

RICE UNIVERSITY

**Optimization Techniques for Minimizing Energy
Consumption in Approximate Circuits**

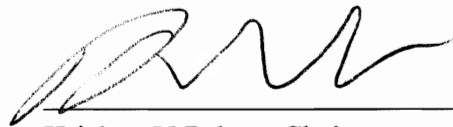
by

Kirthi Krishna Muntimadugu

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:



Krishna V Palem, Chair
Dept. of ECE, Rice University



Behnaam Aazhang
Dept. of ECE, Rice University



Richard G Baraniuk
Dept. of ECE, Rice University



Zvi Kedem
Courant Institute of Mathematical Sciences,
New York University

Houston, Texas

May, 2011

ABSTRACT

Optimization Techniques for Minimizing Energy Consumption in Approximate Circuits

by

Kirthi Krishna Muntimadugu

This work presents different global and local optimization techniques for designing “approximate” circuits which decrease energy consumption, one of the most important criteria in present day circuit design. The concept of “approximate” circuits which trades off energy consumption to output quality, thus creating a new dimension to the design space, is radically different from the conventional design principle in which all circuits operate correctly all the time. But efficient and intelligent designs have to be realized to tap its full potential. These techniques, which have not been explored till date, are based on a rigorous mathematical model and target to improve the output quality of a given circuit keeping the energy consumption to a minimum. They use the value of information and the architecture of the circuit to maximize efficiency. They have been applied to digital signal processing circuits to realize energy savings up to 2X the conventional value.

Acknowledgments

Several people have contributed to the successful completion of this dissertation that it would not be possible to acknowledge them all here. If I have failed to recognize someone's contribution by name please accept my sincere apologies for doing so. I am grateful to everyone who has helped me.

First, I would like to express my deepest gratitude to my adviser Prof. Krishna V. Palem for his vital support and encouragement throughout. His guidance and insights have been invaluable in developing this research. His fundamental work on the relationship between probability and energy consumption is the foundation for this dissertation. Our long discussions on various topics spanning from thermodynamics to mathematical modeling have been key in my development towards becoming a good researcher. I would like to convey my most sincere appreciation to my committee member, Prof. Zvi M. Kedem, who has been integral in developing the ideas proposed in this thesis. His assistance has been central to the development of the models and optimization techniques described here. I am eternally indebted to Prof. Vincent J. Mooney III for his painstaking effort to strengthen this work especially from the perspective of a core hardware engineer. His scrupulous attention to details has been extremely constructive in improving this research and writing.

I would like to thank my committee members Prof. Behnaam Aazhang and Prof. Richard Baraniuk for their valuable time and feedback which has been of immense help for extending this research further. I would like to acknowledge Prof. C. Sidney Burrus for his time and perspective which helped guide this research. I also want to thank Prof. Al Barr for the stimulating discussions that we had which helped me improve the scope and impact of this research. I would like to express my genuine gratitude to Dr. Lakshmi Chakrapani for his encouragement and counseling without which it would have been very difficult for

me to cope in such a competitive environment. I am very thankful for his feedback on my initial unstructured thoughts which helped me to develop them into solid ideas that could be implemented.

The exciting environment at Rice University and Nanyang Technological University has been most beneficial. I appreciate everyone who granted me an opportunity to work at these prestigious institutions.

I would like to express my special thanks to my friend and colleague, Avinash Lingam-
neni, for his continuing support and friendship over the past 10 years and collaboration in
many joint papers. I thank my previous colleagues, Scott Novich, Avani Devarasetty and
Phani Deepak Parasuramuni for their partnership in my research.

I would like to express my heartfelt thanks to Sravani Gullapalli for her friendship,
company, support and help throughout.

I cannot put into words my appreciation towards my parents, Sathyanarayana Munti-
madugu and Kavitha Muntimadugu for their love, blessings, encouragement, sacrifices and
support throughout my life. I really appreciate my brother, Prithvi Krishna Muntimadugu,
for his love and affection.

Finally, I thank God for my health and happiness.

Contents

Abstract	ii
Acknowledgments	iii
List of Illustrations	ix
List of Tables	xiii
1 Introduction and Related Work	1
1.1 Challenges in the World of Computing	1
1.2 The Fourth Dimension	3
1.3 Thesis Statement	6
1.4 Problem Statement	6
1.5 Key Contributions of this Research	7
1.6 The Shift from a Deterministic to a Probabilistic Mind-set	8
1.7 An Overview of Related Approaches	9
1.8 Thesis Organization	14
2 Philosophy of Inexact Circuit Design	16
2.1 Value of Information based Optimization	17
2.2 Energy Accuracy Tradeoff	18
2.2.1 Classification of tradeoff methods	19
2.2.2 Approximate circuit design	21
2.3 Key Contributions	22
2.4 Case Study : An Approximate Ripple Carry Adder	23
2.5 Simulation Results	27
2.5.1 The energy-error relationship of a ripple carry adder	29

2.5.2	Comparing alternate adder architectures and the $E - \mathbf{E}$ relationship	34
2.5.3	Applying the lessons learned from the adder to audio and image data in DFT	36
2.6	Principal Thesis	39

3 Modeling and Optimization of a Single Datapath Element: An

	Approximate Binary Adder	40
3.1	Key Developments	40
3.2	Technology Background	43
3.3	Discussion About Circuit Optimization And Our Target Problem	45
3.3.1	Our target problem: approximate adder supply allocation problem	45
3.3.2	Background about circuit optimization problems	46
3.4	RCA Terminology	47
3.5	RCA Delay Assumptions	53
3.6	Energy Modeling Assumptions	55
3.7	Error Model For An Approximate RCA	57
3.7.1	Description of a ripple carry adder	57
3.7.2	Modeling the error at the output of an approximate RCA	60
3.8	Efficient Evaluation of Average Error of an Approximate RCA	73
3.9	Energy Consumption Models	76
3.9.1	Energy model for an RCA	76
3.9.2	Energy model for an approximate RCA	78
3.9.3	Summary to Section 3.9	82
3.10	Minimizing Average Error of an Approximate RCA Using Geometric Programming	83
3.10.1	Formulation of an optimization problem	84
3.10.2	How to minimize average RCA error using geometric programming	86
3.10.3	Supply voltage binning	94

3.11	Simulation Framework for RCA Experimentation	96
3.12	RCA Experimental Results	98
3.12.1	Simulation results of 16-bit and 32-bit approximate ripple carry adders	98
3.12.2	Non-monotonic error rates	102
3.12.3	Approximate RCA FFT example	103
3.13	Additional Terminology for Carry Lookahead Adders	107
3.14	CLA Delay Assumptions	108
3.15	Energy and Error Models for an Approximate CLA Adder	109
3.15.1	Description of a carry lookahead adder	109
3.15.2	Energy consumption model for an approximate CLA	110
3.15.3	Modeling the error at the output of an approximate CLA	111
3.16	Efficient Evaluation and Minimization of Average Error in an Approximate CLA	117
3.17	Simulation Framework for CLA Experimentation	117
3.18	Simulation Results for CLAs	117
3.18.1	Simulation results of 16-bit and 32-bit approximate carry lookahead adders	118
3.19	Optimizing General Circuits	119
3.19.1	Model of a General Circuit	120
3.19.2	The Average Error Optimization Problem	122
3.20	Impact on Circuit Design	123
3.21	Principal Thesis	124

4 Modeling and Optimization of Dataflow Graph of Approximate

Adders	125
4.1 Primary Contributions	125
4.2 The Model	127

4.2.1	The graph based model	127
4.2.2	The energy optimization problem for our target dataflow graph of adders	129
4.3	The Single Resource Dataflow Energy-Error Optimization Method	130
4.3.1	A single approximate adder	130
4.3.2	Computing the significance of an adder	132
4.3.3	Multiple approximate adders	137
4.3.4	Case study: Ripple carry adder	138
4.3.5	Optimizing energy distribution	138
4.4	Summary of the Method and Extension to Other Adders	139
4.5	Optimizing Designs of DSP Primitives	141
4.5.1	Converting constant-number multipliers to adders and shifters	141
4.5.2	Approximate finite impulse response filter	142
4.5.3	Approximate fast Fourier transform	143
4.6	Simulation Framework and Results	145
4.6.1	Simulation framework	145
4.6.2	Results and comparisons	146
4.7	Impact on Circuit Design	150
4.8	Principal Thesis	150
5	Remarks and Future Directions	151
A	Effect of non-zero carry bits on error and energy models	154
	Bibliography	157

Illustrations

1.1	The energy consumption and accuracy tradeoff in a 16-bit ripple carry adder	4
1.2	Instance of trading accuracy to energy consumption in a fast Fourier transform for image processing	5
1.3	Overview of the classification of related approaches	10
2.1	Classification and characteristics of various currently known energy-accuracy tradeoff techniques	19
2.2	Carry chains and positions for addition of different Binary numbers	24
2.3	The change in the average magnitude of error with respect to the operating delay D	29
2.4	Average magnitude of error with respect to the energy of operation for $D = 0.8ns$ (in the range $d_{n-1} \leq D \leq 2d_{n-1} - \epsilon$)	30
2.5	Average magnitude of error with respect to the energy of operation for $D = 2.3ns$ (in the range $4d_{n-1} \leq D \leq 8d_{n-1}$)	30
2.6	A two point discrete Fourier transform	31
2.7	Average magnitude of error with respect to the energy of operation for $D = 2.0ns$ (in the range $d_0 \leq D \leq \sum_{i=0}^{n-1} d_i$)	31
2.8	Error magnitude and the relative frequency for (a) the uniformly voltage scaled case and (b) the non uniformly voltage scaled case	34
2.9	The $E - E$ relationship for a 16-bit ARCA using the BIVOS and the UVOS schemes, where $\rho = 2\text{ ns}$ and 4 ns	35

2.10	The relationship between energy and delay as the admitted error varies in an ARCA	35
2.11	The $E - E$ relationship for the ARCA, ACLA and ACSA architectures for an eager reading interval of 1 ns and 4 ns	36
2.12	The EDP advantages of BIVOS over UVOS in processing an image where the EDP gain of BIVOS based approach over a UVOS based approach is shown in the figure between the pairs of reconstructed images	38
2.13	The image obtained after applying DFT and then applying Inverse-DFT using conventional correctly operating adders, adders with BIVOS and adders with UVOS	38
3.1	Diagram of a 4-bit ripple carry adder	58
3.2	Diagram of a 1-bit full adder (1-bit FA)	58
3.3	An Example of a carry chain in a binary addition using an RCA	61
3.4	An Example of two contiguous carry chains in a binary addition using an RCA	61
3.5	An example to demonstrate that in an RCA it is possible that by increasing the supply voltages the accuracy of the sum is decreased.	64
3.6	Diagram of an n-bit ripple carry adder to describe the modeling of the critical path of a sum bit in a carry chain	66
3.7	Gate level diagram of a 3-bit ripple carry adder	69
3.8	A sample floorplan with 4 voltage islands based on the binned solution of a 3-bit RCA shown in Table 3.4	93
3.9	Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 16-bit ripple carry adder	100
3.10	Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 32-bit ripple carry adder	101

3.11	Average error and average energy consumption versus clock cycle time for a 16-bit approximate RCA with constant and uniform supply voltage of 1.2V	102
3.12	Average error versus energy consumption for a 16-bit approximate RCA with constant clock cycle time of 4E-10 sec	103
3.13	Flow graph of a complete decimation-in-time decomposition of a 8-point FFT	104
3.14	Graph-theoretic representation of an 8-point FFT	104
3.15	Images generated by (a) Case 1 (b) Case 2 (c) Case 3 and (d) Case 4	107
3.16	Diagram of a 4-bit carry lookahead adder	109
3.17	Diagram of a 1-bit Propagate-Generate full adder (1-bit PG-FA)	109
3.18	Diagram of a 4-bit Carry Look Ahead Block	110
3.19	Illustration of the paths and delay of the carry bit across a carry-lookahead adder	111
3.20	Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 16-bit carry lookahead adder	119
3.21	Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 32-bit carry lookahead adder	120
4.1	Example of a graph-theoretical representation of a circuit	128
4.2	An example of an implicit shifter	128
4.3	The two paths from A_1 to $A_4 (= O_1)$. Error of δ at A_1 , while propagating through the left path contributes an error of 4δ to error at O_1 and an error of 1δ while propagating through the right path. Thus the total error at O_1 is 5δ and the significance of $A_1 = 5\delta/\delta$.	134
4.4	A finite impulse response filter	141
4.5	Graph theoretical representation of a finite impulse response filter	142
4.6	Flow graph of a complete decimation-in-time decomposition of a 8-point FFT	143

4.7	Graph-theoretic representation of an 8-point FFT	143
4.8	Energy consumption vs. Average error for a 8-point FFT with local and global optimization	146
4.9	Energy consumption vs. Average error for a 16-point FFT with local and global optimization	147
4.10	Reconstructed images obtained after processing through a (a) conventional correct 8-point FFT (b) locally optimized approximate 8-point FFT (c) globally optimized approximate 8-point FFT	149
A.1	An example of a 5-bit binary addition using an RCA when the intermediate carry bits are assumed to non-zero at the beginning of the addition.	154

Tables

3.1	Propagation delay and average dynamic energy per transition of an XOR gate in 90nm process technology for various supply voltage values	57
3.2	Maximum and minimum propagation delays of the XOR gate and the MUX in 90nm technology	58
3.3	Maximum and minimum propagation delays of the XOR gate and the MUX in 90nm technology	81
3.4	Propagation delay and supply voltage values from the geometric program and corresponding binned supply voltage values for the gates in Fig. 3.7 . .	91
3.5	Maximum, minimum propagation delays and proportionality constants of the XOR gate and the MUX in 90nm technology	98
3.6	Summary of results of 16-bit and 32-bit approximate ripple carry adders . .	99
3.7	Summary of results of 16-bit and 32-bit approximate carry look ahead adders	118
4.1	Significance values of the adders in the graph shown in Fig. 4.1	137

To my parents who have always believed in me

Chapter 1

Introduction and Related Work

Tom Forester, talking about the information revolution, said “ *If the automobile and airplane business had developed like the computer business, a Rolls Royce would cost \$2.75 and would run for 3 million miles on one gallon of gas. And a Boeing 767 would cost just \$500 and would circle the globe in 20 minutes on five gallons of gas [1].* ”

The reason for such a truly extraordinary revolution can be attributed to many facts, but according to my opinion, one of the most important is electronic technology’s radical reformation from the use of vacuum tubes to current day’s 28nm transistors. This revolution is more popularly known as the Moore’s Law, predicted by one of the founders of Intel, Gordon Moore [2]. But this electronic revolution which has been going on for more than two decades has been possible only by facing numerous challenges. Today, this revolution is again facing some serious challenges. This thesis is an effort to describe an alternative to the conventional electronic design methodology so that some of these challenges can be overcome.

