On the Improving of Approximate Computing Quality Assurance

Alain F.M. Aoun

A thesis
in
The Department
of
Computer and Electrical Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Applied Science at
Concordia University
Montréal, Québec, Canada

May 2021

Abstract

On the Improving of Approximate Computing Quality Assurance

Alain F.M. Aoun
Concordia University 2021

Approximate computing (AC) has been predominantly recommended for implementation in error-tolerant applications as it offers a reduced resource usage, e.g., area and power, for a trade-off in output quality. However, AC implementation has not been adopted in commercial designs yet as it is still falling short in providing a good enough quality. Thus, continued research in the field in the field of improving quality of AC designs is indispensable. In this direction, a recent study exploited the use of machine learning (ML) to improve output quality. Nonetheless, the idea of quality assurance in AC designs could be improved in many aspects.

In the work we present in this thesis, we propose a few practical methods to improve an ML-based quality assurance methodology, which consist of an ML-model that select the most suitable design from a library of AC circuits. For instance, we extend the library of AC designs used for the ML-based approach with larger data path circuits. Larger designs, however, result in an exponential growth of complexity. Thus we propose the use of data pre-processing in order to reduce this hurdle by prioritizing designs based on their physical properties.

Another direction of improving AC circuits designs in general, and the ML-based model in particular is design space exploration (DSE). We therefore propose a novel DSE that drastically reduces the design space based on the aimed targets for area, latency and power of the AC circuit. Moreover, even with a narrowed design space, the number of AC designs to be assessed for their quality could be enormous. Thus, as part of this thesis, we propose a DSE that uses an intricate mathematical modeling for designs to assess their quality.

In another effort in improving quality assurance for AC design, we introduce a highly reliable model that uses a minimal overhead. This work is achieved by using redundant AC modules to form an approximate quadruple modular redundancy

(AQMR) design. The proposed AQMR is superior to the exact triple modular redundancy (TMR) by offering a better reliability on top of the resource savings resulting from the implementation of AC.

In loving memory of my Godfather, my guardian angel, my uncle,
To my father, my mother, my sister and brothers.

Acknowledgments

First, I would like to thank my supervisor, Prof. Sofiène Tahar, for his valuable feedback, support, and encouragement throughout my master thesis. He was always available to share his experience and knowledge which helped in keeping my study on-track. Moreover, during my study under his supervision, I learned a lot about research which helped me improving my abilities in this field. In summary, I can say that I am beholden for his valuable time and effort that he has spent helping me achieving this milestone. Also, I am grateful for the extremely valuable feedback that Dr. Osman Hasan has offered throughout my research. Moreover, I am beholden to Dr. Mahmoud Masadeh, who was more than a colleague at the Hardware Verification Group (HVG) but also inspirational throughout my work, and was always available when needed, to provide support and brainstorm ideas.

I am also deeply grateful for Prof. Abdallah Kassem who was my supervisor at Notre Dame University for my undergraduate studies. Through his endless support and feedback, I acquired a lot of knowledge that was the essence for my Master's study. Moreover, I would like to thank him for introducing me to Prof. Sofiène Tahar, which I consider was one of the best things that ever happened to me.

I would also like to express my gratitude for Dr. Abdessamad Ben Hamza and Dr Otmane Ait Mohamed for serving on my advisory thesis committee and taking the time from their busy schedules to read and evaluate my thesis.

I am very thankful for my friends and colleagues at HVG, specially Yassmeen El-Derhalli, Hassnaa El-Derhalli, Saif Najmeddin, Mohamed Abdelhamid and Mohamed Wagdy Abdelghany, whom made me feel welcomed by a family upon joining HVG. Their company, guidance and the help offered when needed was generously overwhelming.

Finally, I would like to express my deepest gratitude for my family, i.e., my mother, my father, my sister and brothers, for their endless love and support throughout my life, and specifically through this journey. Their embracement made the hard days and chaos of the year 2020 much easier. Last but by no mean the least, I would like to thanks my friends in Montreal whom I call my second family. My sincere gratefulness to my friends from the families of Khalifeh, Saliba and El Kadiri for their honest and great love.

Table of Content

List of Figures

List of Tables

List of Acronyms

AC	Approximate Computing
AMA	Approximate Mirror Adder
ANN	Artificial Neural Netowrk
AQMR	Approximate Quadruple Modular Redundancy
ATMR	Approximate Triple Modular Redundancy
AUGER	Apprximate Units Generator
BDD	Binary Decision Diagram
BER	Bit-Error Rate
DMR	Dual Modular Redundancy
DSE	Design Space Exploration
DT	Decision Tree
ED	Error Distance
FA	Full Adder
GA	Genetic Algorithm
HA	Half Adder
HDL	Hardware Description Language
HPC	High Performance Computer
IoT	Internet of Things
k-NN	k-nearest neighbors
LR	Linear Regression
LSB	Least Significant Bit
LUT	Lookup Table
MAC	Multiply and Accumulate
MED	Mean Error Distance
ML	Machine Learning
MSB	Most Significant Bit

NN	Neural Netowrk
PADP	Power-Area-Delay Product
PC	Personal Computer
PEA	Probabilistic Error Analysis
PSNR	Peak to Signal Noise Ratio
QMR	Quadruple Modular Redundancy
QoS	Quality of Service
RF	Random Forest
RGB	Red Green Blue
SEU	Single-Event Upset
SOP	Sum of Product
TMR	Triple Modular Redundancy
TOQ	Target Output Quality

Chapter 1

Introduction

In this chapter, we first present the context and motivation behind this thesis followed by a review for the state-of-the-art and the problem statement. We conclude the chapter by outlining the main contributions and the organization of the thesis.

1.1 Context

The discovery of transistors in the 20th century and their implementation in computers changed the life on earth forever. Computers nowadays are integrated in day to day activities, e.g., communication and transportation. This deep integration has been achievable by reducing feature size, which enabled the fit of more transistors on a given die. However, this advancement has seen a slower pace in the last few years and computer architecture design has shifted from solely fitting more transistors to architecture modifications, such as the superscalar architecture and multicore processors, etc. These variations in the computer architecture allowed computers to cope with most of the current demands. Nonetheless, improving some of these advanced implementations have or will soon reach its saturation and computers might struggle in the near future to deliver the computation required with the booming of Internet of Things (IoT) and cloud based services. Furthermore, recent chip famine caused by COVID-19 pandemic and its impact on many industries, e.g., automotive [1] and phones [50], demonstrate the vulnerability of chip manufacturing and indicate that a drastic resource optimization, e.g., transistors used to perform a given process, must take place. The work in this thesis offers a solution to this dilemma by offering a reduced resource usage, i.e., area and power, while preserving an acceptable quality of service (QoS), i.e., output quality and reliability.

1.2 Motivation

Approximate Computing (AC), which is well known as best-effort computing, is a nascent computing paradigm that is suitable for error-tolerant applications, e.g., search engines [38], multimedia [38] and big-data analysis [42], which do not require an accurate result. These applications exhibit *intrinsic error-resilience* due to the following factors [53]: (i) iterating and noisy input data; (ii) absence of golden or sole output; (iii) imprecise sense of humans; and (iv) implementing algorithms with self-healing and error attenuation patterns. Based on these concepts, AC could be the essence in delivering the computation power required in the future as it offers a significantly reduced usage of resources.

Diverse approximation techniques in the levels of software and hardware have been investigated by industry and academia, such as IBM [42], Intel [37] and Microsoft [10]. AC in hardware results in a reduced area, delay and power requirements by compromising the accuracy of computation. Such reductions can be achieved by reducing transistor count, e.g., altering logic gates [17], voltage over scaling [41] or bit-wise truncation [20]. The development of circuits aiming to deliver approximate adders, multipliers, dividers have been researched, yet this development has not matured since the perfect trade off, i.e., Golden Goal, in the four dimensions of AC design, i.e., area, delay, power and quality, is not achieved so far. Moreover, for circuits designed with AC in mind, the error persists during the operational-life. Such error in the circuit is classified as hard-error [46]. Nonetheless, many of the proposed approximate designs offer promising results in quality assurance and if improved, an implementation in end-user devices could take place. Thus, future development of AC circuits should be driven by:

1- *Implications of Approximation*: Some modifications that result in the design of an AC circuit could be a little advantageous in some dimensions of AC design, while being very diminishing in the remaining aspects, e.g., improved latency with deteriorated output quality. Hence, the design of approximate circuits is a delicate process that must be carefully practiced.

2- *Assessing Quality*: Output quality assessment and verification of approximate circuits are open challenges with the quality mainly relying on excessive simulation. With larger circuits, an exponential growth in time complexity, e.g., possible inputs combinations of a 16-bit multiplier are 64K times larger compared to the combinations of an 8-bit multiplier. Reduction in simulation time relies on randomly generating a limited set of test cases, which could result in misrepresentation of the actual quality.
3- *Design Space Exploration (DSE)*: Most of the proposed designs have been studied in limited configurations. Furthermore, many variations can take place in these designs, which can generate a large set of undiscovered possibilities. With such large set of possibilities, the chances of finding a configuration that is near the golden goal in the four dimensions of an approximation computing design can be considered reasonable.
4- *Approximate Failure*: An approximate circuit should be designed with the notion of fail-small, fail-rare or fail-moderate where the approximated output should not result in a high loss of quality. Thus, the output-reliability must be studied carefully.

Multiple approximate circuit designs have been proposed in the literature, yet their usage for quality assurance has not matured so far. In the scope of this thesis, we propose to improve an existing quality assurance method, along with a new implementation of AC for quality assurance. In addition, we propose a design space exploration for approximate circuits.

1.3 Sate-of-the-Art

In this section, we review most relevant literature in the area of quality assurance and design space exploration, which are closely related to the objective of this thesis. The literature will discuss the need for AC designs and some of the different techniques discovered so far. Moreover, this section will review state-of-the-art methods for improving quality assurance of AC designs in particular machine learning based quality assurance and design space exploration. Nonetheless, when dealing with AC designs, the main issue is quality assessment. In this area, we will review recent approaches for mathematical modeling of circuits, which can help assessing quality.

1.3.1 Approximate Circuits

Due to the massive explosion in new data, i.e., big-data, for the sake of improving daily services, e.g., smart-cars, search engines etc., powerful computing machines are falling short in delivering all computations needed. Thus, creating the idea of rethinking computer architecture, in order to deliver low power consumption, small footprint, yet powerful computing machines [28] [30] [42] . Moreover, many applications are considered error-tolerant where there is no golden or single correct output, e.g., search engines. Therefore, AC is good trade-off, that can satisfy these requirements. Furthermore, AC can be achieved at the hardware or software level.

Research in approximate sub-blocks, e.g., full adders (FAs), is considered a hot topic in AC circuit design, as their implementation can go beyond the application of adders. For instance, the work in [43] used approximated FAs proposed in [17] to design an array multiplier. Another implementation of approximated FAs is building multiply and accumulate (MAC) circuits as proposed in [29]. Furthermore, research in this area is important, as multiple sub-blocks can be combined together to form a larger functional unit. This would generate a large design space to be explored, since variation at the sub-block level will create different design structure.

For this reason, various approximate hardware designs, specially functional units for arithmetic modules including adders [4] [17] [58], dividers [12] [20] [23] [36] and multipliers [19] [39] [43] [59] [61] [60], have been explored for their practical role in multiple applications. These applications are sporadically tested in computation intensive yet error-tolerant applications, which are susceptible to an approximation.

The approximation of functional units can be done at different levels of abstraction, i.e., transistor, gate, register transfer and application. Transistor-level approximation offers the uppermost versatility, since it is the bottom-line of circuit design. Variations at this level will twist most aspects of the design parameters. The work proposed in [17] is a good example of transistor-level approximation in order to build approximate FA. A similar approximation is accomplished on the gate level by Z. Yang et.al in [58]. Nonetheless, a lower level of abstraction does not guarantee an outstanding improvement in the four dimensions of AC design simultaneously [28]. The work in [28] shows that a good starting point should be a behavioral study for smaller sub-blocks. Such step can be considered as a vital assessment when designing AC circuits, since smaller sub-blocks, e.g., FA, can be used to build larger circuits.

1.3.2 ML-based Quality Assurance for AC Design

The error in AC designs depends on the applied inputs [25]. This could be taken care of if the error is known at early design stages. The solution could take one of the following recently exploited forms: 1) adapting the architecture of approximate components in form of error-compensation input aware [31]; 2) partial reconfiguration as a switch among multiple AC circuits as proposed by [30] or [32]. These solutions rely on machine learning (ML) for quality assurance. ML is a computer algorithm that builds a model by learning properties of a provided training data. The trained data generates an ML-model which is then used to predict the outcome of a given input. ML is a hot topic where algorithms are constantly improved to support a wider range of models, e.g., decision tree (DT), artificial neural network (ANN) and k-nearest neighbors (k-NN) [21]. ML-algorithms are used in different types of applications, e.g., speech recognition and media description [8]. Using ML has been widely used for quality assurance for big-data where the aim is retrieving relevant information to assure quality [47]. Similarly, the work in [30] [32] exploited the use of ML-algorithms for quality assurance in AC designs.

The foundation of the work done in [30] [32] is to have a knob-like setting that selects the most suitable design based on a provided target output quality (TOQ). As shown in Figure 1.1 the proposed model is constructed by first designing various approximate arithmetic modules, which will form a library of approximate designs. Output quality of all models in the library is then assessed using excessive simulation. Afterwards, error metrics, e.g., error distance (ED) and peak signal to noise ratio (PSNR), are quantized for every n-consecutive input entries. The quantized data is then used as a training data for the ML-algorithm, which will generate a ML-predictor. The ML-predictor will be provided with the TOQ and input data in run-time, which will select the most suitable design from a library of designs.

The proposed ML-based quality assurance approaches for AC design could be improved in many directions, and one of them is a design space exploration (DSE), which can improve the library of approximate designs.

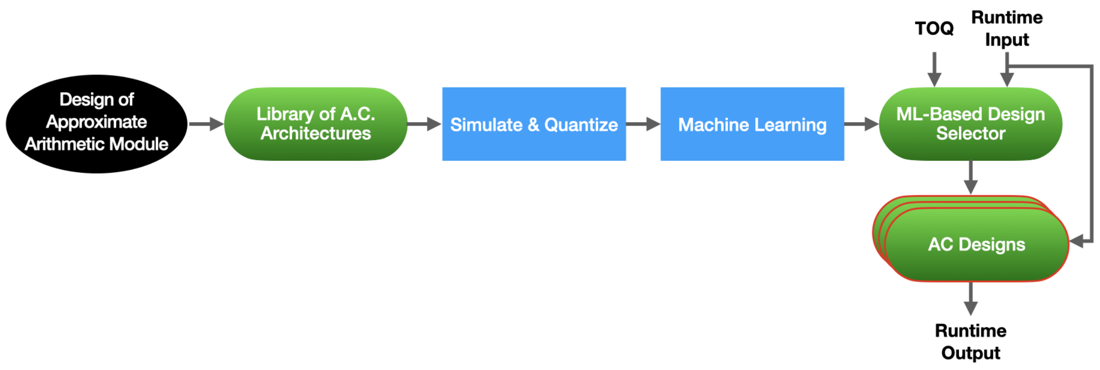
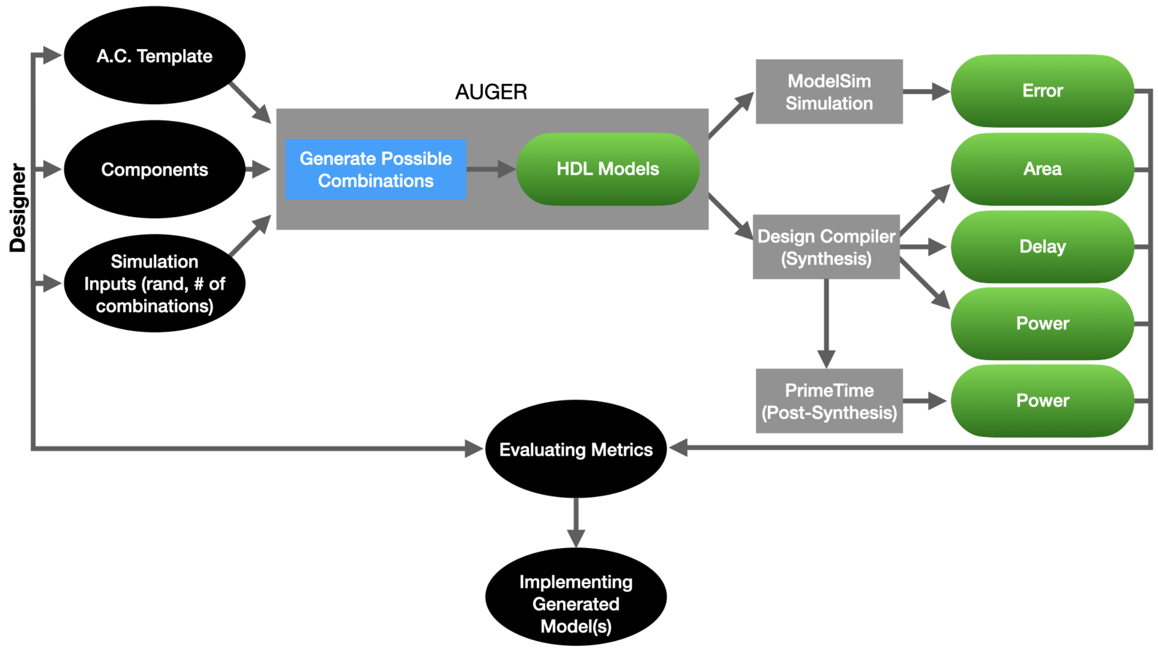


Figure 1.1: ML-Based Quality Assurance for AC Design

1.3.3 Design Space Exploration for AC

Given that the structure of approximate circuits can have multiple configurations, design space exploration (DSE) can become useful as it offers an automated generation of possible configurations. For this purpose, some tool have been developed such as Approximate Units Generator (AUGER) [22] and Automatic Design Space Exploration and Circuit Building (autoAx) [40], which generates approximate adders, multipliers, and dividers using a sub-components library [22]. Figure 1.2 depicts how the AUGER tool allows the designer to provide design specifications, such as bit-width of the arithmetic unit, least significant bits (LSB) to be approximated, the sub-blocks type and the number of random samples to be used to measure probabilities of error distances (EDs). The tool generates hardware description language (HDL) model based on the provided requirements. The generated model is forwarded to synthesis tool for area, delay and power assessment. In addition, a post-synthesis is performed in order to have a more accurate power valuation. Finally, Figure 1.2 shows that the designer will manually assess the output quality and check the area, delay and power usage of the generated structures and choose the design satisfying the requirements. Nonetheless, a DSE is needed for AC designs, and the implementation of AUGER uses the simulation of random inputs for quality assessment. However, a small number of random inputs may not reflect the actual quality of an AC design, while a large set of samples could be time consuming for large designs.



