Analysis, Design and Modeling of Approximate Adders for Error-resilient Applications

A

Dissertation Submitted

by

Sunil Dutt

in Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI – 781039, ASSAM, INDIA

OCTOBER 2018

Certificate

This is to certify that the thesis entitled "**Analysis, Design and Modeling of Approximate Adders for Error-resilient Applications**", submitted by **Sunil Dutt (136102004)**, a research scholar in the *Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati*, for the award of the degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under our supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in our opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any type of degree or diploma.

Place: Guwahati
Date:

Dr. Gaurav Trivedi
Associate Professor
Dept. of Electronics and Electrical Engg.
Indian Institute of Technology Guwahati
Guwahati – 781039, Assam, India.

Place: Guwahati
Date:

Prof. Sukumar Nandi
Professor
Dept. of Computer Science and Engg.
Indian Institute of Technology Guwahati
Guwahati – 781039, Assam, India.

Dedicated

to

My Lovely Daughter, Niyana

Acknowledgments

I first thank my God almighty for graciously giving me his strength and guidance throughout my life. I owe him everything, and I attribute all praise and glory to him.

In fact, I could not write this page without expressing my deepest thanks to my advisors, **Dr. Gaurav Trivedi** and **Prof. Sukumar Nandi**, who are the epitome of a great mentor and guided me to the bleeding-edge in my field of research. They have always been sincere, attentive, available and supportive throughout all the peaks and valleys of my doctorate degree. In all research meeting, they never failed to advise me with their inspiring perspective, amazing engineering-intuition, tireless enthusiasm and firm knowledge. There is much more about their management style that I would like to bring with me as I go into industry and/or academia. All together, I could not asked for better advisors than **Dr. Gaurav Trivedi** and **Prof. Sukumar Nandi**. I am also thankful to the members of doctoral committee for their constructive inputs in my research.

Further, I would not have reached where I am today, had it not been for the guidance and motivation of my parents. I thank them for making it possible for me to reach this major milestone in my life. Thank you mom and dad for supporting me and for giving to me all the best of your heart. For all that I have conquered in my life so far, I need to thank you both. I would also like to acknowledge to all of you that have been with me all my life: my family, my friends and my colleagues. Every person that I know is important to me and each one has something that has made the difference. Everyday when I make decisions, I know that each one has contributed in some how for these choices and I am sure that my life is so wonderful because of that support. Finally, I would like to thank the Govt. of India, who helped me in terms of financial support during my research.

Sunil Dutt

Abstract

Over the decades, *Complementary Metal-Oxide-Semiconductor* (CMOS) technology scaling has been the fundamental driver for computing. However, we are now in a phase where CMOS technology scaling is becoming less effective in improving the system capability. The consequence is that we must either accept that the computing systems are good enough or look for alternate avenues to advance them without significant technology progress. Recent studies show that there are several promising alternate avenues that jointly can improve the system capability equivalent to $2-3$ decades of Moore's law. Approximate computing is one of them and in recent years, has attracted a lot of attention of researchers as well as industry. It should be noted that the concept of approximate computing trade-offs computation quality for computation efforts.

In recent years, several approximate adders have been proposed in the literature. The key design approach behind these approximate adders is to truncate the carry-chain. The two most commonly used approaches to truncate the carry-chain are: (i) *Approximate Full Adder* (AFA); and (ii) *Equal Segment Adder* (ESA). In the first approach, an N-bit adder is segmented into two sub-adders: (i) Accurate sub-adder that includes the higher order k bits; and (ii) Approximate sub-adder that includes the remaining lower order $(N-k)$ bits. For accurate sub-adder, *Full Adders* (FAs) are used, whereas for approximate sub-adder, AFAs are used. In the second approach, an N-bit adder is segmented into several smaller disjoint or overlapping equally sized accurate sub-adders. The *Carry-in* (C_{in}) of all sub-adders is considered as 0. Consequently, all sub-adders become independent and operate in parallel. This thesis is divided into three parts in which analysis, designing, analytical modeling, optimization and applications of AFAs and ESAs are presented.

In the first part, four AFAs are proposed with design objective that *Carry-out* (C_{out}) should be independent of C_{in} with minimal error probability. Using the proposed AFAs, an N-bit approximate adder is designed which we call as "ApproxADD". In order to improve the *Error Distance* (ED) and

Error Rate (ER) of ApproxADD, the concept of carry-lifetime and *Error Detection and Correction* (EDC) logic, respectively, is used. In this way, two more versions of ApproxADD – ApproxADD$v1$ and ApproxADD$v2$ are introduced. Further, analytical models are provided to estimate the accuracy, delay, power and area of ApproxADDs. The efficiency of ApproxADDs is evaluated by comparing them with existing approximate adders. Finally, the effectiveness of the proposed approach in real-life applications is examined via an image processing application.

In the second part, analytical models are proposed to estimate the accuracy, delay, power and area of ESAs. It should be noted that the proposed analytical models are generalized and superior (or at par) to the existing analytical models. From the proposed analytical models, it is observed that in an N-bit ESA, there exist multiple configurations which exhibit similar accuracy, however, these configurations exhibit different delay, power and area. Therefore, for a given accuracy, the configurations which provide minimal delay, power and/or area need to be known apriori for efficient, intelligent and goal oriented implementations of ESAs. In this regard, an optimization framework is presented that exploits the proposed analytical models and reveals the optimal configurations. Further, from the proposed analytical models, it is observed that there is a scope and need of improvement in accuracy-effort curves of ESAs. For improving the accuracy-effort curves, modifications are proposed with design objective that the modified ESAs provide higher accuracy without imposing any additional delay, power and area overheads $w.r.t.$ original ESAs.

In the third part, the applicability of approximate adders in cryptography applications, yield enhancement and designing other approximate arithmetic operations is examined. First, an approximate *Secure Hash Algorithm* (SHA-1) is designed using ApproxADDs and evaluated over cryptographic hash functions. Further, analytical models are proposed to estimate the yield of approximate circuits. From the proposed analytical models, it is observed that approximate circuits can improve the functional yield and parametric yield due to decrease in area and delay, respectively. This is demonstrated with ApproxADDs. Further, in order to examine the effectiveness of approximate adders for designing other approximate arithmetic operations, an approximate *Multiply-and-Accumulate* (MAC) unit is designed using approximate adders. The effectiveness of approximate MAC unit in real-life applications is examined via an image processing application.

Contents

Contents

Contents

List of Figures

List of Tables

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

List of Acronyms

ACA	Accuracy Configurable Adder
ACC_{amp}	Accuracy of Amplitude
ACC_{inf}	Accuracy of Information
AFA	Approximate Full Adder
AMA	Approximate Mirror Adder
AP	Acceptance Probability
AXA	Approximate XOR/XNOR-based Adder
BK	Brent-Kung
BKA	Brent-Kung Adder
CIA	Carry Increment Adder
CMOS	Complementary Metal-Oxide-Semiconductor
CPD	Critical Path Delay
CUT	Chip Under Test
DCT	Discrete Cosine Transform
DR	Distance Rate
DSEC	Dynamic Segmentation and Error Compensation
DSP	Digital Signal Processing
ECL	Error Correction Logic
ED	Error Distance
EDC	Error Detection and Correction
EDL	Error Detection Logic
EF	Error Flag

ER	Error Rate
ESA	Equal Segment Adder
ETA-1	Error-Tolerant Adder 1
FA	Full Adder
FPGA	Field Programmable Gate Array
GeAr	Generic Accuracy Configurable Adder
GP	Geometric Progression
JPEG	Joint Photographic Experts Group
IDCT	Inverse Discrete Cosine Transform
IQA	Image Quality Assessment
KA	Knowles Adder
KS	Kogge-Stone
KSA	Kogge-Stone Adder
LOA	Lower-part-OR Adder
LSB	Least Significant Bit
LA	Lu's Adder
MAA	Minimum Acceptable Accuracy
MAC	Multiply-and-Accumulate
MAD	Maximum Allowed Delay
MED	Mean Error Distance
MRED	Mean Relative Error Distance
MSB	Most Significant Bit
MSE	Mean Squared Error
NMOS	N-Type Metal-Oxide-Semiconductor
PDAA	Power-Delay-Area-Accuracy
PDAP	Power-Delay-Area Product
PMOS	P-Type Metal-Oxide-Semiconductor
PSNR	Peak Signal-to-Noise Ratio

PTL	Pass Transistor Logic
PTM	Predictive Technology Model
RCA	Ripple Carry Adder
RED	Relative Error Distance
RFD	River Formation Dynamics
SA_i	i^{th} Sub-Adder
SA	Soares's Adder
SAC	Strict Avalanche Criterion
SAD	Sum of Absolute Differences
SFA	Simplified Full Adder
SHA-1	Secure Hash Algorithm 1
Sk	Sklansky
SSIM	Structural Similarity Index Metric
VLSA	Variable Latency Speculative Adder

1

Introduction

Contents

1. Introduction

Binary arithmetic is an essential part of digital systems. In binary arithmetic, all basic operations, such as addition, subtraction, multiplication and division use adders as the key components. Besides the arithmetic operations, adders are used to perform increment, decrement and many similar operations. Therefore, at micro-architecture level of abstraction, adders can be treated as the basic building blocks of digital systems, particularly, of digital systems which are designed for data-dominated applications, $e.g.$, multimedia applications. In such applications, adders being the key components determine the overall system performance and power consumption. Over the decades, several adders have been proposed in the literature [6]. Among all existing conventional adders, *Ripple Carry Adder* (RCA) is the simplest one, giving the smallest area ($O(N)$), where N is the bit-width. However, it exhibits linear delay ($O(N)$), and thus, is significantly slow. On the other hand, parallel-prefix computation based adders, such as *Kogge-Stone Adder* (KSA) are the fastest one, exhibiting the logarithmic delay. However, their hardware implementation requires larger area ($O(Nlog_2N)$). In summary, as shown in Fig. 1.1, delay and area are two conflicting design metrics in conventional adders, $i.e.$, improvement of one usually demands the sacrifice of another. As a result, no adder can provide logarithmic delay ($O(log_2N)$) along with linear area ($O(N)$). Further, we know that power consumption of a digital circuit is proportional to its area. Therefore, adders which exhibit larger area are expected to consume more power as compared to the adders which exhibit smaller area [6]. All together, due to the conflicting design metrics, adders have become one of the key delay/power bottlenecks of digital systems, $i.e.$, for a marginal improvement in delay/power, we have to sacrifice a significant amount of power/delay. Moreover, since delay and power of adders increase rapidly with bit-width, this situation is expected to become more worse in the near future.

One possible way to overcome the above-mentioned situation ($i.e.$, to improve the delay-power frontier of conventional adders) is to sacrifice the accuracy for delay and/or power benefits. Several error-resilient computing techniques have been introduced in the literature for such trade-offs. The three most commonly used error-resilient computing techniques are: (i) Approximate computing [19]; (ii) Probabilistic computing [20]; and (iii) Stochastic computing [21]. Among all these three error-resilient computing techniques, approximate computing has recently attracted a lot of attention of researchers as well as industry [22]. This is partially due to the fact that it relies on the existing models

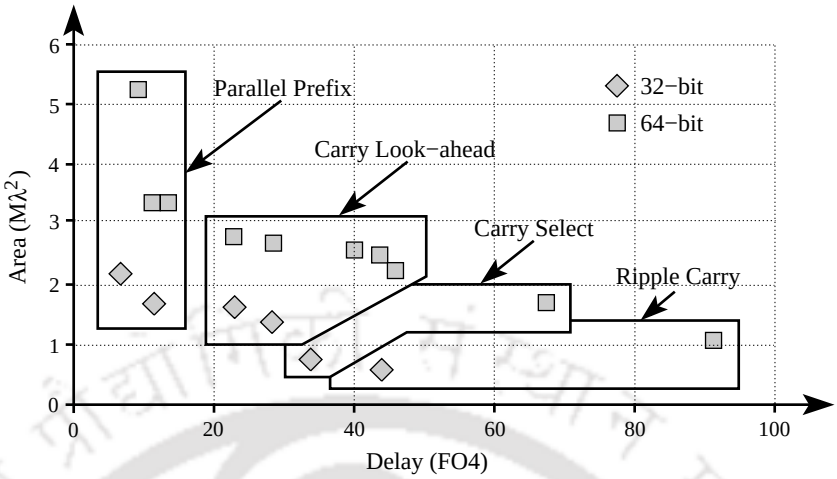


Figure 1.1: Delay versus area of conventional adder architectures [6].

and design approaches, rather than changing the perspective completely. Over the past decade, several research works have been proposed in the literature for approximate computing at different levels of abstraction [23–30]. Meanwhile, at hardware level of abstraction, most of the work has been proposed on arithmetic units. Further, among arithmetic units, adders have attracted the strongest attention for hardware level approximation [31, 32]. In this thesis, we analyse, design, model, optimize and evaluate the effectiveness (at application level) of approximate adders.

1.1 Approximate Computing

The concept of approximate computing features high-performance energy-efficient software and hardware implementations by trading-off the computation accuracy for computation efforts [25, 33]. Over the past decade, several research works have been proposed in approximate computing paradigm both at software level and hardware level of abstraction with encouraging results [23–30]. Software techniques skip algorithm level computations, whereas hardware techniques modify designs at circuit level and architecture level of abstraction. At software level of abstraction, researchers/designers exploit approximate algorithms to optimize the computation complexity of compute-intensive applications ($e.g.$, NP-hard and NP-complete [34–36]) as there can ever be efficient polynomial-time accurate algorithms that can solve such applications due to massive input data size. On the other hand, at hardware level of abstraction, researchers/designers explore approximate computing through trun-

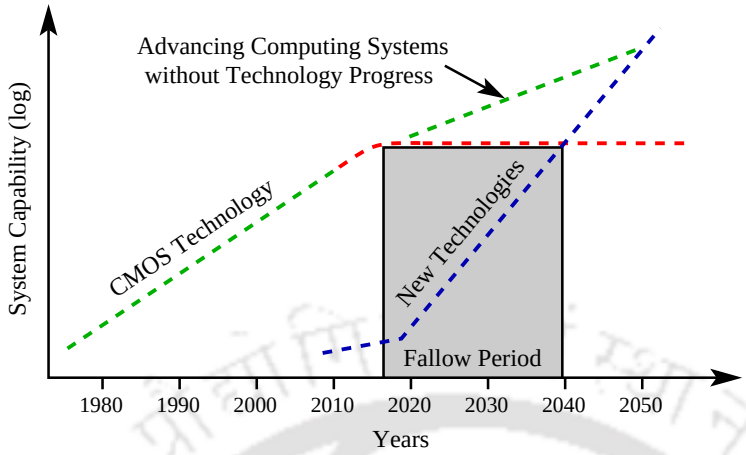


Figure 1.2: Researchers/designers are working to harvest new gains during the fallow period to advance the computing systems without significant technology progress [7].

cation [37–39], voltage over-scaling [40–42], over-clocking [43–45] and approximate storage/accelerators/arithmetic units [31, 46, 47]. The key motivations for approximate computing are as follows.

(i) **Technology perspective:** Over the decades, Moore's law [48] and Dennard scaling [49] have jointly been the fundamental driver of computing. However, with the breakdown of Dennard scaling (around $2004 - 2006$), increase in clock frequency and reduction in supply voltage have greatly slowed down. Under such conditions, *Complementary Metal-Oxide-Semiconductor* (CMOS) technology can follow Moore's law, but the power density increases to a level that processors can be as hot as a rocket nozzle [50]. Therefore, in mid $2000s$, the microprocessor industry has shifted to multicore scaling. The objective behind multicore scaling is to improve the system capability by increasing the number of cores per die per generation. However, it is now becoming difficult because: (a) The performance gain with increased core counts is diminishing due to serial computation, synchronization, global communication, etc. [51]; and (b) More cores are only possible if the cores are slower or less utilized with each generation due to dark silicon[1]. All together, as shown in Fig. 1.2, we are now in a phase where CMOS technology scaling is becoming less and less effective at improving the system capability. For the improvement of system capability, researchers/designers are thinking about the post-CMOS

[1] With the progression of Moore's law, the percentage of a CMOS chip that can actively be used within its own power budget is dropping exponentially [52]. Even at $22nm$, 21% of a CMOS chip must be powered off due to thermal design power constraint. At $8nm$, this number is expected to reach up to $50\% - 80\%$.

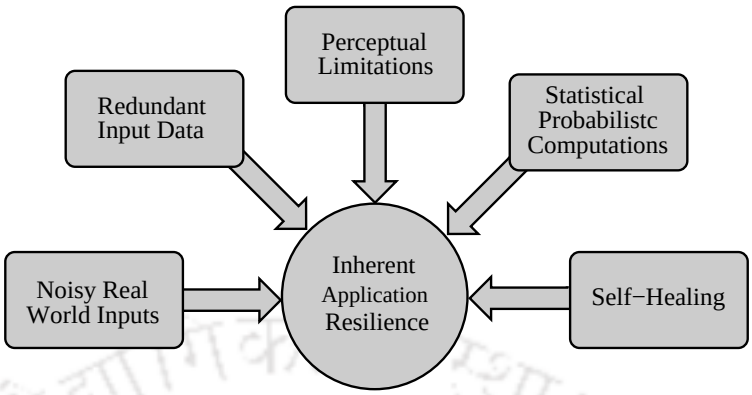


Figure 1.3: Sources of inherent error-resilience of applications [8].

technologies, but there appears to be no obvious alternate technology that can replace *End-of-Roadmap* CMOS over the next $2-3$ decades (see Fig. 1.2). In such a scenario, we must either accept that computing systems are good enough or look for alternate avenues to advance them without (significant) technology progress. Recent studies show that there are several promising alternate avenues which can jointly improve the system capability equivalent to $2-3$ decades of Moore's law [7]. Approximate computing is one of them and in recent years, it has attracted a lot of interest of the researchers as well as industry [19].

(ii) **Application perspective:** We know that today's computing is driven by a very different kind of applications. One of the emerging and important domains of applications is *error-resilient applications*. Error-resilience refers to the property of an application to accept outputs despite some of its underlying computations being executed using approximate software and/or hardware. As shown in Fig. 1.3, the error-resilience of such applications is attributed to several factors, including noisy and redundant input data, absence of a unique golden output, limited perception of human senses, algorithmic features, etc. [8]. Due to the property of error-resilience, final output in such applications need not be fully precise, rather an approximate output is equally acceptable provided the approximate output satisfies the accuracy requirements. It has been observed that a very high degree of error-resilience is prevalent in a broad spectrum of applications, including web search, machine learning, computer vision, statistical and probabilistic analytics, scientific computing, image, audio and video processing, wireless communication, etc [8, 30, 53–55]. All these applications are very compute-intensive, but the

feature of error-resilience provides an opportunity to compute these applications approximately which could lead orders of magnitude in delay and power benefits.

1.2 Approximate Adders

As mentioned earlier, in approximate computing paradigm, most of the work at hardware level of abstraction has been proposed on arithmetic units [23, 29]. Further, among arithmetic units, adders have attracted a significant attention of researchers/designers [31,32]. Here, we discuss briefly the significance/motivation of approximate adders. **(i) Fundamental arithmetic operators:** As discussed earlier, in digital systems, all basic arithmetic operations, such as addition, subtraction, multiplication and division use adders as the key components. Therefore, being the fundamental and most widely used arithmetic operators, adders have attracted a significant interest for approximation. **(ii) Delay/power bottleneck:** We know that in all digital circuits, delay and power are two conflicting design metrics. However, this situation is more severe in case of adders because delay and power of adders increase rapidly with bit-width. Consequently, at micro-architecture level of abstraction, adders have become one of the key delay/power bottlenecks of digital systems. One possible way to overcome this situation is to approximate the adders, $i.e.$, sacrifice the accuracy for delay and/or power benefits. **(iii) Parametric yield loss:** As shown in Fig. 1.4, the parametric yield loss which is increasing rapidly with CMOS technology scaling has become a matter of concern for the semiconductor industry due to the fact that profits are tied directly to the chip yield [9]. Adders, being the basic building blocks, determine the overall parametric yield of digital systems, particularly, of digital systems which are designed for data-dominated applications. The parametric yield of such digital systems can be improved using approximate adders [1].

Based on the above-mentioned motivations, several approximate adders have been proposed in the literature. For the sake of illustration, we classify these approximate adders into two different categories: (i) *Approximate Full Adders* (AFAs) [1–4, 56–60]; and (ii) *Equal Segment Adders* (ESAs) [12, 14, 61–73]. As shown in Fig. 1.5, in the first approach, an N-bit adder is segmented into two sub-adders: (i) Accurate sub-adder that includes the higher order k bits; and (ii) Approximate sub-adder that includes the remaining lower order $(N - k)$ bits. The length of both sub-adders need not

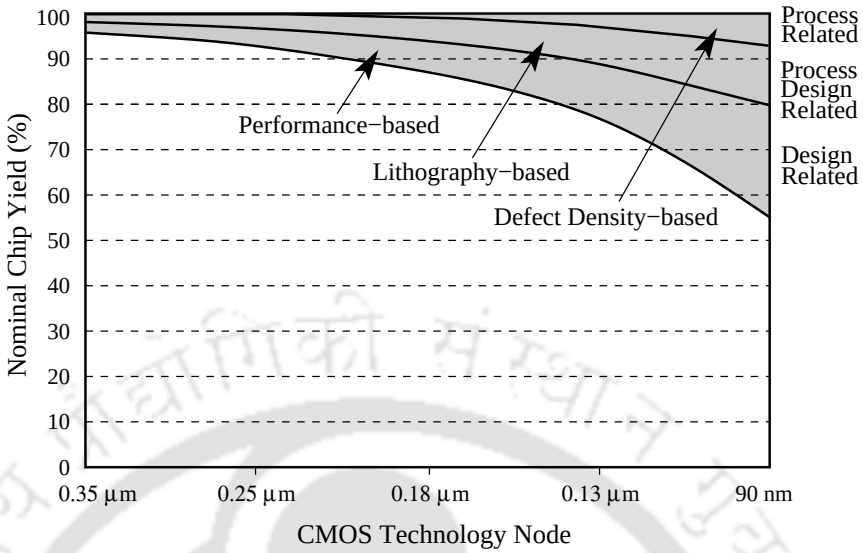


Figure 1.4: In nanometer regime, design related issues begin to exert a more pronounced influence on chip yield as compared to the process related issues [9].

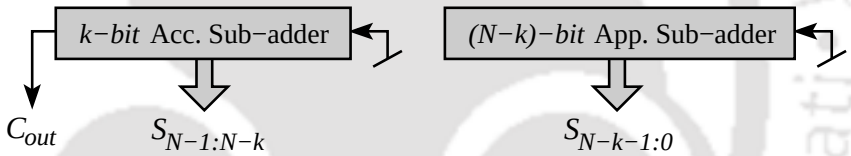


Figure 1.5: Generalized architecture of an N-bit AFA-based approximate adder.

necessary be equal. For accurate sub-adder, conventional carry generation and propagation approach is used. On the other hand, for approximate sub-adder, AFAs are used in which either carry generation and propagation is completely ignored [2, 58] or an approximate carry is generated and propagated [1, 3, 4]. Further, the approximate *Carry-out* (C_{out}) may depend on *Carry-in* (C_{in}) [3, 4] or it may be independent of C_{in} [1]. Any configuration of an AFA-based approximate adder can be specified using the notation AFA(N, k). On the other hand, as shown in Fig. 1.6, in the second approach, an N-bit adder is segmented into several smaller disjoint or overlapping equal size accurate sub-adders. The C_{in} of all sub-adders is considered as 0. Consequently, all sub-adders become independent and operate in parallel. As shown in Fig. 1.6, an N-bit ESA has two primary design parameters: (i) Segment size (k), which represents the maximum length of carry propagation; and (ii) Overlapping bits (l), which represents the minimum number of bits used in carry prediction, where $1 \leq k < N$ and $0 \leq l < k$. Based on the combinations of k and l, an N-bit ESA has $N(N-1)/2$ possible

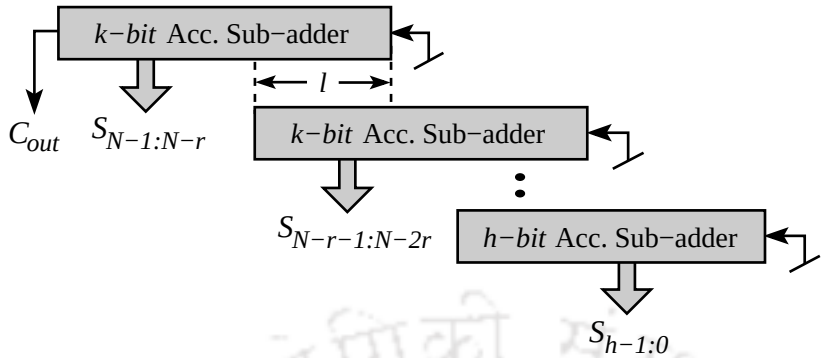


Figure 1.6: Generalized architecture of an N-bit ESA-based approximate adder.

Table 1.1: Primary and secondary design parameters of an N-bit ESA.

k: Segment size or the maximum length of carry propagation; $1 \leq k < N$
l: Overlapping bits or the minimum number of bits used in carry prediction; $0 \leq l < k$
r: Output bits per sub-adder (except the least significant sub-adder) that contribute to the final sum; $r = k - l$
h: Size of the least significant sub-adder; $h = N - (k-l)\lceil\frac{N-k}{k-l}\rceil$
n: Total number of sub-adders; $n = \lceil\frac{N-k}{k-l}\rceil + 1$

configurations. Any configuration of an ESA-based approximate adder can be specified using the notation ESA(N, k, l). In addition to the primary design parameters k and l, an N-bit ESA also has three secondary design parameters: (i) Output bits per sub-adder (except the least significant sub-adder) that contribute to the final sum (r); (ii) Size of the least significant sub-adder (h); and (iii) Total number of sub-adders (n), where $r = k - l$, $h = N - (k-l)\lceil\frac{N-k}{k-l}\rceil$ and $n = \lceil\frac{N-k}{k-l}\rceil + 1$. We call r, h and n as secondary design parameters because of their dependency on k and l. As a summary, these design parameters for an N-bit ESA are tabulated in Table 1.1.

1.3 Quality Metrics

The efficiency of conventional digital circuits is generally evaluated in terms of delay, power and area. On the other hand, in approximate computing paradigm (where the circuits provide approximate results), accuracy is also a concern in addition to the above-mentioned figure of merits. The accuracy of approximate circuits is measured in terms of quality metrics. Based on the requirements of the applications, several quality metrics have been introduced in the literature [1, 62, 65, 74, 75]. The overall accuracy of an approximate circuit is often the result of two concurring quality metrics: (i) *Error Dis-*

tance (ED); (ii) and *Error Rate* (ER). Therefore, ED and ER are considered as the fundamental quality metrics [22]. Besides ED and ER, other commonly used composite quality metrics are *Relative Error Distance* (RED) [23, 69, 75], *Mean Error Distance* (MED) [31, 60, 69, 75–81], *Mean Relative Error Distance* (MRED) [31, 72, 82], *Minimum Acceptable Accuracy* (MAA) [62, 69], *Accuracy of Amplitude* (ACC_{amp}) [65, 69] and *Accuracy of Information* (ACC_{inf}) [65, 69]. In addition to these generic quality metrics, there are application-specific quality metrics also, such as *Mean Squared Error* (MSE) [22, 76, 77, 79, 81, 83, 84] and *Peak Signal-to-Noise Ratio* (PSNR) [22, 31, 77, 81, 83] in case of multimedia applications. Note that the choice of a particular quality metric depends on the applications. For example, in *Digital Signal Processing* (DSP) applications, ED is important, whereas in communication applications, ER is important. Further, ER is important when the erroneous results need to be handled or recovered using *Error Detection and Correction* (EDC) logic, such as in case of *Accuracy Configurable Adders* (ACAs) and *Variable Latency Speculative Adders* (VLSAs) [14, 61, 65, 69, 70, 73, 85, 86]. It has been observed in [76, 78] that in approximate arithmetic circuits, the combination of ED, ER, MED and MSE is sufficient to evaluate the overall accuracy. Consequently, as compared to other quality metrics, we emphasis more on these four quality metrics. A brief description of these four quality metrics is as follows.

(i) For an input, ED is the arithmetic distance between the accurate and approximate outputs as:

$$ED = \mid S_{acc} - S_{app} \mid \tag{1.1}$$

where, S_{acc} is the output of an accurate adder and S_{app} is the output of an approximate adder.

(ii) For an input vector (or set of inputs), ER is the probability of output to be incorrect as:

$$ER = \frac{1}{v} \sum_{i}^{v} P(ED_i \neq 0) \tag{1.2}$$

where, v is the size of input vector (or input set size) and $P(ED_i \neq 0)$ is 1 if $ED_i \neq 0$, otherwise $P(ED_i \neq 0)$ is 0.

(iii) MED is a composite quality metric which encompasses the information of both ED and ER. For an input vector, MED is the mean of all EDs as:

$$MED = E[ED] = \sum_{i}^{v} ED_i P(ED_i) \tag{1.3}$$

where, E[x] represents the expected value of variable x.

(iv) In case of MED (Eqn. (1.3)), both ED and ER are given the same importance. However, ED and ER can be assigned different importances through appropriate weighting. MSE is one of the such quality metrics in which ED is given more importance as compared to the ER. For an input vector, MSE is the mean of all squared EDs as:

$$MSE = E[ED^2] = \sum_{i}^{v} ED_i^2 P(ED_i) \tag{1.4}$$

1.4 Thesis Motivations

The key motivations of this thesis are as follows.

(i) The design metrics, such as accuracy, delay, power and area of an approximate adder can be evaluated either through computer simulations or through analytical models. While evaluating the design metrics through computer simulations, an approximate adder can be simulated either exhaustively ($i.e.$, for all 2^{2N} possible inputs) or for a fixed number of inputs generated using *Monte-Carlo* methods. Exhaustive simulations provide the exact design metrics, but the required simulation time makes them infeasible. In order to evaluate the simulation time, we exhaustively simulate single configuration of an N-bit ESA for $N = 1$ to 16 using *Dell precision R7610 rack workstation* which consists of two 8-core *Intel Xeon E5-2650 processors* operating at a clock frequency of $2GHz$ [10]. Fig. 1.7 shows the simulation time as a function of N. It can be seen from Fig. 1.7 that the required simulation time increases exponentially with bit-width. Consequently, for smaller values of N (particularly, for $N \leq 12$ [78]), simulations can be performed exhaustively, however, for higher values of N, exhaustive simulations become infeasible. For example, exhaustive simulations of a 16-bit ESA for all possible configurations require ≈ 1.44 years. One possible way to handle the simulation time is to simulate approximate adders for a fixed number of inputs generated using Monte-Carlo methods. However, as

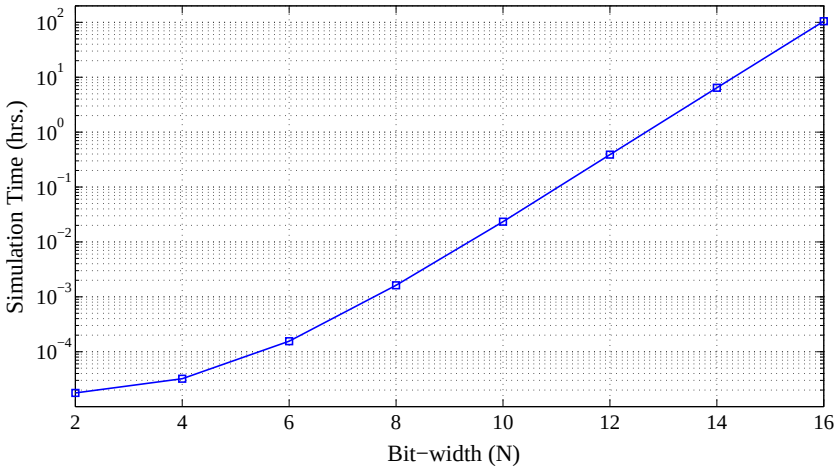


Figure 1.7: Simulation time versus bit-width for exhaustive simulation of a single configuration of an N-bit ESA for different values of N. The simulations are run on *Dell precision R7610 rack workstation* which consists of two 8-core *Intel Xeon E5-2650 processors* operating at a clock frequency of $2GHz$ [10].

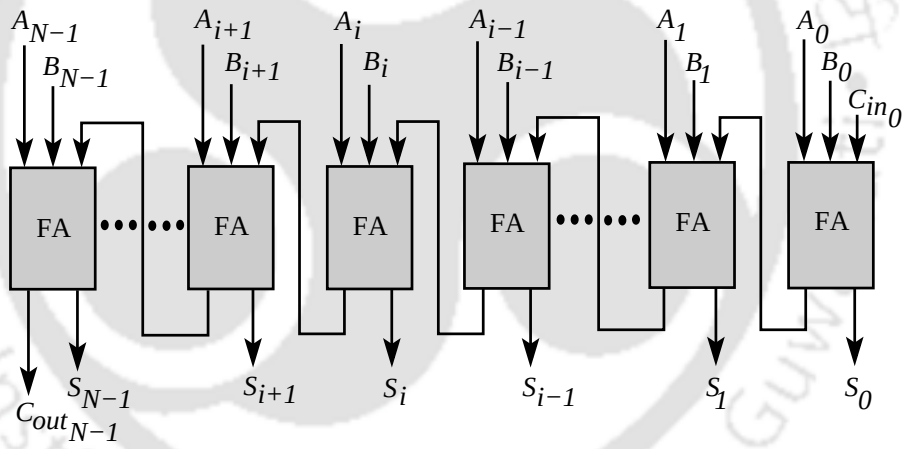


Figure 1.8: Generalized architecture of an N-bit RCA designed by cascading N FAs in series.

discussed in [78], Monte-Carlo based simulations have their own limitations. Therefore, there is a need of analytical modeling of approximate adders.

(ii) Among all existing conventional adders, RCA is the simplest one. As shown in Fig. 1.8, an N-bit RCA is generally constructed by cascading N *Full Adders* (FAs) in series. An emergent property of RCA is that it exhibits the smallest silicon area. However, in RCA, C_{out} of i^{th} FA is feed directly to the C_{in} of $(i+1)^{th}$ FA, and thus, it is significantly slow as carries ripple in a serial manner. Therefore, while designing a FA for RCA, the delay to compute sum is unimportant, whereas the delay to compute $C_{in} \rightarrow C_{out}$ should be minimized. Moreover, a significant portion

Table 1.2: ED_{max} and ER of a 32-bit ESA as a function of k and l.

	k	l								
		0	1	2	3	4	5	6	7	8
ED_{max}	8	1.68E+7	3.35E+7	6.71E+7	1.34E+8	2.69E+8	5.37E+8	1.07E+9	2.15E+9	—
	10	4.19E+6	8.38E+6	1.67E+7	3.35E+7	6.71E+7	1.34E+8	2.68E+8	5.37E+8	1.07E+9
	12	1.04E+6	2.09E+6	4.19E+6	8.38E+6	1.67E+7	3.35E+7	6.71E+7	1.34E+8	2.68E+8
	14	2.62E+5	5.24E+5	1.04E+6	2.09E+6	4.19E+6	8.38E+6	1.67E+7	3.35E+7	6.71E+7
	16	6.55E+4	1.31E+5	2.62E+5	5.24E+5	1.04E+6	2.09E+6	4.19E+6	8.38E+6	1.67E+7
ER	8	0.87352	0.68028	0.40856	0.26822	0.16339	0.10428	0.06809	0.04583	—
	10	0.87463	0.57730	0.32895	0.22591	0.11748	0.07342	0.04314	0.02701	0.01599
	12	0.74987	0.43731	0.23416	0.17570	0.09050	0.04579	0.03040	0.01505	0.00912
	14	0.74996	0.43745	0.23432	0.12103	0.06146	0.03094	0.02316	0.01158	0.00575
	16	0.49999	0.43748	0.23436	0.12107	0.06150	0.03099	0.01554	0.00778	0.00388

of dynamic power consumption in RCA is due to re-switching of internal nodes when carries propagate from *Least Significant Bits* (LSBs) to *Most Significant Bits* (MSBs). One possible way to overcome the above-mentioned limitations is to truncate the length of carry propagation. For truncating the length of carry propagation, AFAs are used in which C_{out} is independent of C_{in}. In a FA, to secure C_{out} independent of C_{in}, C_{out} should be either 0, 1 or a Boolean function of the primary inputs A and B only. In such a scenario, several AFAs are possible. From our analysis, we observe that these AFAs exhibit different logic complexity and error probability. Therefore, designers need to choose the optimal one.

(iii) As mentioned earlier, based on the combinations of k and l, an N-bit ESA has $N(N-1)/2$ possible configurations. Table 1.2 shows ED_{max} and ER of a 32-bit ESA for different configurations ($i.e.$, combinations of k and l), where ED_{max} is the maximum possible magnitude of error. It can be seen from Table 1.2 that in an N-bit ESA, there exist multiple configurations which exhibit similar accuracy. For example, ESA(32, 12, 1), ESA(32, 14, 3) and ESA(32, 16, 5) have similar ED_{max}. Further, ESA(32, 8, 7), ESA(32, 10, 6) and ESA(32, 12, 5) have similar ER. Therefore, an interesting question that may arise here is – for a given accuracy, what are the optimal configurations which provide minimal delay, power and/or area. These optimal configurations need to be known apriori as the prior knowledge of such optimal configurations can lead to efficient, intelligent and goal oriented implementations of ESAs. Further, we know that

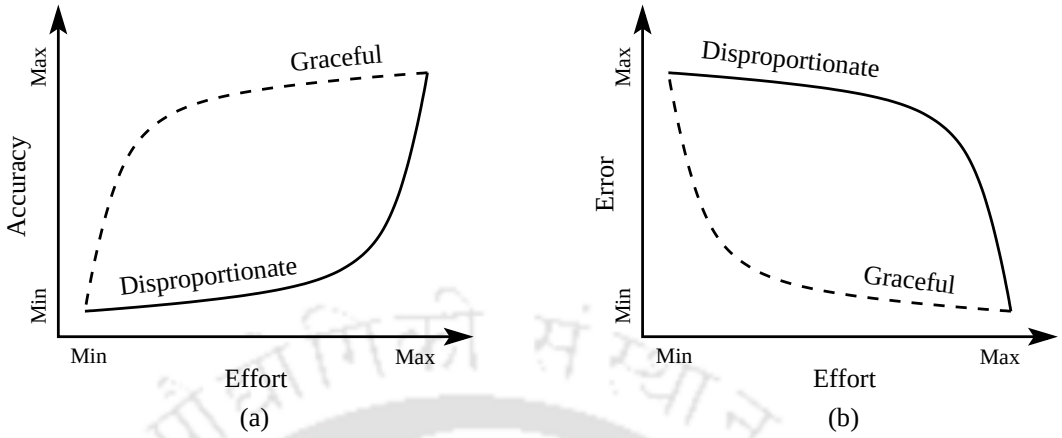


Figure 1.9: (a) Accuracy-effort curves; and corresponding (b) Error-effort curves.

accuracy of an ESA does not depend on the adder architecture used to implement it, however, its delay, power and area depend significantly. Therefore, the optimal configurations vary with adder architectures used to implement the ESA. Accordingly, this type of optimization analysis need to be carried out considering different adder architectures.

(iv) In digital circuit design, the efficiency of conventional circuits is generally evaluated using the delay-power curves. On the other hand, as shown in Fig. 1.9, in approximate computing paradigm (where the circuits provide approximate results), researchers/designers use accuracy-effort curves to evaluate the efficiency of approximate circuits. Note that accuracy-effort curves describe the relationship between the accuracy we can achieve and the efforts we need to pay in return. In case of approximate arithmetic circuits, accuracy is measured in terms of quality metrics (ED, ER, MED and MSE) and efforts are measured in terms of delay, power and area. Therefore, accuracy-effort curves evaluate – how much optimal delay-accuracy, power-accuracy and area-accuracy trade-offs an approximate arithmetic circuit can provide. From our analysis, we observe that there is a scope of improvement in the accuracy-effort curves of ESAs. The accuracy-effort curves of ESAs need to be improved so that they provide optimal delay-accuracy, power-accuracy and area-accuracy trade-offs.

(v) Over the past decade, several approximate adders have been proposed in the literature. In most of the work, researchers/designers examine the feasibility of approximate adders in multimedia

applications. However, they also find applicability in other state-of-the-art applications. As different applications have different level of error-resilience, the effectiveness of approximate adders vary with the applications. Further, in different applications, there are different methods of introducing approximation. For example, in cryptography applications, approximation can be introduced either by reducing the number of rounds or by computing the rounds approximately using approximate adders. In addition to the delay and power benefits, approximate adders can also be used to address other technology/design related issues, such as yield loss [1]. Further, we know that all basic arithmetic operations, such as addition, subtraction, multiplication and division use adders as the key components. Therefore, approximate adders can be utilized to design other approximate arithmetic operations [87–89]. All theses points need to be explored to maximize the benefits of approximate adders.

1.5 Thesis Contributions

The key contributions of this thesis are as follows.

1.5.1 Contribution in AFAs

We propose four AFAs with design objective that C_{out} should be independent of C_{in} with minimal ER. Subsequently, we exploit one of the proposed AFAs (the optimal one) for designing an N-bit approximate adder which we call as "ApproxADD". An emergent property of ApproxADD is that carries do not propagate in it, and thus, it provides bit-width-aware constant delay ($O(1)$). ApproxADD also shows improvement in dynamic power consumption by 46.31% and in area by 28.57% $w.r.t.$ RCA. Although ApproxADD provides a significant improvement in delay, power and area, it may not be preferred for some of the error-resilient applications due to the fact that its: (i) ED is too high; and (ii) ER increases rapidly with bit-width. We know that in arithmetic operations, the impact of errors in higher order bits is more severe as compared to the errors in lower order bits. Therefore, in order to improve the ED of ApproxADD, we perform higher order k-bit operations accurately using FAs and lower order $(N-k)$-bit operations approximately using the proposed AFAs. We call this version as "ApproxADD$v1$". ApproxADD$v1$ provides a significant improvement in ED, but its ER is still

high, particularly, for $k \leq 8$. Consequently, ApproxADD$v1$ may not be preferred for the applications which demand lower ER. In order to improve the ER of ApproxADD$v1$, we propose an EDC logic that detects errors, and accordingly, corrects them. We call this version as "ApproxADD$v2$". In addition to the designing of ApproxADDs, we also provide a detailed analysis and analytical modeling to estimate accuracy, delay, power and area of ApproxADDs.

We evaluate the efficiency of ApproxADDs by comparing them with existing AFA-based approximate adders [1, 2, 5, 58]. Further, in order to inspect the effectiveness of the proposed approach in real-life applications, we demonstrate image compression and decompression by replacing the conventional addition operations in *Discrete Cosine Transform* (DCT) and *Inverse Discrete Cosine Transform* (IDCT) modules with ApproxADD$v2$. For having a fair evaluation of the proposed approach: (i) We use four different images (Lena, Cameraman, Mandrill and Girl [90]); and (ii) We evaluate output images both subjectively as well as objectively [91].

1.5.2 Contribution in ESAs

Due to the additional features of design flexibility and programmability, ESAs have attracted a lot of attention of researchers and industry as compared to the AFAs. Our key contributions (analysis, design, modeling and optimization) for ESAs are as follows.

1.5.2.1 Analytical Modeling

We know that computer simulations have several limitations over analytical models, such as programming efforts, time-consuming simulations, etc. We propose analytical models to estimate accuracy, delay, power and area of ESAs. To the best of our knowledge, no favorable work has been proposed till now on analytical modeling of delay, power and area of ESAs. We the first time propose such analytical models considering different adder architectures. Further, the key features of the proposed analytical models of accuracy (ED, ER, MED and MSE) are that: (i) They are generalized, $i.e.$, work for all possible configurations of an N-bit ESA, whereas the existing analytical models are for specific configurations; and (ii) They are superior ($i.e.$, estimate more accurately) or at par to the existing analytical models [63, 65, 76, 77, 83]. All together, the proposed analytical models can assist

designers to estimate the design metrics of any configuration of an N-bit ESA without going for the programming efforts and time-consuming simulations.

1.5.2.2 Optimization

As discussed earlier, in an N-bit ESA, there exist multiple (more than one) configurations which exhibit similar accuracy, however, these configurations exhibit different delay, power and area. Therefore, for a given accuracy, to apriori select the optimal configurations which provide minimal delay, power and/or area is a challenging decision for designers. We propose an optimization framework which can assist designers in making such important decisions early in the design phase of ESAs. This enables designers for efficient, intelligent and goal oriented implementations of ESAs. Note that finding the optimal configurations is a constrained optimization problem, $i.e.$, minimize or maximize objective functions in the presence of constraints. Further, objective functions and constraints depend on the applications. In our analysis, we consider delay, power and/or area as the objective functions and ED, ER, MED and/or MSE as the constraints. Over the decades, several nature-inspired heuristic algorithms have been proposed in the literature for constrained optimization [92]. For the sake of demonstration of the proposed optimization framework, we use *River Formation Dynamics* (RFD) heuristic algorithm. Further, we know that accuracy of an ESA does not depend on the adder architecture used to implement it, however, its delay, power and area depend significantly. Therefore, the optimal configurations vary with adder architectures used to implement the ESA. In order to cover a wide range of adders, we consider three types of architecture in our analysis: (i) Architectures having smaller area ($O(N)$); (ii) Architectures having smaller delay ($O(log_2 N)$); and (iii) Architectures having in-between delay ($O(N/4)$) and area ($O(2N)$).

1.5.2.3 Accuracy Enhancement

From the proposed analytical models, we observe that there is a scope of improvement in accuracy-effort curves of ESAs. The accuracy-effort curves of ESAs need to be improved so that their accuracy degrade in a graceful manner, and consequently, they provide optimal delay-accuracy, power-accuracy and area-accuracy trade-offs. In order to improve the accuracy-effort curves of ESAs, we propose modifications in existing ESAs. Note that the accuracy-effort curves can be improved either: (i) By

improving the accuracy without imposing any additional delay, power and area overheads; or (ii) By reducing the delay, power and area without loosing accuracy. We achieve our objective using the former approach. The crux of the proposed modifications is that $0.25 \leq P(G_i) \leq 0.50$ in original ESAs, whereas it is 0.25 in modified ESAs, where $P(G_i)$ represents the probability of C_{in} of sub-adders to be incorrect. With this improvement, modified ESAs provide higher accuracy without imposing any additional overhead as compared to the original ESAs.

1.5.3 Application of Approximate Adders

Over the past decade, several approximate adders have been proposed in the literature [1–4, 12, 14, 56–73]. The feasibility/applicability of approximate adders need to be explored to maximize their benefits. We explore the effectiveness of approximate adders in cryptography applications, yield enhancement and designing other approximate arithmetic operations.

1.5.3.1 Cryptography Applications

Cryptographic hash functions play an important role in today's E-commerce as they are the basic building blocks of numerous security applications [93]. *Secure Hash Algorithm 1* (SHA-1) is one of the most widely used cryptographic hash functions and has many future aspects. The overall delay, power and area of SHA-1 are determined by modular-32 adders. In order to examine the effectiveness of approximate adders in cryptographic applications, we replace the conventional modular-32 adders with ApproxADD. We call the resulting approximate SHA-1 as "ApproxSHA-1". We use ApproxADD because in cryptography applications, each output bit has same weight, and thus, the number of incorrect bits matters irrespective of their positions. For verifying the security level of ApproxSHA-1, we perform: (i) Avalanche test; (ii) Maurer's universal statistical test; (iii) Statistical Mobius analysis; and (iv) Near collision test. We conduct first three tests sequentially. If ApproxSHA-1 passes current test, then only we conduct the next test; otherwise, we reduce the approximation level and re-run the test. Finally, we implement SHA-1 and ApproxSHA-1 on Virtex-6 *Field Programmable Gate Arrays* (FPGA) for their delay, power and area evaluation.