1.1 Challenges in the World of Computing

There are two important challenges that the world of computing is facing. Currently, the first challenge is due to the increasingly ubiquitous nature of the present day *portable* electronics ranging from mobile phones to GPS-based navigation devices. Portability demands lower energy consumption without compromising on the functionality. Also, demand for

low energy consuming, also referred to as *green design*, electronics [3] is gaining a lot of momentum. According to the 2008 International Technology Roadmap for Semiconductors (ITRS) “*Energy Consumption has become an increasingly important topic of public discussion in recent years because of global CO2 emission . . . In general, the ITRS documents the impressive trends and, more importantly, sets aggressive targets for future electronics energy efficiency, for example, computational energy/operation (per logic and per memory-bit state changes). The most detailed targets relate directly to semiconductor materials, process and device technologies, which form the bases of integrated-circuit manufacturing and components, respectively.*” [4]

The second challenge is manufacturing reliable and predictable electronic devices. Moore’s Law predicts that the number of transistors on a single die is going to increase at an exponential rate. This has been accomplished by decreasing the size of an individual transistor up to 20nm where particular layers such as the gate oxide layer is about 1.2 nm (equivalent to 5 atoms!). But engineering considerations on lithography have limitations of designing these tiny elements precisely which leads to hindrances like thermal noise, parametric variations and other device perturbations [5, 6, 7] which leads to *unreliable computing*. Again the 2008 ITRS report states as a long term challenge “*Dealing with fluctuations and statistical process variations*”. Also the report mentions that “*Increasing yield loss due to non-visual defects and process variations requires new approaches in methodologies, diagnostics and control.* [4]”

These two challenges have competing requirements in the design of a VLSI system. A straightforward method of lowering the energy consumption is to lower the supply voltage of the circuit. But this would lead to transistors behaving unreliably because noise becomes a dominant factor. To ensure reliability, techniques such as redundancy and majority voting [8] can be used. However these techniques tremendously increase the energy consumption of

the circuit. Thus conventional methods typically have contradictory results and do not offer a common solution to both energy consumption and reliable design.

1.2 The Fourth Dimension

The conventional electronic circuit design methodology utilizes three parameters in the tradeoff argument, the energy consumption of the circuit, the area occupied by the circuit and the speed at which the circuit is being operated. To face the challenges outlined previously, a radically new solution to introduce a new alternate dimension, the *accuracy* of the circuit, to the traditional design approach has been proposed recently. For the first time ever, Palem [9, 10], showed that noise (or randomness) can be exploited as a resource for low energy but still obtain useful computation.

Currently there are three different techniques, applicable in different scenarios, that use this novel fourth dimension in circuit design [11, 12, 13].

The first approach uses CMOS circuits that operate probabilistically due to noise [14, 11]. The concept of a probabilistic CMOS switch (PCMOS) was introduced where a PCMOS inverter is correct with a probability parameter $p \leq 1$. The probabilistic inverter has been characterized in detail in terms of the relation between its energy consumption per switching and its probability of correctness. These probabilistic inverters were then used to design bigger probabilistic gates which switch correctly with a probability of correctness. This work was later extended to develop a probabilistic boolean logic, because it was realized that conventional boolean logic was no more valid in the universe where devices are probabilistic and not deterministic [15]. In probabilistic boolean logic, the basic operators are defined with a probability of correctness, for example, an AND gate with a probability of correctness p is defined as \wedge_p . Further extending the foundational probabilistic boolean logic, a probabilistic arithmetic was developed, where the operators are also defined with a

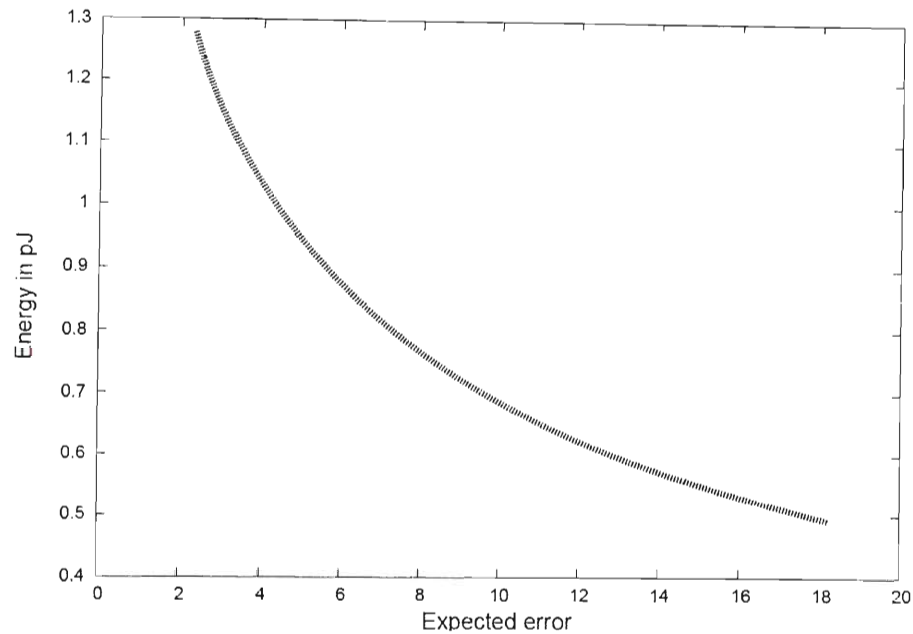


Figure 1.1 : The energy consumption and accuracy tradeoff in a 16-bit ripple carry adder

probability of correctness, such as $+p$.

This concept was applied by George et.al [11], who demonstrated the approach through which the *correctness* of arithmetic primitives may be traded off for energy consumed, while providing an acceptable or “good enough” solution. The principle that enables such an opportunity, is the relationship between energy and the probability of correctness in highly scaled, noise susceptible (future) CMOS technologies. Though impressive energy efficiencies were demonstrated empirically, and though such techniques are likely to be of great value in future noise susceptible CMOS and novel devices (such as molecular devices [16]), such techniques are not applicable to current day technology generations based on CMOS, where the noise levels are not sufficient to enable such a trade off.

To apply the concept of trading accuracy for energy consumption in present day de-

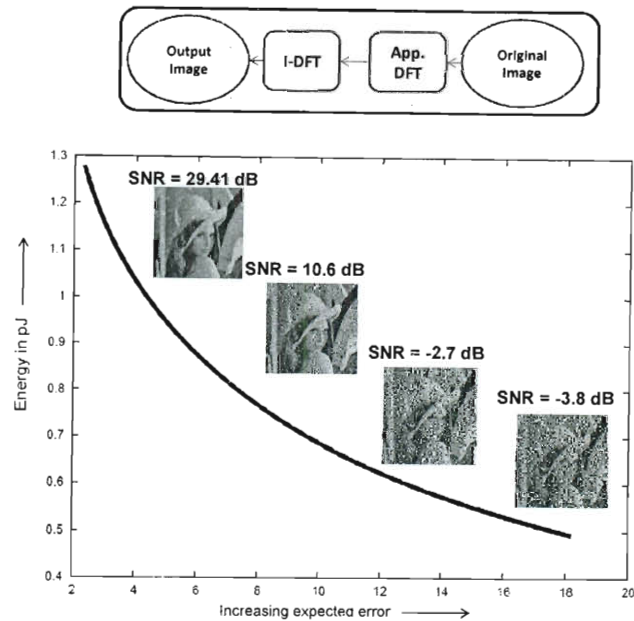


Figure 1.2 : Instance of trading accuracy to energy consumption in a fast Fourier transform for image processing

terministic circuits, we introduced the technique of approximate circuits using voltage overscaling [12]. Every digital circuit operates on a supply voltage. The delay of the circuit, which is the time the circuit takes to compute the output from the given inputs, is generally considered to be inversely proportional to the supply voltage of the circuit in CMOS. Hence the conventional design methodology is that a circuit should be operated at a speed which allows the outputs to be completely computed otherwise the intermediate values of the outputs, which could be different from the final outputs, might be considered as the actual outputs. But in certain applications where *closer to actual* outputs are sufficient, then the supply voltages of the circuit can be scaled down lower than the permitted value which is dependent on the operating speed. This results in *approximate* outputs and such circuits are referred to as *approximate circuits*. To provide further intuition, Figure 1.1 shows an instance where the energy consumption is being traded for accuracy. This figure shows the

results for a 16-bit *approximate* ripple carry adder. It can be inferred from the figure that as the energy consumption of the circuit is lowered, the error of the adder is increased (or the accuracy of the circuit is lowered). This behavior can be extrapolated from a single adder to designing a 16-point fast Fourier transform (FFT) for image processing. The results of an *approximate* FFT are shown in Figure 1.2. The images that are shown in the figure are the result of computing the FFT and then the inverse-FFT where the FFT part is approximate but the inverse is accurate. It can be observed from Figure 1.2 that there is an almost graceful degradation of quality of the image as energy consumption is lowered. A complete description of the simulation framework and assumptions behind these results are given in later chapters.

Another technique to trade accuracy for energy consumption is referred to as probabilistic pruning [13] where under-utilized parts of the circuit are systematically pruned or deleted thereby saving energy consumption.

The philosophy behind the concept of approximate circuits and a discussion of other techniques for implementing energy-accuracy tradeoffs are discussed in further detail in Chapter 2.

1.3 Thesis Statement

The supply voltages of a circuit can be lowered beyond acceptable limits leading to deterministic approximate circuits to achieve low energy computing.

1.4 Problem Statement

The two challenges that are mentioned in Section 1.1 are two opposing issues in the design of VLSI since solving one issue might deteriorate the other. Voltage overscaling leading to

approximate circuits presents one possible solution to both these issues that can be applied to current day deterministic designs. For efficient design and synthesis of these approximate circuits, characterization, analysis and optimization in terms of their accuracy of outputs, energy consumption and performance are essential.

1.5 Key Contributions of this Research

This dissertation presents the modeling and optimization of approximate CMOS circuits resulting from voltage overscaling. The resulting accuracy, energy consumption and performance of approximate circuits are analyzed and optimized. The use of these circuits for different popular DSP applications are also presented.

The following are the primary contributions of this research:

1. **Characterization and Efficient supply voltage allocation of an approximate adder**

A rigorous mathematical model for a general approximate adder with voltage overscaling has been provided. In doing so, the relationship between the resulting accuracy and its switching energy consumption is established. The characterization is based on theoretical models that have been developed, and later validated and supported by circuit simulations and measurement. This detailed characterization considers the following: (i) the type of the adder being used (ii) the effect of carry propagation and (iii) the constraints on the design parameters. This work is the first to provide an optimization model for minimizing error for a given energy budget for a general approximate adder when its supply voltages are overscaled. This optimization model is solved using a technique implemented using geometric programming, a variant of the more popular linear programming model. This work is therefore different from the previous characterization of approximate adders which establishes theoretical bounds on the expected gains through this approach. The characterization in this

work, besides being of interest, has utility since this can be used to design efficient approximate adders which later can be extended to bigger circuits.

2. **Modeling and Optimization of Dataflow Graph of Adders** A concrete model that depicts the behavior of propagation of errors in a dataflow graph (circuit) of adders and/or shifters is developed which is not constrained on the topology of the graph. This model can be used to represent many different circuits including some of the popular circuits used in digital signal processing, such as the finite impulse response filter or the fast Fourier transform. By analyzing the topology of the graph, the average error at the output(s) of the circuit is minimized by optimizing the distribution of energy across the different components of the circuit using the technique of Lagrange multipliers. This analysis, the first of its kind as per the author's knowledge, allows to design optimal approximate circuits that have minimal error for a given energy budget. Thus allowing the designer to build bigger and efficient optimal approximate circuits from smaller modules.