Designer
A.C. Template
Components
Simulation Inputs (rand, # of combinations)
AUGER
Generate Possible Combinations
HDL Models
ModelSim Simulation
Design Compiler (Synthesis)
PrimeTime (Post-Synthesis)
Error
Area
Delay
Power
Power
Evaluating Metrics
Implementing Generated Model(s)

Figure 1.2: DSE for ACusing AUGER Tool [22]

1.3.4 Error Models of Approximate Design

Probabilistic error analysis (PEA) for approximate circuit designs is commonly used to measure its quality. The foundation of this approach is the assumption of equiprobability of inputs being logic 0 or 1. This equiprobability is then carried to study the probability of output bits. An excellent example of quality measurement using a probabilistic form is proposed in [35], where the authors describe a technique to build larger approximate arithmetic units recursively from an energy-efficient smaller component. The probability of error for sub-components is evaluated first. Using a recursive procedure, this probabilistic behavior is later generalized to a larger arithmetic unit. However, applying the proposed PEA in [35] to large designs, e.g., 32-bit multiplier, can be delicate. Moreover, another disadvantage of PEA is the fact that the output assessment relies on bit-error which can have irrelevant indication in some cases. For example, if an exact computation resulted in $(16)_{10}$ while its approximated version resulted in $(15)_{10}$, the bit-error would indicate a 0% accuracy while the arithmetic error distance (ED) is 1. Furthermore, the bit-error could indicate a high accuracy while ED is large, e.g. $(15)_{10}$ and $(7)_{10}$. These extreme cases show that looking at the bit-error may point in the wrong directions and/or may not indicate an absolute behavior of the quality. Nonetheless, the usage of mathematical notions to asses output quality of AC designs is a deviation in the right direction.

1.3.5 Boolean Calculus Modeling

Calculus modeling for circuits is an alternative approach to model logic circuits, which can be applied on AC designs in order to asses their quality. This approach has been researched in the 20th century [26]. This exploration focused on canonical Boolean vector representation. This work earned the name of "Digital Calculus" since it merges Boolean calculus and binary vector Boolean calculus. Moreover, the authors developed the differential and primitives of the proposed equations. This work was poorly adopted since it lacks wide support by commercial tools for this type of equations and thus can be burdensome to deal with. Nonetheless, a calculus modeling for circuits can eliminate the need for excessive simulation, since quality assessment can be examined using mathematical equations. Hence, an alternative approach which relies on mathematical approaches that are supported by commercial tools is crucial.

1.3.6 Modular Redundancy to Improve Reliability

Modern micro-architectural trends and scaling feature size, reduce the susceptibility of logic circuits to external noise such as radiation or internal noise such as variations in applied voltages [48]. Such events can result in lifetime damages (hard errors) [46], or temporary faults (soft errors) [54]. In both cases, the system can be deemed as malfunctioning. Therefore, error mitigation techniques, such as fault tolerance at both software and hardware levels are proposed in order to improve quality of service (QoS). However, the use of these techniques can significantly affect embedded systems and microprocessors as more resources are required, e.g., area and power.

Error-mitigation can be done in hardware in the fashion of components redundancy, such as double modular redundancy (DMR) and triple modular redundancy (TMR) [5]. However, error-correction modules will add a significant overhead in terms of resources, i.e., area, performance and power. To overtake this issue, selective or partial redundancy is proposed in [54]. This technique will protect sensitive components and thus reducing the overhead. Towards achieving a similar goal, the usage of AC has been researched in [44]. AC designs are attractive since they are fast and offer a low resources usage.

TMR is one of the most used technique, which consists of triplicating the module and adding a voter which will mask error, if one of the three units is faulty. The downside of TMR, is the 200% increase in area and a similar rate in power. To reduce overhead, approximate TMR (ATMR) [45] is presented for use at the hardware level. This work targets loop-based applications where approximation is achieved by using duplicate circuits that will be used for fewer iterations. Another work of ATMR is presented in [16] with the usage of an exact unit along with two approximate units. The realization of the approximate units is achieved by first finding a subset architecture for the exact unit to build the first approximate unit, followed by a subset of the approximate unit to generate the second approximate circuit. A good example that illustrates the realization of this work is an exact unit that computes $Exact = (A \times B) + C$, while the approximate units compute $App_1 = (A \times B)$ and $App_2 = A$. This approach would allow to mitigate errors in a given sub-space. A similar approach to [45] has been explored in [49], which uses binary decision diagrams (BDDs) [11] to represent all the functions under consideration when forming the approximate circuits. Another approach of integrating AC to improve reliability is proposed in [7], with the goal here to build three different approximated versions of the exact unit to be used in parallel. Similar approach is used in [6] with the assumption that the usage of three different approximate units would generate an exact computation using a majority voter in the absence of errors, with the possibility of erroneous computation in the presence of a fault. To achieve a better quality, the usage of four different approximate units is proposed in [14] with the aim to improve the probability of achieving a better quality in the absence of faults. Even though the integration of AC has advantages in terms of reducing area and power, it has seen a limited adoption as it lacks quality assurance since all previously proposed models are either application specific, e.g., loop based applications, or designed with the assumption of achieving exact computation in the absence of faults.

1.4 Problem Statements

Developed AC circuits proved to be powerful in reducing area, delay and power. However, the best trade-off to achieve the golden goal in the four dimensions of AC design is not achieved. The list below represents a series of researched work that can be considered as assets of accomplishing this goal, yet with their respective drawback(s) such achievement is paralyzed:

1- ML-Based Quality Assurance [27]: This work, offers a noticeable improvement in quality. However, its scalability is questionable, e.g., 16-bit multipliers library instead of 8-bit, since the simulation time for an n-bit multiplier is almost 2^n times larger compared to m-bit multiplier (where $n = 2 \times m$). Another challenge is handling the large set of data to be classified.

2- Design Space Exploration (DSE): One way of improving AC designs is by performing a DSE. Nonetheless, previous methods of DSE for AC designs do not consider all possibilities in the design space. Nonetheless, a proper DSE must consider all possible configurations.

3- Quality Assessment: Output quality assessment is a vital process to any research in the field of AC. However, using excessive simulation can take an infinite amount of time. Alternatively, a mathematical approach is proposed in [35] which grants a small overhead. Nonetheless, this method suffers from inaccuracy in some cases and can become complex. Thus, research for a different quality assessment method that has a minimal overhead while offering a solid determination of quality is needed.

1.5 Proposed Methodology

As discussed in the previous sections, AC designs have been proposed and tested in limited numbers of configurations. Furthermore, the usage of AC proved saving benefits in area, delay and power, in exchange for reduced output quality. Moreover, the main method used today to assess quality of AC designs is excessive simulation. Such method consumes a good amount of time and may not be feasible for large designs. In this thesis, we aim to improve the quality of service (QoS) with the use of AC circuits.

Towards achieving this goal, we study a state-of-the-art model [32], which is an ML-based design select towards the delivery of quality assurance. The study focuses on finding the limits of the proposed model in terms of extending to larger designs. In addition, a careful examination for further quality improvement is performed. Based on these studies, we propose a methodology that overcomes the challenges by using data pre-processing to reduce the training data. Furthermore, if a design space exploration (DSE) is conducted, the library of approximate designs might be improved and maybe reduced. Therefore, these changes would benefit the quality of the proposed design.

Nonetheless, the state-of-the-art DSE generates designs in a limited number of configurations and relies on excessive simulation to assert quality. To overcome these disadvantages, we propose a novel DSE that aims to study all possible scenarios, yet robustly eliminating configurations and thus narrowing the design space. The elimination is done based on the designer's targets for area, delay and power. Moreover, the proposed DSE offers a new mathematical modeling that can be deployed on commercial tools, e.g., Matlab [34]. The modeling will then be used to assess the output quality of the candidate designs from the reduced design space.

Finally, we propose a highly reliable approximate design to improve quality assurance using AC. This design uses the concept of modular redundancy in order to improve the reliability of logic circuits. The design is an approximate quadruple modular redundancy (AQMR) and consists of three approximate modules and one exact module. Figure 1.3 illustrate a general overview of the proposed methodology for improving AC quality assurance. The methodology consists of black boxes describing manual operations that the designer must perform by hand, blue boxes represent implemented computer based processes, while the green boxes are physical outputs of the individual processes. As the figure shows, the overall methodology consists of four main classes, with the manual, i.e., traditional, generation of AC designs being the base for all the remaining work. The other three components are: 1) ML-based output quality assurance which aims at improving the output quality 2) DSE which will result in an improved library of AC designs; and 3) quadruple modular redundancy (AQMR) which improves reliability yet offers a minimal overhead by using AC. These three components complement each other in the direction of improving quality assurance, i.e., QoS, with the use of AC. For instance, the the ML-based design selector could be enhanced further if an improved library of AC designs is generated by a DSE. The improvement can take place in some or all of the four dimensions of AC

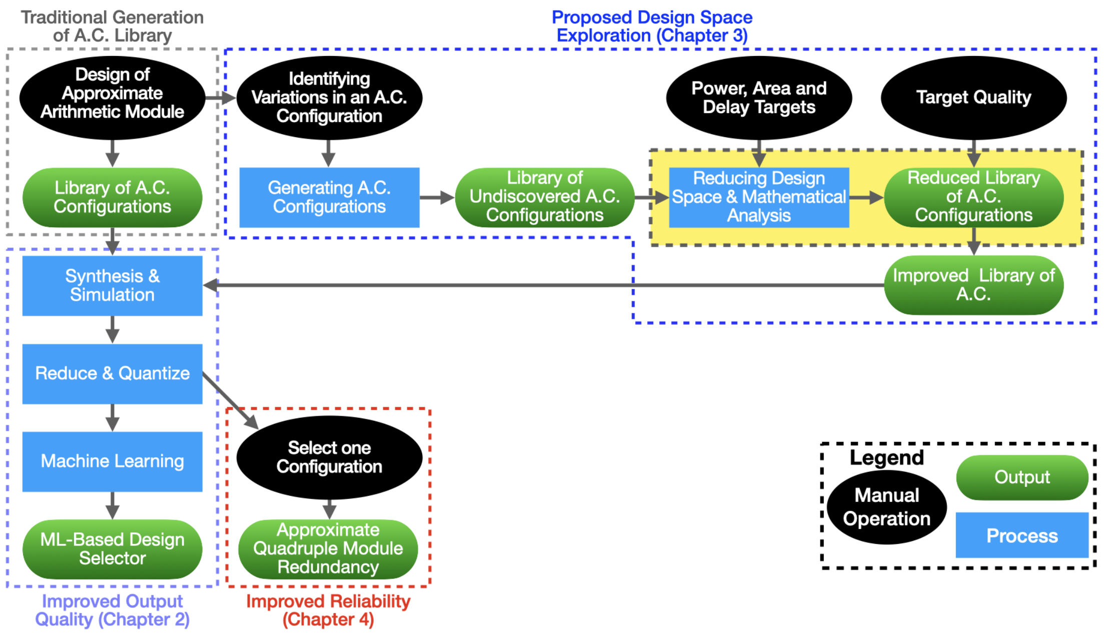

Traditional Generation of A.C. Library
Design of Approximate Arithmetic Module
Library of A.C. Configurations
Proposed Design Space Exploration (Chapter 3)
Identifying Variations in an A.C. Configuration
Generating A.C. Configurations
Library of Undiscovered A.C. Configurations
Power, Area and Delay Targets
Target Quality
Reducing Design Space & Mathematical Analysis
Reduced Library of A.C. Configurations
Improved Library of A.C.
Synthesis & Simulation
Reduce & Quantize
Machine Learning
ML-Based Design Selector
Improved Output Quality (Chapter 2)
Select one Configuration
Approximate Quadruple Module Redundancy
Improved Reliability (Chapter 4)
Legend
Manual Operation
Output
Process

Figure 1.3: General Overview of the Proposed Thesis Methodology

design, based on the designer' inputs to the DSE. Similarly, the AQMR could benefit in a similar manner by using data resulting from any of the processes for improving output quality to select the most suitable design. For instance, in Figure 1.3, AQMR is applied after data reduction and quantization.

The output quality improvement is based ML-based design selector proposed in the literature [27]. To achieve this improvement, the library of designs is first synthesized and simulated for a complete quality analysis. Thereafter, the data generated by the simulation is quantized and reduced. The reduction is based on the synthesis of the designs, which will prioritize designs in the library. Thereafter, reduced and quantized data is used as a training data for the ML-algorithm, which will generate a ML-based design selector. The generated selector will be used in runtime to predict the most suitable design delivering the aimed output quality.

The DSE is a crucial in improving the library of AC configurations. The proposed DSE consists of first identifying the variations in the proposed designs. These variations will generate a large number of undiscovered configurations. Thereafter, the library is reduced based on the targets of area, delay and power provided by the designer. This will generate a smaller library of undiscovered AC configurations. Finally, the quality of configurations in the reduced library are assessed, with designs satisfying target quality carried to form an improved library of AC configurations.

Finally, the improvement in AC design reliability will be achieved with the use of modular redundancy. As Figure 1.3 depicts, the process that consists of selecting one configuration of an AC design that is known to have a good metrics in the four dimensions of AC. The design is then used to built the proposed AQMR.

1.6 Thesis Contributions

In this thesis, the work presented focuses on improving quality assurance when using AC circuits. The contributions of the thesis can be summarized as follows:

- Extending ML-based quality assurance model proposed in [32] from 8-bit to 16-bit. Moreover, the work improves the output quality by giving a priority to specific designs. To provide a relevant priority, the designs are synthesized using Xilinx Vivado [57] with the design resulting in the least power-area-delay-product (PAPD) given the highest priority [Bio-Cf2].

- A novel DSE that drastically reduces design space based on the designer's requirements, i.e., resource usage and output quality. The design space reduction is accomplished in two steps, i.e., area & power optimization and delay optimization. In addition, the presented DSE uses a calculus- based mathematical modeling for logic circuits which will revoke the use of excessive simulation and replace it with a pure mathematical examination of output quality. The use of this modeling remarkably reduces the time needed for quality assessment.

- Improving the quality of circuits by delivering higher reliability with the use of approximate circuits. This improvement is achieved by using modular redundancy. Moreover, unlike previous implementations of approximate modular redundancy, the proposed model adapt to the characteristics of AC circuits and uses a two steps voter. One of the voters is a new magnitude-based voter [Bio-Cf1].

1.7 Thesis Organization

The rest of this thesis is organized as follows: in Chapter 2, the challenges of extending a previously proposed ML-based quality assurance is presented. Subsequently, we proposes a solution that reduces the training data with the aim of reducing complexity of constructing the ML-predictor which also reduces the complexity of the design in runtime. Moreover, this reduction results in an improved output quality. Finally, the currently known limits of this methodology are discussed.

In Chapter 3, we propose design space exploration by first describing the methodology used to reduce design space. Consequently, a detailed explanation of the mathematical modeling is proposed, which includes the reasoning that led to the creation of the equivalences. Lastly, a practical experiment is conducted to validate the proposed methodology.

In Chapter 4, a new approach to implement AC circuits in modular redundancy is proposed. Thereafter, a detailed assessment for resource usage, accuracy and reliability are presented. Lastly, Chapter 5 summarizes the thesis and the outlines potential future research directions.

Chapter 2

Extended Implementation of ML-Based Quality Assurance

This chapter represents a detailed description of the challenges in extending the work in [32], which uses 8-bit multipliers. While 8-bit designs can be used as a proof of concept; however, extensions to larger designs, e.g., 16-bit functional units, are required. This chapter also represents a possible solution for these hurdles and the outcome of the proposed solution.

2.1 Introduction

The proposed model in [32] uses 8-bit array multipliers with variations to its structure to form multiple approximated configurations. Figure 2.1 shows the structure of an 8-bit array multiplier which consists of full adders (FAs) and half adders (HAs).

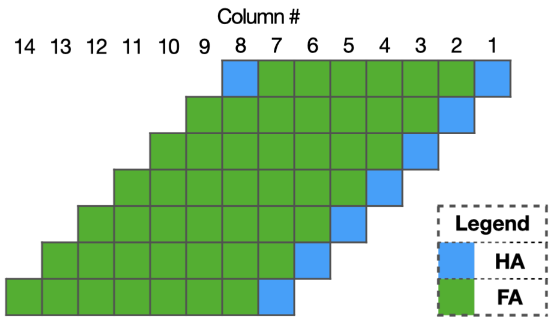


Figure 2.1: Structure of an 8-bit Array Multiplier

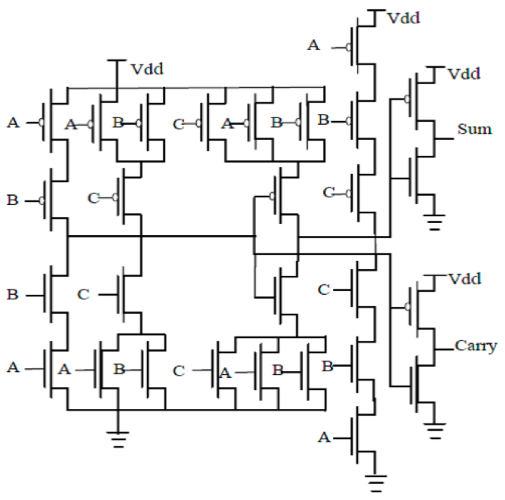


Figure 2.2: Schematic of Conventional Mirror Full Adders [17]

The work in [32] opted for the use of four degrees of approximation: D1, D2, D3 and D4. The chosen degrees are the number of columns in which the exact FAs are replaced with approximated ones. The selected four degrees are: 1) half of the columns, 2) half minus one column, 3) half plus one column, and 4) all columns approximated. Moreover, the authors of [32] used five well known approximate FAs proposed in [17] denoted as AMA1, AMA2, AMA3, AMA4, and AMA5, to form five types of approximate array multiplier. These five approximate FAs resulted from modifications in the exact mirror FA which is shown in Figure 2.2. The work in [17] is based on removing one set of transistors at a time to form a new approximate mirror adder (AMA). This iterative reduction of transistor count leads to a minimum count of two transistors in AMA5, i.e., two buffers. Thus, the library is composed of 20 designs, i.e., four degrees and five types, as shown in Table 1. Furthermore, the structure of *Design 3* can be seen in Figure 2.3.

