INVESTIGATION OF RECONFIGURABLE-ACCURACY APPROXIMATE ADDER DESIGNS FOR IMAGE PROCESSING APPLICATIONS

Khaled Suleiman Al-Ma'aitah

A Thesis Submitted for the Degree of

Doctor of Philosophy at Newcastle University

School of Engineering

Faculty of Science, Agriculture and Engineering

October 2019

RSST: Title Change Approval

Rebecca McLean
on behalf of
Research Student Support Team Role Account

Tue 08/10/2019 10:42 AM

To: Khaled Al-ma''aitah (PGR) <k.almaaitah@newcastle.ac.uk>
Cc: A Bystrov <a.bystrov@newcastle.ac.uk>; Alex Yakovlev <Alex.Yakovlev@newcastle.ac.uk>; contact soe pgr eleceng <soe.pgr.eleceng@newcastle.ac.uk>

RM/120365407
08.10.2019

Research Student Support Team
Student Progress Service
Student Services
Newcastle University
King's Gate
Newcastle upon Tyne
NE1 7RU United Kingdom

Dear Khaled,

I am pleased to inform you that the Dean of Postgraduate Studies, acting on behalf of the Faculty of Science, Agriculture and Engineering has approved the following concession:

Title changed to: **"Investigation of Reconfigurable-Accuracy Approximate Adder Designs for Image Processing Applications"**

Please keep this email as evidence of the concession granted.

Should you have any queries regarding this, please do not hesitate to contact the Research Student Support Team at rssteam@ac.uk.

Yours sincerely,

Rebecca McLean
Research Student Support Assistant

Research Student Support Team

Student Recruitment, Admissions and Progress
Student and Academic Services
Level 3, Kings Gate
Newcastle University
Newcastle upon Tyne
NE1 7RU
United Kingdom

Tel: 0191 208 8713

Team Email: RSSTeam@newcastle.ac.uk

RSST Drop-in Sessions - 10am to 12 pm (Mon to Fri), Level 2, King's Gate

---

Student Services welcomes feedback from students and other customers. Your views about the service we provide are very important to us and will help us make improvements where required. Please use this form to make any comments about the service you have received from us https://my.ncl.ac.uk/students/feedback.php

CC:
Supervisor(s)
School PGR Secretary

ABSTRACT

In the last decades, integrated circuits with CMOS technology show progressive scaling challenges of both increased power density and power dissipation. Meanwhile, high-performance requirements of current and future application operations show rapid demands of computing resources like power. This design conflict has pushed much effort to search for high performance and energy efficient design approach, such as approximate computing.

Approximate computing exploits the error resilience of compute-intensive applications such as image processing applications to implement approximation design techniques with different levels of abstractions and scalability. The basic principle is to relax the strict accuracy requirements in favour of a lower design complexity, thereby achieving more computational performance (i.e., speed) and energy saving. The adder arithmetic unit is considered one of the essential computational blocks in most of the applications. As such, much effort has explored new designs of an efficient approximate adder design.

This thesis presents an investigation into design enhancement, novel approximate adder designs and implementation approaches. The first approach introduces a modification to the error detection technique of a popular configurable-accuracy approximate adder design. The proposed lightweight error detection technique reduces the required gates of the error detection circuit, thus, mitigating the design area overhead. Furthermore, at the error correction process of the adder, we have proposed an extensive error detection while activating more than one correction stage concurrently. As a result, this ensures achieving an optimum accuracy of outputs for the worst case of quality requirements.

In general, approximate (speculative) adder designs use the segmentation technique to divide the adder into multiple short length sub-adders which operate in parallel. Hence, this would limit the long chains of carry propagation and result in a better performance

operations. However, the use of overlapped parts of sub-adders regarding a better carry speculation and then more accuracy becomes a significant challenge of a large design area overhead. The second approach continues mitigating this challenge by presenting a novel and simpler adder dividing technique to a number of sub-adders. The new method uses what is known as the carry-kill signal for both limiting the carry propagation and applying adder segmentation. Further, between every two adjacent sub-adders, one AND gate and one XOR gate are used for carry speculation and error (i.e., carry propagation) detection respectively. Thus, a significant reduction of the design overhead has been achieved, yet, with acceptable levels of output results accuracy. In the third final approach, simple logic OR gates are used to build the approximate adder while compensating the conventional full adders operation. The resulted approximate adder design presents very low complexity, high speed, and low power consumption. Furthermore, instead of augmenting error recovery circuit, short bit-length exact adders are used as correction stages to control the general level of output quality (i.e., without error detection overhead). At the final correction stage, the proposed design would operate the same as an exact adder.

To validate the efficiency of these approaches, a number of adders with different bit-widths are designed and synthesized showing considerable reductions in the critical delay, silicon area and more savings in energy consumption, compared to other existing approaches. In addition to acceptable levels or output errors, which are extensively analysed for each proposed design.

In this study, the proposed configurable adder designs exhibit energy/quality trade-offs at a different number of correction stages. These trade-offs can be effectively exploited to implement adders in applications, where energy can be gracefully minimised within the envelope of quality requirements. As such, designs implementation in an image processing application known as Gaussian blur filter was introduced, demonstrating the loss in the image quality at each error correction stage. The output images showed promising results to use the proposed designs for more energy-efficient applications, where output quality requirements can be relaxed.

Newcastle University Library

DEPOSIT LICENCE FOR PRINT AND ELECTRONIC THESES

Name of Student:	Student Number:
Khaled Suleiman AL-Maaitah	120365407

COVERED WORK

I would like to deposit:

- a print copy of my material in Newcastle University Library and
- an electronic copy in the Newcastle University e-Thesis Repository.

Research referred to below as "Work" is covered by this agreement and when I deposit my Work, whether personally or through an assistant or other agent, I agree to the following:

NON-EXCLUSIVE RIGHTS

Rights granted to Newcastle University Library through this agreement are entirely non- exclusive. I am free to publish the Work in its present version or future versions elsewhere. I agree that Newcastle University Library may, without changing content, translate the Work to any medium or format for the purpose of future preservation and accessibility.

DEPOSIT IN NEWCASTLE UNIVERSITY LIBRARY/THESIS REPOSITORY

The standard period of restriction for consultation is **6 months***, following which both the print and electronic version will be made available. After this time, I understand that:

- the print version will be made available for consultation in the Library or a Library to which it has been issued on inter- library loan, though it will not be permitted to leave the Library in either case.
- the electronic version deposited in the Newcastle University e-Thesis Repository will be accessible to a wide variety of people and institutions – including automated agents - via the Internet.
- An electronic copy of my thesis may also be included in the national British Library database of theses.

I understand that once the Work is deposited, metadata for the work will always remain visible, although the author retains the right to update the Work.

*[*An additional extension of restriction on consultation may be granted in exceptional circumstances with written permission of Supervisor and Dean of Postgraduate Studies. The following webpage provides further details: https://www.ncl.ac.uk/students/progress/student-resources/PGR/keyactivities/Award.htm]*

I AGREE AS FOLLOWS:

i. That I am the author, or have the authority of the author, to make this agreement and to hereby give Newcastle University Library the right to make available the Work in the way described above.
ii. That the electronic version of the Work is the same as the final, approved print version, including all corrections.
iii. That I have exercised reasonable care to ensure that the Work is original, and does not to the best of my knowledge break any UK law or infringe any third party's copyright or other Intellectual Property Right.
iv. Newcastle University Library does not hold any obligation to take legal action on behalf of the Depositor, or other rights holders, in the event of breach of intellectual property rights, or any other right, in the material deposited.

Student Signature:		Date:	14/10/2019

Students in HaSS and SAgE should return this form to: RSST, LEVEL 3, KING'S GATE
Students in FMS should return this form to: MS GRADUATE SCHOOL, LEVEL 3, RIDLEY BUILDING 1

INVESTIGATION OF RECONFIGURABLE-ACCURACY APPROXIMATE ADDER DESIGNS FOR IMAGE PROCESSING APPLICATIONS

Khaled Suleiman Al-Ma'aitah

A Thesis Submitted for the Degree of

Doctor of Philosophy at Newcastle University

School of Engineering

Faculty of Science, Agriculture and Engineering

October 2019

DECLARATION

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Where other sources of information have been used, they have been acknowledged.

Newcastle upon Tyne, November 2018

Khaled Al-Ma'aitah

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Professor Alex Yakovlev for his supervision, unlimited kind support, guidance and patience in all situations and especially hard times during all stages of my study and preparation of this thesis.

I would also like to express my sincere thanks to Dr Alex Bystrov, my co-supervisor, for his kindness, valuable comments and advice.

Huge thanks are also due to my friends, Dr. Ahmad Soltan, Dr. Ghaith Tarawneh and Dr. Issa Qiqieh for their valuable help and support that enabled me to overcome many obstacles during the past years. I would like to take this opportunity to express my profound gratitude to Mutah University for it generous scholarship and financial support for this work.

No words can express my thanks and gratitude to my family, my lovely father (Dr. Suleiman) and my awesome mother (Mrs. Fatima) and all of my sisters (Maes, Batool, Hala and Esraa) for their love, passion, encouragement and support throughout the different stages of my academic and social life.

To my second family, my wife's family, from whom I received both encouragement and compassion; my sincere respect and thanks go to all of you.

To the soul of my father in law (Mr. Esam AlKaraki), may Allah Almighty accept him with His mercy. I really miss your adorable smile, and I have been really blessed to have the pleasure of knowing you even for a short period of time.

To my beloved and gorgeous wife (Sura), thank you for being my best friend, first supporter and the only one who literally lived all the happy and tough times of this long journey with me. I pray to God to bless you and for you to remain the source of happiness in my life. May Allah give me the strength to reward you with a wonderful life ahead.

ABSTRACT

In the last decades, integrated circuits with CMOS technology show progressive scaling challenges of both increased power density and power dissipation. Meanwhile, high-performance requirements of current and future application operations show rapid demands of computing resources like power. This design conflict has pushed much effort to search for high performance and energy efficient design approach, such as approximate computing.

Approximate computing exploits the error resilience of compute-intensive applications such as image processing applications to implement approximation design techniques with different levels of abstractions and scalability. The basic principle is to relax the strict accuracy requirements in favour of a lower design complexity, thereby achieving more computational performance (i.e., speed) and energy saving. The adder arithmetic unit is considered one of the essential computational blocks in most of the applications. As such, much effort has explored new designs of an efficient approximate adder design.

This thesis presents an investigation into design enhancement, novel approximate adder designs and implementation approaches. The first approach introduces a modification to the error detection technique of a popular configurable-accuracy approximate adder design. The proposed lightweight error detection technique reduces the required gates of the error detection circuit, thus, mitigating the design area overhead. Furthermore, at the error correction process of the adder, we have proposed an extensive error detection while activating more than one correction stage concurrently. As a result, this ensures achieving an optimum accuracy of outputs for the worst case of quality requirements.

In general, approximate (speculative) adder designs use the segmentation technique to divide the adder into multiple short length sub-adders which operate in parallel. Hence, this would limit the long chains of carry propagation and result in a better performance

operations. However, the use of overlapped parts of sub-adders regarding a better carry speculation and then more accuracy becomes a significant challenge of a large design area overhead. The second approach continues mitigating this challenge by presenting a novel and simpler adder dividing technique to a number of sub-adders. The new method uses what is known as the carry-kill signal for both limiting the carry propagation and applying adder segmentation. Further, between every two adjacent sub-adders, one AND gate and one XOR gate are used for carry speculation and error (i.e., carry propagation) detection respectively. Thus, a significant reduction of the design overhead has been achieved, yet, with acceptable levels of output results accuracy. In the third final approach, simple logic OR gates are used to build the approximate adder while compensating the conventional full adders operation. The resulted approximate adder design presents very low complexity, high speed, and low power consumption. Furthermore, instead of augmenting error recovery circuit, short bit-length exact adders are used as correction stages to control the general level of output quality (i.e., without error detection overhead). At the final correction stage, the proposed design would operate the same as an exact adder.

To validate the efficiency of these approaches, a number of adders with different bit-widths are designed and synthesized showing considerable reductions in the critical delay, silicon area and more savings in energy consumption, compared to other existing approaches. In addition to acceptable levels or output errors, which are extensively analysed for each proposed design.

In this study, the proposed configurable adder designs exhibit energy/quality trade-offs at a different number of correction stages. These trade-offs can be effectively exploited to implement adders in applications, where energy can be gracefully minimised within the envelope of quality requirements. As such, designs implementation in an image processing application known as Gaussian blur filter was introduced, demonstrating the loss in the image quality at each error correction stage. The output images showed promising results to use the proposed designs for more energy-efficient applications, where output quality requirements can be relaxed.

PUBLICATIONS

The publications that were produced as a part of research reported in this thesis are listed as follows:

- **Khaled Al-Maaitah**; Issa Qiqieh; Ahmed Soltan; Alex Yakovlev, *Configurable-accuracy approximate adder design with light-weight fast convergence error recovery circuit*, IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), 2017, pp 1-6.

- **Khaled Al-Maaitah**; Ghaith Tarawneh; Ahmed Soltan; Issa Qiqieh; Alex Yakovlev, *Approximate adder segmentation technique and significance-driven error correction*, 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2017, pp 1-6.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

CMOS complementary metal-oxide-semiconductor

VOS voltage over-scaling

LSB least significant bits

MSB most significant bits

FPU floating-point unit

PSNR peak signal-to-noise ratio

EDC error detection and correction

LWC lightweight check

LVA load value approximation

SIMD single instruction multiple data

FA full adder

QoS quality of service

VDD voltage source

Ceff effective switched capacitance

DSP digital signal processing

LOA lower-part-oR adder

MA mirror adder

AMAs approximate mirror adders

ER error rate

ED error distance

MED mean error distance

RED relative error distance

MRED mean relative error distance

NED normalized error distance

RCA ripple carry adder

CSL carry select adder

CLA carry look-ahead adder

SPGs sum, propagate, generate signals

ETA Error tolerant adder

VLSA variable latency speculative adder

ACA accuracy-configurable adder

GeAr general architectural design of accuracy-configurable adders

GDA gracefully-degrading accuracy-configurable adder

CPD cumulative probability distribution

PDP power-delay-product

FPGA field-programmable gate array

Part I

Thesis Chapters

1

INTRODUCTION

1.1 DESIGN SCALING CHALLENGE

Over the last decade, the size of computation work-loads has dramatically increased because of extensive data, demanding applications and communication features. Hence, a massive amount of resources should be available in order to meet the current and future computation requirements. However, the continuing advances in technology scaling introduce a critical issue of delivering high-performance designs without a significant increase in energy consumption [9, 10, 15, 33].

Several studies have explored the relationship between technology scaling and designing references like Moore's law (reduction in transistor size leads to increase the total number of transistors per chip with an effective cost) [95]. These studies argue that, for the past three decades, each of Moore's law and Dennard scaling which postulates a constant power density despite transistors size reduction [93, 76, 81] have led to an exponential increase in computational performance, yet, at constant energy consumption and transistor cost [21].

However, the fast development of higher performance technologies and circuits becomes coordinated with a decline of Moore's law. This implies that, as the downscaling of the complementary metal-oxide-semiconductor (CMOS) is wholly stretched to limits; technology scaling would become unlikely to drive computing

shortly [76, 81, 77, 40]. Moreover, with the end of Dennard scaling (2005-2007), the per-transistor performance power efficiency is not held on with known power-reduction techniques at various abstraction levels [21, 61, 22]. Thus, the simultaneous increase of performance with a higher clock frequency and supply voltage reduction have significantly weakened. In other words, CMOS technology scaling might still sustain Moore's law; however, with a large increase in power density since transistor scaling and voltage scaling are no longer consistent with each other.

Further, the failure of Dennard's law would result in the so-called the Dark silicon problem, which imposes the portion of the circuit that cannot be powered at the nominal operating voltage. This happens to assure the circuit stays within the power density and the thermal design power restrictions (i.e., to avoid more heat which might destroy the chip) [21, 79].

Other studies have proposed that is expected that, in the coming decade, the increased demands of performance would shortly outpace the growth in available resource budgets, and a significant gap would take place in the near future [24, 55, 39].

Since CMOS technology scaling is becoming less effective in improving system capability, new approaches are required for more resource-efficient computing systems, instead of just over-provisioning of resources alone. This introduces the necessity of exploring new computing paradigms that convey more energy efficiency and moderate the functionality out of computing platforms across the spectrum, from mobile and deeply-embedded devices to servers and data centres. One of these approaches is approximate computing, which has been globally considered as attractive design approach and has even driven more attention in the scientific community, during the last few years.

A brief description of approximate computing is presented in the following section.

1.2 APPROXIMATE COMPUTING

As illustrated in the previous section, designing of low power consumption and small area circuit becomes a basic requirement for electronic products. Further, due to the increase of technology scaling and embedded mobile applications, the required level of computational workload grows massively. For example, the applications that involve media processing (e.g., images and videos), search engines, recognition, and data mining would consume large computing resources with hard restrictions.

A common characteristic of such applications is their ability to work sufficiently despite the existence of a low level of errors. This characteristic would allow few errors to take place during the intermediate operations and result in a highly acceptable output quality. Thus, this leads to a state that strict exact results are not necessary, and approximate (i.e., less than optimal) result can be sufficient and hard to be distinguished from the optimum one. As a result, this precision-resilient characteristic is exploited by a new design approach known as approximate computing [30].

Remarkably, approximate computing in image processing application mainly leverages the perceptual limitations of users due to the human brain ability to fill in large number missing elements of the resulted images or videos. In other words, approximate computing exploits the different accuracy requirements between the user and the application and the exact operations provided by the computing system, for achieving multiple optimisations.

Approximate computing design approach might include hardware and software-based techniques that expose incorrectness to a specific part of circuit architecture or program code. The trading-off computational quality for computational efforts would result in more speed, lower power dissipation, and smaller area designs, yet, with controlled bounds of quality loss.

However, quality-loss controlling has emerged as a significant challenge while applying an approximation. For example, implementing approximation to critical portions of code, control flow or significant memory access operations can lead to severe unacceptable quality degradation. Hence, a careful selection of approximable code or data portion and the approximation technique can effectively improve error analysis and recovery during approximation. Moreover, efficient output monitoring mechanisms would ensure meeting any predefined quality specifications, which would lead to decrease the probability of severe quality loss [69, 75, 102].

As mentioned, approximate computing can target both hardware and software levels of abstraction. The software-based approximation specifies the approximable portion of code in a program or algorithm and applies approximate techniques such as omitting least significant code portions or early terminating of the program loop. On the other hand, hardware-based techniques would explore the chance of modifying designs at circuit and architecture levels of abstraction. Furthermore, extending common techniques like truncation and voltage over-scaling (VOS) would result in wider ranges of approximation scalability [66, 51].

As an example of hardware level approximation, adders have attracted remarkable interest. This is due to the fact that adders are key arithmetic units in digital systems and intensively used by other arithmetic operations such as multiplication and divi-

sion; thus, they have been regarded as attractive blocks to be approximated for the goal of the more efficient computing system [105, 48, 106, 92, 5].

The following section presents our motivation in this work for more approximate computing effort regarding the approximate adder design.

1.3 MOTIVATION

In a state-of-the-art review, the majority of adder design approximations have exploited two main observations:

1. The addition operation of the lower order part or the least significant bits (LSB) of the adder can be approximated, due to the limited significance (i.e., contribution) of this part in the final result, in contrast to the higher order part or the most significant bits (MSB) of the adder, which should be strict to the exact operations to preserve full correctness.

2. The longest (worst case) carry chains are limited and rarely happened; thus, they present the probability to be speculated in advance. As a result, much low power and high-speed approximate (speculative) adder designs have been introduced.

However, several challenges are still facing such approximate and speculative adder designs. For instance, approximate adder designs with a lower order approximate part and an accurate higher order part show large error rates, especially for small numbers. On the other hand, although speculative adder design presents a more top speed of the addition operation, both the used overlapping techniques and the augmented error detection and correction (EDC) circuits would result in the large overhead of area, delay and power

consumption. Hence, mitigating the effect of these challenge has been considered the primary motivation of this work.

In detail, this work proposes more efficient designs of approximate and speculative adders, when compared to other related efforts. For example, speculative adder designs are presented with a simple sub-adders segmenting technique, better accuracy and a lower error recovery circuit overhead. Further, approximate adder design is performed with a general output quality control and a comparable design overhead to a conventional adder design. The proposed designs are based on the significance-driven structure of accuracy controlling during run-time so as to achieve quick convergence to the exact result. The following section illustrates the major contributions of the proposed approximate adder designs with a different level of accuracy during run-time and their effect on the targeted image processing application.

1.4 AIM OF THE THESIS

This work aims to mitigate or even eliminate the challenge of design overhead of the configurable accuracy adder design. Several techniques have been proposed to meet this goal. However, assuming the error source of the resulted adder outputs is limited to approximate adder operations (i.e., functional level) and not due to gate or transistor levels.

1.5 CONTRIBUTION

The new contributions are mainly based on the previously mentioned motivations of mitigating approximate and speculative adder challenges including the design overhead of sub-adders seg-

mentation and the error recovery circuit. This improves accuracy and scalability of large bit-width adders, and further, controlling the general output quality instead of frequent activation of error recovery processes, thus, maximizing the approximation benefits. In this work, the following points present the main contributions regarding the targeted configurable-accuracy (i.e., the different level of accuracy) approximate adder designs and their implementation results in an image processing application.

- A configurable-accuracy approximate adder design, which has been proposed with augmented significance-driven correction stages. The proposed design shows a lower overhead of error detection by using lightweight checks, and further, achieving the optimum accuracy similar to an exact adder in the last correction stage. Moreover, it improves the approximate design feasibility for adders with larger bit widths [3].

- Presenting a novel speculative adder segmentation technique by using the principle of carry kill signals in order to limit the carry chain of the adder. The new technique divides the adder into an independent number of sub-adders in contrast to other dependent dividing techniques like intensive single-bit shift or overlapping. To preserve accuracy levels, a carry prediction technique is also proposed, in addition to significance-driven configurable correction stages during run-time. This significance-driven error recovery structure would start correcting the errors with the major effect in the final output, which resulted from the higher order or the most significant bits (MSB) of the adder. As a result, this would confirm a quick convergence to the exact addition operation output [4].

- Presenting a new approximate adder design that eliminates frequent error recovery process overhead. For the addition approximation stage, the proposed design has replaced the conventional arithmetic addition with simple OR gates at each bit location and results in a direct sum bit. The correction process has been implemented through a significance-driven multi-stage structure, yet, by using a short bit-width exact adder for each stage. Hence, at the final correction stage, the proposed design would work as an exact adder that guarantees full accuracy. This general output quality controlling eliminates the augmented overhead of error detection and correction circuitry.

- All the proposed approximate adder designs have replaced the conventional adder units in an image processing filter known as Gaussian blur filter. The implementation results have provided a highly acceptable output images quality. As such, this might confirm the feasibility of the proposed design to be applied in similar error resilience applications, yet, with low design overhead.

1.6 THESIS ORGANIZATION

Chapter 2 includes a background of approximate computing approach, image processing and a brief description and evaluation of related work of approximate and speculative adder designs. Chapter 3 presents the first contribution of a scalable speculative adder with configurable-accuracy during run-time, in which the design overhead of error recovery has been mitigated and introduces a smaller area, better accuracy and more scalability for large bit-width adders. The second contribution of the new speculative adder segmentation technique is explored in Chapter 4, where

simpler sub-adders' segmentation technique is introduced without using overlapping. The proposed method is combined with a quick error detection process; thus, it results in lower design overhead and highly acceptable output quality. Approximate adder design with simple logic gates and a general output quality controlling is placed in Chapter 5. However, the proposed adder design has no augmented error detection circuit, and the resulted design parameters are approximately comparable to the exact conventional adder. Finally, Chapter 6 concludes the thesis and points to future work.

2

BACKGROUND AND LITERATURE REVIEW

In this chapter, a brief background of the major parts of this work is presented. In the first part, we start with a general review of approximate computing motivations, challenges and techniques, in addition to a short review of basic adders. The second part explores a literature review of related approximate and speculative adders as well as their current challenges. The definition of the used Gaussian blur image filter is placed at the final part.

2.1 APPROXIMATE COMPUTING

The approximate computing design approach exploits the error tolerable applications characteristics in order to create more chances for efficient designs. The new designs are based on efficiency-quality trades-off (i.e., speed and low power design with a limited output quality loss) [78, 30].

The following sections present a brief background of the motivations, problems, solution approaches and examples of the used approximate computing techniques.

2.1.1 *Motivations*

In many applications, the necessity of using approximation might be different; for instance, frequently rounding the result of floating-point unit (FPU) is an unavoidable operation. Thus, the optimal

exact result may not be known. However, the approximated result can be efficient and sufficient. Generally, several approaches can use the approximate computing as a chance for more efficiency optimization (e.g., more speed and lower power consumption). The following points discuss the motivations of approximate computing and the opportunity of implementation.

2.1.1.1 *Essential Requirements for Approximation*

Common factors can lead to employing unavoidable approximation to the computation operation and then the output result. These factors might include the inherently noisy input, data with limited precision and hard real-time constraints. Hence, the application is forced to compute a sufficient (i.e., not exact) output [56].

2.1.1.2 *Error-Resilience of Users and Applications*

- Perceptual limitations: Applications with analogue inputs and/or outputs which operate on noisy real-world data exploit the ability of the human brain and vision tolerance (i.e., the ability to automatically fill in the missing information) of the distorted image or video frames.

- Redundant input data: The redundancy of data values increases the chances for algorithm approximate computation, and further, can be used to internally recovering from errors. Hence, the algorithm might be lossy, yet, still suitable [75, 86, 49, 72].

- Significance in the final output: Inputs or code portions can be classified, depending on their significant impact on the final output. For instance, the lower order bits or the least significant bits (LSB) have smaller significance and a minor

impact on the final output quality when compared to the higher-order or the most significant bits (MSB) [13, 23, 68, 28].

- Applications with no unique answer, such as the web search.

2.1.1.3 *Performance Optimization*

When exploiting a combined error resiliency characteristic and minimum acceptable quality levels, such as minimum error rate (10%) and peak signal-to-noise ratio (PSNR) values (30 dB) in image processing applications [68], the approximate computing can be extended for more efficiency optimization (improving performance and energy efficiency). For example, introducing techniques like reducing the memory refresh rate and voltage [55, 56], skipping memory access [99], and loop termination [12], can lead to more efficient performance, however, with minor and acceptable quality degradation.

2.1.1.4 *Design Reconfigurability*

Approximate computing might allow a better management of resources. In other words, the user may have the ability to expand efforts (e.g., power, area) as much as dictated by the output quality requirements, providing knobs to trade-off quality with efficiency. This is compared to the conventional computation, where every computation is executed to full reliability [11].

2.1.2 *Challenges of Approximate Computing*

Similar to any design approach, approximate computing has several challenges, which might limit the approximation techniques

benefits. The following points show a few major challenges of approximation regarding hardware computation blocks and software-based implementations.

2.1.2.1 *Critical Domain of Approximate Computing*

Applications like cryptography and compression programs require hard and restricted levels of accuracy. With this being the case, they can be considered not amenable to approximation. Moreover, approximation techniques can be valid for a certain range; for instance, aggressive task skipping might lead to program failure or a corrupted output. Further, the advantages of approximate computing may become limited. For example, an unreliable approximated memory does not reduce the number of operations on the data, and vice versa. Hence, a good selection of the targeted application (i.e., considerable error resilience) and the range of approximation should take place cautiously.

2.1.2.2 *Application Dependent Approximation*

One or multiple approximation techniques can be applied to an approximable application. However, the challenge stated that there is no approximate technique that can be applied universally to all targeted applications. Hence, the approximate techniques are considered an application-based dependent and need to be determined according to each application requirement [6].

2.1.2.3 *Design Overhead and Limited Scalability*

Several approximate computing techniques introduce large implementation overhead; for instance, the augmented error detection and correction (EDC) circuitry of approximate adders incurs a large design overhead, and thus, reducing the gains of approximation.

Other software-based techniques might require the user to write multiple approximate versions of a program. Consequently, this would consume huge designing efforts, and further, would not scale to more complex programs.

2.1.2.4 *Accuracy Recovery*

A key challenge of approximate computing is the output accuracy controlling mechanism. This mechanism should control the level of resulted errors, and thus, maintain the desired output quality. The error control block should have an efficient error detection and correction techniques, based on predefined thresholds (e.g., worst-case bounds) or predetermined tunable knob(s) to trade-off quality with efficiency [88]. Hence, if the error level or quality degradation violates the threshold, the application must recover the desired quality and might need to be executed precisely, a matter which increases computation overhead [26].

2.1.3 *Solution Approaches*

The proposed techniques in the following points aim to address the above-mentioned approximate computing challenges.

2.1.3.1 *Identifying the Approximable Parts*

The basic step in the majority of approximate computing techniques is to find the approximable variables, blocks and code portions. This can be achieved by using a straightforward option such as approximating the lower order bits (LSB) of input data. However, this may require a deep inspection of the application characteristics by using a testing technique like error injection or

statistical measurements, and then, monitoring the correlation between outputs and targeted executed data and operations [71].

2.1.3.2 *Preserving Quality by Output Monitoring*

The overhead of monitoring application output to preserve desired quality might be mitigated by using several approaches. These approaches might include periodic checking of general quality levels, instead of full frequent checking and recovering of errors. Further, error verification and correction can be accomplished using lightweight check (LWC)circuitry. For example, Grigorian and Reinman in [26] proposed (LWC) circuit, which indicates an unacceptable quality loss. Accordingly, the exact computation would be activated to recover the undesired levels of quality. Otherwise, if the output quality shows acceptable levels, the approximation would be deemed satisfactory. Hence, this would bound the worst-case error, and saves energy without compromising reliability.

2.1.3.3 *Programming Language Support for Approximate Computing*

Programming languages can play a vital role in solving approximate computing challenges. For instance, Sampson et al. in [74] proposed Java extension type qualifiers that can classify portions of the code as precise (guaranteed strict accuracy) or approximable. According to this classification, approximated computations, storage and algorithms can be used by variables denoted with an approximate qualifier. They showed that their approach can save a large amount of energy with small accuracy loss.

Another effort by Yazdanbakhsh et al. in [49] presented Verilog annotations, which provide suitable syntax and semantics for an approximate hardware design. They showed that the designer

can specify both critical (precise) and approximable portions of the design. Furthermore, they provide the chance of reusing the approximate modules in different designs that might have different accuracy requirements, yet, without requiring reimplementation.

Based on these solution approaches, the following sections present several approximate computing techniques that have been explored by many state-of-the-art efforts.

2.2 APPROXIMATION TECHNIQUES

Once the process of identifying the approximable variables and operations within the application has been accomplished, different approximation methods might be implemented. We now discuss these methods, in the context of the approximation techniques and related applications in which they are used. Fig. 2.1 shows the general taxonomies of approximate computing techniques. The following points present a brief description of each approximation method with related examples.

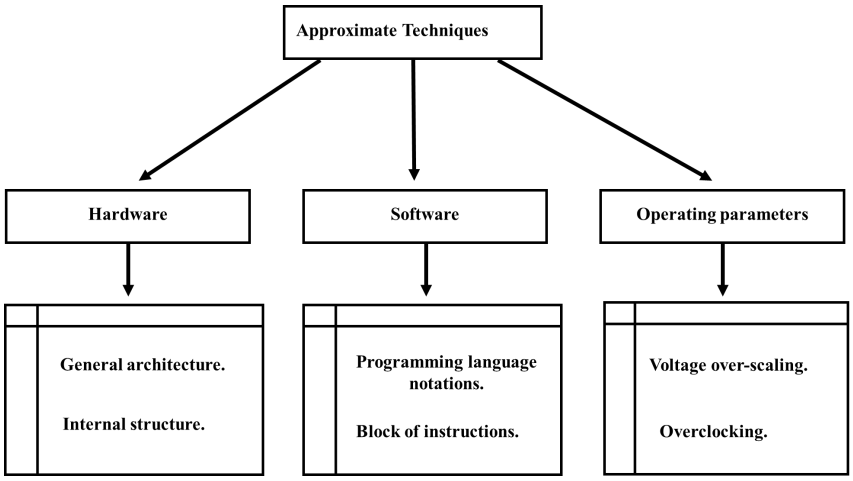


Figure 2.1: General taxonomies of approximate computing techniques.

2.2.1 *Software Techniques*

In software domain, approximate computing introduces the programmers an opportunity for creating lower restricted programs and applications versions. These new versions would contain several approximate code techniques that allow the application to work more efficiently, yet, with different trade-off error levels (i.e., accuracy) [60].

The crucial point of this software-based approximation is to identify the portions of code (instruction, operation and storage) that are tolerable to be approximated. These selected approximable portions must not harmfully affect the final output quality.

Several techniques can be used to make application's code operating with more performance and controlled levels of quality degrading. Such techniques might include skipping tasks, loop perforation, omitting portions of code, replicating local statistically correlated matrices of inputs, and using memory values to compensate computations. Nevertheless, error levels have to be controlled with specific user-defined metrics (e.g., predefined error rate threshold).

The following points present several examples of software-based approximate techniques.

2.2.1.1 *Loop Perforation*

The idea of the loop perforation method has been implemented by skipping some iterations of a loop in order to reduce computational overhead.

For example, Sidiroglou et al. in [85] introduced several global computational algorithms that work with loop perforation such as search space enumeration, in which some items computations

can be skipped and one of the remaining items is returned from the search space. On the other hand, for exploring performance versus accuracy trade-off, they used given perforation rates (i.e., a fraction of iterations to skip) as a test threshold. While exploring combinations of all tunable loops (those loops whose perforation produces efficient and still acceptable computations) on training inputs, the combinations that produce error are discarded and those with acceptable accuracy are selected. As a result of this effort, performance improvements have been shown, along with the ability of performance-accuracy trade-off exploration.

2.2.1.2 *Load Value Approximation*

In general, once a load's miss in a cache memory happened, the required data must be fetched from the next-level cache or main memory, and thus, inquiring large latency. However, the error tolerance characteristics of the approximable applications allow estimating the missed load value, which is known as load value approximation (LVA). Hence, this would hide the cache miss latency and allow a processor to progress without any delay for a response [54].

An example of LVA-based efforts, Miguel et al. [86] presented an LVA technique for graphics applications. In their proposed technique, memory block would be fetched occasionally to train the approximator, in contrast to traditional block fetching on every cache miss to confirm the correctness of prediction. However, in case of not matching of the estimated value and exact value, no rollbacks are required in such error tolerance applications. As a result, this would reduce the overhead of memory accesses significantly.

2.2.1.3 *Memoization*

The memoization methodology is an optimization technique that exploits the results of inputs functions similarity. In other words, storing the results of expensive function calls for later reuse with identical values of inputs. Consequently, exploring more speed up and reducing the computation overhead. This approach is used by several approximate computing techniques [94, 73].

An example of memoization concept, Rahimi et al. [67] exploited the value locality of a parallel program, which is exposed to all lanes in the single instruction multiple data (SIMD) architecture. Based on this, they proposed a technique of reusing the result of data item's error-free execution across different parallel lanes of the SIMD architecture, and thus, reducing the delay of error recovery overhead. The memoized result would be used to correct, either precisely or approximately, an erroneous execution on same or adjacent data items. Precise correction is implemented by comparing the corresponding bits of the inputs of the instructions (i.e., bit-by-bit matching). On the other hand, approximate correction is implemented by matching a certain number of a least significant fraction of inputs. Their SIMD architecture consists of a single strong lane and multiple weak lanes. Hence, the result of an exact floating point FPU instruction on a strong lane can be memoized and reused to approximately recover any weak erroneous lane, and as a result, reducing the overhead of timing recovery for a large fraction of erroneous instructions.

2.2.1.4 *Multiple Inexact Program Versions*

Code approximation extension was implemented in well-known programming languages, such as Java and Verilog [49, 74]. This can be conducted by a specific user's notations that classify the

portions of code as critical and approximable, and thus, identifying the possibility of applying any kind of programme-based approximation strategies, mentioned before.

As an example of multiple inexact program versions, Baek and Chilimbi [7] presented a programming model framework that provides approximated versions of the program by approximating functions and the loops perforation technique. However, the loss in quality of service (QoS) is measured by a user-defined function for checking the difference of output results of the precise and approximate function versions. Additionally, they used periodically statistical measuring of the QoS loss at runtime in order to update the approximation and guarantee the targeted quality levels.

