Systematic Design of Low-power Processing Elements Using Stochastic and Approximate Computing Techniques

Dissertation zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.) im Fach Elektrotechnik und Informationstechnik

Ardalan Najafi

1. Gutachter:	Prof. Dr. Alberto García-Ortiz
2. Gutachter:	Prof. Dr. Guillermo Payá Vayá
Eingereicht am:	21.12.2020
Tag des Promotionskolloquiums:	26.01.2021

Universität Bremen

To my beloved family

Abstract

The approximate and stochastic computing have been developed, on the one hand, to address the diminishing gains of technology scaling, and on the other hand, to exploit the intrinsic error resilience of many applications. They, indeed, take advantage of the disparity between the level of accuracy required by the application and that provided by the computing system, for achieving energy efficiency. As of the most important constitutes of an integrated circuit, arithmetic units often lie within the critical path of a processing system. They play a vital role in determining the performance and power consumption of the computing system.

In the past decade, the design of the approximate arithmetic units has been in the center of attentions of the VLSI design research community; resulting in a numerous proposed approximate designs in the literature. In spite of a decade work on the approximate computing, there are still unresolved challenges faced by digital designers.

The concept of acceptable quality of the results forms the foundation of the approximate and stochastic computing. In view of this fact, it is crucially decisive to have a clear, quantifiable definition of what signifies an acceptable quality. Indeed, the current metrics most often do not capture the requirements of a target application, and hence, mislead to sub-optimal design options for the application. Moreover, non-systematic designs, lack of fair comparisons and reproducible research have resulted in somewhat limited progresses in the field of approximate and stochastic computing. Besides, the accuracy requirements of an application is not a static property and may change across the different phases of the application. Therefore, it is important to systematically develop approximate and stochastic computing platforms which offer a variety of output qualities.

In this dissertation, the aim is to take fundamental steps towards resolving the aforementioned challenges. Correspondingly, the following contributions are made in this dissertation. First, to palliate the lack of expressiveness of current metrics, a new parameterizable metric which correlates more precisely to the accuracy of the applications is proposed in this dissertation. Afterwards, the importance of fair comparisons for approximate computing units is underlined in this work. Subsequently, through generalizing and systematically optimizing an architectural template for approximate adders, an architecture is proposed which outperforms its existing counterparts. A conceptual framework for the systematic design of approximate adders including hybrid and non-equally segmented approaches is developed next. The framework discriminates the scenarios where approximate processing does not provide significant benefits from those where it does; in this latter case, it aids in obtaining optimal configurations for the adders. Furthermore, in order to address the dynamic

configuration of the error characteristics, a stochastically-tunable adder is proposed which reduces the energy-delay product considerably in comparison with its conventional counterpart. In addition, we develop data-dependent corrections for truncated multipliers, where the proposed architectures surpass the existing approximate multipliers in the literature.

The applicability of the proposed methods, and in general approximate computing units is eventually studied in modern applications. The correlation between the errors of a single unit and the whole system's accuracy is also investigated in the applications.

Kurzfassung

Die approximative und stochastische Berechnung wurde entwickelt, um einerseits die abnehmende Gewinne der Technologieskalierung zu adressieren und um andererseits die intrinsische Fehlerresistenz vieler Anwendungen auszunutzen. Sie nutzen den Vorteil der Diskrepanz zwischen der von der Anwendung geforderten und der vom Rechensystem bereitgestellten Genauigkeit, um eine Energie-Effizienz zu erzielen. Als einer der wichtigsten Bestandteile einer integrierten Schaltung liegen Recheneinheiten oft im kritischen Pfad eines Verarbeitungssystems. Sie spielen eine wesentliche Rolle bei der Bestimmung der Leistung und des Stromverbrauchs des Rechensystems.

Im vergangenen Jahrzehnt stand der Entwurf der approximativen Recheneinheiten im Zentrum der Aufmerksamkeit der VLSI-Design-Forschungsgemeinschaft, was zu zahlreichen Vorschlägen von approximativen Designs in der Literatur führte. Trotz der Arbeit eines Jahrzehnts mit approximativen Recheneinheiten gibt es immer noch ungelöste Herausforderungen für digitale Designer.

Das Konzept der akzeptablen Qualität von Ergebnissen bildet die Grundlage für die approximative und stochastische Berechnung. In Anbetracht dieser Tatsache ist es von entscheidender Bedeutung eine klare, quantifizierbare Definition dessen zu haben, was eine akzeptable Qualität bedeutet. In der Tat erfassen die aktuellen Messwerte meist nicht die Anforderungen einer Zielanwendung und verleiten daher zu suboptimalen Designoptionen der Anwendung. Außerdem haben unsystematische Designs, fehlende faire Vergleiche und reproduzierbare Forschung eher begrenzte Fortschritte auf dem Gebiet der approximativen und stochastischen Berechnung nach sich gezogen. Darüber hinaus sind die Genauigkeitsanforderungen einer Anwendung keine statische Eigenschaft und können sich über die verschiedenen Phasen der Anwendung ändern. Daher ist es wichtig, systematisch approximative und stochastische Berechnungsplattformen zu entwickeln, die eine Vielzahl von Output-Qualitäten bieten.

In dieser Dissertation sollen grundlegende Schritte zur Lösung der oben genannten Herausforderungen unternommen werden. Dementsprechend werden die Beiträge dieser Dissertation folgende sein: Erstens, um die mangelnde Aussagekraft aktueller Messwerte zu lindern, wird ein neuer parametrisierbarer Messwert in dieser Dissertation vorgeschlagen, der präziser mit der Anwendungsgenauigkeit korreliert. Anschließend wird die Bedeutung fairer Vergleiche für approximatives Computing in dieser Arbeit hervorgehoben. Darauffolgend wird durch Verallgemeinerung und systematische Optimierung einer Architekturvorlage für approximative Addierer eine Architektur vorgeschlagen, die ihr existierendes Gegenstück übertrifft. Als Nächstes wird ein konzeptioneller Rahmen für den systematischen Entwurf von approximativen Addierern einschließlich hybrider und nicht-gleich-segmentierten Ansätzen entwickelt. Die Rahmenkonstruktion unterscheidet die Szenarien, in denen die approximative

Verarbeitung keine signifikanten Vorteile bietet, von denen, in denen sie signifikante Vorteile bietet; im letzteren Fall hilft sie bei der Erlangung optimaler Konfigurationen für die Addierer.

Weiterhin wird ein stochastisch abstimmbarer Addierer vorgeschlagen, um die dynamische Konfiguration der Fehlereigenschaften zu berücksichtigen, der das Energieverzögerungsprodukt im Vergleich zu seinem konventionellen Gegenstück erheblich reduziert. Zusätzlich entwickeln wir datenabhängige Korrekturen für abgeschnittene Multiplikatoren, wobei die vorgeschlagenen Architekturen die bestehenden approximativen Multiplikatoren in der Literatur übertreffen.

Die Anwendbarkeit der vorgeschlagenen Methoden und generell approximative Recheneinheiten werden schließlich in modernen Anwendungen untersucht. Die Korrelation zwischen den Fehlern einer einzelnen Einheit und der Genauigkeit des Gesamtsystems wird ebenfalls in den Anwendungen untersucht.

Acknowledgment

I, personally, consider myself a lucky person. Throughout my life, I have always been destined to meet amazing people. I cannot name all of the people, but I thank anyone who helped me to get to this point. Yet, I would like to name the people most directly responsible for this dissertation becoming a reality.

First and foremost, I would like to thank Prof. Alberto Garcia-Ortiz. I was blessed to be supervised by him during my PhD study. I strongly believe that this work could not be succeeded without his supervision. He is such a phenomenal person who not only improved my research and teaching skills but also impacted on my character and made me a better person. Besides being a professional boss who is strict in the work, he has developed an international team and a genuinely friendly workplace. Alberto is a thoughtful, supportive and diligent person. He loves his job which is the best motivation for his PhD students. I do greatly appreciate all what he has done for me.

Next, I would like to thank Kerstin Janssen, the administrator of ITEM.ids group at university of Bremen. Indeed, I cannot thank you Kerstin with words. You have helped each of us significantly. From the very beginning, by finding a flat for my brother and me, which is out of the scope of your responsibilities, to bringing the very delicious "Grünkohl and Kasseler" for me when I was writing my defense, to the very last one with helping me binding the printed dissertation, I was fortunate to have you as my teammate. You literally made my life much easier in Bremen. Thank you Kerstin, a million.

I could not ask for a better working group and for better colleagues. Among my teammates, I would like to first thank the one who is not only my brother, but also the most supportive friend of mine throughout my life. Amir has been there for me whenever I needed help. In my difficulties, happiness, and sickness, Amir has always been the one I counted on, in whole my life. Clearly, without you I could not easily get to this point. Wanli Yu and Yanqiu Huang, the lovely couple, who have been valuable colleagues and are great friends, helped me a lot during my PhD. Thank you for all your supports and motivations. I am proud of having you two as my friends. Robert Schmidt, it has been a fantastic experience to work with you. Yarib Nevarez, thank you for bringing your positive energy to our group. And more than anybody else, I spent time with Lennart Bamberg. He is an ambitious, determined, genius person. I have learned a lot by working with him. I enjoyed each and every moment with you Lennart, during working hours as well as hanging out with you outside of the institute. I am happy to have you as my friend.

Besides the IDS group, I would like to thank Maike Schröder, Peter Lutzen, Jakob Döring, Andreas Beering, Jochen Rust, and Mingjie Hao, who helped me in different ways. Andreas and Jakob are not only awesome colleagues, but also amazing friends. The collaboration experience with Mingjie and Jochen

was also superb.

I would also like to thank IMS institute of Leibniz university of Hannover, and more specifically, prof. Guillermo Paya-Vaya and Moritz Weissbrich. We collaborated on a DFG project. It was a very convenient and pleasant experience working with you. I would also like to thank Prof. Goerschwin Fey from Hamburg University of Technology (TUHH), and Prof. Karl-Ludwig Krieger from University of Bremen for serving as two of the examiners for this dissertation.

Among all my beloved friends, I would like to thank Freya Gröning and Charlotte Wagner who made me feel at home from the very beginning in Bremen. I do appreciate their presence in my life.

Last but not least, special thanks goes to my family as well as my partner. Without the supports of my parents, Akram and Akbar, I could not attend at university of Bremen to pursue my PhD. The first credit of this dissertation belongs to my parents without any doubt. I owe them until the end of my life, and I am grateful to have their supports. My sister, Elahe, did a lot for me. Elahe has been the most supportive sister I have ever seen. I also thank Shahin Vaziri, my brother-in-law. I never forget your supports specially during my master study. I also would like to thank Stefanie Jahn. Steffi and her family are wonderful people who I consider as my family here in Germany. Thank you Steffi also for your helps with this dissertation. Above all, I am grateful to have Hedieh Farhandi as my partner. She is such an energetic, positive, supportive, thoughtful and sympathetic person. Hedieh, you are not only my partner, but also my best friend to whom I share all my feelings with. I am fortunate to have you in my life. I thank you for all your supports, and more specifically during the last months when I was writing this dissertation. I am happy to have you beside me.

Table of Contents

Specific Mathematical Symbols

$Pr\big[x_i\big]$	probability of x_i being 1.
T_{clk}	period time of the clock determining the pattern duration.
T_{pd}	50%–50% propagation delay.
V_{dd}	power-supply voltage.
$\neg$	bit-wise logical/Boolean negation.
$\mathbb{E}\big[X\big]$	expectation operator for a value discrete variable X.
$\mathbb{N}$	set of natural numbers: $\{0,1,2,3,\dots\}$.
μ	average error.
$\oplus$	Boolean Logical exclusive disjunction (XOR) operator.
σ^2	variance of error.
σ	standard deviation of error.
f	frequency equal to $1/T_{\mathrm{clk}}$.
$\lvert X\rvert$	the absolute value of variable X.
$\log_2(x)$	binary (base 2) logarithm of x.
$x_1 \bmod x_2$	modulo operation for x_1 over x_2 (*i.e.*, the remainder after the division).

Acronyms

ACA	Almost Correct Adder.
ADP	Area-Delay Product.
AI	Artificial Intelligence.
ALU	Arithmetic and Logical Unit.
AND	Logical conjunction.
ANT	Algorithmic Noise Tolerance.
ASIC	Application-Specific Integrated Circuit.
BAM	Broken-Array Multiplier.
CAD	Computer-Aided Design.
CDF	Cumulative Density Function.
CLA	Carry Lookahead Adder.
CMOS	Complementary Metal-Oxide-Semiconductor.
CPU	Central Processing Unit.
CSA	Carry-Save Adder.
DAG	Directed Acyclic Graph.
DFG	*German Research Foundation.*
DMA	Direct Memory Access.
DRAM	Dynamic Random-Access Memory.
DSP	Digital Signal Processing.
EDA	Electronic Design Automation.
EDP	Energy-Delay Product.
ESA	Equally Segmented Adder.
ETAII	Error Tolerant Adder type II.
ETM	Error Tolerant Multiplier.
FA	Full Adder.
FF	Flip-Flop.
FFT	Fast Fourier Transform.
FIFO	First-In First-Out.
FO4	Fan-Out of 4.
FOS	Frequency Over-Scaling.
FPGA	Field-Programmable Gate Array.
FSM	Finite State Machine.
GeAr	Generic Accuracy-configurable Adder.
HA	Half Adder.

IC	Integrated Circuit.
IoT	Internet of Things.
ITRS	International Technology Roadmap for Semiconductors.
LOA	Lower-part OR Adder.
LSB	Least Significant Bit.
LSP	Lower Significant Partial product.
MAC	Multiply-Accumulate Unit.
MAE	Mean Absolute Error.
MOS	Metal-Oxide-Semiconductor.
MOSFET	MOS Field-Effect Transistor.
MRAE	Mean Relative Absolute Wrror.
MSB	Most Significant Bit.
MSE	Mean Squared Error.
NAND	Logical non-conjunction.
NMSE	Normalized Mean Squared Error.
NN	Neural Network.
NOR	logical non-disjunction.
OLOCA	Optimized Lower-part OR-Constant Adder.
OR	logical disjunction.
PDF	Probability Density Function.
PDK	Process Design Kit.
PDP	Power-Delay Product.
PP	Partial Product.
PPA	Parallel-Prefix Adder.
PPT	Partial Product Tree.
PVT	Process Voltage Temperature.
RAM	Random-Access Memory.
RCA	Ripple-Carry Adder.
RF	Radio Frequency.
RGB	Red-Green-Blue.
RISC	Reduced Instruction Set Computer.
ROM	Read-Only Memory.
RTL	Register-Transfer Level.
Rx	Receiver.
SAIF	Switching Activity Interchange Format.
SIFT	Scale-Invariant Feature Transform.
SIMD	Single Instruction, Multiple Data.

SMSE	Saturated Mean Squared Error.
SNR	Signal-to-Noise Ratio.
SoC	System On Chip.
SPICE	Simulation Program with Integrated Circuit Emphasis.
SRAM	Static Random Access Memory.
SSIM	Structural Similarity Index Measure.
STA	Static Timing Analysis.
STD	Standard Deviation.
Tx	Transmitter.
UDM	UnderDesigned Multiplier.
VCD	Value Change Dump.
VHDL	Very high speed integrated circuit Hardware Description Language.
VLSI	Very-Large-Scale Integration.
VOS	Voltage Over-Scaling.
XNOR	logical exclusive non-disjunction.
XOR	Logical exclusive disjunction.

CHAPTER 1

Introduction and Overview

Nowadays, a transcendent challenge is growing as the increasing approach to Artificial Intelligence (AI) requires huge amount of data and complex calculations. The new systems demand higher performance and more energy efficiency. General purpose computers and Application-Specific Integrated Circuits (ASICs) are employed to step up the development of these new technologies. However, as technology scales down, it is getting increasingly expensive to ensure exact functionality of digital integrated circuits. The variability of devices and interconnects is increasing dramatically due to intrinsic (e.g., unequal dopant concentrations) and extrinsic (e.g., temperature) variations. Meanwhile, exact or high-precision computation is not always necessary. Applications such as multimedia, machine learning and data mining are inherently error-tolerant and do not require a perfect accuracy in computation. In addition, some errors may compensate each other, or their effects on the final result might be negligible. Therefore, Approximate Computing has emerged as a new technique to tackle the growing concerns by saving energy consumption and increasing the performance of a computer system, with an accepted loss of accuracy [1, 2].

1.1 Motivation

As feature size of transistors shrink with the advances in the technology, it becomes more and more expensive for designers and manufacturers to keep transistors functioning deterministic, even under normal operating conditions. The nondeterminism holds back the continuous technology progress, inspired by the Moor's law [3], in the past few decades [4].

Moreover, as transistor counts grow exponentially following Moore's law, the transistor threshold and supply voltages do not scale proportionately, and the power consumption of the additional transistors can no longer be alleviated through circuit-level techniques. As a result, a percentage of transistors of an integrated circuit cannot be powered-on at the nominal operating voltage to mitigate a given thermal design power constraint. The powered-off transistors are called *dark silicon* [5, 6]. Based on the technological data from ITRS and Intel, at the 8nm node, the area of dark silicon escalate over 50% of the chip area [5, 7, 8]. It brings new unprecedented challenges for the design community. Different design methodologies such as sub/near-threshold design [9–11] and approximate computing [2, 12–14] have been studied to address this problem.

Many applications such as computer vision, multimedia, and machine learning do not always require exact and high-precision computations. For instance, the impact of an extent of errors on the output quality of image, video and audio processing is imperceptible for humans. One approach is to exploit the ability of some applications to tolerate errors and deliberately expose the underlying system to the errors. This approach is a new vision with a radical view of the design of hardware systems: a plea for the early targeted identification and development of the faulty properties of hardware during the design phase [1, 2]. By matching the statistical hardware error characteristics to the statistical processing requirements of the target application, it is possible to achieve an enormous increase in energy efficiency or performance.

There are two fundamentally different approaches to assess the trade-off between hardware efficiency and precision: Approximate computing [2] and stochastic data processing [1]. With approximate computing, the requirements for an exact implementation of a Boolean function are relaxed; the exact functionality is replaced by an approximated functionality. The approximated functionality is chosen so that it can be implemented more effectively. In contrast, with stochastic data processing, the requirement for correct operation over the entire operating range is reduced. For example, frequency or voltage over-scaling are stochastic techniques which result in timing violation of some paths in the circuit.

Despite the fundamental differences, the key innovation in both approximate and stochastic data processing is that errors are allowed and are considered part of the design; the causes and effects of errors differ, however, and therefore

require different design methods. Indeed, it is even possible to combine methods from both approaches, which leads to better results in terms of improving hardware costs as well as accuracy.

1.1.1 Approximate computing:

The idea of approximating a logical functionality is so general that it can be applied to both software [15] and hardware [16]. The history of approximate computing dates back to early 1960s, where it was embraced to develop multiplication and division based on logarithm [17]. In this work, the binary logarithm is determined approximately from the number itself by simple shifting and counting. The next remarkable contributions during the next decades were truncation based approximate designs. In truncated computing units, some Least Significant Bits (LSBs) are pruned resulting in reduction in the circuit complexity. At the same time, the truncation generates errors which appear frequently, albeit when appear they are relatively small. In the case of multipliers, the truncation is applied to the Partial Products (PPs) which results in faster and more energy efficient partial product accumulation stage. Fixed-width multiplier, where the bit-width of input data and the output result are the same, were studied in [18–20]. The concept of approximate computing, afterwards, was applied to adders in [21]. The critical path of an adder is determined by its full carry chain. An exact adder obtains the final carry out and calculates the final results considering all the input bits. However, in real applications, due to the fact that the data distribution is not fully uniform, the effective carry chain of the adder is much shorter for most combinations of inputs. Therefore, a faster adder is developed with a much shorter carry chain which approximates the result. In an n-bit approximate adder, the carry is obtained by its previous k bits where $k < n$, resulting in a significantly shorter critical path.

In the past decade, approximate adders and multipliers have been in the center of researchers' attention resulting in a plethora of proposed approaches. The notable designs include the the Error Tolerant Multiplier (ETM) [22], Broken-Array Multiplier (BAM) [23], the UnderDesigned Multiplier (UDM) [24], Lower-part OR Adder (LOA) [23], the Equally Segmented Adder (ESA) [25], the Almost Correct Adder (ACA) [26], the Generic Accuracy-configurable Adder (GeAr) [27], Error Tolerant Adder type II (ETAII) [28]. Approximate adders and multipliers have also been generated using evolutionary design approaches [29–33]. Besides, in recent years, the approximate design of Multiply-Accumulate Unit (MAC) is gaining significant attentions [34–36]. The MAC is the core arithmetic computation performed during Neural Network (NN) inference.

On a closer inspection of approximate arithmetic units, current approaches use two philosophies for the error: small errors or unlikely errors. In the first

philosophy, approximate units are constructed so that the magnitude of errors is small even if they occur frequently. This is justified by error masking due to intrinsic rounding and noise errors, which is why no significant degradation of the application is to be expected. One example of this group is the LOA [23]. In the second philosophy, errors are constructed so that they rarely occur, even if they are large when they appear. The basic principle is based on the property that an application can survive errors as long as they are rare. An example of this group is the ETAII [28].

1.1.2 Stochastic computing:

Stochastic processing is a promising approach for designing energy-efficient embedded hardware systems. It has been emerged to address the problem of the languishing benefits of technology scaling. Stochastic processing uses the ability of many applications to tolerate a loss of quality in terms of precision of results. In stochastic computing, the requirement for correct operation over the entire operating range is reduced. Traditionally, the aim of the designers is to ensure correct operation of the hardware under all possible Process Voltage Temperature (PVT) variations. However, in stochastic computing, instead of hiding restrictions in hardware implementation behind expensive guardbands, the causal variability of the hardware is specifically revealed. Correspondingly, designers can palliate traditional constraints, which leads to significantly increased processing speed and results in energy benefits [37–42]. Stochastic techniques can be applied on different levels of abstraction including transistor and gate level, as well as Register-Transfer Level (RTL) and architecture level.

The error rate of the error resilient design determines the energy benefits offered by stochastic computing techniques. For example, when the error rate rises sharply with voltage/frequency over-scaling, due to confined scalability, only limited benefits are feasible. In case the error rate increases gradually, the benefits are considerable [37].

The better-than-worst-case strategy is advocated by the researchers to build designs which operate correctly under nominal conditions, and effectively address the challenges of deep sub-micron design [43]. This approach relaxes design constraints on core components, reduces the effects of physical design challenges, and correspondingly, creates opportunities to optimize performance and power characteristics. Razor [44] is an approach to dynamic voltage scaling, based on dynamic detection and correction of circuit timing errors. Employing this capability to tolerate timing errors, a Razor design consolidates self-checking circuits to enable pushing clock frequency and/or supply voltages beyond nominal worst-case levels. Further strategies of better-than-worst-case designs have been reviewed in [45].

Recovery-driven design [38] emphasizes that instead of designing and opti-

mizing hardware systems for correct operations, they should be optimized for a target error rate. Recovery-driven design is developed based on slack distribution. By redistributing timing slack from paths which generate infrequent errors to the critical paths that frequently cause errors, the energy benefits of exploiting error resilience are maximized. As a result, the minimum supply voltage and consequently power consumption are reduced; or the maximum frequency and consequently performance are increased for a target error rate [46].

1.2 Research Aims and Objectives

The aim of this dissertation is to identify the problems of the state-of-the-art and address those problems systematically. As mentioned above, a huge number of approximate designs have been proposed in the literature. However, they share a common characteristic: they have been obtained with an ad-hoc and non-systematic methodology. Although there are some exceptions, a majority of the designs aim to achieve an improvement (even if marginal) based on different trade-offs among energy efficiency, silicon area, performance and accuracy. Besides, the evaluations are not systematic. In the one hand, the evaluations are carried out using different synthesis tools and technologies; on the other hand, the metrics employed to quantify the quality of the approximate designs are not methodical. Indeed, the comparisons are mostly deficient. There is lack of systematic and fair comparisons. The basic rules for a fair comparison are often violated in the literature. In a majority of the research works, the proposed design is compared with sub-optimal designs, and the significant baseline designs are missing in the comparisons. Furthermore, even though approximate computing has been proved to be pivotal in the digital Very-Large-Scale Integration (VLSI) design and has received significant attention in the past decade, there is still no comprehensive library which can be used as a reference for comparisons as well as reproducible research.

The aforementioned shortcomings make it difficult to choose a suitable approximate design for a target application. Accordingly, the main objectives of this dissertation are described briefly.

A review and evaluation of existing approximate architectures is presented first in the next chapter. The goal, here, is to classify the approximate adders and multipliers, and review the characteristics of each class. Afterwards, with the overview of the existing approximate architecture, the problem of the state-of-the-art is identified; where fair comparisons of approximate adders and multipliers are performed. A simultaneous comparison of approximate architectures and stochastic exact designs is specifically considered which is believed as a missing point in the existing research works. We believe that approximate computing and stochastic computing have to be studied together, and in some scenarios even be combined to achieve the best trade-off between the

hardware cost and accuracy. In addition, a new metric to quantify the quality of approximate computing units when employed in more realistic scenarios, is presented. The new metric quantify the accuracy of an architecture according to the target application.

A majority of the existing approximate adders have been obtained non-systematic. In order to address this problem, a systematic approximate adder is proposed. The proposed adder is designed by optimizing a developed architectural template for maximum accuracy. Additionally, a comprehensive template for approximate adders is presented along with all the error formulas associated with it. Using the proposed template, various approximate adders can be developed, where the related formulas to the developed approximate adder can be derived from the general formulas. Rather than focusing on approximate computing, the systematic design of inexact adders is expanded to stochastic computing units. A stochastically-tunable low-error adder is proposed afterwards, where frequency over-scaling considered as the stochastic technique to compromise accuracy for energy efficiency.

A data-dependent truncated multiplier is proposed according to the observations from comparison of multipliers. A template for truncated multipliers is also developed, where fixed-width multipliers are also considered as part of the design.

The final goal is to show the applicability of the proposed architectures in real scenarios, where in multiple case-studies the functionality of the designs are evaluated.

1.2.1 Publications

The outcome of the research works for this dissertation, including the collaborative works with our research partners is a list of publications including [47–57]. In the following, a complete list of the related publications are itemized.

Journal Articles

1. Ayad Dalloo, **Ardalan Najafi*** and Alberto Garcia-Ortiz, "Systematic Design of an Approximate Adder: The Optimized Lower Part Constant-OR Adder," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 8, pp. 1595-1599, Aug. 2018, doi: 10.1109/TVLSI.2018.2822278.
   * *All the authors contributed equally to this manuscript.*

2. **Ardalan Najafi**, Moritz Weißbrich, Guillermo Payá-Vayá and Alberto Garcia-Ortiz, "Coherent Design of Hybrid Approximate Adders: Unified Design Framework and Metrics," in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 8, no. 4, pp. 736-745, Dec. 2018, doi: 10.1109/JETCAS.2018.2833284.

3. **Ardalan Najafi** and Alberto Garcia-Ortiz, "Stochastic Mixed-PR: A Stochastically-Tunable Low-Error Adder," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 10, pp. 2144-2148, Oct. 2020, doi: 10.1109/TCSII.2019.2953617.

4. Wanli Yu, **Ardalan Najafi**, Yanqiu Huang and Aalberto Garcia-Ortiz, "Combination of Task Allocation and Approximate Computing for Fog Architecture based IoT," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2020.3040892. (Early Access)

5. Amir Najafi, **Ardalan Najafi** and Alberto Garcia-Ortiz, "Stochastic Wave-Pipelined On-Chip Interconnect," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 5, pp. 841-845, May 2020, doi: 10.1109/TCSII.2020.2984194.

6. Amir Najafi, Lennart Bamberg, **Ardalan Najafi** and Alberto Garcia-Ortiz, "Integer-Value Encoding for Approximate On-Chip Communication," in IEEE Access, vol. 7, pp. 179220-179234, 2019, doi: 10.1109/ACCESS.2019.2959446.

7. Moritz Weißbrich, Lukas Gerlach, Holger Blume, **Ardalan Najafi**, Alberto García-Ortiz, Guillermo Payá-Vayá, "FLINT+: A runtime-configurable emulation-based stochastic timing analysis framework," in Integration (Elsevier), Vol. 69, pp. 120-137, 0167-9260, 2019, doi: 10.1016/j.vlsi.2019.01.002.

Conference Proceedings

8. **Ardalan Najafi**, Moritz Weißbrich, Guillermo Payá Vayá and Alberto Garcia-Ortiz, "A fair comparison of adders in stochastic regime," 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Thessaloniki, 2017, pp. 1-6, doi: 10.1109/PATMOS.2017.8106990.

9. Wanli Yu, **Ardalan Najafi**, Yarib Nevarez, Yanqiu Huang and Alberto Garcia-Ortiz, "TAAC: Task Allocation Meets Approximate Computing for Internet of Things," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Sevilla, 2020, pp. 1-5, doi: 10.1109/ISCAS45731.2020.9180895.

10. Mingjie Hao, **Aardalan Najafi**, Alberto García-Ortiz, Ludwig Karsthof, Steffen Paul, Jochen Rust, "Reliability of an Industrial Wireless Communication System using Approximate Units," 2019 29th International

Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Rhodes, Greece, 2019, pp. 87-90, doi: 10.1109/PATMOS.2019.8862161.

11. Moritz Weißbrich, **Ardalan Najafi**, Alberto Garcia-Ortiz, and Guillermo Paya-Vaya. "ATE-Accuracy Trade-Offs for Approximate Adders and Multipliers in Pipelined Processor Datapaths", Third Workshop on Approximate Computing, 2019, Bremen, Germany (AxC18, www.lirmm.fr/axc18)

12. Yizhi Chen, **Ardalan Najafi**, and Alberto Garcia-Ortiz, "On the Effects of Data Distribution on Small-error Approximate Adders," 2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST), Bremen, Germany, 2020, pp. 1-4, doi: 10.1109/MOCAST49295.2020.9200260.

13. Amir Najafi, Lennart Bamberg, **Ardalan Najafi** and Alberto Garcia-Ortiz, "Misalignment-aware delay modeling of narrow on-chip interconnects considering variability," 2018 7th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, 2018, pp. 1-4, doi: 10.1109/MOCAST.2018.8376593.

14. Moritz Weißbrich, Guillermo Payá-Vayá, Lukas Gerlach, Holger Blume, **Ardalan Najafi**, and Alberto García-Ortiz, "FLINT+: A runtime-configurable emulation-based stochastic timing analysis framework," 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Thessaloniki, 2017, pp. 1-8, doi: 10.1109/PATMOS.2017.8106956.

15. Amir Najafi, Lennart Bamberg, **Ardalan Najafi**, and Alberto Garcia-Ortiz, "Energy modeling of coupled interconnects including intrinsic misalignment effects," 2016 26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), Bremen, 2016, pp. 262-267, doi: 10.1109/PATMOS.2016.7833697.

1.3 Dissertation Outline

This dissertation is organized into 7 chapters. The review of the existing approximate architectures is presented following to the introduction in Chapter 2. The chapter is divided into two main sections where approximate adders and approximate multipliers are classified and the prominent existing architectures are highlighted. Fair comparisons of approximate adders and multipliers are provided in Chapter 3. In this chapter, first, the problems of the state-of-the-art are discussed. Subsequently, a new metric is proposed for the evaluation of the accuracy of approximate computing unit considering a target

application. In Chapter 4, the proposed inaccurate adders are discussed. In the first Section of this chapter, a low-power optimized approximate adder is proposed. The proposed adder is designed systematically from a developed template for small-error approximate adders. In section 4.3, hybrid approximate adders are discussed based on the proposed unified design framework. A comprehensive template for approximate adders is proposed in this section. Afterwards, in section 4.4, the error behaviour of exact adders in stochastic regime is mathematically analyzed, where accordingly, a stochastic mixed adder is proposed. Truncated multipliers with data-dependent correction are subsequently proposed in Chapter 5. A template for truncated multipliers along with two correction strategies are presented in this chapter. In Chapter 6, the applicability of the proposed approximate architectures are evaluated in multiple case-studies. Finally, Chapter 7 concludes this dissertation.

CHAPTER 2

Background and Previous Works

2.1 Introduction

A huge number of approximate computing units have been proposed in the literature. Although approximate adders have been studied more than other approximate computing units, the attention of the researchers have been increasingly attracted to approximate multipliers in the recent years. In this chapter, the goal is to review the state-of-the-art and different approximation methodologies. The comparison of the approximate architectures, however, is left for the next chapter where we discuss the problems in the state-of-the-art. As a result, here, relevant approximate adders and approximate multipliers are classified and briefly reviewed. First, in the next section, different approximate adders with different design strategies are studied. Then, in section 2.3, different classes of approximate multipliers are investigated.

2.2 Approximate Adders

Addition of two binary numbers is one of the most basic arithmetic circuits in a digital computer. Adders, as one of the key components of arithmetic

circuits, have attracted lots of researchers' attention in the field of approximate computing. The most basic and fundamental adder architecture is the Ripple-Carry Adder (RCA). An n-bit RCA is constructed by n Full Adders (FAs), where the input carry of each full adder is fed by the output carry of the previous full adder; resulting in a long propagated carry chain. Correspondingly, the delay and silicon area of the RCA increase proportionally with n (O(n)). Another basic adder architecture is the Carry Lookahead Adder (CLA). In an n-bit CLA, there are n units which work in parallel to produce the generate ($g_i = a_i b_i$) and propagate signals ($p_i = a_i + b_i$). The generate and propagate signals are then employed to generate the look-ahead carries. The delay of CLA is logarithmic in n (O(log(n)) which is much shorter than delay of RCA. However, a CLA requires a larger Circuit area (O(nlog(n)), resulting in a higher power dissipation.

High speed adders are based on well established parallel-prefix architectures [58], including Brent-Kung [59], Kogge-Stone [60], Sklansky [61], Han-Carlson [62], etc. A Parallel-Prefix Adder (PPA) formulates a binary addition using three stages: pre-processing, prefix-processing and post-processing. The pre-processing stage computes generate g_i and propagate p_i signals using bit-wise operations:

$$g_i = a_i b_i, \tag{2.1}$$

$$p_i = a_i \oplus b_i, \tag{2.2}$$

where a and b are the inputs, and i denotes the bit position. The prefix-processing stage exploits the prefix operator, to speed-up carry computation. The prefix-processing stage is core of the parallel-prefix adders. Since the prefix operator is associative and idempotent, the individual operations can be carried out in any order. This has resulted to emergence of diverse existing parallel-prefix architectures.

The post-processing stage computes the sum bit s_i as follows:

$$s_i = p_i \oplus c_{i-1} \tag{2.3}$$

where $c_{-1} = C_{in}$ and $c_i = g_{[i:0]}$.

Various approximation schemes have been proposed in order to reduce the critical path and hardware complexity of exact adders. The concept behind all the approximate adders is to break the carry propagation chain and consequently perform the addition faster than exact adders. A conventional scheme to cut the carry propagation chain and reduce the delay of the critical path as well as the power dissipation is to use approximate full adders to implement the LSBs of an adder. This class of approximate adders are known as approximate full adders [16, 23, 63–66]. Another methodology is based on a speculative operation [21, 26, 67]. In a speculative adder, each sum bit is calculated using

its previous k LSBs, where k is less than n (k<n). Because of the fact that the carry chain is shorter than n, a speculative adder is faster than a conventional adder. A segmented adder is composed by several smaller sub-adders working in parallel [25,27,28,68,69]. Accordingly, the carry propagation chain is reduced into shorter segments. The segmentation technique is used in [70–74] with a similar concept. However, the carry input of each sub-adder block is selected with an altered strategy. Correspondingly, this class of approximate adders is called speculative carry selection adder. Therefore, the approximate adders are divided into four categories, which are briefly summarized below.

2.2.1 Approximate Full Adders

In this class of approximate adders, the adder is divided into two sub-adders. The higher significant sub-adder is an exact adder, while the lower significant is constructed employing approximate full adders. This class includes a simple approach of truncation. A truncated adder is an adder which its output LSBs are replaced with constant bits. In the case where the LSBs are replaced with constant ones we call it *TruncOne* and if the LSBs are replaced with constant zeros, it is called *TruncZero* in this dissertation. The Lower-part OR Adder (LOA) [23] is the most well-known design in this class. It divides an n-bit adder into two sub-adders. While the higher significant sub-adder is an $(n - n_{or})$-bit exact adder, the lower part sub-adder is simply constructed by n_{or} number of OR gates. To generate the Carry-in signal for the higher significant exact sub-adder, an extra AND gate is used which ANDs the most significant input bits of the lower significant sub-adder. The critical path delay of LOA then depends on the size of the exact sub-adder. The other figures of merit are also dependent on the exact sub-adder architecture. It is obvious that using different architectures as the sub-adder in LOA results in different performances as well as in different silicon areas and/or power consumptions. Fig. 2.1 shows the topology of a LOA. As can be seen, an n-bit LOA exploits a regular smaller precise adder that computes the precise values of the $(n_h - 1)$ most significant bits of the result along with OR gates that approximate the n_{or} least significant result bits by applying bit-wise OR to the respective input bits.

In addition, the approximate XOR/XNOR-based adders proposed in [63] is also classified as approximate full adders. The approximate adders proposed in [63] are based on using XOR and XNOR gates with multiplexers implemented by pass transistors. In the design of the approximate XOR/XNOR-based full adders, some of the transistors required in the accurate adder design have been removed to achieve lower logic complexity. In a similar approach, in [16], several approximate mirror adder designs are proposed by removing some of the transistors. They reported up to 60% power saving by approximating LSBs with this approach. In [66] majority logic based approximate full adders are proposed, and in [65] a proposed majority gate is employed to design an

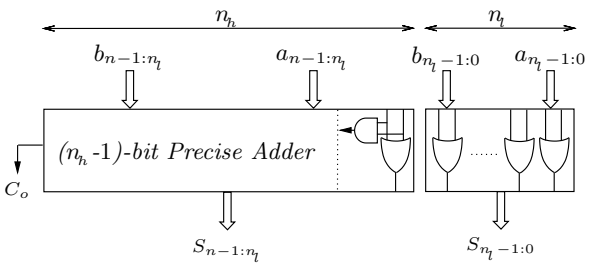


Fig. 2.1: Hardware architecture of the *L*ower-part OR Adder (LOA)

accuracy-configurable full adder cell. The above-mentioned designs except LOA, require customized layout. As a result, for the comparison, LOA is considered for our evaluations. In addition, the truncated adders are considered as baseline designs. Since the approximate full adders approximate the LSBs, they generate frequent errors. However, the errors generated by this class of adders are small in magnitude.

2.2.2 Speculative Adders

In a speculative adder, each sum bit is calculated using its previous k LSBs, where k is less than n (k<n). The Almost Correct Adder (ACA) [26], is the most applicable adder of this class designed based on [21]. The rationale for ACA design is that the probability of having $k-2$ Consecutive propagate bits is low enough to produce correct results for majority of the input combinations. However, for large k, the expected improvement in the speed is not achieved with ACA compared with an exact adder; and with small k, the accuracy does not correlate with the name of the adder. In addition, the area overhead of having multiple sub-adders is considerable in comparison with an exact adder. The ACA design can be seen in Fig. 2.2.

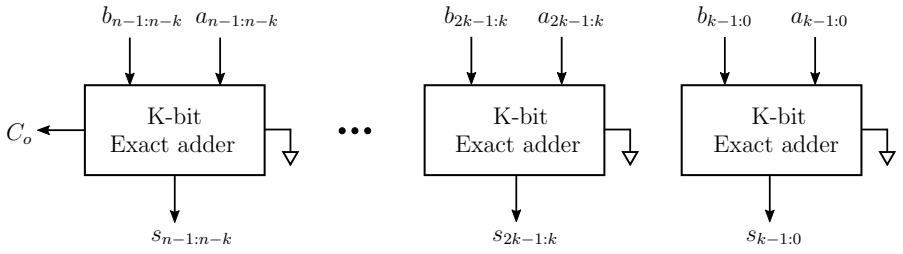


Fig. 2.2: Almost Correct Adder (ACA), k is the number of previous LSBs used for the calculation of the sum bits.

2.2.3 Segmented Adders

A segmented adder divides an n-bit adder into a number of smaller sub-adders which operate in parallel with fixed carry inputs. Let $\mathbb{K}=\{k_1, k_2, \ldots, k_s\}$ denote a vector including size of sub-adders, where s is the number of sub-adders. In Fig. 2.3, a segmented adder divided into s sub-adders is shown, where k_1 is the size of the first (the lowest significant) sub-adder, k_2 is the size of the second sub-adder, and so on.

The Equal Segmentation Adder (ESA) is a type of segmented adders with equally sized sub-adders, i.e. $k_1 = k_2 = \cdots = k_s$. Conventionally, ESA is considered as an n-bit adder divided into $\frac{n-k_1}{k}$ equally sized sub-adders in addition to the lowest significant sub-adder with the size k_1 where k is the size of equal sub-adders and $k_1 < k$. Accordingly, the delay and the area of an ESA is dependent to the structure of the sub-adders.

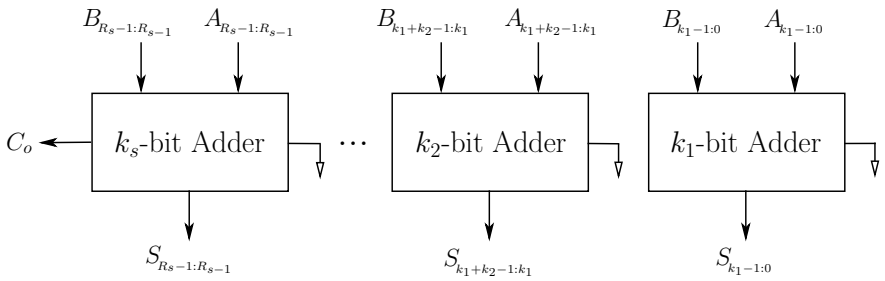


Fig. 2.3: The structure of the Segmented Adder divided into s sub-adders with arbitrary sizes, where $R_x = \sum_{i=1}^{x} k_i$.

ETAII, proposed in [28], is another approximate adder based on segmented adders. It splits the entire carry propagation path into a number of short paths and completes the carry propagations in these short paths concurrently. Let us consider the general case of the ETAII with arbitrary block sizes. As depicted in Fig. 2.4, the architecture of ETAII is divided into smaller blocks. Each block has an arbitrary number of bits and, different from ESA, consists of two separate circuitries - Carry Generator and Sum Generator. As the name implies, the Carry Generator creates the carry-out signal. It does not take the carry signal from the previous block. The Sum Generator, however, takes the carry-in signal from the previous block to generate its sum output bits. Consequently, the carry propagation only exists between two neighboring blocks instead of lying along the entire adder structure [28]. Conventionally, ETAII is divided into $\frac{n-k_1}{k}$ equally sized blocks, in addition to the lowest significant block with the size k_1. Although the circuit complexity of ETAII is similar to ESA (due to the fact that sub-adders in ESA consist both carry and sum generators), considering

parallel prefix adder as the internal sub-adders, ETAII has one extra stage in its prefix structure which exacerbates its speed.

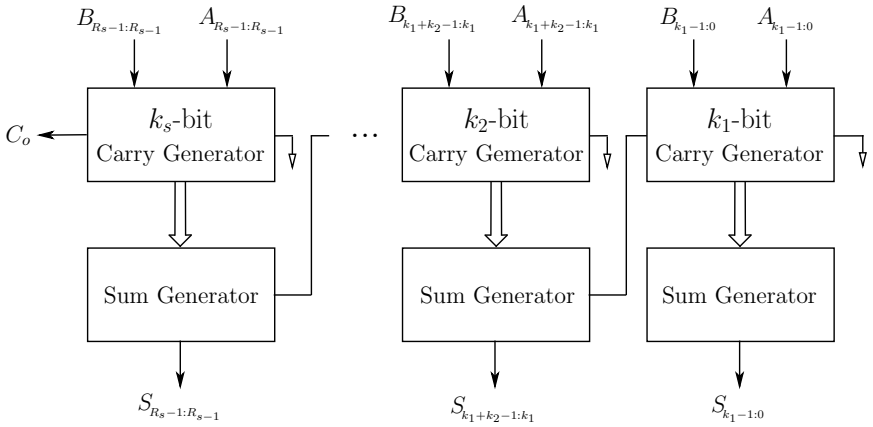


Fig. 2.4: The structure of the ETAII divided into s sub-blocks with arbitrary sizes, where $R_x = \sum_{i=1}^{x} k_i$.

A low latency generic accuracy configurable (GeAr) adder [27] is a generalized design which employs the concept of segmented adders. It uses multiple sub-adder units of equal length to provide a wide-range of accuracy configurability and variable approximation modes. The GeAr provides a higher number of potential configurations compared to state-of-the-art, and provides a template which enabling a high degree of design flexibility and trade-off between performance and output quality.

The error behaviour of segmented adders is completely different from approximate full adders. The segmented adders, due to the method which they cut the carry chain, make big infrequent errors.

2.2.4 Speculative Carry Select Adders

A speculative carry select adder is an approximate adder divided into sub-adders where the carry inputs of the sub-adders are selected with different methodologies. The speculative carry selection adder (SCSA) [71] is divided into $\lceil \frac{n}{k} \rceil$ blocks called "window". Each window adder consists of two k-bit exact adders where the input carry of $Adder_0$ is connected to zero and the input carry of $Adder_1$ is connected to One. The sum result of each window adder is selected with a multiplexer which its select input is connected to the output carry of $Adder_0$ of the previous (lower significant) window adder.

The approximate architecture proposed in [72] uses the generate signals for carry speculation. Here, the carry selection depends on the propagate signal of each sub-adder. In case the propagate signal is one, the carry-in of the block is

the most significant generate signal of the previous block; and if the propagate signal is zero, the carry-in of the block is the carry-out of the previous carry generator block.

The Carry Cut-Back Adder (CCBA) [74] is composed of sub-adder blocks where multiplexers are inserted in order to cut the carry chain to reduce the critical path delay. The cut multiplexes the real carry with a carry speculated from a much shorter chain. The decision to cut the chain is taken in the carry propagate block (PROP) which monitors a group of carry stages and generates the cut signal if those are all in propagate states. The accuracy of the CCBA adder depends on how large is the bit-width between the propagate block and the cut.

The speculative carry select adders usually provide a high accurate results. However, the high circuit overheads of those adders make them less interesting than the approximate full adders and segmented adders.

2.3 Approximate Multipliers

Multiplication is the main operation in many signal processing applications. Multipliers are the most power-hungry units of Arithmetic and Logical Units (ALUs). In most of the DSP applications, multipliers are part of the critical path and hence determine the speed of the system. A multiplier consists of three stages: 1) partial product generation, 2) partial product reduction (accumulation), 3) a final addition, as shown if Fig. 2.5(b).