1.6 The Shift from a Deterministic to a Probabilistic Mind-set

Since 20000 BC when the first number representation, the tally marks [17] were introduced, to the era of the Egyptians and Sumerians in 3000 BC who invented some of the first arithmetic concepts [18] leading to Classical Physics, all phenomena in the universe were modeled deterministically. Boltzmann, a pioneer who introduced statistical considerations in physical phenomena, was in fact challenged by many of his colleagues on his statistical interpretation of the Second Law of Thermodynamics. It is very interesting to realize that the debate between deterministic and probabilistic modeling of physical phenomena persisted due to its theological dimension on the lines of "*Would God order His universe through*

Chance occurrences [19].” A much more popular quote in this topic is by Albert Einstein, “*God does not play dice with the universe [20].*” Whether randomness and chance really exist in the universe is a topic that deserves a lot more attention and is out of the scope of this thesis.

In this thesis I aim to demonstrate that randomness and chance in a given interpretation can be utilized in ways that are profitable and propitious to electronic design. The domain of electronics which deal with multimedia (audio and video) signal processing is one of the many areas which can tolerate inexact computing. The reason is that the quality of these electronics is evaluated only by human perception which can interpret useful information from slightly erroneous data. Thus lowering the accuracy would still be tolerable. This leads to a new design methodology in which the computations are approximate. A logical shift from reliable and deterministic design to probabilistic or stochastic design is inevitable. A prominent researcher from Intel Corp. Shekhar Borkar says “*We need to evolve from today’s deterministic design to probabilistic and statistical design for the future, comprehending variations, and optimizing for yield, performance, and power*” [21, 22, 23, 24].

1.7 An Overview of Related Approaches

The first work to address the issue of reliable computation in the presence of unreliable components was by Von Neumann [8]. Specifically, he showed that if the failure probability of a component $p_e \leq 0.0073$, then a circuit which fails less than half of the time can be designed using a cascade of three-input majority gates. There have been further developments on the bound on p_e , where Pippenger [25] and Feder [26] showed that if $p_e \geq 0.5 - 0.5k$ it is impossible to construct reliable networks using k -input gates.

There have been recent approaches that designers have adopted aimed at solving a similar problem that is being addressed in this thesis. To the best of the knowledge of the

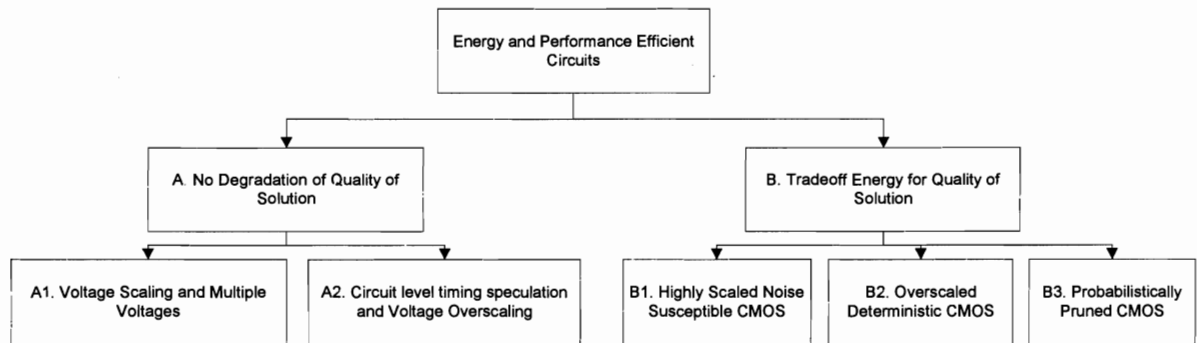


Figure 1.3 : Overview of the classification of related approaches

author, most of the approaches can be classified as shown in Figure 1.3.

The basic aim of all these approaches is to design energy and performance efficient circuits mostly targeted towards applications in digital signal processing. These can be divided into two categories

1. Techniques that always produce the correct answer and do not trade “accuracy for energy” at the output. This category also includes those approaches that might produce some errors during the computation but then specific error correction schemes are applied prior to the final output.
2. Techniques that aim to trade accuracy for energy under two assumptions, (i) If the application level impact is low and (ii) the savings in energy consumption are disproportionate to the loss in accuracy

Let us first discuss about some popular techniques in the first category. These can be divided into two sub-categories where the first set uses voltage scaling and multiple voltage levels (A.1) and the other set uses circuit level timing speculation and voltage overscaling for energy efficiency (A.2). Some of the techniques [27, 28] in the former set are *adaptive* which

means that the throughput of the circuit is based on the workload. Essentially the throughput is adjusted based on the input load at that particular time as opposed to operating at the worst case frequency at all times. Non-adaptive techniques typically operate the circuit at multiple voltages—critical paths of the circuit being implemented at higher voltages when compared to the non-critical paths—or rely on circuit implementation techniques like transistor sizing for energy efficiency [29]. Manzak and Chakrabarti [30] and Yeh et al. [31, 32] present techniques that are non-adaptive which operate the critical paths of the circuit at higher voltages than the non-critical paths and also use transistor sizing. This is similar to a biased voltage scaling but the bias is because of the time criticality of the output rather than the importance of the data that we use.

The techniques that use circuit level timing speculation and voltage overscaling are broadly known as the “Razor” approach [33, 34] championed by researchers at the University of Michigan which allows error at the circuit level, but only temporarily (for a clock cycle or two). In this case, errors are corrected by inserting delay in order to continue from a known “correct” logic state. In the case of overclocked circuits it is assumed that prior approaches to handle flip-flops and meta-stability issues [33, 34] can be used. A recent announcement in *Technology Review* (Published by MIT) [35] describes recent advances in error resilient circuit including a prototype chip designed by Tschanz et al. [36] at Intel that lets errors happen and then corrects them using less power overall. Shim et al. [37] show a design in which circuit level timing errors are not corrected at the circuit level, rather techniques borrowed from signal processing are used to correct such errors. Soft digital signal processing by Hegde and Shanbhag et al. [38, 37, 39] allows errors in the computation, but then uses digital noise tolerance schemes to attempt to correct these errors though partially in some cases.

In contrast, the other category of techniques, are ones which allow limited loss in accu-

racy of the output for substantial gains in savings of energy consumption *without* completely compensating for error. The first cases in which these techniques have been discussed are when the circuits are deep sub-micron and are highly susceptible to noise which will henceforth be referred to as *probabilistic circuits*(B.1). In this case, each component is considered to be unreliable similar to the analysis done by Von Neumann [8]. Rigorous modeling has been done in this case, to characterize the *energy-probability of error* relationship. The first of the results were by Palem [40, 41, 9] which established quantitatively that error can be traded for energy consumed and established that as the tolerable error limit is increased the energy saved increases exponentially. A rigorous quantitative analysis and model for a energy consumption versus probability of error relationship in a CMOS inverter was established by Korkmaz et al. [14]. This was further extended by George et al. [11] to show that if efficiently applied, such circuits can result in useful computation (shown in terms of digital signal processing applications) for arithmetic. The core concept was to identify that some components are more important than others in arithmetic circuits (such as adders or multipliers) and hence the investment of energy in these components should be biased with respect to the impact the particular component has on the output. Hence the concept of Biased VOLTage Scaling (BIVOS) was introduced.

The modeling of probabilistic circuits, required a complete overhaul of the basic boolean logic that is considered universally true for all boolean circuits. It was shown by Chakrapani et al. [15, 42] that conventional boolean logic fails in certain areas for circuits which behave probabilistically. Hence a radically novel *probabilistic boolean logic* was developed with all associated laws established (such as a probabilistic De-Morgan's law).

The modeling and analysis of probabilistic circuits would be applicable only when the transistors are highly scaled and hence the noise would be a significant factor. But at this time, noise is still a tolerable parameter and the devices are still deterministic in nature.

Hence we use voltage overscaling where because the critical path is violated there are errors in the output of the circuit. This kind of implementation will henceforth be called as *approximate circuits* (B.2). This technique was introduced by Lakshmi et al. [12] in which an approximate ripple carry adder was rigorously modeled and theoretical savings in energy consumption due to biased voltage scaling versus uniform voltage scaling was shown. An exception to these techniques is by Banerjee et al. [43] where in the specific case of a 2-dimensional discrete cosine transform they modify the circuit topology such that computations that are more important to output quality take shorter time than the ones that do not affect the output quality as much. So when they overclock the circuit, the computations that take shorter time would be computed correctly but the other ones might have errors in them.

A third form of energy-accuracy tradeoff is referred to as probabilistic pruning (B.3) where selective components in a circuit are deleted/pruned such that effect on the overall output is minimal and thus saving energy. This technique has been implemented on various arithmetic adders and substantial savings in energy-delay-area product have been shown by Lingamneni et al. [13].

A straightforward technique to reduce the energy consumption is power gating [44], which is simply cutting off power to *less important* circuitry. But in power gating we are neglecting the switched part of information completely, whereas in our approach we relatively invest based on the significance of the data.

This thesis focuses on this technique and provides modeling and optimization techniques for designing efficient low energy approximate circuits (B.2).

1.8 Thesis Organization

This thesis has been organized into 5 chapters. A brief description of their organization is given below.

- **CHAPTER 1:** We present the motivation behind designing approximate circuits and describe the effect of approximate circuits on the traditional circuit design paradigm. We also describe the thesis and problem statements along with the key contributions in this thesis. A philosophical viewpoint of the shift from a completely deterministic to a probabilistic way of thinking is presented. We also present a classification of related approaches and thereby position this work in current state of the art research.
- **CHAPTER 2:** We first describe the notion of “value of information” which is the primary foundation of the optimization techniques that are described in this thesis. We then present the classification of currently known techniques to trade accuracy for energy consumption and then elaborate on the particular tradeoff technique that is used in this thesis. To better describe our particular technique we pick the ripple carry adder as a case study and analyze its behavior in multiple ways.
- **CHAPTER 3:** This chapter presents the first key contribution of this thesis which is the modeling of error and energy for a given approximate adder design. We then use these models to formulate the problem of *optimization of an approximate adder* as a geometric program which is a type of convex optimization. Essentially we compute an allocation of supply voltages to different gates in the adder for minimal loss in accuracy for a given energy budget. We present the results of our optimization techniques on two different adder designs. We also present the simulation results of an 8-point FFT with image data to demonstrate the perceptual impact of approximate

circuits. We also present an extension of this modeling that can be applied to any general circuit design.

- **CHAPTER 4:** Our second key contribution in this thesis is the modeling of a network of “inexact” adders. As our primary target for inexact circuits is signal processing, we consider the primitive signal processing blocks such as finite impulse response filters and fast Fourier transforms (FFT) as our target. Most of these primitives can be modeled as a network of adders and shifters. Hence we developed a rigorous mathematical model to estimate the loss in accuracy at the output of such a network and consequently optimize the distribution of energy to minimize the loss in accuracy. We present simulation results for FFT circuits of different sizes to demonstrate the improvement due to our optimizations.
- **CHAPTER 5:** We finally present our conclusions and list a few directions in which this research can be further extended. Some of these extensions are key for a practical implementation of these optimization techniques and to also conform to the current design methodologies.

Chapter 2

Philosophy of Inexact Circuit Design

Inexact circuit design is a design philosophy where the conventional constraint of requiring 100% accuracy in circuits is relaxed. Fundamentally, this philosophy adds a fourth dimension of accuracy to the current 3-dimensional circuit design space spanning around power consumption, area and delay. This philosophy is applicable in the following two situations.

- The first situation is where the circuits are inherently “unreliable” and “probabilistic”. Increasing parameter variations, noise susceptibility and decreasing process sizes are causing CMOS devices to be non-deterministic. To address these issues and precisely model the effect of these probabilistic circuit elements, the metric of accuracy needs to be introduced into the entire circuit design framework.
- The second situation is where the circuits themselves are not probabilistic in nature but are deterministic, but the application does not demand 100% accuracy. In such cases, relaxing the very rigid constraint of accuracy can be used to decrease energy consumption which is one of the leading challenges in current day circuit design.

The first challenge in adopting an inexact circuit design methodology is to prove that circuits with less than 100% accuracy can still be used to perform useful computations in many applications which are energy constrained. There are two types of applications where inexact circuit design can be implemented. They are

- The first set of applications is where randomness is a required quality. For example, many encryption applications use pseudo-random number generators to produce ran-

dom numbers. But if there was a circuit that was inherently random then the extensive overhead of a pseudo-random number generator can be removed. It has been shown by Lakshmi et al. [45] that there are numerous such applications/algorithms which actually benefit from having a circuit which is inherently probabilistic. Some of the algorithms are Bayesian inference, Random neural networks, probabilistic cellular automata, and hyper-encryption.

- The second set of applications where accuracy can be relaxed is where a less than 100% correctness can be tolerated. This set primarily consists of traditional digital signal processing (DSP) applications whose output is consumed/judged by a human being. For example, consider a music player whose quality can be tuned based on how much battery power is left. In fact, many DSP algorithms are “approximate” in nature. The discrete Fourier transform, for example, represents the entire signal in the frequency spectrum with a few samples, thus introducing both quantization and sampling error. However, this still works because the human brain can interpret stimuli to the body’s senses and obtain useful information even if they are not completely accurate. A very common illustration of this fact is that a person can understand human speech even if there is extensive background noise corrupting the signal. It has also been clinically demonstrated that the human brain can spot known visual patterns even if interlaced with extensive noise. [46].