2.2.2 *Operating Parameters Techniques*

2.2.2.1 *Voltage Scaling*

For static-timing analysis of conventional design methodology, all the operations (circuit paths) would be guaranteed to meet timing limits. However, when voltage source (VDD) starts to be scaled down, timing errors will be induced rapidly, and thus, degrading the output signal quality, then, reducing the benefits of potential energy reduction. Voltage scaling is considered to be the most key factor of decreasing the digital circuit energy consumption. This is due to the fact that the power consumption is dominated by dynamic power dissipation P_dynamic, which is given by [65, 64, 60]

$$\mathbf{P}_{dynamic} = \mathbf{C}_{eff} \,.\mathbf{V}_{DD}^{2} \,.\, \mathbf{f}\,, \tag{2.1}$$

where VDD is the supply voltage, effective switched capacitance (Ceff), and (f) is the clock frequency. It can be noticed that scaling down the supply voltage would lead to an overall quadratic reduction in the energy to complete a task. Nevertheless, while preserving a fixed performance (i.e., operating frequency), aggressive scaling of supply voltage will cause timing errors.

Several techniques of approximate computing have been explored for new procedures to gracefully over-scale voltage below the circuit's lower voltage [66, 51]. Generally, they differ in the way they deal with timing induced errors due to insufficient voltage, which cannot guarantee timing correctness on all paths (i.e., timing starvation).

In some efforts, correction mechanisms are introduced in such a way that the system becomes able to tolerate timing errors induced by voltage over-scaling (VOS). For example, design efforts in [86, 32, 82], specifically targeted digital signal processing (DSP) type circuits, such as filters. A main computing block in the circuit is targeted with lower voltage for more energy reduction. On the other hand, error correcting block is implemented with a normal voltage value, and thus, error-free results.

A general approach to mitigate the effect of the induced timing errors is to identify the significance of computations that need to be protected against voltage-over-scaling and those that can tolerate them as shown in the following examples.

- In [59], meta-functions accumulator was considered the main block which might experience time-starvation under voltage over-scaling. By using techniques such as dynamic segmentation and delay budgeting of chained units, the accumulator can work more gracefully under VOS, resulting in that the quality-energy trade-offs are improved.

- Even in the same block, not all computations have the same significance to the final output quality. In [58], The significant computations are identified and then protected under VOS, by allowing them to consume an additional clock cycle for completion. In opposite, the insignificant computations are allowed to produce occasional errors.

Furthermore, using methods of predicting the occurrence of early timing errors can lead to a significant reduction of quality loss under VDD scaling. This might be accomplished by using operand statistics such as the design effort in [31], which is based on the knowledge of operand statistics. They stated that while VDD is scaling down, the large magnitude of timing errors shows high probability to happen in the addition of small numbers with different signs. Due to such additions timing-critical paths, they are proposed to apply the addition of opposing signs small number at the last step of computation, and thus, limiting the magnitude of error and improving the output quality.

2.2.2.2 *Over-Clocking*

In general, for any given voltage, the circuit will have a maximum "stable" speed where it still operates correctly. Approximation by using over-clocking is to gain further performance from a given component by increasing its operating speed while applying constant voltage. This happens by setting the circuit's working frequency at the higher end of the margin of the clock frequency. An example of work towards "over-clocking friendly" is exercising an over-clocking technique for serial operations in online arithmetic implementations which can gain substantial performance benefits with graceful degradation of timing violations [17, 18, 64].

2.2.3 *Hardware Techniques*

In this part, we investigate the efforts of the inexact basic arithmetic units' design, especially for the approximate adders. This part also includes a brief description of approximation efforts of multiplier units.

From state-of-the-art literature, the general methodologies of conventional adder approximation can be classified into two groups. The first group concerns about the production of a new adder architecture with better efficiency. The second group applies changes to the internal structure of the computing units in the adder. The change can be implemented at the gate level by using a few logic gates to apply the addition operation rather than the standard circuit of the full adders, and further it might take place at the transistor level of the adder by removing a number of transistors which would not significantly harm the adder operation and outputs.

2.2.3.1 *General Architecture Approximation*

In this section, adder approximation techniques would target the general architecture of the adder (i.e., not the internal gates or transistors levels). The following two parts explore examples of the key ideas of both approximate and speculative adders that follow this classification.

- Bit-significance approximate adders

  Several approximate adder designs exploit the general classification of the significance of the adder bits in the final sum result. The main idea is that the lower adder bits (LSB) contribute with the lower portion of the final sum value and have

the smaller effect of errors (i.e., not harmful to the general accuracy), compared to the higher order bits (MSB) of the adder. This motivation of significance analysis leads to dividing the multiple bit adder into two parts:

1. The most significant or accurate part, which retains the exact conventional carry propagation addition to be performed to the higher order bits (MSB) of the adder.
2. The less significant or inaccurate part, in which a less complex circuit design or a modified carry free (i.e., not generated or propagated from previous bit location) addition function can be implemented to each bit of the lower order bits (LSB) of the adder.

This architecture of the approximate design with accurate and inaccurate parts would maintain the low magnitude of errors since the higher order bits of the sum are expected to result in values that are close to the exact sum result. On the other hand, for the inaccurate part, the modified inexact addition function can be performed by using new less complex or simplified versions of the conventional adder at the transistor level [28, 29] or using simple logic gates to compensate the conventional arithmetic addition operation [19, 98, 50]. However, the length of both parts not necessarily needs to be equal.

As an example, the lower-part-oR adder (LOA) design divided the adder into two parts, the accurate part for the higher order (MSB) bits and the inaccurate part for the lower order (LSB) bits range [50]. In the inaccurate part, the conventional full adder (FA) unit was replaced with simple OR gate at each bit location in order to estimate the sum value. Additionally, an AND gate was used to generate the carry-in to the accurate upper part that

performs the exact addition operation, specifically, when both inputs values to the most significant bit in the lower part are '1'. In summary, the LOA performs a carry-free addition in the less-significant lower part of the adder, which a result, shows very low power dissipation with smaller area and better performance.

- Segmented Speculative Adders

  This type of approximate adders is generally based on the fact that the carry propagation critical path rarely implies the worst-case adder delay (i.e., carry propagation from the first bit to the last bit of the adder).

  Approximate speculative adder designs have proposed that the carry propagation chain can be limited to a number of previous bits to calculate the sum with a high probability. Hence, the conventional adder can be segmented into a number of blocks (sub-adders), which have an equal number of bits. The carry-in to each sub-adder would be speculated to either '0' value or by using a carry prediction technique that is related to the previous sub-adder. Additionally, the input bits to the sub-adder can internally manipulate the propagated carry-in values with a high probability (i.e., input bits might generate correct carry and can be propagated to a considerable number of bits within the sub-adder).

  All the segmented sub-adders would work in parallel, calculating the sum bits values of each sub-adder at the same time. Thus, the speculative adder designs would result in high speed (limited to the bit width of the sub-adder) achieving sub-logarithmic delays (i.e., the operation complexity is less than log(n) in an n-bit adder [48, 91]), yet, with a high probability of acceptable accuracy.

However, assuming that the bit width of each sub-adder equals (k), speculative adders' trade-off between accuracy and delay would depend crucially on the size of (k). Although the smaller value of k speeds up the addition operation, it might limit the accuracy. Conversely, larger values of k can improve the accuracy level; Nevertheless, they limit delay reduction benefits.

Speculative adders with different segmenting techniques can show more accuracy and a higher level of output quality without affecting speed benefits. For example, speculative almost correct adder (ACA) in [91] used intensive blocks overlapping (overlaps (k-1)-bits between successive sub-adders) to increase the probability of the correct carry speculation. The proposed speculative adder involved N-k+1 sub-adders, and thus, enforced power and area overheads. Moreover, the design was augmented with error detection and correction circuitry (EDC) to recover any detected errors to reach the full accuracy with variable latency (i.e., error recovery consumes an additional clock cycle of the addition function).

- Accuracy-Configurable Adders

  The previously discussed approximate adders have the challenge of the fixed accuracy level at the design time (i.e., not flexible to be changed within multi-levels during run-time). Hence, the designed approximate adder will be dependent on the accuracy level of the targeted application and might be not fit another error-resilience application with a different level of accuracy. Therefore, one of the key problems of approximate adders is the redesign efforts. To resolve this challenge, several design efforts propose accuracy-configurable adders [91, 37, 8, 80, 100]. In this approach, Multi-Stages error recovery structure is used. Here at each stage, error detection and correction (EDC) circuit

can be enabled to detect and correct errors or just be disabled according to the required accuracy level during run-time. In this manner, the accuracy-configurable adder trades-off speed-power-accuracy during run-time, according to requirements of applications, and consequently, generalises the design.

An example of this type of accuracy-configurable adders, the design effort in [37], in which a speculative segmented adder was proposed by introducing a middle sub-adder that shares an overlapped part from the previous sub-adder in terms of increasing accuracy as a first interest. Further, a pipelined multi-stage error recovery circuit structure was implemented. Each stage here employs a different level of accuracy and can be enabled or disabled during run-time by a kind of a power gating technique.

2.2.3.2 *Internal Structure Approximation*

In the approximate adder circuit, the inexact addition function can be performed by using less complex versions of the conventional adder at the transistor level [28, 29]. This happens by making changes to the internal structure of the transistors of the adder.

For example, the approximation of the 28-transistors mirror adder (MA) architecture [28]. The idea of analysing the effect of truth table entries to the sum result at each bit was the motivation to produce five approximate MAs (AMAs). The new approximate mirror adders (AMAs) explored different attempts to reduce the logic complexity by removing a number of transistors from the original MA circuit architecture, yet, by ensuring the low significance of resulted output errors. This logic reduction at a transistor level leads to faster charging and discharge of the node capacitance in the new approximate MA. Consequently, this results in a lower

circuit complexity, more speed (i.e., a shorter critical path inside the MA), and lower power dissipation. Hence, this would trade-off accuracy for energy, area and performance.

2.2.3.3 *Approximate Multipliers*

Approximate multipliers design starts to receive better attention in the last few years. Generally, the key design idea is to reduce the critical path of adding the partial products. This can be done by the use of speculative adders to compute the sum of the partial products [101, 103], or omitting some less significant bits in the partial products (i.e., some adders can be removed for more delay benefit) [50, 62], or by using approximation techniques like logic compression as in [27] in order to reduce the levels of partial addition, or using inexact small block in order to be used as a building block in a larger multiplier for an approximate computation [89]. As a result of approximate multiplier implementation, the area has been reduced, compared to the exact multiplier, leading to a shorter critical path and a better speed.

2.3 SCALABLE EFFORT DESIGN FOR APPROXIMATE COMPUTING

In several cases, a more efficient approximation can happen while tuning the effort expended by the approximate technique. This tunable effort would be based on the tolerance level of the targeted portion, and the accuracy significance of operation results in the output. In the state of art, several efforts have used this approach per task basis in order to maximize the gains and reliability of approximation [57].

For example, Grigorian et al. [27] presented a technique that uses both precise and approximate accelerators and uses error analysis to set the accuracy constraints. During execution, first steps operate with relaxed approximate computation, and consecutively, the approximation complexity increases and ends with a precise computation. Based on the output quality measurement at each step, the tasks that have met the specified output quality are committed, and only remaining tasks move to the next stage. They showed that most of the tasks are expected to be committed in early stages, thus, reducing the overall overhead.

Venkataramani et al. [89] presented a scalable approximation technique for improving the energy efficiency of supervised machine-learning classifiers. In their technique, the difficulty of input data would determine the tuning of the computational effort. Here simple inputs would be processed within few stages, and hard inputs would require going through multiple stages. They showed that the number of operations per input is decreased, and thus, results in reducing the energy consumption of the classification process.

2.4 SIGNIFICANCE-DRIVEN DESIGN

The main idea of significance-driven design is based on exploring the most significant part of the design regarding the resource consumption and/or the final output. For example, some image processing applications have specific operations that consume a large portion of execution time or power consumption, and further have mathematical coefficients that affect the final output result significantly. Approximate designs would exploit these significant

parts since they present the chance for new efficient designs with more speed and a lower power consumption [58, 57].

In this work of approximate adder designs, we consider the higher order bits as the most significant part of the adder. This is due to their significant influence in the final sum results when compared to the lower order bits. Thus, we have implemented a significance driven error correction structure for each proposed approximate adder design. In the proposed error recovery structure, the most significant (higher order) bits or sub-adders would be corrected first (i.e., at early stages). Hence, this would guarantee fast convergence to exact addition values with a small delay.

As a summary of the targeted levels of the approximate computing techniques, Table. 2.1 presents the three mentioned taxonomies with their features and design limitations.

2.5 APPROXIMATE ERROR METRICS

Since the approximate computing has been explored increasingly by many efforts, a new performance metrics become required in order to evaluate the efficiency of the approximate design. Several analytical efforts have proposed error metrics, which can be used for quantifying approximation errors, and then the reliability of approximate designs. In the following, a list of generally used error metric is introduced aligned with a brief explanation [37, 42, 44, 53, 83, 84, 87, 90].

- The error rate (ER): also known as error frequency [53], is defined as the fraction of incorrect outputs out of a total number of inputs in an approximate circuit. ER is expressed by the fol-

Table 2.1: Taxonomies of approximate computing approaches

Approach	**Methodology**	**Features and Limitations**
Hardware-based approximation	Approximate the internal architecture of computing blocks such as adders and multipliers.	Achieve a better performance with a lower power consumption. However, the error recovery process might incur a larger execution delay and a design area.
Software-based approximation	Apply code instructions to limit parts of the running programs such as loop perforation.	Apply a lower execution time by stopping or neglecting some parts of an executed program or instructions to achieve better performance and total power consumption. However, a careful selection of tolerable parts and approximation techniques should exist to avoid sever final quality degradation.
Operating parameters scaling	Apply the parameter value scaling to the operated circuit, such as voltage over-scaling and overclocking.	Very beneficial of decreasing the power consumption by scaling the supply voltage below the nominal value, and it achieves better performance by applying high frequency value, instead of the stable frequency range of the circuit. However, this will cause timing errors that happen when the task does not meet the maximum (critical) path delay of the circuit, which in turn, results in an output value error.

lowing equation where Re is the number of incorrect bits, and R is the total number of inputs.

$$\mathbf{ER} = \frac{\mathbf{R_e}}{\mathbf{R}} \tag{2.2}$$

In general, ER is important when the erroneous results need to be handled or recovered by a costly technique. Hence, an accurate estimation of ER is important in case where approximate circuits spend additional cycles of error corrections.

- The error distance (ED): is defined as the arithmetic distance (magnitude of deviation) between an inexact output and the correct output for a given input. ED is used for evaluating the quality of approximate adders [42], and expressed by the following equation where Re is the number of incorrect bits, and Rc is the number of correct bits.

$$\mathbf{ED} = |\mathbf{R_c} - \mathbf{R_e}| \tag{2.3}$$

Several error metrics can be derived from the formal definition of the error distance ED such as the mean error distance (MED), which studies the average error effect of multiple inputs [34]. In other words, MED reflects the average "closeness" of the approximate adder to the accurate adder and is used as common criteria. Another related metrics to ED is the relative error distance (RED) that equals to the ratio of ED over the exact output, as presented in Eq. 2.4, and the mean relative error distance (MRED) which is defined as the average of all RED values obtained from all possible input combinations. These metrics of RED and MRED are used in this thesis to demonstrate the effect of our proposed

approximate adder designs since they are considered as a significant indicator of outputs quality degradation.

$$\mathbf{RED} = \frac{|\mathbf{R_c} - \mathbf{R_e}|}{\mathbf{R_c}} \tag{2.4}$$

- The normalized error distance (NED) is the normalization of MED for multiple-bit adders. Remarkably, while MED measures multiple-bit adders' implementation accuracy, NED is independent of the size of the adder, and thus, analyses the general reliability of a specific design.

  Based on the definition of the error distance (ED) metric, error rate (ER) can be defined as the probability of inputs for which ED is greater than zero as described in Eq. 2.5.

$$\mathbf{ER} = \frac{\mathbf{1}}{\mathbf{n}} \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{P(ED(i) > 0)} \tag{2.5}$$

  However, if MED value is large, and ER value is small, then the design would generate large error magnitudes.

The following section presents more illustration about the adder importance, and makes a brief review of basic conventional adder designs.

2.6 BASIC ADDERS BACKGROUND

The addition operation is an essential part of any digital system where the fast and accurate operation of a digital system is greatly involved in existed adders' performance. Adders are also extensively used in other basic digital operations such as subtraction,

multiplication and division. Then, improving the efficiency of the digital adder module would significantly enhance the whole processes of binary operations inside a circuit compromised of such blocks.

The majority of approximate adder designs have exploited the idea of the low probability of the worst case of the carry propagation critical path. In other words, for an adder with bit width equals N, the probability of carry-chains having length=N is extremely low, and generally, carry signal can be determined by considering a number of bits to the right of the current bit position [91, 25]. Adders critical path reduction would result in decreasing hardware complexity and general power consumption,yet, with limited output quality degradation.

In the following part of this section, a short revision for three basic designs of the adder is presented.

2.6.1 *Ripple Carry Adder*

The ripple carry adder (RCA) in Fig. 2.2 is constructed by cascading full adders (FA) blocks in series. One full adder is responsible for the addition of two inputs binary digits at any stage of the ripple carry. The carry-out of one stage is directly inputted to the carry-in of the next stage. Major characteristics of RCA can be summarized as follows:

- Cascade N(1-bit) full adder.
- The delay grows in proportion to N or O(N), thus, has a long carry path (i.e., very slow for wide numbers).
- Each full adder requires waiting for the carry bit to be calculated from the previous adder.

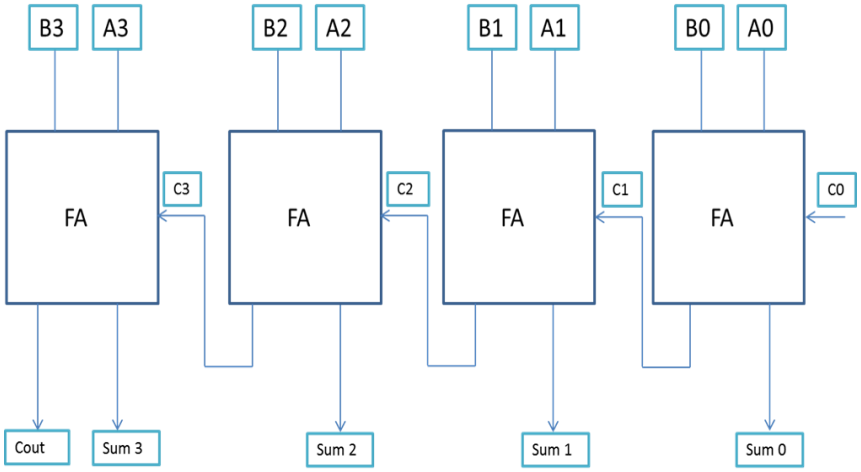


Figure 2.2: 4-Bits ripple carry adder.

- Shows a low power consumption due to a small area and the basic addition operation.

2.6.2 *Carry Select Adder*

In the carry select adder (CSL) shown in Fig. 2.3, the results of the addition operation are computed in parallel for the two alternatives: input carry '0' and '1'. These simultaneous additions take place in advance of receiving the actual carry value. Once the carry value becomes available, the correct computation is chosen (using a multiplexer (MUX)) to produce the desired output. Hence, mitigating the delay value of waiting for the carry-in to calculate the sum. However, although the fast speed and the high accuracy of the CSL adder, it shows considerable area requirements and an extensive level of computation (replicates the addition process).

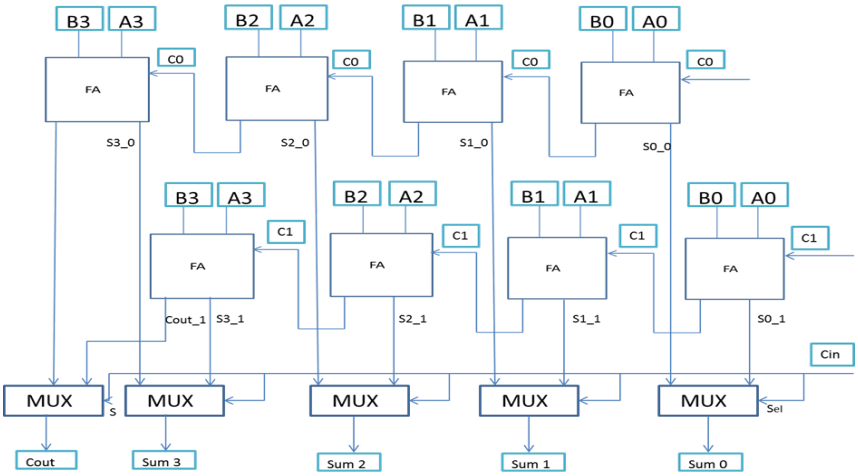


Figure 2.3: 4-Bits carry select adder.

2.6.3 *Carry Look Ahead Adder*

The carry look-ahead adder (CLA) in Fig. 2.4, solves carry propagation problem by calculating the carry signals in advance, based on the input signals. Further, the use of both generate (G(i)), and propagate (P(i)) signals takes place in this adder design.

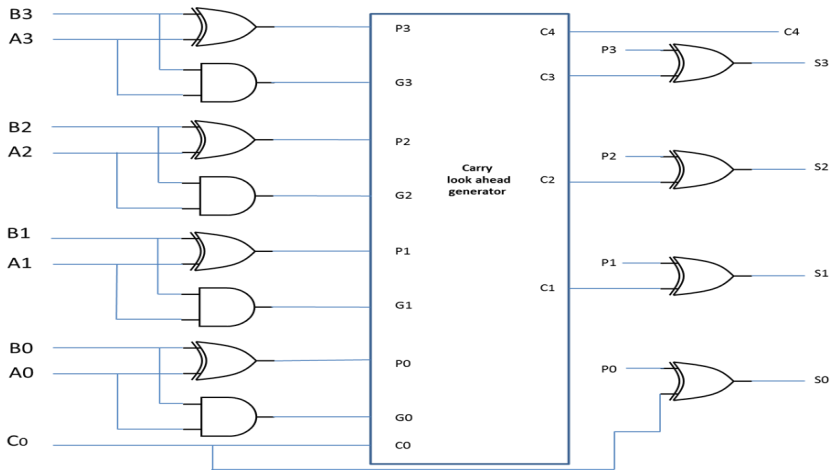


Figure 2.4: 4-Bits carry look-ahead adder.

The following points describe the main boolean expressions for generating sum and the new carry signals or propagating the previous carry value:

- Consists of (n) (sum, propagate, generate signals (SPGs)) which operate in parallel to produce the sum.
- $G_i = Ai \,.\, Bi$, $P_i = Ai \oplus Bi$, both signals are connected to the carry look-ahead (CLA) generator, which is illustrated in Fig. 2.5.
- $S_i = P_i \oplus C_{i-1}$, $C_{i+1} = G_i + P_i C_i$,where (Si) is the sum result and (Ci) is the carry value of adder (i). These equations show that a carry signal will be generated in two cases:
  1. If both bits Ai and Bi are '1',
  2. If either Ai or Bi is 1 and the carry-in Ci is '1'.
- For CLA generator depicted in Fig. 2.5
  1. All carries are generated directly, thus shorter carry path compared to RCA (i.e., high-speed computation).
  2. Anticipating the carry-in of every module according to a calculus of the carry-out from the previous (less significant) module.
  3. Require a large circuit.

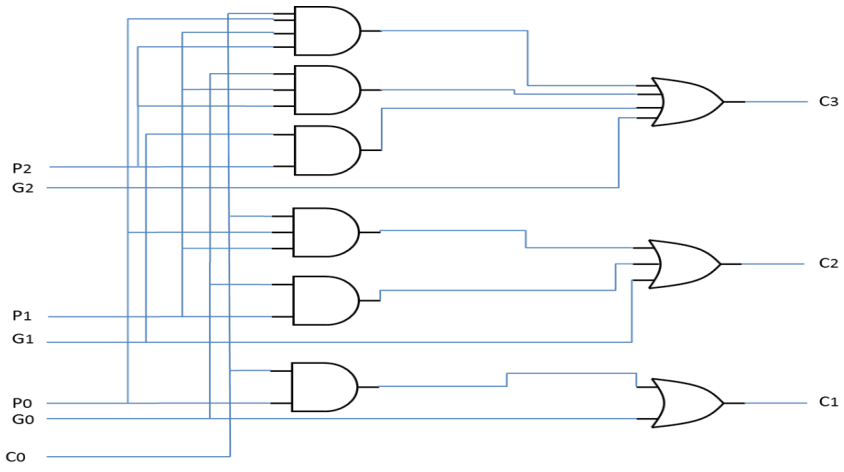


Figure 2.5: Carry look-ahead generator.

In summary, the comparison between the three types of adders shows that the RCA design has the simplest design, the smallest area and a lower power consumption, yet, has the slower computation speed. On the other hand, although both CSL and CLA adder designs have larger areas and complex circuits (hardware and computations), CSL adder designs shows high accuracy for outputs and CLA adder design behaves more efficiently in terms of propagation delay.

The following sections provide a review and examples of approximate and speculative adder designs that are related to our proposed work. They also present adders taxonomy table and summary points that illustrate the differences and challenges of the two design approaches.

2.7 APPROXIMATE ADDERS

Since adder module plays a key role in digital system efficiency, several efforts have targeted the approach of approximated adder designs. The general idea is to use an approximate computation technique to implement addition in the lower order bits (LSB) of the adder. This is due to the observed limited significance of lower part bits in adder's final result, and thus, preserving very low degradation of output quality.

In the following part, several related examples of approximated adder's designs are shown. These designs explain the utilizing of dividing the adder into an accurate and approximated "carry free" inaccurate parts. The approximation technique of inaccurate part is used to compensate the conventional exact addition operation, thus, more efficiency and a lower power consumption can be achieved.

2.7.1 *Lower-Part-OR Adder (LOA)*

Lower-Part-OR adder(LOA) [98, 50, 42, 35, 36, 47, 46] is considered a basic example of the approximated adder design. As depicted in Fig. 2.6, the adder structure of bit width equals (P) is divided into two parts, accurate part of a bit width of (m) and an inaccurate part of a bit width of (n), where (m+n=P).

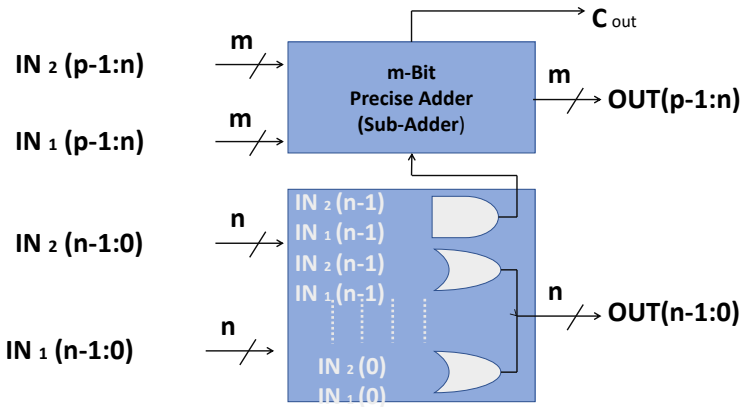


Figure 2.6: Lower-Part-OR adder(LOA) architecture.

In the approximated design, the accurate part has the higher order bits (MSB) and uses the exact addition operation to produce correct results. On the other hand, the lower inaccurate part (LSB) uses OR gates to perform bitwise OR operation to the corresponding input bits. Further, an AND gate is placed at the higher order bit location of the inaccurate part and utilised to generate carry-in value to the upper accurate part when the two input bits are both equal to '1' at that location. As a result, this would help to propagate a generated carry value of the lower part of the adder, and then, increase the total accuracy.

For the general evaluation of the LOA design, it can be summarised that it achieves a considerable increase of computation speed, in addition to reductions in area and power consumption.

However, these benefits are limited to the size selection of the bit width of the two parts of the adder, since a large inaccurate part bit width would achieve high performance and energy efficiency, yet, with a high percentage of output errors. On the other hand, if the accurate part has a larger bit width, the output accuracy would increase, but approximation gains would be decreased.

2.7.2 *Error Tolerant Adder (ETA)*

Error tolerant adder (ETA)) [106] design follows the same adder division strategy of accurate and inaccurate parts, yet, the bit width of the two parts are not necessarily equal. Fig. 2.7 illustrates the proposed approximate addition operations for each part of the adder.

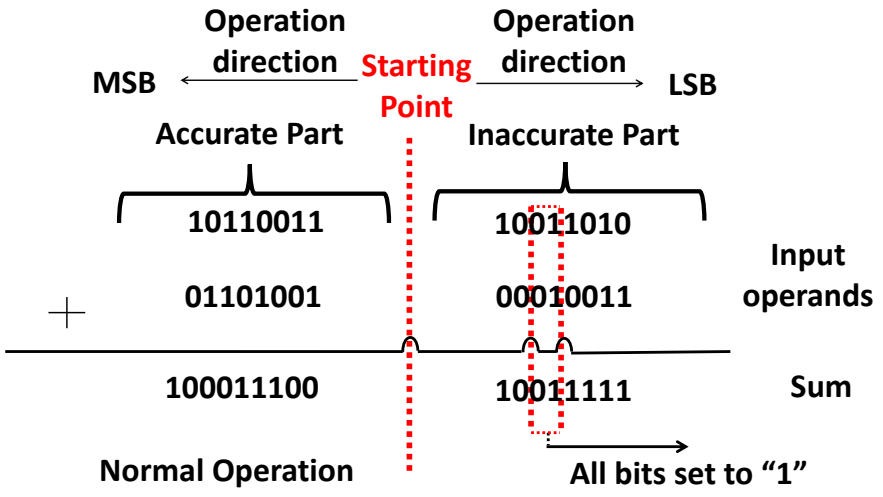


Figure 2.7: Error-tolerant adder (ETA).

The addition operation is started simultaneously from the middle of the adder and in opposite directions. The accurate part that has the higher order or most significant bits (MSB), performs the exact addition and results in accurate outputs. On the other hand,

for the inaccurate part, a specific approximate addition technique is applied on the lower order or least significant bits (LSB).

In detail, starting carry free (i.e., not generate or propagate of the carry) addition operation from the MSB to LSB bits of the inaccurate part, control signals would check the value of input bits at each location. In case the checked bits are both equal '0' or different (i.e., '0', '1'), the addition operation is performed by modified XOR gate. In the other case, when the checked two input bits to the same location both have the value of '1', the sum bit value of the corresponding modified XOR gate is set to 1, and further, all the remaining bits to the right of this bit location are set to '1' (by selecting the output nodes connected to VDD). As a result, this would increase the accuracy and limit the error of carry chain elimination in the inaccurate part of the adder.

In summary of this effort, the proposed design has achieved an overall delay reduction due to a simultaneous implementation of the addition operation in the opposite direction, and to the carry-free addition in the inaccurate part. In addition, introducing smaller area and a lower power consumption. However, a considerable level of output errors might be produced, especially for the case of small number additions, and further, the control signal overhead in the inaccurate part.

For addressing the challenge of error levels, ETA has been extended to several modified designs (ETAII, ETAIIM [104]and ETAIII [107]), in which the first interest was to improve predicting the carry-in to the higher order or most significant bits (MSB) due to their high efficiency in the output quality. In the following, we will explore the (ETA IV) [105] approximate adder design, which is considered the last version of ETA adder series.

2.7.3 *ETA IV*

The proposed ETAIV approximate adder [105] follows the concept of carry select adder (CSL) [102] in its implementation. An (N-bit) adder is divided into (N/X) blocks, where (X) equals the bit width of the sub-block (Sum Generator). However, the carry chain has been divided into two stages by the multiplexer (MUX2) unit (i.e., instead of dividing sum generator blocks in the original (CSL) adder adder described in section. 2.6.2).

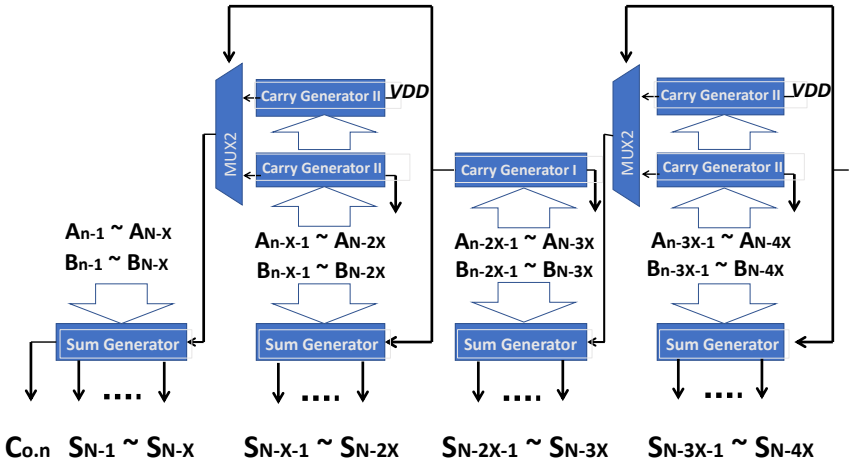


Figure 2.8: Block diagram of ETA type IV (ETAIV).

As illustrated in Fig. 2.8, two carry generators, type I and type II, are proposed to select the carry in input to the successive most significant sum generator. Each carry generator of type II calculates the carry out for the cases when inputted carry-in has the values '0' and '1'. Nevertheless, the accuracy of the carry-out of the carry generator would be limited to the bit width (X) value.

Based on the carry select concept, the carry-out of one of the two carry generators (type II) would be selected depending on the carry-out value from the previous stage carry generator (type I). In detail, when the carry-out of carry generator (type I) is '0', then the carry-out of Carry Generator (type II) that has inputted

carry-in equals '0' will be selected. On the contrary, if the carry-out of carry generator (type I) is '1', then the carry-out of carry generator (type II) that has inputted carry-in equals '1' will be selected. Consequently, the selected carry-out value is inputted to the successive most significant sum generator block. This leads that the carry predicting has been increased to (2X) bits, which would increase the accuracy of the result of the most significant sum generator block.

To analyse the design effort, although the improved accuracy level of outputs, the bit width value (X) might limit the gains of the approximation since a larger value of X would increase the accuracy, yet, with more delay and power dissipation and vice versa. Moreover, the general architecture based on carry select adder would introduce a large area overhead, therefore, limiting the approximation approach benefits.

2.8 SPECULATIVE ADDERS

The speculative adder design approach exploits the observation of rare occurrence of carry propagation worst case (critical) chains during addition operation. The general design architecture of speculative adder follows the approach of dividing N-bit adder into several smaller blocks (sub-adders), which are operating in parallel. Assuming that the bit width of sub-adder equals K, then the number of divided sub-adder is (M=N/K). The key idea in such designs is to speculate the carry-in value, which has to be propagated to a bit location, and then, approximate the correspondence bit sum result. Majority of speculative adder designs use K previous bits to predict the carry and then the sum value of bit location.

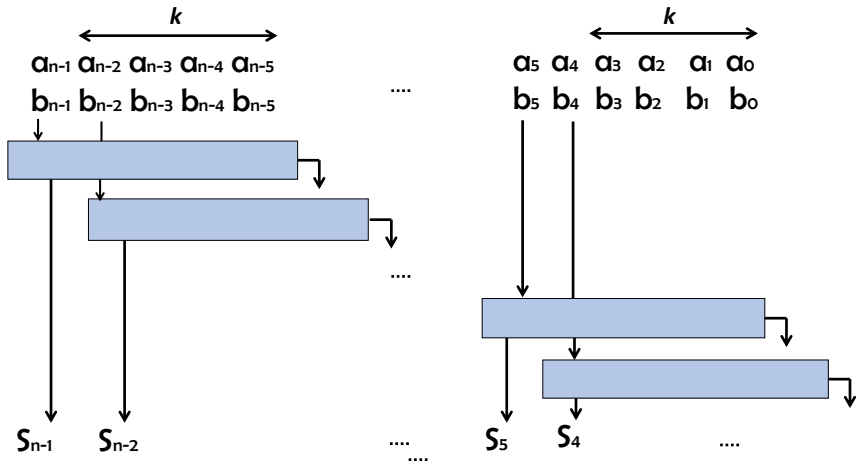


Figure 2.9: VLSA single-bit shift segmentation.

However, they might differ in the segmentation procedure and the accounted number of sum bits result in the output.

In the following, several related efforts of speculative adders are presented with a brief description and evaluation of each design.

2.8.1 *Variable Latency Speculative Adder (VLSA)*

The design of the variable latency speculative adder (VLSA) [91] in Fig. 2.9 proposed sub-adders segmentation by overlapping (K-1)) bits of previous sub adder in order to speculate the carry value in the current sub-adder. Each sub-adder contributes with one bit in the resultant sum (i.e., one resultant bit per sub-adder). As a result, a limited delay value to the bit width of the sub-adder is achieved (i.e., high speed). Further, the error levels become restricted, a matter which results in preserving a high accuracy level of the adder output. Nevertheless, due to using an intensive fixed single-bit shift operation, the number of sub-adders will increase with large fan-out of the input, and thus, increasing the area and power consumption overhead.