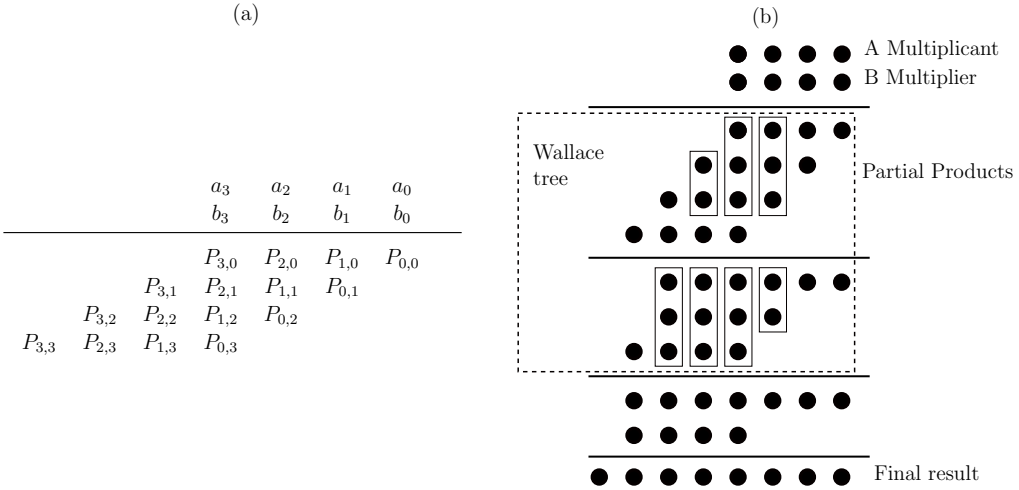


Fig. 2.5: (a) The partial product tree, $P_{i,j} = a_i b_j$. (b) A 4×4 Wallace multiplier.

Considering A and B as the inputs of a multiplier, partial products of the multiplier $P_{i,j}$ are the products of the two bits (a_i and b_i) generated by AND

gates (i.e. $P_{i,j} = a_i b_j$), where i and j are the bit position of inputs a and b, respectively. Since in binary multiplication b_i is in {0, 1}, each row of the partial product tree can be either 0 or $a \times 2^j$ which is the shifted version of input "a". Because of this fact, the multiplication sometimes is considered as the combination of shift and add operations.

The frequently used partial product reduction strategies are the carry-save adder array, Wallace and Dadda trees [75,76]. A carry save adder array (Fig. 2.6) employs a row of binary FAs to reduce three numbers to two numbers. The carry and sum signals produced in each row are fed to the next row of the FAs. Since the adders which are in a column work in a serial order, the partial product reduction with carry save adders is relatively slow. The multipliers using carry save adder array as the partial product reduction stage are called Array multipliers. The Wallace and Dadda trees [77, 78] employ (3,2) and (2,2) counters (i.e. Full Adder (FA) and Half Adder (HA), respectively) at each level of partial product accumulation to achieve required reduction. Dadda multipliers use a minimal number of FAs and HAs at each level, while in Wallace trees, the number of operands are reduced at the earliest possibility. In other words, if there are h dots in a column, $\lceil \frac{h}{3} \rceil$ FAs are immediately applied to that column. This results in minimizing the overall delay by making the final addition as small as possible.

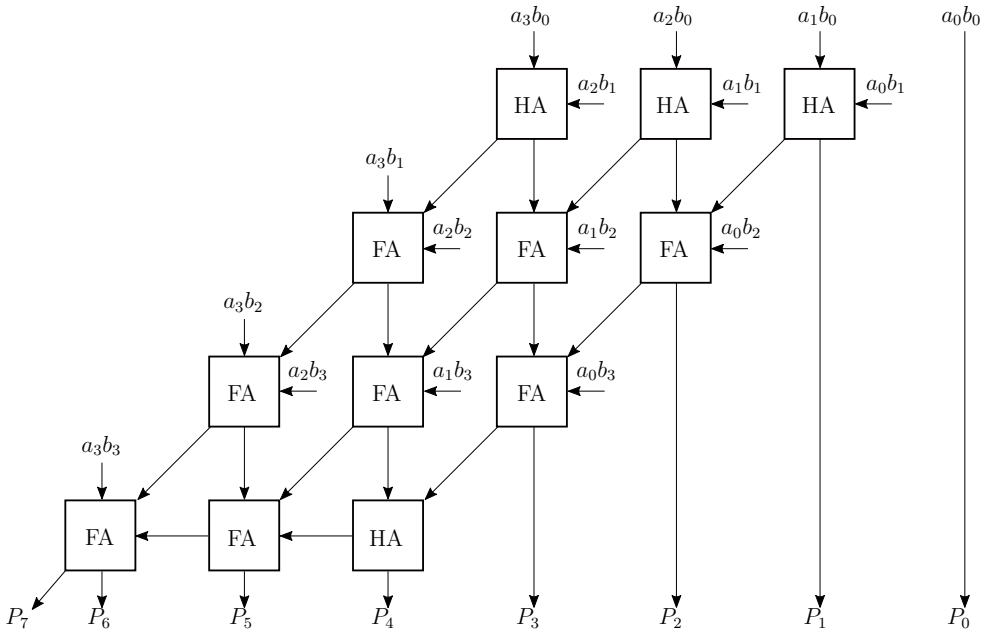


Fig. 2.6: The structure of partial product reduction of a 4×4 unsigned array multiplier.

The approximate multipliers can be classified in four main classes. The

approximation in generating the partial products is the first class of approximate multipliers [24]. Another class of approximate multipliers designs a multiplier by approximating the partial product tree [22, 23, 79–82]. This class of multipliers include the conventional truncated multipliers. The third class of approximate multipliers employs approximate blocks for accumulation of the partial products. In [83], an approximate multiplier with error correction is proposed where approximate counters are used for the reduction of the partial products. There are additionally a huge number of designs in the literature where approximate compressors are employed to accumulate the partial products of the multipliers [84–90]. Approximate multipliers can also be classified as the units developed using evolutionary design approaches [29–33].

2.3.1 Approximation in Generating the Partial Products

The most straightforward approach of this class is to utilize smaller approximate multipliers to construct larger multipliers [24, 91] which results in approximated partial products. In [24], the proposed underdesigned multiplier employs an approximate 2×2 multiplier. The approximate 2×2 multiplier is designed to make error only when both the inputs are one (i.e. "11"). In this case, the multiplier output is "111" instead of the precise result of "1001", saving one output bit. Consequently, considering a uniform distribution, where each input bit has 50% probability to be 0 or 1, the error rate of the 2×2 approximate multiplier is $\frac{1}{16}$. The error introduced by this approximate multiplier generates approximate partial products, while the accumulation of the partial products is still performed precise.

2.3.2 Approximation in the Partial Product Tree

The simplest approach in this methodology is input truncation where k-bit LSBs of the input operands are pruned. In fact, the multiplication is performed for the Most Significant Bits (MSBs) of the operands. This approach results in a relatively high level of inaccuracy. Conventional truncated multipliers are the eminent architectures of this class of approximate multipliers. A truncated multiplier (TruncM) pruns the k LSBs (k least significant columns) of the partial products resulting the k LSBs of the output of the multiplier in constant zeros. Removing the partial products of a multiplier not only reduces the silicon area due to the removal of AND gates, but also scales the speed of the multiplier up as a result of cutting the critical path. Moreover, the area reduction benefits is twofold since the reduced size partial products require lesser hardware to be accumulated. Broken-Array Multiplier (BAM) [23] is a type of truncated multipliers which has a structure similar to an array multiplier. The BAM omits the Carry-Save Adder (CSA) cells horizontally and vertically which lead to a smaller and faster circuit while provides inaccurate results. As shown in Fig. 2.7 the number and position of the omitted cells (that are hatched) depend

on two parameters: Horizontal Break Level (HBL) and Vertical Break Level (VBL). The BAM technique results in a same accuracy when applied to other multipliers such as Wallace-tree. The Error-Tolerant Multiplier (ETM) [22] splits a multiplier into LSBs and MSBs. A control unit checks the product of the MSBs (using NOR gates). If the product of MSBs is zero, then the LSBs are multiplied normally. But if the product of the MSBs is nonzero, a non-multiplication block is used for the the LSBs [22].

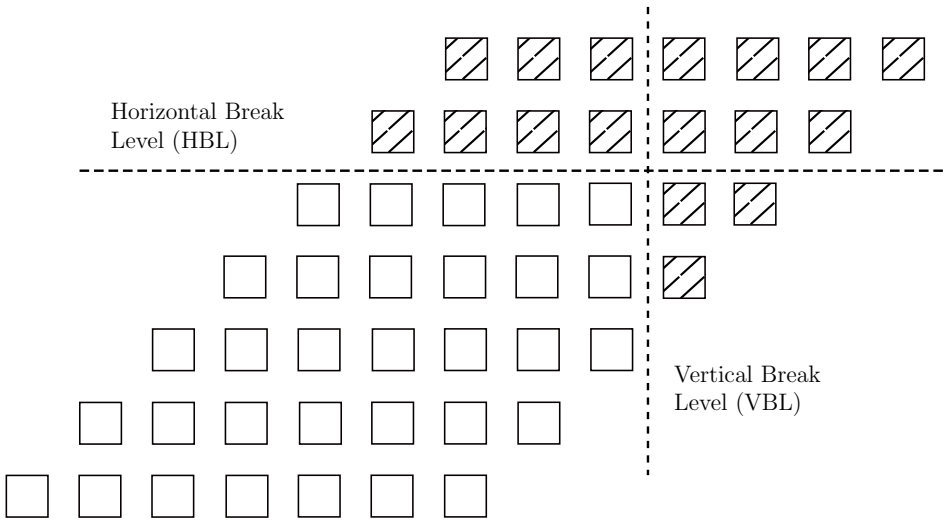


Fig. 2.7: The structure of a 7×7 Broken Array Multiplier (BAM). The hatched carry-save adders are the omitted cells (Horizontally and/or Vertically).

2.3.3 Approximate Compressors

In the design of a fast multiplier, compressors have been widely used to speed up the partial product reduction tree and decrease power dissipation [92–97].

Two approximate compressors are proposed in [84] and then used for the partial product accumulation of a Dadda multiplier. These approximate compressors are designed by simplifying the outputs of the compressor and thereby decreasing the transistor count of the compressors. In [87], a simple but novel idea is proposed to design approximate compressors. In addition, an algorithm to exploit the designed approximate compressors in the partial product reduction stage is presented. The compressors designed in [87] approximate the arithmetic sum. For most of the input combinations, they compute the exact value resulting in a low average error. The outputs of the proposed approximate compressors have the same weight as of the inputs and they do not produce carry outputs, which is the difference from conventional compressors where the weight of the carry output is two times the weight of the inputs.

In [98], a new approximate adder cell is proposed. The adder operates on a set of pre-processed inputs. This input pre-processing is done considering the interchangeability of the input bits with the same weights. In the input pre-processing stage, the rule is to swap a_i and b_i when $b_i > a_i$, otherwise the inputs are kept untouched. By doing so, more 1's are expected in A and more 0's are expected in B. As a result, the pre-processed inputs can be calculated using the following equations:

$$\begin{aligned} \dot{a}_i &= a_i + b_i, \\ \dot{b}_i &= a_i b_i. \end{aligned} \tag{2.4}$$

The Eq. (2.4) corresponds to the propagate and generate signals used in a parallel prefix adders. The proposed adder can process data in parallel by cutting the carry propagation chain. A carry signal is produced in the proposed adder only by the generate signal (i.e., $a_i = b_i = 1$), which based on the pre-processed inputs, it corresponds to when $\dot{b}_i = 1$. In addition, the produced carry can only be propagated to the next higher bit. The truth table of the approximate adder cell is tabulated in Fig. 2.8(a).

(a)

$s_i e_i$		$\dot{b}_i\dot{b}_{i-1}$ 00	01	10	11
$\dot{a}_i\dot{a}_{i-1}$	00	0/0	X	X	X
	01	0/0	1/0	X	X
	10	1/0	1/1	1/0	0/0
	11	1/0	X	X	0/0

(b)

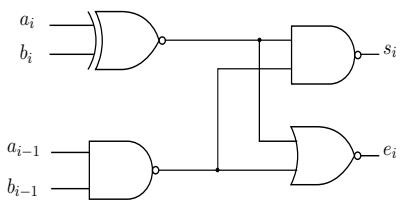


Fig. 2.8: (a) Truth table of the approximate adder cell proposed in [98]. 'X' represents the combinations which are not possible to occur due to the input pre-processing; (b) approximate adder cell of [98].

The idea of [98] is extended in [99] where besides using the approximate adder cell of Fig. 2.8(b), to achieve further improvement in area and power consumption of the multiplier, truncation is applied to the partial products tree. Depending on the error recovery circuit, the multipliers are called AM1 for the error recovery with OR gates, and AM2 when error recovery circuit employs OR gates and the approximate adder cell of Fig. 2.8(b). The truncated version of these adders are called TAM1 and TAM2, respectively.

2.3.4 Multipliers Derived by Optimization Algorithms

Another approach to design approximate computing units is optimization algorithms such as genetic algorithm [29–33]. EvoApprox library [33] is developed based on conventional adder and multiplier designs. The method used to obtain the library follows the the evolutionary approach introduced in [32]. The methodology is a general-purpose approach to develop approximate combinational circuits based on a multi-objective Cartesian genetic programming. It represents candidate circuits as directed acyclic graphs and develops a set of approximate circuits along a Pareto front by simultaneously optimizing delay, power consumption and accuracy of the approximate unit.

2.4 Conclusion

In this chapter, relevant approximate adders and multipliers have been reviewed. An extensive comparison of these architectures are presented in the next chapter. In general, the appoximate full adders, due to applying the approximation to the LSBs, generate small errors but with a high probability. Segmented adders, on the other hand, produce large erros with low probability. In regards to multipliers, applying approximation in the generating the partial products and also in the partial product tree by truncation are effective strategies to reduce the hardware cost. However, the truncation of the partial products results in a higher accuracy than truncating the inputs operands. The multipliers which use approximate compressors in their structures provide the higher accuracy with the cost of higher energy consumption as well as silicon area.

The existing research works optimize an approximate computing unit for different trade-offs among accuracy, silicon area, energy efficiency and perfomrmance. Although in each of those designs marginal improvements might have been achieved, due to inconsistency in the error analyses and circuit characterisitcs, it is challenging to choose a proper architecture for a desired application. In the next chapter, we study the problems in the state-of-the-art in detail and we discuss our approaches to address those problems.

CHAPTER 3

Identification of Problems in State-of-the-art

3.1 Introduction

The languishing benefits of technology scaling has pushed the designers to look for new energy efficient strategies. One approach is approximate computing which has attracted lots of researchers' attention. A vast amount of time and resources of the research companies have been dedicated to approximate-computing-related projects. Correspondingly, a prodigious number of approximate designs have been proposed in the literature. However, it is still difficult to choose a suitable approximate design for a target application. Indeed, there are four fundamental problems which are aimed to be addressed in this dissertation. These problems are listed below:

1. **Systematic design approaches:** The existing approximate designs in the literature share a common characteristic: they have been obtained with an ad-hoc and non-systematic methodology. Although there are some exceptions (like GeAr adder which is based on the concept of a

template), a majority of the designs aim to achieve an improvement based on different trade-offs among energy efficiency, silicon area, performance and accuracy. Our approach, however, is to study the approximate architectures systematically. Rather than improving any existing architecture for a target application or for a specific trade-off, we aim to characterize the error behaviours of the approximate architectures. Afterwards, based on these understandings, architectural templates are developed. Finally, optimized architectures can be obtained from the templates. The promising architectures obtained with our approaches are detailed in Chapter 4 and Chapter 5.

2. **Fair comparison:** As mentioned above, the interest of stochastic and approximate processing has resulted in a huge number of publications and contributions; each work claiming a superior functionality over the others. The comparisons among the alternatives, however, are not always systematic and fair. On the one hand, different synthesis tools and technologies are employed for the evaluations; on the other hand, the metrics employed to quantify the quality of the approximate designs are not methodical. There is also lack of systematic studies comparing approximate and stochastic architectures. Indeed, the comparisons are mostly deficient. The rules for a fair comparison are often violated in the literature. For example, the selection of an approximate architecture is strongly influenced by the timing constraints of the hardware [48]. The design constraints determine the internal structure of the unit (in case of adders for example, sequential structure with linear cost, parallel-prefix structure with a logarithmic cost, etc.), which determines the reduction in cost achieved by the approximate units. With relaxed timing constraints, an exact adder is implemented with a ripple-carry structure and an ESA almost decreases the delay by a factor of two. With more stringent timing constraints, an exact adder is implemented with a parallel-prefix architecture, where the delay increases as $\lceil \log_2(n) \rceil$ and the use of an ESA is just marginally reducing the delay. In this case, non-equally segmented sub-adders may be preferable. The effects of the internal adder architecture have been studied in [48]; it illustrates the potentials for non-equally segmented approximate adders which have been disregarded in the literature. In majority of the research works, the proposed design is compared with sub-optimal designs, and the significant baseline designs are missing in the comparisons. For a fair comparison, it is vital to consider all the best possible counterparts. This simple fact has often been violated in the literature, and even in the review articles, which can be misleading for the research community. The aforementioned shortcomings make it difficult to choose a suitable approximate design

for a target application. In this chapter, a fair comparison of adders as well as multipliers are presented and discussed.

3. **Reproducible research:** Reproducible research is an important factor to accelerate advances in any research domain. Despite the fact that approximate computing has received significant attentions, in the digital VLSI design community, in the past decade, there is still no comprehensive library which can be used as a reference for comparisons as well as reproducible research. Indeed, lack of the library of a comprehensive approximate computing unit has slowed down the research in this domain. In conjunction with this dissertation, we have developed an open-source repository where the Very high speed integrated circuit Hardware Description Language (VHDL) codes of the approximate computing units and the golden references are stored. The comparison of approximate computing units can be accessed there for different figures of merit. In addition, the users can add their own designs in the repository. This way, not only the researchers in this domain can access to the code and descriptions of the existing architectures, but also they can evaluate and compare their own architectures with the existing ones on the developed platform. Due to the fact that the architectures are classified and also categorized based on their entities, the fair comparison is ensured using the developed platform. Above all, this way the comparisons are reproducible.

4. **Metrics:** A closer investigation into the variety of approximate units shows that the current approaches use two philosophies for the error: a) *small errors* or b) *unlikely errors*. In the first philosophy, the errors of the approximate unit are engineered to be small in magnitude, even if they are frequent. The rationale is that those errors are masked by the intrinsic truncation and noise error of the system, and therefore they do not degrade considerably the quality of the application. Examples of this philosophy are the LOA [23] and truncated adders and multipliers. In the second philosophy, the errors are engineered to appear infrequently, even if they are large when they appear. The rationale is that the application can overcome errors if they are sporadic. Examples of this philosophy are the ACA [68], the GeAr [27], the ETAII [28] and the ESA [25].

   A key problem when considering simultaneously the two philosophies is the quantification of the errors. The authors working with the small errors philosophy tend to prefer error metrics as the standard deviation, or the mean average error that measure the average magnitude of the errors. This metric, however, strongly penalizes large infrequent errors. The authors working with the infrequent error philosophy tend to favor metrics as the average number of errors which quantify the error probability;

however, this metric strongly penalizes architectures with small errors. In a real scenario, the two effects have to be considered in a single metric, but it is not possible with the current approaches.

In the rest of this chapter, first, a new metric to quantify the quality of approximate computing units when employed in real applications, is proposed. Afterwards, fair comparisons for existing approximate adders and multipliers for different trade-offs are presented and discussed.

3.2 Metrics

The error is defined as the difference between approximate and accurate output results of the computing unit:

$$\varepsilon = \tilde{Y} - Y, \tag{3.1}$$

where $\tilde{Y}$ is the approximate (erroneous) output of the unit and Y is the accurate result. It is a random variable that can be characterized by its probability density function (i.e. $Pr[\varepsilon_j]$). However, from the perspective of an automated design framework, it is more convenient to use an error metric (a single number) to quantify the importance of the error. Several metrics have been proposed; among them, the most common ones are the *Average Error* (μ), the average number of errors (PE), the *Standard Deviation* (STD or σ), the *Mean Squared Error* (MSE), the *Mean Absolute Error* (MAE), and the *Mean Relative Absolute Error* (MRAE). In the literature (e.g. [100]) the Error Distance and the Mean Error Distance (MED) are used to evaluate the arithmetic performance of approximate computing units. These metrics are actually the absolute error and the MAE, respectively. References [101, 102] use MED and error rate (similar to PE), while references [103–105] use relative error metrics. In summary, the most common metrics are defined as follows:

$$PE = \mathbb{E}\big[\delta(\varepsilon)\big] = \sum_j Pr[\varepsilon_j] \ , \tag{3.2}$$

$$\mu = \mathbb{E}\big[\varepsilon\big] = \sum_j \varepsilon_j Pr[\varepsilon_j] \ , \tag{3.3}$$

$$\sigma = \sqrt{\mathbb{E}\big[(\varepsilon-\mu)^2\big]} = \sqrt{\sum_j (\varepsilon_j-\mu)^2 Pr[\varepsilon_j]} \ , \tag{3.4}$$

$$MSE = \mathbb{E}\big[\varepsilon^2\big] = \mu^2 + \sigma^2 \ , \tag{3.5}$$

$$MAE = \mathbb{E}\big[|\varepsilon|\big] = \sum_j |\varepsilon_j| Pr[\varepsilon_j] \ , \tag{3.6}$$

$$MRAE = \mathbb{E}\big[|\varepsilon|\big] = \sum_j \Big|\frac{\varepsilon_j}{y_j}\Big| Pr[\varepsilon_j] \ , \tag{3.7}$$

where $\mathbb{E}$ is the expectation operator. It should be mentioned that it is also common to employ the normalized version of the previous metrics.

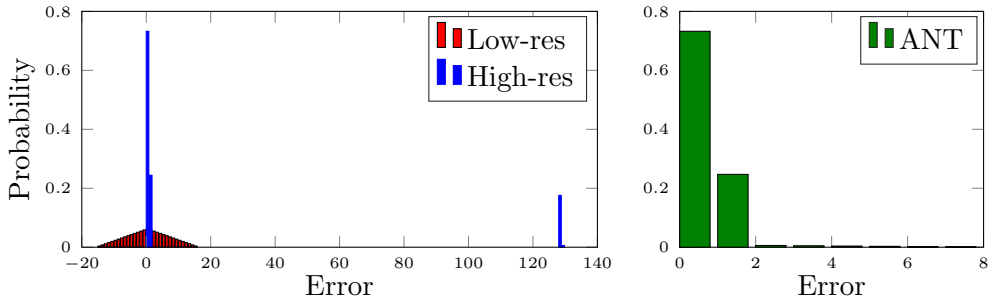


Fig. 3.1: Error histograms of a low-resolution input, a high-resolution input and an ANT output.

The metric PE favors the *infrequent error* philosophy, while Standard Deviation (STD) and Mean Squared Error (MSE) favor the *small error* philosophy. The Mean Absolute Error (MAE) is more robust than MSE to the presence of large errors and it is commonly used as a compromise. However, it is insufficient. Let us consider a typical stochastic approach as the Algorithmic Noise Tolerance (ANT) approach [106]. In ANT, two versions of an algorithm are computed; one with a low-cost and low-resolution unit and another one with a high-resolution approximate processing unit. Then, the two outputs are compared; if the difference is small, the approximate processing output is chosen, otherwise the low-resolution one is chosen. After this selection process, the quality of the system is notably improved. An example is shown in Fig. 3.1, where the low-resolution signal uses 5 effective bits and the high-resolution signal is generated with a hybrid adder. The error histogram of the ANT output shows the quality of the resulting signal. Beside the simplicity of this example, none of the current metrics can be reliably used to quantify the error-cost of the high-resolution approximate processing needed by ANT. To illustrate this fact, Table 3.1 reports the error metrics (PE and MSE) of the signal produced by three different adders as well as the final signal quality measured as the MSE at the output of the ANT process. Analyzing only the MSE, ESA-4 should be the best adder, while analyzing the PE, ETAII-3 is preferable. In fact, a hybrid adder, whose MSE and PE are worse than the previous examples, provides the best solution. Accordingly, current metrics are misleading.

The key problem is that the current metrics do not capture the different behaviors of small and large errors. Low-magnitude errors get added to the final output and can be quantified with the MSE or STD; high-magnitude errors are

Table 3.1: Error metrics for approximate adders and resulting output quality after ANT processing

Adder	ANT input MSE	ANT input PE	ANT output MSE
ESA-4	10.9545	0.4688	7.4330
ETAII-3	14.9666	0.0547	0.8101
Hybrid	19.6405	0.2676	0.7539

detected by ANT and replaced by the low-resolution version of the algorithm. Thus, they contribute with a factor dependent on the error probability (PE) but relatively independent of the actual error magnitude. To palliate the lack of expressiveness of current metrics, we propose a new parameterizable metric that captures the requirements of stochastic applications, the *Saturated Mean Squared Error* (SMSE_τ). Mathematically, it is defined as:

$$SMSE_\tau = \mathbb{E}\big[\min(\tau, |\varepsilon|)^2\big] \tag{3.8}$$

The parameter τ controls the behavior of the metric; large values of τ produce a cost similar to the MSE, while small values produce a cost similar to PE. As shown with more details in Chapter 6, the new metric captures more precisely the error cost and can be used to explore different approximate units.

In order to evaluate our new metric, we have compared the adders in a simple image processing algorithm which first calculates the average of the pixels using approximate adders and then the error probability is calculated after binarization. In Fig. 3.2, the probability of errors for 14 different 8-bit adders are shown versus the calculated error metrics. The colors correspond to different families of adders. The red color is the precise adder, the green ones are different configurations of ETAII (including the non-equal segmented ones) and the dark blue ones are ESA adders as the representatives of infrequent error philosophy, the light blue ones are OLOCA [47] architectures as the representative of small error philosophy. As can be seen in the figure, our saturated metric predicts the errors in this algorithm more precisely. There is an almost linear relationship between quality (probability of errors) and our saturated metric. The conventional metrics are misleading in some cases. For instance, in Fig. 3.2(b), the adder marked with $*$ is among the best configurations with the lowest probability of errors; but, conventional MSE cannot predict this and misleads to ignore the adder.

In a similar scenario, two images can be multiplied pixel by pixel. Here, in this example, the Astronaut image is multiplied with a *windowed image*.

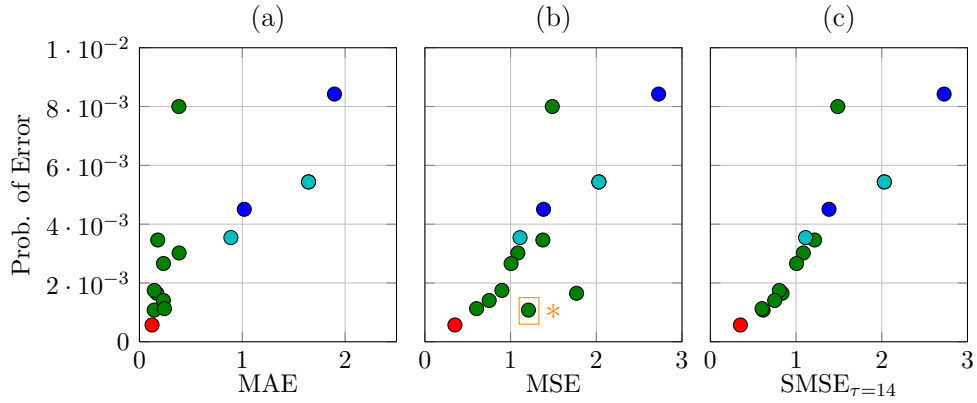


Fig. 3.2: The error prediction of different metrics in an image processing algorithm. Different colors correspond to different classes of approximate adders. The star * is used to mark an adder as an example to show a case where conventional metrics are misleading.

The windowing can be employed to smoothly reduce the amplitude of the signal as it reaches the edges, removing the effect of the artificial discontinuity which results from the Fast Fourier transform (FFT). The input images as well as the resulting images from exact and approximate multipliers are shown in Fig. 3.3. The approximate multipliers are TruncWM-4, TruncM-8, and BAM-3. The TruncWM-4 is a truncated Wallace multiplier where the 4 LSBs of its input operands are pruned; the TruncM-8 is a truncated multiplier where truncation is applied to the 8 LSBs of its partial products; and BAM-3 is the Broken Array multiplier where 3 rows (horizontal) and 3 columns (vertical) of its partial products are removed. As can be seen in Fig. 3.3, the BAM multiplier outperforms the other multipliers in terms of output image quality, while the TruncWM is providing the least quality. The quality of output images are quantified with the Structural Similarity Index Measure (SSIM) as tabulated in Table 3.2. The multipliers are also compared using MSE and MAE metrics as well as our proposed metric Saturated Mean Squared Error (SMSE), tabulated in the Table 3.2. It can be observed that the conventional metrics

Table 3.2: Error metrics for approximate multipliers and resulting output quality after multiplication of two images

	$SSIM$	MSE	MAE	$SMSE_{\tau=256}$
TruncMW-4	0.845	1.131e6	8.807e2	58.979e3
TruncM-8	0.877	0.261e6	4.477e2	56.741e3
BAM-3	0.927	1.669e6	9.542e2	52.448e3

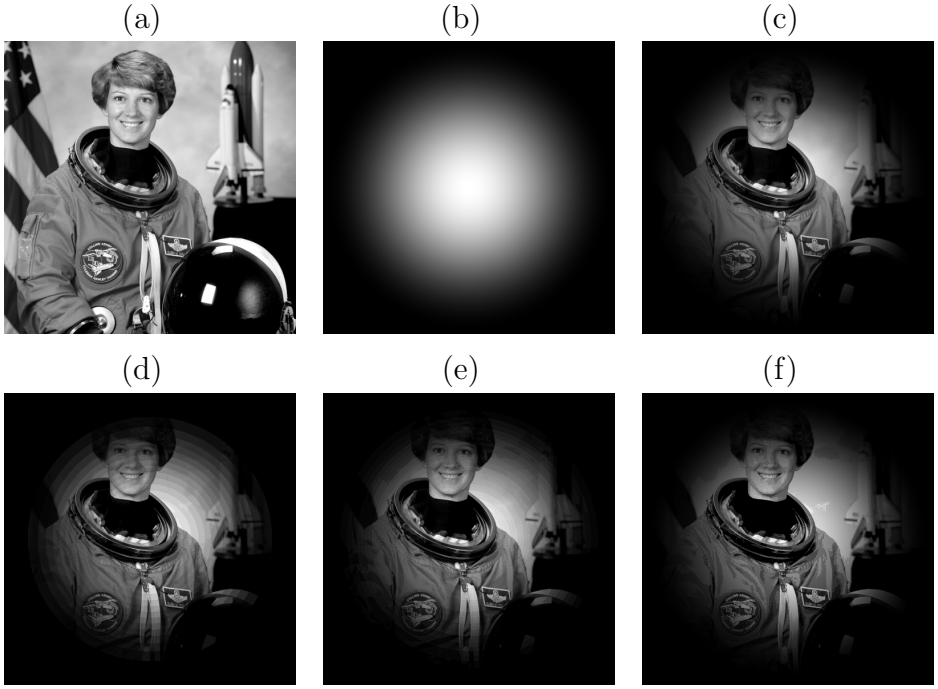


Fig. 3.3: Comparison of multiplication of (a) the *Astronaut* image with (b) the Window image using (c)exact multiplier, (d)input truncated Wallace multiplier (TruncWM-4), (e)truncated multiplier (TruncM-8), and (f)Broken array multiplier (BAM-3).

are again misleading in this application. Based on the MSE and MAE metrics, the TruncM-8 should have the best quality among the compared approximate multipliers, while as our metric suggests, the BAM-3 is outperforms the other two approximate architectures in this application (see Table 3.2).

3.3 A Fair Comparison of Approximate Adders

To assess the circuit characteristics and compare the existing approximate architectures reviewed in Chapter 2, the VHDL description of the adders have been generated. Different configurations of these adders are synthesized in a commercial low-power 40nm library, for 16-bit operands. Using back-annotated simulations, dynamic power dissipation of the adders are evaluated after synthesis. All the adders have been simulated for 10^7 uniformly distributed random input patterns. Besides, in order to include the stochastic analysis of exact adders, frequency over-scaling has been used as the technique to trade-off the accuracy and hardware cost.

In this section, in order to provide a comprehensive comparison of the adders, they are compared for different figures of merit as well as different accuracy metrics. All the adders share the same entity to ensure a fair comparison. As

discussed in [48], considering the input carry and the output carry ports in the entity of the adder architectures can considerably impact on the accuracy the architectures. In this work, input carry port is not considered in the entity of the adder architectures, while the bit-width of the output of the adders are 17 bits, which considers the output carry. In addition, the adder architectures are compared for various timing constraint to consider the impact of internal architectures in the behaviour of the approximate adders.

The approximate adders which are compared in this section are the ESA, ETAII, LOA, ACA, and GeAr. In addition, the conventional truncated adders are considered as the baselines for the comparison. TruncZero is a truncated adder where the pruned LSBs are constant zeros. The TruncOne, on the other hand, replaces the pruned LSBs of the adder's output by constant 1's. The 16-bit approximate adders which are compared in this chapter are configured as follows: the number of approximated LSBs for LOA and truncated adders are depicted for $n_l = 4$ to $n_l = 10$; the ESA and ACA adders are configured for $k = 4$ to $k = 8$; the ETAII is configured for $k = 4$ to $k = 8$; and finally regarding the GeAr adder, the set of resultant and previous bits are (2,4),(4,4),(4,8), and (6,4). Note that, the 16-bit ETAII adder with $k = 8$ operates like an exact adder, as a result it is not considered.

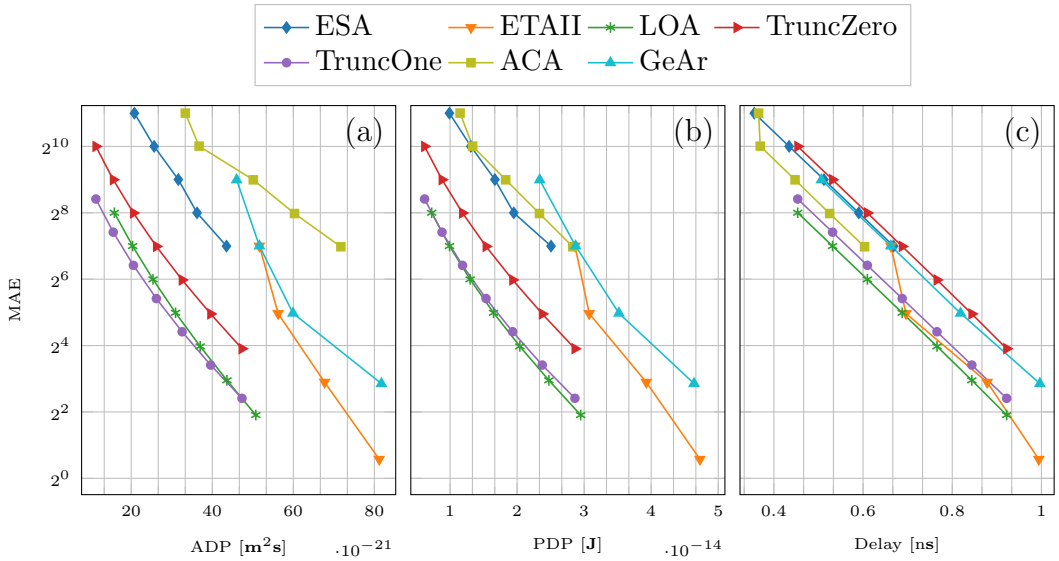


Fig. 3.4: Comparison of the area-optimized approximate adders for their Mean Absolute Error (MAE) vs. (a)Area-Delay Product (ADP), (b)Power-Delay Product (PDP), (c)Delay.

The comparison of the aforementioned approximate adders for different error metrics are shown in Fig. 3.4, Fig. 3.5, Fig. 3.6, and Fig. 3.7. These figures show

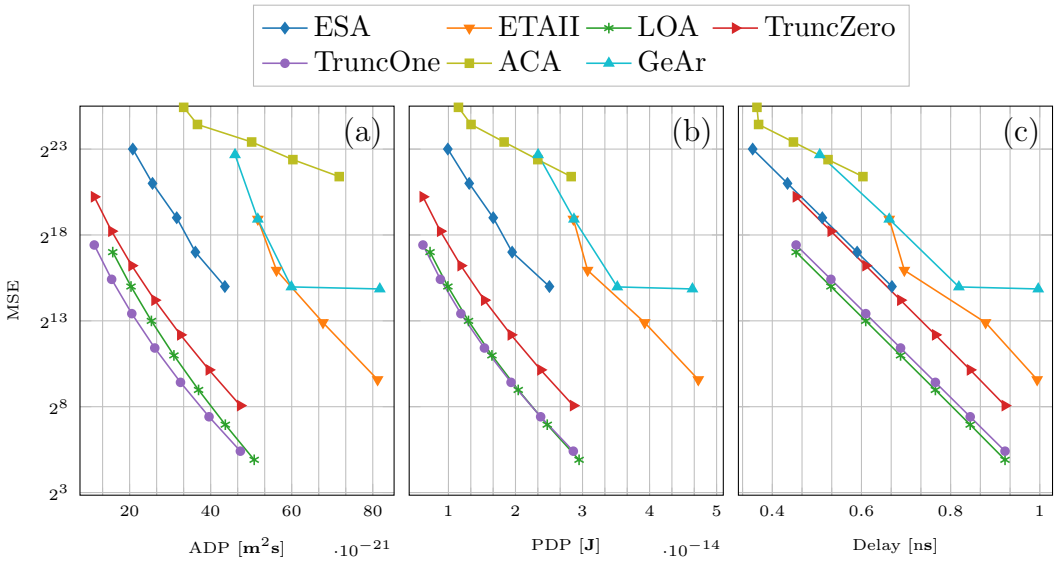


Fig. 3.5: Comparison of the area-optimized approximate adders for their Mean Squared Error (MSE) vs. (a)Area-Delay Product (ADP), (b)Power-Delay Product (PDP), (c)Delay.

the comparison of the area-optimized adders (a relaxed timing constraint).

It can be seen that only considering one single figure of merit (either speed, silicon area, or power consumption) is not sufficient for the comparison. As can be seen, in the Fig. 3.4 for instance, the ETAII architectures considering delay and accuracy shows a promising architecture which outperforms the other architectures for most of its configurations. However, when considering the silicon area, and power consumption, the ETAII has a worse behaviour that a truncated adder. This is also valid for only considering silicon area, or power consumption as the cost metric. Correspondingly, the Power-Delay Product (PDP) and Area-Delay Product (ADP) represent fair hardware cost metrics to compare the approximate architectures.

Comparing Fig. 3.4 and Fig. 3.5, it can be observed when the goal is to achieve energy efficiency (lower PDP), considering MAE, the ACA and ESA architectures have almost the same characteristics. However, when MSE is considered as the accuracy metric, the ESA adder performs better than ACA. The other notable observation is the error rate (PE) of the adders. The small error adders such as LOA, and truncated adders, due to the approximating of the LSBs, generate errors frequently. As a result, as can be seen in Fig. 3.7, those adders have a probability of errors close to one. This is particularly important for the applications where the number of errors impact on the quality of the results.

It can be concluded that, the selection of metrics is important to choose an

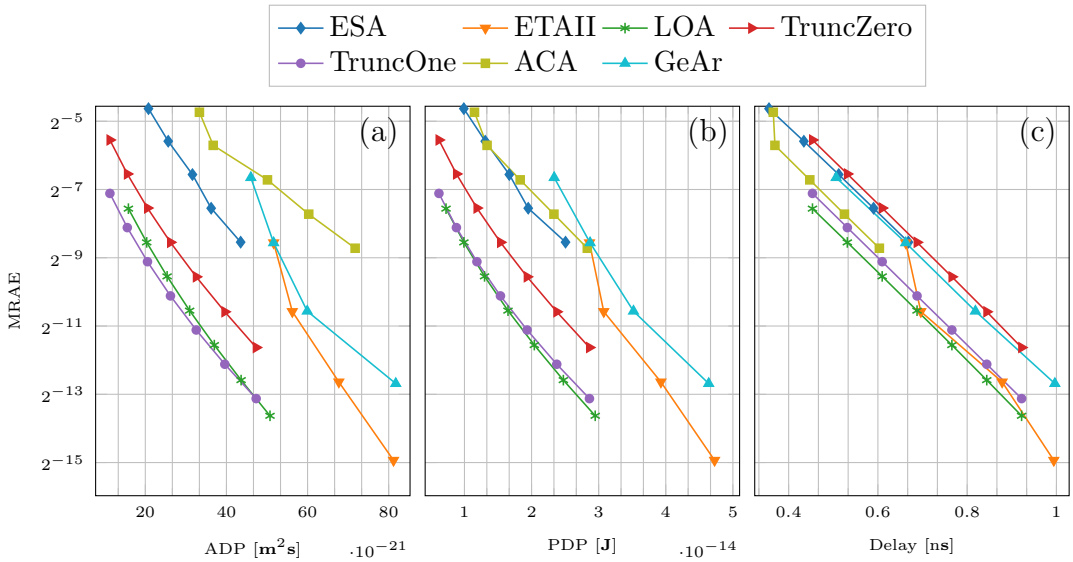


Fig. 3.6: Comparison of the area-optimized approximate adders for their Mean Relative Absolute Error (MRAE) vs. (a)Area-Delay Product (ADP), (b)Power-Delay Product (PDP), (c)Delay.

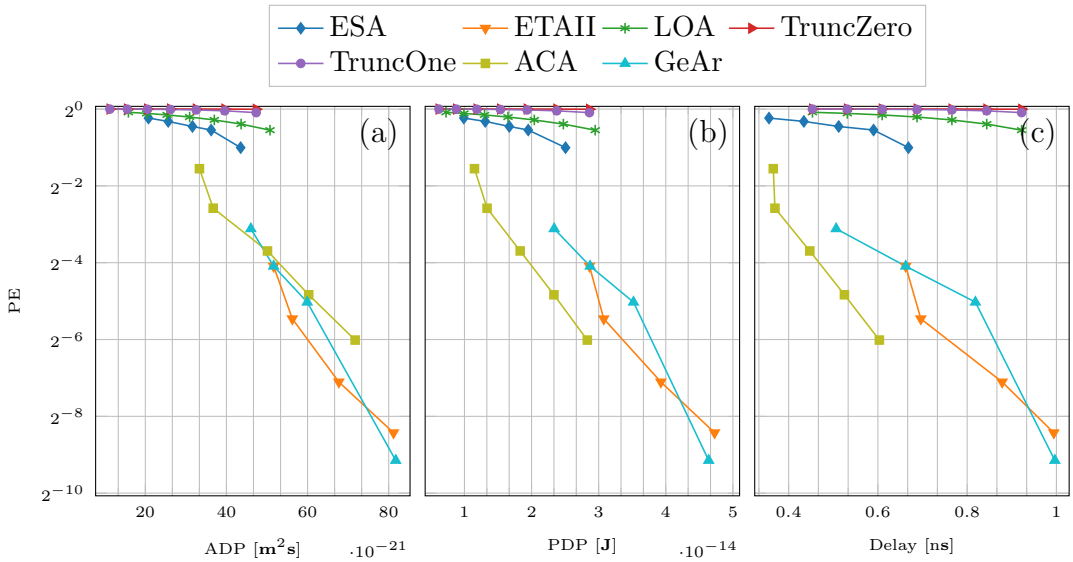


Fig. 3.7: Comparison of the area-optimized approximate adders for their error rate (PE) vs. (a)Area-Delay Product (ADP), (b)Power-Delay Product (PDP), (c)Delay.

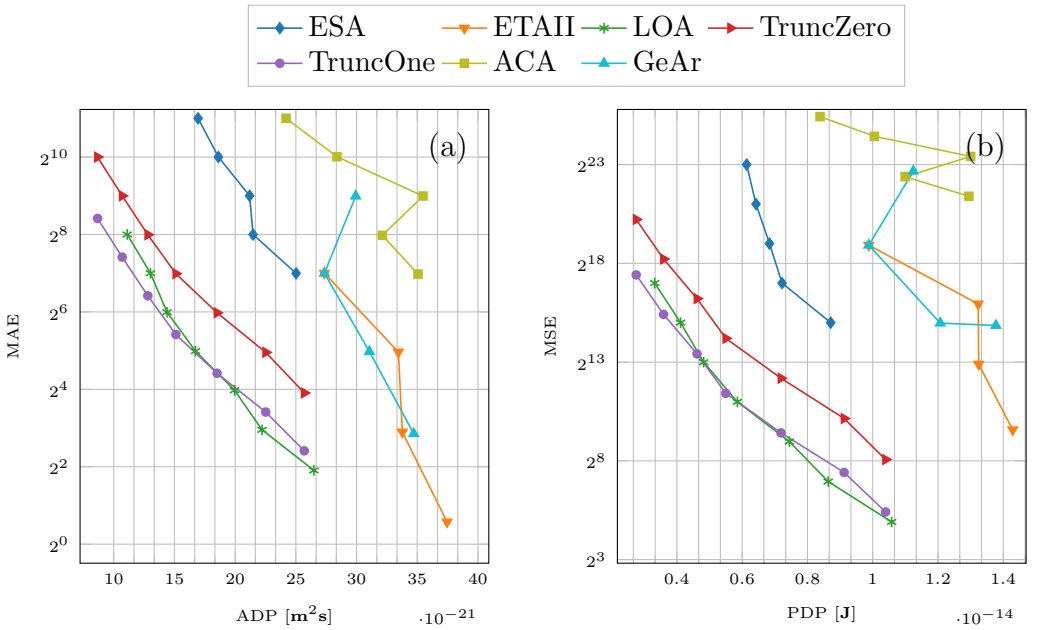


Fig. 3.8: Comparison of the approximate adders in a stringent timing constraints of 0.25ns: (a) Mean Absolute Error (MAE) vs. Area-Delay Product (ADP), (b) Mean Squared Error vs. Power-Delay Product (PDP).

approximate architecture for an application. In fact, to show that a proposed architecture outperforms the others for only one specific metric does not guarantee that the architecture is indeed a superior architecture. This fact has been violated in majority of the contributions in the approximate architecture design domain. In addition, it is important to choose appropriate baselines for the comparisons. As can be observed in the figures, the truncated adders are promising architectures which are often disregarded in the literature.