Table 1: Library of 20 Static Approximate Designs based on *Degree* and *Type* [27]

Approximate Designs		**Degree**			
		D1	**D2**	**D3**	**D4**
Type	**AMA1**	*Design 1*	*Design 2*	*Design 3*	*Design 4*
	AMA2	*Design 5*	*Design 6*	*Design 7*	*Design 8*
	AMA3	*Design 9*	*Design 10*	*Design 11*	*Design 12*
	AMA4	*Design 13*	*Design 14*	*Design 15*	*Design 16*
	AMA5	*Design 17*	*Design 18*	*Design 19*	*Design 20*

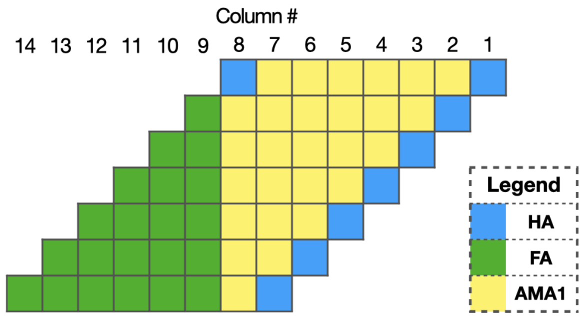


Figure 2.3: Structure of *Design 3*

As proposed in [32], the modified array multipliers are used to generate a library of AC designs. Moreover, with the help of machine learning (ML), an ML-based design selector is developed in order to choose the most suitable design in runtime, which will deliver the aimed output quality. In the context of extending the models proposed in [32], we chose the implementation of 16-bit array multipliers, with the same type of FAs, and a similar approach in the degrees of approximation, which is based on four levels of column-driven approximation. Lastly, the extended implementation uses the same error metric, namely PSNR, as used in [32].

2.2 Challenges Extending the ML-based Quality Assurance for AC

Extending the ML-based quality assurance model proposed in [32] to larger designs comes with its challenges. Thus, in this section, we will discuss in details the main challenges, which are:

1- *Simulation Time*: Simulation time and possible outcomes for larger circuit have a ratio of more than 1:1. This is due to the fact that larger circuits have more components that require more simulation time. For instance, if the 8-bit and 16-bit array multipliers are to be compared, the 16-bit model is four times bigger in terms of FAs usage. This overhead alone would imply that the 16-bit simulation would require roughly four times the time to compute a single combination of inputs. On top of that, the 16-bit multiplier has more possible inputs (64K times more). Considering these two facts, it is obvious that the simulation time will be much higher for larger citcuits.

2- *Classifying Diverse and Large Training Data*: Machine learning is a computer algorithm that learns to act based on a set of training data. In the proposed model by [32], the provided training data is composed of quantized inputs and their corresponding errors. However, when expanding to larger models, i.e., larger input-width, classification can become more challenging. For instance, for 20 approximate 16-bit multipliers, if simulation data is quantized for every 16 consecutive inputs, i.e., clusters of size 16×16, the number of entries for the training data is: $\frac{20\times(2^{16}\times2^{16})}{16\times16} = 335,544,320$. Moreover, with each entry containing a set of three data, i.e., two-inputs and corresponding error, the size of the training data to be classified by the ML-algorithm is almost 1 billion. Errors can be similar in magnitude, and thus the job can be easy for the classifier. However, the fact that errors can vary in magnitude, cannot be omitted. Thus, with the growth and possible variance of the training data, the classifier can easily fail to classify the training data, and hence aborting the process of building a ML-based design selector.

2.3 Proposed Solution

For ML-algorithms, classifying large and diverse training data is a delicate process, since the computation power can be beneficial, yet not a sufficient factor to build an ML-based predictor. Furthermore, with the exponential growth of data entries mentioned earlier, larger circuits will provoke much more data. To surmount this challenge, the usage of data pre-processing can be a viable solution, which is often neglected, yet it represents a significant step in the data mining process [15]. The authors of [32] applied a class of pre-processing (quantizing); however, this pre-processing is not sufficient to handle the simulation of large circuits. Moreover, modifying the proposed pre-processing by enlarging the array of data to be quantized could potentially result in a loss of accuracy in the training data and hence might compromise the proposed quality assurance model. Nonetheless, applying other techniques can aid in reducing data, yet without the loss of indispensable data. For this reason, the reduction of entries based on given priority is proposed. A valuable priority for the AC circuits can be the Power-Area-Delay Product (PADP), with the design offering the lowest PADP, would have the highest priority. Subsequently, with accordance to this priority, training data entries are eliminated on the basis of the existence of designs with the same or better accuracy and a higher priority, i.e., lower PADP. Additionally, input instances with high accuracy, e.g., $PSNR \geqslant 70dB$, are excluded if there exists another instance with high accuracy. Finally, input entries with very low quality, e.g., $PSNR \leqslant 15dB$, which are considered worthless in real applications are also eliminated. This data pre-processing, would keep training data of circuits with lower PADP, yet offering a better, similar, or high enough quality compared to circuits with higher PADP. This data-processing resulted in a light-weight training data, as it will be demonstrated in the next section.

Figure 2.4 depicts the flow of the proposed methodology when extending the work proposed in [32] to larger models. The methodology consists of first designing a library of AC designs. These designs are synthesized on the RTL level and prioritized based on their PADP. Moreover, the models are simulated in parallel computing using high performance computation (HPC) to generate training data, with this data being quantized based on consecutive entries to produce quantized training data. Afterwards, based on the synthesis-based priority, the quantized training data is reduced, and then sent to the ML-algorithm in order to be classified. The ML-algorithm will generate a design selector which will predict the appropriate design based on a given TOQ and the runtime inputs. Finally, it must be noted that the proposed reduction of training data set may result in a different behavior of the ML-based design selector. Nonetheless, this modification is accepted since the aim here is to develop an ML-based design selector that will select the most suitable design satisfying the TOQ and not the classification of generated data from a data analysis point of view.

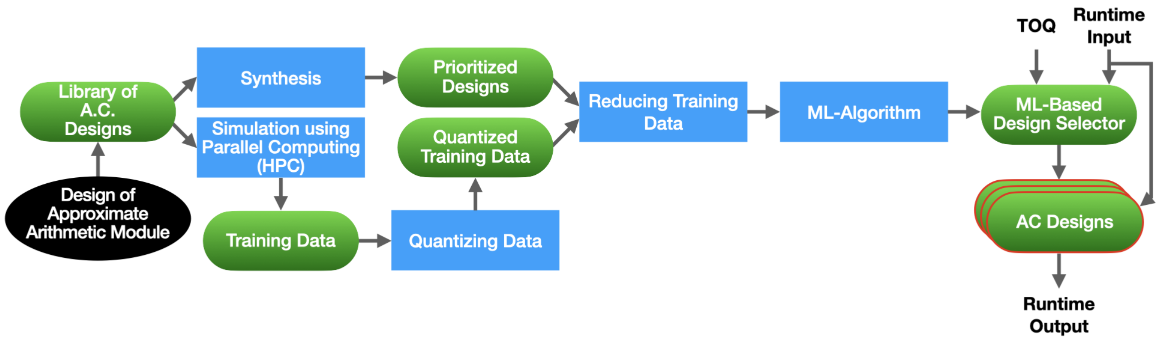


Figure 2.4: Improved ML-Based Quality Assurance

2.4 Experimental Results

In this section, we conduct experiments in order to demonstrate the performance gains from applying the methodology proposed in the previous section. The attained benefits in each stage, are explained in the following subsections.

2.4.1 Simulating Extended Model

The excessive simulation to evaluate the accuracy of 20 designs of 16-bit array multipliers was conducted on high-performance computing (HPC) server [52]. This HPC offers 32 physical cores, 512GB of RAM per node and a total of 24 nodes. The runtime gained access to the full power of this machine, by utilizing all 32-cores available per node. Furthermore, with the HPC server offering multiple computational nodes, different models were simulated simultaneously each on a given node, which drastically reduced the overall runtime. Moreover, the simulation of each model took between 8 to 10 hours, reduced from 24 days if they were simulated using a PC. Furthermore, the overall runtime was less than 2 days compared to 1 year and 4 months if it had to be conducted on a PC. This speed-up was achievable due to the availability of multiple nodes, thus running multiple simulations in parallel. Moreover, it must be noted that the extension to 32-bit can be deemed impractical since its projection simulation time would be in the range of 3.4×10^{10} to 4.3×10^{10} hours when using a similar HPC.

2.4.2 Quantazing & Reducing Training Data

Next, as shown in Figure 2.4, the training data resulting from the excessive simulation is first quantized, which reduced the data from 8.5×10^{10} down to 3.35×10^{8}. Afterwards, the 20 designs are synthesized using Xilinx Vivado [57]. We conducted the synthesis with an XC7VX330T FPGA from the Xilinx Virtex-7 family [56]. Table 2 shows the synthesis results, i.e. area, power, delay. The power considered in this study is the FPGA dynamic power and measured in mW, while the area is measured by two factors represented in the number of slices and lookup tables (LUTs) used by each of the designs. Finally, the delay is measured in ns and represents the time required by the slowest output bit. The PADP is computed by multiplying the sum of the two columns of area, i.e., Slice and LUT, by the dynamic power and delay.

Moreover, as proposed in the previous section, each design was given a priority, where the design with the lowest PADP has the highest priority while the design with the highest PADP has the lowest priority. As shown in Table 2 the design resulting in the lowest PADP (Type = AMA4, Degree = D4), is provided the highest priority which is 1. Furthermore, the benefit of AC circuits can be noticeable, since the circuits with priority 1, has 0.18% the PADP of the exact multiplier. Moreover, the design with least priority (Type = AMA2, Degree = D1), i.e., approximate circuits with highest PADP, resulted in a PADP that is 55.63% the value of the exact multiplier design.

Table 2: Design Characteristics of the Approximate Library, i.e., Power, Area, Delay and Power-Area-Delay Product (PADP)

Design	**Degree**	**Dynamic Power**(mW)	**Area** (Slice)	**Area** (LUT)	**Delay** (ns)	**PADP**	**Priority**
AMA1	D1	290	166	552	18.297	3809.80	19
AMA1	D2	259	165	536	18.472	3353.76	17
AMA1	D3	230	151	487	13.620	1998.6	11
AMA1	D4	52	53	115	7.547	65.93	3
AMA2	D1	318	165	504	18.479	3931.26	20
AMA2	D2	300	153	483	18.690	3560.45	18
AMA2	D3	289	148	473	18.329	3289.49	15
AMA2	D4	98	80	207	8.221	231.22	5
AMA3	D1	309	156	451	17.796	3337.87	16
AMA3	D2	292	147	467	18.876	3204.95	14
AMA3	D3	271	133	415	17.134	2544.54	13
AMA3	D4	93	38	63	7.330	68.85	4
AMA4	D1	268	143	439	15.109	2356.64	12
AMA4	D2	249	128	423	14.434	1980.33	10
AMA4	D3	222	128	413	14.366	1725.39	8
AMA4	D4	32	27	34	6.787	13.25	1
AMA5	D1	287	128	413	14.366	1725.39	8
AMA5	D2	270	99	312	13.989	1552.36	7
AMA5	D3	241	93	255	13.343	1119.05	6
AMA5	D4	74	23	24	6.046	21.03	2
Exact	-	473	183	603	19.008	7066.76	-

Consequently, for every distinctive applied input, training data is reduced based on the set priority. The implementation of the proposed data reduction proposed in the methodology shown in Figure 2.4 is summarized in the list below along with the data excluded with each of the three reductions:

1- *Reducing entries based on output quality and priority*: 73.11% of data instances are excluded.
2- *Entries with high accuracy*, e.g., PSNR $\geqslant 70$, yet a design with higher priority, i.e., lower PADP, offering similar or higher accuracy: additional 9.96% of the data is eliminated.
3- *Entries with very low accuracy*, e.g., PSNR $\leqslant 15$: additional 7.29% of the data is reduced.

In total, 90.36% of the data is eliminated resulting in approximately 3.24×10^7 instances, down from 3.35×10^8. Table 3 shows the remaining number of training entries for each of the 20 designs after pre-processing, i.e., quantizing then reducing, data. From Table 3, it can be noticed that the design with (Type = AMA5, Degree=D4), i.e., *Design 19* in Table 8 has the most number of entries after data pre-processing. This design is given a priority of 6 as shown in Table 2 and thus has one of the lowest PADP. Moreover, it can be noticed that designs with low priority, e.g., priorities 15 to 20, have minimal to zero entries after data pre-processing as shown in Table 3. Finally, it must be noted that this data pre-processing eliminated 7 designs, i.e., zero training entries, from the library of AC designs, since other designs offer similar or better quality with better PADP.

Table 3: The Number of Training Entries of each Approximate Design After Pre-processing

	AMA1	**AMA2**	**AMA3**	**AMA4**	**AMA5**
Degree1	264	12,677	7	441,404	1,859,393
Degree2	0	0	0	29,437	315,541
Degree3	0	0	0	75,752	16,761,883
Degree4	0	1,315,321	1,493,624	4,987,277	2,230,190

2.4.3 Building the ML-Based Design Selector

Based on the proposed methodology of Figure 2.4, the quantized and reduced training data was sent to the ML-algorithm, in order to build an ML-predictor, that would select the design that is projected to meet the TOQ. The research conducted in [30] found out that the usage of decision tree (DT) as ML-algorithm proved superiority compared to linear regression (LR), random forest (RF) and neural network (NN) for the ML-based quality assurance model. Furthermore, over-fitting of training data is needed since all possible combinations have been generated when simulating the designs. Thus, an ML-algorithm that delivers an over-fitting of training data is needed. Therefore DT has been chosen and the best algorithm for the ML-based design selector. The DT-modeling was constructed using Classification Learner Toolbox in Matlab [33]. The generated DT-predictor that will select the appropriate design is depicted in Figure 2.5, with an accuracy of 83.9%. Moreover, the data was simplified in order to create a lightweight ML-predictor, and indeed this goal was achieved, since predictions can be achieved with a minimum of 5 nodes, and a maximum of 9 nodes. Moreover, it is remarkable that when data pre-processing is applied, many designs were excluded, either for having 0 entries, or a small contribution as shown in Table 3. However, with such enormous unbalanced contribution for different designs, it must be noted that the ML-algorithm tend to disregard training data entries with small contribution, unless a true-fit predictor is set. Nonetheless, a true-fit predictor would result in unnecessary overhead by the predictor, since the ML-model will have further nodes and quality improvement that could be intangible.

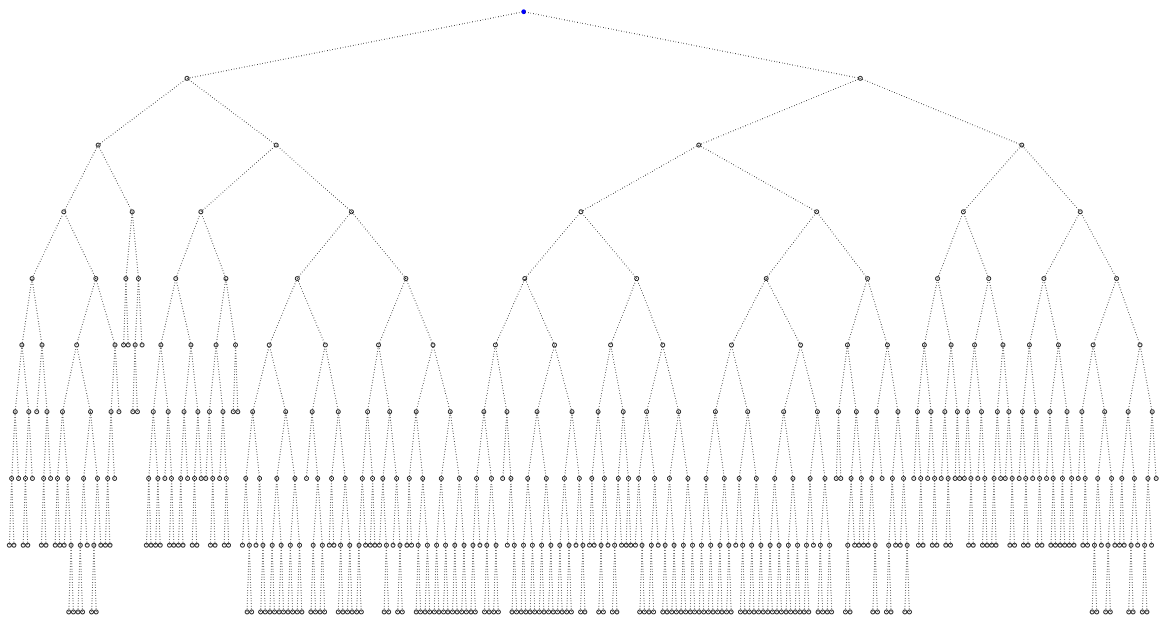


Figure 2.5: The Structure of the Constructed DT-Model

2.4.4 Quality Check

In order to assess the design quality, we now test the generated model that uses the DT-predictor shown in Figure 2.5. For the experimental results, we have chosen multimedia blending applications based on multiplication as they are known to be error-tolerant. The two multimedia services chosen are audio and image, with 2×10^9 and 12×10^6 unique inputs, for each of the two applications, respectively. The experimental work is executed with a variation in the TOQ, i.e., PSNR, and a variation in the numbers of samples used, which are provided to the DT-predictor. The number of samples vary from 1 sample, to the maximum number of 2^n samples. For each run, the predictor is used once, since the samples are averaged, and thereby, the prediction is based on this value. To proceed with the multimedia processing, one should understand the physical nature of each of the applications, and its digital representation. Thus below, the physical behavior is explained, along with their corresponding digital construction:

1- *Characteristics of Audio Files*: Sounds can be stored as digital audio in a series of bitstreams, with each bit representing the physical wave length, i.e., amplitude. With the use of 16-bits per sample, the amplitude is represented in a more accurate representation, as a wider range is covered $\pm 2^8$. The database of WAV sound files available in [9], is a good resource for researchers, conducting experiments in the field of audio-processing.

2- *Characteristics of Image Files*: Similar to audio, images are constructed of the mix in three channels, i.e., Red, Green and Blue (RGB). The 16-bit representation for images, means that each pixel can take one of 2^{48} unique combination of colors, since each of the three channels has a set of 2^{16} unique colors, i.e., 2^{16} different red colors. A library of raw images found in [2] is used in order to have access to 16-bit images.