1.5.3.2 Yield Enhancement

With continued innovations in fabrication process steps, CMOS technology is moving toward finer geometries, exhibiting higher performance, higher energy efficiency and lower silicon area per computation. However, with the relentless scaling of CMOS technology in sub-nanometer regime, chip yield has become a matter of concern for the semiconductor industry due to the fact that profits are tied directly to the chip yield [9]. We explore the effectiveness of approximate adders for yield enhancement. We propose analytical models to estimate the functional yield and parametric yield of approximate arithmetic circuits. From the proposed analytical models we observe that approximate data path modules can improve the functional yield and parametric yield due to decrease in area and delay, respectively. We demonstrate this in respect to ApproxADDs.

1.5.3.3 MAC Unit

Multiply-and-Accumulate (MAC) unit is the heart of multimedia applications. As MAC unit lies in the critical path, it determines the overall delay, power and area of multimedia systems. We know that all basic arithmetic operations, such as addition, subtraction, multiplication and division use adders as the key components. In order to examine the effectiveness of approximate adders for designing other approximate arithmetic operations, we design an approximate multiplier using approximate adders. Further, we design an approximate MAC unit using approximate multiplier and adder. We call the resulting approximate MAC unit as "ApproxMAC". We evaluate the feasibility of ApproxMAC unit in real-life applications through an image processing application.

1.6 Thesis Organization

Based on the above-discussed contributions, this thesis is organized into six different chapters. The chapter-wise contents of the rest of this thesis are as follows.

Chapter 2: Literature Survey

In this chapter, we first discuss design philosophy and design approach of approximate adders. Further, we present a survey of state-of-the-art approximate adders (both AFA-based as well as ESA-

based). In the end of this chapter, we briefly discuss different directions in which the research work is recently going on in respect to approximate adders.

Chapter 3: AFA-based Approximate Adders

In this chapter, we discuss the proposed AFAs. Further, we present ApproxADD – an N-bit approximate adder designed using the proposed AFAs. We explain our strategies (concept of *carry-lifetime* and EDC logic) used to improve the ED and ER of ApproxADD. In this way, we introduce two more (improved) versions of ApproxADD – ApproxADD$v1$ and ApproxADD$v2$. We provide analytical models to estimate the accuracy, delay, power and area of ApproxADDs. We validate the proposed analytical models through simulation results. We evaluate the effectiveness of the proposed ApproxADDs by comparing them with existing AFA-based approximate adders. Further, in order to inspect the effectiveness of the proposed approach in real-life applications, we demonstrate image compression and decompression using ApproxADD$v2$.

Chapter 4: ESA-based Approximate Adders

In this chapter, we discuss the proposed analytical models to estimate the accuracy (ED, ER, MED and MSE), delay, power and area of ESAs. We validate the proposed analytical models by comparing them with simulation results as well as with existing analytical models. Based on the analytical models and simulation results, we discuss some important observations regarding the design metrics of ESAs. Further, we present an optimization framework that exploits the proposed analytical models to find the optimal configurations of an N-bit ESA which provide minimal delay, power and/or area for a given accuracy. In the end, we discuss modified ESAs.

Chapter 5: Application of Approximate Adders

In this chapter, we evaluate the effectiveness of approximate adders in cryptography applications, yield enhancement and designing other approximate arithmetic operations. We present an approximate SHA-1 (ApproxSHA-1) and a framework to determine the maximum level of approximation in ApproxSHA-1. Further, we discuss the proposed analytical models to estimate the functional yield and parametric yield of approximate arithmetic circuits. Based on the proposed analytical models, we

demonstrate the yield enhancement in respect to ApproxADDs. In the end, we present an approximate MAC unit (ApproxMAC) and its feasibility in real-life applications.

Chapter 6: Conclusion and Future Aspects

In this chapter, we conclude the work presented in this thesis. In addition to the contributions and benefits of this thesis, we also discuss some of the very important design challenges and future aspects of approximate adders (AFAs, ESAs, ACAs and VLSAs).

2

Literature Survey

Contents

There are two fundamental approaches used for designing the approximate adders: (i) AFA; and (ii) ESA. We briefly discussed these approaches in Chapter 1. Over the past decade, several approximate adders have been proposed in the literature based on these approaches [1–4, 12, 14, 56–72]. In this Chapter, we first discuss the design philosophy and design approach of approximate adders. Further, we present a survey of state-of-the-art AFA-based and ESA-based approximate adders. We also briefly discuss different directions in which the recent research work is going on in respect to approximate adders. In the end, we summarize the Chapter.

The rest of this Chapter is organized as follows. Section 2.1 presents the design philosophy and Section 2.2 presents the design approach of approximate adders. Section 2.3 and Section 2.4 provide the literature survey of AFAs and ESAs, respectively. Section 2.5 discusses the research directions of approximate adders. Finally, Section 2.6 concludes the Chapter.

2.1 Design Philosophy

We know that the critical path in conventional adders depends on the length of carry propagation. In the worst-case scenario, carry generates at LSBs and propagates to MSBs. In such conditions, the length of carry propagation is close to bit-width. However, in real-life applications, such conditions rarely happen and in most of the cases, the length of carry propagation is much shorter as compared to the bit-width. It has been observed that in most of the cases, the longest length of carry propagation in an N-bit conventional adder is close to log_2N [62, 94, 95] and is less than $log_2N + 12$ [61] with extremely high probability. In order to evaluate the length of carry propagation, we implement N-bit conventional adders in C/C++ for different values of N. We then simulate them individually for one billion pseudo-random inputs drawn from a sample space between 0 and $2^N - 1$. Our simulation results for $N = 8$, 16, 32, 64 and 128 are tabulated in Table 2.1. It can be seen from Table 2.1 that in 99% cases, the length of carry propagation is less than $log_2N + C$, where C is a bit-width dependent constant. Consequently, in an N-bit conventional adder, if carry is computed by considering $log_2N + C$ input bits on the right of i^{th} bit position, then the probability of getting correct carry at i^{th} bit position is greater than 99%. Further, as C increases, the probability of getting correct carry increases. For example, in a 64-bit conventional adder, for $C = 1$, the probability of getting correct carry is

Table 2.1: Probability (in %) of the length of carry propagation to be less than (or at least equal to) $log2N + C$ in N-bit conventional adder for different values of N.

Bit-width (N)	Length of Carry Propagation			
	log_2N	$log_2N + 1$	$log_2N + 2$	$log_2N + 3$
8	92.1254	96.8469	98.8179	99.5021
16	95.3017	97.8502	99.0171	99.6604
32	97.3623	98.7318	99.3934	99.7078
64	98.5489	99.2879	99.6508	99.8303
128	99.2879	99.6504	99.8282	99.9159

99.29%, whereas for $C = 3$, it is 99.83%. In addition to these, it should be noted from Table 2.1 that the logarithmic term (log_2N) increases, whereas the bit-width dependent constant (C) decreases gradually with bit-width. Due to this conflict, the length of carry propagation may be independent of bit-width or it may increase very slowly with bit-width. For example, 99% carry propagation has a length of 6, 6, 7, 7 and 8 for $N = 8$, 16, 32, 64 and 128, respectively. Note that a similar observation has been reported in [61, 62]. Further, using Table 2.1, the probability that length of carry propagation is less than or equal to p bits can be empirically given by $1 - 1/2^p$.

2.2 Design Approach

In recent years, several approximate adders have been proposed in the literature [1–4, 12, 14, 56–72]. Based on the above-mentioned observations, the key design approach behind approximate adders is to truncate the length of carry propagation. With the reduced length of carry propagation, approximate adders can provide high performance (by increasing the clock frequency) or low power (by decreasing the supply voltage). As discussed in Chapter 1, the two fundamental approaches used to truncate the length of carry propagation are: (i) AFA [1–4, 56–60]; and (ii) ESA [12, 14, 61–72]. A comparative survey/review of some of these approximate adders can be found in [23, 32] and their libraries are available at *http://www.fit.vutbr.cz/research/groups/ehw/approxlib/* [74] and *https://sourceforge.net/projects/approxadderlib/* [85]. Note that both these design approaches have their own pros and cons. The key feature of AFAs is that they provide a significant reduction in logic complexity and power consumption. On the other hand, the key features of ESAs is that they provide a high degree of design flexibility and programmability.

2.3 State-of-the-Art AFAs

We know that in arithmetic operations, the impact of errors in higher order bits is more severe as compared to the errors in lower order bits. Therefore, as shown in Fig. 1.5, in AFA-based approximate adders, the higher order k-bit operations are performed accurately using conventional carry generation and propagation approach, whereas the remaining lower order $(N-k)$-bit operations are performed approximately using AFAs. In recent years, several AFAs have been proposed in the literature [1–4, 56–60]. AFAs are generally designed by modifying the conventional FAs at gate level [1, 2, 56–58] and transistor level [3, 4] of abstraction. While designing an AFA, the key objective is to reduce the logic complexity subjected to minimal ED and/or ER. Note that truncated adder is a special case of AFA-based approximate adders in which the lower order $(N-k)$ bits are totally ignored. Therefore, it is considered as a base case of AFA-based approximate adders. It has been observed in [56] that for the similar accuracy, an AFA-based approximate adder consumes less power as compared to the truncated adder. This is also one reason that AFAs are gaining prominence. Further, independent of the bit-width (N), the accuracy of an AFA-based approximate adder depends on the number of lower order $(N-k)$ bits which are performed approximately using AFAs. Here, we discuss state-of-the-art AFAs (both at gate level and transistor level) in detail.

2.3.1 Gate Level AFAs

Consider a FA which is the basic building block of commonly used adder architectures. A FA can be expressed by the well-known Boolean functions as:

$$S = A \oplus B \oplus C_{in} \tag{2.1}$$

$$C_{out} = A.C_{in} + B.C_{in} + A.B \tag{2.2}$$

where A, B and C_{in} are the binary inputs, and $Sum\ (S)$ and C_{out} are the binary outputs. Using Eqn. (2.1) and Eqn. (2.2), there are different ways (both at transistor level and gate level of abstraction) to implement a FA [6]. The most commonly used gate level implementation is shown in Fig. 2.1(a) and transistor level implementation is shown in Fig. 2.3(a).

Table 2.2: Examples of gate level design simplifications.

Original Functions	AND	OR	NAND	NOR	XOR	XNOR
Simplified Functions	0	1	1	0	OR	NOR
ER	0.25	0.25	0.25	0.25	0.25	0.25

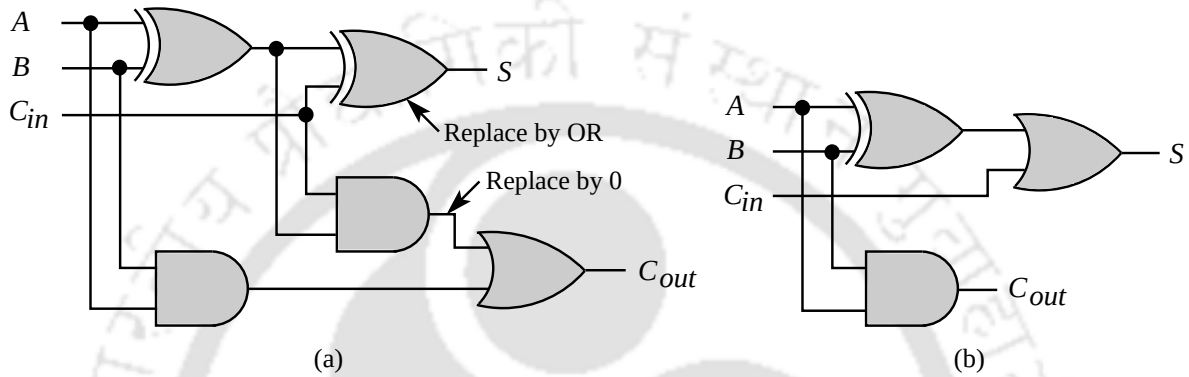


Figure 2.1: Gate level implementations: (a) Conventional FA; and (b) SFA [1].

2.3.1.1 Simplified Full Adder [1]

The straightforward way of designing an AFA is to simplify the on-path logic gates individually. For example, we know that in a 2-input AND gate, the output is 0 in 3 out of 4 input combinations. Therefore, in the critical path of a circuit, if we replace a 2-input AND gate with 0, then the probability of output to be incorrect is $1/4$. Note that the probability of output to be incorrect further reduces as the number of inputs increases, $e.g.$, if we replace an n-input AND gate with 0, then the probability of output to be incorrect is $1/2^n$. By doing so, we cut the critical path as well as reduce the logic complexity. Few examples of gate level design simplifications for 2-input logic gates are shown in Table 2.2. Assuming that the inputs are independent and uniformly distributed, Table 2.2 also summarizes the ER of simplified functions. Based on this gate level simplification approach, Fig. 2.1(b) shows *Simplified Full Adder* (SFA) [1]. In addition to replacing the output of AND gate by 0, the authors also replace XOR gate by OR gate as this reduces ED while keeping the ER intact. This is because for any combination of inputs for which there is an error due to the simplification of AND gate, there is another error due to the simplification of XOR gate that reduces ED by causing a compensating deviation in the output (see Table 2.3).

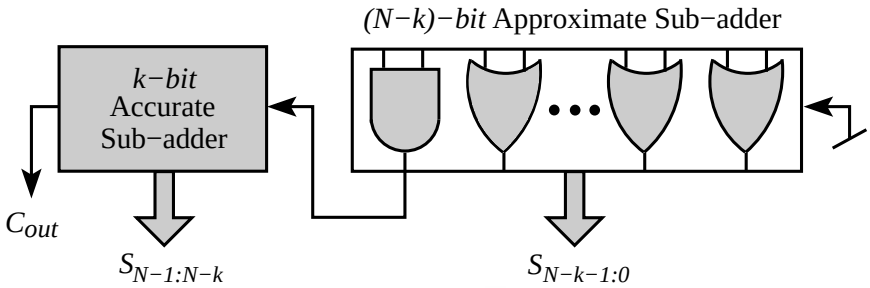


Figure 2.2: Generalized architecture of an N-bit LOA [2].

2.3.1.2 Lower-part-OR Adder [2]

As shown in Fig. 2.2, in *Lower-part-OR Adder* (LOA) [2], the lower order $(N-k)$-bit operations are performed approximately using OR gates. Note that the authors use an additional AND gate to generate C_{in} for the accurate sub-adder when the MSBs of approximate sub-adder ($i.e.$, A_{N-k-1} and B_{N-k-1}) are 1. This helps to consider the trivial carries from approximate sub-adder to improve the accuracy. Note that a similar approach has been discussed in [56].

The truth table of SFA and LOA are shown in Table 2.3. Note that in Table 2.3, tick mark denotes a match with the accurate output and cross mark denotes an error. Assuming that the inputs are independent and uniformly distributed, Table 2.3 also summarizes the ER of SFA and LOA. Further, the logic complexity in terms of the number of 2-input basic gates (AND, OR, NAND and NOR) is mentioned in Table 2.3. As shown in Table 2.3, conventional FA requires 9 2-input basic gates. On the other hand, SFA requires 5 and LOA requires only 1. Further, SFA has 2 errors in each C_{out} and S, whereas LOA has 4 errors in each. Although ER of LOA is double than the SFA, but it provides an enormous improvement in logic complexity (88.89% $w.r.t.$ FA).

2.3.2 Transistor Level AFAs

At transistor level of abstraction, AFAs are designed by removing the transistors from conventional FAs. Note that we can not remove the transistors in an arbitrary fashion. We need to make sure that any input combination does not result in short circuit or open circuit. Another important criterion is that the resulting AFA should introduce minimal ED and/or ER. The key feature of transistor level AFAs is that they provide additional dynamic power saving due to the reduction in switched capaci-

Table 2.3: Truth table and design metrics of gate level AFAs.

Inputs			Outputs					
			FA		SFA [1]		LOA [2]	
A	B	C_{in}	C_{out}	S	C_{out}	S	C_{out}	S
0	0	0	0	0	0 ✓	0 ✓	—	0 ✓
0	0	1	0	1	0 ✓	1 ✓	—	0 ×
0	1	0	0	1	0 ✓	1 ✓	—	1 ✓
0	1	1	1	0	0 ×	1 ×	—	1 ×
1	0	0	0	1	0 ✓	1 ✓	—	1 ✓
1	0	1	1	0	0 ×	1 ×	—	1 ×
1	1	0	1	0	1 ✓	0 ✓	—	1 ×
1	1	1	1	1	1 ✓	1 ✓	—	1 ✓
ER			0	0	0.25	0.25	—	0.5
#Gates			9		5		1	

tance. Further, removal of the transistors facilitates faster charging/discharging of node capacitances which results in shorter delay as compared to the gate level AFAs.

2.3.2.1 Approximate Mirror Adders [3]

In general, CMOS implementation of a FA requires 32 transistors [6]. However, a more compact design is possible based on the observation that S can be factored to reuse C_{out} as:

$$S = A.B.C + (A + B + C).\overline{C}_{out} \tag{2.3}$$

Such a design is shown in Fig. 2.3(a) and requires only 28 transistors (including the 4 transistors required to derive C_{out} and S from $\overline{C}_{out}$ and $\overline{S}$, respectively). Note that in Fig. 2.3(a), *P-Type Metal-Oxide-Semiconductor* (PMOS) network and *N-Type Metal-Oxide-Semiconductor* (NMOS) network are identical to each other, rather than being the complement. Therefore, this topology is called as *Mirror Adder* (MA) in the literature. MA is one of the most widely used economical implementations of FA at transistor level of abstraction. MA reduces the number of series connected transistors and makes the layout more uniform. It is possible because the addition function is symmetric, $i.e.$, the function of complemented inputs is the complement of function. MA has a greater delay to compute S than C_{out}. However, in RCA, the critical path goes from C_{in} to C_{out} through many FAs, and thus, the extra delay required in computation of S is unimportant.

Gupta et al. [3] propose four *Approximate Mirror Adders* (AMAs) by removing the transistors from MA. A judicious selection of transistors to be removed (ensuring no open circuit or short circuit) results in the schematics, as shown in Fig. 2.3. The truth table and ER of AMAs are shown in Table 2.4. Further, the logic complexity in terms of transistor count is also mentioned in Table 2.4. It can be seen from Fig. 2.3 and Table 2.4 that MA requires 28 transistors, whereas AMA#1 requires 20 transistors. In AMA#1, there is 1 error in C_{out} and 2 errors in S.

In the truth table of MA, $S = C_{out}$ in 6 out of 8 input combinations. Further, as shown in Fig. 2.3(a), in MA, C_{out} is computed in the first stage. Thus, an easy way to get a simplified schematic is to set $S = C_{out}$. However, as shown in Fig. 2.3(c), the authors use a buffer for implementing the same functionality. The reason for this can be explained as follows. If we set $S = C_{out}$, then the total capacitance at S node would be a combination of 4 source-drain diffusion and 2 gate capacitances. This is a considerable increase as compared to the MA or AMA#1 and it would lead to delay penalty when multi-bit approximate adders are connected in series. Fig. 2.3(c) shows AMA#2 obtained using the above-mentioned approach. As shown in Table 2.4, AMA#2 requires 16 transistors and has 2 errors in S, while C_{out} is correct for all input combinations.

More simplified schematic can be further obtained by combining AMA#1 and AMA#2, $i.e.$, compute C_{out} similar to AMA#1 and S similar to AMA#2. The corresponding schematic is shown in Fig. 2.3(d). As shown in Table 2.4, AMA#3 requires 13 transistors and has 1 error in C_{out} and 3 errors in S. Note that AMA#3 requires the lowest number of transistors.

Further, a close observation of the truth table of MA shows that $C_{out} = A$ for 6 out of 8 input combinations. Similarly, $C_{out} = B$ for 6 out of 8 input combinations. Since A and B are interchangeable, we can consider the either. The corresponding schematic is shown in Fig. 2.3(e) in which C_{out} is computed using an inverter and S is computed similar to AMA#1. As shown in Table 2.4, AMA#4 requires 15 transistors and has 2 errors in C_{out} and 3 errors in S.

2.3.2.2 Approximate XOR/XNOR-based Adders [4]

Designing the logic gates using *Pass Transistor Logic* (PTL) reduces the number of transistors by eliminating the redundant transistors. Note that in PTL, transistors are used as switches to pass the

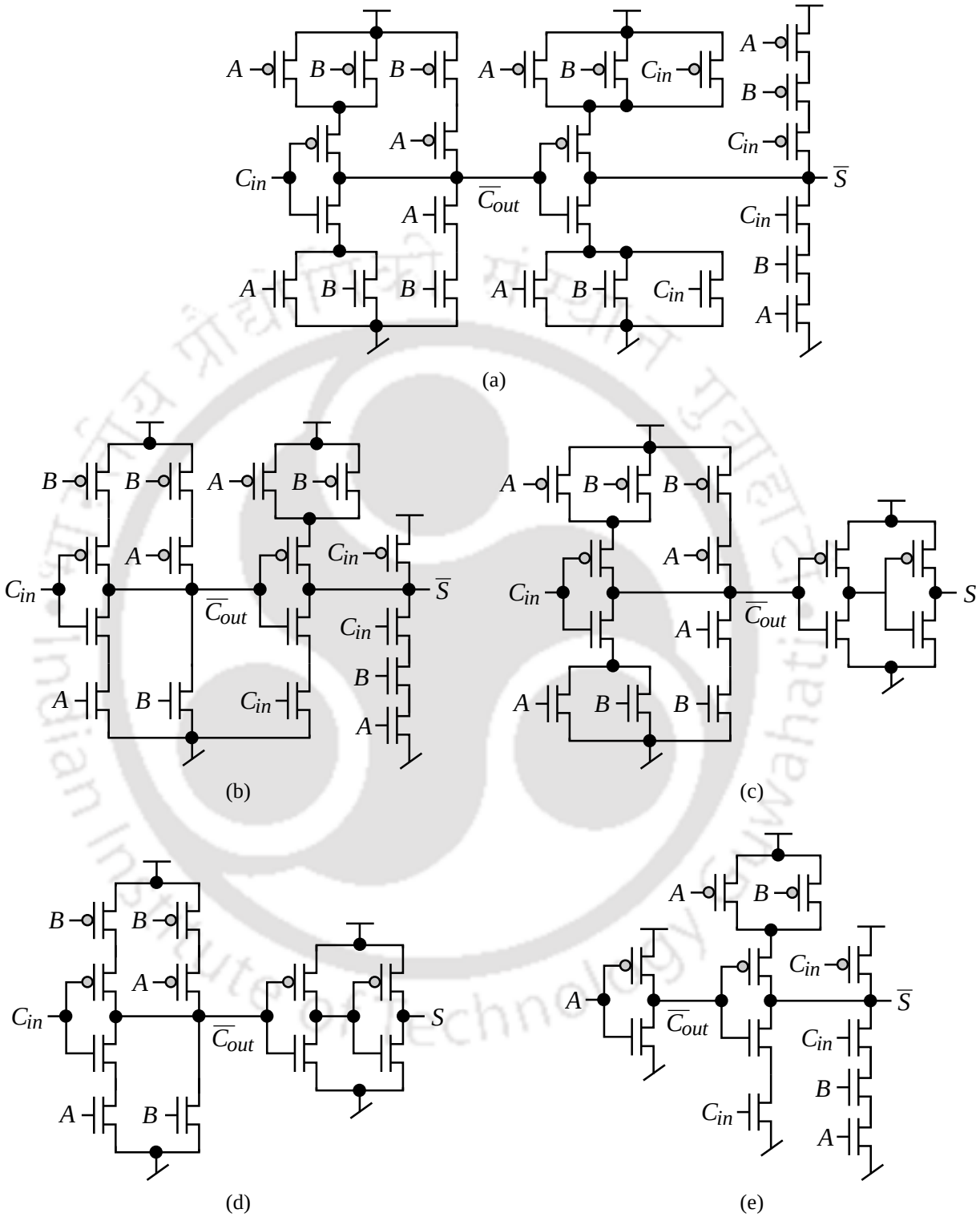


Figure 2.3: Transistor level implementations of: (a) Conventional MA; and AMAs proposed in [3]: (b) AMA#1; (c) AMA#2; (d) AMA#3; and (e) AMA#4.

logic levels between the nodes, instead of as switches connected directly to the supply voltages. This reduces the number of active transistors. All together, for the same functionality, PTL uses fewer tran-

Table 2.4: Truth table and design metrics of transistor level AFAs proposed in [3].

Inputs			Outputs									
			MA		AMA#1		AMA#2		AMA#3		AMA#4	
A	B	C_{in}	C_{out}	S	C_{out}	S	C_{out}	S	C_{out}	S	C_{out}	S
0	0	0	0	0	0 ✓	0 ✓	0 ✓	1 ×	0 ✓	1 ×	0 ✓	0 ✓
0	0	1	0	1	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓
0	1	0	0	1	1 ×	0 ×	0 ✓	1 ✓	1 ×	0 ×	0 ✓	0 ×
0	1	1	1	0	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	0 ×	1 ×
1	0	0	0	1	0 ✓	0 ×	0 ✓	1 ✓	0 ✓	1 ✓	1 ×	0 ×
1	0	1	1	0	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓
1	1	0	1	0	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓
1	1	1	1	1	1 ✓	1 ✓	1 ✓	0 ×	1 ✓	0 ×	1 ✓	1 ✓
ER			0	0	0.125	0.25	0	0.25	0.125	0.375	0.25	0.375
#Transistors			28		20		16		13		15	

sistors, runs faster and consumes less power as compared to the conventional CMOS logic. However, it has the disadvantage that the difference of voltage between high and low logic levels decreases at each stage. Each series transistor is less saturated at its output than at its input. Therefore, if several transistors are connected in series in a logic path, then a conventional CMOS logic is required to restore the logic level. A PTL based FA implemented with 10 transistors is shown in Fig. 2.4(a), where A, B and C_{in} are the inputs and D is the intermediate signal [96].

Yang et al. [4] propose three *Approximate XOR/XNOR-based Adders* (AXAs). These AXAs are designed by removing the transistors from PTL based FAs proposed in [11, 96]. AXA#1 is based on the FA proposed in [96], while AXA#2 and AXA#3 are based on the FA proposed in [11]. Note that both these FAs are designed using 4-transistor XOR/XNOR gates and consists of total 10 transistors. The truth table, ER and logic complexity (in terms of transistor count) of AXAs are shown in Table 2.5. In AXA#1 (see Fig. 2.4(b)), XOR operation is implemented using an inverter and two pass transistors connected to A and B, respectively. When B is 1, $D = \overline{A}$; otherwise, $D = A$; $i.e.$, $D = A \oplus B$. As shown in Table 2.5, AXA#1 requires 8 transistors and has 4 errors in each C_{out} and S. AXA#2 (see Fig. 2.4(c)) is implemented with 6 transistors. It consists of a 4-transistor XNOR gate and a pass transistor. As shown in Table 2.5, AXA#2 has 4 errors in S, while C_{out} is correct for all input combinations. AXA#3 is an extension of AXA#2. As shown in Fig. 2.4(d), it uses one addition pass transistor for the better accuracy of S. In total, AXA#3 requires 8 transistors and it has 2 errors

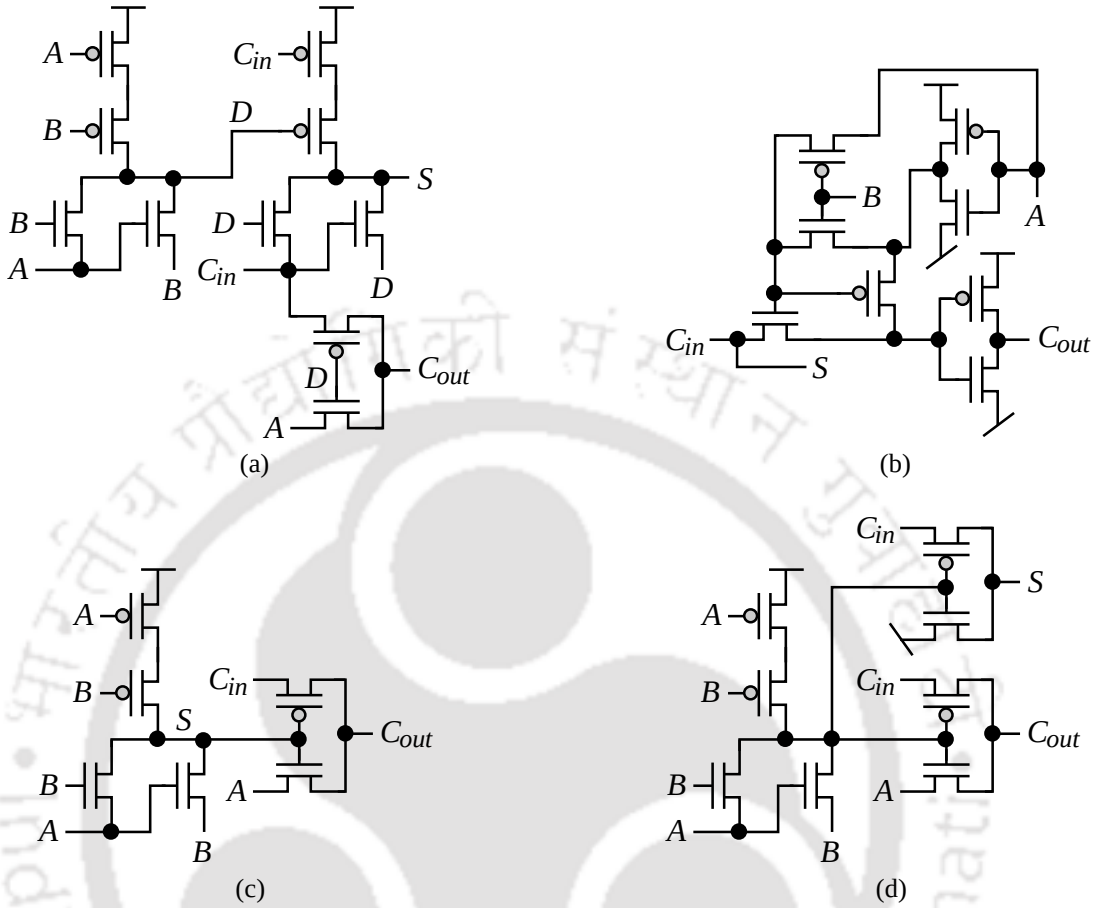


Figure 2.4: Transistor level implementations of: (a) PTL based FA proposed in [11]; and PTL based AXAs proposed in [4]: (b) AXA#1; (c) AXA#2; and (d) AXA#3.

in S, while C_{out} is correct for all input combinations (see Table 2.5). Here, it should be noted that in SFA, LOA and AMA#4, C_{out} is independent of C_{in}, whereas in AMA#1, AMA#2, AMA#3, AXA#1, AXA#2 and AXA#3, C_{out} is a function of C_{in}. Consequently, only SFA, LOA and AMA#4 can be used to truncate the length of carry propagation.

2.3.3 Error-Tolerant Adder 1 [5]

As compared to the AFAs, the design approach of *Error-Tolerant Adder 1* (ETA-1) is different. However, since accuracy of ETA-1 is similar to AFA-based approximate adders, we keep it in the same category. Similar to the AFA-based approximate adders, in ETA-I, an N-bit adder is segmented into two sub-adders: (i) Accurate sub-adder that includes the higher order k bits; and (ii) Approximate sub-adder that includes the remaining lower order $(N - k)$ bits. However, ETA-1 uses a modified XOR gate and control signal (CTL) to perform the lower order $(N - k)$ bit operations (see Fig.

Table 2.5: Truth table and design metrics of transistor level AFAs proposed in [4].

Inputs			Outputs							
			PTL based FA		AXA#1		AXA#2		AXA#3	
A	B	C_{in}	C_{out}	S	C_{out}	S	C_{out}	S	C_{out}	S
0	0	0	0	0	0 ✓	0 ✓	0 ✓	1 ×	0 ✓	0 ✓
0	0	1	0	1	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓
0	1	0	0	1	1 ×	0 ×	0 ✓	0 ×	0 ✓	0 ×
0	1	1	1	0	0 ×	1 ×	1 ✓	0 ✓	1 ✓	0 ✓
1	0	0	0	1	1 ×	0 ×	0 ✓	0 ×	0 ✓	0 ×
1	0	1	1	0	0 ×	1 ×	1 ✓	0 ✓	1 ✓	0 ✓
1	1	0	1	0	1 ✓	0 ✓	1 ✓	1 ×	1 ✓	0 ✓
1	1	1	1	1	1 ✓	1 ✓	1 ✓	1 ✓	1 ✓	1 ✓
ER			0	0	0.5	0.5	0	0.5	0	0.25
#Transistors			10		8		6		8	

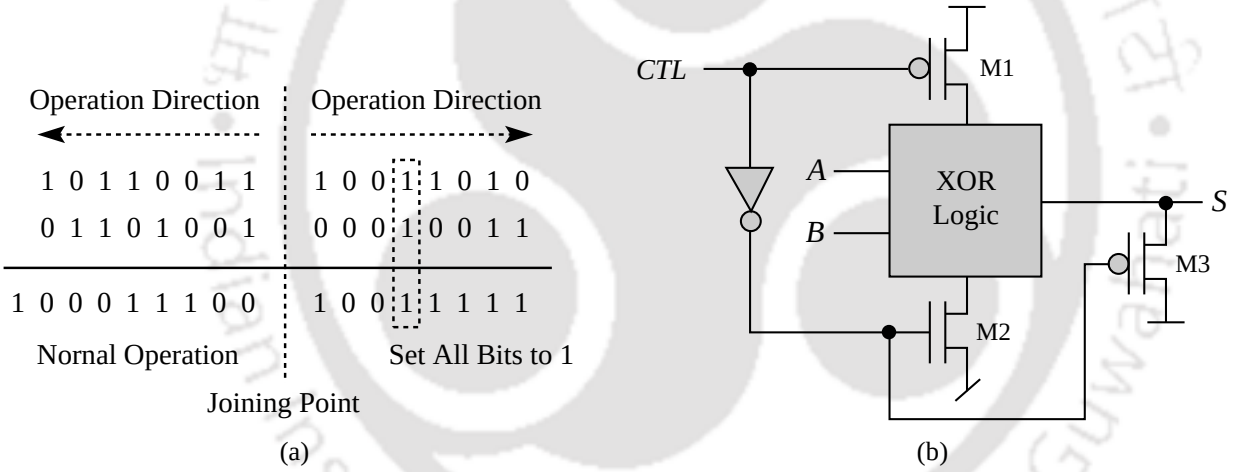


Figure 2.5: ETA-1: (a) Illustration via an example; and (b) Schematic diagram of modified XOR gate.

2.5(b)). We explain the ETA-1 approach via an example, as shown in Fig. 2.5(a). For the ease of illustration, we consider two 16-bit numbers. As shown in Fig. 2.5(a), addition process starts from the middle (joining point of the sub-adders) toward two opposite directions simultaneously. For accurate sub-adder, conventional carry generation and propagation approach is used. On the other hand, for approximate sub-adder, a special mechanism is used in which no carry is generated and propagated at any bit position. This special mechanism is described as follows.

(i) Check every input bit (A_i and B_i) from left to right.

(ii) If both input bits are 0 or different, then normal 1-bit addition is performed and the operation

proceeds to the next bit position.

(iii) If both input bits are 1, then the checking process is stopped and from this bit onwards, all sum bits to the right are set to 1.

As shown in Fig. 2.5(b), the modified XOR gate consists of XOR gate, NOT gate and three transistors (M1, M2 and M3). The control signal which is generated by the control block sets the operational mode of the modified XOR gate. When $CTL = 0$, M1 and M2 are turned on, while M3 is turned off, leaving the circuit to operate as normal XOR gate. When $CTL = 1$, M1 and M2 are turned off, while M3 is turned on, connecting the S to V_{DD}, and thus, setting it to 1. The function of the control block is to: (i) Detect the first bit position when both input bits are 1; and (ii) To set the control signal on this position as well as those on its right to 1.

2.4 State-of-the-Art ESAs

In recent years, ESAs has attracted a lot of attention of researchers/designers due to the key features of design flexibility and programmability. Different from the AFA-based approximate adders, ESA-based approximate adders do not eliminate the entire or a part of the carry-chain. Instead, they split the entire carry-chain into a number of short paths and complete the carry propagations in these short paths concurrently. As shown in Fig. 1.6, in this approach, an N-bit adder is segmented into several smaller disjoint or overlapping equally sized accurate sub-adders. An N-bit ESA has two primary design parameters: (i) Segment size (k), which represents the maximum length of carry propagation; and (ii) Overlapping bits (l), which represents the minimum number of bits used in carry prediction, where $1 \leq k < N$ and $0 \leq l < k$. Based on the relationship between k and l, several ESAs have been proposed in the literature [12, 14, 61–72]. In case of ESAs proposed in [14, 61–67], k need to be fixed apriori. Further, for a given k, l is also fixed in these ESAs ($l = 0$ in [14], $l = k/2$ in [62, 64–67] and $l = k - 1$ in [61, 63]). Consequently, ESAs proposed in [14, 61–67] can be designed for a given configuration, and thus, their design flexibility is limited. On the other hand, ESAs proposed in [12, 69] provide a very high degree of design flexibility by supporting multiple configurations. However, they provide this feature at the cost of hardware complexity.

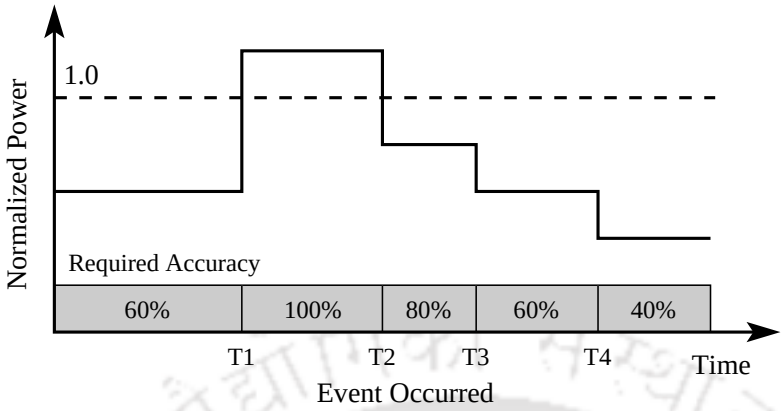


Figure 2.6: Power benefits from accuracy configurable adders.

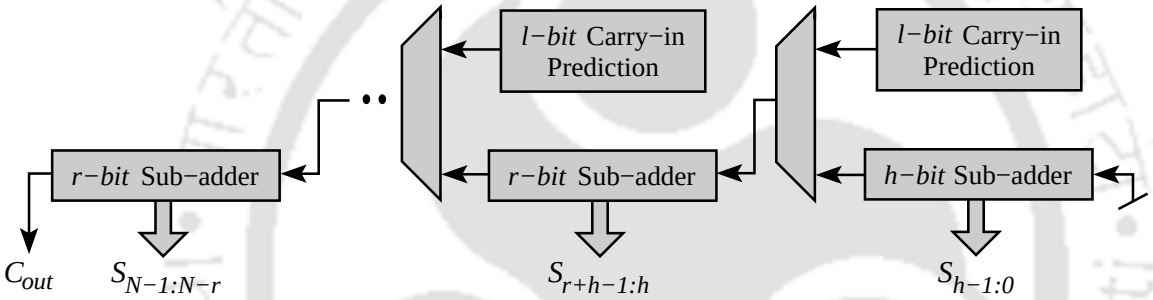


Figure 2.7: Generalized architecture of an N-bit ACA designed by dynamic tuning of k and l [12].

Accuracy Configurable Adders: Depending on the requirements of the applications, ESAs are designed for a targeted accuracy. Since different applications have different level of error-resilience, an approximate adder designed for an application can not be used for another application. Therefore, one of the major drawbacks of approximate adders is re-design efforts. Further, sure enough, not all the digital systems can accept the concept of approximate adders. For example, general purpose digital systems which run a variety of applications (including the error-resilient applications) refuse the use of approximate adders. In contexts where the accuracy requirements changes during runtime, the accuracy should be configurable to maximize the benefit of approximate adders. In the literature, such adders are known as ACAs. Fig. 2.6 illustrates how power benefits can be achieved with an ACA. We know that accuracy of an N-bit ESA depends the primary design parameters k and l. Therefore, as shown in Fig. 2.7, the accuracy of ESAs can be configured by tuning k and/or l dynamically using multiplexers. [12, 64, 70]. Multiplexers select carry-in either from the previous sub-adder or from the carry-in prediction. Based on the requirements of the applications, we can combine sub-adders

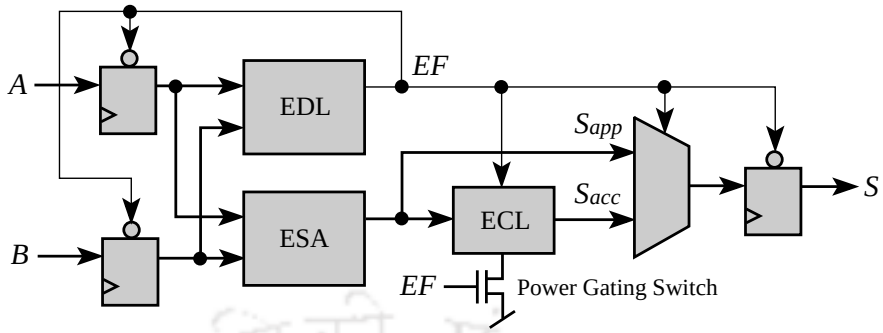


Figure 2.8: Generalized architecture of ACAs and VLSAs using EDC logic [13].

together as groups by setting the control signals of multiplexers.

Besides the tuning of k and l, another approach to configure the accuracy of ESAs is EDC logic [14, 61, 65, 69, 70, 85, 86]. As shown in Fig. 2.8, *Error Detection Logic* (EDL) detects errors and *Error Correction Logic* (ECL) corrects them based on the requirements of the applications. Note that EDC logic based ACAs are augmented with an *Error Flag* (EF) to reduce the power consumption. We know that *power gating* is one of the most effective approaches used to reduce the power consumption of digital circuits [97]. In this approach, power consumption is reduced by shutting off the current to the circuit blocks that are not in use. In ACAs also, for saving the power consumption, EF signal shuts off the ECL block when ESA provides correct result [65].

Variable Latency Speculative Adders: ESAs integrated with EDC logic can also work as VLSAs. The design approach of ACAs and VLSAs is similar, but ACAs configure accuracy during runtime based on the requirements of the applications, whereas VLSAs always provide accurate results. In ACAs, EDC logic starts correction errors from the least significant sub-adder and moves toward the most significant sub-adder till the required accuracy is not achieved. On the other hand, in VLSAs, EDC logic corrects errors in all sub-adders. VLSAs take one clock cycle when the ESA provides correct result and take two clock cycles when the ESA provides incorrect result. In this way, VLSAs provide results with variable latency. In recent years, several EDC logic have been proposed in the literature for ACAs and VLSAs [14, 61, 65, 69, 70, 73, 85, 86].

As the underlying structure of ESAs, ACAs and VLSAs is similar, it is not convenience to discuss them separately (like we do in the previous Section for AFAs). Therefore, here, we discuss ESAs, ACAs and VLSAs altogether in a proper sequence as follows.

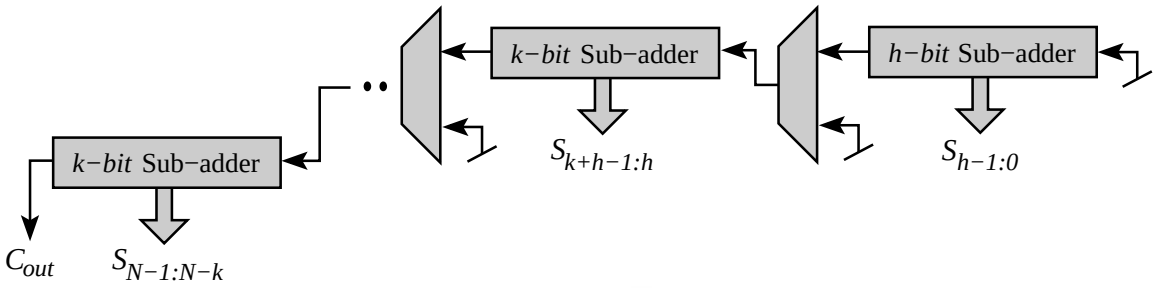


Figure 2.9: Generalized architecture of an N-bit DSEC [14].

In the literature, most of the ESAs have been proposed for $l = 0$ [14], $l = k/2$ [62, 64–67] and $l = k - 1$ [61, 63]. Later in Chapter 4, we discuss some important observations regarding ESAs. Here, it should be noted that for a fixed k, an N-bit ESA with $l = 0$ provides the lowest ED and with $l = k - 1$, it provides the lowest ER. Therefore, ESAs with $l = 0$ are considered as ED-optimal and ESAs with $l = k - 1$ are considered as ER-optimal. On the other hand, ESAs with $l = k/2$ provide trade-off between ED and ER to smooth the overall behavior.

Mohapatra et al. [14] propose an ESA with $l = 0$, called as *Dynamic Segmentation and Error Compensation* (DSEC), to exploit the concept of voltage over-scaling. As shown in Fig. 2.9, depending on the degree of voltage over-scaling, value of k can be configured using the control signals of multiplexers. Under the nominal V_{DD}, multiplexers select the original carry-in and feed it to the subsequent sub-adder. When operating under the voltage over-scaling, sub-adders are isolated from each other due to a carry-in of 0 being feed by the multiplexers.

Kahng et al. [65] propose an ESA with $l = k/2$. The authors also present an EDC logic where error correction happens in various stages. The stages can be pipelined for increasing the throughput of the design. Shafique et al. [69] propose a more generalized version of [65], called as *Generic Accuracy Configurable Adder* (GeAr), which allows the flexible values of k and l. By configuring these parameters, we can control the accuracy of GeAr, however, at the design time only. The authors also provide an EDC logic which is similar to the one proposed in [65]. Note that in [65, 69], the error correction starts from the least significant sub-adder and moves toward the most significant sub-adder. Hence, the accuracy of ACAs proposed in [65, 69] improves slowly in the progression of configurations. In order to overcome this drawback, Benara et al. [73] propose a correction technique

in which the error correction starts from the most significant sub-adder. Consequently, the accuracy of ACA proposed in [73] degrades in a graceful manner.

In ACAs proposed in [65, 69, 73], the error correction is pipelined. Therefore, when highly accurate results are required, they take multiple clock cycles and cause data stalls. In order to overcome these drawbacks, dynamic tuning of k and l based ACAs have been proposed in the literature [12, 64, 70]. As shown in Fig. 2.7, in this approach, each sub-adder contains an approximate carry-in prediction circuit. By selecting between the carry-out from sub-adder or carry-in prediction, the accuracy can be configured to different levels. Such an approach does not need EDC logic and incur any data stall. Further, the configuration of higher order sub-adders is independent of the lower order sub-adders. This leads to graceful degradation of accuracy.

