Design and Analysis of Approximate Adders and Multipliers

by

Cong Liu

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Integrated Circuits & Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Approximate adders have been considered as a potential alternative for error-tolerant applications to trade off some accuracy for gains in other circuit-based metrics, such as power, area and delay. Existing approximate adder designs have shown substantial advantages in improving many of these operational features. However, the error characteristics of the approximate adders still remain an issue that is not very well understood. A simulation-based method requires both programming effort and time-consuming simulations for evaluating the effect of errors. This method becomes particularly expensive when dealing with various sizes and types of approximate adders. As the first contribution of this thesis, a framework based on analytical models is proposed for evaluating the error characteristics of approximate adders. Error features such as the error rate and the mean error distance are obtained using this framework without developing functional models of the approximate adders for time-consuming simulation. As an example, the estimate of peak signal-to-noise ratios (PSNRs) in image processing is considered to show the potential application of the proposed analysis. This analytical framework provides an efficient method to evaluate various designs of approximate adders for meeting different figures of merit in error-tolerant applications.

In addition to adders, multipliers are also key arithmetic circuits in many error-tolerant applications such as digital signal processing (DSP). As the second contribution of this dissertation, a novel approximate multiplier with a lower power consumption and a shorter critical path than traditional (accurate) multipliers is proposed for high-performance DSP applications. This multiplier leverages a newly designed approximate adder that limits its carry propagation to the nearest neighbors for fast partial product accumulation. Different levels of accuracy can be achieved through a configurable error recovery by using different error reduction strategies. These designs use OR gates and the proposed approximate adder for two configurations of the multiplier: approximate multiplier 1 (AM1) and approximate multiplier 2 (AM2). Both AM1 and AM2 have a low mean error distance, i.e., most of the errors are not significant in magnitude. Compared to the Wallace multiplier, a 16×16 bit AM1 implemented in a 28-nm CMOS process shows a reduction in delay and power of 20% and up to 69%, respectively. AM2 has a better accuracy compared to AM1 but with a longer delay and higher power. Image processing applications such as image sharpening and smoothing are used to show the quality of the approximate multipliers in error-tolerant applications. It is shown that by utilizing an appropriate error recovery, the proposed approximate multipliers achieve similar processing accuracy as traditional accurate multipliers, but with significant improvements in power and performance.

A comparative evaluation of existing approximate multipliers, including the proposed ones, is also presented in this thesis. Monte Carlo simulations are performed to evaluate the error characteristics of these multipliers. Circuit simulations are further run to compare the delay, area and power consumption of these multipliers. The proposed approximate multipliers have high accuracies and lowest power-delay-products among all the designs, while the other designs have at least one major shortcoming in terms of error and/or circuit characteristics. Therefore the proposed designs achieve the best tradeoff between accuracy, delay and power.

To my family

Acknowledgements

I would like to give my deepest gratitude to my supervisor, Dr. Jie Han, for his continuous support and encouragement during my graduate study. This thesis would not have been finished without his profound knowledge and insightful suggestions. Furthermore, his motivation and attention to detail impressed me a lot. I would also like to express my gratitude to Dr. Fabrication Lombardi, who not only provided valuable feedbacks on my research but also polished my papers to a higher standard.

I would like to thank Dr. Bruce Cockburn and Dr. José Nelson Amaral for serving on my thesis committee. I also enjoyed the Digital System Testing and Design for Testability course instructed by Dr. Cockburn.

I am grateful to all my colleagues in the research group for their friendship and help. Many thanks to Jinghang Liang, Zhixi Yang, Peican Zhu, Ran Wang and Honglan Jiang. Thanks to Jinghang and Zhixi for the orientation on using software tools in the lab. Thanks to Honglan and Ran for sharing their knowledge in using simulation tools. I would also like to thank Naman Maheshwari for his contribution on the simulations during his summer internship.

I owe my deepest gratitude to my entire family for their unconditional love and support.

Last but not least, I would also want to express my sincere thanks to Mr. and Mrs. Jones, Alberta Innovates Technology Futures (AITF) and Natural Sciences and Engineering Research Council (NSERC) of Canada for their financial supports and CMC Microsystems for providing software tools used during the research.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	**Description**	**Page number of first use**
DSP	Digital Signal Processing	1
ALU	Arithmetic Logic Unit	2
CLA	Carry Look Ahead	2
VOS	Voltage Over Scaling	2
CPA	Carry Propagation Adder	3
ETM	Error-Tolerant Multiplier	3
MSB	Most/more Significant Bit	3
LSB	Least/less Significant Bit	3
ER	Error Rate	3
MED	Mean Error Distance	3
PSNR	Peak Signal-to-Noise Ratio	3
ASM	Application Specific Metrics	3
AM1	Approximate Multiplier 1	4
AM2	Approximate Multiplier 2	4
ACA	Almost Correct Adder	5
ESA	Equal Segmentation Adder	5
DSEC	Dynamic Segmentation and Error Compensation	5
ETAII	Error-Tolerant Adder Type II	6
ETA	Error-Tolerant Adder	6
SCSA	Speculative Carry Selection Addition	6
LOA	Lower-Part-OR Adder	8
UDM	Underdesigned Multiplier	9
BAM	Broken-Array Multiplier	9
BBM	Broken-Booth Multiplier	9
PDP	Power-Delay-Product	9
MSE	Mean Squared Error	9
AWTM	Approximate Wallace Tree Multiplier	10
ICM	Inaccurate Compressor based Multiplier	11
ED	Error Distance	13
CV	Coefficient of Variation	20

EA	Estimate Accuracy	24
MSED	Mean Squared Error Distance	27
RD	Relative Discrepancy	34
PDEM	Power-Delay-Error Metric	36
IPP	Input Pre-Processing	36
FA	Full Adder	39
HA	Half Adder	39
TAM1	Truncated Approximate Multiplier 1	45
TAM1	Truncated Approximate Multiplier 2	45
NMED	Normalized Mean Error Distance	45
RED	Relative Error Distance	45
MRED	Mean Relative Error Distance	45
LEM	Low-Error Multiplier	64
MEM	Medium-Error Multiplier	64
HEM	High-Error Multiplier	64

Chapter 1

Introduction

1.1 Background and Motivation

As the physical dimensions of CMOS circuits are scaled down to a few tens of nanometers, it has been increasingly difficult to improve circuit performance and/or to enhance power efficiency. Considering the explosive growth of portable devices such as smart phones, improving the power efficiency of digital circuits becomes increasingly important. Approximate computing, in both software and hardware, has been considered as a new approach to saving area and power, as well as increasing performance, however at the cost of a certain loss in accuracy.

While computation errors are not desirable, applications such as multi-media (image, audio and video) processing, wireless communications, recognition, and mining are tolerant to certain errors. Due to perceptual limitations of human beings, some errors do not make an obvious difference in applications such as video that are dedicated for human sensing. In many digital signal processing (DSP) systems, inputs from the real world are noisy, so there is no strict correctness in these systems. There are also many applications that are based on statistical/probabilistic computation, such as some clustering and recognition algorithms used in processing data. Due to the statistical/probabilistic nature of these applications, small errors in the computation will not impose a noticeable performance degradation. Therefore, approximate computing is applicable in various applications that can tolerate the loss of certain accuracy [37].

Research on approximate computing has been pursued in primarily two directions: software and hardware. Sampson et al. developed the EnerJ language, an extension to Java, that supports approximate data types for low-power computation [36]. In [5], an approximate squaring circuit is proposed. A new logic synthesis approach is proposed to maximally reduce the area of a synthesized circuit for a given error rate threshold in [9]. In [4], the so-called "scalable effort hardware design" approach is proposed to implement high efficiency hardware solutions for error resilient applications.

There lie huge opportunities for digital computing systems to embrace a significant improvement in computing speed and power efficiency by exploiting approximate computing methodologies in both software and hardware. To effectively implement an approximate computing system, hardware that supports approximate operation is in urgent need. Hence, research on basic but key arithmetic circuits is essential toward building an approximate computing hardware platform or a general purpose processor that supports approximate operations. This dissertation is focused on the design and analysis of two such arithmetic circuits, namely, adders and multipliers.

1.2 Current Research on Approximate Arithmetic Circuits

Logic circuits for binary arithmetic (referred to as arithmetic circuits throughout this thesis) have been studied for many decades [32]. By encoding a number into a binary format, arithmetic circuits perform arithmetic operations on the binary numbers, such as addition/subtraction, multiplication and division. Arithmetic circuits such as adders and multipliers are key components in an arithmetic logic unit (ALU), which is a fundamental building block in microprocessors. The speed and power consumption of the arithmetic circuits highly influence the performance of a processor. High-performance arithmetic circuits such as carry look ahead adders (CLAs) and Wallace tree multipliers have been proposed. However, there are theoretical bounds for the complexity of the critical path delay of arithmetic operations [6]. Limited by these bounds, traditional arithmetic circuits that perform exact operations can hardly be improved. Approximate arithmetic, which allows a certain loss of accuracy, is able to further reduce the critical path delay. Since most approximate designs leverage simplified logic, they tend to have less power consumption and area overhead as well. Approximate computing has thus become promising for overcoming power, area and delay constraints in VLSI design, albeit at the expense of a loss in computational accuracy [11].

Generally, there are two methodologies for reducing accuracy by approximations. The first methodology uses a voltage-over-scaling (VOS) technique for CMOS circuits to save power, while it also introduces errors into the circuit [12,24,29]. The second methodology is based on redesigning a logic circuit into an approximate version. While the VOS technique is applicable to most circuits for error-tolerant applications, an approximate redesign requires us to consider the functionalities of different logic circuits.

As one of the simplest, but key components of arithmetic circuits, adders have attracted extensive interest for redesign and implementation as approximate schemes. Approximate adders have been proposed by using a reduced number of transistors [10,41] and by truncating the carry propagation chain for a speculation-based operation [6,16,25,38,43]. The approximate speculative designs achieve a better performance in terms of area, power and delay compared to conventional adders. New metrics and simulation-based approaches have been proposed to model and evaluate approximate adders [15,20,28,37]. Monte Carlo or an exhaustive simulation has been employed to acquire data for analysis. This class of approaches are however time-consuming and require building functional models of the approximate designs. To improve efficiency, a mathematical characterization of the arithmetic accuracy of approximate adders is then required for a better understanding of the design prior to a simulation-based evaluation. In addition to generic metrics (such as the error rate), application specific measures (ASMs) such as the peak signal-to-noise ratio (PSNR) for image processing are well suited in practice. Without an approach to modeling the relationship between the generic metrics and the ASMs, extensive programming and simulation efforts are required to obtain the ASMs for assessing the impact and the potential of approximate computing in different applications. Therefore, an effective approach to obtain or estimate the ASMs from generic error metrics is needed; however, there appear to be no formal methodologies or analytical approaches for these purposes in the technical literature.

Compared to approximate adders, there has been relatively less effort in the design of approximate multipliers. A multiplier usually consists of three stages: partial product generation, partial product accumulation and a carry propagation adder (CPA) at the final stage [7]. [25] considers using approximate adders to generate the radix-8 Booth encoding 3X with error reduction. In [17], approximate partial products are computed using inaccurate 2×2 multiplier blocks, while accurate adders are used in an adder tree to accumulate the approximate partial products. [15] briefly discusses the use of approximate speculative adders for the final stage addition in a multiplier. The error tolerant multiplier (ETM) of [18] is based on the partition of a multiplier into an accurate multiplication part for the Most-Significant-Bits (MSBs) and a non-multiplication part for the Least-Significant-Bits (LSBs).

1.3 Contributions of this Thesis

In this thesis, an analytical framework for evaluating approximate adders, a novel approximate multiplier design and a comparison of approximate multipliers are presented.

- A mathematical characterization of the arithmetic accuracy (i.e., the error rate (ER) and mean error distance (MED)) of approximate adders is first presented. The revealed error characteristics provide insights into the quality of an appropriate adder for achieving a desired operational accuracy. In addition to analyzing generic metrics such as error rate, the peak signal to noise ratio (PSNR), as an example of application specific metrics (ASMs), is also analyzed as another important contribution of this framework. A model is presented for estimating the PSNR from the MED obtained from the proposed framework. Experimental results show that the estimated PSNRs are very close to the PSNRs obtained by simulation. The utilization of the proposed framework to PSNR estimate provides an analytical approach for assessing and designing an image processing system based on approximate adders.

- This thesis also contributes to the design of approximate arithmetic circuits by proposing a novel low-power high-speed approximate multiplier. This novel approximate multiplier design is based on a simple, yet fast approximate adder. This newly designed adder can process data in parallel by cutting the carry propagation chain (and thus, introducing an error). It has a critical path delay that is even shorter than a conventional one-bit full adder. Albeit having a high error rate, this adder simultaneously computes the sum and generates an error signal; this feature is employed to reduce the error in the final result of the multiplier. In the proposed approximate multiplier, a simple tree of the approximate adders is used for partial product accumulation and the error signals are used to compensate the error to obtain better accuracy as well as a reduced circuit complexity. The proposed multiplier can operate in either an approximate or accurate mode; in the approximate mode, it can be configured to be Approximate Multiplier 1 (AM1) and Approximate Multiplier 2 (AM2) based on different error recovery methods. Different levels of error recovery can also be achieved by using a different number of MSBs for error recovery in both AM1 and AM2. Compared to the traditional (exact) Wallace tree, the proposed multipliers have significantly lower power and shorter critical paths. Functional and circuit simulations are performed to evaluate the performance of the multipliers. Image sharpening and smoothing are considered as approximate multiplication-based DSP applications; experimental results indicate that the proposed approximate multipliers achieve a high performance in these error-tolerant image processing applications.

- As an increasing number of approximate multiplier designs are found in the literature, a comparative study is performed to evaluate these designs, including the proposed ones. Functional models of these designs are developed and Monte Carlo simulations using these models are performed to evaluate the error characteristics. Gate level models were also developed in VHDL and circuit measures (delay, area and power) are obtained using Synopsys Design Vision based on a 65-nm process. Comparisons of the error and circuit characteristics provide unique insights into the specific features of each multiplier design.

1.4 Outline of this Thesis

The outline of the remainder of the thesis is as follows. Chapter 2 presents the related work in both approximate adder and multiplier design. In chapter 3, an analytical framework that analyzes the error characteristics of three typical approximate adder designs is discussed. Chapter 4 presents the low-power and high-performance multiplier designs. A comparison of existing approximate multiplier designs is presented in chapter 5. Chapter 6 concludes the dissertation.

Chapter 2

Review of Approximate Adders and Multipliers

Arithmetic circuits (e.g., adders) are basic components of a digital processor. This chapter reviews existing approximate adder and multiplier designs.

2.1 Approximate Adders

The Speculative and Almost Correct Adder (ACA)

The so-called almost correct adder (ACA) [38] is based on the speculative adder design in [25]. The ACA utilizes insufficient information, i.e., k LSBs for predicting the sum of each bit in an n-bit adder ($n > k$). The same illustration (Fig. 2.1) as in [28] is used for ACA (and ESA in the following subsection). In Fig. 2.1, four bits (i.e., $k = 4$) are used to calculate each bit in the sum of an n-bit adder. The identical vertical rectangular blocks on the top denote the inputs, while the horizontal rectangles under them show the carry propagation paths for each sum bit. This design is based on the observation that the carry propagation chain is usually shorter than n, i.e., in practice, the truncation of the chain up to some length has a very low probability to be erroneous.

The Equal Segmentation Adder (ESA)

A dynamic segmentation and error compensation (DSEC) scheme is presented in [29] for an approximate adder design. This approximate adder consists of several sub-adders of different sizes divided from an n-bit adder; each of the sub-adders operates in parallel and has a truncated carry input. For convenience, but with no loss in correctness, sub-adders of equal size are considered in this thesis. Moreover the error compensation part [29] is neglected because the focus of this thesis is on analyzing the approximate operation. Thus, a simplified DSEC adder referred to as an equal segmentation adder (ESA) is analyzed in this thesis (Fig. 2.2).

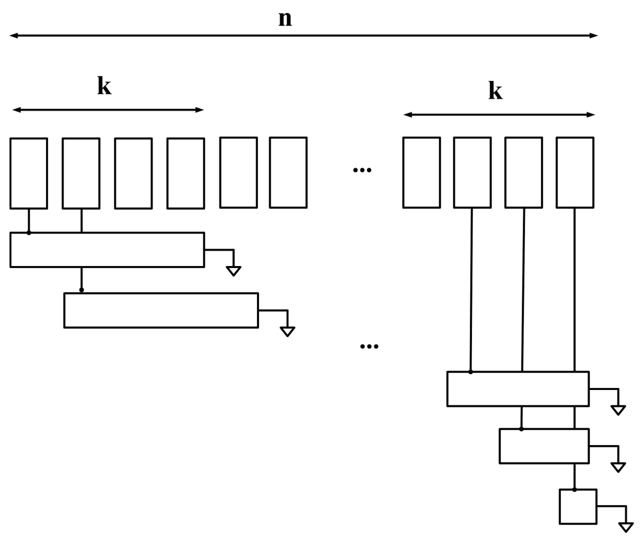


Figure 2.1: The almost correct adder. n: the adder size; k: the maximum carry chain length.

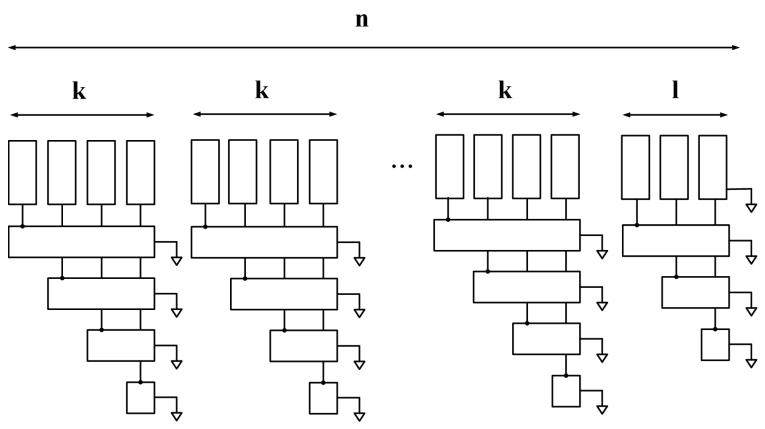


Figure 2.2: The equal segmentation adder. n: the adder size; k: the maximum carry chain length; l: size of the first sub-adder ($l \leq k$).

The Error-Tolerant Adder Type II (ETAII)

ETAII is also based on the truncation of the carry propagation chain and the segmentation of a full-sized adder [43]. Compared to ESA, the predicted carry input for each segmented k-bit sub-adder (or the sum generator in Fig. 2.3) is generated by k LSBs. ETAII has an improved accuracy compared to ESA because it uses more information to predict the carry when the same k is used. In addition to ETAII, several other Error Tolerant Adders (ETAs) were proposed by Zhu et al. [42, 44, 45].

The Speculative Carry Select Adder (SCSA)

A theoretical bound for the complexity of the critical path delay of an adder is $O\log(n)$, where n is the word length of the adder [6]. However, the so-called speculative carry selection addition

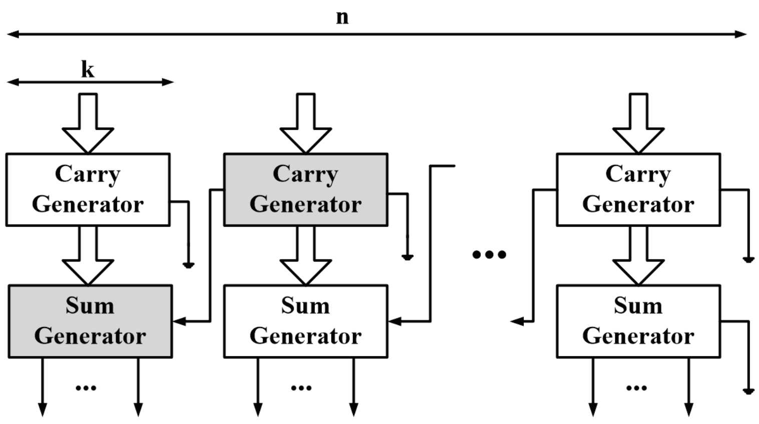


Figure 2.3: Block Diagram of error-tolerant adder type II. n: the adder size; k: half of the maximum carry chain length

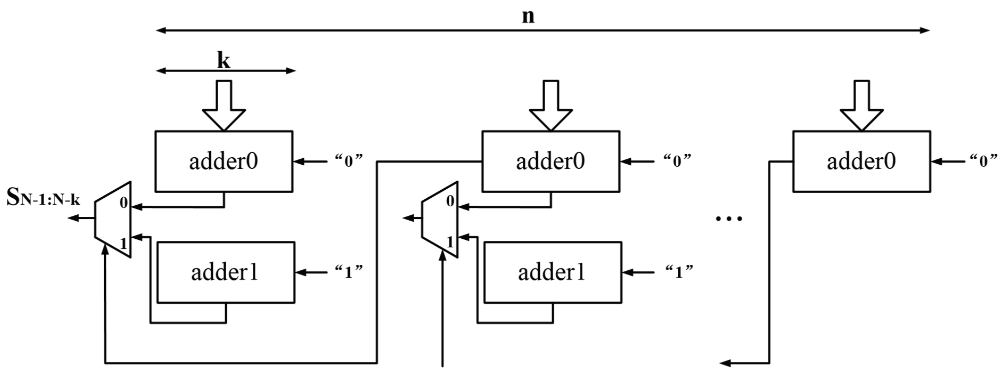


Figure 2.4: The speculative carry selection adder. n: the adder size; k: the maximum carry chain length.

(SCSA) [6] is able to achieve a sub-logarithmic delay by neglecting rare inputs that activate the critical path. In SCSA, an n-bit adder is first divided into $\lceil \frac{n}{k} \rceil$ sub-adders (also known as "window adders"); each sub-adder consists of two k-bit adders: adder0 and adder1 (Fig. 2.4). The only difference between the two k-bit adders is the carry input; the carry of adder0 is "0" while it is "1" for adder1. The output of the ith sub-adder is selected from adder0 and adder1 based on the carry out signal generated by the $(i-1)$th sub-adder. The carry out of each sub-adder is generated based on the k-bit in the sub-adder rather than all previous bits. Therefore, the carry selection process is still approximate and faster than a traditional carry selection scheme. Even though SCSA and ETAII have different circuit implementations, they share a similar functionality if their sub-adders have the same length. SCSA and ETAII generate the same carry signal for each sub-adder (or the Sum Generator in ETAII) but they use different circuit implementations.

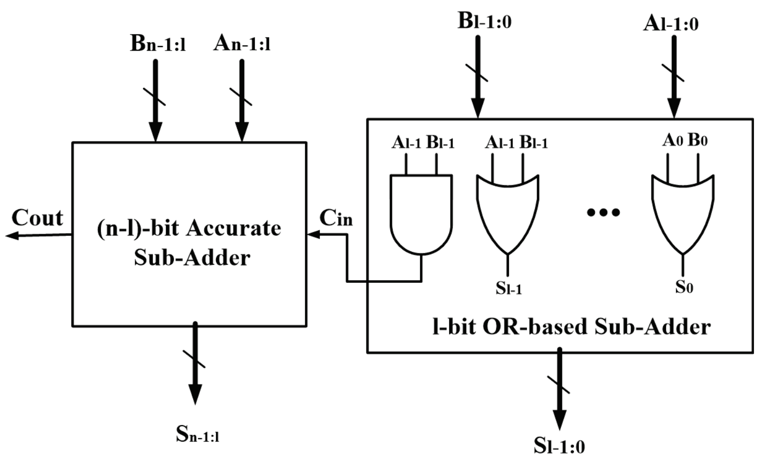


Figure 2.5: Lower-Part-OR Adder (LOA) [27]

The Lower-Part-OR Adder (LOA)

LOA [27] divides an n-bit adder to an $(n - l)$-bit more significant sub-adder and an l-bit less significant sub-adder. Since the l-bit sub-adder is less significant, its inputs are simply processed by applying OR gates to each bit, while the $(n - l)$-bit more significant sub-adder is an accurate adder. An extra AND gate is used to generate the carry-in signal for the more significant sub-adder by ANDing the most significant bits in the inputs of the less significant sub-adder.

Accuracy-Configurable Approximate Adder

Kahng and Kang [16] proposes an accuracy-configurable approximate adder. As accuracy can be configured during runtime by changing the circuit structure, a better tradeoff of accuracy versus performance and power can be achieved.

The Dithering Adder

The dithering adder [28] starts by dividing a multiple-bit adder into two sub-adders, and then uses a “Dither Control” signal to configure an upper or lower bound of the sum, resulting in a smaller overall error variance.

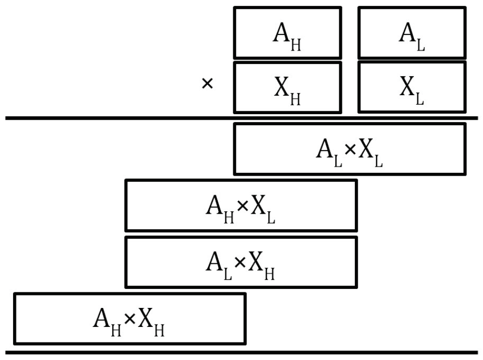
2.2 Approximate Multipliers

Approximate multiplier designs mainly use three approximation approaches: i) approximation in generating the partial products, ii) applying truncation in the partial product tree, and iii) using approximate adders and/or compressors to accumulate the partial products.

B_1B_0

A_1A_0	00	01	11	10
00	000	000	000	000
01	000	001	011	010
11	000	011	$\boxed{111}$	110
10	000	010	110	100

(a)



(b)

Figure 2.6: (a) K-Map for the 2x2 underdesigned multiplier block and (b) a 4x4 multiplier built on 2x2 blocks [17]

2.2.1 Approximation in Generating Partial Products

The Underdesigned Multiplier (UDM)

[17] proposes an approximate 2x2 multiplier block by altering one entry in the K-Map of a 2x2 multiplier. Based on the 2x2 block, larger underdesigned multipliers (UDMs) can be built (Fig. 2.6). This multiplier design introduces error in generating partial products while the adder tree remains accurate.

2.2.2 Approximation in the Partial Product Tree

Fixed-Width Multipliers

[34] presented an error compensation method for a modified Booth fixed-width multiplier, where quantization error is compensated using Booth encoder outputs. Another high-accuracy error compensation circuit for the fixed-width modified Booth multiplier was presented in [39]. This error compensation method significantly reduces the mean and mean-square errors by making the errors symmetric as well as centralizing error distribution in zero errors. Even though the authors did not explicitly name their designs as "approximate," these two fixed-width multipliers can be considered as approximate designs.

Brocken-Array Multiplier (BAM) and Broken-Booth Multiplier (BBM)

A bio-inspired imprecise multiplier referred to as the Broken-Array Multiplier (BAM) is proposed in [27]. BAM operates by omitting certain lines of carry-save adder cells in the carry-save adder tree both horizontally and vertically (Fig. 2.7).