The simulation is conducted with a target PSNR ranging from $15\,dB$ up to $70\,dB$. Furthermore, since the ML-selector requires runtime inputs to predict the appropriate design, switching designs for every computation would defeat the purpose of using AC. Thus, in order to reduce the overhead of switching among designs, the images and audio files are sampled with the runtime input being the average of the samples. The samples are taken in the form of 2^n with n varying from 0 up to 12 and 17 for image and audio processing, respectively. The patterns used to collect 2^n samples from audio files and images are shown in Figures 2.6 and 2.7, respectively. As these figures depict, the space is divided after every iteration with the samples taken from the middle of each of the sub-spaces resulting from the division.

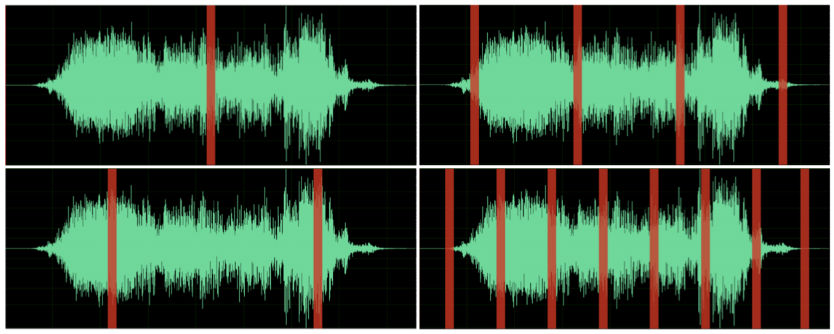


Figure 2.6: Pattern used to Sample Audio Files

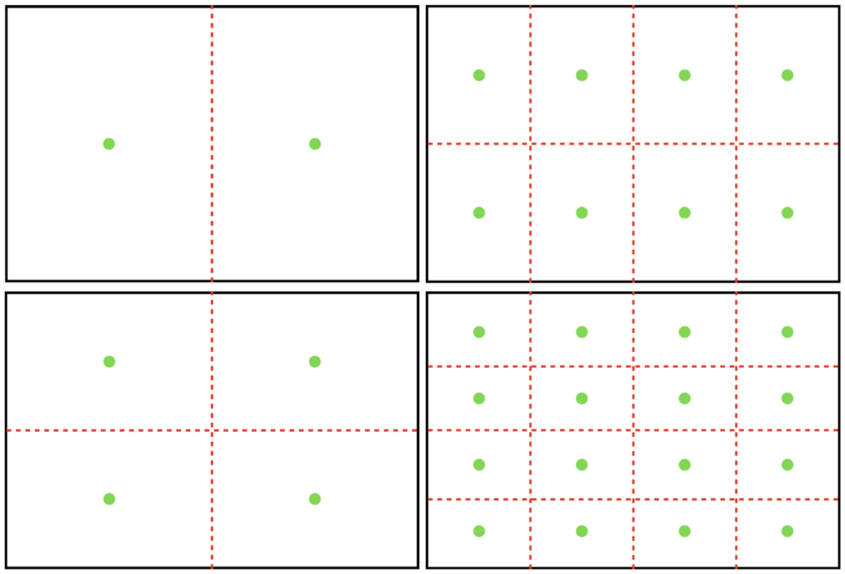


Figure 2.7: Pattern used to Sample Images

Multiple runs of audio and image processing are conducted, and the output quality is monitored. Afterwards, the average quality based on the number of samples taken is computed. Figure 2.8 shows the quality obtained for audio processing when the various 2^n samples are collected with n varying from 0 to 17. Figure 2.9 shows similar data for image sampling when collecting 2^0 to 2^{12} samples, i.e., n varying from 0 to 12. Moreover, both figures provide the target output quality (TOQ) provided to the

ML-predictor. As shown in Figure 2.8, the audio blending resulted in a measured quality that is better or equal to the chosen TOQ. On the other side, we can notice from Figure 2.9 that the actual quality was better than the chosen TOQ for most of the cases. Furthermore, the simulation shows that additional numbers of samples does not always imply an improvement in quality. Thus, the number of samples to be taken must be carefully selected in order to minimize the overhead of sampling. Finally, it must be noted that the reduction of training data we proposed in this chapter improved the output quality. This can be noticed in Figures 2.8 and 2.9 since the actual quality is higher than the TOQ, while the work in [32] resulted in an actual

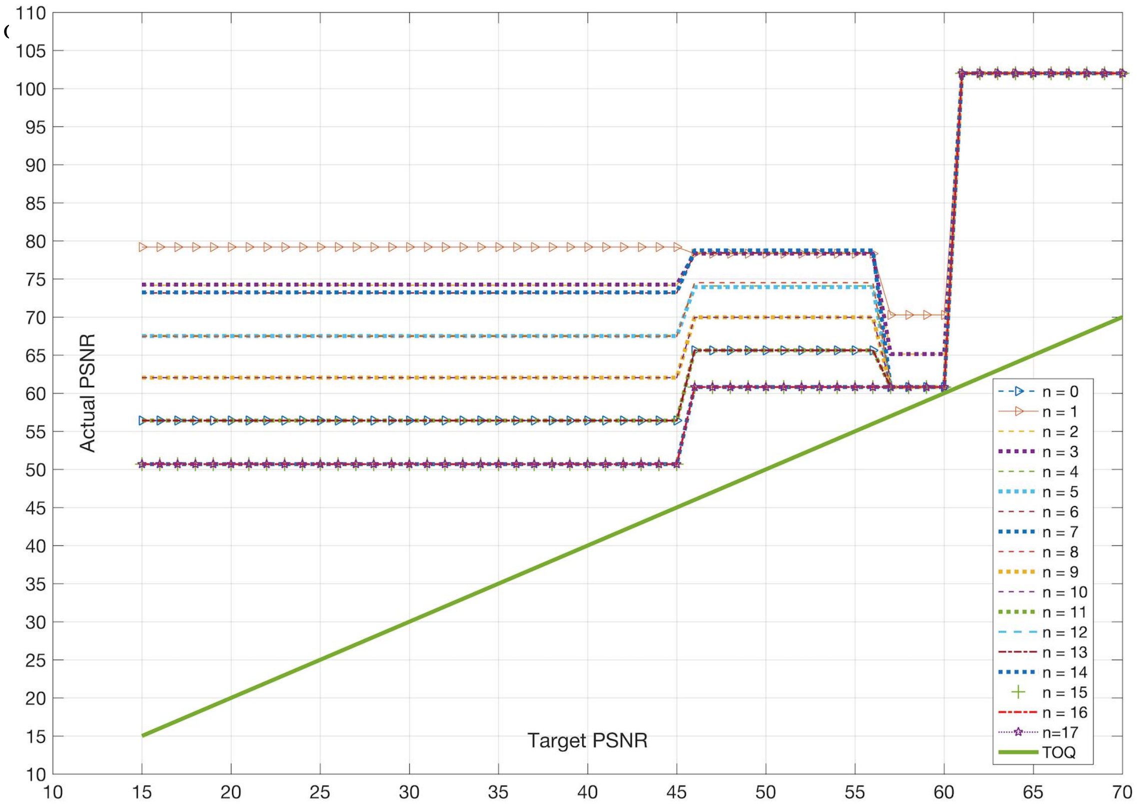


Figure 2.8: TOQ versus Obtained Output Quality for Adaptive Audio Blending

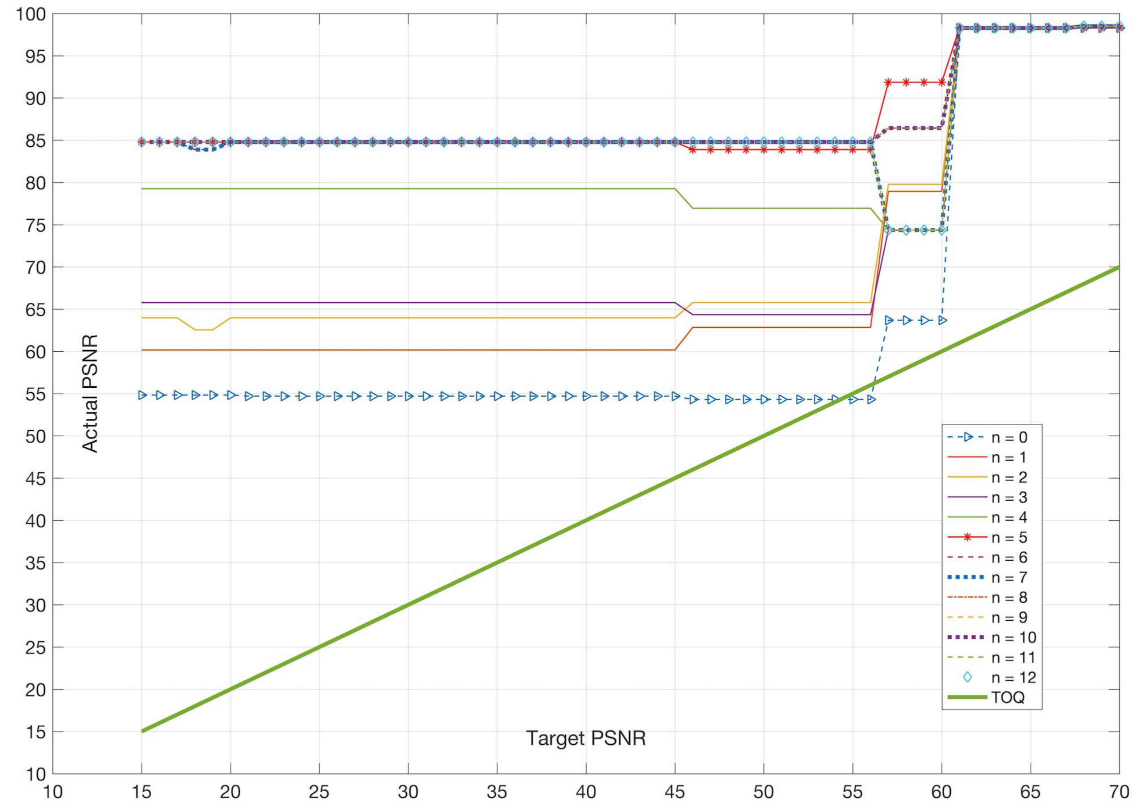


Figure 2.9: TOQ versus Obtained Output Quality for Adaptive Image Blending

2.5 Summary

In this chapter, we studied the extension of previously proposed ML-based quality assurance [32]. Such extension is hard when dealing with larger designs, e.g., 16-bit multiplier instead of 8-bit. Nonetheless, the proposed methodology in Figure 2.4 uses an HPC server [52] to reduce simulation time and introduce, a new reduction to the training data. Using this proposed methodology, the ML-based model was successfully extended to support 16-bit multipliers. Moreover, the extension to 16-bit model was achievable, however with today's computation power, simulating a 32-bit models or larger is unpractical as it would require an indefinite amount of time. Moreover, it is remarkable that the data pre-processing proposed in this chapter resulted in extensive reduction of training data. This reduction resulted in a simple DT-predictor which has a small overhead along with an improved output quality.

Nonetheless, the output quality of the ML-based quality assurance could be enhanced, if the library of AC designs used is improved. A viable approach to improve the library is an implementation of a DSE. Such study could also result in a design that has an improved metrics in the four dimensions of AC design. If such design is found, it could be a good solution for circuits where ML-based quality assurance results in undesirable overhead, e.g., IoT. Moreover, extension to larger designs, e.g., 32-bit multiplier, could be feasible if the time to assess the quality is reduced. Towards this goal, in the next chapter we propose a methodology of a DSE for AC designs that allows such achievements.

Chapter 3

Design Space Exploration for Approximate Circuits

In this chapter, a methodology for design space exploration (DSE) for AC circuits is proposed. The methodology is based on a new mathematical modeling for circuits, which allows for a faster assessment of logical circuits.

3.1 Introduction

Research of AC circuits has been concentrated on modifications at the architecture level. Furthermore, the reported AC architectures such as the work in [32] [43] have been explored in a small number of structural configurations. Some of the reasons why the proposed designs are studied in a limited number of configurations are: 1) common fallacies of circuit approximation; and 2) time required to asses large set of configurations. In order to illustrate some of the common fallacies and interpret the proposed DSE for AC, array multipliers are used in this chapter.

One common fallacy is an effort to approximate least significant bits (LSBs) only. For instance, in order to approximate the array multiplier shown in Figure 3.1, techniques have been based on swapping exact sub-blocks in the right columns with approximated ones, e.g., replacing exact full adders (FAs) in column 1 to column 15 with approximated FAs. A good example of this approach's adoption is AUGER tool [22] and the work in [32]. The objective of this approach is the desire to approximate LSBs only, which will result in a negligible impact on the output quality. Nonetheless,

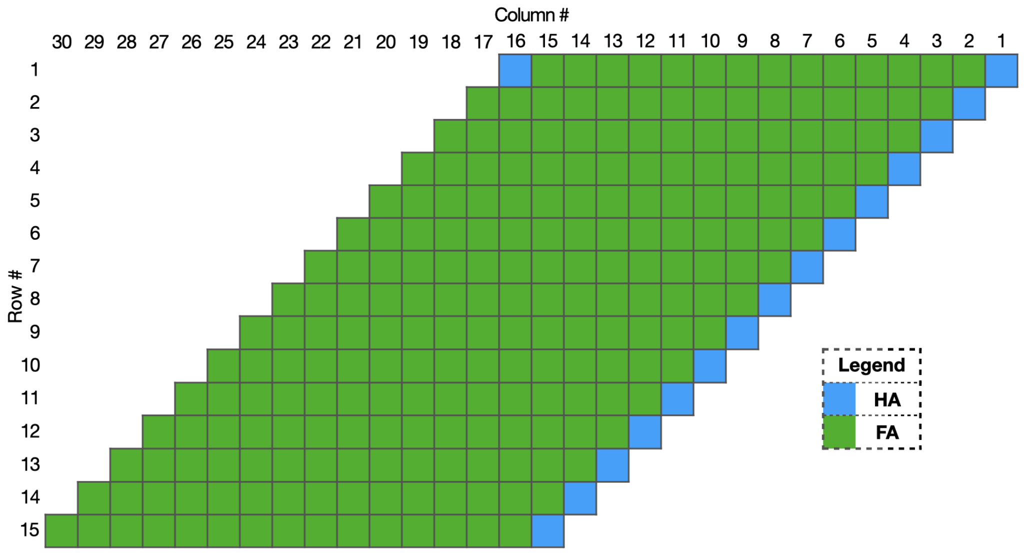


Figure 3.1: Architecture of a 16-bit Array Multiplier

no study has previously shown a direct relation between output quality and the position of an approximate sub-component in an AC circuit. Hence, the current approach used to generate approximate configurations is not accurate as the generated error can always propagate and has an effect on the most significant bit (MSB) as well.

Moreover, taking into consideration all possible variations in the previously proposed AC designs will generate multi-trillion configurations. Nonetheless, since a limited number of configurations has been studied, a DSE is indispensable as it can spot a design with better metrics, i.e., area, delay, power and quality. However, a DSE requires quality assessment, which can be time consuming if conducted using excessive simulation when assessing large models, e.g., 16-bit multiplier.

In this chapter, we propose a new DSE methodology, which will allows a vast exploration of AC circuits that satisfy designers' requirements, i.e., area, delay, power and quality. This procedure will reduce the multi-trillion possibilities to a few hundreds of candidate circuits. The proposed methodology is shown in Figure 3.2. As the figure depicts, the designer must choose the AC architecture, e.g., array multiplier, and set the aimed area, delay and power usage. In addition, the designer must identify interchangeable sub-blocks, e.g., FAs, and provide a list of candidates of approximate sub-blocks along with their synthesis. Based on the designer inputs, we will reduce the design space in two phases. The first phase is position independent, i.e., optimizing area and power, regardless of the position of the FA or HA in the array. However, the second is position dependent, i.e., optimizing delay, by using a genetic algorithm (GA). Such approach is needed since even after the first reduction, varying positions of sub-blocks, e.g., FA, could change the latency of the functional unit. Once the design space is reduced, a quality assessment for candidates is required. Thereafter, we use mathematical modeling to establish a quality assessment. This modeling requires generation of the Boolean output functions of each configuration. The Boolean functions are then converted to a single decimal function. This conversion is applied to each of the candidates and will result in a list of decimal functions. The generated functions are then assessed using mathematical analysis, e.g., ED and derivatives. Afterwards, the assessment is evaluated in order to check compliance with the target quality requirement identified by the designer. If the design satisfies the requirements, the model can be carried for implementation; otherwise, a further DSE can be conducted.