Lu [63] proposes an ESA with $l = k - 1$. One concern about *Lu's Adder* (LA) is the area overhead due to the multitude of sub-adders. As per Table 1.1, implementation of an N-bit LA requires $(N - k + 1)$ k-bit sub-adders. For example, to implement a 32-bit LA, 25 8-bit sub-adders are required. This also imposes a large fanout on the primary inputs which results in delay penalty. In order to reduce the area overhead and fanout on the primary inputs, Verma et al. [61] introduce a reliable version of [63] by sharing the computation blocks among the sub-adders. In addition to the area and fanout reduction, the authors also propose an EDC logic.

2.5 Research Directions

The key directions in which recent research is going on for approximate adders are as follows.

(i) Over the past decade, several approximate adders have been proposed in the literature based on AFAs [1–4, 56–60] and ESAs [12, 14, 61–72]. Further, there are other mechanisms also which can be used for designing the approximate adders, such as ETA-1 [5], truncation [37–39], voltage over-scaling [40–42] and over-clocking [43–45]. Accordingly, one of the research directions is to introduce new approximate adders or modify the existing approximate adders so that their accuracy degrade in a graceful manner, and consequently, they provide optimal delay-accuracy, power-accuracy and area-accuracy trade-offs.

(ii) Traditionally, designers evaluate the design metrics of approximate adders using computer simulations. However, as discussed in Chapter 1, computer simulations have their own limitations, including programming efforts and time-consuming simulations. In recent years, several analytical models have been proposed in the literature to estimate the design metrics of approximate adders [63, 65, 69, 76–79, 81, 83, 98, 99]. Therefore, the another research direction is to introduce new generalized analytical models or modify the existing analytical models so that they estimate more accurately without imposing any additional complexity.

(iii) ESA is a specific category of segment based approximate adders in which each sub-adder (except the least significant sub-adder) has same value of k and l. However, besides ESAs, other types of segment based approximate adder are also possible. For example, it is possible that different sub-adders can have different values of k and l [62, 79, 83]. Therefore, one of the research directions is to explore segment based approximate adders which are more efficient (in terms of accuracy, delay, power and area) than the ESAs.

(iv) Over the past decade, most of the research work on approximate adders have demonstrated their effectiveness in multimedia applications. However, it has been observed in [8, 61, 100, 101] that a very high degree of error-resilience ($\approx 83\%$) is prevalent in a broad spectrum of applications, including machine learning, web search, probabilistic and statistical analytics, scientific computing, image, audio and video processing, wireless communication, cryptography, etc. Therefore, the next research direction is to explore the applicability/feasibility of approximate adders for state-of-the-art applications to maximize their benefits.

(v) In addition to the delay, power and area benefits, approximate adders can also be used to address various technology/design related issues, $e.g.$, yield enhancement [1]. Approximate adders need to be exploited to resolve such issues. Moreover, we know that in binary arithmetic, all basic operations, such as addition, subtraction, multiplication and division use adders as the key components. Therefore, approximate adders can further be used for designing other approximate arithmetic operations [87–89]. Accordingly, the another research direction is to utilize approximate adders for designing other approximate arithmetic operations efficiently.

2.6 Summary

In this Chapter, we first discussed the design philosophy and design approach of approximate adders. Further, we presented a survey of state-of-the-art AFA-based and ESA-based approximate adders. In case of AFAs, we considered two gate level AFAs (SFA and LOA) and two transistor level AFAs (AMAs and AXAs). We also discussed ETA-1. Although the design approach of ETA-1 is different, we included it in the AFAs category because its accuracy is similar to AFA-based approximate adders. In case of ESAs, we also described ACAs and VLSAs designed using EDC logic and dynamic tunning of k and l. As the underlying structure of ESAs, ACAs and VLSAs is similar, we discussed them altogether in a proper sequence. Finally, we presented the directions in which recent research is going on for approximate adders.

In this Chapter, we discussed state-of-the-art AFAs and ESA. In subsequent chapters, we present the proposed work (design, analytical modeling, optimization and applicability) on AFAs and ESAs.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

3

AFA-based Approximate Adders

Contents

Over the past decade, several AFA-based approximate adders have been proposed in the literature [1–4,56–60]. As discussed in Chapter 2, AFA-based approximate adders has attracted the attention of researchers/designers due to the two main features: (i) They provide a significant reduction in logic complexity; and (ii) For a given accuracy, they consume less power as compared to the truncated adders. Note that in AFA-based approximate adders, the lower order $(N - k)$-bit operations are performed using AFAs, whereas in truncated adders, the lower order $(N-k)$-bit operations are totally ignored. AFAs are generally designed by modifying the conventional FAs at gate level [1, 2, 56–58] and transistor level [3, 4] of abstraction. While designing an AFA, the key objective is to reduce the logic complexity subjected to minimal ED and/or ER.

In this Chapter, we propose four AFAs with design objective that C_{out} should be independent of C_{in} with minimal ER. We exploit one of the proposed AFAs (the optimal one) for designing an N-bit approximate adder which we call as "ApproxADD". For improving the ED and ER of ApproxADD, we avail the concept of carry-lifetime and EDC logic, respectively. In this way, we introduce two more (improved) versions of ApproxADD – ApproxADD$v1$ and ApproxADD$v2$. Apart from the designing, we provide a detailed analysis and analytical models to estimate accuracy, delay, power and area of ApproxADDs. For evaluating the efficiency of the proposed approach, we compare ApproxADD$v1$ and ApproxADD$v2$ with existing AFA-based approximate adders. Further, to inspect the effectiveness of the proposed approach in real-life applications, we demonstrate image compression and decompression by replacing the conventional addition operations in DCT and IDCT modules with ApproxADD$v2$. In the end, we summarize the Chapter.

The rest of this Chapter is organized as follows. Section 3.1 describes the method of logical effort which we use to estimate delay, power and area of ApproxADDs. Section 3.2 discusses the proposed AFAs and their delay, power and area analysis. Section 3.3 presents ApproxADD – an N-bit approximate adder designed using the proposed AFAs. Section 3.4 explains our strategies (probability of carry-lifetime and EDC logic) for improving the ED and ER of ApproxADD. Section 3.5 provides simulation results and discussion of the proposed approach in respect of acceptance threshold, parallel-prefix adder architectures, comparison with existing work and applicability in real-life applications. Finally, Section 3.6 concludes the Chapter.

3.1 Estimation Method

In digital circuit design, there are different levels of design abstraction, such as transistor, gate, etc. At each level of design abstraction, several delay, power and area estimation methods have been proposed in the literature [6]. These methods are generally classified based on the trade-off between their estimation accuracy and computational complexity. For example, estimation of delay using the method of *logical effort* is more accurate as compared to estimate the delay in terms of *gate level depth*. However, estimating the delay in terms of logical effort terminology is more complex as compared to estimate the delay in terms of gate level depth. Therefore, for simple circuits ($e.g.$, FA and AFAs), the method of logical effort is used, whereas for complex circuits ($e.g.$, CIA and KSA), researchers/designers estimate delay in terms of gate level depth. In this Chapter, we use the method of logical effort for estimating delay, power and area of ApproxADDs, while in Chapter 4, we use the lateral approach for estimating delay, power and area of ESAs.

3.1.1 Delay Estimation

According to the method of logical effort [15], total delay (d) incurred by a static CMOS logic gate (measured in terms of basic delay unit τ[1]) can be given by:

$$d = f + p_d \tag{3.1}$$

where, f is the effort delay and p_d is the parasitic delay. The effort delay further consists of two components: (i) Logical effort (g), which captures the effect of logic complexity; and (ii) Electrical effort (h), which captures the effect of external load being driven by the logic gate. The effort delay is given by the product of these two components as:

$$f = g \times h \tag{3.2}$$

where, logical effort is given by the ratio of input capacitance of the logic gate ($C_{in_{gate}}$) to the input

[1]Delay of CMOS logic gates depends on the process technology. In order to isolate the effects of a particular process technology, unless otherwise indicated, we measure all delays in this Chapter in units of τ.

capacitance of a unit standard CMOS inverter ($C_{in_{inv}}$) as:

$$g = \frac{C_{in_{gate}}}{C_{in_{inv}}} \tag{3.3}$$

and, electrical effort is given by the ratio of load capacitance (C_L) being driven by the logic gate to the input capacitance of the logic gate ($C_{in_{gate}}$) as:

$$h = \frac{C_L}{C_{in_{gate}}} \tag{3.4}$$

Substituting the value of effort delay from Eqn. (3.2) into Eqn. (3.1), we obtain the basic equation that models delay[2] of a static CMOS logic gate as:

$$d = (g \times h) + p_d \tag{3.5}$$

We can now compute the total path delay (D[3]) of an n stage CMOS digital circuit by adding the delay (d) of individual logic gates in that path as:

$$D = \sum_{i=1}^{n} d_i = \sum_{i=1}^{n} \Big((g_i \times h_i) + p_{d_i} \Big) \tag{3.6}$$

3.1.2 Power Estimation

The power consumption of a CMOS logic gate is mainly consists of two components: (i) Static power consumption; and (ii) Dynamic power consumption. The dynamic power consumption (p_{dyn}) which dominates the total power consumption is given by:

$$p_{dyn} = \alpha \, C_{out} \, V_{DD}^2 \, f_{clk} \tag{3.7}$$

where, α, C_{out}, V_{DD} and f_{clk} are the switching activity factor, output capacitance, supply voltage and clock frequency, respectively. Note that the output capacitance is further consists of two components: (i) Load capacitance (C_L); and (ii) Parasitic capacitance (C_p). Substituting the value of output

[2]In [15], to study the effect of logic complexity, the authors introduce logical effort (g) and to study the effect of external load being driven by logic gates, the authors introduce electrical effort (h). Therefore, simplification of Eqn. (3.2) as $(C_{in_{gate}}/C_{in_{inv}}) \times (C_L/C_{in_{gate}}) = C_L/C_{in_{inv}}$ is not advisable.

[3]In this Chapter, to represent delay, power and area of logic gates, we use small letters ($d,\ p$ and a), whereas to represent delay, power and area of digital circuits, we use capital letters ($D,\ P$ and A).

capacitance ($C_{out} = C_L + C_p$) into Eqn. (3.7) yields:

$$p_{dyn} = \alpha \left(C_L + C_p\right) V_{DD}^2 \, f_{clk} \tag{3.8}$$

It can be seen that estimating the dynamic power consumption using Eqn. (3.8) is very cumbersome and time consuming. In order to make the analysis simple, Kabbani [102] characterizes the dynamic power consumption in terms of logical effort terminology by normalizing it $w.r.t.$ the dynamic power consumption of a unit standard CMOS inverter as:

$$p_{nm\_inv} = \frac{\alpha_{gate}}{\alpha_{inv}} \left(\frac{h \times C_{in_{gate}}}{C_{in_{inv}}} + \frac{C_{p_{gate}}}{C_{p_{inv}}} \right) = \alpha_{nm} \left(h \times g + p_p\right) \tag{3.9}$$

where, p_{nm_inv} and α_{nm} are the normalized dynamic power consumption and switching activity factor of the logic gate $w.r.t.$ a unit standard CMOS inverter, respectively, and p_p is the parasitic power consumption of the logic gate. Since dynamic power consumption is a strong function of the logic gate area (a), Eqn. (3.9) is valid only for *template gates*. A template gate refers to the logic gate whose output current is equal to the output current of a unit standard CMOS inverter. For any other logic gate, the normalized dynamic power consumption can be given by:

$$p_{nm\_inv} = a_{nm} \, \alpha_{nm} \left(h \times g + p_p\right) \tag{3.10}$$

where, a_{nm} is the normalized area of the logic gate $w.r.t.$ corresponding template gate. We can now compute the total normalized dynamic power consumption (P_{nm_inv}) of a CMOS digital circuit that contains m logic gates by adding the normalized dynamic power consumption (p_{nm_inv}) of individual logic gates in that circuit as:

$$P_{nm\_inv} = \sum_{i=1}^{m} p_{nm\_inv_i} = \sum_{i=1}^{m} \left(a_{nm_i} \, \alpha_{nm_i} \left(h_i \times g_i + p_{p_i}\right) \right) \tag{3.11}$$

While determining the normalized dynamic power consumption using Eqn. (3.11), it is necessary to compute the switching activity factor at each node of the circuit. Switching activity factor of a node is the probability that it switches from 0 to 1, $i.e.$, probability of the node to be 0 on one cycle and to be 1 on the next cycle. Assuming that the probability is uncorrelated from cycle to cycle [6], switching activity factor of a node can be given by:

Table 3.1: Probability of output node of logic gates to be 1 as a function of their input nodes probability.

Logic Gates	Input Nodes	P_1
NOT	A	$1 - P_A$
2-input NAND	A and B	$1 - (P_A \times P_B)$
2-input NOR	A and B	$(1 - P_A)(1 - P_B)$
2-input XOR	A and B	$P_A(1 - P_B) + P_B(1 - P_A)$
3-input NAND	A, B and C	$1 - (P_A \times P_B \times P_C)$

$$\alpha = P_0 \times P_1 = P_1(1 - P_1) \tag{3.12}$$

where, P_0 is the probability of the node to be 0 and P_1 is the probability of the node to be 1. Table 3.1 lists P_1 of various logic gates (used in this Chapter) as a function of their input nodes probability, where P_A, P_B and P_C represent the probabilities of input nodes A, B and C to be 1, respectively, and P_1 represents the probability of output node to be 1 [6]. Assuming that inputs are uniformly distributed ($i.e.$, P_1 of input nodes is 0.5), we first compute P_1 of the nodes using Table 3.1. We then compute switching activity factor of the nodes using Eqn. (3.12).

3.1.3 Area Estimation

Researchers/designers generally measure area in terms of *gate count* or *transistor count*. However, this type of analysis does not include the effect of transistor sizes, and therefore, their estimation accuracy is poor. For example, as shown in Fig. 3.1, both NAND gate and NOR gate have same transistor count, but due to different transistor sizes, their actual silicon areas are different. In order to improve the estimation accuracy without imposing any additional complexity, we measure area in terms of transistor sizes. According to our analysis, NAND gate has area, $a = 8$ units and NOR gate has area, $a = 10$ units (see Fig. 3.1). Similarly, we compute area of other logic gates used in this Chapter. We then compute the total area (A) of a CMOS digital circuit that contains m logic gates by adding the area (a) of individual logic gates in that circuit as:

$$A = \sum_{i=1}^{m} a_i \tag{3.13}$$

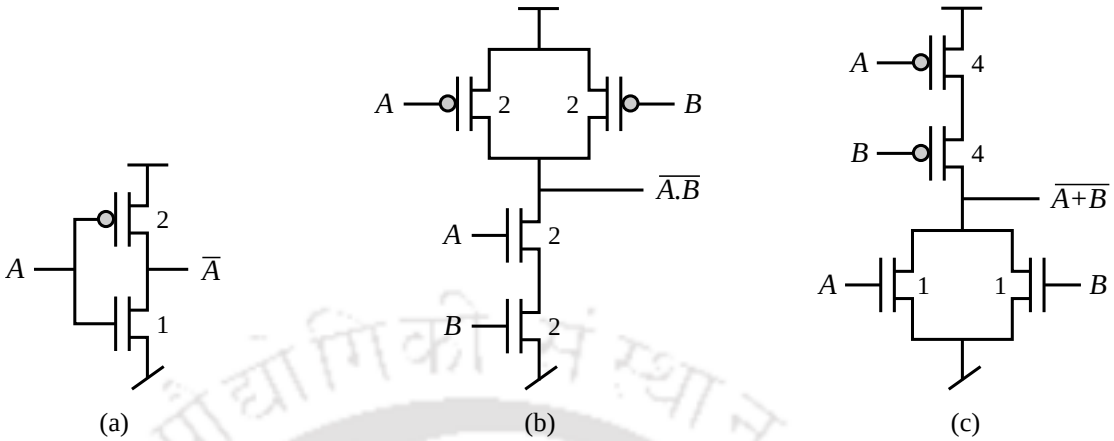


Figure 3.1: Transistor level implementations: (a) NOT gate, which has $a = 3$ units and $C_{in_{gate}} = 3$ units; (b) 2-input NAND gate, which has $a = 8$ units and $C_{in_{gate}} = 4$ units; and (c) 2-input NOR gate, which has $a = 10$ units and $C_{in_{gate}} = 5$ units. Note that we size these logic gates according to the method of logical effort.

3.2 Proposed AFAs

AFA is one of the commonly used approaches to curtail the length of carry propagation. As discussed in Chapter 2, AFAs are generally designed by modifying the conventional FAs at gate level [1,2,56–60] and transistor level [3,4] of abstraction. In a FA, to secure C_{out} independent of C_{in}, C_{out} should be either 0, 1 or a Boolean function of primary inputs A and B only. In such a scenario, there can be total $2^{(2^2)} = 16$ approximate functions of C_{out}. All these 16 approximate functions are tabulated in Table 3.2 with corresponding correct and incorrect entries, where tick marks denote match and cross marks denote mismatch with accurate output. Assuming that inputs (A, B and C_{in}) are uniformly distributed, the probability of approximate C_{out} to be correct is also mentioned in Table 3.2. From Table 3.2, our important observations are as follows.

(i) In an AFA, if we consider C_{out} as 0, 1, $A \oplus B$, $A \odot B$, $\overline{A}.B$, $A.\overline{B}$, $\overline{A}+B$ or $A+\overline{B}$, then the C_{out} has correct output in four out of total eight input combinations, and therefore, the probability of approximate C_{out} to be correct is $4/8 = 0.5$.

(ii) In an AFA, if we consider C_{out} as $\overline{A}$, $\overline{B}$, $\overline{A.B}$ or $\overline{A+B}$, then the C_{out} has correct output in two out of total eight input combinations, and therefore, the probability of approximate C_{out} to be correct is $2/8 = 0.25$.

Table 3.2: Approximate functions of C_{out} in which C_{out} is independent of C_{in}. Assuming that inputs (A, B and C_{in}) are independent and uniformly distributed, the last row summarizes the probability of approximate C_{out} to be correct.

Inputs				Approximate functions of C_{out} in which C_{out} is independent of C_{in}													
A	B	C_{in}	C_{out}	0	1	A	B	$\overline{A}$	$\overline{B}$	$A.B$	$A+B$	$\overline{A.B}$	$\overline{A+B}$	$A \oplus B$	$A \odot B$	$\overline{A}.B$	$A.\overline{B}$	$\overline{A}+B$	$A+\overline{B}$
0	0	0	0	0 ✓	1 ×	0 ✓	0 ✓	1 ×	1 ×	0 ✓	0 ✓	1 ×	1 ×	0 ✓	1 ×	0 ✓	0 ✓	1 ×	1 ×
0	0	1	0	0 ✓	1 ×	0 ✓	0 ✓	1 ×	1 ×	0 ✓	0 ✓	1 ×	1 ×	0 ✓	1 ×	0 ✓	0 ✓	1 ×	1 ×
0	1	0	0	0 ✓	1 ×	0 ✓	1 ×	1 ×	0 ✓	0 ✓	1 ×	1 ×	0 ✓	1 ×	0 ✓	1 ×	0 ✓	1 ×	0 ✓
0	1	1	1	0 ×	1 ✓	0 ×	1 ✓	1 ✓	0 ×	0 ×	1 ✓	1 ✓	0 ×	1 ✓	0 ×	1 ✓	0 ×	1 ✓	0 ×
1	0	0	0	0 ✓	1 ×	1 ×	0 ✓	0 ✓	1 ×	0 ✓	1 ×	1 ×	0 ✓	1 ×	0 ✓	0 ✓	1 ×	0 ✓	1 ×
1	0	1	1	0 ×	1 ✓	1 ✓	0 ×	0 ×	1 ✓	0 ×	1 ✓	1 ✓	0 ×	1 ✓	0 ×	0 ×	1 ✓	0 ×	1 ✓
1	1	0	1	0 ×	1 ✓	1 ✓	1 ✓	0 ×	0 ×	1 ✓	1 ✓	0 ×	0 ×	0 ×	1 ✓	0 ×	0 ×	1 ✓	1 ✓
1	1	1	1	0 ×	1 ✓	1 ✓	1 ✓	0 ×	0 ×	1 ✓	1 ✓	0 ×	0 ×	0 ×	1 ✓	0 ×	0 ×	1 ✓	1 ✓
Probability of C_{out} to be correct				0.5	0.5	0.75	0.75	0.25	0.25	0.75	0.75	0.25	0.25	0.5	0.5	0.5	0.5	0.5	0.5

Table 3.3: Estimated D, P_{nm_inv}, A and $PDAP$ of conventional FA (Fig. 3.2) and the proposed AFAs (Fig. 3.3). Table also summarizes the percentage change (Δ) in D, P_{nm_inv} A and $PDAP$ of the proposed AFAs $w.r.t.$ conventional FA.

Adder Architectures	D		ΔD (%)		P_{nm_inv}	ΔP_{nm_inv} (%)	A	ΔA (%)	$PDAP$	$\Delta PDAP$ (%)
	S	C_{out}	S	C_{out}						
FA	13.33	17.99	0	0	26.54	0	84	0	40.106E+3	0
AFA#1	12	0	-9.98	-100	14	-47.25	60	-28.57	10.080E+3	-74.86
AFA#2	12	0	-9.98	-100	14	-47.25	60	-28.57	10.080E+3	-74.86
AFA#3	12	7	-9.98	-61.09	19.25	-27.47	71	-15.48	16.401E+3	-59.11
AFA#4	12	7	-9.98	-61.09	19.25	-27.47	73	-13.09	16.863E+3	-57.95

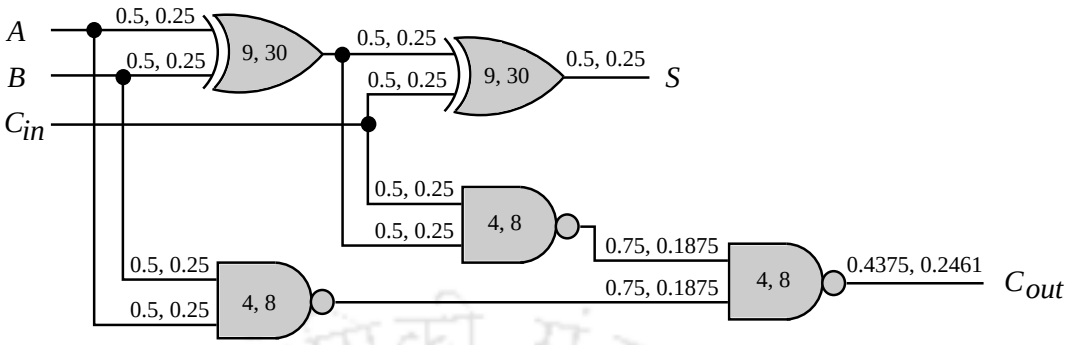


Figure 3.2: Gate level implementation of conventional FA. Here, each logic gate is shown with two numbers, where the first number indicates input capacitance ($C_{in_{gate}}$) of the logic gate and the second number indicates area (a) of the logic gate [15]. Further, input and output nodes of the logic gates are also shown with two numbers, where the first number indicates P_1, *i.e.*, probability of the node to be 1 and the second number indicates switching activity factor (α) of the node [6].

(iii) In an AFA, if we consider C_{out} as $A,\ B,\ A.B$ or $A+B$, then the C_{out} has correct output in six out of total eight input combinations, and therefore, the probability of approximate C_{out} to be correct is $6/8 = 0.75$.

It should be noted from above observations that in an AFA, if we consider C_{out} as A, B, $A.B$ or $A+B$, then the AFA provides lower ER. Consequently, an N-bit approximate adder designed using either of these four AFAs provides higher accuracy. We call these AFAs as AFA#1, AFA#2, AFA#3 and AFA#4, respectively. The gate level implementations of all these AFAs are shown in Fig. 3.3.

3.2.1 Delay, Power and Area Estimation

For estimating delay, power and area of conventional FA and the proposed AFAs, we use the method described in Section 3.1. We first compute the design parameters, such as α, g, h, p_d, p_p, a, d and p_{nm_inv} of all the logic gates used in conventional FA and the proposed AFAs. Our computed values are tabulated in Table 3.4. Using Table 3.4, we then compute delay (D), normalized dynamic power consumption (P_{nm_inv}) and area (A) of conventional FA and the proposed AFAs by adding the delay (d), normalized dynamic power consumption (p_{nm_inv}) and area (a) of individual logic gates in that circuit, respectively. For example, in conventional FA (Fig. 3.2), output S has delay, $D = 8.33 + 5 = 13.33$ and output C_{out} has delay, $D = 8.33 + 3.33 + 6.33 = 17.99$; whereas, in AFA#3 (Fig. 3.3(c)), output S has delay, $D = 7 + 5 = 12$ and output C_{out} has delay, $D = 3 +$

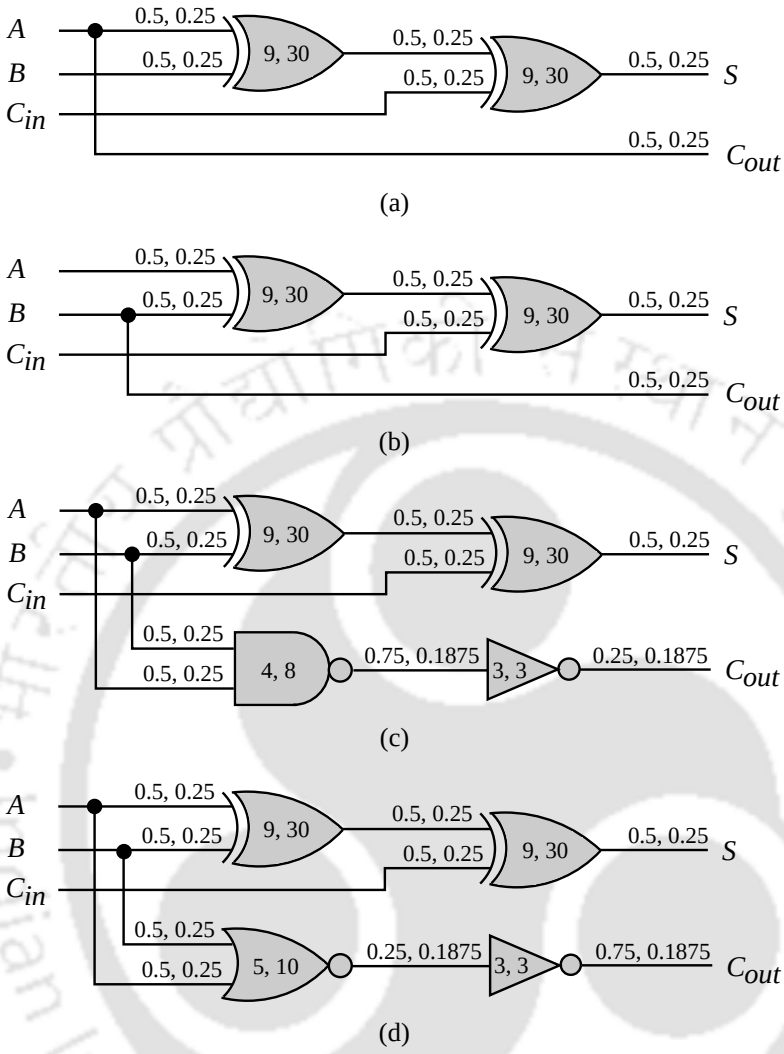


Figure 3.3: Gate level implementations of the proposed AFAs: (a) AFA#1; (b) AFA#2; (c) AFA#3; and (d) AFA#4. Note that the numbers indicated on the logic gates and on the input and output nodes of the logic gates have already been discussed in Fig. 3.2.

$4 = 7$. Accordingly, as compared to the conventional FA, AFA#3 improves the delay for output S by 9.98% and for output C_{out} by 61.09%. Further, conventional FA has normalized dynamic power consumption, $P_{nm\_inv} = 9.33 + 6 + 2.49 + 2.49 + 6.23 = 26.54$ and area, $A = 30 + 30 + 8 + 8 + 8 = 84$ units; whereas AFA#3 has normalized dynamic power consumption, $P_{nm\_inv} = 8 + 6 + 2.25 + 3 = 19.25$ and area, $A = 30 + 30 + 8 + 3 = 71$ units. Accordingly, as compared to the conventional FA, AFA#3 improves the dynamic power consumption by 27.47% and area by 15.48%. Similarly, we compute delay, normalized dynamic power consumption and area of AFA#1, AFA#2 and AFA#4. Our computed values are tabulated in Table 3.3. Note that Table 3.3 also summarizes *Power-Delay-*

Table 3.4: Design parameters of the logic gates used in FA (Fig. 3.2) and the proposed AFAs (Fig. 3.3).

Logic Gate	α	g	h	p_d	p_p	a	d	p_{nm_inv}
INV	0.1875	1	3^*	1	1	3	4	3
NAND	0.1875	4/3	1	2	2	8	3.33	2.49
NAND	0.2461	4/3	$13/4^*$	2	2	8	6.33	6.23
NAND	0.1875	4/3	3/4	2	2	8	3	2.25
NOR	0.1875	5/3	3/5	2	2	10	3	2.25
XOR	0.25	3	13/9	4	5	30	8.33	9.33
XOR	0.25	3	$1/3^{\#}$	4	5	30	5	6
XOR	0.25	3	1	4	5	30	7	8

Here, $*$ and $\#$ signify that while computing the electrical effort (h) of output logic gates, we assume that C_{out} drives C_{in} and S drives a unit standard CMOS inverter, respectively.

Area-Product (PDAP) of conventional FA and the proposed AFAs. In addition to this, the percentage change in delay (ΔD), normalized dynamic power consumption (ΔP_{nm_inv}), area (ΔA) and PDAP ($\Delta PDAP$) of the proposed AFAs $w.r.t.$ conventional FA is also mentioned in Table 3.3.

3.3 ApproxADD

We now design an N-bit approximate adder using the proposed AFAs. Table 3.2 shows that all the four proposed AFAs exhibit similar error probability. Therefore, an N-bit approximate adder designed using either of these four AFAs provides similar error characteristics. However, as shown in Table 3.3, these AFAs have different delay, power, area and PDAP. Among all the four proposed AFAs, AFA#1 and AFA#2 provide smaller delay, power, area and PDAP. Therefore, while designing an N-bit approximate adder that provides optimal *Power-Delay-Area-Accuracy* (PDAA) trade-off, one should choose either AFA#1 or AFA#2. An N-bit approximate adder (which we call as "ApproxADD") designed using AFA#1 is shown in Fig. 3.4.

3.3.1 Delay, Power and Area Estimation

In RCA, the worst-case delay is given by the delay to compute sum in last stage and the delay to compute carry propagation in the remaining $(N-1)$ stages. On the other hand, in ApproxADD, since there is no carry propagation, the worst-case delay is given by the delay to compute sum in AFA#1 only. Therefore, using Table 3.3 and Table 3.4, the normalized delay (D_{nm}) of an ApproxADD $w.r.t.$

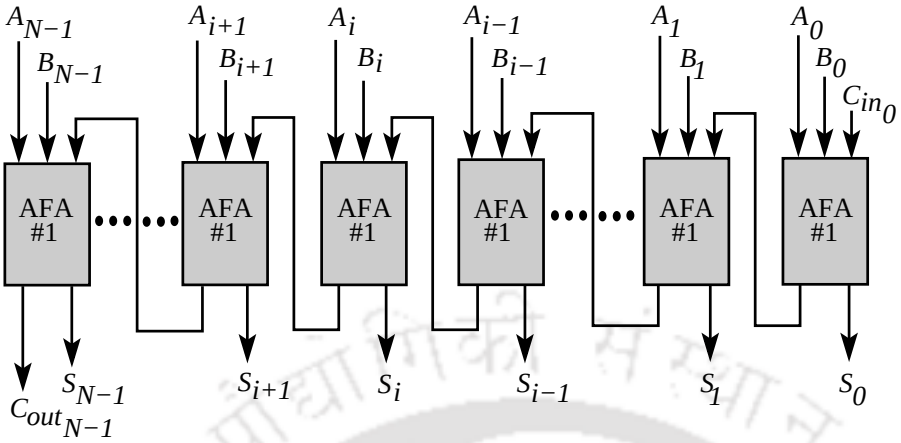


Figure 3.4: Generalized architecture of an N-bit ApproxADD designed by cascading N AFA#1 in series.

RCA can be given by:

$$\begin{aligned} D_{nm} &= \frac{D_{AFA\#1}}{D_{FA[S]} + (N-1) \times D_{FA[C_{in}:C_{out}]}} \\ &= \frac{12}{13.33 + (N-1) \times 9.66} = \frac{1.2422}{N + 0.3799} \end{aligned} \tag{3.14}$$

where, $D_{AFA\#1}$ is the delay to compute sum in AFA#1, $D_{FA[S]}$ is the delay to compute sum in conventional FA and $D_{FA[C_{in}:C_{out}]}$ is the delay to compute carry propagation ($C_{in} \rightarrow C_{out}$) in conventional FA. Further, using Table 3.3 and Table 3.4, the normalized dynamic power consumption (P_{nm}) and normalized area (A_{nm}) of an ApproxADD $w.r.t.$ RCA can be given by:

$$P_{nm} = \frac{N \times P_{nm\_inv_{AFA\#1}}}{N \times P_{nm\_inv_{FA}}} = 0.5275 \tag{3.15}$$

$$A_{nm} = \frac{N \times A_{AFA\#1}}{N \times A_{FA}} = 0.7143 \tag{3.16}$$

where, $P_{nm\_inv_{FA}}$ and $P_{nm\_inv_{AFA\#1}}$ are the normalized dynamic power consumptions of conventional FA and AFA#1 $w.r.t.$ a unit standard CMOS inverter, and A_{FA} and $A_{AFA\#1}$ are the areas of conventional FA and AFA#1, respectively.

3.3.2 ED and ER Estimation

As shown in Table 3.2, AFA#1 provides erroneous carry for two out of eight input combinations. These two input combinations are $\overline{A}.B.C_{in}$ and $A.\overline{B}.\overline{C}_{in}$. Further, in ApproxADD, C_{in_i} is A_{i-1}. Therefore, in ApproxADD, an erroneous carry generates at i^{th} bit position only when $\overline{A}_i.B_i.A_{i-1} = 1$

or $A_i.\overline{B}_i.\overline{A}_{i-1} = 1$. We know that in addition operation, erroneous carry generated at i^{th} bit position affects the output at $(i+1)^{th}$ bit position. Accordingly, ED of an ApproxADD can be given by the weighted sum of input combinations $\overline{A}_i.B_i.A_{i-1}$ and $A_i.\overline{B}_i.\overline{A}_{i-1}$ as:

$$ED \approx \sum_{i=0}^{N-1} 2^{i+1} \times \Big((\overline{A}_i.B_i.A_{i-1}) + (A_i.\overline{B}_i.\overline{A}_{i-1}) \Big) \tag{3.17}$$

where, $\cdot$ and $+$ represent the binary AND and OR operations, respectively. Note that in ApproxADD, either of the input combination ($\overline{A}_i.B_i.A_{i-1}$ or $A_i.\overline{B}_i.\overline{A}_{i-1}$) can not occur consecutively. However, these input combinations can occur in an alternative fashion. This is the worst-case in which ApproxADD provides maximum error. For example, let us assume that A is 01010101, B is 10101010 and C_{in_0} is 0. In this case, $\overline{A}_i.B_i.A_{i-1}$ OR $A_i.\overline{B}_i.\overline{A}_{i-1}$ is 11111111, which implies that if we add A and B using ApproxADD, then error will occur at all bit positions. Since accurate sum is 11111111, approximate sum (computed using ApproxADD) should be 00000000. However, ApproxADD computes it as 01010101. This difference is because in ApproxADD, two consecutive erroneous carries generated from i^{th} bit position do not propagate erroneous carry for $(i+2)^{th}$ bit position. The reason behind this is the fact that twice the complement of a function is the function itself. In other words, an erroneous carry generated at i^{th} bit position masks the effect of erroneous carry generated at $(i+1)^{th}$ bit position. Therefore, Eqn. (3.17) can be corrected by subtracting the weighted sum of input combinations which generate consecutive erroneous carries as:

$$\begin{aligned} ED \approx & \sum_{i=0}^{N-1} 2^{i+1} \times \Big((\overline{A}_i.B_i.A_{i-1}) + (A_i.\overline{B}_i.\overline{A}_{i-1}) \Big) \\ & - \sum_{i=0}^{N-1} 2^{i+1} \times \Big((\overline{A}_i.B_i.A_{i-1}) \,.\, (A_{i-1}.\overline{B}_{i-1}.\overline{A}_{i-2}) \Big) \\ & - \sum_{i=0}^{N-1} 2^{i+1} \times \Big((A_i.\overline{B}_i.\overline{A}_{i-1}) \,.\, (\overline{A}_{i-1}.B_{i-1}.A_{i-2}) \Big) \end{aligned} \tag{3.18}$$

Note that Eqn. (3.18) is still an approximation because overlapping cases are not taken into consideration. Further, in most of the error-resilient applications, RED is more meaningful as compared to the absolute ED, where RED is defined as the ED normalized to corresponding correct output

(R_c) [75]. For a set of n inputs, MRED of an ApproxADD can be given by:

$$MRED = \frac{1}{n}\sum_{i=1}^{n} RED_i = \frac{1}{n}\sum_{i=1}^{n} \frac{ED_i}{R_{c_i}} \tag{3.19}$$

Further, in order to model the ER of ApproxADD, we use empirical approach. Note that empirical approaches are generally based on the information received by the means of observation or experience of patterns and behavior through experimentation. As per our experimental observations, ER of ApproxADD forms a *Geometric Progression* (GP) with scalar factor $\approx 1/4$ and common ratio $\approx 3/4$. Accordingly, ER of an ApproxADD can be given by:

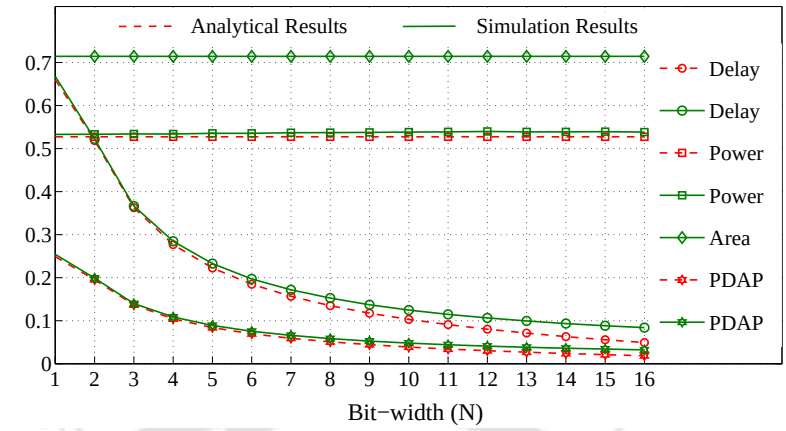
$$ER = \sum_{i=1}^{N} \frac{1}{4} \times \left(\frac{3}{4}\right)^{i-1} = \sum_{i=1}^{N} \frac{3^{i-1}}{4^i} \tag{3.20}$$

3.3.3 Simulation Setup

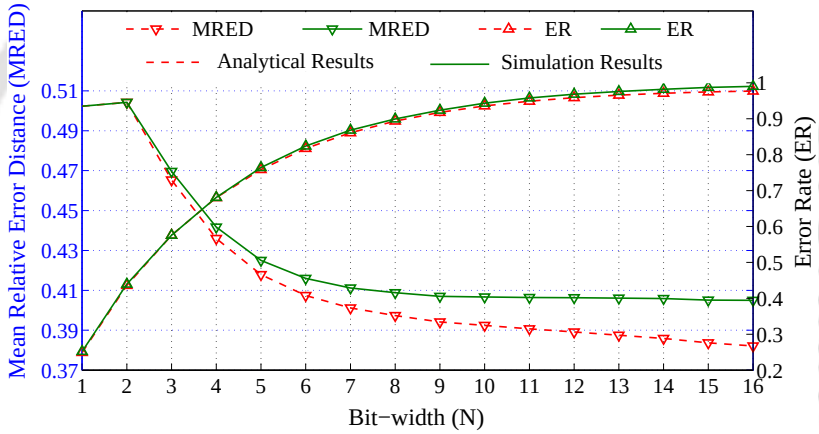
For assessing the simulation results of delay, power and area, we design different configurations of ApproxADD using *Mentor Graphics Tanner schematic capture* [103]. While designing the schematics, we size all the logic gates to template gate according to minimum-size inverter. We then extract netlists from schematics and simulate the extracted netlists using *Synopsys HSPICE circuit simulator* [104] with *Predictive Technology Model* $32nm$ model files [105]. While evaluating the area of ApproxADD, we neglect area imposed by interconnects and extract layout area of the logic gates only from HSPICE netlists. Further, for assessing the quality metrics of ApproxADD, we implement different configurations of ApproxADD in C/C++. We simulate each configuration individually for one billion pseudo-random inputs drawn from a sample space between 0 and $2^N - 1$. The same simulation setup is carried-out in the rest of this Chapter.

3.3.4 Results and Discussion

The analytical results and simulation results of ApproxADD as a function of N for normalized delay, power, area and PDAP are shown in Fig. 3.5(a) and for MRED and ER are shown in Fig. 3.5(b). It can be seen from Fig. 3.5(a) that the normalized delay of ApproxADD decreases with bit-width at a rate $\approx 1/N$. This implies that ApproxADD provides bit-width-aware constant delay



(a)



(b)

Figure 3.5: Analytical and simulation results of N-bit ApproxADD as a function of N: (a) Normalized ($w.r.t.$ N-bit RCA) delay, power, area and PDAP; and (b) MRED (on left y-axis) and ER (on right y-axis).

($O(1)$). ApproxADD also improves dynamic power consumption by 46.31% and area by 28.57% $w.r.t.$ RCA. Further, as shown in Fig. 3.5(b), MRED of ApproxADD decreases with bit-width, but for $N \geq 9$, it saturates. On the other hand, ER of ApproxADD increases rapidly with bit-width (≈ 1 for $N \geq 13$). Here, it should also be noted that the difference between analytical results and simulation results is very small. It means our analytical results agree closely with simulation results. Further, analytical results and simulation results of area are identical as in both cases, we neglect area imposed by interconnects and consider the area of logic gates only.

3.4 Improving ED and ER

It can be seen from Fig. 3.5(a) that ApproxADD provides a significant improvement in delay, power, area and PDAP. However, as shown in Fig. 3.5(b), its: (i) MRED is too high (≈ 0.4 for $N \geq 9$); and (ii) ER increases rapidly with bit-width (≈ 1 for $N \geq 13$). Therefore, ApproxADD may not be preferred for applications in which error-resilience is very low. Moreover, according to the approximate computing design concept of *fail small or fail rare* [8], approximate schemes should be constrained either in ED or in ER. In this Section, we explain our strategies (probability of carry-lifetime and EDC logic) to improve the ED and ER of ApproxADD.

3.4.1 ApproxADD$v1$

We know that in arithmetic operations, impact of errors in higher order bits is more severe as compared to the errors in lower order bits. Therefore, in order to improve the ED of ApproxADD, we perform higher order k-bit operations accurately (see Fig. 3.6). For estimating the number of higher order k-bit that should be performed accurately, we avail the probability of carry-lifetimes. In conventional binary addition process, we define carry-lifetime as the number of bits a carry propagates before annihilation. For example, if a carry generates at m^{th} bit position and propagates to p^{th} bit position, then the carry has a lifetime of $p - m$ bits, where $p > m$. For evaluating the probability of carry-lifetimes in N-bit conventional adders, we implement N-bit conventional adders in C/C++ for different values of N. We then simulate them individually for one billion pseudo-random inputs drawn from a sample space between 0 and $2^N - 1$. Our simulation results for $N = 8$, 16, 32, 64 and 128 are tabulated in Table 3.5. It can be seen from Table 3.5 that the worst-case carry propagation ($i.e.$, carry having lifetime $\approx N$) rarely happens. Therefore, in most of the cases, carry can be determined by considering just few input bits (l) on the right of current bit position [62]. For example, in a 16-bit conventional adder, for $l = 1$, 2 and 3, the probability of getting correct carry is 0.5036, 0.7544 and 0.8789 respectively. However, if the value of l is increased to 8, 9 and 10, then the probability of getting correct carry is 0.9980, 0.9991 and 0.9996, respectively.

Based on the above observations, we perform higher order k-bit operations using conventional FA and lower order $(N - k)$-bit operations using AFA#1. Note that the higher order k-bit operations can

Table 3.5: Probability of carry-lifetimes in N-bit conventional adder for different values of N.

Bit-width (N)	Length of Carry Propagation									
	1	2	3	4	5	6	7	8	9	10
8	0.5622	0.8121	0.9213	0.9685	0.9882	0.9950	0.9990	1.0000	—	—
16	0.5317	0.7815	0.8985	0.9530	0.9785	0.9902	0.9966	0.9980	0.9991	0.9996
32	0.5161	0.7659	0.8866	0.9450	0.9736	0.9873	0.9939	0.9971	0.9986	0.9993
64	0.5036	0.7544	0.8789	0.9405	0.9708	0.9856	0.9929	0.9965	0.9983	0.9991
128	0.5024	0.7531	0.8780	0.9398	0.9704	0.9855	0.9928	0.9965	0.9983	0.9992

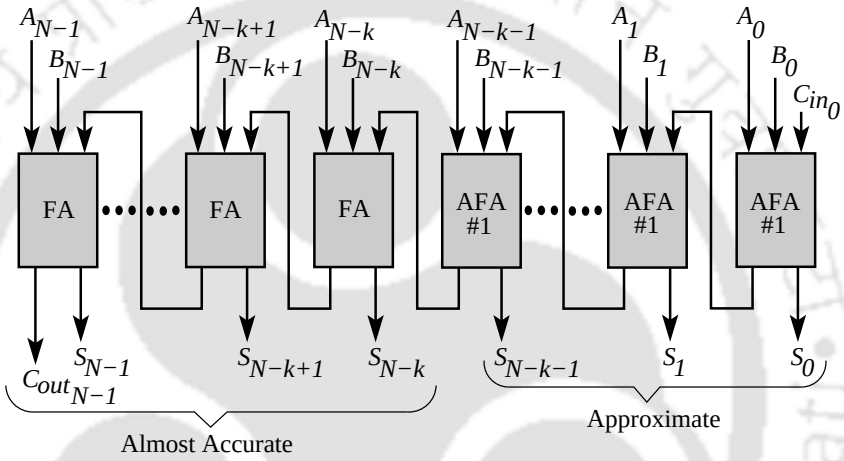


Figure 3.6: Generalized architecture of an N-bit ApproxADD$v1$ which consists of k-bit RCA and $(N-k)$-bit ApproxADD. Here, "almost accurate" signifies that ED and ER of higher order k-bit is very low.

be performed using any adder architecture. Considering RCA architecture, Fig. 3.6 shows modified version of ApproxADD which we call as "ApproxADD$v1$".

