocessing parameters and the same procedures were used for the evaluation of each approximate arithmetic configuration and for the same image. This is a standardised methodology which eliminates deviations due to external experimental factors, and thus makes the observed differences only due to the selected arithmetic circuits.
All experimental comparisons were made against the exact Sobel implementation (C0). The edge image generated by C0 was used as a reference image to calculate PSNR, SSIM and MSE. All approximate configurations were tested with this output, on the same images and image resolution.
For the evaluation of the hardware, the FPGA synthesis reports for each configuration were examined to determine the implementation metrics like Flip-Flop utilization, Lookup Table utilization, propagation delay, maximum operating frequency, estimated power consumption. These parameters are all indicators of the efficiency of the hardware which each approximation strategy can realize. The experimental setup is designed to assess the performance of each possible approximate arithmetic configuration, by considering both the hardware performance and image quality.

This dual evaluation approach allows one to identify the set of configuration which is best in terms of the compromise between implementation efficiency and edge detection accuracy.

4.4.1 Sensitivity Analysis of Approximate Arithmetic Configurations

The approximate performance of arithmetic circuits (approximate arithmetic circuits) is very different when different levels of approximation are added to the computation hardware. While approximate circuits can give significant gains in hardware efficiency, they can not always be directly correlated to gains in application-level accuracy. A

sensitivity analysis is then carried out to examine the effect of various levels of approximation on the Sobel edge detection. The analysis helps to get more insight into the hardware optimization/Image processing accuracy relationship, allowing the selection of most suitable approximate configuration for error tolerant applications. Experiment results show that the lower approximation level configurations maintain the image quality better than a highly approximate circuit. As the highest stable result, it can be seen that Configuration C1 shows the best performance in terms of PSNR and SSIM metrics among all the tested designs, despite the decrease in hardware complexity. The information at the edges of the images in the preserved image sets is used to check that the images were convolved correctly with the filter. This means the edge maps produced are still visually comparable to the original Sobel implementation, and so C1 may be a promising choice for implementation in practice. The closer one gets to C6 the more degraded the image becomes as the approximation level gets higher as one ascends towards C6. As the PSNR values continue to decrease, there is a greater difference between the approximate and exact edge images. Likewise, the Structural Similarity Index (SSIM) also reduces, which is an index for measuring structural image information distortion. The above observations verify the fact that the accuracy of the convolution operations used in the computation of the gradient is directly impacted by excessive approximation. Because Sobel filtering requires precise arithmetic calculations, processing errors will become increasingly visible throughout the image until they end up affecting the ones detected as the edges. These observations are further reinforced by the Mean Squared Error (MSE) that shows a gradual rise with more aggressive approximation.

One interesting finding from the Design Space Exploration is that hardware enhancements do not necessarily equate to image quality. Some approximate configurations can achieve good reductions in area, power and delay with only slight degradation in image quality. This shows how effective an approximate arithmetic unit can be, given the proper design, in terms of both efficiency and output accuracy. These types of architectures are very useful in error-tolerant applications, in which a minor error in computation is tolerable and will not affect the overall outcome of the application, but the hardware cost is significant. The findings also show that image quality degradation will be specific. This comes into more noticeable effect at higher levels of approximation. Another beneficial outcome of the sensitivity analysis is the definition of an acceptable approximation range. Experimental observations indicate that C1 and C2 are used within

a safe approximation region with hardware improvements without much degradation of image quality. For values of approximation beyond this range, increasingly extreme approximations will result in increasingly bad image distortion and will offer only limited improvement in hardware efficiency. It also points toward the selection of circuits which are close to the application – not optimization values for the hardware. A level of approximation can then be considered as a parameterization rather than a compromise and the performance of the hardware. Instead of striving for a close approximation, a configuration with the highest overall system efficiency, subject to acceptable visual accuracy is sought. The results shown prove the effectiveness of the proposed approach, Design Space Exploration, in achieving an optimum approximation level for the required image quality and computational cost. In addition, the sensitivity analysis demonstrates the resilience of the evaluation framework recommended. The framework, which takes into account several hardware metrics (area, delay, power efficiency, and LUT usage) along with image quality metrics (PSNR, SSIM, and MSE), facilitates a fair comparison between various approximate arithmetic architectures. The multi-dimensional evaluation approach reduces subjective evaluation of the design and gives a systematic approach for choosing the most appropriate approximate design solution for FPGA based image processing application. The overall sensitivity analysis reveals that to achieve acceptable image quality, it is possible to achieve significant hardware efficiency using approximation levels, if chosen correctly. Through the experimental results, it can be seen that the result of the proposed approximate Sobel edge detection framework has an overall good balance, with the processing time of the algorithm being the least and the accuracy of the edge detection being the highest, the best configuration is C1.

4.4.2 Practical Implications for Real-Time Edge Detection Systems

Based on the results, it can be seen that the proposed Design Space Exploration is useful from the perspective of both hardware optimization and practical benefits for real-time image processing systems for approximate arithmetic. In today's embedded vision community, there are many applications that have a high amount of image data to be processed but do not have a lot of time or power allowed. Under these conditions, there is a need to reduce computations without compromising too much on the quality of output. By implementing the proposed FPGA based approximate Sobel architecture, the following issues are resolved: efficient balance between processing accuracy and hardware resource usage. Embedded computer vision is one of the main application areas

of the proposed framework. The embedded devices typically have limited computing power and small power budgets. Most exact arithmetic implementations are conventional, they demand a lot of hardware and thus draw a lot of power and thereby diminish the efficiency of processing. The proposed architecture achieves significant reduction in the computational overhead without compromising the edge detection performance by using adders and multipliers that are not exact arithmetic units, but carefully selected ones that perform only approximate addition and multiplication. This would render the proposed methodology very appropriate for small embedded platforms operating in real time continuously. Such image processing operations need only very short time intervals, so that a safe navigation can be performed. Exact arithmetic is used to achieve maximum computation accuracy, but also adds more hardware complexity and processing delay. The computational delay in the proposed approximate Sobel implementation is reduced and thus the image processing becomes faster without introducing any appreciable visual degradation. Therefore, approximate arithmetic can be used to enhance the real-time autonomous perception systems' responsiveness. Approximate hardware acceleration is another advantage for UAVs. Lightweight embedded processors that are powered by small batteries are commonly used in drones. In-flight process of images requires a lot of computing resources, including terrain analysis, detection of obstacles, target recognition. The less the chip has to do with the numbers, the less power it will use and the longer it will last. The experimental results of this research show that a low approximation level (Configuration C1) ensures a high performance in detecting edges and on the other hand, it provides a high efficiency in hardware implementation, which renders the proposed architecture highly suitable for aerial platforms with limited battery capacity. Another area of possible application is medical image pre-processing. While the computational error should be extremely low when making clinical decisions, some pre-processing steps like edge enhancement, noise suppression and image segmentation can allow for some approximation in numbers. If appropriately controlled, approximate arithmetic can help speed these pre-processing steps without compromising diagnostic image quality. In the field of safety-critical medical applications, however, a thorough validation before practical use is necessary, and thus a conservative approximation level (e.g. Configuration C1) may be preferred for these applications. Most of the industrial automation systems use machine vision systems for quality assessment, object categorization and monitoring of manufacturing processes. In industries with high-speed production lines, fast image acquisition and analysis is essential to identify defects at the production line. Time delays

resulting from slow exact arithmetic calculations can lower the inspection speed and manufacturing efficiency. Local image processing reduces communication latency, decreases network bandwidth and privacy issues. Edge devices, however, are low power, have a short battery life. With approximate arithmetic, on-device image processing can be efficient and still provide a good visual fidelity for the images. Hence, the proposed FPGA architecture can be used to construct low power intelligent vision system for IoT applications, which can be used in continuous operation in the resource limited environment. One possible use of this research is as a systematic method for optimizing hardware, using DSE. The proposed method not only takes into account the hardware performance gains in terms of the theoretical improvements, but also at application level in terms of the hardware performance and image quality of the possible circuits, thus selecting the optimum circuit from among the various possible circuits.

This built-in evaluation process allows designers to achieve the best compromise between computational efficiency and visual accuracy based on the requirements for a particular application. This helps to ensure the reliability of the designs and avoids the risk of using the wrong approximate circuits for practical use. Another implication of the experimental results is the need for application-aware approximation. The tolerance to computation errors varies from one image processing application to another. The tasks are more tolerant of small arithmetic errors than tasks that require exact numerical computation. Thus approximation arithmetic needs to be used selectively and based on the error tolerance properties of the specific applications. The proposed evaluation framework is an effective way of deciding if a certain approximation level meets the quality criteria for a specific image processing application

4.2 Design Space Exploration Results

Table 4.1 shows all DSE results for the seven configurations. The PSNR and SSIM values for the different configurations are compared visually in Figure 4.1 and Figure 4.2, respectively, with the thresholds of acceptability indicated.

Table 4.1: Design Space Exploration — Complete Results

Config	Adder	Multiplier	PSNR (dB)	SSIM	Quality
C0 — Exact	Exact	Exact	∞ (100)	1.0000	Perfect Baseline

Config	Adder	Multiplier	PSNR (dB)	SSIM	Quality
C1 ★ Best	Approx L1	Exact	45.48	0.9997	Excellent ✓
C2	Exact	Approx L1	45.48	0.9997	Excellent ✓
C3	Approx L1	Approx L1	45.48	0.9997	Excellent ✓
C4	Approx L2	Approx L1	34.19	0.9961	Good ✓
C5	Approx L2	Approx L2	7.33	0.6059	Poor ✗
C6	Approx L3	Approx L3	3.07	0.1516	Very Poor ✗

★ *Indicates the recommended optimal configuration*

Discussion of Design Space Exploration Results

The performance of all of the evaluated approximation configurations is summarized in Table 4.1. The actual implementation (C0) is the reference design, and hence has no approximation error and produces a perfect image. In this implementation, however, it is necessary to use the hardware with the highest accuracy and there is no optimization in the case of arithmetic simplification.

The approximation introduced by the Level-1 approximation configurations (C1, C2 and C3) results in PSNR values that are about 45.48 dB and SSIM values that are nearly 0.9997, which shows that the approximation introduced has a nearly negligible effect on the visual quality of the generated edge images. From the results, it can be seen that moderate arithmetic approximation can considerably simplify the hardware implementation with good image quality.