The adder design is extended by augmenting error detection and correction (EDC) circuitry, which consists of carry look-ahead (CLA) adder with the same bit width of segmented sub-adder. In case of error detection, the CLA adder will perform a precise addition to the erroneous sub-adder, while exploiting carry generate and propagate signals from the previous sub-adder. However, the process of the error recovery will consume another clock cycle to be performed.

As a summary, the VLSA design has shown more speed and high accuracy levels. However, it introduced a large area overhead due to a single-bit shift operation and the augmented EDC circuitry, in addition to the critical delay of the error recovery. This limitation of the design would decrease the total gain of the approximation effort.

2.8.2 *Accuracy-Configurable Approximate Adder (ACA)*

The design of accuracy-configurable adder (ACA) adder [37] proposed a speculative adder that mitigates the design challenges exist in previous efforts such as VLSA. Since the accuracy level was the first interest in ACA design, a middle sub-adder was introduced between every two basic sub-adders as depicted in Fig. 2.10. In detail, each sub-adder has a 2K number of bits, and after the first (least significant sub-adder), the successive sub-adder has overlapped half number of bits from the previous adder for the purpose of carry speculation to its upper part bits. Hence, half of the sub-adder length would result in the final sum as compared to the VLSA design that provides a single sum bit result for each sub adder. As a result, the number of sub-adders would decrease,

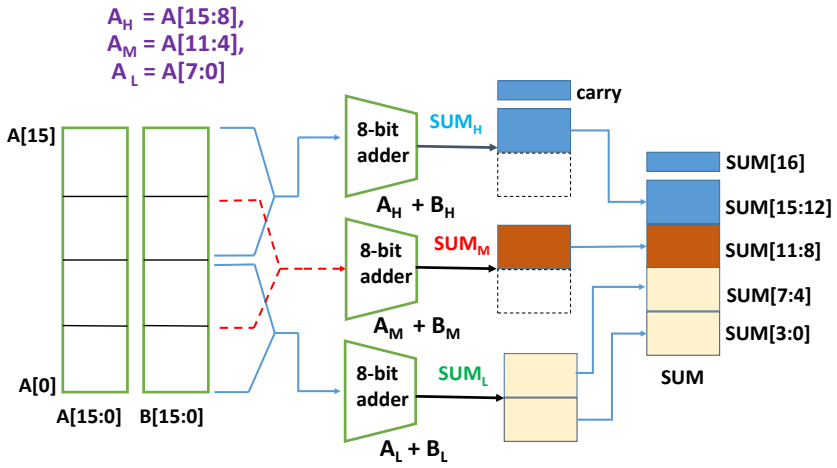


Figure 2.10: ACA adder example (16-bit adder), where H=High, M=Middle and L=Low.

and then, reducing the area and power consumption overhead. Fig. 2.10 shows a 16-bits example of ACA adder implementation.

For the error detection and correction (EDC) circuit depicted in Fig. 2.11, the proposed error detection uses simple logic gates such as AND gates, and is based on checking the following two conditions:

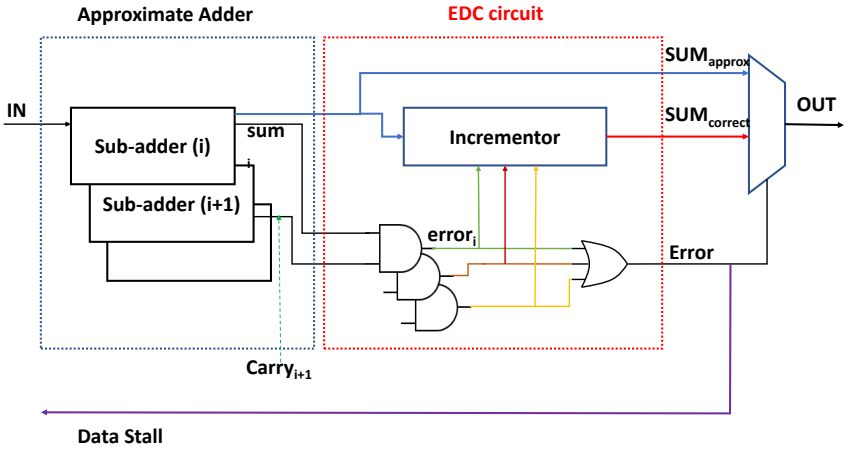


Figure 2.11: ACA error detection and correction circuit.

1. In the current sub-adder, all resulted sum bits of the overlapped part are equal to '1'.

2. In the previous sub-adder (i.e., from where the overlapped part comes from), the carry-in to overlapped bits is equal '1'.

In case that two conditions have been met, then this indicates that a carry propagation should take place to the current sub-adder. To overcome the detected erroneous sub-adder, an incrementor circuit is used to propagate the missed carry value of '1' to the resulted sum bit (i.e., not to the sub-adder itself), thus, preserving a high level of accuracy.

Another major advantage of ACA design is the multi-stage structure of the error recovery process of which each correction stage recovers a determined number of erroneous sub-adders' sum result using incrementors.

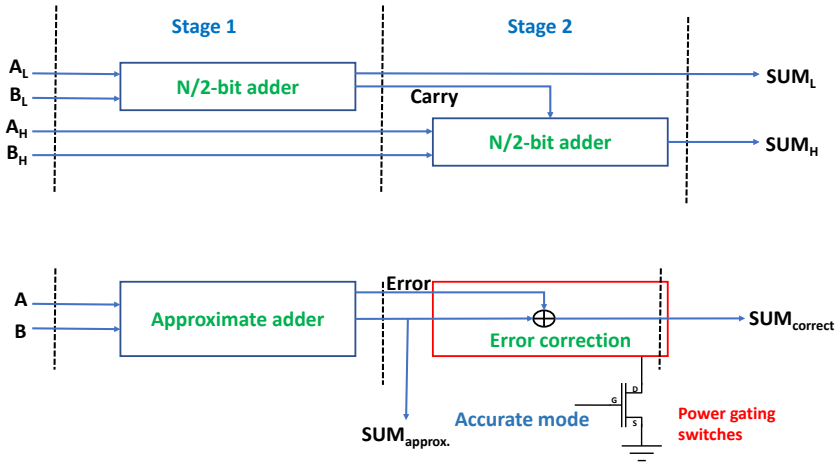


Figure 2.12: Two stages implementation of the approximate adder (below) with power gated correction stage, a conventional adder (above).

As illustrated in Fig. 2.12, the first stage has the approximate addition operation and the second stage consists of a correction stage (incrementor in the ACA design). It can be noticed that the activation of the correction stage is controlled by power gating the added footer transistor. In case an error recovery operation is

required, the transistor would power on the correction circuit to implement the correction process.

Furthermore, this two-stage structure proposed to be extended to multi-correction stages. The new correction stages would be organised in the pipelined architecture where the first correction stage would recover errors in the least significant sub-adders and the last correction stage recovers the error in the most significant sub-adder sum. The activation of these correction stages is controlled by a power gating technique by using one transistor for each stage. This added transistor would control the activation of the correction stage based on the required level of accuracy, thus, the error correction process becomes configurable during run-time. The worst case of the accuracy level can be fulfilled by operating all the correction stages.

Since we consider the ACA design [37] as a significant example of accuracy configurable approximate adders, the proposed adder designs of this work in chapters 3, 4 and 5 have been compared to its design parameters such as delay, area and power consumptions, in addition to the related output error values.

2.8.3 *General Architectural Design of Accuracy-Configurable Adders (GeAr)*

The general architectural design of accuracy-configurable adders (GeAr) [80] is based on the previous accuracy-configurable adder (ACA) design [37] with equal bit-width sub-adders. However, this design proposed a generalised model and architecture for the accuracy-configurable adders.

The main idea of generalisation is to exploit the chance of multiple ranges of a number of overlapped bits, which are used for

carry speculation, and the number of sum resulted bits from each sub-adder. As a result, different architectures and configurations of adder design can be available, thus, introducing multiple energy-accuracy trade-offs.

In detail, for a sub-adder, if the number of the overlapped bits (P) is larger than resulted bits (R), the accuracy will increase, yet, with more delay. Conversely, if the (R) bits have a larger number than propagate bits (P), the speed will increase with the reduction of accuracy. Figs. 2.13 and 2.14 below show two different implementations of 12-bits GeAr adder design regarding the defined values of result bits (R) and propagate bits (P) and the number of sub-adders (M).

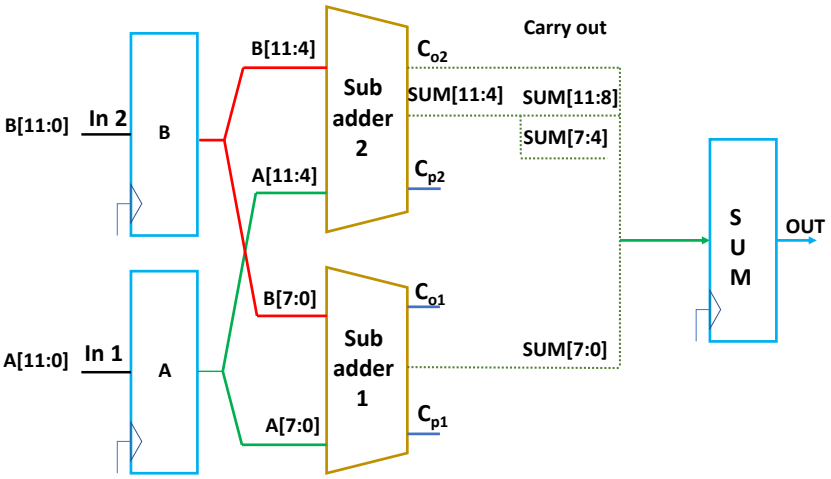


Figure 2.13: GeAr adder with N=12, R=4, P=4 and M=2.

On the other hand, regarding the configurable correction scheme which is depicted in Fig. 2.15, the incrementor in ACA design is replaced by a new approach of error detection and correction. The proposed correction technique includes the following points:

1. The error detection is implemented with an AND gate (i.e., ANDING Cpi and Co(i-1)).

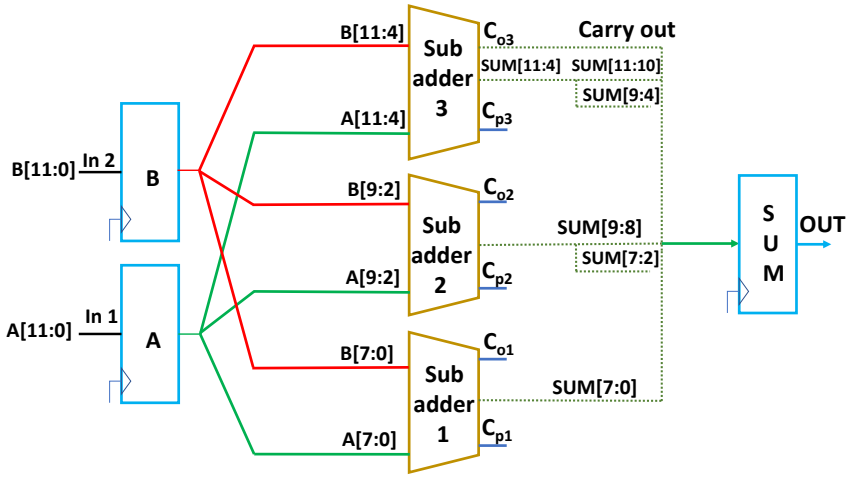


Figure 2.14: GeAr adder with N=12, R=2, P=6 and M=3.

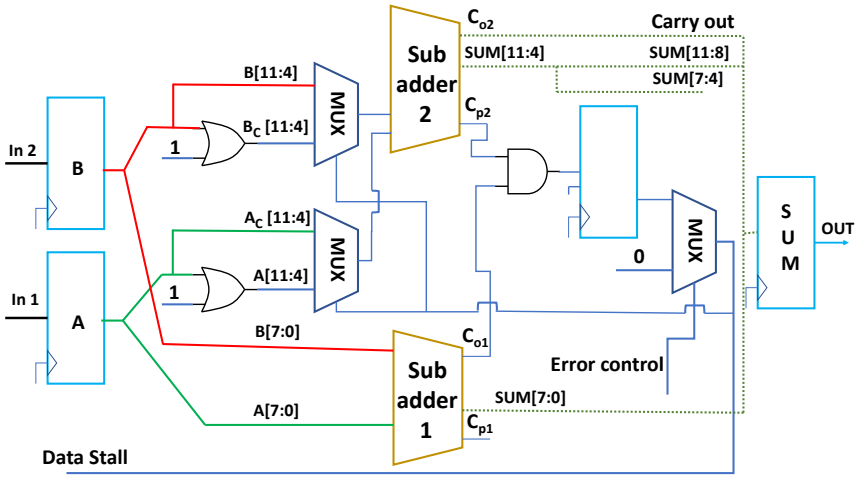


Figure 2.15: Error detection and correction for GeAr adder with N=12, R=4, P=4 and k=2.

2. In case Cpi and Co(i-1) both are '1', then the accuracy shall be compromised, and this confirms that a carry with a value of '1' should be propagated from the previous sub-adder, otherwise, an error is detected.

3. For error correction, both inputs to sub-adder are passed through an OR gate, and their LSBs are set to '1', and hence, while activating the correction process, this will generate the required carry to recover the erroneous sum.

4. Further, a control signal is used to select the sub-adder that needs error detection and recovery. Consequently, providing a higher level of the architectural support for configurable error correction to avoid correction overhead, based on application accuracy specifications, yet, with a larger area challenge.

As a summary, Table. 2.2 presents taxonomies of the three types of approximate adders with their features and design limitations. Additionally, a detailed evaluation of both approximate and speculative adders is provided in the following section.

2.9 COMPARISON AND CHALLENGES

As a general evaluation for both approximate and speculative adders, it can be noticed that the main difference between speculation and approximation depends on the error rate concern. Speculative adders' principle depends on the very low probability of error occurrence, which results in more efficiency in terms of high speed and low activation of the error recovery circuit (EDC). Conversely, approximated adder depends on the significance of an error in the resulted output quality, thus, presents the ability for

Table 2.2: Taxonomies of approximate adder based on error recovery circuit implementation.

Category	Features	Limitations
Designs without error recovery circuits. LOA [50], ETA [106]	High speed, small area, and low power consumption	1- Accuracy depends on the number of bits in the full accurate part of the adder design. 2- High error rate for the small numbers additions which are processed at the approximated part of the adder design. 3- New architectures for improving the accuracy will incur a substantial design overhead regarding area and power consumption.
Designs with an error recovery circuit and a fixed level of accuracy. VLSA [91]	1- High speed due to the segmented short carry chain and parallel addition operations mainly in the case of no augmented error recovery circuit. 2- Better output accuracy when compared to the designs in the previous category due to the quick carry-in speculation technique at each bit location in the adder.	1- EDC circuit would incur considerable design overhead of area and power consumption. 2- A fixed accuracy level at the design time would make the whole approximation process less beneficial. 3- Frequent erroneous sum detection during addition operation would result in more latency.
Designs with an error recovery circuit and a configurable level of accuracy. ACA [37]	1- High speed due to the segmented short carry chain and parallel addition operations. 2- The error recovery is divided into multiple stages and controlled during the addition operation run-time.	1- The segmentation and speculation techniques of the adder design still result in a larger design area when compared to the conventional adder. 2- At the worst case of error recovery (i.e., full accuracy of output), considerable delay and power consumption levels would be introduced. 3- The activation of all error recovery stages does not guarantee the full accuracy of the final output. This due to the fact that a few carry propagations might be not fully detected during error recovery.

accepting high rates of error in its lower order or least significant (LSB) part, yet, with low magnitude values. Therefore, the research efforts are focusing on minimizing the error amplitude in a way to conserve highly accepted quality for the outputs. The following points will summarise the major characteristics, approximation gains and challenges of both adder design approaches.

- The bit width of sub-adders of the speculative adder and the inaccurate part of the approximate adder plays a key role in trading off the approximation efficiency and the output quality.
- Approximate adder designs do not have an error recovery circuit; However, several techniques might be used to limit the error significance (i.e. magnitude) in the final output, yet, with increased design overhead.
- For the speculative adders, the overlapping speculation technique and the number of resulted sum bits of each sub-adder, both determine the number of sub-adder blocks, and thus, affecting the gains of area and power saving.
- Error detection and correction circuitry (EDC) in speculative adders results in more area and delay overhead, which limits the benefits of approximation.

2.10 IMAGE PROCESSING

Image processing is defined as a technique of converting an image to a computational form in order to perform several operations on it in a digital system. As shown in Fig. 2.16, The image in mathematical form is regarded as an array or matrices of squared picture elements known as Pixels, which are organized in adjacent

columns. Pixels "Picture Elements" are the basic building blocks of a digital image or display and are created using geometric coordinates.

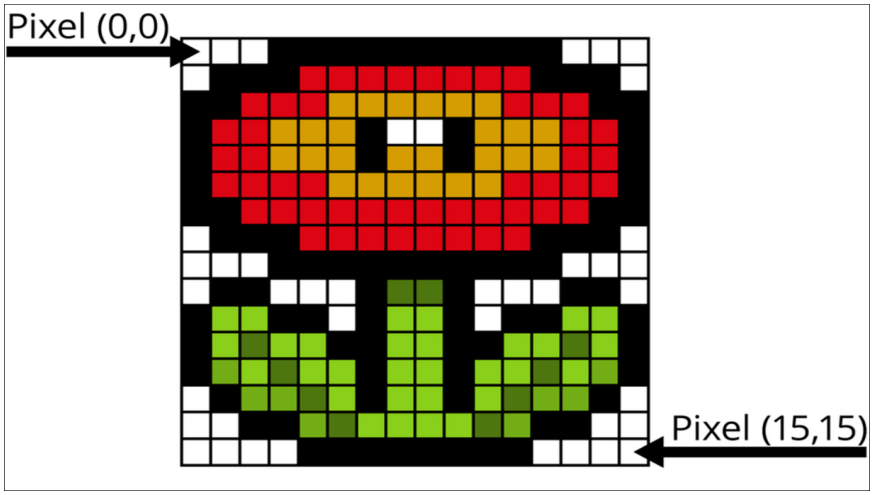


Figure 2.16: (15X15) Matrix of image-array of pixels.

Operations on the digitally captured image (e.g., photograph or video frame) might include enhancing the original image by a kind of filtering such as sharpening and smoothing filters or using the characteristic of the image in some processing operations [38].

Generally, images are considered two-dimensional (i.e., including rows and columns of pixels) signals, or a function of two variables (e.g., F (x,y)) with the amplitude value of the image at the real coordinate position (x,y). Thus, different signal processing methods can be applied to them in the image processing system.

Image processing plays an integral role in most of growing multimedia applications and related technologies today. It is considered a core part of many aspects of business like social and medical applications, in addition to high-speed communications. Image processing presents a major and wide research area within engineering and computer science disciplines for more optimization and computation efficiency.

The following parts present a brief background of the major steps, general purposes and techniques of the image processing implementation.

2.10.1 *Image Processing Steps*

Basically, the image processing implementation follows the next three steps [38].

1. Capturing and importing the image to the digital system by means of input devices such as a scanner or digital camera.

2. Performing the computational manipulation on the imported image, which might include image processing techniques such as enhancement or compression, or analysing image specific patterns and characteristics.

3. The last step presents the output either as a new altered image or a report of the analysed data of the image.

2.10.2 *Image Processing Motivations*

General purposes of image processing can be classified into five groups.

1. Visualization: in which the not-easily visible objects can be observed.

2. Image restoration and enhancing: in which the amendments on the original image can produce a better image quality.

3. Image retrieval: that allows the application's user to search for the image of interest.

4. Measurement of the pattern: which allows making measurements on multiple characteristics or objects in the image.

5. Image Recognition: that helps to distinguish a specific object in an image.

2.10.3 *Image Processing Techniques*

Once the image has been captured, a number of basic computational operations are implemented in advance to convert the image into a digital (i.e., bit information) form, and then, becomes available for further image processing.

Two basic operations in the image processing system are the digitization operation that includes a sampling of the image by selecting subsets of inputs, based on determined algorithms, and the quantization operation which gives amplitude values of each image sample. Consequently, after finishing these two steps, the image would be converted to a bits form and the processing can be performed [38, 97, 41].

Among widely known computation blocks in image processing, the Gaussian blur filter presents a remarkable example. Therefore, it has been considered in this work for analysing the implementation of the proposed adder designs in the image processing application.

2.10.4 *Gaussian Blur Image Filter*

In image processing, a common filter known as Gaussian blur (smoothing) filter is used to reduce image noise and details. The process of blurring an image is resulted from convolving each

pixel in the image with Gaussian function. Blurring is a widely used effect in graphics software, and specifically, with algorithms that are sensitive to noise such as edge-detection algorithms, and thus, improving the result of the algorithm. Further, it is generally considered a pre-processing stage in computer vision algorithms for enhancing image structures at different scales [20].

In general, Gaussian blur filter can be classified as a low pass filter and would result in a reduction effect of image's high-frequency components and signals.

The Gaussian blur filter uses a Gaussian function (which also expresses the normal distribution in statistics), which calculates the transformation that would be applied to each pixel in the image. The following equation presents the mathematical of a Gaussian function in one dimension [20].

$$\mathbf{G}(x) = \frac{\mathbf{1}}{\sqrt{2\pi\sigma^2}} e^{-\frac{\mathbf{x}^2}{2\sigma^2}}, \tag{2.6}$$

While in two dimensions, the Gaussian function equals the product of two such Gaussians, one in each dimension:

$$\mathbf{G}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{1}}{2\pi\sigma^2} e^{-\frac{\mathbf{x}^2+\mathbf{y}^2}{2\sigma^2}}, \tag{2.7}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and (σ) is the standard deviation of the Gaussian distribution.

This equation gives a way to calculate coefficients for a Gaussian template that is then convolved with an image. Convolution operation would result in Gaussian averaging, which means that the point in the averaged picture is calculated from the sum of

the region where the central parts of the picture are weighted to contribute to more than the peripheral points. Standard deviation is essentially the average distance from the mean of all points and a small standard deviation will have a tight bell curve, which is essentially prioritizing the weights in the middle.

The result of the two dimensions equation would result in a convolution matrix to be applied to the original image. The values of this matrix have a Gaussian distribution from the centre point. Hence, each pixel in the image must be included in the calculations and then get a new value (non-zero). Each new pixel value is set to a weighted average of the neighbourhood pixel. In such values distribution, the original pixel's value receives the heaviest weight (i.e., highest Gaussian value), and as the distance from the original pixel increases, neighbouring pixels receive smaller Gaussian values. As a result of this blur, better preservation of image boundaries and edges would happen.

The following Figs. 2.17and 2.18 present a simple comparison between mean averaging and blur averaging methodologies.

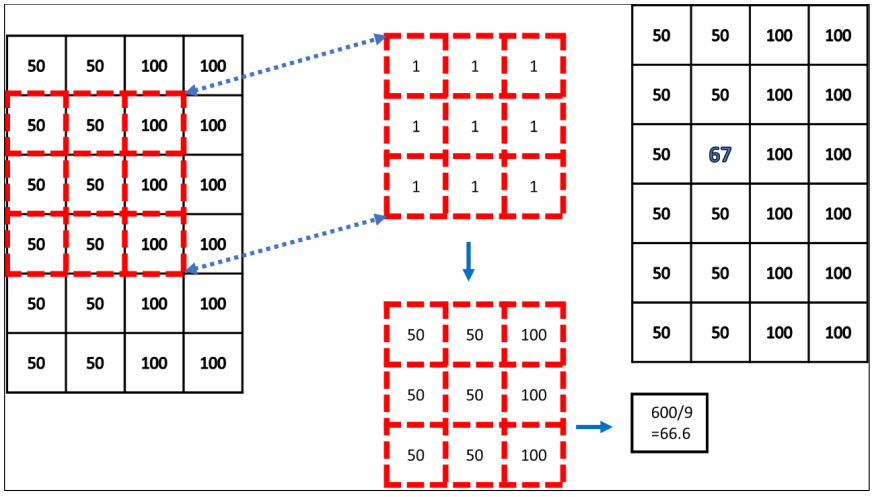


Figure 2.17: Mean averaging of image values matrix.

From Fig. 2.17, it can be noticed that the mean averaging operation uses unity kernel matrix (i.e., has values of ones) in order to

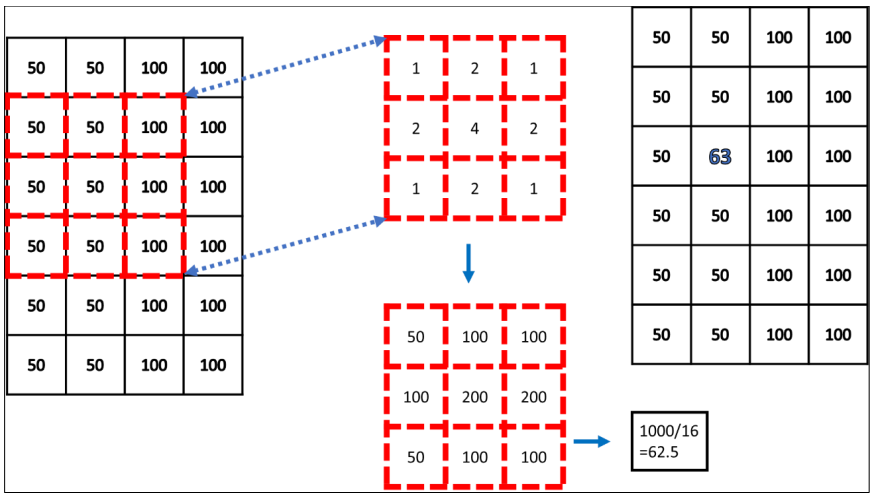


Figure 2.18: Blur averaging of image values matrix.

calculate the direct average of neighboured values in the matrix of the original image. This centre location average value would be the new value in the resulted image matrix. On the other hand, blur averaging operation in Fig. 2.18 uses different values in its kernel matrix, which results in new better values of the smoothed image matrix.

As a summary, Gaussian blur filter is used to produce a smoothed image, and thus, a better quality for analysis operations. Further, it has the characteristic of the low pass filter that limits high-frequency data in the new blurred image. However, this would require an intensive level of computations.

3

SCALABLE LOW-POWER AND CONFIGURABLE-ACCURACY APPROXIMATE ADDER DESIGN

3.1 INTRODUCTION

The previous chapter explores the approximate computing principle and the targeted design levels of approximation by research efforts. It has shown that the main idea of approximate computing is about enhancing the design parameters such as execution speed, area, and power consumption, meanwhile allowing computing errors to occur within acceptable frequency and magnitudes.

In this chapter, an approximate (speculative) adder design is introduced. The proposed design is grounded on the idea of the accuracy-configurable adder design in ACA [37]. This design has been considered because it was investigated by other research efforts such as (gracefully-degrading accuracy-configurable adder (GDA) [100], GeAr [80], Accurus [8]), nevertheless, with the common challenge of additional design overhead regarding delay, power, and area.

As presented in chapter 2, the general adder architecture of ACA design is based on dividing the adder of length N into several sub-adders equals N/2K, where k is the half-length of each sub-adder. The primary design point in the ACA adder design is using an overlapping number of bits equals (k) from the previous sub-adder to be used as the low order part of the current sub-adder. The main

two interests of the design are to break the carry chain for more execution speed and to increase the accuracy of the speculated carry-in value to the remaining higher order k bits within the sub-adder. As a result, the higher order k bits in each sub-adder would be extracted to the final sum result with highly acceptable accuracy.

The error detection (i.e., carry propagation) technique in ACA design needs to check the sum result of the overlapped k bits in the current sub-adder and to check the carry-in value to these k bits in the previous sub-adder if all equals to logic one. If the two checks flagged high, this would confirm an erroneous sum result of the current sub-adder which should be corrected. As a result, the challenge of the large design area overhead is existing and referred to the large number of logic gates used for error detection between every two successive sub-adders. Additionally, although the correction process was implemented with controlled stages, it did not consider the significance (i.e., impact) of the corrected sub-adders sum in the final full-length sum result (i.e., N bits result). This is because it starts correction from the lower order sub-adder with the overlapped k bits part.

The contribution in this chapter[1] introduces two main modifications to the baseline design of the ACA design in order to mitigate the mentioned challenges of design area overhead and low significance of premier correction stages. This effort proposes a new lightweight error detection technique, which results in lower design overhead and more scalability for larger bit-width adder designs, yet, without any accuracy scarifying. For correction stages, a

1 This effort has been published in IEEE Xplore as, Khaled Al-Maaitah; Issa Qiqieh; Ahmed Soltan; Alex Yakovlev, Configurable-accuracy approximate adder design with lightweight fast convergence error recovery circuit, IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), 2017, pp 1-6.

significance-driven structure has been proposed to ensure correcting errors with high magnitude early and achieve fast convergence to the exact adder outputs. Moreover, it demonstrates an extended version of the proposed design, which guarantees 100% accuracy at the final correction stage.

Compared to other equivalent approximate adders, the proposed design has drastically reduced the logic counts used for error detection process, thus, achieving lower overhead of the silicon area. In addition to improving the energy efficiency of the adder design with faster convergence to the exact results.

A number of different bit-widths of the proposed adder (32-bit to 256-bit) are designed in Verilog and synthesized using Synopsys design compiler. Our post-synthesis experiments showed significant reductions of 12% and 10% for dynamic and leakage power respectively, and 8% in the silicon area for the design with full correction stages. Remarkably, the proposed adder with large bit-widths has reserved these reduction ratios while presenting better scalability overhead. For design implementation, the proposed adder design has been applied in an image processing application (Gaussian blur filter), which resulted in high PSNR output values of (29 and 42 dB) for the two premier correction stages, and the optimum image quality while using the full correction stages [3].

3.2 GENERAL DESIGN ARCHITECTURE

The proposed approximate adder exploits the idea that the critical (worst case) carry propagation through the whole adder width would take place on rare occasions (i.e. not often happened). Hence, the proposed approximate adder is divided into several sub-adders by cutting the length of the critical carry path. On the other

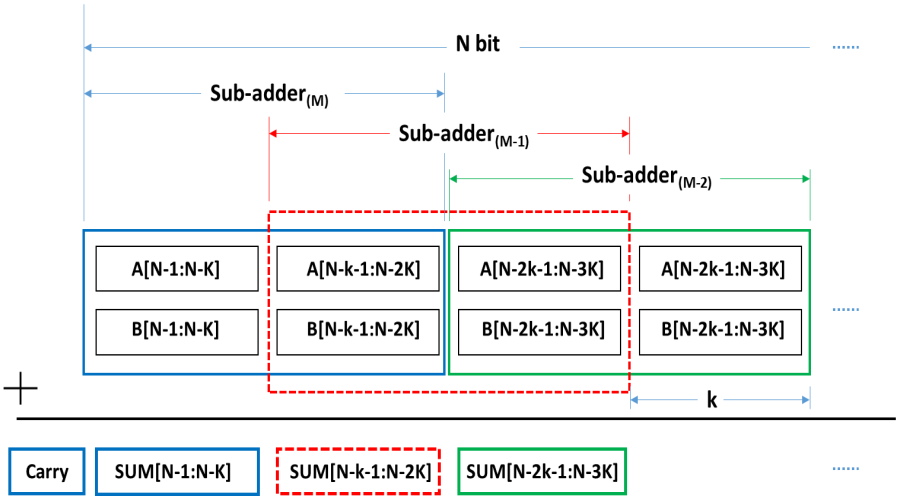


Figure 3.1: General implementation of the proposed approximate adder.

hand, redundant parts of the addend inputs are overlapped at successive sub-adders. These redundant parts are used to predict a carry-in value within the new sub-adders. This would lead to a high probability of carry value handling, and then, increasing the outputs accuracy significantly. This general approach of the proposed approximate adder might be considered similar to the efforts in [80], [8] and [37]. However, the proposed design in this work would introduce better design overhead, scalability and high levels of accuracy.

Fig. 3.1 shows the general implementation of the proposed approximate adder design, which has the same general architecture in ACA design, however, the proposed design has been augmented with a different lightweight error detection technique. The following points summarize the design main parts.

- 2K is the bit-width of the sub-adder.
- K equals the half-length of the carry-chain in the sub-adder.
- Non-overlapped K-bits at each sub-adder would result in final output sum.

- The first sub-adder produces a 2k-bit result.
- The approximate adder consists of M= ((N/K)-1)sub-modules, where N equals to the total bits number of the main adder.

3.3 ERROR DETECTION CIRCUIT

The second part of the proposed design is the error detection and correction (EDC) circuit. This augmented circuit gives the proposed adder design the ability to work gracefully in the approximate and accurate modes during run-time. In the proposed design, the error will signal high when the overlapped (redundant) part of the sub-adder failed to handle a correct carry value which should be propagated from the previous sub-adder. For error detection between two sub-adders, the proposed lightweight error detection technique uses an XOR gate as shown in Fig. 3.2 to check the equality of the carry-out of the previous sub-adder and the carry-out of the redundant part of the current sub-adder. Hence, when an error is detected; the approximated sum value has an error and needs to be corrected by adding '1' to its current value.

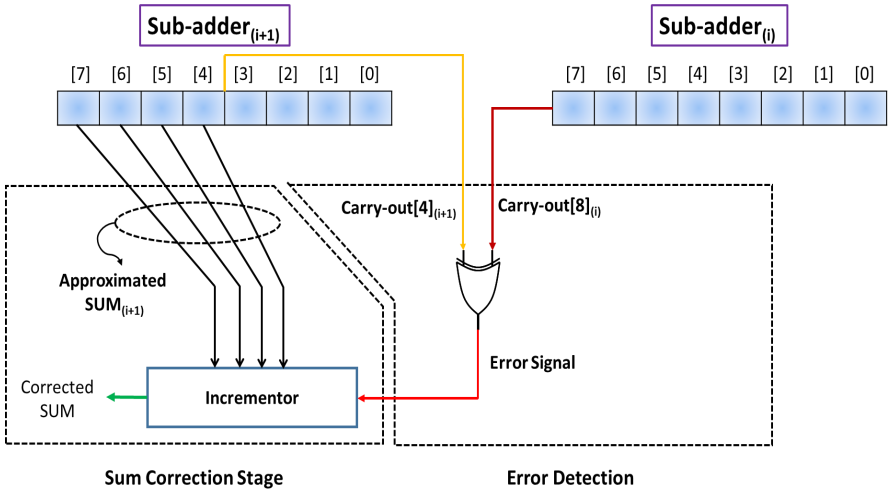


Figure 3.2: Proposed error detection technique.

This correction process guarantees the correct carry value propagation and can be accomplished by an incrementor circuit. The proposed technique decreases the logic complexity used for error detection between every two sub-adders when compared to the previous effort of ACA [37] that used more logic to check values of a larger number of bits (sum bits of the redundant (overlapped) part of the current sub-adder, and the carry-in to this part in the previous sub-adder).

Although the proposed error detection technique incurs a limited degradation of delay, significant enhancements of power consumption and area happened. Moreover, this technique is considered more beneficial in terms of scalability of larger adder widths as will be shown in section 3.8 of this chapter.

3.4 ERROR CORRECTION CIRCUIT

The correction operation basically uses an incrementor circuit that increments the erroneous sum value by one. The bit width of the incrementor is equal to the width of the resulted approximated sum bits of each sub-adder.

The structure of the correction circuit is applied through several correction stages. The multi-stage structure would give the ability to control the active correction stage independently while changing the accuracy level during the run-time. This would achieve more power saving for low-level accuracy modes [37].

Fig. 3.3 illustrates the proposed significance-driven error correction structure. This structure would guarantee that the resulted sum from the most significant sub-adders will be corrected first, and the final correction stage will correct the resulted sum from the least significant sub-adders. As a result, the correction of high mag-

nitude errors and fast convergence to the exact sum value will take place at the premier correction stages. This significance-driven correction scheme might be analogous to [8]; However, there was no remarkable modification for the error detection mechanism, in opposite to our proposed design that incurs lower design overhead.

On the other hand, in order to guarantee 100% accuracy of outputs at the final correction stage (highest level of accuracy), the carry-out values of the active correction stages should not be overlooked. Conversely, these values have to be considered and propagated to the successive most significant correction stages. Hence, the proposed design has introduced a new extended version, which checks the correction stages carry-out values and their propagation in a way that achieves full accuracy at the final correction stage. The extended full accurate version of the proposed design is denoted by (Proposed_Accurate).