2.1 Value of Information based Optimization

The fourth dimension of accuracy is directly related to the information content that is being processed. Therefore to design inexact circuits efficiently, the consideration of the “value of information” is of paramount importance. Analogous to the case of conventional circuit

optimization techniques, inexact circuits also need to be optimized before the full advantage of this tradeoff can be realized. This optimization will revolve around the “value of information” that is being processed. To clarify the importance of “value of information” we present Example 1.

Example 1. Consider the addition of two simple decimal numbers, 743 and 8342. Assume that the accuracy of each digit position can be individually controlled. If there is an error of +1 in the unit’s positions, then the output of the addition would be 9086 (instead of 9085) leading to an error of +1. But consider the case where the exact same “magnitude” of error occurs in the hundred’s position instead of the unit’s position, then the output of the addition would be 9185 (instead of 9085) leading to an error of $+100 = +1 \times 100$. Therefore the same amount of error occurring at different places can lead to very different consequences.

□

This thesis will focus on the “value of information” as the key towards developing optimization techniques to design more efficient inexact circuits.

2.2 Energy Accuracy Tradeoff

A particular technique through which the new dimension of accuracy is introduced in circuits is referred to as an energy-accuracy tradeoff method. These techniques are based on practical circuit constraints and are also classified based on them. An energy-accuracy tradeoff method gives the circuit designer an approach to control the loss of accuracy in a given circuit for a particular energy constraint.

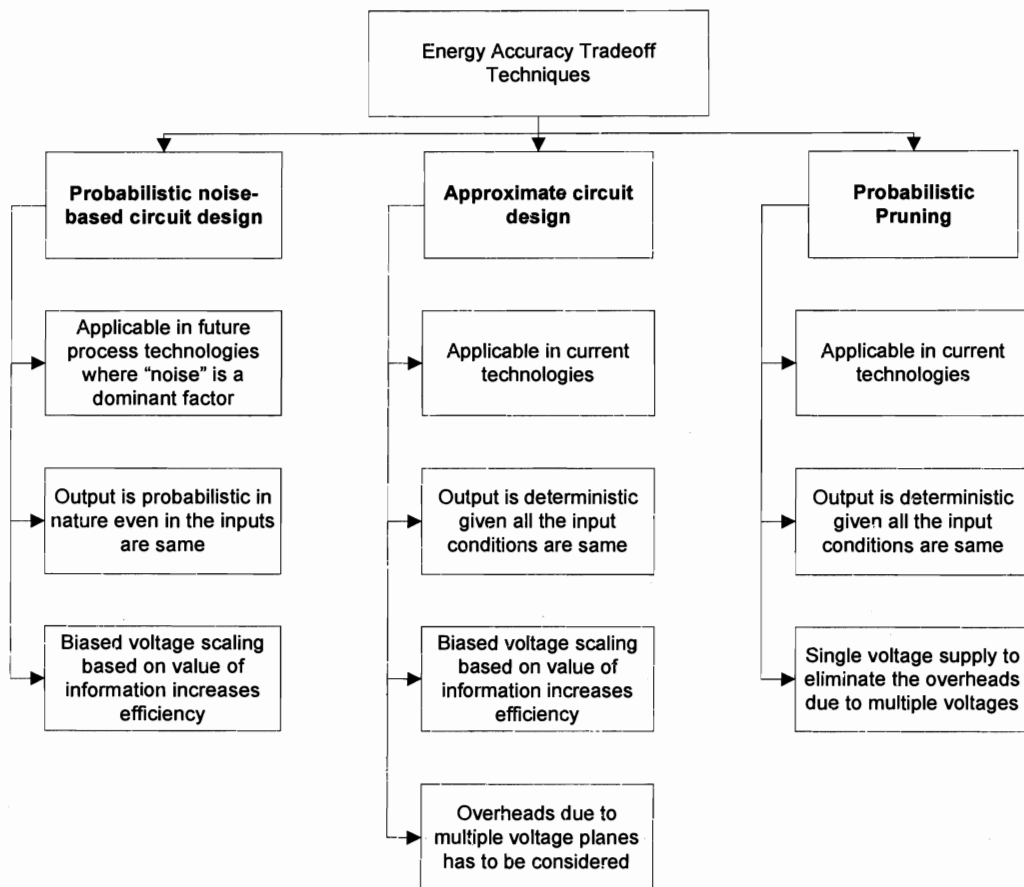


Figure 2.1 : Classification and characteristics of various currently known energy-accuracy tradeoff techniques

2.2.1 Classification of tradeoff methods

Currently known energy-accuracy tradeoff methods are classified based on their mode of implementation and characteristics. They are

- **Probabilistic noise-based circuit design:** This technique is applicable to circuits in “futuristic” process technologies where process variations and thermal noise are significant enough to make the individual circuit elements, non-deterministic. For example,

one of the current day approaches to combat the probabilistic effects of thermal noise and maintain the deterministic properties of circuits is to keep the supply voltages high enough such that thermal noise is “drowned”. But this inevitably leads to an increase in energy consumption. In applications where a “slight” loss in accuracy can be tolerated, the supply voltages could be lowered resulting in thermal noise based effects, which might lead to an “inexact and probabilistic output”. The phrase “inexact and probabilistic output” implies that the outputs of such circuits might not always be exactly equal to the correct output and the amount of deviation from the correct output also can vary. [40, 41, 9, 11]

- Approximate circuit design: This technique is applicable to current day deterministic circuits. One of the most rigorous constraints in current day circuit design is to adhere to timing constraints. Essentially the operating speed of a circuit is directly related to the critical path delay of the circuit. Thus if there is a constraint on the operating speed of a circuit, then, the operating supply voltage of the circuit is automatically fixed, which in turn, determines the energy consumption of the circuit. Analogous to noise-based circuit design, if the target application is such that accuracy of the output can be relaxed to a certain threshold then the conventional rigid binding between the supply voltage of the circuit and the operating frequency can be broken, then the supply voltage of the circuit can be lowered (reducing energy consumption) beyond the threshold at which the circuit is guaranteed to be correct. This might lead to an inexact output but, in contrast to the noise-based circuit design, is still deterministic. That means given the same set of input conditions the output will always be the same, albeit inexact. Such circuits are referred to as the *approximate circuits* and the process of operating at higher speeds than permitted by the critical path delay is referred to as *overclocking*. [12, 47, 48].

- **Probabilistic Pruning:** This technique is applicable to current day deterministic circuits. In this approach the components (which could be gates or collection of gates) and their associated wires are systematically “pruned” or deleted. The decision of which components are pruned is based on the probability of those components being active during circuit operation and the value of information being produced by those components. The extent of pruning is based on the constraints imposed on the maximum loss of accuracy that can be tolerated by the application. [13].

The classification and characteristics of the three energy-accuracy tradeoff methods is also presented in Fig. 2.1 for clarification and quick reference.

2.2.2 Approximate circuit design

Approximate circuits as described previously in Section 2.2.1, is a technique where the the relationship between the operating frequency of a circuit and its supply voltage is broken. Traditionally the operating frequency is based on the circuit’s critical path delay which is dependent on the circuit topology and its supply voltage. Assuming that the circuit topology is not being changed, the operating frequency is a function of the supply voltage. Thus if the operating frequency is fixed there is a minimum voltage threshold over which the circuit has to be operated to guarantee 100% accuracy in the output. In applications which can tolerate a loss in accuracy, the supply voltage of the circuit is lowered below the minimum threshold which might lead to an “approximate” output.

Also traditionally the entire circuit is operated at the same voltage because all parts of the circuit are computed correctly. But when we introduce accuracy as a parameter, as described in Section 2.1, the reduction in accuracy is affected by the value of information that is being computed by the particular part of the circuit. Hence a BIased VOLTage Scaling (BIVOS) scheme was proposed by Lakshmi et al. [12] wherein parts of the circuit which

compute information of higher value are supplied with a higher voltage and a lower voltage is supplied to other parts. We presented the following two key contributions in [12].

1. A rigorous theoretical foundation to quantify an energy-accuracy trade off at the arithmetic level which can be applied to current day deterministic electronic devices , and
2. A practical technique to realize this trade off in current day technology generations that is inspired by the foundational model.

A primary advantage of this technique is that it can be applied without any alteration to the existing circuit design and can be adjusted in real-time by modifying either the operating frequency or the supply voltages of the circuit. To further clarify the application of this technique and the resulting energy-accuracy tradeoff a case study of an approximate ripple carry adder is presented in Section 2.4.

2.3 Key Contributions

This chapter first presents the philosophy of inexact circuits and its classification. It will provide the intuition behind the approach of approximate circuits but will not discuss the theoretical foundations [12] for energy and probability of correctness in approximate circuits.

The key points that are presented are as follows

1. The scenarios in which inexact circuits can be designed are described. Then the two types of application domains in which inexact circuits can be used is presented.
2. The classification of current energy-tradeoff methods is shown and their respective characteristics are described.

3. The notion of value of information and its prime importance in the design of inexact circuits is presented.
4. A viable approach through which a tradeoff between energy consumption and accuracy of the output in current day technologies is demonstrated.
5. This approach uses a technique called overscaling, where the supply voltages are scaled much higher than allowed by the operating frequency thereby rendering the output *inexact* or *approximate*.
6. This technique is applied to arithmetic circuits, specifically a ripple carry adder to demonstrate its validity.
7. To demonstrate the tradeoff with respect to the three dimensions, analysis of different designs of approximate arithmetic adders are presented along with simulation results.
8. To demonstrate the tradeoff between energy consumption and output accuracy using approximate circuits, a two-dimensional fast Fourier transform is built using approximate adders.

2.4 Case Study : An Approximate Ripple Carry Adder

As an example consider the 8 bit binary addition of two numbers A and B where $A = 01001010$ and $B = 01000110$, where the least significant bit (the “first” bit) is written on the right and the most significant bit (the “eighth bit”) on the left. Consider the “ripple carry technique” to perform such an addition. As illustrated in Figure 2.2(i), it can be noticed that the addition of the second bit *generates* a carry, which is *propagated* to the third bit, which, in turn generates a carry, which when added to the fourth bit, generates a carry and sets the results of the addition of the fifth bit to 1. This will be referred to as a *carry chain*

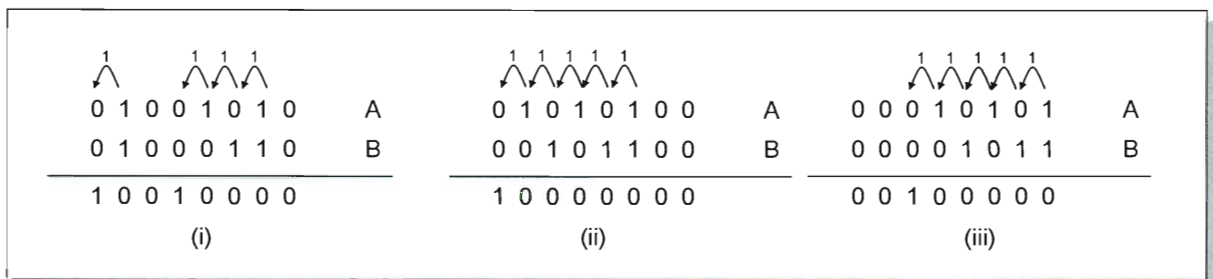


Figure 2.2 : Carry chains and positions for addition of different Binary numbers

of length 3 originating at *position 2*. In this example, there is also a carry chain of length 1, originating at position 7. If the delay for one addition combined with the wire delay (the delay for computing and propagating this carry to the next significant bit position) is d , in conventional circuit implementations of this adder, the total delay is taken as $D = 8d$ and the circuit operated at a frequency $1/D$. This is because, in the worst case (adding 10101011 to 01010101 for example) the carry chain is of length 8 and originates at position 1. The total delay of $8d$ determines the operating frequency $f = 1/8d$ of the circuit, which in turn, determines the operating voltage V , since in CMOS circuits, $V \propto 1/d$. This in turn, determines the energy consumed, where $E \propto V^2$.