In order to analyze the impact of internal architectures on the behaviour of the approximate adders, we have synthesized them for various timing constraints. In Fig. 3.8, the comparison the approximate adders for a stringent timing constraint of 0.25ns is depicted. Since, changing the timing constraint changes the netlist of the adders, resulting in different internal architectures, dissimilar trade-offs are expected to be observed. For example, in Fig. 3.8, the GeAr adder, for some configurations, outperforms the ETAII; while considering a relaxed timing constraint (see Fig. 3.4 and Fig. 3.5, ETAII performs better that the GeAr adder for all the configurations.

For a better understanding of the impact of the design constraints on the approximate adders' trade-off, let us consider an application with error tolerance of $MAE < 16$. As a result, we choose the the adder architectures which meet

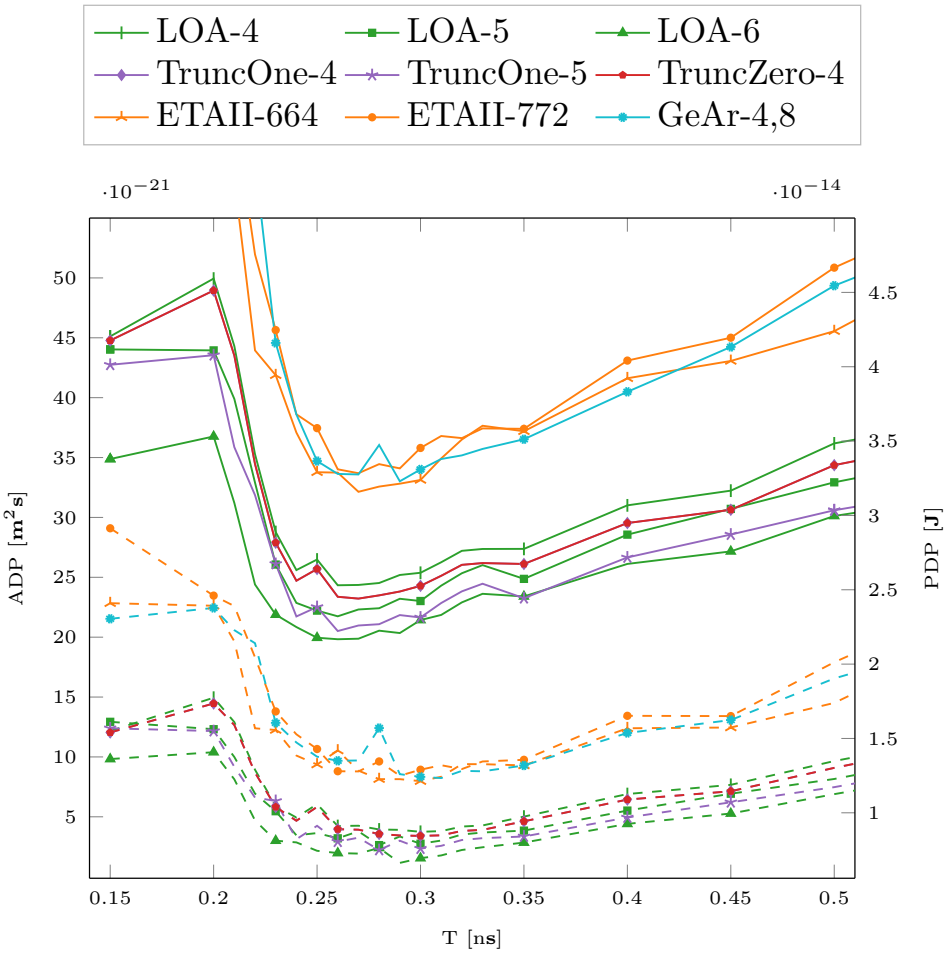


Fig. 3.9: Comparison of adders with $MAE < 16$ for different timing constraints. The dashed lines are the Power-Delay Product (PDP) of the adders, while the solid lines illustrate the Area-Delay Product (ADP) of the adders.

this restriction from Fig. 3.4. These architectures are compared for various timing constraints in Fig. 3.9. The numbers following the name of the adders in this figure are their configurations. For the LOA, and truncated adders, the number is the approximated LSBs; in the case of ETAII, the numbers are the bit-widths of the sub-modules from most significant module to the least significant module, from left to right, respectively; and for the GeAr adder, the left number is the width of the resultant bits and the right number is the number of previous bits used to calculate the results. Note that, throughout the whole range of the timing constraints (the whole line for each adder), the error of the approximate adders are unaltered. As a result, the crossing lines in the figure indicate that superiority of the architectures in terms of the hardware

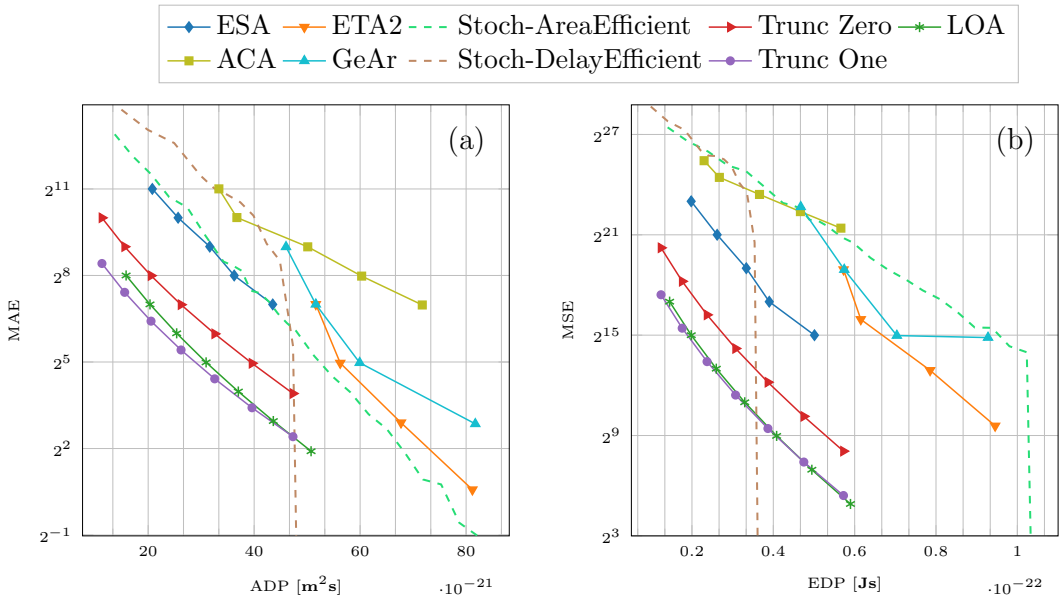


Fig. 3.10: Comparison of approximate adders and stochastic adders. (a) Mean Absolute Error (MAE) vs. Area-Delay Product (ADP), (b) Mean Squared Error (MSE) vs. Energy-Delay Product (EDP).

cost is changing. This might affect the trade-off between the accuracy and hardware cost of approximate adders, and consequently affecting the decision of choosing an architecture for a specific application.

As a matter of fact, the internal architecture plays an important role in the behaviour of adders in the stochastic regime. In Fig. 3.10, the comparison of the approximate adders and stochastic adders are illustrated. The stochastic adders are exact adders synthesized considering two different timing constraint and simulated for overscaled frequencies: a relaxed timing constraint (Stoch-AreaEfficient), and a rigorous timing constraint (Stoch-DelayEfficient). The Stoch-AreaEfficient adder is realized as a RCA, while the Stoch-DelayEfficient has a structure very close to parallel-prefix adders.

As observed in the figure, when considering the trade-off between accuracy and energy efficiency, the approximate adders are better choices than the stochastic adders. Note that the stochastic adder Stoch-DelayEfficient because of its abrupt increase in MSE and MAE is not an attractive selection. On the other hand, when considering the trade-off between accuracy and ADP, even the Stoch-AreaEfficient is a better selection than a majority of the approximate adders.

3.4 A Fair Comparison of Approximate Unsigned Multipliers

To assess the circuit characteristics and compare the existing approximate multipliers reviewed in Chapter 2, the VHDL description of the multipliers have been generated. Different configurations of these multipliers are synthesized in a commercial low-power 40nm library, for 16-bit operands. Using back-annotated simulations, dynamic power dissipation of the adders are evaluated after synthesis. All the multipliers have been simulated for 10^7 uniformly distributed random input patterns. Besides, the stochastic frequency-over-scaled multipliers are included in the comparisons, in order to study the approximate and stochastic multipliers simultaneously.

In this section, in order to provide a comprehensive comparison of the unsigned multipliers, they are compared for different figures of merit as well as different accuracy metrics. All the multipliers share the same entity to ensure a fair comparison. Accordingly, the bit-width of the product of the compared multipliers are 32 bit. In addition, similar to the previous section, the multipliers are compared for various timing constraint to consider the impact of internal architectures in the behaviour of the approximate multipliers.

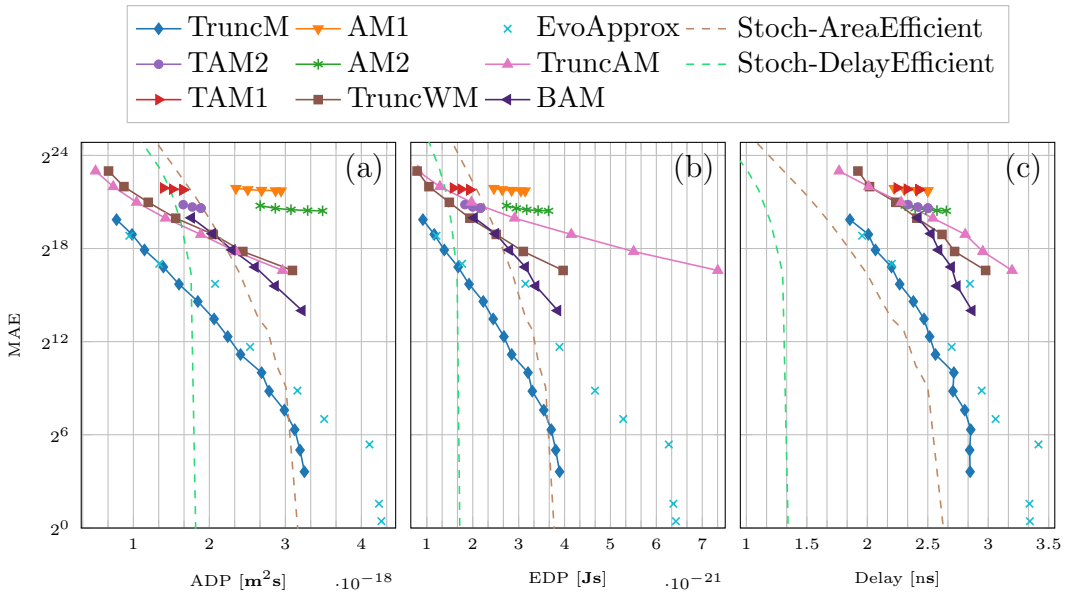


Fig. 3.11: Comparison of the area-optimized approximate multipliers and stochastic multipliers for their Mean Absolute Error (MAE) vs. (a)Area-Delay Product (ADP), (b)Energy-Delay Product (EDP), (c)Delay.

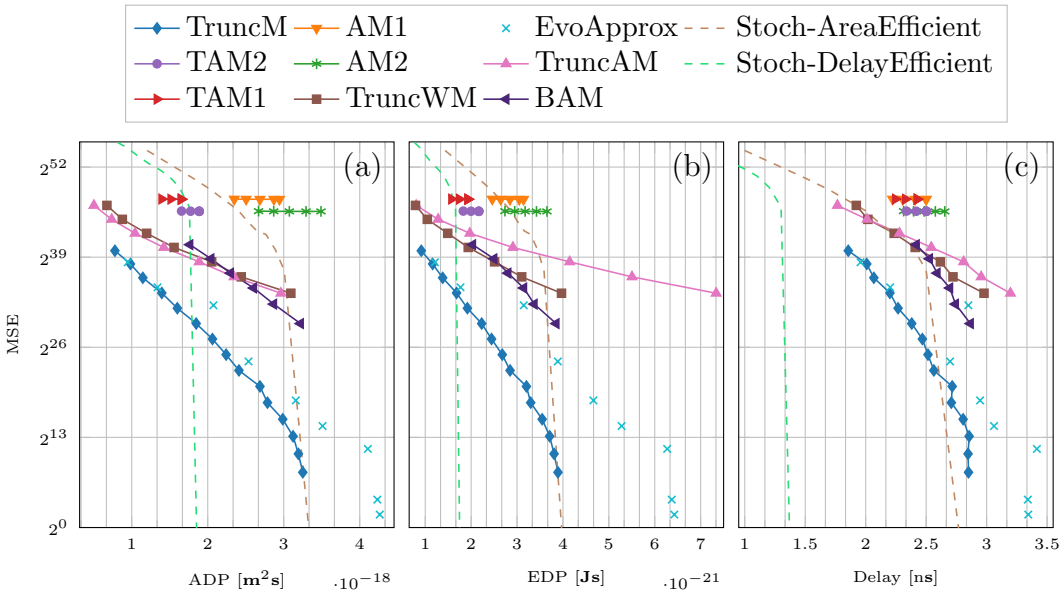


Fig. 3.12: Comparison of the area-optimized approximate multipliers and stochastic multipliers for their Mean Squared Error (MSE) vs. (a)Area-Delay Product (ADP), (b)Energy-Delay Product (EDP), (c)Delay.

The approximate multipliers which are compared in this section are the AM1, AM2 proposed in [98], the TAM1, TAM2 proposed in [99], the BAM [23], input-truncated Wallace multiplier (TruncWM), input-truncated array multiplier (TruncAM), and automatically generated EvoApprox multipliers [33]. In addition, the conventional truncated multiplier (TruncM) is considered as the baseline for the comparison. TruncM approximates the multiplication by pruning the k LSBs from the partial products, while in case of TruncAM and TruncWM k bits from the LSBs of their input operands are truncated. In regards to the BAM multiplier, we consider that equally k bits from both the rows (horizontally) and columns (vertically) of the partial products are pruned.

The 16-bit approximate multipliers which are compared in this chapter are configured as follows: the number of approximated LSBs for AM1 and AM2 are from $k = 12$ to $k = 16$; the TAM1 and TAM2 are configured for $k = 12$ to $k = 15$; the TruncWM and TruncAM are truncated with $k = 2$ to $k = 8$; the TruncM prunes the LSBs from $k = 4$ to $k = 18$; the BAM is horizontally and vertically truncated from $k = 1$ to $k = 6$, and finally the EvoApprox multipliers are the Pareto optimal selections from the EvoApprox database [33]. In addition, the stochastic multipliers, Stoch-DelayEfficient and Stoch-AreaEfficient, are included in the comparisons to concurrently analyze the existing approximate and stochastic multipliers. The Stoch-AreaEfficient

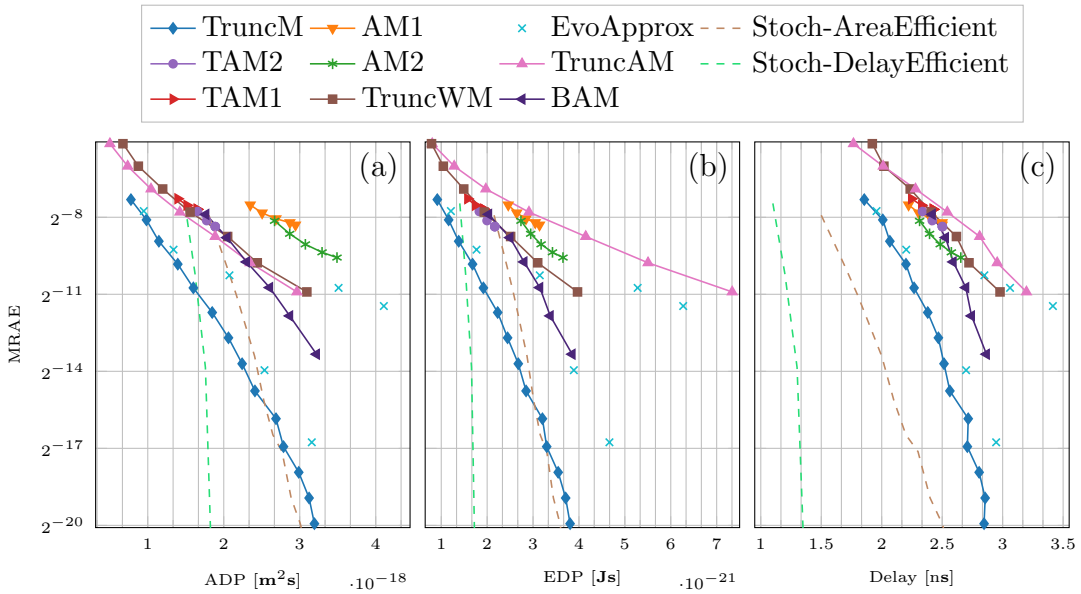


Fig. 3.13: Comparison of the area-optimized approximate multipliers and stochastic multipliers for their Mean Relative Absolute Error (MRAE) vs. (a)Area-Delay Product (ADP), (b)Energy-Delay Product (EDP), (c)Delay.

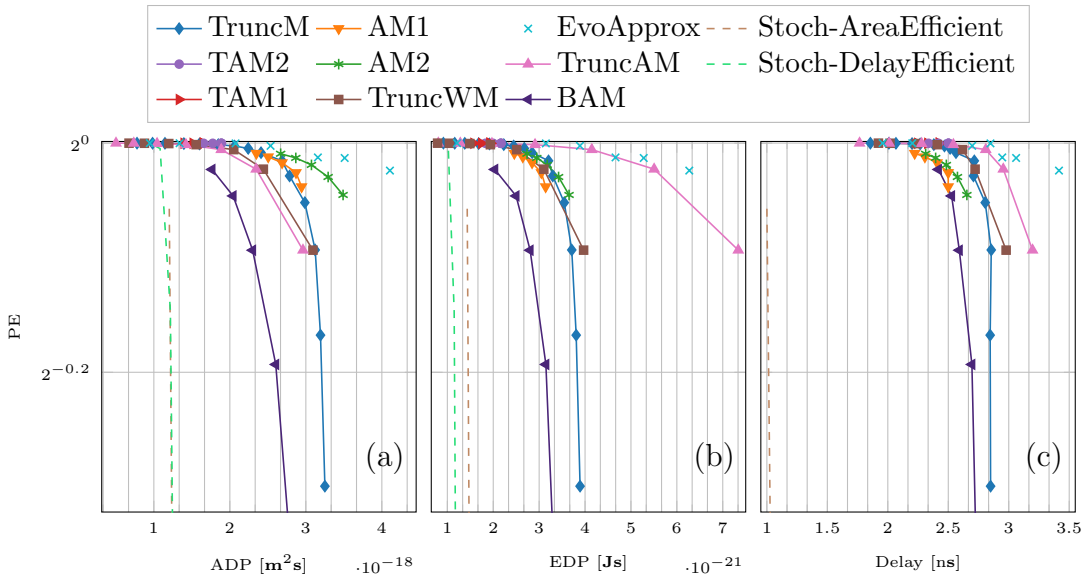


Fig. 3.14: Comparison of the area-optimized approximate multipliers and stochastic multipliers for their error rate (PE) vs. (a)Area-Delay Product (ADP), (b)Energy-Delay Product (EDP), (c)Delay.

and the Stoch-DelayEfficient multipliers are exact multipliers synthesized for a relaxed timing constraint and a rigorous timing constraint, respectively, which are simulated for over-scaled frequencies.

The comparison of the aforementioned multipliers for different error metrics are shown in Fig. 3.11, Fig. 3.12, Fig. 3.13, and Fig. 3.14. Note that the approximate multipliers are area-optimized (synthesized for a relaxed timing constraint).

Here, similar to the adders' comparison, it can be observed that, only considering one single figure of merit is not sufficient for a fair comparison. For example, considering only the speed of the multipliers, AM1 and TAM1 have the same trade-off, while considering the PDP and ADP, the TAM1 outperforms AM1, as can be seen in Fig. 3.11. The same is valid for AM2 and TAM2 multipliers. Similarly, as can be seen in Fig. 3.13, when considering the delay and MRAE trade-off, the AM2 multiplier outperforms the TruncWM, while considering the silicon area, and power consumption of the multipliers, the TruncWM is performing better than the AM2. Moreover, the Stoch-AreaEfficient multiplier, regardless of its abrupt error increment in higher frequencies, outperforms a majority of the multipliers in Fig. 3.13(b); but when considering the MAE or MSE, it generates very large errors in the stochastic regime.

Comparing Fig. 3.11, Fig. 3.12, and Fig. 3.13, it can be observed that, the TAM1 and TAM2 architectures have significantly higher MAE and MSE than the truncated multipliers, while their MRAE are comparable with truncWM. There are also evident differences between the MAE and MRAE of some EvoApprox multipliers, as can be observed in the figures. It is notable that considering the error rate (PE), the BAM and AM1 multipliers can be considered as the superior architectures. Consequently, choosing a right metric is decisive for the selection of the multipliers for a target application. For a fair general and comprehensive comparison, it is essential to consider different metrics. This fact has been repeatedly violated for the comparison of multipliers in the literature [107]. Such contributions despite being valuable can be misleading for the research community.

It is also fundamentally important to select right architectures for the comparison. For instance, as can be seen in the figures, the TruncM offers the best accuracy versus hardware cost trade-off. Consequently, it should be chosen as the baseline for the comparison of the multipliers. However, in majority of the contributions in the literature, it has been disregarded.

To study the impact of the timing constraint on the behaviour of multipliers, a selection of the multipliers are compared for different timing constraints in Fig. 3.15. It can be observed that, unlike the a relaxed timing constraint, the hardware cost of the selected multipliers are pretty close to each other for more stringent timing constraints. This observation is specifically more relevant,

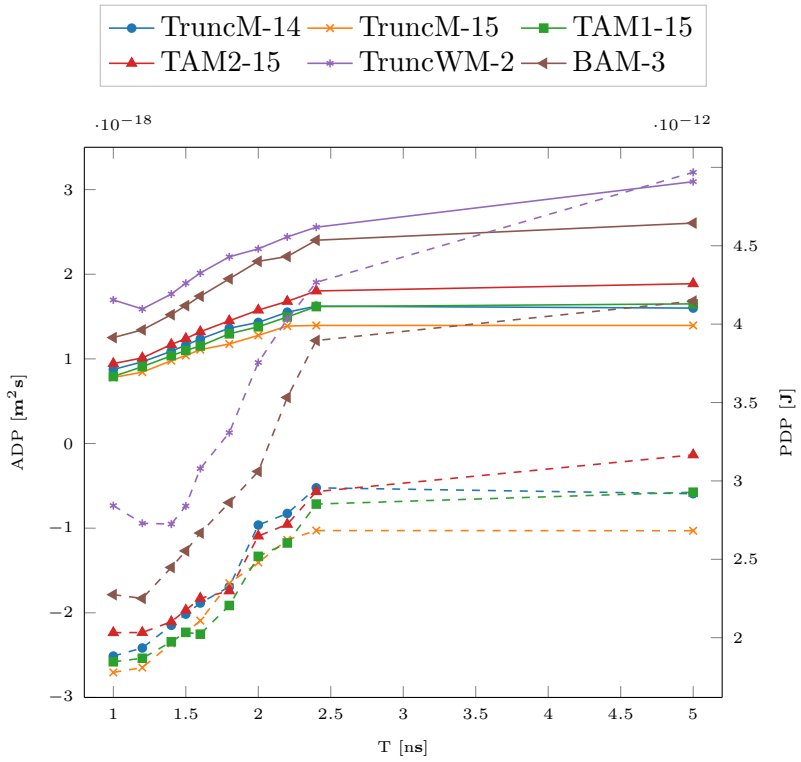


Fig. 3.15: Comparison of Area-Delay Product (ADP) and Power-Delay Product (PDP) of a selection of multipliers for different timing constraints. The dashed lines are the PDP of the multipliers, while the solid lines illustrate the ADP of the multipliers.

when the architectures have similar error behaviour or meet the accuracy requirements of the application. Note that, the impact of timing constraint on the multipliers in Fig. 3.15 is less chaotic than what observed for the adders in Fig. 3.9. The rationale is that the approximation strategy of the compared multipliers are analogous.

3.5 Conclusion

In this chapter, we have discussed the main deficiencies in the existing research works and our approaches to address those shortcomings. It has been shown using practical examples how the conventional metrics can be misleading in choosing approximate architectures for an application. In response, we proposed a new parameterizable metric called SMSE. The proposed metric predicts the output accuracy of the whole application with a higher precision than the conventional metrics such as MSE or MAE. Subsequently, along with presenting fair comparisons of approximate adders and approximate multipliers,

we discussed the prerequisites of a fair comparison. It has been shown that in a comparison figures of merit, accuracy metrics, timing constraint, and the entity of the architectures play important roles and must be well defined. Based on the comparison of approximate adders, it can be seen that the LOA outperforms the other architectures considering the trade-off between hardware cost and accuracy. In regards to multipliers, the conventional truncated multiplier (TruncM) shows the best trade-off among the existing approximate multipliers. Nonetheless, the LOA and TruncM have been ignored in majority of the research works. Taking the results of the comparisons into consideration, in the next chapters, we systematically improve the best existing architectures to obtain obtimized designs.

CHAPTER 4

Inaccurate Adders

4.1 Introduction

One of the fundamental arithmetic operations in many applications is addition. The full carry chain of an adder determines its critical path. A precise adder considers all the input bits to calculate the final carry out signal. Indeed, in real scenarios, the effective carry chain is much shorter than the full carry chain. The rationale is that the inputs to the adder are not uniformly distributed. This property can be exploited to design approximate adders with much shorter carry chains than that of an exact adder.

The researchers in the field of approximate computing have paid special attention to adders, one of the key components of arithmetic circuits. In fact, a surprisingly large number of approximate adders [16, 23, 25, 27, 28, 68, 71, 73] have been proposed in the literature: segmented adders where an n-bit adder is divided into k-bit sub-adders [25, 28, 68]; carry select adders in which multiple sub-modules are used [71, 73], approximate full adders where the full adder

is approximated [16, 23] and speculative adders which are built upon the observation that the critical path is rarely activated in traditional adders [21, 26, 108]. The current situation is such, that even a fair comparison of approximate adders is a challenging endeavor [48,103]. As mentioned in Chapter 3, a majority of the existing approximate adders, despite their conceptual differences, share a common characteristic: they have been obtained with an ad-hoc and non-systematic methodology. A remarkable exception is the Generic Accuracy Configurable Adder (GeAr) that uses the idea of template [27] but is not optimal.

In this chapter, the main aim is to tackle the non-systematic approaches of existing research works. In addition, we study different classes of approximate adders to understand the error behaviours of them. Correspondingly, first, in the next section, with generalizing a template based on LOA, we systematically propose an optimized approximate adder which shows upto 58% lower MSE than LOA. Note that, as observed in Chapter 3, cosidering the trade-off between MSE and hardware cost, LOA outperfoms the other existing approximate adders. We further progress in the third section by developing a template for approximate adders. The precise error formulas are presented along with the template. The proposed template covers different classes of approximate adders including hybrid structures which combine the two error philosophies discussed in the previous chapter. In the last section of this chapter, we study exact adders working in the stochastic regime. By developing precise models, we discuss why those stochastic adders have received less attention in comparison with the approximate adders. Finally, we propose a mixed adder which works promising in the stochastic regime and successfully overcomes the problems of its counterparts.

4.2 Optimized Lower-part Constant-OR Adder

As discussed in Chapter 3, among all the purely combinatorial approximate adders, the *L*ower-part OR Adder (LOA) [23] shows the best error versus hardware-cost trade-off [48,103]. As can be seen in Fig 2.1, LOA [23] divides an n-bit adder into two sub-adders. While the higher significant sub-adder consists of an (n_h-1)-bit exact adder, the lower part sub-adder is simply constructed by n_l OR gates (bits 0 to n_l-1). To generate the carry-in signal for the accurate adder, an extra AND gate is used which combines the adder inputs of bit position n_l (i.e., a_{n_l} and b_{n_l}). The key advantage of LOA with respect to other architectures as Equal Segmentation Adder (ESA) [25], Error Tolerant Adder (ETAII) [28] or Almost Correct Adder (ACA) [68] is that the approximation is restricted to the least significant bits and therefore, the magnitude of the errors is limited.

The goal of this section is to improve LOA systematically. First, we generalize

the LOA architecture in the form of an architectural template; then, studying all the possible choices to implement that template, we obtain an *optimal* architecture for the presented template focusing on Mean Squared Error (MSE). We call it *O*ptimized Lower-part Constant-OR Adder (OLOCA). The optimized architecture outperforms all the existing approximate adders when considering the trade-off between hardware-cost and accuracy. The experimental evidence reported in this section corroborate this fact.

Following the aforementioned goals, this section is organized as follows: subsection 4.2.1 describes the structures of the architectural template and of OLOCA. Afterwards, in subsection 4.2.2, we quantify the advantages of OLOCA using experimental results; furthermore, the mathematical formulas developed in next subsection will be validate.

4.2.1 Architecture

To obtain systematically an optimal[1] approximate adder, we progress in three steps. First, we describe the error metrics and hardware-cost quantifying the quality of the architecture; second, we generalize the architecture of LOA into a more abstract template; third, we optimize the template, regarding MSE, to produce OLOCA.

Metrics

To quantify the quality of the approximate architectures, error metrics need to be considered. The error is defined as the difference between approximate and accurate output results of the adder, i.e.,

$$\varepsilon = \tilde{S} - S, \tag{4.1}$$

where $\tilde{S}$ is the approximate (erroneous) output of the adder and S is the accurate result. The magnitude of the error can be quantified with several metrics; among them, the most common ones are the *Average Error* (μ) (Eq. (3.3)), the STD (Eq. (3.4)), the MSE (Eq. (3.5)), and the MAE (Eq. (3.6)). It should be mentioned, that it is also common to employ the normalized version of the previous metrics dividing them by the range of the adder.

Besides the accuracy of the approximate architectures, the hardware efficiency of them need to evaluated. In the mathematical analysis, we use the unit-gate model [58] where simple monotonic 2-input gates (AND,OR,NAND,etc.) have a cost of one in area and delay, and simple non-monotonic 2-input gate (XOR,XNOR) have a cost of two in area and delay. In the rest of this section, A and D denote the hardware area and delay, respectively, based on the unit gate delay. Obviously, in the experimental results, the actual silicon area and latency of the circuits are reported.

[1] Throughout this section, "optimal" refers to "optimal for the given template".

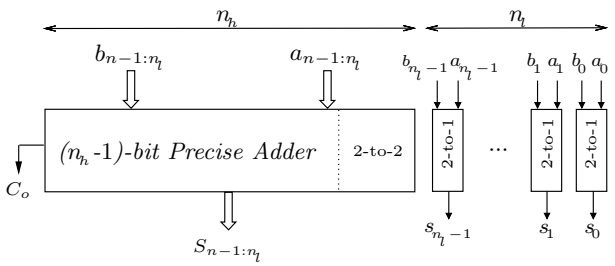


Fig. 4.1: The hardware structure of the general template

General Template Architecture Based on LOA

As discussed before, considering the error versus hardware-cost trade-off, experimental results show that LOA is the best architecture among all the existing approximate adders [48, 103] considering a uniform distribution. Studying LOA's architecture carefully, it can be generalized as Fig. 4.1: the lower significant sub-adder can be divided into n_l 2-to-1 logic blocks (bits 0 to n_l-1), and a single 2-to-2 logic block. This latter block receives the inputs of the adder in bit position n_l to generate the input carry for the exact part using an AND gate, and its sum signal can be generated inexactly. Finally, the higher significant sub-adder is an exact adder. Clearly, the architecture of LOA can be described by taking the proposed general template, putting OR gates in each bit of the lower significant sub-adder, and replacing the first bit of the higher significant sub-adder with approximate circuitry of OR_AND.

Table 4.1: All the 1-bit addition possibilities: (a) exact results, (b) approximate outputs.

(a)

		b_i	
		0	1
a_i	0	0	1
	1	1	2

(b)

		b_i	
		0	1
a_i	0	$\tilde{s}0_i$	$\tilde{s}1_i$
	1	$\tilde{s}2_i$	$\tilde{s}3_i$

In principle, any Boolean function with the right size provides a choice for the blocks. Note that even a constant function equal to one (Cte-1) or zero (Cte-0) is a valid selection. All the possible choices for 2-to-1 blocks and 2-to-2 blocks are tabulated in Table 4.2 and Table 4.3, respectively. The error values presented in these tables are calculated with the help of Table 4.1, where Table 4.1(a) represents the accurate 1-bit additions, and Table 4.1(b) shows

Table 4.2: all possibilities for 2-to-1 blocks

$\tilde{s}0, \tilde{s}1, \tilde{s}2, \tilde{s}3$	HW	$\hat{\mu}$	$\hat{\sigma}^2$	$\hat{MSE}$	HW Cost	
0,0,0,0	Cte-0	-1	$1/2$	$3/2$	0	←Pareto point (Cte-0)
0,0,0,1	$a.b$	$-3/4$	$3/16$	$3/4$	1	←Pareto point (AND)
0,0,1,0	$a.\overline{b}$	$-3/4$	$11/16$	$5/4$	1	
0,0,1,1	a	$-1/2$	$1/4$	$1/2$	0	←Pareto point (Buffer)
0,1,0,0	$\overline{a}.b$	$-3/4$	$11/16$	$5/4$	1	
0,1,0,1	b	$-1/2$	$1/4$	$1/2$	0	
0,1,1,0	$a \oplus b$	$-1/2$	$3/4$	1	2	
0,1,1,1	$a + b$	$-1/4$	$3/16$	$1/4$	1	←Pareto point (OR)
1,0,0,0	$\overline{a + b}$	$-3/4$	$19/16$	$7/4$	1	
1,0,0,1	$a\overline{\oplus}b$	$-1/2$	$3/4$	1	2	
1,0,1,0	$\overline{b}$	$-1/2$	$5/4$	$3/2$	0	
1,0,1,1	$a + \overline{b}$	$-1/4$	$11/16$	$3/4$	1	
1,1,0,0	$\overline{a}$	$-1/2$	$5/4$	$3/2$	0	
1,1,0,1	$\overline{a} + b$	$-1/4$	$11/16$	$3/4$	1	
1,1,1,0	$\overline{a.b}$	$-1/4$	$19/16$	$5/4$	1	
1,1,1,1	Cte-1	0	$1/2$	$1/2$	0	←Pareto point (Cte-1)

the approximate 1-bit additions. As can be seen in Table 4.1(b), $\tilde{s}0, \tilde{s}1, \tilde{s}2$, and $\tilde{s}3$ are approximate sums of "a" and "b", where "ab" are 00,01,10, and 11, respectively. Note that, in the case of 2-to-2 blocks in Table 4.3, the error values are calculated by considering that the carry bit is constructed with an AND gate. For different values of average error and hardware cost, the Pareto points have been marked. In the case of 2-to-2 blocks of Table 4.3, Cte-0 and AND gate because of their negative average errors are not selected despite of being Pareto points. Although we have studied all the possibilities, the blocks with higher error values for the same cost are not considered for the proof. Correspondingly, for concreteness, the relevant choices for 2-to-1 and 2-to-2 blocks are tabulated in Table 4.4 and Table 4.5, respectively. As mentioned above, an AND gate is used in all the cases to produce carry input for the upper part adder. That is why in the name of 2-to-2 blocks are formed as $*$_AND. In order to have an *optimal* architecture for the template, the best combination of blocks from each table should be chosen. For uniform distributed data, each bit is uncorrelated and the error metrics of the template (T) can be calculated as a function of the error characteristics of each block. Since the total error, ε_T, is the summation of the errors of each block , $\hat{\varepsilon}_i$, with the corresponding

Table 4.3: all possibilities for 2-to-2 blocks

$\tilde{s}0, \tilde{s}1, \tilde{s}2, \tilde{s}3$	HW	$\hat{\mu}$	$\hat{\sigma}^2$	$\hat{MSE}$	HW Cost	
0,0,0,0	Cte-0	$-1/2$	$1/4$	$1/2$	0	
0,0,0,1	$a.b$	$-1/4$	$11/16$	$3/4$	1	
0,0,1,0	$a.\overline{b}$	$-1/4$	$3/16$	$1/4$	1	
0,0,1,1	a	0	$1/2$	$1/2$	0	
0,1,0,0	$\overline{a}.b$	$-1/4$	$3/16$	$1/4$	1	
0,1,0,1	b	0	$1/2$	$1/2$	0	
0,1,1,0	$a \oplus b$	0	0	0	2	←Pareto point (XOR)
0,1,1,1	$a + b$	$1/4$	$3/16$	$1/4$	1	←Pareto point (OR)
1,0,0,0	$\overline{a + b}$	$-1/4$	$11/16$	$3/4$	1	
1,0,0,1	$a\overline{\oplus}b$	0	1	1	2	
1,0,1,0	$\overline{b}$	0	$1/2$	$1/2$	0	←Pareto point (Buffer)
1,0,1,1	$a + \overline{b}$	$1/4$	$11/16$	$3/4$	1	
1,1,0,0	$\overline{a}$	0	$1/2$	$1/2$	0	
1,1,0,1	$\overline{a} + b$	$1/4$	$11/16$	$3/4$	1	
1,1,1,0	$\overline{a.b}$	$1/4$	$3/16$	$1/4$	1	
1,1,1,1	Cte-1	$1/2$	$1/4$	$1/2$	0	←Pareto point (Cte-1)

weight, i.e., $\varepsilon_{_T} = \sum_{i=0}^{n_l} \hat{\varepsilon}_i 2^i$, we obtain:

$$\mu_{_T} = \sum_{i=0}^{n_l} \hat{\mu}_i 2^i \tag{4.2}$$

$$\sigma^2_{_T} = \sum_{i=0}^{n_l} \hat{\sigma}^2_i 2^{2i} \ , \tag{4.3}$$

$$MSE_{_T} = \sum_{i=0}^{n_l} \hat{\sigma}^2_i 2^{2i} + \left(\sum_{i=0}^{n_l} \hat{\mu}_i 2^i \right)^2 , \tag{4.4}$$

where $\hat{\mu}_i$ and $\hat{\sigma}^2_i$ are the average error and the variance of error associated with the instantiated block in bit position i. The corresponding values are given in Table 4.4 for bits 0 to $n_l - 1$ and in Table 4.5 for the bit n_l, under the column names $\hat{\mu}$ and $\hat{\sigma}^2$, respectively. The error of each block $\hat{\varepsilon}_i$ is calculated based on Eq.(4.1), where the exact result is tabulated in Table 4.1(a), and the approximate addition results are obtained using Table 4.1(b). For example, using this method, we obtained the error metrics for LOA shown in Table 4.6

which agree with the simulation results of [48]. The key question, now, is whether the particular choices made by LOA are optimal, and if not, which is the optimal alternative for the selected template. Next subsection addresses this topic.

Table 4.4: Error metrics and unit gate characteristics of the possibilities for 2-to-1 blocks

	$\hat{\mu}$	$\hat{\sigma}^2$	$\hat{MSE}$	$\hat{A}$	$\hat{D}$
AND	$-3/4$	$3/16$	$3/4$	1	1
OR	$-1/4$	$3/16$	$1/4$	1	1
Buffer	$-1/2$	$1/4$	$1/2$	0	0
Cte-0	-1	$1/2$	$3/2$	0	0
Cte-1	0	$1/2$	$1/2$	0	0

Table 4.5: Error metrics and unit gate characteristics of the possibilities for the 2-to-2 blocks

	$\hat{\mu}$	$\hat{\sigma}^2$	$\hat{MSE}$	$\hat{A}$	$\hat{D}$
Half-Adder	0	0	0	3	2(1)
OR_AND	$1/4$	$3/16$	$1/4$	2	1(1)
Cte-1_AND	$1/2$	$1/4$	$1/2$	1	0(1)
Buffer_AND	0	$1/2$	$1/2$	1	0(1)

The Optimized Architecture

Depending on the error metrics which are chosen, different optimization results might be obtained. Illustratively, here, the MSE is chosen as the error metric because of its relevance in data processing applications. In order to obtain the optimal architecture out of the general template, all the possible combinations of 2-to-1 and 2-to-2 logic blocks of Table 4.4 and Table 4.5 need to be evaluated. Let us proceed firstly intuitively and then more formally.

The errors in the upper bits have a higher weight than in the lower ones (see Eq. (4.4)). Thus, it is more profitable to expend resources in the 2-to-2 block than in the lower 2-to-1 blocks. The best 2-to-2 blocks are the OR_AND and Half-adder. Replacing the Half-adder with an OR_AND does not improve the delay and improves the area only marginally; the penalty is a large increase in the MSE. For this reason, the idea of LOA (to use the OR_AND for the 2-to-2 block) is not efficient. Form Table 4.5, it can be observed that the average error introduced by the 2-to-2 block is zero or positive, while the 2-to-1 blocks introduce a zero or negative average error. Thus, once the 2-to-2 block is set to a Half-adder, it is only useful to use blocks with small $\hat{\mu}$ (the Cte-1) or

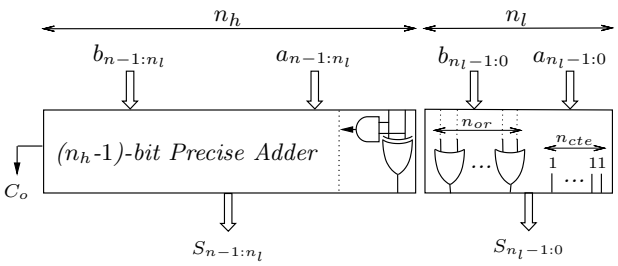


Fig. 4.2: The structure of LOCA; $n_l = n_{cte} + n_{or}$

small $\hat{\sigma}$ (the OR). Therefore, the optimal disposition of 2-to-1 blocks should be OR blocks followed by Cte-1 blocks in the lower bits where the errors are less relevant. Since the adder is constructed using Cte-1s and OR gates, we call it *L*ower-part Constant-OR Adder (LOCA). The structure of LOCA is depicted in Fig. 4.2 and its error metrics can be expressed as follows:

$$\mu_{LOCA} = 2^{n_{cte}-2} - 2^{n_l-2} \ , \tag{4.5}$$

$$\sigma^2_{LOCA} = 2^{2n_l-4} + \frac{5}{3}2^{2n_{cte}-4} - \frac{1}{6} \ , \tag{4.6}$$

$$MAE_{LOCA} = 2^{n_l-2} - 2^{n_{cte}-2} + \tag{4.7}$$

$$+ \frac{1}{3}\left(\frac{3}{4}\right)^{n_l-n_{cte}}\left(2^{n_{cte}} - \frac{1}{2^{n_{cte}}}\right) \ , \tag{4.8}$$

$$MSE_{LOCA} = \frac{1}{6}2^{2n_l-2n_{or}} + 2^{2n_l-3} - 2^{2n_l-n_{or}-3} - \frac{1}{6} \tag{4.9}$$

To determine the optimal number of OR gates, we can minimize the Eq. (4.9) versus n_{or}, resulting the optimal value in $n_{or} = \log_2\left(\frac{8}{3}\right)$. The closest integer numbers, $n_{or} = 1$ and $n_{or} = 2$, produce the same MSE and are optimal. We prefer $n_{or} = 2$ that provides a better STD. We call this architecture *Optimized*

Table 4.6: Formulas of error metrics, area and delay

	LOA	OLOCA
μ	$\frac{1}{4}$	$\frac{-3}{16}2^{n_l}$
σ^2	$\frac{1}{4}4^{n_l} - \frac{1}{16}$	$\frac{53}{768}4^{n_l} - \frac{1}{6}$
MSE	$\frac{1}{4}4^{n_l}$	$\frac{5}{48}4^{n_l} - \frac{1}{6}$
MAE	$\frac{3}{8}2^{n_l} - \frac{3}{8}$	$\frac{15}{64}2^{n_l} - \frac{3}{4}2^{-n_l}$
A	$(n_h-1).A_{FA} + A_{AND} + (n_l+1).A_{OR}$	$(n_h-1).A_{FA} + A_{HA} + (n_l-n_{cte}).A_{OR}$
D	$(n_h-1).t_c + T_{AND}$	$(n_h-1).t_c + T_{AND}$

Lower-part Constant-OR Adder (OLOCA). Although remarkable simple, it outperforms LOA regarding STD, MSE and MAE.

The error metrics, area and delay of LOA and OLOCA are tabulated in Table 4.6. Those formulas provide a better understanding of the architectures and make the comparison easier. As can be seen in the table, the average error of OLOCA is slightly larger than that of LOA's, while its STD is much smaller. Hence, the MSE of OLOCA is almost 2.4 times smaller than the MSE of LOA, which represents a considerable improvement for practical circuits. Regarding the MAE, LOA has a 1.6 times larger error with respect to OLOCA. Although OLOCA does not improve the delay over LOA, its silicon area is clearly smaller (for $n_l > 2$).

It is also possible to obtain the optimal architecture out of the general template more rigorously. Firstly, let us observe that Eq. (4.4) and Table 4.5 imply that an architecture where the 2-to-2 block is not a Half-Adder has necessarily a MSE of at least $MSE_T \geq \hat{\sigma}^2_{n_l} 4^{n_l} \geq \frac{3}{16} 4^{n_l}$ (due to the fact that the second term of Eq. (4.4) cannot be negative), which is worse than the MSE of OLOCA (see Table 4.6). Thus, the 2-to-2 block has to be a Half-Adder in the optimal architecture.

In order to demonstrate that the selection of 2-to-1 blocks of OLOCA is optimal in terms of MSE for the given template (Fig. 4.1), we can proceed by induction, using n_l as the induction variable. A simple computation of all the possibilities, using Eq. (4.4), shows that OLOCA is indeed optimal for $n_l = 1$, $n_l = 2$ and $n_l = 3$. For the case that $n_l = 1$, looking at Table 4.4, it is clear that the best 2-to-1 block regarding MSE is the OR gate. For the case that $n_l = 2$, there are 25 possibilities to be considered. Using Eq. (4.4), calculating these 25 possibilities, both the 2-to-1 blocks should be OR gates. Table 4.7 shows the MSE of the template for all the combinations of relevant 2-to-1 blocks when $n_l = 2$. In a similar manner, all the possibilities for $n_l = 3$ can be calculated

Table 4.7: Mean Squred Error for all the combinations of relevant 2-to-1 blocks when n_l=2.

		block 0				
		AND	OR	Buffer	Cte-0	Cte-1
block 1	AND	6.00	4.00	5.00	7.50	3.50
	OR	2.50	1.50	2.00	3.50	1.50
	Buffer	4.25	2.75	3.50	5.50	2.50
	Cte-0	9.75	7.25	8.50	11.50	6.50
	Cte-1	2.75	2.25	2.50	3.50	2.50

using Eq. (4.4). There are 125 relevant possibilities when $n_l = 3$. Calculating

all the possibilities, OLOCA is optimal regarding MSE. In the Table 4.8, 25 of those 125 possibilities are tabulated considering that the 2-to-1 block in bit position 2 (third block) is an OR gate.