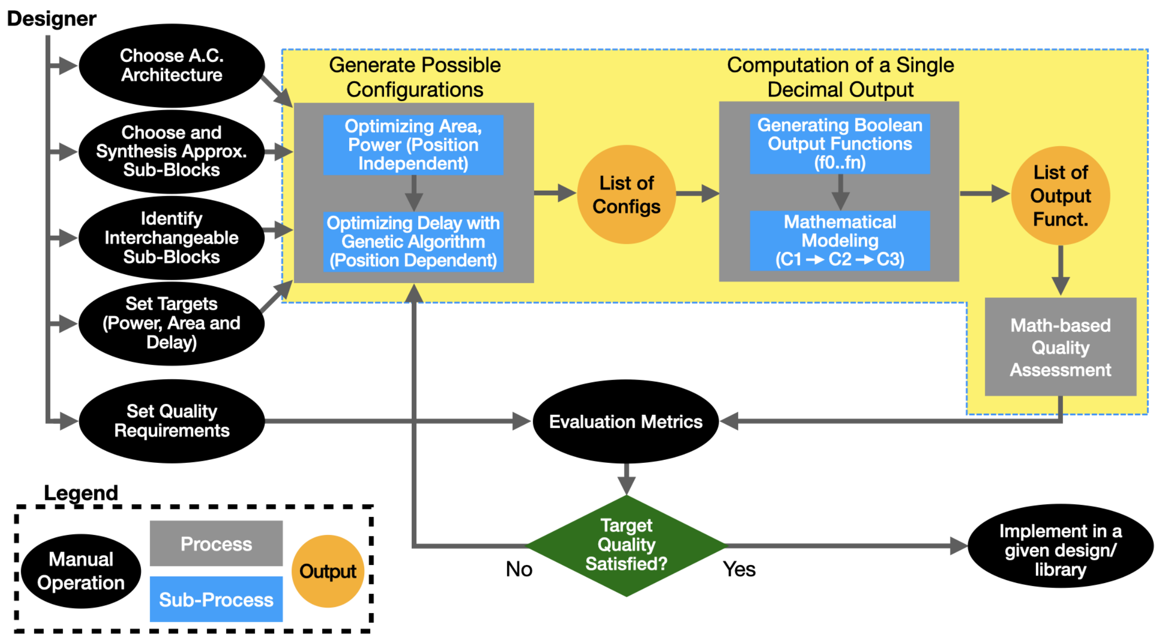

Designer
Choose A.C. Architecture
Choose and Synthesis Approx. Sub-Blocks
Identify Interchangeable Sub-Blocks
Set Targets (Power, Area and Delay)
Set Quality Requirements
Generate Possible Configurations
Optimizing Area, Power (Position Independent)
Optimizing Delay with Genetic Algorithm (Position Dependent)
List of Configs
Computation of a Single Decimal Output
Generating Boolean Output Functions (f0..fn)
Mathematical Modeling (C1 → C2 → C3)
List of Output Funct.
Math-based Quality Assessment
Evaluation Metrics
Target Quality Satisfied?
No
Yes
Implement in a given design/ library
Legend
Manual Operation
Process
Sub-Process
Output

Figure 3.2: Proposed Design Space Exploration for AC Design

In the remaining of this chapter, we will detail the proposed DSE. Afterwards, five well-known mirror-based approximate FAs [17] are used for the DSE of a 16-bit array multiplier. Furthermore, unlike previous works [32] [43], different types of approximated FAs are used in a given multiplier. Moreover, the accuracy of the generated multipliers are assessed based on image and audio blending applications. Finally, a brief study on different AC architectures is exhibited.

3.2 Design Space Reduction

Finding optimal AC circuits for large arithmetic units, e.g., 16-bit multiplier, is challenging since these circuits can become complex, and many variations can occur. For instance, an n-bit array multiplier contains $[(n-1)^2 - 1]$ FAs and n half adders (HAs). For the rest of this chapter, the HA is assumed to remain indifferent. For a case of a 16-bit array multiplier, with five types of approximate FA to be used, $5^{224} = 3.7 \times 10^{156}$ possible configurations could be generated. Studying all potential configurations is unrealistic with today's computers and could take forever. For this reason, we use a two-phases reduction which can be classified in as follow:
1- Optimizing area and power (position independent)
2- Optimizing delay using genetic algorithm (position dependant variable)

Since the proposed optimizations are based on the configuration characteristics, i.e., area, delay and power, a synthesis is required. However, performing synthesis using third-party tools, e.g., Synopsys Design Vision [51] and Xilinx Vivado [57], may result in communication overhead among the tools and the DSE. In addition, synthesis using such tools can actually be slow especially when certain factors, such as routing optimization, are considered. For this reason, a fast synthesis that neglects advanced factors, e.g., optimized routing, is proposed. Towards this goal, synthesizing sub-blocks, i.e., FA, must be conducted first. The results of the synthesis, i.e., area, delay and power, are then used as a sum of product to estimate area and power, while the delay is assessed as a lumped value of sum or carry signals. For instance, for the array multipliers in Figure 3.1, the propagation delay of the circuit is computed by first evaluating the time to generate the sum and carry for each series of FAs in the first row. Thereafter, in each of the subsequent rows, the latency of each FA is added to the arrival time of the slower input signal, i.e., sum signal from the top-side and carry signal from the right-side.

The rest of this section will show the steps of our proposed methodology, by first synthesizing sub-block, then finding candidate circuit configurations that meet targets, i.e., area and power. Afterwards, configurations meeting target delay are retrieved with the help of a proposed genetic algorithm. Finally, the generated designs are modeled using arithmetic formulation. The models are used for quality assessment and final synthesis. For the sake of simplification, for the rest of this chapter, the symbols n, q and T will be used as a reference to input bit-width, quantity of a given FAs and target metric, respectively.

3.2.1 Position-Independent Constraint

The area and power of a given array multiplier are position-independent of the sub-components but only relate to n and q. This constraint can be illustrated in the the array multiplier shown in Figure 3.1 when a given approximate FA is placed in a different position, i.e., change of row and/or column, where the total area and power are assumed to remain constant. Since the position does not count, search space is a combination of unordered sampling with replacement [55]. Following this approach, one or more circuit configuration(s) would fall under a single category, since they share the same metric, i.e., total area or total power. Thus, design space reduction is performed at an abstract level by eliminating categories not meeting the target metric. Subsequently, two sets of candidates, namely S_A and S_P, are formed where each set satisfies its corresponding target, i.e., area and power. Finally, a set named S is constructed by finding the intersection of both S_A and S_P. This step is essential since we have to satisfy the area and power requirements simultaneously.

3.2.2 Position-Dependent Constraint

Unlike area and power, delay constraint depends on the position of sub-blocks, e.g., FA, since the changes in circuit configuration could result in different propagation delay. For a given candidate chosen from the set S satisfying both area and power constraint, the total number of unique configurations that can be formed by relocating sub-blocks follow the form of permutations with repetition. This permutation will result in one possible configuration if all sub-blocks are of the same type. On the other hand, this permutation would generate the maximum number of unique configurations if the load is evenly distributed through different types of sub-blocks, i.e, $q_0 = q_1 ... = q_n$. For the case of 16-bit a array multiplier with five types of approximate FAs, the maximum would be: $max = \frac{224!}{45!\times45!\times45!\times45!\times44!} \approx 1.03 \times 10^{152}$. Given the fact that a candidate set can have a large number of possibilities, studying all potential circuit configurations can be time consuming. To overcome this challenge, the first step is to understand the behavior of the delay. Towards this goal, the delay is monitored for a chosen set when the positions of sub-blocks are shuffled multiple times. Figure 3.3 shows the frequency of occurrence of a given delay for randomly generated configurations, i.e., random shuffling positions of sub-blocks. From Figure 3.3 it is noticeable that the delay has a normal distribution characteristic. For the sake of simplification, the delay is always assumed to follow a normal distribution trend when generating random design structures. Thus, at least half of the possibilities could be disregarded, since the chances of finding a circuit that has good quality, yet a delay less than the average are great (half of the set-space). With this assumption, the next step is to find an appropriate method that finds the circuit configurations meeting the target delay. For this, we propose an innovative method based on a genetic algorithm (GA). Moreover, this method will allow the designer to control the time spent searching for the appropriate circuit. The proposed GA consists of the following main steps:

1- Randomly placing sub-blocks.

2- Search for the longest ($\boldsymbol{L}$) and shortest ($\boldsymbol{S}$) paths and find the propagation delays D_L and D_S for ($\boldsymbol{L}$) and ($\boldsymbol{S}$), respectively.

3- Find uncommon FAs in ($\boldsymbol{L}$) and ($\boldsymbol{S}$) paths.

4- Swap an uncommon FA that is known to have high latency from ($\boldsymbol{L}$) with low latency FA from ($\boldsymbol{S}$) paths.

5- If the new D_L becomes larger, revert the modification and repeat step 4 with another combination of FA.

6- Loop to step 2 for α-iterations.

7- If $D_L < T_D$ (target delay), then save the circuit configuration, and loop to step **1** for β-times.

8- Sort and save all circuit configurations or few best in case of a large set.

Using this genetic algorithm will intensely reduce the time finding circuits meeting the target delay. In addition, designers can control the search time by changing α and β which are the loop bounds for the genetic algorithm.

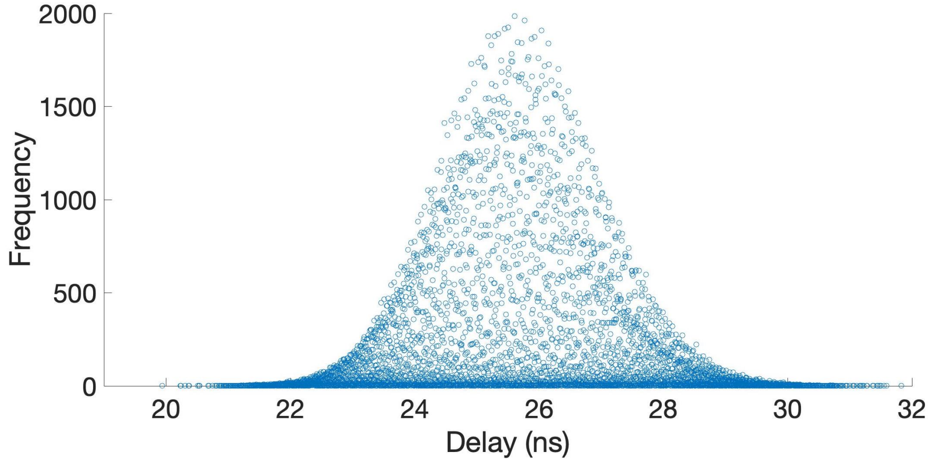


Figure 3.3: Frequency of Delay for 1 Million Randomly Generated Configurations

3.3 Mathematical Modeling

The quality assessment of AC circuits has been mainly based on excessive simulation [22] [28] [30]. However, as proven in the previous chapter, this can consume a large amount of time, which would make the usage of such method impractical. Furthermore, research conducted in [35] offered an alternative to assess quality based on bit-error. Nonetheless, the proposed method can fall short in offering a relevant assessment. Moreover, its application can fail if smaller blocks are not modeled at first hand, and then generalized.

In this section, a mathematical modeling that can be applied to any logic circuits (AC circuits included) is proposed. This model will generate a function in the form of $Z = g(X, Y)$, where X and Y are the applied inputs, and Z is the expected output. The function $g(X, Y)$ can be used to assess quality. For instance, since a similar function $f(X, Y)$ for exact computation is already in place, e.g., $f(X, Y) = X \times Y$ for exact multiplication, then $ED = |f(X, Y) - g(X, Y)|$. To achieve this goal, the following mathematical models are needed:

1- An order 1, *O(1)*, Decimal to Binary converter.

2- Equivalence of logic gates in arithmetic calculus.

3- An *O(1)* Binary to Decimal converter.

If the above mathematical models are in place, then any logic circuit can have a mathematical representation that offers an arithmetic-based relation between inputs and output. The binary to decimal conversion currently in place is enough, i.e.,

$(D)_{10} = \sum_{i=0}^{n} Z_i \times 2^i$ with $(D)_{10}$ being the decimal number while Z_i is a binary number. Thus, a work that achieves **1** and **2** from the list above is needed. The following subsections will focus on the decimal to binary conversion and logic gates equivalence.

3.3.1 Decimal to Binary

Conversion from base 10 to base 2 can either be done by dividing the number in base 10 by two and checking for the remainder with the first output being the LSB, or subtracting the biggest possible power of 2 with the first output being the MSB. In this work, the first approach is used. When dividing by 2, the remainder will be either 0 or 1, which would identify the n-th bit computed. Moreover, a remainder 0 is equivalent to the number being even; otherwise, it would be an odd number. On the other side, this provides an interesting property when the number is divided by two. This property can be noticeable in the fraction part of a number x, i.e., $frac(x) = x - \lfloor x \rfloor$. It can be noticed that when dividing a number (integer or floating-point) by two, the result of $frac(x)$ will belong to one of two intervals. If the number is even, then $frac(x) \in [0; 0.5[$ while the division of odd numbers would result in $frac(x) \in [0.5; 1[$. Thus, this interesting property can be used to form a 1:1 relation between the binary conversion and $frac(x)$.

Since $frac(x)$ will always belong to one of the two fixed domains, then a periodic function can form a relation for decimal and binary representation. The trigonometric function *sin* is a periodic function, where its period can be manipulated. Furthermore, any function or number, if divided by its absolute value (given that the absolute value is none-zero), the result will be either -1 or 1. Using these notions, the function $h(x)$ below is proposed.

$$h(x) = -\frac{sin(2\pi x)}{|sin(2\pi x)|} \tag{1}$$

The graphical representation of $h(x)$ is shown in Figure 3.4. It can be noted that the proposed $h(x)$ function will provide a constant value, i.e., 1 and -1, for all x. Moreover, the constants have the a similar pattern compared to $frac(x)$ for all values in the domain of x.

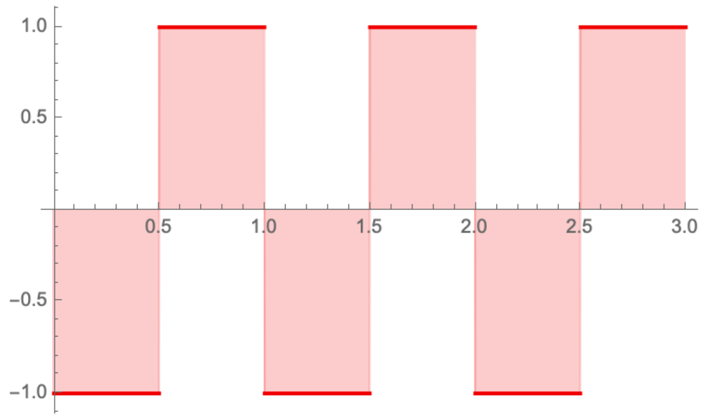


Figure 3.4: Graphical Representation of the proposed $h(x)$

Since the binary numbers are 0,1 instead of -1,1; and x is a result of division by 2^n, the function $h(x)$ proposed in Equation 1 shall be transformed. In this case, x shall be replaced with a division of the number X by 2^n. Furthermore, the equation is undefined if $frac(x) = 0.5$ or $frac(x) = 1$. For this reason, a small positive number, i.e., ε shall be added to the argument of sin. The equation that adapts to all these conditions is shown in Equation 2. Thus, the n-th bit representation of a decimal number X can be computed by replacing the corresponding values in Equation 2.

$$k(X) = 0.5\left(1 - \frac{sin\left(\frac{\pi X}{2^n} + \varepsilon\right)}{\left|sin\left(\frac{\pi X}{2^n} + \varepsilon\right)\right|}\right) \tag{2}$$

In Table 4 the conversion of $(19)_{10}$ to its binary representation is shown. The binary output, can be computed by replacing $(X/2^n)$ with its value in Equation (2). It can be noted that at a certain point, the division by 2^n will result in a zero when the proposed conversion is used. The conversion using Equation (2) from decimal to binary has been tested for all $X \leq 2^{16}$, and showed compliance with traditional methods of conversion.

Table 4: Decimal to Binary Conversion for $X = 19$

$\boldsymbol{n}$	1	2	3	4	5	6	7	8
$X/2^n$	9.5	4.75	2.375	1.1875	0.59375	0.29688	0.14844	0.07422
$frac(X) \in [0; 0.5[$	No	No	Yes	Yes	No	Yes	Yes	Yes
$frac(X) \in [0.5; 1[$	Yes	Yes	No	No	Yes	No	No	No
Binary Output	1	1	0	0	1	0	0	0

3.3.2 Logic Operators Equivalence

After the decimal to binary conversion, the next step is to figure out the Boolean operators equivalence in arithmetic calculus. For instance, the Boolean operation $Z_0 = X_0 \vee Y_0$, Z_0 can be written in terms of the decimals X and Y by converting X_0 and Y_0 using the Decimal-Boolean equivalence proposed in the previous section. Nonetheless, Z_0 cannot be considered an arithmetic equation, since the *OR* operator is logic. Towards this goal, we propose the equivalences in Table 5. Using these equivalences, Z_0 can now be converted to pure arithmetic equation, i.e., $Z_0 = X_0 + Y_0 - (2 \times X_0 \times Y_0)$.

Table 5: Logic-Arithmetic Equivalence

Gate	Boolean Operator	Arithmetic Equivalence
Inverter	$!A$	$(1 - A)$
AND	$A \wedge B$	$A \times B$
OR	$A \vee B$	$A + B - (A \times B)$
XOR	$A \oplus B$	$A + B - (2 \times A \times B)$

However, the proposed equivalences for *OR* and *XOR* will double the size of the equation if used recursively. Hence, the proposed methodology suggests the use of sum of product (SOP) equations, where *OR* operators are converted to inverted *AND*. For example, $A \vee B \vee C$ can be converted using the following Boolean equivalences $A \vee B \vee C = \overline{\overline{A \vee B \vee C}} = \overline{\overline{A} \wedge \overline{B} \wedge \overline{C}}$. The equivalence of $\overline{\overline{A} \wedge \overline{B} \wedge \overline{C}}$ in arithmetic calculus can now be written as $\{1 - [(1 - A) \times (1 - B) \times (1 - C)]\}$. The advantage of this approach is obvious compared to the recursive usage of the proposed *OR* equivalence in Table 5, which doubles the size of the equation after every iteration.

3.4 Experimental Results

In this section, the proposed DSE is applied on a 16-bit array multiplier and the obtained results are discussed. Moreover, this section studies the extension of the proposed DSE to other designs, such as (i) 32-bit and 64-bit array multiplier; (ii) multiply and accumulate (MAC) units [29]; and (iii) approximate divider [20].

3.4.1 Applying the Proposed DSE on a 16-bit Array Multiplier

The proposed methodology of DSE shown in Figure 3.2 is applied on 16-bit unsigned array multiplier since it does offer a wide range of variations. Five well known approximate FAs proposed in [17] denoted as AMA1, AMA2, AMA3, AMA4, and AMA5 have been chosen for this implementation. The used approximate FAs and the HA were synthesized in CMOSP18/TSMC [13] using Synopsys Design Vision [51]. The result of the synthesis is represented in Table 6. Moreover, the synthesis showed that unlike exact FAs, the generation of the carry could be faster than the generation of the sum in the approximate FAs, e.g., the carry of AMA3 takes 0.26 ns while the generation of the sum takes 0.2 ns.