Now, consider the addition operation from the previous paragraph when the entire adder circuit is operated at $V' = \frac{1}{2}V$ and hence $d' = 2d$ and let the frequency of operation be f . Since the individual carry computation and propagation delay is doubled, in the time $1/f$, any carry can propagate to only half as many (or $n/2$) bit positions. Now, since the length of the larger carry chain is $3 < 8/2$, the addition would be performed correctly with this voltage lowering. Moreover, in this case, since energy consumed by CMOS circuits is quadratically related to the operating voltage, the operating energy $E' = E/4$. Hence in this case, operating voltage can be halved, and the energy consumed can be improved by

a factor of 4, without compromising the correctness of operation. However, considering Figure 2.2 (ii) where the length of carry chain is 5, if the adder is operated at voltage V' , the addition would be performed incorrectly, with 01000000 as the computed result instead of 10000000, an error magnitude of 64. However this adder achieves a factor of 4 in energy efficiency when compared to the case where the adder is operated at voltage V wherein the answer is computed correctly. This illustrates the first principle [12]: *There is a trade off between energy consumption and error induced by propagation delay, in circuits which implement arithmetic operations, that can be exploited to garner energy savings*

Now, consider the case when the inputs are 00010101 and 00001011 (Figure 2.2(iii)). Even though the length of the carry chain is 5—the same length as the case described in Figure 2.2(ii)—since the carry chain originates in a less significant position and though the adder is operated at voltage V' , the error is significantly lower in this case—a magnitude of 16—for the same factor of 4 in energy savings when compared to the correct operation with voltage V . This case illustrates the second principle [12]: *Errors in bits of a higher value affect the quality of solution more than similar number of errors in bits of a lower value.*

Combining these two principles, the techniques to design and optimize approximate circuits will be presented in this dissertation. Specifically, in the case of a ripple carry adder, the full adders in the more significant position will be operated with a higher supply voltage, when compared to the full adders in the less significant positions. Thus the error rate or “probability” of error in the more significant positions is decreased when compared to the less significant positions. This approach [12] has been referred to as the “non uniformly voltage scaled addition”. A conventional (“uniform”) aggressive voltage scaling on the other hand, would operate all full adders with the same (reduced) supply voltage, and hence the error rate would be the same irrespective of the bit position.

In the work cited before [11], the methodology is ad-hoc, a mathematical model which

trades off the energy investment in each full adder to the expected *magnitude* of error is vital for a systematic exploration to guide the design of arithmetic units, and to provide an intellectual framework with sound foundation and to avoid this ad-hoc design. To this end, we had introduced in [12], a theoretical characterization of an adder which combines the energy consumption with the notion of the *value* of a bit—in the binary representation of numbers, more significant bits are of a higher value than less significant bits for example—and characterize the ability to trade off energy for *quality of solution* of arithmetic operations. The quality of solution has been quantified through the expected magnitude of error at the arithmetic level, and is determined by the error of the constituent bits as well as their values, the latter determined by their position significance. This detailed theoretical discussion is not presented in this dissertation.

To better present the intuition behind an approximate circuit, some empirical results are discussed in the next section for which we require to define some parameters for the circuit. We consider the following four parameters to describe our empirical results (of which three are independent):

1. n the width of the adder,
2. D^* the total computation time for each set of inputs for the adder
3. $f = 1/D$ the frequency of operation of the adder,
4. $v_i \propto \frac{1}{d_i}$, the supply voltage for each individual full adders.

For the purpose of illustration, all our simulations are in the context of a ripple carry design. While theoretically all the n full adders in an n bit adder can be operated at different supply voltages, practical considerations such as routing and the limitation of the number of

*This is independent of the inputs given or the total critical path delay

power planes will limit the number of realizable levels of supply voltage in any CMOS based implementation. Hence, in the theoretical foundation presented in [12] we had considered an additional independent parameter m , the number of “bins” or distinct levels of supply voltages. For example, in the uniformly voltage scaled case, there is $m = 1$ bin, since all full adders are operated at the same supply voltage. In future implementations, typically, $m \leq 4$ distinct levels of supply voltages are anticipated; a restriction that we will observe throughout the rest of this dissertation.

Based on this model, we had shown [12] that for a n bit addition with an operating frequency f and with the same energy consumption *the ratio of the expected magnitude of error between the uniform voltage scaling scheme and the non uniform voltage scaling scheme is $\Omega(2^{n/c})$ for a constant number of bins m* : as the number of bits $n \rightarrow \infty$ the gap (ratio) between *expected* error introduced by biased voltage scaling and conventional voltage scaling for an *equal amount of energy consumed* grows exponentially as $\Omega(2^{n/c})$. The expectation is determined by averaging over the inputs to the adder drawn uniformly from the set of all possible inputs.

2.5 Simulation Results

This section presents practical implementation techniques, provides simulation results and relates the theoretical foundation to the DSP domain. The erroneous behavior and energy efficiency of the circuits considered in this work are due to the mismatch of the critical path delay and the actual frequency with which the circuits are operated. The behavior of a ripple carry adder and later that of a circuit that implements the discrete Fourier transform (DFT) built from ripple carry adders is analyzed. The energy and (erroneous) behavior of the ripple carry adder as well as the DFT is characterized in the following three contexts:

- Case (a) correct operation (the baseline), where the operating frequency is less than that determined by the critical path delay of the circuit. In the context of a n bit ripple carry adder, if the delays of the individual full adders is represented by the uniform delay vector $D = \langle d_{n-1}, d_{n-2}, \dots, d_1, d_0 \rangle$, where $d_i = d$, this corresponds to the case where $D > nd$.
- Case (b) uniform aggressive voltage overscaling where all of the constituent full adders are operated with identical supply voltages, but the ripple carry adder itself is operated with a delay less than that of its critical path delay. That is, D is less than nd .
- Case (c) non uniform voltage overscaling, where some full adders are operated at a higher voltage than the others, and the entire ripple carry adder is operated with a delay less than that of its critical path delay. Hence, again D is less than the sum of the delays of the individual full adders. In our experiments, we consider delay vectors with 4 bins.

In all of these cases, the metrics of interest are the average energy consumed and the loss in accuracy. The loss in accuracy is quantified in terms of the average magnitude of (absolute) error in the case of a ripple carry adder, and the signal to noise ratio (SNR) in the context of the DFT. The comparisons of interest are

- Case (i) the energy consumption and the average magnitude of error of case (b) and case (c) above, when compared to the baseline.
- Case (ii) for identical energy and operating frequencies of case (b) and case (c), their average magnitude of errors (and SNR). This was defined to be the *relative magnitude* of error in the context of a ripple carry adder in the theoretical analysis.

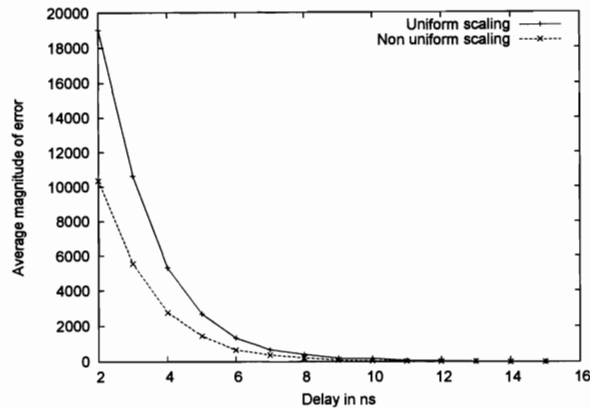


Figure 2.3 : The change in the average magnitude of error with respect to the operating delay D

- Case (iii) for identical operating frequency and error (or SNR) magnitude, the energy consumption of case (b) and case (c).

We call this relationship between error, energy and operating frequency, the energy-error relationship.

2.5.1 The energy-error relationship of a ripple carry adder

We consider the ripple carry adder to be composed of a chain of full adders of length n , where n is the length of the input operands. In our experiments, a ripple carry adder of length 16 is considered. The behavior of an individual full adder is modeled using HSPICE using TSMC $0.25\mu m$ libraries to derive the delay and energy consumption for supply voltages in the range $0.7v$ to $2.5v$. The behavior of a ripple carry adder is modeled using a C based behavioral simulator which utilizes the data generated by the HSPICE simulations.

To study the energy-error relationship for various operating frequencies and supply voltage configurations, the average error rate of the output and the expected magnitude of error for the three cases—case (a), case (b) and case (c) described above—is modeled

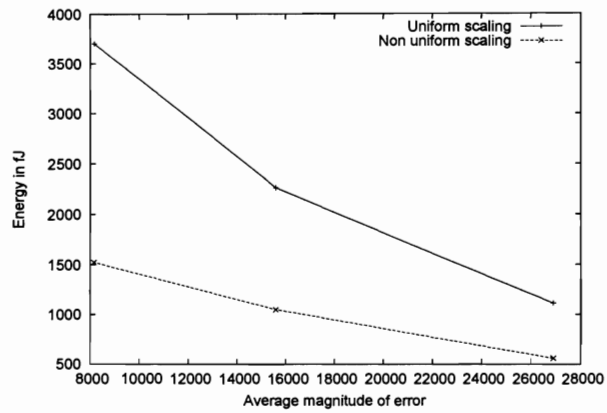


Figure 2.4 : Average magnitude of error with respect to the energy of operation for $D = 0.8ns$ (in the range $d_{n-1} \leq D \leq 2d_{n-1} - \epsilon$)

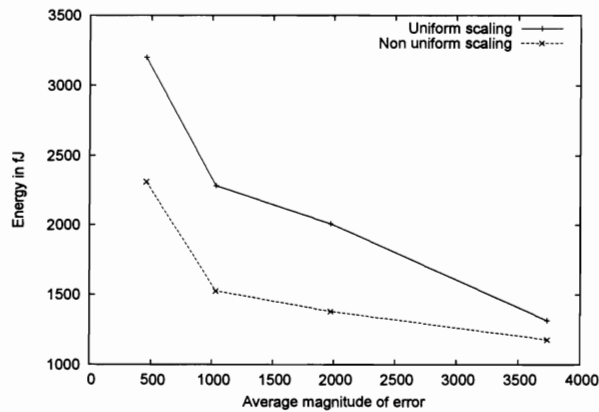


Figure 2.5 : Average magnitude of error with respect to the energy of operation for $D = 2.3ns$ (in the range $4d_{n-1} \leq D \leq 8d_{n-1}$)

using the behavioral simulator for varying operating frequencies. Three ranges of operating frequencies are of interest. If D is the delay vector of a non uniformly voltage scaled adder, and if d_{n-1} is the delay of the fastest full adder and d_0 is the delay of the slowest full adder, the first range of operating frequencies of interest is between d_{n-1} and $2d_{n-1} - \epsilon$, or equivalently, $d_{n-1} \leq D \leq 2d_{n-1} - \epsilon$. We shall refer to this case as the “first range of operating

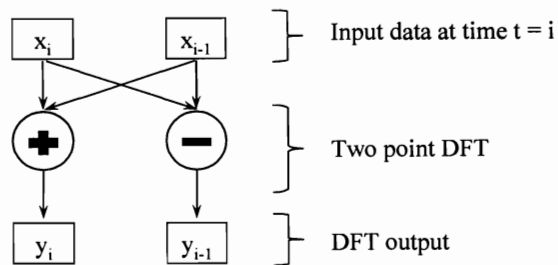


Figure 2.6 : A two point discrete Fourier transform

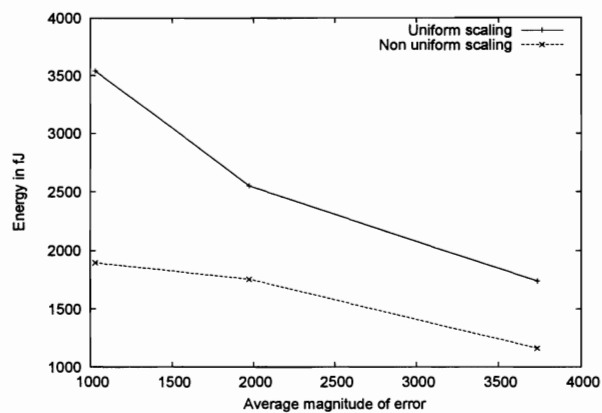


Figure 2.7 : Average magnitude of error with respect to the energy of operation for $D = 2.0ns$ (in the range $d_0 \leq D \leq \sum_{i=0}^{n-1} d_i$)

frequencies". Similarly a second range of interest is the case where $4d_{n-1} \leq D \leq 8d_{n-1}$ and finally the range of operating frequencies such that $d_0 \leq D \leq \sum_{i=0}^{n-1} d_i$.

For these three ranges of operating frequencies, the energy consumption and error magnitude of a 16-bit ripple carry adder is derived for various voltage allocations, where a voltage allocation refers to a set of voltages assigned to different full adders in the 16-bit ripple carry adder. To compute the average magnitude of error, this adder is operated on 10000 pairs of numbers derived from a uniform distribution.

Results and discussion

In all the three ranges, we observe certain general trends. For a specific voltage configuration, as illustrated in Figure 2.3, the average magnitude of error decreases with decreasing operating frequency. For a specific voltage configuration, the average energy consumption increases with operating delay as higher operating delays allow for more transitions in the constituent full adders as carries propagate across positions. Similarly, as illustrated in Figure 2.4, Figure 2.5 and Figure 2.7, for a fixed operating frequency, the magnitude of error decreases with increasing energy investment, since increasing the energy investment decreases the critical path delay. For a fixed operating frequency and energy consumption, the non uniformly voltage scaled adder, achieves a lower magnitude of error than the uniformly voltage scaled adder, since errors are confined to the lower order bits and hence are of a lesser magnitude. A histogram which compares the magnitude of errors and their relative frequency of occurrence is presented in Figure 2.8(a) and (b), which shows that in the non uniformly voltage scaled case, errors of a lesser magnitude are quite frequent, whereas error of a higher magnitude are quite rare.