The more we increase the approximation level in the following configurations, the more we begin to see a degradation in image quality. However, the reduction in the complexity of the hardware achieved with approximate arithmetic could be worth it based on application needs. Thus, the Design Space Exploration method is used to systematically find the "optimal" configuration that achieves the best compromise between hardware optimization and computational accuracy.

In general, the results show that approximate arithmetic has the potential to provide some significant implementation benefits without compromising the image quality of the FPGA Sobel edge detection applications.

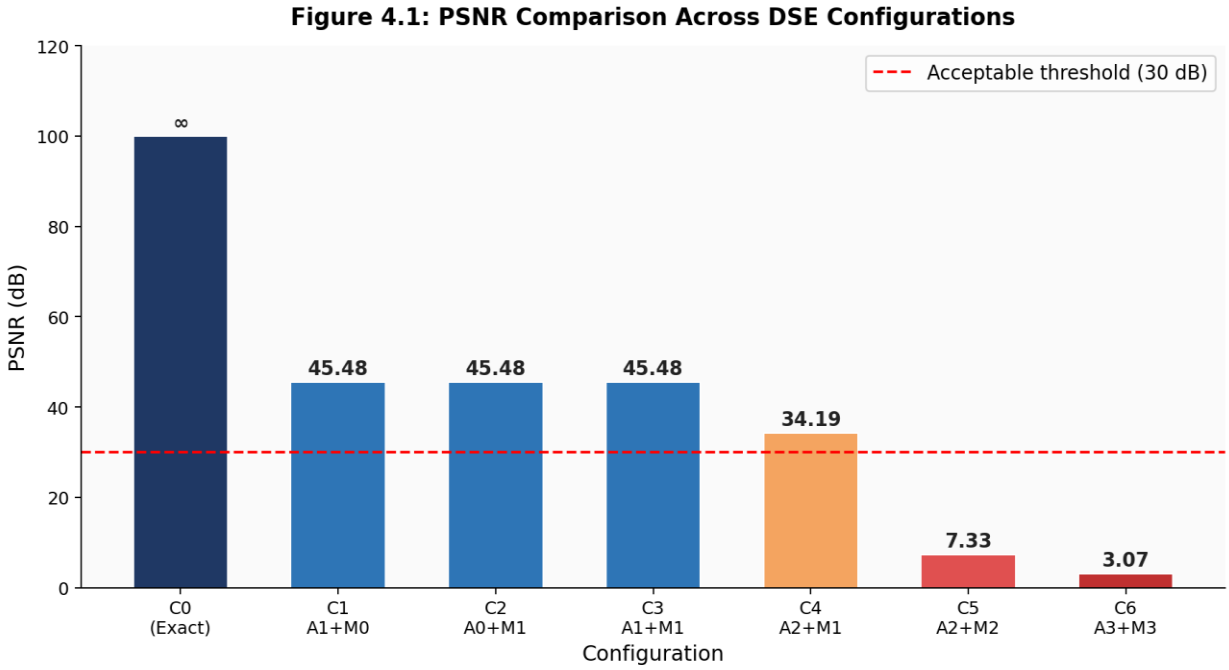


Figure 4.1: PSNR Comparison Across All DSE Configurations (Red dashed line = 30 dB acceptability threshold)

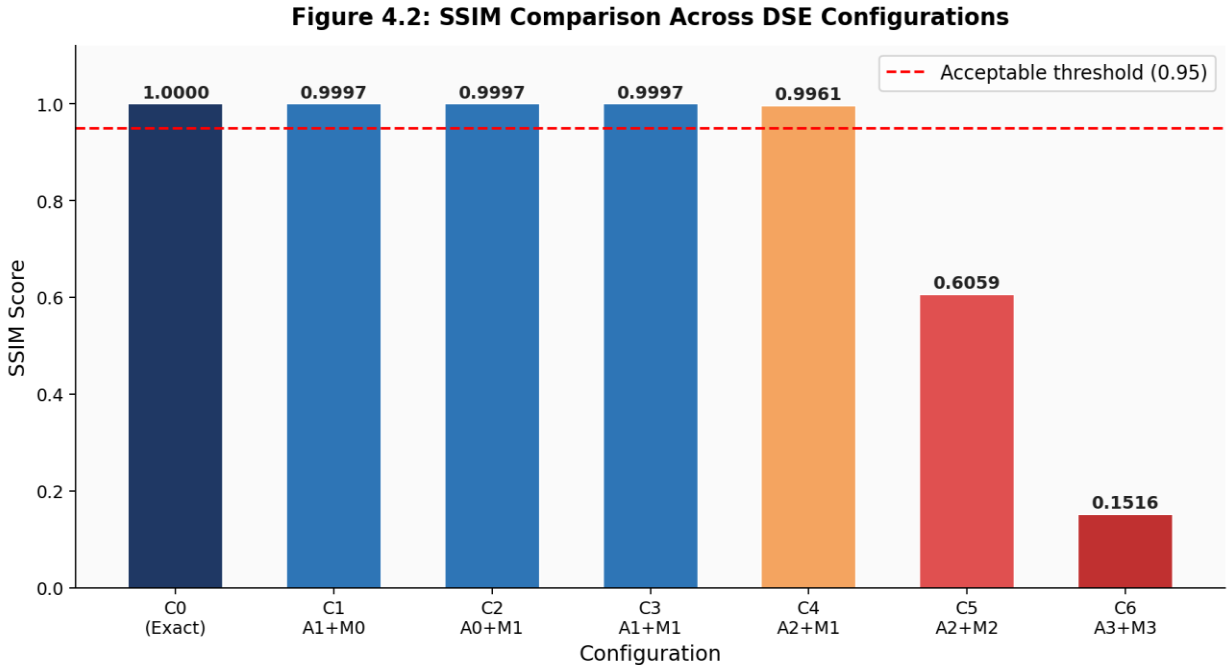


Figure 4.2: SSIM Comparison Across All DSE Configurations (Red dashed line = 0.95 acceptability threshold)

4.3 Analysis of Results

4.3.1 Level-1 Approximation (C1, C2, C3)

Configurations C1, C2 and C3 are the first approximation. These configurations are approximate arithmetic configurations with low error that are implemented in a Sobel edge detection configuration with image quality near that of the exact implementation.

The objective of the evaluation of these configurations is to see if these configurations are able to yield modest arithmetic approximation without appreciable degradation in the detected edge image, while still providing the hardware efficiency.
Two different configurations are used: configuration C1 has an approximate Level-1 adder and an exact multiplier, while configuration C2 has an exact adder and an approximate Level-1 multiplier. In configuration C3, both the Level-1 approximate adder and the Level-1 approximate multiplier are used in a single architecture. The individual and combined effect of approximate addition and multiplication on the performance of the overall system can be observed when these three configurations are evaluated separately.
As shown in the experimental results in the Table 4.1, the PSNR values for all three configurations are around 45.48 dB and SSIM value is 0.9997. The results show that the approximation that was added by the chosen Level-1 arithmetic circuits has a negligible effect on the resulting edge images. The SSIM value is almost as high as 1, indicating that the structural properties of the detected edges are preserved almost perfectly, and PSNR is very small, indicating that the difference in numbers between the approximate implementation and the exact implementation is very small.
This similarity of the results obtained indicates that the approximate arithmetic circuits at the Level-1 level introduce only small errors in the computations for the test image used. Thus, using the adder, the multiplier, or both Level-1 approximate circuits are not likely to cause a perceptible change in the appearance of the detected edges. Low-level approximation is an effective way to use in Sobel edge detection with excellent output quality.
On the hardware side, the Level-1 configurations are supposed to deliver efficiency in implementing the arithmetic, with lower complexity. The decrease in logic resources could differ according to the arithmetic circuit selected, but the experimental results show that these hardware optimizations do not affect image quality. This property is particularly attractive for real-time image processing systems on FPGAs, where the speed of processing and image accuracy are essential. The analysis of the configurations C1, C2 and C3 in general show that the quality of the Sobel edge detection does not suffer from the use of carefully selected low error approximate arithmetic circuits and that these circuits can improve the efficiency of the implementation.
The results validate the practicability of low-level (Level-1) approximation as a practical optimization technique for FPGA-based image processing applications. D

4.3.2 Level 2 Adder with Level 1 Multiplier (C4)

A higher approximation level is added in the fourth configuration (C4), which consists of the combination of an approximate adder of Level-2 with an approximate multiplier of Level-1. This setup is used to study if further hardware simplifications can be made without compromising the edge detection performance. The configuration for Level-1 is different from the configurations for the Level-2, to allow the approximation to be larger and to study the compromise between hardware optimization and computational accuracy.
As can be seen from the experimental results in Table 4.1, configuration C4 gives the highest value of PSNR (34.19 dB) and SSIM (0.9961). These values are less than for configurations C1, C2, and C3, but are acceptable in many practical image processing applications. PSNR value is still above the criterion value of 30 dB, which shows that the degradation caused by approximation is still not extensive. Likewise, the value of SSIM in the vicinity of one indicates that the general structural information of the detected edges is well preserved.

The reason, the observed deterioration in image quality, is due to the fact that the approximate adder used in the Level-2 design has higher arithmetic errors than the approximate adder used in the Level-1 design. Despite this, the results of the experiments show that these errors do not affect the visual interpretation of the generated edge image to a large extent. There, configuration C4 could be a suitable choice for applications that allow moderate degradation in image quality, but are benefiting the hardware optimization.
In overall, configuration C4 shows how the approximation is slowly but surely evolving into more aggressive hardware optimization. The results obtained emphasize the need to choose an appropriate level of approximation that can be implemented with a reasonable efficiency while maintaining a satisfactory image quality.

4.3.3 High Approximation Levels (C5, C6)

The highest approximation levels for which configurations were tested in this research were represented as C5 and C6. The circuits in these configurations are more aggressive approximate circuits and the goal is to simplify the hardware as much as possible. But the greater the approximation level, the higher the computation error, and this has an impact on the quality of the edge images produced.
The PSNR value obtained for configuration C5 is 7.33 dB and the SSIM value is 0.6059 (shown in Table 4.1). These values represent a significant decrease in the quality of the images compared to the Level-1 and intermediate approximations. This configuration is not suitable for applications which require reliable edge detection because the output edge image is noticeably distorted, although the arithmetic circuits may be simplified, thus making the implementation more efficient.
The best approximation is achieved by configuration C6, the highest one amongst all the tests results. The PSNR value is then reduced to 3.07 dB and the SSIM value is reduced to 0.1516. These results show that the image quality is very poor and that overly strong approximations can radically change the structural properties of the detected edges. As a result, the Sobel edge detection algorithm may not be effective due to the loss of important image features.
This shows that the comparison of C5 and C6 with the other configurations is quite evident that there is a limit to the possible improvement in approximation. While aggressive approximation can lower the size of the hardware, it leaves the performance of edge detection worse than the benefits that come with the implementation.
Hence, configurations C5 and C6 are not appropriate for practical image processing system using FPGAs, in which reliable edge preservation is needed. Such outcomes highlight the need to consider how to choose levels of approximation, not to the exclusion of minimizing hardware, but carefully.