Based on BAM, a Broken-Booth Multiplier (BBM) was presented in [8]. Compared to BAM, BBM uses modified Booth algorithm to generate partial products and only omits carry-save adders to the right of a vertical line. According to [8], BBM has a smaller power-delay-product (PDP) than BAM when they have the same mean squared error (MSE).

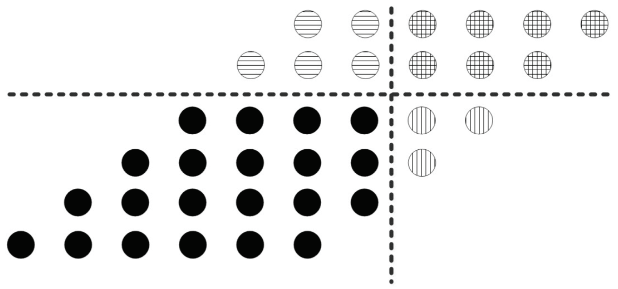


Figure 2.7: Structure of the Broken-Array Multiplier (BAM) with 4 vertical lines and 2 horizontal lines omitted [27]

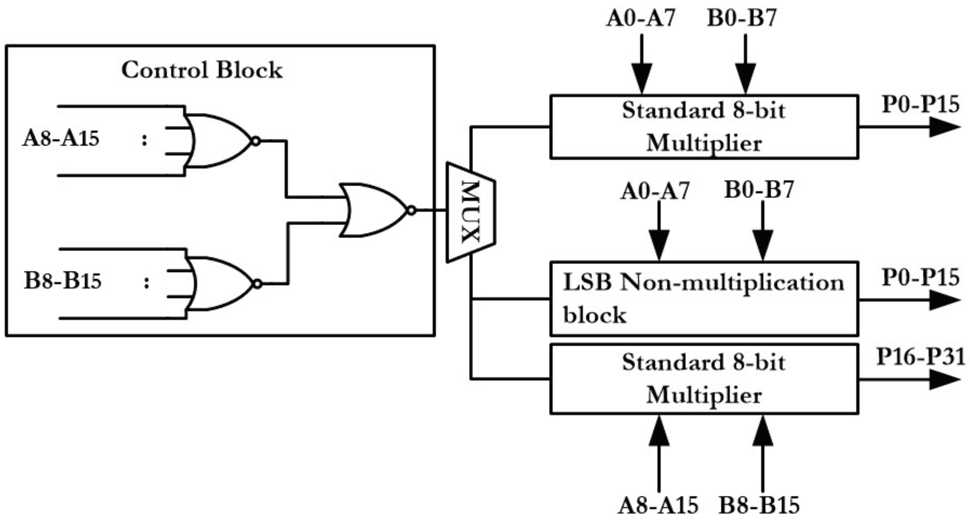


Figure 2.8: Architecture of a 16-bit Error-Tolerant Multiplier (ETM) [18]

Error-Tolerant Multiplier (ETM)

The Error-Tolerant Multiplier (ETM) [18] is divided into a multiplication part for the MSBs and a non-multiplication part for the LSBs, a control block is used to deal with two cases: i) if the product of the MSBs is zero, then a standard (accurate) multiplier is used to process the LSBs, and ii) if the product of the MSBs is non-zero, a standard multiplier is used to multiply the MSBs while a simple approximate multiplier is used to process the LSBs.

Approximate Wallace Tree Multiplier (AWTM)

A power and area-efficient AWTM is based on a bit-width aware approximate multiplication algorithm and a carry-in prediction method [1]. The architecture of this design is shown in Fig. 2.9, where the multiplier $A_H X_H$ is made accurate while the other three multipliers are approximate.

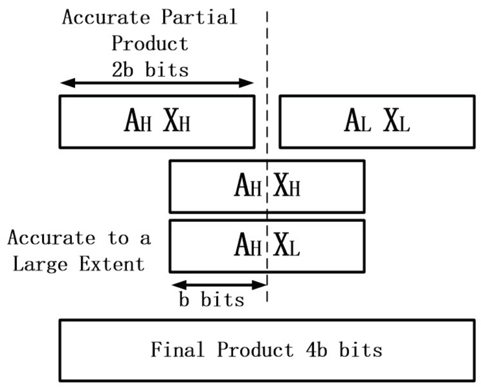


Figure 2.9: Architecture of an Approximate Wallace Tree Multiplier (AWTM) [1]

2.2.3 Using Approximate Compressors in the Partial Product Tree

In [26], a new approximate (4:2) compressor is used in a bottom-up tree topology to implement a high-speed, area-efficient and power-aware approximate multiplier. [21] proposed an inaccurate 4-bit Wallace multiplier based on a newly designed (4:2) approximate compressor. The inaccurate 4-bit multiplier is then used to build larger multipliers with error detection and correction circuits. This multiplier is referred to as Inaccurate Compressor based Multiplier (ICM) in this thesis. Two approximate (4:2) compressor designs are presented in [30] and then these compressors are used in a Dadda multiplier with four different schemes.

Chapter 3

An Analytical Framework for Evaluating the Error Characteristics of Approximate Adders

In this chapter, an analytical framework is presented for assessing the arithmetic accuracy, i.e., the error rate (ER) and mean error distance (MED), of approximate adders. These results are then used to estimate the PSNR in image processing. Three types of approximate adders are considered and their error features are compared using the proposed analysis.

The results of this chapter have been published in [22]. The organization of this chapter is as follows. Section 3.1 describes the analysis for modeling the error characteristics of the approximate adders. Discussion follows in Section 3.2. Section 3.3 investigates the PSNR estimate in image processing. Conclusions are given in section 3.4.

3.1 Error Analysis

The adders to be analyzed in this chapter are ESA, ACA, and ETAII, which have been introduced in section 2.1. For establishing the error characteristics, adders with the same functionality are considered to be the same type. For example, an ETAII and a SCSA with the same k and n values generate the same output for the same inputs. The accuracy-configurable approximate adder proposed in [16] can adjust the accuracy during runtime. For a given accuracy, the approximate configuration of the adder performs a similar function as ETAII. Thus, they have the same error characteristics. However, characteristics related to a circuit implementation such as delay and power are not necessarily the same. Some of the approximate adders also have an error correction circuit that permits an additional accurate operation mode; only the approximate operation of each adder is considered in this chapter.

3.1.1 Preliminaries

Metrics

The *error distance* (ED) and the *mean error distance* (MED) are proposed in [20] to evaluate the arithmetic performance of approximate circuits. For an n-bit approximate adder, the ED is defined as the absolute value of the difference between the approximate and accurate sums, i.e.,

$$ED = |S^{'} - S|, \tag{3.1}$$

where $S^{'}$ is the sum of the approximate adder and S is the sum of an accurate adder for a given input combination. The MED is defined as the average ED for a given set of input vectors, i.e.,

$$MED = E[ED] = \sum_{i=0}^{S_{MAX}} iP(i), \tag{3.2}$$

where S_{MAX} is the maximum output of an n-bit adder (i.e., $2^{n+1}-1$) and $P(i)$ is the probability of $ED = i$. The *error rate* (ER) is defined as the percentage of erroneous outputs among all outputs [3], i.e.,

$$ER = \sum_{i=1}^{S_{MAX}} P(i). \tag{3.3}$$

The above metrics (ER and MED) are of interest for evaluating the arithmetic performance of approximate adders. In the following section, an analytical method is presented to calculate these metrics for different types of adders.

Notation

The notation used in the error analysis throughout this chapter is introduced next. We consider an n-bit approximate adder with inputs A, B and C_0, and an output S. A_i, B_i, S_i are the corresponding input and output bits at the ith position. C_i is the carry to be added to the ith bit. Let $p_i = P(C_i = 1) = P(A_{i-1}B_{i-1} = 1) + P(A_{i-1} \oplus B_{i-1} = 1, C_{i-1} = 1)$, for uniformly-distributed inputs, $p_i = \frac{1}{4} + \frac{1}{2}p_{i-1}$, which leads to

$$p_i = \frac{1}{2} + \frac{1}{2^i}(p_0 - \frac{1}{2}), \tag{3.4}$$

where p_0 is the probability that the initial carry in bit is 1. Assume $p_0 = 0$, then

$$p_i = \frac{1}{2}(1 - \frac{1}{2^i}). \tag{3.5}$$

Let $\bar{X}_i$ and $\tilde{X}_i$ denote the events that the ith approximate sum bit is the same as or different from the ith exact sum bit, respectively, i.e., $\bar{X}_i = \{S_i = S_i^{'}\}, \tilde{X}_i = \{S_i \neq S_i^{'}\}$. An $\mathbf{X}$ vector consisting of $\bar{X}_i$'s or $\tilde{X}_i$'s is used to denote a set of outputs of the approximate adder compared to the accurate one. For example, for a 4-bit approximate adder, $\{\bar{X}_3\bar{X}_2\tilde{X}_1\}$ denotes a set of outputs in which S_1 is incorrect, both S_2 and S_3 are correct and S_0 could be either correct or incorrect.

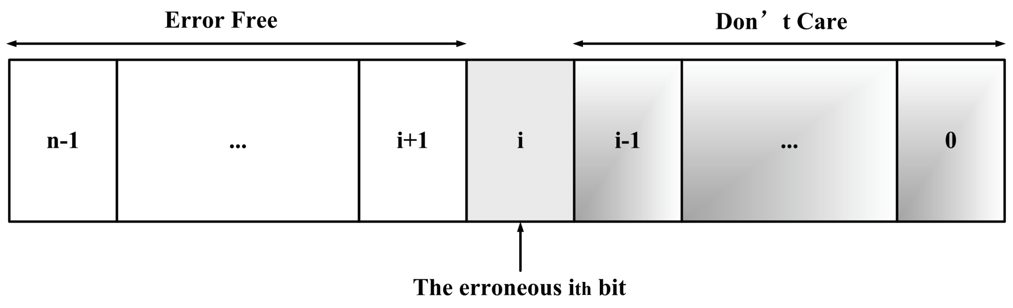


Figure 3.1: Sub error set $\mathbf{\Pi_i}$ in ACA

3.1.2 Error Characteristics of the Almost Correct Adder (ACA)

A *universal error set* is said to be formed by all possible errors of the ACA. To calculate the MED of an n-bit ACA, this error set is divided into n disjoint subsets and then the MED is calculated for each subset. The universal error set, denoted by $\mathbf{\Pi}$, is divided into the subsets of $\mathbf{\Pi_i}(i = 0, 1, ..., n-1)$, i.e.,

$$\mathbf{\Pi} = \cup \mathbf{\Pi_i}, \tag{3.6}$$

where $\mathbf{\Pi_i} = \{\bar{X}_{n-1}, \bar{X}_{n-2}, ..., \tilde{X}_i\}$. The errors in $\mathbf{\Pi_i}$ are those whose ith bit is erroneous, while the more significant bits are correct and the less significant bits are "don't cares", i.e., they can be either correct or erroneous, as shown in Fig. 3.1. Based on the error set division, the mean error distance of the approximate adder is calculated as:

$$MED = \sum_{i=0}^{n-1} E[|e_i|], \tag{3.7}$$

where $E[|e_i|]$ is the mean error distance of the subset $\mathbf{\Pi_i}$.

In the subset $\mathbf{\Pi_i}$, the errors in the ith bit (i.e., $\pm 2^i$) are dominant. Moreover, some of the errors in the lower bits can cancel each other. Therefore, the errors in $\mathbf{\Pi_i}$ have on average a magnitude of approximately 2^i; so the MED of $\mathbf{\Pi_i}$ is calculated as

$$E[|e_i|] \approx 2^i q_i, \tag{3.8}$$

where q_i is the probability that the error fall in $\mathbf{\Pi_i}$. The total mean error distance is then given by

$$MED = \sum_{i=0}^{n-1} 2^i q_i. \tag{3.9}$$

The error rate of the ACA is given by

$$ER = \sum_{i=0}^{n-1} q_i. \tag{3.10}$$

Next the calculation of q_i is presented. Let $\bar{P}_l$ be the probability that l consecutive bits in the approximate sum are correct (Fig. 3.2(a)) and $\tilde{P}_l$ be the probability that $l-1$ consecutive bits in the approximate sum are correct, but the next lower bit is erroneous (Fig. 3.2(b)), i.e.,

$$\bar{P}_l = P(\bar{X}_N, \bar{X}_{N-1}, ..., \bar{X}_{N-l+2}, \bar{X}_{N-l+1}), \tag{3.11}$$

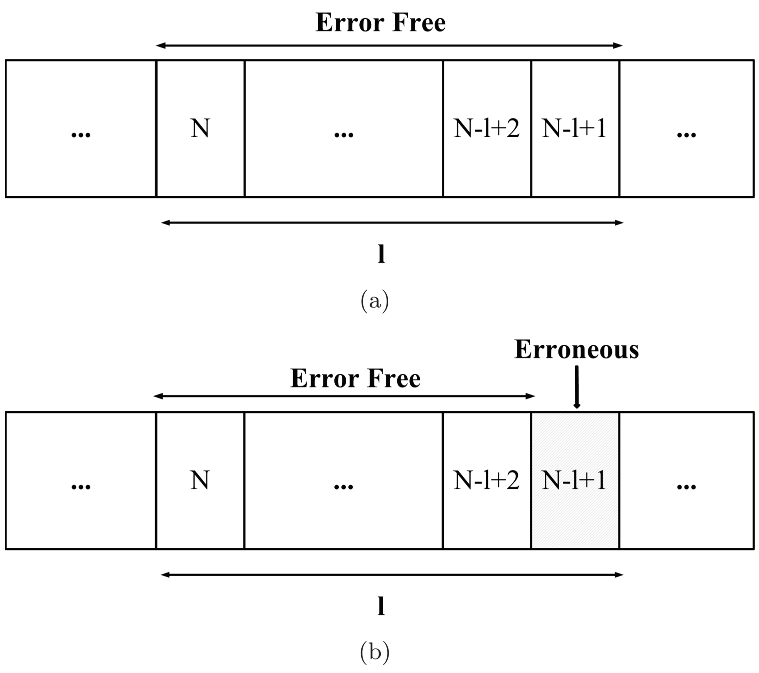


Figure 3.2: (a) $\bar{P}_l$, (b) $\tilde{P}_l$.

$$\tilde{P}_l = P(\bar{X}_N, \bar{X}_{N-1}, ..., \bar{X}_{N-l+2}, \tilde{X}_{N-l+1}), \tag{3.12}$$

Furthermore, let $\bar{Q}_l$ denote the conditional probability that one approximate sum bit is correct given that $l-1$ consecutive lower bits in the approximate sum are correct, and let $\tilde{Q}_l$ denote the conditional probability that one approximate sum bit is correct given that $l-2$ consecutive lower bits in the approximate sum are correct but the next lower bit is erroneous, i.e.,

$$\bar{Q}_l = P(\bar{X}_N | \bar{X}_{N-1}, ..., \bar{X}_{N-l+2}, \bar{X}_{N-l+1}), \tag{3.13}$$

$$\tilde{Q}_l = P(\bar{X}_N | \bar{X}_{N-1}, ..., \bar{X}_{N-l+2}, \tilde{X}_{N-l+1}), \tag{3.14}$$

Note that in these probabilities, only the length of the error is important, i.e., the exact index of each error, N, is less important.

Let the truncated carry propagation length be k; $\tilde{P}_l(l \leq k)$ is calculated first. Further denote the accurate sum as S and the inaccurate sum as $S^{'}$. As $S^{'}_{N-l+1}$ is the most significant erroneous bit, i.e., all higher bits in the sum are correct, the input carry, $C^{'}_{N-l+1-k}$, to calculate $S^{'}_{N-l+1}$, must propagate all the way to $S^{'}_{N-l+1}$, i.e., $P_i = A_i \oplus B_i = 1$, for $i = N-l-k+2, N-l-k+3, ..., N-l$. Also, $P_{N-l+1} = 0$ because, otherwise, the wrongly estimated carry would propagate to $S^{'}_{N-l+2}$. As $P_{N-l+1} = 0$, it is sufficient to show that $S^{'}_{N-l+2}, S^{'}_{N-l+3}, ..., S^{'}_{N}$ are correct because $P_{N-l+1} = 0$ prevents the error from propagating to the higher $k-1$ bits. Therefore ,

$$\begin{aligned}\tilde{P}_l &= P(P_{N-l-k+2}, ..., P_{N-l} = 1, P_{N-l+1} = 0, C^{'}_{N-l-k+2} \neq C_{N-l-k+2}) \\ &= \tfrac{1}{2^{k+1}}, l \geq 2.\end{aligned} \tag{3.15}$$

Let $\tilde{P}_1 = P(\tilde{X}_N)$, then

$$\tilde{P}_1 = P(P_{N-k+1}, ..., P_{N-2} = 1, P_{N-1} = 0, C^{'}_{N-k+1} \neq C_{N-k+1}) = \frac{1}{2^k}. \tag{3.16}$$

Thus,

$$\tilde{P}_l = \begin{cases} \frac{1}{2^{k+1}}, l \geq 2. \\ \frac{1}{2^k}, l = 1. \end{cases} \tag{3.17}$$

Since $\bar{P}_1 = \tilde{P}_2 + \bar{P}_2 = \tilde{P}_2 + \tilde{P}_3 + \bar{P}_3 = ... = \tilde{P}_2 + \tilde{P}_3 + ... + \tilde{P}_l + \bar{P}_l$, then

$$\bar{P}_l = \bar{P}_1 - \sum_{i=2}^{l} \tilde{P}_i, \tag{3.18}$$

where $\bar{P}_1 = 1 - \tilde{P}_1$. For $l \leq k$,

$$\tilde{Q}_l = \frac{\tilde{P}_l}{\tilde{P}_{l-1}}, l \leq k. \tag{3.19}$$

For $l > k$, since only k bits are used to calculate the current sum, the bits that are less significant than these k bits, have no influence on $\tilde{Q}_l$. Thus,

$$\tilde{Q}_l = \bar{Q}_k = \frac{\bar{P}_k}{\bar{P}_{k-1}}, l > k. \tag{3.20}$$

From all of the above, we obtain

$$\tilde{Q}_l = \begin{cases} \frac{\tilde{P}_l}{\tilde{P}_{l-1}} = 1, k \geq l > 2, \\ \frac{\tilde{P}_2}{\tilde{P}_1} = \frac{1}{2}, l = 2, \\ \frac{\bar{P}_k}{\bar{P}_{k-1}} = \frac{2^{k+1}-k-1}{2^{k+1}-k}, l > k. \end{cases} \tag{3.21}$$

This leads to

$$\begin{aligned} & q_i \approx P(\bar{X}_N, \bar{X}_{N-1}, ..., \bar{X}_{i+1}, \tilde{X}_i) \\ & = P(\bar{X}_N | \bar{X}_{N-1}, ..., \bar{X}_{i+1}, \tilde{X}_i) P(\bar{X}_{N-1}, ..., \bar{X}_{i+1}, \tilde{X}_i) \\ & = P(\bar{X}_N | \bar{X}_{N-1}, ..., \bar{X}_{i+1}, \tilde{X}_i) P(\bar{X}_{N-1} | \bar{X}_{N-2}, ..., \bar{X}_{i+1}, \tilde{X}_i) P(\bar{X}_{N-2}, ..., \bar{X}_{i+1}, \tilde{X}_i) \\ & = P(\bar{X}_N | \bar{X}_{N-1}, ..., \bar{X}_{i+1}, \tilde{X}_i) P(\bar{X}_{N-1} | \bar{X}_{N-2}, ..., \bar{X}_{i+1}, \tilde{X}_i) ... P(\bar{X}_{i+1} | \tilde{X}_i) P(\tilde{X}_i) \\ & = \tilde{Q}_{N-i+1} \tilde{Q}_{N-i} ... \tilde{Q}_2 \tilde{P}_1. \end{aligned} \tag{3.22}$$

Hence, (3.17), (3.21) and (3.22) can be used to calculate q_i in ACA error analysis.

3.1.3 Error Characteristics of the Equal Segmentation Adder (ESA)

Consider an n-bit ESA divided into r ($r = \lceil \frac{n}{k} \rceil - 1$) sub-adders of equal size k and 1 sub-adder of size $l = n - kr$. Thus there are $(r+1)$ sub-adders in total. Since the lowest sub-adder (i.e., the first sub-adder) is always error-free, only the higher r sub-adders that can be erroneous are considered. Initially the error in the $(m+2)$th sub-adder ($m = 0, 1, ..., r-1$) is considered; this is always 2^{mk+l}, as introduced by the wrong estimate of the input carry to this sub-adder.

When the speculative carry into each sub-adder is truncated to 0, the error rate of the $(m+2)$th sub-adder is given by

$$P(error = -2^{mk+l}) = P(C'_{mk+l} < C_{mk+l}) = p_{mk+l} \approx \tfrac{1}{2}. \tag{3.23}$$

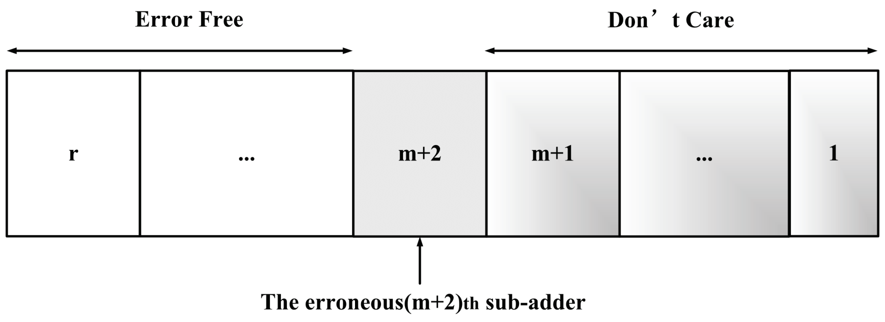


Figure 3.3: An example of the error in ESA dominated by the $(m+2)$th sub-adder.

The error rate of each sub-adder (except for the first one) is approximately $\frac{1}{2}$. All sub-adders are independent because there is no connection between them; so, the error rate of the entire adder is

$$ER = 1 - (1 - \frac{1}{2})^r = 1 - (\frac{1}{2})^r. \tag{3.24}$$

An approximate method is now introduced to calculate the mean error distance of ESA. Since the errors in a lower sub-adder are significantly smaller than those in a higher sub-adder, the error magnitude of the approximate adder is dominated by the highest erroneous sub-adder. For example, for 8-bit sub-adders, the error in the third sub-adder is 256 times greater than the error in the second sub-adder. Therefore, if a sub-adder is erroneous, the errors in the lower sub-adders become insignificant and thus, they can be ignored. Hence, for an ESA with $(r+1)$ sub-adders, the error magnitudes can be approximately divided into r levels, i.e., $\{2^l, 2^{k+l}, ..., 2^{(r-1)k+l}\}$. Fig. 3.3 shows a case when the error is dominated by the $(m+2)$th sub-adder, and the error probability is given by

$$P(error = -2^{mk+l}) \approx (\tfrac{1}{2})^{r-m-1} \times \tfrac{1}{2}. \tag{3.25}$$

The mean error distance is therefore given by

$$MED = \sum_{m=0}^{r-1} 2^{mk+l}(\tfrac{1}{2})^{r-m-1} \times \tfrac{1}{2} = \tfrac{2^{kr+r}-1}{2^{k+1}-1} \times 2^{l-r} \approx 2^{n-k-1}. \tag{3.26}$$

3.1.4 Error Characteristics of the Error-Tolerant Adder Type II (ETAII)

Similar to the analysis of ESA, ETAII uses the same partition scheme and its error analysis starts with the evaluation of the error rate in the $(m+2)$th sub-adder. The $(m+1)$th sub-adder generates the approximate carry, C'_{mk+l}, to the $(m+2)$th sub-adder based on the assumption that the input carry to itself, $C'_{(m-1)k+l}$, is 0. Thus, when the exact carry $C_{(m-1)k+l}$ is 1 and propagates through the $(m+1)$th sub-adder, the carry generated by the $(m+1)$th sub-adder is erroneous; thus, this results in an error in the $(m+2)$th sub-adder. Hence, the error rate of the $(m+2)$th sub-adder is given by

$$\begin{aligned}&P(error = -2^{mk+l}) = P(C'_{mk+l} < C_{mk+l}) = (\tfrac{1}{2})^k p_{mk+l}\\ &= (\tfrac{1}{2})^{k+1}(1 - \tfrac{1}{2^{mk+l}}) \approx \tfrac{1}{2^{k+1}}.\end{aligned} \tag{3.27}$$

In ETAII, the first and second sub-adders are always error-free, so there are in total $(r-1)$ sub-adders that can be erroneous. Hence, the ER of ETAII is given by

$$ER \approx 1-(1-\frac{1}{2^{k+1}})^{r-1}. \tag{3.28}$$

Similarly as for ESA, the error in ETAII is approximately divided among the $(r-1)$ levels. Hence

$$P(error=-2^{mk+l})=(1-\frac{1}{2^{k+1}})^{r-m-1}\times\frac{1}{2^{k+1}}, \tag{3.29}$$

and

$$MED=\sum_{m=1}^{r-1}2^{mk+l}(1-\tfrac{1}{2^{k+1}})^{r-m-1}\times\tfrac{1}{2^{k+1}}=\frac{2^{kr-k+l}-2^{l}(1-\frac{1}{2^{k+1}})^{r-1}}{2^{k+1}-2+2^{-k}}. \tag{3.30}$$

(3.30) can be simplified by ignoring $-2+2^{-k}$ in the denominator and $2^{l}(1-\frac{1}{2^{k+1}})^{r-1}$ in the numerator, hence

$$MED \approx 2^{n-2k-1}. \tag{3.31}$$

3.1.5 Error Analysis Using Different Carry Estimation Methods

ACA, ESA and ETAII use a fixed carry (with a value of 0) estimate such that the carry propagation chain is truncated. For example, for $k=10$ in ACA, an assumed input carry $C_1^{'}=0$ is used to calculate $S_{10}^{'}$. This assumption may lead to under-estimated results and the average error is non-zero. A straightforward solution to avoid the non-zero average error encountered in a method using a fixed carry is to use a random carry. If the input operands are uniformly distributed, one of the less significant input bits can be used as the random carry signal [15]. For example, $C_1^{'}=A_0$ or $C_1^{'}=B_0$ can be used to estimate C_1 in the previous ACA example. Hereafter, these two methods are referred to as the *fixed carry estimate* and *1-LSB carry estimate*, respectively.

In the previous section, the error characteristics of approximate adders with a fixed carry estimate have been discussed. Next, the error characteristics under the 1-LSB carry estimate case are considered.