Table 4.8: Mean Squred Error for all the combinations of relevant 2-to-1 blocks when n_l=3, considering block in bit position 2 is an OR gate.

		block 0				
		AND	OR	Buffer	Cte-0	Cte-1
block 1	AND	14.50	11.50	13.00	16.50	10.50
	OR	9.00	7.00	8.00	10.50	6.50
	Buffer	11.75	9.25	10.50	13.50	8.50
	Cte-0	19.25	15.75	17.50	21.50	14.50
	Cte-1	8.25	6.75	7.50	9.50	6.50

Let us analyze an architecture with $n_l = K$, assuming the optimality of OLOCA for an architecture with $n_l = K - 1$. Observe that the total error, ε_T, can be decomposed into the independent contributions of the block in bit position 0, $\hat{\varepsilon}_0$, and the remaining blocks, ε_{MSBs}. Since $\varepsilon_T = \varepsilon_{MSBs} + \hat{\varepsilon}_0$, the MSE_T can be expressed as a function of the statistical characteristics of $\hat{\varepsilon}_0$ and ε_{MSBs}; more precisely:

$$MSE_T = MSE_{MSBs} + \hat{MSE}_0 + 2\hat{\mu}_0\mu_{MSBs} \ , \tag{4.10}$$

where μ_{MSBs} and MSE_{MSBs} can be calculated using Eq. (4.2) and Eq. (4.4), respectively, iterating i from 1 to K.

Note that the optimization of the block 0 and the remaining $K-1$ blocks are not independent due to the term $2\hat{\mu}_0\mu_{MSBs}$. However, if we prove that the block 0 is a Cte-1 in the optimal architecture, then it follows that $\hat{\mu}_0 = 0$ and the term $2\hat{\mu}_0\mu_{MSBs}$ disappears. In this case, the optimization of MSE_{MSBs}, consisting of $K-1$ blocks, yields an OLOCA architecture by the induction hypothesis. Consequently, considering the fact that OLOCA is optimal for $n_l = 3$, it will be optimal for $n_l = 4$; being optimal for $n_l = 4$, it will be optimal for $n_l = 5$; and so on.

Let us show that the block in bit position 0 has to be Cte-1 (for $K \geq 3$) in order to have the optimal architecture regarding MSE. Considering all the possibilities for 2-to-1 blocks of Table 4.4, using Eq. (4.10), the MSE of the

template for the different 2-to-1 blocks can be represented as follows:

$$MSE_T = \begin{cases} MSE_{MSBs} + \frac{3}{4} + 2\mu_{MSBs}(\frac{-3}{4}) & \text{when} \quad block\ 0\ is\ \text{AND} \\ MSE_{MSBs} + \frac{1}{4} + 2\mu_{MSBs}(\frac{-1}{4}) & \text{when} \quad block\ 0\ is\ \text{OR} \\ MSE_{MSBs} + \frac{1}{2} + 2\mu_{MSBs}(\frac{-1}{4}) & \text{when} \quad block\ 0\ is\ \text{Buffer} \\ MSE_{MSBs} + \frac{3}{2} + 2\mu_{MSBs}(-1) & \text{when} \quad block\ 0\ is\ \text{Cte-0} \\ MSE_{MSBs} + \frac{1}{2} + 0 & \text{when} \quad block\ 0\ is\ \text{Cte-1}. \end{cases} \tag{4.11}$$

The plot of the equations discarding the common term MSE_{MSBs}, as can be seen in Fig. 4.3, shows that for $\mu_{MSBs} \leq \frac{-1}{2}$, Cte-1 must be instantiated for the block in bit position 0 to develop the optimal architecture regarding MSE. As a side note, observe that for $\frac{-1}{2} \leq \mu_{MSBs} \leq \frac{1}{2}$, for example, OR gate must be used to have the optimal architecture regarding MSE. Note that the alternative of

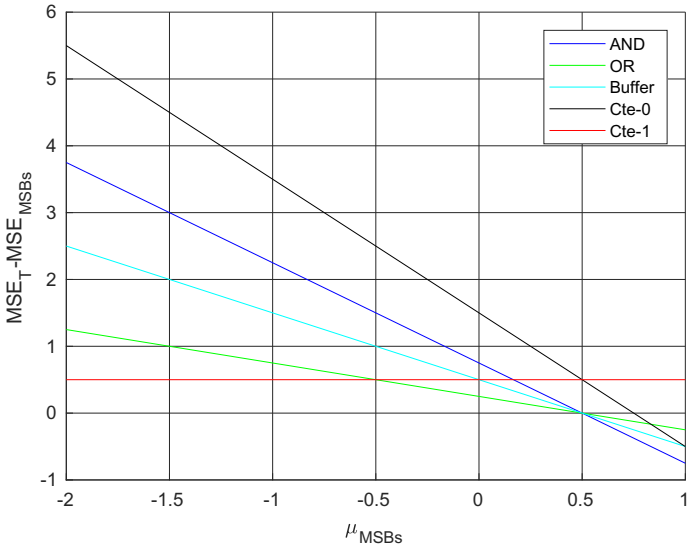


Fig. 4.3: The relation between MSE_T and the μ_{MSBs}, depending on the block in bit position 0.

choosing Cte-1 for the blocks 1 to $K-1$, which produces $MSE_{MSBs}=\frac{1}{6}4^K - \frac{2}{3}$ and $\mu_{MSBs} = 0$, is sub-optimal. This is due to the fact that the resulting MSE_T, which is greater than or equal to MSE_{MSBs}, is worst than that of OLOCA. As a result, at least one of the blocks should not be a Cte-1 in the optimal architecture. For any of those cases, $\mu_{MSBs} = \sum_{i=1}^{K-1} \hat{\mu}_i 2^i \leq \frac{-1}{2}$, because each term in the addition is strictly negative. Now that the optimal architecture necessarily satisfies $\mu_{MSBs} \leq \frac{-1}{2}$, a Cte-1 is the optimal selection according to Fig. 4.3 as discussed above. Considering the fact that we know the block 0

should be Cte-1 for the optimal architecture, we know that:

$$MSE_{_T} = MSE_{_{MSBs}} + MSE_{_0} + 2\mu_{_{MSBs}}\mu_{_0} \ = MSE_{_{MSBs}} + \frac{1}{2}. \tag{4.12}$$

Therefore, the optimization of $\text{MSE}_{_{MSBs}}$ results in the optimization of $\text{MSE}_{_T}$. Since we already know that OLOCA is optimal regarding MSE for the $n_l = 3$, we can conclude that it is optimal for the given template regarding MSE for $n_l = 4$, $n_l = 5$ and so on. This observation concludes the proof.

4.2.2 Experimental Results

To assess the circuit characteristics and evaluate the presented architectures in the previous subsection, we have generated VHDL description of the adders. Different configurations of these adders are synthesized in a commercial low-power 65 nm library, for 16-bit and 8-bit operands. Using back-annotated simulations, dynamic power dissipation of the adders are evaluated after synthesis for the freq.=1GHz. Ripple Carry Adders (RCA) are used as the sub-adders of all the approximate adders. All the adders have been simulated for 10^7 uniformly distributed random input patterns. In this section, each adder's name is followed by one number. For ESA and ETAII, this number is the size of the equal segments. Regarding LOA, and OLOCA the number is the size of the lower significant sub-adder; i.e. n_l.

In order to check the accuracy of the formulas, as well as comparing the adder architectures, the error versus cost of the adders for different values of n_ls are depicted in Fig. 4.4. MAE and MSE versus Area-Delay Product (ADP) of the 16-bit adders are shown in two graphs. LOCA has been simulated for different number of constants in each n_l case. As can be seen in the graphs, replacing OR gates with Cte-1s decreases the MAE and MSE values and at the same time the ADP; the trend continues until the point where 2 OR gates remain. After that point, the error values start increasing while the cost of the adder decreases. As a result, the optimal architecture, considering the error-cost trade-off, is obtained keeping 2 OR gates and place Cte-1s for the rest of 2-to-1 blocks. This verifies the discussion in the previous section that the optimal architecture has 2 OR gates. Replacing all the 2-to-1 blocks with Cte-1s considerably increase the error values. Although, replacing all the 2-to-1 blocks with Cte-1s results in an architecture which is still better that LOA, it is not the optimal architecture, as shown in the figure. It can also be seen that the OLOCA and LOA's formulas (see Table 4.6) perfectly predict the behavior of the adders for all the n_ls. Moreover, for all n_ls, OLOCA outperforms LOA, both from cost and error points of view; for the same values of errors, OLOCA improves the cost almost 25% and for the same values of cost, the error values of OLOCA are almost half of the LOA's. As an example, a 16-bit OLOCA-8 improves the cost by 13.6%, MAE by 37.4% and MSE by 58% in comparison

with LOA-8.

In order to evaluate OLOCA with another bit-width, we have studied 8-bit adders as well; the results are tabulated in Table 4.9. The table shows the accuracy of the presented formulas versus the simulation results, as well as the superiority of OLOCA over LOA for all the n_ls. As an example, OLOCA-4 improves MAE by 36.9%, MSE by 58.5% and cost by 7.2% in comparison with LOA-4.

To show the superiority of OLOCA over all the existing approximate adders, besides LOA, we consider ESA , ETAII and GeAr. Among the existing combinational approximate adders, the above-mentioned architectures have proved to have the best performance [48, 103] after LOA. Different configurations of the

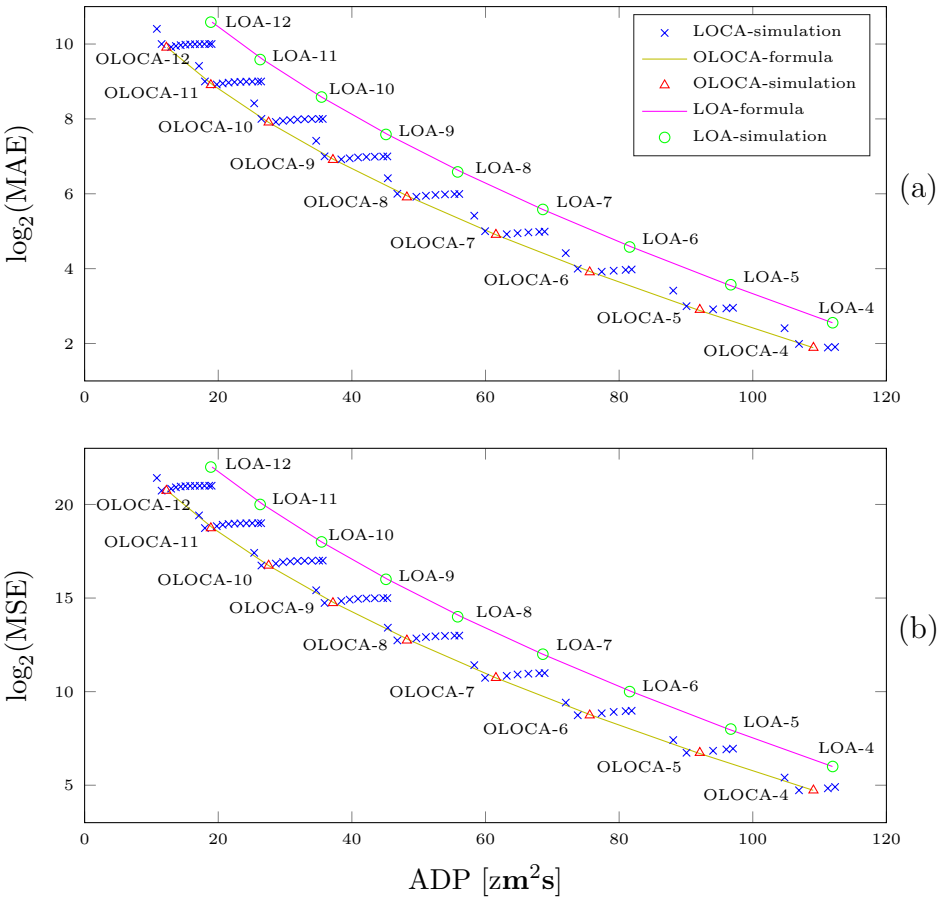


Fig. 4.4: Comparison of 16-bit LOA and OLOCA synthesized in a 65nm tech.: Simulation and formulas results. (a)Mean Absolute Error (MAE) vs. Area-Delay Product (ADP), (b)Mean Squared Error (MSE) vs. Area-Delay Product (ADP).

Table 4.9: Simulation and formulas results for 8-bit adders synthesized in a commercial 65nm technology

		n_l=2		n_l=3		n_l=4		n_l=5		n_l=6	
		Sim.	Formula	Sim.	Formula	Sim.	Formula	Sim.	Formula	Sim.	Formula
MAE	LOA	1.38	1.38	2.88	2.88	5.87	5.88	11.87	11.88	23.86	23.88
	OLOCA	0.75	0.75	1.78	1.78	3.70	3.70	7.48	7.48	14.96	14.99
MSE	LOA	4.00	4.00	16.00	16.00	63.93	64.00	255.90	256.00	1023.39	1024.00
	OLOCA	1.50	1.50	6.53	6.50	26.50	26.50	106.57	106.50	424.95	426.50
STD	LOA	1.99	1.98	3.99	3.99	7.99	8.00	16.00	16.00	31.99	32.00
	OLOCA	0.97	0.97	2.06	2.06	4.18	4.18	8.40	8.40	16.79	16.81
ADP	LOA	26.82	26.82	19.19	19.50	13.19	13.32	7.95	8.28	3.94	4.38
[$\mathbf{zm^2s}$]	OLOCA	27.00	27.00	18.89	19.20	12.24	12.36	6.74	7.02	3.05	3.18

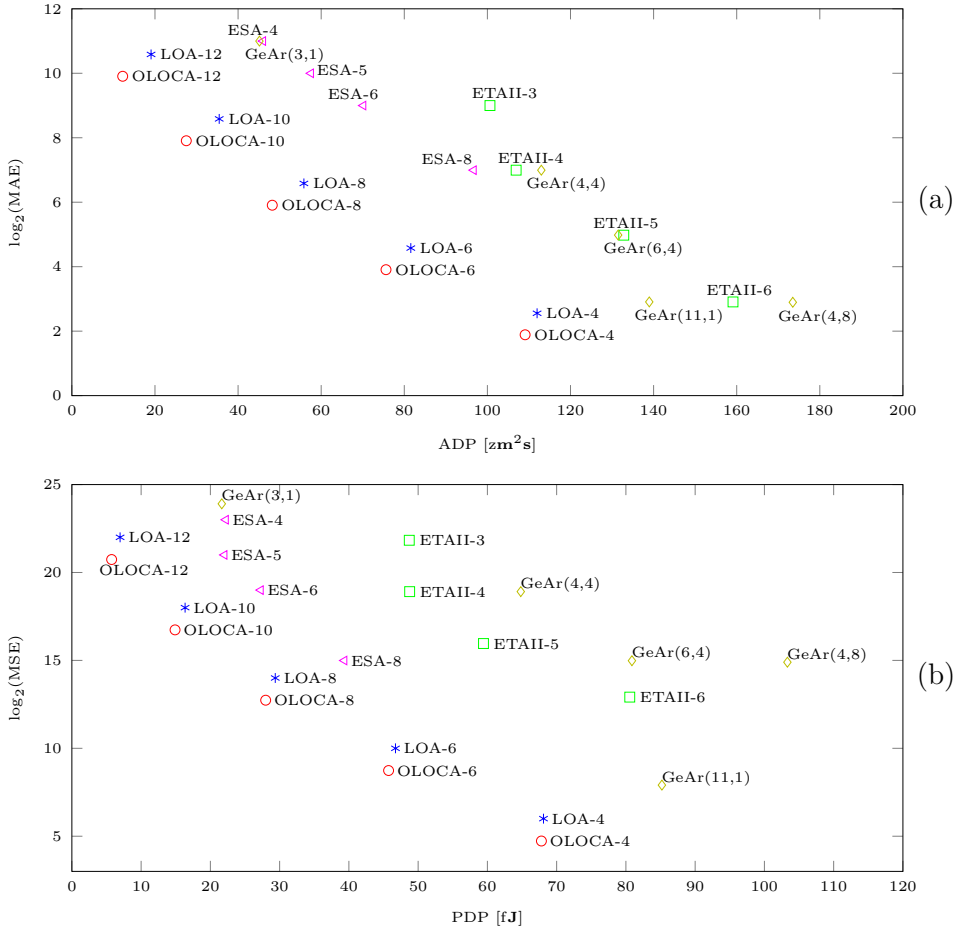


Fig. 4.5: Comparison of 16-bit approximate adders synthesized in a commercial 65nm tech. with various configurations: (a)Mean Absolute Error (MAE) vs. Area-Delay Product (ADP), (b)Mean Squared Error (MSE) vs. Power-Delay Product (PDP).

adders have been simulated and the results are depicted in Fig. 4.8. Fig. 4.8(a) depicts the MAE of the adders versus ADP. Similarly, MSE versus Power-Delay Product (PDP) of the adder architectures are illustrated in Fig. 4.8(b). Although ESA is hardware-efficient, it is the least accurate adder architecture. As an example, for the almost same value of ADP, OLOCA-8 improves the error value by 97% in comparison with ESA-4. OLOCA-8 improves error, ADP and PDP by 53%, 54.9% and 42.6% compared with ETAII-4, respectively. The improvements for the MSE are even larger.

4.3 Unified Design Framework and Metrics of Hybrid Approximate Adders

The selection of an approximate architecture is strongly influenced by the timing constraints of the hardware [48]. The constraints determine the internal structure of the unit (sequential structure with linear cost, parallel-prefix structure with a logarithmic cost, etc.), which determines the reduction in cost achieved by the approximate units. As mentioned in Chapter 3, when the timing constraint is relaxed, an exact adder is implemented with a ripple-carry structure. In such a scenario, an ESA with $k = \frac{n}{2}$, reduces the delay by a factor of two. With more stringent timing constraints, an exact adder is implemented with a parallel-prefix architecture, where the delay increases as $\lceil \log_2(n) \rceil$. In this scenario, however, the ESA just marginally reduces the delay. In this case, non-equally segmented sub-adders may be preferable. The effects of the internal adder architecture have been studied in [48]; it illustrates the potentials for non-equally segmented approximate adders which are currently disregarded. A unified description for approximate adders is still an open problem.

The goal in this section is to provide a conceptual framework for the systematic design of approximate adders, including hybrid and non-equally segmented approaches. Consequently, this section is organized as follows: Firstly, a template for approximate adders is developed, including the *s*mall-errors and the *i*nfrequent-errors philosophies; it includes a detailed mathematical analysis of errors. Afterwards, it is described how our design framework automatically chooses best configurations of the proposed template for given application constraints. Finally for the validation, an experimental evaluation using a commercial CMOS technology in presented.

4.3.1 Hardware modeling

As previously mentioned, the existing approximate adders have been proposed based on two major philosophies: 1) *Infrequent errors*: Segmented adders where an n-bit adder is divided into k-bit sub-adders [25, 28, 68]. This class of adders mostly produce big errors with low probability of happening. 2) *Small errors*: Approximate full adders where the full adder is approximated [16, 23]. This

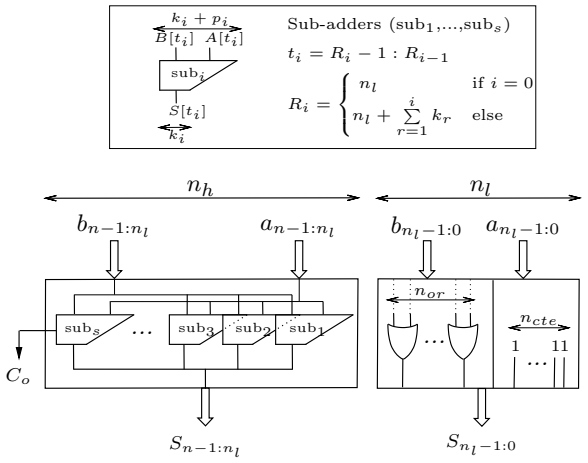


Fig. 4.6: The proposed generic template of approximate adders

group of adders approximate lower bits and hence, produce small errors with high probability of happening. Since the superiority of approximate adders depends on the application they are used in, a systematic and unified template for hybrid solutions, covering a relatively wide range of approximate adders, are desirable and even required.

At the same time, designers have so far considered segmented adders as *equally* segmented. The reason to consider only equal segmentation is to have the minimum delay and thereupon, an optimal approximate architecture. This consideration is appropriate when working with area-efficient, slow adders like RCA. However, working with fast and/or optimal exact adders replaced as sub-adders, asymmetric and non-equally segmented adders might outperform the equally-segmented ones. One example is the system reported in Table 3.1.

In order to address the above-mentioned problems, in this section, systematically, we propose a unified generic template for approximate adders, which combines the two error philosophies.

Nomenclature

In this section, each adder's name is followed by numbers. For ETAII and ESA, the number is the size of equal segmented sub-adders. For GeAr, the left number is the number of resultant bits contributing to the final sum, and the right one is the number of previous bits used for carry prediction. Regarding LOA, it is the number of OR gates. And finally, for LOCA [47], the numbers indicate the number of OR gates and constant-1s, from left to right, respectively.

In addition to that, we have a new naming system based on the proposed template architecture, which will be discussed later in this section.

The Generic Template of Approximate Adders

The generic template for approximate adders is shown in Fig. 4.6. As can be seen in the figure, the template is constructed using the two aforementioned approximation methodologies. An upper part which is constructed using a segmented adder (the infrequent-errors philosophy) and a lower part composed of OR gates and constant-1s (the small-errors philosophy). Variable n denotes the length of operands to be added, while n_l and n_h represent the bit-width of the lower and higher parts of the template, respectively.

As depicted in Fig. 4.6, while the lower part is constructed using n_{or} OR gates as well as n_{cte} constants, the upper part is instantiated by an adder. The adder can be either exact or approximate. In order to have a range of adders using the template, we replace the higher-significant part of the template with a segmented adder. The segmented adder splits the entire carry propagation path into a number of short paths and completes the carry propagations in these short paths concurrently. Let $\mathbb{K} = \{k_s, k_{s-1}, \ldots, k_1\}$ denote a vector including the size of resultant bits of each sub-adder, where s is the number of its sub-adders, k_1 is size of the resultant bits of the first (the lowest significant) sub-adder, k_s is size of the resultant bits of the last (the most significant) sub-adder, and so on. In addition, $\mathbb{P} = \{p_s, p_{s-1}, \ldots, p_1\}$ represents a vector including number of previous bits used for carry prediction. The vectors $\mathbb{K}$ and $\mathbb{P}$ describe the upper part of the architecture; the i^{th} sub-adder of the segmented adder which has a size of k_i+p_i, uses p_i previous bits to construct the input carry for the sub-adder. In order to name each adder developed using the template, we use the following format:

$$\boxed{k_s(p_s)k_{s-1}(p_{s-1})\ldots k_1(p_1)|n_{or}, n_{cte}}$$

This way, using the template, not only a wide range of the existing approximate adders can be developed, but also hybrid and non-equal segmented approximate adders can be generated.

Table 4.10 tabulates some examples to illustrate how some of the existing approximate adders are constructed using our template. As can be seen in the table, different classes of approximate adders can be implemented, changing variables of the template. Since the GeAr adder can be constructed using our proposed template, ACA is also in the list of the adders which can be developed by the template.

In order to obtain the error formulas for the template, we divide the architecture into lower and higher parts. Then, we formulate the error metrics for each part. Finally, the error metrics of the template can be obtained combining the error formulas for each part of the template. Let us first introduce a term in order to make the formulas, presented in the rest of this section more compact:

Table 4.10: Examples to illustrate how to generate some of the existing approximate adders using the template, 16-bit adders

Adder	$k_s(p_s)k_{s-1}(p_{s-1})\dots k_1(p_1)\|n_{or}, n_{cte}$
Exact adder	$16(0)\|0,0$
ETAII-4	$4(4)4(4)8(0)\|0,0$
ETAII-6	$6(6)10(0)\|0,0$
ESA-4	$4(0)4(0)4(0)4(0)\|0,0$
GeAr-2,6	$2(6)2(6)2(6)2(6)8(0)\|0,0$
LOA-8	$8(1)\|8,0$
LOCA-2,6	$8(0)\|2,6$

$$B_i = \begin{cases} 0 & \text{if } i = 0 \\ \sum_{r=1}^{i} k_r & \text{otherwise} \end{cases} \tag{4.13}$$

In order to have the histogram of the upper part of the template which is a segmented adder, first, we need to review the Generate and Propagate concepts. Generate G_j and propagate P_j signals are computed using bit-wise operations:

$$P_j = a_j \oplus b_j \quad , \quad G_j = a_j b_j \ . \tag{4.14}$$

where a_j and b_j are the primary inputs of the adder corresponding to the bit position j.

From Fig. 4.6, when $p_i + k_i - p_{i+1}$ bits generate a carry signal and the p_{i+1} bits propagate the carry, the sub-adder sub_i produces error with the magnitude -2^{B_i}:

$$\varepsilon_{h_i} = -2^{B_i} \ . \tag{4.15}$$

Given the fact that for uniformly-distributed inputs $Pr[G_j] = \frac{1}{4}$ and $Pr[P_j] = \frac{1}{2}$; the probability of error ε_{h_i} is calculated as follows:

$$\begin{aligned} Pr_{h_i} = Pr\big[\varepsilon_{h_i}\big] &= \frac{1}{2}\big(1 - (\frac{1}{2})^{k_i+p_i-p_{i+1}}\big)(\frac{1}{2})^{p_{i+1}} \\ &= \frac{2^{k_i+p_i-p_{i+1}} - 1}{2^{k_i+p_i+1}}. \end{aligned} \tag{4.16}$$

where Pr_{h_i} denotes the probability of error ε_{h_i}, for the uniformly distributed input vectors, produced by the i^{th} sub-adder for $i \in \{1, 2, \dots, s-1\}$.

Replacing Eq. (4.15) and Eq. (4.16) in Eq. (3.5) as well as Eq. (3.6), MSE

and MAE of the segmented adder of Fig. 4.6 can be expressed as:

$$MAE_h = \sum_{i=1}^{s-1} |\varepsilon_{h_i}| Pr_{h_i} = 2^{n_h-1-k_s-p_s} - \frac{1}{2^{p_1+1}}, \tag{4.17}$$

and

$$\begin{aligned} MSE_h =& \sum_{i=1}^{s-2} \varepsilon_{h_i}^2 Pr_{h_i} \\ =& 2^{2n_h-2k_s-p_s-1} + \sum_{i=1}^{s-1} 2^{2B_{i-1}-p_i-1}\left(1-2^{k_i}\right) . \end{aligned} \tag{4.18}$$

Since all the errors have the same sign, the mean error value of the segmented adder has the same value as its MAE:

$$\mu_h = -2^{n_h-1-k_s-p_s} + \frac{1}{2^{p_1+1}}, \tag{4.19}$$

The metrics presented above, implicitly show the relation between the error metrics and the higher significant sub-adder variables, (i.e. k_s and p_s).

Regarding the lower part of the template, for the uniform distributed data, each bit is uncorrelated and the error metrics can be calculated as a function of the error characteristics of each block. The total error of the lower part, ε_l, is the summation of the errors of each block, ε_i, with the corresponding weight, i.e., $\varepsilon_l = \sum_{i=0}^{n_l-1} \varepsilon_i 2^i$ [47]. Given the fact for the blocks used in the lower part of the template, i.e., OR gate and Cte-1, the error values are $\mu_{or} = -\frac{1}{4}$, $\sigma_{or}^2 = \frac{3}{16}$ and $\mu_{cte1-1} = 0$, $\sigma_{cte-1}^2 = \frac{1}{2}$, the error metrics of the lower part of the template are as follows:

$$\mu_l = 2^{n_{cte}-2} - 2^{n_l-2} , \tag{4.20}$$

$$\sigma_l^2 = 2^{2n_l-4} + \frac{5}{3}2^{2n_{cte}-4} - \frac{1}{6} , \tag{4.21}$$

$$\begin{aligned} MAE_l =& 2^{n_l-2} - 2^{n_{cte}-2} + \\ &+ \frac{1}{3}\left(\frac{3}{4}\right)^{n_l-n_{cte}} \left(2^{n_{cte}} - \frac{1}{2^{n_{cte}}}\right) . \end{aligned} \tag{4.22}$$

The more details of the equations for the lower part of the template can be found in [47].

Combining the error metrics for the higher and lower part of the template, the error metrics for the template adder are calculated as follows:

$$\mu_{temp} = 2^{n_l} . \mu_h + \mu_l , \tag{4.23}$$

$$\sigma^2_{temp} = 2^{2n_l}.\ \sigma^2_h + \sigma^2_l\ , \tag{4.24}$$

$$MAE_{temp} = 2^{n_l}.\ MAE_h + (1 - PE_h).MAE_l\ . \tag{4.25}$$

In order to have the formulas of the saturated metrics, the understanding of the histogram of the template is required. The histogram of the lower part of the template (LOCA) is a bi-modal distribution with mean value of μ_l and standard deviation of σ_l presented in Eq. (4.20) and Eq. (4.21), respectively. The number of bins (intervals) in the histogram of the upper part of the template (Segmented adder) is equal to the number of segments. These intervals are -2^{B_i}, as mentioned earlier in this section, in addition to the one at zero with $Pr_0 = 1 - PE_h$. The histogram of the template, as a result, is the repetition of the lower part histogram at the intervals of the higher part histogram. Let us consider that there is no overlap in the histogram of the template (which is true in the practical cases). Hence, the histogram consists of multiple bi-modal distribution. Applying a threshold between these distributions, the following formula calculates the precise saturated MSE:

$$\begin{aligned} SMSE_\tau &\approx \sum_{i=1}^{s-1} Pr_{h_i} \min(\tau^2\ ,\ (\varepsilon_{h_i} + \mu_l)^2 + \sigma_l^2) \\ &= \sum_{i=1}^{s-1} Pr_{h_i} \min(\tau^2\ ,\ \varepsilon_{h_i}^2 + 2\varepsilon_{h_i}.\mu_l + MSE_l)\ . \end{aligned} \tag{4.26}$$

where s denotes the number of segments in the upper part segmented adder.

Note that this formula is a piecewise linear approximation of the exact error. It provides a remarkable accuracy for any τ that does not collide with one of the bi-modal distributions that compose the Probability Density Function (PDF) of the error. In the unlikely situation that τ hits one of those bi-modal distributions, the actual error decreases slightly and Eq. (4.26) provides an upper bound of the error.

The other metrics can be calculated in a similar way. We have precisely modeled all the metrics, from conventional to saturated ones (SMSE, SMAE, SSTD), using the histogram of the template. It is part of the developed framework which will be discussed in the next section with more details.

4.3.2 Design Methodology

In this part, the methodology of our framework is discussed. The proposed framework discriminates between the scenarios where approximate processing does not provide significant benefits from those where it does. In the latter case, it aids in obtaining optimal configurations of the template.

In the first step, using precise error models, the framework finds all the possible configurations of the template for a given error constraint. As men-

tioned in the previous subsection, the possible configurations are conventional approximate adders, new hybrid and/or non-equal segmented adders, truncated exact adders, etc. The designer can choose between a traditional metric (MSE, MAE, STD, etc) or the saturated counterpart. In fact, in this step, a given threshold, based on the target application, needs to be specified. Based on the definition presented earlier in Chapter 3, large enough thresholds result in conventional metrics.

In the next step, for a target frequency, the framework sorts the adders based on their expected silicon area. The sorting process is done based on the bit-width of the upper part adder, i.e. $\sum_{i=1}^{s}(k_i + p_i)$, as well as the length of the sub-adders, i.e. $\max(k_i + p_i) for\ i \in \{1, 2, \ldots, s\}$.

Finally, out of several possible architectures which have been sorted, the top ten architectures are synthesized to determine precisely the figures of merit. The complete procedure is automatic.

Observe that the current framework does require a precise characterization of the error (see the equations presented in the previous subsection), but just a relative rank for the area.

4.3.3 Error philosophies of approximate adders

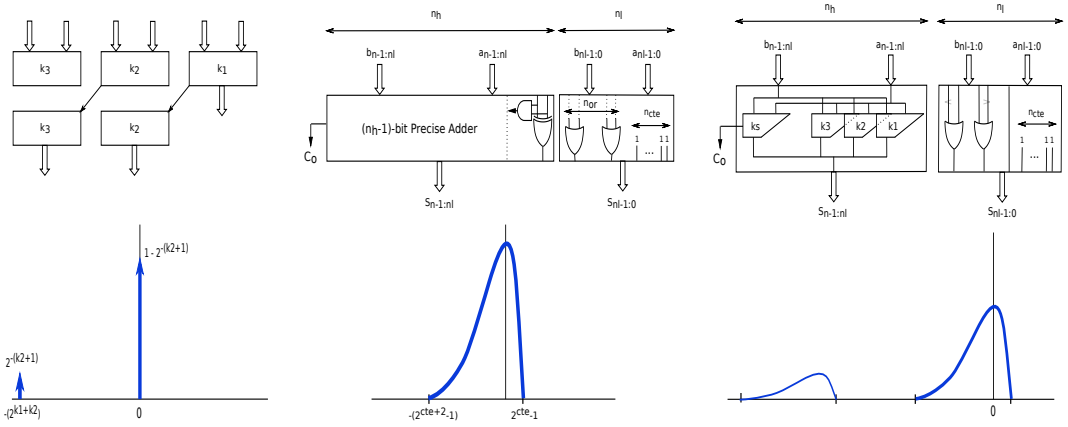


Fig. 4.7: Architecture and error probability distribution of three illustrative approximate adders with different approximation philosophies. From left to right: Error Tolerant Adder (ETA-II), Optimal Lower-part Constant-OR adder (OLOCA), and Hybrid adder.

The conceptual diagram of Fig. 4.7 is the summary of the presented architectures and ideas in this dissertation, in regards to approximate adders. It illustrates the concept behind the error philosophies of the approximate adders. The ETAII is shown in this figure as a representative of *infrequent error* philosophy. As can be observed from its PDF, ETAII rarely fails to produce the exact result, but when it does, the magnitude of the error is extremely

high. The *small error* philosophy is represented by OLOCA adder. From the PDF of OLOCA, it can be seen that the adder generates the errors frequently (with high probability); however, the magnitude of errors are small when they appear. The hybrid adder, combines the two philosophies and alleviates the extremes of the philosophies.

4.3.4 Experimental Results

We have structured this subsection in two steps: Firstly, we evaluate the exactness of the presented error formulas; and secondly, we assess the quality of the adders designed with our framework.

In order to evaluate the accuracy of the presented error formulas, the simulation and formulas results of 16-bit approximate adders, developed using our proposed template, are tabulated in Table 4.11. The numbers presented in the table verify the perfect accuracy of the presented formulas for the generic template of approximate adders. As a result, using one compact accurate formula, the error behavior of a wide range of approximate adders can be modeled.

To assess the circuit characteristics and evaluate the approximate architectures, we have generated VHDL description of the proposed template. Different configurations of approximate adders ranging from existing to new hybrid structures are synthesized in a commercial low-power 65 nm library, for 16-bit operands. Using back-annotated simulations, dynamic power dissipation of the adders are evaluated after synthesis. All the adders have been simulated for 10^7 uniformly distributed random input patterns. In addition, we have developed a framework to distinguish the Pareto optimal architectures for a given accuracy and hardware quality, based on the presented formulas.

Fig. 4.8 illustrates a comparison of approximate adders generated by the proposed template. The compared adders are all optimal (area-delay efficient) architectures. In Fig. 4.8(a), STD of approximate adders versus their Power-Delay Product (PDP) are depicted. As can be seen in this figure, considering the trade-off between error and PDP of the adders, OLOCA architectures outperform all the other adders. Furthermore, hybrid architectures, developed by the proposed template, perform better than other existing approximate adders including ETAII, ESA and GeAr. Another notable fact in this figure is the superiority of non-equal segmented ETAIIs over the equal segmented ones. As discussed in the previous subsection, since the optimal adders are instantiated as sub-adders, non-equal segmentation boosts the efficiency of segmented adders.

On the other hand, Fig. 4.8(b) depicts the Saturated STD (SSTD) of each adder versus its PDP, applying a relatively small threshold. As can be seen in the figure, considering the trade-off between error and energy consumption of the adder architectures, the hybrid architectures developed using the proposed

Table 4.11: Simulation and Formulas results for 16-bit adders

Adder	μ		MAE		σ		$SMSE_{\tau=16}$		$SMAE_{\tau=64}$		$SSTD_{\tau=256}$	
	sim	*math*	*sim*	*math*	*sim*	*math*	*sim*	*model*	*sim*	*model*	*sim*	*model*
4(8)12(0)—0,0	-7.54	-7.50	7.54	7.50	175.57	175.11	0.47	0.47	0.12	0.12	10.97	10.94
4(4)4(4)8(0)—0,0	-127.56	-127.50	127.56	127.50	690.95	690.78	15.01	15.00	3.75	3.75	60.14	60.12
7(1)1(1)8(0)—0,0	-127.51	-127.50	127.51	127.50	181.03	181.02	95.50	95.50	23.88	23.88	123.80	123.81
6(6)10(0)—0,0	-7.54	-7.50	7.54	7.50	87.53	87.32	1.88	1.87	0.47	0.47	21.88	21.83
8(4)4(1)4(0)—0,0	-7.50	-7.50	7.50	7.50	32.00	32.00	60.07	60.00	4.51	4.50	32.00	32.00
8(0)8(0)—0,0	-127.54	-127.50	127.54	127.50	127.99	128.00	127.50	127.50	31.89	31.88	128.00	128.00
12(0)—2,2	-3.00	-3.00	3.70	3.70	4.18	4.18	26.50	26.50	3.70	3.70	4.18	4.18
7(1)5(0)—2,2	-123.02	-123.00	123.55	122.83	216.93	217.22	80.31	80.30	17.84	17.84	107.20	107.24
8(4)5(0)—2,1	-5.50	-5.50	5.78	5.75	31.82	32.31	10.40	10.40	2.75	2.75	31.62	31.61

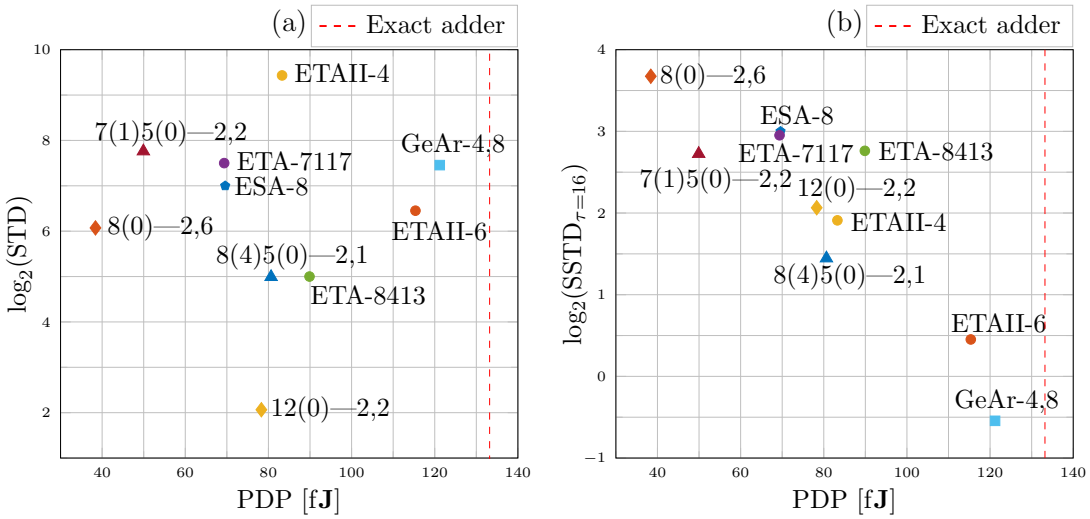


Fig. 4.8: Simulation results Comparison of 16-bit approximate adders: (a) Standar Deviation (STD) vs. Power-Delay Product (PDP), (b) Saturated Standard Deviation (SSTD) applying threshold=2^4 vs. Power-Delay Product (PDP). In order to keep the graphs readable, ETA-8413 denotes the configuration: 8(4)4(1)4(0)—0,0 ; and ETA-7117 is: 7(1)1(1)8(0)—0,0.

template outperform the rest of the adders. For the applications whose error can be modeled using SSTD with a small threshold, OLOCAs are not superior adders anymore, due to their high probability of errors. The superiority of adders depends on how stringent the threshold of the SSTD metric is. Our new error metric helps to understand qualitatively and to quantify numerically the intrinsic characteristics of the adders. Furthermore, Fig. 4.8 shows that the optimal solutions cannot be obtained using the standard and conventional methodologies in most of the cases.

Finally, at the end of this subsection, we evaluate our framework. In an attempt to assess our framework, we progress in two steps: First, using our framework, we analyze various 16-bit approximate adders ranging from conventional approximate adders to new architectures developed from our template. In this step, we show the functionality of our framework, while comparing the adder architectures. Secondly, in order to show the exactness of our framework, we synthesize more architectures for a given error constraint. Here, the goal is to show that the selected architectures by our framework are actually the optimal architectures for a defined error limit.

Fig. 4.9 depicts a comparison of 16-bit adders developed using our template. The SMSE of the architectures for four different thresholds have been calculated using the error models. The applied thresholds range from small, stringent

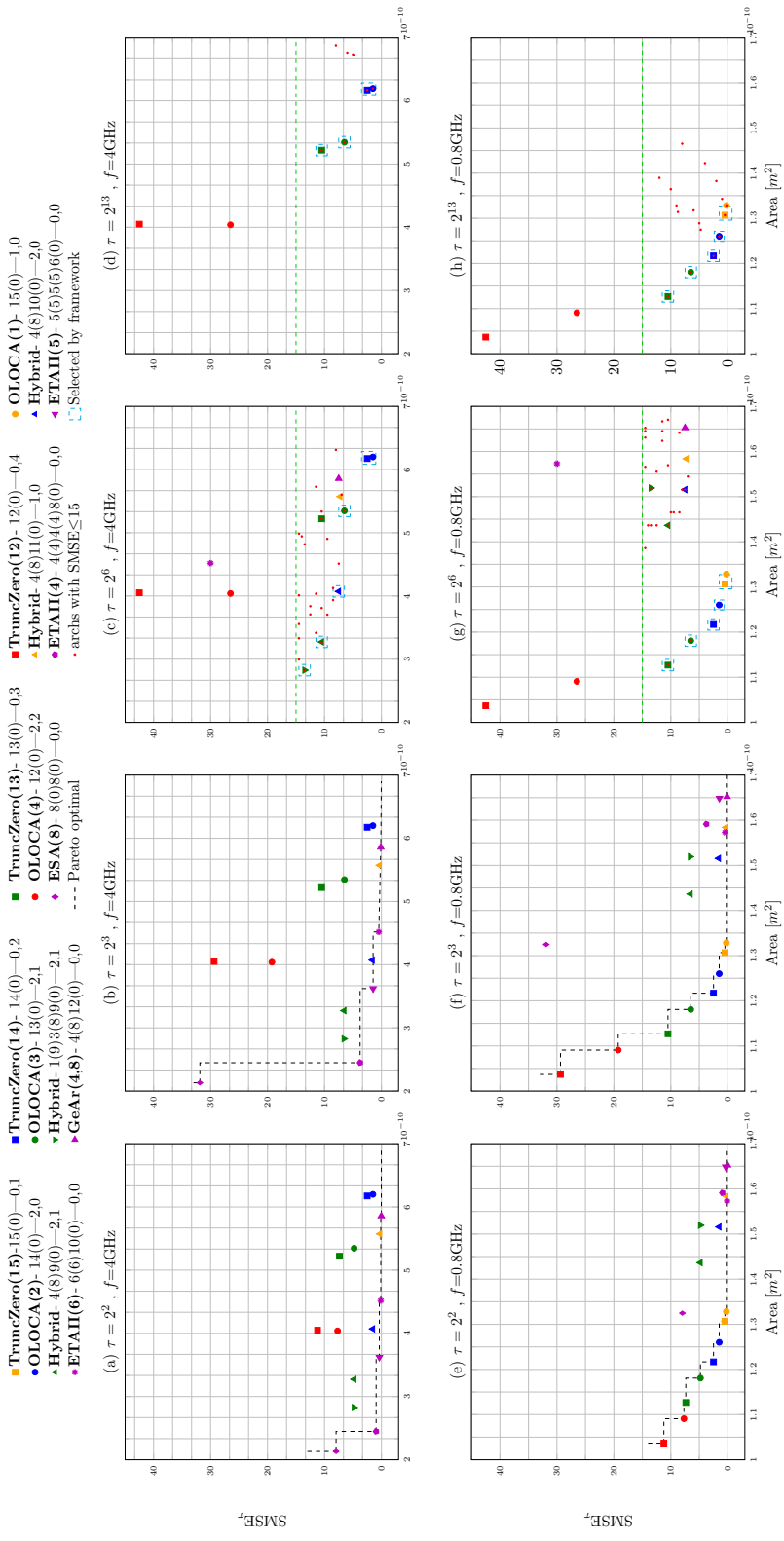


Fig. 4.9: Comparison of 16-bit adders using the proposed framework, applying two timing constraints and different thresholds. Saturated Mean Squared Error (SMSE) versus silicon area of adders: (a)-(d) Stringent timing constraint: frequency=4GHz ; (e)-(h) Relaxed timing constraint: frequency=800MHz. The final silicon area has been collected from synthesis. The green dashed line is used to show the error limit applied, as an example, to our framework.

thresholds to the one which is large enough to represent conventional MSE. The silicon area of the architectures have been obtained synthesizing the selected architectures. Different colors in the figure correspond to different bit-width of the upper part adder. For example, the TruncZero(13) is a truncated exact adder, in which the upper part adder is a 13-bit exact adder, and the lower part bits are constant-1s. As can be seen in Fig. 4.9(a)-(d), for relatively small timing constraints, when a stringent threshold applied, ETAII architectures are superior adders due to their low probability of errors. As the threshold increases, SMSE of ETAII architectures increase and hybrid architectures show superiority, as illustrated in Fig. 4.9(c). Continuing the threshold increment, the OLOCA adders tend to be superior architectures, as depicted in Fig. 4.9(d). As a result, qualitatively, the range of possible thresholds can be divided into three regions: small, medium and large thresholds. For small thresholds, conventional ETAIIs; for medium thresholds, hybrid architectures; and for large thresholds (conventional MSE), OLOCA adders are superior architectures.