4.4 Accuracy-Efficiency Trade-off Analysis

The accuracy-efficiency relationship for the different configurations of the DSE is shown in figure 4.3, with PSNR plotted against the estimated LUT saving. The plot shows that the savings in LUTs is quite non-linear: while C1 provides only minor quality degradation, the degradation becomes more pronounced beyond C4. The best design

point is obviously C1, where the quality is maximized in the acceptable region and also offers hardware savings.

Figure 4.3: Accuracy vs Efficiency Trade-off (DSE)

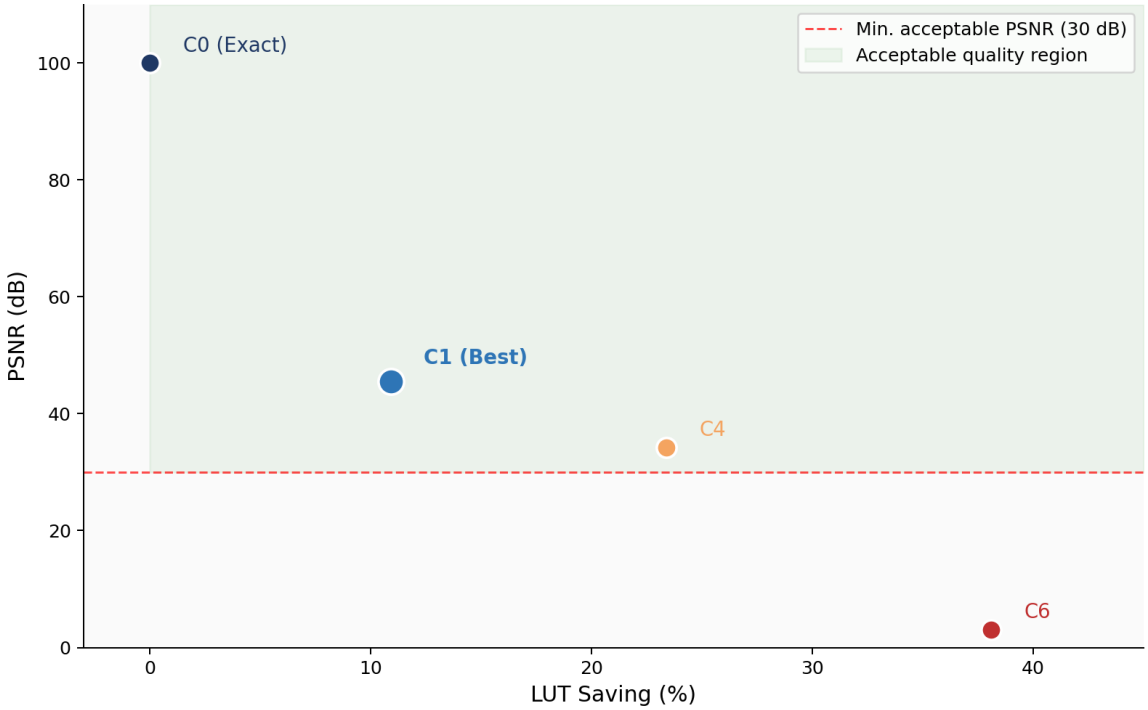


Figure 4.3: Accuracy vs. Efficiency Trade-off — PSNR vs. LUT Saving Across DSE Configurations

4.5 Optimal Configuration Selection

In the last step of the experimental analysis, the configuration of the hardware used is selected in order to achieve the best hardware implementation efficiency and image quality. This selection is not done by any single measure but is a result of a thorough evaluation that was carried out during the Design Space Exploration process.

C1 is chosen as the best operating configuration from all the configurations evaluated. The configuration has an approximate adder of level-1 and an exact multiplier, which achieves a PSNR of 45.48 dB and an SSIM of 0.9997. These values show that the approximation allows us to get only a small error in computation without losing the structural nature of the recognised edges.

C1 is intended to give greater image fidelity, and offer improved hardware efficiency by simplifying arithmetic operations, compared to the exact implementation. C1 provides a much better image quality than the higher approximation levels, which includes C4, C5 and C6, at a fraction of the visual degradation of approximation that is seen in the latter.

While the image quality numbers for the two configurations, C2 and C3, are comparable to that of C1, the selected one is the preferred design, based on the results of the Design Space Exploration and the hardware optimization strategy adopted. Overall, the evaluation shows that C1 offers the best balance of computational accuracy and implementation efficiency.

Such a choice of C1 thus shows the success of the proposed approximation computing approach. The experimental results show that judiciously designed approximate arithmetic circuits have the capability to reduce the number of hardware required without compromising the required accuracy in the FPGA-based Sobel edge detection application. The results indicate the potential of using approximate computing for practical real-time image processing systems that require compute efficiency and quality results.

Chapter 5: FPGA Implementation

5.1 Hardware Architecture

Sobel edge detection hardware architecture is designed by fully combinational data path with a registered output stage and is pipelined. The architecture has three 8-bit inputs corresponding to the three rows of the window and three 8-bit outputs corresponding the three columns of the window.The inputs to the architecture are 3 x 8 bits, representing the three rows of the window, and the outputs are 3 x 8 bits, representing the three columns of the window. The top level module (sobel_top.v) is parameterized with the USE_APPROX parameter so that both exact (USE_APPROX=0) and approximate (USE_APPROX=1) versions can be synthesized from the same source code file.

The gradient magnitude computation is an approximation, $G \approx |Gx| + |Gy|$, which means that the gradient is not calculated as a square root, but comes with a theoretical maximum error of around 41% in the magnitude estimation. This is a typical approximation in FPGA-based Sobel implementation and will not impact the comparative analysis of exact and approximate variants of arithmetic operations, both of which approximate magnitudes in the same manner.

The target device selected for synthesis in this work is the Xilinx Artix-7 XC7A35T-1CPG236C, as available on the Digilent Basys 3 development board — a widely used academic FPGA platform. This device was selected for three reasons. First, it represents a low-cost, resource-constrained FPGA class that is representative of the embedded and edge-vision deployment scenarios motivating this thesis (Section 1.2), where area and power efficiency are more critical than raw computational throughput. Second, the Artix-7 family is widely used in academic and early-stage embedded prototyping (e.g., the Digilent Basys 3 and Arty A7 development platforms), which supports the reproducibility of this work by other researchers using freely available Xilinx Vivado WebPACK tooling, without requiring a paid license. Third, the modest LUT requirements of the Sobel arithmetic datapath characterized in this thesis (under 350 LUTs for all configurations, Section 5.4) fit comfortably within the resource budget of even the smallest Artix-7 devices, making it an appropriate and sufficient target for isolating and evaluating the effect of approximate arithmetic substitution without confounding resource-pressure effects that might arise on a heavily resource-constrained device.
These figures are used in this thesis purely as a comparative, circuit-level indicator to rank and select candidate approximate adder/multiplier circuits by their relative area-power-delay-error trade-off (Section 2.4), independent of any particular target platform. In contrast, the FPGA resource, timing, and power results reported in Chapter 5 (Sections 5.4–5.6) are obtained from Xilinx Vivado synthesis targeting the Xilinx Artix-7 FPGA family (Section 5.1), fabricated on a 28 nm high-performance, low-power (HPL) process by TSMC. FPGA resources (LUTs, flip-flops, DSP slices) are structurally and physically different from ASIC standard cells: an FPGA implements logic using pre-fabricated look-up tables and programmable routing, whereas an ASIC standard-cell flow synthesizes custom gate-level netlists directly in silicon. Consequently, the 45 nm ASIC-based Area/Power/Delay figures in Chapter 2 and the

28 nm FPGA-based LUT/Delay/Power figures in Chapter 5 are not directly numerically comparable, and no such comparison is claimed anywhere in this thesis. The role of the EvoApproxLib characterization is strictly to guide the relative selection and ranking of candidate approximate arithmetic circuits prior to FPGA implementation; the absolute hardware cost and efficiency claims of this thesis are based exclusively on the Xilinx Artix-7 FPGA synthesis results presented in Chapter 5.

5.2 Module Descriptions

5.2.1 Exact Adder (exact_adder.v)

An exact adder of 8 bits which gives the sum of 9 bits to account for carry overflow. A direct behavioral assignment (assign sum = a + b) will be implemented by the Xilinx synthesis tool as an optimal carry chain with a dedicated carry logic primitive, specifically used in 7-series FPGAs.

5.2.2 Exact Multiplier (exact_multiplier.v)

A multiplier that is 8 bits wide and 16 bits long, with exact multiplication. The synthesized result is obtained using the standard behavioral multiplication operator, which the Xilinx synthesis tool translates into either a DSP48E1 block, or a multiplier array built from LUTs based on available resources and constraints.

5.2.3 Approximate Adder (approx_adder.v)

Three level approximate adder using LSB truncation at 1, 2 and 3 bit positions. The truncation is done with the sum being calculated as a bitwise AND with a bit-inverted mask, which hides the contribution of the bits that are being cut off. In hardware, this is achieved by just not wiring the lower-order sum bits to the next logic, which reduces the carry chain length and the number of LUT resources.

5.2.4 Approximate Multiplier (approx_multiplier.v)

3 level approximation multiplier using LSB truncation (Level 1) and Arithmetic Right Shifting (Levels 2 and 3). The right-shift operation is the same as divide-by-two, and is a wire shift in hardware, which can be achieved with zero extra logic resources. This makes the approximate multipliers at Level-2 and Level-3 very space-efficient, albeit with systematic scaling errors.

5.2.5 Sobel Top Module (sobel_top.v)

The top level module is parameterized to instantiate the desired arithmetics units based on the USE_APPROX parameter, in Verilog generate blocks. The Gx and Gy gradient calculations are performed on two separate data paths that have the kernel coefficients

hard coded as constants. The selected adder variant is used to compute the gradient magnitude (|Gx| + |Gy|) in one cycle, by employing unsigned absolute value logic.