Table 6: Synthesis of used FAs

Design	Power (μW)				Area (μm^2)	Delay (ns)	
	Internal	Switching	Leakage	Total	-	Sum	Carry
AMA1	13.4	6.93	2.99E-03	20.4	77.24	0.58	0.25
AMA2	83.9	4.42	1.52E-03	12.8	48.78	0.25	0.36
AMA3	3.6	2.69	1.33E-03	6.29	24.39	0.20	0.26
AMA4	4.31	3.76	1.5E-03	7.97	32.52	0.21	0.1
AMA5	1.8	1.31	3.34E-04	3.1	16.26	0.05	0.05
HA	9.57	1.91	2.1E-06	11.5	48.78	0.42	0.13

Using these results, a weighted target for power and area is chosen with the weight distributed among the five FAs. Moreover, a small margin of $\pm\varepsilon$ is added to the targets to assure a formation for the joint set S. With the chosen area and power targets, 112 unique combinations are generated. Then, the proposed genetic algorithm for delay optimization is implemented to find the circuit configurations that satisfy the timing constraints. Each candidate of the sample space S is shuffled 50,000 times, i.e., $\beta = 50,000$ in the proposed GA. For the improvement process, i.e., looping to step 1 in the proposed GA, the loop is applied 100-times. However, the chosen target delay turned relatively high, since many candidate configurations are found. Hence for the sake of simplification, only the fastest 2 circuits are carried for quality assessment. This led to the generation of 224 candidate configurations. Using Mathematica [3], simplified Boolean equations for each of the 32 output bits are found and converted to decimal equations using the proposed technique. Once output bits are converted to decimal format, a function $g(X, Y)$ is generated. Finally, the quality of AC circuits is assessed. On the other side, if these models are to be assessed on the same computer used previously for benchmarking, they would have taken approx. 4,480 days (or almost 12 years) using excessive simulation. were

After carefully assessing the quality for all candidate circuits, no model could meet the target accuracy -Golden Goal-, yet many multipliers are identified to operate well in a given region from the domain space. Such multipliers can be considered as a good opportunity for applications where the occurrence of inputs in that region is high. Moreover, the fact of not finding a multiplier that has an absolute quality is expected since the test set included 224 models only. The tested set can be considered a microscopic set when compared to all possible circuit configurations.

Furthermore, out of the 224 configurations, 10 designs are selected for implementation on real-world applications. For this purpose, multimedia applications, i.e., audio and image blending, are considered. Six combinations of pictures and six blends of audio were retrieved from [2] and [9], respectively, and then put under test. The results of the simulation for the 10 selected designs are summarized in Table 7. The average PSNR ranges between $51.63dB$ and $73.55dB$. The chosen results showed concurrence with the quality assessment performed previously. Moreover, with an average PSNR $= 73.55dB$ of Design 3 in Table 7, a broader study for configurations including different types of sub-blocks is needed, since a good quality can be achieved.

Table 7: Simulation-Based Quality Assessment of Chosen Designs

Type	**PSNR** (dB)					
	Run #	Design 1	Design 2	Design 3	Design 4	Design 5
Image	1	58.86	58.34	69.01	50.11	66.43
	2	64.62	63.80	71.05	49.68	69.61
	3	61.77	61.41	65.58	50.07	64.88
	4	58.88	58.06	68.38	49.20	67.08
	5	58.50	58.72	63.26	49.41	62.85
	6	64.16	64.28	67.03	49.91	65.61
Audio	1	59.78	58.65	81.74	50.63	78.73
	2	59.54	59.27	74.41	54.84	72.66
	3	54.96	54.98	81.37	57.47	72.46
	4	55.95	57.39	86.41	56.07	82.57
	5	55.42	54.31	74.26	51.03	72.06
	6	57.80	57.23	80.08	51.17	73.58
Average	-	59.19	58.87	73.55	51.63	70.71

Type	**PSNR** (dB)					
	Run #	Design 6	Design 7	Design 8	Design 9	Design 10
Image	1	60.20	65.74	61.14	53.07	50.59
	2	66.43	69.14	66.47	56.71	51.64
	3	62.71	64.75	62.96	57.04	51.70
	4	61.42	67.21	61.73	52.58	50.06
	5	60.35	64.67	62.05	52.76	50.05
	6	64.39	66.36	66.17	57.36	50.71
Audio	1	61.18	77.84	69.23	50.65	50.56
	2	61.40	70.56	66.01	54.90	54.66
	3	55.59	67.88	63.35	57.48	57.13
	4	57.26	78.45	68.44	56.09	56.00
	5	57.64	71.70	60.81	51.13	50.83
	6	58.87	72.38	64.32	51.27	50.75
Average	-	60.62	69.72	64.39	54.25	52.06

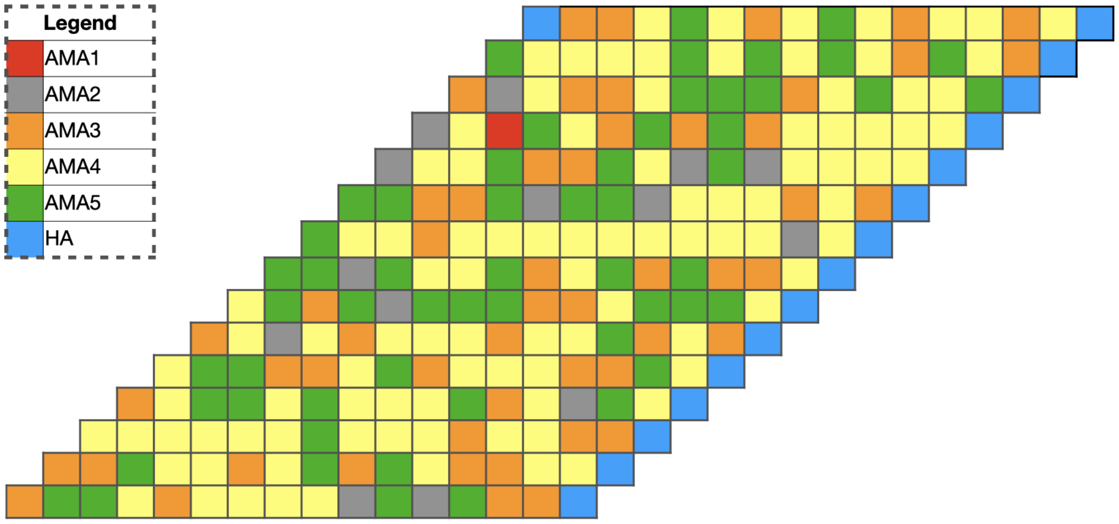


Figure 3.5: Configuration of Array Multiplier Consisting of Multiple types of Approx. FAs

Figure 3.5 depicts the configuration of the design that scored the highest average PSNR, i.e., Design 3 in Table 7. Moreover, the circuit scoring the highest PSNR is synthesized using Synopsys Design Vision. The synthesis documented savings of 76.63% and 56.72% are achieved in power and area, respectively, in comparison to the exact array multiplier.

Finally, a noticeable error is observed when comparing power estimated using the proposed fast synthesis and the result from third-party tools, e.g., Synopsys Design Vision [51]. This could be explained by the fact that power loss due to wiring is not included in the proposed estimation of power, yet in big circuits this loss can be significant. The marginal error resulting from the use of the proposed power synthesis is 58.72% and 41.47% for the exact and the approximate circuits, respectively. Thus, the accuracy of the proposed synthesis can be improved by error-compensation, e.g., adding 50%, and hence reducing the gap in error. On the other side, the marginal error for estimated area turned out to be less than 0.2%.

3.4.2 Extending the Proposed DSE to other AC Designs

In the previous subsection, we studied the implementation of the proposed DSE on a 16-bit unsigned array multiplier. However, this type of approximation is one of many proposed architectures. For this purpose, we study the extension of the proposed DSE to other architectures, namely a multiply and accumulate (MAC) unit proposed in [29] and an approximate divider proposed in [20]. Moreover, an extension to larger data-width, e.g., 32-bit and 64-bit array multiplier, is another characteristic of the DSE that we examined.

Approximate MAC Unit

Since the array multiplier can be considered as a straight forward architecture that is formed by connecting the sum and carry of FAs to inputs of other FAs, we would like now to study more complex architectures. For this purpose the we consider the usage of the MAC architecture which consists of small multipliers interconnected to adders as shown in Figure 3.6 [24]. Nonetheless, if the MAC is formed out of smaller array multipliers and carry-ripple adders, its structure would mainly consist of FAs. Thus, the previously proposed optimizations, i.e., position dependent and position independent, are still applicable. Hence, the only process that has to be tested is quality assessment using the proposed mathematical modeling. In this direction, the quality assessment of a given configuration of 16-bit approximate MAC is conducted as a proof of concept.

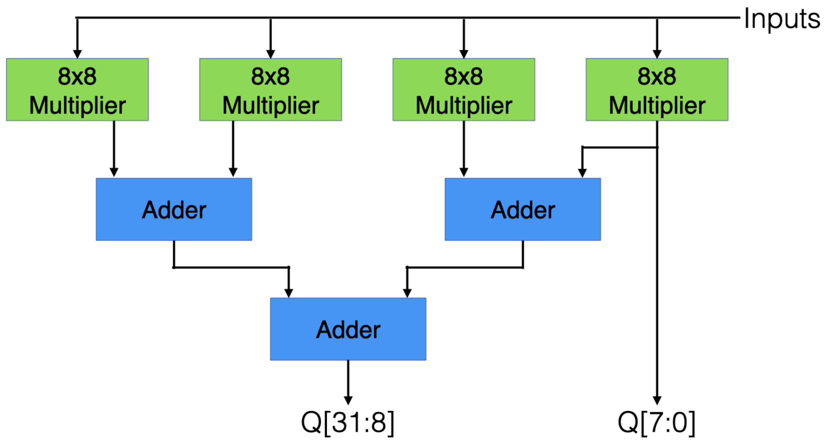


Figure 3.6: Structure of a Basic 16-bit Multiply and Accumulate Unit [24]

Approximate Divider

Afterwards, we study the approximate divider proposed in [20]. Unlike the array multiplier and MAC, the architecture of the approximate divider proposed in [20] does not offer a wide symmetry as sub-blocks are not heavily redundant. As shown in the diagram of the approximate divider shown in Figure 3.7 [20], the chosen divider offers an additional level of complexity as it consists of two leading one detectors, multiplexers and a barrel shifter which are interconnected to exact functional units, i.e., subtracter and divider. Moreover, the divider, as proposed, offers a small room to form variant configurations since the only variation is the number of bits selected, i.e., k and $k/2$ selected bits. Thus, the variations can be classified as position independent since sub-blocks can be positioned in a single form when the bit-width changes and the proposed DSE can be applied. The only remaining question is whether the mathematical modeling can be applied. Towards addressing this concern, the model is tested in the configuration of dividing 16-bit by 8-bit with $k = 8$. The output quality of the chosen configuration is successfully assessed using the proposed mathematical modeling.

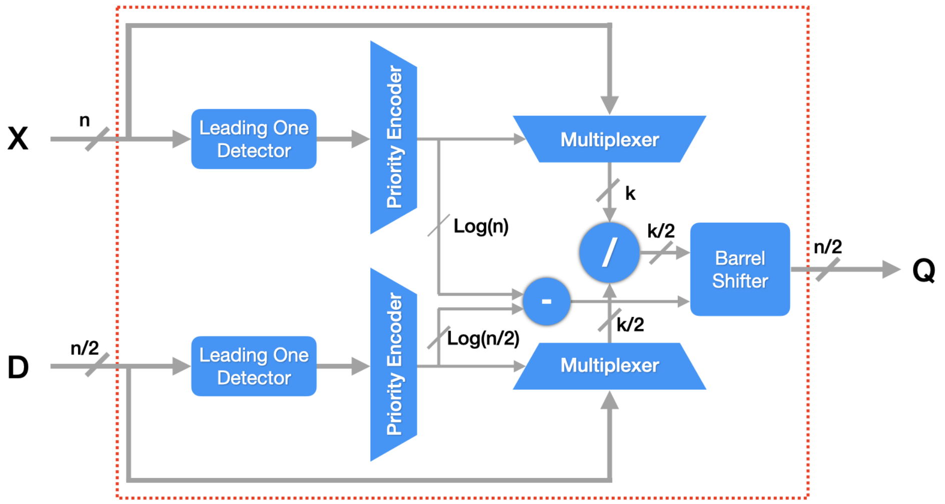


Figure 3.7: Approximate Divider [20]

Large Approximate Multipliers

Finally, we study the extensibility of the proposed DSE to larger designs, e.g., 32-bit array multiplier. The runtime to generate Boolean equations for a given approximate 32-bit array multiplier configuration is depicted in Figure 3.8, with a total of 1453 seconds ($\sim$ 24 minutes). Furthermore, assessing the output quality, i.e., g(X, Y), took 11 hours the same configuration. All the proposed mathematical modeling took almost 11.5 hours. These measurements are conducted on Mathematica [3] running on a computer with Intel(R) Core(TM) i5-4278U CPU 2.60GHz and 8GB of RAM. The operating system of the machine is macOS 10.15. However, if the same quality assessment had to be conducted using excessive simulation on an HPC, the projected runtime would be in the range of 3.4 to 4.3×10^{10} hours (thousands of decades).

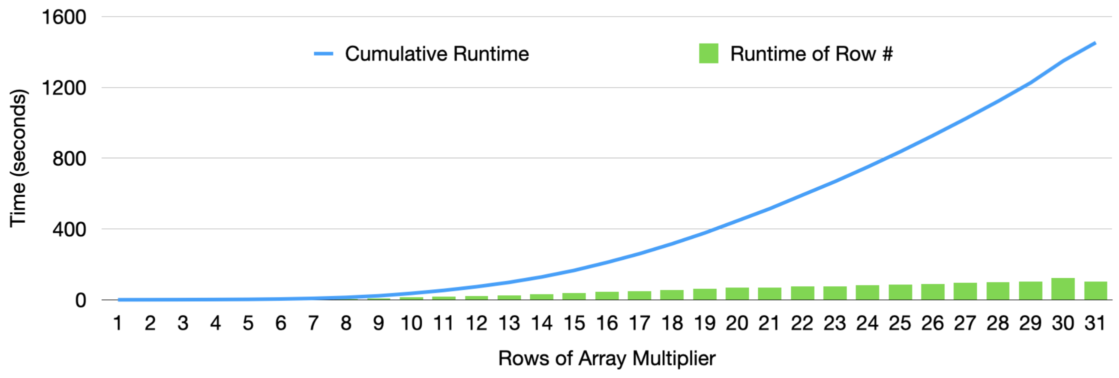


Figure 3.8: Runtime to Generate Output Equations for a 32-bit Array Multiplier

During a similar experiment with a 64-bit array multiplier and due to the exponential growth of the total runtime, the process was killed after generating equations for the first 22 rows (less than half). The runtime for these rows along with the total runtime are illustrated in Figure 3.9. The cumulative runtime was 25560 seconds ($\sim$ 7 hours).

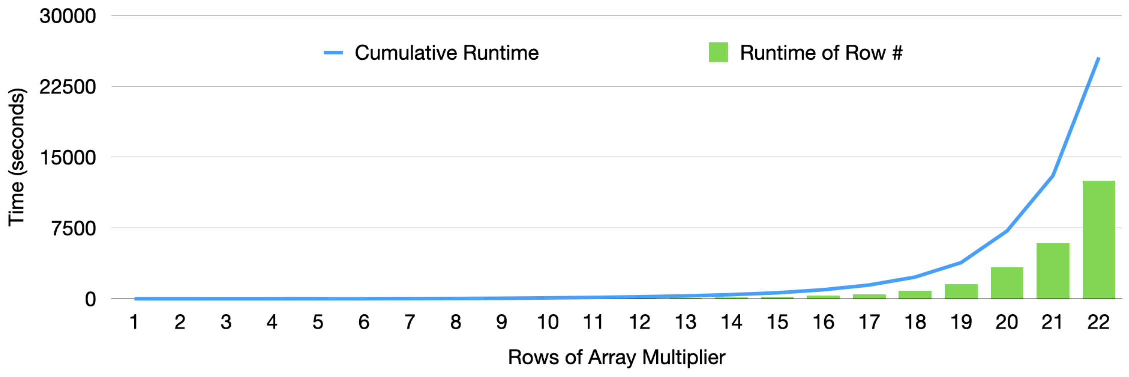


Figure 3.9: Runtime to Generate Some of the Output Equations for a 64-bit Array Multiplier

3.5 Summary

Design space exploration refers to the process of examining design alternatives that best meet the design requirements. Since multiple AC architectures have been proposed which have been studied for a small number of configurations, a DSE is indisputable. Nonetheless, previously proposed methods of DSE for AC designs can generate AC circuits in a very small subset from the design space. Furthermore, previously proposed DSEs relies on excessive simulation which can be either: 1) insignificant if studied in limited number of random samples; or 2) time consuming if all possible combinations are studied.

The proposed DSE drastically reduces time by cutting corners in the synthesis, and offering a new and faster technique to asses quality. The main contributions of the proposed methodology shown in Figure 3.2 include:
1- A two-phases design space reduction, which incorporates a genetic algorithm in order to generate a high-quality solution based on user-given design constraints, i.e., area, power, and delay.
2- A decimal modeling for logic circuits. This method will eliminate the need for excessive simulation, which can consume an indefinite amount of time.

Moreover, the proposed DSE methodology proved to be efficient for studying various types of AC designs and different input-widths. Nonetheless, applying the proposed DSE on large circuit designs, e.g., 64-bit multiplier, cannot be performed on a regular PC, and an HPC would be an appropriate machine for such task. Finally, if a wider DSE is conducted, a golden goal, or near golden goal AC circuit could be found. This golden goal would achieve improved quality assurance. However, since AC circuits offer a great cut in area, delay and power, they could be handful in improving QoS in circuits. Towards this goal, in the next chapter, a new approach is proposed that will improve the reliability of circuits by using AC modules, and thus improving QoS.

Chapter 4

Approximate Quadruple Module Redundancy

Quality assurance can be considered as the golden goal for any circuit implementation. The pillars of quality assurance are output quality and reliability of the circuit. In the previous chapters, multiple techniques that aim to drastically reduce resource usage yet offering high output quality were presented. To complement this work, this chapter will focus on improving the reliability of circuits with the usage of AC.

4.1 Introduction

Redundancy has been a common method to improve reliability. In this chapter, we present a design consisting of redundant AC circuits for improved design reliability, yet offering high quality. The design we propose aims to offer redundant modules in case the exact unit fails. However, since the usage of exact modules for redundancy would result in high overhead, the design we propose uses approximate modules for the sake of reducing the overhead. This approach will improve reliability for a trade in quality if the exact unit fails. Figure 4.1 depicts the methodology used in building the proposed approximate modular redundancy and its relationship to the general methodology of this thesis and the subjects discussed in previous chapters. The process consists of choosing one AC circuit configuration from a library of AC designs, which provides the best quantitative features in terms of quality and resource usage. This selection is based on a quantitative superiority in the area of quality and resource usage. Moreover, such library can be generated by using the traditional method of building a library of AC designs or with the use of design space exploration (DSE) proposed in the previous chapter. Previous usage of AC for improved design reliability is based on domain specific applications e.g., loop perforation [45] or highly optimistic speculations, e.g., different approximate units will generate exact results [6] [14]. Such assumption may result in a deterioration of the output quality in the presence or absence of faults. Thus, with the aim of this thesis to improve quality assurance of AC designs, we propose a technique that offers a two-step approximation-aware voter based on the output magnitude, which will be explained in details in the next sections.