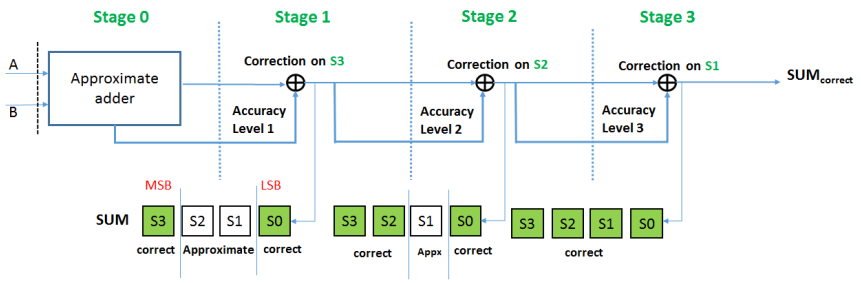


Figure 3.3: Significance-driven error correction stages (32-bit adder example).

From Fig. 3.3, the main points of multi correction stages can be summarized as follows.

1. Stage (0) has the approximated sum result (without any correction).

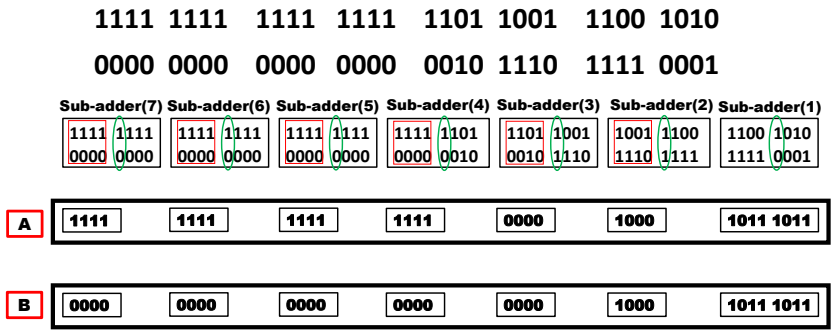
2. The correction stage (incrementor) will result in the accurate sum part (coloured green).

3. S0 is always correct as it uses carry-in = ’0’.

For more clarification for how the proposed design works, a numerical example is provided in the following section.

3.5 NUMERICAL EXAMPLE

In this section, a (32-bit) binary numerical addition example is presented. The detailed example in Fig. 3.4 shows the main methodology of adder division, approximate addition and error detection and correction techniques of the proposed design versions (Proposed design and Proposed_Accurate) versus ACA design.



A= Approximated final sum result of Proposed and ACA designs. Approximated Final Carry equals to ‘0’.

B= Exact (correct) final sum result. Exact Final Carry equals to ‘1’.

Figure 3.4: Proposed design numerical (32-bit) approximate addition example.

Fig. 3.4 presents an example of (32-bit) addition of the proposed design. The main adder is divided into seven sub-adders of (8-bit) width for each. Sub-adders (except the first one) has two parts, in which the first lower order part has a redundant 4 bits from the previous sub-adder, and the most significant part has new added 4 bits, which are used to result in the approximated sum.

In the proposed architecture, the carry-in to each sub-adder is truncated to ’0’, while the redundant bits at each sub-adder are

used to speculate the carry-in to the most significant part of the sub-adder, thus, approximate the sum results with high accuracy.

In Fig. 3.4, part A presents the approximated sum of the proposed design, while part B shows the exact (correct) sum that accomplished by a conventional adder. The two results A and B show the difference between the approximated and exact results. It can be noticed that a carry at A had to be propagated after the fourth sub-adder to the final carry-out, and hence, the exact value of carry-out should be '1' instead of approximated carry-out of '0'.

Table 3.1 shows the methodology of error detection of the eight sub-adders of the proposed design versions (Proposed design and Proposed_Accurate) and ACA design. Error detection simply implies checking (i.e., active '1') if there is a carry-out value of '1' from the previous sub-adder, which should be propagated to the current sub-adder. As a result, if an error is detected, the approximated 4-bit sum value should be incremented by '1' to match the correct value.

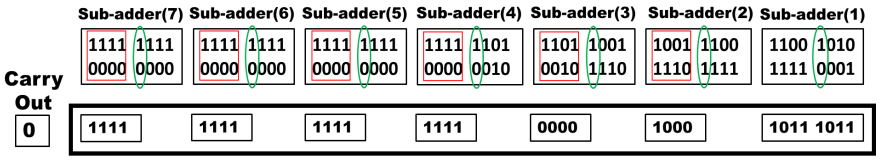
Table 3.1: Proposed design versions vs. ACA error detection.

Proposed Accurate Check	**Proposed Design Check**	**ACA Check**	**Carry out**	**Redundant =1111**	**Carry [3]**
-	**-**	**-**	**1**	**-**	**0**
0	**0**	**0**	**1**	**No**	**1**
0	**0**	**0**	**1**	**No**	**1**
1	**1**	**1**	**0**	**Yes**	**0**
1	**0**	**0**	**0**	**Yes**	**0**
1	**0**	**0**	**0**	**Yes**	**0**
1	**0**	**0**	**0**	**Yes**	**0**

The proposed design in ACA [37] detects an error when the redundant 4-bit sum is equal to (1111), and the carry-out of the fourth bit of the previous sub-adder is equal to '1'. Conversely, in

this work, the proposed design error detection concerns about just checking the equality of two bits locations (the carry-out of the previous sub-adder and the carry-out of the fourth bit of the current sub-adder), then, the equality check will signal high if there is any detected error (i.e., carry propagation).

Moreover, the extended full accurate version of the proposed design (denoted as Proposed_Accurate) introduces more accuracy through detecting the carry-out of the active correction stage to be propagated to the sum result of successive correction stages.



Proposed_Accurate Active stage	Proposed Design Active stage	ACA Active stage	Error	Sub-adder
0	0	0	No	1
0	0	0	No	2
0	0	0	No	3
2	2	2	Yes	4
2	0	0	Yes	5
1	0	0	Yes	6
1	0	0	Yes	7

Figure 3.5: Proposed design versions vs. ACA active error correction stages.

Fig. 3.5 presents a comparison between the proposed design versions and ACA design regarding the active correction stages while detecting an error. For the ACA design error recovery technique, an error is detected at sub-adder number four. Based on the ACA correction stages structure, which starts correcting the errors of the least significant sum results, stage number 2 will be active to correct the error. The proposed design version detects the same error at the fourth sub-adder, and regarding the significance-driven architecture in Fig. 3.3, correction stage number 2 will be active to correct the error.

On the other hand, for the Proposed_Accurate design version, all the errors of carry propagation starting from the fourth sub-adder to the final seventh sub-adder will be detected. The Proposed_Accurate design version has the same significance-driven structure of correction stages, thus, for the errors at the fourth and fifth sub-adders, correction stage number 2 will be active, and for the errors at the sixth and seventh sub-adders, correction stage number 1 will be active. As a result, all the errors will be corrected with 100% accuracy of the final sum result.

In the following sections, we present an evaluation of design metrics improvements and the error analysis of the proposed design when compared to ACA design [37].

3.6 DESIGN TRADE-OFFS

To demonstrate the proposed approach, Verilog was used to apply different sizes of adders ranging from (32-bit to 256-bit). However, our main comparison for approximate and all correction stages used 32-bit adder with half carry chain (K) equals to (4-bit) for each sub-adder (i.e., 2K = 8 bits). These codes were synthesized and implemented using two different off-the-shelf tools; Firstly, Modelsim was used for compiling the Verilog codes and running the associated test benches of functionality and error analysis. Secondly, the Synopsys design compiler was utilized for synthesizing all sizes of the proposed adder versions when mapping the circuits to the UMC (Faraday 90nm) technology and evaluating for power, delay and area.

In order to make hardware evaluation, the proposed design was compared to the design effort in ACA [37]. The proposed design has two versions, where the first version was applied without consid-

ering the carry-out of correction stages, and the second version considers the carry-out of each active correction stage. For simplifying, the design version considering the correction stage carry-out is denoted by (Proposed_Accurate).

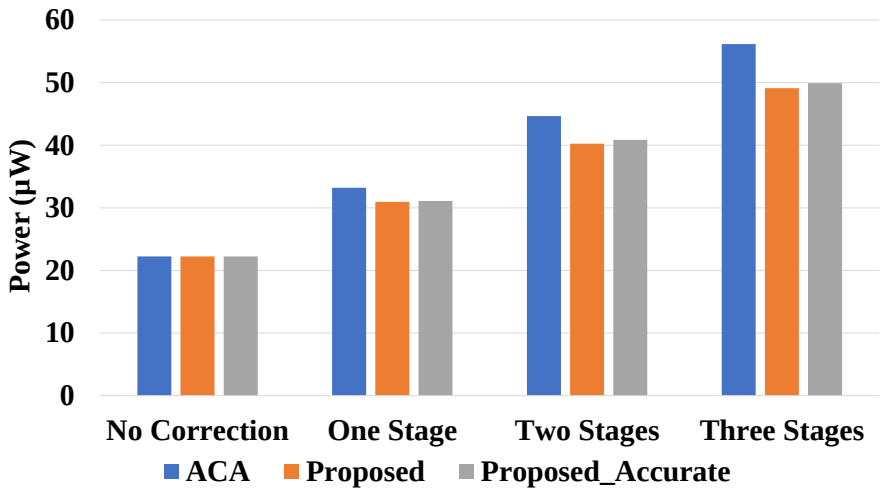


Figure 3.6: 32-Bit proposed design versions vs. ACA dynamic power (μW) comparison.

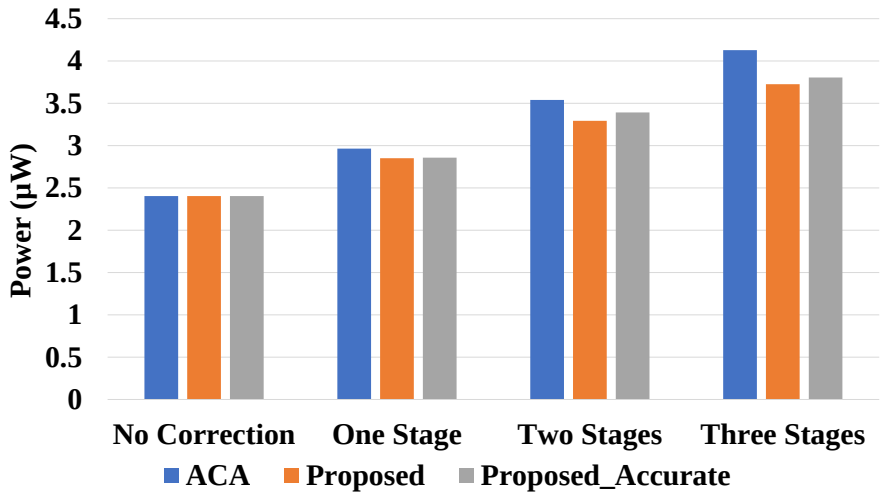


Figure 3.7: 32-Bit proposed design versions vs. ACA leakage power (μW) comparison.

From Figs. 3.6, 3.7 and 3.8, it can be noticed that the proposed design behaves better than the accuracy-configurable adder (ACA) [37] in terms of design parameters such as dynamic power,

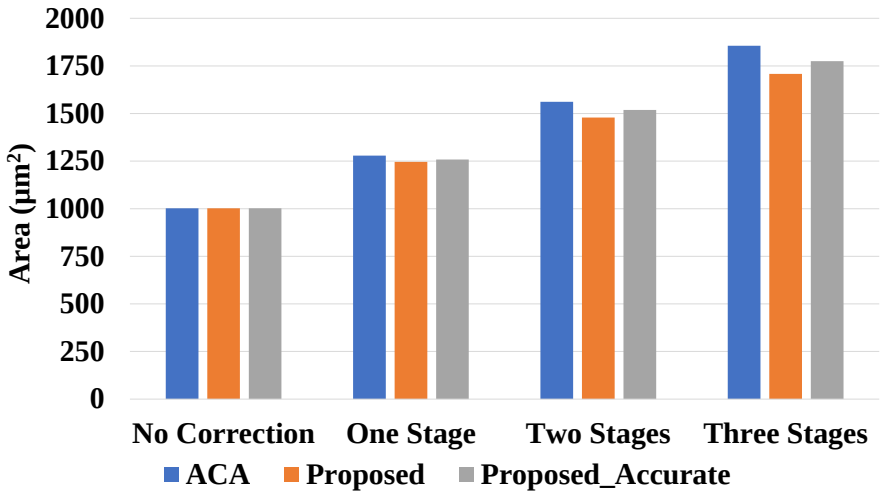


Figure 3.8: 32-Bit proposed design versions vs. ACA area (μm^2) comparison.

leakage power and Area. This is because the proposed design incurs a smaller silicon area due to limiting the error detection logic gates counts when compared to the ACA error detection technique. Consequently, the decreased hardware would result in lower levels of the dynamic and leakage power consumption. For the Proposed_Accurate design version, due to the additional carry-out signal detection of the active correction stages, it shows a limited increase of logic gates, when compared to the proposed design version. This explains the increase of the total design area and the levels of dynamic and leakage power consumption. However, it still shows smaller values when compared to the ACA design.

For delay values in Fig. 3.9, the proposed design shows a small enhancement of the execution speed at the final correction stage, besides stable delay values through all correction stages. These values can illustrate the independence characteristic of each sub-adder in which the critical path delay is the same for all segmented blocks. This is referred to the limited equality checking of two signals values for error detection.

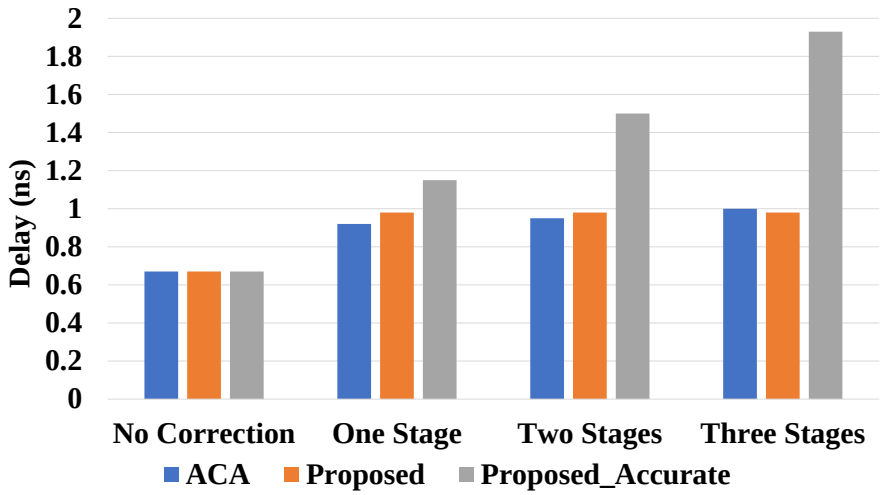


Figure 3.9: 32-Bit proposed design versions vs. ACA delay (ns) comparison.

In the case of the Proposed_Accurate version with an additional carry-out signal detection of the active correction stages, despite the degradation of correction stages delay values, it still behaves with higher speed regarding conventional adders like ripple carry adder (RCA) as will be shown in section 3.9 of further comparison. Moreover, the Proposed_Accurate designs keeps lower design overhead in terms of area and power consumption levels when compared to ACA for all stages.

Table 3.2: Average reduction ratio values of the proposed design version compared to ACA design

Parameter	Proposed Design vs. ACA
Dynamic Power	**7.3%**
Leakage Power	**5.2%**
Area	**4.6%**
Delay	**-1.9%**

Table 3.2 provides the average reduction ratios resulted from the proposed design version when compared to the ACA design for the approximate addition and all correction stages. Obviously, due to the resulted smaller design area, significant improvements

are introduced in terms of power (dynamic and leakage) and area values for all stages of the proposed design. However, the speed value is limited to the carry-out delay that equals to the full length of the sub-adder. As a result, this would introduce more delay when compared to the ACA design, thus, explain the limited negative average ratio of delay.

3.7 ERROR ANALYSIS

In this section, we present the second part of the proposed design evaluation by analysing the expected error levels of its resulted outputs. The analysis includes the mathematical models and the approximate error metrics simulations.

3.7.1 *Error Probability Model*

In first part of this section, we attempted to analyse the probability of erroneous sum occurrence (i.e., the chance of undetected carry propagation) through the sub-adders. This analysis would lead to predicting the maximum error percentage (i.e., the error bound) of the proposed design results within the total space of the outputs.

To analyse error occurrence of the proposed design, assume that the final carry-out from the previous sub-adder is C1 equals to '1', and the carry-out of the overlapped four bits in the current sub-adder is C2 equals to '0'. This case would check the carry propagation from the lower order (i.e., least significant) part of the previous sub-adder. All other cases would introduce no errors because the carry can be generated by the addition of inputs having values of ones (i.e., C1 = C2 = '1'), or the carry might be stopped in

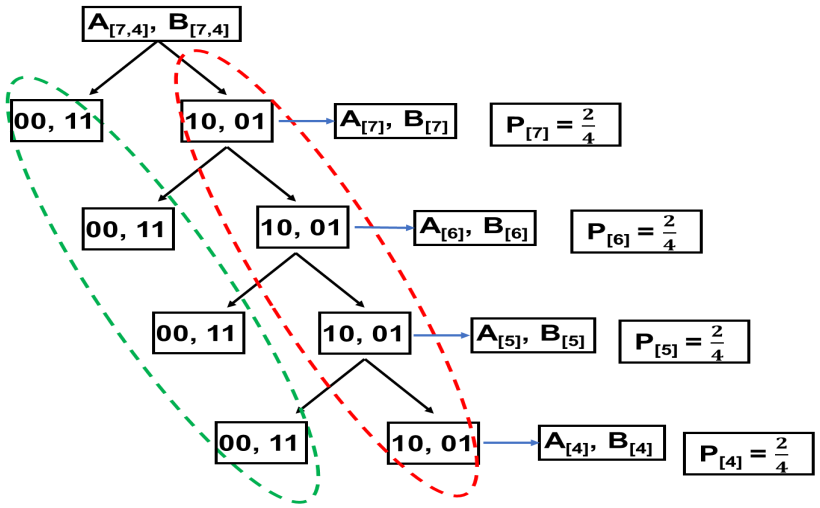


Figure 3.10: Tree of probability of carry propagation based on inputs bits values assuming C1= '1' and C2= '0'.

the case of addition of inputs having values of zeros (i.e., C1 = C2 = '0').

The red dashed line in Fig. 3.10 shows the case of carry propagation for the input bits (A [7,4], B [7,4]), which in turns would result in an error (i.e., mismatch) between the two carry values C1 and C2.

The analysis also needs to consider the carry generate or propagate from the higher order (most significant) bit of the lower part of the previous sub-adder. This bit location might either generate carry or propagate a carry signal to the overlapped 4-bits part.

The maximum error bound can be analysed using the following equation (Eq. 3.1).

$$|E_i| = P_i \quad \times \quad W_i, \tag{3.1}$$

where (i) is the bit index number in the sub-adder, E_i is the expected magnitude of error due to bit (i), P_i is the probability of error of bit (i), and W_i is the weight of bit (i).

The probability of correct result at each bit location would be equal to $\frac{1}{2^i}$,then, the probability of error would be equal to $1-\frac{1}{2^i}$.

$$P_i = \left(1-\frac{1}{2^i}\right) \tag{3.2}$$

The bit weight (i) within the sub-adder is calculated by the following equation.

$$W_i = \frac{1}{2^{((MSB-i)+1)}} \tag{3.3}$$

$$|E_i| = \left(1-\frac{1}{2^n}\right) \times \frac{1}{2^{((MSB-i)+1)}} \tag{3.4}$$

where n is the number of the overlapped bits between the sub-adders. In the proposed design, the overlapped part has four bits. As a result, the error bound can be expressed by the following equations.

$$\begin{aligned} |E_i| &= \left(1-\frac{1}{2^4}\right) \times \frac{\sum_{i=3}^{i=7} p_i}{2^{((7-4)+1)}}, \\ &= \left(1-\frac{1}{16}\right) \times \frac{\frac{2}{4}+\frac{2}{4}+\frac{2}{4}+\frac{2}{4}+\frac{3}{4}}{16} \end{aligned}$$

$$P(C_{1=1}, C_{2=0}) = 0.94 \times 0.172 = 0.162 \tag{3.5}$$

This result shows that at maximum, 16.2% of the resulted approximated outputs would have erroneous sum values (i.e., not equal to the exact sum value). In our experiments of error levels, we have used the relative error distance (RED) metric which mea-

sures the magnitude difference value between the approximated and exact output result.

The following section of the (RED) results simulation will show that the correct results of the proposed addition approximately equal to 84% of the outputs space. Hence, this might confirm the mathematically derived error bound of the proposed approximate adder design.

3.7.2 *Error Metrics Evaluation*

In this part, we explore the behaviour of the proposed design regarding three error metrics described in the previous chapter, in section 2.5 of approximate error metrics. The targeted error metrics are the relative error distance (RED), mean relative error distance (MRED) and the cumulative probability distribution (CPD) of error.

Relative error distance (RED) distribution analysis has been done for each design over different error values. RED simply measures how far the significance of resulted output's error when compared to the exact output of the conventional adder. RED analysis has been made for both proposed design versions (Proposed and Proposed_Accurate) and the ACA adder design, as they are compared to the exact outputs from a conventional adder. However, despite the simplicity of this measurement, it would show the effect (i.e., error magnitude) of the proposed design stages regarding the final quality of the outputs.

The following equation shows the arithmetic expression of the RED value.

$$RED = \frac{|Exact\ output - Approximated\ output|}{Exact\ output}, \quad (3.6)$$

For clarifying, an example when the RED value equals '0', then the approximated output value is correct, and there is no difference between it and the output value of the exact adder. On the other hand, if the RED value equals '0.01', then the approximated output value is not fully correct, and there is a difference between its value and the exact value by the percentage of 1% (i.e. it has 99% of accuracy).

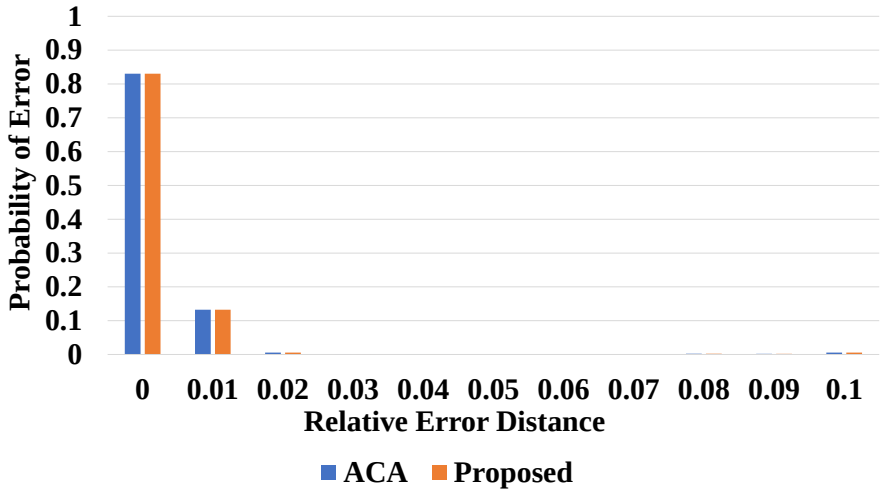


Figure 3.11: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (No correction stages).

Fig. 3.11 shows the case of designs (proposed design version versus ACA design) without any correction stages (i.e., the approximate addition stage). The two compared designs have the same general implementation of adder segmenting to sub-adders with the same bit widths. Each sub-adder has the same number overlapped bits from the previous sub-adder, which equal to the half

of its bit width except the first sub-adder. As a result, the error analysis shows that the proposed design version has the same behaviour as ACA. However, since the targeted design modification is about the enhancing error detection with the lightweight technique, the following figures of error analysis of the correction stages would show more enhancements of limiting errors, yet, with a lower design overhead.

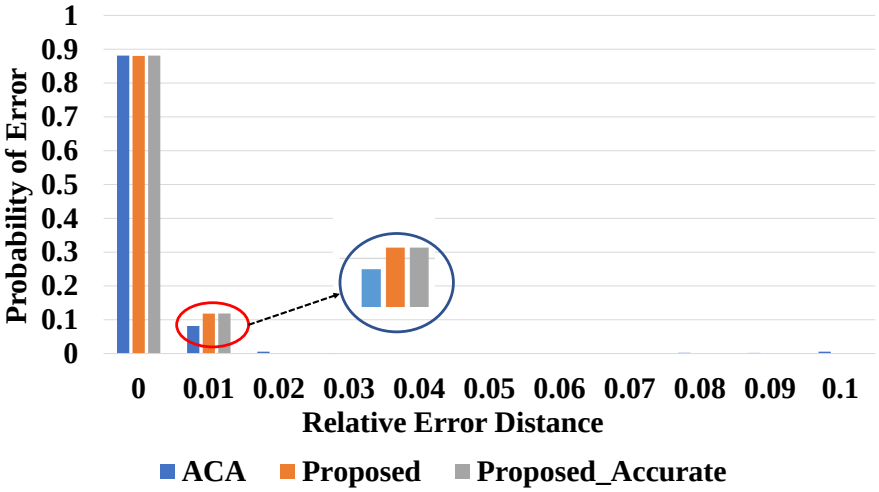


Figure 3.12: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (One correction stage).

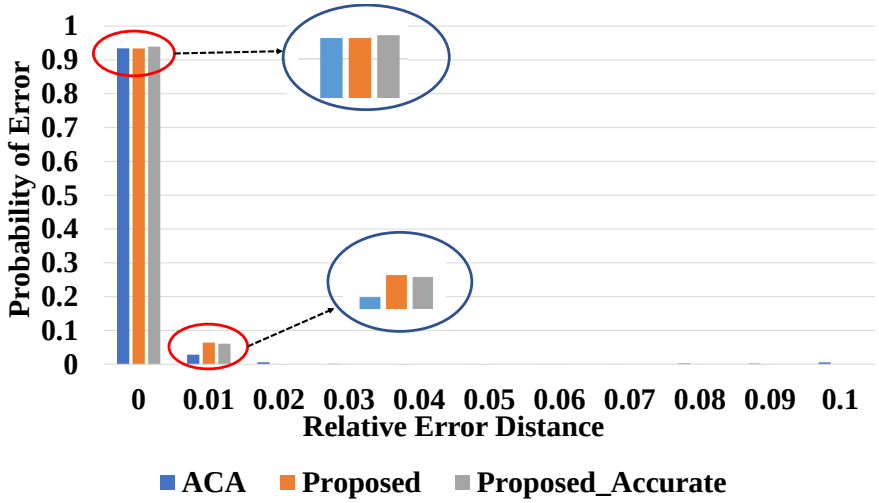


Figure 3.13: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (Two correction stages).

Figs. 3.12 and 3.13 show the error analysis of designs with one correction stage and two correction stages, respectively. It can be noticed that our design versions (Proposed and Proposed_Accurate) have approximately the same error distribution as in ACA design. Nevertheless, due to the fact that the proposed design versions start correcting the most significant (i.e., higher order) sub-adders' errors, they show results with fast convergence to exact sum values. Further, they present more stability in terms of RED values as they start to be limited strictly between 100% and 99% of accuracy.

The analysis of the highest level of accuracy is provided in Fig. 3.14. In this mode, the full three correction stages would be in action. It is obviously shown that the behaviour of the proposed design version continues to show the same behaviour as ACA. However, the improved error detection mechanism of the Proposed_Accurate design version (by considering the carry propagation of the correction stages) guarantees the full error detection and correction of all induced sum errors, thus, shows the best result of 100% accurate outputs.

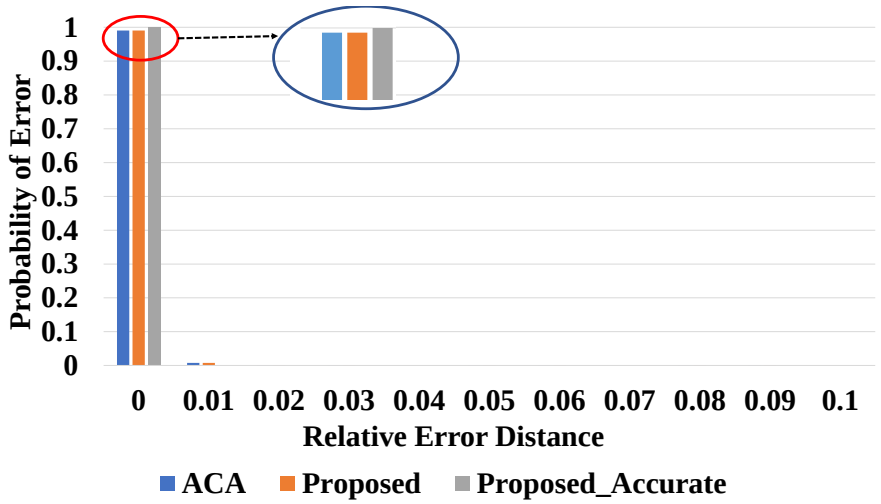


Figure 3.14: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (Three correction stages).

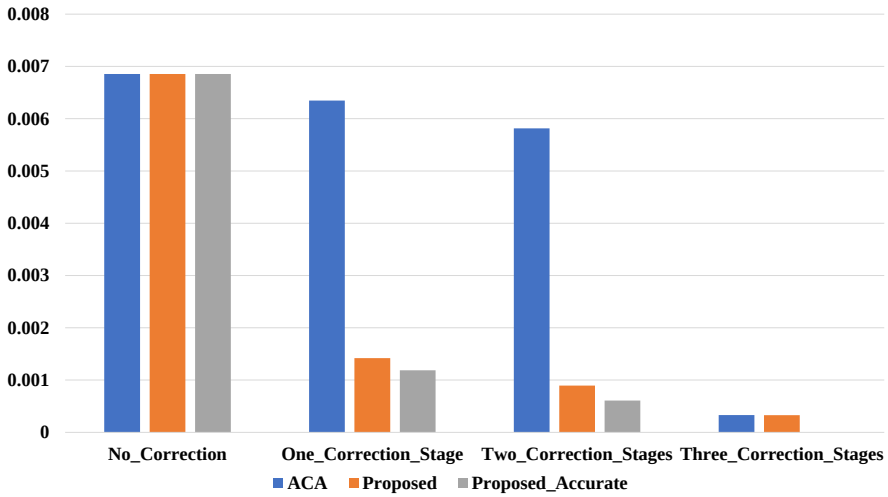


Figure 3.15: The 32-Bit adder designs MRED values comparison through different correction stages.

Fig. 3.15 presents the mean relative error distance(MRED) values comparison, which shows the mean value of errors within test space or input vectors at each correction stage [63]. At the approximate addition stage with no error detection and correction, the proposed design versions and the ACA design show the same mean error values. This is due to the fact that they have the same sub-adder segmenting implementation. However, by using the first and second correction stages, the proposed versions' MRED values are drastically decreased showing better accuracy and consistency. This referred to the significance-driven structure of the proposed correction stages, which guarantee the fast convergence to the exact sum values and limit the final outputs to low magnitude values of errors early.

For the final correction stage, our proposed design version provides MRED values similar to ACA design. However, for the extended proposed version (Proposed_Accurate) with full error detection and correction, the MRED evaluation dropped to zero value, which eliminates the probability of not handled errors and confirms the final outputs full accuracy.

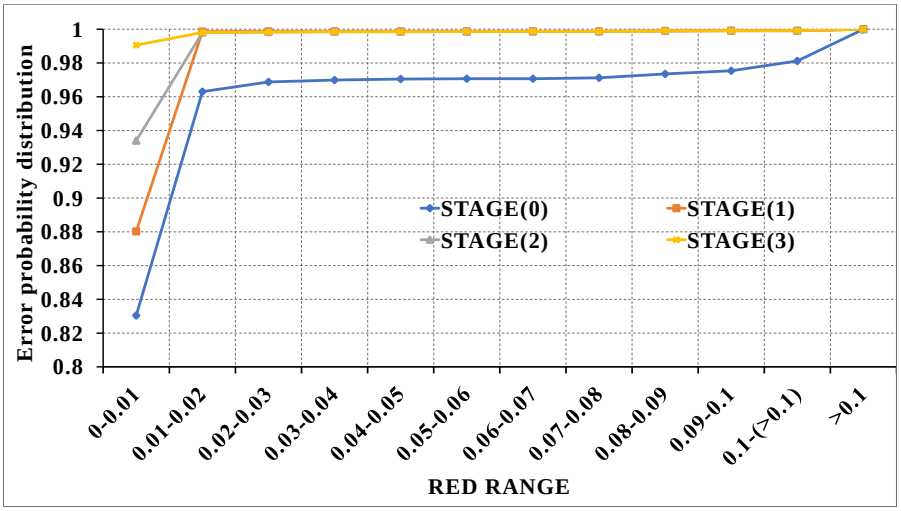


Figure 3.16: The cumulative probability distribution for the error through different correction stages.

Furthermore, Fig. 3.16 presents the cumulative probability distribution (CPD) for relative error distance (RED) levels through approximate addition and error correction stages. This analysis describes the change of percentages of the output space regarding the RED values. Additionally, it shows the speed of the resulted outputs become closer to the exact addition results. Thus, it would explore how each correction stage might affect the total output accuracy.

It can be noticed that the cumulative value at approximate addition stage without any error correction shows a quick move to the RED range ($0.0 \leq RED < 0.02$). This means that approximately 96% of the output space hasRED values of ($0 \leq RED < 0.02$). This percentage would be gradually increased with small values and reached '1' (i.e., 100%) of the output space at RED > 0.1.

For the correction stages, the approximately show the same cumulative behaviour of the sharp jump to the RED range ($0.1 \leq RED < 0.02$). However, they present the high speed to cover all the output space (i.e., reach 100% of the outputs number) when compared to the previous stage of approximate addition with no

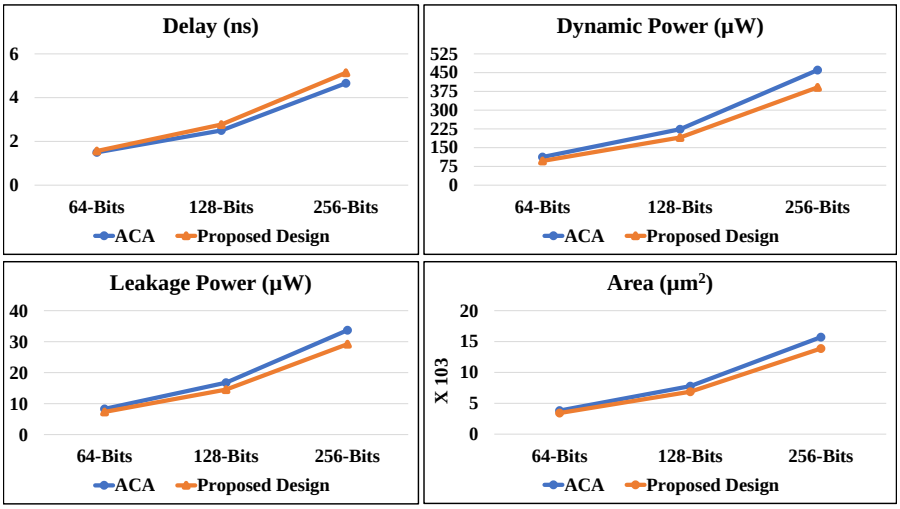


Figure 3.17: ACA vs. Proposed large bit-width adder designs hardware comparisons.

error recovery. The only difference of MRED values between the correction stages is the percentage of the premier resulted exact values. It can be noticed that as the number of active correction stages increased, the portion of the correct results (i.e., RED ='0') is increased, and thus, improves the quickstep to cover all the resulted output space.

3.8 LARGE BIT WIDTH ADDERS EVALUATION

For design scalability checking, a further hardware evaluation was implemented for different adder designs (ACA and Proposed design version) with larger bit widths (64-bits,128-bits and 256-bits). The hardware metrics evaluation for each design with its full correction stage architecture (i.e. using three correction stages) is shown in Fig. 3.17.

It can be noticed that the proposed design version preserves lower values in terms of power (dynamic and leakage) and area when compared to the ACA design. Nevertheless, it still shows a very limited delay degradation when larger bit-width adders

are in use. Remarkably, these reduction values would confirm the scalability advantage of the proposed design for large bit-width adders over the ACA design.

3.9 FURTHER HARDWARE COMPARISON

Table 3.3 shows an extended comparison of the proposed design two versions (Proposed and Proposed_Accurate) regarding previous efforts.