3.4.1.1 Delay, Power and Area Estimation

In ApproxADD$v1$, the higher order k-bit and lower order $(N-k)$-bit operations are performed simultaneously. We know that delay of lower order $(N-k)$-bits is constant irrespective of the bit-width. Consequently, the worst-case delay of N-bit ApproxADD$v1$ is same as the k-bit RCA, and therefore, the normalized delay (D_{nm}) of an ApproxADD$v1$ $w.r.t.$ RCA can be given by k/N. Further, using Table 3.3 and Table 3.4, the normalized dynamic power consumption (P_{nm}) and normalized area

(A_{nm}) of an ApproxADD$v1$ $w.r.t.$ RCA can be given by:

$$\begin{aligned} P_{nm} &= \frac{k \times P_{nm\_inv_{FA}} + (N-k) \times P_{nm\_inv_{AFA\#1}}}{N \times P_{nm\_inv_{FA}}} \\ &= \frac{k}{N} \times \left(1 - \frac{P_{nm\_inv_{AFA\#1}}}{P_{nm\_inv_{FA}}}\right) + \frac{P_{nm\_inv_{AFA\#1}}}{P_{nm\_inv_{FA}}} \\ &= 0.5275 + \left(0.4725 \times \frac{k}{N}\right) \end{aligned} \tag{3.21}$$

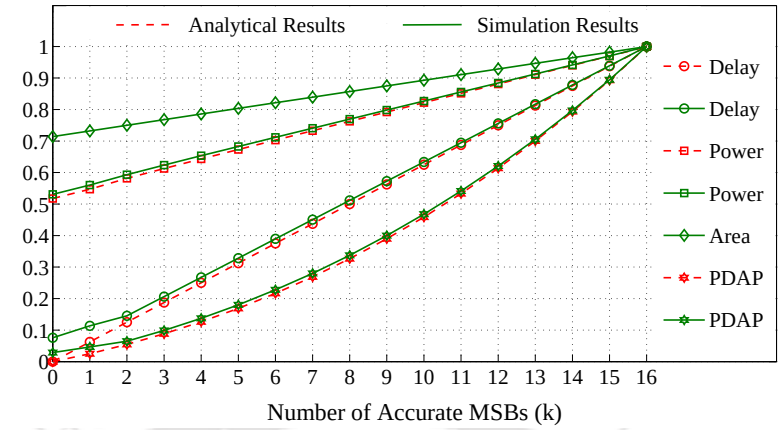
$$\begin{aligned} A_{nm} &= \frac{k \times A_{FA} + (N-k) \times A_{AFA\#1}}{N \times A_{FA}} \\ &= \frac{k}{N} \times \left(1 - \frac{A_{AFA\#1}}{A_{FA}}\right) + \frac{A_{AFA\#1}}{A_{FA}} \\ &= 0.7143 + \left(0.2857 \times \frac{k}{N}\right) \end{aligned} \tag{3.22}$$

3.4.1.2 ED and ER Estimation

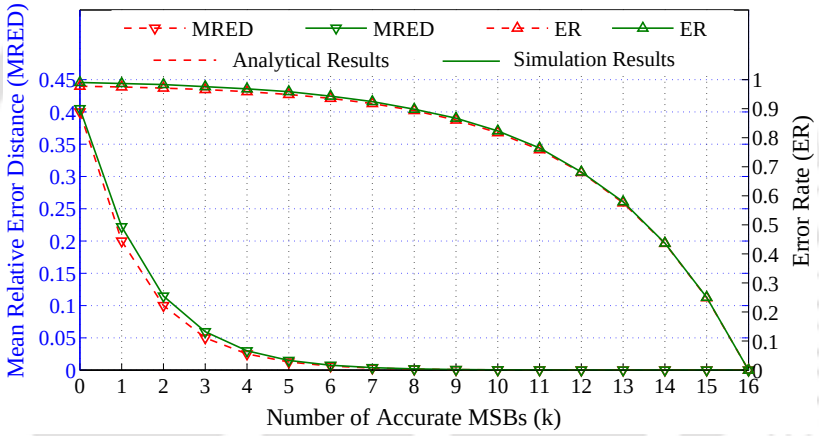
In ApproxADD$v1$, C_{in} of accurate sub-adder is considered as A_{N-k-1}. Based on the discussion presented in Section 3.2, the probability of C_{in} of accurate sub-adder to be erroneous is 0.25. Further, the probability that this erroneous C_{in} will propagate to MSBs depends on k. For example, as per the results shown in Table 3.5, in a 16-bit ApproxADD$v1$, for $k = 7$, 8, 9 and 10, the probability that erroneous C_{in} will propagate to MSB is 0.0044, 0.0020, 0.0009 and 0.0004, respectively. All together, we can deduce that due to cumulative effect, the probability of error in higher order bits is very low (particularly, for $k \geq 8$). Consequently, the error characteristics of N-bit ApproxADD$v1$ are similar to $(N-k)$-bit ApproxADD, and therefore, ED and ER of an N-bit ApproxADD$v1$ can be estimated by replacing N with $(N-k)$ in Eqn. (3.18) and Eqn. (3.20), respectively.

3.4.1.3 Results and Discussion

The analytical results and simulation results of 16-bit ApproxADD$v1$ as a function of k for normalized delay, power, area and PDAP are shown in Fig. 3.7(a) and for MRED and ER are shown in Fig. 3.7(b). It can be seen from Fig. 3.5(b) and Fig. 3.7(b) that the approach used in ApproxADD$v1$ ($i.e.$, to perform higher order k-bit operations accurately) improves MRED significantly. According to our experimental observations, MRED of an N-bit ApproxADD$v1$ can be empirically given by



(a)



(b)

Figure 3.7: Analytical and simulation results of 16-bit ApproxADD$v1$ as a function of k: (a) Normalized ($w.r.t.$ 16-bit RCA) delay, power, area and PDAP; and (b) MRED (on left y-axis) and ER (on right y-axis).

$0.4/2^k$. ApproxADD$v1$ also shows improvement in ER, but the improvement is marginal (particularly, for $k \leq 8$). Here, it should be noted that MRED of ApproxADD$v1$ decreases with k, whereas its normalized delay, power, area and PDAP increase with k. It means ApproxADD$v1$ provides improvement in MRED at the cost of delay, power, area and PDAP. However, the rate at which MRED decreases is higher as compared to the rate at which delay, power, area and PDAP increases. This shows that ApproxADD$v1$ improves MRED in a graceful manner.

3.4.2 ApproxADD$v2$

Although ApproxADD$v1$ provides a significant improvement in ED, it may still be not preferred for some of the error-resilient applications, particularly, for applications which demand low ER. For

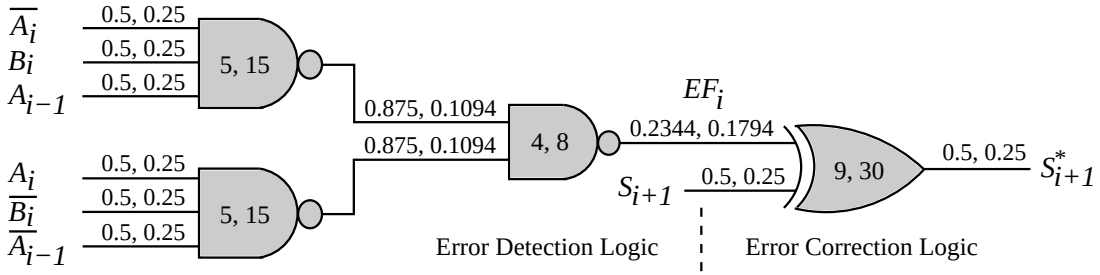


Figure 3.8: Gate level logic implementation of the proposed EDC logic. Note that the numbers indicated on the logic gates and on the input and output nodes of the logic gates have already been discussed in Fig. 3.2.

Table 3.6: Design parameters of the logic gates used in EDC logic (Fig. 3.8) .

Logic Gate	α	g	h	p_d	p_p	a	d	p_{nm_inv}
NAND	0.1094	5/3	4/5	3	3	15	4.33	1.89
NAND	0.1794	4/3	9/4	2	2	8	5	3.59
XOR	0.25	3	$1/3^{\#}$	4	5	30	5	6

Here, $\#$ signifies that while computing the electrical effort (h) of output logic gates, we assume that S^* drives a unit standard CMOS inverter.

improving the ER of ApproxADD$v1$, we exploit the concept of EDC logic that detects errors, and accordingly, corrects them. As shown in Fig. 3.6, an N-bit ApproxADD$v1$ can be considered as k-bit RCA followed by $(N-k)$-bit ApproxADD. As discussed earlier, the probability of error in higher order k-bit is very low. Further, in lower order $(N-k)$-bit, an erroneous carry generates at i^{th} bit position only when $\overline{A}_i.B_i.A_{i-1} = 1$ or $A_i.\overline{B}_i.\overline{A}_{i-1} = 1$, where $i \leq N-k$. The occurrence of such input combinations can be divulged using the error detection logic, as shown in Fig. 3.8. Here, true value of error flag EF_i indicates that an erroneous carry is generated at i^{th} bit position. Further, we know that in addition operation, erroneous carry generated at i^{th} bit position affects the output at $(i+1)^{th}$ bit position. Therefore, output at $(i+1)^{th}$ bit position can be corrected (S_{i+1} is complemented when $EF_i = 1$) using the error correction logic, as shown in Fig. 3.8. The design parameters, such as α, g, h, p_d, p_p, a, d and p_{nm_inv} (which are described in Section 3.1) of the logic gates used in EDC logic are tabulated in Table 3.6. Using Table 3.6, the proposed EDC logic has delay, $D = 4.33 + 5 + 5 = 14.33$, normalized dynamic power consumption, $P_{nm\_inv} = 1.89 + 1.89 + 3.59 + 6 = 13.37$ and area, $A = 15 + 15 + 8 + 30 = 68$ units.

The proposed approach and EDC logic are explained via an example in Fig. 3.9. For the ease

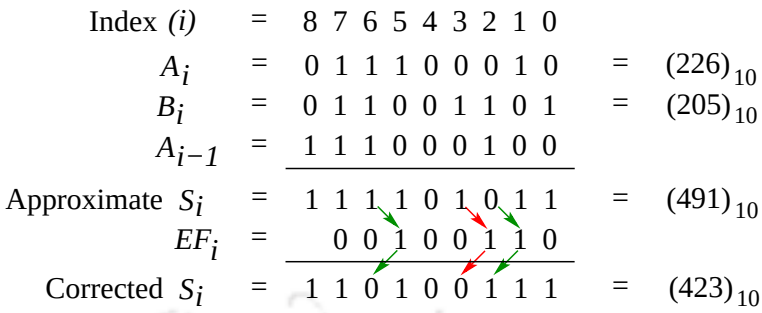


Figure 3.9: Illustration of the proposed approach and EDC logic.

of illustration, we consider two 8-bit numbers. Further, for persuading the compatibility between inputs and result, 0 is appended at MSB position. It can be seen from Fig. 3.9 that the proposed EDC logic is not effective for the consecutive errors. In the proposed approach, two consecutive erroneous carries generated from i^{th} bit position do not introduce error at $(i+2)^{th}$ bit position. However, the proposed EDC logic complements output at $(i+2)^{th}$ bit position, and thus, introduces errors for such consecutive errors. For example, in Fig. 3.9, regardless of the output at 3^{rd} bit position is correct, the proposed ECL complements the output as $EF_2 = 1$. The proposed EDC logic can be customized to handle such cases, however, it imposes hardware complexity.

Now we integrate ApproxADD$v1$ and EDC logic which we call as "ApproxADD$v2$". As shown in Fig. 3.10, ApproxADD$v2$ has two stages. In the first stage, sum bits are generated using k-bit RCA and $(N-k)$-bit ApproxADD. Since error occurrences can be detected simultaneously with sum bits, an $(N-k)$-bit EDL is also implemented in the first stage to actuate EF signals. In the second stage, depending on EF signals, errors are corrected using an $(N-k)$-bit ECL.

3.4.2.1 Delay, Power and Area Estimation

In ApproxADD$v2$, the higher order k-bit and lower order $(N-k)$-bit operations are performed simultaneously. We know that delay of lower order $(N-k)$-bit is constant irrespective of the bit-width. Consequently, the worst-case delay of N-bit ApproxADD$v2$ is same as the k-bit RCA, and therefore, the normalized delay (D_{nm}) of an ApproxADD$v2$ $w.r.t.$ RCA can be given by k/N. Further, using Table 3.3 and Table 3.6, the normalized dynamic power consumption (P_{nm}) and normalized area (A_{nm}) of an ApproxADD$v2$ $w.r.t.$ RCA can be given by:

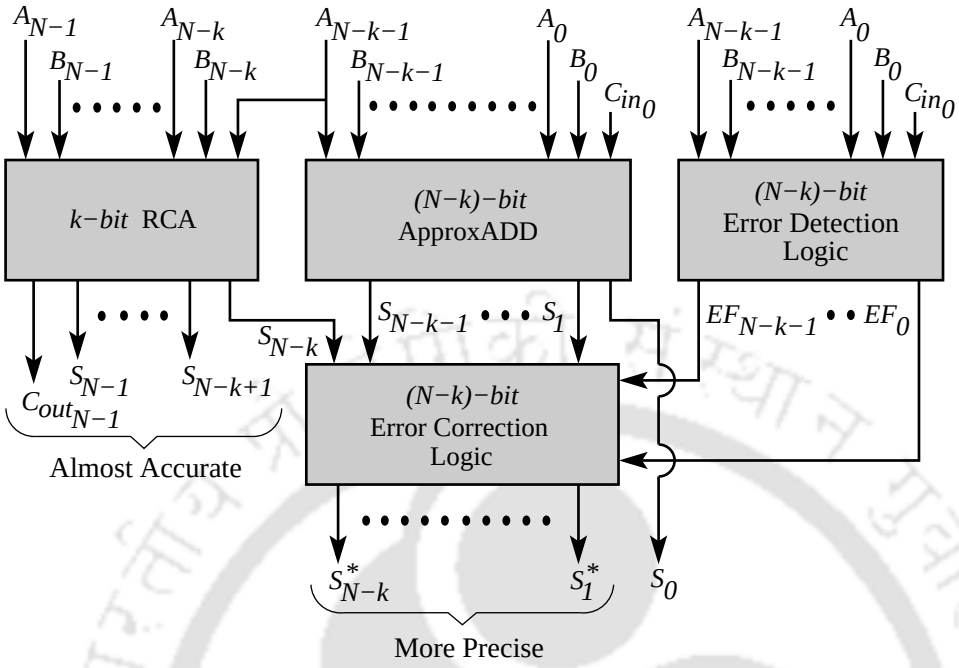


Figure 3.10: Generalized architecture of an N-bit ApproxADD$v2$ which consists of k-bit RCA, $(N-k)$-bit ApproxADD and $(N-k)$-bit EDC logic. Here, "more precise" signifies that the probability of error in $S^*_{N-k:1}$ is less as compared to the probability of error in $S_{N-k:1}$.

$$\begin{aligned} P_{nm} &= \frac{k \times P_{nm\_inv_{FA}} + (N-k) \times (P_{nm\_inv_{AFA\#1}} + P_{nm\_inv_{EDC}})}{N \times P_{nm\_inv_{FA}}} \\ &= \frac{k}{N} \times \left(1 - \frac{P_{nm\_inv_{AFA\#1}}}{P_{nm\_inv_{FA}}} - \frac{P_{nm\_inv_{EDC}}}{P_{nm\_inv_{FA}}}\right) + \frac{P_{nm\_inv_{AFA\#1}}}{P_{nm\_inv_{FA}}} + \frac{P_{nm\_inv_{EDC}}}{P_{nm\_inv_{FA}}} \\ &= 1.0313 - \left(0.0313 \times \frac{k}{N}\right) \end{aligned} \tag{3.23}$$

$$\begin{aligned} A_{nm} &= \frac{k \times A_{FA} + (N-k) \times (A_{AFA\#1} + A_{EDC})}{N \times A_{FA}} \\ &= \frac{k}{N} \times \left(1 - \frac{A_{AFA\#1}}{A_{FA}} - \frac{A_{EDC}}{A_{FA}}\right) + \frac{A_{AFA\#1}}{A_{FA}} + \frac{A_{EDC}}{A_{FA}} \\ &= 1.5238 - \left(0.5238 \times \frac{k}{N}\right) \end{aligned} \tag{3.24}$$

where, $P_{nm\_inv_{EDC}}$ and A_{EDC} are the normalized dynamic power consumption $w.r.t.$ a unit standard CMOS inverter and area of the proposed EDC logic, respectively. Eqn. (3.23) and Eqn. (3.24) show that ApproxADD$v2$ consumes more power and area as compared to the RCA. However, it should be noted that the power and area overheads decrease with k. Further, the rate at which power and area overheads decrease depends on the bit-width.

3.4.2.2 ED and ER Estimation

An N-bit ApproxADD$v2$ can be considered as k-bit RCA followed by $(N-k)$-bit ApproxADD and $(N-k)$-bit EDC logic. As discussed in Fig. 3.9, two consecutive erroneous carries generated from i^{th} bit position do not introduce error at $(i+2)^{th}$ bit position. However, the proposed EDC logic complements output at $(i+2)^{th}$ bit position, regardless of the output at $(i+2)^{th}$ bit position is correct. This implies that the proposed EDC logic corrects error when input combinations $\overline{A}_i.B_i.A_{i-1}$ and $A_i.\overline{B}_i.\overline{A}_{i-1}$ occur separately, but introduces errors when these input combinations occur consecutively. Therefore, ED of an ApproxADD$v2$ can be given by the weighted sum of input combinations which generate consecutive erroneous carries as:

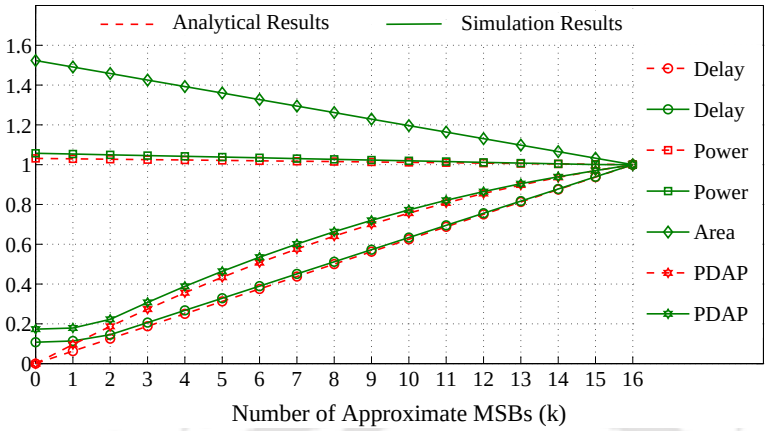
$$\begin{aligned} ED \approx & \sum_{i=0}^{N-k-1} 2^{i+1} \times \Big((\overline{A}_i.B_i.A_{i-1}) \,.\, (A_{i-1}.\overline{B}_{i-1}.\overline{A}_{i-2}) \Big) \\ + & \sum_{i=0}^{N-k-1} 2^{i+1} \times \Big((A_i.\overline{B}_i.\overline{A}_{i-1}) \,.\, (\overline{A}_{i-1}.B_{i-1}.A_{i-2}) \Big) \end{aligned} \tag{3.25}$$

Further, in order to model the ER of ApproxADD$v2$, we use empirical approach. As per our experimental observations, ER of ApproxADD$v2$ forms a GP with scalar factor $\approx 1/9$ and common ratio $\approx 8/9$. Accordingly, ER of an ApproxADD$v2$ can be given by:

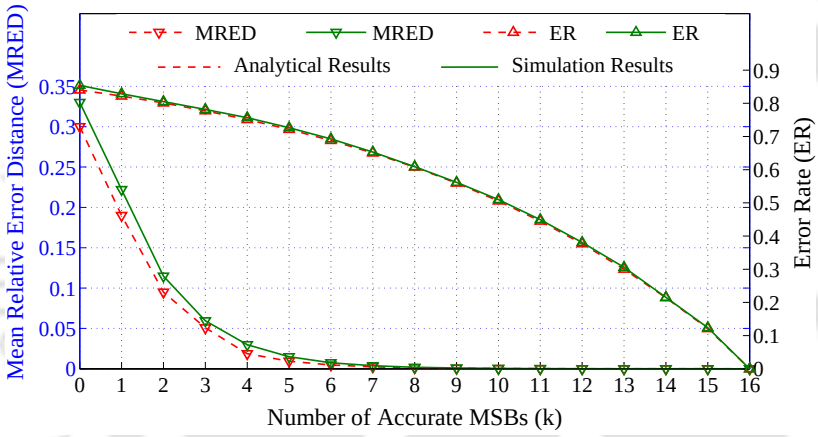
$$ER = \sum_{i=1}^{N} \frac{1}{9} \times \left(\frac{8}{9}\right)^{i-1} = \sum_{i=1}^{N} \frac{8^{i-1}}{9^i} \tag{3.26}$$

3.4.2.3 Results and Discussion

The analytical results and simulation results of 16-bit ApproxADD$v2$ as a function of k for normalized delay, power, area and PDAP are shown in Fig. 3.11(a) and for MRED and ER are shown in Fig. 3.11(b). It can be seen from Fig. 3.7(b) and Fig. 3.11(b) that ApproxADD$v2$ provides a significant improvement in ER. ApproxADD$v2$ provides this improvement at the cost of power and area overheads (see Fig. 3.7(a) and Fig. 3.11(a)). Although ApproxADD$v2$ imposes power and area overheads, but it shows a significant improvement in overall performance which is generally measured in terms of PDAP. Further, in case of parallel-prefix adders (discussed in Section 3.5.1), the proposed approach provides improvement in power as well as area.



(a)



(b)

Figure 3.11: Analytical and simulation results of 16-bit ApproxADD$v2$ as a function of k: (a) Normalized ($w.r.t.$ 16-bit RCA) delay, power, area and PDAP; and (b) MRED (on left y-axis) and ER (on right y-axis).

3.5 Performance Analysis

As discussed in Chapter 1, accuracy of approximate circuits is generally measured in terms of quality metrics. Based on the requirements of the applications, several quality metrics have been introduced in the literature [1, 62, 65, 74, 75]. Two of the quality metrics (MRED and ER) and analytical results as well as simulation results of the proposed ApproxADDs $w.r.t.$ these quality metrics have been discussed in previous Sections. Besides these quality metrics, in error-resilient DSP applications, ACC_{amp} [65] is used to evaluate EDs for the amplitude data as:

$$ACC_{amp}(\%) = \left(1 - \frac{ED}{R_c}\right) \times 100 \tag{3.27}$$

On the other hand, in error-resilient communication applications that mainly handle information data, the number of incorrect bits, $i.e.$, Hamming distance is more meaningful. For such applications, ACC$_{inf}$ [65] is used to evaluate EDs for the information data as:

$$ACC_{inf}(\%) = \left(1 - \frac{B_e}{N}\right) \times 100 \tag{3.28}$$

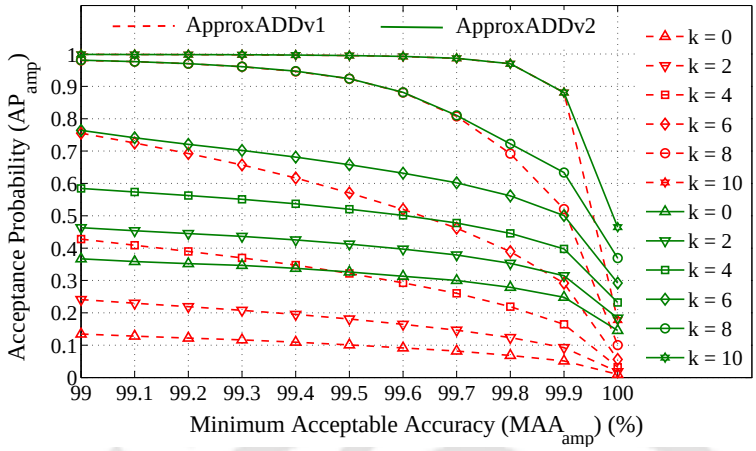
where, B_e is the number of incorrect bits. Further, in error-resilient applications, errors are allowed to exist, however, accuracy of the output should be higher than the acceptance threshold. Different applications have different level of acceptance threshold depending on their inherent error-resilience. Researchers/designers measure acceptance threshold in different ways: (i) Shin et al. [1] measure acceptance threshold in terms of *Distance Rate* (DR), $i.e.$, product of ED and ER; and (ii) Zhu et al. [62] measure acceptance threshold in terms of MAA. If MAA represents acceptance threshold, then the criterion of acceptability is given by the *Acceptance Probability* (AP) as:

$$AP_{amp} = P(ACC_{amp} \geq MAA_{amp}) \tag{3.29}$$

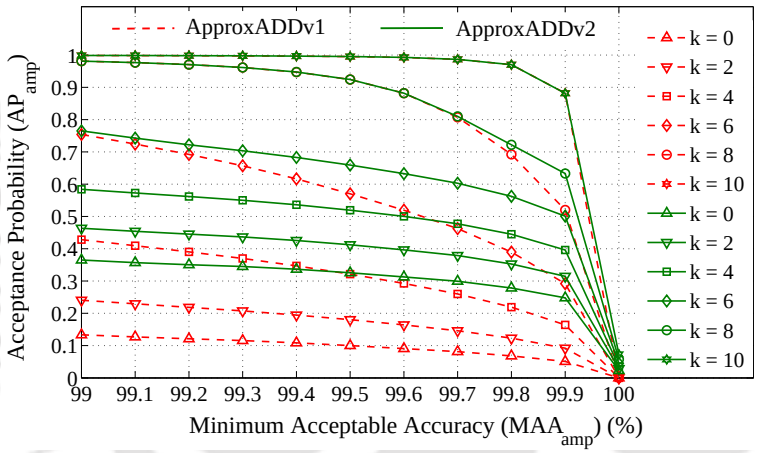
$$AP_{inf} = P(ACC_{inf} \geq MAA_{inf}) \tag{3.30}$$

where, AP$_{amp}$ and MAA$_{amp}$ are the AP and MAA for amplitude data, and AP$_{inf}$ and MAA$_{inf}$ are the AP and MAA for information data, respectively. In order to inspect the effectiveness of the proposed approach, we simulate ApproxADD$v1$ and ApproxADD$v2$ over the above-mentioned quality metrics. Our simulation results of AP$_{amp}$ and AP$_{inf}$ for 16- and 32-bit ApproxADD$v1$ and ApproxADD$v2$ as a function of k are shown in Fig. 3.12 and Fig. 3.13, respectively. Further, for some particular values of DR, Fig. 3.14 shows the relationship between N and k for ApproxADD$v2$. Based on Fig. 3.12, Fig. 3.13 and Fig. 3.14, some of our important observations are as follows.

(i) Fig. 3.12(a) and Fig. 3.12(b) show that for a fixed value of k, both ApproxADD$v1$ and ApproxADD$v2$ provide similar AP$_{amp}$ for $N = 16$ and 32 (except for MAA$_{amp} = 100\%$). This implies that both in ApproxADD$v1$ and ApproxADD$v2$, for a fixed value of k, AP$_{amp}$ does not depend on the bit-width. In other words, for a particular value of AP$_{amp}$, k is independent of bit-width. Therefore, in case of AP$_{amp}$, ApproxADD$v1$ and ApproxADD$v2$ provide



(a)

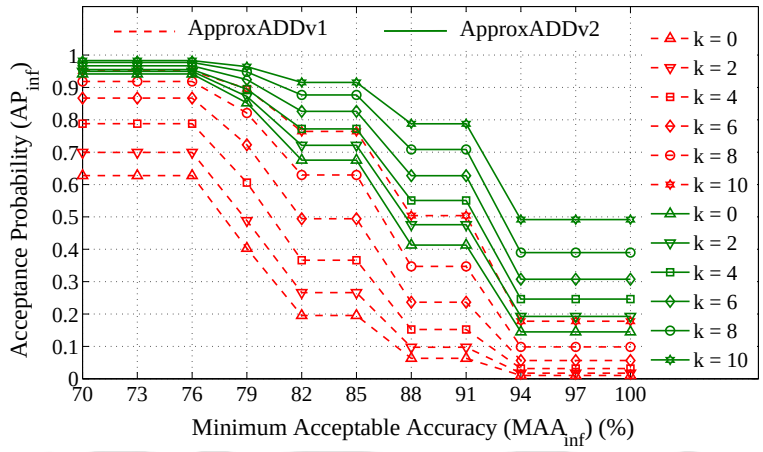


(b)

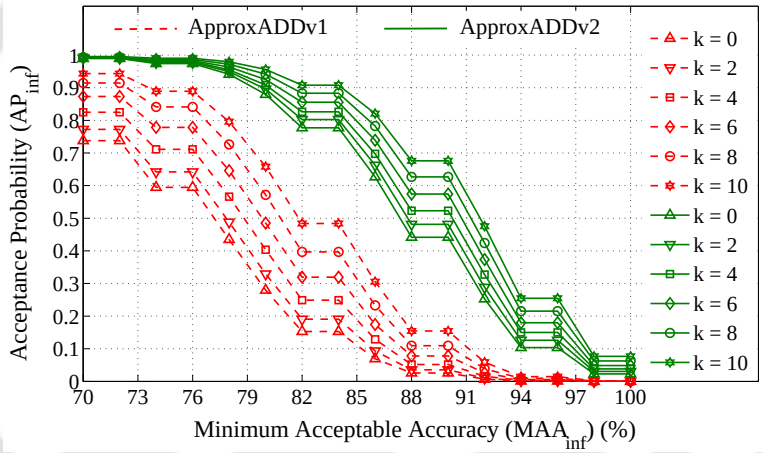
Figure 3.12: AP_{amp} of ApproxADD$v1$ and ApproxADD$v2$ for different values of k: (a) 16-bit ApproxADD$v1$ and ApproxADD$v2$; (b) 32-bit ApproxADD$v1$ and ApproxADD$v2$.

bit-width-aware constant delay ($O(1)$).

(ii) Fig. 3.12(a) and Fig. 3.12(b) show that for higher values of k ($k \geq 10$), ApproxADD$v1$ and ApproxADD$v2$ provide similar AP$_{amp}$ (except for MAA$_{amp} = 100\%$). Therefore, in error-resilient DSP applications that demand higher AP$_{amp}$, ApproxADD$v1$ is preferred over ApproxADD$v2$ as both ApproxADD$v1$ and ApproxADD$v2$ provide similar AP$_{amp}$, but ApproxADD$v1$ consumes less power and area than ApproxADD$v2$ (see Fig. 3.7(a) and Fig. 3.11(a)). On the other hand, Fig. 3.13(a) and Fig. 3.13(b) show that for a fixed value of k, ApproxADD$v2$ provides higher AP$_{inf}$ than ApproxADD$v1$. Therefore, in error-resilient communication appli-



(a)



(b)

Figure 3.13: AP_{inf} of ApproxADD$v1$ and ApproxADD$v2$ for different values of k: (a) 16-bit ApproxADD$v1$ and ApproxADD$v2$; (b) 32-bit ApproxADD$v1$ and ApproxADD$v2$.

cations that demand higher AP$_{inf}$, ApproxADD$v2$ is preferred over ApproxADD$v1$.

(iii) Fig. 3.14 shows that for higher values of DR (DR $\geq$ 1%), k is almost independent of N, whereas for smaller values of DR (DR $\leq$ 0.01%), k varies linearly with N. This implies that for smaller values of DR, delay of N-bit ApproxADD$v2$ varies linearly with bit-width. However, if constraints on DR are relaxed, then ApproxADD$v2$ provides bit-width-aware constant delay ($O(1)$) in this case also. Our simulation results show that ApproxADD$v1$ also shows similar behaviour, but ApproxADD$v1$ provides bit-width-aware constant delay ($O(1)$) for higher values of DR as compared to ApproxADD$v2$.

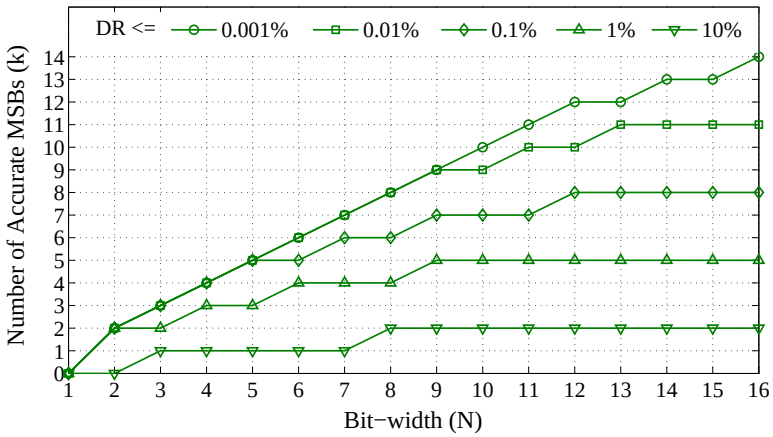


Figure 3.14: For some particular values of DR (0.001%, 0.01%, 0.1%, 1% and 10%), relationship between N and k for N-bit ApproxADD$v2$, where $N = 1, 2, 3 \ldots 16$.

3.5.1 Effect of Adder Architectures

The final version of the proposed approach ($i.e.$, ApproxADD$v2$) is shown in Fig. 3.10, where we perform higher order k-bit operations using RCA and lower order $(N - k)$-bit operations using ApproxADD and EDC logic. For the ease of illustration, we use RCA architecture for higher order k-bit operations. However, the higher order k-bit operations can be performed using any adder architecture. In order to evaluate – how the effectiveness of the proposed approach varies with higher order k-bit adder architectures, we perform higher order k-bit operations using *Brent-Kung* (BK) [106], *Kogge-Stone* (KS) [107] and *Sklansky* (Sk) [108] architectures. We choose BK, KS and Sk architectures as these are the fundamental and most widely used parallel-prefix architectures. We call the resulting approximate adders as ApproxBK, ApproxKS and ApproxSk, respectively.

Since approximation is introduced in lower order $(N - k)$-bit only, the adder architecture used for higher order k-bit makes no difference to the quality metrics. Therefore, the analytical results and simulation results shown in Fig. 3.11(b), Fig. 3.12 and Fig. 3.13 are valid for ApproxBK, ApproxKS and ApproxSk also. On the other hand, different adder architectures have different delay, power and area. Therefore, the adder architecture used for higher order k-bit shows a significant difference in delay, power and area. Our simulation results for normalized delay, power and area of 16-bit ApproxBK, ApproxKS and ApproxSk $w.r.t.$ BK, KS and Sk adders, respectively, as a function of k are shown in Fig. 3.15. It can be seen from Fig. 3.11(a) and Fig. 3.15 that the higher order

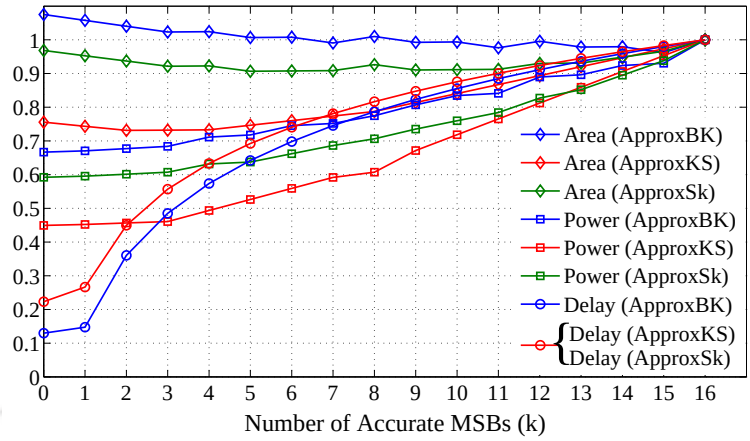


Figure 3.15: Normalized delay, power and area of 16-bit ApproxBK, ApproxKS and ApproxSk $w.r.t.$ BK, KS and Sk adders, respectively, as a function of k.

Table 3.7: For $k = 8$ and 10, percentage change (Δ) in D, P, A and $PDAP$ of 16-bit ApproxADD$v2$, ApproxBK, ApproxKS and ApproxSk $w.r.t.$ 16-bit RCA, BK, KS and Sk adders, respectively.

Approximate Adders	$k = 8$				$k = 10$			
	ΔD	ΔP	ΔA	$\Delta PDAP$	ΔD	ΔP	ΔA	$\Delta PDAP$
ApproxADD$v2$	-48.840	3.0687	26.190	-39.880	-36.630	2.3112	19.642	-27.985
ApproxBK	-21.319	-22.537	0.9973	-38.444	-14.454	-16.536	-0.6299	-29.050
ApproxKS	-18.340	-39.284	-21.254	-60.957	-12.441	-28.152	-15.940	-47.119
ApproxSk	-18.340	-29.347	-7.3758	-46.560	-12.441	-24.018	-8.8416	-39.353

k-bit adder architecture shows contradictory behavior. In Fig. 3.11(a), the normalized power and area are higher than one and decrease with k, whereas in Fig. 3.15, the normalized power and area are lower than one and increase with k. This is because the proposed approximate cell (AFA#1 + EDC) has overall power and area more than FA cell, but lower than BK, KS and Sk cells. Further, the rate at which the normalized delay, power and area changes also depends on the higher order k-bit adder architecture. Here, it should also be noted that if we perform higher order k-bit operations using RCA architecture, then the proposed approach imposes power and area overheads $w.r.t.$ RCA. However, if we perform higher order k-bit operations using state-of-the-art adder architectures (such as, BK, KS and Sk), then the proposed approach provides a significant improvement in power and area $w.r.t.$ corresponding conventional adder. Further, in arithmetic operations, higher order bits are more important as compared to the lower order bits. As discussed earlier, for $k \geq 8$, ED and ER of higher order k-bits is very low. Therefore, for $k = 8$ and 10, the percentage change in delay (ΔD),

Table 3.8: Comparison of the proposed approach with existing work.

Design Metrics	RCA	ETA-1 [5]	SFA [1]	LOA [2]	SA [58]	ApproxADD$v1$	ApproxADD$v2$
Delay (ps)	335.64	335.64	335.64	335.64	335.64	335.64	335.64
Power (μW)	592.68	272.17	213.84	187.81	173.43	224.53	349.78
Area (μm^2)	408.82	236.28	183.61	112.47	101.79	194.16	313.76
MRED	0.0000	1.4E-4	3.3E-4	1.2E-4	1.6E-4	9.5E-5	2.5E-5
ER	0.0000	0.9991	0.9733	0.9992	0.9999	0.9967	0.8745
Acc$_{amp}$ (avg.)	1.0000	0.9986	0.9993	0.9990	0.9982	0.9991	0.9997
Acc$_{inf}$ (avg.)	1.0000	0.6295	0.7303	0.6049	0.5817	0.8362	0.9221

power (ΔP), area (ΔA) and PDAP ($\Delta PDAP$) of 16-bit ApproxADD$v2$, ApproxBK, ApproxKS and ApproxSk $w.r.t.$ RCA, BK, KS and Sk adders, respectively, are tabulated in Table 3.7. It can be seen from Table 3.7 that the proposed approach provides the highest improvement in delay for RCA architecture and in power, area and PDAP for KS architecture.

3.5.2 Comparison

For evaluating the efficiency of the proposed approach, we compare ApproxADD$v1$ and ApproxADD$v2$ with existing approximate adders. For comparison purposes, we choose ETA-1 [5], SFA [1], LOA [2] and *Soares's Adder* (SA) [58]. In order to have a fair comparison: (i) We design these approximate adders using RCA architecture; and (ii) While designing, we keep the maximum length of carry propagation ($i.e.$, value of k) same in these approximate adders.

Our simulation results for RCA (as a reference) and all the above-mentioned approximate adders along with ApproxADD$v1$ and ApproxADD$v2$ for $N = 32$ and $k = 12$ are tabulated in Table 3.8. It can be seen from Table 3.8 that AFA-based approximate adders provide higher ER. Therefore, these are suitable for applications in which error-resilience is very high. As compared to SFA, LOA and SA, ApproxADD$v1$ consumes more power and area, but it provides higher ACC$_{inf}$ than SFA, LOA and SA. Consequently, in error-resilient communication applications, ApproxADD$v1$ is preferred over SFA, LOA and SA. Moreover, ApproxADD$v1$ provides better MRED, ER and ACC$_{amp}$ than LOA and SA. Further, as compared to ETA-1, ApproxADD$v1$ provides smaller power and area, smaller MRED and ER, and higher ACC$_{amp}$ and ACC$_{inf}$. Therefore, both in error-resilient DSP applications and communication applications, ApproxADD$v1$ is preferred over ETA-1.

3.5.3 Multimedia Applications

We use many electronic systems equipped with multimedia applications in our daily lives. As the end users of multimedia applications are human beings, multimedia applications are error-resilient due to the limited perception of human senses. The limited perception of human senses removes the strict restrictions on accuracy and allows the output of multimedia applications to be approximate, rather than fully accurate. Moreover, in most of the multimedia applications, the commonly used image processing algorithms, such as *Joint Photographic Experts Group* (JPEG) have inherent error-resilience. In such algorithms, DCT and IDCT are the integral components. DCT and IDCT involve a large number of multiplication and addition operations. Multiplication can further be computed using shift and add operations. In summary, adders can be treated as the basic building blocks of DCT and IDCT modules. In order to inspect the effectiveness of the proposed approach in real-life applications, we replace the conventional addition operations in DCT and IDCT modules with AFA-based approximate adders. As Lena picture is one of the most widely used standard test image, we use the same Lena picture as benchmark. Further, there are two principle methods of evaluating the image quality [91]: (i) Subjective; and (ii) Objective. For having a fair assessment of image quality, we evaluate output images both subjectively as well as objectively.

In subjective method, image quality is evaluated by human beings. For subjective evaluation, Fig. 3.16 shows the original Lena image and output images after performing DCT and IDCT using 24-bit ApproxADD$v2$ with different values of k. It can be seen from Fig. 3.16 that for $k \geq 16$, visual quality loss to the output images is negligible. For $12 \leq k \leq 14$, output images show some blockiness that may be tolerated by human vision, and thus, may be acceptable for some of the error-resilient applications. However, with further decrease in the value of k (particularly, for $k < 10$), there is severe degradation in the image quality which is not favorable.

Subjective method is the only correct method for evaluating the image quality [109]. However, this method is very expensive and time consuming. Besides this, evaluation of image quality using subjective method is not feasible for real-time applications. In such a scenario, researchers/designers evaluate the image quality using objective method. In objective method, image quality is evaluated by automated computer programs using *Image Quality Assessment* (IQA) metrics. Based on the

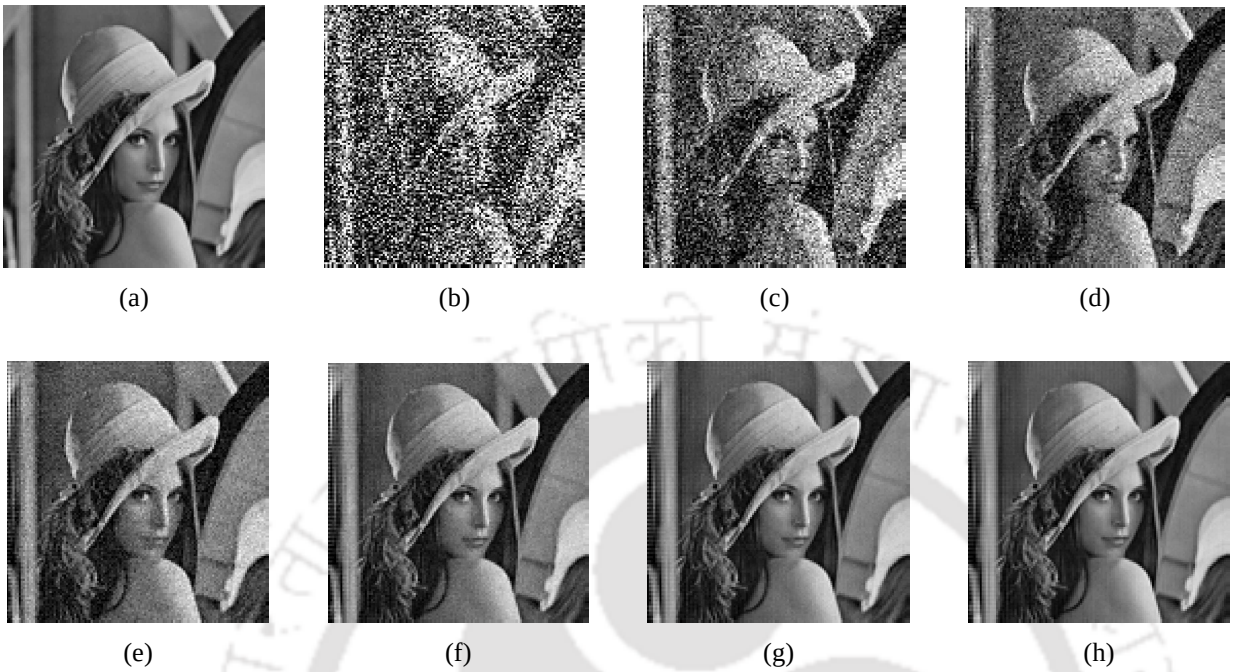


Figure 3.16: Image processing results: (a) Original image and images processed using 24-bit ApproxADD$v2$ with: (b) $k = 8$; (c) $k = 10$; (d) $k = 12$; (e) $k = 14$; (f) $k = 16$; (g) $k = 18$; and (h) $k = 20$.

applications, several IQA metrics have been introduced in the literature [110]. The three fundamental and most widely used IQA metrics are: (i) MSE; (ii) PSNR; and (iii) *Structural Similarity Index Metric* (SSIM) [111]. We evaluate the output images (shown in Fig. 3.16) for all the above-mentioned IQA metrics. Our simulation results are tabulated in Table 3.9. Note that the numbers tabulated in Table 3.9 correspond to different delay, power and area points. It can be seen from Table 3.9 that for $k \leq 8$, values of IQA metrics are not acceptable (particularly, value of SSIM). For $10 \leq k \leq 12$, values of IQA metrics may be acceptable for some of the error-resilient applications. However, for $k \geq 14$, 24-bit ApproxADD$v2$ shows favorable values of IQA metrics. Further, Table 3.9 also summarizes error characteristics, such as mean (μ) and standard deviation (σ) of the output images shown in Fig. 3.16. Here, it should be noted that mean of the error is very close to zero. This implies that the proposed approach overestimates and underestimates the accurate results likely, $i.e.$, error is equally distributed around zero. Therefore, the proposed approach is very suitable for computing the operations in which adder is first used for multiple times and then the output is used for further processing. One example of such operations is matrix multiplication. If ApproxADD$v2$ is used in such operations, then the overall error is expected to be negligible.

Table 3.9: Error characteristics and IQA metrics of the output images processed using 24-bit ApproxADD$v2$ for different values of k. Table also summarizes IQA metrics of the output images processed using 24-bit existing AFA-based approximate adders for different values of k.