The same comparison is shown in Fig. 4.9(e)-(h), carried out with relaxed timing constraint corresponding to area-efficient implementation of the architectures. As can be seen in the graphs, regardless of the threshold, there is no reason to use conventional approximate adders in this scenario. With relaxed timing constraints, for all the thresholds, truncated exact adders or OLOCA adders are superior architectures. Accordingly, the framework discriminates this scenario, where approximate adders do not provide considerable benefits from the scenarios which they do.

To assess the exactness of our framework, in this step of the experimental validation, we consider an illustrative error limit and analyze that case in detail. Then, we generate all the possible architectures which meet the defined constraint. As an example, we consider a practical value for the error constraint ($\text{SMSE}_\tau \leq 15$). The architectures which meet the above-mentioned constraint are shown with small red dots in Fig. 4.9. Additionally, the error limit is marked with green dashed lines in the figure. The number of architectures that meet the constraint increases, as the threshold is decreased. Because of the fact that there are many architectures that meet the constraint for small thresholds, in order to keep the graphs readable, we show the example for larger thresholds: τ=64 and τ=8192. In addition, we limit the architectures for the case of τ=64, and we consider only the architectures where the bit-width of their upper-part adders are 13-bit, as an illustration.

As can be seen in the figure, the architectures which have been chosen by our framework (marks with light blue border), among all the possible architectures (red dots), are in fact the optimal ones. In short, Fig. 4.9 not only shows the functionality of our framework, but also depicts how, in different scenarios, different approximate adders are superior. A notably important message of Fig. 4.9 is that not always using approximate adders results in a better design. It

should be discriminated where approximate adders provide significant benefits from where they do not. As mentioned, our framework is able to distinguish these scenarios.

4.4 Stochastic Mixed-PR: A Stochastically-Tunable Low-Error Adder

Stochastic computing is a technique to trade off energy and quality of a circuit. Instead of hiding variations under expensive guard-bands, designers have begun to relax traditional correctness constraints and deliberately expose hardware variability to higher levels of the computing stack [1]. The considerable power and performance overheads imposed by worst-case design practices can be relaxed by scaling up the frequency, i.e. Frequency Over-Scaling (FOS), or scaling down the operating voltage, i.e. Voltage Over-Scaling (VOS). An over-scaled design consumes lower power than its worst-case designed counterparts. At the same time, since the delay of some paths may now be greater than the clock period under certain conditions, timing violation may occur [1]. Over-scaling can be considered as a knob to dynamically tune the exactness of the adder architectures [48]. This technique, however, has not been remarkably embraced by the researchers. The rationale is that the conventional fast adder architectures start to make big errors as soon as they enter the stochastic regime.

Approximate adders, as key components of approximate arithmetic units, have attracted special attention of the researchers in the past few years. However, most of the existing architectures fix the level of hardware approximation statically, and hence, the level of approximation can not be dynamically tuned based on the application demand.

Nowadays, adaptation plays a major role in approximate processing [109]. In several scenarios, such as general purpose processors, DSP applications with variable quality signals (SNR), etc., both approximate and exact functionalities are required [110–112]. Fundamentally, this feature can be obtained in two ways: 1. by configuring the circuit logic or 2. by applying stochastic techniques.

In the first case, reconfigurability may be obtained by shutting down some parts of the logic [110] or bit truncation strategies [111]. These techniques use a control signal to switch between approximate and exact modes dynamically. However, both techniques are able to switch between an exact mode and an approximate mode with a fixed approximation level. The system can also be reconfigurable by adding an error-correction unit to the approximate architecture [73, 108]. This technique, however, increases the latency, silicon area and power consumption of the design.

Depending on the application demands for performance, different adder architectures are preferred. For a relaxed timing constraint, an exact adder is

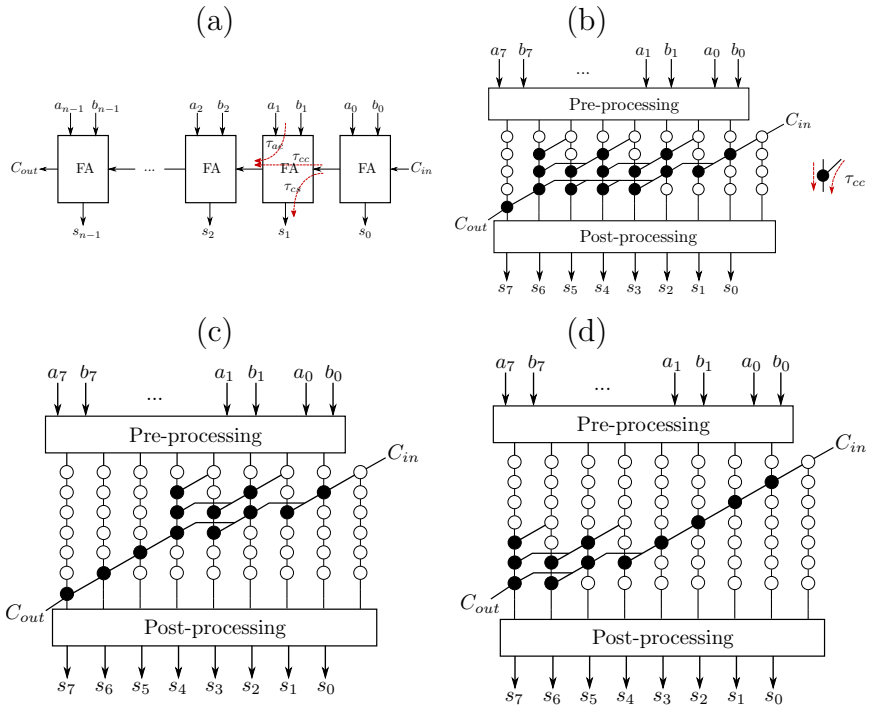


Fig. 4.10: The illustrative architectures of the exact adders: (a) Ripple-carry adder, (b) Parallel-prefix adder (Sklansky), (c) Std-mixed adder, (d) Mixed-PR adder

implemented with a Ripple-Carry Adder (RCA), Fig. 4.10(a). With stringent timing constraints, an exact adder is implemented with a Parallel-Prefix Adder (PPA), Fig. 4.10(b) as an example. For the cases in between, the synthesis tools implement an exact adder with a mixed adder of PPA and RCA, where the RCA is used for the MSBs to decrease the number of worst paths. Fig. 4.10(c) is as an example of the above-mentioned mixed adder. In this section, we propose the use of a mixed adder with late input MSB arrival time, which allows embracing the idea of over-scaling for the sake of reconfigurability. Aiming that, a precise analysis of the error behavior of adders in stochastic regime is presented in this section.

In the rest of this section, first, in subsection 4.4.1, the stochastic error behavior of adders are mathematically analyzed. Afterwards, we quantify the advantages of the proposed mixed adder using experimental results in subsection 4.4.2. Finally, the mathematical analyses developed in subsection 4.4.1 are validated.

4.4.1 Stochastic Analysis

Frequency (voltage) over-scaling is a conventional technique to dynamically trade-off the energy and accuracy of an adder. However, even a slight frequency (voltage) over-scaling in conventional adders causes large timing errors to occur and degrades the quality of the adders rapidly. This rapid quality loss under over-scaling was one of the main reasons to disregard stochastic computing techniques to dynamically tune the adder architectures.

As discussed in [48] and [40], with over-scaling, RCA fails moderately when entering the stochastic regime. Unlike that, a pure PPA fails catastrophically with over-scaling. However, RCA is extremely slow and starts to make errors at lower frequencies than a PPA. To solve the problem, we propose to use the mixed adder architecture with late input MSB arrival time, which shows a gradual increment in error when entering the stochastic regime. Hence, the proposed adder can be dynamically configured using FOS (VOS) techniques. We call the mixed adder with late input MSB arrival time, *Mixed-PR*, while the conventional mixed adder realized by the synthesis tool is called *Std-mixed*.

In this subsection, firstly, we introduce the idea of our stochastic analysis and the nomenclature used in this section. Then, the error characteristics of RCA in stochastic regime is analyzed. Once the behavior of an over-scaled RCA is understood, we can justify why our proposed adder can be efficiently used in the stochastic regime. Subsequently, the error formulas of the proposed adder are presented mathematically.

Nomenclature and analysis strategy

In order to analyze the error behavior of the adders in the stochastic regime, we consider the abstract adder depicted in Fig. 4.11, divided into two sub-blocks, where n_l and n_h are the bit-width of the blocks D_l and D_h, respectively. Let us consider the transition of inputs at time 0, and sampling the data at time T. Correspondingly, a^- and b^- represent the previous values of inputs a and b, respectively. In addition, due to the fact that we are analyzing blocks with late arrival signals, in order to perform our analysis, we have to differentiate between the logic (ideal) values (e.g. a and a^-), and the temporal evolution of

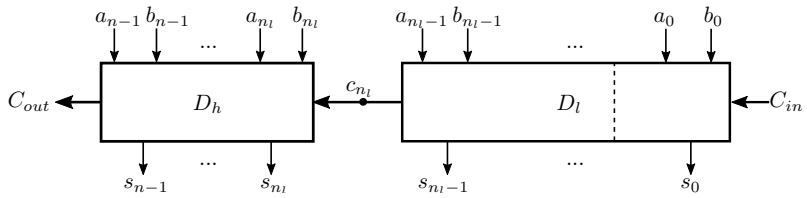


Fig. 4.11: The template of an adder split into two sub-blocks, used for the error analysis.

signal (e.g. $a[t]$ where $a[0^-]$=a^- and $a[0^+]$=a). Considering FOS, if we assume a scenario (clock period) in which D_h is the block that makes errors, with the information of the signal $c_{n_l}[t]$, we can characterize the stochastic errors of the adder. As a result, the problem of analyzing the error behavior of the adder of Fig. 4.11 in the stochastic regime can be split into two parts: 1. to characterize the temporal evolution of signal $c_{n_l}[t]$, and 2. to calculate the error as the result of the temporal evolution of signal $c_{n_l}[t]$.

Since the concepts of prefix structures are used often in this section, P, G and K are used for the terms propagate, generate and kill, respectively. Traditionally, these terms are defined as $p_i = a_i \oplus b_i$, $g_i = a_i.b_i$ and $k_i = \overline{a_i}.\overline{b_i}$, where i is the bit position. The theory of the parallel prefix architectures, [58], can be used to accelerate the calculation of carrys using the prefix operator $\bullet$ defined by:

$$g_{i:m} = g_{i:j} + p_{i:j}g_{l:m} \quad , \quad p_{i:m} = p_{i:j}p_{l:m} \ . \tag{4.27}$$

where $i \geq l \geq j \geq m$. Using the generate and propagate signals, the carry signals c_i can be computed as follows:

$$c_i = g_{i-1:0} + p_{i-1:0}.C_{in}. \tag{4.28}$$

In order to develop our error models, we use real delays instead of unit-gate delays. Therefore, the terms τ_{cc}, τ_{cs} are used. The notations are defined in Fig. 4.10. In addition, the error is defined as Eq. (4.1).

In order to develop the error formulas of the adders, let us first consider a single FA with a late carry input signal. As a result, the temporal evolutions of $S[t]$ and $C_{out}[t]$ depend on the temporal evolution of $C_{in}[t]$ and logical values of inputs a and b. More precisely, the logical equation of sum bit, $s_i = p_i \oplus c_i$, can be rewritten arithmetically, as follows:

$$\begin{aligned} s_i[T] &= p_i + c_i[T - \tau_{cs}] - 2p_ic_i[T - \tau_{cs}] \\ &= p_i + (-1)^{p_i}c_i[T - \tau_{cs}]. \end{aligned} \tag{4.29}$$

Taking Eq. (4.1) and Eq. (4.29) into consideration, the error corresponding to each sum bit of an adder is:

$$\varepsilon_{s_i}[T] = (-1)^{p_i}.\Big(c_i[T - \tau_{cs}] - c_i\Big).2^i, \tag{4.30}$$

where ε_{s_i} is the error corresponding to the output at bit position i. In a similar manner, the following equation models the error of the output carry of an n-bit adder:

$$\varepsilon_{C_{out}}[T] = p_{n-1}.\Big(c_{n-1}[T - \tau_{cc}] - c_{n-1}\Big).2^n \ . \tag{4.31}$$

Stochastic error analysis of RCA

In order to analyze the error behavior of RCA in stochastic regime, the blocks of Fig. 4.11 are instantiated with RCAs. Consequently, Eq. (4.30) and Eq. (4.31) can be rewritten in respect to node c_{n_l} as follows:

$$\varepsilon_{s_i}[T] = (-1)^{p_i}.\Big(c_{n_l}[T - \tau_{cs} - (i - n_l)\tau_{cc}] - c_{n_l}\Big).2^i.p_{i-1:n_l} \ , \tag{4.32}$$

$$\varepsilon_{C_{out}}[T] = \Big(c_{n_l}[T - n_h\tau_{cc}] - c_{n_l}\Big).2^n.p_{n-1:n_l} \ . \tag{4.33}$$

For the case where $\tau_{cc} = \tau_{cs}$, Eq. (4.32) and Eq. (4.33) can be combined as follows:

$$\varepsilon_{rca}[T] = \sum_{i=0}^{n_h-1}\Big(p_{n_l+i-1:n_l}\Big).\big(c_{n_l}[T - (i+1)\tau_{cc}] - c_{n_l}[T - i\tau_{cc}]\big).2^{n_l+i} \ . \tag{4.34}$$

The equations presented above show that to calculate the error of a RCA for a given clock period, the temporal evolution of $c_{n_l}[t]$ is needed.

In order to analyze the temporal evolution of $c_{n_l}[t]$, let us split the block D_l of Fig. 4.11 into two sub-blocks in a way that the higher significant sub-block meets the timing constraints of the specified clock period, while the lower significant sub-block violates the timing. The value of $c_{n_l}[t]$ is constant "1" or "0" over time, if the higher significant sub-block does not propagate the carry. Correspondingly, the adder does not make any error, due to constant behavior of signal $c_{n_l}[t]$. Alternatively, if the higher significant sub-block propagates the carry, the behavior of the carry out signal of the lower significant sub-block is copied to the signal $c_{n_l}[t]$. Consequently, $c_{n_l}[t]$ is just the delayed version of $c_i[t]$.

Let us consider $n_h = 1$, which results in a correspondence between $c_{n_l}[t]$ of Fig. 4.11 and $c_1[t]$ of Fig. 4.10(a). Considering the possible transitions of the inputs of the first FA, i.e. a_0 and b_0, the transitions of signal $c_1[t]$ can be found in Table 4.12. Depending on C_{in}, the probabilities of changes in $c_1[t]$ are different. The probabilities of transitions on $c_1[t]$ are tabulated in Table 4.13 for the scenario in which C_{in} is constant "0", as well as the scenario that C_{in} is an uniformly distributed random input (i.e. equals "0" or "1" with a probability of 1/2). In order to generalize the relation between input transitions and the carry signals, Table 4.14 tabulates the changes on the output carry of the FA in bit position i, i.e. $c_{i+1}[t]$, depending on the transitions of its inputs, i.e. a_i and b_i, and the input carry of the FA, i.e. $c_i[t]$. Considering Table 4.14, and having $c_1[t]$, the possible changes of $c_2[t]$ and corresponding probabilities can be derived. Having the changes of $c_2[t]$, the characteristic of the signal

Table 4.12: possible transitions on $c_1[t]$

Transition of a_0,b_0	Transition of $c_1[t]$	Prob.
$k_0^- \rightarrow k_0$	$0 \rightarrow 0$	$1/16$
$k_0^- \rightarrow p_0$	$0 \rightarrow C_{in}$	$1/8$
$k_0^- \rightarrow g_0$	$0 \rightarrow 1$	$1/16$
$g_0^- \rightarrow k_0$	$1 \rightarrow 0$	$1/16$
$g_0^- \rightarrow p_0$	$1 \rightarrow C_{in}$	$1/8$
$g_0^- \rightarrow g_0$	$1 \rightarrow 1$	$1/16$
$p_0^- \rightarrow k_0$	$C_{in}^- \rightarrow 0$	$1/8$
$p_0^- \rightarrow p_0$	$C_{in}^- \rightarrow C_{in}$	$1/4$
$p_0^- \rightarrow g_0$	$C_{in}^- \rightarrow 1$	$1/8$

Table 4.13: probability of changes on $c_1[t]$

$C_{in} = 0$			random C_{in}		
$c_1[0]$	$c_1[\tau]$	prob.	$c_1[0]$	$c_1[\tau]$	prob.
0	0	$9/16$	0	0	$1/4$
0	1	$3/16$	0	1	$1/4$
1	0	$3/16$	1	0	$1/4$
1	1	$1/16$	1	1	$1/4$

$c_3[t]$ can be derived and so on. Let us formulate characterizing of the $c_i[t]$ embracing the information presented in Table 4.13, and Table 4.14. If $Pr[C_1]$ is the prob. column of the table containing the characterization of $c_1[t]$, i.e. Table 4.13, $Pr[C_i]$ for $i \in \{2, n-1\}$ can be calculated as follows for the case C_{in} is uniformly random:

$$Pr[C_i] = \frac{1}{8}\begin{bmatrix} M \\ M \end{bmatrix} + \frac{1}{8}\begin{bmatrix} Pr[C_{i-1}] \\ Pr[C_{i-1}] \end{bmatrix} + \frac{1}{4}\begin{bmatrix} Pr[C_{i-1}]_h \\ 0 \\ 0 \\ Pr[C_{i-1}]_l \end{bmatrix}, \tag{4.35}$$

where M is a vector with the same size as $Pr[C_{i-1}]$ which only the first and the last elements of this vector are one and the rest are zero, i.e. $[1\ 0\ 0\ ...\ 0\ 1]^T$; $Pr[C_{i-1}]_h$ and $Pr[C_{i-1}]_l$ are the upper and lower half of the probability column of the previous table, i.e. c_{i-1}. Note that, the number of changes on a carry bit is linked to its bit position, and accordingly, the size of the table representing

transition possibilities of a carry bit depends on its bit position. Therefore, the size of the table for c_1, Table 4.13, is 2^2, the size of the table for c_2 is 2^3 and so on.

Stochastic error analysis of Mixed-PR adder

Conventionally, when the timing constraints lie between the delay of RCA and PPA, these two adder structures are mixed, where the lower bits a parallel prefix structure is realized, and a serial prefix is used for the upper bits. However, if D_l in Fig. 4.11 is a PPA, the behavior of signal $c_{n_l}[t]$ is different from that analyzed above. The rationale is that the critical path to node c_{n_l} instead of being one path as in RCA, are multiple paths because of a parallel prefix structure. Hence, the transition combinations of signal $c_{n_l}[t]$ that produce errors are more probable which results in a dramatic increase of error in stochastic regime [40, 48].

Let us analyze mathematically the error behavior of the adder of Fig. 4.11 when instead of RCA, a PPA is instantiated as the upper block (D_h).

Using Eq. (4.1), Eq. (4.29) and Eq. (4.28), we can develop the error formulas of the Mixed-PR adder in the stochastic regime, considering an Sklansky architecture, as an instance. Obviously, the general equations Eq. (4.30) and Eq. (4.31) are also valid here for the PPA. Similar to the RCA, we can write these equations in respect to the node c_{n_l}:

$$\varepsilon_{s_i}[T] = (-1)^{p_i}.\big(c_{n_l}[T - \tau_{cs} - \lceil \log_2(i - n_l + 1)\rceil \tau_{cc}] - c_{n_l}\big).2^i.p_{i-1:n_l}, \quad (4.36)$$

$$\varepsilon_{C_{out}}[T] = \left(c_{n_l}[T - \lceil \log_2(n_h + 1)\rceil \tau_{cc}] - c_{n_l}\right).2^n.p_{n-1:n_l} \ . \quad (4.37)$$

If we assume that $\tau_{cs} = \tau_{cc}$, and if the size of D_h is a power of 2, Eq. (4.36)

Table 4.14: probability of changes on $c_{i+1}[t]$ for the transitions of input

a_i,b_i transition	changes in $c_{i+1}[t]$					Prob.
	t=0	t=τ	t=2τ	...	t=T	
$\overline{p_i}^- \rightarrow \overline{p_i}$	c_{i+1}^-	c_{i+1}	c_{i+1}	...	c_{i+1}	1/4
$p_i^- \rightarrow \overline{p_i}$	c_i^-	c_{i+1}	c_{i+1}	...	c_{i+1}	1/4
$\overline{p_i}^- \rightarrow p_i$	c_{i+1}^-	c_i^-	c_i	...	c_i	1/4
$p_i^- \rightarrow p_i$	c_i^-	c_i^-	c_i	...	c_i	1/4

and Eq. (4.37) can be combined as follows:

$$\varepsilon_{mix}[T] = \sum_{i=0}^{\log_2(n_h-1)} \left(p_{n_l+i-1:n_l}\right).(c_{n_l}[T-(i+1)\tau_{cc}]-c_{n_l}[T-i\tau_{cc}]).2^{n_l+i} \ . \quad (4.38)$$

Considering the RCA as an architecture with a serial prefix-processing stage, Eq. (4.34) and Eq. (4.38) show that decreasing the clock period, the magnitude of error of each level of the prefix-processing stage corresponds to the first bit of that level. In other words, as Eq. (4.36)-(4.38) show, the Mixed-PR adder groups the bits of each level of the prefix-processing stage, and their errors compensate each other. This fact explains the reason why the error increment in Mixed-PR adder is gradual in stochastic regime.

4.4.2 Experimental Results

To assess the circuit characteristics and evaluate the architectures presented in this section, we have generated VHDL description of the adders. The adders are synthesized in a commercial low-power 65-nm library for 12- and 16-bit operands. Using back-annotated simulations, dynamic power dissipation of the adders is evaluated after synthesis. All the adders have been simulated for 10^7 uniformly distributed random input patterns. In this section, the mixed adders' name are followed by two numbers. The numbers represent the bit-widths of the higher significant and the lower significant blocks, respectively.

In order to check the accuracy of the presented models, the mean absolute error (MAE) versus clock period of the RCA and the Mixed-PR adder are depicted in Fig. 4.12. The model for 12- and 16-bit adders are compared with the simulation results. As can be seen in the graph, the behavior of the adders in the stochastic regime is predicted precisely using the presented models.

In order to evaluate the efficiency of the Mixed-PR adder, we have studied and compared 12-bit adders for different clock periods. The results are shown in Fig. 4.13. In this figure, the MAE versus energy-delay product (EDP) as well as area-delay product (ADP) of the adders are depicted. As illustrated in this figure, for the same values of error, the Mixed-PR outperforms all the other architectures from both the EDP and the ADP points of view. As an example, if an application limits the maximum error to $log_2(MAE) = 2.5$, the Mixed-PR-84 reduces the EDP and ADP about 40% in comparison with the conventional mixed adder, i.e. the Std-mixed-48 adder. The Mixed-PR-84 also reduces the EDP and ADP about 21% and 31%, respectively, in comparison with the adder of Ref. [111] when k is fixed to the optimal value of 4.

The energy consumption, silicon area, and the maximum clock period for a given error are reported in Table 4.15. As can be seen in the table, in contrast

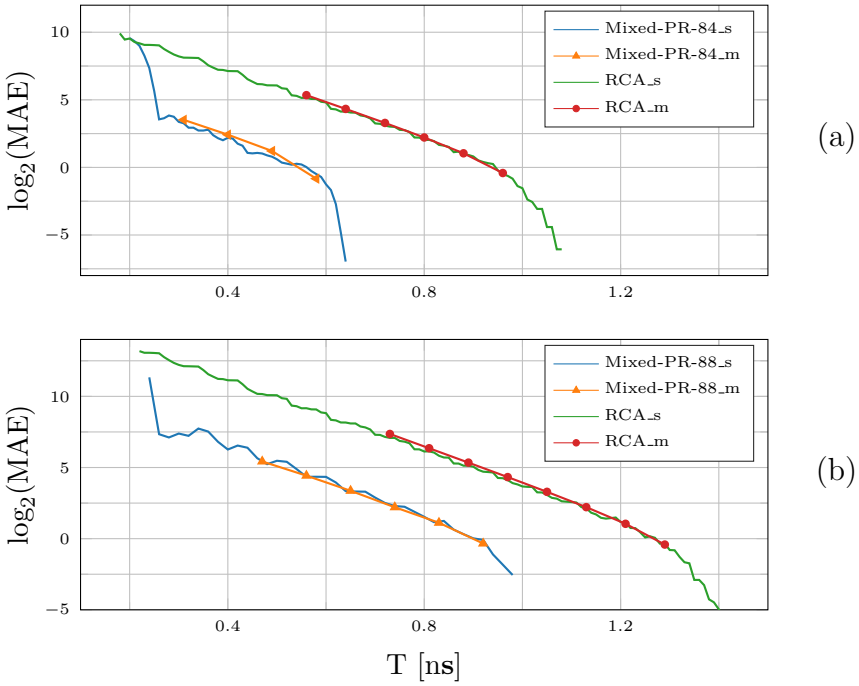


Fig. 4.12: Comparison of the simulation (_s) and the model (_m) results for the RCA and Mixed-PR for (a) 12-bit adders and (b) 16-bit adders, synthesized in a 65-nm technology.

to the extreme architectures (i.e. RCA and PPA), our Mixed-PR-84 provides a promising trade-off; it requires only 147 fJ and achieves an speed of 0.65 ns. Furthermore, regarding the maximum achievable speed for a given MAE (5 in this example), we observe that Mixed-PR-84 improves the speed by a factor of 1.75x when entering into the approximate domain. It is the largest improvement of any other adder. Incidentally, we can note that the performance of the traditional Std-mixed-48 is very poor in terms of the maximum speed for a given MAE, which is 0.63 ns versus the 0.37 ns of our adder. Because of this, stochastic techniques had been rejected until now as a suitable method to construct inexact adders.

To exemplify the advantages of our approach, let us consider two scenarios. In the first one, an adder needs to work in exact mode with a clock period of 1.1 ns, while in approximate mode a maximum MAE of 5 is allowed. For this low-demanding scenario, even the RCA and the adder of Ref. [111] with k=4 will work. They run in exact mode very efficiently (143 fJ), but the maximum speed in the inexact mode is very limited. The use of a PPA will allow a max speed of 0.31 ns in the inexact mode, but in the relaxed exact operation as much as 232 fJ is required. Our approach will operate in exact mode consuming only 147 fJ (factor 1.6 smaller than the PPA) and allowing a maximum speed in

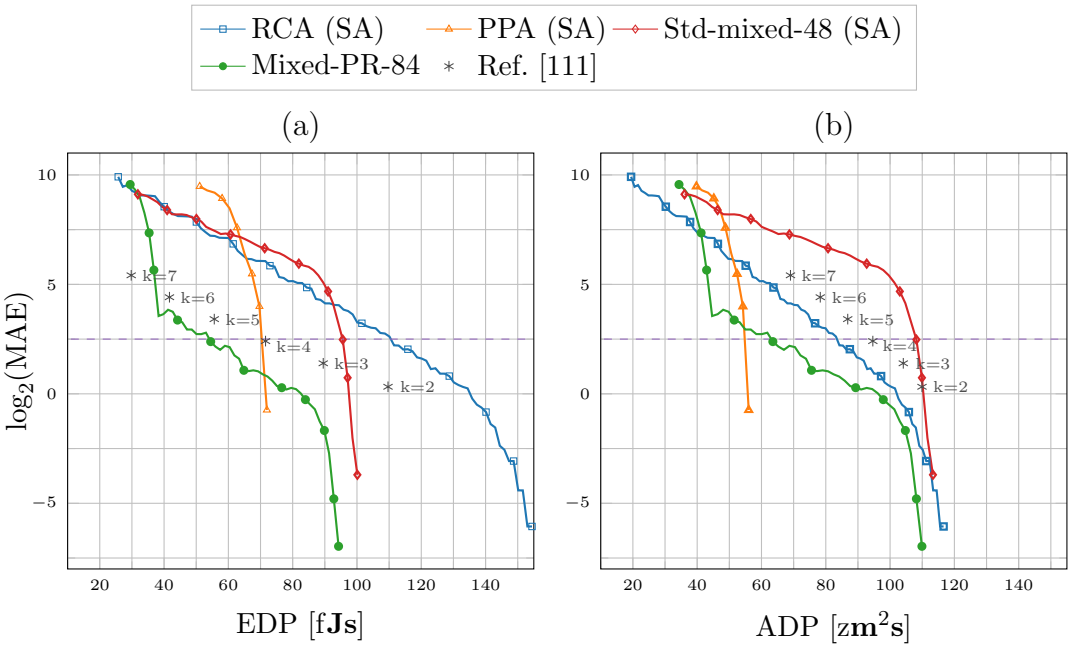


Fig. 4.13: 12-bit adders in stochastic regime: comparison of the Mixed-PR adder with the conventional exact adders generated by the synthesis tool (SA) as well as the approximate mode of the configurable adder of Ref. [111], synthesized in a commercial 65nm technology : (a) Mean Absolute Error (MAE) vs. Energy-Delay Product (EDP), (b) Mean Absolute Error (MAE) vs. Area-Delay Product (ADP).

the inexact mode of 0.37 ns (factor 2.5 faster than RCA). As a second scenario, let us consider that the adder needs to work with a period of 0.66 ns in the exact mode. In this case the RCA approach (and that of Ref. [111]) will not meet the timing requirements. Our approach still works and provides the same energy advantages versus PPA, as mentioned above.

In order to show the superiority of the Mixed-PR adder over its conventional counterpart, i.e. Std-mixed adder, the adders are evaluated in a more realistic scenario where the inputs have non-uniform distribution. In this case-study, the adders are used in a two-step image processing algorithm. In the first step, the Astronaut image is filtered with the average filtering using a 3×3 square kernel. In the second step, the result of the average filtering is fed to the Sobel filter to do the edge detection. The results of the process using the exact, Std-mixed and Mixed-PR adders are shown in Fig. 4.14. As can be seen in the figure, while the resulting image processed by the use of the Std-mixed is not usable, the Mixed-PR adder is still working perfectly in a relatively critical frequency.

Fig. 4.14: Comparison of the *Astronaut* image after Average and Sobel filtering using (a)exact adder, (b)adder of [111], (c)stochastic PPA, (d)stochastic RCA, (e)stochastic Std-mixed adder, and (f)proposed stochastic Mixed-PR adder. Stochastic adders work with a clock period of 0.5 ns.

4.5 Conclusion

In this chapter, an optimal approximate adder from a developed architectural template has been proposed. Using experimental results as well as mathematical analyses, it has been proved that the proposed adder *"Optimized Lower-part Constant-OR Adder (OLOCA)"* improves both the accuracy and hardware-cost in comparison with the previously reported best architectures. It has been shown that a 16-bit OLOCA provides a better trade-off between error and energy efficiency than that of LOA for all the possible configurations. Since, as discussed in Chapter 3 , LOA is the best existing approximate adder, the proposed OLOCA outperforms the state-of-the-art.

In addition to OLOCA architecture, a generic template for approximate adders combining the *small-errors* and the *infrequent-errors* philosophies has been proposed in this chapter. Using the proposed template, different classes of approximate adders can be developed. This includes hybrid and non-equal segmented adders which have not been studied before in the literature. The accurate mathematical analyses of the template have also been presented where using one compact formula, the error metrics of a wide range of approximate adders can be calculated. The detailed mathematical analyses enabled us to develop a framework to find the most suitable adder architecture for a target application. Using experimental results, it has been shown that for the scenarios that approximate adders provide considerable benefits, the develop framework finds an optimal approximate adder. Using extensive experimental analyses, we considered different threshold for SMSE error metric which correspond to different application scenarios. It has been shown that, for a relaxed timing constraint where the approximate architectures are area optimized, the OLOCA adder is a superior architecture for all the application scenarios. However, when considering delay optimized architectures, where the internal architectures of the approximate adders are faster but require more silicon area, the selection of the adders depend on the threshold of SMSE (i.e. error resiliency of the applications). In fact, for stringent timing constraints, OLOCA, hybrid, and

Table 4.15: Comparison of Mixed-PR-84 adder with 12-bit adders generated by the synthesis tool in a commercial 65nm.

Adder	Energy [fJ]	Area [μm^2]	T(ε=0) [ns]	T(ε=5) [ns]
Mixed-PR-84	147.30	171.72	0.65	0.37
RCA	143.00	108.00	1.09	0.78
Std-mixed-48	151.70	171.72	0.66	0.63
PPA	232.20	182.88	0.32	0.31

ETAII architectures are superior classes of adder architectures for large, medium and small thresholds, respectively.

Finally, in the last section, the use of a mixed adder with late input MSB arrival time has been proposed. Using detailed mathematical analyses, we have been shown the reason why conventional exact adders demonstrate a steep increase of error in stochastic regime resulting in being overlooked by the researchers. In response, we suggest a stochastically tunable Mixed-PR adder which shows a gradual decrease in accuracy makes it suitable to use for error resilient applications. In addition, unlike the existing reconfigurable adders which switch between two modes (i.e. fixed approximate and exact modes), the proposed stochastic adder can be configured in multiple modes depending on the operating frequency and supply voltage.

CHAPTER 5

Inaccurate Multipliers

5.1 Introduction

The multiplication of two n-bit numbers results in a 2n-bit product. For the sake of simplicity, unsigned multipliers are considered in the following. The inputs of the multipliers are of the form:

$$A = \sum_{i=0}^{n-1} a_i 2^i \quad ; \quad B = \sum_{j=0}^{n-1} b_j 2^j, \tag{5.1}$$

and the product of inputs A and B is:

$$P = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j 2^{i+j}, \tag{5.2}$$

where i and j are the bit position of a and b, respectively. In Eq. (5.2), the AND of input bits ($a_i b_j$) are the partial products.

As discussed in Chapter 3, the conventional partial product truncation is a promising approximation strategy which outperforms most of the recently proposed designs in the literature. Therefore, in this chapter, first, we study the concepts of truncated multipliers. Then, constant correction of the truncated multipliers is discussed. Afterwards, a template for truncated multipliers is developed where a data-dependent correction is presented. It is shown using experimental results that the developed multipliers outperform the existing architectures.

5.2 Truncated Multipliers

The truncated multipliers reduce power consumption as well as silicon area, and increase the speed of multiplication by removing K_p columns of the partial products of a multiplier. Fig. 5.1 shows a truncated multiplier. The lower significant part of the partial products tree is called Lower Significant Partial product (LSP) where the K_p LSB columns of the partial products are dismissed. Removing the partial products, not only reduces the area of a multiplier because of the removed AND gates of partial product generation stage, but also cuts part of the partial product accumulation stage out, where further improvements in power consumption reduction is achieved. Obviously, the improvements in the performance and hardware cost are traded off with the accuracy of the multiplication result.

Indeed, the error of a truncated multiplier corresponds to the addition of the removed partial products (i.e. the elements in LSP). Let us define the error as the difference of the approximate multiplication result and the exact output result:

$$\varepsilon = \tilde{P} - P, \tag{5.3}$$

where $\tilde{P}$ is the approximate (erroneous) product and P is the accurate multiplication result. Correspondingly, the error of a truncated multiplier can be formulated as follows:

$$\varepsilon_{trunc} = -\sum_{i=0}^{K_p-1}\sum_{j=0}^{K_p-i-1} a_i b_j 2^{i+j}, \tag{5.4}$$

where K_p is the number of truncated columns from the partial products of the multiplier as illustrated in Fig. 5.1. It is worth mentioning that the error of a truncated multiplier is independent from the bit-width of a multiplier (i.e. n) as can be seen in Eq. (5.4). For example, an 8-bit multiplier with 4-bit truncation ($K_p = 4$) has exactly the same error characteristics as a 16-bit multiplier with 4-bit truncation.

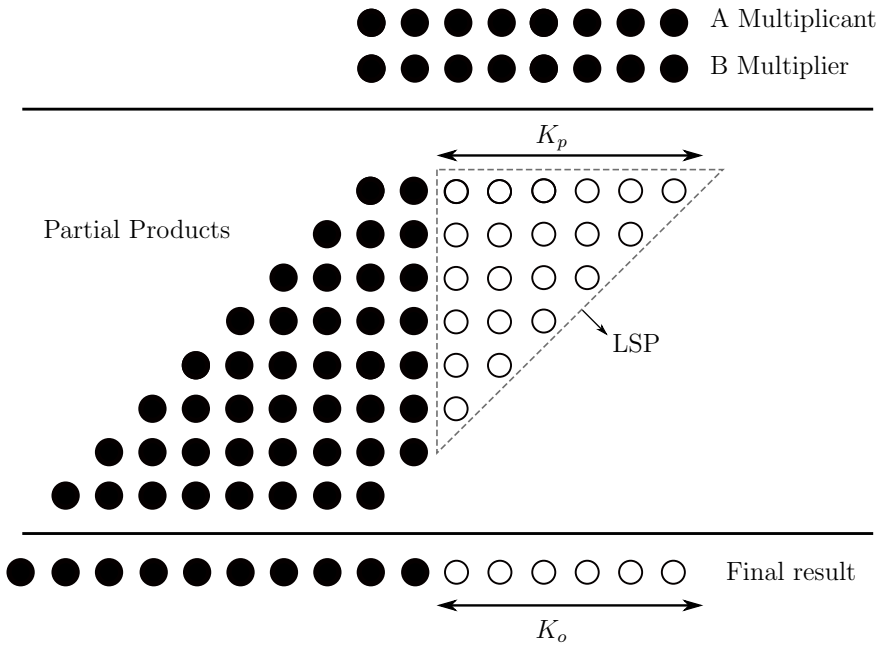


Fig. 5.1: The dot diagram of an 8×8 truncated multiplier with K_p-bit truncated partial products.

From Eq. (5.4), the maximum error of a truncated multiplier is produced when all removed partial products are '1'. As a result, the maximum error of a truncated multiplier (ε_{trunc_MAX}) can be formulated as below:

$$\varepsilon_{trunc\_MAX} = -\sum_{i=0}^{K_p-1}\sum_{j=0}^{K_p-i-1} 2^{i+j} = -(K_p-1)2^{K_p}-1, \tag{5.5}$$

Considering a uniform distribution where the probability of any input bit (a_i or b_j) being one is 1/2, the probability of each single partial product (a_ib_j) to be one is 1/4. Correspondingly, the probability of each element of LSP is 1/4. Illustratively, the LSP of a truncated multiplier with $K_p = 4$ is shown in fig. As a result, taking Eq. (5.4). and Fig. 5.2 into account, the average error of a truncated multiplier is calculated as follows:

$$\begin{aligned}\mu_{trunc} &= -\sum_{i=0}^{K_p-1}\sum_{j=0}^{K_p-i-1} Pr[a_ib_j]2^{i+j} \\ &= -\frac{1}{4}\sum_{i=0}^{K_p-1}(i+1)2^{i+j} \\ &= -(K_p-1)2^{K_p-2}-\frac{1}{4},\end{aligned} \tag{5.6}$$

where $Pr[a_i b_i]$ represents the probability of each partial product which equals ¼ in a uniform distribution, as mentioned above. Since the truncation errors are always negative, the mean absolute error of a truncated multiplier has the same magnitude as the average error of the multiplier:

$$MAE_{trunc} = (K_p - 1)2^{K_p - 2} + \frac{1}{4}, \tag{5.7}$$

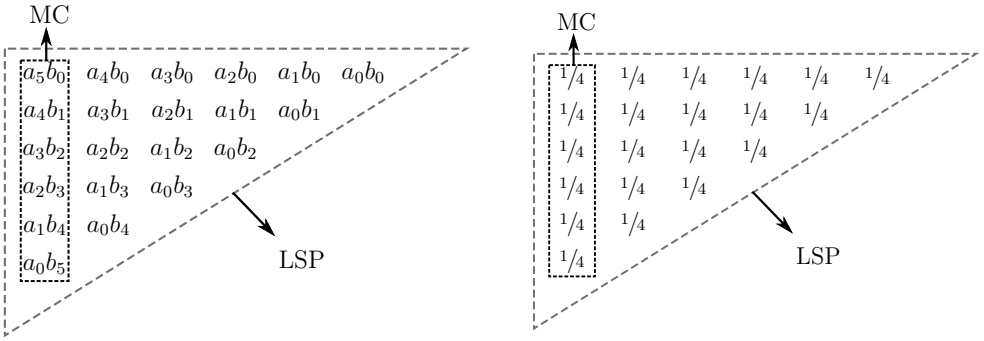


Fig. 5.2: (a) The LSP of a truncated multiplier with $K_p = 6$. (b) The probabilities of the partial products for a uniform distribution where $Pr[a_i b_j] = \frac{1}{4}$.

5.3 Constant Correction of Truncated Multipliers

The error of a truncated multiplier can be compensated with a constant correction. The best correction is clearly the summation of the removed partial products (see Eq. (5.4)). However, the summation of the LSP elements depends on the input data. In order to compensate the error with one constant, the additive inverse of the average error of the truncated multiplier results in the best correction. If C is defined as the correction constant, the mean squared error of the multiplier is calculated as below:

$$MSE_{trunc} = {\sigma^2}_{trunc} + (\mu_{trunc} + C)^2, \tag{5.8}$$

where ${\sigma^2}_{trunc}$ is the variance of the error of the truncated multiplier. In Eq. (5.4), since all the terms of the right hand of the equation is positive, it can easily be concluded that the best correction to have the minimum MSE is:

$$C = -\mu_{trunc}. \tag{5.9}$$

By substituting Eq. (5.6) in Eq. (5.9), one obtains the constant correction of the truncated multipliers as:

$$C = (K_p - 1)2^{K_p-2} + \frac{1}{4}. \tag{5.10}$$

The correction can be added to the partial products and correspondingly accumulated in the partial product reduction stage of the multiplier.

The average error of the truncated multipliers can be obtained with a more general formula as follows:

$$\mu_{trunc} = -\frac{1}{2^{K_p+1}} \sum_{i=0}^{2^{K_p}-1} \sum_{j=0}^{K_p-1} (i \times 2^j) \ mod \ 2^{K_p}. \tag{5.11}$$

For the applications in which the input data is not uniform, Eq. (5.11) can help to find the correction constant.

5.4 Data-dependent Correction of Truncated Multipliers

In order to achieve a higher accuracy, the error compensation of the truncated multipliers can be carried out dependent to the input data. Clearly, the most significant column of LSP has the the largest effect on the error due to its large weight. Let us call it MC in this section as highlighted in Fig. 5.2. Therefore, the partial products of the most significant column of LSP can be used for the error correction. Let us define m as the number of partial products of the MC used for the data-dependent correction of a truncated multiplier, where $m \in \{1, K_p\}$. In addition, we call the data-dependent correction using m partial products, the *m-bit correction*. The m-bit correction has 2^m different correction biases depending on particular combinations of the m partial products of the MC used for the correction. The selection of m is a trade-off between the hardware cost and accuracy. The higher m clearly results in a more accurate correction, while increasing the hardware cost. From Fig. 5.2, it can be seen that each partial product has correlation with its diagonal as well as horizontal neighboring partial products. This implies the fact that for the m-bit correction, where $m < K_p$, the middle partial products are the best selections for the correction due to the larger embrace of the correlated partial products. Therefore, the correction with middle m-bit of the MC results in the most accurate error correction with m partial products.

5.4.1 1-bit Correction

In order to do the error correction of truncated multipliers using only one partial product, the middle partial product of the MC is employed to estimate the correction values. Indeed, the correction biases are the summation of LSP elements for the cases that the middle partial product of MC is '0' or the case where it is '1'. As an illustration, Fig. 5.3 depicts the probability of the LSP elements of a truncated multiplier with $K_p = 6$. The probabilities are shown for the cases where the correction partial product is '0' and '1'. From the figure, it can be seen that the partial products which are in diagonal and horizontal line with the correction partial product are the ones which are correlated with it, and thereby, their probabilities differ from Fig. 5.2. As can be seen in the Fig. 5.3, when the correction partial product (i.e. a_2b_3) is '0', the partial products in its diagonal and horizontal lines have the probability of $1/6$ to be one. The rationale is that the partial products of the diagonal line include a_2 and the partial products of the horizontal line include b_3 in their partial products. Since $a_2b_3 = 0$, it can be concluded that probabilities of a_2 and b_3 to be one is $1/3$ (i.e. $Pr[a_2] = Pr[b_3] = 1/3$). Besides, by considering that another bit of each partial product in the diagonal and horizontal lines has the probability of $1/2$ to be one, by a simple calculation the probability of the aforementioned partial products is $1/6$. In a similar way, when the correction partial product is '1' (i.e. $a_2b_3 = 1$), it implies that both a_2 and b_3 are '1', and considering the above-mentioned descriptions, the probability of the partial products in the diagonal and horizontal line can easily be calculated and is equal to $1/2$.

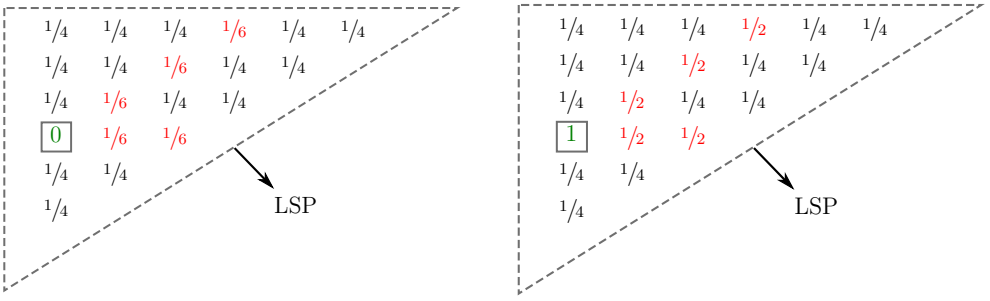


Fig. 5.3: The probabilities of the LSP elements of a truncated multiplier with $K_p = 6$ for a uniform distribution (a)when $a_2b_3 = 0$, (b)when $a_2b_3 = 1$.