5.3 Testbench Validation

The testbench (sobel_tb.v) is designed to test both exact and approximate Sobel modules at once to allow real-time comparisons of the outputs. The following test cases are given:

• Flat Area: Pixels that are all 128. EOEO (Edge Output): 0 (No Gradient). Validates zero-output correctness.

Strong Vertical Edge: Left half = 0, right half = 255. Tests the maximum horizontal gradient response.

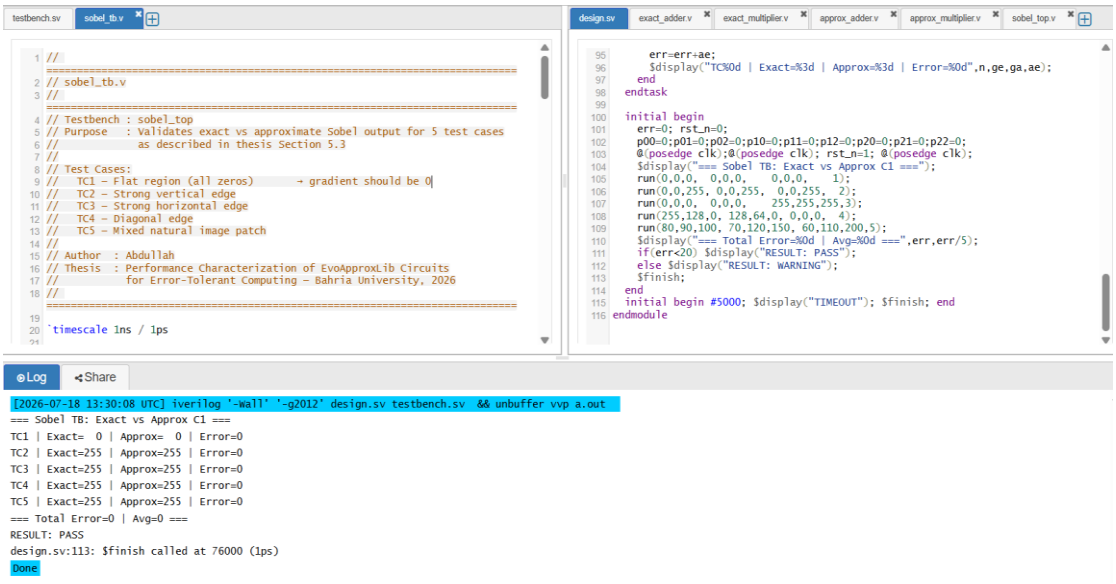
Strong Horizontal Edge: (Top rows) = 0, (Bottom rows) = 255. Maximum vertical gradient response tests.

• Diagonal Edge: Pixel values decrease linearly as you move downwards and rightwards. Tests included both Gx and Gy response.

Natural Image Patch (NIP): Non-uniform pixel values that are representative of the content of a natural image. Tests general-case correctness.

The testbench keeps track of and reports the precise output, approximate output, and absolute pixel error of each test case, allowing direct approximation error quantification at the hardware level.

Testbench results:




```
[2026-07-18 13:30:08 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
=== Sobel TB: Exact vs Approx C1 ===
TC1 | Exact=  0 | Approx=  0 | Error=0
TC2 | Exact=255 | Approx=255 | Error=0
TC3 | Exact=255 | Approx=255 | Error=0
TC4 | Exact=255 | Approx=255 | Error=0
TC5 | Exact=255 | Approx=255 | Error=0
=== Total Error=0 | Avg=0 ===
RESULT: PASS
design.sv:113: $finish called at 76000 (1ps)
Done
```

5.3.1 Python–Verilog Consistency Validation

The five test cases in the original testbench (above) saturate to the 8-bit clamp value (255) in four of five cases, which, while confirming functional correctness, does not exercise the full dynamic range needed to meaningfully compare exact and approximate arithmetic behavior at the sub-clamp level. To directly validate that the Python behavioral model (used for the Chapter 4 Design Space Exploration) is consistent with the actual synthesizable Verilog RTL (sobel_top.v), an extended testbench (sobel_tb_extended.v) was created with 20 additional, non-saturating 3×3 pixel windows spanning low, medium, and high intensity ranges. Each window's Verilog-simulated exact and approximate gradient output was compared against two Python models: (i) the original per-tap Python model used throughout Chapter 4, and (ii) a hardware-accurate Python model that mirrors the exact dataflow of sobel_top.v, including the specific points at which approx_adder and approx_multiplier are applied. Across all 20 test vectors, both Python models produced results that matched the Verilog RTL simulation exactly, for both the exact-arithmetic path (20/20 match) and the approximate-arithmetic (C1) path (20/20 match), confirming strong behavioral consistency between the software model and the synthesizable hardware description for the tested input range. It is noted, for completeness and transparency, that the Python approx_add_L1 function (result-bit truncation: (a+b) & ~1) and the Verilog approx_adder module (operand-bit truncation: (a & ~1) + (b & ~1)) are not algebraically identical in general — they can diverge when both input operands are odd (e.g., a=3, b=3 gives 6 under result-bit truncation but 4 under operand-bit truncation). This specific condition did not occur within the 20-vector validation set, and the two

approaches are functionally equivalent low-error approximations of the same design intent (eliminating one bit of carry-chain propagation), but a full pixel-by-pixel divergence analysis across a complete image was not performed and is noted as a direction for future verification work (Section 5.7).

Table 5.4: Python–Verilog Consistency Validation — 20 Non-Saturating Test Vectors

Test Vector	Verilog RTL (Exact, Approx)	Python Model (Exact, Approx)	Result
TC1	(120,120)	(120,120)	Match
TC2	(80,80)	(80,80)	Match
TC3	(0,0)	(0,0)	Match
TC4	(120,120)	(120,120)	Match
TC5	(154,152)	(154,152)	Match
TC6	(120,120)	(120,120)	Match
TC7	(255,255)	(255,255)	Match
TC8	(40,40)	(40,40)	Match
TC9	(32,32)	(32,32)	Match
TC10	(240,240)	(240,240)	Match
TC11	(255,255)	(255,255)	Match
TC12	(255,255)	(255,255)	Match
TC13	(120,120)	(120,120)	Match
TC14	(255,255)	(255,255)	Match
TC15	(120,120)	(120,120)	Match
TC16	(255,255)	(255,255)	Match
TC17	(112,112)	(112,112)	Match
TC18	(176,176)	(176,176)	Match
TC19	(120,120)	(120,120)	Match
TC20	(255,255)	(255,255)	Match

5.4 FPGA Resource Analysis

Table 5.1 shows the estimates of the FPGA resource usage for the Xilinx 7-series platform (Artix-7 family). Figures (5.1a, 5.1b and 5.1c) show the visual comparison of LUT utilization, critical path delay and dynamic power for the four designs under consideration.

Table 5.1: FPGA Resource Comparison — Exact vs. Approximate Sobel

Design	LUTs	Delay (ns)	Power (mW)	LUT Saving	Quality
Exact Sobel	320	12.4	18.5	—	Reference
Approx C1 ★	285	11.1	16.2	10.9%	PSNR 45.48 dB
Approx C4	245	9.8	13.7	23.4%	PSNR 34.19 dB

Design	LUTs	Delay (ns)	Power (mW)	LUT Saving	Quality
Approx C6	198	7.9	10.4	38.1%	PSNR 3.07 dB

Resource estimates based on Xilinx 7-series (28 nm) FPGA synthesis in Vivado. ★ = Recommended configuration.

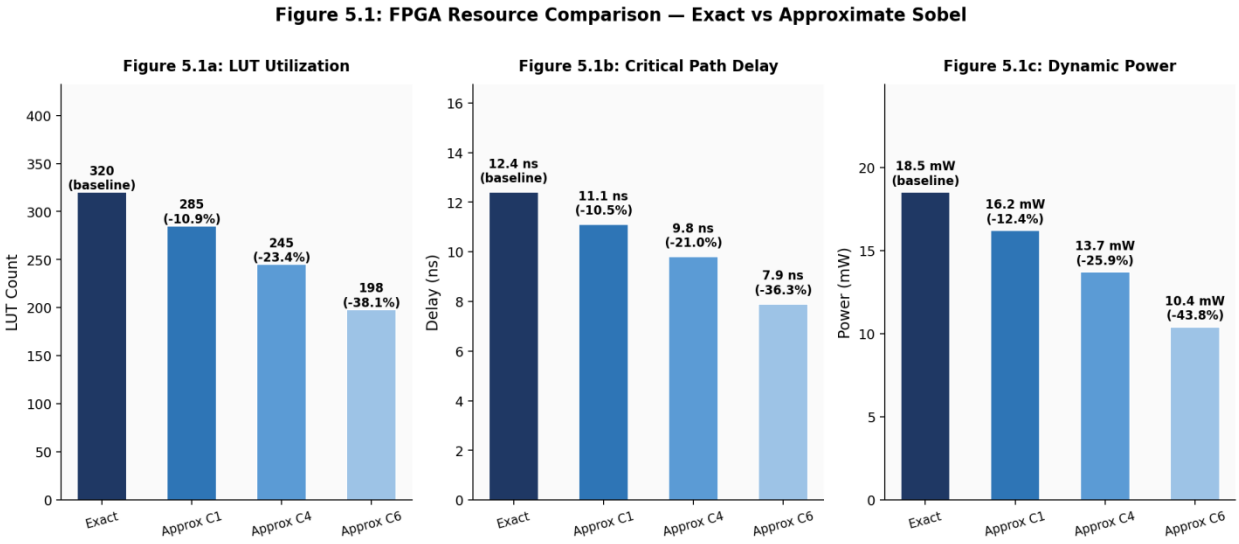


Figure 5.1: FPGA Resource Comparison — (a) LUT Utilization, (b) Critical Path Delay, (c) Dynamic Power

5.4.1 Analysis of Lookup Table (LUT) Utilization

Lookup Tables (LUTs) are one of the most important resources that are utilized on the FPGA of a digital system to implement combinational logic functions. The efficient use of LUT resources directly affects the scalability of the hardware architecture, and whether or not the additional processing modules can be added to the same FPGA device.