Quality Metrics			$k=8$	$k=10$	$k=12$	$k=14$	$k=16$	$k=18$	$k=20$
Approx-ADD$v2$	Error	μ	-1.41E-14	-1.37E-14	-1.34E-14	-1.32E-14	-1.30E-14	-1.29E-14	-1.28E-14
		σ	153.3015	121.8938	98.2511	86.1178	78.1383	73.5977	69.5073
	MSE		2.35E+4	1.12E+4	7.69E+3	4.30E+3	2.09E+3	9.13E+2	4.81E+2
	PSNR (dB)		4.42031	7.62429	9.27493	11.7951	14.9168	18.5239	21.3007
	SSIM		0.06187	0.21220	0.42724	0.61751	0.76165	0.85263	0.89768
ETA-1 [5]	MSE		5.01E+4	3.18E+4	1.99E+4	9.49E+3	3.92E+3	1.38E+3	6.25E+2
	PSNR (dB)		1.12654	3.10157	5.12845	8.35362	12.1927	16.7026	20.1664
	SSIM		0.01253	0.04606	0.10136	0.32812	0.58895	0.80081	0.88745
SFA [1]	MSE		3.81E+4	1.83E+4	1.20E+4	6.30E+3	2.79E+3	1.07E+3	5.22E+2
	PSNR (dB)		2.31735	5.49011	7.31658	10.1319	13.6710	17.8206	20.9464
	SSIM		0.03368	0.12530	0.20078	0.51543	0.70428	0.82775	0.89049
LOA [2]	MSE		5.55E+4	4.18E+4	2.48E+4	1.25E+4	4.93E+3	1.67E+3	6.66E+2
	PSNR (dB)		0.68401	1.91422	4.17132	7.14613	11.1946	15.8931	19.8960
	SSIM		0.00776	0.02188	0.05834	0.16709	0.54232	0.78979	0.88464
SA [58]	MSE		6.07E+4	5.25E+4	3.21E+4	1.42E+4	5.45E+3	1.84E+3	7.11E+2
	PSNR (dB)		0.29431	0.92786	3.05849	6.59293	10.7662	15.4752	19.6092
	SSIM		0.00130	0.00877	0.03892	0.12190	0.53621	0.75788	0.88161

For the sake of comparison at application level, we perform DCT and IDCT using the existing AFA-based approximate adders [1, 2, 5, 58]. Our simulation results of IQA metrics after performing DCT and IDCT using 24-bit above-mentioned AFA-based approximate adders for different values of k are tabulated in Table 3.9. It can be seen from Table 3.9 that ApproxADD$v2$ provides better IQA metrics (smaller MSE, and higher PSNR and SSIM) as compared to the ETA-1, SFA, LOA and SA. Note that the improvement in IQA metrics (due to the proposed approach $w.r.t.$ ETA-1, SFA, LOA and SA) depends significantly on k. As k increases, less number of bits are left for approximation, and thus, improvement in IQA metrics (due to the proposed approach $w.r.t.$ ETA-1, SFA, LOA and SA) decreases. However, for $k \leq 14$, ApproxADD$v2$ provides a significant improvement in IQA metrics, particularly, $w.r.t.$ ETA-1, LOA and SA.

Further, we know that the effectiveness of approximate adders depends on the input data type. Based on the brightness, sharpness, contrast, etc., different images represent different data types. Therefore, in order to have a fair evaluation of the proposed approach in real-life applications, we perform DCT and IDCT using 24-bit ApproxADD$v2$ for different images. In addition to Lena, other

Table 3.10: Error characteristics and IQA metrics of the output images (Cameraman, Mandrill and Girl) after performing DCT and IDCT using 24-bit ApproxADD$v2$ for different values of k.

Quality Metric			$k=8$	$k=10$	$k=12$	$k=14$	$k=16$	$k=18$	$k=20$
Camera-man	Error	μ	-1.67E-14	-1.60E-14	-1.56E-14	-1.51E-14	-1.48E-14	-1.46E-14	-1.45E-14
		σ	168.0842	140.8599	114.2347	99.3723	84.6212	78.4631	72.2417
	MSE		2.51E+4	1.20E+4	8.42E+3	4.65E+3	2.29E+3	9.71E+2	5.26E+2
	PSNR (dB)		4.11937	7.33169	8.87510	11.4550	14.5248	18.2573	20.9134
	SSIM		0.05767	0.20456	0.40822	0.59874	0.74183	0.84235	0.88213
Mandrill	Error	μ	-1.18E-14	-1.14E-14	-1.11E-14	-1.08E-14	-1.06E-14	-1.05E-14	-1.04E-14
		σ	131.9697	102.8413	88.1139	78.4913	69.4467	66.4362	64.4001
	MSE		1.97E+4	9.69E+3	6.79E+3	3.52E+3	1.71E+3	7.36E+2	3.82E+2
	PSNR (dB)		5.16941	8.26603	9.81173	12.6629	15.7841	19.4589	22.3059
	SSIM		0.07496	0.23071	0.45167	0.66942	0.79542	0.88678	0.93004
Girl	Error	μ	-9.83E-15	-9.79E-15	-9.76E-15	-9.74E-15	-9.72E-15	-9.71E-15	-9.71E-15
		σ	119.7677	94.4769	80.6031	71.9206	63.9084	61.5285	62.2895
	MSE		1.36E+4	7.04E+3	4.61E+3	2.56E+3	1.22E+3	4.94E+2	2.60E+2
	PSNR (dB)		6.78530	9.65301	11.4845	14.0346	17.2634	21.1869	23.9704
	SSIM		0.09471	0.28663	0.52942	0.72547	0.86614	0.91204	0.95101

commonly used standard test images are Cameraman, Mandrill and Girl [90]. Our simulation results of IQA metrics after performing DCT and IDCT using 24-bit ApproxADD$v2$ for above-mentioned images are tabulated in Table 3.10. It can be seen from Table 3.9 and Table 3.10 that for different images, ApproxADD$v2$ provides different IQA metrics. ApproxADD$v2$ provides better IQA metrics for Girl and Mandrill images as compared to the Lena and Cameraman images.

3.6 Summary

In this Chapter, we proposed four AFAs by re-designing the conventional FA (at gate level of abstraction) in such a way that C_{out} remains independent of C_{in} with minimal ER. Using the proposed AFA#1 (the optimal one), we designed an N-bit approximate adder which we called as "ApproxADD". An emergent property of ApproxADD is that carries do not propagate in it, and thus, it provides bit-width-aware constant delay ($O(1)$). ApproxADD also improves dynamic power consumption by 46.31% and area by 28.57% $w.r.t.$ RCA. In order to improve the ED and ER of ApproxADD, we exploited the concept of carry-lifetime and EDC logic, respectively. In this way, we introduced two more (improved) versions of ApproxADD – ApproxADD$v1$ and ApproxADD$v2$. In addition to the designing of ApproxADDs, we provided a detailed analysis and analytical modeling

to estimate accuracy, delay, power and area of ApproxADDs. For the ease of illustration, we discussed the proposed approach $w.r.t.$ RCA, however, we demonstrated that the proposed approach can be customized for any adder architecture. For some of the adder architectures ($e.g.$, RCA), the final version of the proposed approach imposes minor power and area overheads, but it shows significant improvement in overall performance which is generally measured in terms of PDAP. Further, for state-of-the-art adder architectures ($e.g.$, BK, KS and Sk), the final version of the proposed approach shows significant improvement in power as well as area. As compared to the existing AFA-based approximate adders (ETA-1, SFA, LOA, SA), ApproxADDs provide smaller MRED and ER, and higher ACC_{amp} and ACC_{inf}. Therefore, both in error-resilient DSP applications and communication applications, ApproxADDs are preferred over the existing AFA-based approximate adders. Further, for inspecting the effectiveness of the proposed approach in real-life applications, we demonstrated image compression and decompression by replacing the conventional addition operations in DCT and IDCT modules with 24-bit approximate adders. Our image processing results showed that for $k \geq 16$, ApproxADD$v2$ provides favorable image quality. Further, as compared to the existing AFA-based approximate adders, ApproxADD$v2$ provides better IQA metrics.

There are two fundamental approaches used for designing the approximate adders: (i) AFA; and (ii) ESA. Both these design approaches have their own pros and cons. In this Chapter, we have presented the proposed work on AFAs. In the next Chapter, we discuss the proposed work on ESAs – analytical modeling, optimization and accuracy enhancement.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

4

ESA-based Approximate Adders

Contents

Due to the additional features of design flexibility and programmability, ESA-based approximate adders have attracted a lot of attention of researchers and industry as compared to the AFA-based approximate adders. Further, ESAs are the basic building blocks of ACAs and VLSAs. Over the past decade, several approximate adders have been proposed in the literature based on ESA [12, 14, 61–73]. An N-bit ESA has two primary design parameters: (i) Segment size (k), which represents the maximum length of carry propagation; and (ii) Overlapping bits (l), which represents the minimum number of bits used in carry prediction, where $1 \leq k < N$ and $0 \leq l < k$. Based on the combinations of k and l, an N-bit ESA has $N(N-1)/2$ possible configurations.

In this Chapter, we first propose analytical models to estimate accuracy, delay, power and area of ESAs. From the proposed analytical models, we observe that in an N-bit ESA, there exist multiple configurations which exhibit similar accuracy, however, these configurations exhibit different delay, power and area. Therefore, for a given accuracy, the configurations which provide minimal delay, power and/or area need to be known apriori for efficient, intelligent and goal oriented implementations of ESAs. In this regard, we present an optimization framework that exploits the proposed analytical models and reveals these optimal configurations. Further, we observe that there is a scope of improvement in accuracy-effort curves of ESAs. The accuracy-effort curves of ESAs need to be improved so that their accuracy degrade in a graceful manner, and consequently, they provide optimal delay-accuracy, power-accuracy and area-accuracy trade-offs. Intended to improve the accuracy-effort curves of ESAs, we propose modifications. As mentioned in Chapter 1, accuracy-effort curves can be improved either: (i) By improving the accuracy without imposing any additional delay, power and area overheads; or (ii) By reducing the delay, power and area without loosing accuracy. We achieve our objective using the former approach. For evaluating the effectiveness of the proposed approach in real-life applications, we demonstrate Lena image processing using original ESAs and modified ESAs. In the end, we summarize the Chapter.

The rest of this Chapter is organized as follows. Section 4.1 presents the analytical models to estimate accuracy, delay, power and area of ESAs. Section 4.2 provides the optimization framework and its results and discussion using RFD heuristic algorithm. Section 4.3 discusses the modified ESAs and evaluates their effectiveness over the original ESAs. Finally, Section 4.4 concludes the Chapter.

4.1 Analytical Models

The design metrics, such as accuracy, delay, power and area of an ESA can be evaluated either through computer simulations or through analytical models. In case of computer simulations, design metrics are evaluated by implementing the ESA and then simulating it, whereas in case of analytical models, design metrics are evaluated by solving mathematical equations. While evaluating the design metrics through computer simulations, ESAs can be simulated either exhaustively ($i.e.$, for all possible inputs) or for a fixed number of inputs generated using Monte-Carlo methods. Exhaustive simulations become infeasible (particularly, for $N \geq 12$ [78]) as the required simulation time increases exponentially with bit-width (see Fig. 1.7). One possible way to handle the simulation time is to simulate ESAs for a fixed number of inputs generated using Monte-Carlo methods. However, Monte-Carlo simulations have their own limitations [78]. These are: (i) In computer simulations, it is difficult to observe the effect of different configurations as each configuration needs to be implemented and simulated individually which requires extensive programming efforts and considerable simulation time; (ii) Computer simulations do not provide any information about the causes of error, whereas analytical models provide a systematic understanding of the relationship between different configurations and their design metrics; and (iii) While designing EDC logic for ESAs, knowing the input conditions that lead to error can provide efficient, intelligent and goal oriented implementations [85], however, computer simulations do not provide any such information. In summary, there is a requirement of analytical models to estimate the design metrics of ESAs.

4.1.1 Accuracy Estimation

As discussed in Section 1.3, accuracy of approximate circuits is generally measured in terms of quality metrics. Over the decades, several quality metrics have been introduced in the literature [1, 62, 65, 74, 75]. The choice of a particular quality metric depends on the applications. However, it has been observed in [76, 78] that the combination of ED, ER, MED and MSE is sufficient to evaluate the overall accuracy of ESAs. Here, we discuss the proposed analytical models to estimate these four quality metrics. We evaluate the effectiveness of the proposed analytical models by comparing them with simulation results as well as with existing analytical models.

Table 4.1: Latest research work on analytical modeling of ESAs.

ESAs	ER	ED_{max}	MED	MSE
$l = 0$	[76, 77]	—	[76, 77]	[76, 77]
$l = k/2$	[65, 76, 77]	—	[76, 77]	[76, 77]
$l = k - 1$	[63, 76, 77]	[83]	[76, 77]	[76]
Generalized[#]	[69, 78, 79]	—	[78, 79]	[79]

[#] Although the approaches discussed in [69, 78, 79] are generalized, but the derivation of generalized model is not feasible (particularly, for $n > 5$ [78]) due to the computational complexity.

4.1.1.1 ER Estimation

Among all the above-mentioned quality metrics, ER is considered the most important quality metric, particularly, in case of ESAs. This is because the effectiveness of ACAs and VLSAs (which are two major future design aspects of ESAs) depends on ER [14, 61, 65, 68–70, 73, 85, 86]. Consequently, as shown in Table 4.1, most of the existing work on analytical modeling has been proposed on ER. We also emphasize more on ER as compared to other quality metrics.

In ESAs, C_{in} of all sub-adders is considered as 0. Therefore, output of the least significant sub-adder is always correct as C_{in} of the sub-adder is correct. However, output of any other sub-adder can be incorrect due to the truncation of carry-chain. For any other sub-adder, output is incorrect only if: (i) C_{in} of the sub-adder is incorrect; and (ii) This incorrect C_{in} propagates through carry prediction bits to the output of the sub-adder. Let $P(G_i)$ represents the probability of C_{in} of i^{th} *Sub-Adder* (SA_i) to be incorrect and $P(P_i)$ represents the probability of this incorrect C_{in} to propagate through carry prediction bits to the output of SA_i. Now, ER of SA_i can be given by:

$$ER_{SA_i} = P(G_i)P(P_i) \tag{4.1}$$

Further, we know that probability of output to be correct is equal to one minus the probability of output to be incorrect. Accordingly, $PR_{SA_i} = 1 - ER_{SA_i}$, where PR represents the *Pass Rate* and is the probability of output to be correct. PR of SA_i can be given using Eqn. (4.1) as:

$$PR_{SA_i} = 1 - P(G_i)P(P_i) \tag{4.2}$$

In an ESA, there are total n sub-adders, where $n = \lceil \frac{N-k}{k-l} \rceil + 1$. Since final output is obtained by

concatenating the outputs of all sub-adders, error in any sub-adder can contribute to the error in final output. Therefore, to produce the final correct output, any of the sub-adders ($2 \leq \text{SA}_i \leq n$) must not has the error conditions. Consequently, PR of an ESA can be given by:

$$PR = \prod_{i=2}^{n} PR_{SA_i} = \prod_{i=2}^{n} \Big(1 - P(G_i)P(P_i)\Big) \tag{4.3}$$

and accordingly, ER of an ESA can be given using Eqn. (4.3) as:

$$ER = 1 - \prod_{i=2}^{n} \Big(1 - P(G_i)P(P_i)\Big) \tag{4.4}$$

Next, we discuss three different cases to compute $P(G_i)$ and $P(P_i)$. Based on these cases, we have three different analytical models of ER. Results obtained from these three analytical models as well as from simulations for a 16-bit ESA as a function of k and l are shown in Fig. 4.1. Further, accuracy of these analytical models for different values of N is tabulated in Table 4.2. Note that "accuracy of an analytical model" represents the closeness of analytical results to simulations results. Therefore, it shows – how much accurately an analytical model estimates the design metric. Higher is the accuracy of an analytical model, better is the model. For a given value of l, we evaluate accuracy of the proposed and existing analytical models of an N-bit ESA as:

$$Accuracy\ (in\%) = \frac{1}{(N-2)/2} \sum_{k=2,+2}^{N-2} \left(\frac{AR_k}{SR_k}\right) \times 100 \tag{4.5}$$

where, AR and SR represent the analytical result and simulation result, respectively. For assessing the simulation results, we implement different configurations ($k = 2, 4, 6 \ldots N-2$ for $l = 0$, $k/2$ and $k-1$) of N-bit ESAs in C/C++. We simulate each configuration individually for one billion pseudo-random inputs drawn from a sample space between 0 and $2^N - 1$.

Case I: As discussed earlier, in ESAs, C_{in} of all sub-adders is considered as 0 to limit the length of carry propagation. However, C_{in} of a sub-adder can be 0 or 1 depending on the computation of the least significant bits which are not included in that sub-adder. Consequently, the probability of C_{in} of SA_i to be incorrect is $1/2$. Further, we know that a carry is propagated by i^{th} bit position only when the primary inputs A_i and B_i are either 0 and 1, or 1 and 0. Therefore, the probability of C_{in} to propagate through carry prediction bits to the output of SA_i is $1/2^l$. Substituting these values of

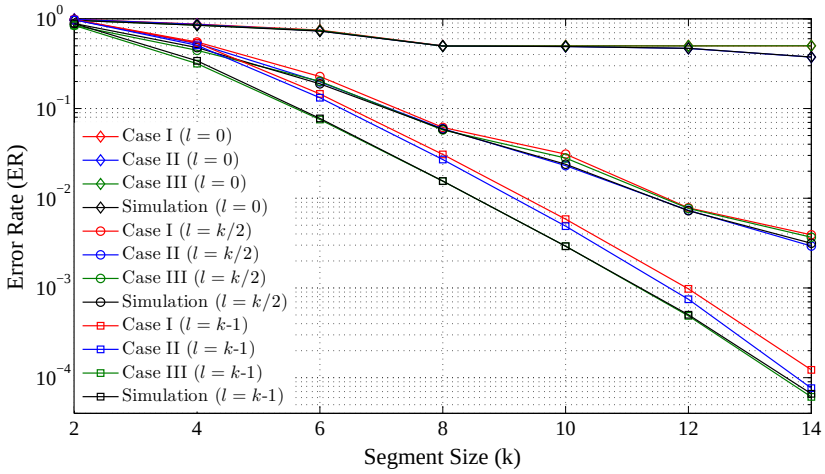


Figure 4.1: Analytical results (for three different cases) and simulation results of ER of a 16-bit ESA for different configurations, $i.e.$, different combinations of k and l.

Table 4.2: Accuracy (in %) of the proposed analytical models of ER.

ESAs		Bit-width (N)					
		8	12	16	20	24	28
$l = 0$	Case I	81.44	89.37	92.75	94.24	95.74	96.37
	Case II	96.66	98.39	99.24	99.63	99.81	99.89
	Case III	89.59	92.11	93.84	94.71	95.98	96.49
$l = k/2$	Case I	68.38	82.08	85.06	88.16	90.72	91.02
	Case II	91.82	93.79	94.06	95.81	96.56	96.88
	Case III	90.09	91.02	92.78	95.01	96.17	96.81
$l = k$–1	Case I	27.72	25.09	23.49	18.95	16.43	13.79
	Case II	69.09	58.68	50.29	44.48	41.51	35.18
	Case III	95.51	96.71	97.07	97.11	97.14	97.18

$P(G_i)$ and $P(P_i)$ in Eqn. (4.4), ER of an ESA can be given by:

$$ER = 1 - \prod_{i=2}^{n} \left(1 - \frac{1}{2} \times \frac{1}{2^l}\right) = 1 - \left(1 - \frac{1}{2} \times \frac{1}{2^l}\right)^{(n-1)} \tag{4.6}$$

Case II: It can be seen from Fig. 4.1 and Table 4.2 that for lower values of l (particularly, for $l < k/2$), Eqn. (4.6) provides a good estimation of ER. However, for $l > k/2$, accuracy of Eqn. (4.6) drops abruptly. This is partially due to the assumption that all sub-adders exhibit same $P(G)$ which is practically not true [65, 79]. From our analysis, we observe that $P(G_i)$ depends on the number of least significant bits which are not included in SA_i. Therefore, $P(G)$ of each sub-adder should be computed individually by taking into account $P(g)$ and $P(p)$ at all the least significant bits

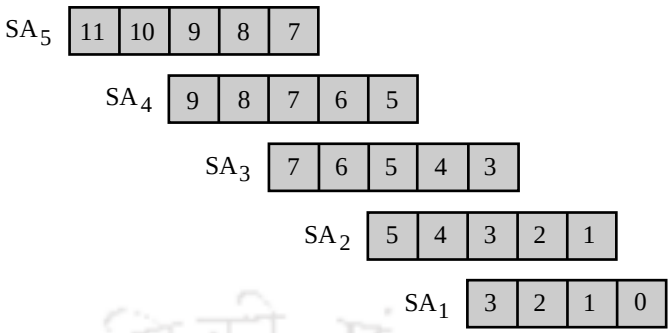


Figure 4.2: ESA(12, 5, 3): A 12-bit ESA with $k = 5$ and $l = 3$. In this case, $r = k - l = 2$, $h = N - (k - l)\lceil\frac{N-k}{k-l}\rceil = 4$ and $n = \lceil\frac{N-k}{k-l}\rceil + 1 = 5$.

which are not included in that sub-adder, where $P(g) = 1/4$ and $P(p) = 1/2$ represent the bit-wise probability of carry generation and carry propagation, respectively. Let m_i represents the number of least significant bits which are not included in SA$_i$. Now $P(G_i)$ can be given as $(\frac{1}{4}) + (\frac{1}{4} \times \frac{1}{2}) + (\frac{1}{4} \times \frac{1}{2} \times \frac{1}{2})$ ... $= \sum_{j=0}^{m_i-1} P(g)P(p)^j$. For the sake of illustration, consider an ESA(12, 5, 3), as shown in Fig. 4.2. According to Case I, $P(G_2)$, $P(G_3)$, $P(G_4)$ and $P(G_5)$ are 0.50. On the other hand, if we compute $P(G_i)$ by taking into account $P(g)$ and $P(p)$ at all the least significant bits which are not included in SA$_i$, then $P(G_2)$, $P(G_3)$, $P(G_4)$ and $P(G_5)$ are 0.25, 0.4375, 0.4844 and 0.4961, respectively. It can be seen that according to Case I, $P(G_i) = 0.50$, whereas according to Case II, $0.25 \leq P(G_i) \leq 0.50$. To be generalized, $P(G)$ of SA$_i$ can be given by:

$$P(G_i) = \sum_{j=0}^{m_i-1} \left(\frac{1}{4} \times \frac{1}{2^j}\right) = \sum_{j=0}^{h-l+r(i-2)} \left(\frac{1}{4} \times \frac{1}{2^j}\right) \tag{4.7}$$

Substituting the value of $P(G_i)$ from Eqn. (4.7) into Eqn. (4.4), ER of an ESA can be given by:

$$ER = 1 - \prod_{i=2}^{n} \left(1 - \left(\sum_{j=0}^{h-l+r(i-2)} \left(\frac{1}{4} \times \frac{1}{2^j}\right)\right) \times \frac{1}{2^l}\right) \tag{4.8}$$

Case III: As shown in Fig. 4.1 and Table 4.2, Eqn. (4.8) provides a better estimation of ER and is valid for a wider range of l as compared to Eqn. (4.6). However, for $l \gg k/2$, Eqn. (4.8) also fails. This is due to the assumption that all sub-adders are independent, *i.e.*, $P(G)$ of all sub-adders are mutually exclusive which is practically not true [78]. Therefore, the right way to estimate ER of an ESA is to compute $P(G)$ of each sub-adder individually using the approach discussed in Case II and then subtract the joint probability terms [78]. Although this approach estimates ER exactly, but

its complexity increases exponentially with the number of sub-adders. This is because the number of probability terms (which is given by $\sum_{i=1}^{n-1} \binom{n-1}{i}$ [78]) increases rapidly with the number of sub-adders. Consequently, this approach can be used only for $n \leq 5$ [78].

The above-discussed approach estimates ER exactly, but due to its complexity, the generalized derivation of $P(G)$ is not feasible (particularly, for $n > 5$). In order to make the analysis simple while maintaining accuracy of the model at the same time, we compute $P(G)$ empirically. In this regard, we exhaustively simulate an N-bit ESA for all possible configurations for $N = 1$ to 12. We then derive $P(G)$ in terms of primary design parameters k and l using the empirical approach. As mentioned in Chapter 2, empirical approaches are generally based on the information received by the means of observation or experience of patterns and behavior through experimentation. According to our experimental observations, $P(G)$ of SA$_i$ can be given by:

$$P(G_i) = \frac{2^{k-l} - 1}{2^{k-l+1}} = \frac{2^r - 1}{2^{r+1}} \tag{4.9}$$

Substituting the value of $P(G_i)$ from Eqn. (4.9) into Eqn. (4.4), ER of an ESA can be given by:

$$ER = 1 - \prod_{i=2}^{n} \left(1 - \frac{2^r - 1}{2^{r+1}} \times \frac{1}{2^l}\right) \tag{4.10}$$

Summary: It can be seen from Fig. 4.1 and Table 4.2 that for lower values of l (particularly, for $l \leq k/2$), Eqn. (4.8) provides a better estimation of ER as compared to Eqn. (4.6) and Eqn. (4.10). Consequently, for $l \leq k/2$, Eqn. (4.8) is preferred for ER estimation. However, Eqn. (4.8) fails for higher values of l, and thus, it can not be used as a generalized model. In case of ESAs, for an analytical model to be generalized, it should work for all possible configurations of an N-bit ESA. Among all the above-discussed three cases, only Eqn. (4.10) provides an overall good estimation of ER for all possible configurations of an N-bit ESA. Therefore, Eqn. (4.10) can be used as a generalized model. In the rest of this Chapter, Eqn. (4.10) is referred as the proposed analytical model of ER and the same is used for ER estimation.

Results of ER obtained from the proposed and existing analytical models as well as from simulations for a 16-bit ESA as a function of k and l are shown in Fig. 4.3. Further, accuracy of the proposed and existing analytical models of ER for different values of N is tabulated in Table 4.3. It

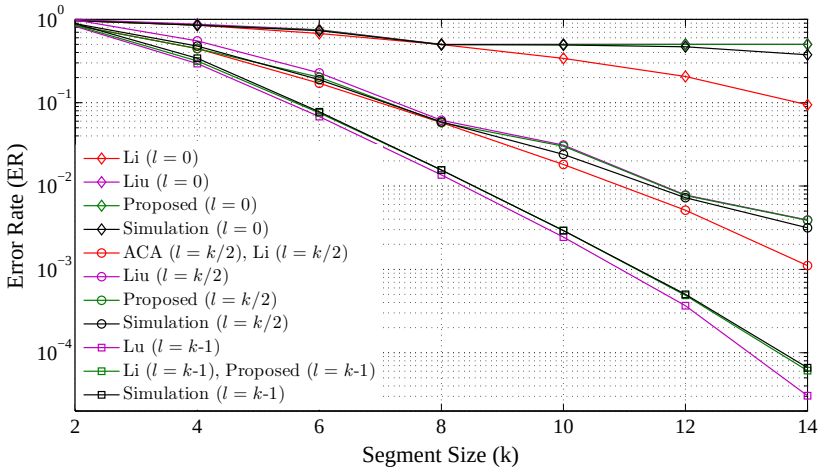


Figure 4.3: Analytical results (for the proposed and existing work) and simulation results of ER of a 16-bit ESA for different configurations, *i.e.*, different combinations of k and l.

Table 4.3: Comparison of the proposed analytical model of ER with existing work.

ESAs		Bit-width (N)					
		8	12	16	20	24	28
$l = 0$	Li [76]	84.63	79.33	75.84	73.64	73.39	72.04
	Liu [77]	87.44	89.37	92.75	94.24	95.74	96.37
	Proposed	89.59	92.11	93.84	94.71	95.98	96.49
$l = k/2$	ACA [65]	88.46	87.22	80.58	79.82	78.43	77.35
	Li [76]	88.46	87.22	80.58	79.82	78.43	77.35
	Liu [77]	68.38	80.17	87.08	93.16	94.02	94.72
	Proposed	90.09	91.02	92.78	95.01	96.17	96.81
$l = k$–1	Lu [63]	67.86	75.65	80.42	83.36	84.68	87.34
	Li [76]	95.51	96.71	97.07	97.11	97.14	97.18
	Proposed	95.51	96.71	97.07	97.11	97.14	97.18

can be seen from Fig. 4.3 and Table 4.3 that the proposed analytical model of ER is superior (or at par) to the existing analytical models. Here as well as in the rest of this Chapter, superior signifies that the proposed analytical models estimate the design metrics more accurately as compared to the existing analytical models. Further, the analytical models of ER discussed in [63, 65, 76, 77] are for specific configurations, whereas the proposed analytical model is generalized.

4.1.1.2 ED Estimation

In ESAs, the least significant sub-adder always provides correct output. Therefore, ED of the least significant sub-adder remains 0. On the other hand, any other sub-adder can provide correct

or incorrect output. If any other sub-adder ($2 \leq \text{SA}_i \leq n$) provides correct output, then its ED is 0, otherwise its ED is equal to the weight of C_{out} from the previous sub-adder [85]. Accordingly, ED of SA_i can be given by $T_i \times 2^{m_i+l}$, where $T_i = 0$, if SA_i provides correct output, otherwise $T_i = 1$. Now, the overall ED of an ESA can be given by summing all individual EDs as:

$$ED = \sum_{i=2}^{n} T_i \times 2^{m_i+l} = \sum_{i=2}^{n} T_i \times 2^{h+r(i-2)} \tag{4.11}$$

Since T_i depends on input data, estimation of ED without the prior knowledge of input data is not feasible. This is the only reason that no analytical model has been proposed till now for ED estimation. However, in respect of ED, researchers/designers propose analytical models to estimate other variants of ED, such as possible EDs [78, 79] and ED_{max} [83].

Possible EDs: We know that in an ESA, error can occur in any sub-adder (except the least significant sub-adder) individually as well as in multiple (let's say, x) sub-adders simultaneously, where $2 \leq x \leq n$. Error can occur in x sub-adders simultaneously only if there is no overlapping between them [79, 85]. For the sake of illustration, consider an ESA(12, 5, 3), as shown in Fig. 4.2. As mentioned earlier, ED of SA_i is equal to the weight of C_{out} from the previous sub-adder. Therefore, in case of ESA shown in Fig. 4.2, possible EDs are 2^4, 2^6, 2^8 and 2^{10} when error occurs in SA_2, SA_3, SA_4 and SA_5 individually, respectively. Other possible EDs are when error occurs in x sub-adders simultaneously, where $2 \leq x \leq 5$ in this case. It can be seen from Fig. 4.2 that only SA_2 and SA_5 are non-overlapping. Therefore, another possible ED is $2^4 + 2^{10}$ when error occurs in SA_2 and SA_5 simultaneously. Besides these, no other ED is possible in case of ESA(12, 5, 3). To be generalized, possible EDs when error occurs in sub-adders individually can be given by:

$$EDs = 2^{h+r(i-2)} \tag{4.12}$$

where, $2 \leq i \leq n$. Further, other possible EDs are when error occurs in x sub-adders simultaneously. These can be given by summing the EDs of sub-adders which do not overlapping with each other as:

$$EDs = \sum_{i=1}^{x} ED_i \tag{4.13}$$

In order to realize the non-overlapping criteria, we exhaustively simulate an N-bit ESA for all

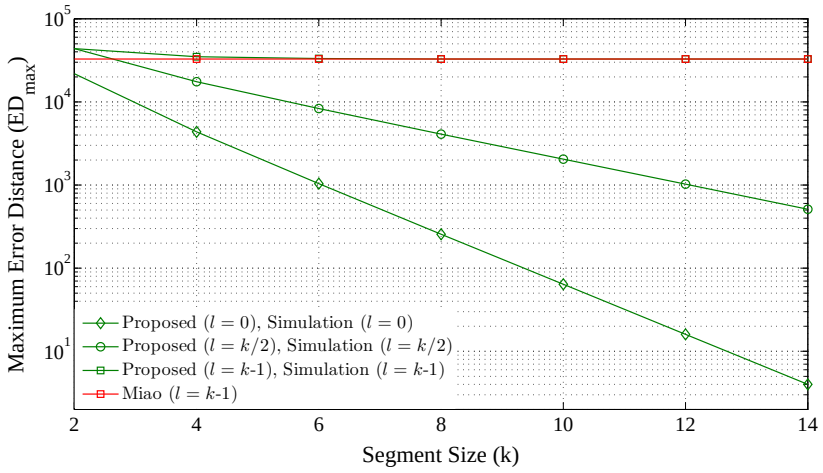


Figure 4.4: Analytical results (for the proposed and existing work) and simulation results of ED_{max} of a 16-bit ESA for different configurations, $i.e.$, different combinations of k and l.

possible configurations for $N = 1$ to 12. According to our experimental observations, x sub-adders do not overlap with each other only if they are at a distance $\geq d$ from each other, where $d = \lceil l/r \rceil + 1$. The distance between two sub-adders SA_i and SA_j can be computed as $|i - j|$, where i and $j \geq 2$. For example, consider an ESA(12, 5, 3), as shown in Fig. 4.2. In this case, $d = 3$. It can be seen from Fig. 4.2 that only SA_2 and SA_5 are at a distance $\geq d$ from each other. Consequently, SA_2 and SA_5 are non-overlapping, and thus, error can occur simultaneously in these two sub-adders only.

Maximum Possible ED: We know that ED_{max} of an ESA is dominated by the ED of the most significant sub-adder. Therefore, ED_{max} of an ESA can be given by summing the EDs of all sub-adders (starting from SA_n toward SA_2) which are at a distance equal to d from each other as:

$$\begin{aligned} ED_{max} &= ED_{SA_n} + ED_{SA_{n-d}} + ED_{SA_{n-2d}} + \ \ ... \\ &= \sum_{i=n,\ -d}^{2} 2^{h+r(i-2)} \end{aligned} \tag{4.14}$$

Results of ED_{max} obtained from the proposed and existing analytical models as well as from simulations for a 16-bit ESA as a function of k and l are shown in Fig. 4.4. From our analysis, we observe that Eqn. (4.14) estimates ED_{max} exactly for any value of N. This implies that accuracy of the proposed analytical model of ED_{max} is 100%. On the other hand, accuracy of the analytical model discussed in [83] is 92.06%, 93.84%, 95.32%, 96.31%, 96.97% and 97.44% for $N = 8$, 12, 16,

20, 24 and 28, respectively. Moreover, the analytical model of ED_{max} discussed in [83] is for specific configurations, whereas the proposed analytical model is generalized.

4.1.1.3 MED and MSE Estimation

In addition to the ED and ER, other most widely used quality metrics are MED and MSE. The two major reasons for selecting these quality metrics are: (i) Since MED encompasses the information of both ED and ER, it can be considered a "common design objective" while designing an approximate circuit; and (ii) Most of the multimedia applications are error-resilient and in these applications, MSE is one of the most widely used quality metrics. Further, we know that MSE is inversely proportional to PSNR, which is another consistently used quality metric in multimedia applications.

MED Estimation: As per Eqn. (1.3), MED of an ESA can be estimated exactly by summing the product of all possible EDs and their respective probability of occurrences. However, estimation of possible EDs when error occurs in multiple sub-adders simultaneously is complex. Therefore, in order to make the analysis simple, we consider the cases when error occurs in sub-adders individually and neglect the cases when error occurs in multiple sub-adders simultaneously. Considering this assumption, MED of an ESA can be given using Eqn. (4.10) and Eqn. (4.12) as:

$$MED = \sum_{i=2}^{n} 2^{h+r(i-2)} \left(\frac{2^r - 1}{2^{r+1}} \times \frac{1}{2^l} \right) \tag{4.15}$$

MSE Estimation: Likewise MED estimation, MSE of an ESA can be given using Eqn. (4.10) and Eqn. (4.12) as:

$$MSE = \sum_{i=2}^{n} (2^{h+r(i-2)})^2 \left(\frac{2^r - 1}{2^{r+1}} \times \frac{1}{2^l} \right) \tag{4.16}$$

Results of MED and MSE obtained from the proposed and existing analytical models as well as from simulations for a 16-bit ESA as a function of k and l are shown in Fig. 4.5 and Fig. 4.6. Further, for different values of N, accuracy of the proposed and existing analytical models of MED is tabulated in Table 4.4 and of MSE is tabulated in Table 4.5. It can be seen from Fig. 4.5, Fig. 4.6, Table 4.4 and Table 4.5 that the proposed analytical models of MED and MSE are superior (or at par) to the existing analytical models. Moreover, the analytical models of MED and MSE discussed in [76, 77] are for specific configurations, whereas the proposed analytical models are generalized.

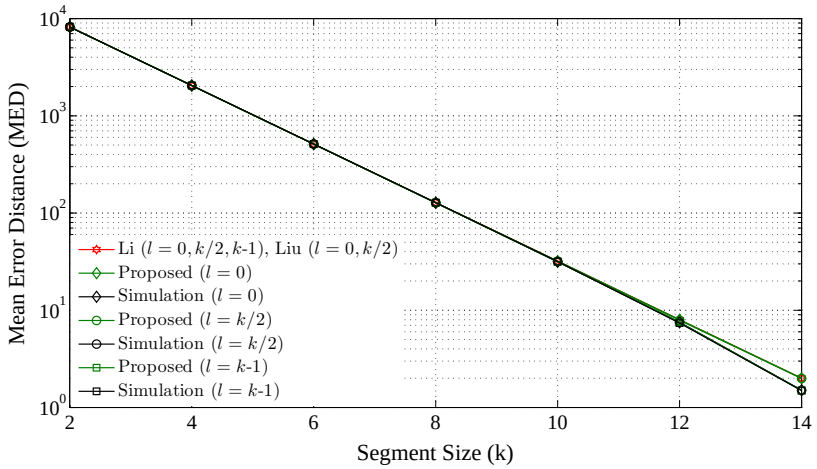


Figure 4.5: Analytical results (for the proposed and existing work) and simulation results of MED of a 16-bit ESA for different configurations, $i.e.$, different combinations of k and l.

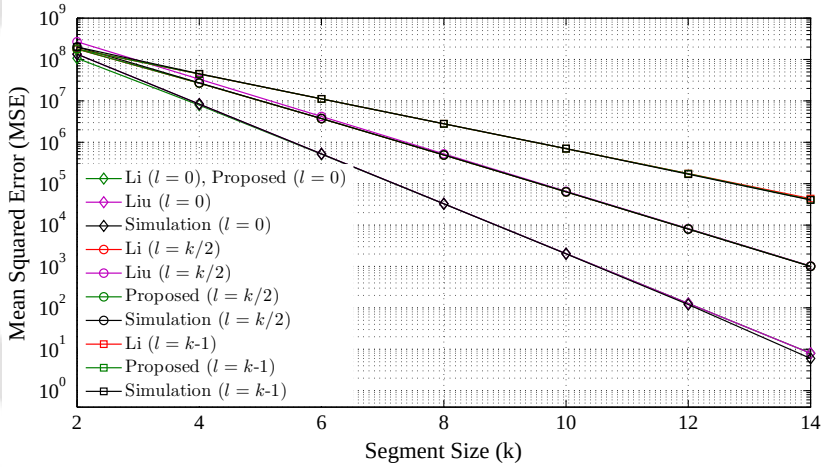


Figure 4.6: Analytical results (for the proposed and existing work) and simulation results of MSE of a 16-bit ESA for different configurations, $i.e.$, different combinations of k and l.

Table 4.4: Comparison of the proposed analytical model of MED with existing work.

ESAs		Bit-width (N)					
		8	12	16	20	24	28
$l = 0$	Li [76]	86.15	91.59	93.98	95.31	96.16	96.75
	Liu [77]	86.15	91.59	93.98	95.31	96.16	96.75
	Proposed	89.57	92.11	94.06	95.33	96.16	96.75
$l = k/2$	Li [76]	86.15	91.59	93.98	95.31	96.16	96.75
	Liu [77]	86.15	91.59	93.98	95.31	96.16	96.75
	Proposed	94.03	94.35	94.77	95.48	96.37	96.83
$l = k-1$	Li [76]	86.15	91.59	93.98	95.31	96.16	96.75
	Proposed	99.95	99.82	99.51	98.54	98.14	97.92

Table 4.5: Comparison of the proposed analytical model of MSE with existing work.

ESAs		Bit-width (N)					
		8	12	16	20	24	28
$l = 0$	Li [76]	83.18	87.04	90.17	92.26	93.64	94.61
	Liu [77]	86.15	91.59	93.98	95.31	96.16	96.75
	Proposed	83.30	87.04	90.17	92.26	93.64	94.61
$l = k/2$	Li [76]	94.75	96.91	97.76	98.08	98.56	98.63
	Liu [77]	73.80	88.75	89.19	91.42	92.55	93.39
	Proposed	95.35	97.00	97.76	98.08	98.56	98.63
$l = k-1$	Li [76]	94.30	96.31	96.55	96.87	97.40	97.46
	Proposed	96.58	97.43	97.57	97.66	97.70	97.86

4.1.2 Delay, Power and Area Estimation

Conventional adders are generally implemented using the architecture shown in Fig. 4.7. The adder architecture (shown in Fig. 4.7) consists of three parts – PG block, carry block and sum block, where PG stands for the propagate and generate. Note that PG block and sum block are identical for all adders. Only different implementations of carry block differentiate between various adders. Different implementations of carry block using the gray and black cells (shown in Fig. 4.8) can be found in any standard text on VLSI design [6] or computer arithmetic [94]. Based on the implementation of carry block, different adders have different delay, power and area. ESAs can be implemented using any of these adder architectures. Here, it should be noted that accuracy ($e.g.$, ED, ER, MED and MSE) of an ESA does not depend on the architecture used to implement it, however, its delay, power and area depend significantly. Therefore, the choice of a particular architecture depends only on delay, power and area requirements of the applications. As mentioned in Chapter 1, in order to cover a wide range of adders, we consider three types of architecture in our analysis.

4.1.2.1 Estimation Method

As discussed in Section 4.1, there are different levels of design abstraction in digital circuit design. At each level of design abstraction, several delay, power and area estimation methods have been proposed in the literature [6]. For complex circuits ($e.g.$, KSA), the most commonly used method is to measure delay in terms of *gate level depth* and area in terms of *gate count*. In this method, the circuit is first modeled in terms of 2-input basic gates. This means the circuit is seen as it is

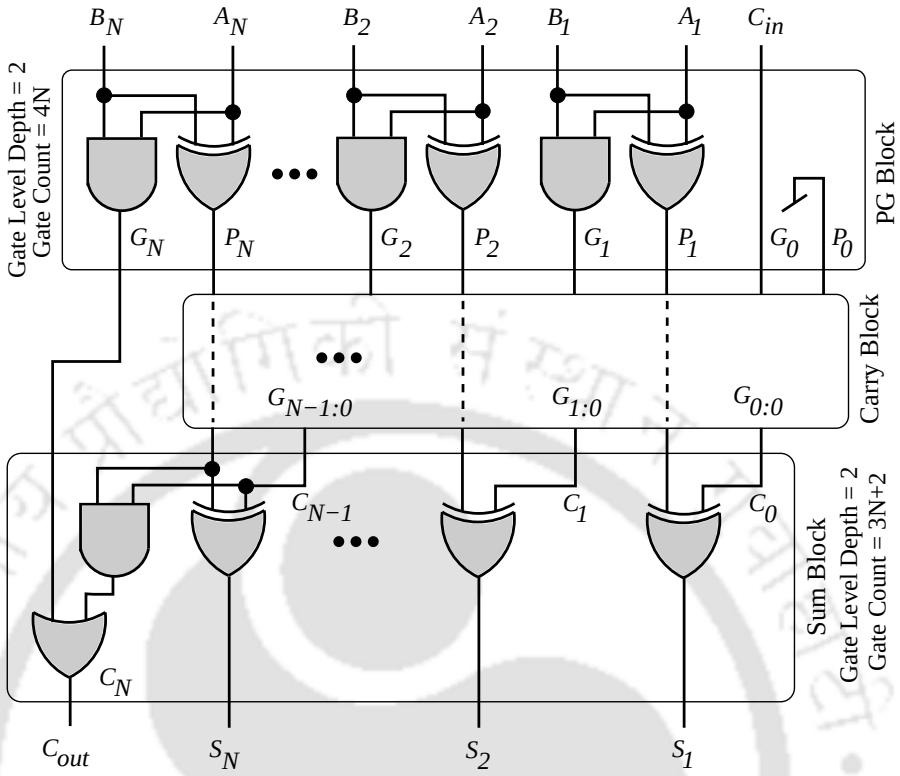


Figure 4.7: Generalized adder architecture of conventional adders [6].

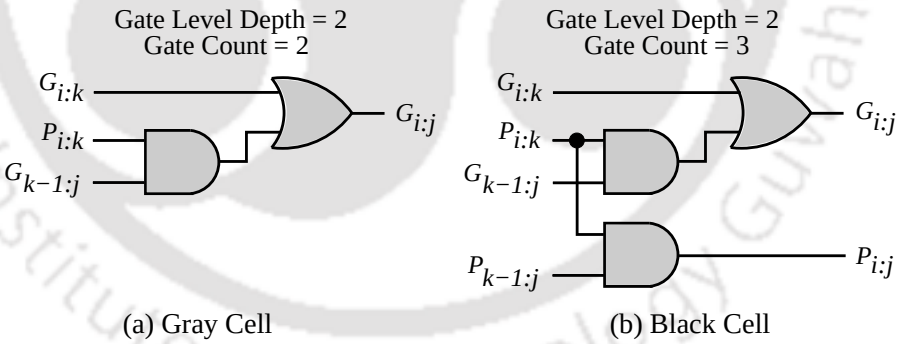


Figure 4.8: Gate level implementations: (a) Gray cell; and (b) Black cell.

constructed from 2-input AND, OR, NAND and/or NOR gates only. Note that NOT gates of the circuit are neglected in this method. Further, for the gates of other types or the gates having more than two inputs, they are expressed using their equivalent 2-input basic gates. For example, a 2-input XOR gate can be seen as it is constructed from 2 2-input AND gates and one 2-input OR gate. According to this method, gate level depth and gate count of PG block and sum block are shown in Fig. 4.7 and of gray cell and black cell are shown in Fig. 4.8. Further, the number of logic levels, gray cells and black cells required in the implementation of carry block in RCA, CIA and KSA architectures are

Table 4.6: The number of logic levels, gray cells and black cells required in the implementation of carry block in RCA, CIA and KSA architectures.

Adder Architectures	Number of Logic Levels	Number of Gray Cells	Number of Black Cells
RCA	$N-1$	$N-1$	0
CIA	$N/4+2$	$N-1$	$N-5$ [#]
KSA	$\lceil log_2 N \rceil$	$N-1$	$\lceil N(log_2 N-2) \rceil$

[#] for $N \leq 4$, the number of black cells required is 0.

tabulated in Table 4.6. In further discussion on delay, power and area estimation of ESAs, we use the data mentioned in Fig. 4.7, Fig. 4.8 and Table 4.6.

4.1.2.2 Delay Estimation

As mentioned earlier, PG block and sum block are identical for all adder architectures. Therefore, irrespective of the adder architecture, PG block and sum block each have 2 gate level depth. Unlike PG block and sum block, gate level depth of carry block depends on the adder architecture. In case of RCA architecture, carry block has $(N-1)$ logic levels, where each logic level consists of a gray cell. A gray cell has 2 gate level depth. Consequently, RCA has $2+2(N-1)+2=2(N+1)$ gate level depth. On the other hand, in case of ESAs, all sub-adders operate in parallel. Therefore, gate level depth of an ESA is equal to the gate level depth of a k-bit sub-adder. The gate level depth of a k-bit sub-adder implemented using RCA architecture is $2(k+1)$. Accordingly, delay (D) of an ESA implemented using RCA architecture can be given by:

$$D = 2C_d(k+1) \tag{4.17}$$

where, C_d is a technology dependent constant for delay. Similarly, we derive analytical models to estimate delay of an ESA implemented using CIA and KSA architectures. The proposed analytical models are tabulated in Table 4.7.