To calculate the biases of a 1-bit correction, similar to the constant correction, the average value of LSP elements has to be calculated. In order to calculate the average error of the truncated multiplier the weighted probabilities of the partial products have to be accumulated. The average error of the truncated multiplier for a 1-bit correction can be calculated employing the difference of

the probabilities of Fig.5.3 with the probabilities of Fig. 5.2. From Fig. 5.3(a), when the correction partial product (for the illustrative example of the figure a_2b_3) is '0', the average error of a truncated multiplier can be calculated as:

$$\begin{aligned}\mu_{1-bit_0} &\approx \mu_{trunc} + \frac{1}{6}\sum_{i=\frac{K_p}{2}}^{K_p-2} 2^i + \frac{1}{4}2^{K_p-1} \\ &\approx -(K_p-1)2^{K_p-2} + \frac{1}{6}2^{K_p-1} - \frac{1}{6}2^{K_p/2} + 2^{K_p-3} \\ &\approx -(K_p - \frac{11}{6})2^{K_p-2},\end{aligned} \tag{5.12}$$

where μ_{1-bit_0} is the average error of the truncated multiplier when the correction partial product is '0'. Note that the above equation has been simplified by discarding the negligible terms. As discussed before, the correction biases are the additive zero of the average errors. As a result, the bias for the 1-bit correction when the correction partial product is '0' which we call it $bias_0$ is calculated as:

$$bias_0 = (K_p - \frac{11}{6})2^{K_p-2}. \tag{5.13}$$

Similarly, for the case that the correction partial product is '1' (Fig. 5.3(b)), the average error of the truncated multiplier is formulated as below:

$$\begin{aligned}\mu_{1-bit_1} &\approx \mu_{trunc} - \frac{1}{2}\sum_{i=\frac{K_p}{2}}^{K_p-2} 2^i - \frac{3}{4}2^{K_p-1} \\ &\approx -(K_p-1)2^{K_p-2} - \frac{1}{2}2^{K_p-1} + \frac{1}{2}2^{K_p/2} - \frac{3}{4}2^{K_p-1} \\ &\approx -(K_p + \frac{3}{2})2^{K_p-2},\end{aligned} \tag{5.14}$$

where μ_{1-bit_1} is the average error of the truncated multiplier when the correction partial product is '0'. As a result, the additive zero of μ_{1-bit_1} which we call it $bias_1$ is calculated as follows:

$$bias_1 = (K_p + \frac{3}{2})2^{K_p-2}, \tag{5.15}$$

which can be rewritten as:

$$\begin{aligned} bias_1 &= bias_0 + \frac{10}{3}2^{K_p-2} \\ &\approx bias_0 + 2^{Kp-1} + 2^{Kp-2}. \end{aligned} \tag{5.16}$$

As discussed above when the correction bit (in the illustrative example of Fig. 5.3 the partial product a_2b_3) is zero the $bias_0$ has to be added for the correction, and when it is '1', $bias_1$ has to be added in order to correct the truncated multiplication results. Eq. (5.16) suggests that for a 1-bit correction, the $bias_0$ can be added to the partial products of the truncated multiplier, while the correction partial product should be placed in bit positions $K_p - 1$ and $K_p - 2$. Therefore, when it is one, $2^{K_p-1} + 2^{K_p-2}$ is added to $bias_0$ as in Eq. (5.16). Fig. 5.4 shows a 1-bit correction for a 4-bit truncated multiplier with $K_p = 4$ as an illustration.

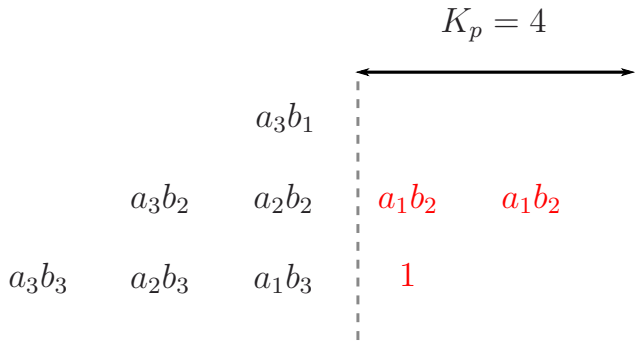


Fig. 5.4: The partial product tree of a 1-bit corrected truncated multiplier with 4-bit truncation ($K_p = 4$). The red colored elements in LSP are the correction biases.

5.4.2 2-bit Correction

In order to compensate the error of a truncated multiplier using 2 partial products of MC, the middle partial products are employed to estimate the correction biases. Here, similar to the 1-bit correction, first the probabilities of LSP elements for the four different values of the correction partial products are calculated. Then, the average error of the truncated multiplier for each of these scenarios is calculated. Correspondingly, the correction biases are the additive zero of the average errors. Since each partial product has correlation with its diagonally as well as horizontally lined partial products, the probability of the elements of the LSP can be calculated based on the probabilities of the correlated partial products of the MC. Each partial product of the LSP has correlation with two partial products of MC, one in its diagonal line and the other on its horizontal line. As an instance, in Fig. 5.2, partial product a_2b_1

has correlation with a_4b_1 and a_2b_3 which are two partial products of the MC. Table 5.1 tabulates all the possible combinations of the two elements of the MC column. In this table, a_ib_j is the partial product of the LSP, and the a_ib_x and a_yb_j are its correlated partial products located on MC. The indices x and y can be any number in the range 0 to $K_p - 1$. Using this table, the probabilities of the LSP elements can be found.

Table 5.1: The resulting probability for the elements of LSP based on the combination of the probabilities of the correlated partial products on the MC.

$Pr[a_ib_x]$	$Pr[a_yb_j]$	$Pr[a_ib_j]$
0	0	$1/9$
0	1	$1/3$
1	0	$1/3$
1	1	1
0	$1/4$	$1/6$
$1/4$	0	$1/6$
1	$1/4$	$1/2$
$1/4$	1	$1/2$
$1/4$	$1/4$	$1/4$

Fig. 5.5 shows the probabilities of the LSP elements of a truncated multiplier with $K_p = 6$ calculated with the help of Table 5.1. Similar to the 1-bit correction, the average error of the truncated multiplier for a 2-bit correction can be calculated employing the difference of the probabilities of Fig. 5.5 with the probabilities of Fig. 5.2. From Fig. 5.5(a), when the correction partial products (for the illustrative example of the figure a_2b_3 and a_3b_2) are '0', the average error of a truncated multiplier can be calculated as:

$$\begin{aligned}\mu_{2-bit_{00}} &\approx \mu_{trunc} + \frac{1}{3}\sum_{i=\frac{K_p}{2}}^{K_p-3} 2^i + \frac{11}{36}2^{K_p-2} + \frac{1}{2}2^{K_p-1} \\ &\approx -(K_p-1)2^{K_p-2} + \frac{1}{3}2^{K_p-2} - \frac{1}{3}2^{K_p/2} + \frac{11}{36}2^{K_p-2} + 2^{K_p-2} \\ &\approx -(K_p - \frac{7}{3})2^{K_p-2},\end{aligned} \tag{5.17}$$

where $\mu_{2-bit_{00}}$ is the average error of the truncated multiplier when the correction partial products are both '0'. The correction bias $bias_{00}$ which is the additive

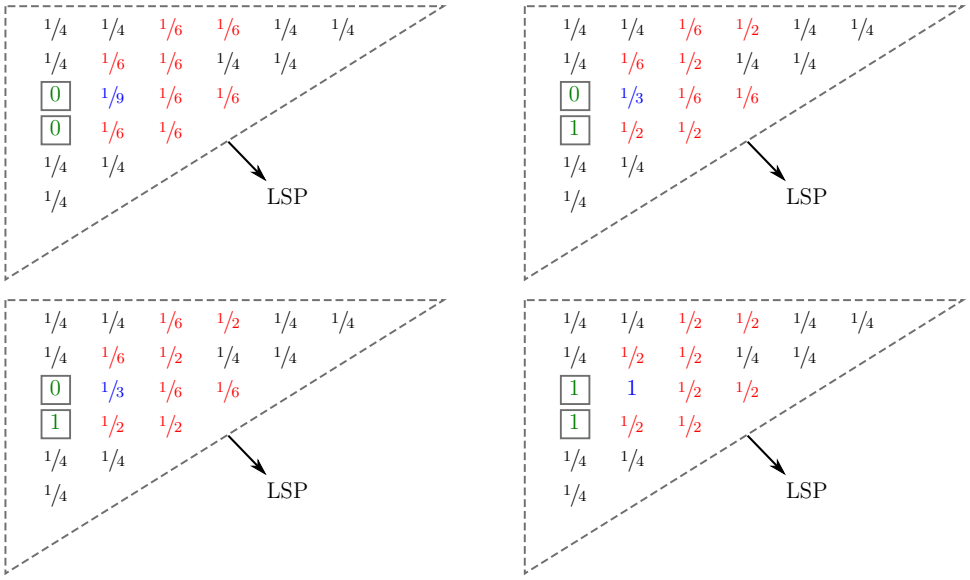


Fig. 5.5: The probabilities of the LSP elements of a truncated multiplier with $K_p = 6$ for a uniform distribution (a)when $a_2b_3 = 0$ and $a_3b_2 = 0$ (b)when $a_2b_3 = 0$ and $a_3b_2 = 1$ (c)when $a_2b_3 = 1$ and $a_3b_2 = 0$ (d)when $a_2b_3 = 1$ and $a_3b_2 = 1$.

zero of the average error is calculated as:

$$bias_{00} = (K_p - \frac{7}{3})2^{K_p-2} \approx (K_p - 2)2^{K_p-2}. \tag{5.18}$$

By the same token, for the case that one of the correction partial products is '1' (Fig. 5.3(b)(c)), the average error of the truncated multiplier is formulated as below:

$$\begin{aligned} \mu_{2-bit_{01}} &\approx \mu_{trunc} - \frac{1}{3}\sum_{i=\frac{K_p}{2}}^{K_p-3} 2^i - \frac{1}{4}2^{K_p-2} - \frac{1}{2}2^{K_p-1} \\ &\approx -(K_p - 1)2^{K_p-2} - \frac{1}{3}2^{K_p-2} + \frac{1}{3}2^{K_p/2} - \frac{1}{4}2^{K_p-2} - \frac{1}{2}2^{K_p-1} \\ &\approx -(K_p + \frac{7}{12})2^{K_p-2}, \end{aligned} \tag{5.19}$$

where $\mu_{1-bit_{01}}$ is the average error of the truncated multiplier when one the correction partial products is '0' and the other one is '1'. As a result, the additive zero of μ_{2-bit_01} which we call it $bias_{01}$ is calculated as follows:

$$bias_{01} = (K_p + \frac{7}{12})2^{K_p-2}, \tag{5.20}$$

which can be rewritten as:

$$\begin{aligned} bias_{01} &= bias_{00} + \frac{35}{12}2^{K_p-2} \\ &\approx bias_{00} + 2^{Kp-1} + 2^{Kp-2}. \end{aligned} \tag{5.21}$$

In addition, when both correction partial products are '1' (Fig. 5.3(d)), the average error of the truncated multiplier is calculated using the following formula:

$$\begin{aligned} \mu_{2-bit_{11}} &\approx \mu_{trunc} - \sum_{i=\frac{K_p}{2}}^{K_p-3} 2^i - \frac{5}{4}2^{K_p-2} - \frac{3}{2}2^{K_p-1} \\ &\approx -(K_p-1)2^{K_p-2} - 2^{K_p-2} + 2^{K_p/2} - \frac{5}{4}2^{K_p-2} - \frac{3}{2}2^{K_p-1} \\ &\approx -(K_p + \frac{17}{4})2^{K_p-2}, \end{aligned} \tag{5.22}$$

where $\mu_{1-bit_{11}}$ is the average error of the truncated multiplier when both of the correction partial products are '1'. The additive zero of $\mu_{2-bit_{11}}$ called $bias_{11}$ can correspondingly calculated as follows:

$$bias_{11} = (K_p + \frac{17}{4})2^{K_p-2}, \tag{5.23}$$

which relative to $bias_{00}$ can be rewritten as:

$$\begin{aligned} bias_{11} &= bias_{00} + \frac{79}{12}2^{K_p-2} \\ &\approx bias_{00} + 2 \times 2^{Kp-1} + 2 \times 2^{Kp-2}. \end{aligned} \tag{5.24}$$

Eq. (5.21) and Eq. (5.24) indicate that to correct a truncated multiplier with 2 bits, besides a constant bias ($bias_{00}$), the correction partial products should be placed in their bit positions (i.e. $K_p - 1$) and also be repeated in bit position $K_p - 2$. Fig. 5.6 shows a 2-bit correction for a 4-bit truncated multiplier with $K_p = 4$ as an illustration.

5.4.3 m-bit Correction

The correction of truncated multipliers can be extended to m-bit correction, where increasing m increases the hardware cost of the multiplier while the error behaviour of the multiplier is improved. Consequently, choosing the right m, depends on the requirements of the application. Fig. 5.7 shows an illustrative example with probabilities of the LSP elements where correction is carried out with $m = K_p$. In a similar way as for 1-bit and 2-bit corrections, first, the

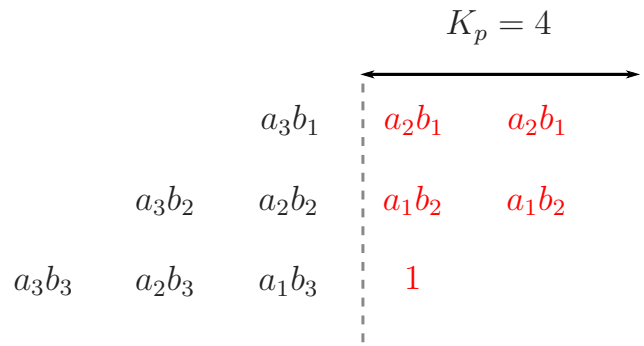


Fig. 5.6: The partial product tree of a 2-bit corrected truncated multiplier with 4-bit truncation ($K_p = 4$). The red colored elements in LSP are the correction biases.

correction is calculated when all the correction partial products are zero (in the example of Fig. 5.7(a) when all the elements of MC are '0'). Let us call this correction bias as $bias_0$. Afterwards, the correction corresponding to each partial product of MC is calculated. In other words, the average error of the truncated multiplier when only one element of the MC is '1' is calculated. From Fig. 5.7(b)-(c), it can be seen that regardless of which element is '1' the bias (the average of the LSP elements) remains almost the same. Let us call this $bias_1$.

From Fig. 5.7, Table 5.1 and earlier discussions, the $bias_0$ which is the additive zero of the average error in case all the correction bits are zero, can be calculated by the following formula:

$$\begin{aligned} bias_0 &\approx \mu_{trunc} - \frac{5}{36}\sum_{i=K_p-m}^{K_p-2}(i+1+m-K_p)2^i - \frac{1}{4}n2^{K_p-1} - \frac{1}{6}2^{K_p-2} - \frac{1}{12}(K_p-m)2^{K_p-3} \\ &\approx (4K_p - 3m - 3)2^{K_p-4}. \end{aligned} \tag{5.25}$$

The coefficient of Eq. (5.25) are the difference of the probabilities of the LSP elements of the truncated multiplier (Fig. 5.2(b)) and the probabilities extracted from Table 5.1 for the truncated multiplier with correction bits.

Similar to 1-bit and 2-bit corrections, $bias_1$ can be calculated relative to $bias_0$. By observing the difference of probabilities of LSP elements of the case where all the correction bits are zero, and the case which only one of the correction bits is '1', the formula for $bias_1$ is be computed as:

$$bias_1 = bias_0 + 2^{K_P-1} + 2^{K_p-2}. \tag{5.26}$$

Eq. (5.26) implies that in order to correct a truncated multiplier with m bit, besides a constant bias (i.e. $bias_0$), a pair of each correction partial product

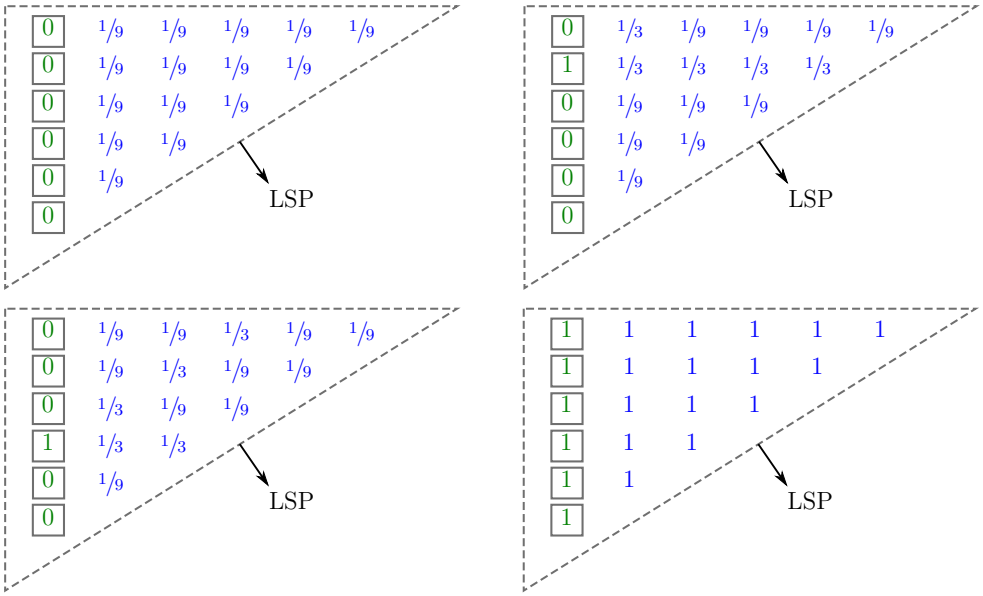


Fig. 5.7: The probabilities of the LSP elements of a truncated multiplier with $K_p = 6$ for a uniform distribution (a) when all the elements of MC are '0',(b)-(c) when only one element of MC is '1' and the rest are '0', (d) when all the elements of MC are '1'.

need to be placed in bit positions $K_p - 1$ and $K_p - 2$ to achieve the best error compensation.

5.5 A Template for Truncated Multipliers

The main challenge for the design of the data-dependent multipliers after finding the best compensation is how to efficiently apply the corrections to hardware. In this section, a template for truncated multipliers is proposed, where based on the variables of the template, the best correction is found using the formulas presented in previous sections.

A full-width digital $n \times n$ multiplier computes the 2n-bit product as a weighted sum of the partial products [75, 76]. However, in many signal processing applications, the data path is of a fixed-width. In these applications, the precision of 2n bits at the output of multiplier is often more than required. As a result, a multiplier with less output bits where the accuracy is traded-off with hardware cost can be used in these applications. An n-bit truncated multiplier with n-bit output is a widely used design in digital signal processors where the input and outputs of the data path can be stored in registers with the same bit-width.

Besides, considering the whole system, rounding the output of the multiplier or set some output bits to constants, reduces the number of registers at

multiplier's output. Moreover, the reduced bit-width when passed to the next block requires less hardware for computation and communication of the data, resulting in a twofold improvement in hardware cost and performance. As a result the template which is introduced here combines the idea of full-width multipliers which are of interest of approximate computing community and the fixed-width multipliers where used frequently in signal processing applications. As a result the template employs two variables. One is the bit-width of removed partial products (K_p) and the other one is the number of LSBs of the multipliers output which are pruned or are set to constant (K_o). A conceptual diagram of the proposed template is depicted in Fig. 5.8. In the figure partial product tree of a truncated multiplier is illustrated where the LSP of the multiplier is hatched.

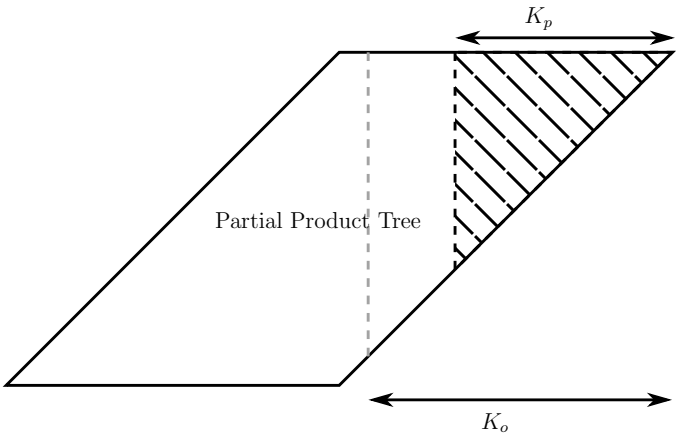


Fig. 5.8: The conceptual diagram of the template of the truncated multipliers. The hatched part is the LSP with K_p the bit-width of the removed partial products. K_o is the width of output truncation or constant output.

In order to adapt the correction biases calculated in previous sections to the proposed template, the formulas need to be modified. Indeed, the output truncation with K_o bits introduces a rounding error which is added to the previously calculated formulas. The K_p bits of the multiplier output has already been truncated due to removing the LSP. Therefore, the additional rounding error corresponds to the $K_o - K_p$-bit of bit positions K_p to $K_o - 1$. Since the probability of the output bits in these bit positions to be one is $1/2$, the

additional rounding average error can be calculated as:

$$\begin{aligned} \mu_{rounding} &= -\frac{1}{2}\sum_{i=K_p}^{K_o-1} 2^i \\ &= 2^{K_p-1}2^{K_o-1}. \end{aligned} \tag{5.27}$$

Since the rounding error and truncation error are independent, the average error of template is calculated as:

$$\mu_{temp} = \mu_{trunc} + \mu_{rounding}. \tag{5.28}$$

As discussed earlier, the correction constants are the additive inverse of the average error of the multiplier. As a result, the rounding error need to be added to the Eq. (5.25) and Eq. (5.26) to modify the bias corrections and adapting them for the presented template. In fact, it is enough to add the additional rounding error to $bias_0$, while the $bias_1$ is determined relative to $bias_0$. The modified correction biases, as a result, are as follows:

$$\begin{aligned} bias_{0_{temp}} &= (4K_p - 3m - 3)2^{K_p-4} + 2^{K_o-1} - 2^{K_p-1} \\ &= (4K_p - 3m - 11)2^{K_p-4} + 2^{K_o-1}. \end{aligned} \tag{5.29}$$

$$bias_{1_{temp}} = bias_{0_{temp}} + 2^{K_P-1} + 2^{K_p-2}, \tag{5.30}$$

Let us call the m-bit corrected template multiplier as *MultA*. Due to the fact that K_o bits of the LSBs of the MultA output are truncated, the sum of the correction partial products in bit position $K_p - 1$ and $K_p - 2$ are subsequently pruned. As a result, this rounding error need to be considered as well. Considering the probability of the output sum of the correction bits as $1/2$ (which is almost precise for $m > 3$), the modified correction biases of MultA are:

$$\begin{aligned} bias_{0_{MultA}} &= (4K_p - 3m - 11)2^{K_p-4} + 2^{K_o-1} + \frac{1}{2}(2^{K_p-1} + 2^{K_p-2}) \\ &= (4K_p - 3m - 5)2^{K_p-4} + 2^{K_o-1}. \end{aligned} \tag{5.31}$$

$$bias_{1_{MultA}} = bias_{0_{MultA}} + 2^{K_P-1} + 2^{K_p-2}, \tag{5.32}$$

As a result the best error correction to the truncated multiplier of the template architecture is to employ m partial product form MC in LSP, duplicate them in bit position $K_p - 2$ and add the constant correction of $bias_{0_{MultA}}$ to the partial product tree.

Although MultA introduces an optimal error correction, the fact that the m

correction partial products of MC need to be duplicated increases the hardware cost of the corrected multiplier. In consideration of the fact that the duplication of correction bits, which are in bit position $K_p - 1$, to bit position $K_p - 2$ is indeed corresponds to multiply the correction bits by '1.5'. In other words, the weight of the correction partial products need to be '1.5'. Alternatively, one can employ the weight of 2 for the correction which correspond to put the correction bits in one higher bit position (i.e. K_p). However, a modification to $bias_0$ is required due to the excess 0.5 weight of $bias_1$. Let us call this alternative design *MultB*. Thus, the $bias_{1_{MultA}}$ is formulated as:

$$bias_{1_{MultB}} = bias_{0_{MultB}} + 2^{K_P}, \tag{5.33}$$

Regarding the $bias_0$ the extra rounding error added in Eq. (5.31) does not appear in MultB. As a result, the only modification to $bias_0$ of MultB is to consider the excess 0.5 weight of correction bits. Correspondingly, because the m correction bits are in bit position K_P instead of $K_p - 1$ and $K_p - 2$, an extra correction has already been added which needs to be deducted from Eq. (5.29). Since the probability of each partial product is 1/4, the aforementioned extra correction can be calculated by the difference of the weighted sum of the correction elements when they have the weight of 2 (MultB) with when their weight is 1.5 (MultA).

$$\mu_{E_b} = m(\frac{1}{4})2^{K_p} - m(\frac{1}{4})(2^{K_p-1} + 2^{K_p-2}) = m.2^{K_p-4}, \tag{5.34}$$

where μ_{E_b} is the extra correction which has already added to MultB due to the use of correction bits with weight 2 instead of weight 1.5 as in MultB. Correspondingly, $bias_1$ of multB can be calculated as:

$$\begin{aligned} bias_{0_{MultB}} &= (4K_p - 3m - 11)2^{K_p-4} + 2^{K_o-1} - m.2^{K_p-4}) \\ &= (4K_p - 4m - 11)2^{K_p-4} + 2^{K_o-1}. \end{aligned} \tag{5.35}$$

5.6 Experimental Results

To assess the circuit characteristics and evaluate the architectures presented in this chapter, we have generated VHDL description of the multipliers. The architectures are synthesized in a commercial low-power 40-nm library for 16-bit operands. Using back-annotated simulations, dynamic power dissipation of the multipliers is evaluated after synthesis in the frequency of 200MHz. All the multipliers have been simulated for 10^7 uniformly distributed random input patterns.

In this section, first, the proposed data-dependent correction strategies are evaluated considering different number of bits for correction (m-bit correction

for $m \in \{1, K_p\}$). Following the evaluation and comparison of the existing approximate multiplier in Chapter 3, the truncated multiplier shows the best trade-off between energy consumption and accuracy. Correspondingly, if we show that the proposed architectures outperform the truncated multiplier for all the m-bit corrections, they outperform the exiting approximate multipliers in the literature. As a result, the proposed architectures *MultA* and *MultB* are compared, for different K_p truncation and all the possible m-bit truncation, with the truncated multiplier. Afterwards, in a more concrete illustration, MultB with a maximum m (i.e. K_p-correction) and a low m (i.e. 2-bit correction) are compared with the existing approximate multipliers, as well as two stochastic exact multipliers. Subsequently, we configure the template to have a fixed-width multiplier, and the proposed architectures are then compared with the fixed-width multiplier of [79].

In order to evaluate the proposed architectures, they are compared for 3 different truncations ($K_p = 12$, $K_p = 14$, and $K_p = 16$) with the truncated multiplier (TruncM). The comparison for mean absolute error is shown in Fig. 5.9. The K_p for TruncM is changed from 10 to 16 in the figure. The rationale to choose only 3 different K_p for the proposed architecture is to keep the figure readable. As can be seen in the figure, both MultA and MultB outperform the TruncM for all the "m"s, which means the proposed architectures even with 1-bit corrections outperform the existing architectures in the literature. In addition, it can be observed that, as expected, the difference between MultA and MultB is noticable for larger ¡m¿s, where MultB is slightly more accurate with the cost of more ADP and PDP. In the case of smaller *'m'*s, the behaviours of MultA and MultB are marginally different, and the reason is that the area and power consumption overhead of MultA over MultB reduces by decreasing the m, while at the same time, the superiority of its accuracy decreases.

The differences between the proposed multipliers and the truncated multiplier are highlighted on 3 spots on the Fig. 5.9 as illustration. As can be see in the figure, MultB reduces the area-delay product by about 15%, for the same accuracy, in comparison with the TruncM. In another scenario, for the same power-delay product, the proposed multiplier mitigates the mean absolute error by 67%, as shown in Fig. 5.9(b). In the same figure, the 18% reduction in the energy consumption is pointed out for the same MAE.

In Fig. 5.10, a similar comparison for mean squared error is demonstrated. In the same way, in this graph, the TruncM is shown for K_p from 10 to 16, while the proposed multipliers are shown for $K_p = 12$, $K_p = 14$, and $K_p = 16$. Considering the trade-off between the MSE and the hardware cost, it can be seen that the proposed multipliers perform better than the TrunM. As an illustration, a scenario where the TruncM and MultA have the same PDP is highlighted. It can be observed that the MultA with $K_p = 14$ and 1-bit

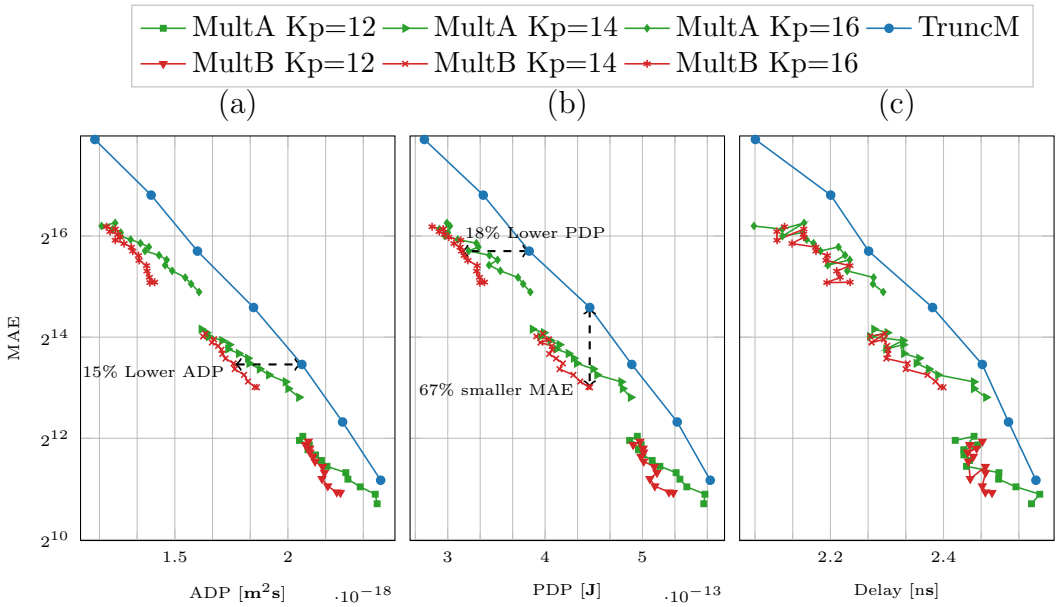


Fig. 5.9: Comparison of Mean Absolute Error (MAE) of the multipliers vs. their (a)Area-Delay Product (ADP), (b)Power-Delay Product (PDP), (c)Delay. The K_p of TruncM varies from 10 to 16.

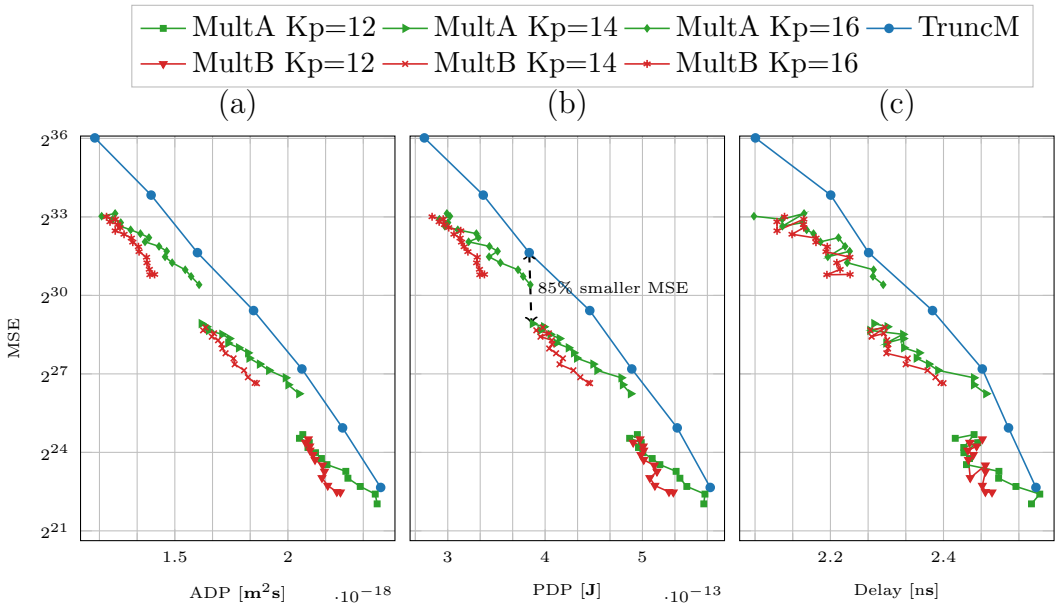


Fig. 5.10: Comparison of Mean Squared Error (MSE) of the multipliers vs. their (a)Area-Delay Product (ADP), (b)Power-Delay Product (PDP), (c)Delay. The K_p of TruncM varies from 10 to 16.

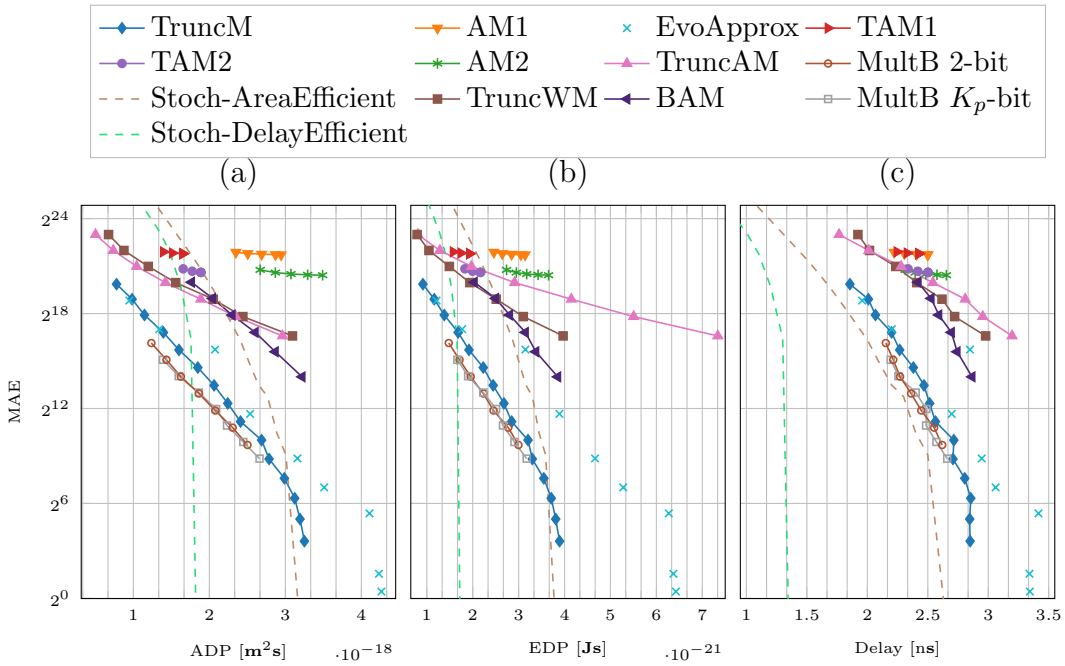


Fig. 5.11: Comparison of MultB with existing approximate multipliers for Mean Absolute Error (MAE) vs. (a) Area-Delay Product (ADP), (b)Energy-Delay Product (EDP), (c)Delay.

correction reduces the mean squared error by 85% in comparison with the TruncM with $K_p = 14$.

The comparison of the proposed multipliers with the state-of-the-art is depicted in Fig. 5.11 and Fig. 5.12 for mean absolute error and mean squared error, respectively. Due to reason that the difference of MultA and MultB are hardly observable when depicted in these graphs, only MultB is compared with the existing multipliers. In both of the graphs, MultB is shown for K_p from 10 to 16. In addition, two divergent m are selected for the MultB to be depicted in the figures; K_p-bit correction where all the bits of the MC column of LSP are employed for the correction, and 2-bit correction where two middle partial products of MC are used for the correction. The TruncM is shown for K_p from 4 to 18, while the BAM multiplier is depicted for horizontally and vertically truncated from 1 to 8 bits. The truncation for truncated array multiplier and truncated Wallace multiplier is from 2 bits to 8 bits of the input data. The AM1, AM2 are shown for 12-bit to 16-bit approximation of their LSBs, while the TAM1 and TAM2 are shown for 12 to 14 bits. The EvoApprox multipliers have been taken directly from the library and they are the pareto optimal selection of their multipliers. Finally, the stochastic multipliers are the exact

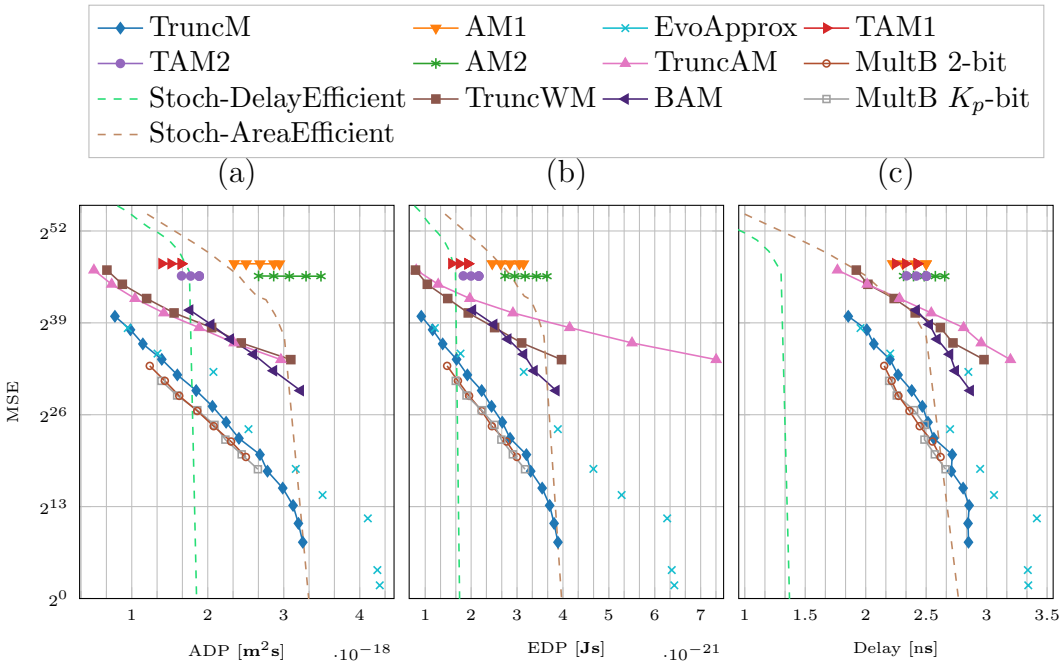


Fig. 5.12: Comparison of MultB with existing approximate multipliers for Mean Squared Error (MSE) vs. (a)Area-Delay Product (ADP), (b)Energy-Delay Product (EDP), (c)Delay.

mutipliers synthesized for a relaxed timing constraint (Stoch-AreaEfficient) and a relatively stringent timing constraint (Stoch-DelayEfficient).

As discussed in Chapter 3, the comparisons in the literature lack the baseline design of truncated multiplier. Considering the TruncM as the baseline for the comparison, Fig. 5.11 compares the proposed multipliers with the state-of-the-art for mean absolute error. As can be seen in the figure, MultB for both K_p-bit correction and 2-bit correction outperform the existing architectures. It is observed that, the TAM2 for the same EDP and ADP, has a considerably higher MAE in comarison with MultB. The multipliers AM1, AM2, TAM1, and TAM2 are listed in [107, 113] as the best existing approximate multipliers. In Fig. 5.12 the comparison of the MultB with the existing approximate adders is illustrated for the mean squared error. Irrespective of TruncM, the automatic generated multipliers of EvoApprox show the best trade-off, and still different configurations of MultB outperform the EvoApprox multipliers. As an instance the MultB with K_p-bit correction when $K_p = 16$ mitigates the MSE by 93% in comparison with the EvoApprox multiplier with a similar EDP; and it reduces the EDP by 47% in comparison with another EvoApprox multiplier with a similar MSE.

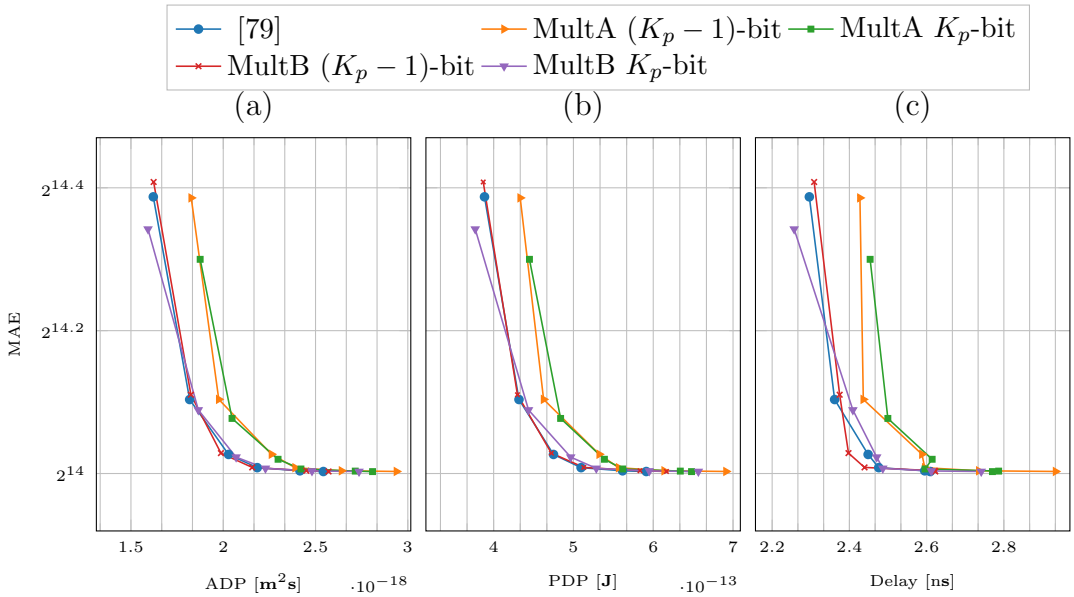


Fig. 5.13: Comparison of the proposed multipliers with the multiplier of [79] for Mean Absolute Error (MAE) versus (a)Area-Delay Product (ADP), (b)Power-Delay Product (PDP), (c)Delay. The K_p-bit and (K_p-1)-bit, following the name of MultA and MultB, indicate the number of bits employed for the correction: K_p-bit correction and (K_P-1)-bit correction, respectively.

In order to evaluate the proposed multipliers when K_o is different form K_p, let us set the $K_o = 16$ and compare them with the fixed-width multiplier proposed in [79]. The comparison in shown in Fig. 5.13 and Fig. 5.14 for mean absolute error and mean squared error, respectively. The multipliers depicted in both figures are synthesized and simulated for K_p from 10 to 15. In the case of MultA and MultB, the Kp-bit correction and (K_p-1)-bit correction are selected for the comparison. It can be seen in the figures that MultB when corrected with (K_p-1)-bit has almost the same behaviour as multiplier of [79]. MultB with K_p-bit correction outperforms the multiplier of [79] when $K_p = 15$ and for the other K_ps it has lower error and slightly higher PDP and ADP. MultA compared with the fixed-width multiplier of [79] has higher ADP and PDP, while it reduces the MAE and MSE. Obviously, by decreasing m (the number of bits employed for the correction), the energy efficiency can be improved with the cost of more error. It is required mentioning that the multiplier of [79] is a promising design which probably because of being limited to fixed-width has not gotten deserved attention from the researchers during the last years.

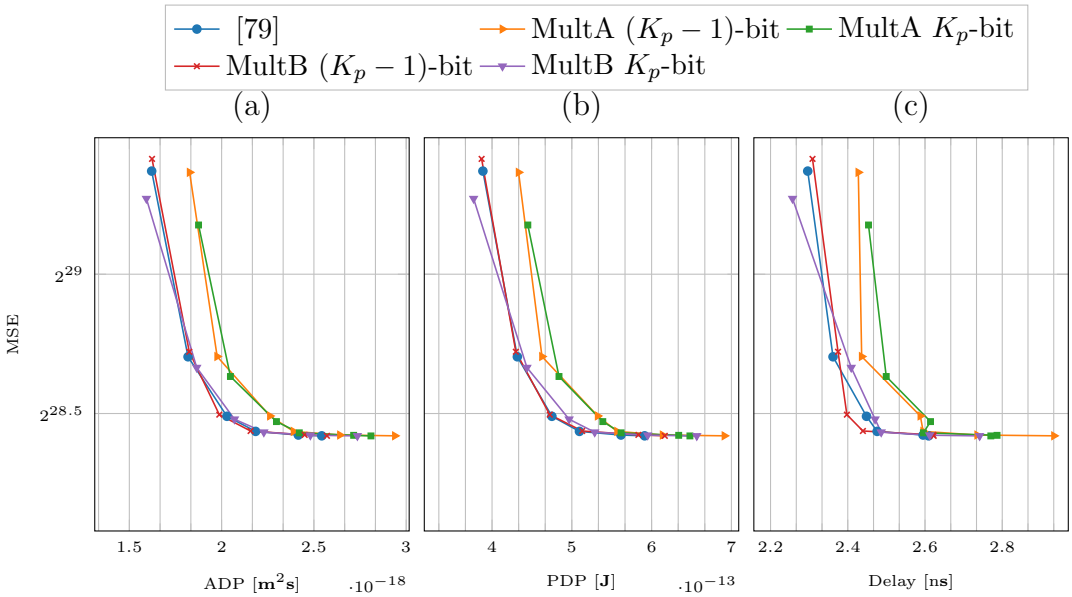


Fig. 5.14: Comparison of the proposed multipliers with the multiplier of [79] for Mean Squared Error (MSE) versus (a)Area-Delay Product (ADP), (b)Power-Delay Product (PDP), (c)Delay. The K_p-bit and (K_p-1)-bit, following the name of MultA and MultB, indicate the number of bits employed for the correction: K_p-bit correction and (K_P-1)-bit correction, respectively.

5.7 Conclusion

In this chapter, a data-dependent correction strategy for truncated multipliers has been proposed. Based on this methodology, two multipliers have been developed. Using experimental results, it has been shown that the proposed multipliers outperform the existing approximate multipliers. The proposed multipliers mitigate the error by up to 85% in comparison with the baseline design (i.e. truncated multiplier) for similar hardware cost. Furthermore, for the similar error, the proposed multipliers reduce the ADP and PDP up to 18%. In addition, a template for truncated multipliers have been presented in this chapter. The template is proposed based on the fact that by reducing the bit-width of the output of the multiplier, the overall system performance is improved. As a result, K_p and K_o have been defined for partial product truncation and output truncation of the template multiplier. By choosing the K_o equal to bit-width of the input data, a fixed-width multiplier is configured. Correspondingly, the proposed multipliers have also been compared with the fixed-width multiplier of [79]. It has been shown that some configurations of the

template can result in less error as well as lower hardware cost in comparison of the multiplier of [79].

CHAPTER 6

Error tolerant applications - case studies

6.1 Introduction

After the quantitative evaluation of different approximate and stochastic units, in this chapter, we study the relevance of the approximate computing units in modern applications. The goal of this chapter is to show the applicability of the approximate computing units in real world applications, rather than just to evaluate the metrics of the proposed approximate and stochastic architectures. In order to fulfill this purpose, we study different case studies including a Sobel filter on an approximate Single Instruction, Multiple Data (SIMD) coprocessor, a motion estimation, combination of task allocation and approximate computing for Internet of Things (IoT), and employing approximate computing units for industrial wireless communication systems.