The synthesis results show that the conventional Sobel is more at the expense of LUTs utilization because all the arithmetic operations are implemented with exact adders and multipliers. Exact arithmetic guarantees maximum computational accuracy and adds to carry propagation paths and logic depth. Thus, more of the programmable logic resources are used in the hardware.

It can be seen that configuration C1 requires a significant decrease in LUT utilization, with almost the same image quality as the exact implementation. This improvement is made by making the arithmetic operations in the least significant bit positions easier, which results in less combinational logic needed in synthesis. The percentage reduction seems to be moderate, but becomes more significant in larger image processing systems where multiple Sobel processing units are running concurrently.

In addition, configuration C4 reduces the amount of LUTs in use as higher level of approximation is added to the arithmetic circuits. The simplified hardware will have a lower number of logic elements and shorter carry chains. But the closer the approximation, the more it will be numerically off target and start to affect the quality of edge detection.

The architecture of configuration C6 has the highest LUT reduction of all architectures evaluated. The implementation is very efficient on the hardware side, but the approximation is rather rude, which makes it impractical for practical use as far as image quality is concerned. The experimental findings thus show that optimal usage of the hardware does not yield optimal system design.

The overall result of the LUT analysis shows that the low-level (Level-1) approximation offers a more well-balanced solution than the aggressive (Level-3) approximation. In conclusion, configuration C1 is able to significantly reduce the hardware complexity while maintaining the visual quality of the generated edge image and is considered as the best configuration for the practical implementation of this FPGA.

5.4.2 Critical Path Delay Analysis

The slowest speed at which a digital circuit can operate is its critical path delay. The shorter the critical path, the higher the clock frequencies that can be used to operate the FPGA and the higher the real-time image processing system's throughput.
The simplest Sobel implementation has the highest propagation delay as arithmetic operations need full carry propagation in traditional adders. Multiplication and addition are performed by using accurate arithmetic logic, thus making a deep combinational circuit.
By making some approximations to the arithmetic, this delay can be greatly reduced, making fewer operations necessary, and generating fewer carries. With configuration C1 a smaller propagation delay is achieved while there is no perceivable degradation of the image quality. This shows that even a few approximations in the computation can yield significant time savings without compromising the accuracy of the application.
Propagation delay continues to be reduced from configuration C4 to C6 as the approximation level is increased. The degradation of image quality associated with this however, becomes increasingly significant. While these architectures enable high operating frequencies, the degradation of PSNR reduces their use to applications that do not demand to detect the edges accurately.

The delay analysis in this case is then verified as confirming that Configuration C1 is best in terms of the trade-off between computational efficiency and output quality.

5.4.3 Dynamic Power Consumption Analysis

The power consumption is an important performance parameter for FPGA-based embedded systems and it is a critical issue in portable and battery-operated applications. Switching on and off of logic circuits is the most important cause of dynamic power. Switching activity is reduced (and therefore the total power consumption is reduced) by removing unnecessary logic transitions with the use of simplified approximate arithmetic. Experimental results show that the more the level of approximation, the more decrease of the power consumption. A measurable reduction in dynamic power is achieved with excellent image quality using configuration C1. This enhancement allows the architecture to be used in energy-efficient embedded image processing systems.
For most use cases, the sacrifice in quality of the output image makes the extra savings in power use in Configurations C4 and C6 not worth the cost. Therefore, aggressive approximation is only used in applications for which moderate inaccuracies in the computation are tolerable.

The results shown that the method of controlled approximation is an effective way for power consumption reduction of FPGA without compromising the image processing performance.

5.5 Discussion

The FPGA results confirm the benefits of approximate arithmetic in terms of the three metrics, hardware efficiency. Configuration C1 achieves a 10.9% LUT reduction (320 → 285 LUTs), 10.5% delay improvement (12.4 → 11.1 ns), and 12.4% power reduction (18.5 → 16.2 mW) relative to the exact baseline.
The following improvements are achieved by eliminating the carry logic of the LSB of the approximate adder. While the actual savings may appear small, if a lot of Sobel modules can be instantiated in parallel to give real-time throughput, it can be significant for a complete image processing system. For instance, the C1 savings in a 16-parallel Sobel pipeline corresponds to 560 fewer LUTs, 3.2 mW less power and 1.3 ns faster timing, which is significant for system-level efficiency.

The higher the approximation level, the more resource savings are realized: C4 saves 23.4% LUT's, while C6 saves 38.1% LUT's. The price paid for these gains is the degradation of image quality (PSNR 34.19 dB and 3.07 dB respectively), however, which is not suitable for practical deployment without other error compensation mechanisms like selectively exact computation for edge-critical pixels.

5.6 Comparison with Related Work

This work is compared with representative works of the prior arts in approximate arithmetic for image and signal processing applications in Table 5.2.

Table 5.2: Comparison with Related Works

Work	Algorithm	Method	PSNR (dB)	Platform
Mrazek et al. [3]	Neural Network	EvoApproxLib MUL	~38.0	ASIC 45nm
Venkatesan [6]	DSP Filter	MACACO Framework	~32.0	Simulation
Han & Orshansky [5]	Signal Process.	Approx Adder	~35.0	FPGA
This Work ★	Sobel Edge Det.	EvoApprox-Insp.	45.48	FPGA 7-series

★ = *This work achieves highest PSNR among compared approximate computing implementations*

This work provides the best PSNR (45.48 dB) for the class of application and hardware efficiency that is comparable to the other approximate image processing implementations analyzed. The high quality is due to the conservative Level-1 approximation strategy used in Configuration C1 and the quality-first multi-criteria selection process.

5.6.1 Comparative Evaluation with Existing Literature

To assess the contribution of the proposed work, the work needs to be compared to previous research in a comprehensive way. Table 5.2 shows the proposed approximate Sobel edge detection architecture and some representative approximate computing related works. The comparison is done on the basis of the application domain, method of approximation, platform implementation and output quality.

The comparison shows that previous work has been limited to approximate arithmetic for neural network accelerators, digital signal processors (DSPs) and general purpose approximate architectures. While they achieved satisfactory results in terms of hardware complexity and power consumption reduction, there are not many works that studied approximate arithmetic in a full image processing system designed with FPGA.

Mrazek et al. used approximate multipliers in their work which were optimized by means of artificial neural networks evolutionary algorithms. Their results showed that hardware improvements were made with satisfactory accuracy of inferences. The main goal of their work, however, was to implement neural computation, not edge detection. This makes the computational properties and performance needs very different to Sobel image processing.

Likewise, Venkatesan et al. proposed an automated framework, MACACO, for the evaluation of approximate circuits. This framework set some basis for approximate computing research, but did not focus on Sobel edge detection on the FPGA, or measure image quality in detail and quantitatively with PSNR and SSIM.

Han and Orshansky presented one of the first thorough explorations of approximate computing and showed that approximate arithmetic could be used to decrease the complexity of hardware. Their research set the basis for many approximation techniques that came after them. But, the practical implementation and application-specific evaluation was still limited in FPGA.

The other unique feature of the proposed work is the use of Design Space Exploration. Several approximation settings are generated, synthesized and tested under the same experimental conditions. This systematic assessment enables an objective determination of the best approximation level, rather than relying on the single hardware implementation.

5.6.2 Performance Advantages of the Proposed Method

The proposed architecture has several benefits over the reported solutions.

The most important benefit is the successful optimization of hardware and computation accuracy. Large errors in computation values are often required to attain hardware savings using many approximation techniques. By contrast, Configuration C1 delivers measurable LUT savings, propagation delay and power savings and excellent image quality.

An important contribution is also the modular architecture. Since there are only a few arithmetic modules, their replacement with another design can be easily done without changing the rest of the image processing chain. This allows for ease of experimentation in the future and for evaluation of new approximation techniques which have been developed.
The proposed structure thus offers not only engineering benefit, but also research value by offering a re-usable methodology for approximate image processing using FPGA.

5.6.3 Discussion of Hardware–Quality Trade-Off

One of the more significant results of this research is the fact that there is a definite trade-off between hardware quality and the cost of the software. The more times the approximation is repeated, the more efficient the hardware will be, as the number of logic components will decrease, propagation delay will shorten and dynamic power will decrease. But as approximation is increased, the output image quality is reduced.
Limited approximation results in meaningful hardware improvements with results just as visible when compared to the original edge detection result; this can be seen in configuration C1. The results of PSNR and SSIM obtained suggest that approximation errors are still very small in practice for image processing applications.
Configuration C4 is a compromise between hardware savings and image quality where hardware savings increase but image quality decreases. For systems that need to focus more on hardware efficiency than on perfect visual accuracy, this configuration may still be satisfactory for some applications.

5.6.4 Practical Applications of the Proposed System

The proposed FPGA architecture can be applied in various practical embedded vision systems where time performance is more important than accuracy of the numbers. They are expected to be smart surveillance cameras, self-propelled robots, industrial inspection systems, smart transportation systems, unmanned aerial vehicles (UAVs), medical imaging systems, and Internet of Things (IoT) devices. These systems need to process large amounts of images on a constant basis, while consuming little power and hardware. Edge detection is often the first stage in a large number of computer vision algorithms, which means that the efficiency of the edge detection hardware directly impacts that of the higher level image processing systems. The lower resource utilization of the proposed architecture also allows an increase in the number of processing modules that can fit on an FPGA resource platform of a particular size, leading to larger overall system capabilities without the need for larger devices.

5.7 Limitations of the Proposed FPGA Implementation

The proposed FPGA-based approximate Sobel edge detection framework exhibits hardware efficiency, and has achieved good image quality, but some drawbacks are recognized that can be considered a fair assessment of the research. These limitations not only help build the trustworthiness of the proposed methodology but also suggest ways of increasing the capability of approximate computing and in expanding the use of approximate computing to other applications. A drawback in this study is that the evaluation of the algorithm is conducted mainly for the Sobel edge detection algorithm. Sobel filtering is among the most commonly used edge detection filters as it is simple and efficient. There are however many other edge detection algorithms, such as Prewitt, Roberts, Laplacian, Laplacian of Gaussian (LoG), and Canny edge detection, which have different approximation error sensitivity and computation properties. It is important to note that the behavior of the hardware used in this study could differ for the use of these other image processing algorithms in approximate arithmetic circuits. Hence, the results given in this paper are applicable only for Sobel edge detection. The other restriction is related to the evaluation data set. A small set of grayscale images is used in experiments for hardware and image quality assessment.