4.1.2.3 Area Estimation

Irrespective of the adder architecture, PG block has $4N$ gate count and sum block has $3N+2$ gate count. Unlike PG block and sum block, gate count of carry block depends on the adder architecture.

Table 4.7: The proposed analytical models to estimate delay and area of ESAs implemented using RCA, CIA and KSA architectures.

Adder Architectures	Delay of ESAs	Area of ESAs
RCA	$2C_d(k+1)$	$C_a(7N+2h+2(k-1)\lceil\frac{N-k}{k-l}\rceil)$
CIA	$C_d(k/2+8)$	$C_a(7N+5h-15+(5k-17)\lceil\frac{N-k}{k-l}\rceil)^{\#}$
KSA	$2C_d(\lceil log_2 k\rceil+2)$	$C_a(7N+2h+3(\lceil h(log_2 h-2)\rceil)+(2(k-1)+3(\lceil k(log_2 k-2)\rceil))\lceil\frac{N-k}{k-l}\rceil)$

$^{\#}$for $k \leq 4$, area of an ESA implemented using CIA architecture is $7N+2h+2(k-1)\lceil\frac{N-k}{k-l}\rceil$.

In case of RCA architecture, carry block consists of $(N-1)$ gray cells, where each gray cell has 2 gate count. Consequently, RCA has $7N+2+2(N-1)=9N$ gate count. On the other hand, an ESA has $\lceil\frac{N-k}{k-l}\rceil+1$ sub-adders. As shown in Fig. 1.6, the lower order l bits of a sub-adder (except the least significant sub-adder) are used to generate carry signals only. Therefore, an ESA can be considered as it consists of a PG block, a sum block and $\lceil\frac{N-k}{k-l}\rceil+1$ carry blocks, where the least significant carry block is of h-bit and the remaining $\lceil\frac{N-k}{k-l}\rceil$ carry blocks are of k-bit. The h-bit carry block and k-bit carry block implemented using RCA architecture consist of $(h-1)$ gray cells and $(k-1)$ gray cells, respectively. Accordingly, area (A) of an ESA implemented using RCA architecture can be given by:

$$A = C_a\left(7N+2h+2(k-1)\left\lceil\frac{N-k}{k-l}\right\rceil\right) \tag{4.18}$$

where, C_a is a technology dependent constant for area. Similarly, we derive analytical models to estimate area of an ESA implemented using CIA and KSA architectures. The proposed analytical models are tabulated in Table 4.7.

4.1.2.4 Power Estimation

The total power consumption of a digital circuit mainly consists of two components: (i) Static power consumption; and (ii) Dynamic power consumption. We know that the dynamic power consumption is given by $\alpha C_{out} V_{DD}^2 f_{clk}$, where α, C_{out}, V_{DD} and f_{clk} are the switching activity factor, output capacitance, supply voltage and clock frequency, respectively. At a fixed clock frequency, the dynamic power consumption with voltage scaling is proportional to the capacitance and V_{DD}^2. Further, capacitance is proportional to area and cell delay is proportional to $1/(V_{DD}-V_{th})^{\beta}$, where V_{th} is the threshold voltage and β is the velocity saturation index. Considering $\beta=2$ and $V_{th} \ll V_{DD}$, V_{DD}^2 is

roughly proportional to 1/(cell delay). Since the product of cell delay and path depth is constant at a fixed clock frequency, cell delay is proportional to 1/(path depth). Consequently, V_{DD}^2 is proportional to path depth. In summary, at a fixed clock frequency, the dynamic power consumption with voltage scaling of a digital circuit is roughly proportional to its area and path depth [65]. In our analysis, we measure area in terms of gate count and path depth in terms of gate level depth. Consequently, at a fixed clock frequency, the dynamic power consumption (P_d) with voltage scaling of an ESA implemented using RCA architecture can be given using Eqn. (4.17) and Eqn. (4.18) as:

$$P_d = 2C_{p_d}\left((k+1)\left(7N + 2h + 2(k-1)\left\lceil\frac{N-k}{k-l}\right\rceil\right)\right) \tag{4.19}$$

where, C_{p_d} is a technology dependent constant for dynamic power consumption. Further, we know that the static power consumption of a digital circuit is proportional to its area [65]. As mentioned above, in our analysis, we measure area in terms of gate count. Consequently, the static power consumption (P_s) of an ESA implemented using RCA architecture can be given using Eqn. (4.18) as:

$$P_s = C_{p_s}\left(7N + 2h + 2(k-1)\left\lceil\frac{N-k}{k-l}\right\rceil\right) \tag{4.20}$$

where, C_{p_s} is a technology dependent constant for static power consumption. Now the total power consumption (P) of an ESA implemented using RCA architecture can be given by the sum of Eqn. (4.19) and Eqn. (4.20). Similarly, using Table 4.7, we derive analytical models to estimate static, dynamic and total power consumptions of an ESA implemented using CIA and KSA architectures.

4.1.2.5 Model Validation

In Section 4.1.1, we validate the proposed analytical models of accuracy (ED_{max}, ER, MED and MSE) by comparing them with simulation results as well as with existing work. However, as mentioned earlier, no favorable work has been proposed till now on analytical modeling of delay, power and area of ESAs. Therefore, in order to validate the proposed analytical models of delay, power and area, we compare them with simulation results only. For assessing the simulation results of delay, power and area, we design different configurations (k = 2, 4, 6 ... $N-2$ for $l = 0$, $k/2$ and $k-1$) of N-bit ESAs using *Mentor Graphics Tanner schematic capture* [103]. We then extract netlists from

schematics and simulate the extracted netlists using *Synopsys HSPICE circuit simulator* [104] with PTM $32nm$ model files [105]. While designing the schematics, we size all the transistors using the method of *logical effort* [15]. According to logical effort, transistors of a logic gate are sized to achieve the unit resistance, $i.e.$, pullup transistors and pulldown transistors should have the same conductance. Further, in order to make the analysis simple, we neglect delay, power and area imposed by interconnects. Here, it should be noted that from simulations, we have delay, power and area in terms of ps, μW and μm^2, respectively, whereas from analytical models, we have delay, power and area in terms of C_d, C_p and C_a, respectively, where $C_p = C_{p_d} + C_{p_s}$ is a technology dependent constant for total power consumption. Note that these constants vary with foundry, process, technology node, size of logic gates, etc. Therefore, knowing the actual value of these constants (prior fabrication) is not feasible. In such a scenario, researchers/designers determine these constants approximately using different methods. We know that 2-input NAND gate forms the basis for most practical digital circuits. Therefore, the simplest method to determine these constants is to consider delay of a 2-input NAND gate as C_d, power consumption of a 2-input NAND gate as C_p and area of a 2-input NAND gate as C_a. In our analysis, we also determine C_d, C_p and C_a using the same method. In this regard, we design a 2-input NAND gate according to the method of *logical effort* using *Mentor Graphics Tanner schematic capture*. We then extract netlist from schematic and simulate the extracted netlist using *Synopsys HSPICE circuit simulator* with PTM $32nm$ model files. While simulating the netlist, we assume that the NAND gate derives a load capacitance equal to the input capacitance of a unit standard CMOS inverter. Further, while evaluating the area, we neglect area imposed by interconnects and extract only layout area of the NAND gate from HSPICE netlist. As per our simulation results, delay, power and area of the NAND gate are $12.14ps$, $9.24\mu W$ and $0.70\mu m^2$, respectively. Consequently, $C_d = 12.14ps$, $C_p = 9.24\mu W$ and $C_a = 0.70\mu m^2$. For the sake of illustration, consider an ESA(8, 2, 0) implemented using RCA architecture. From the proposed analytical models, we have delay, power and area of the above-mentioned ESA as $6C_d$, $462C_p$ and $66C_a$, respectively. Now substituting the value of C_d, C_p and C_a, we have delay as $72.84ps$, power as $4268.88\mu W$ and area as $46.20\mu m^2$. On the other hand, from simulations, we have delay, power and area as $76.84ps$, $4649.67\mu W$ and $49.96\mu m^2$, respectively. Therefore, in case of ESA$(8, 2, 0)$ implemented using RCA

Table 4.8: Accuracy (in %) of the proposed analytical models of delay, power and area.

ESAs		Delay			Power			Area		
		$N=8$	$N=16$	$N=32$	$N=8$	$N=16$	$N=32$	$N=8$	$N=16$	$N=32$
RCA	$l=0$	93.92	93.84	93.55	90.27	90.12	89.80	91.88	91.73	91.53
	$l=k/2$	93.46	93.29	93.04	89.61	89.44	89.11	91.44	91.31	91.09
	$l=k-1$	92.91	92.78	92.53	88.97	88.84	88.50	90.93	90.85	90.61
CIA	$l=0$	92.41	92.23	91.96	88.34	88.22	87.85	90.52	90.37	90.07
	$l=k/2$	91.78	91.67	91.41	87.73	87.49	87.08	89.95	89.81	89.58
	$l=k-1$	91.30	91.11	90.83	86.93	86.72	86.45	89.44	89.35	89.02
KSA	$l=0$	90.72	90.58	90.24	86.25	86.08	85.70	88.85	88.73	88.49
	$l=k/2$	90.13	90.02	89.72	85.57	85.37	84.96	88.36	88.21	87.94
	$l=k-1$	89.57	89.49	89.28	84.84	84.65	84.31	87.78	87.70	87.42

architecture, accuracy of the proposed analytical model of delay is $(72.84/76.84) \times 100 = 94.79\%$, power is $(4268.88/4649.67) \times 100 = 91.81\%$ and area is $(46.20/49.96) \times 100 = 92.47\%$. Similarly, we determine analytical results and simulation results of delay, power and area of different configurations ($k = 2, 4, 6$ ... $N-2$ for $l = 0$, $k/2$ and $k-1$) of N-bit ESAs implemented using RCA, CIA and KSA architectures. We then compute accuracy of the proposed analytical models of delay, power and area as per Eqn. (4.5). For $N = 8$, 16 and 32, accuracy of the proposed analytical models is tabulated in Table 4.8. It can be seen from Table 4.8 that analytical results of delay, power and area are in good agreement with simulation results. From our analysis, we observe that the difference between analytical results and simulation results is basically due to two main reasons: (i) In analytical models, we neglect NOT gates, whereas our simulation results are for complete circuit including NOT gates; and (ii) Since we are not aware about the actual values of C_d, C_p and C_a, we consider approximate values of these technology dependent constants.

4.1.3 Important Observations

The design metrics, such as accuracy, delay, power and area of an N-bit ESA depend on the primary design parameters k and l. However, these design metrics have unalike behavior with k and l. Based on the above-discussed analytical models and simulation results, some of our important observations regarding the behavior of these design metrics are as follows.

(i) The delay-accuracy trade-off of an ESA depends on k only. The accuracy ($e.g.$, ED, ER, MED

and MSE) of an ESA improves with k. However, as k increases, delay benefits (such as, increasing the clock frequency and reducing the supply voltage) decline.

(ii) The power-accuracy and area-accuracy trade-offs of an ESA depend on both k and l. The number of sub-adders required in the implementation of an ESA decreases with k, whereas size of the sub-adders increases with k. Due to this conflict, area of an ESA may increase/decrease with k. On the other hand, power of an ESA always increases with k. Further, for a fixed k, the number of sub-adders required in the implementation of an ESA increases with l. Therefore, for a fixed k, both power and area of an ESA increase with l.

(iii) We know that all the quality metrics, such as ED, ER, MED and MSE of an ESA depend on k and l. However, unlike in case of k, where all these quality metrics decrease with k; in case of l, these quality metrics have contradictory behavior. For a fixed k, ER of an ESA decreases with l, whereas its ED and MSE increase with l. Further, for a fixed k, MED of an ESA does not change with l. This is because with increase in l, ER of an ESA decreases, but its ED increases by the same ratio. Consequently, MED of an ESA is independent of l.

(iv) For a fixed k, an ESA with $l = 0$ provides the lowest ED and with $l = k - 1$, it provides the lowest ER. Therefore, ESAs with $l = 0$ are considered as ED-optimal and ESAs with $l = k - 1$ are considered as ER-optimal. On the other hand, ESAs with $l = k/2$ provide trade-off between ED and ER to smooth the overall error characteristics.

4.2 Optimization

In error-resilient applications, the final output need not be fully precise, rather approximate output is equally acceptable. However, for an approximate output to be acceptable, it must satisfy the accuracy requirements of the applications, where accuracy is measured in terms of quality metrics, such as ED, ER, MED and MSE. We know that in an N-bit ESA, there exist multiple (more than one) configurations which exhibit similar accuracy (see Table 1.2). However, these configurations exhibit different delay, power and area. Therefore, for a given accuracy, the configurations which provide minimal delay, power and/or area need to be known apriori for efficient, intelligent and goal oriented

implementations of ESAs. Note that finding the optimal configurations is a constrained optimization problem, $i.e.$, minimize or maximize objective functions in the presence of constraints. Based on the requirements of the applications, there can be four types of optimization: (i) Single-objective optimization subjected to single constraint; (ii) Single-objective optimization subjected to multiple constraints; (iii) Multi-objective optimization subjected to single constraint; and (iv) Multi-objective optimization subjected to multiple constraints.

4.2.1 Objective Functions and Constraints

Objective functions and constraints depend on the applications. Based on the applications, objective function can be delay, power and/or area. For example, our objective: (i) In high-performance applications is to minimize delay; (ii) In low-power applications is to minimize power; and (iii) In area-constrained applications is to minimize area. Further, constraints can be either soft or hard. Hard constraints must be satisfied at any cost, whereas soft constraints should be satisfied, but may be violated. As mentioned earlier, in approximate computing paradigm, an approximate output is acceptable only if it satisfies the accuracy requirements of the applications. Researchers/designers introduce several quality metrics to characterize the accuracy of the applications [1, 62, 65, 74, 75]. It has been observed that in case of approximate arithmetic circuits, the combination of ED, ER, MED and MSE is sufficient to evaluate the overall accuracy [76, 78]. Therefore, in our analysis, we consider ED, ER, MED and/or MSE as the hard constraints. Now based on the combinations of objective functions (delay, power and/or area) and constraints (ED, ER, MED and/or MSE), several cases of optimization are possible. Using the combination formula, $\sum_{i=1}^{3} \binom{3}{i} \times \sum_{j=1}^{4} \binom{4}{j} = 105$ cases of optimization per adder architecture are possible. Since we consider three types of adder architecture (RCA, CIA and KSA) in our analysis, overall 315 cases of optimization are possible. For the sake of demonstration, we consider 9 cases of optimization in this thesis – single-objective optimization (delay, power or area) subjected to single constraint (PR). However, the proposed approach can be customized/extended for the remaining cases of optimization also.

4.2.2 Optimization Framework

In single-objective optimization (delay, power or area) subjected to single constraint (PR), our objective is to find the optimal configurations of an N-bit ESA which provide minimal delay, power or area for a given PR. In this regard, our optimization framework is as follows.

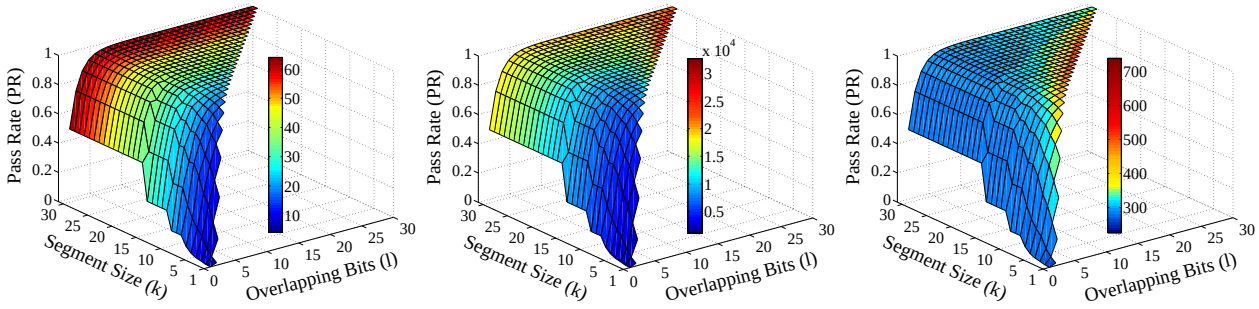
(i) Using the proposed analytical models discussed in Section 4.1, we first determine delay, power and area surfaces $w.r.t.$ PR of all $N(N-1)/2$ possible configurations of an N-bit ESA as a function of k and l. Such results for a 32-bit ESA implemented using RCA, CIA and KSA architectures are shown in Fig. 4.9, Fig. 4.10 and Fig. 4.11, respectively.

(ii) We then define search space by specifying the constraint on PR as PR $\geq$ threshold value. By doing so, we are shortlisting the qualifying configurations which satisfy the accuracy (PR) requirements of the applications. Note that in this case, the number of qualifying configurations is inversely proportional to the threshold value of PR.

(iii) Now we execute search operations within the search space (defined in point (ii)) to find the optimal configurations. Theses optimal configurations exhibit minimal delay, power and area while satisfying PR requirements of the applications at the same time.

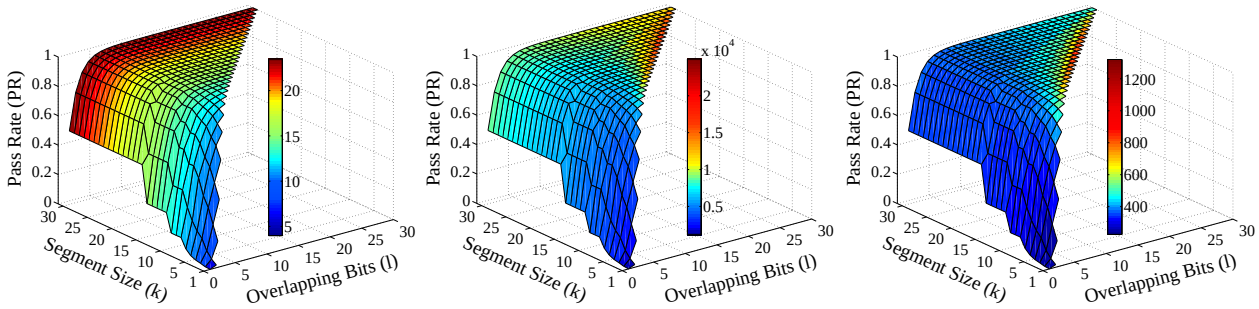
(iv) The search operations (mentioned in point (iii)) can be executed exhaustively or heuristically. Here, exhaustive search means all configurations within the search space need to be searched. On the other hand, heuristic search means only some random configurations within the search space need to be searched using heuristic algorithms.

(v) We know that an N-bit ESA has $N(N-1)/2$ configurations. Depending on the constraint on PR, some of these configurations (defined as the qualifying configurations) exist within the search space and the remaining configurations (defined as the disqualifying configurations) exist out of the search space. Let's assume that for a given constraint on PR, $x\%$ configurations qualify and exist within the search space. Now in case of exhaustive search, we need to search $\frac{N(N-1)}{2} \times \frac{x}{100}$ configurations. It can be seen that the number of configurations which need to be searched increases quadratically with N and linearly with x. For example, for $x = 50\%$,



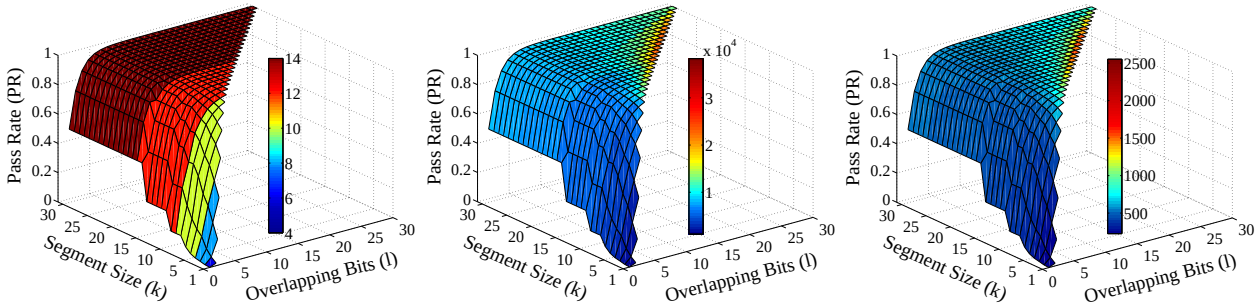
(a) (b) (c)

Figure 4.9: Delay, power and area surfaces with respect to PR of a 32-bit ESA implemented using RCA architecture as a function of k and l: (a) Delay; (b) Power; and (c) Area.



(a) (b) (c)

Figure 4.10: Delay, power and area surfaces with respect to PR of a 32-bit ESA implemented using CIA architecture as a function of k and l: (a) Delay; (b) Power; and (c) Area.



(a) (b) (c)

Figure 4.11: Delay, power and area surfaces with respect to PR of a 32-bit ESA implemented using KSA architecture as a function of k and l: (a) Delay; (b) Power; and (c) Area.

we need to search 60, 248, 1008, 4064 and 16320 configurations for $N = 16$, 32, 64, 128 and 256, respectively. As a result, for smaller values of N (particularly, for $N \leq 32$), search can be executed exhaustively. However, for higher values of N (particularly, for $N \geq 64$), exhaustive

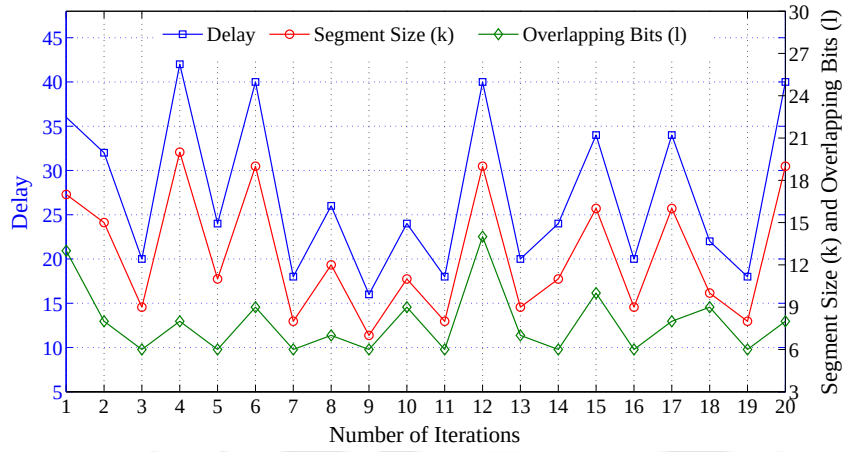
search becomes infeasible due to programming efforts, time-consuming simulations, etc. In such conditions, researchers/designers generally employ heuristic search.

4.2.3 Results and Discussion

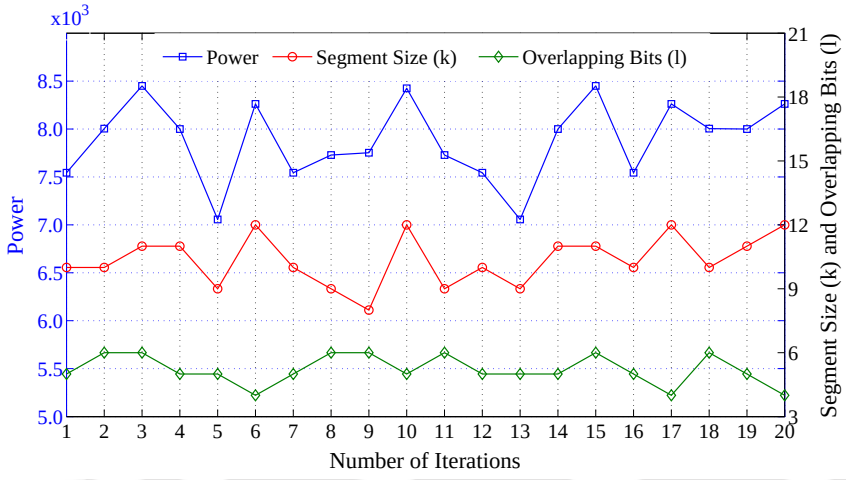
The above-discussed optimization steps can be collectively defined as the constrained optimization. Over the decades, several nature-inspired heuristic algorithms have been proposed in the literature for constrained optimization [92]. For the sake of demonstration, we use RFD heuristic algorithm. RFD heuristic algorithm is based on the formation of rivers by water drops. As compared to other heuristic algorithms, RFD requires longer time to converge to a solution, but it provides a better quality of solutions. The detailed theory and working of RFD heuristic algorithm can be found in [112]. Note that heuristic algorithms do not guarantee for the best optimal solution every time. Therefore, repeated simulations need to be executed to find the best optimal solution.

Our search results of minimal delay, power and area subjected to PR ≥ 0.90 for a 32-bit ESA implemented using RCA architecture for different iterations are shown in Fig. 4.12. The corresponding optimal configurations, $i.e.$, values of k and l are also shown in Fig. 4.12. As mentioned earlier, exhaustive search can be used for $N \leq 32$, however, we use heuristic algorithm even for a 32-bit ESA to demonstrate its effectiveness. It can be seen from Fig. 4.12 that in this case, minimal delay is $16C_d$ at $k = 7$ and $l = 6$, minimal power is $7056C_p$ at $k = 9$ and $l = 5$, and minimal area is $292C_a$ at $k = 18$ and $l = 3$. Similarly, we search for the optimal configurations subjected to different constraint on PR. Our search results for a 32-bit ESA implemented using RCA, CIA and KSA architectures are tabulated in Table 4.9, Table 4.10 and Table 4.11, respectively. Note that these search results are in terms of technology dependent constants C_d, C_p and C_a. As discussed in Section 4.1.2.5, for a particular technology, the approximate values of these constants can be determined by designing and simulating a 2-input NAND gate. Accordingly, the real values of minimal delay, power and area can be assessed by multiplying with C_d, C_p and C_a, respectively.

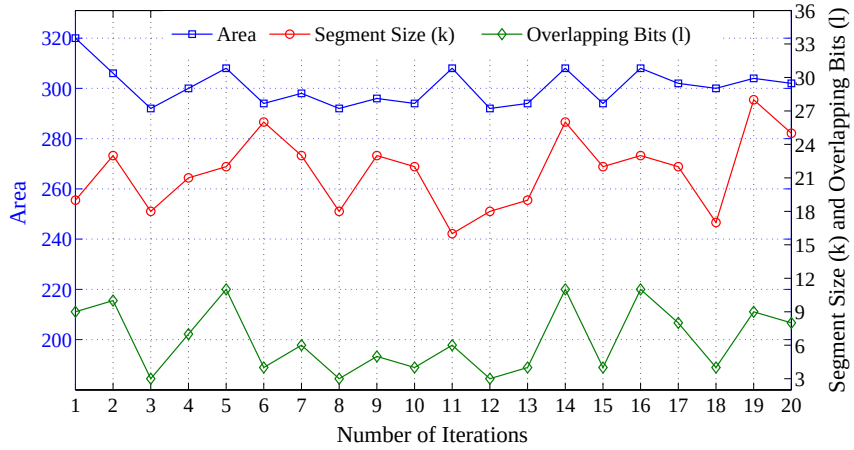
In a similar fashion, using the proposed optimization framework, designers can find the optimal configurations of an N-bit ESA which provide minimal delay, power and/or area for a given ED, ER, MED and/or MSE. This enables designers for efficient, intelligent and goal oriented implementations



(a)



(b)



(c)

Figure 4.12: Search results for minimal delay, power and area subjected to PR ≥ 0.90 of a 32-bit ESA implemented using RCA architecture for different iterations: (a) Delay; (b) Power; and (c) Area. Here, left y-axis shows minimal delay, power and area, and right y-axis shows the corresponding optimal configurations.

Table 4.9: Minimal delay, power and area (and corresponding optimal configurations) with different constraint on PR of a 32-bit ESA implemented using RCA architecture.

Constraints	$D_{min}\ (k,\ l)$	$P_{min}\ (k,\ l)$	$A_{min}\ (k,\ l)$
PR ≥ 0.90	$16C_d$ (7, 6)	$7056C_p$ (9, 5)	$292C_a$ (18, 3)
PR ≥ 0.91	$18C_d$ (8, 6)	$7056C_p$ (9, 5)	$292C_a$ (18, 3)
PR ≥ 0.92	$18C_d$ (8, 6)	$7544C_p$ (10, 5)	$292C_a$ (18, 3)
PR ≥ 0.93	$18C_d$ (8, 6)	$7728C_p$ (9, 6)	$292C_a$ (18, 3)
PR ≥ 0.94	$18C_d$ (8, 6)	$7728C_p$ (9, 6)	$294C_a$ (18, 4)
PR ≥ 0.95	$18C_d$ (8, 6)	$8004C_p$ (10, 6)	$294C_a$ (18, 4)
PR ≥ 0.96	$20C_d$ (9, 7)	$8450C_p$ (11, 6)	$294C_a$ (18, 4)
PR ≥ 0.97	$20C_d$ (9, 7)	$8832C_p$ (10, 7)	$296C_a$ (19, 5)
PR ≥ 0.98	$22C_d$ (10, 8)	$9072C_p$ (12, 7)	$296C_a$ (19, 5)
PR ≥ 0.99	$24C_d$ (11, 9)	$9666C_p$ (12, 8)	$298C_a$ (19, 6)

Table 4.10: Minimal delay, power and area (and corresponding optimal configurations) with different constraint on PR of a 32-bit ESA implemented using CIA architecture.

Constraints	$D_{min}\ (k,\ l)$	$P_{min}\ (k,\ l)$	$A_{min}\ (k,\ l)$
PR ≥ 0.90	$11.5C_d$ (7, 6)	$5481C_p$ (11, 4)	$367C_a$ (18, 3)
PR ≥ 0.91	$12C_d$ (8, 6)	$5629.5C_p$ (9, 5)	$367C_a$ (18, 3)
PR ≥ 0.92	$12C_d$ (8, 6)	$5726C_p$ (10, 5)	$367C_a$ (18, 3)
PR ≥ 0.93	$12C_d$ (8, 6)	$5814.5C_p$ (11, 5)	$367C_a$ (18, 3)
PR ≥ 0.94	$12C_d$ (8, 6)	$5895C_p$ (12, 5)	$372C_a$ (18, 4)
PR ≥ 0.95	$12C_d$ (8, 7)	$5895C_p$ (12, 5)	$372C_a$ (18, 4)
PR ≥ 0.96	$12.5C_d$ (9, 7)	$6160C_p$ (14, 5)	$372C_a$ (18, 4)
PR ≥ 0.97	$12.5C_d$ (9, 8)	$6324C_p$ (13, 6)	$377C_a$ (19, 5)
PR ≥ 0.98	$13C_d$ (10, 8)	$6517.5C_p$ (15, 6)	$377C_a$ (19, 5)
PR ≥ 0.99	$13.5C_d$ (11, 9)	$6885C_p$ (16, 7)	$382C_a$ (19, 6)

Table 4.11: Minimal delay, power and area (and corresponding optimal configurations) with different constraint on PR of a 32-bit ESA implemented using KSA architecture.

Constraints	$D_{min}\ (k,\ l)$	$P_{min}\ (k,\ l)$	$A_{min}\ (k,\ l)$
PR ≥ 0.90	$10C_d$ (7, 6)	$6630C_p$ (11, 4)	$510C_a$ (11, 4)
PR ≥ 0.91	$10C_d$ (8, 6)	$6708C_p$ (14, 4)	$516C_a$ (14, 4)
PR ≥ 0.92	$10C_d$ (8, 6)	$6708C_p$ (14, 4)	$516C_a$ (14, 4)
PR ≥ 0.93	$10C_d$ (8, 6)	$6708C_p$ (14, 4)	$516C_a$ (14, 4)
PR ≥ 0.94	$10C_d$ (8, 7)	$6994C_p$ (14, 5)	$531C_a$ (19, 4)
PR ≥ 0.95	$10C_d$ (8, 7)	$6994C_p$ (14, 5)	$531C_a$ (19, 4)
PR ≥ 0.96	$12C_d$ (9, 7)	$6994C_p$ (14, 5)	$531C_a$ (19, 4)
PR ≥ 0.97	$12C_d$ (9, 8)	$7280C_p$ (15, 6)	$545C_a$ (19, 5)
PR ≥ 0.98	$12C_d$ (10, 8)	$7280C_p$ (15, 6)	$545C_a$ (19, 5)
PR ≥ 0.99	$12C_d$ (11, 9)	$7566C_p$ (16, 7)	$556C_a$ (19, 6)

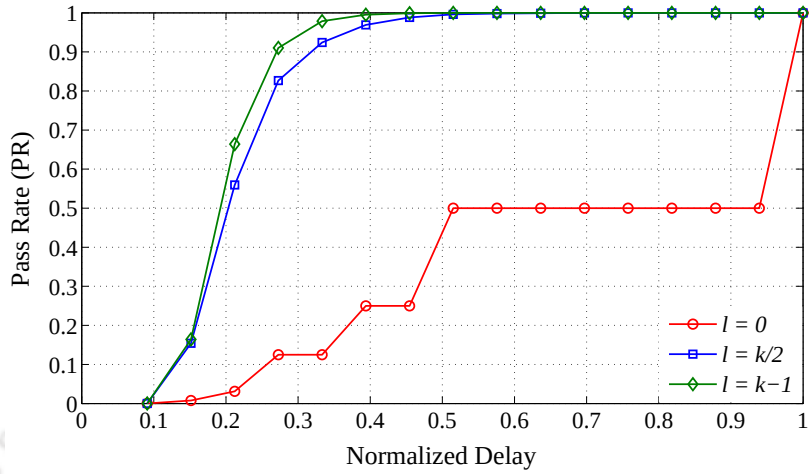
of ESAs. Further, based on Table 4.9, Table 4.10 and Table 4.11, some of our important observations regarding the optimization of ESAs are as follows.

(i) The search results tabulated in Table 4.9, Table 4.10 and Table 4.11 also provide insights into 32-bit ESA implemented using the adder architectures which exhibit delay, power and area similar to RCA, CIA and KSA architectures, respectively. For example, *Sklansky Adder* (SA) and *Knowles Adder* (KA) architectures have similar delay as the KSA architecture, and *Brent-Kung Adder* (BKA) architecture has similar area as the CIA architecture [6]. Therefore, search results of minimal delay tabulated in Table 4.11 are moderately valid for the ESA implemented using SA and KA architectures, and search results of minimal area tabulated in Table 4.10 are moderately valid for the ESA implemented using BKA architecture.

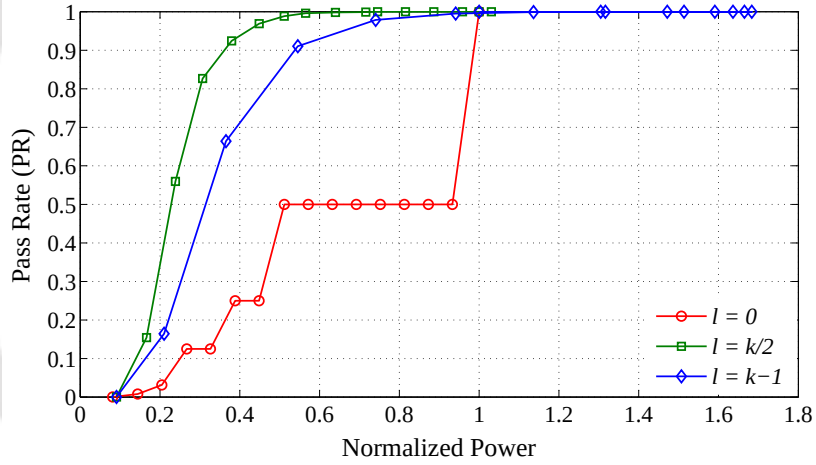
(ii) It can be seen from Table 4.9, Table 4.10 and Table 4.11 that for a given constraint on PR: (a) ESA designed using KSA architecture provides minimum delay; (b) ESA designed using CIA architecture provides minimum power; and (c) ESA designed using RCA architecture provides minimum area. Consequently, it is preferred to design ESAs using KSA, CIA and RCA architecture for high-performance, low-power and area-constrained applications, respectively.

4.3 Accuracy Enhancement

As mentioned earlier, in approximate computing paradigm (where the circuits provide approximate results), accuracy is also a concern in addition to the conventional design metrics, such as delay, power and area. In such a scenario, researchers/designers use accuracy-effort curves for evaluating the effectiveness of approximate circuits. Note that accuracy-effort curves describe the relationship between the accuracy we can achieve and the efforts we need to pay in return. In case of approximate arithmetic circuits, accuracy is measured in terms of quality metrics (ED, ER, MED and MSE) and efforts are measured in terms of delay, power and area. Therefore, accuracy-effort curves evaluate – how much optimal delay-accuracy, power-accuracy and area-accuracy trade-offs an approximate arithmetic circuit can provide. Fig. 4.13 shows the accuracy-effort curves of 32-bit ESA as a function of k and l, where accuracy is measured in terms of PR and effort is measured in terms of delay in



(a)



(b)

Figure 4.13: Accuracy-effort curves of 32-bit ESA (implemented using RCA architecture) as a function of k and l: (a) PR versus delay; and (b) PR versus power. Here, the markers/points in curves correspond to the values of k as 2, 4, 6 ... 32 from left to right.

Fig. 4.13(a) and in terms of power in Fig. 4.13(b). It can be seen from Fig. 4.13 that there is a scope of improvement in accuracy-effort curves of ESAs. The accuracy-effort curves of ESAs need to be improved so that their accuracy degrade in a graceful manner, and consequently, they provide more optimal delay-accuracy, power-accuracy and area-accuracy trade-offs. The accuracy-effort curves of ESAs can be improved either: (i) By improving the accuracy without imposing any additional delay, power and area overheads; or (ii) By reducing the delay, power and area without loosing accuracy. We achieve our objective using the former approach.

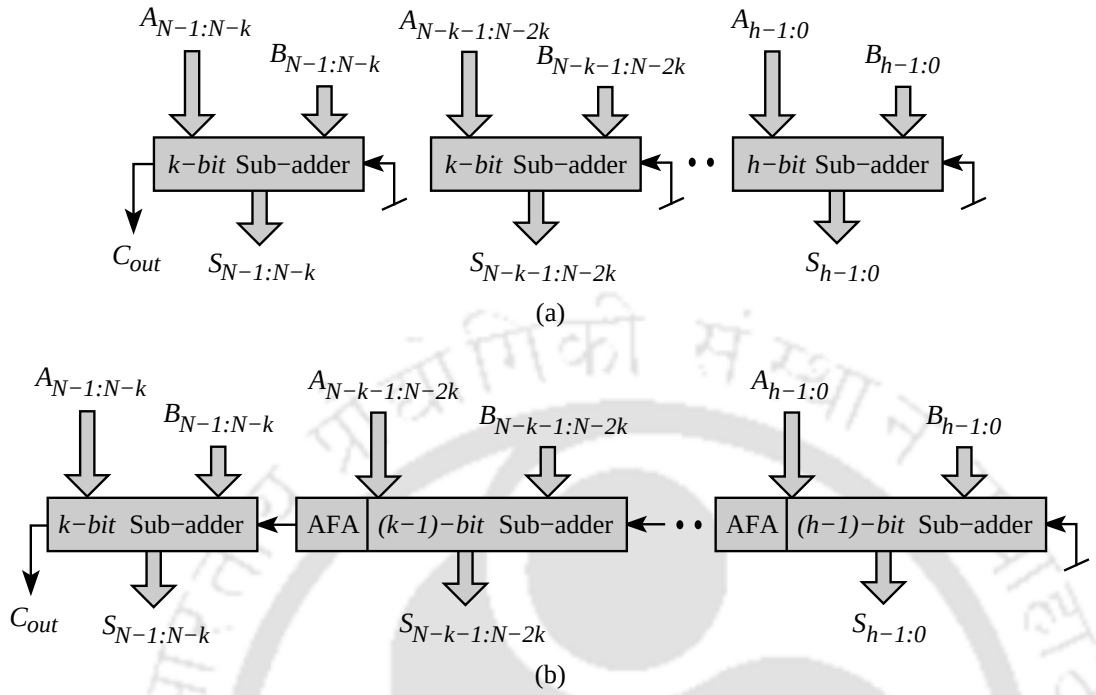


Figure 4.14: Generalized architecture of N-bit ESAs in which sub-adders are disjoint ($i.e.$, $l = 0$): (a) Original ESA; and corresponding (b) Modified ESA.

4.3.1 Proposed Modifications

In existing ESAs [12, 14, 61–73], C_{in} of all sub-adders is considered as 0. The objective behind this consideration is to limit the length of carry propagation to k-bit by securing C_{in} of sub-adders to be independent of C_{out} of previous sub-adders. We achieve the same objective in a different way. Our modification steps are: (i) Design an AFA in such a way that C_{out} is independent of C_{in}; (ii) Replace the most significant FA in all sub-adders (except the left most sub-adder) with AFA; and (iii) Feed C_{out} of AFAs to the C_{in} of next sub-adders. We re-design existing ESAs using the above-mentioned three steps. For the ease of illustration, original ESA in which sub-adders are disjoint ($i.e.$, $l = 0$) is shown in Fig. 4.14(a) and the corresponding modified ESA is shown in Fig. 4.14(b). Note that in modified ESAs, the length of carry propagation is still limited to k-bit as C_{out} of AFAs is independent of C_{in}. But now, as compared to the original ESAs (in which accuracy depends on k and l), accuracy of modified ESAs depends on the AFA used also. In this way, we introduce a third parameter ($i.e.$, AFA) which determines accuracy of the modified ESAs.

As discussed in Section 4.1.1.1, according to our experimental observations, $0.25 \leq P(G_i) \leq$

0.50 in original ESAs, where $P(G_i)$ represents the probability of C_{in} of SA$_i$ to be incorrect. On the other hand, in case of modified ESAs, $P(G)$ depends on the AFA used. Note that this AFA has design constraint that C_{out} should be independent of C_{in}. In a FA, to secure C_{out} independent of C_{in}, C_{out} should be either 0, 1 or a Boolean function of primary inputs A and B only. In such a scenario, 16 AFAs are possible (see Table 3.2). Among all these 16 AFAs, only AFA#3 and AFA#4 are optimal in terms of accuracy, delay, power and area. Therefore, if modified ESAs are designed using AFA#3 or AFA#4, then modified ESAs can provide higher accuracy without imposing any additional delay, power and area overheads. Consequently, modified ESAs designed using AFA#3 or AFA#4 can provide better accuracy-effort curves and more optimal delay-accuracy, power-accuracy and area-accuracy trade-offs as compared to the original ESAs.

In summary, in original ESAs, C_{in} of all sub-adders is considered as 0. In other words, it can be assumed that in original ESAs, all sub-adders (except the left most sub-adder) generate a 0-constant carry and this carry is feed to the next sub-adder. The probability of this 0-constant carry to be correct is $0.50-0.75$. On the other hand, in modified ESAs, instead of using 0-constant carry, we use AFA#3 or AFA#4 at the most significant bit of sub-adders to generate a random carry whose probability to be correct is 0.75 (see Table 3.2). Consequently, in modified ESAs, the probability of C_{in} of sub-adders to be correct is higher than that in case of original ESAs without imposing any additional overhead.

4.3.2 Results and Discussion

The absolute difference (in %) between the PR of 32-bit modified ESA and original ESA as a function of k and l is tabulated in Table 4.12. While designing an ESA, first k is decided based on delay requirements and then l is decided based on accuracy requirements of the applications. Generally, k is greater than log_2N [94] and less than $log_2N + 12$ [61]. Therefore, in Table 4.12, we include results only for $5 < k < 17$. It can be seen from Table 4.12 that: (i) Modified ESAs provide higher PR than the original ESAs; (ii) The improvement in PR depends on k and l; and (iii) The proposed approach provides a maximum improvement in PR of 31.25%.

Further, Fig. 4.15 shows the accuracy-effort curves of 32-bit original ESAs and modified ESAs, where accuracy is measured in terms of PR and effort is measured in terms of delay in Fig. 4.15(a) and

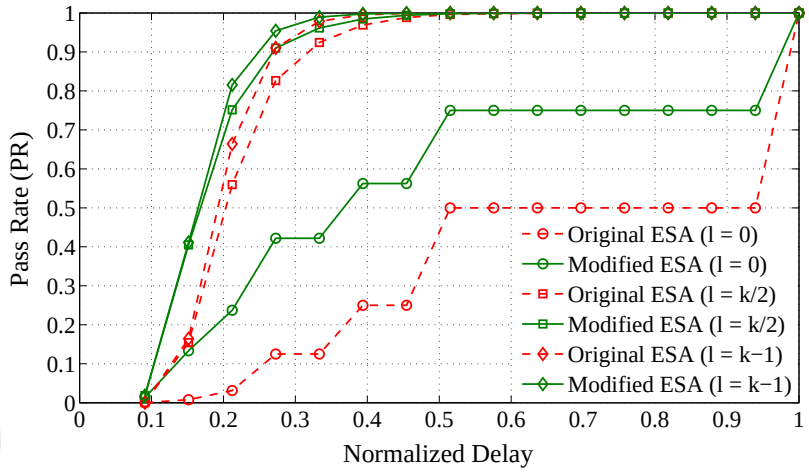
Table 4.12: Absolute difference (in %) between the PR of 32-bit modified ESAs and original ESAs.

k	l												
	0	1	2	3	4	5	6	7	8	9	10	11	12
6	20.61	27.08	24.38	19.20	15.30	15.15	—	—	—	—	—	—	—
7	25.39	27.56	21.12	16.42	11.63	8.819	8.484	—	—	—	—	—	—
8	29.68	26.97	18.62	12.90	8.328	5.755	4.394	4.381	—	—	—	—	—
9	29.68	24.80	18.62	10.82	7.106	4.419	2.999	2.269	2.174	—	—	—	—
10	29.68	24.80	15.41	10.82	5.821	3.726	2.275	1.531	1.058	1.057	—	—	—
11	31.25	24.80	15.41	8.517	4.470	3.016	1.907	1.154	0.677	0.533	0.508	—	—
12	31.25	20.31	11.32	8.517	4.470	2.289	1.535	0.774	0.485	0.340	0.243	0.243	—
13	31.25	20.31	11.32	5.957	4.470	2.289	1.158	0.774	0.388	0.243	0.171	0.121	0.115
14	31.25	20.31	11.32	5.957	3.051	1.544	0.776	0.582	0.292	0.146	0.097	0.061	0.054
15	31.25	20.31	11.32	5.957	3.051	1.544	0.776	0.582	0.292	0.146	0.097	0.060	0.036
16	25.00	20.31	11.32	5.957	3.051	1.544	0.776	0.389	0.195	0.146	0.073	0.048	0.024

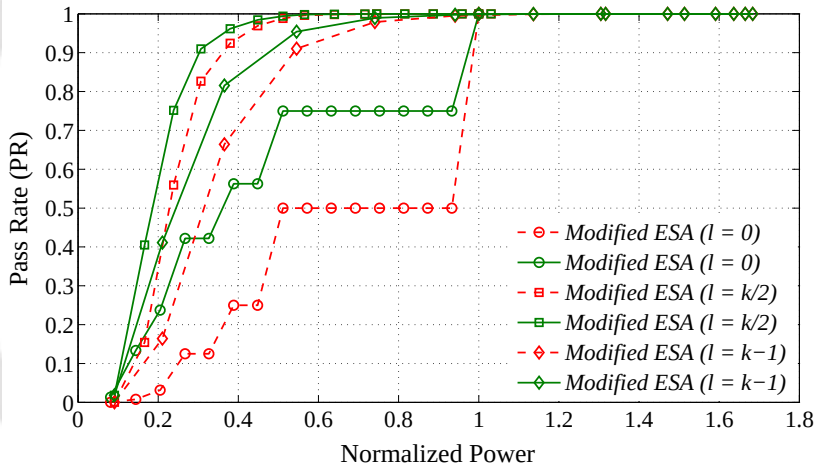
in terms of power in Fig. 4.15(b). It can be seen from Fig. 4.15 that modified ESAs provide better accuracy-effort curves as compared to the original ESAs. With improved accuracy-effort curves, modified ESAs provide more optimal PDAA trade-off than original ESAs. Accordingly, as compared to the original ESAs: (i) For a given delay and power, modified ESAs provide higher PR; or (ii) For a given PR, modified ESAs provide lower delay and power. Another very important feature of the proposed approach is that it provides a significant reduction in hardware complexity. As discussed in Section 4.1.3, for a given k, area of an N-bit ESA increases with l. For a given k, modified ESAs can provide same PR as that of original ESAs with lower values of l. Consequently, for a given PR, implementation of modified ESAs require less area than the original ESAs.