Considering the first range of operating frequencies where $d_{n-1} \leq D \leq 2d_{n-1} - \epsilon$ and referring back to Figure 2.4, the non uniformly overscaled case (case (c)) achieves up to a 2.43x reduction in energy consumption when compared to uniformly scaled configuration (case (b)) for similar error magnitude of 8156.63 and operating frequency of $0.8ns$. When the operating frequency is in the first range, where $d_{n-1} \leq D \leq 2d_{n-1} - \epsilon$, the ripple carry adder yields the best energy savings over the baseline, since fastest full adders switch only once and the rest may not have completed even one computation. In this context, since the operating frequency is much higher than that determined by the critical path delay—the ratio of the critical path delay to the operating delay D is 23.5 in the case considered in Figure 2.4—the average (absolute) magnitude of error when compared to the baseline can

be as high as about 26894.

For operating frequencies in the second range, when the delay of operation of the circuit $4d_{n-1} \leq D \leq 8d_{n-1}$ —and is greater than the sum of the delays of the full adders in the fastest bin in the non-uniformly overscaled case—the non-uniformly voltage overscaled case has an energy savings of about 1.49x over the uniformly overscaled configuration for similar error magnitude of 1030. Since in this case the frequency of operation is lesser than the case considered above—in the case considered in Figure 2.5, the ratio of the critical path delay to the operating delay is about 8.5—we note that the lowest average magnitude of (absolute) error that can be achieved has decreased as shown in Figure 2.5.

Finally, for the third range of operating frequencies $d_0 \leq D \leq \sum_{i=0}^{n-1} d_i$, considering the faster operation where $D = 2ns$, case (c) achieves a 1.86x reduction in energy consumption over case (b). Since the operating frequency is higher than that of the representative example considered above, for a similar error magnitude of about 1030, the energy consumption of case (b) as well as case (c) is higher.

To summarize, the relationship between energy consumed (E) to compute a 16-bit addition and its associated expected error \mathbf{E} , henceforth referred to as the $E - \mathbf{E}$ relationship, is shown in Fig. 2.9 for the BIVOS as well as the UVOS cases. At an expected error of 17%, a BIVOS adder operating in 2 ns is 3.25X faster and consumes 3.8X lower energy when compared to a conventional adder, i.e., the BIVOS based adder is 12.3X more efficient in terms of EDP.

In Fig. 2.10, we present the relationship between energy (E) and delay (\mathcal{E}), henceforth referred to as the $E - \mathcal{E}$ relationship. In this figure, we present the relationship for two different error values, 5% and 15%.

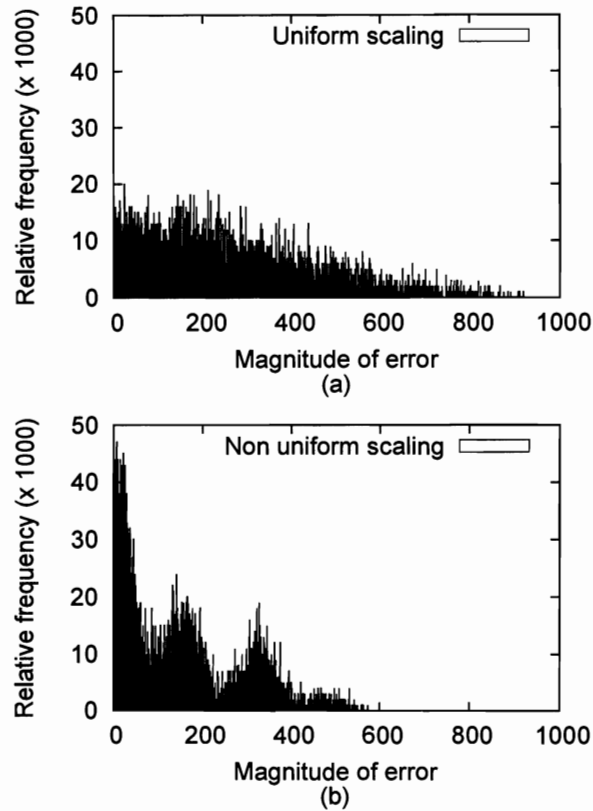


Figure 2.8 : Error magnitude and the relative frequency for (a) the uniformly voltage scaled case and (b) the non uniformly voltage scaled case

2.5.2 Comparing alternate adder architectures and the $E - E$ relationship

We will now compare the behavior of an ARCA to alternate adder architectures, specifically Approximate Carry Skip Adder (ACSA) and Approximate Carry Look-ahead Adder (ACLA). Consider the manner in which the behavior of adders in Fig. 2.11 vary as we increase the operating speed from 1 ns to 4 ns. The basic observation is that for a fixed amount of error, while an ARCA is indeed the most energy efficient adder design at 1 ns, its relative advantage decreases with increasing \mathcal{E} . This is because when \mathcal{E} increases, then the ACSA and ACLA propagate the carry information more than an ARCA for a relatively small additional energy

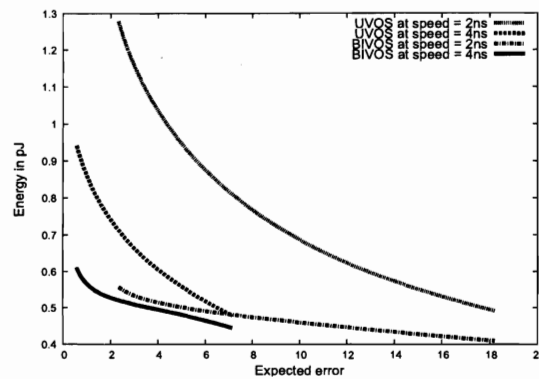


Figure 2.9 : The $E - \mathcal{E}$ relationship for a 16-bit ARCA using the BIVOS and the UVOS schemes, where $\rho = 2 \text{ ns}$ and 4 ns

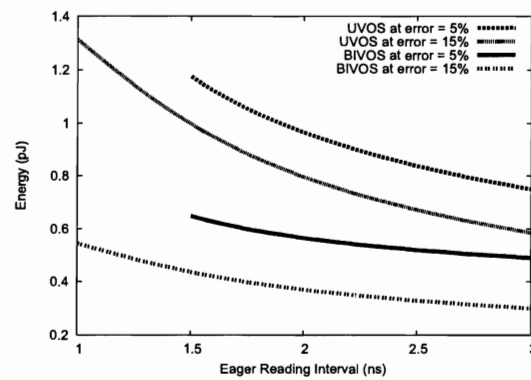


Figure 2.10 : The relationship between energy and delay as the admitted error varies in an ARCA

investment. And thus the ACSA and ACLA have increasingly lower error. *Thus, at higher \mathcal{E} when the advantage of the additional circuitry is utilized completely, the ACSA performs the best, followed by the ACLA, with the ARCA trailing both of them.* This represents a reversal of relative energy and EDP efficiency of the three architectures in going from an \mathcal{E} equals 1 ns to \mathcal{E} equals 4 ns .

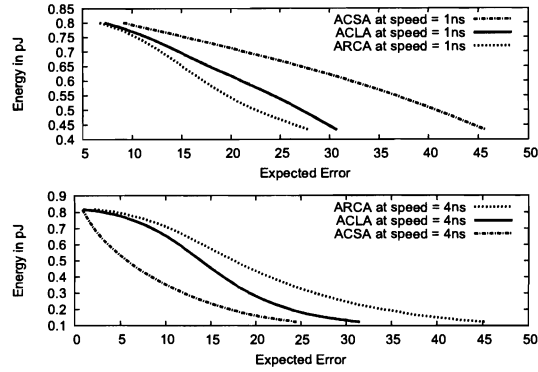


Figure 2.11 : The $E - \mathbf{E}$ relationship for the ARCA, ACLA and ACSA architectures for an eager reading interval of 1 ns and 4 ns

2.5.3 Applying the lessons learned from the adder to audio and image data in DFT

To demonstrate the utility of our voltage overscaling technique, we build a circuit which implements the discrete Fourier transform. Discrete Fourier transform is a mathematical technique that transforms a signal in time domain to the frequency domain. The input is a sequence of n complex numbers x_0, x_1, \dots, x_{n-1} and is transformed into the sequence of n complex numbers y_0, y_1, \dots, y_{n-1} , where for $0 \leq k \leq n - 1$,

$$y_k = \sum_{l=0}^{n-1} x_l e^{-\frac{2\pi jkl}{n}} \quad (2.1)$$

We have considered a 2-point DFT, where each DFT operation is performed on two subsequent values in time. Hence the DFT formula reduces to

$$y_0 = x_0 + x_1 \quad \text{and} \quad y_1 = x_0 - x_1 \quad (2.2)$$

and is illustrated in Figure 2.6

To study the energy and signal to noise ratio (SNR) of DFT operations, the behavioral

simulator is extended to simulate a DFT circuit and this circuit is simulated using representative data derived from audio files and images. Each simulation of a DFT described in this section consists of about 1000000 operations of the individual full adders, and the frequency of operation of the circuit is confined to the third range of operating frequencies. In the context of case (b) and case (c) above, after the DFT is computed the inverse DFT of the output is computed using a conventional correct technique. This is compared with the original input to compute the signal to noise ratio SNR. Since 2^s complement addition is present in the DFT data and since the operating frequency we consider is less, for increasing the efficiency of simulation, the simulation is performed in the granularity of a full adder and the internal state of a full adder as characterized by the signal values of the individual gates is not maintained.

Results and discussions

Based on the insights obtained from the experiments with the ripple carry adder, which indicate the conditions under which the greatest energy savings of case (c) over case (b) can be achieved for a low average magnitude of error of case (c) over the baseline, we perform the DFT experiment to quantify the energy savings as well as the improvement of SNR of the non-uniformly voltage overscaled case over a uniformly overscaled case.

For a operating delay of $12ns$ for the entire circuit which implements the DFT operation, in the context of the DFT, the non uniformly voltage overscaled case achieves a SNR of $19.63dB$ whereas for identical operating frequency and energy consumption, the signal to noise ratio in the context of uniform aggressive voltage overscaling is $5.79dB$. Thus non-uniform voltage overscaling yields an improvement of 3.4x in the SNR when compared to conventional uniform voltage scaling. If the energy investment in the non-uniform voltage scaling context is lowered such that its SNR matches that of case (b), the non uniformly

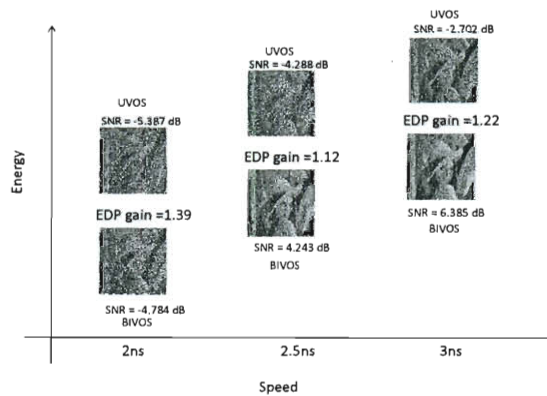


Figure 2.12 : The EDP advantages of BIVOS over UVOS in processing an image where the EDP gain of BIVOS based approach over a UVOS based approach is shown in the figure between the pairs of reconstructed images

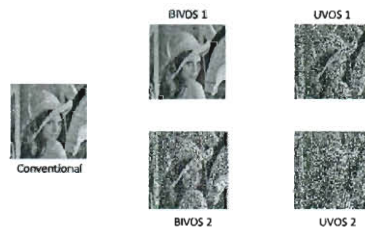


Figure 2.13 : The image obtained after applying DFT and then applying Inverse-DFT using conventional correctly operating adders, adders with BIVOS and adders with UVOS

overscaled technique achieves a 27.2% savings in energy consumption when compared to the uniformly scaled case.

The significance of approximate arithmetic as a viable approach to realizing energy savings will become apparent from Fig. 2.12 and Fig. 2.13. The images in Fig. 2.12 demonstrate the following two ideas

- As the operating delay is decreased, that is, the time provided to the circuits to compute the outputs is decreased the visual quality gradually degrades.

- Also a biased voltage allocation scheme for the same energy consumption as an uniform voltage allocation scheme has a higher output quality since it takes the notion of *value of information* into account.

In Fig. 2.13, the images labeled as BIVOS 1, BIVOS 2, UVOS 1 and UVOS 2 correspond to different voltage biasing schemes used for their respective computation. Going from a completely correct computation (which is labeled Conventional to that using BIVOS (labeled BIVOS 1 and BIVOS 2 in Fig. 2.13) an EDP savings of 1.3X and 1.75X respectively was achieved with an associated *signal to noise ratio* (SNR) of 29.41 dB and 1.4 dB, while preserving the computational speed or performance. In contrast and with similar EDP values, a UVOS based approach (labeled as UVOS 1 and UVOS 2 in Fig. 2.13) results in obviously unacceptable images with an associated SNR of -2.7 dB and -3.8 dB respectively.

2.6 Principal Thesis

The primary argument in this chapter is radically different from the conventional techniques— an energy consumption versus accuracy tradeoff in arithmetic circuits is demonstrated using a circuit design technique called overclocking. This tradeoff can be used to design low energy consuming circuits for applications where lower than 100% accuracy can be tolerated.

But the issue being that brute force overclocking does not lead to an efficient implementation of this tradeoff. An optimal investment of energy based on value of information is required. The rest of this thesis proposes various solutions for *optimal* design of overclocked circuits.