ACA

A detailed solution under the 1-LSB carry estimate case is not provided because it can be obtained in a manner similar to the fixed carry case. When 1-LSB is used to estimate the carry, (3.17) and (3.21) are replaced by (3.32) and (3.33), respectively, while (3.22) remains the same.

$$\begin{aligned}\tilde{P}_l &= P(q_{N-l-k+2},...,q_{N-l}=1,q_{N-l+1}=0,C_{N-l-k+2}^{'}\neq C_{N-l-k+2})\\ &=\tfrac{1}{3}\tfrac{1}{2^k}(1+\tfrac{1}{2^{2l-1}}),l\le k.\end{aligned} \tag{3.32}$$

$$\tilde{Q}_l=\begin{cases}\frac{\tilde{P}_l}{\tilde{P}_{l-1}}=\frac{2^{2l-1}+1}{4(2^{2l-3}+1)},k\ge l\ge 2\\ \frac{\tilde{P}_k}{\tilde{P}_{k-1}}=\frac{1}{4}\frac{9\times2^{3k-1}-(6k+4)2^{2k-2}-1}{9\times2^{3k-3}-(6k-2)2^{2k-4}-1},l>k.\end{cases} \tag{3.33}$$

After q_i is calculated by (3.22), (3.9) and (3.10) can still be used to obtain the MED and ER in the 1-LSB carry estimate case.

ESA

The error rate of each sub-adder in the 1-LSB carry estimate case is half of the error rate in the fixed carry case (as discussed later). Thus, the procedure in the previous section can be utilized by changing the error rate of each sub-adder at the beginning. If 1-LSB is used to estimate the truncated carry, (3.23) is changed to:

$$\begin{aligned}&P(error = 2^{mk+l}) = P(C'_{mk+l} > C_{mk+l})\\&= P(A_{mk+l-1} = 1, B_{mk+l-1} = C_{mk+l-1} = 0)\\&= \tfrac{1}{4}(1 - p_{mk+l}) \approx \tfrac{1}{8},\end{aligned} \tag{3.34}$$

and

$$P(error = -2^{mk+l}) = P(C'_{mk+l} < C_{mk+l}) \approx \frac{1}{8}. \tag{3.35}$$

The mean error distance is then given by

$$MED = \sum_{m=0}^{r-1} 2^{mk+l}(\tfrac{3}{4})^{r-m-1} \times \tfrac{1}{4} = \tfrac{2^{kr}-(\frac{3}{4})^r}{2^{k+2}-3} \times 2^l \approx 2^{n-k-2}. \tag{3.36}$$

The ER is:

$$ER = 1 - (1 - \frac{1}{4})^r = 1 - (\frac{3}{4})^r. \tag{3.37}$$

ETAII

If 1-LSB is used to estimate the truncated carry, the analysis of ETAII is very similar to ESA, so the ER and MED for ETAII are given as follows:

$$ER \approx 1 - (1 - \frac{1}{2^{k+2}})^{r-1}, \tag{3.38}$$

and

$$MED = \sum_{m=1}^{r-1} 2^{mk+l}(1 - \tfrac{1}{2^{k+2}})^{r-m-1} \times \tfrac{1}{2^{k+2}} \approx 2^{n-2k-2}. \tag{3.39}$$

3.1.6 Monte Carlo Simulation

Error Analysis for Monte Carlo Simulation

Assume the accurate MED is μ and $\hat{\mu}_T$ is an estimate for MED obtained by averaging EDs from T iterations of Monte Carlo simulation. This can be modeled as a Monte Carlo integration approach [35]. The variance of $\hat{\mu}_T$ is

$$var(\hat{\mu}_T) = \frac{v}{T}, \tag{3.40}$$

where v is the variance of EDs given by

$$v = \sum_i (ED_i)^2 P(ED_i) - \mu^2. \tag{3.41}$$

For large T, by the Law of Large Numbers,

$$\hat{\mu}_T \sim N(\mu, \frac{v}{T}). \tag{3.42}$$

For a given confidence level, a parameter z_c can be determined to show the corresponding confidence interval. Therefore the error of $\hat{\mu}_T$ becomes

$$e = \frac{z_c}{\mu}\sqrt{\frac{v}{T}}. \tag{3.43}$$

For a confidence level of 95%, $z_c = 1.96$. Therefore, for $T = 1,000,000$, the error is

$$e = \frac{0.00196\sqrt{v}}{\mu} = 0.00196CV, \tag{3.44}$$

where $CV = \sqrt{v}/\mu$, the coefficient of variation.

(3.44) will be used in the following to analyze the error for Monte Carlo simulations of three approximate designs with a confidence level of 95%.

Considering (3.25), (3.26) and (3.41), the variance of the EDs of ESA for a fixed carry estimate is given by

$$\begin{aligned} v &= \sum_{m=0}^{r-1}(2^{mk+l})^2(\tfrac{1}{2})^{r-m-1} \times \tfrac{1}{2} - (2^{n-k-1})^2 \\ &\approx 2^{2n-2k-2}. \end{aligned} \tag{3.45}$$

By taking into consideration (3.26) and (3.45), the percentage error is 0.196% by (3.44), i.e., we are 95% confident that a simulated MED is within 0.196% of the true MED. Similarly, the variance of EDs of ESA for the 1-LSB carry estimate is

$$v \approx 3 \times 4^{n-k-2}. \tag{3.46}$$

The corresponding error is 0.34%.

For ETAII in the fixed carry estimate case, based on (3.29), (3.31) and (3.41), the variance of EDs is

$$\begin{aligned} v &= \sum_{m=1}^{r-1}(2^{mk+l})^2(1-\tfrac{1}{2^{k+1}})^{r-m-1} \times \tfrac{1}{2^{k+1}} - (2^{n-2k-1})^2 \\ &\approx 2^{2n-3k-1}. \end{aligned} \tag{3.47}$$

The corresponding error for a confidence level of 95% and 1,000,000 iterations of simulation is

$$e = 0.00196\sqrt{2^{k+1}}. \tag{3.48}$$

In the simulation, the smallest and largest k for ETAII is 4 and 10, thus the error is in the range of 1.11%-8.87%. For ETAII in the 1-LSB carry estimate case, the variance can be obtained in a similar way as in the fixed carry estimate case:

$$v \approx 2^{2n-3k-2}. \tag{3.49}$$

The corresponding error is

$$e = 0.00196\sqrt{2^{k+2}}. \tag{3.50}$$

The error is in the range of 1.57%-12.54% for k between 4 and 10.

Since it is difficult to get the theoretical variance for ACA, simulated variances are used. The CVs (i.e., $\sqrt{v}/\mu$) for different k and different carry estimate methods are presented in Table 3.1. According to Table 3.1 and (3.44), the errors of ACA are in the ranges of 1.3%-10.5% and 1.5%-12.5% for the fixed and 1-LSB carry estimates, as shown in Table 3.2.

Table 3.1: Coefficient of Variation (CV) values for ACA

	fixed carry estimate	1-LSB carry estimate
k=6	6.8	7.7
k=7	9.7	11.0
k=8	13.6	15.5
k=9	19.6	21.9
k=10	27.0	31.4
k=11	39.0	43.7
k=12	53.7	63.7

Table 3.2: Errors of Monte Carlo simulation for ACA

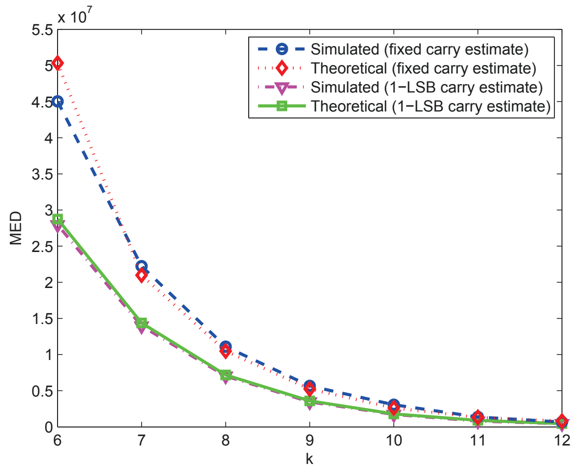
	fixed carry estimate	1-LSB carry estimate
k=6	1.3%	1.5%
k=7	1.9%	2.2%
k=8	2.7%	3.0%
k=9	3.8%	4.3%
k=10	5.3%	6.1%
k=11	7.7%	8.6%
k=12	10.5%	12.5%

Simulation Results

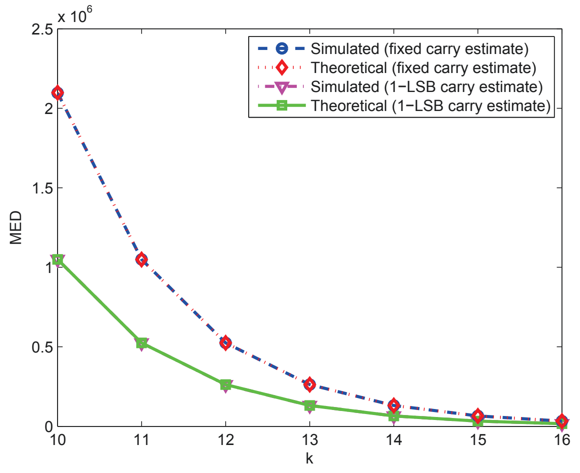
Fig. 3.4 and Fig. 3.5 show the simulation results and the analytical results for ACA, ESA and ETAII. The functional models of both the accurate and approximate adders were implemented in Matlab. 1,000,000 random input combinations are used to find the MED and ER values.

The simulation and the analytical MEDs are well matched especially for ESA for which the theoretical and simulated curves overlap. There are mainly two sources of discrepancy. The first source is due to the simulation method, i.e., Monte Carlo simulation is not exhaustive. The second source is caused by the approximation used in the analytical framework. As shown in the figures, the MED drops exponentially as k is increased: the MED drops approximately to half of its previous value when k is increased by 1 for all three approximate adders. The difference between MED values with different k values for the same adder is very large; therefore, the small discrepancy between the analytical and simulated results is rather negligible, i.e., the discrepancy will not result in an incorrect assessment of k.

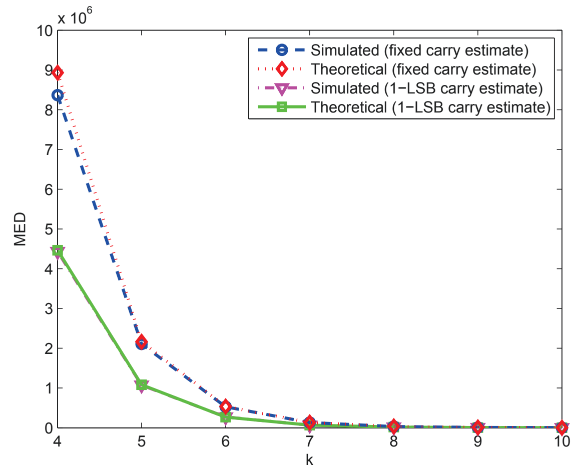
The simulation results for the error rate (ER) are shown in Fig. 3.5. For ACA, ER decreases to half of its previous value when the value of k is increased by 1. For any k larger than 9, the ER drops below 2% for both 1-LSB and fixed carry estimates. Due to the design, the ER of ESA is nearly constant for a value of k in a range between 11 and 15; it starts to decrease significantly when k is 16. In general, ESA has a very high error rate, even though it has a very small MED. ETAII can significantly reduce ER. When $k = 6$, for example, the ER of ETAII with 1-LSB carry estimate is below 2%.



(a)

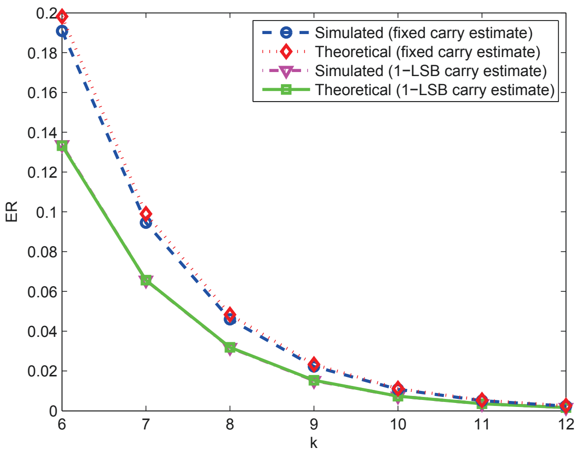


(b)

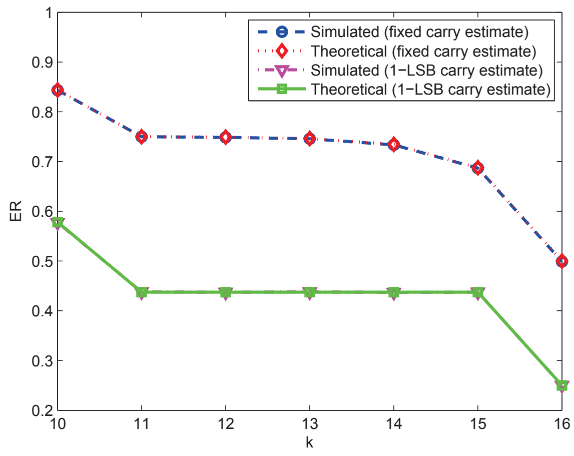


(c)

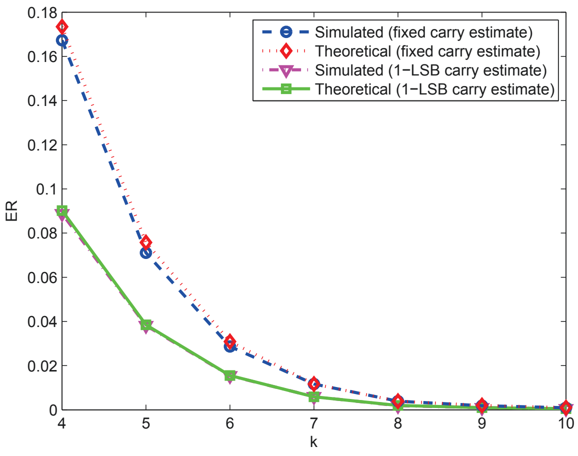
Figure 3.4: Simulated and theoretical MED for a 32-bit (a) ACA, (b) ESA, and (c) ETAII.



(a)



(b)



(c)

Figure 3.5: Simulated and theoretical ER for a 32-bit (a) ACA, (b) ESA, and (c) ETAII.

3.2 Discussion

In this section, different features as related to carry estimate are analyzed for the various approximate adders.

3.2.1 Generalizing Carry Estimation Methods

The 1-LSB carry estimate generates rather symmetrical and centralized errors, while a fixed carry tends to generate biased errors. Another significant advantage of 1-LSB carry estimate is that it reduces the MED. As shown in the simulation results (Fig. 3.4), the MED of 1-LSB carry estimate is approximately half of the MED of a fixed carry for all three types of approximate adder. Consider the carry estimate accuracy (EA), i.e., the probability that the estimated carry is equal to the exact carry. If the carry is fixed to 0, then $EA = P(C_i = 0) = 1 - p_i \approx \frac{1}{2}$. For the 1-LSB carry estimate, $EA = P(C_i = A_{i-1}) = P(C_i = 1|A_{i-1} = 1)P(A_{i-1} = 1) + P(C_i = 0|A_{i-1} = 0)P(A_{i-1} = 0) = \frac{3}{4}$. It is also intuitively true that the 1-LSB carry estimate method has a higher accuracy than a fixed carry estimate because it uses more information (when $A_{i-1} = 1$, C_i is more likely to be 1). It is then evident that the use of the 1-LSB to estimate carry reduces the ER as well as the MED.

The estimate of the carry signals is a significant issue for an approximate adder design. Intuitively, the utilization of more less significant bits results in a better performance to predict the carry signal. Hereinafter, "k-LSB carry estimate" refers to using the k bits in both A and B that are less significant than the current index to estimate the current carry. The fixed carry approach fails to use the LSBs in the estimate process, while the 1-LSB carry estimate method uses only 1 LSB. For ETAII and SCSA, if the Sum Generator is considered as a sub-adder, then the Carry Generator (Fig. 2.3) is the circuitry for the k-LSB carry estimate. Thus, ETAII uses more LSBs for the carry estimate than ESA and this increases the accuracy. Nevertheless, the use of more LSBs incurs a larger area overhead and longer delay; a trade-off must be made between the number of LSBs for carry prediction and circuit performance.

Table 3.3 shows the truth table of C_{i+1} given 1-LSB, i.e., A_i and B_i, where "U" means "unknown". More LSBs must be used to determine the unknown values. Without the information provided by the additional LSBs, the unknown values have approximately a probability of 0.5 to be either 1 or 0. Therefore, the best EA using 1-LSB information is $\frac{2+2/2}{4} = \frac{3}{4}$. The 1-LSB estimated carry, i.e., $C'_{i+1} = A_i$, uses a logic function based on 1-LSB information for achieving an EA of $\frac{3}{4}$. Consider the case in which k LSBs are used to estimate the carry C'_m. If the propagates in these k positions are all 1's, i.e., $P_i = A_i \oplus B_i = 1, i = m-1, m-2, ..., m-k$, C'_{m-k} is required to determine C'_m. Therefore, without the information of C'_{m-k}, there are 2^k unknown entries in the truth table of ${C_m}'$ based on previous k LSBs. In the k-LSB carry estimate method, these entries are arbitrarily assigned with certain values (1 or 0), which on average can successfully estimate half (i.e., 2^{k-1}) of the unknown entries. There are totally 2^{2k} input combinations, with 2^k unknown entries and $2^{2k} - 2^k$ known entries in the truth table. Thus, the highest EA based on k LSBs is $\frac{2^{2k}-2^{k-1}}{2^{2k}} = 1 - 2^{-k-1}$. However, the utilization of more LSBs for carry prediction increases both the delay and the circuit complexity (i.e., the number of transistors). To reduce complexity, additional

Table 3.3: Truth table of C_{i+1} given A_iB_i

A_iB_i	C_{i+1}
00	0
01	U
10	U
11	1

“approximate” carry estimates based on k LSBs can be derived by slightly changing some entries in the truth table. Two examples are provided by using 2 LSBs, as given in (3.51) and (3.52), with EA being 0.8125 and 0.875, respectively. Note that (3.52) achieves the highest EA of the 2-LSB carry estimate while (3.51) has a lower EA with a simpler logic implementation compared to (3.52).

$$C_{i+1} = A_iB_i + A_iA_{i-1}. \tag{3.51}$$

$$C_{i+1} = A_iB_i + A_{i-1}(A_i \oplus B_i). \tag{3.52}$$

3.2.2 Comparison of the Different Approximate Adders

ACA has an area complexity of $O(n \log\log n) \approx O(n)$, and a delay complexity of $O(\log k)$ [38], where n is the size of the adder and k is the maximum carry chain length. ESA has the same complexity as ACA if a parallel adder structure, such as carry look-ahead (CLA), is used to implement each sub-adder in ESA. However if real area and delay values are considered, ACA has a more complicated structure than ESA for the same k, and thus, it has a larger area overhead and probably a longer delay. As an example consider $k = 10$ and the 1-LSB carry estimate case. The simulated MEDs of ACA and ESA are 1.54×10^6 and 1.05×10^6, respectively. In terms of MED, ESA is better than ACA. If area and delay are taken into consideration, ESA is certainly a better scheme because it has a smaller MED as well as smaller area and delay. However, ACA has a smaller ER: the ER of ACA is 0.0074 while the ER of ESA is 0.5775, which is 78 times of the ER of ACA.

In ESA, $\lceil \frac{n}{k} \rceil$ sub-adders have to be implemented. The area and delay complexities are $\lceil \frac{n}{k} \rceil A(k)$ and $\tau(k)$, respectively, where $A(k)$ and $\tau(k)$ are the area and delay complexities of a k-bit adder. In ETAII, the area complexity is $(2\lceil \frac{n}{k} \rceil - 2)A(k)$ because every sub-adder is duplicated except for the first and last sub-adders. The delay complexity of ETAII is $2\tau(k)$ because the critical path contains two k-bit sub-adders in series. SCSA has the same error characteristics as ETAII, but it has a different circuit implementation. The delay of SCSA is $\tau(k)$ with the delay of a multiplexer. Compared to ETAII, the last k-bit sub-adder still needs duplication and $(\lceil \frac{n}{k} \rceil - 1)$ k-bit multiplexers are needed (they are not used in ETAII). Therefore, SCSA is approximately two times faster than ETAII at the cost of an increased area overhead.

For comparison purposes, choose $k = 10$ for ESA and $k = 5$ for ETAII with a 1-LSB carry estimate because these two implementations have roughly the same delay (even though ETAII requires more area). The MED of ESA is 1.05×10^6, while the MED of ETAII is 1.07×10^6; the

Table 3.4: Comparison between the adders

	ACA	ESA	ETAII	SCSA
k	10	10	5	10
Delay	O(logk)	O(logk)	O(log2k)	O(logk)
Area	O(nlog(log(n)))	O(n)	O(n)	O(n)
MED ($\times 10^3$)	1540	1050	1074	1.074
ER (%)	0.74	57.8	3.81	0.05

ERs are 0.5775 and 0.0381 for ESA and ETAII, respectively. Therefore for the compared schemes, ESA and ETAII have roughly similar MED, but ETAII has a significantly smaller ER. Hence, ETAII tends to generate large error magnitudes because it has a similar MED as ESA, but a very small ER.

Consider $k = 10$ for ACA, ESA and SCSA, and $k = 5$ for ETAII as further examples. These three adders have similar critical path delays. ESA has the least area overhead, but the largest ER. ACA has the smallest ER, but it has the largest MED. According to [16], ACA occupies 36% more area but it incurs in a 30% smaller delay compared to ETAIIM (a modified version of ETAII), with both adders having the same carry propagation length. ETAII has a significantly reduced MED compared to ACA and an acceptable ER of 3.81%. SCSA is faster than ETAII at the cost of an increase in area (due to the additional multiplexers). If the design objective is an extremely fast approximate adder, then SCSA is the best choice among these four approximate adders because it achieves a similar MED to other approximate adders with a smaller k (i.e., a shorter critical path delay). SCSA ($k = 10$) has approximately the same delay as ETAII ($k = 5$). However, SCSA with $k = 10$ has an extremely small MED and ER: the ER is only 0.05%, while the MED is 10^{-3} of ETAII ($k = 5$), as shown in Table 3.4.

3.3 Application: Image Quality Evaluation Using PSNRs

3.3.1 Peak Signal-to-Noise Ratio (PSNR)

PSNR is widely used in many DSP applications (such as image processing) as an important figure of merit. In image processing, if **I** is the noise-free image and **K** is the noisy image, the PSNR is defined as [13]:

$$PSNR = 20\log(MAX_I/\sqrt{MSE_K}), \tag{3.53}$$

where MAX_I is the maximum possible pixel value of image **I** and MSE is the mean squared error defined as

$$MSE_K = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[\mathbf{I}(i,j) - \mathbf{K}(i,j)]^2. \tag{3.54}$$

When approximate circuits are used for image processing, **I** can be the resulting image using an exact computation while **K** is the image obtained by approximate computing. For a good agreement with a precisely processed image, the PSNR of a noisy image should be very large - usually larger than a threshold P, i.e.,

$$PSNR > P,$$

or

$$\sqrt{MSE_K} < C, \tag{3.55}$$

where $C = 10^{\log(MAX_I) - \frac{P}{20}}$.

For (3.54), define an error matrix as $\mathbf{E} = \mathbf{K} - \mathbf{I}$; then the MSE and MED of image $\mathbf{K}$ are

$$MSE_K = E(\mathbf{E} \circ \mathbf{E}), \tag{3.56}$$

$$MED_K = E(|\mathbf{E}|). \tag{3.57}$$

In (3.56) and (3.57), $E(\mathbf{X})$ denotes the average value of all the elements in matrix $\mathbf{X}$, the "$\circ$" operation obtains the element-wise product of two matrices, which is also known as the *Hadamard product* [14], and $|\mathbf{X}|$ obtains the absolute value of each element in $\mathbf{X}$.

Let d and μ denote the ED and MED of the approximate adder and σ^2 represent the mean squared error distance (MSED):

$$\sigma^2 = MSED = E[d^2]. \tag{3.58}$$

The MED and MSED of the approximate adders are obtained under the assumption that the inputs are uniformly distributed. In this analysis, the pixel values of an image are assumed to be sufficiently random, i.e., if an image is processed by an approximate addition for each pixel, the mean and mean squared value of the error matrix are considered to be the same as the MED and the MSED of the approximate adder. Hereinafter, μ and σ^2 denote the MED and MSED of the approximate adder, as well as the mean and mean squared values of the error matrix if a resulting image is obtained by one approximate addition of each pixel. If the image $\mathbf{K}$ is processed by a approximate addition operations, the mean squared error, MSE_K, is given by

$$MSE_K = E[(\sum_{i=1}^{a} \mathbf{E_i}) \circ (\sum_{i=1}^{a} \mathbf{E_i})], \tag{3.59}$$

where $\mathbf{E_i}$ is the error matrix of the ith addition. Assume each $\mathbf{E_i}$ is independent, then (3.59) becomes:

$$\begin{aligned} MSE_K &= E[\sum_{i=1}^{a} \mathbf{E_i} \circ \mathbf{E_i} + 2 \sum_{1 \leqslant i < j \leqslant a} \mathbf{E_i} \circ \mathbf{E_j}] \\ &\leqslant a\sigma^2 + 2 \sum_{1 \leqslant i < j \leqslant a} E(|\mathbf{E_i}|)E(|\mathbf{E_j}|) \\ &\approx a\sigma^2 + a(a-1)\mu^2. \end{aligned} \tag{3.60}$$

Note that for adders whose error distribution is symmetrical (e.g., LOA), i.e., the mean error is close to zero, $E[\sum_{1 \leqslant i < j \leqslant a} \mathbf{E_i} \circ \mathbf{E_j}] \approx 0$, therefore (3.60) becomes

$$MSE_K \approx a\sigma^2. \tag{3.61}$$

Assume the relationship between $\sqrt{MSE}$ (i.e., σ) and MED (i.e., μ) is given by $\sigma = f(\mu)$ (the function $f(\cdot)$ will be derived next), then (3.60) and (3.61) can be converted to the following equations, respectively.

$$\sqrt{MSE_K} \approx \sqrt{af(\mu)^2 + a(a-1)\mu^2} \leqslant C \tag{3.62}$$

$$\sqrt{MSE_K} \approx \sqrt{af(\mu)^2} \leqslant C \tag{3.63}$$

Based on (3.53), (3.62) and (3.63), the PSNR can be estimated by

$$PSNR \approx \begin{cases} 20log(MAX_I/\sqrt{af(\mu)^2}), \text{ for symmetrical error distribution.} \\ 20log(MAX_I/\sqrt{af(\mu)^2 + a(a-1)\mu^2}), \text{ for biased error distribution.} \end{cases} \tag{3.64}$$

The solutions of (3.62) and (3.63) give the maximum value of MED for deriving the parameter k in ESA and ETAII using the corresponding equations (i.e., (3.26) and (3.30)) given in section 3.1. Hence, this analytical framework can be used to select the proper approximate adder type and parameter (i.e., carry propagation length k) for image processing applications, instead of building functional models of the approximate adder and running time-consuming simulations with different parameters.