4.3.3 Delay and Power Improvements

Throughout the discussion, we claim that the proposed approach improves accuracy of existing ESAs keeping the delay, power and area intact. However, the proposed approach also provides improvements in delay and power when ESAs are used with an EDC logic to detect errors and correct them accordingly. As shown in Fig. 2.8, EDL detects error and ECL corrects them. Here, we explain briefly how the proposed approach provides improvements in delay and power when modified ESAs are used for designing the ACAs and VLSAs. As shown in Fig. 2.8, ACAs and VLSAs are augmented with an EF which activates or deactivates ECL. The EF signal activates ECL when ESA provides in-



(a)



(b)

Figure 4.15: Accuracy-effort curves of 32-bit original ESAs and modified ESAs as a function of k and l: (a) PR versus delay; and (b) PR versus power. Here, the markers/points in curves correspond to the values of k as 2, 4, 6 ... 32 from left to right.

correct result. If the ESA provides correct result, then EF signal deactivates ECL. Consequently, an ACA/VLSA takes one clock cycle when ESA provides correct result and takes two clock cycles when ESA provides incorrect result. We know that PR of an ESA represents the probability of output to be correct. Accordingly, average delay (D_{avg}) of an ACA/VLSA can be given by:

$$\begin{aligned} D_{avg} &= (PR \times D_{ESA}) + ((1 - PR) \times (D_{ESA} + D_{ECL})) \\ &= D_{ESA} + D_{ECL} - (PR \times D_{ECL}) \end{aligned} \tag{4.21}$$

where, D_{ESA} and D_{ECL} are the delays of ESA and ECL, respectively. Further, as discussed in Chapter 2, *power gating* is one of the most effective approaches used to reduce the power consumption of digital circuits [97]. In this approach, power consumption is reduced by shutting off the current to the circuit blocks that are not in use. In ACAs and VLSAs also, EF signal shuts off the ECL block when it is not in use [65]. Therefore, similar to Eqn. (4.21), average power consumption (P_{avg}) of an ACA/VLSA can be given by:

$$\begin{aligned} P_{avg} &= (PR \times P_{ESA}) + ((1 - PR) \times (P_{ESA} + P_{ECL})) \\ &= P_{ESA} + P_{ECL} - (PR \times P_{ECL}) \end{aligned} \tag{4.22}$$

where, P_{ESA} and P_{ECL} are the power consumptions of ESA and ECL, respectively. It is pretty visible from Eqn. (4.21) and Eqn. (4.22) that delay and power of an ACA/VLSA decrease with PR. As discussed earlier, modified ESAs provide higher PR than the original ESAs. Consequently, ACAs and VLSAs designed using the modified ESAs provide lower delay and power as compared to those designed using the original ESAs. Here, it should be noted that improvements in delay and power depend on k and l as the improvement in PR (due to the proposed approach) depends on k and l. Further, improvements in delay and power also depend on the ECL.

4.3.4 Multimedia Applications

ESAs are generally used for $N \geq 32$. For $N \leq 16$, these are less effective. In order to evaluate the effectiveness of the proposed approach in the worst-case scenario, we choose multimedia applications which use adders of $N \leq 16$. One such multimedia application is to find the absolute difference of two images. Absolute difference and its variants, such as *Sum of Absolute Differences* (SAD) and *Mean of Absolute Differences* (MAD) are used for a variety of purposes in multimedia applications, including object recognition, generation of disparity maps for stereo images, motion estimation for video compression, etc. [113]. As mentioned in Section 3.5.3, for evaluating the effectiveness of digital image processing approaches, several standard test images have been introduced in the literature as benchmark [90]. Among all existing test images, Lena picture is the most widely used standard test image. We also use the same Lena picture as benchmark. Further, we know that there are two

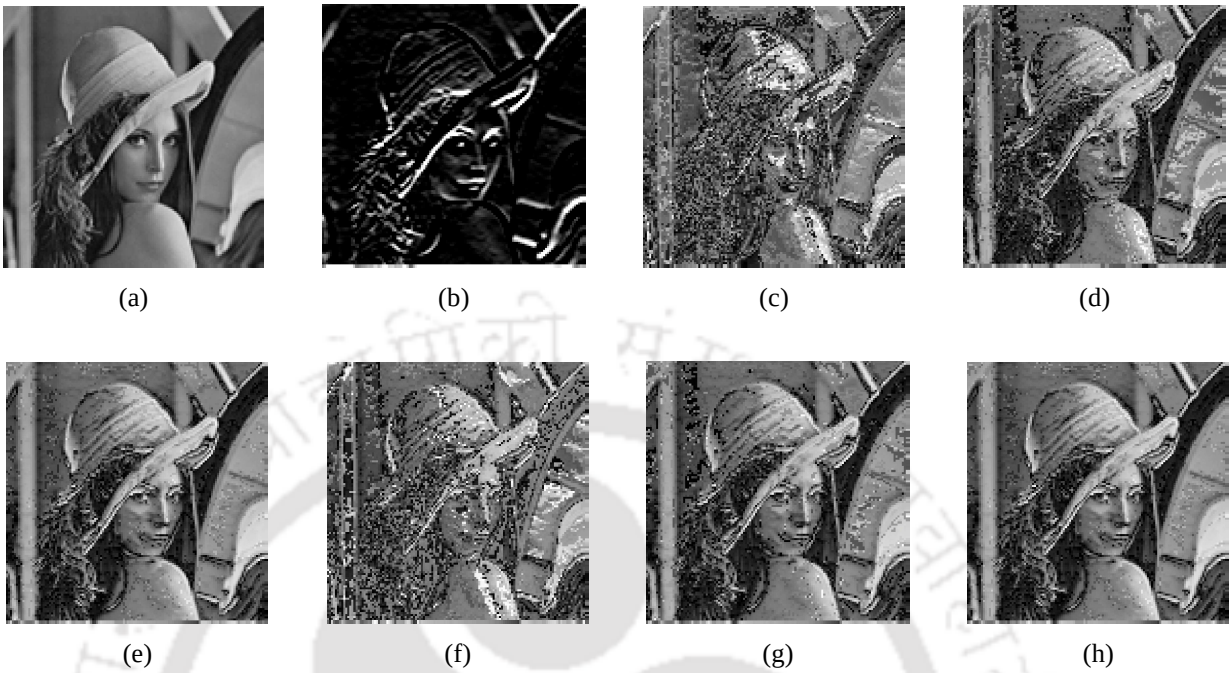


Figure 4.16: Image processing results: (a) Original Lena image; (b) Noisy Lena image; and absolute difference images generated using: (c) Original ESA with $k = 2$; (d) Original ESA with $k = 4$; (e) Original ESA with $k = 6$; (f) Modified ESA with $k = 2$; (g) Modified ESA with $k = 4$; and (h) Modified ESA with $k = 6$. Note that all the image processing results presented here are for $l = k/2$.

principle methods of evaluating the image quality [91]: (i) Subjective; and (ii) Objective. In order to have a fair assessment of image quality, we evaluate output images both subjectively as well as objectively. Our image processing framework is as follows.

(i) We first generate a noisy Lena image by applying Sobel filter to the original Lena image. The original and noisy Lena images are shown in Fig. 4.16(a) and Fig. 4.16(b), respectively.

(ii) We then compute the absolute difference between the original and noisy Lena images using 8-bit approximate absolute difference units. We implement approximate absolute difference units by replacing the 8-bit accurate adder (used in conventional absolute difference unit [114]) with 8-bit original ESAs and modified ESAs for different values of k and l.

(iii) We compare the absolute difference images generated using original ESAs and modified ESAs based on both subjective and objective methods. For objective assessment, we use three most widely used IQA metrics: (i) MSE; (ii) PSNR; and (iii) SSIM.

Table 4.13: IQA metrics of the absolute difference images shown in Fig. 4.16.

Quality Metrics	Original ESAs			Modified ESAs		
	$k=2$	$k=4$	$k=6$	$k=2$	$k=4$	$k=6$
MSE	2.63E+3	731.887	98.1931	1.82E+3	373.318	74.2734
PSNR	11.9922	19.8461	27.2106	15.4822	22.4104	29.9108
SSIM	0.19975	0.60952	0.86207	0.37958	0.74941	0.91709

Fig. 4.16(c) – Fig. 4.16(h) show the absolute difference images generated using 8-bit original ESAs and modified ESAs as a function of k for $l = k/2$. It can be subjectively visualized from Fig. 4.16(c) – Fig. 4.16(h) that the absolute difference images generated using modified ESAs are more precise as compared to the absolute difference images generated using original ESAs. Further, for the sake of objective assessment, our simulation results of IQA metrics are tabulated in Table 4.13. It can be seen from Table 4.13 that modified ESAs provide better IQA metrics ($i.e.$, lower MSE, and higher PSNR and SSIM) than original ESAs. It is clear from Fig. 4.16 and Table 4.13 that in DSP applications, use of modified ESAs provides more precise final outputs (both subjectively and objectively) than the use of original ESAs. Consequently, modified ESAs preferred over original ESAs while designing digital circuits for current/future error-resilient applications.

4.4 Summary

ESA is the most widely used approach for designing the approximate adders, accuracy configurable adders and variable latency speculative adders. In this Chapter, we proposed analytical models to estimate accuracy (ED, ER, MED and MSE), delay, power and area of ESAs. To the best of our knowledge, we the first time proposed analytical models to estimate delay, power and area of ESAs considering different adder architectures. Further, the key features of the proposed analytical models of ED, ER, MED and MSE are that: (i) They are generalized, $i.e.$, work for all possible configurations of an N-bit ESA; and (ii) They are superior (or at par) to the existing analytical models. The proposed analytical models can assist designers for estimating the design metrics of any configuration of an N-bit ESA more accurately without going for the programming efforts and time-consuming simulations. From the proposed analytical models, we observed that in an N-bit ESA, there exist multiple

(more than one) configurations which exhibit similar accuracy. However, these configurations exhibit different delay, power and area. Therefore, for a given accuracy, to apriori select the optimal configurations which provide minimal delay, power and/or area is a challenging decision for designers. We presented an optimization framework that can assist designers in making such important decisions early in the design phase of ESAs. This enables designers for efficient, intelligent and goal oriented implementations of ESAs. We demonstrated the proposed optimization framework using RFD heuristic algorithm. Although we demonstrated single-objective optimization (delay, power or area) subjected to single constraint (PR), the proposed approach can be customized/extended for the other cases of optimization. Further, we know that accuracy of an ESA does not depend on the adder architecture used to implement it, however, its delay, power and area depend significantly. Therefore, the optimal configurations vary with adder architectures used to implement the ESA. In our analysis, we considered three types of architecture. However, due to similar behavior/structure, our analysis implicitly provides fair insights into a wide range of adders.

Further, from the proposed analytical models, we observed that there is a scope of improvement in accuracy-effort curves of ESAs. The accuracy-effort curves of ESAs need to be improved so that they provide optimal delay-accuracy, power-accuracy and area-accuracy trade-offs. Intended to improve the accuracy-effort curves of ESAs, we proposed modifications. The crux of the proposed modifications is that the probability of C_{in} of sub-adders to be correct in original ESAs is $0.50 - 0.75$, whereas in case of modified ESAs, it is 0.75 without imposing any additional delay, power and area overheads. With this improvement, modified ESAs provide better accuracy-effort curves and more optimal trade-offs. Accordingly, as compared to the original ESAs: (i) For a given accuracy, modified ESAs provide lower delay, power and area; or (ii) For a given delay, power and area, modified ESAs provide higher accuracy. In addition to the accuracy enhancement, the proposed approach also provides improvements in delay and power when modified ESAs are used for designing the ACAs/VLSAs. In order to evaluate the effectiveness of the proposed approach in real-life applications, we processed Lena image using original ESAs and modified ESAs. Our image processing results showed that the output images processed using modified ESAs are more precise (both subjectively and objectively) than the output images processed using original ESAs. As a whole, modified ESAs are more efficient (in terms

of accuracy, delay, power and area) than the original ESAs.

In Chapter 3, we have discussed the proposed work on AFAs and this Chapter, we have discussed the proposed work on ESAs. Now these approximate adders need to be explored. In the next Chapter, we examine the effectiveness of approximate adders in cryptography applications, yield enhancement and designing other approximate arithmetic operations.

5

Application of Approximate Adders

Contents

Over the past decade, most of the research work on approximate adders have demonstrated their effectiveness in multimedia applications. In Chapter 3 and Chapter 4, we also evaluate the effectiveness of AFAs and ESAs in two different image processing applications. However, it has been observed that a very high degree of error-resilience is prevalent in a broad spectrum of applications [8,100,101]. Therefore, the applicability of approximate adders need to be explored for state-of-the-art applications. In addition to the delay and power benefits, approximate adders can also be used to address various technology/design related issues, such as yield enhancement. Further, we know that in binary arithmetic, all basic operations, such as addition, subtraction, multiplication and division use adders as the key components. Therefore, approximate adders can be used as the basic building blocks for designing other approximate arithmetic operations [87–89].

In this Chapter, we explore the feasibility of approximate adders in cryptography applications, yield enhancement and designing other approximate arithmetic operations. We know that the overall delay, power and area of SHA-1 are determined by modular-32 adders. In order to examine the effectiveness of approximate adders in cryptography applications, we design an approximate SHA-1 (ApproxSHA-1) using approximate modular-32 adders. Further, we explore the effectiveness of approximate adders for yield enhancement based on the observation that approximate data path modules can improve the functional yield and parametric yield due to decrease in area and delay, respectively. We also propose analytical models to estimate the yield of approximate arithmetic circuits. Moreover, MAC unit is the heart of real-time multimedia applications. As MAC unit lies in the critical path, it determines the overall delay, power and area of multimedia systems. In order to improve the efficiency of multimedia systems, we present an approximate MAC unit (ApproxMAC) which exploits approximate hybrid redundant adder as the basic building blocks. For evaluating the effectiveness of the proposed approach in real-life applications, we demonstrate Lena image processing using ApproxMAC unit. In the end, we summarize the Chapter.

The rest of this Chapter is organized as follows. Section 5.1 presents the approximate SHA-1 and its test results. Section 5.2 provides the analytical models to estimate the yield of approximate arithmetic circuits and evaluates the effectiveness of approximate adder for yield enhancement. Section 5.3 discusses the approximate MAC unit and its results. Finally, Section 5.4 concludes the Chapter.

5.1 Cryptography Applications

Over the past decades, CMOS technology scaling has been the primary key for continuous progress of semiconductor industry. However, as discussed in Chapter 1, we are now in a phase where CMOS technology scaling is becoming less and less effective at improving the system capability. On the other hand, today every kind of businesses is using E-commerce to sustain in the market. In 2017, retail E-commerce sales worldwide amounted to 2.3 trillion US dollars and E-retail revenues are projected to grow to 4.88 trillion US dollars in 2021 [115]. The simultaneous demand of secured E-commerce along with limitations of CMOS technology scaling motivated us to evaluate the effectiveness of approximate computing in cryptography applications. Hash functions play an important role in today's E-commerce as they are the basic building blocks of security applications [93]. While designing a hash function, designers do not aim for the golden outputs, rather they aim for a set of random outputs with some specific properties for a correlated set of inputs. Therefore, the intermediate computations in conventional hash functions can be executed approximately subjected to the output sets fulfill the hash function criteria. Since SHA-1 is one of the most widely used cryptographic hash functions and has many future aspects, we consider the same in our analysis.

5.1.1 Conventional SHA-1

SHA-1 takes a 64-bit message (M) as input and produces a 160-bit hash value known as message digest (H). As shown in Algorithm 1, SHA-1 consists of the following steps.

(i) **Variables initialization:** Initialize variables as: $h_0 = 0x67452301$, $h_1 = 0xEFCDAB89$, $h_2 = 0x98BADCFE$, $h_3 = 0x10325476$ and $h_4 = 0xC3D2E1F0$.

(i) **Pre-processing:** If message length is multiple of 8 bits, then append 1 to the message by adding $0x80$. Now append k $0s$ such that the resulting message length in bits is congruent to $-64 \equiv 448 \pmod{512}$, where $0 \leq k < 512$. Append M, the original message as a 64-bit length. Thus, the total message length is multiple of 512 bits.

Algorithm 1: Conventional SHA-1

Input: M
Output: H
Variables initialization:
$h_0 = 0x67452301$, $h_1 = 0xEFCDAB89$, $h_2 = 0x98BADCFE$, $h_3 = 0x10325476$ and $h_4 = 0xC3D2E1F0$.
Pre-processing:
Do proper appending so that the total message length is multiple of 512 bits.
Chunks formation:
Break message into 512-bit chunks.
for each chunk X_i **do**
  for $0 \leq j \leq 79$ **do**
    if $0 \leq j \leq 15$ **then**
      $w_j =$ break the chunk X_i into 16 32-bits words;
    else if $16 \leq j \leq 79$ **then**
      $w_j = LS^1[w_{j-3}$ XOR w_{j-8} XOR w_{j-14} XOR $w_{j-16}]$;
    end if
    Hash initialization:
    $a = h_0$, $b = h_1$, $c = h_2$, $d = h_3$, and $e = h_4$;
    Procerssing:
    if $0 \leq j \leq 19$ **then**
      $f =$ (b AND c) OR ($\overline{B}$ AND D);
      $k = 0x5A827999$;
    else if $20 \leq j \leq 39$ **then**
      $f = b$ XOR c XOR d;
      $k = 0x6ED9EBA1$;
    else if $40 \leq j \leq 59$ **then**
      $f =$(b AND c) OR (b AND d) OR (c AND d);
      $k = 0x8F1BBCDC$;
    else if $60 \leq j \leq 79$ **then**
      $f = b$ XOR c XOR d;
      $k = 0xCA62C1D6$;
    end if
    $temp = LS^5(a) + f + e + k + w_j$;
    $e = d$;
    $d = c$;
    $c = LS^{30}(b)$;
    $b = a$;
    $a = temp$;
  end for
  $h_0 = h_0 + a$;
  $h_1 = h_1 + b$;
  $h_2 = h_2 + c$;
  $h_3 = h_3 + d$;
  $h_4 = h_4 + e$;
end for
$H = \{h_0, h_1, h_2, h_3, h_4\}$;

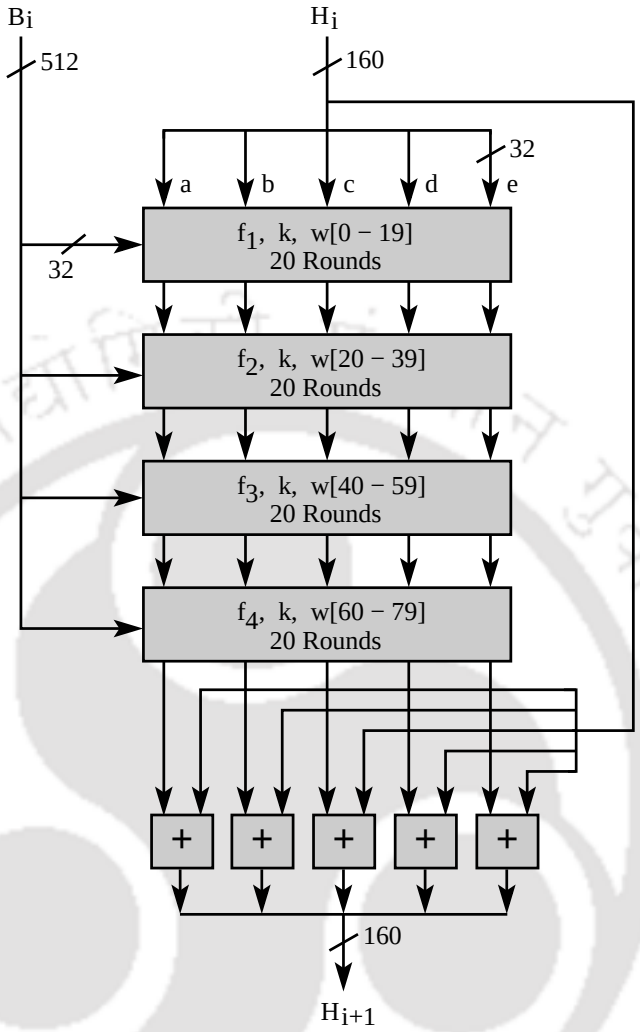


Figure 5.1: Block diagram of processing block of SHA-1.

(ii) **Chunks formation:** Break message into 512-bit chunks. For each chunk, break it into 16 32-bit words $w[j]$ for $0 \leq j \leq 15$. Extend the 16 32-bit words into 80 32-bit words using Eqn. (5.1).

$$w_j = LS^1[w_{j-3} \oplus w_{j-8} \oplus w_{j-14} \oplus w_{j-16}] \quad \text{for} \quad 16 \leq j \leq 79 \tag{5.1}$$

(iii) **Hash initialization**: Initialize hash value to five variables for every chunk as: $a = h_0$, $b = h_1$, $c = h_2$, $d = h_3$ and $e = h_4$.

(iv) **Processing:** This is the main loop of SHA-1 which determines the overall complexity. As shown in Fig. 5.1, the processing block has 80 rounds, $i.e.$, for every chunk, the elementary block of processing block (see Fig. 5.2) gets executed 80 times.

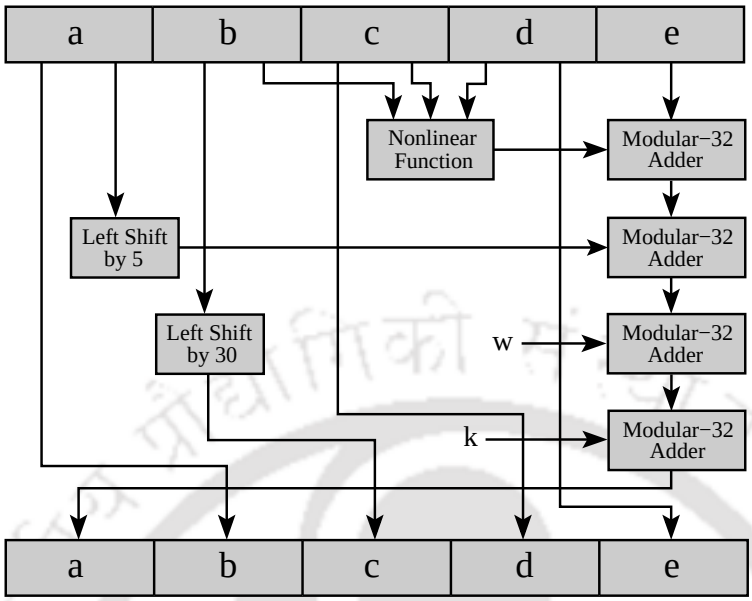


Figure 5.2: Block diagram of elementary block of processing block.

5.1.2 Approximate SHA-1

The elementary block (see Fig. 5.1) gets executed 80 times for every chunk, and thus, it can be considered as the basic building block of SHA-1. As shown in Fig. 5.2, it consists of a non-linear function, two shifters and four modular-32 adders. Note that the four modular-32 adders are connected in series. Further, as mentioned earlier, these gets executed 80 times for every chunk. Therefore, the overall delay, power and area of SHA-1 are determined by modular-32 adders. In order to examine the effectiveness of approximate adders in cryptography applications, we replace the conventional modular-32 adders with approximate modular-32 adders. We call the resulting approximate SHA-1 as "ApproxSHA-1". For the sake of demonstration, we use ApproxADD (discussed in Chapter 3) because in cryptography applications, each output bit has same weight, and thus, the number of incorrect bits matters irrespective of their positions. Since we are computing all N-bit operations using AFAs, this is known as full approximation. This full approximation in cryptography applications leads to better delay, power and area benefits as compared to the multimedia applications. Note that full approximation is not feasible in multimedia applications because in multimedia applications, higher order bits have more weight as compared to the lower order bits. Consequently, impact of errors in higher order bits is more severe than the errors in lower order bits.

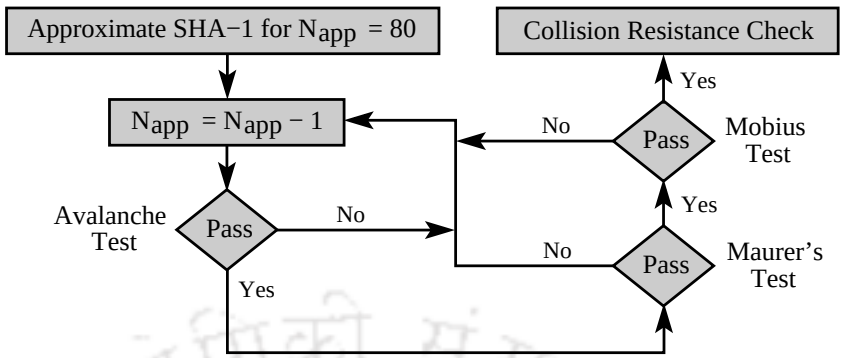


Figure 5.3: Testing approximate SHA-1 and finding $N_{app_{max}}$.

5.1.3 Maximum Approximation

The approximation level in cryptography applications is decided by a set of tests which determines whether the approximate hash function is cryptographic or not. In other words, there is a set of properties which should be fulfilled by the approximate hash function to be cryptographic. In order to determine the maximum level of approximation in SHA-1, we use two types of elementary blocks: (i) Accurate block, which consists of conventional modular-32 adders; and (ii) Approximate block, which consists of approximate modular-32 adders. We compute first N_{app} rounds using approximate block and the next $(80 - N_{app})$ rounds using accurate block. Consequently, in approximate hash functions, the approximation level increases with N_{app}. For evaluating the maximum possible approximation level, we test ApproxSHA-1 for $N_{app} = 80$ and then decrease N_{app} till the resulting output sets fulfill all cryptographic hash function criteria. Therefore, the maximum value of N_{app} which fulfills all cryptographic hash function criteria is $N_{app_{max}}$.

5.1.4 Results and Discussion

A cryptographic hash function must satisfy three criteria: (i) Pre-image resistance; (ii) Second pre-image resistance; and (iii) Collision resistance. For examining all these criteria, the commonly used four tests are: (i) Avalanche test; (ii) Maurer's universal statistical test; (iii) Statistical Mobius analysis; and (iv) Near collision test. As shown in Fig. 5.3, we conduct three tests sequentially. If ApproxSHA-1 passes current test, then only we conduct the next test; otherwise, we reduce the approximation level ($i.e.$, value of N_{app}) and re-run the test. In this way, we find $N_{app_{max}}$

Table 5.1: Implementation results on Virtex-6 FPGA.

Design Parameters	Conventional SHA-1	ApproxSHA-1
Number of occupied slices	686	506
Maximum clock frequency (MHz)	98	300
Throughput ($Mbps$)	628	1921
Total power consumption (W)	3.57	3.60

for ApproxSHA-1 which passes all tests. For testing ApproxSHA-1 over cryptographic hash function criteria, we simulate it for one billion messages in C/C++. From our analysis, we observe that ApproxSHA-1 passes all these three tests with all 80 rounds approximated. In the end, we evaluate ApproxSHA-1 for its collision resistance quality. Further, we implement conventional SHA-1 and ApproxSHA-1 on Virtex-6 *Field Programmable Gate Arrays* (FPGA). Our implementation results are tabulated in Table 5.1. It can be seen from Table 5.1 that almost for similar power consumption, ApproxSHA-1 provides $3.12\times$ speedup. Further, to the best of our knowledge, this is the first time that throughput of a SHA-1 reaches $\approx 2Gbps$ without pipeline.

5.2 Yield Enhancement

With continued innovations in the fabrication process steps, CMOS technology is moving toward finer geometries, exhibiting higher performance, higher energy efficiency and lower silicon area per computation [116]. Consequently, the ongoing complexity of CMOS integrated circuits, $i.e.$, transistor count has reached up to billions. However, with the relentless scaling of CMOS technology in sub-nanometer regime, chip yield has become a matter of concern for the semiconductor industry due to the fact that profits are tied directly to the yield [9]. Though researchers are thinking about the post-CMOS technologies, however, as shown in Fig. 1.2, there appears to be no alternate technology that can replace *End-of-Roadmap* CMOS over the next $2-3$ decades [7].

In such a scenario, researchers exploit the inherent error-resilience of applications and introduce the concept of *threshold testing* [16, 117–119]. In traditional testing, a normal set of test vectors is applied to a *Chip Under Test* (CUT) and the responses are compared with those expected from a fault-free circuit. Thus, traditional testing has capability only to classify chips as either perfect, $i.e.$,

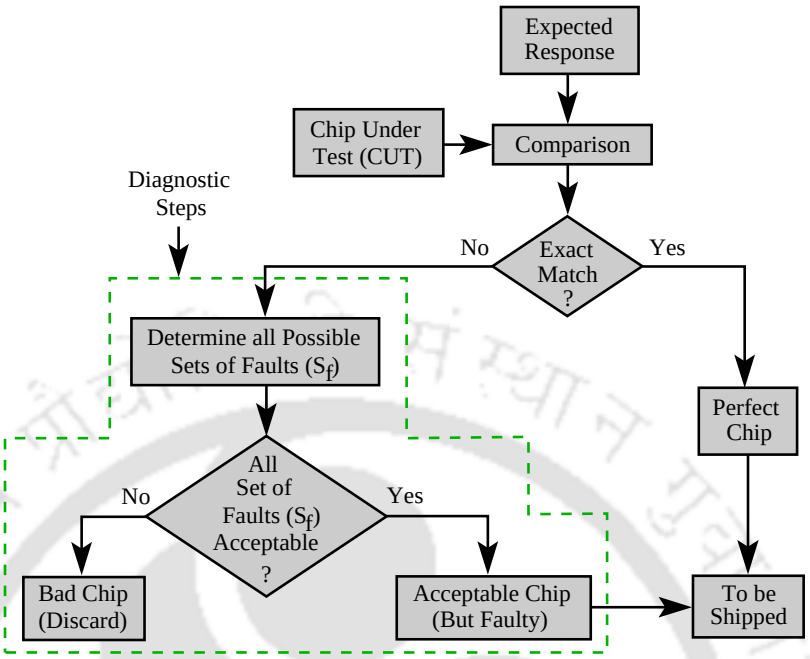


Figure 5.4: Threshold testing that identifies acceptable (but faulty) chips [16].

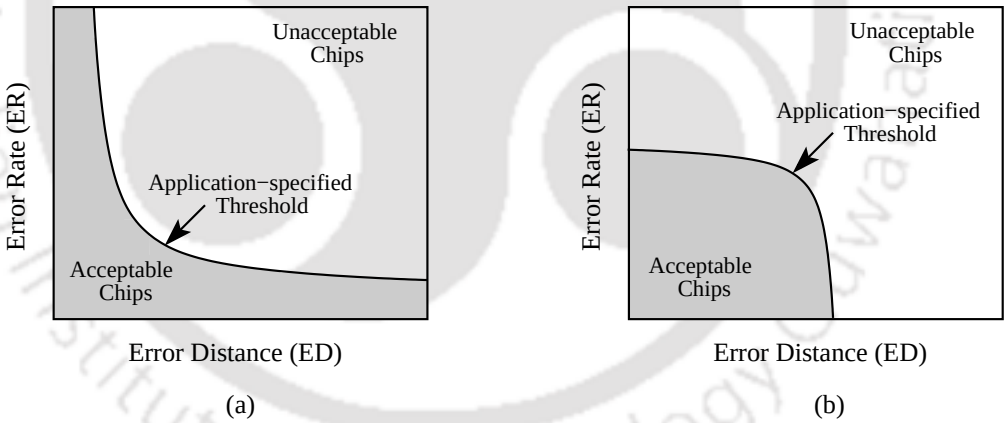


Figure 5.5: Application-specified threshold: (a) Fail small or fail rare; and (b) Fail small and fail rare.

without any error-producing defect or imperfect, $i.e.$, those that have one or more error-producing defects. In traditional testing, since every imperfect chip is discarded, yield measured in terms of perfect chips is decreasing drastically (see Fig. 1.4). However, as shown in Fig. 5.4 and Fig. 5.5, for error-resilient applications, imperfect chips may still be used, provided the errors are of certain types and their severities are within application-specified threshold, where the application-specified threshold is characterized in terms of quality metrics. Researchers/designers exploit threshold testing to identify such acceptable chips. As shown in Fig. 5.4, for CUTs that do not match the exact

response, the response of the CUT is used to perform diagnosis to separate the acceptable chips from the bad chips. In this diagnostic step, all possible sets of faults (S_f) in each imperfect chip are identified. Further, if all possible sets of faults meet the criteria of application-specified threshold, then the chip is considered as acceptable for that application.

If defects that produce error within application-specified threshold are acceptable, then we can exploit this opportunity during hardware design to improve the chip yield. Accordingly, we explore the effectiveness of approximate adders for yield improvement based on the observation that approximate data path modules can improve the functional and parametric yields due to decrease in area and delay, respectively [1]. We discuss the proposed approach $w.r.t.$ approximate adders, however, the proposed approach can be customized for any data path module.

5.2.1 Yield Models

Under manufacturing yield model [17], the total chip yield (Y) is represented by the product of functional yield (Y_{func}) and parametric yield (Y_{para}), $i.e.$, $Y = Y_{func} \times Y_{para}$. The functional yield loss ($1 - Y_{func}$) is associated with spot defects and occurs when there is shorts or opens in the chip. Highly localized spot defects are the predominant source that characterize the area dependency nature of functional yield. Let's assume A represents the active area of the device and D represents the average density of spot defects. Now the probability of a chip to be perfect can be given by:

$$P_{perfect} = e^{-nAD} \tag{5.2}$$

where, n is the total number of devices in the chip. Since the value of D varies from chip to chip, the average functional yield must be summed over all the chips. Considering $f(D)$ as a normalized distribution function of chips in defect densities, Murphy [120] predicts the functional yield as:

$$Y_{func} = \int_0^\infty e^{-nAD}\, f(D)\, dD \tag{5.3}$$

In Eqn. (5.3), only perfect chips are considered. However, as discussed earlier, in threshold testing, chips with defects that produce error within application-specified threshold should also be taken into consideration. Let's assume γ is the probability of defects that produce error within application-

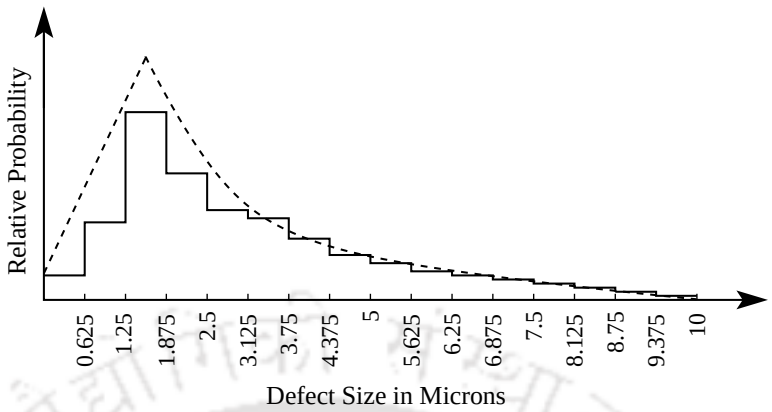


Figure 5.6: Defect size distribution [17].

specified threshold. Now the probability of a chip to be acceptable can be given by:

$$P_{acceptable} = \gamma nADe^{-nAD} \tag{5.4}$$

We know that faults in digital circuits/systems are just another source of errors. Analogically, an approximate adder can be considered equivalent to an accurate adder with defects. Further, since approximate adders are designed for a targeted application-specified threshold, $\gamma = 1$. Accordingly, the functional yield in approximate computing paradigm can be given by:

$$Y_{func_{approx}} = \int_0^\infty (e^{-nAD} + \gamma nADe^{-nAD})\, f(D)\, dD \tag{5.5}$$

The functional yield depends not only on the defect density, but it also on the defect size distribution. Researches/designers model the distribution of defect sizes using different mathematical functions. For the sake of simplicity, the most widely accepted distribution is the exponential distribution (see Fig. 5.6). Assuming $f(D) = \frac{e^{-D/D_0}}{D_0}$ (where D_0 is a technology dependent parameter), the functional yield in approximate computing paradigm can be given as:

$$Y_{func_{approx}} = \frac{1}{1+nAD_0} + \frac{nAD_0}{(1+nAD_0)^2} \tag{5.6}$$

On the other hand, parametric yield is the percentage of functionally correct chips that meet the design specifications, such as delay and power. The parametric yield loss ($1 - Y_{para}$) is associated with process parameter variations and occurs when there is an unacceptable mismatch between the intended and obtained design specifications. As shown in Fig. 5.7, while estimating the parametric

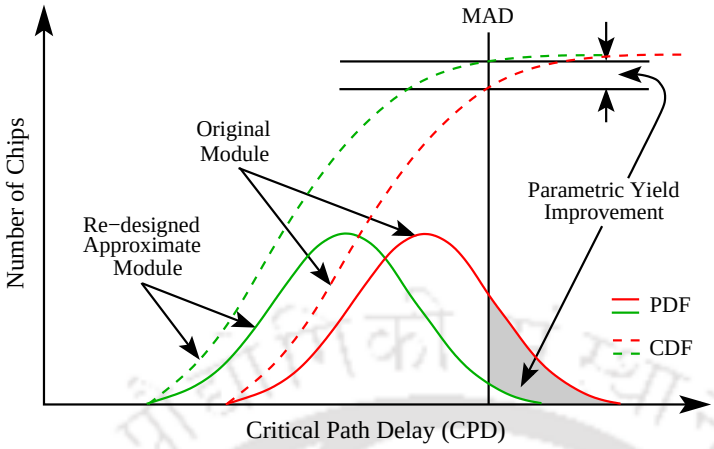


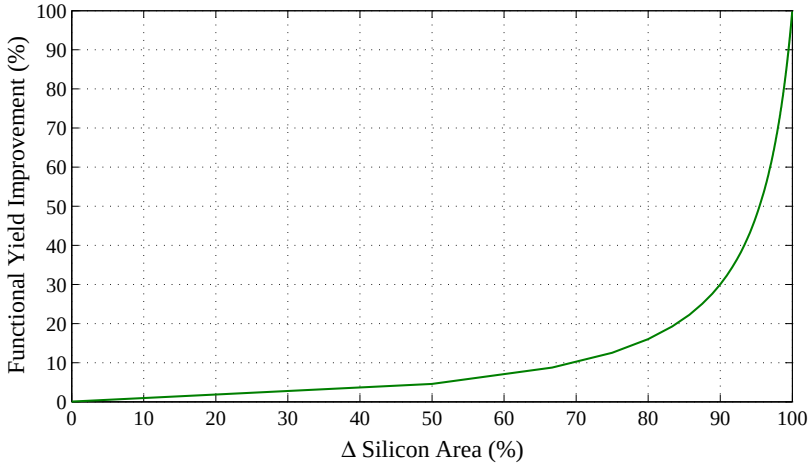
Figure 5.7: Distribution of CPDs under process parameter variations.

yield, we assume that: (i) *Critical Path Delay* (CPD) is normally distributed [121]; and (ii) Chip having any data path module with delay higher than *Maximum Allowed Delay* (MAD) is rejected. In such a scenario, area under the curve on the left of MAD in Fig. 5.7 represents the parametric yield. Further, for assessing the shaded region in Fig. 5.7 (which represents the parametric yield improvement due to decrease in delay), we describe the parametric yield as:

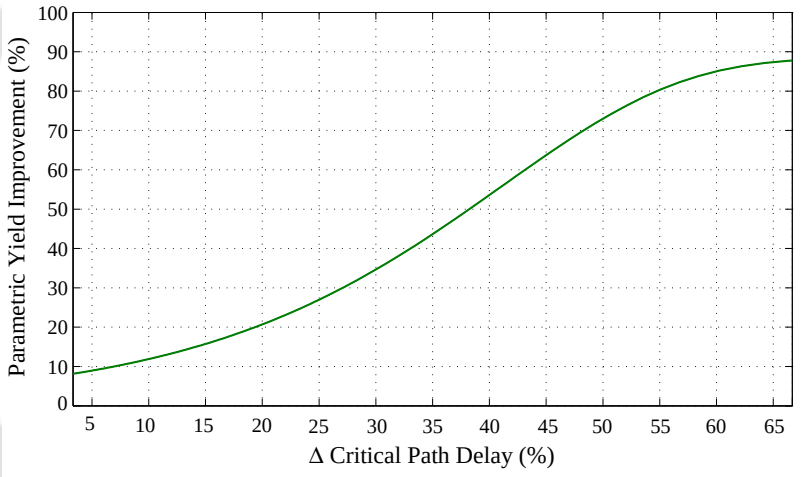
$$\begin{aligned} Y_{para} &= \int_{-\infty}^{MAD} CPD(t)\, dt \\ &= \int_{-\infty}^{MAD} \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(t-\mu)^2}{2\sigma^2}}\, dt \\ &= \frac{1}{2}\left(1 + erf\left(\frac{MAD-\mu}{\sigma\sqrt{2}}\right)\right) \end{aligned} \tag{5.7}$$

where, μ and σ are the mean and standard deviation of the critical path delay distribution in the original data path module, respectively. Similarly, if μ_{approx} is the mean of the critical path delay distribution in the re-designed approximate module, then the parametric yield in approximate computing paradigm can be given by:

$$Y_{para_{approx}} = \frac{1}{2}\left(1 + erf\left(\frac{MAD-\mu_{approx}}{\sigma\sqrt{2}}\right)\right) \tag{5.8}$$



(a)



(b)

Figure 5.8: Yield improvements for a 16-bit RCA: (a) Functional yield improvement due to decrease in silicon area; and (b) Parametric yield improvement due to decrease in critical path delay.

5.2.2 Results and Discussion

Using Eqn. (5.3) and Eqn. (5.5), Fig. 5.8(a) shows the functional yield improvement for a 16-bit RCA due to decrease in silicon area. Similarly, using Eqn. (5.7) and Eqn. (5.8), Fig. 5.8(b) shows the parametric yield improvement for a 16-bit RCA due to decrease in critical path delay. In $32nm$ CMOS technology, 3σ variations can be as worst as 54% of the mean of critical path delay distribution [116]. Therefore, while estimating the parametric yield improvement (Fig. 5.8(b)), we consider $\sigma = 18\%$. We then extract the functional yield and parametric yield improvements due to decrease in silicon area and critical path delay from Fig. 5.8(a) and Fig. 5.8(b), respectively.

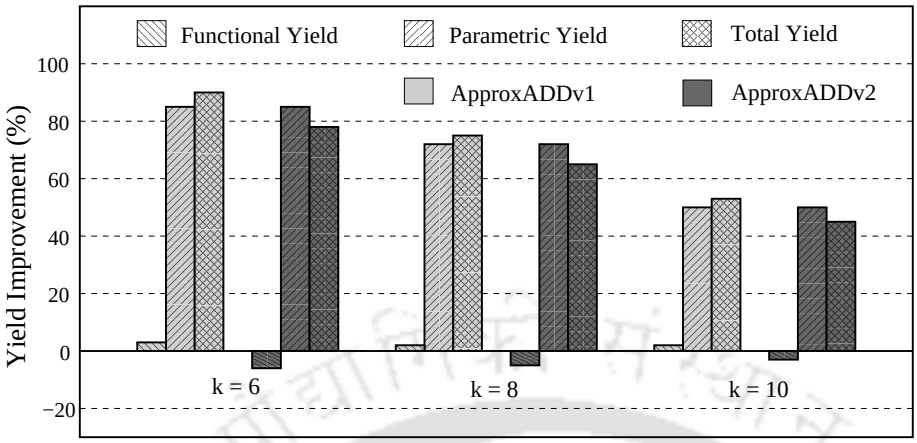


Figure 5.9: Yield improvements for 16-bit ApproxADD$v1$ and ApproxADD$v2$.

Our results of the effective yield improvement for the proposed 16-bit ApproxADD$v1$ and ApproxADD$v2$ (discussed in Chapter 3) for different values of k are shown in Fig. 5.9. While plotting Fig. 5.9, the functional and parametric yield improvements due to decrease in silicon area and critical path delay are extracted from Fig. 5.8. It can be seen from Fig. 5.9 that 16-bit ApproxADD$v1$ improves the chip yield by 89.63%, 74.44% and 52.63%; and 16-bit ApproxADD$v2$ improves the chip yield by 79.08%, 65.81% and 43.72% for $k = 6$, 8 and 10, respectively. The negative values in Fig. 5.9 show the functional yield loss due to increase in silicon area imposed by EDC logic. However, the overall yield improvement is still satisfactory. One important observation from Fig. 5.9 is – in sub-nanometer regime, parametric yield improvement due to decrease in delay is more important than the functional yield improvement due to decrease in silicon area.

5.3 MAC Unit

MAC units are extensively used in DSP processors. As tabulated in Table 5.2, CEVA-XC family of DSP processors has reached 128 (16×16)-bit and 32 (32×32)-bit MAC units [18]. This continuous increase in bit-width and the number of MAC units motivated us to explore approximate adders for the optimization of MAC unit. Binary MAC unit consists of multiplication and accumulation. We know that multiplication is performed by summing the partial products. Binary partial products summation imposes inherent carry propagation. On the other hand, redundant number system eliminates the carry propagation, however, it requires complex redundant manipulations and additional binary-redundant-

Table 5.2: CEVA-XC family of DSP processors [18].