Obviously, the proposed design shows more advantages and enhancements of the design parameters such as power, delay and area. In detail, for both the power and area, the proposed design versions present better values when compared to ACA [37] and Accurus [8] designs. For the delay values, the proposed design version shows a higher speed regarding ACA and Accurus designs. These results of the proposed design version have been confirmed by the delay-power-product power-delay-product (PDP) values, which shows smaller numbers while using the full correction stages when compared to ACA and Accurus designs. On the other hand, despite delay degradation of the Proposed_Accurate version, it still presents better speed than conventional ripple carry adder (RCA).

3.10 IMAGE PROCESSING APPLICATION

To evaluate the proposed design in a real-world implementation, this work exploits a key block in the image processing application known as Gaussian blur image filter, which has been previously defined in section 2.10.4.

Table 3.3: Hardware metrics comparison of proposed adders and previous efforts

Design	Max Correction Stages	Delay (ns)	Dynamic Power (μW)	PDP	Leakage Power (μW)	Area (μm^2)
ACA* [37]	No correction	0.67	22.243	14.9	2.4038	1001.95
ACA [37]	Three	1	56.161	56.161	4.1278	1855.72
Accurus [8]	Three	1.13	56.916	64.31	4.1796	1895.71
RCA	Accurate	2.22	18.127	40.24	2.0263	702.81
Proposed*	No correction	0.67	22.243	14.9	2.4038	1001.95
Proposed	Three	0.98	49.117	48.13	3.726	1708.33
Proposed_Accurate	Three	1.93	49.905	96.31	3.805	1774.97

In the Gaussian blur image filter, the process of blurring an image is resulted from convolving each pixel in the image with the Gaussian function. The filter is commonly used since blurring is a major effect in graphics software, and specifically, with algorithms that are sensitive to noise such as edge-detection algorithms, and thus, improving the result of the algorithm. Further, it is generally considered a pre-processing stage in computer vision algorithms for enhancing image structures at different scales [38].

For the implementation analysis, a general Matlab test bench was proposed to apply the Gaussian blur image filter test. The test bench checks the actual behaviour of (20-bits) Proposed_Accurate adder design version during multiple correction stages. The peak signal to the noise ratio (PSNR) is used to measure the quality of the output images after applying the Gaussian blur filter by comparing the resulted image quality to the optimum blurred image of the conventional circuit of the Gaussian blur filter.

From Fig. 3.18, at Mode=1, the most significant correction stage is in operation; the resulted image shows a PSNR value of (29.8 dB) with acceptable quality and limited distortion. However, for Mode=2 where two correction stages are in action, a remarkable improvement happened to the PSNR value by (42.6 dB) with a very acceptable quality of the output. In Mode=3 with all three active correction stages, the output image presents the optimum PSNR value, the same as the resulted image of the exact adder.

Inspite of the appearance of low PSNR value at the first correction stage, the proposed design might be considered an attractive adder design for some application like the biomedical applications, which are generally interested in high speed, very low power and acceptable outputs quality [96, 2, 1].

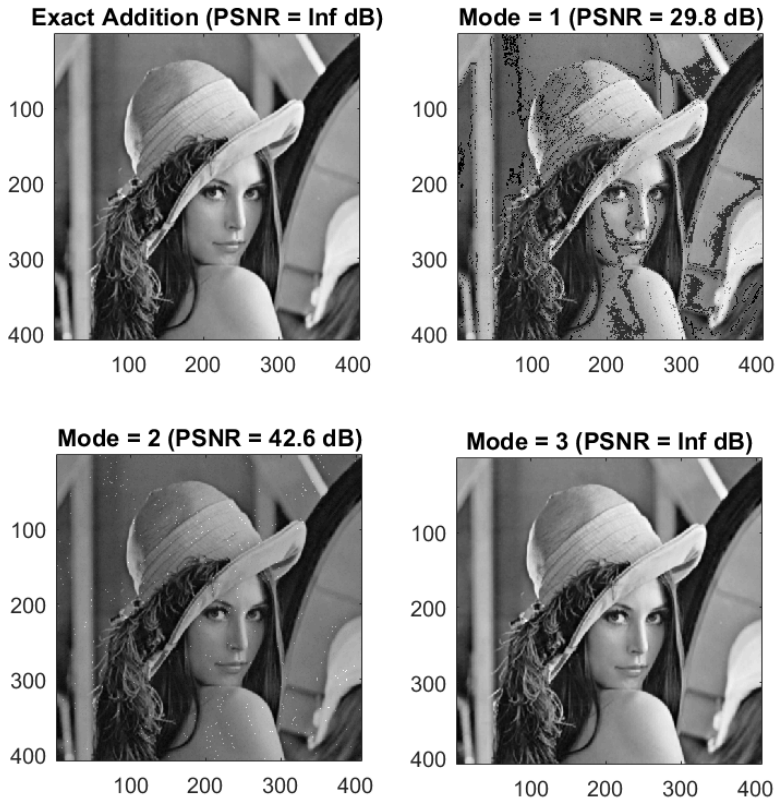


Figure 3.18: Gaussian blur image filter test.

3.11 SUMMARY

In this chapter, a configurable-accuracy approximated adder with a new error detection technique was proposed. The new technique mitigated the error detection overhead by limiting the number of the targeted error signals for equality checking, thus, decreasing the number of the required logic (XOR) gates in the error recovery circuit.

As a result, this incurred lower design overhead with 12%, 10% and 8% reduction ratios for dynamic power, leakage power, and area, respectively, at the final correction stage (i.e. highest level of accuracy). Furthermore, the error correction stages were structured in a significance-driven scheme, in which the first correction

stage starts correcting the most significant (highest impact) erroneous sum bits. Hence, this guarantees a fast convergence to the exact output values at the premier correction stages.

On the other hand, the proposed design was extended to a modified version, which considers the propagation of the correction stages' carry-out to successive significant stages during the active correction process. Remarkably, the (Proposed_Accurate) design version continued to show a lower design overhead and the best results of accuracy consistency (i.e. between 99% and 100% accuracy levels), and further, 100% accurate results at the final stage.

The proposed adder with large bit-widths has reserved the design parameters reduction ratios with acceptable scalability overhead. For the design implementation in the image processing application, the proposed adder design was applied in the Gaussian blur filter block, which resulted in high PSNR output values of (29 and 42 dB) for the two premier correction stages, and the optimum image quality while using the full correction stages.

The next chapter explores another modification of approximate (speculative) adder, in which a simple sub-adders' segmenting technique is introduced. The newly proposed technique uses the principle of carry kill signals that limit or stop the propagated carry chain when the addend inputs have zero values. The main advantage of this technique is to avoid sub-adders' parts overlapping, hence, reducing the resulted area and design overhead. Further, the proposed design in the following chapter keeps using the error recovery circuit in order to keep highly acceptable output quality.

4

APPROXIMATE ADDER DESIGN WITH CARRY KILL SEGMENTATION TECHNIQUE

4.1 INTRODUCTION

In the previous chapter 3, we have introduced several modifications on a widely known accuracy-configurable approximate (speculative) adder design ACA. A new lightweight error detection technique has been proposed, in addition to organising the controlled correction stages in a significance-driven structure. As a result, lower design overhead is introduced with a smaller area and lower levels of power consumption which present better design scalability of large bit width adders. Furthermore, the proposed modifications would guarantee high convergence to the exact result at the premier error correction stages.

In this chapter, we continue to address the challenge of approximate adder large design overhead. This effort introduces a novel technique for segmenting the sub-adders in the approximated (speculative) adder. The proposed technique is based on using the principle of the carry-kill signal in order to stop the carry chain propagation at specific bit locations, which in turn, specify the number of sub-adders. The new dividing technique does not propose any overlapping parts from the previous sub-adders. Conversely, it preserves approximately the same length of the basic N bits adder. Hence, a smaller area is introduced which results in lower levels of design overhead and power consumptions. However, in order

to preserve highly acceptable accuracy of the approximated sum result, a carry-in value prediction technique has been augmented to the proposed adder design with high probability.

In general, each sub-adder in the approximate (speculative) adder design uses overlapped bits to predict a carry-in value, and then, produces a number of resultant bits that contribute to the final summation output. Consequently, approximate (speculative) adders might be roughly categorised into three techniques, regarding the number of the resulted approximate sum bits at each sub-adder: the first one proposed the use of multiple overlapping sub-adders with one resultant bit per sub-adder to the final sum [91, 14, 45]. The second technique divided addition into multiple blocks with overlapping parts, and each block is responsible for generating a range of bits to the final sum [105, 37, 100, 45, 16, 43]. The proposed design in this chapter follows the third technique, in which each sub-adder results in sum bits number that equals to its full bit-width as can be found in [16].

For the error correction process, we maintain the configurable multi correction stages in a significance-driven structure. This stages structure would guarantee that the higher order (i.e., most significant) sub-adders to be corrected first, thus, imply fast convergence to the exact result. The multi-stage error correction mechanism allows the designer to limit the delay of error correction and the ratio of consumed power. This is done by controlling the activation of the number of correction stages (accuracy level) and limiting the error checks number. Hence, the more the pipelined stages, the smaller the carry chain length of the design sub-blocks and the more performance achieved [37].

The proposed work in this chapter[1] focuses on improving the procedure of segmenting the sub-adders in order to increase the expected benefits of such approximate (speculative) adder designs. A novel segmenting technique of sub-adders has been introduced by using carry-kill bit locations (with input bits values equal to zero). Further, a lightweight carry-in prediction and error detection techniques are proposed, which leads to lower design overhead and more scalability for larger adder size. The proposed design is augmented with a significance-driven multi-stage error recovery circuit, which implies fast convergence to correct outputs.

The proposed design presented improvements of (16%), (17%) and (18.6%) for dynamic, leakage power and area respectively. Nevertheless, outputs reserved a general high accuracy level, which limited between 99% and 100% for the majority of input space. The proposed design was implemented in an image processing application, which resulted in high PSNR values of (53 and 83 dB) for the two premier correction stages, and 100% exact results while using full correction stages [4].

4.2 PROPOSED DESIGN

In general, the carry kill signal shown in Eq. 4.1 participates in a vital role by limiting the chain of carry propagation in the adder, and then, the critical path delay. In this effort, we exploited this characteristic in order to divide the adder into smaller sub-adders, and further, to apply the real carry detection in parallel. The new segmentation technique would lead to the decreased silicon

1 This effort has been published in IEEE Xplore as, Khaled Al-Maaitah; Ghaith Tarawneh; Ahmed Soltan; Issa Qiqieh; Alex Yakovlev, Approximate adder segmentation technique and significance-driven error correction, 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2017, pp 1-6.

area and lower power consumption, thus, mitigating the emerging design overhead challenge in configurable-accuracy approximate adders.

$$Carry\ Kill\ Signal_{(j)} = SUM_{(j)}[0+0] + Carry_{(j-1)}, \quad (4.1)$$

4.2.1 *Segmenting Technique*

The proposed design follows the general technique of dividing the adder into independent (i.e., not overlapped) smaller sub-adders. However, a one-bit location is added after each sub-adder to limit the long carry chain as depicted in Fig. 4.1 that shows the general architecture of the proposed adder design. In detail, for an N bit adder, the number of segments will equal to M where (M=N/K); K is the sub-adder bit width (K=L-1), and L is the total bit width of the sub-adder (i.e., includes sub-adder bits and the added bit location for limiting the carry chain).

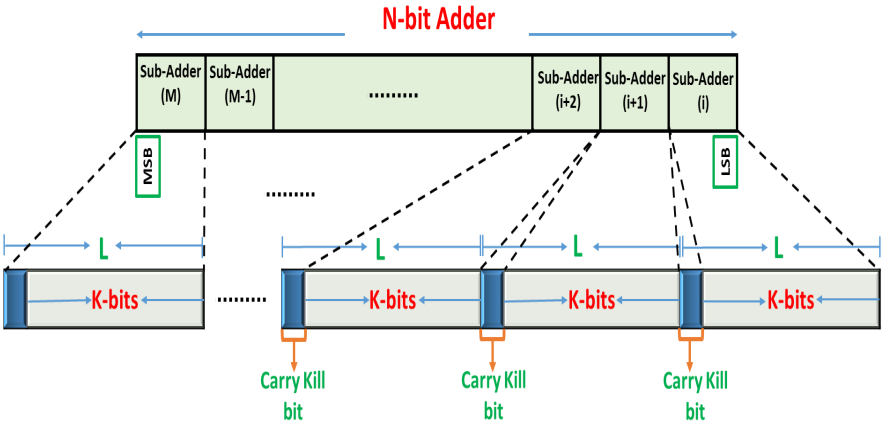


Figure 4.1: The proposed general adder segmentation technique using the carry kill new bit locations.

The new bit locations have input values of '0' (e.g., A+B= 0+0 =0), thus, result in sum values equal to '0'. The added location keeps its value to '0', or can be changed to '1' in the case of carry propagation or generating from the neighbouring most significant bit (MSB) of the sub-adder. The new sum value of the added bit location would hold the real output carry of the sub-adder. As a result, the value of the real carry at each bit location will be used in the process of the erroneous sum value detection and correction of each sub-adder.

4.2.2 *Carry Prediction Technique*

Carry-in prediction to each sub-adder in the proposed design is presented in Fig. 4.2. It can be noticed that carry prediction applies an AND logic gate between every two successive sub-adders. As illustrated in Eq. 4.2, the predicted carry value of sub-adder (i) is equal to the value of generate signal (G) of the most significant (i.e., higher order) bits of previous sub-adder (i-1). In detail, the predicted carry value will be equal to '1' if both inputs to the AND logic gate have values of '1', otherwise, the predicted value will be equal to '0'. The carry-in to the first sub-adder will be truncated to '0' value. A similar technique of carry prediction by AND gate was previously used in the lower-part-OR (LOA) adder design effort [50]; however, with completely different approximate adder architecture.

The following Eq. 4.2 presents the carry-in generate signal (G), which is used for carry-in value prediction between every two sub-adders in the proposed design. The result of AND-ing the values of

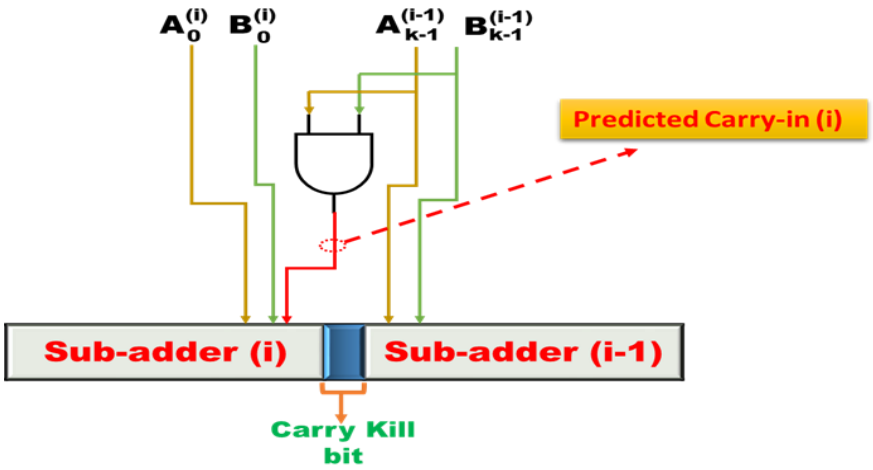


Figure 4.2: The proposed carry-in prediction technique for each segmented sub-adder.

the most significant bits (MSB) of the previous sub-adder will be the value of the predicted carry-in to the current sub-adder.

$$Carry-in_{(i)} = G_{(MSB(i-1))} = A_{(MSB(i-1))} . B_{(MSB(i-1))}, \quad (4.2)$$

where i is the current sub-adder, i-1 is the previous sub-adder.

An example of the (32-bit) approximate adder uses the proposed segmenting technique is presented in Fig. 4.3, and the following points summarise its main parts:

- Assuming the bit width of each sub-adder (K= 8-bits), the number of sub-adders equals to M=N/K= 32/8 = 4. The length of each segment (sub-adder) is increased by one additional bit location in order to limit the carry propagation and hold the real carry-out of the adder segment. Hence, L equals to (K+1)= 9 bits.

- The carry-in of each sub-adder is predicted as the result of AND-ing the most significant (MSB) input bits of the previ-

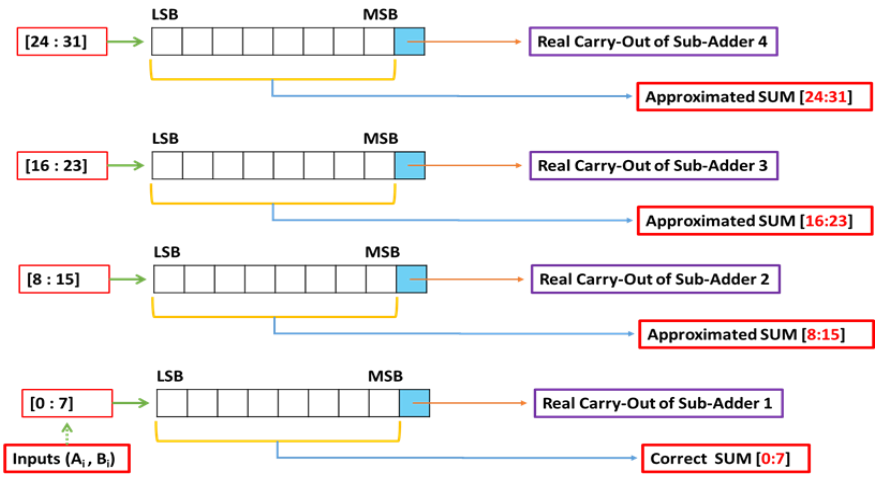


Figure 4.3: Example of (32-bit) proposed approximate adder using the segmenting technique of carry kill bit locations.

ous sub-adder, except the first sub-adder, which has been truncated carry-in = 0.

- Each bit in sub-adder participates in one sum bit in the final approximated sum output value. However, the carry kill bit location value is not regarded as a sum bit and it will be discarded.

- The final carry kill bit location at sub-adder (4) is considered the final carry-out value of the whole approximate adder.

- The length of the sub-adders might be configurable at the design time, depending on the application requirements.

4.2.3 *Error Detection and Correction*

The proposed circuit in Fig. 4.4 illustrates that for error detection at each prediction circuit, one XOR gate is used, and the error signal will be high if both the predicted and the real carry (in the carry kill bit location) are not equal as presented in Eq. 4.3.

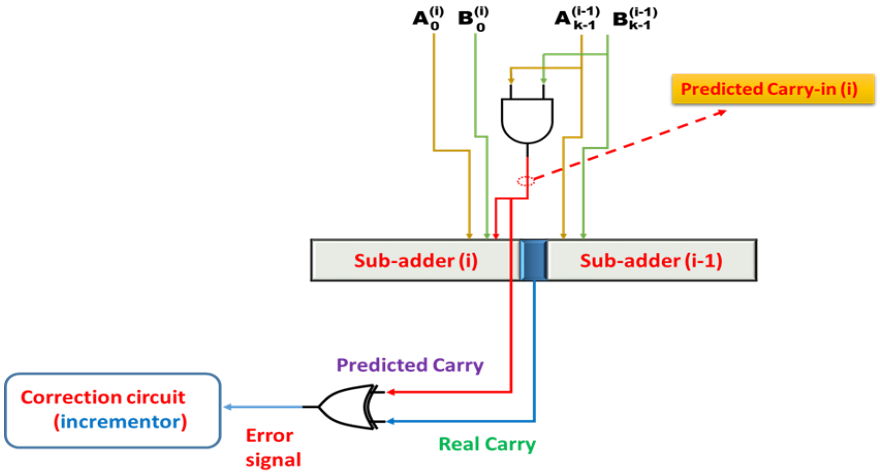


Figure 4.4: The proposed error detection technique augmented with the carry-in prediction circuit.

$$Error_{(i)} = G_{(MSB(i-1))} \oplus Carry\ Kill\ Bit_{(i-1)}, \tag{4.3}$$

This implies that error will signal high when there is a carry propagation from lower order bits than the used MSBs for prediction. This case will happen just once when the predicted carry-in is equal to '0' and the real carry-out of the previous adder is equal to '1'. Table 4.1 presents the inputs combinations and the probability of error correction when there is a carry propagation.

Table 4.1: One-bit inputs probability of carry prediction and error detection and correction

A	B	Predicted Carry	Real Carry	Error	SUM Correction
0	0	0	0	NO	NO
0	1	0	1	YES	SUM + 1
1	0	0	0	NO	NO
1	1	1	1	NO	NO

To correct the detected erroneous approximate sum value, we used the same procedure as in the proposed design in chapter 3.

For error recovery process, an incrementor circuit is used to compensate the missed carry-in value of '1' in the final output sum. Thus, the bit width of each incrementor would have the same number of bits resulted from sub-adder. Further, the error correction is implemented through multiple controlled stages during runtime (i.e., each correction stage has an incrementor for correction), which can be activated or turned off based on the required output accuracy level.

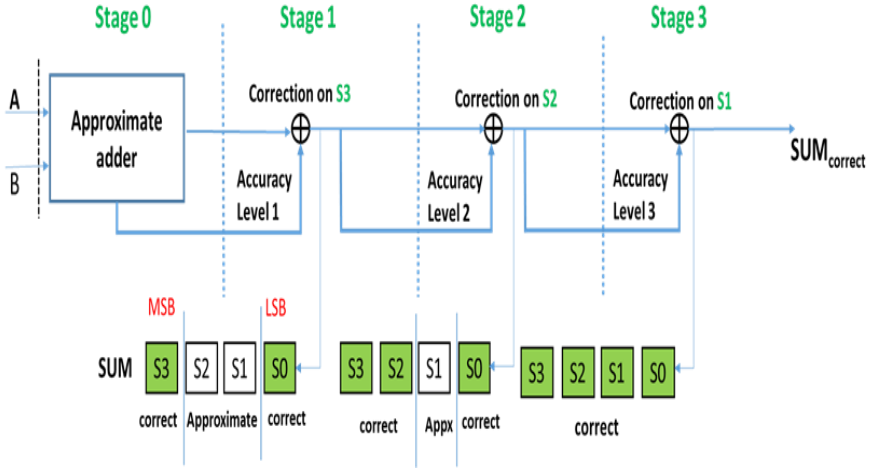


Figure 4.5: Significance-driven structure of error correction stages.

The correction stages are illustrated in Fig 4.5 and have the same structure of significance-driven principle in section 3.4 in the previous effort of chapter 3. In such a structure, the most significant (i.e., higher order) erroneous segment will be corrected first. This is due to the fact that higher order bits have the largest impact on the output result to the final sum when compared to the bits of the lower order sub-adders. Hence, this significance-driven structure of error correction would guarantee high convergence to the exact result, yet, with a small delay and power overhead. In this implementation, the least significant segmented part (sub-adder) will be corrected at the final stage, and only in the worst case of the full accuracy mode.

This structure of arranging the correction stages is similar to what exists in [8]. However, this work presents an extension of the error detection and correction process. The extended version of the proposed design is denoted by (Proposed_Accurate), in which the carry-out of each correction stage will not be overlooked in case its value was high. Conversely, the high carry-out value of a correction stage has to be propagated in order to correct the successive sub-adder output sum. As a result, this extension of error detection would guarantee the full accuracy of outputs at the final correction stage.

Based on Fig. 4.5, the following points can be noticed:

- The approximated adder gives the approximated SUM at each stage.
- (S0) is always correct as it uses truncated (not predicted) carry-in = '0'.
- The correction stage (incrementor) will result in the accurate sum part (coloured green).

Comparing the proposed error recovery process with other efforts such as accuracy-configurable adder (ACA) [37] that is described previously in section 2.8 of chapter 2, the proposed error detection and correction mechanism introduces lower design overhead. For instance, in the (32-bit) adder example in Fig. 4.6, it is obviously shown that four segmented sub-adders have to make just three error checks (since the first sub-adder is always correct), in contrast to six error checks have to take place for the same adder bit length in ACA. Additionally, we use just one incrementor circuit to correct the resulted erroneous 8-bit sum at each sub-adder, then, three incrementors are required for the whole correction process. Further, the extended error correction mechanism of the

(Proposed_Accurate) design version would guarantee full output accuracy by checking whether the results of the successive correction stages need a further correction or not.

4.3 NUMERICAL EXAMPLE

In this section, a (32-bit) binary numerical addition example is presented. The detailed example shows the principal methodology of adder division, approximate addition and error detection techniques of the proposed design.

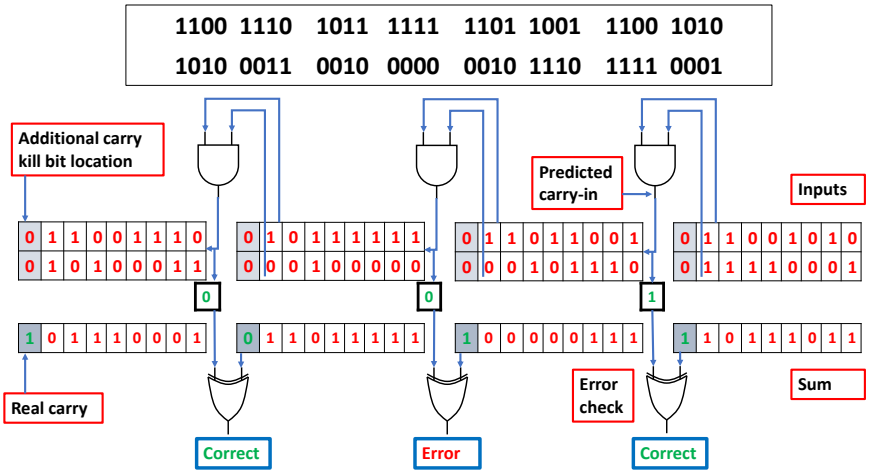


Figure 4.6: Proposed design numerical example of (32-bit) inputs addition, sub-adders carry-in prediction and error (carry propagation) detection.

The numerical example depicted in Fig. 4.6 presents the approximate addition and the segmentation technique of the adder into equal length (not overlapped) sub-adders. The new bit location at each sub-adder has the addition result of inputs with values of zero (i.e., A='0', B='0'). As a result, the propagated carry of the higher order bit of the sub-adder would be stopped in that location, thus, would have the real carry-out value of the sub-adder. On the other hand, an AND gate with two inputs is used for carry-in

value prediction to each sub-adder except the first adder which has a truncated carry-in value equals to zero '0'. It can be noticed that the inputs to each AND gate are the higher order (i.e., most significant) input values of the previous sub-adder.

For carry-in value misprediction detection, one XOR gate is used between every two sub-adders to check the inequality of the predicted carry-in value to the current sub-adder and the real carry-out of the previous sub-adder (located at the additional bit location). If an error has been detected, the resulted erroneous approximated sum would be corrected and incremented by one using the incrementor block.

4.4 DESIGN TRADE-OFFS

To examine the contribution of the proposed design, our experiments used Verilog to build (32-bit) different adder designs with their different correction stages, and test benches were used to test the functionality of each design with different accuracy modes. For the part of the comparison, we used Modelsim for error analysis simulations, which is based on Monte Carlo method that implies a large number of random variable as input values to the proposed approximate addition function. This test would provide the general behaviour of the proposed approximate adder and the error levels of each correction stage. Further, we exploited the Synopsys design compiler UMC (Faraday 90nm) technology to synthesise and evaluate the design parameters such as delay, power and area values.

To evaluate the modification of hardware design metrics, the proposed design is compared to the design effort of ACA [37]. The proposed design has two versions, where the first version has been

applied without considering the carry-out of correction stages, and the second version considers the carry-out of each active correction stage regarding the selected accuracy mode. For simplifying, the design version considering the correction stage carry-out is denoted as (Proposed_Accurate).

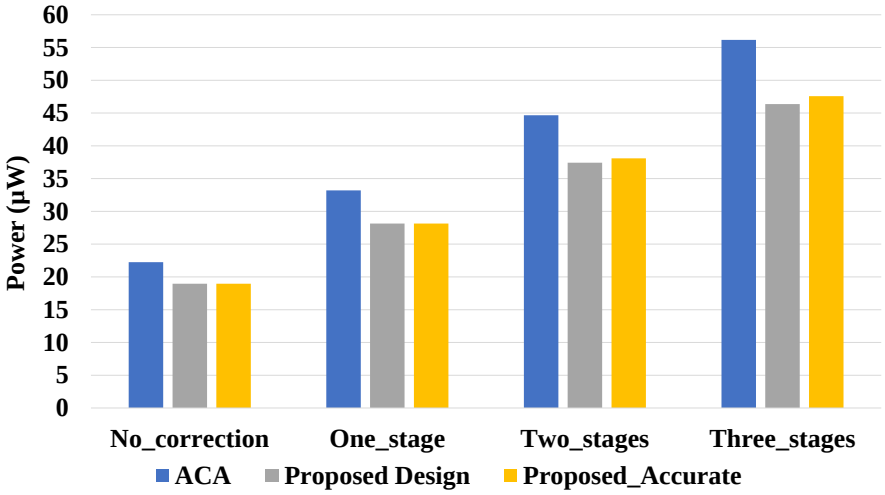


Figure 4.7: 32-Bit proposed design versions vs. ACA dynamic power (μW) comparison.

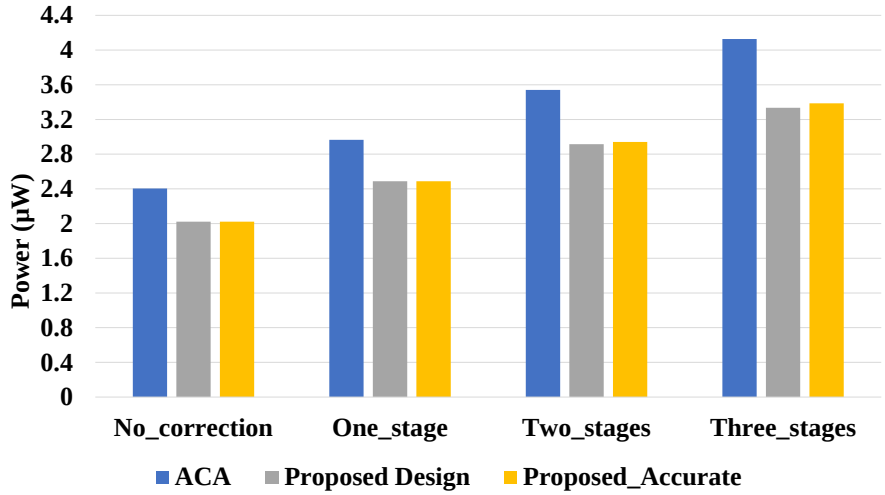


Figure 4.8: 32-Bit proposed design versions vs. ACA leakage power (μW) comparison.

From Figs. 4.7, 4.8 and 4.9, it can be noticed that the proposed design behaves better than the accuracy configurable adder (ACA)

in terms of design parameters such as dynamic and leakage power and area. These enhancements are referred to not using any overlapped (redundant) parts of the addend inputs, which are used in terms of carry speculation in other efforts. As a result, the number of the introduced sub-adders used for addition is decreased, resulting in that the total design area and power consumption are decreased as well. Further, the augmented lightweight error detection circuit would not incur high overhead to the design.

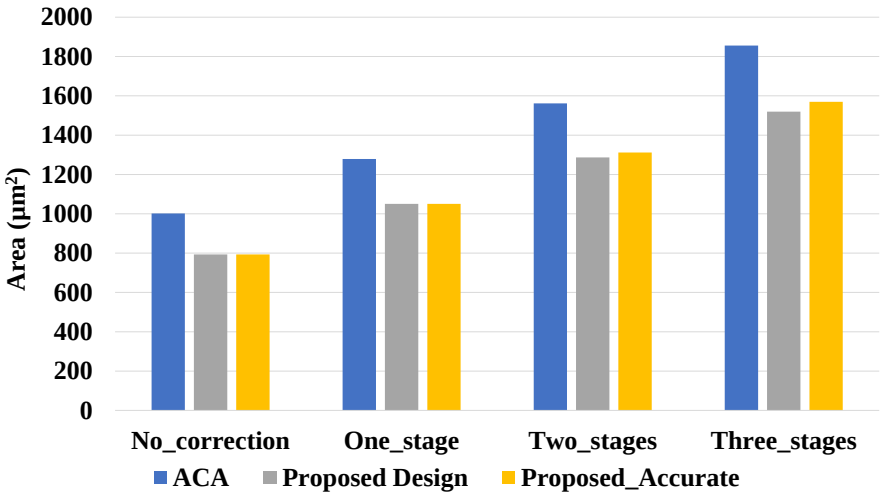


Figure 4.9: 32-Bit proposed design versions vs. ACA area (μm^2) comparison.

For delay values in Fig. 4.10, the proposed design has larger values with a limited range compared to the ACA design. This is due to the use of the carry prediction technique with AND gates, and the increased length of each sub-adder with more one-bit location. For erroneous addition recovery, (8-bit) length incrementors are used (instead of 4 bits length as in ACA), thus, needing more execution time. However, the proposed design version (i.e. not Proposed_Accurate), shows more stability regarding delay values through all correction stages. This, in turn, presents the independence characteristic of each sub-adder in which the critical path delay is the same for all segmented blocks.

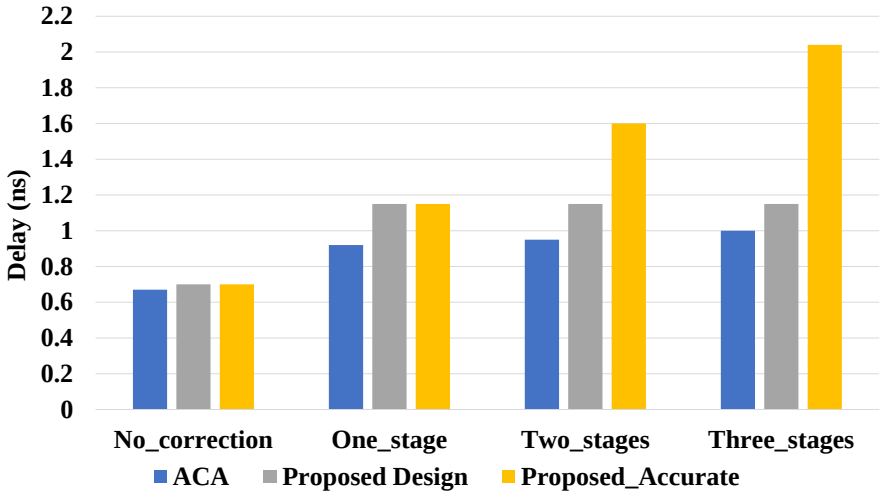


Figure 4.10: 32-Bit proposed design versions vs. ACA delay (ns) comparison.

On the other hand, for the case of the Proposed_Accurate design version (correction stages carry-out in concern), it reaches the full accuracy at the highest correction mode. However, it continues to behave better in terms of power consumption and area when compared to ACA for all stages, and with a comparable speed to the exact ripple carry adder (RCA).

Table 4.2: Average reduction ratio values of the proposed design compared to ACA design for all correction stages.

Parameter	Proposed design vs. ACA
Dynamic power	**16%**
Leakage power	**17.2%**
Area	**18.6%**
Delay	**-16%**

Table 4.2 provides the average values of reduction ratios resulted from the proposed design through approximate and all correction stages. Obviously, due to the decreased design overhead of using less number of sub-adders and logic gates for error detection, significant improvements are introduced in terms of positive reduction values of dynamic power (16%), leakage power (17%),

and area (18%). On the other hand, although of the degradation of the delay values in the proposed design, which are referred to the addition of carry prediction logic gates, it still shows higher speed when compared to exact adder like ripple carry adder (RCA)as will be shown in section 4.7 of further comparison in this chapter.

4.5 ERROR ANALYSIS

Since approximate designs error characteristics drive a great attention to previous efforts such as in [44, 52], this section presents the second part of the proposed design evaluation by analysing the expected error levels of its resulted outputs. The analysis includes the mathematical model and the approximate error metrics simulations.

4.5.1 *Error Probability Model*

In the first part of this section, we attempted to analyse the probability of erroneous sum occurrence (i.e., the chance of carry misprediction and unhandled carry propagation) through the sub-adders. This analysis would lead to predicting the maximum error percentage (i.e., the error bound) of the proposed design results within the total space of the outputs.

To analyse error occurrence in the proposed design, assume that the final carry-out from the previous sub-adder is C1 equals to '1', and the predicted carry-in to the current sub-adder is C2 equals to '0'. This case would check carry misprediction and the unhandled carry propagation from the previous sub-adder. As shown in Table 4.3, all other cases would introduce no errors since the carry can be predicted to value of '1' by ANDing most significant

Table 4.3: Error probability of carry value regarding input combinations assuming C1= '1' and C2= '0'.