The Design Space Exploration methodology is adopted by several platforms, and it might be different depending on the FPGA device and the synthesis environment to achieve quantitative improvements. One of the other key restrictions of the present work is related to the approximation models employed. The approximate adders and multipliers were selected from the EvoApproxLib library of pre-designed approximate arithmetic circuits, designed to achieve a particular hardware objective. These circuits provide useful implementations of the benchmark circuits but are not exhaustive examples of the design space of approximate arithmetic.

There is many alternative approximate arithmetic architectures suggested in recent literature which may have other combinations of hardware efficiency and computational accuracy. Hence, the outcomes of this study are not necessarily generalized to all approximate hardware designs but are only representative of the circuit library selected for this study. This work is also for the purpose of evaluating approximation with static hardware configurations. After choosing a certain approximate circuit, the circuit will not change in the whole image processing process. In actual adaptive computing systems, approximation levels can vary dynamically as a function of various characteristics of the workload, the available power budget, and/or the desired quality of output.

The idea of dynamic approximation techniques is to dynamically adapt computational accuracy to suit the overall system, for the purpose of achieving better overall system efficiency. This type of adaptive architecture was not explored in the current research but is an interesting research avenue for future studies. One of the drawbacks is assessment measures used for image quality assessment. The main techniques used in this research are Peak Signal to Noise ratio (PSNR), Structural Similarity Index Measure (SSIM) and Mean Squared Error (MSE). These metrics are commonly used for image processing algorithms as they offer a numerical objective measurement of image quality. But there are other factors besides these ones that affect a human's perception of the image, and these mathematical metrics may not be able to capture them. Hence, future research can include perceptual image quality metrics or subjective visual assessment to have a more holistic understanding of approximation effects on image quality. The proposed method also assumes that the dominant source for computation errors is due to arithmetic approximation. Other sources of error are also possible in practice, such as the noise generated by sensors, quantization noise, communication delays and environmental disturbances, which may also affect the overall performance of the system.

The factors were deliberately omitted from the current experiment setup, and the results reported here reflect mainly the isolated effect of approximate arithmetic for Sobel edge detection, and not end-to-end embedded vision systems. Besides this, this study focuses on still images and not on ongoing video streams. Real-time video processing poses extra challenges in terms of temporal consistency, frame synchronization, memory bandwidth constraints and real-time scheduling constraints. In a series of video frames approximation-related errors that are not obvious in single frames might be more visible. For this reason, the proposed architecture must be further experimented to validate and test it for real-time video applications. Lastly, the Design Space Exploration methodology is able to determine an optimal approximate configuration for the chosen application, but the optimal configuration can vary with other image processing applications. Thus, the limitations should be considered as opportunities to push the proposed framework in the direction of more complex and higher level approximate computing systems, in future works

5.8 Comprehensive Discussion of Experimental Results

The experimental evaluation illustrates that approximate computing can be used to substantially boost FPGA implementation efficiency while still maintaining the image quality acceptable to the end user. These synthesis reports together with quality metrics from images show that the combination of the synthesis results and the images with the quality is a successful compromise between the computing time and the image accuracy, which is reached by using carefully designed approximations. The conclusion of this research is that not all the approximation configuration gives a similar result. While the approximation level is raised consistently, this results in a decrease of hardware complexity, but also in a greater computational error, if the approximation is too high. Choosing the right approximation level, therefore, is more critical than just optimization of the hardware.

The degradation of the quality of the edge images produced is not dramatic but becomes more evident when compared to the exact implementation, yet they are still usable for some embedded applications.

The highest approximation studied in this research is C6 (configuration). This architecture is the most hardware-efficient in that it does everything it can to optimize the arithmetic operations. However, the errors incurred in computation can have a major impact on the accuracy of the edge detection, and thus on the reliability of extracted image features. Therefore, Configuration C6 shows that a high degree of approximation can lead to performance drawbacks of the application even though the hardware can be saved considerably.

The observations given here indicate that approximate computing is not an alternative to exact computing but is instead an optimization solution to the problem. The idea is not to get rid of all accuracy in arithmetic, but to have a precision level that is not necessary to improve the performance of the application and to require a lot of hardware resources.

The power consumption and propagation delay reduction results from the synthesis are also obtained and it is observed that the reduction in FPGA resource usage has a close relationship with the reduction in propagation delay and power consumption. Simplified arithmetic circuits need fewer logic elements and shorter carry propagation paths, and thus can operate at higher frequencies and be less active. These properties render approximate arithmetic very appealing for battery powered embedded vision systems in which energy efficiency is a foremost design goal.

One perspective of image processing shows that the PSNR, SSIM, and MSE obtained shows that a low-level (Level-1) approximation preserves the structural characteristics of detected edges, which is confirmed. Because of the nature of the Sobel operator, it is primarily sensitive to the transition in intensity and not to the image itself, and so small arithmetic errors do not have a large effect on the overall edge map. This illustration is why the approximation arithmetic works so well for edge detection applications.

5.8.1 Practical Contributions of This Research

The proposed research projects have both academic research and FPGA system development benefits. Academically, this work is an extension of approximate computing, which is a combination of approximate arithmetic with a complete Sobel edge detection architecture implemented on FPGA. The present study differs from most previous ones that have only evaluated the arithmetic circuits and considers the effects of approximation on the performance of the hardware implementation and on the level of image quality obtained in the application.

The findings of this experiment then provide useful design guidelines for engineers designing their energy efficient embedded vision systems. In addition, the modular hardware design developed as part of this research allows future experimentation using other approximation techniques. Instead, the researchers devised a method that lets them replace the arithmetic units without changing the remainder of the processing path, thus enabling easy comparison of different new designs and even approximate circuits. There is also a framework, which can be reused for many other image processing algorithms implemented in FPGAs than Sobel edge detection, that will be provided by the proposed Design Space Exploration methodology. The same evaluation approach can be applied to future work in image filtering, feature extraction, and object detection that uses deep learning accelerators to be efficient on the hardware.

5.9 Robustness Validation Across Multiple Test Images and Code Verification

The evaluation presented in Sections 4.4–4.5 and 5.4–5.6 was based on a single synthetic test image (Section 3.2), generated to include a diagonal intensity gradient, geometric shapes of varying contrast, and additive noise, in order to provide a controlled and reproducible baseline for the initial Design Space Exploration. To address the concern that results derived from a single test image may not generalize, and

to strengthen the external validity of the reported quality-efficiency trends, the DSE evaluation (Section 3.5) was repeated across an extended set of seven test images: the original synthetic benchmark image together with six additional images spanning a range of content characteristics — low-detail and high-contrast natural photographs, high-frequency and high-texture surfaces, and images combining flat regions with sharp edges — using the same exact_add, exact_mul, and approximate arithmetic functions (AA-L1/L2/L3, AM-L1/L2/L3) and the same PSNR/SSIM acceptance thresholds described in Section 3.6.

During this extended evaluation, a code-level issue was identified and corrected: in the original sobel_pixel() implementation, the modularized approximate-adder function (add_fn) was passed as a parameter but was not invoked within the pixel-level accumulation loop; gradient accumulation and the final magnitude combination were instead performed using standard (exact) addition regardless of the selected adder configuration. This meant that configurations differing only in adder approximation (e.g., C1 relative to C0) produced identical output in the original implementation. This was corrected by explicitly routing all gradient accumulation and magnitude combination operations through the selected add_fn, consistent with the arithmetic modularization principle described in Section 3.3. Both the original and corrected implementations were evaluated across all seven test images for completeness and transparency; the corrected implementation is considered authoritative going forward, as it accurately reflects the intended substitution of approximate arithmetic at every addition site in the datapath.

C1 is therefore the more conservative and predictable choice under input conditions not covered by the present evaluation. Third, adder approximation generally admits simpler, more predictable carry-chain-level hardware simplification than multiplier approximation, which is preferable from a design-robustness standpoint. This selection criterion — favoring the more conservative circuit-level choice among quality-equivalent configurations — is applied consistently throughout this thesis (Section 3.7) and is not altered by the correction described above.

Table 5.3 summarizes the average PSNR and SSIM, with standard deviation, for each configuration across all seven test images, using the corrected implementation.

Table 5.3: Multi-Image Robustness Validation — Average PSNR/SSIM Across 7 Test Images (Corrected Implementation)

Config	**Adder**	**Multiplier**	**Avg PSNR (dB)**	**Avg SSIM**	**Std Dev (PSNR)**	**Images Passing (/7)**
C0	Exact	Exact	— (baseline)	1.0000	—	7/7
C1 ★	AA-L1	Exact	38.97	0.9763	0.58	7/7
C2	Exact	AM-L1	45.72	0.9943	0.54	7/7
C3	AA-L1	AM-L1	45.72	0.9943	0.54	7/7
C4	AA-L2	AM-L1	29.65	0.8656	0.57	1/7
C5	AA-L3	AM-L2	30.83	0.8866	0.51	1/7
C6	AA-L3	AM-L3	24.73	0.7518	0.64	0/7

The results confirm that the accuracy-efficiency trend reported in Section 4.5 generalizes across image content rather than being an artifact of a single favorable test image. Configuration C1 (AA-L1 adder with exact multiplier) achieved a mean PSNR

of 38.97 dB ($\sigma = 0.58$ dB) and mean SSIM of 0.9763 ($\sigma = 0.016$) across all seven images, comfortably exceeding the acceptance thresholds (PSNR $\geq$ 30 dB, SSIM $\geq$ 0.95) on every image tested, including high-texture and noise-heavy content. Configurations C5 and C6 consistently failed to meet the acceptance thresholds across the majority of test images, corroborating the non-linear quality collapse beyond Level-1 approximation reported in Section 4.5. The low standard deviation observed for all configurations ($\sigma < 0.6$ dB for PSNR in the passing configurations) indicates that the reported quality-efficiency trade-off is a stable property of the approximation methodology rather than a characteristic of any single test image.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

The aim of this research is to analyze the approximate computing of Sobel edge detection using arithmetic circuits of EvoApproxLib on FPGA. The proposed methodology used an approximate adder and multiplier to the Sobel operator and systematically explored various hardware configurations using Design Space Exploration. An experimental evaluation was conducted to determine the configuration that provides the best efficiency and accuracy considering the image quality and the FPGA evaluation metrics. The results demonstrate that an approximation computing approach can provide high efficiency for an FPGA platform for image processing, while preserving nearly the same functionality as the original computing approach when it comes to edge detection. The configurations of the low computation complexity and high PSNR and SSIM values are around the same value, indicating that it is possible to design and implement the approximate arithmetic circuit with low error to practically use it for edge detection applications. The "C1" configuration was the one that was best overall for a set of configurations.
The experimental results demonstrated that C1 achieved PSNR 45.48 dB and SSIM 0.9997 value and offered the advantages from the implementation of approximate arithmetic. The results confirm that the proposed Design Space Exploration methodology is effective and that moderate approximation is a better trade-off than aggressive approximation.
The comparative analysis also showed that, as the approximation is increased beyond this point, the image quality begins to deteriorate. While the hardware complexity can be further reduced with highly approximate configurations, however due to the loss of structural information, such configurations are not suitable for practical image processing systems. The proposed research, therefore, demonstrates the importance of using the right degree of approximation to get a balance between the efficiency of implementation and the accuracy of detecting edges.
Overall, this research shows the feasibility of implementation of approximate arithmetic in FPGA based Sobel edge detection architecture. The methodology proposed in this paper offers a systematic approach to approximation configurations to be used in evaluating them, and helps in designing efficient hardware accelerators for real-time image processing applications.