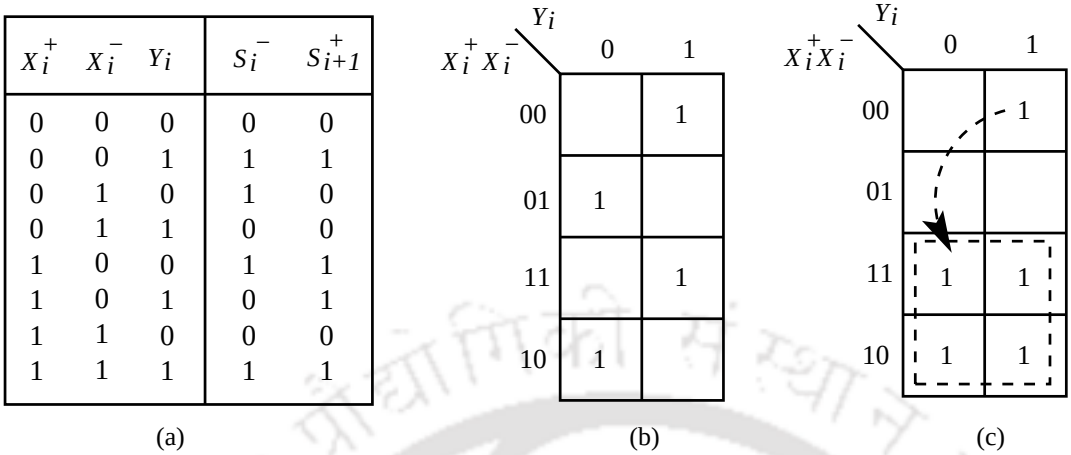
DSP Processors	Number of MAC Units		Bandwidth (in bits)	SoC Integration
	16×16	32×32		
CEVA-XC4100	16	4	512	AMBA3-FIC
CEVA-XC4110	32	8	512	AMBA3-FIC
CEVA-XC4200	32	8	1024	AMBA3-FIC
CEVA-XC4210	64	16	1024	AMBA3-FIC
CEVA-XC4400	64	16	2048	AMBA4-FIC
CEVA-XC4410	128	32	2048	AMBA4-FIC

binary converters. Therefore, we consider *hybrid redundant number system* in which one operand is in binary number system and the another operand is in redundant number system. The three key advantages of the MAC unit designed using hybrid redundant number system are: (i) Carry-free addition, leading to lesser delay and lesser number of pipelining latches; (ii) Reduced latency of the MAC operation from $2N$ to N, where N is the bit-width; and (iii) Since binary bit set is a subset of the redundant bit set, no binary-redundant converter is required.

5.3.1 Hybrid Redundant Adder

A MAC unit computes $A = A + (B \times C)$. It consists of three operations: (i) Multiply two numbers; (ii) Add the product with an accumulator; and (iii) Store the results in the accumulator. As mentioned earlier, multiplication is performed by summing the partial products. Therefore, at micro-architecture level of abstraction, adders can be treated as the basic building blocks of MAC units. Consider the addition operation $S = X + Y$, where S and X are in redundant number system and Y is in binary number system. The binary number system allows $\{0, 1\}$ bit set, whereas the redundant number system allows $\{-1, 0, +1\}$ bit set. The value of a redundant bit X is given by the difference of two bits used to encode it, *i.e.*, $X = X^+ - X^-$, where $X^+, X^- \in \{0, 1\}$ and $X \in \{-1, 0, +1\}$. For example, if $X^+ = 0$ and $X^- = 0$, then $X = 0$; if $X^+ = 0$ and $X^- = 1$, then $X = -1$; if $X^+ = 1$ and $X^- = 0$, then $X = +1$; and if $X^+ = 1$ and $X^- = 1$, then $X = 0$.

The above-mentioned coding scheme can be implemented using a generalized type-1 adder. The truth table of a hybrid redundant adder is shown in Fig. 5.10(a), where X_i^+, X_i^- and Y_i are the inputs, and S_i^- and S_{i+1}^+ are the outputs. Note that hybrid redundant adder adds a binary bit Y_i with a



X_i^+	X_i^-	Y_i	S_i^-	S_{i+1}^+
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

(a)

(b) (c)

Figure 5.10: Hybrid redundant adder: (a) Truth table; (b) K-map for S_i^-; and (c) K-map for S_{i+1}^+.

redundant bit X_i, where $X_i = X_i^+ - X_i^-$. The final sum bit S_i is given by the difference of two bits S_i^+ and S_i^- with the initial conditions of $S_0^+ = 0$ and $S_N^- = 0$.

5.3.2 Proposed Approach

In hybrid redundant adder, the overall delay, power and area is dominated by the implementation of $S_{i+1}^+ = \overline{X^-}_i Y_i + X_i^+ \overline{X^-}_i + X_i^+ Y_i$. In order to improve the efficiency of hybrid redundant adder, we manipulate the Boolean function of S_{i+1}^+. Fig. 5.10(c) shows the modified K-map along with the indication of entry changed from $\overline{X^+}_i \overline{X^-}_i Y_i$ to $X_i^+ X_i^- \overline{Y}_i$. The motivation behind this change is the observation that by moving 1 from $\overline{X^+}_i \overline{X^-}_i Y_i$ to $X_i^+ X_i^- \overline{Y}_i$ yields a significant reduction in logic complexity. For approximate hybrid redundant adder, S_i^- remains the same, *i.e.*, $S_i^- = X_i^+ \oplus X_i^- \oplus Y_i$, whereas S_{i+1}^+ simply reproduce the redundant encoded input X_i^+, *i.e.*, $S_{i+1}^+ = X_i^+$.

Now we design N-bit approximate hybrid redundant adder and $(N \times N)$-bit approximate hybrid redundant multiplier using the above-discussed 1-bit approximate hybrid redundant adder. As discussed in Chapter in 3, in arithmetic operations, impact of errors in higher order bits is more severe as compared to the errors in lower order bits. Therefore, we perform higher order k-bit operations accurately using conventional hybrid redundant adder and lower order $(N-k)$-bit operations approximately using the proposed approximate hybrid redundant adder. We exploit these N-bit approximate hybrid redundant adder and $(N \times N)$-bit approximate hybrid redundant multiplier for designing the approximate hybrid redundant MAC unit which we call as "ApproxMAC".

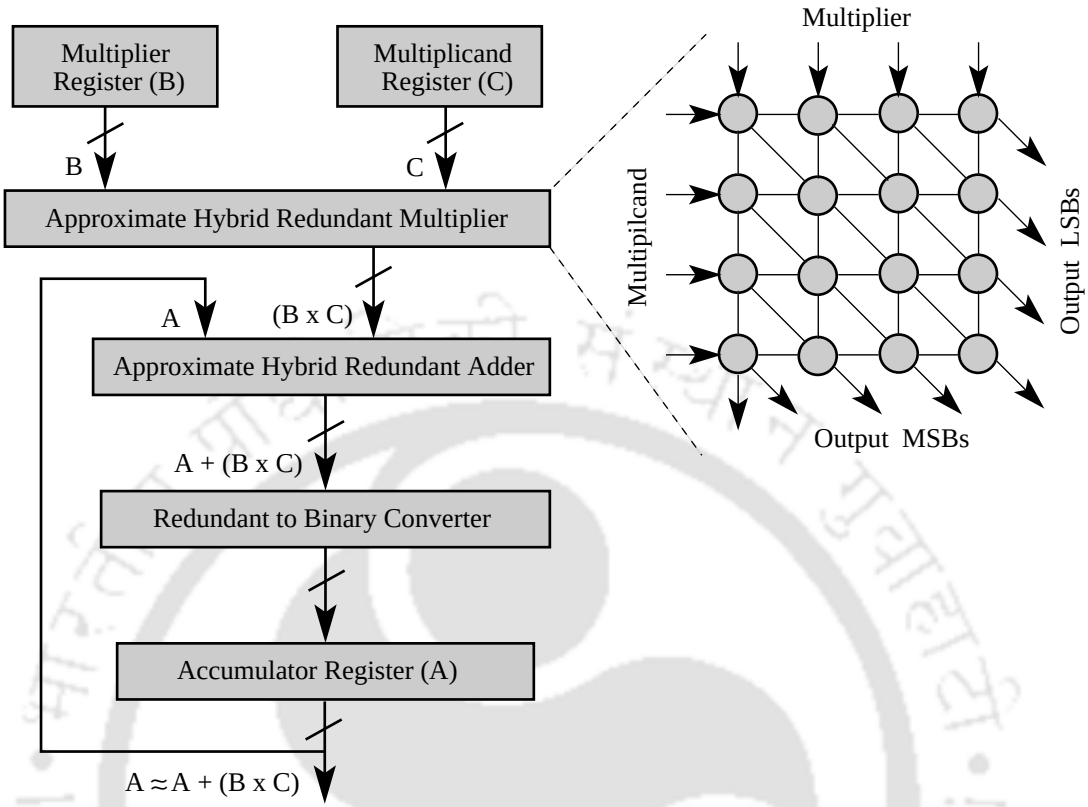


Figure 5.11: Block diagram of the proposed hybrid redundant ApproxMAC unit.

5.3.3 ApproxMAC Unit

The block diagram of the proposed ApproxMAC unit is shown in Fig. 5.11. ApproxMAC unit computes the approximate product of two numbers and adds the results approximately with the contents of an accumulator, $i.e.$, $A \approx A+(B \times C)$. It has a multiplier and a multiplicand registers to store the inputs B and C, respectively. As shown in the inset of Fig. 5.11, the ApproxMAC unit exploits a dedicated approximate hybrid redundant array multiplier to perform the multiplication operation. Next, an approximate hybrid redundant adder is used to compute $A + (B \times C)$. In order to maintain the consistency with hybrid redundant operations ($i.e.$, one of the operand should be in binary number system while the another operand should be in redundant number system), the output is converted back to the binary number system using a redundant-to-binary converter. Finally, an accumulator register is used to store the output for the next MAC operation.

Table 5.3: Comparison of (32×32)-bit ApproxMAC unit with binary and hybrid redundant MAC units.

Design Metrics	Binary MAC unit	Hybrid redundant MAC unit	ApproxMAC					
			$k = 56$	$k = 48$	$k = 40$	$k = 32$	$k = 24$	$k = 16$
Delay (ns)	69.774	14.841	13.762	12.781	11.464	10.114	9.721	8.609
Power (mW)	6.921	18.072	16.308	13.625	10.869	7.237	4.341	1.017
Area (μm^2)	7158.78	8109.37	8064.07	7837.55	7475.12	6976.78	6342.53	5617.67
MED	0.0000	0.0000	-5.8E+1	-1.8E+2	-3.8E+3	-1.5E+4	-5.2E+6	-1.3E+8
ACC_{amp} (avg.)	1.0000	1.0000	1.00001	1.00001	1.00001	1.00001	1.00016	1.02449
ACC_{inf} (avg.)	1.0000	1.0000	0.9602	0.8885	0.8209	0.7558	0.6968	0.6190

5.3.4 Results and Discussion

In order to evaluate the proposed approach over quality metrics, we implement (32×32)-bit ApproxMAC unit with different values of k in C/C++. We simulate each configuration individually for one billion pseudo-random inputs drawn from a sample space between 0 and $2^N - 1$. Further, for assessing the delay, power and area, we design (32×32)-bit ApproxMAC unit with different values of k using *Mentor Graphics Tanner schematic capture* [103]. We extract netlists from schematics and simulate the extracted netlists using *Synopsys HSPICE circuit simulator* [104] with PTM $32nm$ model files [105]. Our simulation results are tabulated in Table 5.3. It can be seen from Table 5.3 that as k decreases, the improvements in delay, power and area increase. However, for $k < 32$, quality metrics start degrading. ApproxMAC unit with $k = 32$ improves delay by 85.50% and 31.85%, power by -4.56% and 59.95%, and area by 2.54% and 13.96% $w.r.t.$ binary and hybrid redundant MAC units, respectively. As both hybrid redundant MAC unit and the proposed ApproxMAC unit render a carry-free addition, a marginal speedup is reported $w.r.t.$ hybrid redundant MAC unit. However, as intended, ApproxMAC unit provides a significant improvements in power and area $w.r.t.$ hybrid redundant MAC unit. Further, the overall performance, $i.e.$, PDAP of ApproxMAC unit is much better than binary and hybrid redundant MAC units. Note that $\text{Acc}_{mag} > 1$ and negative MEDs in Table 5.3 shows that ApproxMAC unit overestimates the outputs.

Further, in order to inspect the effectiveness of the proposed approach in real-life applications, we replace the conventional (32×32)-bit MAC unit in DCT and IDCT modules with (32×32)-bit ApproxMAC unit. Fig. 5.12 shows the output images after performing DCT and IDCT using

(a) (b) (c) (d)

Figure 5.12: Image processing results using (32×32)-bit ApproxMAC with: (a) $k = 48$; (b) $k = 40$; (c) $k = 32$; and (d) $k = 24$.

(32×32)-bit ApproxMAC unit with different values of k. It can be seen from Fig. 5.12 that for $k \geq 32$, visual quality loss to the output images is negligible. However, for $k \leq 24$, there is severe degradation in the image quality which is not favourable.

5.4 Summary

In this Chapter, we explored the effectiveness of approximate adders in cryptography applications, yield enhancement and designing other approximate arithmetic operations. SHA-1 is the most widely used cryptographic hash function and has many future aspects. The basic building block of SHA-1 has four modular-32 adders connected in series and it gets executed 80 times for every chunk. Therefore, the overall delay, power and area of SHA-1 are determined by modular-32 adders. In order to examine the effectiveness of approximate adders in cryptography applications, we replaced the conventional modular-32 adders with approximate modular-32 adders. We called the resulting approximate SHA-1 as "ApproxSHA-1". Our simulation results showed that ApproxSHA-1 fulfills all cryptographic hash function criteria. Further, almost for similar power consumption, ApproxSHA-1 provides $\approx 3\times$ speedup as compared to the conventional SHA-1. To the best of our knowledge, this is the first time that throughput of a SHA-1 reaches $\approx 2Gbps$ without pipeline.

Further, with the relentless scaling of CMOS technology, chip yield has become a matter of concern for the semiconductor industry due to the fact that profits are tied directly to the yield. We proposed analytical models to estimate the functional yield and parametric yield of approximate circuits. From the proposed analytical models, we observed that approximate data path modules can improve

the functional and parametric yields due to decrease in area and delay, respectively. We demonstrated this in respect to ApproxADDs. Our simulation results showed that 16-bit ApproxADD$v1$ improves the chip yield by 89.63%, 74.44% and 52.63%; and 16-bit ApproxADDv2 improves the chip yield by 79.08%, 65.81% and 43.72% for $k = 6$, 8 and 10, respectively.

We know that adders are the basic building blocks of all arithmetic operations. Therefore, approximate adders can further be utilized to design other approximate arithmetic operations. In this regard, we designed an approximate hybrid redundant MAC (ApproxMAC) unit using approximate hybrid redundant adders. We considered hybrid redundant number system as it has several advantages. Our results showed that the overall performance, $i.e.$, PDAP of ApproxMAC unit is much better than the binary and hybrid redundant MAC units. We demonstrated the likelihood of ApproxMAC unit in real-life applications through an image processing application.

In Chapter 3 and Chapter 4, we have evaluated the effectiveness of approximate adders in two different image processing applications. In this Chapter, we have examined the effectiveness of approximate adders in cryptography applications, yield enhancement and designing other approximate arithmetic operations. In the next Chapter, we conclude this thesis and discusses some very important design challenges and future aspects of approximate adders.

6

Conclusion and Future Aspects

Contents

In recent years, the concept of approximate computing is gaining prominence [23–30]. The key motivation behind approximate computing is the *error-resilient applications* [8, 30, 53–55]. Due to the property of error-resilience, final output in these applications need not be fully precise, rather an approximate output is equally acceptable. Accordingly, approximate computing trade-offs computation quality for computation efforts and provides high-performance energy-efficient software and hardware implementations for error-resilient applications.

6.1 Conclusion

Over the past decade, several research work have been proposed in approximate computing paradigm at different levels of abstraction. Meanwhile, at hardware level of abstraction, adders (being the fundamental and most widely used arithmetic operators) have attracted a lot of attention for approximation. There are two basic approaches used for designing the approximate adders: (i) AFA; and (ii) ESA. Both the design approaches have their own pros and cons. In this thesis, we discussed the significance, motivation, design philosophy, design approach and background of approximate adders. This can provide an elementary aid to the beginners to understand approximate adders in a systematic manner. Further, we analysed, designed, modeled and optimized both types of approximate adder. We also evaluated the effectiveness of approximate adders in multimedia applications, cryptography applications, yield enhancement and designing other approximate arithmetic operations. The brief conclusion of the proposed work in this thesis is as follows.

AFA-based Approximate Adders: We proposed four AFAs with key design objective that C_{out} should be independent of C_{in} with minimal ER. Using one of the proposed AFAs (the optimal one), we designed an N-bit approximate adder which we called as "ApproxADD". An emergent property of ApproxADD is that carries do not propagate in it, and thus, it provides bit-width-aware constant delay ($O(1)$). ApproxADD also improves power by 45.54% and area by 28.57% $w.r.t.$ RCA. Although ApproxADD provides a significant improvement in delay, power and area, it may not be preferred for some of the error-resilient applications because its: (i) ED is too high; and (ii) ER increases rapidly with bit-width. We exploited the concept of carry-lifetime and EDC logic for improving the ED and ER of ApproxADD. In this way, we introduced two more (improved) versions of ApproxADD –

ApproxADD$v1$ and ApproxADD$v2$. In addition to the designing of ApproxADDs, we provided a detailed analysis and analytical modeling to estimate accuracy, delay, power and area of ApproxADDs. For the ease of illustration, we discussed the proposed approach $w.r.t.$ RCA, however, we demonstrated that the proposed approach can be customized for any adder architecture. For some of the adder architectures ($e.g.$, RCA), the final version of the proposed approximate adder imposes minor power and area overheads, but it shows significant improvement in overall performance which is generally measured in terms of PDAP. Further, for state-of-the-art adder architectures ($e.g.$, BK, KS and Sk), the final version of the proposed approximate adder shows significant improvement in power and area also. We evaluated the efficiency of the proposed approach by comparing ApproxADD$v1$ and ApproxADD$v2$ with existing AFA-based approximate adders. Our results showed that as compared to the existing AFA-based approximate adders (ETA-I, LOA, SA), ApproxADDs provide smaller MRED and ER, and higher ACC_{amp} and ACC_{inf}. Further, for inspecting the effectiveness of the proposed approach in real-life applications, we demonstrated image compression and decompression by replacing the conventional addition operations in DCT and IDCT with 24-bit approximate adders. Our image processing results showed that ApproxADD$v2$ provides favorable image quality and better IQA metrics than the existing AFA-based approximate adders.

ESA-based Approximate Adders: From our analysis, we observed that computer simulations have several limitations. Therefore, we proposed analytical models to estimate the accuracy (ED, ER, MED and MSE), delay, power and area of ESAs. We validated the proposed analytical models by comparing them with simulation results as well as with existing work. The key features of the proposed analytical models are that: (i) They are generalized, $i.e.$, work for all possible configurations of an N-bit ESA, whereas the existing analytical models are for specific configurations; (ii) They are superior ($i.e.$, estimate more accurately) or at par to the existing analytical models. The proposed analytical models can assist designers for estimating the design metrics of any configuration of an N-bit ESA more accurately without going for the programming efforts and time-consuming simulations. Based on the proposed analytical models and simulation results, we presented some very important observations regarding the behavior of design metrics of ESAs.

Further, from the proposed analytical models, we observed that in an N-bit ESA, there exist

multiple configurations which exhibit similar accuracy, however, these configurations exhibit different delay, power and area. Therefore, for a given accuracy, to apriori select the optimal configurations which provide minimal delay, power and/or area is a challenging decision for designers. We presented an optimization framework which can assist designers in making such important decisions early in the design phase of ESAs. This leads designers for efficient, intelligent and goal oriented implementations of ESAs. We demonstrated the proposed optimization framework for single-objective optimization (delay, power or area) subjected to single constraint (PR). However, the proposed approach can be customized/extended for the other cases of optimization also. Further, we know that accuracy of an ESA does not depend on the adder architecture used to implement it, however, its delay, power and area depend significantly. Therefore, the optimal configurations vary with adder architectures used to implement the ESA. In order to cover a wide range of adders, we considered three types of adder architecture in our analysis: (i) Architectures having smaller area; (ii) Architectures having smaller delay; and (iii) Architectures having in-between delay and area.

Further, from the proposed analytical models, we observed that there is a scope of improvement in accuracy-effort curves of ESAs. The accuracy-effort curves of ESAs need to be improved so that their accuracy degrade in a graceful manner, and consequently, they provide more optimal delay-accuracy, power-accuracy and area-accuracy trade-offs. Intended to improve the accuracy-effort curves of ESAs, we proposed modifications. The crux of the proposed modifications is that the probability of C_{in} of sub-adders to be correct in original ESAs is $0.50 - 0.75$, whereas in case of modified ESAs, it is 0.75 without imposing any additional delay, power and area overheads. With this improvement, modified ESAs provide better accuracy-effort curves. Accordingly, as compared to the original ESAs: (i) For a given accuracy, modified ESAs provide lower delay, power and area; or (ii) For a given delay, power and area, modified ESAs provide higher accuracy. In addition to the accuracy enhancement, the proposed approach also provides improvements in delay and power when modified ESAs are used for designing the ACAs and VLSAs. We evaluated the effectiveness of the proposed approach in real-life applications by processing Lena image using original ESAs and modified ESAs. Our image processing results showed that the output images processed using modified ESAs are more precise than the output images processed using original ESAs. All together, modified ESAs are more efficient

(in terms of accuracy, delay, power and area). Thus, they can be preferred over original ESAs while designing the digital circuits for error-resilient applications.

Application of Approximate Adders: The applicability/feasibility of approximate adders need to be explored to maximize their benefits. Over the past decade, most of the research work on approximate adders have demonstrated their effectiveness in multimedia applications. We examined the likelihood of approximate adders in cryptography applications. We know that the overall delay, power and area of SHA-1 are determined by modular-32 adders. In order to explore the effectiveness of approximate adders in cryptography applications, we replaced the conventional modular-32 adders with ApproxADD. We called the resulting approximate SHA-1 as "ApproxSHA-1". We also presented a framework to determine the maximum approximation in SHA-1. Our results showed that ApproxSHA-1 fulfills all cryptographic hash function criteria. Further, as compared to the SHA-1, ApproxSHA-1 requires $0.74\times$ area. Almost for the similar power consumption, ApproxSHA-1 provides $3.12\times$ speedup over the SHA-1. To the best of our knowledge, this is the first time that throughput of a SHA-1 has reached $\approx 2Gbps$ without pipeline.

In addition to the delay, power and area benefits, approximate adders can also be used to address various technology/design related issues. In sub-nanometer regime, chip yield has become a matter of concern for the semiconductor industry due to the fact that profits are tied directly to the yield. We evaluated the effectiveness of approximate adders for yield enhancement. We proposed analytical models to estimate functional yield and parametric yield of approximate arithmetic circuits. From the proposed analytical models, we observed that approximate data path modules can improve the functional and parametric yields due to decrease in area and delay, respectively. We demonstrated this in respect to ApproxADDs. Our results showed that 16-bit ApproxADD$v1$ improves the chip yield by 89.63%, 74.44% and 52.63%; and 16-bit ApproxADDv2 improves the chip yield by 79.08%, 65.81% and 43.72% for $k = 6$, 8 and 10, respectively.

Further, we know that all basic arithmetic operations, such as addition, subtraction, multiplication and division use adders as the key components. Therefore, approximate adders can be utilized to design other approximate arithmetic operations. In order to explore the effectiveness of approximate adders for designing other approximate arithmetic operations, we designed an approximate hybrid

redundant MAC (ApproxMAC) unit using approximate hybrid redundant adders as the basic building blocks. We considered hybrid redundant number system as it has several advantages. Our results showed that the overall performance, $i.e.$, PDAP of ApproxMAC unit is much better than the binary and hybrid redundant MAC units. We also demonstrated the likelihood of ApproxMAC unit in real-life applications through an image processing application.

6.2 Future Aspects

The design concept of approximate adders (AFAs, ESAs, ACAs and VLSAs) came a long way and still has a long way to go. Based on the work proposed in this thesis, the key design challenges and future aspects regarding approximate adders are as follows.

(i) While designing the proposed AFAs, our objective is to obtain C_{out} independent of C_{in} with minimal ER. Note that we do not focus on the another fundamental quality metric, $i.e.$, ED. The proposed AFAs can be optimized in terms of ED by designing the approximate functions for S [1]. There can be total $2^{(2^3)} - 1 = 255$ approximate functions of S. These approximate functions of S can be clubbed with the approximate functions of C_{out} to find the optimal AFAs in terms of minimal ED, ER, MED and/or MSE.

(ii) We know that ACAs and VLSAs are two major future design aspects of ESAs. ESAs integrated with EDC logic can work as ACAs and VLSAs. The design approach of ACAs and VLSAs is similar, but ACAs configure accuracy during runtime based on the requirements of the applications, whereas VLSAs always provide accurate results. Recently, several EDC logic have been proposed in the literature for ACAs and VLSAs [14, 61, 65, 69, 70, 73, 85, 86]. Since EDC logic requires extra hardware, the major design challenge is to reduce delay, power and area overheads imposed by the EDC logic. To a certain extent, this can be achieved by first finding all possible EDs of an N-bit ESA (using the proposed analytical models) and then design an intelligent EDC logic to correct those EDs only [85].

(iii) We know that accuracy of an N-bit ESA depends on k and l. Therefore, the another approach to configure accuracy during runtime is to tune k and l dynamically [12, 64, 70]. However, tunning

of k and l also requires extra hardware. Therefore, the another major design challenge is to reduce delay, power and area overheads imposed due to tunning of k and l. To a certain extent, this can be achieved by first finding the optimal configurations of an N-bit ESA (using the proposed optimization framework) and then design the ESA for tunning k and l over a narrow range, rather than tunning them over a wider range.

(iv) In Chapter 4, we considered ESAs. Note that ESA is a specific category of segment based approximate adders in which each sub-adder (except the least significant sub-adder) has same value of k and l. We concern ourselves to ESAs because: (a) The design concept of ESA is the basis for segment based approximate adders; and (b) ESAs benefit from the design principle of regularity which reduces design complexity using similar blocks (sub-adders in case of ESAs). However, other types of segment based approximate adder are also possible. For example, it is possible that different sub-adders can have different values of k and l [62, 79, 83]. Further, it may be possible that for a given accuracy, a segment based approximate adder with different values of k and l can provide less delay, power and/or area than the ESAs. Therefore, the proposed work (analytical models, optimization and accuracy enhancement) can be extended for other types of segment based approximate adder.

(v) It has been observed that a very high degree of error-resilience ($\approx 83\%$) is prevalent in a broad spectrum of applications, including machine learning, web search, probabilistic and statistical analytics, scientific computing, image, audio and video processing, wireless communication, cryptography, etc. [8, 100, 101]. We explored the effectiveness of approximate adders in multimedia applications, cryptography applications, yield enhancement and designing ApproxMAC unit. In addition to these, the applicability/feasibility of approximate adders can be explored for state-of-the-art applications, to deal with various technology/design related issues and for designing other approximate arithmetic operations.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Bibliography

[1] D. Shin and S. K. Gupta, "A Re-design Technique for Datapath Modules in Error Tolerant Applications," in *17th Asian Test Symposium (ATS)*, Nov. 2008, pp. 431–437.

[2] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-computing Applications," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 4, pp. 850–862, April 2010.

[3] 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*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[4] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate XOR/XNOR-based Adders for Inexact Computing," in *13th IEEE Conference on Nanotechnology (IEEE-NANO)*, 2013, pp. 690–693.

[5] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of Low-power High-speed Truncation-error-tolerant Adder and its Application in Digital Signal Processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

[6] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. USA: Addison-Wesley Publishing Company, 2010.

[7] M. D. Hill and C. Kozyrakis, "Advancing Computer Systems without Technology Progress," in *ISAT outbrief, DARPA/ISAT Workshop*, March 2012.

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

[9] Electronic Design. Nanometer Yield Enhancement Begins in the Design Phase. [Online]. Available: https://www.electronicdesign.com/products/nanometer-yield-enhancement-begins-design-phase

[10] Dell Precision R7610 Rack Workstation. [Online]. Available: https://www.dell.com/learn/us/en/04/shared-content~data-sheets~en/documents~dell-precision-r7610-spec-sh

[11] J. Lin, Y. Hwang, M. Sheu, and C. Ho, "A novel high-speed and energy efficient 10-transistor full adder design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 5, pp. 1050–1059, May 2007.

[12] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On Reconfiguration-oriented Approximate Adder Design and its Application," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2013, pp. 48–54.

[13] D. Esposito, D. D. Caro, and A. G. M. Strollo, "Variable latency speculative parallel prefix adders for unsigned and signed operands," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 8, pp. 1200–1209, Aug. 2016.

[14] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, "Design of Voltage-scalable Meta-functions for Approximate Computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2011, pp. 1–6.

[15] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. USA: Morgan Kaufmann Publishers Inc., 1999.

[16] Z. Jiang and S. Gupta, "An ATPG for Threshold Testing: Obtaining Acceptable Yield in Future Processes," in *Proceedings of International Test Conference (ITC)*, 2002, pp. 824–833.

[17] W. Maly, "Cost of Silicon Viewed from VLSI Design Perspective," in *31st Design Automation Conference (DAC)*, June 1994, pp. 135–142.

[18] CEVA-DSP. [Online]. Available: https://www.ceva-dsp.com/product/ceva-xc4500/

[19] S. Mittal, "A Survey of Techniques for Approximate Computing," *ACM Computing Surveys*, vol. 48, no. 4, pp. 62:1–62:33, March 2016.

[20] K. Palem and A. Lingamneni, "Ten Years of Building Broken Chips: The Physics and Engineering of Inexact Computing," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, pp. 1–23, May 2013.

[21] A. Alaghi and J. P. Hayes, "Survey of Stochastic Computing," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, pp. 92:1–92:19, May 2013.

[22] N. Bombieri, M. Poncino, and G. Pravadelli, *Smart Systems Integration and Simulation*, 1st ed. Springer Publishing Company, Incorporated, 2016.

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

[24] A. Sampson, "Hardware and software for approximate computing," Ph.D. dissertation, University of Washington, 2015.

[25] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate Computing and the Quest for Computing Efficiency," in *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. ACM, 2015, pp. 120:1–120:6.

[26] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Invited: Cross-layer Approximate Computing: From Logic to Architectures," in *53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 1–6.

[27] M. Wyse, "Modeling Approximate Computing Techniques," University of Washington, Tech. Rep.

[28] A. Aponte-Moreno, A. Moncada, F. Restrepo-Calle, and C. Pedraza, "A Review of Approximate Computing Techniques Towards Fault Mitigation in HW/SW Systems," in *IEEE 19th Latin-American Test Symposium (LATS)*, March 2018, pp. 1–6.

[29] A. G. M. Strollo and D. Esposito, "Approximate Computing in the Nanoscale Era," in *International Conference on IC Design Technology (ICICDT)*, June 2018, pp. 21–24.

[30] F. Betzel, K. Khatamifard, H. Suresh, D. J. Lilja, J. Sartori, and U. Karpuzcu, "Approximate Communication: Techniques for Reducing Communication Bottlenecks in Large-scale Parallel Systems," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1:1–1:32, Jan. 2018.

[31] C. Liu, "Design and Analysis of Approximate Adders and Multipliers," Master's thesis, University of Alberta, Canada, 2014.

[32] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A Review, Classification, and Comparative Evaluation of Approximate Arithmetic Circuits," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 4, pp. 60:1–60:34, Aug. 2017.

[33] L. Sekanina, "Introduction to Approximate Computing: Embedded Tutorial," in *IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2016, pp. 1–6.

[34] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*, 1st ed. USA: Cambridge University Press, 2011.

[35] S. Arora, "The Approximability of NP-hard Problems," in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 337–348.

[36] B. S. Baker, "Approximation Algorithms for NP-complete Problems on Planar Graphs," *ACM*, vol. 41, no. 1, pp. 153–180, Jan. 1994.

[37] M. Schulte, J. Stine, and J. Jansen, "Reduced Power Dissipation through Truncated Multiplication," in *IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, Mar. 1999, pp. 61–69.

[38] E. Swartzlander, "Truncated Multiplication with Approximate Rounding," in *33rd Asilomar Conference on Signals, Systems and Computers*, vol. 2, Oct. 1999, pp. 1480–1483.

[39] M. Schulte and E. Swartzlander, "Truncated Multiplication with Correction Constant [for DSP]," in *Workshop on VLSI Signal Processing*, Oct. 1993, pp. 388–396.

[40] P. Krause and I. Polian, "Adaptive Voltage Over-scaling for Resilient Applications," in *Design, Automation & Test in Europe (DATE)*, March 2011, pp. 1–6.

[41] R. Hegde and N. Shanbhag, "A Voltage Overscaled Low-power Digital Filter IC," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 2, pp. 388–391, Feb. 2004.

[42] D. Jeon, M. Seok, Z. Zhang, D. Blaauw, and D. Sylvester, "Design Methodology for Voltage-overscaled Ultra-low-power Systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 12, pp. 952–956, Dec. 2012.

[43] K. Shi, D. Boland, and G. Constantinides, "Overclocking Datapath for Latency-error Tradeoff," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2013, pp. 2537–2540.

[44] G. Memik, M. Chowdhury, A. Mallik, and Y. Ismail, "Engineering Over-clocking: Reliability-performance Trade-offs for High-performance Register Files," in *International Conference on Dependable Systems and Networks*, June 2005, pp. 770–779.

[45] R. Duarte and C. S. Bouganis, "Zero-latency Datapath Error Correction Framework for Over-clocking DSP Applications on FPGAs," in *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec. 2014, pp. 1–7.

[46] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, "Approximate Storage in Solid-state Memories," in *46th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-46, 2013, pp. 25–36.

[47] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural Acceleration for General-purpose Approximate Programs," in *45th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-45, 2012, pp. 449–460.

[48] G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol. 38, no. 8, April. 1965.

[49] R. H. Dennard, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of Ion-implanted MOSFET's with Very Small Physical Dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, Oct. 1974.

[50] J. Rabaey, *Low Power Design Essentials*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[51] A. Gupta, D. E. Culler, and J. P. Singh, *Parallel Computer Architecture: A Hardware/Software Approach*, 1st ed. USA: Morgan Kaufmann Publishers Inc., 1997.

[52] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling," in *38th Annual International Symposium on Computer Architecture (ISCA)*, June 2011, pp. 365–376.

[53] D. D. Thaker, D. Franklin, J. Oliver, S. Biswas, D. Lockhart, T. Metodi, and F. T. Chong, "Characterization of Error-tolerant Applications when Protecting Control Data," in *IEEE International Symposium on Workload Characterization*, Oct. 2006, pp. 142–149.

[54] X. Li and D. Yeung, "Application-level correctness and its impact on fault tolerance," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2007, pp. 181–192.

[55] Z. Vasicek and L. Sekanina, "Evolutionary Approach to Approximate Digital Circuits Design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 432–444, June 2015.

[56] P. Albicocco, G. Cardarilli, A. Nannarelli, M. Petricca, and M. Re, "Imprecise Arithmetic for Low Power Image Processing," in *Conference Record of the 46th Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Nov. 2012, pp. 983–987.

[57] C. Allen, D. Langley, and J. Lyke, "Inexact Computing with Approximate Adder Application," in *IEEE National Aerospace and Electronics Conference (NAECON)*, June 2014, pp. 21–28.

[58] L. B. Soares, S. Bampi, and E. Costa, "Approximate Adder Synthesis for Area- and Energy-efficient FIR Filters in CMOS VLSI," in *IEEE 13th International New Circuits and Systems Conference (NEWCAS)*, June 2015, pp. 1–4.

[59] T. Zhang, W. Liu, E. McLarnon, M. O'Neill, and F. Lombardi, "Design of Majority Logic (ML) Based Approximate Full Adders," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[60] M. Rezaalipour, S. Tajasob, M. Dehyadegari, and M. N. Bojnordi, "DrAx: An Automatic Approach to Designing More Precise and Energy-efficient Approximate Adders," in *Real-Time and Embedded Systems and Technologies (RTEST)*, May 2018, pp. 8–15.

[61] A. Verma, P. Brisk, and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design," in *Design, Automation & Test in Europe (DATE)*, 2008, pp. 1250–1255.

[62] N. Zhu, W. L. Goh, and K. S. Yeo, "An Enhanced Low-power High-speed Adder for Error-tolerant Application," in *12th International Symposium on Integrated Circuits (ISIC)*, Dec. 2009, pp. 69–72.

[63] S. L. Lu, "Speeding up Processing with Approximation Circuits," *IEEE Transactions on Computers*, vol. 37, no. 3, pp. 67–73, 2004.

[64] J. Hu and W. Qian, "A New Approximate Adder with Low Relative Error and Correct Sign Calculation," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 1449–1454.

[65] A. Kahng and S. Kang, "Accuracy-configurable Adder for Approximate Arithmetic Designs," in *49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2012, pp. 820–825.

[66] Y. Kim, Y. Zhang, and P. Li, "An Energy Efficient Approximate Adder with Carry Skip for Error Resilient Neuromorphic VLSI Systems," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2013, pp. 130–137.

[67] X. Yang, Y. Xing, F. Qiao, Q. Wei, and H. Yang, "Approximate Adder with Hybrid Prediction and Error Compensation Technique," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2016, pp. 373–378.

[68] K. Du, P. Varman, and K. Mohanram, "High Performance Reliable Variable Latency Carry Select Addition," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2012, pp. 1257–1262.

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

[70] W. Xu, S. S. Sapatnekar, and J. Hu, "A Simple Yet Efficient Accuracy-configurable Adder Design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–14, 2018.

[71] T. Yang, T. Ukezono, and T. Sato, "A low-power yet high-speed configurable adder for approximate computing," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[72] K. Al-Maaitah, I. Qiqieh, A. Soltan, and A. Yakovlev, "Configurable-accuracy Approximate Adder Design with Light-weight Fast Convergence Error Recovery Circuit," in *IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, Oct. 2017, pp. 1–6.

[73] V. Benara and S. Purini, "Accurus: A Fast Convergence Technique for Accuracy Configurable Approximate Adder Circuits," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2016, pp. 577–582.

[74] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "EvoApproxSb: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2017, pp. 258–261.

[75] J. Liang, J. Han, and F. Lombardi, "New Metrics for the Reliability of Approximate and Probabilistic Adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, Sep. 2013.

[76] L. Li and H. Zhou, "On Error Modeling and Analysis of Approximate Adders," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2014, pp. 511–518.

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

[78] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, and J. Henkel, "Probabilistic Error Modeling for Approximate Adders," *IEEE Transactions on Computers*, vol. 66, no. 3, pp. 515–530, March 2017.

[79] M. A. Hanif, R. Hafiz, O. Hasan, and M. Shafique, "QuAd: Design and Analysis of Quality-Area Optimal Low-Latency Approximate Adders," in *54th ACM/EDAC/IEEE Annual Design Automation Conference (DAC)*, 2017.

[80] D. Celia, V. Vasudevan, and N. Chandrachoodan, "Optimizing Power-accuracy Trade-off in Approximate Adders," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 1488–1491.

[81] D. Celia, V. Vasudevan, and N. Chandrachoodan, "Probabilistic Error Modeling for Two-part Segmented Approximate Adders," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[82] A. G.-O. M. Weibrich, A. Najafi and G. Pay-Vay, "ATE-Accuracy Trade-offs for Approximate Adders and Multipliers in Pipelined Processor Datapaths," in *3rd Workshop on Approximate Computing (AxC18)*, 2018.

[83] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and Synthesis of Quality-energy Optimal Approximate Adders," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, Nov. 2012, pp. 728–735.

[84] A. Najafi, M. Weibrich, G. P. Vay, and A. Garcia-Ortiz, "Coherent Design of Hybrid Approximate Adders: Unified Design Framework and Metrics," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2018.

[85] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, and J. Henkel, "An Area-efficient Consolidated Configurable Error Correction for Approximate Hardware Accelerators," in *53rd ACM/EDAC/IEEE Annual Design Automation Conference (DAC)*, 2016, pp. 96:1–96:6.

[86] D. Esposito, D. D. Caro, and A. G. M. Strollo, "Variable Latency Speculative Parallel Prefix Adders for Unsigned and Signed Operands," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 8, pp. 1200–1209, Aug. 2016.

[87] C. Liu, J. Han, and F. Lombardi, "A Low-power, High-performance Approximate Multiplier with Configurable Partial Error Recovery," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–4.

[88] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate Radix-8 Booth Multipliers for Low-power and High-performance Operation," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2638–2644, Aug. 2016.

[89] M. Masadeh, O. Hasan, and S. Tahar, "Comparative Study of Approximate Multipliers," in *Great Lakes Symposium on VLSI (GLSVLSI)*, ser. GLSVLSI '18. ACM, 2018, pp. 415–418.

[90] The USC-SIPI Image Database. [Online]. Available: http://sipi.usc.edu/database/database.php

[91] J. Vanne, E. Aho, T. D. Hamalainen, and K. Kuusilinna, "A high-performance sum of absolute difference implementation for motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 7, pp. 876–883, July 2006.

[92] I. Fister Jr., X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A Brief Review of Nature-inspired Algorithms for Optimization," *Elektrotehniski vestnik*, vol. 80, no. 3, pp. 116–122, 2013.

[93] B. Preneel, "Cryptographic Hash Functions," *European Transactions on Telecommunications*, vol. 5, no. 4, pp. 431–448, 1994.

[94] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. UK: Oxford University Press, 2010.

[95] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, "Opportunities and Challenges for Better Than Worst-case Design," in *10th ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2005, pp. 2–7.

[96] H. A. Mahmoud and M. A. Bayoumi, "A 10-transistor low-power high-speed full adder cell," in *IEEE International Symposium on Circuits and Systems VLSI (ISCAS*, vol. 1, May 1999, pp. 43–46.

[97] M. B. Henry, "Emerging Power-gating Techniques for Low Power Digital Circuits," Ph.D. dissertation, Virginia Tech, USA, 2011.

[98] D. Sengupta, F. S. Snigdha, J. Hu, and S. S. Sapatnekar, "An Analytical Approach for Error PMF Characterization in Approximate Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2018.

[99] Y. Wu, Y. Li, X. Ge, Y. Gao, and W. Qian, "An Efficient Method for Calculating the Error Statistics of Block-based Approximate Adders," *IEEE Transactions on Computers*, pp. 1–1, 2018.

[100] D. D. Thaker, D. Franklin, J. Oliver, S. Biswas, D. Lockhart, T. Metodi, and F. T. Chong, "Characterization of Error-tolerant Applications when Protecting Control Data," in *IEEE International Symposium on Workload Characterization*, Oct. 2006, pp. 142–149.

[101] X. Li and D. Yeung, "Application-level Correctness and its Impact on Fault Tolerance," in *IEEE 13th International Symposium on High Performance Computer Architecture*, Feb. 2007, pp. 181–192.

[102] A. Kabbani, "Modeling and Optimization of Switching Power Dissipation in Static CMOS Circuits," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, April 2008, pp. 281–285.

[103] Tanner S-Edit Schematic Capture. [Online]. Available: https://www.mentor.com/tannereda/s-edit

[104] The Gold Standard for Accurate Circuit Simulation. [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Pages/default.aspx

[105] PTM. (2007) Predictive Technology Model (PTM). [Online]. Available: http://ptm.asu.edu/modelcard/32nm_MGK.pm

[106] R. P. Brent and H. Kung, "A Regular Layout for Parallel Adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260–264, 1982.

[107] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Transactions on Computers*, vol. C-22, no. 8, pp. 786–793, 1973.

[108] J. Sklansky, "Conditional-sum Addition Logic," *IRE Transactions on Electronic Computers*, vol. EC-9, no. 2, pp. 226–231, June 1960.

[109] M. Kudelka, "Image Quality Assessment," in *21st Annual Conference of Doctoral Students (WDS)*, May 2012, pp. 94–99.

[110] Z. Wang and A. C. Bovik, *Modern Image Quality Assessment*. Morgan & Claypool, 2006.

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

[112] P. Rabanal, I. Rodríguez, and F. Rubio, "Using River Formation Dynamics to Design Heuristic Algorithms," in *6th International Conference on Unconventional Computation*, 2007, pp. 163–177.

[113] J. Chen, K. J. Liu, and U.-V. Koc, *Design of Digital Video Coding Systems: A Complete Compressed Domain Approach*. New York, NY, USA: Marcel Dekker, Inc., 2001.

[114] M. Kudelka, "Image Quality Assessment, Annual Conference of Doctoral Students (WDS)," pp. 894–99, May 2012.

[115] The Statistics Portal. (2018) Retail E-commerce Sales Worldwide from 2014 to 2021. [Online]. Available: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/

[116] International Technology Roadmap for Semiconductors. [Online]. Available: http://www.itrs2.net/itrs-reports.html

[117] M. A. Breuer and S. K. Gupta, "Intelligible Testing," *Proceedings Int'l Workshop on Microprocessor Test and Verification*, 1999.

[118] M. Breuer, "Intelligible Test Techniques to Support Error-tolerance," in *13th Asian Test Symposium (ATS)*, Nov. 2004, pp. 386–393.

[119] S. Shahidi and S. Gupta, "ERTG: A Test Generator for Error-rate Testing," in *IEEE International Test Conference (ITC)*, Oct. 2007.

[120] B. Murphy, "Cost-size Optima of Monolithic Integrated Circuits," *Proceedings of the IEEE*, vol. 52, no. 12, pp. 1537–1545, Dec. 1964.

[121] A. Srivastava, D. Sylvester, and D. Blaauw, *Statistical Analysis and Optimization for VLSI: Timing and Power*. Springer, 2005.

List of Publications

Based on the work presented in this thesis, our key publications are listed as follows.

Journals:

(i) Sunil Dutt, Satyabrata Dash, Sukumar Nandi and Gaurav Trivedi, "Analysis, Modeling and Optimization of Equal Segment Based Approximate Adders", in *IEEE Transactions on Computers*, vol. 68, no. 3, pp. 314 – 330, 2019. doi: 10.1109/TC.2018.2871096

(ii) Sunil Dutt, Sukumar Nandi and Gaurav Trivedi, "Accuracy Enhancement of Equal Segment Based Approximate Adders", in *IET Computers & Digital Techniques*, vol. 12, no. 5, pp. 206 – 215, 2018. doi: 10.1049/iet-cdt.2017.0171

(iii) Sunil Dutt, Sukumar Nandi and Gaurav Trivedi, "Analysis and Design of Adders for Approximate Computing", in *ACM Transactions on Embedded Computing Systems*, vol. 17, no. 2, pp. 40:1 – 40:28, 2017. doi: https://doi.org/10.1145/3131274

Conference Proceedings:

(i) Sunil Dutt, Bikram Paul, Anshu Chauhan, Sukumar Nandi and Gaurav Trivedi, "Approxhash: Delay, Power and Area Optimized Approximate Hash Functions for Cryptography Applications", in *10th International Conference on Security of Information and Networks (SIN)*, Jaipur, India, 2017, pp. 291 – 294. doi: https://doi.org/10.1145/31 36825.3136858

(ii) Sunil Dutt, Sukumar Nandi and Gaurav Trivedi, "A Comparative Survey of Approximate Adders", in *26th International Conference Radioelektronika*, Kosice, Slovakia, 2016, pp. 61 – 65. doi: 10.1109/RADIOELEK.2016.7477392

(iii) Sunil Dutt, Harsh Patel, Sukumar Nandi and Gaurav Trivedi, "Exploring Approximate Computing for Yield Improvement via Re-design of Adders for Error-resilient Applications", in *29th*

International Conference on VLSI Design (VLSID), Kolkata, India, 2016, pp. 134 – 139. doi: 10.1109/VLSID.2016.101

(iv) Sunil Dutt, Anshu Chauhan, Rahul Bhadoriya, Sukumar Nandi and Gaurav Trivedi, "A High-Performance Energy-efficient Hybrid Redundant MAC for Error-resilient Applications", in *28th International Conference on VLSI Design (VLSID)*, Bangalore, India, 2015, pp. 351 – 356. doi: 10.1109/VLSID.2015.65