B_{MSB-1}	A_{MSB-1}	B_{MSB}	A_{MSB}	**Carry prediction error**
0	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**
1	**1**	**0**	**0**	**0**
0	**0**	**1**	**0**	**0**
1	**0**	**1**	**0**	**0,1**
0	**1**	**1**	**0**	**0,1**
1	**1**	**1**	**0**	**1**
0	**0**	**0**	**1**	**0**
1	**0**	**0**	**1**	**0,1**
0	**1**	**0**	**1**	**0,1**
1	**1**	**0**	**1**	**1**
0	**0**	**1**	**1**	**0**
1	**0**	**1**	**1**	**0**
0	**1**	**1**	**1**	**0**
1	**1**	**1**	**1**	**0**

inputs having values of ones (i.e., C1 = C2 = '1'), or it might be predicted to value of '0' in the case of ANDing inputs having values of zeros (i.e., C1 = C2 = '0').

The coloured rows in Table 4.3 show the cases of carry propagation to the higher order (i.e., most significant) bits input in the previous sub-adder which are used for carry prediction (A[7], B[7]). These inputs combinations would result in wrong carry prediction, thus, an error (i.e., mismatch) between the two carry values C1 and C2 might occur.

The analysis also needs to consider the carry generate or propagate from the lower order bit location from the used bits for carry prediction. This bit location might either generate carry or propagate a carry signal from the least significant bits in the sub-adder.

The maximum error bound of the proposed design can be calculated as the addition of the probability of not correctly predicting the carry-in value to the current sub-adder, and the probability of error of the carry-out (i.e., carry propagation) from the previous sub-adder.

The maximum error bound can be analysed by the following equation (Eq. 4.4).

$$|E_i| = P_i \quad \times \quad W_i, \tag{4.4}$$

where (i) is the bit index number in the sub-adder, E_i is the expected magnitude of error due to bit (i), P_i is the probability of error of bit (i), and W_i is the weight of bit (i).

The probability of correct result at each bit location would be equal to $\frac{1}{2^i}$,then, the probability of error would be equal to $1-\frac{1}{2^i}$.

$$P_i = \left(1-\frac{1}{2^i}\right) \tag{4.5}$$

The bit weight (i) within the sub-adder is calculated by the following equation.

$$W_i = \frac{1}{2^{((MSB-i)+1)}} \tag{4.6}$$

$$|E_i| = \left(1-\frac{1}{2^n}\right) \times \frac{1}{2^{((MSB-i)+1)}} \tag{4.7}$$

where n is the bit-width or the index of the MSB bit in the previous sub-adder. The carry-out error probability of the previous sub-adder is as follows:

$$P(C_1) = |E_7| \quad = \left(1 - \frac{1}{2^7}\right) \times \frac{1}{2^1} = 0.496, \tag{4.8}$$

The probability of not correct carry-in value prediction equals to the probability of unhandled carry generate and carry propagate from the previous least significant bit.

$$P(C_2) = \left(\frac{2}{16}\right) + \left(\frac{\left(\frac{1}{2}\right)^4}{16}\right) = 0.125 + 0.004 = 0.129, \tag{4.9}$$

$$P(Error) = P(C_1 + C_2) = 0.496 + 0.129 = 0.625 \tag{4.10}$$

This result shows that at maximum, 62.5% of the resulted approximated outputs would have erroneous sum values (i.e., not equal to the exact sum value).

In our experiments of error levels, we have used the relative error distance (RED) metric which measures the magnitude difference value between the approximated and exact output result.

The following section of RED results simulation would show that the correct results of the proposed addition approximately equal to 41% of the outputs space. Hence, this might confirm the derived error bound of the proposed approximate adder design.

4.5.2 *Error Metrics Evaluation*

In this part, we explore the behaviour of the proposed design regarding three error metrics which are the relative error distance (RED), the mean relative error distance (MRED) and the cumulative probability distribution (CPD) of error. We targeted these error metrics as they are considered an important evaluation of approximate design as mentioned in section 2.5 of approximate error metrics in chapter 2.

Relative error distance (RED) distribution analysis has been done for each design over different error distance values range. RED simply measures how far the significance of resulted output's error when compared to the exact output of the conventional adder. RED analysis has been made for both proposed design versions (Proposed and Proposed_Accurate) and the ACA adder design when compared to the correct outputs of exact adder. However, despite the simplicity of this measurement, it would show the effect (i.e., error magnitude) of the proposed design stages, regarding the final accuracy of the outputs.

The following equation shows the arithmetic expression of the RED value.

$$RED = \frac{|Exact\ output - Approximated\ output|}{Exact\ output}, \quad (4.11)$$

Fig. 4.11 presents the case of designs (proposed design version versus ACA design), yet, without any correction stages. From the resulted analysis, it can be shown that the proposed design has an acceptable range of outputs with no errors (more than 40% of the tested space), and approximately 55% with a very limited

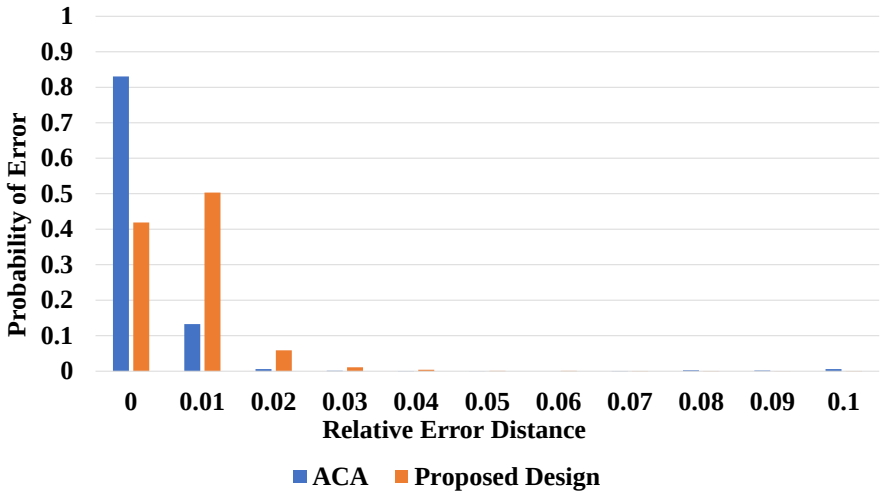


Figure 4.11: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (No correction stages).

magnitude of error (50% with 99% and 5% with 98% of accuracy). These levels of erroneous outputs can be referred to the number of misprediction of the propagated carry between the adjacent sub-adders during the addition process. Regarding the last 5% of the tested inputs space, the proposed design behaves the same as ACA, which lies on different RED values. However, the proposed design presents higher percentages of outputs with a smaller magnitude of error values.

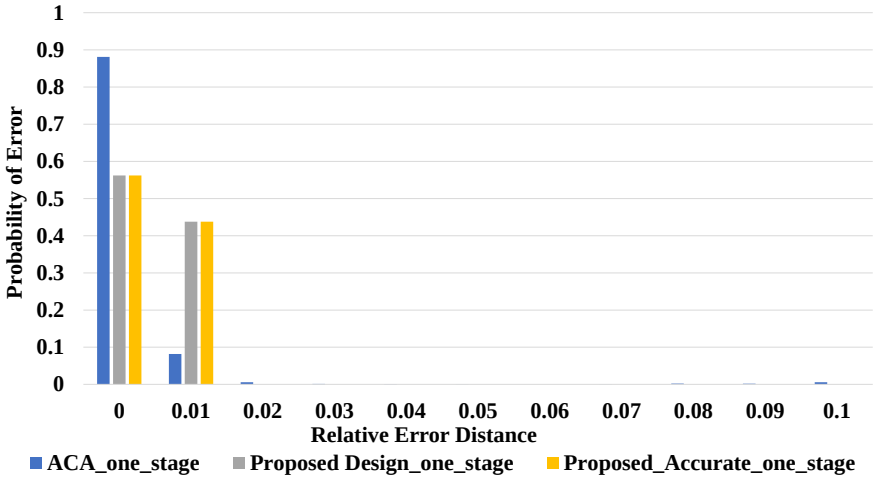


Figure 4.12: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (One correction stage).

Fig. 4.12 presents the error analysis of designs with one correction stage. It can be noticed that our design versions (Proposed and Proposed_Accurate) have increased the ratio of the correct results, and further, more stability in terms of RED values which are started to be limited strictly between (RED = 0 ,57%) and (RED = 0.01, 43%) values, in contrast to ACA which still shows different values of RED. This can be explained because the correction process uses the significance-driven principle that starts correcting the most significant (i.e., higher order) bits of the adder due to their significant impact on the final sum result. As a result, very small magnitudes of error are expected.

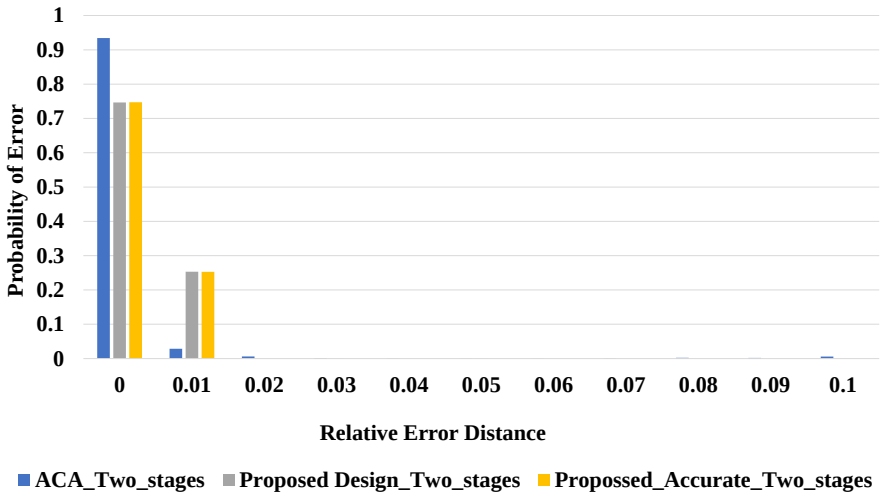


Figure 4.13: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (Two correction stage).

For the case of two correction stages in Fig. 4.13, the proposed design two versions and ACA have improved the ratio of the fully correct output values; however, our design versions still generally show more acceptable results as they continue to be limited to 100% and 99% of accuracy, in contrast to ACA that still owns scattered values of RED.

Finally, at Fig. 4.14, the case of three (full) active correction stages (worst case of accuracy level) is presented. It is obviously

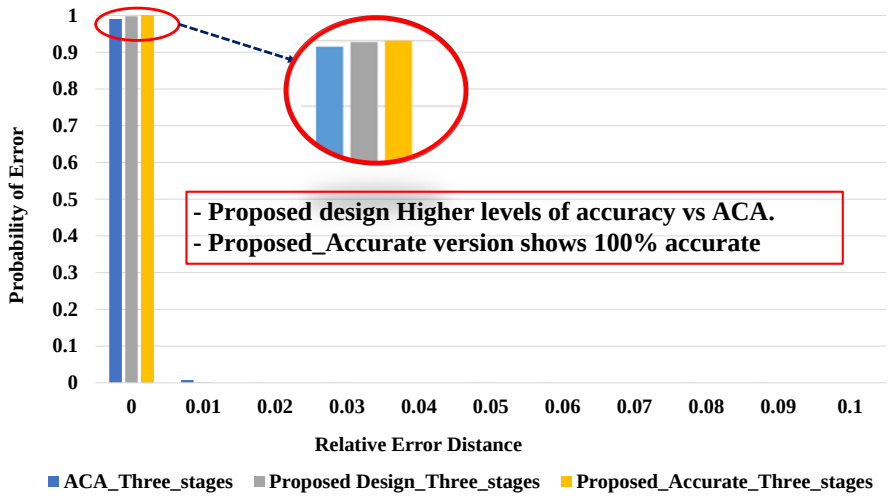


Figure 4.14: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (Three correction stage).

shown that the behaviour of the proposed design versions shows higher levels of accurate results compared to ACA design, and further, guarantees of 100% correct results in the Proposed_Accurate version (i.e., when carry-out values of correction stages are in concern).

These results of accuracy levels of the proposed design versions in the previous figures are referred to the effective carry-in values prediction and the error correction structure. It can be clearly noticed that the significance-driven implementation of the correction stages would guarantee the fast convergence to the exact addition results and furthermore, the small magnitude of the output error. For the case of Proposed_Accurate design version, all the carry propagation signals during error detection and correction would be handled through all the correction stage. As a result, the full accuracy at the last correction stage would be guaranteed.

Fig. 4.15 presents a comparison of the mean relative error distance (MRED) values, which would show the mean value of errors within the test space or input vectors at each correction stage.

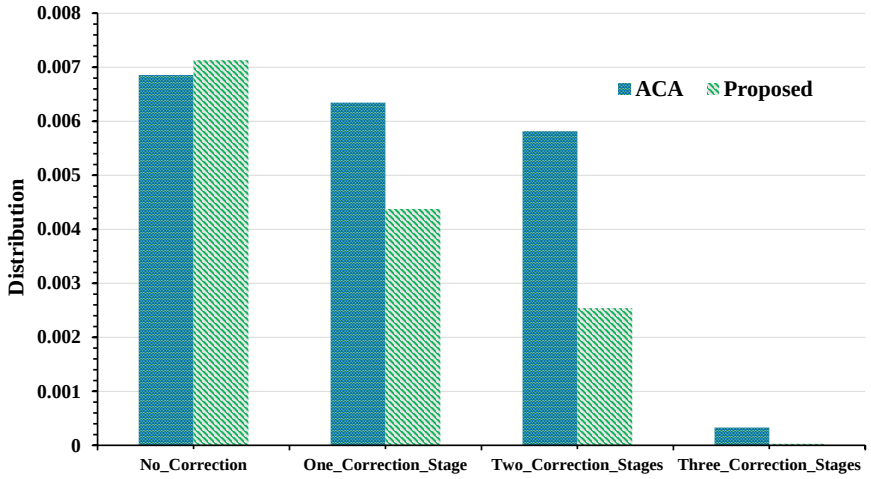


Figure 4.15: The 32-Bit adder designs MRED values comparison through different correction stages.

It can be noticed that at the approximate addition stage with no error detection and correction, the proposed design shows a limited higher level regarding the ACA design. This is due to the fact that the proposed design does not have overlapped parts within each sub-adder used for intense carry speculation, thus, introduces lower accuracy. However, by using the first and second correction stages, the proposed design MRED values are drastically decreased showing better accuracy and consistency. This can be referred to the significance-driven structure of the proposed correction stages, which guarantee the fast convergence to the exact sum values and limit the final outputs to low magnitude values of errors early.

For the final third correction stage, the proposed design provides MRED value equals to zero, in contrast to the ACA design that still shows a small mean error value. Hence, this would confirm that the proposed design presents a full handling of errors with optimum final output accuracy.

Fig. 4.16 presents the cumulative probability distribution (CPD) for relative error distance (RED) levels through approximate addition and error correction stages. This analysis describes the change

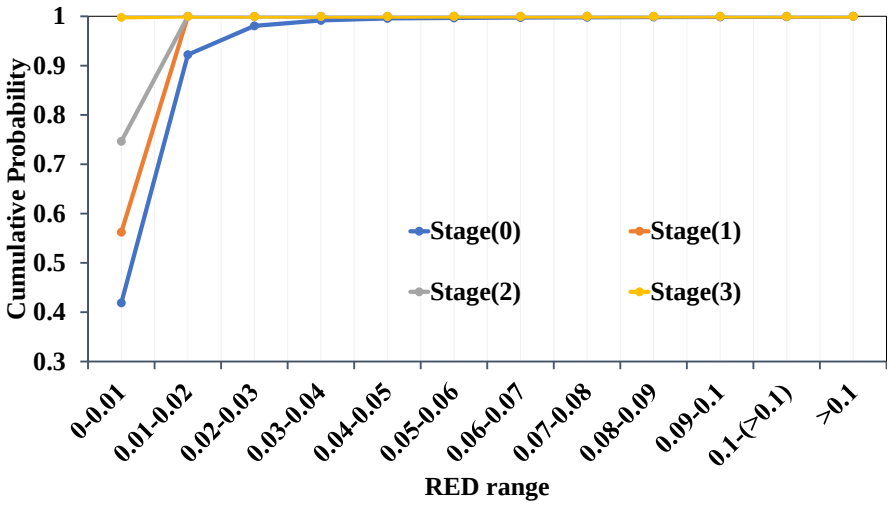


Figure 4.16: The cumulative probability distribution (CPD) for the error through different correction stages.

of percentages of the output space regarding the RED values. Additionally, it shows the speed of the resulted outputs become closer to the exact addition results. In other words, it would explore how each correction stage might affect the total output accuracy.

From Fig. 4.16, it can be noticed that the cumulative value at approximate addition stage without any error correction shows a quick move to the RED range ($0.01 \leq RED < 0.02$). This means that approximately 90% of the output space has RED values of($0 \leq RED < 0.02$). The remaining 10% percentage of the output space would be covered very quickly within the RED range($0.03 \leq RED < 0.04$) to reach '1' (i.e., 100%) of the output space.

For the first and second correction stages, both of them show better and approximately same cumulative behaviour of the sharp jump the RED range ($0.01 \leq RED < 0.02$). However, they become limited to this RED range and entirely cover all the output space (i.e., reach 100% of the outputs number). The only difference between these correction stages is the percentage of the resulted exact values. It is clearly shown that as the number of active correc-

tion stages increased, the portion of the correct results (i.e., RED = 0) is increased, then, speeding up the step to cover all the resulted output space. Remarkably, the last correction stage presents the better behaviour of accuracy with approximately 100% of exact results of the whole output space.

4.6 LARGE BIT WIDTH ADDERS EVALUATION

For design scalability checking, a further hardware evaluation has been implemented for different adder designs (ACA and Proposed versions) with larger bit widths (64-bits,128-bits and 256-bits). The values from each design with full correction stage architecture (i.e. using three correction stages) are presented in the following figures.

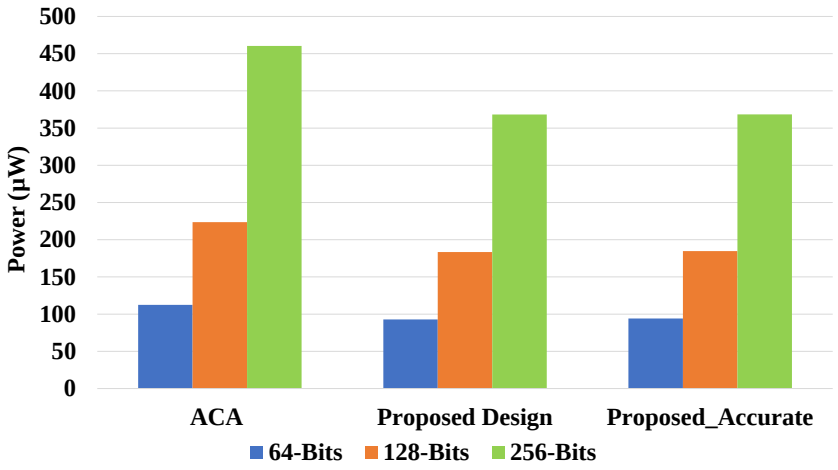


Figure 4.17: ACA vs. proposed large adder designs dynamic power(μW) comparison.

It can be noticed that the proposed design versions (Proposed and Proposed_Accurate) keep the reduction ratio values in terms of dynamic power in Fig. 4.17 , leakage power in Fig. 4.18, and the area in Fig. 4.19. Furthermore, these values start to increase positively as the bit width of the adder becomes larger. This is

due to the increased number of the overlapped sub-adders in the ACA design, thus, increasing the total area and power consumption. However, our proposed design continues to limit the resulted number of segmented sub-adders, and hence, shows better design parameters overhead.

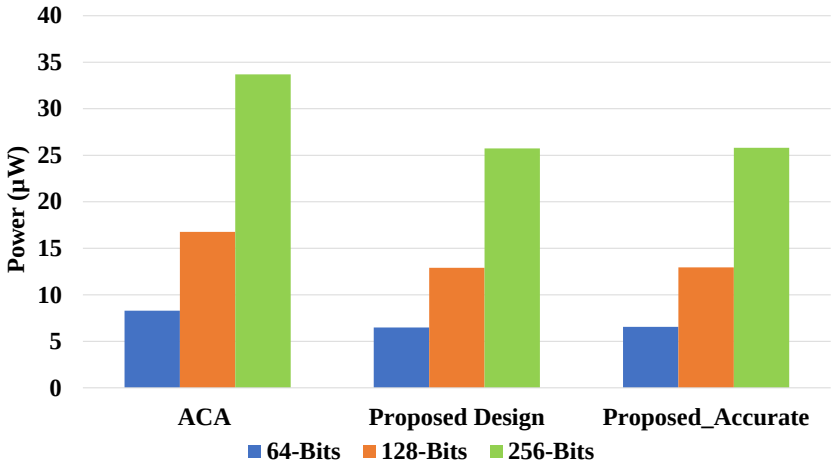


Figure 4.18: ACA vs. proposed large adder designs leakage power (μW) comparison.

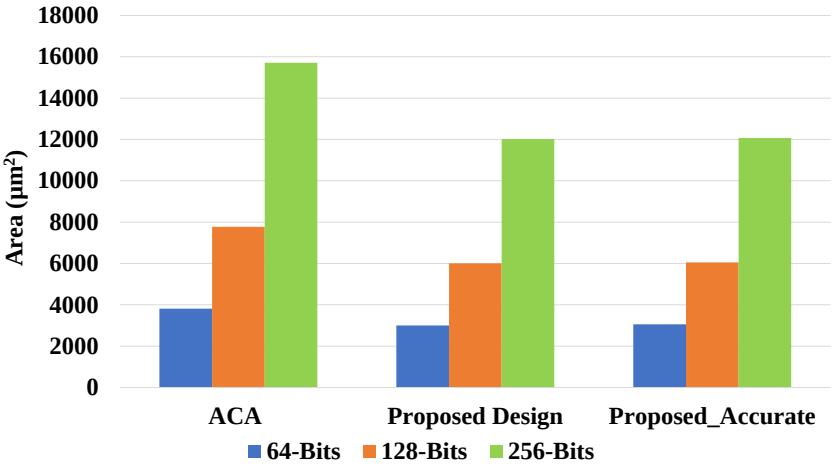


Figure 4.19: ACA vs. proposed large adder designs area (μm^2) comparison.

Fig. 4.20 apparently shows that the delay degradation of the proposed design version compared to ACA design becomes smaller as larger bit-width adders are in use. This might be referred to the

difference of the resulted design area between the proposed design and the ACA design. Thus, the negative ratio of the delay values would be slightly decreased.

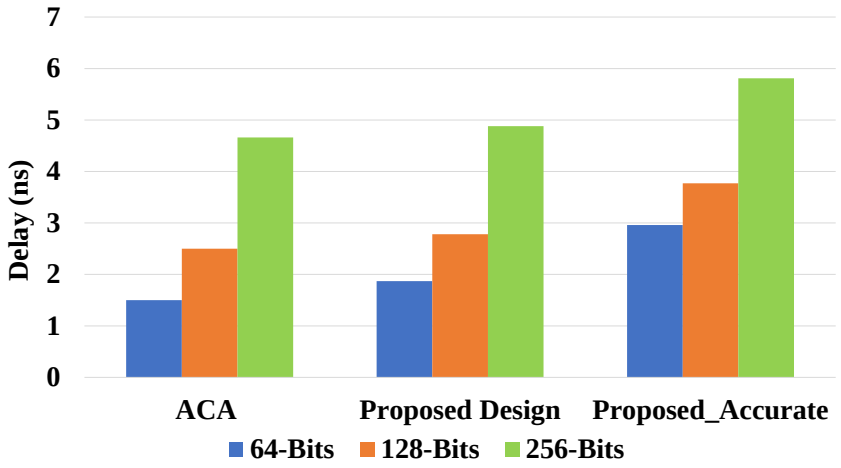


Figure 4.20: ACA vs. proposed large adder designs delay (ns) comparison.

Remarkably, the results in Fig. 4.21 illustrate that the percentage of the large adder designs reduction values which can clearly confirm the scalability advantage of the proposed design versions. Compared to the ACA design, the positive reduction values of the proposed design are referred to the lower overhead of design parameters such as area, dynamic and leakage power as explored in the previous hardware evaluation figures.

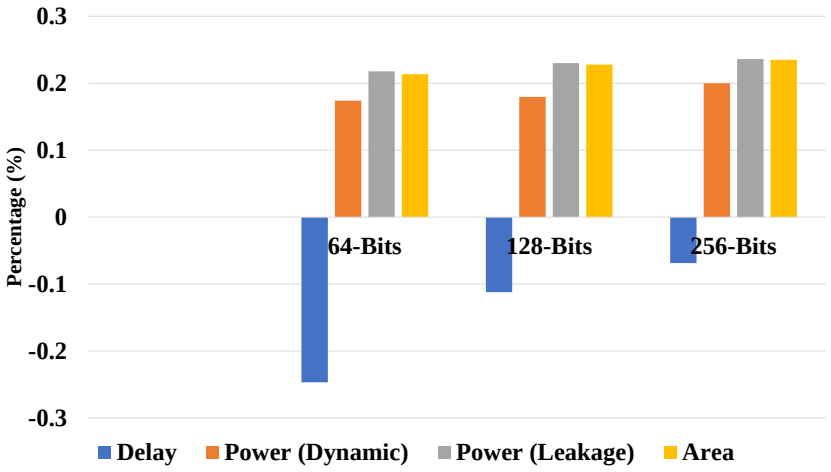


Figure 4.21: ACA vs. proposed large adder designs reduction ratio values.

Furthermore, although the exist of negative reduction ratios of delay values, it can be noticed that they become limited while increasing the bit width of the adder. Hence, it can be concluded that the proposed design versions would be adaptive for use in very large bit width adders with acceptable design overhead.

4.7 FURTHER HARDWARE COMPARISON

Table 4.4 shows an extended comparison of the proposed design two versions (Proposed and Proposed_Accurate) regarding previous efforts. The comparison values confirm the privileges of the proposed design in terms of all design parameters such as delay, power and area. For the delay values, the proposed design versions show a higher speed than Accurus [8] design and has an acceptable difference when compared to ACA [37] design versions.

Moreover, the proposed design versions show the best behaviour regarding the power (dynamic and leakage) and area when compared to all designs in the table except the accurate ripple carry adder (RCA) design. These results of the proposed design versions have been confirmed by the delay-power-product PDP values, which show smaller numbers when compared to ACA and Accurus designs. However, it is interesting to notice that the proposed design version without a correction mode might be considered a very high-speed version of RCA, but with an acceptable output quality and small design overhead.

4.8 IMAGE PROCESSING APPLICATION

To evaluate the proposed design in a real-world implementation, we exploit a key block in the image processing application known

Table 4.4: Hardware metrics comparison of the proposed adder designs and several previous efforts

Design	Max Correction Stages	Delay (ns)	Dynamic Power (μW)	PDP	Leakage Power (μW)	Area (μm^2)
ACA* [37]	No correction	0.67	22.2436	14.9	2.4038	1001.952
ACA [37]	Three	1	56.1614	56.16	4.1278	1855.728
Accurus [8]	Three	1.13	56.916	64.31	4.1796	1895.712
RCA	Accurate	2.22	18.1273	40.24	2.0263	702.816
Proposed*	No correction	0.7	18.9542	13.26	2.0221	793.408
Proposed	Three	1.15	46.369	53.32	3.3348	1519.392
Proposed_Accurate	Three	2.04	47.5747	97.05	3.3866	1569.568

as Gaussian blur image filter, which is already defined in section 2.10.4.

For implementation analysis, a general Matlab test bench was proposed to apply the Gaussian blur image filter test. The test bench checks the actual behaviour of (20-bits) Proposed_Accurate adder design version during multiple correction stages. The peak signal to noise ratio (PSNR) is used to measure the quality of the output images after applying the Gaussian blur filter and comparing the resulted image quality to the optimum blurred image of the conventional circuit of the Gaussian blur filter.

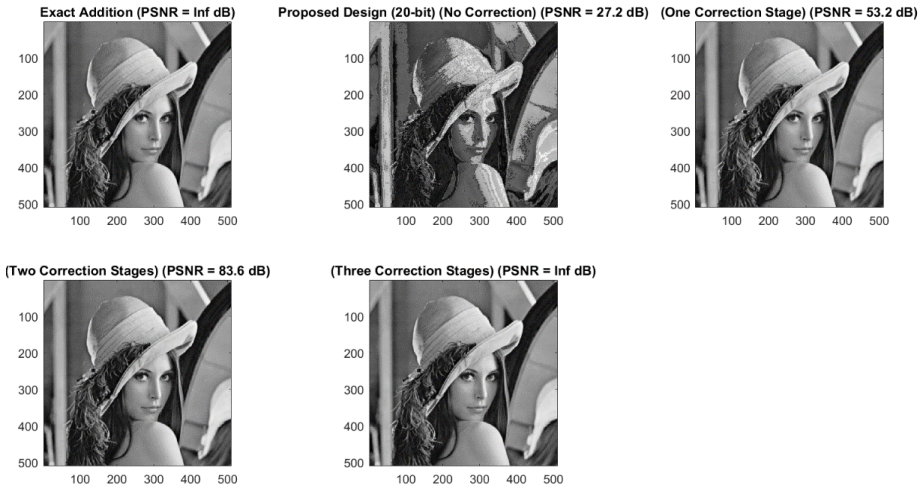


Figure 4.22: Gaussian blur image filter test.

Based on the resulted images of approximate addition and different correction stages, the PSNR results in Fig. 4.22 confirm the advantage of this design version. In detail, for the approximate addition stage (without any correction), an acceptable value of PSNR is provided with (27.2. dB). Further, for the correction stages, the results show high PSNR magnitude values, especially when starting the error recovery at the first correction stage with more than (53 dB). Remarkably, while using two correction stages, the resulted PSNR value has presented a well-noticed jump and reached

(83.6 dB). For the case when the implemented design operates at the full accurate mode (i.e., three correction stages), it guarantees the same accuracy as the original picture which is implemented with exact computations.

The appearance of low PSNR value of the proposed design without correction stages might be referred to the low level of exact outputs as described previously in Fig. 4.5. This level of erroneous addition might impact the Gaussian kernels calculation in the used image filter, as a result, an image distortion can be expected. However, this low level of PSNR value can be considered an attractive choice for adder design in some application like the biomedical applications, which are generally interested in high speed, very low power and acceptable outputs quality [96, 2, 1].

4.9 SUMMARY

In this chapter, a new segmentation technique has been proposed for designing configurable-accuracy approximate adder with low power and area requirements. The concept of carry propagation kill signal was used to introduce a new bit location, which was exploited for both dividing conventional adder into a number of sub-blocks and holding the real carry of each sub-adder.

This new architecture of dividing sub-adders was augmented with a lightweight carry-in prediction technique by using the result of AND-ing the MSB bits of the previous sub-adder. Further, simple error (carry propagation) detection technique was proposed by using XOR gate between successive sub-adders to check the equality value of the predicted carry-in and the real carry at the new bit locations. For error correction, a significance-driven multi-stage structure was used, while considering the carry-out of each

active stage. Thus, this would guarantee full accuracy at the final correction stage.

While evaluating the proposed design in this chapter, the results have presented average reduction ratios of (16%), (17.2%) and (18.6%) for dynamic power, leakage power and area respectively. On the other hand, for error analysis, the results showed fast convergence to exact results at premier correction stages and increased stability of output accuracy levels between (99% and 100%) through all correction stages.

Further, the proposed design feasibility was confirmed by a real-time implementation in an image processing application, which resulted in high PSNR values of (27, 53 and 83 dB) for the approximate addition and the two premier correction stages respectively. Moreover, the optimum PSNR quality was provided as an exact filter circuit when the proposed design was working with full correction stages.

The following chapter continues to explore more chances to mitigate the design overhead of the configurable accuracy approximate adder design. For this target, an approximate adder design is proposed with simple logic gates in the approximate sum stage. The operation of the used logic gates would compensate the conventional arithmetic addition. Further, the proposed design will not have any special augmented error detection and correction (EDC) circuit. Conversely, the error recovery technique would use multi-stages of short bit-width exact adders to overcome and control the general quality of the outputs. The main advantage of the proposed design in the following chapter is that it would approximately eliminate the design overhead and the frequent processing of the error recovery circuit, yet, it would show the optimum output quality at its final recovery stage similar to an exact adder.

5

GENERAL QUALITY-CONTROL APPROXIMATE ADDER WITH LOW DESIGN OVERHEAD

5.1 INTRODUCTION

Based on the general aim of this work, this chapter continues to explore more chances to mitigate or even eliminate the augmented error recovery circuit overhead of the approximated adder design, yet, while preserving acceptable levels of the output accuracy. In the previous two chapters (chapter 3 and chapter 4), design improvements have been made regarding the error recovery process and sub-adders segmentation of the approximate (speculative) adders. However, in this effort, we propose a low-power approximate adder that combines the approximate logic addition design with general output quality controlling during run-time.

In the state of the art literature, approximate adder designs such as lower-part-OR adder (LOA) [50] and error tolerant adder (ETA)design series [106, 105, 104, 107] reduce the maximum carry propagation by dividing the total number of bits of adder into accurate and inaccurate parts. In such architecture, the exact addition is used for the precise part and the approximated addition methods such as the simple logic addition is implemented in the inaccurate part. However, several challenges have been observed regarding these approximated designs, such as, the high output error rates without proposing an error recovery method to keep an accepted level of accuracy. On the other hand, approximate

(speculative) adder designs such as VLSA [91] and ACA [37] used an error recovery circuit to avoid severe outputs quality degradation. However, these efforts still have the common challenge of additional design overhead in terms of error recovery delay, power, and area, and moreover, they have not reached the full accuracy at the final correction stages.

The proposed design in this chapter presents three contributions regarding a hybrid adder design of approximated logic addition and general output accuracy controlling. The proposed contributions introduce an energy-efficient approximate adder which replaces conventional arithmetic addition by a simple logic operations. Additionally, the proposed design is augmented with a general accuracy recovery stages that exploit a short length exact adder at each correction stage (instead of incrementor circuits in the previous two chapters). The correction stages maintain the significance-driven structure to ensure general quality controlling and early correction of higher-magnitude errors to achieve fast convergence to the exact addition results.

Compared to other approximate adders, the proposed design has drastically reduced the overhead of approximate addition and error recovery process. In this work, a (32-bit) approximate adder is designed in Verilog and synthesized using the Synopsys design compiler. Our post-synthesis results showed significant average reductions of 70% and 62% for dynamic and leakage power respectively, and 61% in the silicon area for the design with full correction stages. Delay values show positive reduction ratios at the approximated addition stage and the first correction stage. However, an acceptable degradation at the remaining stages happened because the proposed adder starts to work as a conventional exact adder. The implementation test of the image processing application con-

firms our results with high PSNR values of (50.6 dB and 81.2 dB) for the second and third correction stages, respectively, and the optimum PSNR value while activating all correction stages.

5.2 PROPOSED DESIGN

The proposed design is depicted in Fig.5.1, uses simple logic (OR) gates to approximate the full arithmetic addition operation with high speed and very low design overhead. Previous efforts such as the lower-part-OR (LOA) approximate adder design [50] used the simple logic OR gates to result directly the sum bits in the lower 'carry free' inaccurate part of the adder. However, in this work, the proposed design uses logic OR gate to compensate the full conventional arithmetic addition operation.

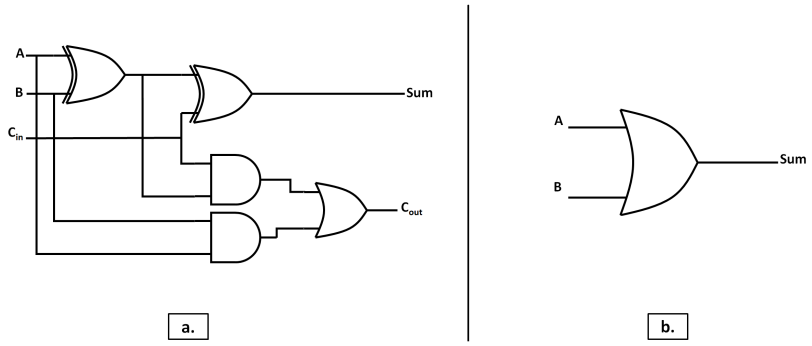


Figure 5.1: The conventional full adder circuit in (a.) versus the proposed approximate OR gates addition operation in (b.).

Fig. 5.1 illustrates the proposed approximated addition, where the logic OR gates are used to compensate the exact full adder arithmetic operations. The following points summarize the proposed design's main parts.

- We OR the corresponding bits of the two inputs,(i.e., Si = Ai | Bi), to get the approximated Sum result.