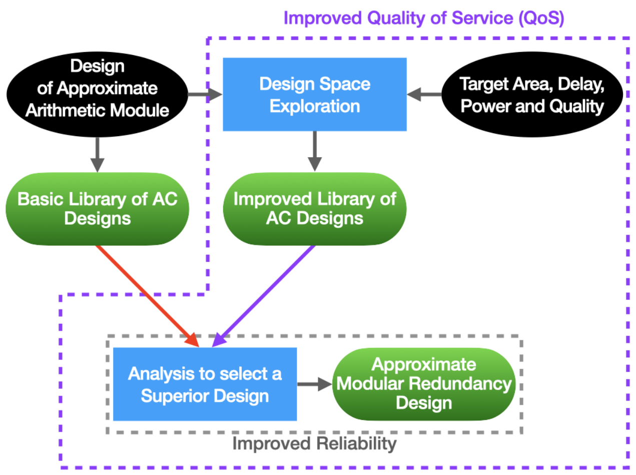

Improved Quality of Service (QoS)
Design of Approximate Arithmetic Module
Design Space Exploration
Target Area, Delay, Power and Quality
Basic Library of AC Designs
Improved Library of AC Designs
Analysis to select a Superior Design
Approximate Modular Redundancy Design
Improved Reliability

Figure 4.1: Integrating Modular Redundancy for Improved AC Quality of Service

Previously presented approaches of ATMR use different versions of AC circuits, with the assumption that (at most) one of the approximated modules will provide a faulty bit in the output vector coming from each of the redundant modules [6]. Based on this assumption, a majority voter is then used, which will select the two bits matching. However, this assumption can be considered as unrealistic, since using approximate modules can lead to multiple bits being misrepresented (flipped bits) at the same time, and thus the majority voter can be mislead. Alternatively, to make a benefit of approximate circuit, the concept of error-threshold in ATMR to perform logic masking of soft-errors is proposed in [7]. Moreover, a low-area overhead quadruple approximate modular redundancy is proposed by [14]. The proposed solution consists of four non-identical approximate modules, with different bits under the condition of at least one bit of the output vector to remain unmasked.

All proposed approaches have a common base that relies on the usage of different approximate circuits in parallel in order to offer functional units redundancy. Moreover, the design where one bit can fail to deliver the correct value, while the remaining modules are error-free. Such assumption is too optimistic since in an AC circuit, every bit has the chance to provide a faulty value. To overcome such an infeasible assumption, we propose a highly-reliable innovative scheme by using three identical approximate modules in addition to the exact one.

4.2 Proposed System Architecture

When using a new technology, e.g., approximate module, a detachment from previous notions must take place. The proposed approximate QMR (AQMR) is detached from previous concepts by delivering:

1- Approximation-aware design that will tolerate a small difference in result of AC and exact computation.

2- Error-tolerant result in the case that the exact computation error-induced, yet the error is smaller than the tolerable error, i.e., ED.

Approximate modular redundancy is a new perception and little work has been accomplished in this area. The proposed AQMR is innovative in this area, by adapting to the characteristics of AC with the use of a two-steps voter. Concurrently, the proposed architecture delivers highly reliable designs with small overhead. Furthermore, the introduced model is designed with a commonly used hypothesis that a voting circuitry does not fail, and thus these modules are not redundant.

As shown in Figure 4.2, the architecture we propose is composed of two main blocks, namely functional modules and a two-step voter. The functional modules consist of an approximate module redundant three times (same module) and one exact module. Even though the approximate units can be heterogeneous (different structures of modules), we opted for the usage of the same module, since the selected approximate design is superior compared to other explored designs in the library. Moreover, the functional modules will take the run-time inputs and will generate four outputs. The first output is an exact computation, i.e., $Result_E$, while the three remaining outputs are approximated values, i.e., $Result_{A1}$, $Result_{A2}$ and $Result_{A3}$.

The two-step voter block shown in Figure 4.2 consists of first step voter, i.e., *Voter 1*, being the commonly used majority voter, and a second step voter, i.e., *Voter 2*, as a new magnitude-based voter, which will forward the appropriate output based on the difference between the inputs, i.e., error distance (ED).

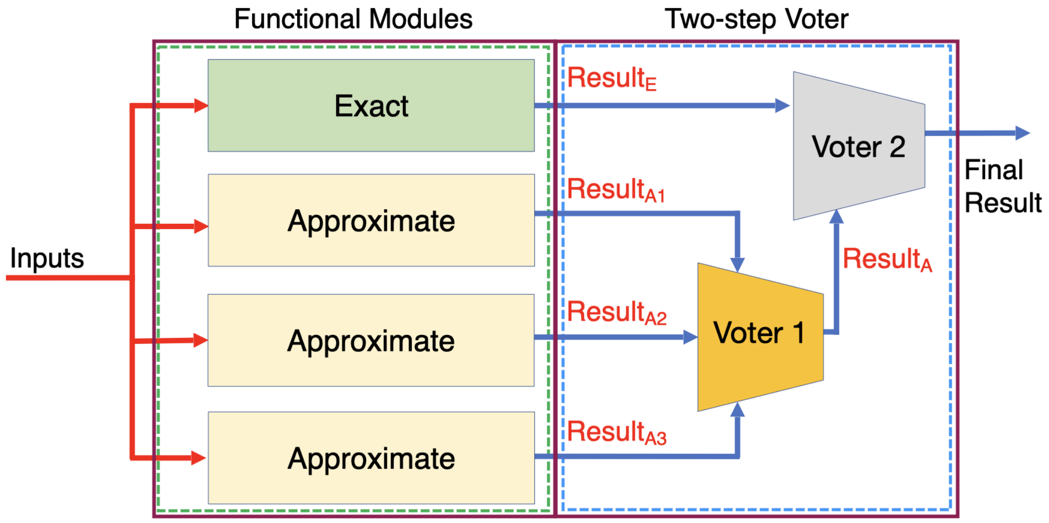


Figure 4.2: Architecture of Approximate Quadruple Modular Redundancy

The architecture we propose takes advantage of the two-steps voter by feeding the *Voter 1*, with three approximate values. This system will mask the faulty bit, if any of the three approximate circuits is faulty. The *Voter 1* will provide an error-free approximate result, i.e., $Result_A$. The generated value is then used to feed *Voter 2* along with an exact result, i.e., $Result_E$, coming from the exact functional module. Since the *Voter 2* is magnitude-based, a marginal difference between ResultA and ResultE will be tolerated. If the difference exceeds the expected value, the ResultA will be used as a final result. Furthermore, since *Voter 2* is magnitude based, the error thresholds must be clustered, and thus the tolerable error will be input based. For instance, if the 8-bit multiplier is used, then all the possible combinations are $2^8 \times 2^8 = 65536$. Hence, the number of adjacent entries used to form a cluster (C) is equal to $\frac{65536}{C}$. If the design has 64 clusters, then each cluster has 1,024 adjacent entries. When inputs are applied, the tolerable error will be based on error-threshold for their correspondent cluster.

4.3 Experimental Results

In this section, we conduct a validation and evaluation of the proposed methodology for AQMR shown in Figure 4.1. In the conducted study, 8-bit array multipliers are used. The redundant approximate multiplier is formed by replacing the FAs in the nine right columns with the approximate mirror adder 5 (AMA5) [17]. In the following subsections, area and power are first assessed, followed by accuracy, and finally the reliability of the system is studied.

4.3.1 Area and Power Assessment

The resources utilization of the proposed AQMR is compared with the traditional TMR, i.e., exact TMR, which consists of three parallel exact functional units that feed a majority voter. For the analysis, the models are synthesized using Xilinx Vivado [57], with the target device being XC7VX485T FPGA, from the Xilinx Virtex-7 FPGA family [56]. The power consumption for the exact TMR and the proposed AQMR turned out to be 14.347W and 7.24W, respectively. Furthermore, a similar drop in area usage with the exact TMR using 79 lookup tables (LUT), while the AQMR uses 39 LUT. Thus, the total savings are 62% and 49.5% in area and power, respectively.

4.3.2 Accuracy Assessment

The accuracy of the system is conducted on the basis of single-event upset (SEU), and thus, one of the functional units can provide a faulty answer. The SEU can be stimulated by a flip of one bit from the input or output of the four functional units. Furthermore, since the design we propose uses three identical approximate modules, a faulty result from one of the approximate functional units will be hidden, as the outputs of these units are fed to a majority voter, i.e., *Voter 1*, as shown in Figure 4.2. Furthermore, if two approximate units fail at the same time, and in the same manner, i.e., the same output bit is flipped, this will result in the error bypassing the *Voter 1.* However, this error might be accepted by the the second voter, i.e., *Voter 2*, as this flip can result in a closer value to the exact value. Hence, the *Voter 2* will select the exact value, since the difference is still in the tolerable threshold and exact value, i.e., $Result_E$ will be forwarded to the final output, i.e., *Final Result.* Finally, a fault detection in *Voter 2*, means that the $Result_A$ is forwarded to the output.

The exact functional unit can also be affected by a SEU which will modify $Result_E$. However, similar to the failure of two approximate units simultaneously, if the error is still in the acceptable range, then a small error, i.e., small ED, will be present in the final output. Nonetheless, if the erroneous exact value is selected, then the error will be small, and thus the accuracy can still be considered high.

The proposed QAMR was set under test using simulation, and on average, the design was able to detect 99.88% of the cases where the exact unit was faulty, and thus the final output was based on the approximately computed value. This high percentage was achieved when a large the number of clusters is used, where a cluster has minimal number of entries. A detailed fault detection for different number of (C) is shown in Table 8.

Table 8: The Percentage (%) of Fault Detection if the Exact Module is Faulty

	Number of Clusters							
	2	**4**	**8**	**16**	**32**	**64**	**128**	**256**
Run#1	56.28	56.82	57.43	58.43	58.51	59.87	61.10	63.64
Run#2	54.60	55.26	55.97	57.17	57.76	58.58	60.57	62.55
Run#3	52.20	52.44	53.39	54.63	55.08	55.92	57.83	60.35
Average	54.36	54.84	55.60	56.74	57.12	58.12	59.83	62.18

	Number of Clusters							
	512	**1024**	**2048**	**4096**	**8192**	**16384**	**32768**	**65536**
Run#1	65.82	68.91	72.20	77.11	80.55	87.82	93.78	99.81
Run#2	65.31	68.13	72.50	76.97	80.18	88.75	94.62	99.90
Run#3	63.03	65.77	69.59	75.66	78.32	87.50	95.21	99.93
Average	64.72	67.60	71.43	76.58	79.69	88.03%	94.54	99.88

However, an undetected fault, i.e., selecting $Result_A$ as *Final Result*, does not imply a failure of the system, nor a complete deterioration in the quality. Figure 4.3 represents the average PSNR obtained for an image processing application based on a different number of clusters. From Figure 4.3 and Table 8, it can be noticed that more clusters would imply more detection of faults in the exact module, and thus a higher PSNR (higher is better). Nonetheless, the minimum achieved PSNR is $82.07dB$ and maximum of $86.61dB$ while the average PSNR of all configurations is $84.16dB$. Thus, all configurations can be deemed usable since an acceptable PSNR is achieved.

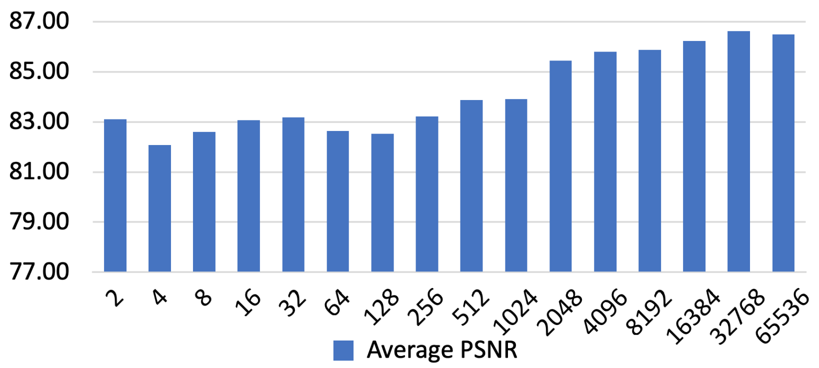


Figure 4.3: Average PSNR in dB for Different Numbers of Clusters

Another case study is conducted, with the condition that $Result_A$ is faulty, while the $Result_E$ is flawless. In this case, theoretically, $Result_E$ must be carried to the final output. However, this would depend on the forwarded output, i.e., $Result_E$ or $Result_A$, which is related to the number of used clusters. Table 9 shows the percentage of fault detection, i.e., selecting $Result_A$ as *Final Result*, for the case of faulty $Result_A$ when the number of clusters varies. From Table 9 it can be noticed that with a higher number of clusters, *Voter 2* tends to select an approximate value, i.e., $Result_A$ even if this result is faulty. Thus, from the results in Tables 8 and 9, it is recommended to keep the number of clusters to less than 512, as a balance can be achieved in fault detection. Furthermore, a lower number of clusters would result in less complexity in *Voter 2*. Hence, the advantages of using a smaller number of clusters, i.e., $(C) \leqslant 512$, can be noticed at different levels.

Table 9: The Percentage (%) of Fault Detection if the Approximate Module is Faulty

	Number of Clusters								
	2	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**
Run#1	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.12	0.43
Run#2	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.15	0.83
Run#3	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.13	0.58
Average	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.14	0.61

	Number of Clusters						
	1024	**2048**	**4096**	**8192**	**16384**	**32768**	**65536**
Run#1	6.19	14.38	31.47	49.67	69.20	80.19	96.04
Run#2	5.82	18.98	35.23	48.58	74.63	85.18	96.36
Run#3	6.14	17.25	39.99	49.56	74.29	88.21	96.31
Average	6.05	16.87	35.56	49.27	72.71	84.53	96.23

4.3.3 Reliability Assessment

In the study of accuracy, the failure of two approximate functional units does not imply that the quality will be deteriorated. However, for the reliability analysis, the study must be based on actual functionality, i.e., performing as per the standard of design, and not based on delivery of a lesser quality nor a probable failure. Thus, the analysis conducted in this subsection will consider that a failure of two approximate functional units will result in a failure of the system. The failure of two approximate modules will result in a "faulty" result coming from *Voter 1*. Hence, the failure of *Voter 1* in providing a "fault-free" result follows the rule of binomial distribution. The probability of faulty (Pf) $Result_A$ is provided by Equation (3), with n being the number of approximate results, in this case three, and p the failure probability of the approximate functional units. Moreover, the *Voter 2* will fail to provide a correct value, if the exact unit fails, and *Voter 1* fails to provide a correct value, i.e., failure of two or more approximate units. In this case, the probability of faulty (Pf) *Final Result*, i.e., failure of the system, follows the theory of two independent events. The failure of the system is provided by Equation (4), where $P(A)$ and $P(E)$ are the probabilities of faulty $Result_A$ and $Result_E$.

$$Pf = \binom{1}{n}(1-p)(p)^{n-1} + \binom{0}{n}(p)^{n} \tag{3}$$

$$Pf = P\left(A \cap E\right) = P(A) \times P(E) \tag{4}$$

For the sake of simplification, all functional units, i.e., exact and approximate modules, are considered to have the same failure rate. Figure 4.4 illustrates the system's relative failure in terms of the module's failure for the traditional TMR, i.e., exact TMR, and approximate module redundancy, i.e., $\mathrm{AMR_n}$ is a design composed of n approximate modules and one exact module.

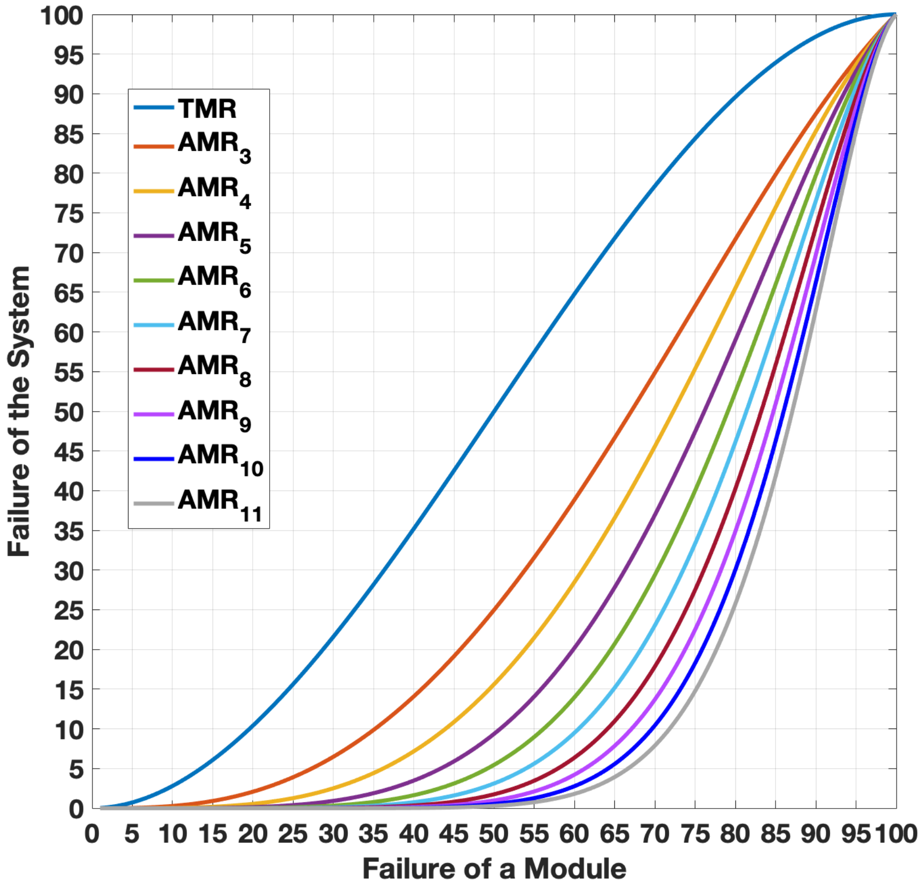


Figure 4.4: Relative Probability of System Failure vs Component Failure for Different Module Redundancy Circuits

The proposed AQMR has an average of 15% less chances of failure compared to the traditional exact TMR. Furthermore, it can be noticed that increasing the number of approximate modules, will decrease the overall system's failure rate. Nonetheless, adding more approximate modules will increase the area and power usage in addition to a more complex design for *Voter 1*. Hence, the number of redundant approximate modules will be limited by the budget of area and power available in a given design. Moreover, the 8-bit approximate and exact multipliers used are synthesized and found to have $P_a = \frac{3}{11} \times P_e$ and $A_a = \frac{3}{8} \times A_e$, where P_a, P_e, A_a and A_e are the power for approximate, power for exact, area for approximate and area for exact multipliers, respectively. Thus, with savings in area and power, along with an improved system's reliability, benefits of AC as modular redundancy are clear.