Chapter 3

Modeling and Optimization of a Single Datapath Element: An Approximate Binary Adder

In approximate circuits, the accuracy of the output is traded for energy consumption through deliberate overclocking and multiple voltage levels. It was shown that in such circuits, where outputs need not be completely accurate all the time due to delay errors, accuracy can be improved by *biased voltage scaling* (BIVOS) in which the computation of more significant bits is given higher voltage than that of lower significant bits, compared to *uniform voltage scaling* (UVOS) in which the same lower voltage is supplied across the adder. But the voltage assignment to the components in the circuit has been ad-hoc. There has not been a well-grounded approach to guide a circuit designer in determining the voltage allocation for the gates in an adder to maximize accuracy while staying within the available energy budget.

3.1 Key Developments

In this chapter, we present a methodology which very efficiently finds a supply voltage investment in the case of approximate arithmetic adders at the gate level. The primary contributions are:

- We develop a formal model to characterize the average error in an approximate adder as a function of its design parameters.
- We also develop an energy model to estimate the energy consumption of an approximate adder. We present an algorithm to compute the reduced switching activities for a

ripple carry adder due to overclocking.

- We then pose our target problem as an optimization problem where the objective function is the average error of the approximate adder. The constraint is that the energy consumption of the adder should be under the given Energy Budget.
- We compute an approximation of our non-linear optimization problem as a geometric program. We then solve it using a standard toolbox which results in a continuous allocation of supply voltages to all the gates in the adder.
- For pragmatic concerns, we propose a heuristic algorithm to limit the number of supply voltages in the adder which we refer to as “supply voltage binning”.
- The improvement that results from this methodology is demonstrated by applying it to two example 16-bit adder designs, the ripple carry adder (RCA) and the carry lookahead adder (CLA). The error was reduced in an RCA by around 2.2X and by approximately 2.25X in a CLA.

We present here a roadmap to this chapter. In Section 3.2, we present a brief background on current day CMOS VLSI technology and also discuss some physical constraints on the implementation of multiple voltages in fabrication. technique of approximate arithmetic and the technology that motivates it. To avoid ambiguity about the exact problem that we target in this chapter, we discuss our goal briefly differentiating it against conventional circuit design optimization problems in Section 3.3. We present a comprehensive list of the variables and their definitions that will be used throughout the chapter in Section 3.4. We describe some assumptions that we use in the chapter on variables related to delay in Section 3.5 and with respect to energy modeling in Section 3.6.

The crux of the chapter is discussed in Section 3.7 where we present our model for the output error estimate of an approximate RCA. To elaborate, Section 3.7.1 presents the gate level description of the ripple carry adder. In Section 3.7.2, we present the model of an approximate RCA and the mathematical characterization of the output error in an approximate RCA. We describe our approach to reducing the computational complexity, which is one of the primary reasons for the practical validity of our approach, of the mathematical characterization of the average output error of an approximate adder over all possible inputs, in Section 3.8. We present our approach to model the energy consumption in an approximate RCA in Section 3.9. We present our target optimization problem and the technique of geometric programming that we use for solving our target optimization problem, which results in a globally optimized adder, in Section 3.10.2. We propose a supply voltage binning scheme in Section 3.10.3 where we map the optimal solution obtained from the optimization problem in Section 3.10.2 to a given set of supply voltages.

We then describe our simulation framework using which we performed all of our experiments in Section 3.11. We present the experimental results in Section 3.12 that compare our optimal supply voltage allocation scheme to a uniform voltage allocation scheme on our target adder designs. In Section 3.12, we also discuss about the non-monotonic behavior of output error rates of adders and present the impact of a globally optimized RCA design in the context of an FFT.

Till now we discussed only a ripple carry adder. But the RCA is not the most popular adder design. Hence we extend the entire formulation to other kinds of adders. Actually, the formulation was developed keeping in mind other adder designs which employ additional carry circuitry to increase parallelism in contrast to an RCA which is a completely serial architecture. To demonstrate the applicability of our techniques to parallel adders we use a carry look-ahead adder as a case study. In Section 3.13 we present the a few additional

definitions and terminology that we will use to model the error and energy of the carry lookahead adder. If any assumptions/definitions are not stated then it is assumed that they are carried over from the RCA case study. Then we present the energy and error model for an approximate CLA in Section 3.15. This section will have a similar flow when compared to Section 3.7. The most important characteristic of our formulation is that the geometric programming based optimization framework remains exactly the same when compared to the description in Section 3.10.2. Therefore we do not present those details again. We then describe the simulation framework for the approximate CLA in Section 3.17 and then present the experimental results in Section 3.18.

We now show our initial thoughts into extending the same geometric programming based framework to optimize any general circuit and not just an adder in Section 3.19. In Section 3.19.1 we describe our model of a general circuit. We then frame our general average error optimization problem in Section 3.19.2.

The impact of our optimization methodology on circuit design is discussed in Section 3.20.

3.2 Technology Background

In this section we discuss some background about the technology that we will be using in this chapter. Specifically, we explain the concepts of approximate arithmetic, overclocking and the use of multiple supply voltages in a circuit.

The problem of reducing energy consumption has in the current day circuit design industry taken the role of a “first citizen.” There are multiple approaches that have been proposed to combat this problem. In this thesis, we focus on one such approach by Chakrapani et al. [12] where a technique called “approximate arithmetic” was proposed. In conventional design methodology, a circuit is operated at a speed that is directly related to the critical path

delay of the circuit. The critical path delay, in turn, is directly related to the topology, process technology and supply voltage(s) of the circuit. The supply voltage dependence is due to the switching delay of a transistor being inversely proportional to its supply voltage. Also, it is known that the dynamic energy consumption of a transistor is in general proportional to the square of its supply voltage. Hence, a very direct way of reducing the switching power of a circuit is to reduce its supply voltage. But this would lead to a reduction in the operating frequency of the circuit and thus impact performance which is also a serious constraint. Keeping these constraints in mind, Chakrapani et al. [12] proposed a methodology in which the operating frequency of the circuit is maintained at the same rate but the supply voltage is decreased past the limit at which the critical path of the circuit is guaranteed to not be violated. This could result in an erroneous output of the circuit but, as shown by Chakrapani et al., can also be used in many applications which do not require strict 100% accuracy of computed values such as in audio and video signal processing. Such circuits which are “overclocked,” being operated at a frequency higher than required to guarantee 100% accuracy, we call “approximate circuits.”

In this chapter we propose a methodology to find an assignment of multiple voltages to an approximate ripple carry adder circuit to minimize error for a given energy consumption. The first design constraint that arises out of this design methodology is the number of multiple voltages that could be useful in practice in the fabricated chip. Circuit designers are using an increasing number of different voltages in their architectures. Typical high end chips seem to have four or five different voltages [49, 50]. To benefit from having multiple voltages on the die, the circuit designer has to overcome the challenge of creating power distribution networks that feed from the voltage regulator modules that supply all the devices on the fewest number of interconnect layers. But the important point to note here is that the number of different voltages is the bottleneck here and not the actual magnitude of each

voltage. The circuit designer has the freedom, albeit at design level, to choose the number of voltage levels and the exact values of the different voltages. With the freedom of using multiple voltage levels the use of voltage shifters becomes a necessity at least in some cases such as when a circuit with lower supply voltage is driving a circuit with higher supply voltage (and the difference in the supply voltages is not negligible) and when the output of a circuit is being stored in a register. It has been shown by Chang et al. [51] that the area/delay overhead of level shifters for using multiple supply voltages can be relatively small. Hence in the analysis presented in this thesis for the sake of simplicity of our mathematical model and experimental methodology we do not consider the overhead of voltage level shifters.

We have given a brief background about approximate circuits and overclocking which is used throughout in this work. We also use multiple voltages in our circuits and hence we mention some constraints and issues in using multiple supply voltages.

3.3 Discussion About Circuit Optimization And Our Target Problem

In this section we first describe our target problem. We then present a brief background about circuit design optimization problems.

3.3.1 Our target problem: approximate adder supply allocation problem

In this subsection we present the problem that we target in this chapter.

An approximate circuit as described in Section 3.2 is a circuit whose output might not be equal to the correct output because the circuit is overclocked. In this chapter we specifically target approximate adders. The two most basic parameters that characterize an approximate adder are the average output error over all possible input cases and the average energy consumption. The two main directions in which an approximate adder can be optimized are as follows:

1. Minimize the energy consumption of an approximate adder for a given average output error
2. Minimize the average output error of an approximate adder for a given energy consumption

While the former direction may likely be most appropriate for some applications, in this chapter we address the latter problem as a first step and are attempting to target the former problem for future work.

In short, our target problem is to minimize the average output error of an approximate adder for a given energy consumption.

3.3.2 Background about circuit optimization problems

In this subsection we briefly discuss about circuit optimization problems in general and the distinction of our target problem with respect to general circuit optimization problems.

Most circuit design optimization problems are computationally expensive to solve. For example, a typical place and route problem may be modeled as a linear programming or an integer linear programming problem which in the worst case are very expensive to solve. Hence such optimization problems of practical sizes are usually solved using heuristics and not by exact methods.

On the other hand, the problem that we target is the “approximate adder supply allocation problem” described in Subsection 3.3.1. Adders are typically used in a few options as far as number of bits are concerned: 16, 32 and 64 are the most common. In contrast to regular place and route problems which typically target 100,000 to a million transistors, we currently do not need to look at adders larger than 64 bits wide. Furthermore, due to limitations in the design of power converter circuitry [52] and layout overheads for power planes, typically

only a small number of voltages, e.g., four or five, are used in practice. As a result, in the design of an energy-optimized adder, at most 64 bits and at most five voltages is in practice a reasonable limitation leading to a limited range in the variables of the problem. As such, the problem can be formulated and solved using exact methods which can be exponential in terms of run time in the worst case. We use geometric programming which typically has polynomial run time but can be exponential in the worst case.

In summary, general circuit optimization problems are computationally very expensive to solve exactly and hence typically heuristics are used. The distinction of our target problem with respect to general circuit optimization problems is that we target specifically approximate arithmetic adders of a set of exact sizes (64 bits or less) and thus are able to utilize an exact solution method.

3.4 RCA Terminology

In this section, we define the terms that we use in this chapter to solve the approximate ripple carry adder (RCA) supply allocation problem. The reader may choose to skip this section and refer back only as needed.

1. \mathbf{a} : A multi-bit binary number with n bits
2. a_i : Represents the i^{th} bit in the multi-bit binary number \mathbf{a} where $0 \leq i < n$.
3. \mathbf{s} : Represents the correct $n + 1$ -bit sum output of an n -bit RCA. As a consequence, s_r , where $0 \leq r < n + 1$, denotes the r^{th} bit of \mathbf{s} .
4. Approximate adder: An approximate adder is an adder circuit whose sum output might not be the correct output because the adder is overclocked. In this chapter we will first consider only approximate RCAs.

5. s^a : Represents the approximate $n + 1$ -bit sum output of an n -bit RCA. As a consequence, s_r^a , where $0 \leq r < n + 1$, denotes the r^{th} bit of s^a . The sum output of an approximate n -bit RCA, s^a , may have errors, i.e., it may be the case that $s^a \neq s$.
6. c_q : Represents the q^{th} , where $0 \leq q < n + 1$, carry bit in an RCA.
7. \mathbf{c} : The sequence of carry bits in the RCA. Therefore $\mathbf{c} = c_n c_{n-1} \dots c_0$.
8. $0 \leq i < j \leq n - 1$: Represents all cases for i and j which satisfy that equation. For example, $\sum_{0 \leq i < j \leq n-1} f(i, j)$ denotes the sum of $f(i, j)$ for all values of i and j which satisfy $0 \leq i < j \leq n - 1$.
9. Carry chain : Given an n -bit RCA and two specific n -bit binary numbers \mathbf{a} and \mathbf{b} as inputs to the RCA, a carry chain is said to be present from position i to position j if and only if
 - $a_i = b_i = 1$. This case is referred to as the generation of a carry.
 - $a_w \neq b_w$. This case is referred to as the propagation of a carry.
 - $a_j = b_j$. If $a_j = 0$, the carry is said to be killed and if $a_j = 1$ another carry is said to be generated. In both the cases the carry chain that was generated at position i ends at position j .

where $0 \leq i < w < j \leq n - 1$. Please note that this definition of a carry chain is borrowed from [53].

10. $C_{ij}(\mathbf{a}, \mathbf{b})$: Consider an n -bit RCA and two specific n -bit binary numbers \mathbf{a} and \mathbf{b} as inputs to the RCA. A boolean variable C_{ij} is defined on whether there is carry chain

starting from position i and ending at position j such that

$$C_{ij}(\mathbf{a}, \mathbf{b}) = \begin{cases} 1 & \text{if there is a carry chain from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

where $0 \leq i < j < n$.

11. N : Represents the number of gates in the RCA.
12. v_ℓ : Represents the supply voltage of the ℓ^{th} gate, where $0 \leq \ell < N$.
13. \mathbf{v} : Represents a vector of size N of all the supply voltages, i.e., the ℓ^{th} element in the vector \mathbf{v} is v_ℓ , the supply voltage of the ℓ^{th} gate.
14. $\epsilon_\ell(v_\ell)$: Represents the worst-case propagation delay of the ℓ^{th} gate with a supply voltage of v_ℓ , where $0 \leq \ell < N$.
15. $\epsilon(\mathbf{v})$: Represents a vector of size N where the elements are the worst-case propagation delays of the gates in the adder which is a function of the voltage vector.
16. t_{in} : Represents the time instant when the inputs are provided to the RCA.
17. t_r : Represents the absolute time instant when the correct s_r is computed where $0 \leq r < n + 1$. Therefore the time taken for the correct s_r to be computed is equal to $t_r - t_{\text{in}}$.
18. d_{pr} : Given an n -bit RCA and two specific n -bit binary numbers \mathbf{a} and \mathbf{b} and assuming that $C_{pq} = 1$ for $p < r < q$, then we define $d_{pr} = t_r - t_{\text{in}}$. d_{pr} is the time elapsed from the instant when the inputs are provided to the RCA till the correct sum bit, s_r , is generated.