- The proposed approximated adder can be usefully used in a low accuracy level and extremely low power applications. Moreover, it has a high speed and a very small area (limited to the size of the used OR gates).

5.3 QUALITY CONTROL CIRCUIT

The proposed accuracy recovery circuit is shown in Fig. 5.2, in which the correction process is implied by an accurate adder of short bits length. However, the bit width of correction adders can be configurable and based on the application design requirements (i.e., the bit width of adders might be different; yet, their total bit width should be equal to the original bits' number of the approximated adder).

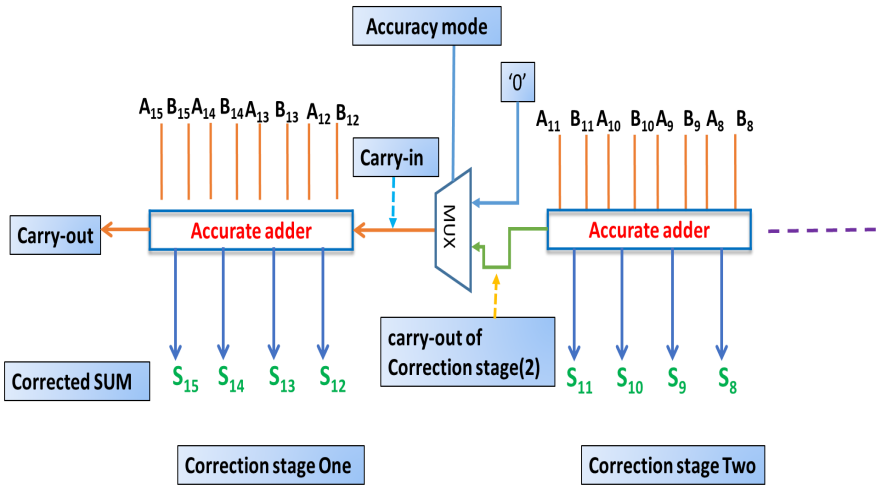


Figure 5.2: The proposed accuracy recovery circuit.

In the proposed circuit in Fig. 5.2, each short length exact adder used for correction has its inputs from the addend inputs, which are the same corresponding inputs to the approximated adder. Further, the carry-in to each correction adder should be multiplexed between two choices, the carry-out from the previous correction

adder (stage) if the two successive stages are active in parallel, or a truncated carry-in value equals to '0'. The multiplexed carry-in value depends on the selected accuracy mode, which specifies the number of the active correction stages.

5.4 ERROR CORRECTION STAGES

The proposed significance-driven structure of the accuracy recovery circuit is depicted in Fig.5.3. The proposed structure would guarantee that the results of the most significant approximated sum bits are first corrected. As a result, the correction of high magnitude errors would take place at the premier correction stages, and then, achieve a fast convergence to the exact sum value.

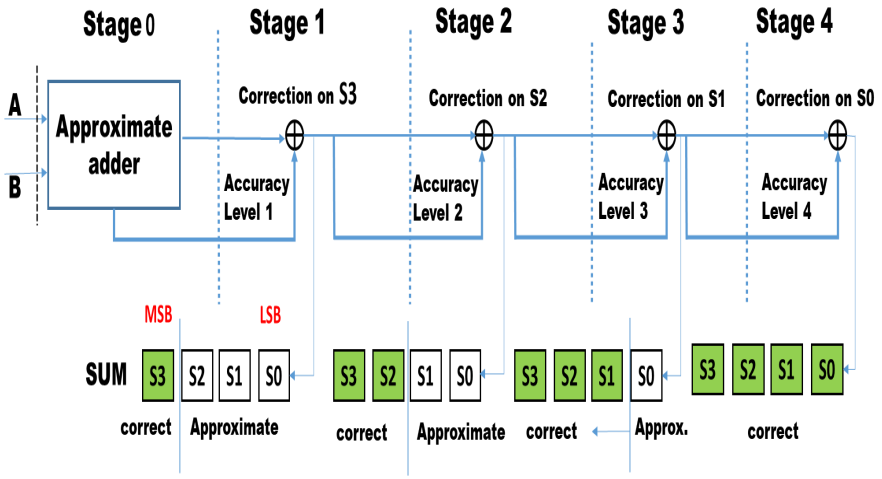


Figure 5.3: Significance-driven error correction stages (32-bit adder example).

This significance-driven structure is similar to what we have used for error recovery stages of the proposed designs in the previous chapters 3 and 4. However, they have used incrementors as correction circuits to correct the erroneous sum values with more delay, power and area overheads. Conversely, the proposed design in this chapter does not imply any error detection process

and uses short length exact adders to overcome the general quality degrading. Thus, it would incur a limited design overhead, which approximately equals to a conventional adder.

Furthermore, the proposed design would follow the procedure of detecting the carry-out value of the active correction stages, which has been used in the Proposed_Accurate design version in the previous chapters 3 and 4, Thus, guaranteeing full output accuracy at the final correction stage.

Based on Fig. 5.3, the main points of the multi correction stages can be summarized as follows.

- Stage 0 has the approximated sum result (without any correction).
- The correction stage (short-length exact adder) will result in the accurate sum part (coloured green).
- S0 will be corrected at the final correction stage.

5.5 NUMERICAL EXAMPLE

In this section, a (32-bit) numerical addition example is presented. The detailed example shows the main methodology of adder division, approximate addition and accuracy recovery techniques of the proposed design.

From Fig. 5.4, the 'Appx Sum' stage presents 32-bit inputs approximate addition using one logic OR gate at each bit location. At this first stage, the brown coloured bit locations have erroneous sum values, when compared to the exact result. Stage 1 is the first correction or quality controlling stage. In this stage, the exact adder with 8-bits width starts to correct the eight higher order (i.e., most significant) bits of the result of the previous approximated

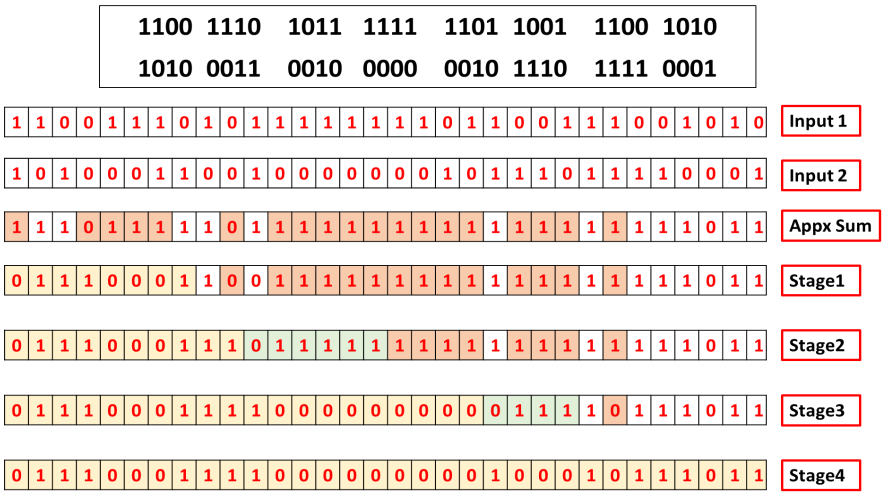


Figure 5.4: Proposed design numerical example of (32-bit) inputs addition using OR logic gates and multi-stage correction by using (8-bit) exact adder at each stage.

stage 'Appx Sum'. The yellow coloured bit locations in this stage present the correct sum values when compared to the exact result.

Stage 2 continues to correct the successive eight higher order (i.e., most significant) bits of the result of the approximated stage 'Appx Sum'. As a result, the number of bits with exact values is increased. However, the green coloured bit locations have erroneous values due to the truncated carry-in zero '0' value to stage 2. This might be referred to not handling the carry propagation with a value of '1' at this stage. Stage number 3 presents the same behaviour as the previous stages with exact sum results and a limited number of erroneous bits.

Finally, stage 4 presents the case when the proposed design starts to work as a 32-bits exact adder with full accurate sum result. At this final stage, all the approximate addition results from 'Appx Sum' are neglected. Thus, the proposed adder would provide the optimum quality of the final outputs.

5.6 DESIGN TRADE-OFFS

To demonstrate the proposed approach, we implemented the design in Verilog to apply (32-bit) adder design with different correction stages. For the correction circuit of the proposed design, we used (8-bit) ripple carry adder (RCA) at each correction stage. These codes were synthesized and implemented using two different off-the-shelf tools. Firstly, Modelsim was used for compiling the Verilog codes and running the associated test benches. Secondly, Synopsys design compiler was utilized for synthesizing all versions of the proposed adder when mapping the circuits to the UMC (Faraday 90nm) technology and evaluating for power, delay and area. Monte Carlo simulation was used for error analysis.

We compare our proposed design against the design effort of accuracy-configurable adder (ACA) [37] in terms of hardware design parameters such as power, area and delay.

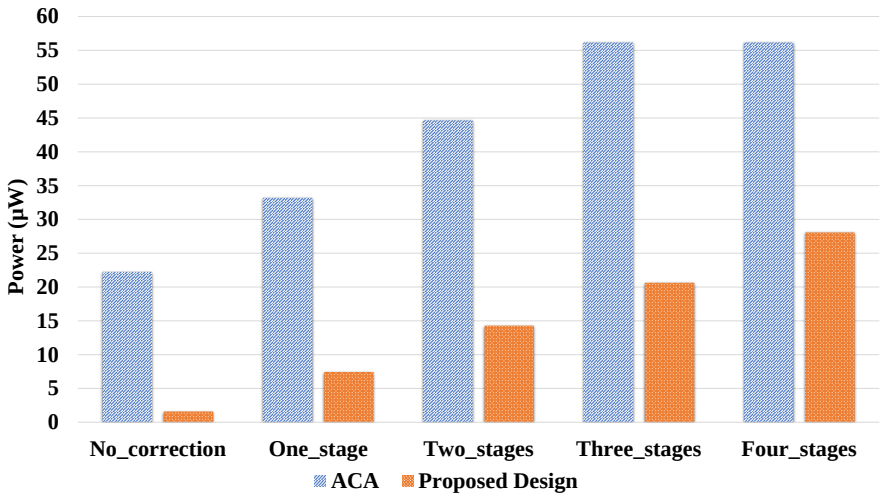


Figure 5.5: 32-Bit proposed design versions vs. ACA dynamic power (μW) comparison.

Based on Figs. 5.5, 5.6 and 5.7 , it can be observed that our proposed design behaves better than the (ACA) design in terms of power (dynamic and leakage), and area. The proposed design

incurs a smaller area overhead due to using OR logic gates number equals to the number of one addend input vector, in addition to limiting the correction circuit overhead by avoiding frequent error detection checks and error correction processes. Conversely, it concerns about general output quality controlling during run-time. As a result, the general design incurs a smaller area and lower levels of power consumption.

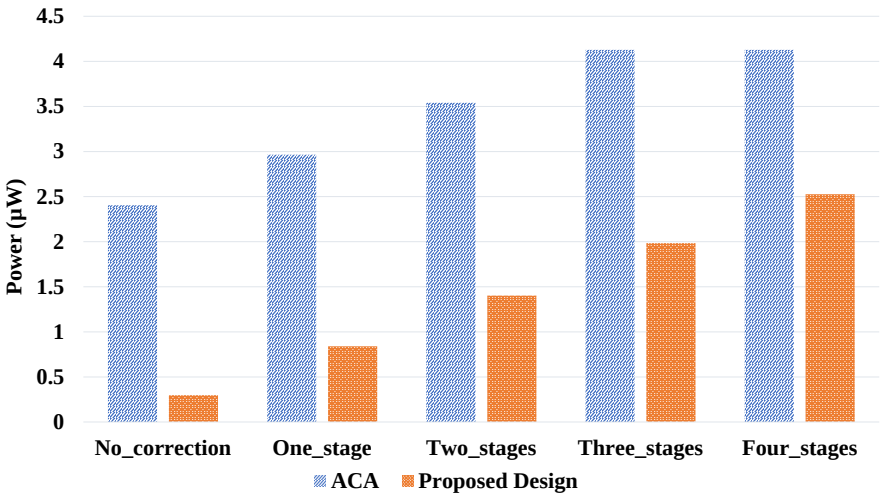


Figure 5.6: 32-Bit proposed design versions vs. ACA leakage power (μW) comparison.

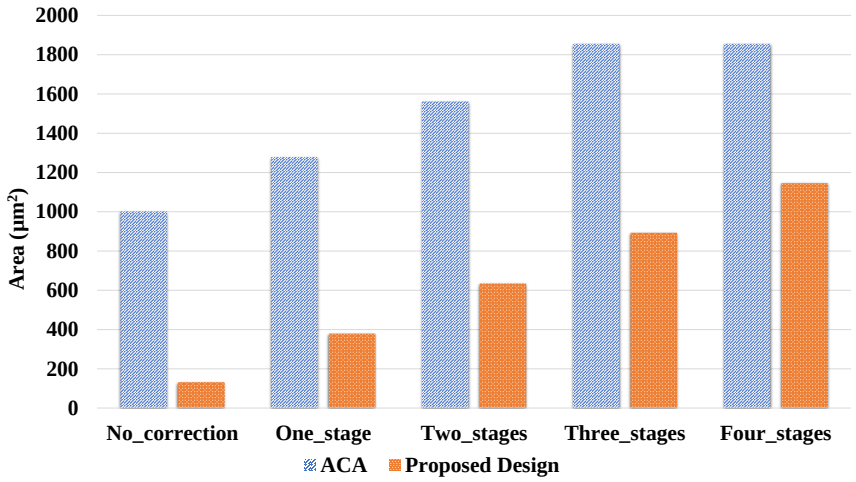


Figure 5.7: 32-Bit proposed design versions vs. ACA area (μm^2) comparison.

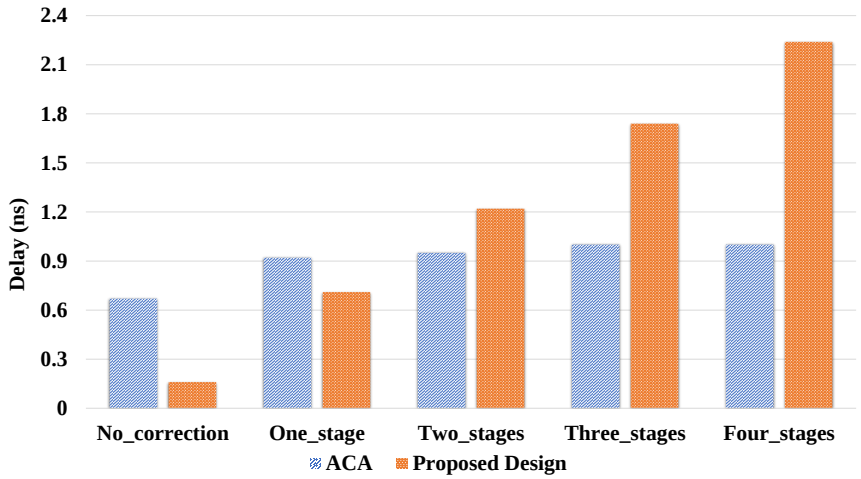


Figure 5.8: 32-Bit proposed design versions vs. ACA delay (ns) comparison.

For the delay values in Fig. 5.8, the proposed design shows higher speed values for approximated addition with no correction overhead and for the addition with one correction stage when compared to the ACA design. However, for the remaining correction stages, the delay presents higher values as the proposed design starts to work as an exact conventional adder.

To analyse the general design enhancement against the ACA design, the reduction ratios of design parameters of the proposed design through approximated addition and all four correction stages are presented in Fig. 5.9. The average reduction values of the proposed design stages result in 70.3% for dynamic power, 62% for leakage power, 61.3% for area and -25% for the delay. In Fig. 5.9, all the positive results are referred to using the simple limited count of logic gates instead of conventional full adders for addition, and further, not using any augmented error detection and correction EDC circuit to handle the erroneous outputs. Instead, the proposed design uses a short bit width exact adder in each correction stage in order to control the general output quality levels. This in turn

would explain the results of positive and negative values of delay reduction ratios analysis through the design stages.

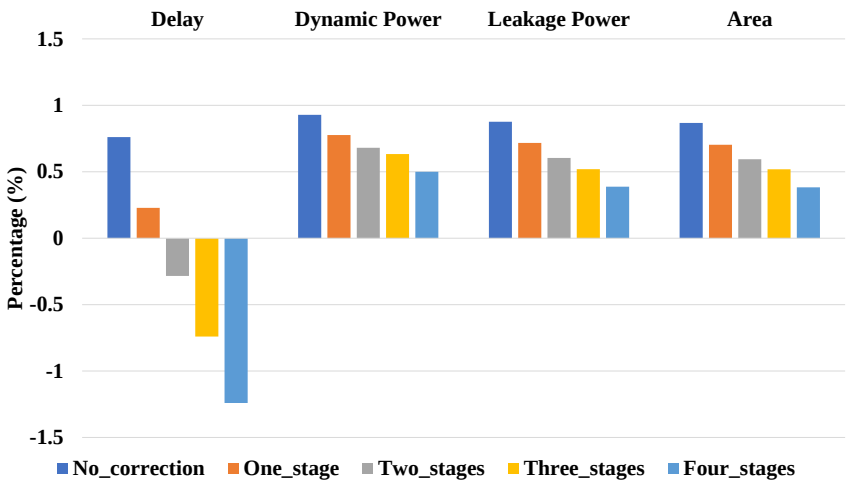


Figure 5.9: 32-Bit proposed design vs. ACA reduction ratio values.

For the positive reduction delay values in the approximated addition stage and the first correction stage, it can be explained that the delay would equal to the processing time of a simple logic gate and a short bit width exact adder. On the other hand, for the remaining correction stages (i.e., second, third and the fourth final stage), they show negative delay reduction ratios since the proposed adder starts gradually to work as a conventional adder with its conventional addition process time. However, the proposed design with its full recovery stages still shows lower values of design parameters such as power and area when compared to the approximated design of ACA.

5.7 ERROR ANALYSIS

In this section, we present the second part of the proposed design evaluation by analysing the expected error levels of its resulted outputs and several detailed approximate error metrics.

5.7.1 *Error Probability Analysis*

In the first part of this section, the probability of the maximum error percentage (i.e., the error bound) of the proposed design results is examined within the total space of the outputs. The analysis would mainly explore the behaviour of error distribution while using the logic OR gates for addition in both small and large bit-width adders instead of the conventional addition process by using exact full adders' blocks.

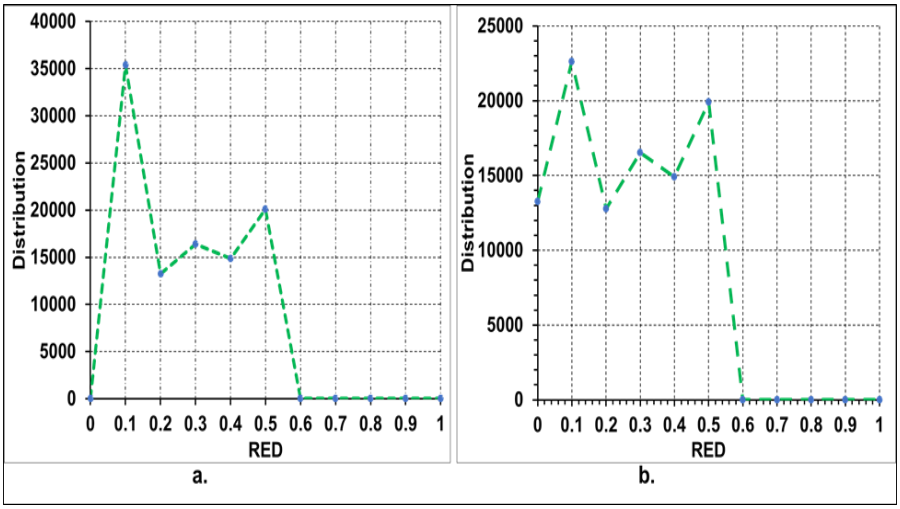


Figure 5.10: Monte Carlo analysis of logic OR addition implementation in(a) 32-bits inputs, and (b) 8-bits inputs.

To analyse the sum results' error distribution, we used Monte Carlo to check the relative error distance (RED) distribution of the addition operation results. Monte Carlo method applies random variables as inputs to the targeted function to check the general behaviour of the resulted outputs. In our experiments, the simulations use repetitions of random variables within the range of examined adders' bit-widths. Fig. 5.10 presents the RED distributions of the addition results of two 32-bits numbers in (a.), and for the addition of two 8-bits numbers in (b.).

The results show two main points; the first one is that error rate (i.e., RED>0) of the resulted outputs is increased as the bit-width of the adder is getting larger. The second point is that the maximum RED value or the error bound is located at 50%, with percentage of 20% for both (a.) and (b.) tests. This result means that while using OR gates for addition, 20% of the resulted sum values would have the half value of the exact addition results for the same inputs. However, it can be noticed that although the error rate is increased for the large bit-width addition in (a.), the percentage of the resulted outputs which has a limited RED value of 10% is increased to approximately 35% of the total output space.

The following section has the RED distribution analysis of the proposed approximate (32-bits) adder with different stages of output quality controlling. The experiments results would confirm the results of Monte Carlo simulations with a maximum RED value of 50% and the majority of outputs has a small RED value of 10% compared to the exact result.

5.7.2 *Error Metrics Evaluation*

In this part, we explore the behaviour of the proposed design regarding three error metrics which are relative error distance (RED), mean relative error distance (MRED) and cumulative probability distribution (CPD) of error.

Relative error distance (RED) distribution analysis has been done for each design over different error distance values range. RED simply measures how far the significance of resulted output error when compared to the exact output of the conventional adder. RED distribution analysis has been made for both proposed design and the ACA adder design [37] when compared to the correct outputs

of the exact adder. This test would show the effect of the proposed design stages regarding the final quality of the outputs.

The following equation shows the arithmetic expression of the RED value.

$$RED = \frac{|Exact\ output - Approximated\ output|}{Exact\ output}, \quad (5.1)$$

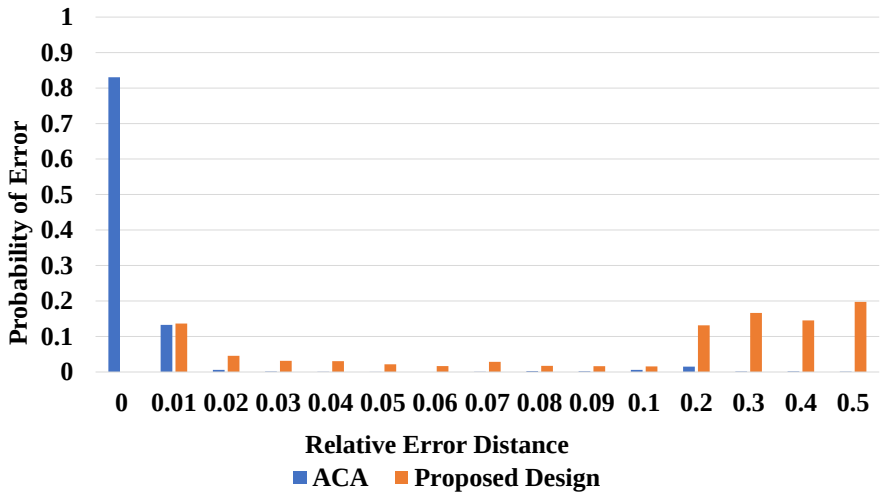


Figure 5.11: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (No correction stages).

Fig. 5.11 presents the case of designs (Proposed and ACA) without any correction stages. From the results, it can be shown that the proposed design has a high RED rate (20% of the test space has a maximum error of 50% far away from the exact result). This is due to the fact of compensating the full conventional arithmetic bit addition operation with just one logic gate. However, for the low accuracy level, this can be considered an acceptable error rate especially when compared to the large improvements in the design parameters like delay, power and area.

On the other hand, at one correction stage in Fig. 5.12, it can be noticed that the proposed design made a remarkable improvement

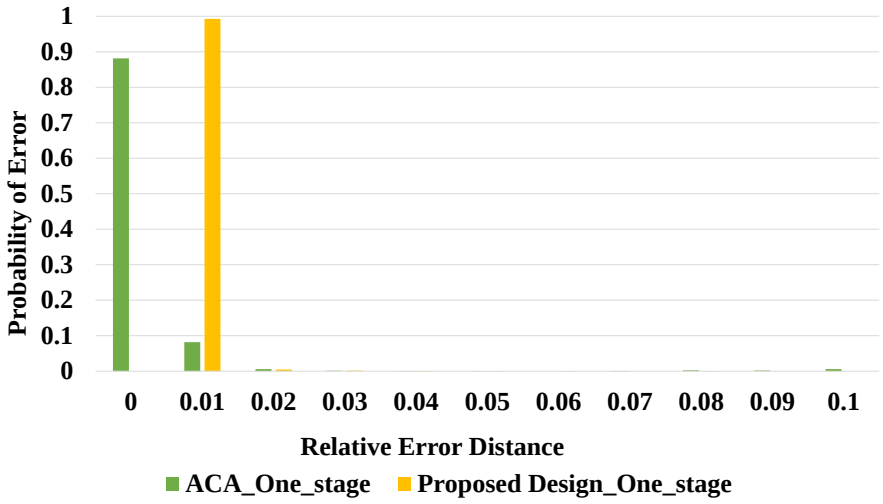


Figure 5.12: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (One correction stage).

in terms of RED values, in which 99% of the test space presents (0.01) RED value, implying that the output result is far away from the exact result by just 1%.

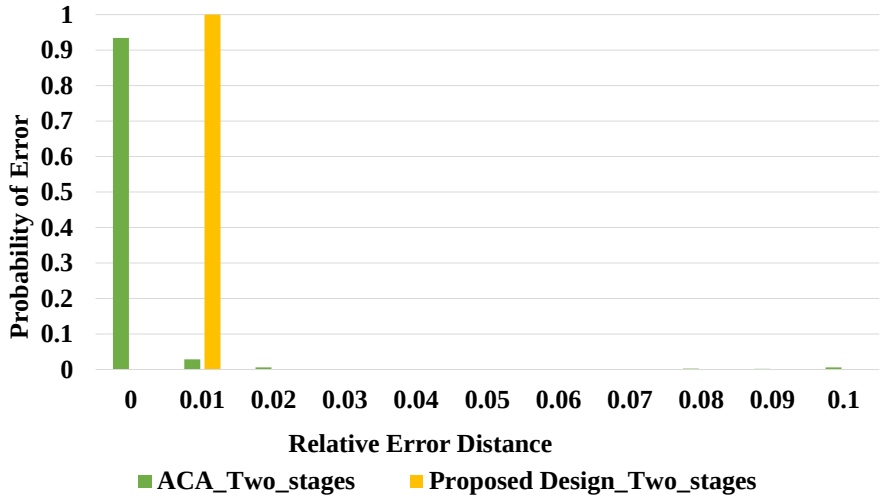


Figure 5.13: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis (Two correction stage).

In the case of two and three correction stages in Fig. 5.13 and Fig. 5.14 respectively, the proposed design has improved the ratio of (0.01) RED value and reached approximately to 100% of the test space, in contrast to ACA that still show different values of

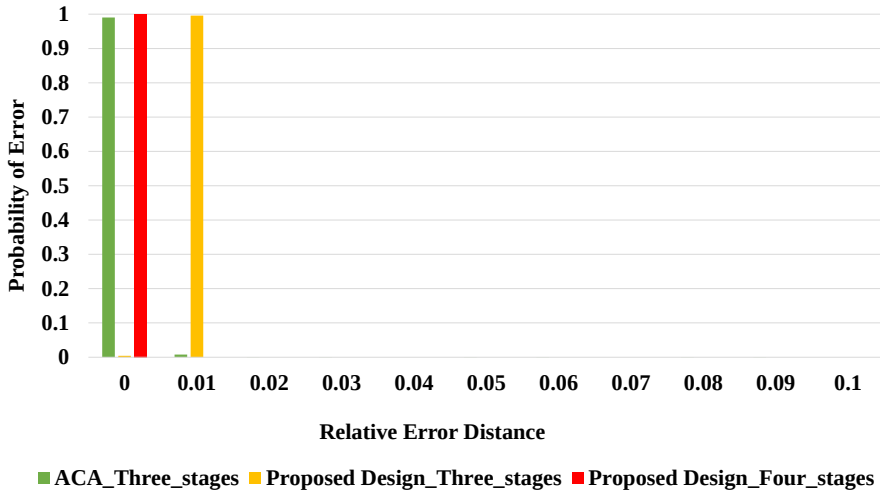


Figure 5.14: 32-Bit proposed design vs. ACA relative error distance (RED) distribution analysis with three and four correction stages.

RED. Hence, this might show that the proposed design has more consistency in terms of resulted outputs than ACA design.

All previous figures of error analysis results can be referred to the correction process implementation while using significance-driven stages structure. In the used structure, it starts correcting the most significant (i.e., higher order) bits of the adder first due to their significant impact in the final sum result. As a result, very small magnitudes of error would be expected.

The highest level of accuracy when all four correction stages are in operation is presented in Fig. 5.14 (red column). The result clearly confirms that the behaviour of the proposed design would guarantee the optimum exact result and achieve full accuracy at the final outputs. This due to the fact that at the final correction or quality control stage; the proposed design starts to work as a conventional adder with full exact output results.

Fig. 5.15 presents the mean relative error distance (MRED) values comparison, which shows the mean value of errors within test space or input vectors at each correction stage. At the approximate OR logic gates addition stage, the proposed design shows a higher

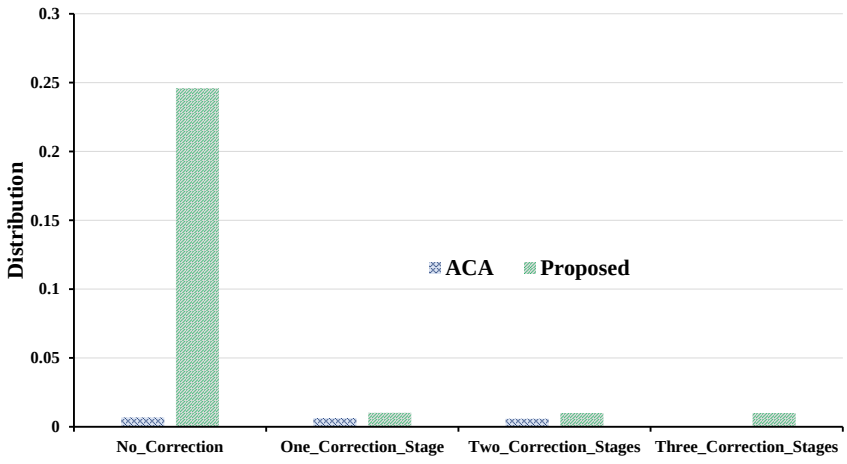


Figure 5.15: The 32-Bit adder designs MRED values comparison through different correction stages.

level of MRED values when compared to ACA design. This level of error is referred to the use of simple logic gates with a considerable error probability to compensate the conventional full addition. However, by start using the correction stages, the proposed design MRED values are drastically decreased showing the approximately consistent low level of errors through all stages. The significance-driven structure of the proposed output quality controlling stages (i.e., short bit-width exact adders) would guarantee exact addition sum for the higher order (i.e., most significant) bits early.

For the final fourth stage, the proposed design operates as an exact adder, thus, providing the MRED value equals to zero with optimum output accuracy.

Fig. 5.16 presents the cumulative probability distribution (CPD) for relative error distance (RED) levels through approximate addition and accuracy recovery stages. This analysis describes the change of percentages of the output space regarding the RED values. Additionally, it shows the speed of the resulted outputs become closer to the exact addition results.

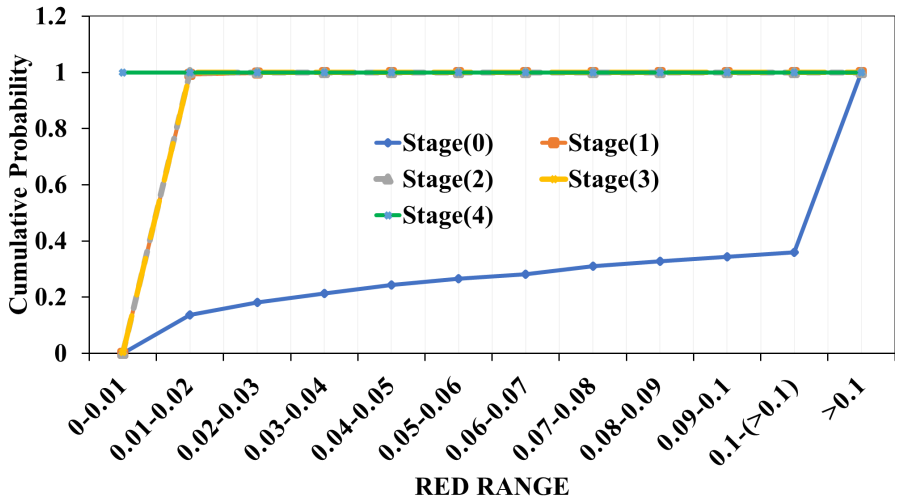


Figure 5.16: The cumulative probability distribution (CPD) for the error through different correction stages.

It can be noticed that the cumulative value at approximate addition stage without any error correction presents a quick step to the RED range ($0.01 \leq RED < 0.02$) with approximately 20% of the outputs space. This percentage would be gradually increased by small numbers for different values of RED and reach about 40% at $RED = 0.1$. The remaining portion of outputs space (60%) would have relative error values greater than 0.1 ($RED > 0.1$).

However, for the first three correction stages, all of these stages show an efficient and approximately the same cumulative behaviour since they step up quickly to the RED range ($0.01 \leq RED < 0.02$), and further, they become limited to this RED range and entirely cover all the output space. The percentage difference of the exact results between these correction stages is very small and can be considered 'zero'. Remarkably, the last fourth correction stage presents the better behaviour of accuracy with approximately 100% of exact results of the whole output space. This due to the fact that the proposed design at this final stage would start to work as a conventional adder with full accuracy.

5.8 IMAGE PROCESSING APPLICATION

To implement the image processing application test, we followed the same procedure in the previous chapters 3 and 4. Hence, a Gaussian blur image filter [20] was implemented by using Matlab test-bench to check the actual behaviour of the proposed design during multiple correction stages. The proposed adder of (20-bits) width was implemented, and the peak signal to noise ratio (PSNR) metric is used to measure the quality of the output images after applying the Gaussian blur filter.

From Fig. 5.17, it can be noticed that for Mode= 0 (without any correction), the resulted image shows a level of distortion with PSNR value of (19.7 dB). At the first correction stage (Mode=1), the resulted image PSNR value is (21.8 dB) with acceptable image blurring. This is referred to the considerable level of outputs error rate due to the use of simple logic OR gate to compensate the exact addition operation. However, for (Mode=2) where two correction stages are in action, a significant improvement has happened to the PSNR value by (50.6 dB) with a very acceptable output quality. Further, for (Mode=3) with three active correction stages, a better-quality output image has been provided with a very high PSNR value of (81.2 dB). At the final fourth stage of Mode=4, the proposed adder works as an exact adder; thus, the output image presents the optimum PSNR value and the best quality.

Nevertheless, both modes(0 and 1) might be considered attractive options for some biomedical applications, which are interested in high speed, low power and acceptable outputs quality [96, 2, 1].