3.3.2 Relationship between μ and σ

ESA

The relationship between μ and σ is analyzed for the fixed carry case. Similar to the MED calculation in (3.26), the MSE of ESA is calculated as:

$$\sigma^2 = \sum_{m=0}^{r-1} (2^{mk+l})^2 (\tfrac{1}{2})^{r-m-1} \times \tfrac{1}{2} \approx 2^{2n-2k-1}. \tag{3.65}$$

Based on (3.26) and (3.65), the relationship between μ and σ for ESA is given by:

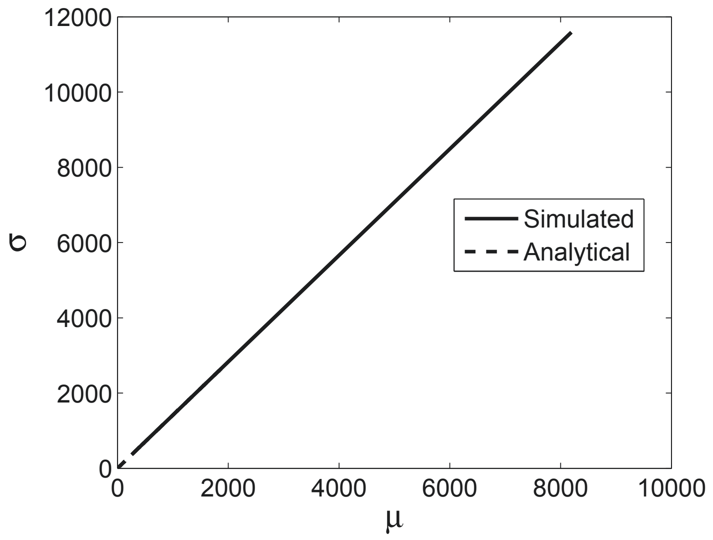
$$\sigma = \sqrt{2}\mu. \tag{3.66}$$

ETAII

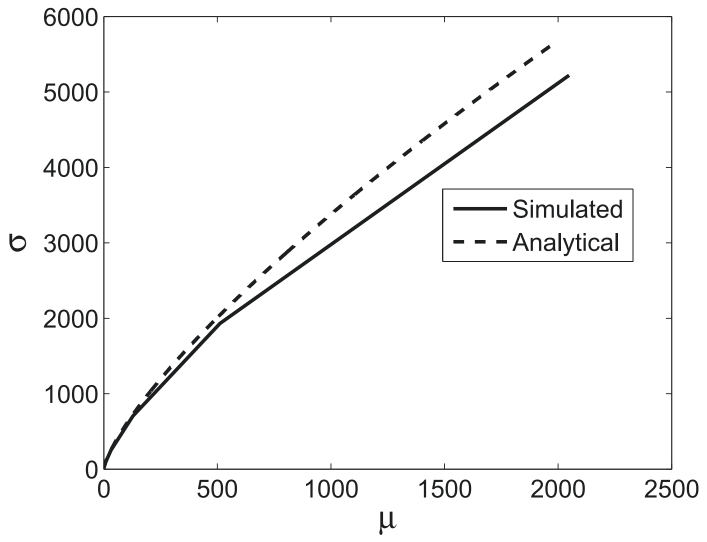
Similar to the analysis for ESA, the MED and MSE for ETAII are found as $\mu \approx 2^{n-2k-1}$,$\sigma^2 \approx 2^{2n-3k-1}$. The relationship between μ and σ is:

$$\sigma = (2^{n+1}\mu^3)^{\frac{1}{4}}. \tag{3.67}$$

The simulated relationship between μ and σ and the analytical functions in (3.66) and (3.67) are plotted as in Fig. 3.6 for $n = 16$. For ESA, the simulated curve and the analytical results match very closely, as shown by the nearly perfect overlapping curves. For ETAII, there is a slight discrepancy in the simulated and analytical plots. This discrepancy is due to the approximation when calculating MED and MSE, as outlined previously.



(a)

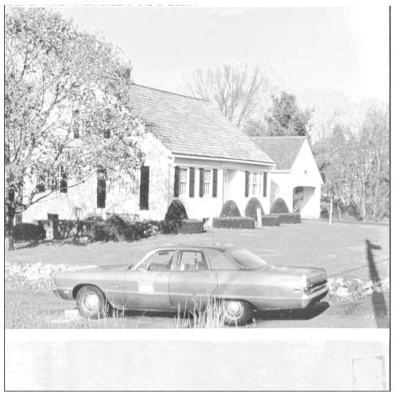


(b)

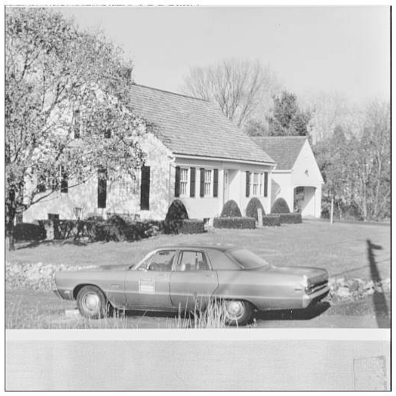
Figure 3.6: Simulated and analytical $\sigma - \mu$ relationships for (a) ESA, and (b) ETAII.

(a)

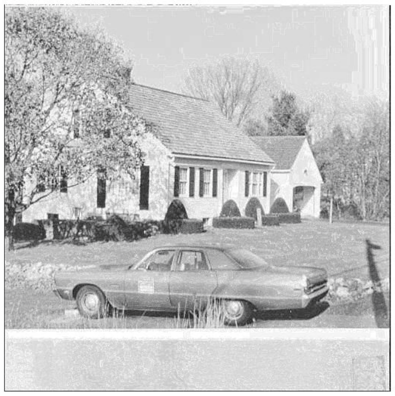
(b)



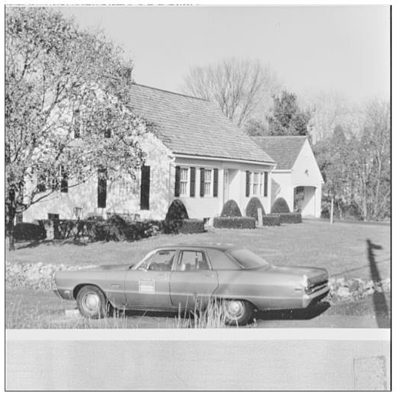
(c)



(d)

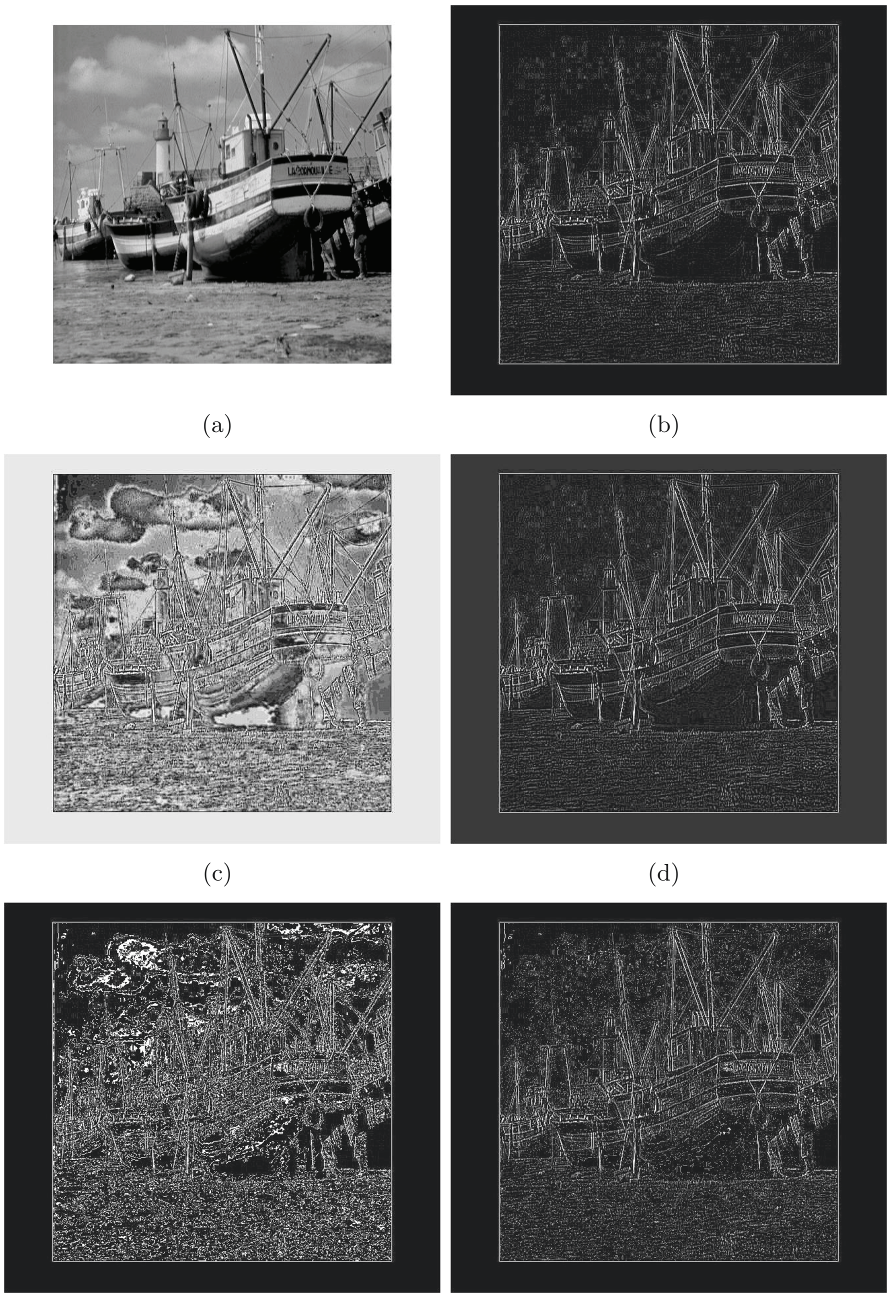


(e)



(f)

Figure 3.7: Image sharpening: (a) original image, (b) using exact adders, (c) using ESA with k=6 and PSNR=18.1dB, (d) using ESA with k=10 and PSNR=41.4dB, (e) using ETAII with k=4 and PSNR=27.4dB and (f) using ETAII with k=7 and PSNR=56.8dB.

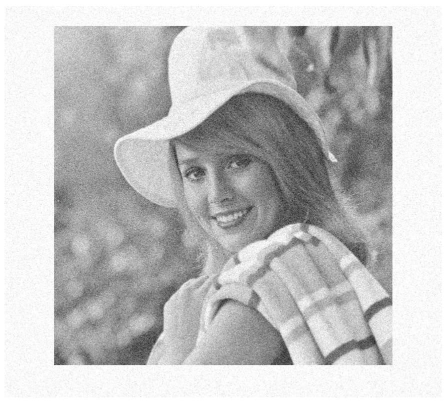


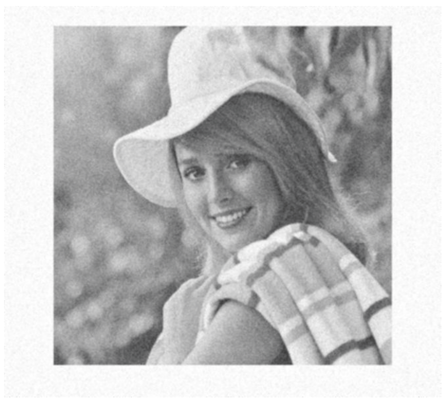
(a) (b)

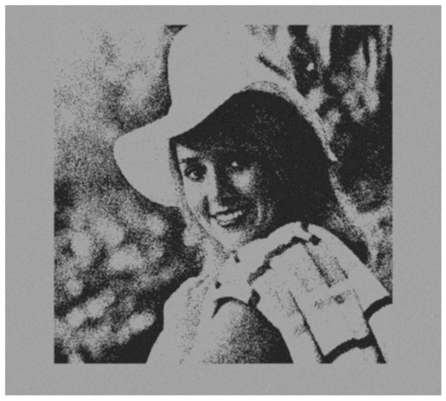
(c) (d)

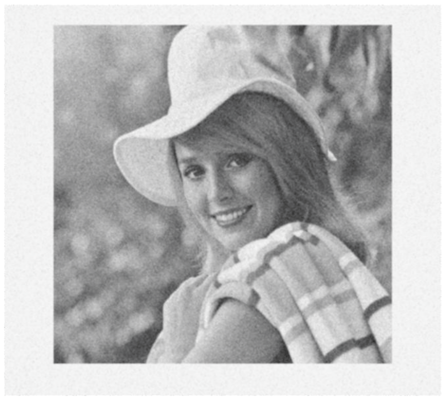
(e) (f)

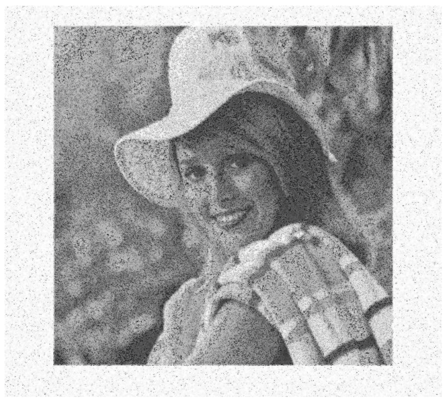
Figure 3.8: Point detection: (a) original image, (b) using exact adders, (c) using ESA with k=7 and PSNR=4.4dB, (d) using ESA with k=10 and PSNR=24.6dB, (e) using ETAII with k=4 and PSNR=10.5dB and (f) using ETAII with k=5 and PSNR=19.9dB.


(a)


(b)


(c)


(d)


(e)

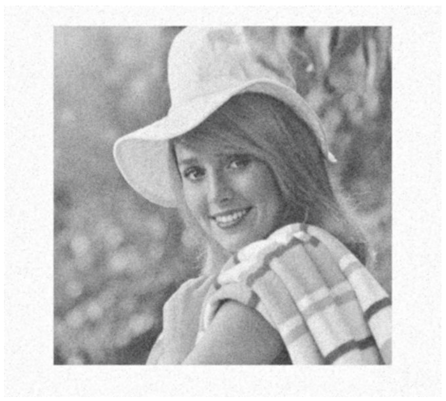

(f)

Figure 3.9: Arithmetic mean filter: (a) original image with noise, (b) using exact adders, (c) using ESA with k=5 and PSNR=10.5dB, (d) using ESA with k=8 and PSNR=27.2dB, (e) using ETAII with k=3 and PSNR=16.9dB and (f) using ETAII with k=5 and PSNR=36.3dB.

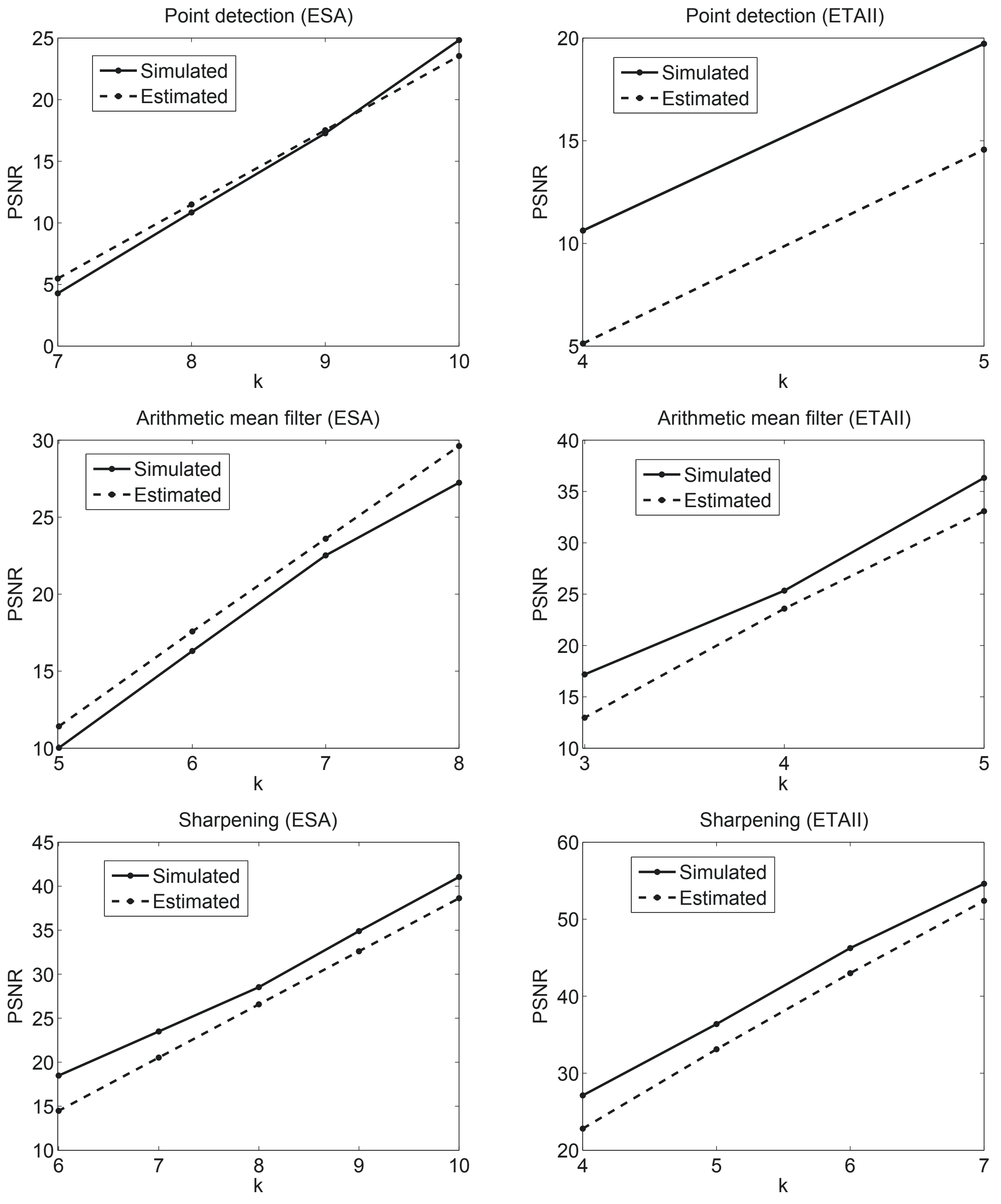


Figure 3.10: Simulated and estimated PSNR (dB).

3.3.3 Estimate of the PSNR for Image Processing

Three image processing algorithms are evaluated next: image sharpening, point detection and arithmetic mean filter.

If **I** is the original image and **S** is the processed image, the sharpening algorithm [19] is performed as

$$\mathbf{S}(x,y) = 2\mathbf{I}(x,y) - \tfrac{1}{273}\sum_{i=-2}^{2}\sum_{j=-2}^{2}\mathbf{G}(i+3,j+3)\mathbf{I}(x-i)(y-j), \tag{3.68}$$

where **G** is a matrix given by:

$$\mathbf{G} = \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}. \tag{3.69}$$

The point detector operation [31] is given by

$$\begin{aligned} &\mathbf{S}(x,y) = 8\mathbf{I}(x,y) - \mathbf{I}(x-1,y-1) - \mathbf{I}(x-1,y) - \mathbf{I}(x-1,y+1) \\ &-\mathbf{I}(x,y-1) - \mathbf{I}(x,y+1) - \mathbf{I}(x+1,y-1) - \mathbf{I}(x+1,y) - \mathbf{I}(x+1,y+1). \end{aligned} \tag{3.70}$$

A 3×3 arithmetic mean filter [31] is also implemented as

$$\mathbf{S}(x,y) = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1}\mathbf{I}(x+i,y+j). \tag{3.71}$$

For all three algorithms, the exact additions are replaced by approximate additions using ESA and ETAII. All other operations (i.e., multiplication, division and subtraction) are performed accurately. The mean squared error is estimated using (3.62), in which the number of approximate additions, a, is 25, 8 and 9, respectively for image sharpening, point detection and arithmetic mean filter applications. The PSNR is then given by (3.53).

In the sharpening algorithm, 16-bit approximate adders are used because the maximum possible sum is 255×273, i.e., approximately $2^{16} - 1$. 12-bit approximate adders are used for the point detection and arithmetic filer computations. The analytical $\sigma - \mu$ plots shown in Fig. 3.6 are used for the PSNR.

Six images are selected for the three algorithms and the corresponding PSNR values are obtained; three of them are shown in Figs. 3.7, 3.8 and 3.9. The images are selected such that they are quite "typical" images to be processed, i.e., they show features commonly found in multimedia applications. For the same algorithm and approximate adder, the six images have PSNR values that are very close; this indicates that the PSNR is not strongly correlated to an image, hence the estimated PSNR can be readily applied to them. For further analyzing this feature, define the *relative discrepancy* (RD) as the maximum difference between the PSNR values of the six images divided by their mean PSNR value. The RD values are shown in Table 3.5; the largest RD is 13.2%, however in most cases the RD is below 5%, i.e., at an acceptable level for this type of applications.

Table 3.5: Relative discrepancy.

Point detection						
	ESA				ETAII	
k	7	8	9	10	4	5
RD (%)	13.2	4.8	2.1	1.4	10.5	5.2

Arithmetic mean filter							
	ESA				ETAII		
k	5	6	7	8	3	4	5
RD (%)	9.9	5.0	2.7	1.7	5.0	2.0	0.8

Sharpening									
	ESA					ETAII			
k	6	7	8	9	10	4	5	6	7
RD (%)	5.7	5.0	3.4	1.7	0.9	2.1	1.7	2.6	4.1

The simulated and estimated PSNR values are shown in Fig. 3.10. The best match between simulated and estimated PSNR values is achieved by ESA-based point detection application, while the worst occurs for ETAII-based point detection. In the worst case, the maximum PSNR is about 20dB that is generally considered to be low. For most cases, the estimate tends to be less accurate when the PSNR values are too small ($< 20dB$). The interesting cases are the ones whose PSNR values are larger than 20dB, otherwise the approximate results are not acceptable. For these cases, the estimate is shown to be accurate in this PSNR range. Moreover, the difference between estimated and simulated results is within a 3dB margin; so for a given image processing application and approximate adder, the error in the estimate appears to be almost constant. Hence, the simulation results show that with the proposed methodology, the MED is a good indicator for the PSNR. The estimate procedure proposed in this chapter can be used to evaluate the performance of approximate adders in image processing applications.

3.4 Conclusion

In this chapter, an analytical framework has been proposed for characterizing approximate adder designs. This framework consists of models for the evaluation of different types of approximate adders targeting several error metrics. Time-consuming simulation can then be avoided by using the proposed analytical models. Design criteria with respect to error characteristics in the operations of these approximate adders have been provided based on the analysis. As an example of the application of the framework, the PSNR in image processing has been evaluated. The estimated PSNR can then be utilized for selecting the proper scheme of an approximate adder. Extensive simulation results show that there is a good agreement between the analytical outcomes of the proposed framework and the simulation results for three different computational algorithms commonly used in image processing. The PSNR estimate method proposed in this chapter shows that there is a close relationship between the MED and PSNR, while the ER is less important. This may provide insights into the design of approximate arithmetic circuits for error-tolerant applications.

Chapter 4

A Novel Approximate Multiplier Design with Configurable Error Recovery for Low-Power and High-Performance Operation

This chapter presents the design of a novel approximate multiplier. This novel design leverages a fast parallel approximate adder to accumulate the partial products and an error reduction scheme to configure the accuracy level of the multiplier. An optimal tradeoff of accuracy for performance and power can be achieved by configuring the error reduction circuit. Compared to existing approximate multiplier designs, the proposed design achieves the best tradeoff in terms of power-delay-error metric (PDEM).

Part of the results in this chapter has been published in [23]. The chapter is organized as follows. Section 4.1 presents the proposed approximate adder and the design of the multiplier. Section 4.2 discusses the error reduction schemes for AM1 and AM2, and introduces the accurate operation mode. Section 4.3 shows the accuracy analysis and in section 4.4, delay and power consumption are obtained. Section 4.5 discusses the application oif the proposed multiplier to image processing. Section 4.6 concludes the chapter.

4.1 The Approximate Multiplier

4.1.1 The Approximate Adder

In this section, the design of a new approximate adder is presented. This adder operates on a set of pre-processed inputs. The input pre-processing (IPP) is based on the *interchangeability* of bits with the same weights in different addends. For example, consider two sets of inputs to a 4-bit adder: i) $A = 1010, B = 0101$ and ii) $A = 1111, B = 0000$. Clearly, the additions in i) and ii) produce the same result. In this process, the two input bits $A_iB_i = 01$ are equivalent to $A_iB_i = 10$ (with i being the bit index) because of the interchangeability of the corresponding bits in the two operands.

The basic rule for the IPP is to switch A_i and B_i if $A_i = 0$ and $B_i = 1$ (for any i), while keeping the other combinations (i.e., $A_iB_i = 00, 10$ and 11) unchanged. By doing so, more 1's are expected in A and more 0's are expected in B. If $\dot{A}_i\dot{B}_i$ are the ith bits in the pre-processed inputs, the IPP functions are given by:

$$\dot{A}_i = A_i + B_i, \tag{4.1}$$

$$\dot{B}_i = A_iB_i. \tag{4.2}$$

(4.1) and (4.2) compute the propagate and generate signals used in a parallel adder such as the carry look-ahead (CLA). The proposed adder can process data in parallel by cutting the carry propagation chain. A carry propagation chain starts at the ith bit when $\dot{B}_i = 1, \dot{A}_{i+1} = 1, \dot{B}_{i+1} = 0$. In an accurate adder, S_{i+1} is 0 and the carry propagates to the higher bit. However, in the proposed approximate adder, S_{i+1} is set to 1 and an error signal is generated as $E_{i+1} = 1$. This prevents the carry signal from propagating to the higher bits. Hence, a carry signal is produced only by the generate signal, i.e., $C_i = 1$ only when $\dot{B}_i = 1$, and it only propagates to the next higher bit, i.e., the (i+1)th position. Table 4.1 shows the truth table of the approximate adder, where $\dot{A}_i, \dot{B}_i$ and $\dot{B}_{i-1}$ are the inputs after IPP, C_{i-1} is the carry signal, S_i and E_i are the sum and error bits, respectively. The error signal is utilized for error compensation purposes as discussed in a later section. In this case, the approximate adder is similar to a redundant number system [33] and the logical functions of Table 4.1 are given by

$$S_i = \dot{B}_{i-1} + \overline{\dot{B}_i}\dot{A}_i, \tag{4.3}$$

$$E_i = \overline{\dot{B}_i}\dot{B}_{i-1}\dot{A}_i. \tag{4.4}$$

By replacing $\dot{A}, \dot{B}$ using (4.1) and (4.2), the logic functions with respect to the original inputs are given by

$$S_i = (A_i \oplus B_i) + A_{i-1}B_{i-1}, \tag{4.5}$$

$$E_i = (A_i \oplus B_i)A_{i-1}B_{i-1}. \tag{4.6}$$

Consider as an example a 6-bit adder with two inputs given by $A = 001111$ and $B = 000110$. The correct (exact) sum S is 010101; however, the approximate adder produces the sum $S^{'} = 001101$ and an error $E = 001000$. It is easy to show that

$$S = S^{'} + E. \tag{4.7}$$

Note that in (4.7) '+' means the addition of two binary numbers rather than the 'OR' function. The error E is always non-negative and the approximate sum is always equal to or smaller than the accurate sum. This is an important feature of this adder because an additional adder can be used to add the error to the approximate sum as a compensation step. While this is intuitive in an adder design, it is particularly a useful feature in a multiplier design as only one additional adder is needed to reduce the error in the final product.