In the next section, we evaluate the results of a well-known edge detection algorithm in an approximate SIMD coprocessor, where the hybrid adder archi-

tectures selected by the framework proposed in Chapter 4 are employed for the approximation. The evaluation results presented in this section are the outcome of a coopertaion with IMS group from university of Hannover, under a *German Research Foundation* (DFG) project entitled "Quantification of the Trade-off between Energy and Exactness in Computer Vision Processor Architectures Enhanced with Stochastic Computing Mechanisms". The evaluation results presented in this section have been published in an IEEE journal [49].

In section 6.3, a motion estimation algorithm (Egomotion [114]) is used to evaluate the impact of combination of stochastic communication and approximate computing techniques. The aim of this study is to understand the error of the whole system due to the combination of the techniques is additive or there are errors overlap which increase the potentials for even more gains in terms of performance and energy efficiency. This work is also the result of the collaboration with IMS institute from university of Hannover, under the above-mentioned DFG project.

In section 6.4, the applicability of approximate computing in IoT applications is studied by combining task allocation and approximate computing. The main goal is to show what are the gains of considering approximation from the very beginning in the design flow. The results of this collaborative work carried out in ITEM.ids group of university of Bremen, have been published in [52, 54].

Finally, in section 6.5, the impact of approximate computing on an industrial wireless communication system is evaluated. The main aim of this work is to study the relation of the errors of a single approximate unit and the whole system's accuracy. The evaluation results presented in this section are the outcome of a collaboration with ITEM.me group of university of Bremen. The evaluation results presented in section 6.5 have been partially published in [55].

In the following, each of these case studies are detailed and the impacts of applying approximation on the results are discussed.

6.2 Approximate SIMD Coprocessor: Sobel filter

Among feature detection algorithms, edge detection is an image processing technique to find boundaries of objects within an image. Edge detection finds sharp changes in intensity or color in an image, where a high value indicates a steep change and a low value corresponds to a trivial change. A well-known technique for edge detection is a Sobel Operator. The Sobel operator calculates the first derivative of an image by performing a convolution between an input image and two kernels. Indeed, two kernels and the convolution operation are used to detect the edges in the image. One kernel to approximate intensity change in the x-direction (horizontal) and another kernel to approximate intensity change at a pixel in the y-direction (vertical).

In this subsection, a case study using different approximate adders in the ALU

of a single instruction multiple data (SIMD) coprocessor for a microprocessor without interlocked pipelined stages (MIPS) CPU is presented. The goal of the case study is to assess the applicability of the proposed design framework in Chapter 4 in more complex scenarios and not to evaluate the gains achieved using the approximate adders. Clearly, the gains are not considerable here since we are changing only one adder in the ALU of a SIMD processor. The performance enhancement as well as circuit area and power consumption reduction are evaluated, while executing a Sobel image filtering application. The Sobel filtering has been chosen as a relatively simple application which uses addition in its algorithm. In addition, Sobel algorithm is one of the typical applications used for approximate computing bechmarking listed in [115].

Processor: Approximate SIMD Coprocessor

The underlying processor substructure for this case study is depicted in Fig. 6.1. To decouple the performance of the SIMD coprocessor from the main MIPS CPU, the coprocessor resides in an own clock domain. The main MIPS CPU is used for handling the application control flow, issuing vector instructions over a clock-domain-crossing FIFO and initiating DMA transfers between the system main memory and the coprocessor's local memory, which also crosses the clock domain boundary.

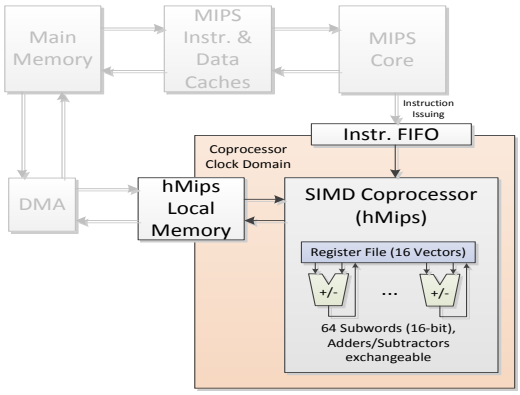


Fig. 6.1: SIMD coprocessor framework. Components that are not considered in the case study are grayed out.

For a meaningful evaluation, the coprocessor contains only the necessary hardware resources required by the executed Sobel filtering application, i.e., a register file containing 16 registers with 64 16-bit element subwords each. Furthermore, each subword ALU is reduced to an exchangeable and configurable approximate adder/subtractor and fixed element subword permutation multiplexers required to calculate the Sobel matrix convolution. The critical path and thus the performance limitation is located in the approximate adder/subtractor unit.

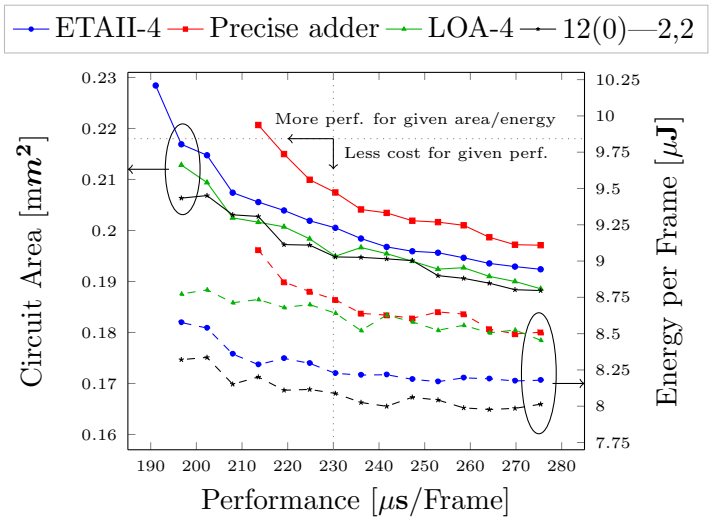


Fig. 6.2: Comparison of performance, circuit area and energy per frame of the SIMD coprocessor equipped with different approximate adder entities and parameterizations. The solid lines denote the circuit area, the dashed lines denote the energy per frame.

Evaluation Flow

For evaluation, the coprocessor ALUs are implemented with different approximate adder entities, including those identified by our framework. The SIMD coprocessor is synthesized to ASIC gate-level netlists for coprocessor clock periods between 3.4 and 4.9 ns, using a TSMC 40 nm low-power standard cell library. After the circuit area is obtained from synthesis, gate-level netlist-based timing simulations using annotated parasitics are performed to obtain the switching activity in the coprocessor for power analysis.

The remaining main MIPS CPU system is simulated functionally and clocked fast enough to ensure that the coprocessor's performance is not limited by the instruction issuing. Moreover, all necessary DMA image transfers are performed prior to all switching activity and performance measurements. Therefore, the pure computational performance of the coprocessor is considered, being only a function of the clock period. For the Sobel filtering application, the performance ranges between 275 us per 512×512 pixels grayscale image frame for a clock period of 4.9 ns and 190 us per frame for 3.4 ns. The power consumption of the coprocessor is converted to an energy budget per image frame using these performance values.

Evaluation Results

In order to evaluate the adder architectures inside the coprocessor, first we compare a precise adder, an OLOCA architecture [47], and two conventional

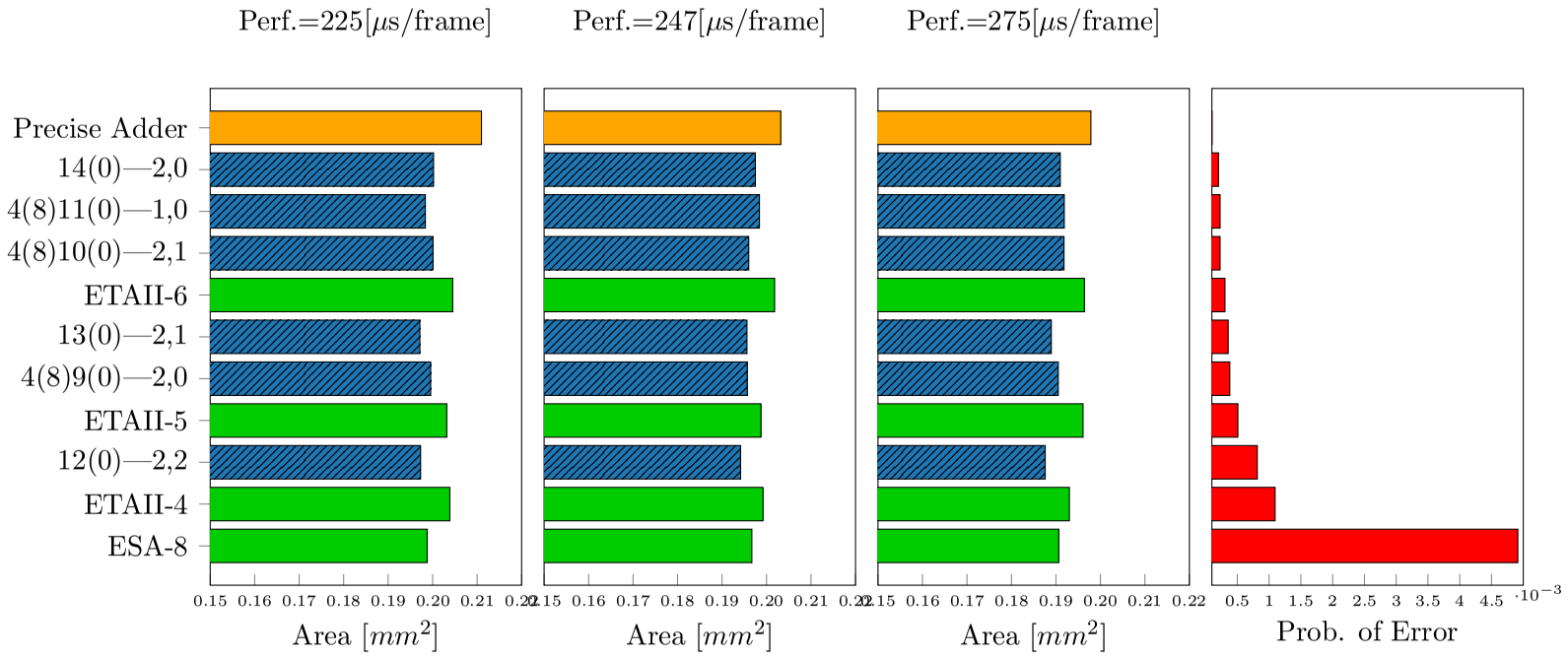


Fig. 6.3: Comparison of accuracy, and circuit area of the SIMD coprocessor implemented with different approximate adder entities and parameterizations. The circuit area of the coprocessor are compared for different adder architecture for three different frequencies: the yellow bar is the precise adder, the green bars are conventional approximate adders, and the hatched blue bars are the hybrid architectures selected by the framework discussed in section 4.3.

approximate adders for all the frequencies. This way, the figure is readable because of choosing limited number of adder architectures. In a separate figure, using bar graphs, we compare more adder architectures for their error values as well as their silicon area for three different frequencies.

Fig. 6.2 compares the total coprocessor circuit area and total energy per frame as a function of the performance for selected adder configurations. This figure compares a precise adder, a selected adder from our framework, and two classical approximate adders for the performance ranges between 275 us per frame and 190 us per frame. The arrows indicate how to find area- and energy-efficient configurations for a desired performance value or more performant implementations for given area and energy budget constraints. It can be seen in the figure that the OLOCA architecture developed using our template outperforms the existing approximate architectures, from both circuit area and energy consumption points of view.

The reference precise adder configuration reaches a maximum performance of 215 us per frame. By inserting approximate adders, the performance can be boosted by up to 12%. At a fixed performance, the evaluated approximate adders can reduce the coprocessor circuit area by up to 10% and the energy dissipation by up to 15%. Given the fact that the precise ALU occupies 12% of the total coprocessor area and consumes 7% of the total power, the significant reduction in hardware cost becomes clear. Especially, the energy dissipation in non-ALU parts of the coprocessor is reduced as well.

Fig. 6.3 depicts the comparison of the total coprocessor circuit area for three different frequencies as well as the probability of the error of adder architectures. More adder architectures are compared in this figure: the precise adder, conventional approximate adders and hybrid adders selected by the framework. The probability of the error of the adders are shown for a chosen binarization which corresponds to the threshold ($\tau = 16$) of our new metric. As can be seen in the figure, considering the error-area trade-off for all the three frequencies, the selected architectures by the framework outperform the conventional approximate architectures for the selected application.

6.3 Motion Estimation: Combining Stochastic Communication and Approximate Computation

A less-than-optimal or an approximate result is sufficient for many application domains such as computer vision and machine learning. The combined approximation across all the subsystems has considerable potential. In [116], the full system approximation and its advantages are compared with a single subsystem approximation. It presents the trade-off between the quality and energy efficiency in the context of the overall embedded computing for an example of a smart camera system. Even though they demonstrate practically the potential energy saving of the full-system approximation, they have not analyzed the interaction of subsystems and the resulting error distributions which is important for decision-making in the architectural level.

In this section, a combined approximation of the computational and communication subsystems is studied. First, the impact of two system components on the error in a single system is analyzed, where we show that the error due to the combined approximation cannot be modeled using simple models and the principle of superposition. The approach is studied on an Egomotion estimation which is widely used for autonomous driving applications.

The research works in the field of approximate communication can be roughly categorized into the lossy compression/decompression [117], dual-voltage approximation [118], and approximate encoding techniques [53]. All these techniques leverage the intrinsic resilience in computations to improve communication performance and reduce the energy consumed by interconnects.

One of the promising approximate communication techniques is the memoryless encoding. An example of this approach is presented in [53]. It uses a combined integer-value coding (CIV) technique which is the combination of *Swap-Coding* and *Inversion-Coding* to reduce the integer value deviation of the received values while signals are transmitted stochastically through the interconnect.

The *Swap-Coding* scheme, which is developed for uniform distributed signals, leverages the physical properties of CMOS links and changes the assignment of the signals to the wires so that the integer-value deviation is reduced. A careful swap of the signals to the wires can reduce errors. On the other hand, there are many real-world applications such as image/video processing, wireless sensor networks and voice processing in which correlated signals are transmitted and processed. *Inversion-Coding* leverages the value locality of the real-world applications to exchange the probability of the worst-case and best-case transitions. The combination of these two coding techniques form the CIV coding technique.

In this study, we transmit data through the interconnect stochastically with voltage and frequency over-scaling. To examine our approach, we use an approximate *CIV* coding technique along with *Conventional* transmission of signals with no coding. Besides, in order to approximate the computing subsystem, we employ approximate adders, where our goal is to analyze the impact of different error philosophies (see Fig. 4.7). Correspondingly, the OLOCA adder from *small errors* philosophy, the ETAII adder from *errors infrequently*, and the Hybrid adder, which combines the two approaches, are employed for our case study.

Egomotion estimation

Egomotion estimation is a widely used technique that holds great significance for computer vision applications, robotics, augmented reality and visual simultaneous localization and mapping. In this section, SIFT-based *Egomotion estimation* has been implemented as a reference application for interpreting scenes based on the extraction of distinctive image features.

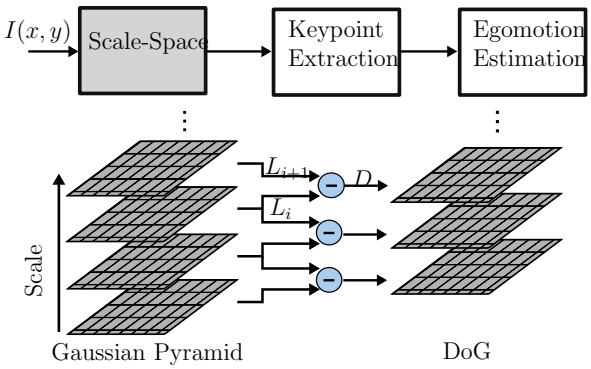


Fig. 6.4: The general flow of our application. The simplified Scale-Space creation step is illustrated which consists of Gaussian pyramid creation and difference of Gaussian (DoG).

SIFT Feature Extraction The purpose of SIFT is to detect distinctive image keypoints and to generate a unique description, which can be used to identify features and objects in images independent of their size and orientation, i.e. scale- and rotation-invariant.

The SIFT algorithm creates a scale space of the original image which is the consecutive Gaussian convolution of the original image with increasing standard deviation and downsampling of the resulting images. The scale-space $L(x, y, \sigma)$ is defined by convolving the Gaussian kernel $G(x, y, \sigma)$ with an input image $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) \circledast I(x, y, \sigma), \tag{6.1}$$

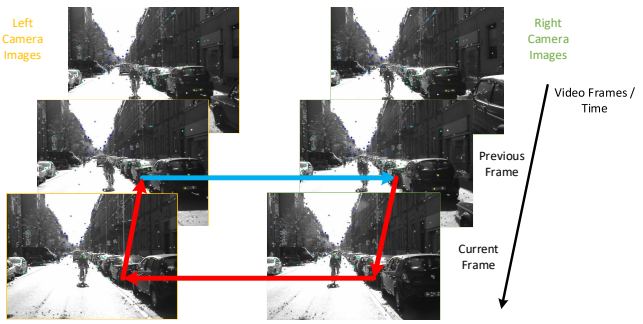


Fig. 6.5: Circular matching of keypoints in a stereo image sequence

where σ is the scale parameter and depends on the total number of pyramid scales, which are organized in octave. The next octave is derived by the downscaling of the Gaussian image. Via a pixel subsampling with a factor of two, the image width and height are halved to obtain higher scales of the pyramid without an excessive increase of the Gaussian kernel size. After the Gaussian pyramid construction, a Difference of Gaussian (DoG) pyramid is formed by subtracting adjacent images within an octave:

$$D(x, y, \sigma_i) = L(x, y, \sigma_{i+1}) - L(x, y, \sigma_i). \tag{6.2}$$

The parameter i is the scale index for the Gaussian-filtered images in one octave.

Fig. 6.4 shows the flow of our application. It provides a simplified Scale-Space creation operation for one scale. L_{i+1} and L_i is used to refer to the Gaussian images $L(x, y, \sigma_{i+1})$ and $L(x, y, \sigma_i)$, respectively, in the rest of this work. Please note that the imprecise adders are implemented in this stage of the application when DoG is derived.

To extract key points, in the next step, the algorithm refines the keypoints' location and assigns orientation to them. This effectively makes the algorithm rotation invariant. Finally, keypoints descriptor as a final representation of the image is generated. Using this representation the features can be identified and keypoints of different images can be matched. A detailed description of the SIFT algorithm can be found in [119].

Egomotion Estimation Egomotion estimation provides a method to reconstruct the 3D scene and measure the movement of a stereo camera system from a sequence of images [120]. This algorithm is based on keypoint and feature matching. A basic keypoint matching method and circular image frame matching to find stable keypoints is illustrated in the following.

Brute-Force Matching is the basic *keypoint matching* method to find similar

keypoints in different images. The feature descriptors extracted by SIFT from two images are compared one by one. Because the descriptor is rotation-invariant and normalized, further processing is not needed. The distance of two feature indicates the similarity between the two keypoints. Two feature vectors with the smallest distance are accepted as the most similar. To reduce false matchings, a position check is used to filter out impossible keypoint pairs, whose vertical differences are too large to be reasonable for a stereo camera scene.

To reconstruct the 3D scene, a process called *circular matching* is used to trace the same keypoints over time in sequential image frames. First, keypoint matching is made between left and right images, current and previous images. As a result, four pairs of matching are created: left current and right current, left current and left previous, right current and right previous, left previous and right previous frames. The keypoints that exist in all matchings are valid matched keypoints. In order to find such keypoints, a circular matching is created through 4 images (Fig. 6.5). For example, a circular matching starts from a keypoint in the left previous image. The blue line points to the matched keypoint in the right previous image. Then, along the red lines, keypoints in the current image frames are matched. After a complete circular matching process, valid matched keypoints are found if start and end keypoint in the left previous image are the same. The 3D position of valid matched keypoints is computed in each frame by using the horizontal disparity, and the movement of the same keypoints overtime is generated by their positions in different frames, which is used to obtain the egomotion of the stereo camera system and thus, the vertical movement.

Egomotion Estimation Quality Metrics Quantitative quality metrics are inevitable for the evaluation of computer vision algorithms. Although these metrics do not necessarily reflect subjective perception, quantitative measures are required for a quick and automated evaluation of the impact of approximate and stochastic mechanisms on the application compared to a golden reference. In the following, the used metrics for egomotion estimation quality evaluation are introduced, i.e., the *rotation angle mean absolute error*, R_{MAE}, and the *translation velocity mean relative error*, T_{MRE}:

$$R_{MAE} = \frac{1}{N}\sum_{i=1}^{N} |\vec{r_{reference}}(i) - \vec{r_{estimated}}(i)| \tag{6.3}$$

$$T_{MRE} = \frac{1}{N}\sum_{i=1}^{N} |\frac{\vec{v_{reference}}(i) - \vec{v_{estimated}}(i)}{\vec{v_{reference}}(i)}|, \tag{6.4}$$

where N is the number of frames of the evaluated video sequence, $\vec{r} = (r_x, r_y, r_z)$ the angular rotation rate per frame around the vehicle axes, and $\vec{v} = (v_x, v_y, v_z)$ the translation velocity along the vehicle axes. These metrics do not compare accumulated movements, i.e., trajectory endpoints of estimated and reference movements, but an averaged error for the estimated movement parameters per frame of a video sequence. The reason is that an evaluation based on trajectory endpoints can be misleading because it is dependent on the frame where an error occurred. In order to evaluate the correlation between the errors of stochastic communication and approximate computing units, a sequence of images is transmitted using ideal, conventional and CIV-based communication strategies. The resulting images are fed to the feature detection algorithm (i.e. SIFT). Further on, in order to study the effects of computation errors, we approximate the subtractor unit of the DoG stage of the SIFT. It should be noted that the goal here is not to show the best approximate unit, but to illustrate the effects of the combined stochastic communication and approximate computing strategies. In the rest of this section, we refer to the combination of ideal communication and exact computation as *exact-approach*. Besides, L_i and L_{i+1} are the inputs and D is the output of the subtractor unit of the DoG stage, as defined in Fig. 6.4.

Table 6.1: The effects of communication strategies (ideal, conventional and CIV-coding) on the input signal of the DoG stage.

	L_i	Err_{L_i}	PSNR
	Mean , Sigma	Mean , Sigma	
ideal	3.79e7 , 2.04e7	-	-
Conventional	4.77e7 , 1.82e7	-9.71e6 , 2.26e6	52.68
CIV	3.80e7 , 2.04e7	-3.47e4 , 2.26e5	85.46

In order to compare the effects of different communication strategies, Table 6.1 tabulates the distribution of a signal after the Gaussian filtering stage, i.e. on of the input signals of the DoG stage, as well as the error characteristic of the input. It can be seen that using the CIV coding, the error at input L_i, has a 10-time smaller sigma, while its mean value is also considerably smaller than the conventional communication strategy (no coding). The resulting changes in the distribution of L_i is important because the errors due to this distribution can be masked by the selected adder in the next stage of the application. Table 6.1 shows that not only a wise selection of stochastic communication may enhance the approximation results, but also may open up tremendous potentials for joint approximation and correspondingly can exploit the most benefits from

the approximation techniques.

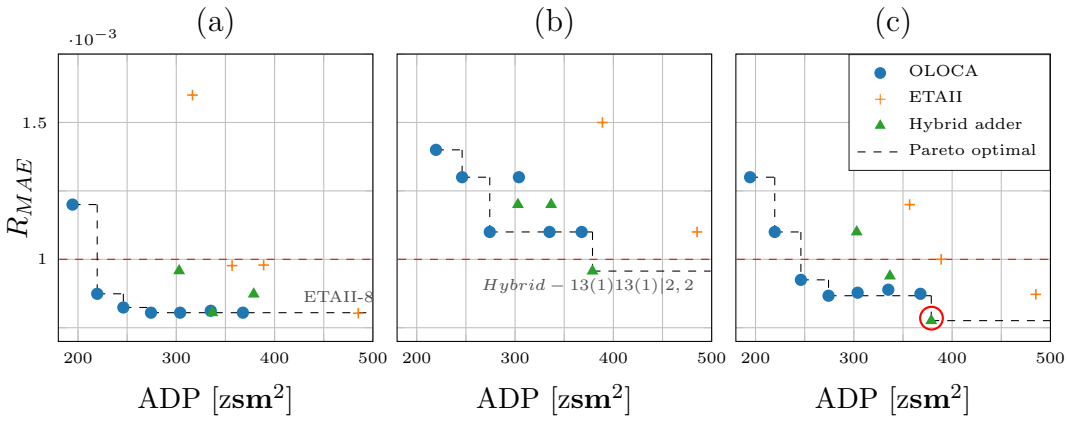


Fig. 6.6: The effect of combining stochastic communication and approximate computation on motion estimation. Rotation angle accuracy (R_{MAE}) versus Area-Delay Product. a) ideal communication, b) conventional communication, and c) CIV coding communication.

Fig. 6.6 compares the effects of different communication strategies on the accuracy of the motion estimation application using different approximate adders. In order to keep the figure readable, only some of the adders are named in the figure. For the OLOCA adders, from left to right in each figure, OLOCA-16 to OLOCA-10 are included. The numbers correspond to the lower significant sub-adder of the OLOCA. The hybrid adders are selected to be optimized for the CIV coding and to be fair in the comparison, they are the same for all the three communication strategies. The sub-adders in each adder architecture are Ripple-Carry adders. As can be seen in this figure, a combination of CIV communication and Hybrid adder (the marked one with a red circle in the figure) has the same quality as an exact-approach. This corresponds to a 20% improvement in the speed of the communication subsystem (from time period of 2.1 ns of the exact to 1.7 ns using stochastic CIV), as well as a 50% improvement in Area-Delay product (ADP) of the adder architecture (from 7.5e-19 sm^2 of exact to 3.7e-19 sm^2 of the Hybrid adder) without any accuracy penalty. As illustrated in this figure, for a tolerable maximum error (the red dashed line for example), there are different combinations of CIV coding and approximate adders which can be selected to improve the energy consumption, performance and silicon area of the system.

Considering the error behavior of ETAII and Hybrid adders, presented in Chapter 4, in order to model the interaction of the errors of these architectures with stochastic communication, the Gaussian mixture model needs to be used.

Table 6.2: The effects of the combination of stochastic communication and approximate computation on the SIFT-based motion estimation. The error characteristics of the output of the DoG stage (D) as well as the Egomotion estimation evaluation.

Communication	Adder	D	R_{MAE}	T_{MRE} (%)	Valid / Invalid Frames	PSNR
ideal	Exact	-	8.05e-4	4.29	153 / 0	-
	ETAII	-1.67e7, 0 , 0.027 -6.55e4, 0 , 0.019 0 , 0 , 0.954	8.03e-4	4.37	153 / 0	63.88
	Hybrid	-3.00e0 , 4.18e0 , 0.75 -3.27e4 , 4.19e0 , 0.25	8.73e-4	4.02	153 / 0	108.37
Conventional (NoCode)	Exact	-4.79e2 , 2.15e5	1.10e-3	5.59	151 / 2	85.99
	ETAII	-1.67e7 , 8.06e4 , 0.029 -6.03e2 , 2.18e5 , 0.971	1.10e-3	5.87	150 / 3	63.44
	Hybrid	-8.66e3 , 2.16e5	9.57e-4	4.94	153 / 0	85.96
CIV Coding	Exact	-1.36e0 , 5.97e4	8.78e-4	4.52	153 / 0	97.14
	ETAII	-1.67e7 , 4.16e4 , 0.027 -1.86e2 , 6.02e4 , 0.973	8.72e-4	4.52	153 / 0	63.91
	Hybrid	-8.17e3 , 6.13e4	7.76e-4	4.02	153 / 0	96.83

Table 6.2 compares the accuracy of these adder architectures considering different communication strategies. An immediate phenomena which can be noted analyzing the information of Table 6.1 and Table 6.2 is that the noise on the output of the DoG stage is smaller than its input noise considering exact computation for both conventional and CIV-based communication. It can be concluded that the errors at output D cannot be described by uncorrelated assumptions. The real errors are much better, and the rationale is the correlation of the input signals. The highly correlated input signals result in masking of some of the errors of the approximate adder architecture. This effect has not been considered conventionally in the analysis of the approximate units.

A careful study of Table 6.2 shows that some of the lobes of the ETAII and hybrid adders are integrated into the neighboring lobes. This phenomenon happens due to the increase in the sigma of some lobes. In addition to the information on the table, Fig. 6.6 illustrates that the selection of an optimized Hybrid adder avoids some of the lobes to appear, resulting in better accuracy.

The combination of approximate computation and stochastic communication in a single system has been studied in this section. It has been shown that due to the error masking, combination of approximations provides an untapped potential to exploit the most benefits from approximate computing. However, because of the highly correlated signals, the interaction of the errors cannot be modeled by conventional simple linear models. It has been observed that the right selection of approximation techniques may provide the same accuracy as the exact approach. For instance, combining a CIV coding and an optimized Hybrid architecture, 20% and about 50% performance improvement can be achieved in communication and computation subsystems, respectively.

6.4 IoT: Combination of Task Allocation and Approximate Computing

The emerging IoT technology can provide promising solutions for numerous present-day society issues, such as intelligent manufacturing, smart cities, remote healthcare, speech or face recognition, real-time online gaming and autonomous driving [121]. The Internet of Things (IoT) devices are typically small and battery-operated. Consequently, energy efficiency is always the primary concern in regards to IoT applications [122, 123]. Two promising techniques at circuit and system levels are approximate computing and energy aware task allocation, respectively, which can be employed to address the energy efficiency for the IoT applications. However, the existing task allocation approaches are designed without considering the aspect of approximate computing. In [52], this gap is filled and the network lifetime subject to the accuracy requirements of the applications is maximized. By considering both the approximate computing and task allocation simultaneously, a non-linear problem is obtained to allocate the tasks for the fog nodes (FN) and IoT end devices(EN), and to select the corresponding execution modes (tasks in approximate or exact modes). To efficiently solve this problem, a centralized Task Allocation algorithm taking the Approximate Computing into account (TAAC) is proposed in [52] by converting the non-linear problem into a linear programming problem. As executing the centralized algorithm is a challenge for the resource limited IoT devices, it is further extended to an optimal distributed algorithm based on Dantzig-Wolfe decomposition to solve the problem of tasks distribution and execution modes selection.

An IoT application is typically composed of a set of dependent computing tasks which can be represented by Directed Acyclic Graph (DAG) as in [124,125]. The problem of task allocation combining approximate computing is modeled as partitioning the DAG into two parts: one part is assigned to ED and the other part will be executed by FN. Meanwhile, ED and FN also need to decide whether to execute each of the assigned tasks in the exact execution mode or approximate mode. Taking Fig. 6.7 for example: ED will execute task v_1 in exact mode and task v_2 in approximate mode; FN will execute task v_3 in approximate mode, task v_4 in exact mode and task v_5 in approximate mode. The objective is to maximize the network lifetime by assigning the tasks and selecting the execution modes for all EDs and FN in each round.

The solution of this problem is out of the scope of this dissertation. The details regarding the centralized and the distributed TAAC algorithms can be found in [52]. Indeed, the goal in this section is to show the applicability of approximate computing in IoT applications. The impact of approximate computing on the energy saving in IoT devices is discussed in this chapter. We

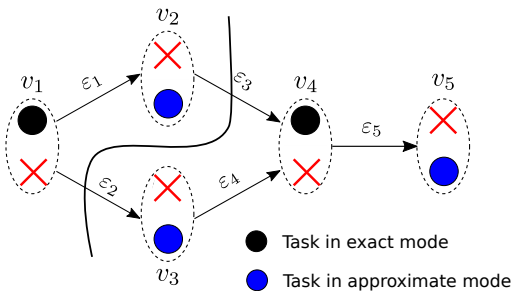


Fig. 6.7: Diagram of task allocation considering approximate computing.

show that if the errors of approximation are considered from the beginning in the design flow and the system is optimized considering those errors, significant improvements can be achieved.

In the following the centralized and distributed TAAC algorithms proposed in [52] are compared with *Weighted scheduling* [126], the *Exhaustive search* and *All-approximate* approaches. *Weighted scheduling* [126] is an optimal task allocation algorithm for cluster based wireless sensor networks. All the tasks in weighted scheduling are executed in the exact modes. *Exhaustive search* runs the *Weighted scheduling* [126] for all possible combinations of the execution modes of tasks and select the best solution among them. *All-approximate* considers that all the tasks are executed in the approximate modes and then runs [126] to get the maximum gain.

The Fog-IoT network is randomly generated in a two-dimensional area of $100\times100\ m^2$ with one FN at the center and n randomly distributed EDs. The parameter values of EDs and FN are taken from the datasheets of Texas Instruments CC2538 [127] and TMS320C5509A [128], respectively. The processing related parameters are $f_i = 32\,MHz$, $f_F = 200\,MHz$, $P_i = 36.9\,mW$, $P_F = 192\,mW$. We assume the EDs and FN have the same RF modules with the bandwidth of 250 $Kbps$. The communication related parameters are $e_{ot} = e_{or} = 3.69\ \mu J$, $t_{tx} = t_{rx} = 4\ \mu s$, $P_{T0} = 59.8\ mW$, $\eta = 0.05$, $P_{rin} = -85\ dBm$, $F_R = 2.4\,GHz$ and $\alpha = 2$. The battery capacities of EDs are randomly generated from 1 to 5 KJ; and the battery energy of FN is randomly generated from 10 to 15 KJ.

The DAG associated with each ED is also randomly generated: the computing workload of each task in exact mode is within the range of $[100, 500]$ KCCs (kilo clock cycles) and the communication data on each edge is in the range of $[100, 1000]$ bits. When the tasks are executed in the approximate modes, the corresponding energy costs are determined by the energy ratio between approximate and exact modes, $r_{a/e}$. We consider the DAGs for all EDs are the same. The error generated by each task in the approximate mode and the corresponding coefficient, σ_k^2 and c_k, are randomly distributed within $[0, 0.01]$

and $[0.5, 1.5]$, respectively. The tasks executed in exact modes do not generate any error and the error limitation for each application is 0.02.

The performance in terms of network lifetime extension and system error variance as well as the algorithm execution time (algorithm computation complexity) in Matlab 2017a are investigated by changing three configuration parameters: a) the energy ratio between approximate and exact modes, $r_{a/e}$; b) the number of EDs, n; c) the number of tasks in each DAG, K_i. Table 6.3 illustrates the configurations of the parameters. Note that only one parameter is changed in each simulation. The reported results in this section correspond to the average values and the standard deviations of 500 test instances for each simulation scenario.

Table 6.3: Configurations of the parameters for the simulations.

Configuring Parameters	Selected values	
	Default	Changing range
Energy ratio, $r_{a/e}$	0.3	0.9, 0.7, 0.5, 0.3, 0.1
Number of EDs, n	10	5,10,15,20,25,30,35,40
Number of tasks, K	10	5, 10, 15, 20

The first set of simulations investigate the impact of energy ratio between executing the tasks in approximate and exact modes, $r_{a/e}$, in terms of network lifetime increase, system output errors by executing tasks in approximate modes and the algorithm runtime of the proposed algorithms. The results of extending the network lifetime are depicted in Fig. 6.8(a). In order to clearly illustrate the superiority of applying approximate computing, we use the normalized network lifetime with respect to [126]. Obviously, both centralized and distributed TAAC algorithms perform the same as *Exhaustive search* and significantly extend the network lifetime comparing with [126]. The gains increase from 1.11 to 8.02 when $r_{a/e}$ changes from 0.9 to 0.1. The is because more energy can be saved by executing the tasks in approximate modes when the $r_{a/e}$ becomes smaller. Although applying the *All-approximate* policy prolongs the network lifetime the longest, the system output errors generated by executing all tasks in approximate modes greatly exceeds the user pre-defined error threshold 0.2, as shown in Fig. 6.8(b). In contrast, both centralized and distributed TAAC algorithms satisfy the error requirements with the Error Tolerance constraint. Fig. 6.8(c) shows the execution time of the proposed algorithms. Clearly, the *Exhaustive search* approach consumes much more time than the others. The algorithm runtime of all approaches keep stable as $r_{a/e}$ decreases. This is due to the fact that the computation complexities of proposed algorithms are only

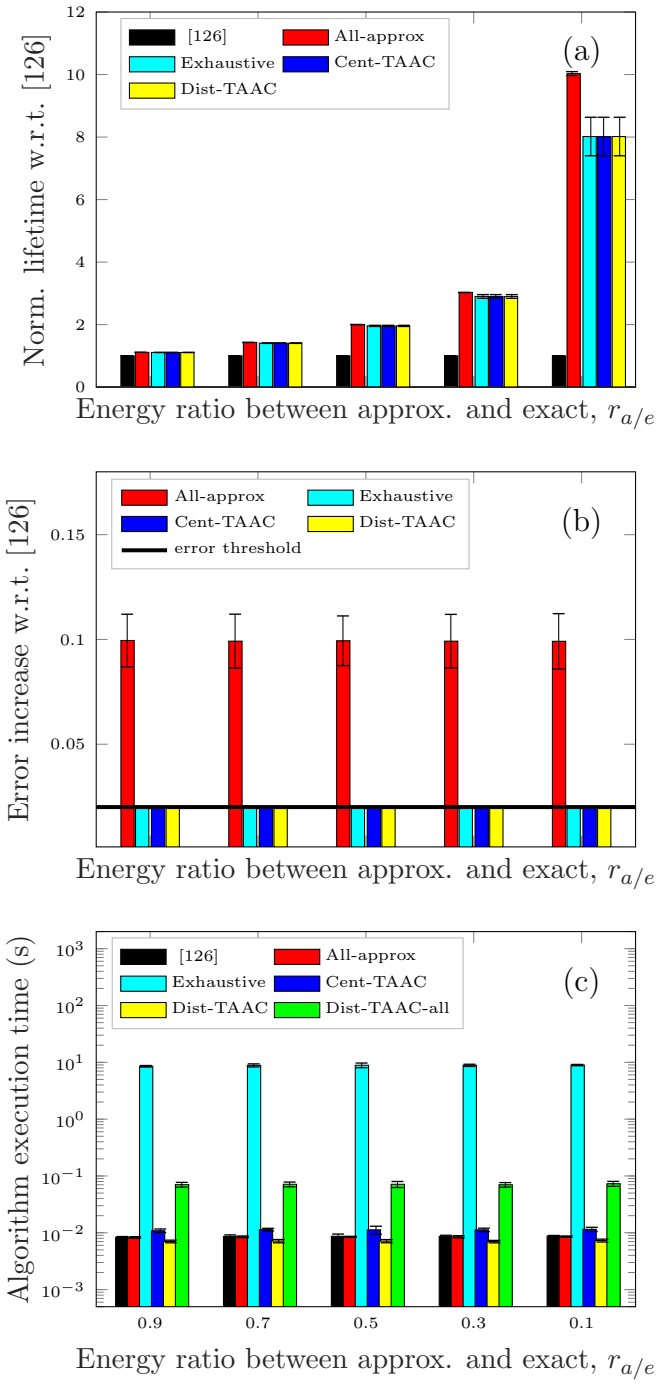


Fig. 6.8: The impact of the energy ratio between executing the tasks in approximate and exact modes on (a) extending the network lifetime, (b) system output quality and (c) execution time of the algorithms (there are 10 EDs and each application contains 10 tasks).

affected by the number of EDs and the number of tasks in each DAG. As distributed TAAC consists of n sub-problems which are executed by each ED in parallel and one linking problem executed on FN, its execution time is actually the maximum runtime among the sub-problems plus the runtime of the linking problem. It can be seen from Fig. 6.8(c) that distributed TAAC requires much smaller execution time than centralized TAAC. For example, when $r_{a/e}$ equals 0.7, distributed TAAC requires in average only 6.9×10^{-3} second while the centralized TAAC needs 11.3×10^{-3} second. In addition, the overall execution time of distributed TAAC (termed as Dist-TAAC-all), i.e., the summation of all n parallel sub-problems, is larger than the centralized TAAC. Therefore, considering the overall algorithm runtime, we conclude that the centralized TAAC can be efficiently solved for the scenarios where all the information can be known in advance. For the other scenarios, distributed TAAC is a better choice.

The second set of simulations investigate the performances of proposed algorithms by changing the number of the EDs, n. In Fig. 6.9(a), both centralized and distributed TAAC algorithms extend the network lifetime as long as the *Exhaustive search* as expected. They extend the lifetime by 2.82 times in average with respect to [126]. The gains of normalized network lifetime slightly change as n increases from 5 to 40. This can be explained by the fact that the energy ratio between executing the tasks in approximate and exact modes does not change, while [126] can provide the optimal task allocation solutions for all tasks in exact modes. With the Error Tolerance constraint, the proposed algorithms can always make the output errors within the user pre-defined threshold as illustrated in Fig. 6.9(b). As the computation complexities of centralized TAAC and the summation of all sub-problems in distributed TAAC are related with n, the corresponding algorithm runtime gradually increases when n changes from 5 to 40 as depicted in Fig. 6.9(c). While the execution time of distributed TAAC is slightly changed, since the complexities of the parallel problem's are mainly related with the number of tasks. Specifically, comparing with the centralized TAAC which requires from 9.8×10^{-3} to 28.4×10^{-3} second as n increases from 5 to 40, the execution time of distributed TAAC only changes in average from 9.0×10^{-3} to 11.4×10^{-3} second. This phenomenon is consistent with the results in Fig. 6.8(c) of the first set of simulations.

We further conduct the third set of simulations by changing the number of tasks in the DAG, K, to evaluate the performances of proposed algorithms. As depicted in Fig. 6.10(a), the proposed centralized and distributed TAAC algorithms achieve dramatic normalized network lifetimes with respect to [126] and provide the same results as the *Exhaustive search*. When K increases from 5 to 20, the gains of both centralized and distributed TAAC algorithms decrease from 2.97 to 2.38 in average. The reason is that the number of

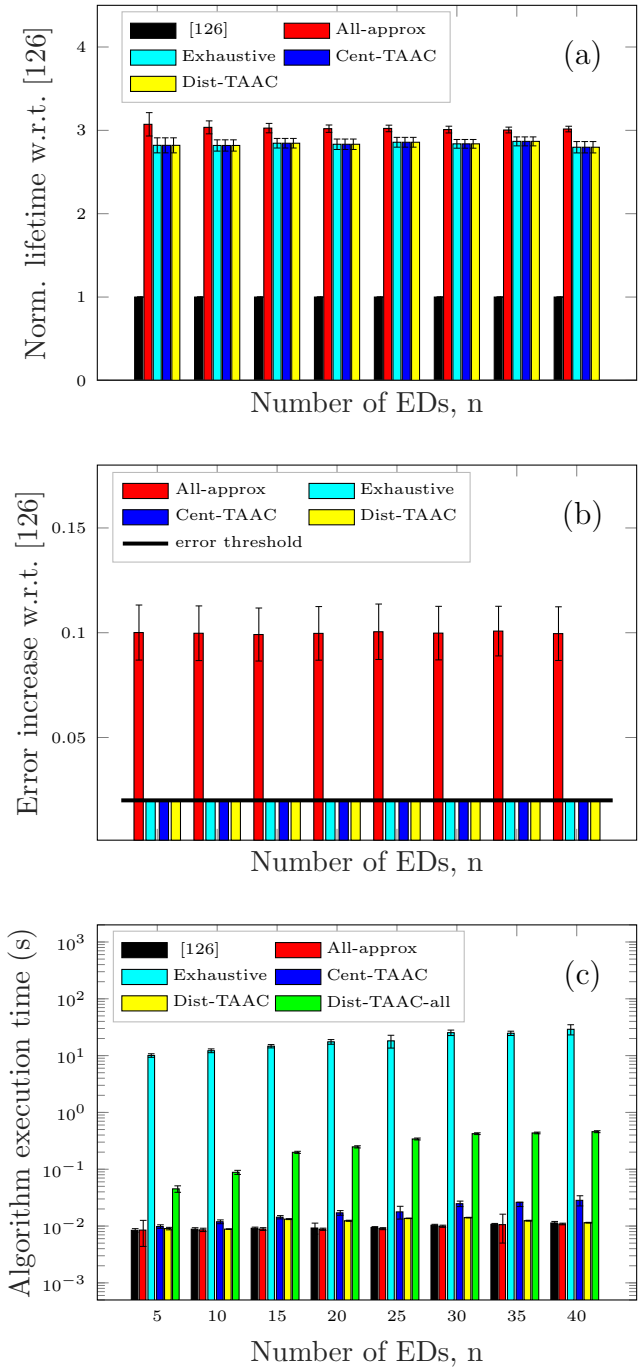


Fig. 6.9: The impact of number of EDs on (a) extending the network lifetime, (b) system output quality and (c) execution time of the algorithms (each application contains 10 tasks, the energy ratio between executing the tasks in approximate and exact modes is 0.3).