4.4 Summary

In this chapter, we proposed a highly reliable approximate modular redundancy for AC designs. The work complements the efforts of the thesis in improving quality assurance, i.e., QoS, with the use of AC. Nonetheless, previous implementations of AC in modular redundancy have been based on very optimistic assumptions. However, the work we presented in this chapter phases out this ideal assumption with the proposition of a new approach to implement AC for modular redundancy. This new approach integrates AC for the sake of reliability improvement while the natural behavior of AC designs is contemplated. Furthermore, the presented architecture offers great savings in terms of area and power usages, compared to traditionally used TMR, while maintaining an acceptable accuracy. The AQMR we proposed in this chapter is an additional step in the direction of improving quality assurance of logic designs with the use of AC designs. This model is complemented by the use of the DSE proposed in Chapter 3. When the two models are implemented together, i.e., design resulting from a DSE implemented in an AQMR, the result is an improved QoS, i.e., improved reliability and output quality, with an optimization of resource usage.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Approximate computing (AC) is an emerging computing paradigm that has gained traction in the past few years. AC reduces output quality for the benefit of savings in resources usage, e.g., power. This computation technique can be applied to error-tolerant applications such as image processing, where a small loss in quality is imperceptible due to the imperfect human sense. Another example application field is search engines where there is no unique or golden result. With the benefits offered, and an existing domain application, AC is a promising area of research for a future integration within computer architectures and algorithms of brain-inspired computing. However, for this computing paradigm to be deemed ready for adoption, some essential questions have be answered such as [18]: 1) how to measure output quality and assure the maintenance of output quality, 2) what are the reasonable "cutting-corners" to be applied while maintaining quality.

Towards answering these questions, in this thesis, the expendability of a previously presented model [32] that aims for improved quality assurance is studied. The model consists of a design selector, that predicts the best-fit design, with the help of machine learning (ML), which is expected to deliver the aimed quality. In Chapter 2, the model was successfully extended from 8-bit to 16-bit functional units by using one of the most powerful computation machine available today. Moreover, an additional data pre-processing is proposed that allowed for the ML classifier to analyse training data and generate ML-based predictor. This led to an improved output quality compared to the previously presented model. However, with today's computation power, this model cannot be extended to even larger designs, e.g., 64-bit floating-point computation, that can be considered as industry-standard nowadays.

Furthermore, the quality of AC designs, in general, can be improved by performing a design space exploration (DSE). However, if all the possible variations for AC designs are considered, the design space consists of multi-trillion possibilities. Moreover, studying all possibilities from this space is almost impossible since assessing all designs will require an infinite amount of computational power. Nonetheless, in order to achieve a DSE for AC, in Chapter 3, an efficient DSE is proposed, which eliminates worthless designs, based on area, delay and power. To complement this work, a mathematical modeling for logic circuits is presented, which will allow to study output quality of AC designs from a mathematical point of view, instead of using excessive simulation.

Last but not least, to supplement the proposed DSE, which aims in finding a design offering a good output quality, a highly reliable functional design is presented in Chapter 4. This design will take advantage of the characteristics of AC circuits, with their use in high quality modular redundancy, while keeping a minimal overhead.

In summary, the work presented in this thesis, aims in the direction of improving quality of service (QoS), while reducing area, delay and power consumption. As AC designs already offer reductions in these fields, it would be trivial to study their implementation in quality assurance designs.

5.2 Future Work

The study of AC has not matured yet and a lot of work is still needed. The work presented in this thesis, lays a foundation for future work in this field. Thus the following list covers some future tasks that can improve AC designs:

- The proposed DSE offers a wide range of support for logic circuits, however, the proposed work can be improved by offering a better modeling for complex gates, e.g., arithmetic modeling for XOR-gate.

- The DSE proposed in this thesis randomly generates approximate structures. However, a better approach is to be aware of the implications on the output quality that would result from such modifications. A good implementation for approximate implication-aware model, can be an ML-based model that can predict such consequences.

- The ML-based quality assurance proposed Design in [32] has been extended from 8-bit to 16-bit multipliers with an improved output quality. However, extensions to other models, e.g., 64-bit multipliers and floating-point units, must be studied. Moreover, additional improvements to the quality must be considered as well.

- Quality measurement of AC designs is still debatable because of the lack of a unified error metrics. For instance some of the AC designs have been assessed based on bit-error rate (BER), while others on error magnitude with error distance (ED), peak signal-to-noise ratio (PSNR), etc. The variety in error metrics is resulting in a subjective quality assessment for various AC designs. Thus a unified error metric, i.e., standard error metric, would allow to classify the quality of AC designs in an objective manner. A good approach can be a weighted error metric that takes various quality metrics into consideration.

Bibliography

[1] GM Update on Semiconductor Production Impact. https://media.gm.com/media/us/en/gm/home.detail.html/content/Pages/news/us/en/2021/feb/0203-semiconductor-statement.html. Last accessed May 7, 2021.

[2] RAW-Samples. http://rawsamples.ch/. Last accessed May 7, 2021.

[3] Wolfram mathematica: The world's definitive system for modern technical computing. https://www.wolfram.com/mathematica/. Last accessed May 7, 2021.

[4] Haider A. F. Almurib, T. Nandha Kumar, and Fabrizio Lombardi. Inexact designs for approximate low power addition by cell replacement. In *IEEE Conference on Design, Automation and Test in Europe*, page 660–665, 2016.

[5] A. Aponte-Moreno, A. Moncada, F. Restrepo-Calle, and C. Pedraza. A review of approximate computing techniques towards fault mitigation in HW/SW systems. In *IEEE Latin-American Test Symposium*, pages 1–6, 2018.

[6] T. Arifeen, A. S. Hassan, and J. Lee. A fault tolerant voter for approximate triple modular redundancy. *Electronics*, 8(3), 2019.

[7] T. Arifeen, A. S. Hassan, H. Moradian, and J. A. Lee. Probing approximate tmr in error resilient applications for better design tradeoffs. In *Euromicro Conference on Digital System Design*, pages 637–640, 2016.

[8] T. Baltrušaitis, C. Ahuja, and L. Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2019.

[9] BBC. BBC Rewind - Sound Effects. http://bbcsfx.acropolis.org.uk/. Last accessed May 7, 2021.

[10] J. Bornholt, T. Mytkowicz, and K. McKinley. Uncertain<T>: A first-order type for uncertain data. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, page 51–66, 2014.

[11] Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[12] L. Chen, J. Han, W. Liu, and F. Lombardi. Design of approximate unsigned integer non-restoring divider for inexact computing. In *ACM Great Lakes Symposium on VLSI*, page 51–56, 2015.

[13] CMC. TSMC 18µm CMOS Process Technology. https://www.cmc.ca/tsmc-0-18-µm-cmos-process-technology/. Last accessed May 7, 2021.

[14] B. Deveautour, M. Traiola, A. Virazel, and P. Girard. Qamr: an approximation-based fully reliable tmr alternative for area overhead reduction. In *IEEE European Test Symposium*, pages 1–6, 2020.

[15] S. García, J. Luengo, and F. Herrera. *Data preprocessing in data mining.* Springer, 2015.

[16] I.A.C. Gomes, M.G.A. Martins, A.I. Reis, and F.L. Kastensmidt. Exploring the use of approximate tmr to mask transient faults in logic with low area overhead. *Microelectronics Reliability*, 55(9):2072–2076, 2015.

[17] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. Low-power digital signal processing using approximate adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(1):124–137, 2013.

[18] J. Han. Introduction to approximate computing. In *IEEE VLSI Test Symposium*, pages 1–1, 2016.

[19] S. Hashemi, R. I. Bahar, and S. Reda. Drum: A dynamic range unbiased multiplier for approximate applications. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 418–425, 2015.

[20] S. Hashemi, R. I. Bahar, and S. Reda. A low-power dynamic divider for approximate applications. In *ACM/IEEE Design Automation Conference*, pages 1–6, 2016.

[21] T. Hastie, R. Tibshirani, and J.H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2017.

[22] D. Hernandez-Araya, J. Castro-Godínez, M. Shafique, and J. Henkel. Auger: A tool for generating approximate arithmetic circuits. In *IEEE Latin American Symposium on Circuits Systems*, pages 1–4, 2020.

[23] M. Imani, R. Garcia, A. Huang, and T. Rosing. Cade: Configurable approximate divider for energy efficiency. In *Design, Automation Test in Europe*, pages 586–589, 2019.

[24] D. Jaina, K. Sethi, and R. Panda. Vedic mathematics based multiply accumulate unit. In *International Conference on Computational Intelligence and Communication Networks*, pages 754–757, 2011.

[25] M. A. Laurenzano, P. Hill, M. Samadi, S. Mahlke, J. Mars, and L. Tang. Input responsiveness: Using canary inputs to dynamically steer approximation. *SIGPLAN Notes*, 51(6):161–176, 2016.

[26] S.C. Lee and Y. M. Ajabnoor. Digital calculus: Ss. In *International Symposium on Multiple-Valued Logic*, page 128–141, 1978.

[27] M. Masadeh. *Adaptive Approximate Computing for Enhanced Quality Assurance*. PhD thesis, Concordia University, August 2020.

[28] M. Masadeh, O. Hasan, and S. Tahar. Comparative study of approximate multipliers. In *ACM Great Lakes Symposium on VLSI*, page 415–418, 2018.

[29] M. Masadeh, O. Hasan, and S. Tahar. Input-conscious approximate multiply-accumulate unit for energy-efficiency. *IEEE Access*, 7:147129–147142, 2019.

[30] M. Masadeh, O. Hasan, and S. Tahar. Using machine learning for quality configurable approximate computing. In *Design, Automation Test in Europe*, pages 1575–1578, 2019.

[31] M. Masadeh, O. Hasan, and S. Tahar. Machine learning-based self-compensating approximate computing. In *Computing Research Repository (CoRR), Electrical Engineering and Systems Science, Signal Processing, arXiv.1908.01343*, 2020.

[32] M. Masadeh, O. Hasan, and S. Tahar. Machine-learning-based self-tunable design of approximate computing. *IEEE Transactions on Very Large Scale Integration Systems*, 29(4):800–813, 2021.

[33] Mathworks. Classification learner toolbox. https://www.mathworks.com/help/stats/classificationlearner-app.html. Last accessed May 7, 2021.

[34] Mathworks. What is MATLAB. https://www.mathworks.com/discovery/what-is-matlab.html. Last accessed May 7, 2021.

[35] S. Mazahir, O. Hasan, R. Hafiz, and M. Shafique. Probabilistic error analysis of approximate recursive multipliers. *IEEE Transactions on Computers*, 66(11):1982–1990, 2017.

[36] J. Melchert, S. Behroozi, J. Li, and Y. Kim. Saadi-ec: A quality-configurable approximate divider for energy efficiency. *IEEE Transactions on Very Large Scale Integration Systems*, 27(11):2680–2692, 2019.

[37] A. Mishra, R. Barik, and S. Paul. iact: A software-hardware framework for understanding the scope of approximate computing. In *Workshop on Approximate Computing Across the System Stack*, pages 1–6, 2014.

[38] S. Mittal. A survey of techniques for approximate computing. *ACM Computing Survey*, 48(4), 2016.

[39] A. Momeni, J. Han, P. Montuschi, and F. Lombardi. Design and analysis of approximate compressors for multiplication. *IEEE Transactions on Computers*, 64(4):984–994, 2015.

[40] V. Mrazek, M. A. Hanif, Z. Vasicek, L. Sekanina, and M. Shafique. autoax: An automatic design space exploration and circuit building methodology utilizing libraries of approximate components. In *ACM/IEEE Design Automation Conference*, pages 1–6, 2019.

[41] R. Ragavan, B. Barrois, C. Killian, and O. Sentieys. Pushing the limits of voltage over-scaling for error-resilient applications. In *Design, Automation Test in Europe*, pages 476–481, 2017.

[42] N. Ravi. Big data needs approximate computing: Technical perspective. *Communications of ACM*, 58(1):104, 2014.

[43] K. M. Reddy, Y. B. Nithin Kumar, D. Sharma, and M. H. Vasantha. Low power, high speed error tolerant multiplier using approximate adders. In *International Symposium on VLSI Design and Test*, pages 1–6, 2015.

[44] G.S. Rodrigues, J. Fonseca, F. Benevenuti, F . Kastensmidt, and A. Bosio. Exploiting approximate computing for low-cost fault tolerant architectures. In *Symposium on Integrated Circuits and Systems Design*, pages 1–6, 2019.

[45] G.S. Rodrigues, J.S. Fonseca, F.L. Kastensmidt, V. Pouget, A. Bosio, and S. Hamdioui. Approximate tmr based on successive approximation and loop perforation in microprocessors. *Microelectronics Reliability*, 100-101:113385, 2019.

[46] B. Schroeder, E. Pinheiro, and W.D. Weber. DRAM errors in the wild: A large-scale field study. *Communications of ACM*, 54(2):100–107, 2011.

[47] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *International Conference on Knowledge Discovery and Data Mining*, page 614–622, 2008.

[48] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proceedings International Conference on Dependable Systems and Networks*, pages 389–398, 2002.

[49] B. D. Sierawski, B. L. Bhuva, and L. W. Massengill. Reducing soft error rate in logic circuits through approximate logic functions. *IEEE Transactions on Nuclear Science*, 53(6):3417–3421, 2006.

[50] K. Sohee. Samsung warns of severe chip crunch while delaying key phone. https://www.bloomberg.com/news/articles/2021-03-17/samsung-warns-of-serious-imbalance-in-the-chips-industry. Last accessed May 7, 2021.

[51] Synopsys. Design compiler graphical. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler-graphical.html. Last accessed May 7, 2021.

[52] AITS Concordia University. High-performance computing (HPC) facility: Speed. https://www.concordia.ca/ginacody/aits/speed.html. Last accessed May 7, 2021.

[53] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan. Approximate computing and the quest for computing efficiency. In *ACM/IEEE Design Automation Conference*, pages 1–6, 2015.

[54] X. Vera, J. Abella, J. Carretero, and A. González. Selective replication: A lightweight technique for soft errors. *ACM Transactions on Computer Systems*, 27(4):1–30, 2010.

[55] E. Weisstein. "ball picking." from mathworld–a wolfram web resource. https://mathworld.wolfram.com/BallPicking.html. Last accessed May 7, 2021.

[56] Xilinx. Virtex-7 fpga family. https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html. Last accessed May 7, 2021.

[57] Xilinx. Vivado design suite. https://www.xilinx.com/products/design-tools/vivado.html. Last accessed May 7, 2021.

[58] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi. Approximate xor/xnor-based adders for inexact computing. In *IEEE International Conference on Nanotechnology*, pages 690–693, 2013.

[59] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram. Roba multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing. *IEEE Transactions on Very Large Scale Integration Systems*, 25(2):393–401, 2017.

[60] G. Zervakis, K. Tsoumanis, S. Xydis, N. Axelos, and K. Pekmestzi. Approximate multiplier architectures through partial product perforation: Power-area tradeoffs analysis. *ACM Great Lakes Symposium on VLSI*, pages 3105–3117, 2015.

[61] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi. Design-efficient approximate multiplication circuits through partial product perforation. *IEEE Transactions on Very Large Scale Integration Systems*, 24(10):3105–3117, 2016.

Biography

Education

- **Concordia University**: Montreal, Quebec, Canada.
  MASc., Electrical & Computer Engineering (September 2019 - May 2021)
- **Notre Dame University**: Zouk-Mosbeh, Lebanon.
  B.E., Electrical Engineering (September 2013 - July 2018)

Awards

- Concordia University, International Award of Excellence (MASc. Program).
- Notre Dame University, Six Academic Honors: Dean's List.

Work History

- **Research Assistant**, Hardware Verification Group, Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada (2019-2021).
- **Teaching Assistant**, Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada (Winter 2021).
- **Webmaster**, Hardware Verification Group, Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada (2019-2021).

Publications

Conference Papers

- **Bio-Cf1** M. Masadeh, A. Aoun, O. Hasan and S. Tahar: Highly-Reliable Approximate Quadruple Modular Redundancy with Approximation-Aware Voting; Proc. IEEE International Conference on Microelectronics (ICM'20), Irbid, Jordan, December 2020, pp. 1-4.

- **Bio-Cf2** M. Masadeh, A. Aoun, O. Hasan and S. Tahar: Decision Tree-based Adaptive Approximate Accelerators for Enhanced Quality, Proc. IEEE International Systems Conference (SysCon'20), Montreal, Quebec, Canada, 2020, pp. 1-5.

- **Bio-Cf3** A. Aoun, A. Iliovits, A. Kassem, P. Sakr and N. Metni: Arthro-Glove a Hybrid Bionic Glove for patients diagnosed with Arthritis, ALS and/or Dysmorphia, Proc. IEEE Cairo International Biomedical Engineering Conference (CIBEC'18), Cairo, Egypt, 2018, pp. 106-109.

- **Bio-Cf4** E. Maalouf, N. Marina, J. B. Abdo, A. Aoun, M. Hamad and A. Kassem, Asthma Irritant Monitoring, Proc. IEEE International Conference on Microelectronics (ICM'18), Sousse, Tunisia, 2018, pp. 120-123.

- **Bio-Cf5** A. Aoun, A. Kassem and M. Hamad, Sun Stimulator for Daylight System, Proc. IEEE International Arab Conference on Information Technology (ACIT'18), Werdanye, Lebanon, 2018, pp. 1-4.

- **Bio-Cf6** A. Kassem, M. Hamad, C. El Moucary, E. Nawfal and A. Aoun, MedBed: Smart Medical Bed, Proc. IEEE International Conference on Advances in Biomedical Engineering (ICABME'17), Beirut, Lebanon, 2017,pp.1-4.