Table 4.1: Truth table of an approximate adder cell.

$\dot{B}_i\dot{B}_{i-1}$	00	01	10	11
$\dot{A}_i$	$\dot{A}_i$	$\dot{A}_i$	1	1
$C_{i-1}/\dot{B}_{i-1}$	0	1	0	1
S_i	$\dot{A}_i$	1	0	1
E_i	0	$\dot{A}_i$	0	0

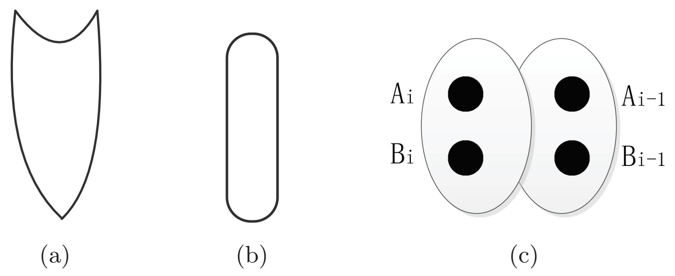


Figure 4.1: Symbols for (a) an OR gate, (b) an FA/HA and (c) an approximate adder cell.

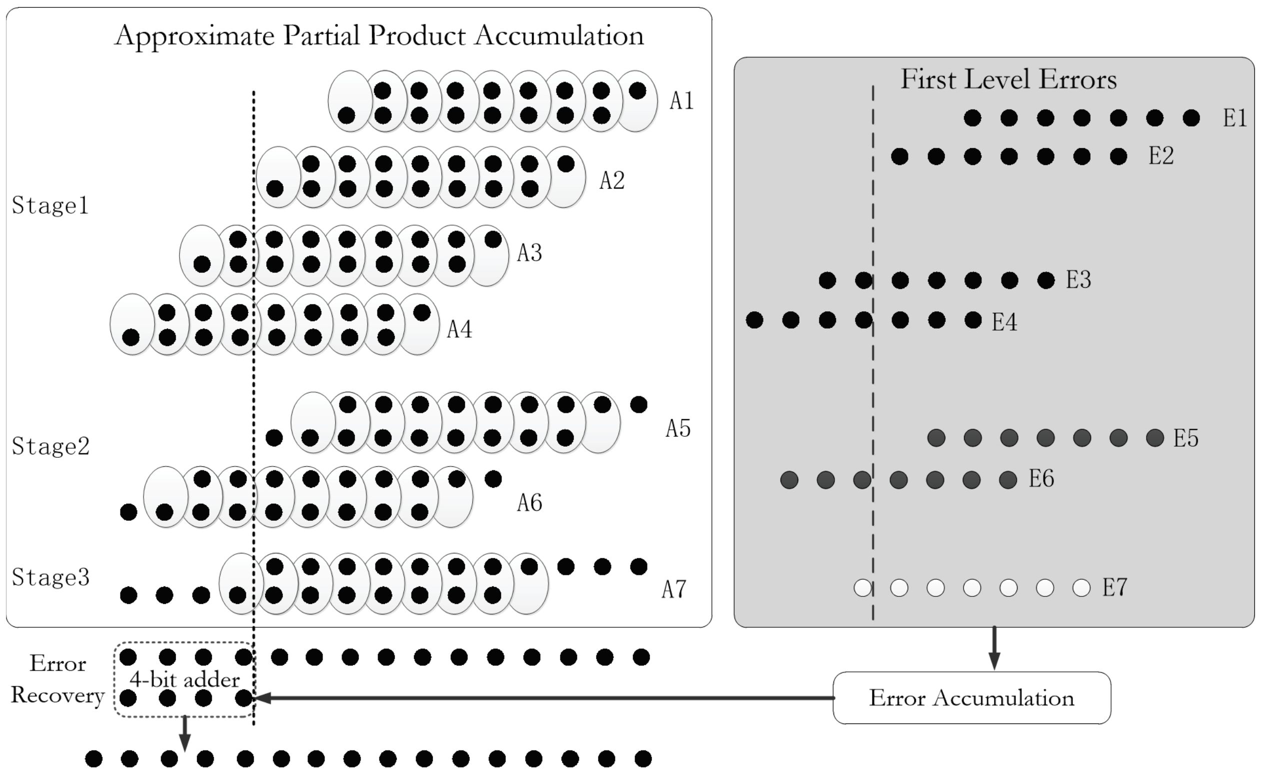


Figure 4.2: An approximate multiplier with OR-gate based partial error recovery using 4 MSBs of the error vector.

4.1.2 Proposed Approximate Multiplier

A significant feature of the proposed approximate multiplier is simply using approximate adders in the partial product accumulation. [17] has shown that this may lead to poor performance because errors may accumulate and it is difficult to correct errors using existing approximate adders. However, the use of the newly proposed approximate adder overcomes this problem by utilizing the error signal. The resulting design has a critical path delay that is shorter than a conventional one-bit full adder because the new n-bit adder can process data in parallel. The approximate adder has a rather high error rate but the feature of generating both the sum and error signals at the same time reduces errors in the final product. An adder tree is utilized for partial product accumulation. The error signals in the tree are then used to compensate the error in the output to generate a product with a better accuracy.

In the proposed approximate multiplier (Fig. 4.2), the simplification of the partial product accumulation stage is accomplished by using an adder tree in which the number of partial products is reduced by a factor of 2 at each stage of the tree. The newly proposed approximate adder is suitable for implementing an adder tree because it is less complex than a conventional adder and has a much shorter critical path delay.

Exact fast multipliers often include a Wallace or Dadda tree using full adders (FAs) and half adders (HAs); compressors are also utilized in the Wallace or Dadda tree to further reduce the critical path with an increase in circuit area. These designs require a proper selection of different circuit modules; for example, 4:2 compressors, FAs and HAs are commonly used in a Wallace tree and a judicious connection of these modules must be considered in a tree design. This increases the design complexity, especially when multipliers of different sizes are considered. The proposed design is simple for various multiplier sizes.

4.2 Error Reduction

The approximate adder generates two signals: the approximate sum S and the error E. The use of the error signal is considered next to reduce the inaccuracy of the multiplier. As (4.7) is applicable to the sum of every single approximate adder in the tree, an error reduction circuit is applied to the final multiplication result rather than to the output of each adder. Two steps are required to reduce errors: i) error accumulation and ii) error recovery by the addition of the accumulated errors to the adder tree output using an adder. In the error accumulation step, error signals are accumulated to be a single error vector, which is added to the output vector of the partial product accumulation tree. Two approximate error accumulation methods are proposed, yielding the approximate multiplier 1 (AM1) and approximate multiplier 2 (AM2). Two accurate error accumulation methods are also presented for an accurate operation. Fig. 4.1 shows the symbols for an OR gate, a full adder and half adder cell and an approximate adder cell used in the error accumulation tree.

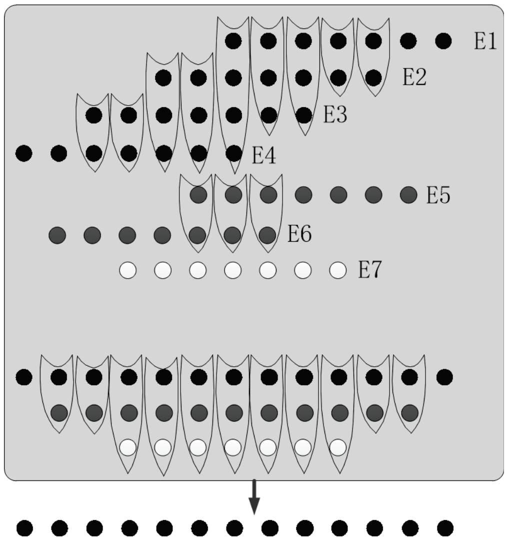


Figure 4.3: Error accumulation tree for AM1

4.2.1 Error Accumulation for Approximate Multiplier 1

If the error signals are added using accurate adders, the accumulated error can fully compensate the inaccurate product; however to reduce complexity, an approximate error accumulation is introduced. Consider the observation that the error vector of each approximate adder tends to have more 0's than 1's. Therefore, the probability that the error vectors have an error bit '1' at the same position is quite small. Hence, an OR gate is used to approximately compute the sum of the errors for a single bit. If m error vectors (denoted by $E1, E2, ..., Em$) have to be accumulated, then the sum of these vectors is obtained as

$$E_i = E1_i \ OR \ E2_i \ OR \ ... \ OR \ Em_i. \tag{4.8}$$

To reduce the error, an accumulated error vector is added to the adder tree output using a conventional adder (e.g., a carry look-ahead adder). However, only several (e.g., k) MSBs of the error signals are used to compensate the outputs and further reduce the overall complexity. The number of MSBs is selected according to the extent that errors must be compensated. For example in an 8×8 adder tree, there are a total of 7 error vectors generated by the 7 approximate adders in the tree. However, not all the bits in the 7 vectors need to be added because the MSBs of some vectors are less significant than the least significant bits of the k MSBs. In the example of Fig. 4.2, 4 MSBs (i.e., the 11-14th bits) are considered for error recovery and therefore, 4 error vectors are considered (i.e., the error vector E3, E4, E6 and E7). Note that the error vectors of the other three adders are less significant than the 11th bit, so they are not considered. The accumulated error E is obtained using (4.8); then, the final result is found by adding E to S using a fast accurate adder. The error accumulation scheme is shown in Fig. 4.3.

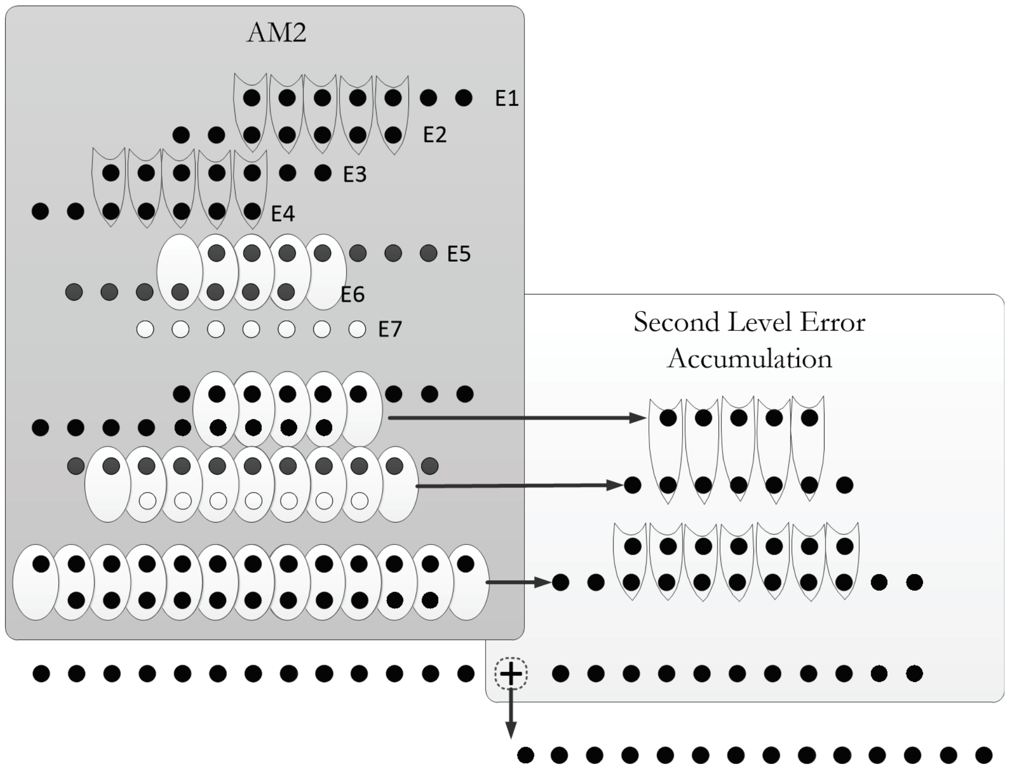


Figure 4.4: Error accumulation tree for ER2 (including AM2)

4.2.2 Error Accumulation for Approximate Multiplier 2

The error accumulation scheme for AM2 is shown in the shaded area marked as "AM2" in Fig. 4.4.

Error Characteristics in the First Stage of the Partial Product Accumulation Tree

To introduce the design of AM2, consider an 8×8 multiplier with two inputs X and Y. For simplicity, consider the first two partial products $X_0Y_7, X_0Y_6, ..., X_0Y_0$ and $X_1Y_7, X_1Y_6, ..., X_1Y_0$ accumulated by the first approximate adder (A1 in Fig. 4.2). Recall (4.6) for the approximate adder, the condition for $E_i = 1$ is

$$A_{i-1} = B_{i-1} = 1 \text{ and } A_i \neq B_i. \tag{4.9}$$

For the first approximate adder in the partial product accumulation tree, its input A_i is X_0Y_i and B_i is X_1Y_{i-1}. If X_0 or X_1 is 0, there will be no error in this approximate adder because either A or B is all-zero. To analyze the error, only $X_0X_1 = 11$ is considered. Under this condition, A_i is reduced to Y_i while B_i is reduced to Y_{i-1}. Then to calculate E_i, A_{i-1}, B_{i-1}, A_i and B_i are replaced by Y_{i-1}, Y_{i-2}, Y_i and Y_{i-1}, respectively. For E_i to be 1, $Y_iY_{i-2}Y_{i-1} = 011$ according to (4.9). Therefore, an error only occurs when the input has "011" as bit sequences. Based on this observation, the "distance" (in bits) between two errors in an approximate multiplier is at least 3. Thus, two neighboring approximate adders in the first stage of the partial product tree cannot have errors at the same bit because the errors in a lower approximate adder are those in the upper

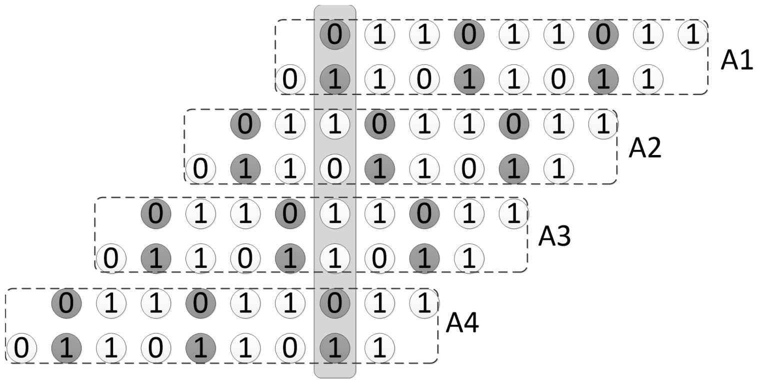


Figure 4.5: An example in which an error occurs when accumulating the errors using approximate adders

adder shifted by 2. The errors in two neighboring approximate adders can then be accurately accumulated by OR gates, e.g., an OR gate can be used for each bit to accumulate the two bits in the error vectors E1 and E2 in Fig. 4.2. After applying the OR gates to accumulate E1 and E2 as well as E3 and E4, the four error vectors are compressed into two.

The proposed approximate adders are then used to accumulate the remaining two error vectors in the first stage of the partial product accumulation tree. Recall that the proposed approximate adder is based on the assumption that the carry only propagates to one higher bit. We then enumerate all the cases for which two errors occur at the same bit (and thus generate a carry) in the two remaining error vectors, and find the cases when the carry propagates to more than 1 bit. The feature that an error only occurs when the input has a bit sequence as "011" significantly simplifies the enumeration. Finally, only 4 input combinations are found to generate a carry propagation length longer than two bits. These input combinations are $Y = (0)11011011, X = 11UU1111$, where 'U' means '0' or '1'. An example is shown in Fig. 4.5, where $Y = (0)11011011, X = 11111111$. In Fig. 4.5, the shaded circles show the error bits in each approximate adder (i.e., A1-A4). When a carry is generated and its higher bit has only one error, a carry propagation path longer than two is created and thus it introduces an error in accumulating these errors. The accumulation results by the approximate adder will always be correct except for these four inputs. Therefore the mean error magnitude caused by the approximate adders is extremely small in the error accumulation of the first stage. The error magnitude is 2^8 with 4 occurrences among 2^{16} input combinations, therefore the mean error magnitude is $4 \times 2^8/2^{16} = 1/64$.

Accumulation of Errors in Other Stages

First an interesting feature of the proposed approximate adder is introduced. Assume $E_i = 1$ in (4.6), then $A_{i-1} = B_{i-1} = 1$ and $A_i \neq B_i$. Since $A_{i-1} = B_{i-1} = 1$, i.e., $A_{i-1} \oplus B_{i-1} = 0$, it is easy to show that $E_{i-1} = 0$. Moreover as $A_i \neq B_i$, i.e., $A_iB_i = 0$, then $E_{i+1} = 0$. Thus, once there is an error in one bit, its neighboring bits are error-free, i.e., there are no consecutive error bits in one approximate adder. Consider next the two error vectors (i.e., $E5$ and $E6$) in the second stage.

$E5$ and $E6$ are accumulated by one approximate adder. Assume a carry is generated in the ith bit, i.e., $E5_i = E6_i = 1$, this carry only propagates to one higher bit because $E5_{i+1} = E6_{i+1} = 0$. Therefore, there is no carry propagation path longer than two bits and the error accumulation using the approximate adder is accurate. After the errors in the first and second stages are accumulated to be two vectors, another two approximate adders are then used to accumulate these two vectors as well as the error vector in the third stage. The errors introduced when accumulating these three vectors have been found by simulation rather than analysis. Simulation results (found in later sections) show that the modified error accumulation outperforms the OR-gate error accumulation with certain overhead on delay and power.

Hereafter, the proposed $n \times n$ approximate multiplier with k-bits OR-gate based error reduction will be referred to as a n/k AM1, while an $n \times n$ approximate multiplier with k-bit approximate adder based error reduction will be referred to as a n/k AM2.

4.2.3 Accurate Operation Mode

For accuracy in critical applications (in which no error can be tolerated), the proposed multiplier can operate in the accurate mode. In the approximate mode, either OR gates or approximate adders are used to accumulate the errors. Intuitively, an accurate adder tree such as the Wallace tree can be used to accumulate the errors; this can give an accurate accumulation of errors. The accumulated error can then be added back to the multiplication result. If the approximate mode uses AM2, then logic sharing is possible because the approximate adder calculates propagates and generates signals that are also used in the full adders. This Wallace tree based error recovery is referred to as "ER1", as shown in Fig. 4.6.

Another full error recovery method, referred to as "ER2", is also proposed for 8-bit approximate multipliers in particular. When accumulating errors using approximate adders, new errors will also be generated. These new errors are the 2nd level errors in Fig. 4.4. If the 2nd level errors are accumulated and then added back to the result, then full error recovery is achieved. Interestingly, for 8-bit approximate multipliers, three 2nd level error vectors introduced by three approximate adders in AM2 can be simply accumulated by OR gates with no loss of accuracy. This phenomenon was verified by exhaustive simulation. Thus, if every error bit in the first level errors is accumulated by approximate adders, and second level errors are accumulated by OR gates, the final result is accurate because both the accumulated first and second level errors are compensated.

The proposed multiplier can be configured to have both an approximate mode (either AM1 or AM2) and an accurate mode (either with ER1 or ER2). A multiplexer can be used to select an approximate or full error recovery. In the approximate mode, the approximate error reduction circuitry are activated, while the full error recovery circuitry are switched off to save power. The design of this multiplier is shown in Fig. 4.7

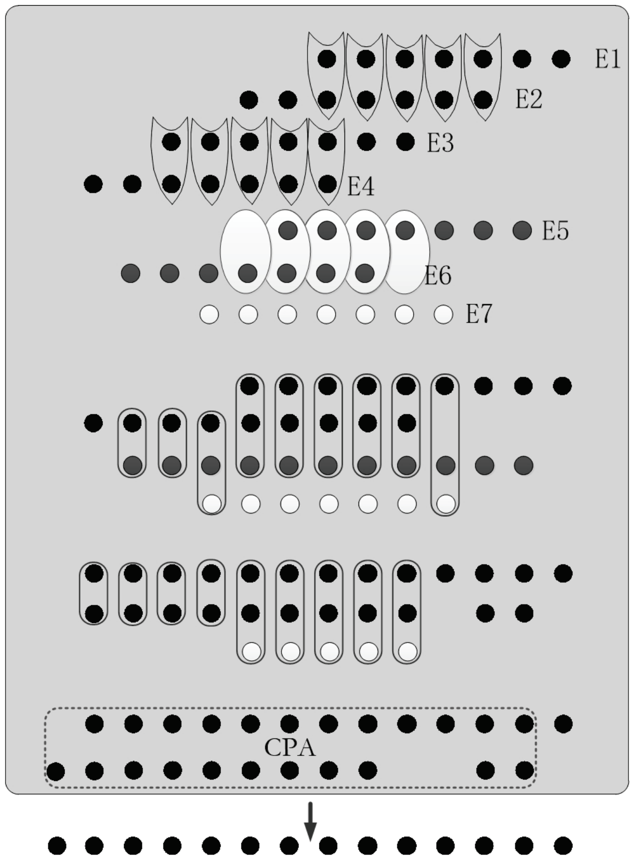


Figure 4.6: ER1: Accurate error accumulation using a Wallace tree

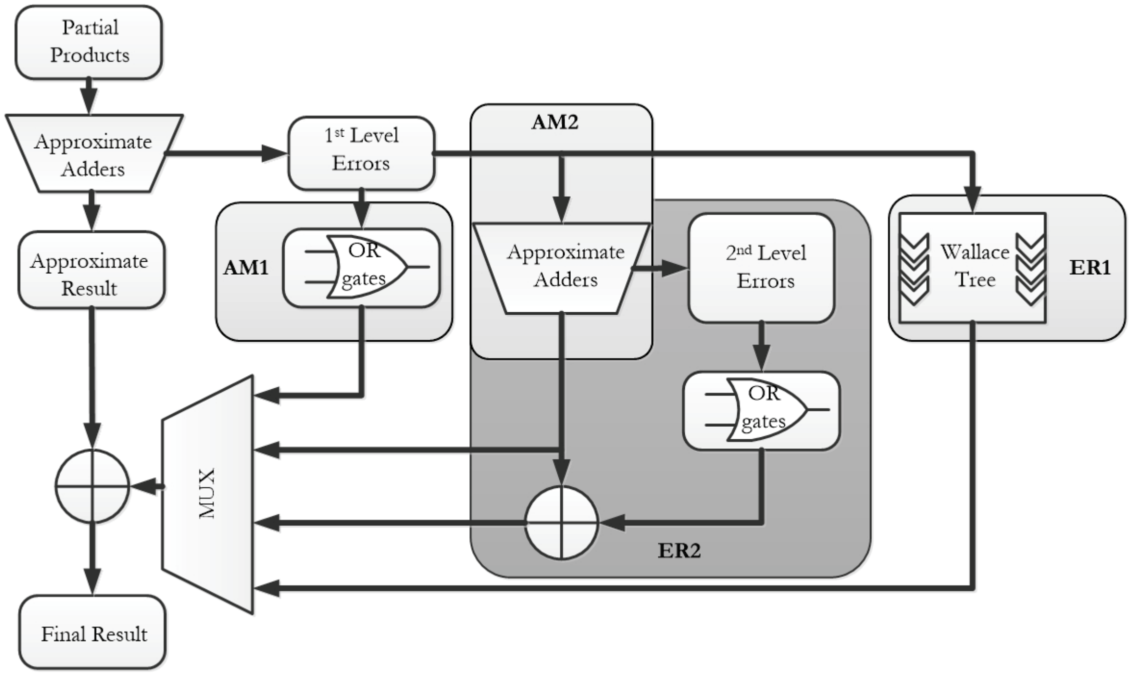


Figure 4.7: Block diagram of the proposed multiplier with four configurations: AM1, AM2, ER1, ER2.

4.2.4 Modifications for the 16×16 bit Approximate Designs

Both AM1 and AM2 compress all the error vectors to one error vector and then add that error vector to the approximate output of the partial product tree. Compared to 8×8 bit designs, 16×16 bit multipliers have more error vectors, therefore only one compressed error vector does not make a good estimation to the overall error. In the modified designs, two final error vectors are introduced. Let us take a 16×16 bit AM1 as an example. The eight error vectors in the first stage of the partial product accumulation tree are compressed to one error vector, EV1, using OR gates. The remaining seven error vectors from the second, third and fourth stages are compressed to the other error vector EV2. Then both EV1 and EV2 are added back to the output of the fourth stage partial product. In 16×16 bit approximate designs, both AM1 and AM2 use this modified error reduction method.

Inspired by ETM [18], BAM [27] and BBM [8], truncation can also be added to the proposed designs if the input operands are "large" enough. In AM1 and AM2, 16 LSBs of the final product are truncated, resulting in truncated AM1 (TAM1) and truncated AM2 (TAM2).

4.3 Accuracy Evaluation

Arithmetic accuracy in approximate circuits is compromised for improvements in other metrics (such as reduced circuit complexity and delay). In [20], the error distance (ED) and mean error distance (MED) are proposed to evaluate the performance of approximate arithmetic circuits. For multipliers with a given input combination, ED is defined to be the arithmetic difference between the accurate product (M) and the approximate product ($M^{'}$), i.e.,

$$ED = |M^{'} - M|. \tag{4.10}$$

MED is the expectation of ED (similar to the definition in (3.2)). A metric applicable for comparing multipliers of different sizes is the normalized MED (NMED), i.e.,

$$NMED = \frac{MED}{M_{max}}, \tag{4.11}$$

where M_{max} is the maximum magnitude of the output of an (accurate) multiplier, i.e. $(2^n - 1)^2$ for an $n \times n$ multiplier. The relative error distance (RED) is defined as:

$$RED = \frac{|M^{'} - M|}{M} = \frac{ED}{M}. \tag{4.12}$$

The mean relative error distance (MRED) can be obtained by averaging all the REDs corresponding to every input combination for an $n \times n$ multiplier:

$$MRED = E[RED] = 2^{-2n} \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^n-1} \frac{|M_{ij}^{'} - M_{ij}|}{M_{ij}}, \tag{4.13}$$

where $M_{ij}^{'}$ and M_{ij} are the corresponding approximate and accurate products for inputs $A = i$ and $B = j$. Note that when either one of the inputs is zero, the MRED is defined to be zero if the approximate product is zero, and is otherwise undefined (in which case the corresponding input combinations are avoided when calculating MRED). The error rate (ER) is defined as the percentage of erroneous outputs among all outputs [3]. These three metrics (NMED, MRED and ER) are used to evaluate the proposed multiplier. A functional model of the proposed multiplier is implemented using Matlab; and an exhaustive simulation is performed for an 8×8 bit approximate multiplier and a Monte Carlo simulation is performed for a 16×16 bit design.