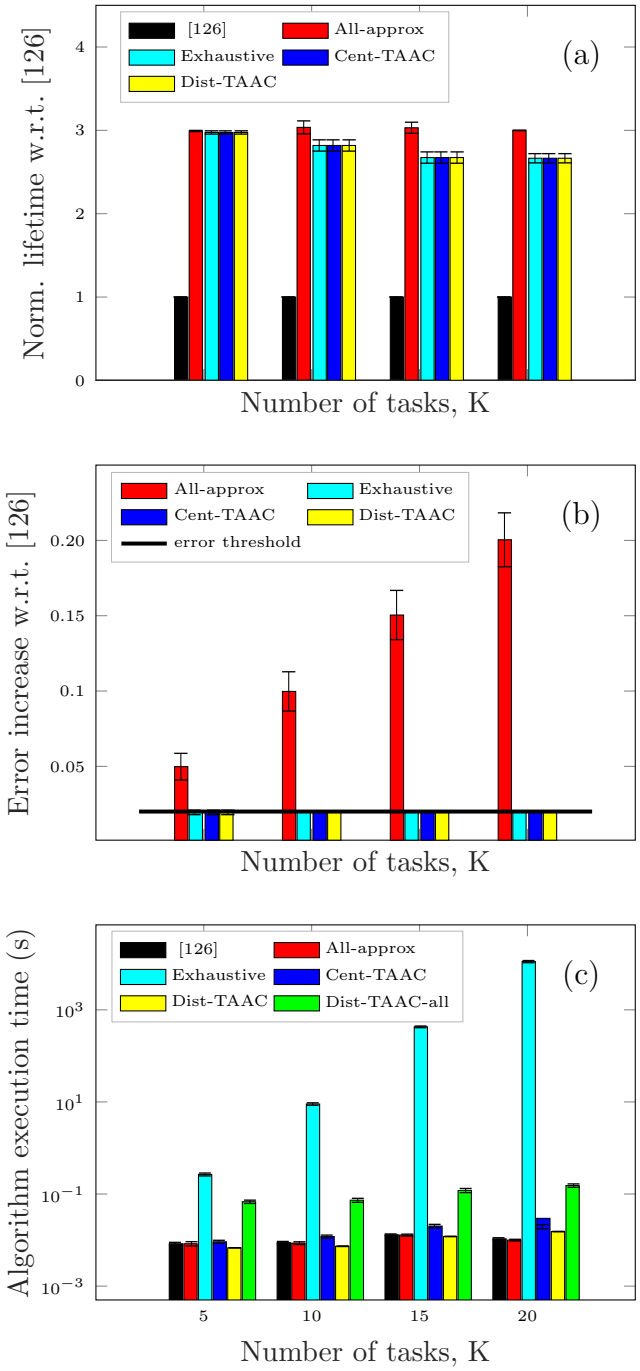


Fig. 6.10: The impact of number of tasks on (a) extending the network lifetime, (b) system output quality and (c) execution time of the algorithms (there are 10 EDs, the energy ratio between executing the tasks in approximate and exact modes is 0.3).

tasks executed in approximate modes is limited, in order to meet the output quality. Consequently, the proportion of saved energy by executing the tasks in approximate modes becomes smaller when the number of tasks increases. As the *All-approximate* policy does not consider the error limits, the normalized network lifetime it achieves does not change too much, which is consistent with the results in Fig. 6.9(a). The drawback of *All-approximate* is that the generated output error by the approximated tasks exponentially increases as shown in Fig. 6.10(b), while both centralized and distributed TAAC algorithms strictly satisfy the requirement of output quality. The results of the impact of K in terms of the execution time of the algorithms are illustrated in Fig. 6.10(c). Obviously, the algorithm runtime of *Exhaustive search* exponentially increases when K changes from 5 to 20. It is due to the fact that *Exhaustive search* considers all of the possible combinations of the execution modes selection of tasks. In contrast, distributed TAAC requires the smallest execution time. As the complexity of each sub-problem is related with the number of tasks, the execution time of distributed TAAC increases from 6.8×10^{-3} to 15.3×10^{-3} second when K changes from 5 to 20.

To sum up, the network lifetime can be further extended by combing approximate computing. When the approximate computing saves more energy cost over executing tasks in exact modes, the more profit can be achieved by the task allocation approaches combining approximate computing.

This work addresses the problem of maximizing the lifetime of fog architecture based IoT networks by applying both the task allocation and approximate computing techniques. The problem of **T**ask **A**llocation combining **A**pproximate **C**omputing (TAAC) is formulated as a non-linear integer optimization problem through partitioning the tasks for all IoT end devices and fog nodes and selecting the corresponding execution modes of all tasks simultaneously. In order to efficiently solve this problem, we convert the non-linear problem to a linear programming problem and propose both a centralized and a distributed TAAC algorithms. By applying the combination of techniques in circuit and system levels, high energy efficiency can be achieved: the proposed algorithms extend the network lifetime by 3 times longer than the previous approaches which only consider task allocation technique. We strongly believe that the future IoT applications can benefit a lot from this cross-level energy optimization combining both circuit and system aspects.

6.5 Industrial Wireless Communication System

Unprecedented challenges to the communication network service are growing along with the rapidly growing wireless communication. Energy efficiency and performance are the main concerns in the wireless communication systems, nowadays, where a huge amount of data need to be transmitted [129]. At the same time, wireless communications always need to compete with quality degradation and noise effects (errors) which undermine the signal characteristics during the transmission. Consequently, several error-resilient components are employed nowadays at the receivers side, e.g. frame and symbol synchronization, frequency offset and channel estimation and compensation steps, forward error correction (FEC), etc. The need for energy efficiency as well as error-resiliency of wireless communication systems meet the criterion where approximate computing can be applied.

A key technology of modern wireless communication with high spectral efficiency is given by general or orthogonal frequency division multiplex approaches, involving the commonly used Fast Fourier Transform (FFT) [130]. Though this has proven its feasibility for many State-of-the-art wireless solutions, the requirements of next generation wireless solutions soon will lead to impractical results when only traditional design approaches are considered. Hence, novel design ideas and strategies must be explored that will lead to an significant increase of efficiency, in particular in terms of energy.

In this section, we explore the impact of approximate computing units within an orthogonal frequency-division multiplexing (OFDM) based industrial wireless communication scenario. Although the benefit of approximate computing units has been essentially proven, neither the relationship between the accuracy of these approximate circuits and the reliability of the baseband wireless system nor the potential of adopting these various approximate units into industrial wireless applications has been so far explored. In particular, when typical industrial applications like close-loop control applications have stringent reliability requirements (e.g. very low frame error rate requirement), this exploration becomes essential. Moreover, conventional evaluation metric for approximate circuits are mostly oriented on single function block. There is a lack of studies exploring the relationship of these evaluation metrics and the accuracy of such complex wireless communication system.

In this case study, the sequential 256-point R2^{2}SDF FFT [132] and the IFFT modules in the baseband signal processing are approximated using approximate adders with different error philosophies, to study the impact of approximate computing a wireless communication system. The system architecture where FFTs and IFFTs are employed in depicted in Fig. 6.11. Each stage of the R2^{2}SDF FFT has one butterfly of either type I or type II, as can be seen in Fig. 6.12 and Fig. 6.13, respectively. More details of the

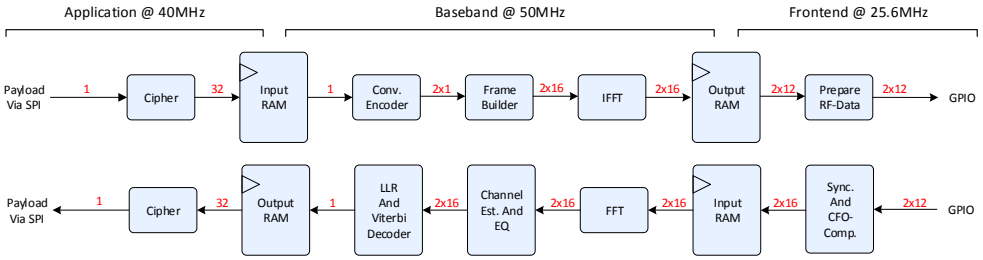


Fig. 6.11: System architecture: Above and below are the TX and RX route. Between the Input RAM and the Output RAM is the sequential signal processing chain. The cipher block with the SPI and the RF controller are operated in separate frequency domains due to the peripheral specification [131].

hardware implementation can be found in [131, 133].

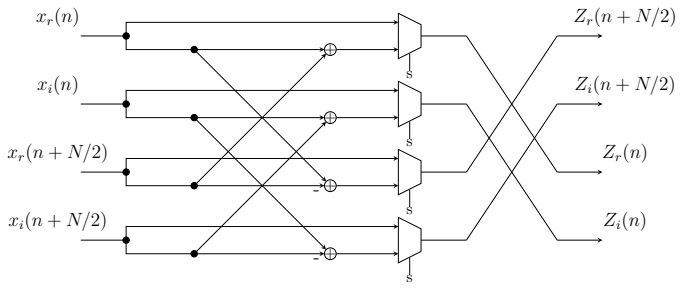


Fig. 6.12: Butterfly type I of R2^{2}SDF

Since in each FFT stage utilizes four adders in its architecture, the 256-point fixed-point pipeline FFT in the system has $4 \cdot log_2(256) = 32$ adder instances. Due to considering overflow of the output carry bit of the addition, 16-bit input of a 256 points FFT, has a 24-bit output. All the four exact adders in each butterfly are replaced by approximate adders. The selected approximate adders are the OLOCA and the truncated adder (TruncZero) from the "small error" philosophy, the ETAII [28] which generates "infrequent errors", and the hybrid approximate adder introduced in Chapter 4. The impact the aforementioned approximate adders on FFT (locally) and on the whole system (globally) is studied. Indeed, the goal is to study the relation between the conventional metric predictions and the actual performance of the approximated system.

In order to evaluate the approximate adder architectures which are employed in the system, individually, the Table 6.4 tabulates the error values of the adders for conventional metrics as well as the saturated error metric discussed in Chapter 3 (i.e. SMSE). As an illustration, here we compare the 17-bit adders,

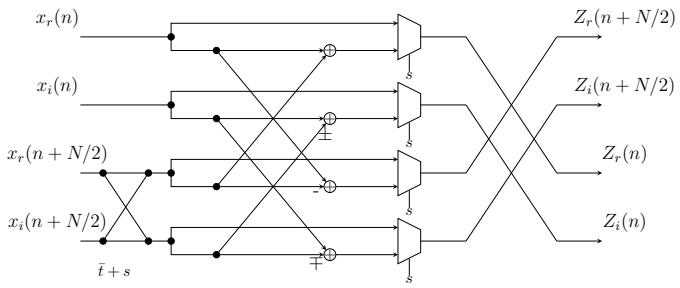


Fig. 6.13: Butterfly type II of R2^{2}SDF

simulated for 10^7 uniform random inputs. The number following the TruncZero and OLOCA names are the number of truncated LSBs. The numbers following the ETAII indicate the bit-widths of the sub-modules of the adder from MSB to LSB, from left to right. In regards to Hybrid adders, the same naming format as of Chapter 4 is used here.

The R2^{2}SDF FFT implementation with different approximate adder configurations have been synthesized and simulated on a Xilinx Artix7-200 FPGA board. In the meantime, software output reference using the same inputs is generated by MATLAB using 32-bit floating point presentation. In this work, 20 groups of 256×16-bit random complex numbers for both weak and strong white Gaussian noise (SNR = 10dB and 40dB) are generated. In reference to the software output, the hardware accuracy is quantified using Normalized Mean Square Error (NMSE). The mean squared error is commonly used as the quality metric for FFT architectures [134].

$$NMSE = 10 \cdot \log_{10} \frac{1}{N} \sum_{i} \frac{(x_i - x_i^{ref})^2}{(x_i^{ref})^2}, \tag{6.5}$$

where N the length of the FFT, x_i the i-th hardware output and x_i^{ref} the i-th software reference output.

The results of local FFT accuracy using the aforementioned approximate adders are tabulated in Table 6.5. The NMSE results (in dB) are shown for the cases when the FFT receives the input with weak noise (SNR = 10dB) as well as strong noise (SNR = 40). Note that in the Table 6.5, the ETAII architectures have fixed bit-widths for LSB sub-blocks, and the bit-width of the most significant block increases from stage 1 to stage 8. In a similar way, the most significant block of the hybrid adders is changing by increasing the bit-width of the adder in each stage. Correspondingly, the x used in the name of the adders indicates variable bit-width.

The FFTs are also integrated to the OFDM receiver/client. The receiver with integrated approximate adders are then synthesized and prototyped on

Table 6.4: Error metrics for the 17-bit AxC adders

Adder	MSE	MAE	MRAE	Err Rate	Avg. Err	$SMSE_{\tau=2^{10}}$
TruncZero-4	267.35	15.00	0.000158	0.9960	-15.00	267.35
TruncZero-8	75829.65	254.74	0.002684	0.9999	-254.74	75829.65
TruncZero-9	304637.08	510.79	0.005375	0.9999	-510.79	304637.08
TruncZero-10	1220953.94	1022.85	0.010730	0.9999	-1022.85	785748.59
TruncZero-11	4890671.53	2047.52	0.021377	0.9999	-2047.52	983075.28
OLOCA-4	26.44	3.69	0.000039	0.8587	-2.99	26.44
OLOCA-8	6810.70	59.91	0.000631	0.9911	-47.88	6810.70
OLOCA-9	27283.48	119.92	0.001261	0.9955	-95.88	27283.48
OLOCA-10	109198.95	239.92	0.002510	0.9978	-191.80	109198.95
OLOCA-11	437261.24	480.26	0.004985	0.9989	-384.11	315711.37
ETAII(11-3-3)	224.38	3.50	0.000037	0.0547	-3.50	224.38
ETAII(7-5-5)	15918.43	15.54	0.000164	0.0151	-15.54	15918.43
ETAII(5-6-6)	129016.79	31.49	0.000331	0.0077	-31.49	8063.54
ETAII(5-4-4-4)	492509.46	127.29	0.001327	0.0586	-127.29	32588.30
ETAII(3-7-7)	1031865.89	62.98	0.000640	0.0038	-62.98	4030.72
Hybrid-7(4)5(0)—2,3	16958.74	23.71	0.000247	0.9303	-22.26	16737.42
Hybrid-6(4)5(0)—2,4	66447.44	46.79	0.000498	0.965	-43.85	16733.13
Hybrid-5(5)6(0)—2,4	130316.05	46.48	0.000488	0.965	-43.52	8493.65
Hybrid-4(6)7(0)—2,4	263805.71	47.02	0.000494	0.964	-44.04	4527.42
Hybrid-5(4)6(0)—2,4	394348.24	110.52	0.001149	0.965	-107.61	24894.13
Hybrid-3(6)7(0)—2,5	1037592.18	92.97	0.000947	0.982	-87.00	5731.87
Hybrid-5(3)6(0)—2,4	924794.61	239.17	0.002494	0.966	-236.35	57838.36
Hybrid-5(2)6(0)—2,4	1981715.14	495.52	0.005152	0.969	-492.89	123486.21

Table 6.5: Post-synthesis Simulation Results of FFT using approximate adders (NMSE in dB) and prototype field test results in 3 Setups (LoS with 0.5m and 7.5m distances; and nLoS with 7.5m distance), values are numbers of lost frames in 2^6 transmissions.

	NMSE		LoS		nLoS
SNR level	10dB	40dB	-	-	-
Distance	-	-	0.5m	7.5m	7.5m
Exact	-84.7275	-84.5612	0	0	0
TruncZero-4	-74,2141	-73.6997	0	0	0
TruncZero-8	-49.4617	-48.7671	0	0	0
TruncZero-9	-43.2173	-42.8701	0	0	364
TruncZero-10	-37.5275	-37.0682	631	1209	2554
TruncZero-11	-31.7743	-31.1915	all	all	all
OLOCA-4	-78.1118	-77.3785	0	0	0
OLOCA-8	-52.8504	-52.5409	0	0	2
OLOCA-9	-46.8423	-46.4592	0	60	138
OLOCA-10	-40.5441	-40.3538	0	16	190
OLOCA-11	-34.7817	-34.3575	44185	37526	38681
ETAII(x-3-3)	-74.8862	-74.5552	0	0	0
ETAII(x-5-5)	-57.0431	-57.1069	0	0	0
ETAII(x-6-6)	-47.6561	-47.9418	0	0	0
ETAII(x-4-4-4)	-41.6742	-41.8792	1	629	1093
ETAII(x-7-7)	-38.3164	-40.2401	6368	16291	32259
Hybrid-x(4)5(0)—2,3	-57.0849	-56.1999	0	0	0
Hybrid-x(4)5(0)—2,4	-49.4440	-48.9199	0	0	0
Hybrid-x(5)6(0)—2,4	-46.6376	-46.6263	0	3	8
Hybrid-x(6)7(0)—2,4	-42.3383	-45.6065	0	3	16
Hybrid-x(4)6(0)—2,4	-42.0923	-42.4949	0	0	0
Hybrid-x(6)7(0)—2,5	-40.3034	-39.4716	6896	9429	13025
Hybrid-x(3)6(0)—2,4	-38.5278	-38.4067	107	571	722
Hybrid-x(2)6(0)—2,4	-34.8894	-34.5372	63210	64601	all

the same FPGA boards. In order to explore the system performance with more realistic conditions, this FPGA receiver prototype has been used together with a transmitter (controller) to perform real field tests. In the initial laboratory tests, three different setups are used: line of sight (LoS) with 0.5 meter, LoS with 7.5 meter and none LoS with 7.5 meter. In the field tests, task of the

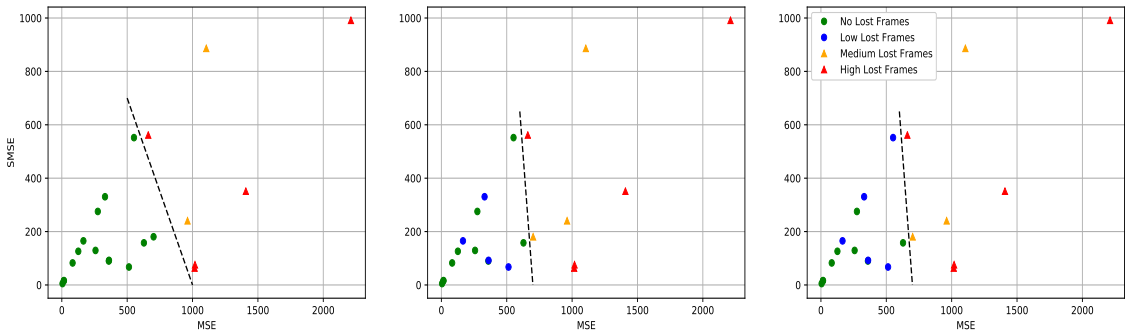
controller is to encode, modulate and transmit the random inputs, where 5×2^{16} input frames are fed to the controller and the corresponding Frame Error Rate (FER) is measured at the client side. Table 6.5 shows the number of the faulty received frame (frame errors) in every 2^{16} transmissions.

The initial observation from Table 6.5 is that employing approximate computing units significant energy efficiency can be achieved, while it even may result in no frame loss. As a distinct instance, the Hybrid adder (x(4)6(0)—2,4) in comparison with an exact adder improves the silicon area and its maximum delay corresponds to a 16-bit adder which is obviously faster than a 24-bit adder (i.e. the exact counterpart), while it results in no frame loss. The analysis of Table 6.5 shows that the selection of approximate units for each scenario (i.e LoS 0.5m, LoS 7.5m, or nLoS 7.5m) is different. For example, TruncZero-9 which works well for LoS, does not work for nLoS; ETAII(x-4-4-4) works promising for LoS 0.5m but makes a huge error in the other cases.

In addition, in some cases small changes in the adder architecture already result in an abrupt degradation of the output quality. As an instance in LoS scenarios, TruncZero-9 works without any frame loss. However, for TruncZero-10, the number of lost frames increases abruptly. Hence, a careful selection of the approximate units is vital.

Comparing the results from Table 6.5, it can be seen that OLOCA adders perform better than the conventional truncation. For example, the OLOCA-10 in the nLoS scenario results in 190 lost frames in comparison with 2554 lost frames for TruncZero-10, which represents a 92% reduction in the frame loss. In a similar way, the hybrid architectures outperform the ETAII adders. For instance, the corresponding architecture for ETAII(x-5-5) is Hybrid-x(4)5(0)—2,3; and the corresponding adder for ETAII(x-6-6) is Hybrid-x(5)6(0)—2,4. In both of the cases, the output quality is almost the same, while the hybrid adders are faster and smaller. The best ETAII architecture which results in no frame loss is ETAII(x-6-6) and the best hybrid adder which results in no frame loss is Hybrid-x(4)6(0)—2,4 as can be seen in the table. Considering the maximum bit-width (24-bit), the hybrid adder is faster and has a delay corresponding to 16-bit adder in comparison with ETAII(x-6-6) which has a delay of 18-bit. Moreover, due to the LSBs truncation, the hybrid adder has smaller silicon area.

As mentioned in Chapter 3, the conventional metrics might be misleading. In fact, none of the conventional metrics can predict the output quality (number of lost frames) shown in Table 6.5 with precision. For example, considering the error metrics of the adders tabulated in Table 6.4, the MAE of Hybrid-5(4)6(0)—2,4 is higher than ETAII(3-7-7). However, based on the results of Table 6.5, this hybrid adder results in no frame loss. Considering the MAE, the Hybrid adder might be discarded which is obviously a wrong prediction. In summary, there is a non-linear relation and even an uncorrelation between the system reliability metrics and the conventional metrics used for the design



(a) (b) (c)

Fig. 6.14: The square root of Saturated Mean Squared Error (SMSE) versus the square root of the Mean Squared Error (MSE), of the adopted approximate adders. The dashed line differentiate between the adder architectures which are producing acceptable and non-acceptable results based of the FER, for 3 different scenarios: (a) LOS 0.5m, (b) LOS 7.5m, and NLOS 7.5m. The architectures are divided into four groups depending on the number of lost frames.

of approximate units. In addition, comparing the NMSE of simulations in Table 6.5 and the field results in the right hand side of Table 6.5, it can be seen that even using conventional error metrics at the local algorithm level (FFT) cannot predict the final quality of the system precisely. For example, considering Table 6.5, Hybrid-x(6)7(0)—2,5 architecture should perform better than Hybrid-x(3)6(0)—2,4 adder due to the better NMSE metric, while considering the field test results given in Table 6.5, it results in considerably higher lost frames. Similarly, ETAII(x-4-4-4) has better NMSE than OLOCA-10, but when used in the system performs worse that the OLOCA counterpart. Moreover, Table 6.5 indicates a notable fact that the baseband signal processing system has good resilience so that its accuracy only begins to decrease when the approximation is fierce. This shows the big optimization potential of the hardware utilization even though the reliability constraints of industrial application is strict. To this end, conventional error metrics can be helpful to qualitatively understand the intrinsic characteristics of different adders but it is not directly related to the system reliability.

To palliate the aforementioned problem, let us take the combination of a conventional metric (MSE) and the SMSE metric (see Chapter 3 for more details) into consideration. We divide the results tabulated in Table 6.5 into four groups: 1) No Frame loss, 2) Low Frame Loss, 3) Medium Frame Loss, and 4) High Frame Loss as a result of employing different approximate adders. The

first and the fourth groups are acceptable and non-acceptable architectures, respectively; while the second and the third groups can be included in either acceptable or non-acceptable division based on the application constraints. As depicted in Fig. 6.14, combining MSE and SMSE, a line can be found to differentiate between acceptable and non-acceptable architectures. However, the quality of the architectures is still impossible to be quantified using this technique. The exploration of novel error metrics is indispensable and must be considered in future works.

6.6 Conclusion

In this chapter, the impact of approximate computing on the whole system's accuracy has been for different case studies. In section 6.2, different approximate adders with different error philosophies have been employed to approximate a SIMD coprocessor for the purpose of Sobel filtering application. The evaluation results have shown that using approximate adders, the performance of the coprocessor has improved by up to 12% in comparison with using exact adders. In addition, for a given performance, the evaluated approximate adders reduce the coprocessor silicon area by upto 10% and the energy consumption by upto 15%. These results, once more, prove the potentials of approximate computing when applied to the error resilience applications. In section 6.3, the combination of approximate computation and stochastic communication has been studied. It has been shown that due to the error masking, even more gains can be achieved by combining different approximation techniques. It has been shown that, these untapped potentials with right selection of approximation techniques may provide the same accuracy as the case that only one technique is used. At the same time, more energy efficiency and performance are gained by the combination of techniques. In section 6.4, the combination of techniques in circuit and system level for achieving higher energy efficiency has been presented. It has been shown that combining approximate computing and task allocation results in 3 times longer network lifetime than the previous approaches which only consider task allocation technique. Observing the results of this section, it can be concluded that there is an enormous untapped potential in the combination of different energy optimization techniques including approximate and stochastic computing. Finally, in section 6.5, the impact of approximate computing on an industrial wireless communication system has been studied. By analyzing the error behaviour of single approximate computing units, FFTs, and the whole system, the relation of error between a single component and the whole system has been explored. The evaluation results conclude that the exploration of more robust metrics is essential for the correct selection of approximate units for a target application.

CHAPTER 7

Conclusion and Future Work

In this work, we have presented a systematic analysis, a design methodology and a set of optimized hardware architectures to use approximate and stochastic techniques to improve the performance and the energy efficiency of the most fundamental arithmetic units, addition and multiplication.

In Chapter 3, a new error metric has been introduced to address the insufficiency of the existing metrics. The new parameterizable metric captures the requirements of the approximate and stochastic applications. It has been shown that, the parameter τ in the proposed metric (i.e. SMSE) can be considered as a knob to model the errors more accurately in real applications. Moreover, a fair comparison for approximate adders as well as a fair comparison of approximate multipliers are presented in this dissertation. The fairness has been ensured by considering various figures of merit along with different accuracy metrics. The fair comparison of approximate computing units has shown outstanding performance of LOA among the approximate adders, and the conventional truncated multiplier (TruncM) among the approximate multipliers. Those architectures have been disregarded in many of the research works.

Previous works in the approximate computing domain show limited improvements due to non-systematic methodologies. In this dissertation, however, with methodical analyses and systematic approaches, four other contributions are presented.

In Chapter 4, an optimal approximate adder, through generalizing an architectural template for approximate adders, has been proposed. The proposed adder Optimized Lower-part OR-Constant Adder (OLOCA) shows considerable improvement in both error and hardware-cost metrics in comparison with the previously reported best architectures. The superiority of OLOCA over the

existing approximate adders has been proved presenting mathematical analysis and further using experimental results. As an instance, a 16-bit approximate adder implemented with the OLOCA approach improves the mean squared error by 58% while reducing the area-delay product by 13.8% at the same time, in comparison with an approximate adder implemented with the LOA approach.

Furthermore, a generic template for approximate adders combining the *small-errors* and the *infrequent-errors* philosophies has been proposed. A wide range of approximate adders, along with new hybrid and non-equal segmented adders, can be developed using the proposed template. The accurate mathematical error formulas of the template has been conjointly presented. Consequently, using one compact formula, the error metrics of a wide range of approximate adders can be calculated. Using experimental results, it has been shown that for the scenarios that approximate adders provide considerable benefits, our framework finds the optimal architectures. Based on the experimental results, for relaxed timing constraints, the conventional truncated adders and OLOCA are the superior architectures, and the other existing approximate adders provide very limited benefits. For stringent timing constraints, OLOCA outperforms the other approximate adders when considering a large threshold for the SMSE accuracy metric (corresponding to the conventional MSE metric). For the applications which their acceptable accuracy can be modeled with medium and small thresholds (τ in SMSE), however, hybrid and ETAII architectures are the superior classes of adder architectures, respectively. These results validate our discussions that the internal architectures of the approximate architectures have to be considered as part of the analyses of the approximate architectures, and specially when comparing them.

Using precise stochastic error analyses, the use of a mixed adder with late input MSB arrival time has been proposed further in Chapter 4. Our stochastically tunable Mixed-PR adder shows a gradual decrease in accuracy in contrast to the abrupt accuracy decrease of its conventional counterparts. Moreover, unlike the existing reconfigurable adders which switch between two modes (i.e. fixed approximate and exact modes), the proposed adder can be configured in multiple modes depending on the operating frequency and supply voltage. Furthermore, using over-scaling techniques, the mixed adder does not require any additional logic to be configured to the approximate modes. Using experimental results, the superiority of the proposed adder over its conventional counterparts as well as existing configurable adders has been shown.

As mentioned above, based on the comparison of approximate multipliers, the conventional truncated multipliers offer the best trade-off between accuracy and hardware cost. As a result, in this dissertation, we studied the correction of truncated multipliers. The data-dependent correction methodology has been proposed in the context of a conceptual template for truncated multipliers.

Using experimental results, it has been shown that some configurations of the our corrected approximate multiplier mitigates the MSE by up to 85% compared to a conventional truncated multiplier with the same PDP.

Future Work

The benefits and challenges of employing approximate computing in modern applications have been discussed in Chapter 6. Accordingly, the following issues are still open to be explored.

1. The accuracy metrics which can precisely relate the errors of approximate computing units to the accuracy of the target application need to be extensively explored. Indeed, the impact of data distribution should be considered in the error metrics.

2. Due to the need for quality configurable systems, a systematic design of optimal reconfigurable multipliers will be considered for our future work.

3. The impact of approximate computing on critical applications such as medical applications will be studied.

Bibliography

[1] John Sartori and Rakesh Kumar. Stochastic computing. *Found. Trends Electron. Des. Autom.*, 5(3):153–210, March 2011.

[2] Sparsh Mittal. A survey of techniques for approximate computing. *ACM Comput. Surv.*, 48(4):62:1–62:33, March 2016.

[3] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[4] 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, 2012.

[5] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6–15, 2011.

[6] 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, 2012.

[7] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pages 365–376, 2011.

[8] M. Shafique, S. Garg, J. Henkel, and D. Marculescu. The eda challenges in the dark silicon era. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2014.

[9] J. Rabaey, J. Ammer, B. Otis, F. Burghardt, Y. H. Chee, N. Pletcher, M. Sheets, and H. Qin. Ultra-low-power design. *IEEE Circuits and Devices Magazine*, 22(4):23–29, 2006.

[10] D. Markovic, C. C. Wang, L. P. Alarcon, T. Liu, and J. M. Rabaey. Ultralow-power design in near-threshold region. *Proceedings of the IEEE*, 98(2):237–252, 2010.

[11] 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, 2010.

[12] 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, 2011.

[13] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger. Neural acceleration for general-purpose approximate programs. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 449–460, 2012.

[14] 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, 2013.

[15] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Architecture support for disciplined approximate programming. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, page 301–312, New York, NY, USA, 2012. Association for Computing Machinery.

[16] 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.

[17] J. N. Mitchell. Computer multiplication and division using binary logarithms. *IRE Transactions on Electronic Computers*, EC-11(4):512–517, 1962.

[18] Y. C. Lim. Single-precision multiplier with reduced circuit complexity for signal processing applications. *IEEE Transactions on Computers*, 41(10):1333–1336, 1992.

[19] M. J. Schulte and E. E. Swartzlander. Truncated multiplication with correction constant [for dsp]. In *Proceedings of IEEE Workshop on VLSI Signal Processing*, pages 388–396, 1993.

[20] E. J. King and E. E. Swartzlander. Data-dependent truncation scheme for parallel multipliers. In *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No.97CB36136)*, volume 2, pages 1178–1182 vol.2, 1997.

[21] Shih-Lien Lu. Speeding up processing with approximation circuits. *Computer*, 37(3):67–73, Mar 2004.

[22] Khaing Yin Kyaw, Wang Ling Goh, and Kiat Seng Yeo. Low-power high-speed multiplier for error-tolerant application. In *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, pages 1–4, 2010.

[23] 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.

[24] P. Kulkarni, P. Gupta, and M. Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In *2011 24th Internatioal Conference on VLSI Design*, pages 346–351, 2011.

[25] 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.

[26] Ajay K. Verma, Philip Brisk, and Paolo Ienne. Variable latency speculative addition: A new paradigm for arithmetic circuit design. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '08, pages 1250–1255, 2008.

[27] 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.

[28] Ning 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.

[29] S. Venkataramani, K. Roy, and A. Raghunathan. Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1367–1372, 2013.

[30] 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, 2014.

[31] Z. Vasicek and L. Sekanina. Evolutionary approach to approximate digital circuits design. *IEEE Transactions on Evolutionary Computation*, 19(3):432–444, 2015.

[32] R. Hrbacek, V. Mrazek, and Z. Vasicek. Automatic design of approximate circuits by means of multi-objective evolutionary algorithms. In *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–6, 2016.

[33] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina. Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 258–261, 2017.

[34] D. Esposito, A. G. M. Strollo, and M. Alioto. Low-power approximate mac unit. In *2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pages 81–84, 2017.

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

[36] Z. G. Tasoulas, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel. Weight-oriented approximation for energy-efficient neural network inference accelerators. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(12):4670–4683, 2020.

[37] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori. Designing a processor from the ground up to allow voltage/reliability tradeoffs. In *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pages 1–11, 2010.

[38] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori. Recovery-driven design: A power minimization methodology for error-tolerant processor modules. In *Design Automation Conference*, pages 825–830, 2010.

[39] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori. Slack redistribution for graceful degradation under voltage overscaling. In *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 825–831, 2010.

[40] S. Narayanan, J. Sartori, R. Kumar, and D. L. Jones. Scalable stochastic processors. In *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, pages 335–338, March 2010.

[41] N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones. Stochastic computation. In *Design Automation Conference*, pages 859–864, 2010.

[42] J. Sartori and R. Kumar. Architecting processors to allow voltage/reliability tradeoffs. In *2011 Proceedings of the 14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, pages 115–124, 2011.

[43] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge. Opportunities and challenges for better than worst-case design. In *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005.*, volume 1, pages I/2–I/7 Vol. 1, 2005.

[44] D. Ernst, Nam Sung Kim, S. Das, S. Pant, R. Rao, Toan Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pages 7–18, Dec 2003.

[45] B. Colwell. We may need a new box. *Computer*, 37(3):40–41, 2004.

[46] J. Sartori, J. Sloan, and R. Kumar. Stochastic computing: Embracing errors in architecture and design of processors and applications. In *2011 Proceedings of the 14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, pages 135–144, 2011.

[47] A. Najafi, A. Dalloo, and A. Garcia-Ortiz. Systematic design of an approximate adder: The optimized lower part constant-or adder. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(8):1595–1599, 2018.

[48] A. Najafi, M. Weißbrich, G. P. Vayá, and A. Garcia-Ortiz. A fair comparison of adders in stochastic regime. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–6, 2017.

[49] A. Najafi, M. Weißbrich, G. Payá-Vayá, and A. Garcia-Ortiz. Coherent design of hybrid approximate adders: Unified design framework and metrics. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(4):736–745, 2018.

[50] A. Najafi and A. Garcia-Ortiz. Stochastic mixed-pr: A stochastically-tunable low-error adder. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(10):2144–2148, 2020.

[51] Amir Najafi, Ardalan Najafi, and A. Garcia-Ortiz. Stochastic wave-pipelined on-chip interconnect. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(5):841–845, 2020.

[52] W. Yu, A. Najafi, Y. Huang, and A. Garcia-Ortiz. Combination of task allocation and approximate computing for fog architecture based iot. *IEEE Internet of Things Journal*, pages 1–1, 2020.

[53] Amir Najafi, L. Bamberg, Ardalan Najafi, and A. Garcia-Ortiz. Integer-value encoding for approximate on-chip communication. *IEEE Access*, 7:179220–179234, 2019.

[54] W. Yu, A. Najafi, Y. Nevarez, Y. Huang, and A. Garcia-Ortiz. Taac: Task allocation meets approximate computing for internet of things. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020.

[55] M. Hao, A. Najafi, A. García-Ortiz, L. Karsthof, S. Paul, and J. Rust. Reliability of an industrial wireless communication system using approximate units. In *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 87–90, 2019.

[56] A. Najafi, L. Bamberg, A. Najafi, and A. Garcia-Ortiz. Misalignment-aware delay modeling of narrow on-chip interconnects considering variability. In *2018 7th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4, 2018.

[57] A. Najafi, L. Bamberg, A. Najafi, and A. Garcia-Ortiz. Energy modeling of coupled interconnects including intrinsic misalignment effects. In *2016 26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 262–267, 2016.

[58] Reto Zimmermann. *Binary adder architectures for cell-based VLSI and their synthesis*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, Hartung-Gorre Verlag, 1998.

[59] R. P. Brent and H. T. Kung. A regular layout for parallel adders. *IEEE Transactions on Computers*, C-31(3):260–264, March 1982.

[60] P. M. Kogge and H. S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, C-22(8):786–793, Aug 1973.

[61] J. Sklansky. Conditional-sum addition logic. *IRE Transactions on Electronic Computers*, EC-9(2):226–231, June 1960.

[62] T. Han and D. A. Carlson. Fast area-efficient vlsi adders. In *1987 IEEE 8th Symposium on Computer Arithmetic (ARITH)*, pages 49–56, May 1987.

[63] 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, 2013.

[64] H. Cai, Y. Wang, L. A. B. Naviner, Zhaohao Wang, and W. Zhao. Approximate computing in mos/spintronic non-volatile full-adder. In *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 203–208, 2016.

[65] S. Angizi, Z. He, R. F. DeMara, and D. Fan. Composite spintronic accuracy-configurable adder for low power digital signal processing. In *2017 18th International Symposium on Quality Electronic Design (ISQED)*, pages 391–396, 2017.

[66] T. Zhang, W. Liu, E. McLarnon, M. O'Neill, and F. Lombardi. Design of majority logic (ml) based approximate full adders. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2018.

[67] Tong Liu and Shih-Lien Lu. Performance improvement with circuit-level speculation. In *Proceedings 33rd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO-33 2000*, pages 348–355, 2000.

[68] Andrew B. Kahng and Seokhyeong Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, pages 820–825, New York, NY, USA, 2012. ACM.

[69] X. Yang, Y. Xing, F. Qiao, Q. Wei, and H. Yang. Approximate adder with hybrid prediction and error compensation technique. In *2016 IEEE*

Computer Society Annual Symposium on VLSI (ISVLSI), pages 373–378, 2016.

[70] V. Camus, J. Schlachter, and C. Enz. Energy-efficient inexact speculative adder with high performance and accuracy control. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 45–48, 2015.

[71] 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.

[72] J. Hu and W. Qian. A new approximate adder with low relative error and correct sign calculation. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1449–1454, 2015.

[73] I. C. Lin, Y. M. Yang, and C. 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, Sept 2015.

[74] V. Camus, M. Cacciotti, J. Schlachter, and C. Enz. Design of approximate circuits by fabrication of false timing paths: The carry cut-back adder. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(4):746–757, 2018.

[75] Behrooz Parhami. *Computer arithmetic*, volume 20. Oxford university press, 2010.

[76] C. R. Baugh and B. A. Wooley. A two's complement parallel array multiplication algorithm. *IEEE Transactions on Computers*, C-22(12):1045–1047, 1973.

[77] C. S. Wallace. A suggestion for a fast multiplier. *IEEE Transactions on Electronic Computers*, EC-13(1):14–17, 1964.

[78] Swartzlander. Merged arithmetic. *IEEE Transactions on Computers*, C-29(10):946–950, 1980.

[79] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo. Truncated binary multipliers with variable correction and minimum mean square error. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(6):1312–1325, 2010.

[80] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo. Design of fixed-width multipliers with linear compensation function. *IEEE*

Transactions on Circuits and Systems I: Regular Papers, 58(5):947–960, 2011.

[81] D. De Caro, N. Petra, A. G. M. Strollo, F. Tessitore, and E. Napoli. Fixed-width multipliers and multipliers-accumulators with min-max approximation error. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(9):2375–2388, 2013.

[82] K. Bhardwaj, P. S. Mane, and J. Henkel. Power- and area-efficient approximate wallace tree multiplier for error-resilient systems. In *Fifteenth International Symposium on Quality Electronic Design*, pages 263–269, 2014.

[83] C. Lin and I. Lin. High accuracy approximate multiplier with error correction. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 33–38, 2013.

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

[85] M. Ha and S. Lee. Multipliers with approximate 4–2 compressors and error recovery modules. *IEEE Embedded Systems Letters*, 10(1):6–9, 2018.

[86] Y. Guo, H. Sun, L. Guo, and S. Kimura. Low-cost approximate multiplier design using probability-driven inexact compressors. In *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 291–294, 2018.

[87] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra. Approximate multipliers based on new approximate compressors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(12):4169–4182, 2018.

[88] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han. Low-power approximate multipliers using encoded partial products and approximate compressors. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(3):404–416, 2018.

[89] H. Pei, X. Yi, H. Zhou, and Y. He. Design of ultra-low power consumption approximate 4-2 compressors based on the compensation characteristic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, pages 1–1, 2020.

[90] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, and G. D. Meo. Comparison and extension of approximate 4-2 compressors for low-power approximate multipliers. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(9):3021–3034, 2020.

[91] V. Mrazek, Z. Vasicek, L. Sekanina, H. Jiang, and J. Han. Scalable construction of approximate multipliers with formally guaranteed worst case error. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(11):2572–2576, 2018.

[92] Zhongde Wang, G. A. Jullien, and W. C. Miller. A new design technique for column compression multipliers. *IEEE Transactions on Computers*, 44(8):962–970, 1995.

[93] Shen-Fu Hsiao, Ming-Roun Jiang, and Jia-Sien Yeh. Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers. *Electronics Letters*, 34(4):341–343, 1998.

[94] D. Radhakrishnan and A. P. Preethy. Low power cmos pass logic 4-2 compressor for high-speed multiplication. In *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems (Cat.No.CH37144)*, volume 3, pages 1296–1298 vol.3, 2000.

[95] Jiangmin Gu and Chip-Hong Chang. Ultra low voltage, low power 4-2 compressor for high speed multiplications. In *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03.*, volume 5, pages V–V, 2003.

[96] Chip-Hong Chang, Jiangmin Gu, and Mingyan Zhang. Ultra low-voltage low-power cmos 4-2 and 5-2 compressors for fast arithmetic circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(10):1985–1997, 2004.

[97] A. Najafi, S. Timarchi, and A. Najafi. High-speed energy-efficient 5:2 compressor. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 80–84, 2014.

[98] C. Liu, J. Han, and F. Lombardi. A low-power, high-performance approximate multiplier with configurable partial error recovery. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–4, 2014.

[99] H. Jiang, C. Liu, F. Lombardi, and J. Han. Low-power approximate unsigned multipliers with configurable error recovery. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(1):189–202, 2019.

[100] J. Liang, J. Han, and F. Lombardi. New metrics for the reliability of approximate and probabilistic adders. *IEEE Transactions on Computers*, 62(9):1760–1771, Sept 2013.

[101] 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.

[102] 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.

[103] Honglan Jiang, Jie Han, and Fabrizio 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.

[104] J. Schlachter, V. Camus, and C. Enz. Near/sub-threshold circuits and approximate computing: The perfect combination for ultra-low-power systems. In *2015 IEEE Computer Society Annual Symposium on VLSI*, pages 476–480, July 2015.

[105] J. Schlachter, V. Camus, K. V. Palem, and C. Enz. Design and applications of approximate circuits by gate-level pruning. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(5):1694–1702, May 2017.

[106] R. Hegde and N. R. Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. In *Proceedings. 1999 International Symposium on Low Power Electronics and Design (Cat. No.99TH8477)*, pages 30–35, Aug 1999.

[107] Honglan Jiang, Francisco J. H. Santiago, Mohammad Saeed Ansari, Leibo Liu, Bruce F. Cockburn, Fabrizio Lombardi, and Jie Han. Characterizing approximate adders and multipliers optimized under different design constraints. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, GLSVLSI '19, page 393–398, New York, NY, USA, 2019. Association for Computing Machinery.

[108] D. Esposito, D. De Caro, and A. G. M. Strollo. Variable latency speculative parallel prefix adders for unsigned and signed operands. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 63(8):1200–1209, Aug 2016.

[109] S. Xu and B. C. Schafer. Toward self-tunable approximate computing. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 27(4):778–789, April 2019.

[110] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram. RAP-CLA: A reconfigurable approximate carry look-ahead adder. *IEEE Trans. on Circuits and Systems II: Express Briefs*, 65(8):1089–1093, Aug 2018.

[111] F. Frustaci, S. Perri, P. Corsonello, and M. Alioto. Energy-quality scalable adders based on nonzeroing bit truncation. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, pages 1–5, 2018.

[112] 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.

[113] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han. Approximate arithmetic circuits: A survey, characterization, and recent applications. *Proceedings of the IEEE*, 108(12):2108–2135, 2020.

[114] M. Weißbrich, A. García-Ortiz, and G. Payá-Vayá. Comparing vertical and horizontal simd vector processor architectures for accelerated image feature extraction. *Journal of Systems Architecture*, 100:101647, 2019.

[115] A. Yazdanbakhsh, D. Mahajan, H. Esmaeilzadeh, and P. Lotfi-Kamran. Axbench: A multiplatform benchmark suite for approximate computing. *IEEE Design Test*, 34(2):60–68, April 2017.

[116] A. Raha and V. Raghunathan. Approximating beyond the processor: Exploring full-system energy-accuracy tradeoffs in a smart camera system. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(12):2884–2897, Dec 2018.

[117] R. Boyapati, J. Huang, P. Majumder, K. H. Yum, and E. J. Kim. Approx-noc: A data approximation framework for network-on-chip architectures. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 666–677, June 2017.

[118] A. Mineo, M. Palesi, G. Ascia, P. P. Pande, and V. Catania. On-chip communication energy reduction through reliability aware adaptive voltage swing scaling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(11):1769–1782, Nov 2016.

[119] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[120] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, June 2011.

[121] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. Internet of things applications: A systematic review. *Computer Networks*, 148:241 – 261, 2019.

[122] Y. Huang, W. Yu, E. Ding, and A. Garcia-Ortiz. Epkf: Energy efficient communication schemes based on kalman filter for iot. *IEEE Internet of Things Journal*, 6(4):6201–6211, 2019.

[123] W. Yu, Y. Huang, and A. Garcia-Ortiz. Optimal task allocation algorithms for energy constrained multihop wireless networks. *IEEE Sensors Journal*, 19(17):7744–7754, 2019.

[124] W. Yu, Y. Huang, E. Ding, and A. Garcia-Ortiz. Joint task allocation approaches for hierarchical wireless sensor networks. In *2018 7th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4, 2018.

[125] W. Yu, Y. Huang, and A. Garcia-Ortiz. Distributed optimal on-line task allocation algorithm for wireless sensor networks. *IEEE Sensors Journal*, 18(1):446–458, 2018.

[126] W. Yu, Y. Huang, and A. Garcia-Ortiz. Modeling optimal dynamic scheduling for energy-aware workload distribution in wireless sensor networks. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 116–118, May 2016.

[127] Texas-instruments. *CC2538 Datasheet*. Chipcon Products from Texas Instruments, USA: Texas Instruments, Copyright, 2013.

[128] Texas-Instruments. *TMS320C5509A Datasheet*. Chipcon Products from Texas Instruments, USA: Texas Instruments, Copyright, 2017.

[129] Y. Huang, W. Yu, and A. Garcia-Ortiz. Pkf: A communication cost reduction schema based on kalman filter and data prediction for wireless sensor networks. In *2013 IEEE International SOC Conference*, pages 73–78, 2013.

[130] A. N. Akansu, P. Duhamel, Xueming Lin, and M. de Courville. Orthogonal transmultiplexers in communication: a review. *IEEE Transactions on Signal Processing*, 46(4):979–995, 1998.

[131] M. Hao, L. Karsthof, J. Rust, J. Demel, C. Bockelmann, A. Dekorsy, A. A. Houry, F. Mackenthun, and S. Paul. Fpga-based baseband solution for high performance industrial wireless communication. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pages 1–5, 2018.

[132] Shousheng He and M. Torkelson. A new approach to pipeline fft processor. In *Proceedings of International Conference on Parallel Processing*, pages 766–770, 1996.

[133] L. Karsthof, M. Hao, J. Rust, and S. Paul. Ultra low latency implementation of robust channel estimation and equalization for industrial wireless communication systems. In *2019 17th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 1–4, 2019.

[134] R. Meyer. Error analysis and comparison of fft implementation structures. In *International Conference on Acoustics, Speech, and Signal Processing,*, pages 888–891 vol.2, 1989.

List of Figures

List of Tables