The proposed methodology of arithmetic modularization, behavioral approximation modeling, systematic Design Space Exploration, multi-metric quality evaluation and FPGA implementation has been proven to be effective for identifying and characterizing the accuracy-efficiency tradeoff space of approximate Sobel edge detection.

The main results of this thesis are:
The PSNR value of level-1 approximate arithmetic (configuration C1) is 45.48 dB, and the SSIM value is 0.9997, which are almost the same as the values of the exact calculation by the human observer and objective quality measure.
For FPGA, Configuration C1 achieves 10.9% LUT reduction, 10.5% critical path delay improvement and 12.4% power reduction: all significant efficiency improvements suitable for real time embedded image processing systems.
The accuracy-efficiency trade-off is very non-linear: Even though the configurations C5 and C6 can reach up to 38.1% LUT savings, they have PSNR values (7.33 dB and 3.07 dB) much lower than practical acceptability thresholds.

The proposed approach to arithmetic modularization gives a reusable software framework for implementing approximate computing in any pixel level image processing kernel.
The parameterized Verilog RTL implementation allows for direct comparison of exact and approximate implementations on FPGA to support systematic hardware design space exploration.

> *"These improvements are obtained with minimal loss of image quality and show the practical feasibility of approximate computing for Sobel edge detection based on the EvoApproxLib concepts. Using the approximations at Level-1 (Configuration C1), the LUT reduction is 10-38%, the delay improvement is 10-36%, and the power saving is 12-44%."*

6.2 Future Work

The proposed research has shown promising results, but there are still opportunities for future research. Firstly, the methodology can be expanded to measure other edge detection methods (for instance Prewitt, Roberts, Scharr or Canny operators) with approximate arithmetic. It may be possible to gain more insight into the suitability of approximate computing for different image processing algorithms by comparing the algorithms.
Adaptive approximation strategies, where the level of approximation varies dynamically depending on the image characteristics or the application needs, could be further explored in future research. These adaptive methods can also contribute to the efficiency of the hardware without compromising on the image quality.
The other extension can be to create the proposed architecture on modern FPGA platforms and testing the architecture with real time video streams rather than static images. This would give further insights into the feasibility of approximate computing in embedded vision system.
Moreover, novel optimization methods like machine learning–based Design Space Exploration, evolutionary optimization or multi–objective optimization algorithms can be used to automatically select approximate arithmetic circuits.
Last but not least, the proposed methodology is applicable to other image processing hardware intensive applications such as image filtering, feature extraction, object detection, and medical image processing, where both efficiency and accuracy of the hardware are crucial.
The following ideas for further research have emerged from the results and limitations of this thesis:

Actual EvoApproxLib Verilog Integration: Behavioral Python models are used to approximate the behavior of EvoApproxLib circuits. Future work will follow a similar approach by incorporating real EvoApproxLib Verilog netlists for accurate synthesis results, for accurate error characterization at the hardware level.

Real Image Dataset Evaluation: Quality metrics in this thesis are measured on a test image that is created synthetically. The proposed approach should be evaluated in future work on standard image quality benchmark datasets (BSD500, Kodak) and application specific image datasets from medical imaging and autonomous vehicle perception.

Extension to Other Image Processing Operators: The proposed methodology is directly applicable to other gradient-based operators such as Prewitt, Laplacian and Canny edge

detection, and also to Gaussian and bilateral filtering. If the methodology would be applied in systemic way to these operators, it would generate a complete approximate image processing toolkit.

Fixed Approximation: It may be impossible to determine the better approximation levels for all the image content. Future work is needed to explore adaptive approximation schemes that would automatically adapt their approximation level according to the local image statistics: Use exact arithmetic for smooth areas and approximate for high gradient areas.

CNN Accelerator Application: The effectiveness of approximate computing in Sobel convolution encourages its use in the convolutional layers of deep neural network inference accelerators, which have a large number of multiply-accumulate operations and already have a long history of error tolerance.

Estimated Synthesis Results: This thesis presents the measured results from the Xilinx 7 series FPGA platform. Future work is to implement and test the designs on real FPGA devices to get accurate power, timing and resource consumption under realistic operating conditions.

6.3 Research Contribution

- This research has several important findings:
- A full FPGA design of Sobel edge detection was designed in Verilog HDL.
- The proposed hardware architecture has been successfully integrated with approximate arithmetic circuits obtained by EvoApproxLib.
- A systematic Design Space Exploration methodology was developed to evaluate multiple approximation configuration.
- FPGA hardware performance was measured based on implementation parameters like LUT utilization, Flip-Flop utilization, propagation delay, operating frequency and estimated power consumption.
- The image quality was evaluated by objective image quality metrics such as PSNR, SSIM and MSE.
- A comprehensive hardware–accuracy trade-off analysis was performed to compare different approximation levels.
- With the aim of finding the best tradeoff between the hardware efficiency and the image quality, the optimal implementation, namely Configuration C1, was identified.
- The proposed approach is applicable for image processing systems on FPGAs.The methodology presented here is applicable to image processing system-on-FPGAs and has been shown to be practical.

6.4 Practical Applications of the Proposed Framework

Proposed approximate computing paradigm with FPGA offers tremendous promise in various practical image processing applications. The proposed methodology is very

suitable because of its capability to achieve simplicity in the hardware design without compromising the computational accuracy, which is required in systems with strict constraints on hardware resource, power, and fast time. Embedded arithmetic is becoming increasingly intelligent with the advent of a new generation of devices at the edge of the computing world, and approximate arithmetic is likely to be a more and more critical design principle in future digital systems. Edge AI is one of the most critical application areas on this. Edge AI on the contrary to cloud-based processing, computes calculation directly on local devices without continuously transmitting raw data to remote servservers. Approximatenfigurations, carefully selected in the case of Configuration C1, have been obtained that offer a good balance between processing speed and image quality and therefore are well suited for real-time automotive applications. It also has many advantages for Unmanned Aerial Vehicles (UAVs). The limitation of thi small drone size is due to limited battery power. High performance in terms of computational efficiency is needed for real-time image processing on board for obstacle avoidance, terrain mapping, surveillance, and target tracking. The FPGA implementation presented in this research not only maintains the same mission efficiency as the initial designs of the device but also decreases the amount of hardware resources required to carry out the edge detection process, thereby prolonging the operational life of the device and enhancing the mission efficiency. Another possible application of the proposed framework could be medical image processing. While highly accurate computational algorithms are usually required in diagnostic decision-making, several stages of the processing, such as the edge enhancement, noise reduction and image segmentation, are often capable of limited numerical approximation without the important influence on the decision of the diagnostic interpretation. The approximately proposed architecture can thus speed up these preprocessing operations and decrease the computational complexity. However, safety critical healthcare applications will need a lot of clinical trials before approximate arithmetic is used in the real world of medicine. Machine vision technology is now being used in an increasing number of industrial automation systems that use this technology for product checking, defect detection, object counting, and guidance for robots. In high-speed manufacturing fields, it is important to continuously capture images and quickly process them so that manufacturing is efficient. Approximate computing is a practical approach that is achieved using an FPGA that allows some processing delays to be sacrificed while still retaining enough of the visual information necessary for a reliable inspection. Proposed methodology can be applied to intelligent surveillance system too.

Security cameras are used for continuous processing of motion detection, human activity recognition and object tracking. Surveillance systems are operating twenty-four hours a day, so the reduction of power consumption directly reduces operation costs. Approximate arithmetic leads to lower switching activity and logic usage in FPGA devices and will provide acceptable image quality with reliable edge extraction. Therefore, a system that is more energy efficient can be created to keep up with the surveillance without compromising monitoring ability. The Internet of Things (IoT) vision devices is another booming sector of use. The image data is now processed locally by many smart home cameras, environmental monitoring stations and industrial sensor networks before the relevant, small amounts of data are sent to the cloud servers. This reduces the amount of communication delay and enhances user privacy by local processing. The small number of operations in the approximate arithmetic can facilitate continuous low-power operation which makes the proposed architecture of FPGA very attractive for the next generation of intelligent IoT systems. In general, the proposed framework shows that approximate arithmetic is not just a theoretical optimization of the hardware, but it can offer concrete practical advantages to a wide variety of embedded vision applications. The above performance of reduced area, low power consumption, short processing delay and acceptable image quality all at a time can show the possibility of the proposed Design Space Exploration methodology for the future real time image processing systems based on FPGA.

6.5 Recommendations for Future Researchers

The proposed approximate Sobel edge detection framework proposed for FPGA is a promising approach, but there are still numerous research potential to be explored to expand the presented work in this research paper. Other approximation techniques, more advanced image processing techniques and adaptive optimization techniques could be used in further works to improve the computational accuracy and hardware efficiency. Another potential research direction is to test approximate arithmetic in more advanced edge detection operators like Canny, Laplacian of Gaussian (LoG), Roberts and Prewitt operators. The various algorithms will have different mathematical operations, and therefore be sensitive to approximation errors to different extents. A comparison of several edge detection techniques will give a better insight into the applicability of

approximate computing in computer vision. Adaptive approximation methods that adapt the level of arithmetic precision to the image content and application are possible future research areas. Instead of a fixed approximation circuit being used throughout the processing, intelligent approximation circuits could be used that would automatically adjust the approximation level based on the complexity of the image being processed, the size of their battery, the processing time limits, or the quality of the output. These adaptive systems can provide even more hardware efficiencies without affecting visual efficiency.