4.3.1 Accuracy Evaluation of an 8×8 Bit Multiplier

Approximate multipliers with both the OR gate and the approximate adder based error reduction, as well as the accurate adder based error reduction are evaluated using exhaustive simulations. Fig. 4.8 shows the three metrics (NMED, MRED and ER) when using different numbers of MSBs for error reduction. Let m denote the number of MSBs used for error reduction. The NMED drops drastically as m is increased from 6 to 10 and continues to drop as m increases, even though at a slower rate. For the approximate multipliers, there is no error in the most significant bit of the output, so the largest number of MSBs used is 15. The NMED finally drops to zero for the exact error accumulation when 15 MSBs are used for error reduction, whereas there is still a non-zero NMED for both AM1 and AM2. Both the MRED and ER also decrease as m is increased, and drop to zero if 15 MSBs are used for an exact error reduction or non-zero if OR gates or the approximate adders are used for approximate error accumulation.

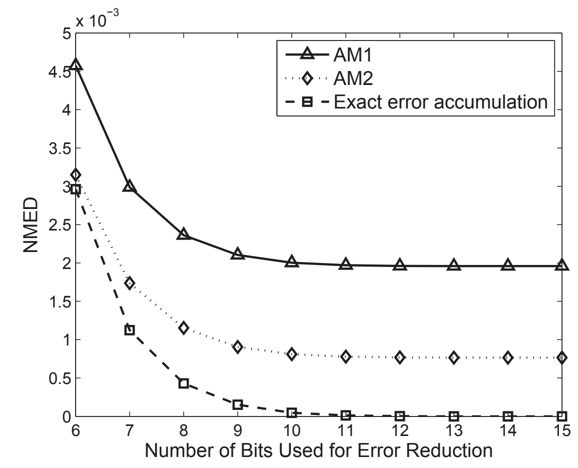
AM2 has a better performance than AM1 in terms of NMED, MRED and ER, i.e., it has lower values for these three metrics. For example, if 10 MSBs are used for error reduction, the NMED of AM2 is 0.08% while it is 0.2% for AM1. Moreover, if 15 MSBs are used for error reduction, AM1 has an error rate of 17.6%, while the error rate of AM2 can be as low as 5.8%.

These three figures also indicate that the proposed approximate multiplier has a rather high error rate, but the errors are usually very small compared to both the accurate and the largest possible output of the approximate multiplier. For example, for m=7, the error rate of AM1 can be as high as 62%, but the MRED is only 1.8%, i.e., most of the errors are not significant.

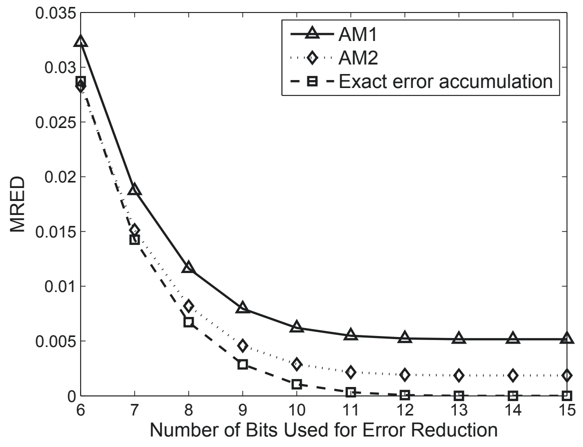
4.3.2 Accuracy Evaluation of a 16×16 Bit Multiplier

Fig. 4.9 shows the Monte Carlo simulation results for 16×16 bit designs of AM1, TAM1, AM2 and TAM2 with 10^8 random inputs. With a larger number of bits for error reduction, the error decreases. However, it is still true that AM2/TAM2 has a better accuracy than AM1/TAM1. Another observation is that AM1/AM2 has a better accuracy than TAM1/TAM2, as expected.

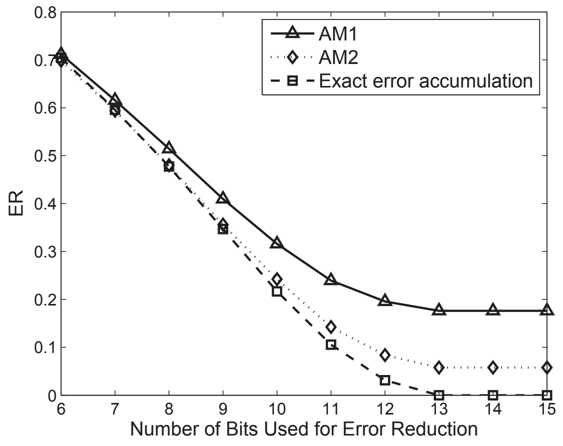
AM1/AM2 has a smaller NMED than TAM1/TAM2 however the difference is very small. This is because truncation of several LSBs does not affect the overall NMED very much. Yet for MRED, we can see that the difference between AM1/AM2 and TAM1/TAM2 becomes more significant because the relative error is more easily affected by truncation. All these four approximate designs have high ERs (98%-100%), and TAM1/TAM2 results in nearly an ER of 100%. This is not surprising since 16×16 bit designs generate more error bits than 8×8 designs, and the truncation



(a)

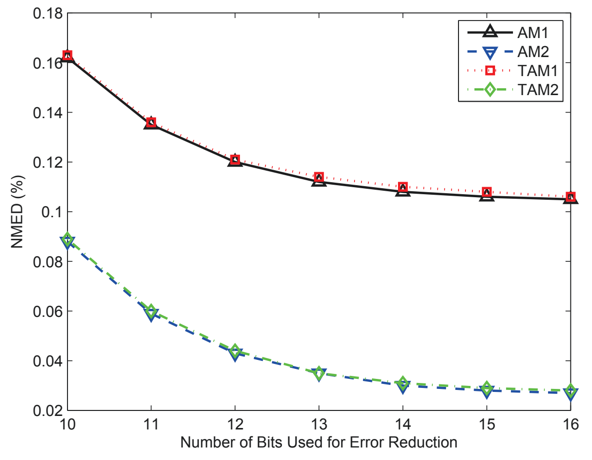


(b)

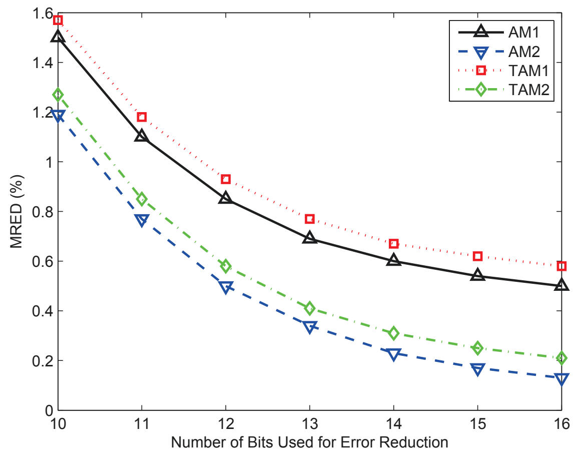


(c)

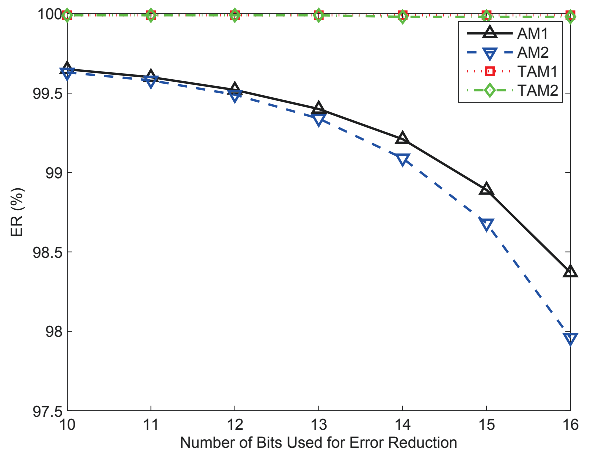
Figure 4.8: Accuracy comparison of the approximate multiplier using approximate and exact error accumulation: (a) NMED, (b) MRED and (c) ER, vs. the number of bits used for error reduction.



(a)

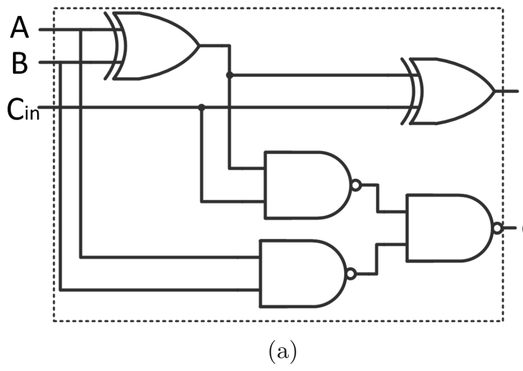


(b)



(c)

Figure 4.9: Accuracy comparison of the approximate multipliers : (a) NMED, (b) MRED and (c) ER, vs. the number of bits used for error reduction.



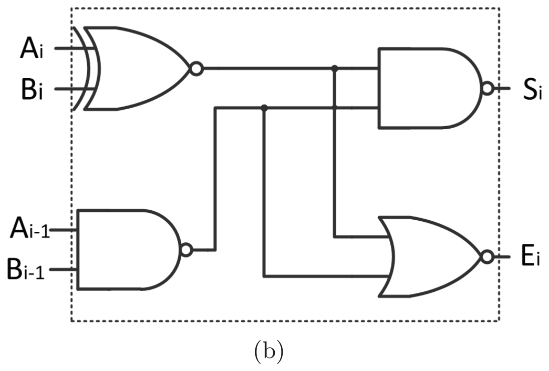


Figure 4.10: (a) An exact full adder and (b) the approximate adder cell.

even generates more errors. However the NMED and MRED are still kept very small.

4.4 Delay and Power Evaluation

4.4.1 Delay Estimation

Based on the linear model of [40], the delays of a full adder (Fig. 4.10 (a)) and the approximate adder cell (Fig. 4.10 (b)) are derived to be approximately $3\tau_g$ and $2\tau_g$, respectively, where τ_g is an approximate "gate delay". For an n-bit approximate multiplier, there are $\log_2 n$ stages in the adder tree. A stage with 2^m rows of partial products are compressed to 2^{m-1} rows of partial products and 2^{m-1} error vectors. These error vectors are then compressed (i.e., accumulated) using OR gates or approximate adders in a similar tree structure. Since the numbers of rows in the new partial product stage and the generated error stage are the same, it takes $m-1$ steps for both stages to be compressed to 1. Thus, when an n-row partial product tree is compressed to 1 row, errors from $\log_2 n$ stages are also compressed to $\log_2 n$ error vectors. Then it takes $\lceil \log_2 \log_2 n \rceil$ stages to finally compress these $\log_2 n$ error vectors. Therefore, the delay of the proposed approximate multiplier is considered to be the delay of compressing the partial product tree and the delay to compress the remaining $\log_2 n$ error vectors, i.e.,

$$D_{AMi} = (\log_2 n) \times 2\tau_g + \lceil \log_2 \log_2 n \rceil \times \tau_i, \tag{4.14}$$

where $\tau_i = \tau_g$ (the delay of an OR gate for AM1) for $i = 1$ and $\tau_i = 2\tau_g$ (the delay of an approximate adder for AM2) for $i = 2$.

There are $\lfloor \log_{1.5} n \rfloor$ stages in the Wallace tree and the delay is given by [2]

$$D_W = 3 \lfloor \log_{1.5} n \rfloor \tau_g. \tag{4.15}$$

Table 4.2 shows the delay of the partial product accumulation tree in both the proposed and Wallace multipliers. For a 16-bit multiplier, the delay of an exact multiplier tree is twice as large as the delay of the proposed multiplier tree; as the size of the multiplier increases, this factor is approximately 2.5. Since the approximate adder cell is simpler than a full adder, the approximate multiplier has no additional area overhead to achieve the shorter delay.

Table 4.2: Delay of partial product accumulation tree of the proposed and conventional multipliers of different sizes.

n	8	16	32	64	2^k
$D_{AM1}(\tau_g)$	8	10	13	15	$2k + \log_2 k$
$D_{AM2}(\tau_g)$	10	12	16	18	$2k + 2\log_2 k$
$D_{W,D}(\tau_g)$	12	18	24	30	$\approx 5k$

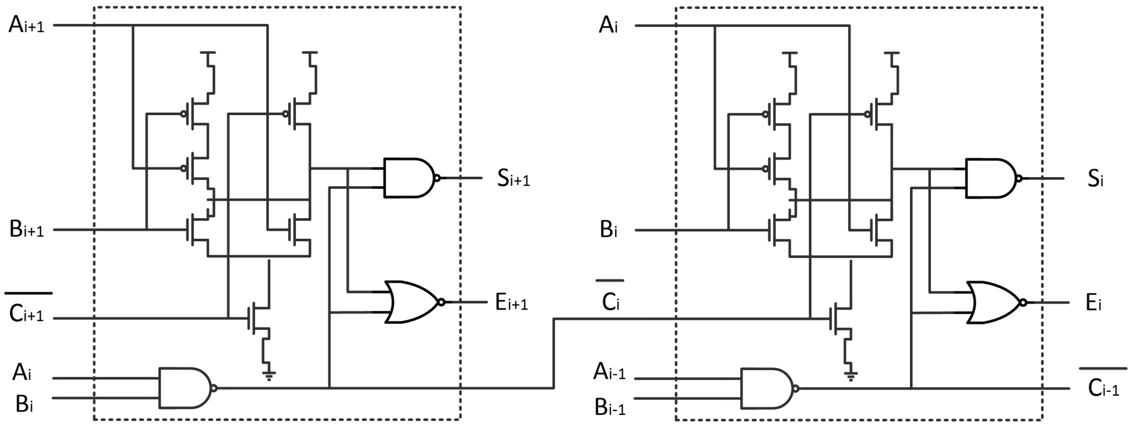


Figure 4.11: Two neighboring approximate adder cells for ASIC implementation.

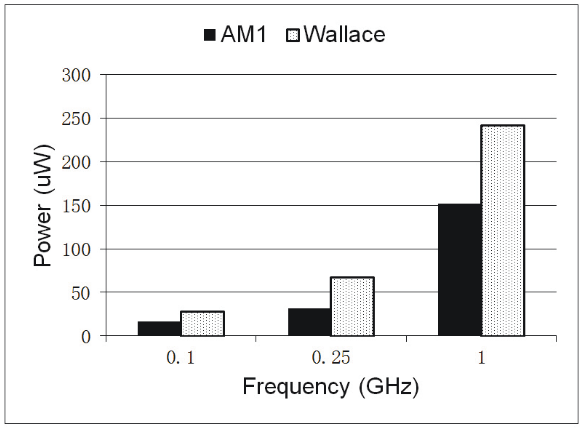
For the UDM in [17], only the partial product generation stage is simplified, so the delay reduction is quite limited. The ETM in [18] can reduce the $n \times n$ partial product tree to $\frac{n}{2} \times \frac{n}{2}$. By (4.15), the difference between the delays of $n \times n$ and $\frac{n}{2} \times \frac{n}{2}$ trees is approximately $3log_{1.5}2\tau_g \approx 5.13\tau_g$. In summary, the other two multipliers reduce the critical path delay by almost a constant value. In contrast, the proposed multiplier can reduce the delay of the partial product accumulation tree by nearly 60% , which scales with the size of the multiplier.

4.4.2 Simulation Results of Implementations in a 28-nm Process

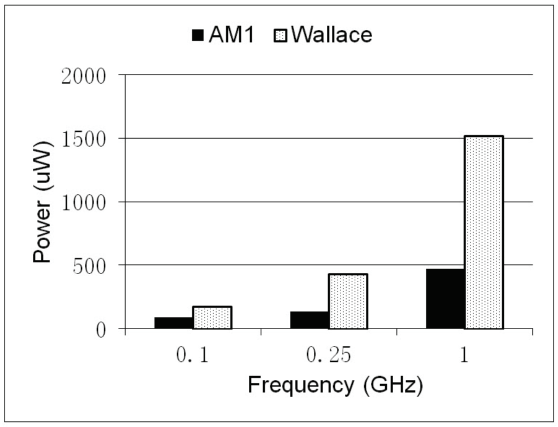
ASIC designs for $n \times n$ ($n = 8, 16$) AM1 with n-bit OR-gate based error reduction, a 8×8 AM2 with 8-bit approximate adder based error reduction, and $n \times n$ ($n = 8, 16$) Wallace multipliers have been implemented in STM 28-nm CMOS process. The approximate adder cell in Fig. 4.10 (b) is implemented using shared logic between two neighboring approximate adder cells, thereby saving additional area, as shown in Fig. 4.11. In Fig. 4.11, the signal $\overline{C_i}$ is given by $\overline{C_i} = \overline{A_i B_i}$ and shared between two cells. The corresponding FA and HA cells in the Wallace multiplier are taken from the 28-nm library as C12T32_LR_FA1×8 and C12T32_LR_HA1×8, respectively.

The delays for an 8×8 AM1, AM2 and Wallace multiplier are 0.51ns, 0.54ns and 0.58ns, respectively. Thus the delay improvements of AM1 and AM2 compared to the conventional Wallace multiplier are 12% and 7%. When considering a larger size, the critical path delays of a 16×16 AM1 and the Wallace multiplier are 0.48ns and 0.6ns, respectively, resulting a delay reduction of 20%. A larger delay reduction can be expected if the multiplier size is larger.

The power consumption for image multiplication is obtained by applying three frequencies (0.1



(a)



(b)

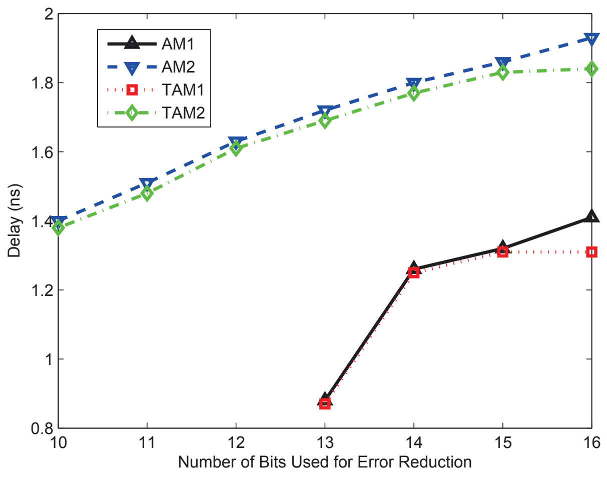
Figure 4.12: Power vs. frequency for (a) 8-bit and (b) 16-bit approximate and Wallace multipliers.

GHz, 0.25 GHz and 1 GHz) to all these multiplier circuits. For the 8×8 multipliers, both AM1 and AM2 have lower power consumption compared to a Wallace multiplier at these three frequencies, and AM2 consumes more power than AM1. The power savings for AM1 and AM2 at these three frequencies are in the ranges of 37%-53% and 16%-29%. For 16×16 multipliers, AM1 achieves power saving of 48%-69% compared to an accurate Wallace multipliers at the three frequencies. The power savings also increase with the multiplier size, as shown in Fig. 4.12 for both the 8×8 and 16×16 bit multipliers.

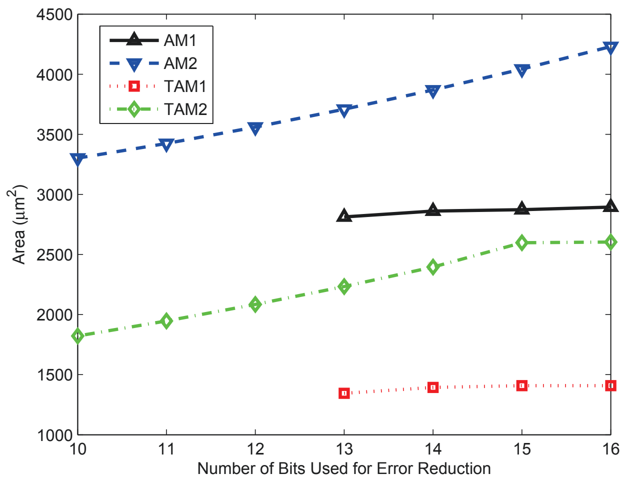
4.4.3 Simulation Results from Synopsys Design Vision in a 65-nm Process

16×16 bit AM1, AM2, TAM1 and TAM2 are implemented in VHDL and their delay, area and power reports are obtained from Synopsys Design Vision based on STM 65-nm libraries (Table 4.3). The delay, area and power values of four designs are plotted separately in Fig. 4.13 as well.

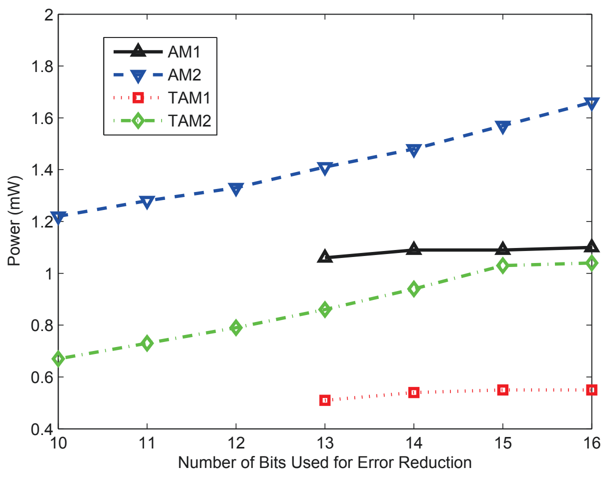
As shown in the table and figures, AM2/TAM2 has larger delay, area and power than AM1/TAM1 at the same number of bits for error reduction. Another observation is that TAM1/TAM2 has s-maller delay, area and power compared to AM1/AM2. TAM1/TAM2 has significantly reduced area and power compared to AM1/AM2, however the reduction in delay is rather limited for most of the cases. It is also shown in Table 4.3 that AM1 (TAM1) with 10-13 bits for error reduction have the same delay, area and power values. Therefore AM1 (TAM1) with 10-12 bits for error reduction are not interesting because their errors are larger than AM1 (TAM1) with 13 bits for error reduction but with no advantage in delay, area and power. Thus they are crossed out in Table 4.3 and not shown in Fig. 4.13.



(a)



(b)



(c)

Figure 4.13: Delay, area and power simulation results for AM1, AM2, TAM1 and TAM2

Table 4.3: Delay, area and power of proposed approximate multipliers obtained from Synopsys Design Vision

Parameter	Delay (ns)	Area (um^2)	Power (mW)	Parameter	Delay (ns)	Area (um^2)	Power (mW)
AM1 Parameter=NO. of MSBs for Error Reduction				TAM1 Parameter= NO. of MSBs for Error Reduction			
~~10~~	~~0.88~~	~~2812~~	~~1.06~~	~~10~~	~~0.87~~	~~1345~~	~~0.51~~
~~11~~	~~0.88~~	~~2812~~	~~1.06~~	~~11~~	~~0.87~~	~~1345~~	~~0.51~~
~~12~~	~~0.88~~	~~2812~~	~~1.06~~	~~12~~	~~0.87~~	~~1345~~	~~0.51~~
13	0.88	2812	1.06	13	0.87	1345	0.51
14	1.26	2861	1.09	14	1.25	1394	0.54
15	1.32	2873	1.09	15	1.31	1408	0.55
16	1.41	2894	1.10	16	1.31	1408	0.55
AM2 Parameter= NO. of MSBs for Error Reduction				TAM2 Parameter= NO. of MSBs for Error Reduction			
10	1.40	3302	1.22	10	1.38	1822	0.67
11	1.51	3424	1.28	11	1.48	1947	0.73
12	1.63	3560	1.33	12	1.61	2084	0.79
13	1.72	3709	1.41	13	1.69	2233	0.86
14	1.80	3869	1.48	14	1.77	2396	0.94
15	1.86	4043	1.57	15	1.83	2598	1.03
16	1.93	4230	1.66	16	1.84	2603	1.04

4.5 Image Processing Applications

4.5.1 Image Processing with Proposed Multipliers

Approximate circuits can be used in error-tolerant applications such as image processing; image sharpening and smoothing applications are studied next. Since multiplication is the arithmetic operation under investigation, an accurate multiplier is replaced by the proposed approximate multipliers (i.e., AM1 and AM2). All other processing steps (such as addition) are kept accurate.

The sharpening algorithm of [19] is simulated using both exact and approximate multipliers (i.e., AM1 and AM2). In the results shown in Fig. 4.14, approximate multipliers with different numbers of bits for error reduction are evaluated and an improvement in performance is achieved when the number of bits for error reduction is increased. The degradation in image quality is evident when 4 bits are used for error reduction for both AM1 and AM2. However, for a 8-bit error reduction in AM1 and AM2, there is no visually distinguishable difference with the exact sharpening result.

The image smoothing algorithm is given by [31]:

$$Y(x,y) = \frac{1}{60} \sum_{m=-2}^{2} \sum_{n=-2}^{2} X(x-m, x-n) Mask(m,n), \tag{4.16}$$

where X is the input image and Y is the output smoothed image, and Mask is a 5×5 matrix given

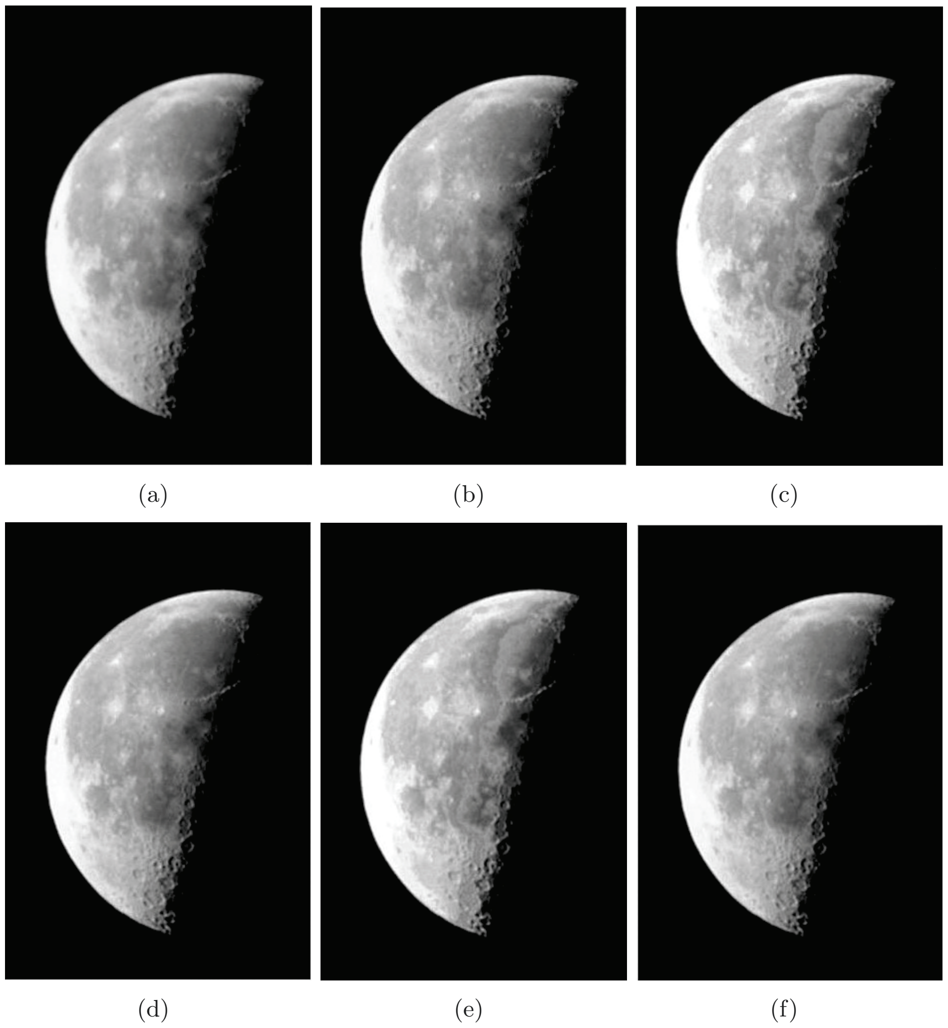


Figure 4.14: Image sharpening (a) original blurred image (b) using an accurate multiplier (c) using an 8/4 AM1 (d) using an 8/8 AM1 (e) using an 8/4 AM2 and (f) using an 8/8 AM2.

by:

$$Mask = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 4 & 4 & 1 \\ 1 & 4 & 12 & 4 & 1 \\ 1 & 4 & 4 & 4 & 1 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}.$$

The smoothing results are shown in Fig. 4.15. Similar to the sharpening application, an AM1 or AM2 with 8-bit error reduction has a visually equivalent performance to an accurate multiplier.