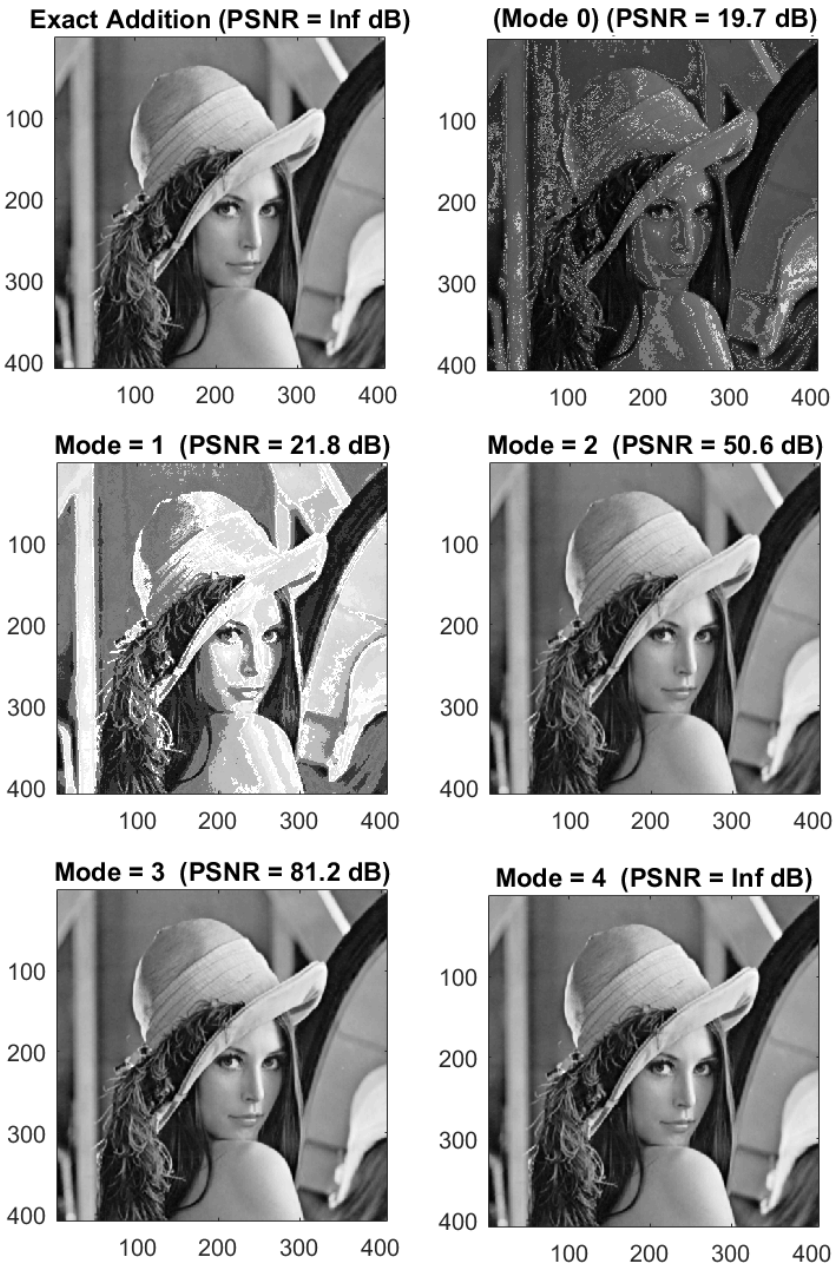


Figure 5.17: Gaussian blur image filter test.

5.9 SUMMARY

In this chapter, a low-power approximated adder design was proposed with output quality recovery stages. The proposed adder used simple OR logic gates to compensate conventional arithmetic

addition. Furthermore, the new accuracy recovery technique has approximately solved the challenge of the severe overhead of error detection and correction circuit. Consequently, the proposed design incurred a lower design overhead with 70%, 62% and 61% average reduction ratios for dynamic power, leakage power, and area respectively at the final correction stage.

The correction stages in the proposed design were structured in a significance-driven scheme, in which the first correction stage starts correcting highest error magnitudes. As a result, the proposed design achieved fast convergence to the exact output values at the premier correction stages. Since the proposed design started to work as a an exact adder at the final correction stage, full output accuracy was introduced, yet, with very limited design overhead.

The implementation test of the proposed design in an image processing application provided fair PSNR values of (19.7 dB and 21.8 dB) for the approximate addition and the first accuracy recovery stage respectively. Remarkably, a noticeable improvement happened to the output image quality at the second stage with a PSNR value of (50.6 dB), and the third stage with a PSNR value of (81.2 dB). Finally, when all the accuracy recovery stages were in action; the optimum PSNR value was shown similar to the result of an exact image filter circuit.

6

CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

Approximate computing has recently introduced a promising approach for the energy-efficient circuit design. Many of imprecision-resilient applications can exploit approximation techniques at different design levels while using inexact computing. These techniques would offer the chance of building low complexity, high performance, and energy-efficient designs, yet, with the cost of induced output errors. As one of the significant examples of approximate circuits, approximate arithmetic blocks (such as adders and multipliers) have received high consideration in the current research efforts.

This thesis proposed an investigation into configurable-accuracy approximate adder design as a promising approach for mitigating the challenges of design parameters overhead, such as critical delay, silicon area, and power consumption in addition to controlling the error recovery process due to approximation during the runtime. This section summarises the main conclusions drawn from this thesis.

The adder is considered an essential hardware module in most of the computing blocks. However, the strict correctness of conventional addition operation starts to be a real challenge in modern applications. This is referred to the necessary exact intensive addition operations to compute outcomes, which in turn, presents a

large portion of performance and power consuming. As a result, adder circuit design has been targeted for an approximation for lower complexity and energy efficient designs. In the adder operation, one of the key factors of dynamic power consumption is the glitches induced by carry propagation delay, which causes an undesired transition that occurs before the intended value in a digital circuit.

Nevertheless, the rare occurrence of long or worst carry chain in uniform random inputs has led to two general methods of adder approximations. The first method divides the adder bits into two parts, low order (least significant) bits, and high order (most significant) bits. An approximate addition technique can be used for the least significant bits part without carry propagation overhead. On the other hand, the most significant bits adder part uses the exact addition operation in order to preserve the general quality of outputs. The second method of adder approximation is to divide the adder into a number of sub-adders while using a carry speculation technique to each sub-adder, such as overlapping bits from the previous sub-adder or using logic gates to predict the carry. In this study, we have mainly targeted the design challenges of the configurable-accuracy approximate adder, which are discussed in Chapter 2.

To mitigate these challenges of design overhead, the first approach in chapter 3 has targeted a modification to the error detection technique of an exist configurable-accuracy approximate adder design. The proposed lightweight error detection technique which has replaced five (multi inputs) AND gates used for error detection (i.e., carry propagation) in the original design with just one XOR gate. The error detection takes place between every two adjacent sub-adders and performs equality checking for two in-

put carry signals. Thus, the proposed technique results in more reduction of the used logic gates, thus, a smaller silicon area and lower power consumption. In the error recovery process, we have ensured a quick convergence to the exact results by organising the correction stages in a bit significance-driven structure. This structure starts correcting the induced error at the high order (i.e., most significant) bits since they have the dominant effect in the final output quality. Moreover, while activating more than one correction stage simultaneously, an extensive error (i.e., carry propagation) detection has been added to the recovery process. As a result, this has guaranteed to attain an optimum accuracy of output at the worst case of quality requirements.

The results obtained after synthesis have shown better enhancement of the performance (i.e., speed) and a substantial decrease in design silicon area and power consumption, when compared to other existing designs. For the design error analysis, an analytical model has been demonstrated, which shows the general accuracy behaviour of the resulted outputs. On a statistical basis, our experiments used various error metrics, such as RED and MRED, which have shown better enhancements of accuracy and error distributions through the approximate addition and all the correction stages. Moreover, we demonstrate how the scalability of adder design has been improved for large bit width adders while preserving reduction ratios of design metrics such as delay, area, and power consuming.

For analysing the implementation effect of the proposed design in our first approach, an image processing filter known as Gaussian blur filter has been used as a test bench block. In this filter, all the exact adder blocks have been replaced by the proposed approximate adder. The peak to signal noise ratio (PSNR) metric was used to

measure the resulted image quality. The implementation results have confirmed the efficiency of our proposed adder designs by producing output images with acceptable levels of quality. In detail, the simulations results show high PSNR values (29 dB, 42 dB) for the first and second correction stages, respectively, and the optimum image quality at the last error correction stage. These results would confirm the feasibility of our design in such error-resilient applications.

The second approach in chapter 4 continues the effort to mitigate the challenge of design area overhead of the overlapped sub-adders in the previous first approach. Hence, we have presented a novel and simpler adder dividing (segmenting) method to a number of sub-adders. The new method has proposed a new bit location at the end of each sub-adder and used the carry-kill signal for both limiting the carry propagation and applying adder segmentation. Further, one AND gate and one XOR gate have been placed between every two adjacent sub-adders for carry speculation and error (i.e., carry propagation) detection respectively. Thus, a significant reduction of the number of sub-adder and logic gates has been introduced, and then, mitigating the total design overhead of the proposed adder. On the other hand, the error recovery process keeps the same bit significance-driven structure and the extensive carry propagation detection of the active correction stages. As such, the error recovery process continues to guarantee the fast convergence to the exact result and the full accuracy when operating all the correction stages.

Design synthesis results have shown more reduction values of the area and required power consumption, when compared to the first approach, and other exist designs. This is referred to eliminating overlapping parts of sub-adders of carry speculation, which

results in smaller number of sub-adders and the required logic gates. The delay values of approximate addition and the first correction stage show an enhancement of operation speed. However, the proposed additional bit locations and the carry prediction technique have incurred a limited delay overhead especially at the time of activating the final correction stages. For design error analysis, an analytical model has been demonstrated, which shows the error bound (i.e., maximum error) of the resulted outputs. By using the same error metric of RED and MRED, the simulation results show different levels of outputs accuracy for approximate addition and correction stages. Although the approximation addition stage shows a high error rate (i.e., error not equal to '0'), the relative error distance (RED) values are still limited between 99% and 98%, which are highly acceptable.

Moreover, the significance-driven structure of the correction stages presents a guarantee for high convergence to exact values, and then, a quick reduction of the error rate of outputs. For scalability validation, the proposed adder has been designed and synthesized with different larger bit-widths. The resulted hardware evaluation has confirmed the scalability characteristic of adder design while preserving reduction ratios of design metrics such as silicon area, power consuming and mitigating the delay values degradations.

The implementation results of the Gaussian blur filter present the feasibility of our proposed adder design while producing output images with acceptable levels of quality. By using the same PSNR metric, the simulations results show a fair value of (27 dB) for the approximate addition stage and values of (53 dB and 83 dB) for the first and second correction stages respectively.

While continuing our attempts to build approximate adder with lower complexity, we have exploited simple logic OR gates in the third approach in chapter 5 to entirely replacing the full adders blocks in the conventional adder. The proposed logic addition would compensate the exact addition operation. For error recovery process, instead of augmenting error recovery circuit, short bit-length exact adders have been used as a correction stages to control the general level of output quality. The proposed accuracy controlling method eliminates the incurred overhead of the augmented error detection circuit. Further, the suggested correction stages are organised in the same significance-driven structure as the previous two approaches, thus, presenting fast convergence to high outputs quality. At the final correction stage, the proposed design would operate the same as a conventional adder with full, accurate outputs.

Since our proposed design uses simple logic gates to compensate the conventional full adder addition operation, the synthesis results have shown significant reductions in delay, design silicon area, and power consumption. However, since the proposed adder works as an exact adder at the final stage of quality recovery, the operation speed would be the same as a conventional adder. For error analysis, we showed the error bound of the used OR gates by using Monte Carlo simulations and test benches of RED and MRED error metrics to evaluate the accuracy levels through the design stages. Although the approximation addition stage shows no exact outputs, the majority of relative error distance (RED) values are less than 90%. These values can be considered acceptable regarding the significant reductions of the design metrics. Further, the significance-driven structure of the correction stages provides fast

convergence to exact values, and then, a quick enhancement of outputs accuracy.

The proposed adder implementation in the Gaussian blur filter has presented an acceptable PSNR value of (19 dB) for the approximate addition stage and values of and (21 dB, 50 dB, 81 dB)for the first, second and third correction stages respectively, and further, the optimum PSNR value while activating all the accuracy recovery stages. These results would confirm the feasibility of our design in high error-resilient applications.

All in all, we can conclude that the proposed designs present simple techniques and a good extension to the configurable-accuracy adder designs, by achieving lower design metrics, such as delay, area, and power when compared to other efforts. Nonetheless, they preserve a high level of accuracy (i.e., output quality), and thus, compatibility with standard image processing applications.

6.2 FUTURE WORK

The objective of this thesis is to demonstrate that there is still a room existing for techniques modifications and novel ideas for the approximate adder designs. This work can be considered an extension for the configurable accuracy adder approximation design efforts. However, more work can be motivated by this thesis to achieve further performance and energy efficiency. The following points would show directions for future research due to some design limitations:

•*Implementations*: Future work might include using the proposed designs in complete circuits that consist of control and memory aspects. Further, it can be targeting other hardware blocks that comprise a large number of adders, such as multipliers. As such,

the field-programmable gate array (FPGA) platform would be useful to conduct more feasibility analysis. The proposed adder designs can be tested over more imprecision-resilient applications, e.g., digital signal processing (DSP) and biomedical applications, which show a large concern for high speed, low power, and acceptable quality outputs designs. These implementations can be combined with new approximation techniques and error recovery techniques while applying the comprehensive analysis of design metrics and accuracy levels.

•*Dynamic Voltage Frequency Scaling (DVFS)*: The proposed approximated adder designs result in a shorter carry path (i.e., more execution speed), especially the designs in chapter 4 and chapter 5. This can be utilized for more energy/performance efficiency by using voltage/frequency scaling during execution of addition operations. For example, while implementing approximate and exact adders to execute the workload operations, the resulted slack time of the tasks executed by approximate adders can be exploited to consciously slow down the execution (e.g., scaling down the operational clock frequency) as stated in slack reclamation approach [70, 64]. Generally, this aims to reduce the total consumed energy while meeting performance targets.

•*Automated analysis and verification*: In this work, several Verilog test benches have been used to check the designs validation. Further, we analysed the resulted error levels of various design stages (approximation and correction stages) by performing Matlab test benches and statistical techniques (Monte-Carlo simulations). These techniques are very time consuming and not flexible enough while supplying design error bounds as a part of the input. As such, adding new tools for automating the approximate design synthesis, validation and error impact analysis becomes an urgent

requirement to facilitate the whole design processes. However, the new tools design should be adaptive and carefully consider a range of aspects, such as parametrisation and analytical error bounds, which are both critical and application dependent. Therefore, the automated analysis and verification of the approximate adders is considered a vital area of future research.

Generally, we believe that the outcomes of this thesis can be considered a useful contribution for the approximate circuit design community, and a guide to further research and development effort for the directions mentioned above.

Part II

Bibliography

BIBLIOGRAPHY

[1] W. Al-Atabany and P. Degenaar. Scene optimization for optogenetic retinal prosthesis. In *2011 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 432–435, Nov 2011. doi: 10.1109/BioCAS.2011.6107820.

[2] W. Al-Atabany, B. McGovern, K. Mehran, R. Berlinguer-Palmini, and P. Degenaar. A processing platform for optoelectronic/optogenetic retinal prosthesis. *IEEE Transactions on Biomedical Engineering*, 60(3):781–791, March 2013. ISSN 0018-9294. doi: 10.1109/TBME.2011.2177498.

[3] K. Al-Maaitah, I. Qiqieh, A. Soltan, and A. Yakovlev. Configurable-accuracy approximate adder design with light-weight fast convergence error recovery circuit. In *2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pages 1–6, Oct 2017. doi: 10.1109/AEECT.2017.8257753.

[4] K. Al-Maaitah, G. Tarawneh, A. Soltan, I. Qiqieh, and A. Yakovlev. Approximate adder segmentation technique and significance-driven error correction. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–6, Sep. 2017. doi: 10.1109/PATMOS.2017.8106986.

[5] P. Albicocco, G. C. Cardarilli, A. Nannarelli, M. Petricca, and M. Re. Imprecise arithmetic for low power image processing. In *2012 Conference Record of the Forty Sixth Asilomar*

Conference on Signals, Systems and Computers (ASILOMAR), pages 983–987, Nov 2012. doi: 10.1109/ACSSC.2012.6489164.

[6] J. Ansel, Y. L. Wong, C. Chan, M. Olszewski, A. Edelman, and S. Amarasinghe. Language and compiler support for auto-tuning variable-accuracy algorithms. In *International Symposium on Code Generation and Optimization (CGO 2011)*, pages 85–96, April 2011. doi: 10.1109/CGO.2011.5764677.

[7] W. Baek and T. Chilimbi. Green: A system for supporting energy-conscious programming using principled approximation. *Microsoft Research, Tech. Rep.*, pages TR–2009, 2009.

[8] V. Benara and S. Purini. Accurus: A fast convergence technique for accuracy configurable approximate adder circuits. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 577–582, July 2016. doi: 10.1109/ISVLSI.2016.58.

[9] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, Sep. 2010. ISSN 0010-4620. doi: 10.1093/comjnl/bxp080.

[10] D.J. Brown and C. Reams. Toward energy-efficient computing. *Commun. ACM*, 53(3):50–58, 2010.

[11] S. T. Chakradhar and A. Raghunathan. Best-effort computing: Re-thinking parallel software and hardware. In *Design Automation Conference*, pages 865–870, June 2010. doi: 10.1145/1837274.1837492.

[12] V. K. Chippa, D. Mohapatra, K. Roy, S. T. Chakradhar, and A. Raghunathan. Scalable effort hardware design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(9):2004–2016, Sep. 2014. ISSN 1063-8210. doi: 10.1109/TVLSI.2013.2276759.

[13] K. Cho, Y. Lee, Y. H. Oh, G. Hwang, and J. W. Lee. edram-based tiered-reliability memory with applications to low-power frame buffers. In *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 333–338, Aug 2014. doi: 10.1145/2627369.2627626.

[14] A. A. Del Barrio, M. C. Molina, J. M. Mendias, E. Andres, R. Hermida, and F. Tirado. Applying speculation techniques to implement functional units. In *2008 IEEE International Conference on Computer Design*, pages 74–80, Oct 2008. doi: 10.1109/ICCD.2008.4751843.

[15] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge. Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits. *Proceedings of the IEEE*, 98(2):253–266, Feb 2010. ISSN 0018-9219. doi: 10.1109/JPROC.2009.2034764.

[16] K. Du, P. Varman, and K. Mohanram. High performance reliable variable latency carry select addition. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1257–1262, March 2012. doi: 10.1109/DATE.2012.6176685.

[17] R. P. Duarte and C. Bouganis. Zero-latency datapath error correction framework for over-clocking dsp applications on fpgas. In *2014 International Conference on ReConFigurable*

Computing and FPGAs (ReConFig14), pages 1–7, Dec 2014. doi: 10.1109/ReConFig.2014.7032566.

[18] R. P. Duarte and C. Bouganis. A unified framework for over-clocking linear projections on fpgas under pvt variation. In *International Symposium on Applied Reconfigurable Computing*, pages 49–60. Springer, 2014.

[19] S. Dutt, H. Patel, S. Nandi, and G. Trivedi. Exploring approximate computing for yield improvement via re-design of adders for error-resilient applications. In *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, pages 134–139, Jan 2016. doi: 10.1109/VLSID.2016.101.

[20] En.wikipedia.org. Gaussian blur, 2018. URL https://en.wikipedia.org/wiki/Gaussian{_}blur.

[21] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. *IEEE Micro*, 32(3):122–134, May 2012. ISSN 0272-1732. doi: 10.1109/MM.2012.17.

[22] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Power challenges may end the multicore era. *Commun. ACM*, 56(2):93–102, February 2013. ISSN 0001-0782. doi: 10.1145/2408776.2408797. URL http://doi.acm.org/10.1145/2408776.2408797.

[23] S. Ganapathy, G. Karakonstantis, A. Teman, and A. Burg. Mitigating the impact of faults in unreliable memories for error-resilient applications. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2015. doi: 10.1145/2744769.2744871.

[24] J. Gantz and D. Reinsel. Extracting value from chaos. *IDC iview*, 1142(2011):1–12, 2011.

[25] J. George, B. Marr, B. E. S. Akgul, and K. V. Palem. Probabilistic arithmetic and energy efficient embedded signal processing. In *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES '06, pages 158–168, New York, NY, USA, 2006. ACM. ISBN 1-59593-543-6. doi: 10.1145/1176760.1176781. URL http://doi.acm.org/10.1145/1176760.1176781.

[26] B. Grigorian and G. Reinman. Dynamically adaptive and reliable approximate computing using light-weight error analysis. In *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 248–255, July 2014. doi: 10.1109/AHS.2014.6880184.

[27] B. Grigorian and G. Reinman. Accelerating divergent applications on simd architectures using neural networks. In *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pages 317–323, Oct 2014. doi: 10.1109/ICCD.2014.6974700.

[28] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy. Impact: Imprecise adders for low-power approximate computing. In *IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 409–414, Aug 2011. doi: 10.1109/ISLPED.2011.5993675.

[29] 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, Jan 2013. ISSN 0278-0070. doi: 10.1109/TCAD.2012.2217962.

[30] J. Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *2013 18th IEEE European Test Symposium (ETS)*, pages 1–6, May 2013. doi: 10.1109/ETS.2013.6569370.

[31] K. He, A. Gerstlauer, and M. Orshansky. Controlled timing-error acceptance for low energy idct design. In *2011 Design, Automation Test in Europe*, pages 1–6, March 2011. doi: 10.1109/DATE.2011.5763129.

[32] R. Hegde and N. R. Shanbhag. Soft digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(6):813–823, Dec 2001. ISSN 1063-8210. doi: 10.1109/92.974895.

[33] T. Higgs. Energy efficient computing. In *Proceedings of the 2007 IEEE International Symposium on Electronics and the Environment*, pages 210–215, May 2007. doi: 10.1109/ISEE.2007.369396.

[34] J. Huang and J. Lach. Exploring the fidelity-efficiency design space using imprecise arithmetic. In *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, pages 579–584, Jan 2011. doi: 10.1109/ASPDAC.2011.5722256.

[35] H. Jiang, J. Han, and F. Lombardi. A comparative review and evaluation of approximate adders. In *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '15, pages 343–348, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3474-7. doi: 10.1145/2742060.2743760. URL http://doi.acm.org/10.1145/2742060.2743760.

[36] H. Jiang, J. Han, F. Qiao, and F. Lombardi. Approximate radix-8 booth multipliers for low-power and high-performance operation. *IEEE Transactions on Computers*, 65(8):2638–2644, Aug 2016. ISSN 0018-9340. doi: 10.1109/TC.2015.2493547.

[37] A. B. Kahng and S. Kang. Accuracy-configurable adder for approximate arithmetic designs. In *DAC Design Automation Conference 2012*, pages 820–825, June 2012. doi: 10.1145/2228360.2228509.

[38] R Saroha Kavita, Rajani Bala, and Sunita Siwach. Review paper on overview of image processing and image segmentation. *International journal of Research in Computer applications and Robotics*, 1(7):1–13, 2013.

[39] S. Kemp. 2018 global digital reports, 2018. URL https://wearesocial.com/blog/2018/01/global-digital-report-2018. [Online; accessed 19-July-2018].

[40] L. B. Kish. End of moore's law: thermal (noise) death of integration in micro and nano electronics. *Physics Letters A*, 305(3-4):144–149, 2002.

[41] J. Kuruvilla, D. Sukumaran, A. Sankar, and S. P. Joy. A review on image processing and image segmentation. In *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, pages 198–203, March 2016. doi: 10.1109/SAPIENCE.2016.7684170.

[42] J. Liang, J. Han, and F. Lombardi. New metrics for the reliability of approximate and probabilistic adders. *IEEE Trans-*

actions on Computers, 62(9):1760–1771, Sep. 2013. ISSN 0018-9340. doi: 10.1109/TC.2012.146.

[43] I. Lin, Y. Yang, and C. Lin. High-performance low-power carry speculative addition with variable latency. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(9):1591–1603, Sep. 2015. ISSN 1063-8210. doi: 10.1109/TVLSI.2014.2355217.

[44] C. Liu, J. Han, and F. Lombardi. An analytical framework for evaluating the error characteristics of approximate adders. *IEEE Transactions on Computers*, 64(5):1268–1281, May 2015. ISSN 0018-9340. doi: 10.1109/TC.2014.2317180.

[45] G. Liu, Y. Tao, M. Tan, and Z. Zhang. Casa: Correlation-aware speculative adders. In *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 189–194, Aug 2014. doi: 10.1145/2627369.2627635.

[46] W. Liu, L. Chen, C. Wang, M. Neill, and F. Lombardi. Inexact floating-point adder for dynamic image processing. In *14th IEEE International Conference on Nanotechnology*, pages 239–243, Aug 2014. doi: 10.1109/NANO.2014.6967953.

[47] W. Liu, L. Chen, C. Wang, M. Neill, and F. Lombardi. Design and analysis of inexact floating-point adders. *IEEE Transactions on Computers*, 65(1):308–314, Jan 2016. ISSN 0018-9340. doi: 10.1109/TC.2015.2417549.

[48] S. L. Lu. Speeding up processing with approximation circuits. *Computer*, 37(3):67–73, March 2004. ISSN 0018-9162. doi: 10.1109/MC.2004.1274006. URL https://doi.org/10.1109/MC.2004.1274006.

[49] D. Mahajan, K. Ramkrishnan, R. Jariwala, A. Yazdanbakhsh, J. Park, B. Thwaites, A. Nagendrakumar, A. Rahimi, H. Esmaeilzadeh, and K. Bazargan. Axilog: Abstractions for approximate hardware design and reuse. *IEEE Micro*, 35(5): 16–30, Sep. 2015. ISSN 0272-1732. doi: 10.1109/MM.2015.108.

[50] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas. Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57 (4):850–862, April 2010. ISSN 1549-8328. doi: 10.1109/TCSI.2009.2027626.

[51] D. May and W. Stechele. Voltage over-scaling in sequential circuits for approximate computing. In *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–6, April 2016. doi: 10.1109/DTIS.2016.7483887.

[52] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, and J. Henkel. Probabilistic error modeling for approximate adders. *IEEE Transactions on Computers*, 66(3):515–530, March 2017. ISSN 0018-9340. doi: 10.1109/TC.2016.2605382.

[53] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. Modeling and synthesis of quality-energy optimal approximate adders. In *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 728–735, Nov 2012.

[54] J. S. Miguel, M. Badr, and N. E. Jerger. Load value approximation. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47,

pages 127–139, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-6998-2. doi: 10.1109/MICRO.2014.22. URL http://dx.doi.org/10.1109/MICRO.2014.22.

[55] S. Mittal. Power Management Techniques for Data Centers: A Survey. *CoRR*, abs/1404.6(1404.6681), 2014. doi: 10.2172/1150909. URL http://www.osti.gov/servlets/purl/1150909/.

[56] S. Mittal. A survey of techniques for approximate computing. *ACM Comput. Surv.*, 48(4):62:1–62:33, March 2016. ISSN 0360-0300. doi: 10.1145/2893356. URL http://doi.acm.org/10.1145/2893356.

[57] D. Mohapatra. *ApproximateComputing:Enabling Voltage Over-Scaling in Multimedia Applications*. PhD thesis, Purdue University, Nov 2011.

[58] D. Mohapatra, G. Karakonstantis, and K. Roy. Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator. In *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '09, pages 195–200, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-684-7. doi: 10.1145/1594233.1594282. URL http://doi.acm.org/10.1145/1594233.1594282.

[59] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy. Design of voltage-scalable meta-functions for approximate computing. In *2011 Design, Automation Test in Europe*, pages 1–6, March 2011. doi: 10.1109/DATE.2011.5763154.

[60] T. Moreau, J. San Miguel, M. Wyse, J. Bornholt, L. Ceze, N. Enright Jerger, and A. Sampson. A taxonomy of approximate computing techniques. *UW CSE Technical Report*, pages 1–5, 2016.

[61] K. Palem and A. Lingamneni. What to do about the end of moore's law, probably! In *DAC Design Automation Conference 2012*, pages 924–929, June 2012. doi: 10.1145/2228360.2228525.

[62] J. Park, J. H. Choi, and K. Roy. Dynamic bit-width adaptation in dct: An approach to trade off image quality and computation energy. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(5):787–793, May 2010. ISSN 1063-8210. doi: 10.1109/TVLSI.2009.2016839.

[63] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev. Energy-efficient approximate multiplier design using bit significance-driven logic compression. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 7–12, March 2017. doi: 10.23919/DATE.2017.7926950.

[64] I. H. Qiqieh. *Investigation into energy-efficient and approximate multiplier design*. PhD thesis, Newcastle University, School of Engineering, aug 2018.

[65] J. Rabaey. *Low power design essentials*. Springer Science & Business Media, 2009.

[66] 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 Conference*

Exhibition (DATE), 2017, pages 476–481, March 2017. doi: 10.23919/DATE.2017.7927036.

[67] A. Rahimi, L. Benini, and R. K. Gupta. Spatial memoization: Concurrent instruction reuse to correct timing errors in simd architectures. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(12):847–851, Dec 2013. ISSN 1549-7747. doi: 10.1109/TCSII.2013.2281934.

[68] A. Rahimi, A. Ghofrani, K. Cheng, L. Benini, and R. K. Gupta. Approximate associative memristive memory for energy-efficient gpus. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1497–1502, March 2015. doi: 10.7873/DATE.2015.0579.

[69] A. Ranjan, S. Venkataramani, X. Fong, K. Roy, and A. Raghunathan. Approximate storage for energy efficient spintronic memories. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2015. doi: 10.1145/2744769.2744799.

[70] N. B. Rizvandi, J. Taheri, and A. Y. Zomaya. Some observations on optimal frequency selection in dvfs-based energy consumption minimization. *Journal of Parallel and Distributed Computing*, 71(8):1154–1164, 2011.

[71] P. Roy, R. Ray, C. Wang, and W. F. Wong. Asac: Automatic sensitivity analysis for approximate computing. *SIGPLAN Not.*, 49(5):95–104, June 2014. ISSN 0362-1340. doi: 10.1145/2666357.2597812. URL http://doi.acm.org/10.1145/2666357.2597812.

[72] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke. Sage: Self-tuning approximation for graphics

engines. In *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 13–24, Dec 2013.

[73] M. Samadi, D. A. Jamshidi, J. Lee, and S. Mahlke. Paraprox: Pattern-based approximation for data parallel applications. *SIGPLAN Not.*, 49(4):35–50, February 2014. ISSN 0362-1340. doi: 10.1145/2644865.2541948. URL http://doi.acm.org/10.1145/2644865.2541948.

[74] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman. Enerj: Approximate data types for safe and general low-power computation. *SIGPLAN Not.*, 46(6):164–174, June 2011. ISSN 0362-1340. doi: 10.1145/1993316.1993518. URL http://doi.acm.org/10.1145/1993316.1993518.

[75] J. Sartori and R. Kumar. Branch and data herding: Reducing control and memory divergence for error-tolerant gpu applications. *IEEE Transactions on Multimedia*, 15(2):279–290, Feb 2013. ISSN 1520-9210. doi: 10.1109/TMM.2012.2232647.

[76] R. R. Schaller. Moore's law: past, present and future. *IEEE Spectrum*, 34(6):52–59, June 1997. ISSN 0018-9235. doi: 10.1109/6.591665.

[77] M. Schulz. The end of the road for silicon? *Nature*, 399 (6738):729, 1999.

[78] L. Sekanina. Introduction to approximate computing: Embedded tutorial. In *2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits Sys-*

tems (DDECS), pages 1–6, April 2016. doi: 10.1109/DDECS.2016.7482460.

[79] M. Shafique, S. Garg, J. Henkel, and D. Marculescu. The eda challenges in the dark silicon era: Temperature, reliability, and variability perspectives. In *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, pages 185:1–185:6, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2730-5. doi: 10.1145/2593069.2593229. URL http://doi.acm.org/10.1145/2593069.2593229.

[80] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel. A low latency generic accuracy configurable adder. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2015. doi: 10.1145/2744769.2744778.

[81] J. M. Shalf and R. Leland. Computing beyond moore's law. *Computer*, 48(12):14–23, Dec 2015. ISSN 0018-9162. doi: 10.1109/MC.2015.374.

[82] B. Shim, S. R. Sridhara, and N. R. Shanbhag. Reliable low-power digital signal processing via reduced precision redundancy. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(5):497–510, May 2004. ISSN 1063-8210. doi: 10.1109/TVLSI.2004.826201.

[83] D. Shin and S. K. Gupta. Approximate logic synthesis for error tolerant applications. In *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, pages 957–960, March 2010. doi: 10.1109/DATE.2010.5456913.

[84] D. Shin and S. K. Gupta. A new circuit simplification method for error tolerant applications. In *2011 Design, Automation*

Test in Europe, pages 1–6, March 2011. doi: 10.1109/DATE.2011.5763248.

[85] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard. Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 124–134, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0443-6. doi: 10.1145/2025113.2025133. URL http://doi.acm.org/10.1145/2025113.2025133.

[86] G. V. Varatkar and N. R. Shanbhag. Energy-efficient motion estimation using error-tolerance. In *ISLPED'06 Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, pages 113–118, Oct 2006. doi: 10.1145/1165573.1165599.

[87] Z. Vasicek and L. Sekanina. Evolutionary design of approximate multipliers under different error metrics. In *17th International Symposium on Design and Diagnostics of Electronic Circuits Systems*, pages 135–140, April 2014. doi: 10.1109/DDECS.2014.6868777.

[88] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan. Axnn: Energy-efficient neuromorphic systems using approximate computing. In *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 27–32, Aug 2014. doi: 10.1145/2627369.2627613.

[89] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan. Approximate computing and the quest for computing efficiency. In *2015 52nd ACM/EDAC/IEEE Design*

Automation Conference (DAC), pages 1–6, June 2015. doi: 10.1145/2744769.2744904.

[90] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan. Macaco: Modeling and analysis of circuits for approximate computing. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 667–673, Nov 2011. doi: 10.1109/ICCAD.2011.6105401.

[91] A. K. Verma, P. Brisk, and P. Ienne. Variable latency speculative addition: A new paradigm for arithmetic circuit design. In *2008 Design, Automation and Test in Europe*, pages 1250–1255, March 2008. doi: 10.1109/DATE.2008.4484850.

[92] M. Weber, M. Putic, H. Zhang, J. Lach, and J. Huang. Balancing adder for error tolerant applications. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 3038–3041, May 2013. doi: 10.1109/ISCAS.2013.6572519.

[93] Wikipedia contributors. Dennard scaling — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Dennard_scaling&oldid=875866716, 2018.

[94] Wikipedia contributors. Memoization — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Memoization&oldid=864455867, 2018. [Online; accessed 20-February-2019].

[95] Wikipedia contributors. Moore's law — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Moore%27s_law&oldid=883976956, 2019.

[96] M. A. Yaman, W. Al-Atabany, A. Bystrov, and P. Degenaar. Fpga design for dual-spectrum visual scene prepa-

ration in retinal prosthesis. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4691–4694, Aug 2014. doi: 10.1109/EMBC.2014.6944671.

[97] A. Yan-Li. Introduction to digital image pre-processing and segmentation. In *2015 Seventh International Conference on Measuring Technology and Mechatronics Automation*, pages 588–593, June 2015. doi: 10.1109/ICMTMA.2015.148.

[98] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi. Approximate xor/xnor-based adders for inexact computing. In *2013 13th IEEE International Conference on Nanotechnology (IEEE-NANO 2013)*, pages 690–693, Aug 2013. doi: 10.1109/NANO.2013.6720793.

[99] A. Yazdanbakhsh, G. Pekhimenko, B. Thwaites, H. Esmaeilzadeh, O. Mutlu, and T. Mowry. Rfvp: Rollback-free value prediction with safe-to-approximate loads. *ACM Trans. Archit. Code Optim.*, 12(4):62:1–62:26, January 2016. ISSN 1544-3566. doi: 10.1145/2836168. URL http://doi.acm.org/10.1145/2836168.

[100] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu. On reconfiguration-oriented approximate adder design and its application. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 48–54, Nov 2013. doi: 10.1109/ICCAD.2013.6691096.

[101] T. Yeh, P. Faloutsos, M. Ercegovac, S. Patel, and G. Reinman. The art of deception: Adaptive precision reduction for area efficient physics acceleration. In *40th Annual IEEE/ACM In-*

ternational Symposium on Microarchitecture (MICRO 2007), pages 394–406, Dec 2007. doi: 10.1109/MICRO.2007.9.

[102] Y. Yetim, M. Martonosi, and S. Malik. Extracting useful computation from error-prone processors for streaming applications. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 202–207, March 2013. doi: 10.7873/DATE.2013.055.

[103] 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 (VLSI) Systems*, 24(10):3105–3117, Oct 2016. ISSN 1063-8210. doi: 10.1109/TVLSI.2016.2535398.

[104] N. Zhu, W. L. Goh, and K. S. Yeo. An enhanced low-power high-speed adder for error-tolerant application. In *Proceedings of the 2009 12th International Symposium on Integrated Circuits*, pages 69–72, Dec 2009.

[105] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo. Enhanced low-power high-speed adder for error-tolerant application. In *2010 International SoC Design Conference*, pages 323–327, Nov 2010. doi: 10.1109/SOCDC.2010.5682905.

[106] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(8):1225–1229, Aug 2010. ISSN 1063-8210. doi: 10.1109/TVLSI.2009.2020591.

[107] N. Zhu, W. L. Goh, and K. S. Yeo. Ultra low-power high-speed flexible probabilistic adder for error-tolerant applications. In *2011 International SoC Design Conference*, pages 393–396, Nov 2011. doi: 10.1109/ISOCC.2011.6138614.