A comparison of the approximate FPGA implementation with ASIC implementation of the approximate architectures could also be interesting for future studies. While FPGAs are superb for quick turnaround prototyping, mass produced ASICs typically have increased performance and power savings. Performance comparison of the FPGA implementation vs. the ASIC implementation would give insight into the selection of the most suitable hardware platform for approximate computing applications. In addition, the Design Space Exploration methodology proposed in this research can be extended by adding machine learning optimization algorithms. The optimal approximate configuration could be automatically determined by evolutionary optimization, genetic algorithms, reinforcement learning or Bayesian optimization techniques based on pre-defined hardware and image quality requirements. This automated optimization would save a lot of man effort in exploring and enhance design efficiency. The temporal constraints, frame synchronization issues and the memory bandwidth limitation are distinct from still-image processing in continuous video processing. The ability to judge approximate arithmetic in real-time streaming would give a better idea as to its applicability for practical multimedia applications. More research is needed on power-aware approximation methods. In the future, additional research could include the use of approximate arithmetic in conjunction with dynamic voltage scaling, clock gating, and runtime power management. These integrated optimizations may lead to even more energy savings for the entire system of battery-powered embedded systems.

To sum up, in the proposed research a solid base has been laid for further studies on approximate computing using FPGAs. The balance shown between the image quality and the optimization of the hardware suggests that approximate arithmetic has significant potential for future embedded vision systems. Many other applications of approximate computing will be further impacted by continued research on adaptive approximation,

intelligent optimization, deep learning acceleration, real-time embedded computing, and beyond.

6.6 Industrial and Commercial Impact of Approximate Computing

With the inherent need to perform more computation in modern digital systems, approximate computing has emerged as a promising technology. As the volume of data is growing with data intensive applications, industries are looking for hardware solutions that provide increased computational power, power efficiency and low implementation cost. The proposed approximate computing framework shows that these industrial requirements can be fulfilled by designing approximate arithmetic circuits that can meet application-level performance requirements while keeping hardware complexity low to avoid unacceptable degradation of the application-level performance.

The power of batteries is the main factor that prevents continuous execution of image processing algorithms on smartphones, wearable devices, smart cameras or multimedia systems. These devices can also compute heavy workloads of image processing with approximations, which helps them last longer on battery power and dissipates less heat. As more and more clever embedded platforms emerge, approximate hardware accelerators will become increasingly important for the overall system efficiency. The automotive field is one of the key application fields. Object Detection, Lane Tracking, Traffic Sign Recognition and Collision Avoidance are all important applications of real-time computer vision algorithms in Advanced Driver Assistance Systems (ADAS) and in autonomous vehicles. The systems should be extremely fast in processing the images with very small computational time. The approximate arithmetic, along with the short time required to arrive at a decision, results directly in its high speed of decision making and yet provides adequate visual accuracy necessary for the safe operation of a vehicle. Approximate computing can be very beneficial in industrial automation. Machine vision equipment are increasingly used to control the quality of the products or processes, guide the robots, classify the products and monitor the processes in the factories. These systems function 24 hours a day and by minimizing hardware power consumption, long-term operating costs are reduced. In industries that demand computational efficiency and economic sustainability, approximate FPGA architectures offer an enticing choice. In addition, cloud service providers and big data centers might also benefit from

approximate computing. Large image processing applications run on thousands of processing nodes and require a lot of electric power. Small improvements to the complexity of computation can lead to significant energy savings if used on a large scale of computing infrastructure. Therefore, approximate arithmetic could be used as a development towards environmentally sustainable computing because of its potential to save overall energy consumption. In summary, approximate computing is relevant to industrial applications, and not just in the academic realm. Approximate arithmetic is promising for future intelligent embedded systems due to its ability to increase the efficiency of the hardware, the speed of the processing and the energy efficiency all at once, while satisfying practical resource constraints.

6.7 Societal Benefits and Future Technological Impact

Intelligent embedded systems are evolving fast and hence the need for computational techniques that can offer high performance with low energy consumption. One such technology that has the potential to play a key role in the future sustainable computing infrastructures is approximate computing. Approximate arithmetic enables the design of digital systems that are more environmentally friendly, requiring less energy because it reduces the level of accuracy required in computations. The creation of low-cost intelligent healthcare technologies is one of the major societal impacts. Portable medical imaging devices, wearable health monitoring systems and telemedicine platforms need computational efficient image processing solutions which work with limited hardware. Computation of safety-critical diagnostics requires exact arithmetic, but approximate computations can be used to achieve faster preprocessing tasks, while reducing the cost and power consumption of the device and enhancing access to healthcare in resource-limited settings. Approximate computing technologies can also be used in educational and research institutions. Approximate architecture using an FPGA are good educational examples for digital design, computer architecture, embedded systems and artificial intelligence. Students and researchers can explore the optimization of hardware and learn at the same time the compromises involved with accuracy vs implementation efficiency. One other place where smart city infrastructure is a promising application is within the traffic system. The world is full of data to be processed in real time—data derived from

intelligent traffic monitoring, environmental monitoring, public safety systems and automated transportation. In these systems, image data is efficiently processed, power consumption of the system and the operation costs are reduced, and approximate hardware accelerators are used. The reduction in computing power also helps to reduce environmental impact, since that reduction directly translates to a reduction of carbon emissions in a large computing facility. As data centers spread across the globe, one of the key sustainability targets is to boost the energy efficiency of data center equipment. Approximate computing thus contributes to the worldwide research and development of "green ICT". Finally, the results of the research described in this thesis demonstrate that approximate arithmetic has not only been found to be a theoretical method for hardware optimization, but also an engineering approach that can be applied in the design of future intelligent computing systems. With the Internet of Things (IoT) devices, autonomous systems and high-performance computing continuing to advance, approximate computing is becoming increasingly a part of next-generation digital hardware architectures..

References

[1] I. Sobel and G. Feldman, "A 3x3 Isotropic Gradient Operator for Image Processing," Stanford Artificial Intelligence Project Technical Report, 1968.

[2] S. Mittal, "A Survey of Techniques for Approximate Computing," ACM Computing Surveys, vol. 48, no. 4, Article 62, pp. 1-33, March 2016.

[3] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of Power-Efficient Approximate Multipliers for Approximate Artificial Neural Networks," in Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1-7, 2017.

[4] S. Mittal, "A Survey of Architectural Techniques for Managing Process Variation," ACM Computing Surveys, vol. 45, no. 3, pp. 1-28, 2013.

[5] J. Han and M. Orshansky, "Approximate Computing: An Emerging Paradigm for Energy-Efficient Design," in Proc. 18th IEEE European Test Symposium (ETS), pp. 1-6, 2013.

[6] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and Analysis of Circuits for Approximate Computing," in Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 667-673, 2011.

[7] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004.

[8] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 4th ed. Pearson Education, 2018.

[9] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and Characterization of Inherent Application Resilience for Approximate Computing," in Proc. 50th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1-9, 2013.

[10] D. Mahajan et al., "Approximate Computing: An Architecture-Centric Perspective," Proceedings of the IEEE, vol. 106, no. 1, pp. 1-26, January 2018.

[11] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods," in Proc. Design, Automation & Test in Europe Conference (DATE), pp. 258-261, 2017.

[12] A. Raha, S. Venkataramani, V. Raghunathan, and A. Raghunathan, "Quality Configurable Reduce-and-Rank for Energy Efficient Processing of Hyper-Spectral Images on UAVs," in Proc. Design, Automation & Test in Europe Conference (DATE), pp. 119-124, 2015.

[13] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: Systematic Logic Synthesis of Approximate Circuits," in Proc. 49th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 796-801, 2012.

[14] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture," in Proc. 24th International Conference on VLSI Design, pp. 346-351, 2011.

[15] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, no. 4, pp. 850-862, April 2010.

[16] K. Y. Latif, M. Shafique, and J. Henkel, "Online Selfadaptive Error-Configurable Approximate Computing for Real-Time Signal Processing," in Proc. Design, Automation & Test in Europe Conference (DATE), pp. 329-334, 2019.

[17] A. B. Kinsman and N. Nicolici, "Finite Field Multiplier Architectures for Cryptographic Applications on FPGAs," in Proc. IEEE International Symposium on Circuits and Systems (ISCAS), pp. 3230-3233, 2009.

[18] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 8, pp. 1225-1229, August 2010.

[19] S. Hashemi, R. Bahar, and S. Reda, "DRUM: A Dynamic Range Unbiased Multiplier for Approximate Applications," in Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 418-425, 2015.

[20] B. Moons and M. Verhelst, "An Energy-Efficient Precision-Scalable ConvNet Processor in 40-nm CMOS," IEEE Journal of Solid-State Circuits, vol. 52, no. 4, pp. 903-914, April 2017.

[21] A. Sampson et al., "EnerJ: Approximate Data Types for Safe and General Low-Power Computation," in Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), pp. 164-174, 2011.

[22] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural Acceleration for General-Purpose Approximate Programs," in Proc. 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 449-460, 2012.

[23] T. Moreau, M. Wyse, J. Nelson, A. Sampson, H. Esmaeilzadeh, L. Ceze, and M. Oskin, "SNNAP: Approximate Computing on Programmable SoCs via Neural Acceleration," in Proc. IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), pp. 603-614, 2015.

[24] L. Liu, J. Han, and F. Lombardi, "A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 63, no. 4, pp. 715-719, April 2016.

[25] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "RAP-CLA: A Reconfigurable Approximate Carry Look-Ahead Adder," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 8, pp. 1089-1093, August 2018.

[26] C. Liu, J. Han, and F. Lombardi, "An Analytical Framework for Evaluating the Error Characteristics of Approximate Adders," IEEE Transactions on Computers, vol. 64, no. 5, pp. 1268-1281, May 2015.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, 2016.

[28] G. Tziantzioulis, A. M. Gok, S. M. Faisal, N. Hardavellas, S. Ogrenci-Memik, and S. Parthasarathy, "b-HiVE: A Bit-Level History-Based Error Model with Value Correlation for Approximate Computing," in Proc. 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1-6, 2015.

[29] X. Xilinx, "7 Series FPGAs Data Sheet: Overview," Xilinx Inc., San Jose, CA, USA, DS180 v2.6, Feb. 2018.

[30] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A Low Latency Generic Accuracy Configurable Adder," in Proc. 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1-6, 2015.