The peak signal-to-noise ratio (PSNR) is used to compare the difference between the images obtained by the accurate and approximate multiplications. Table 4.4 shows the PSNR values with respect to different numbers of bits for error reduction in the proposed multiplier. For example, the resulting image by an 8/8 AM1 has a PSNR of 49.37 dB for image sharpening and 51.56 dB for image smoothing; this is generally considered to be a good match with the accurately processed

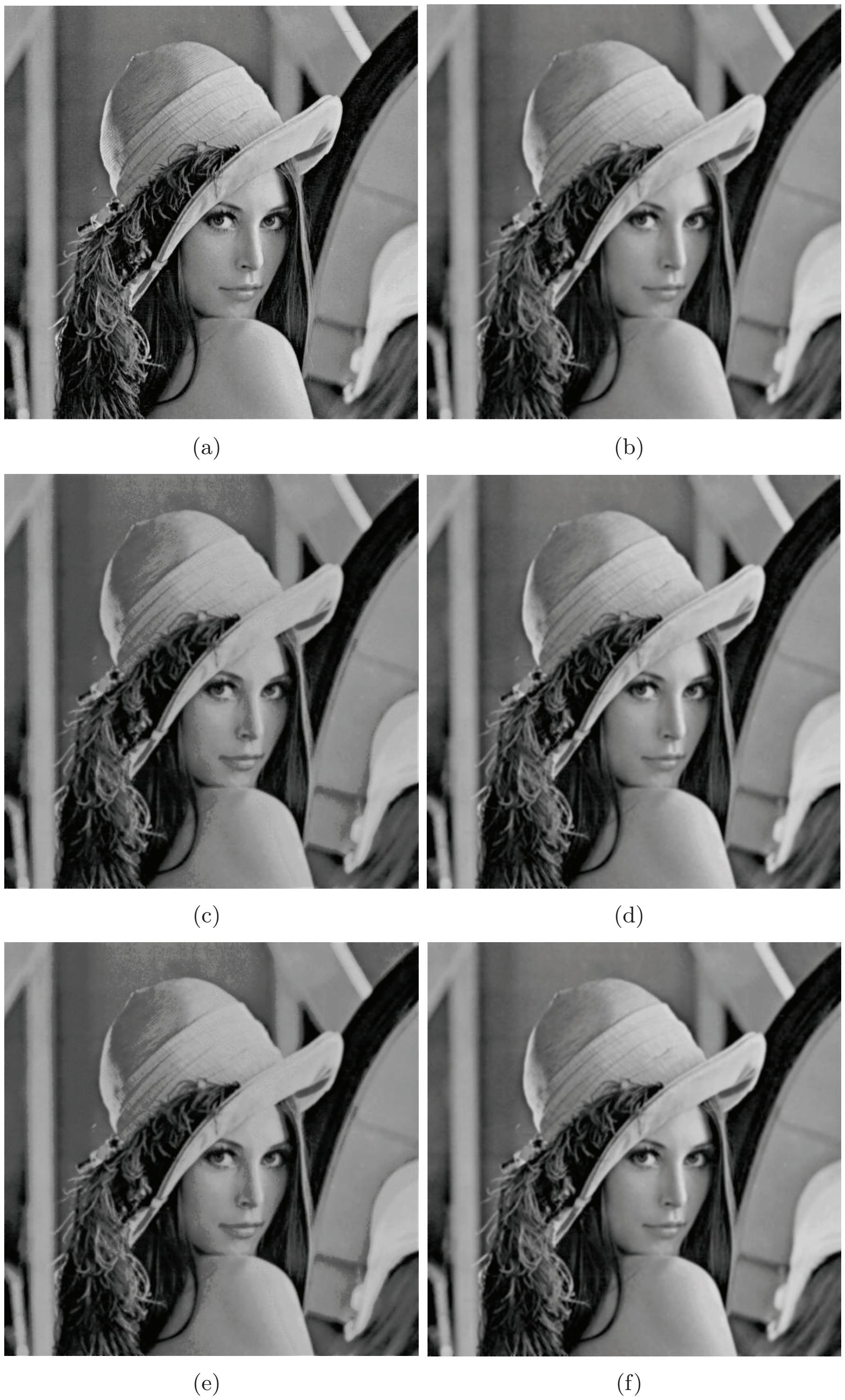
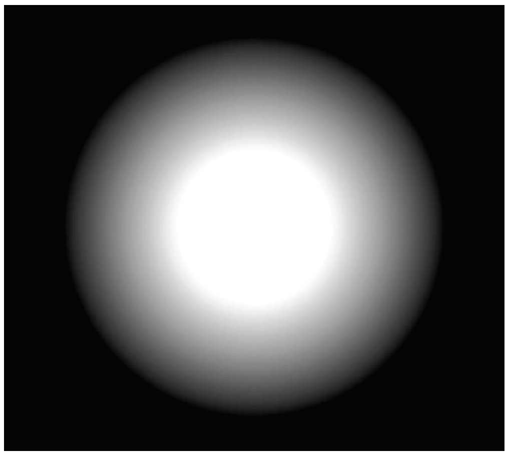
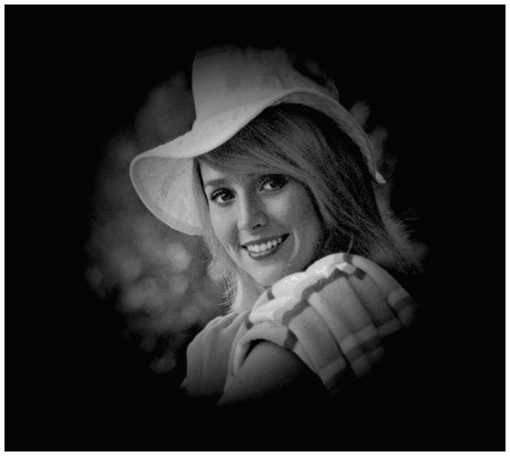
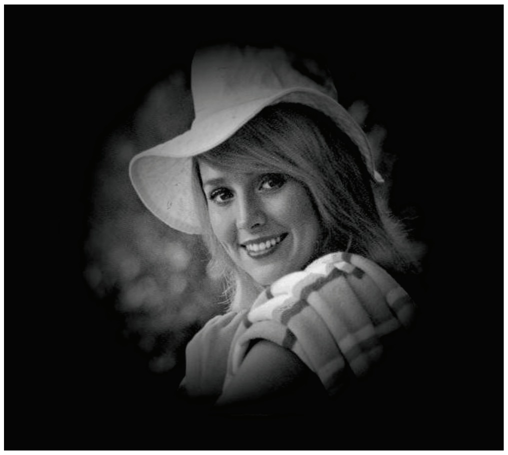


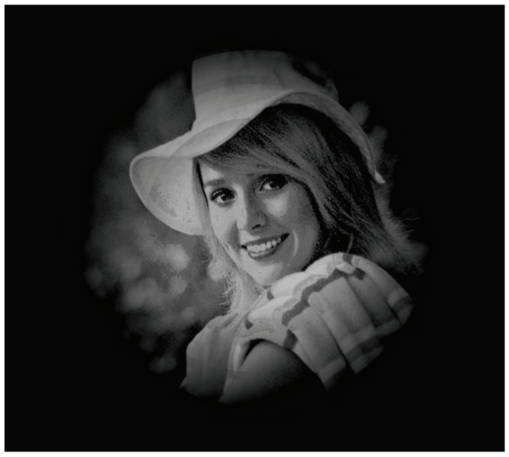
Figure 4.15: Image smoothing (a) original image (b) using an accurate multiplier (c) using an 8/4 AM1 (d) using an 8/8 AM1 (e) using an 8/4 AM2 and (f) using an 8/8 AM2.

(a)


(b)


(c)


(d)


(e)

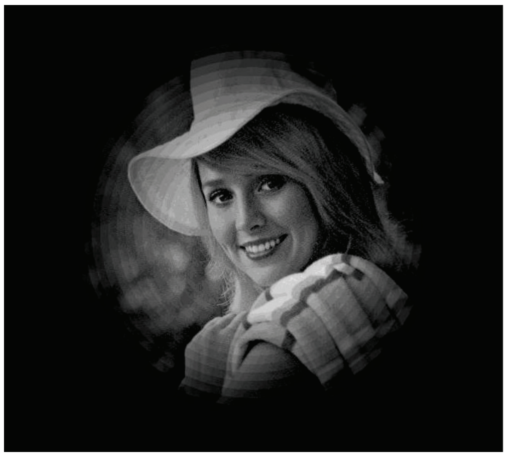

(f)

Figure 4.16: Image multiplication. (a) original image 1, (b) original image 2, (c) result by an 8/8 AM1, (d) result by an 8/8 AM2, (e) result by UDM and (f) result by ETM with 4 MSBs as multiplication part.

Table 4.4: PSNR (dB) of image processing applications for AM1 and AM2.

	Image Sharpening			Image Smoothing		
# of bits for error reduction	4	6	8	4	6	8
AM1	29.08	37.13	49.37	30.64	40.39	51.56
AM2	29.08	37.13	50.56	30.64	40.39	51.56

Table 4.5: PSNR (dB) of image multiplication of 4 different approximate multipliers

8/8 AM2	8/8 AM1	UDM [17]	ETM [18]
46.48	41.93	32.99	30.67

image. Since the result of an approximate multiplication is then processed by an accurate division for both image sharpening and smoothing applications, the error in the approximate multiplication is attenuated. Therefore, the differences in the PSNRs for AM1 and AM2 are very small and, thus, difficult to be observed by the 2-digit accuracy. However, there is a 1dB difference between the PSNRs for AM1 and AM2 with 8-bit error reductions for the image sharpening application.

4.5.2 Comparison with Existing Approximate Multipliers

To evaluate the performance of each approximate multiplier, image multiplication is selected because it directly employs multiplication without any other operations. The multipliers used in image multiplication use exactly the same configuration as those ASIC implementations described previously, i.e., AM1 and AM2 with 8-bit error reduction and ETM with 4 MSB multiplication part. The resulting images are shown in Fig. 4.16 while the corresponding PSNRs are shown in Table 4.5. AM2 achieves the highest PSNR, while ETM has the lowest. This is consistent with the NMED and MRED trends. The PSNRs of AM2 and AM1 are significantly higher than those of both UDM and ETM. As shown in Fig. 4.16, the resulting images by ETM and UDM show a reduction in quality, while there is no visible flaw for both images processed by AM2 and AM1.

4.6 Conclusion

This chapter has proposed a novel approximate partial product accumulation tree for a multiplier design using a newly designed approximate adder. OR-gate based and approximate adder based error reduction schemes are used, yielding two different approximate multiplier designs: AM1 and AM2. Both AM1 and AM2 have been shown to be faster with a lower power dissipation than the exact Wallace multiplier. Functional analysis has shown that on a statistical basis, the proposed multipliers have very small error distances and thus, they achieve a high accuracy. Simulation has also shown that AM2 has a higher accuracy compared to AM1 at the cost of a longer delay and a higher power consumption. The application of the proposed multipliers to image sharpening and smoothing has shown that the proposed designs are very competitive in performance with the accurate counterpart.

Chapter 5

A Comparative Evaluation of Approximate Multiplier Designs

In this chapter, the proposed designs are compared with approximate multiplier designs from the literature in terms of error metrics (NMED, MRED and ER) and circuit metrics (delay, power and area). Both the 8×8 and 16×16 bit multipliers are considered in this comparison.

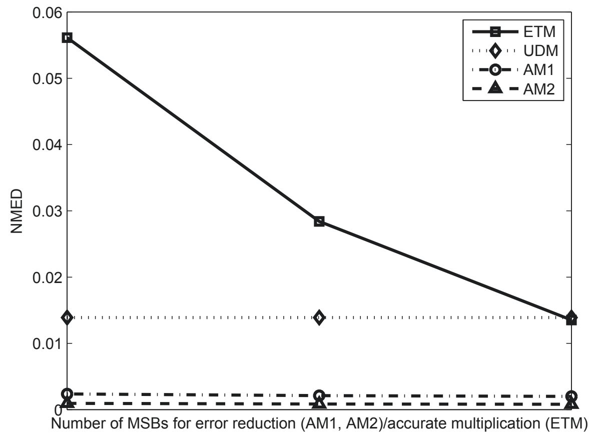
5.1 Comparison of 8×8 Bit Approximate Multipliers

5.1.1 Error Characteristics

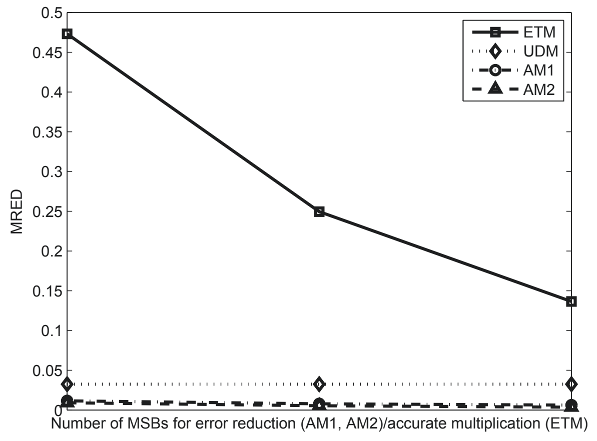
Both an 8×8 AM1 and AM2 are compared with two other approximate multipliers with the same size: the ETM in [18] and the underdesigned multiplier (UDM) in [17], as illustrated in Fig. 5.1. In this comparison, ETM with 5, 4 and 3 MSBs as the accurate multiplication part is considered, as shown in each of the figures in Fig. 5.1, from left to right. The three points from left to right in these figures for both AM2 and AM1 show the corresponding data for 8-bit, 10-bit and 12-bit MSBs for error reduction. There is only one configuration for UDM, so the values for it are all constant.

Among these four multipliers, ETM has the worst accuracy in terms of NMED, MRED and ER. This is not surprising because ETM uses a simple partition scheme. However, it is a straightforward design and as reported in [18], it saves nearly 90% of the power. The proposed multipliers, AM1 and AM2, have significantly smaller NMED and MRED compared to the other two multipliers. The ERs of AM1 and AM2 are slightly higher than UDM for the 8-bit error reduction case. However, the proposed multipliers start to show better performance in all three metrics when at least a 10-bit error reduction is employed.

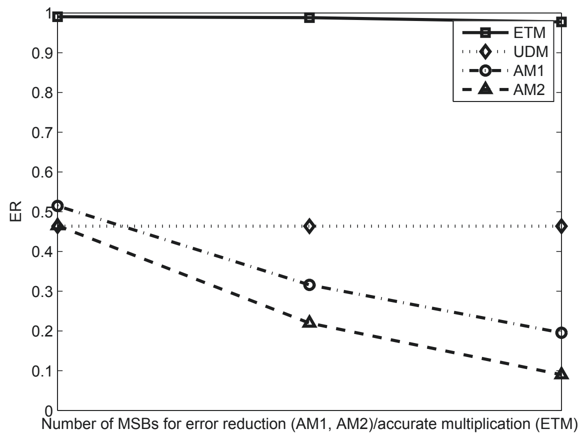
As an example, a more detailed comparison is shown in Table 5.1. In this comparison, the ETM is divided equally into multiplication and non-multiplication parts, while the proposed multipliers use 8 MSBs for error reduction. AM2 has the lowest NMED and MRED, while AM1 is the next. The proposed multipliers have a slightly higher error rate than UDM, but the error rates can be reduced using more MSBs for error reduction.



(a)



(b)

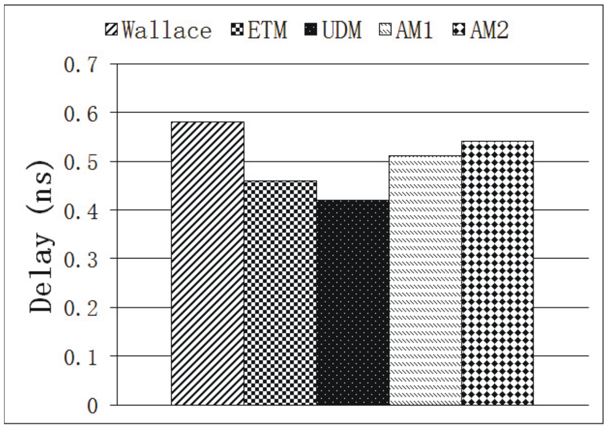


(c)

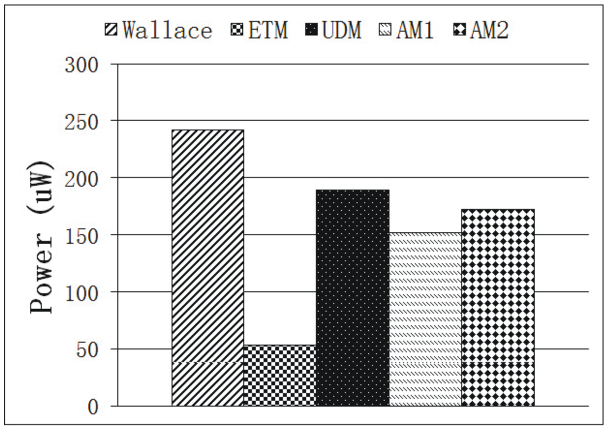
Figure 5.1: Comparison of accuracy among four approximate multipliers (x axis has no definition for UDM since it has only one type of configuration).

Table 5.1: Arithmetic accuracy comparison between three approximate multipliers.

	8/8 AM1	8/8 AM2	ETM [18]	UDM [17]
NMED (%)	0.24	0.12	2.85	1.39
MRED (%)	1.16	0.82	25.21	3.25
ER (%)	51.40	48.00	98.88	46.73



(a)



(b)

Figure 5.2: Delays and power consumptions of different multipliers (@1GHz)

5.1.2 Circuit Characteristics

An 8×8 bit Wallace multiplier, AM1 and AM2 with 8-bit error reduction, UDM and ETM with a 4 MSB multiplication part are implemented in STM 28-nm CMOS process. By applying 20 random input combinations to every circuit, the largest delays of the multipliers are reported in Fig. 5.2 (a). UDM is the fastest while the accurate Wallace multiplier has the longest delay. AM1 has a slightly longer delay than ETM, and AM2 is slightly slower than AM1.

The input data extracted from the two images are applied to each multiplier at a 1GHz frequency and image multiplication is performed. The power consumption for image multiplication is shown in Fig. 5.2 (b). The Wallace multiplier consumes the most power while ETM the least. Both the two proposed approximate multipliers, AM1 and AM2, require less power than UDM. In terms of arithmetic accuracy, both AM1 and AM2 have smaller NMED and MRED, and almost similar ER compared to UDM. ETM consumes the least power among the four approximate multipliers. However, its accuracy is the worst. Due to its limited accuracy, ETM may not be acceptable in many applications, as shown in the following section.

5.1.3 Discussion

The proposed approximate multipliers show significant and competitive accuracy compared to the other two approximate designs. Therefore, image multiplication by the proposed multipliers show a visually equivalent result to accurate processing with excellent PSNR values. The images obtained by the other two approximate multipliers show a lower good quality with lower PSNRs.

Table 5.2: Power-delay product (PDP) and power-delay-error metric (PDEM) of different approximate multipliers

	8/8 AM1	8/8 AM2	ETM [18]	UDM [17]
PDP	3.15	3.78	1	3.23
PDEM	1.67	1	6.29	9.90

The proposed designs reduce power and delay compared to Wallace multiplier. They also require less power consumption compared to UDM. The proposed designs are slightly slower than the other two multipliers; however, they achieve a very competitive accuracy compared to a precise multiplier.

To better compare the four different approximate multipliers, the power-delay product (PDP) and the power-delay-error metric (PDEM) are evaluated. The PDEM is defined to be $PDP \times NMED$. Table 5.2 shows the relative PDP and PDEM (i.e., the smallest PDP or PDEM is converted to 1 as a comparison base) of four approximate multipliers with the same configuration as those used in delay and power comparison. In terms of PDP (that does not consider accuracy), ETM is the best since it has an extremely small power consumption while the other three designs have similar PDPs. However, if accuracy is considered for a more comprehensive comparison, the proposed approximate multipliers show significant advantage. AM2 is the best in terms of PDEP while AM1 is the second. Both ETM and UDM have large PDEMs compared to the proposed ones. Therefore, the proposed approximate multipliers achieve the best trade-off between accuracy, power consumption and circuit performance.

5.2 Comparison of 16×16 Bit Approximate Multipliers

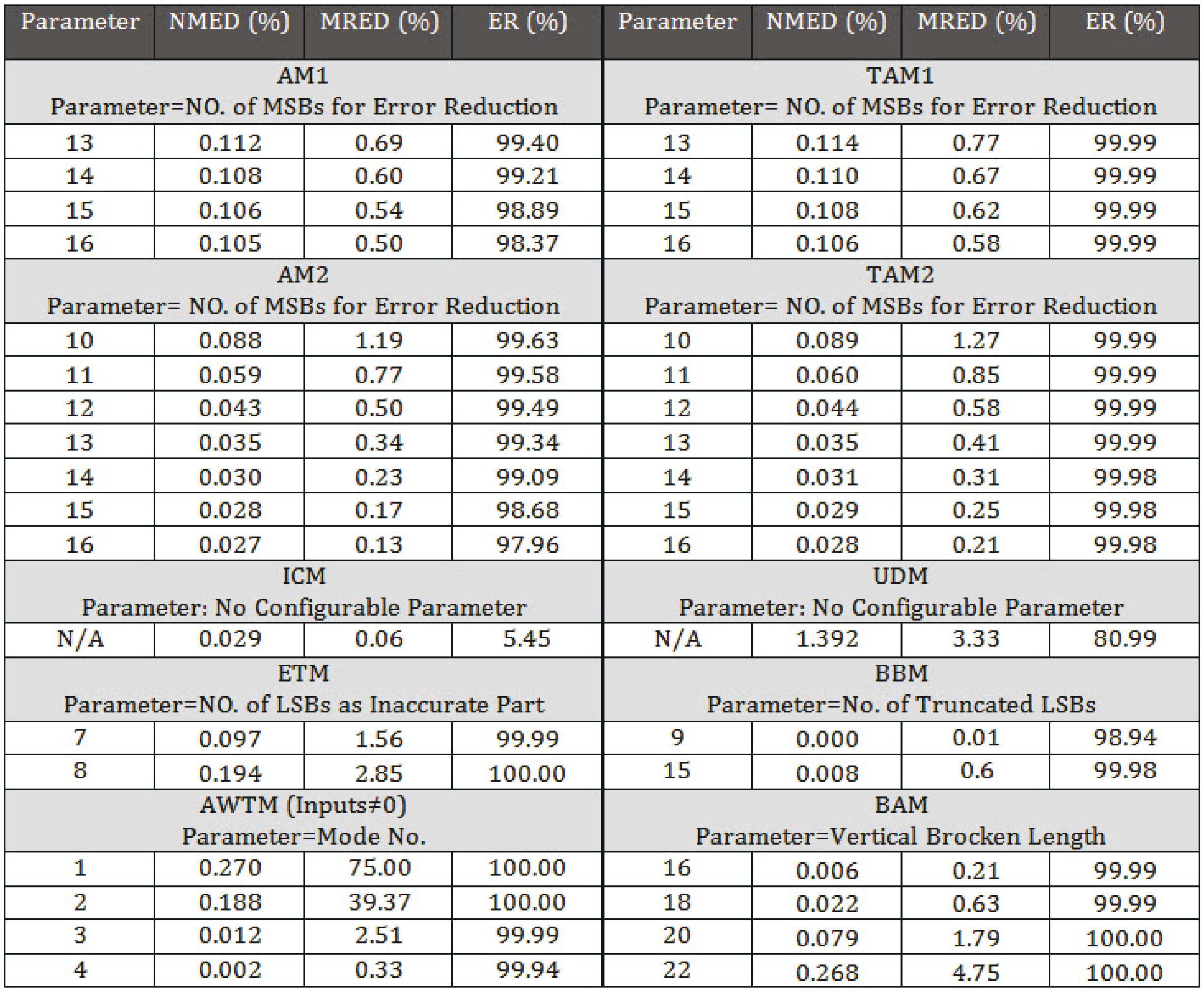
5.2.1 Error Characteristics

The proposed multipliers as well as several existing approximate multipliers have been simulated with 10^8 random input combinations and the error measures in NMED, MRED and ER have been obtained. Table 5.3 shows the results from Monte Carlo simulations.

Among these multipliers, UDM has the largest NMED while BBM has the smallest. Since BBM uses the Booth algorithm before applying the truncation strategy, the number of truncated partial products is very small and therefore it achieves a high accuracy in terms of NMED.

AWTM has low accuracy for small operands, and it even generates non-zero product when the accurate product is zero. In order to calculate the MRED, both input operands are selected to be non-zero. Table 5.4 shows the NMED and MRED of AM2 with 15 bits for error reduction (AM2-15), ICM and BAM with a truncation of 18 LSBs (BAM-18). All the three multipliers have close NMED values, however ICM has the smallest MRED while BAM has the largest. Therefore ICM has the best accuracy in terms of MRED, while BAM is the worst among these three. In fact, multipliers with simple truncation tend to have larger MREDs when NMEDs are similar. It can be seen that ETM, BBM and BAM have relatively larger MREDs due to truncation. AWTM also has very large MREDs especially when one of the input is zero, in which case it has an infinite RED

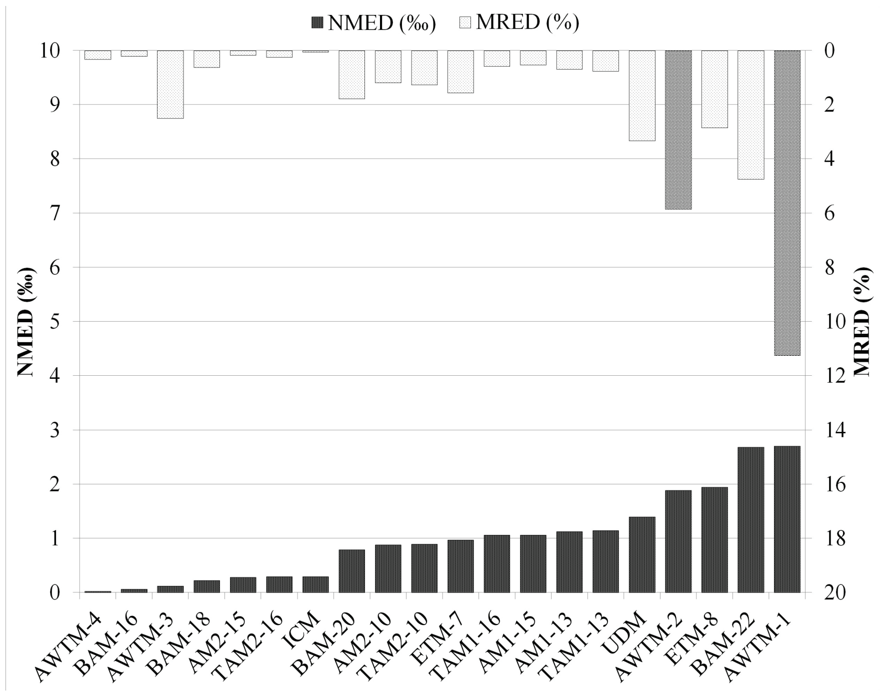
Table 5.3: Comparison of Error Characteristics



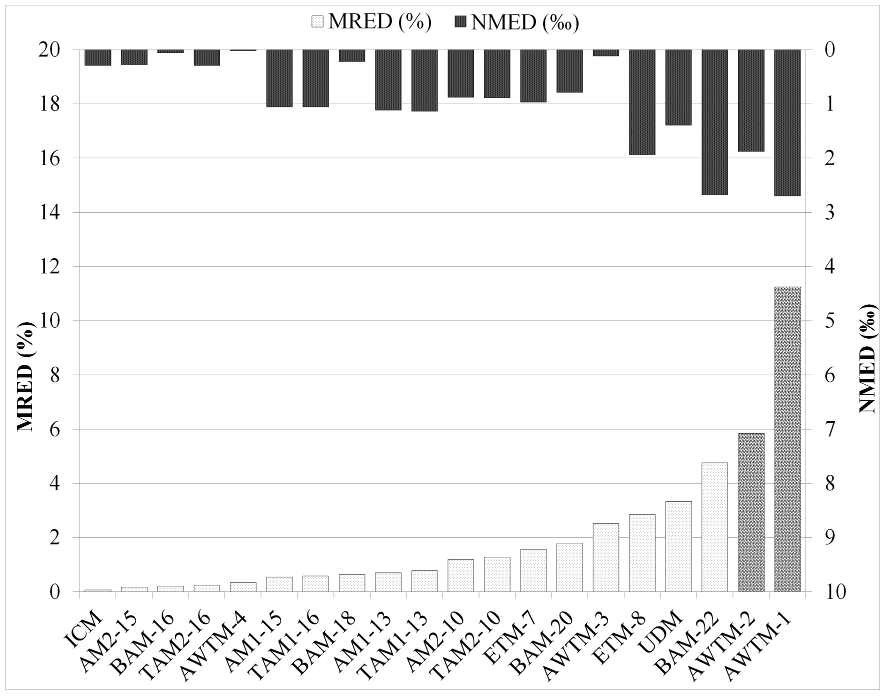
Parameter	NMED (%)	MRED (%)	ER (%)	Parameter	NMED (%)	MRED (%)	ER (%)
AM1 Parameter=NO. of MSBs for Error Reduction				TAM1 Parameter= NO. of MSBs for Error Reduction			
13	0.112	0.69	99.40	13	0.114	0.77	99.99
14	0.108	0.60	99.21	14	0.110	0.67	99.99
15	0.106	0.54	98.89	15	0.108	0.62	99.99
16	0.105	0.50	98.37	16	0.106	0.58	99.99
AM2 Parameter= NO. of MSBs for Error Reduction				TAM2 Parameter= NO. of MSBs for Error Reduction			
10	0.088	1.19	99.63	10	0.089	1.27	99.99
11	0.059	0.77	99.58	11	0.060	0.85	99.99
12	0.043	0.50	99.49	12	0.044	0.58	99.99
13	0.035	0.34	99.34	13	0.035	0.41	99.99
14	0.030	0.23	99.09	14	0.031	0.31	99.98
15	0.028	0.17	98.68	15	0.029	0.25	99.98
16	0.027	0.13	97.96	16	0.028	0.21	99.98
ICM Parameter: No Configurable Parameter				UDM Parameter: No Configurable Parameter			
N/A	0.029	0.06	5.45	N/A	1.392	3.33	80.99
ETM Parameter=NO. of LSBs as Inaccurate Part				BBM Parameter=No. of Truncated LSBs			
7	0.097	1.56	99.99	9	0.000	0.01	98.94
8	0.194	2.85	100.00	15	0.008	0.6	99.98
AWTM (Inputs≠0) Parameter=Mode No.				BAM Parameter=Vertical Brocken Length			
1	0.270	75.00	100.00	16	0.006	0.21	99.99
2	0.188	39.37	100.00	18	0.022	0.63	99.99
3	0.012	2.51	99.99	20	0.079	1.79	100.00
4	0.002	0.33	99.94	22	0.268	4.75	100.00

Table 5.4: Comparing MREDs of Three Multipliers

	NMED (%)	MRED (%)
AM2-15	0.028	0.17
ICM	0.029	0.06
BAM-18	0.022	0.63



(a)



(b)

Figure 5.3: A comparison of NMED and MRED of approximate multipliers with data sorted on (a) NMED and (b) MRED, where the MREDs of AWTM-1 and AWTM-2 are smaller than their real values for better display.

(since its output is non-zero).

Most of the designs, especially those with truncation, have large ERs. For example, ETM and BAM result in nearly 100% error rates. However, there is one exception: ICM. ICM has an extremely low ER of 5.45% because it uses only one approximate compressor in a 4×4 bit sub-multiplier. The error rate for this sub-multiplier is only 1/256, so a 16×16 bit multiplier using the 4×4 bit sub-multipliers has a low error rate too. Take ICM and TAM2 with 15-bit error reduction as an example. Both multipliers have similar NMEDs, however ICM has an extremely low ER. This indicates that ICM tends to generate errors with large magnitudes.

To give a whole picture, the NMED and MRED of approximate multipliers are shown in Fig. 5.3, with ascending NMED values from left to right in Fig. 5.3 (a) and ascending MRED values in Fig. 5.3 (b). Since the MRED values for AWTM-1 and AWTM-2 are too large compared to the other designs, these values are made smaller than their real values to fit in the figure. Based on NMED, these approximate multipliers can be classified into three categories: Low-Error Multiplier (LEM), Medium-Error Multiplier (MEM) and High-Error Multiplier (HEM). From AWTM-4 up to ICM in Fig. 5.3 is the LEM region, BAM-20 up tp TAM1-13 is the MEM region and the others are HEMs. Two configurations of the proposed designs (TAM2-16 and AM2-15) fall in the LEM region and the others are classified to be MEMs. Therefore, the proposed designs are low and medium error multipliers. In MRED, AWTM-3 has a very high MRED (2.51%) in the LEM category, while the other LEMs have very small MREDs. In the MEM category, the proposed multipliers have smaller MREDs compared to the other designs.

In conclusion, the proposed designs have relatively low NMEDs, as well as MREDs, among all the considered multipliers. Some designs may have a low NMED (e.g., BAM-18 and AWTM-3), however their MREDs are relatively high. It is shown in the next section that the proposed designs achieve the best overall trade-off when circuit characteristics are considered.

5.2.2 Circuit Characteristics

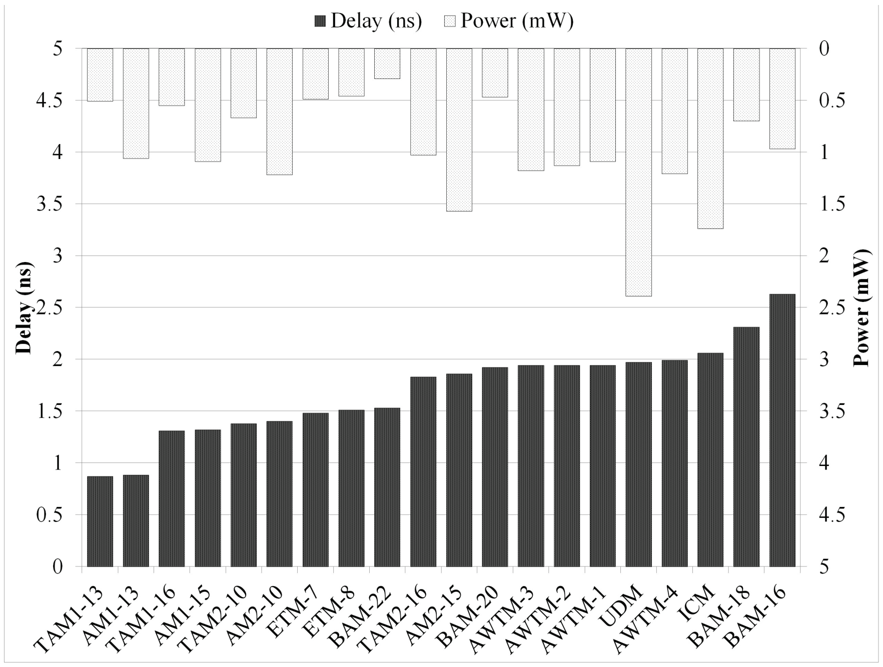
16×16 bit approximate multipliers have been implemented in VHDL and their delay, area and power were obtained using Synopsys Design Vision based on STM 65-nm process. Table 5.5 shows the simulation results from Synopsys Design Vision.

AM1/TAM1 has a smaller delay even with a 16-bit error reduction compared to the other types of designs. For example, AM1-13 has a very short delay of only 0.88ns, while AM2-10 has a delay of 1.40ns, and the next fastest design is ETM-7 with 1.48ns. Although AM2 results in a larger delay than AM1, it is faster than the other designs except for ETM. As analyzed in Chapter 4, the proposed approximate multipliers can significantly reduce the critical path delay.

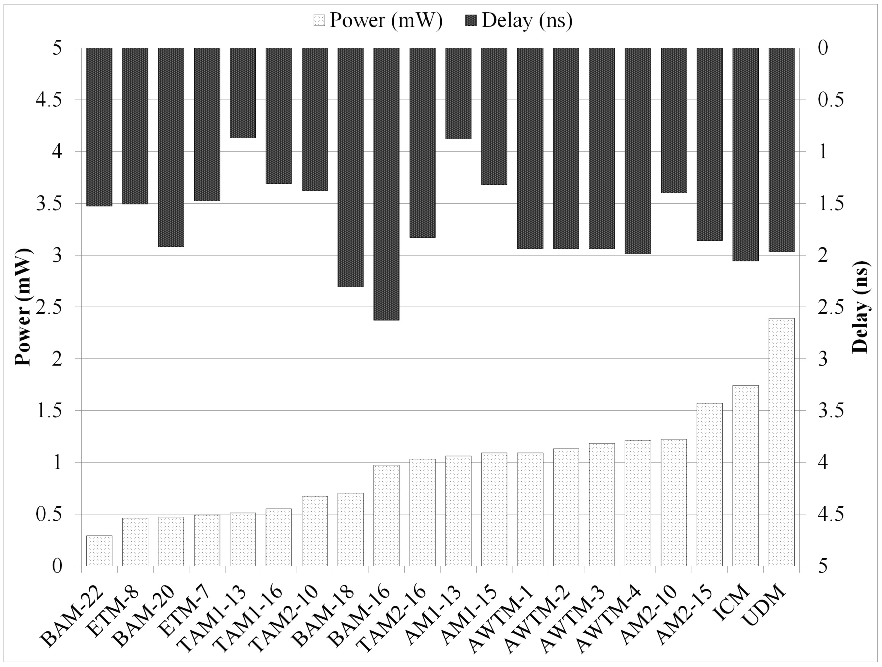
In terms of area and power efficiency, ETM, TAM1/TAM2 and BAM are among the best designs. One similarity of these designs is that they all use truncation. As discussed before, truncation can significantly affect MRED while NMED is not changed much. If most of the inputs are large values, the error introduced by truncation can be tolerated; thus truncation is a useful scheme to save area and power. Otherwise, truncation-based designs may yield unacceptably inaccurate results.

Table 5.5: Comparison of Circuit Characteristics

Parameter	Delay (ns)	Area (um^2)	Power (mW)	Parameter	Delay (ns)	Area (um^2)	Power (mW)
AM1 Parameter=NO. of MSBs for Error Reduction				TAM1 Parameter= NO. of MSBs for Error Reduction			
13	0.88	2812	1.06	13	0.87	1345	0.51
14	1.26	2861	1.09	14	1.25	1394	0.54
15	1.32	2873	1.09	15	1.31	1408	0.55
16	1.41	2894	1.10	16	1.31	1408	0.55
AM2 Parameter= NO. of MSBs for Error Reduction				TAM2 Parameter= NO. of MSBs for Error Reduction			
10	1.40	3302	1.22	10	1.38	1822	0.67
11	1.51	3424	1.28	11	1.48	1947	0.73
12	1.63	3560	1.33	12	1.61	2084	0.79
13	1.72	3709	1.41	13	1.69	2233	0.86
14	1.80	3869	1.48	14	1.77	2396	0.94
15	1.86	4043	1.57	15	1.83	2598	1.03
16	1.93	4230	1.66	16	1.84	2603	1.04
ICM Parameter: No Configurable Parameter				UDM Parameter: No Configurable Parameter			
N/A	2.06	3462	1.74	N/A	1.97	3099	1.85
ETM Parameter=NO. of LSBs as Inaccurate Part				BBM Parameter=No. of Truncated LSBs			
7	1.48	1084	0.49	9	3.38	3759	2.39
8	1.51	1126	0.46	15	2.90	2821	1.69
AWTM Parameter=Mode No.				BAM Parameter=Vertical Brocken Length			
1	1.94	2417	1.09	16	2.63	1679	0.97
2	1.94	2510	1.13	18	2.31	1325	0.70
3	1.94	2603	1.18	20	1.92	1023	0.47
4	1.99	2595	1.21	22	1.53	780	0.29



(a)



(b)

Figure 5.4: A comparison of delay and power of approximate multipliers with data sorted on (a) delay and (b) power.

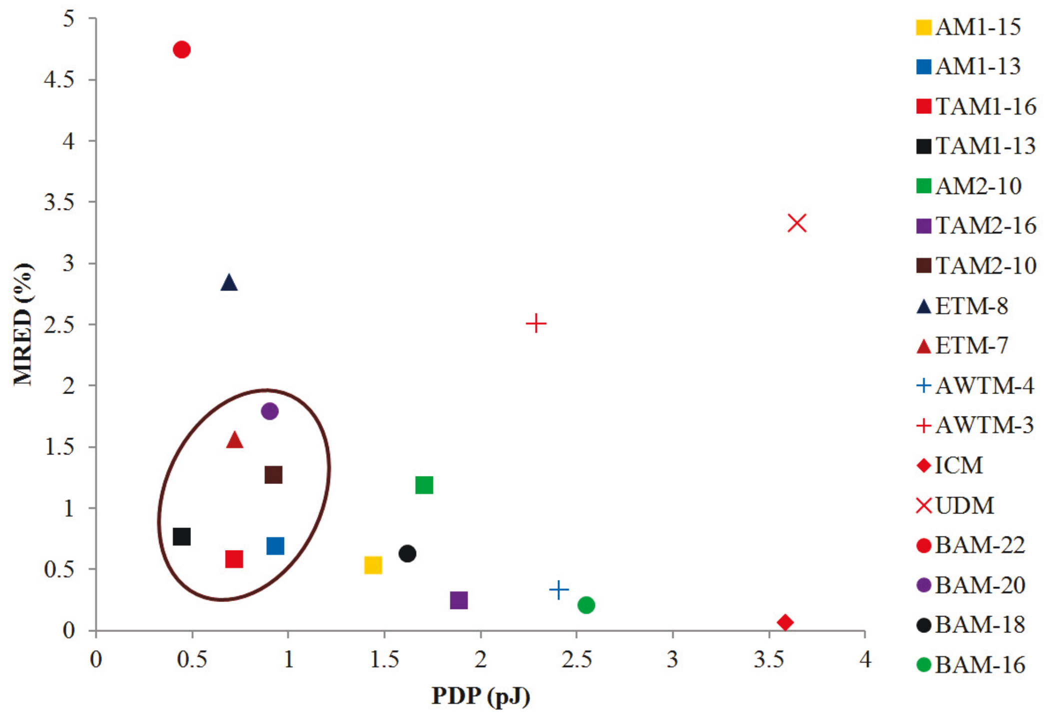


Figure 5.5: MRED and PDP of approximate multipliers

In conclusion, the proposed designs have shorter critical paths than the other designs, with competitive power efficiencies. When truncation is applied, however, the proposed designs are among the most power-efficient designs.

5.2.3 Discussion

For an overall evaluation of different approximate multiplier designs, both error and circuit characteristics should be considered. In this discussion, MRED and power-delay-product (PDP) of the approximate multipliers are used to show the tradeoff between error and circuit measures (Fig. 5.5).

According to Fig. 5.5, the proposed designs have both small PDPs and MREDs, which makes them stand out. Most of the other designs have at least one major shortcoming. For example, ICM incurs an extremely low error, but its PDP is very high. Even though most BAM configurations have small PDPs, their delays are generally large (Fig. 5.4). Moreover, some BAM configurations have low accuracies. AWTMs have large PDPs and only AWTM-4 has a high accuracy.

Those designs that fall in the ellipse in Fig. 5.5 are considered to be desired designs, among which there are four proposed designs. Therefore, some of the proposed designs achieve the best tradeoff between error and circuit measures. Furthermore, the proposed designs have more degrees of freedom to configure, including the number of MSBs for error reduction, the number of LSBs for truncation and the error reduction circuit (either AM1 or AM2). Therefore, the proposed designs are flexible to use for achieving different tradeoffs in approximate circuit design.

Chapter 6

Conclusion

In this thesis, an analytical framework has been proposed to analyze both the generic and application-specific error characteristics of approximate adders. A novel approximate multiplier with low power and high performance has also been presented. A comparative evaluation of various approximate multipliers has been performed.

In the analytical framework, several representative approximate adder designs are considered and generic error metrics such as the error rate are computed. This analysis provides new insights into different approximate adder architectures. For example, the almost-correct adder (ACA) has an extremely low error rate, however its mean error distance is very high, i.e., the magnitudes of the errors are, on average, very large. This framework is further extended to analyze application-specific metrics such as the peak-signal-to-noise ratio (PSNR). The analytical results show that the mean error distance is more relevant to application-specific metrics that is sensitive to accumulated errors, while error rate is less important. The derived mathematical relationships between the generic metrics and PSNR are in fact independent of the analysis of generic metrics. As a result, the analysis of PSNR can be applied to adder designs that have not been considered in this thesis. A different analytical or simulation method can be used to analyze the generic metrics, and then PSNR can further be evaluated using the proposed relationships. Hence, this framework can be divided into two relatively independent parts: the analysis of generic metrics of several types of adder designs and analysis of the PSNR for given generic metrics. While the first part of this framework is only applicable to the adders similar to those considered in the thesis, the second part can be applicable to many adders and even other types of approximate circuits.

A new approximate multiplier design is also proposed in this thesis. This multiplier leverages a newly proposed adder, which generates an approximate result and an error signal. The error signals are then used in the configurable error reduction stage. There are two ways to implement the error reduction: 1) by configuring the number of MSBs for error reduction and 2) by configuring the circuit (either one of the two approximate modes) for error reduction. This configurable feature provides design flexibility in trading off accuracy for a reduced power and delay. With an appropriate configuration of the error reduction, the proposed multiplier can have extremely low power consumption, short critical paths and very high accuracy. When applied in image processing, the multiplier achieves a competitive result with no perceivable loss of quality. Simulations have also been performed for image processing applications. Power savings up to 69% have been obtained, compared to a traditional accurate multiplier.

A comprehensive comparison of the proposed and existing approximate multipliers has been performed on 16×16 bit designs. The proposed multipliers are the fastest in terms of delay, and very competitive in terms of both accuracy and power, while the other designs have at least one major shortcoming in accuracy, delay or power. The proposed design with certain configuration achieves the best tradeoff in accuracy, delay and power.

Bibliography

[1] K. Bhardwaj, P. S. Mane, and J. Henkel. Power- and area-efficient Approximate Wallace Tree Multiplier for error-resilient systems. In *Proceedings of the 15th International Symposium on Quality Electronic Design*, pages 263–269. IEEE, Mar. 2014.

[2] K. Bickerstaff, E. Swartzlander, and M. Schulte. Analysis of column compression multipliers. In *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pages 33–39, 2001.

[3] M. Breuer. Intelligible test techniques to support error-tolerance. In *Proceedings of the 13th Asian Test Symposium*, pages 386–393, 2004.

[4] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar. Scalable effort hardware design. In *Proceedings of the 47th Design Automation Conference - DAC '10*, New York, New York, USA, June 2010. ACM Press.

[5] S. Datla, M. Thornton, and D. Matula. A low power high performance radix-4 approximate squaring circuit. In *Proceedings of the 20th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2009)*, pages 91–97, July 2009.

[6] K. Du, P. Varman, and K. Mohanram. High performance reliable variable latency carry select addition. In *Proceedings of the 2012 Design, Automation Test in Europe Conference Exhibition (DATE 2012)*, pages 1257–1262, March 2012.

[7] M. D. Ercegovac and T. Lang. *Digital arithmetic.* Morgan Kaufmann, 2003.

[8] F. Farshchi, M. S. Abrishami, and S. M. Fakhraie. New approximate multiplier for low power digital signal processing. In *Proceedings of the 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*, pages 25–30. IEEE, Oct. 2013.

[9] S. K. Gupta. Approximate logic synthesis for error tolerant applications. In *Proceedings of the 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pages 957–960. IEEE, Mar. 2010.

[10] 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.

[11] J. Han and M. Orshansky. Approximate Computing: An Emerging Paradigm For Energy-Efficient Design. In *Proceedings of the 18th IEEE European Test Symposium (ETS 2013)*, Avignon, France, May 2013.

[12] R. Hegde and N. Shanbhag. Soft digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(6):813–823, Dec 2001.

[13] A. Hore and D. Ziou. Image Quality Metrics: PSNR vs. SSIM. In *Proceedings of the 20th International Conference on Pattern Recognition*, pages 2366–2369. IEEE, Aug. 2010.

[14] R. A. Horn. The hadamard product. In *Proceedings of Symposia in Applied Mathematics*, volume 40, pages 87–169, 1990.

[15] J. Huang, J. Lach, and G. Robins. A methodology for energy-quality tradeoff using imprecise hardware. In *Proceedings of the 49th ACM Annual Design Automation Conference*, pages 504–509, 2012.

[16] A. B. Kahng and S. Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proceedings of the 49th ACM Annual Design Automation Conference*, pages 820–825, 2012.

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

[18] K. Y. Kyaw, W. L. Goh, and K. S. Yeo. Low-power high-speed multiplier for error-tolerant application. In *Proceedings of the 2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, pages 1–4, 2010.

[19] M. S. Lau, K.-V. Ling, and Y.-C. Chu. Energy-aware probabilistic multiplier: design and analysis. In *Proceedings of the 2009 ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, pages 281–290, 2009.

[20] 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, 2013.

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

[22] C. Liu, J. Han, and F. Lombardi. An analytical framework for evaluating the error characteristics of approximate adders. *IEEE Transactions on Computers*, PP(99):1–1, 2014.

[23] C. Liu, J. Han, and F. Lombardi. A low-power, high-performance approximate multiplier with configurable partial error recovery. In *Proceedings of the 2014 Design, Automation & Test in Europe Conference (DATE 2014)*, 2014.

[24] Y. Liu, T. Zhang, and K. Parhi. Computation error analysis in digital signal processing systems with overscaled supply voltage. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(4):517–526, April 2010.

[25] S.-L. Lu. Speeding up processing with approximation circuits. *Computer*, 37(3):67–73, 2004.

[26] J. Ma, K. L. Man, N. Zhang, S.-U. Guan, and T. T. Jeong. High-speed area-efficient and power-aware multiplier design using approximate compressors along with bottom-up tree topology. In *Proceedings of the 5th International Conference on Machine Vision (ICMV 2012)*. International Society for Optics and Photonics, 2013.

[27] 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*, 57(4):850–862, Apr. 2010.

[28] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. Modeling and synthesis of quality-energy optimal approximate adders. In *Proceedings of the ACM International Conference on Computer-Aided Design*, pages 728–735, 2012.

[29] D. Mohapatra, V. Chippa, A. Raghunathan, and K. Roy. Design of voltage-scalable meta-functions for approximate computing. In *Proceedings of the 2011 Design, Automation & Test in Europe Conference Exhibition (DATE 2011)*, pages 1–6, March 2011.

[30] A. Momeni, J. Han, P. Montuschi, and F. Lombardi. Design and Analysis of Approximate Compressors for Multiplication. *IEEE Transactions on Computers*, PP(99):1–1, 2014.

[31] H. R. Myler and A. R. Weeks. *The pocket handbook of image processing algorithms in C*. PTR Prentice Hall, 1993.

[32] R. M. M. Oberman. *Digital circuits for binary arithmetic*. Macmillan, 1979.

[33] B. Parhami. *Computer arithmetic*. Oxford university press, 2000.

[34] K. Parhi. Design of low-error fixed-width modified booth multiplier. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(5):522–531, May 2004.

[35] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

[36] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman. Enerj: Approximate data types for safe and general low-power computation. 46(6):164–174, 2011.

[37] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan. Macaco: Modeling and analysis of circuits for approximate computing. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 667–673, 2010.

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

[39] J.-P. Wang, S.-R. Kuang, and S.-C. Liang. High-Accuracy Fixed-Width Modified Booth Multipliers for Lossy Applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(1):52–60, Jan. 2011.

[40] N. H. Weste and H. David. *CMOS VLSI Design-A Circuit and Systems Perspective*. Pearson Addison Wesley, 3rd edition, 2005.

[41] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi. Approximate xor/xnor-based adders for inexact computing. In *Proceedings of the IEEE International Conference on Nanotechnology*, Beijing, China, August 2013.

[42] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo. Enhanced low-power high-speed adder for error-tolerant application. In *Proceedings of 2010 International SoC Design Conference*, pages 323–327. School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, IEEE, 2010.

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

[44] N. Zhu, W. L. Goh, and K. S. Yeo. Ultra low-power high-speed flexible Probabilistic Adder for Error-Tolerant Applications. In *Proceedings of the 2011 International SoC Design Conference*, pages 393–396. IEEE, Nov. 2011.

[45] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong. Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(8):1225–1229, Aug. 2010.