19. \mathbf{d} : Represents an $n \times n$ matrix where the r^{th} element in the p^{th} row is d_{pr} . Elements d_{pr} for which $p \geq r$ are invalid and so are not considered, hence, this is an upper triangular matrix.
20. D : This is the clock cycle time of the RCA, which is the difference in time between when the inputs are given to the RCA and the outputs are taken from the RCA.
21. I_k : Variable that denotes if there is an error at the k^{th} bit position, where $0 \leq k < n + 1$, bit output of an n-bit RCA output.
22. Critical Path Delay : The worst case delay of a circuit for all possible inputs is called the critical path delay of the circuit.
23. Overclocked RCA: In general, an RCA's clock cycle time (D) is greater than or equal to the RCA's critical path delay to avoid erroneous outputs. But in our methodology we challenge this constraint and operate the RCA at a clock cycle time lower than the critical path delay of the RCA. We call this type of RCA as an overclocked RCA and the general concept is called overclocking.
24. v_{\min} : The minimum supply voltage permitted to be used for a gate in the process technology used.
25. v_{\max} : The maximum supply voltage permitted to be used for a gate in the process technology used.
26. N_v : Represents the number of voltage levels that we consider in our experiments between v_{\min} and v_{\max} . For the target process technology of 90nm used in this work, $v_{\min} = 0.8V$, $v_{\max} = 1.2V$, and, with a granularity of $0.01V$, N_v is equal to 40.

27. $\epsilon_k(v_{\min})$: The worst-case delay of the k^{th} gate when the supply voltage is equal to v_{\max} .
28. $\epsilon_k(v_{\max})$: The worst-case delay of the k^{th} gate when the supply voltage is equal to v_{\min} .
29. \mathcal{A} : Number of additions in the benchmark used to calculate the switching activities in an RCA and also for an approximate RCA.
30. W_k : Represents the switching activity of the k^{th} gate in a ripple carry adder. Switching activity is defined as the average number of toggles per addition. The average is computed as the ratio of number of toggles at the output of the gate to the number of additions (\mathcal{A}) in the benchmark. Hence this quantity is a positive real number.
31. \mathbf{W} : Represents a vector of size N where the ℓ^{th} element is W_ℓ .
32. W_k^a : Represents the switching activity of the k^{th} gate in an approximate ripple carry adder.
33. \mathbf{W}^a : Represents a vector of size N where the ℓ^{th} element is W_ℓ^a .
34. $E_k^D(v_i)$: The average dynamic energy consumption of the k^{th} gate for a single transition when its supply voltage is equal to v_i where $v_{\min} \leq v_i \leq v_{\max}$. The dynamic energy consumed by a gate for a single transition depends on many other factors such as (i) whether the transition is a LOW to HIGH or a HIGH to LOW, (ii) the input(s) and (iii) parasitic capacitances. For the gates and the target technology used in this analysis, to compute $E_k(v_i)$ we consider an average energy consumption per transition over various input conditions.

35. $P_k^S(v_i)$: Represents the static power consumed by the k^{th} gate at a supply voltage of v_i .
36. γ_k : Represents the proportionality constant between the propagation delay and the average dynamic energy consumption of the k^{th} gate for a single transition averaged over several supply voltages.
37. E : Represents the total energy consumption of an adder over a single addition.
38. Energy Budget : The total energy budget, in the context of a geometric program (see Section 3.9), allocated as per our approximate adder supply allocation problem which is described in Section 3.3.
39. \mathcal{V} : Denotes the possible set of supply voltages for supply voltage binning.
40. M_v : The maximum number of distinct voltages allowed by the circuit designer for supply voltage binning.
41. $P(\mathcal{V})$: The power set of the possible set of supply voltages for supply voltage binning.
42. m_i : Let $p \in P(\mathcal{V})$. Then m_i denotes the number of gates in the adder with a supply voltage equal to the i^{th} element of p .
43. Globally optimized RCAs : RCAs with a voltage allocation scheme that is generated as a result of solving our target problem will be referred to as globally optimized RCAs.

The terms that we have defined in this section serve as a reference for the models and procedures described from Section 3.5 till Section 3.12.

3.5 RCA Delay Assumptions

In this section, we explain some basic assumptions on the variables that are related to propagation delay of gates and timing that we use in this chapter. The variables, which are already defined in Section 3.4, that we discuss in this section are (i) $\epsilon_k(\nu_k)$, (ii) $\text{delay}_{\min}(k)$, and (iii) $\text{delay}_{\max}(k)$.

We need to clarify some concepts about variables related to propagation delay and timing that we use because the technique of “approximate circuits” is, relatively, a novel circuit design technique. Before the conception of Probabilistic CMOS (PCMOS) by Palem et al. [40, 41, 9], four of the major VLSI circuit design techniques were digital VLSI design, asynchronous VLSI design, analog VLSI design and radio frequency VLSI design. In digital design, if we consider a certain circuit, the circuit designer can choose to ignore the intermediate results that the circuit produces as long as the final output of the circuit when the output is read at the end of the clock cycle is correct. But approximate circuits are overclocked, which means that the intermediate values can no longer be ignored. Also, similar to the case of asynchronous logic design, in approximate circuits the consideration of worst-case vs. average-case vs. best-case propagation delays of the circuit is important to properly characterize the output error of the approximate circuit. When we refer to worst-case or average-case or best-case propagation delay of the circuit, we consider the variation in the propagation delay of the circuit for different input transitions in our simulations at a given fixed supply voltage. There are many other factors that affect the propagation delay such as temperature and process variations. For the simulations presented in this chapter we consider that these factors do not change spatially or temporally. Also, for the purpose of this work, we will consider only worst-case errors at all places. To clarify, worst-case delay of a gate is characterized for various values of supply voltages. For example, we compute worst-case delays for $N_\nu = 40$ equally spaced supply voltage levels between $\nu_{\min} = 0.8V$

and $v_{\max} = 1.2V$ for 90nm technology. At this point, we have not modeled the effect on the output error of an approximate adder if average-case or best-case propagation delays are used. Instead we start with analysis of approximate RCAs assuming worst case delays. We do not have a formal proof to show that average and best case delays have lower error rates; however empirically we have found that increasing the delay increases the errors in the adder at least 98% of the time.

In Sections 3.4-3.12 we consider circuits which are RCAs. Each gate in a particular RCA could potentially (but not practically) be supplied with a different supply voltage. As per the definition of $\epsilon_k(v_k)$ in Section 3.4, the worst-case propagation delay of the k^{th} gate is $\epsilon_k(v_k)$ at supply voltage v_k . It has been shown by many independent sources [54, 55] that the propagation delay of a gate is inversely proportional to its supply voltage. This relationship was also used by Chakrapani et al. [12] in the modeling of the effect of biased voltage scaling in an approximate ripple carry adder. For our modeling, the values of propagation delays of gates for a given supply voltage are measured directly from HSPICE for our target technology. To relate the two quantities we will use the same relationship, that is, $\epsilon_k(v_k)$ is inversely proportional to the supply voltage of the k^{th} gate, v_k . From Section 3.4, we know that $\text{delay}_{\min}(k)$ denotes the worst-case delay of the k^{th} gate when the supply voltage is equal to v_{\max} and $\text{delay}_{\max}(k)$ denotes the worst-case delay of the k^{th} gate when the supply voltage is equal to v_{\min} . Thus $\text{delay}_{\min}(k) \leq \epsilon_k(v_k) \leq \text{delay}_{\max}(k)$.

For our error and energy modeling we assume for the sake of simplicity that all the carry bits are 0 when the inputs are provided to the adder for each addition. In reality the carry bits typically retain values from the prior computations and are not reset to 0 every time. Further discussion about the effect of non-zero carry bits on error and energy modeling is presented in Appendix A.

In addition to the above, we will assume that the clock cycle time chosen for overclocking

(D) is never less than the propagation delay of a single full adder.

In this section we have discussed the type of propagation delays that we use for approximate RCAs. We have also described our assumptions regarding the worst-case propagation delay of each gate in the circuits of RCAs that we consider.

3.6 Energy Modeling Assumptions

In this section we will discuss some of the assumptions we make about our energy models.

The total energy consumption of a circuit consists of two separate components, the dynamic energy consumption and the static energy consumption. The dynamic energy consumption constitutes the energy spent during the charging and discharging of capacitive loads during logic changes. The average dynamic energy consumed by a CMOS circuit thus depends on the number of logic changes which is denoted by the switching activity of the adder circuit. The switching activity of gate ℓ , denoted as W_ℓ , is the average number of logic changes that gate undergoes in a single addition. W_ℓ for gates in the case of an RCA is approximately estimated as the ratio of the number of logic changes of gate ℓ to the total number of additions in the benchmark (say \mathcal{A}). Therefore,

$$W_\ell = \frac{\text{Total number of toggles of gate } \ell}{\mathcal{A}}$$

To use our energy model to solve our target problem, we will need to represent the average dynamic energy consumption ($E_\ell^D(v_\ell)$) of a gate in terms of the worst-case propagation delay ($\epsilon_\ell(v_\ell)$) of that gate.

From Section 3.4, $\epsilon_\ell(v_\ell)$ denotes the worst-case propagation delay of the ℓ^{th} gate when its supply voltage is v_ℓ . We compute the average dynamic energy consumption and worst case propagation delays of all the gates in our process technology through simulations. It is known that the dynamic energy consumption of a gate is proportional to the square of

the input supply voltage and, as described in Section 3.5, the propagation delay of a gate is inversely proportional to its supply voltage. To represent $E_\ell^D(v_\ell)$ in terms of $\epsilon_\ell(v_\ell)$ we will use the curve-fit that the average dynamic energy consumption of a gate is proportional to the inverse square of its worst case propagation delay i.e.

$$E_\ell^D(v_\ell) \propto \frac{1}{\epsilon_k^2(v_i)} = \gamma_k \frac{1}{\epsilon_k^2(v_i)} \quad (3.2)$$

where γ_k is the proportionality constant for the k^{th} gate.

The proportionality constant, γ_k , is dependent on the process technology of the k^{th} gate. γ_k is computed by taking the average, over several supply voltage levels, of the product of the square of the worst-case (over all possible input transitions) propagation delay and the switching energy consumption of the k^{th} gate. Thus γ_k is computed for the k^{th} gate as follows

$$\gamma_k = \frac{1}{N_v} \sum_{i=1}^{N_v} \epsilon_k^2(v_i) \times E_k^D(v_i) \quad (3.3)$$

Thus γ_k is computed separately for each type of gate. For example, in the design of the RCA that we consider there are two types of gates, XOR and MUX. An example of computing the proportionality constant for the XOR gate in 90nm technology is given in Example 2.

Example 2. Consider an XOR gate in our target 90nm process technology. We will consider 5 specific voltages for this example and compute the proportionality constant. Refer to Table 3.1 for the values of worst-case propagation delay and average dynamic energy consumption per transition for the different supply voltage values. The first column shows the various supply voltage values. The last column shows the product of the average dynamic energy per transition and the square of the propagation delay which denotes the proportionality constant, γ_k , as per Eq. 3.2. For our modeling and simulations, we take an average of the proportionality constant over various supply voltages as shown in Eq. 3.3

Table 3.1 : Propagation delay and average dynamic energy per transition of an XOR gate in 90nm process technology for various supply voltage values

v_ℓ	$E_k^D(v_\ell)$ (femto-J)	$\epsilon_k(v_\ell)$ (pico-sec)	$E_k^D(v_\ell) \times \epsilon_k^2(v_\ell)$ (10^{-35} Jsec ²)
0.8	8.63	46.89	1.90
0.9	11.18	41.61	1.94
1	14.30	37.93	2.06
1.1	17.71	35.26	2.20
1.2	21.65	33.33	2.41

and in this example the average proportionality constant turns out to be 2.1×10^{-35} Jsec². As can be observed from Table 3.1, γ_k slightly increases with v_ℓ . But the increase is approximately 2% per 0.1V, therefore for simplicity we will assume that this does not affect the final result significantly and is captured by the mean (average) of the proportionality constant estimates for various voltage values. \square

3.7 Error Model For An Approximate RCA

In this section we first present a brief discussion of a ripple carry adder. This is followed a description of the mathematical framework and the models that we use to characterize the average error at the output of an approximate RCA.

3.7.1 Description of a ripple carry adder

A ripple carry adder (RCA) is the most basic adder design that is typically considered. Consider an n-bit ripple carry adder with inputs to the adder being denoted as \mathbf{a} and \mathbf{b} , where \mathbf{a} and \mathbf{b} are two n-bit binary numbers and c_0 is the carry input bit. The RCA is made up of a

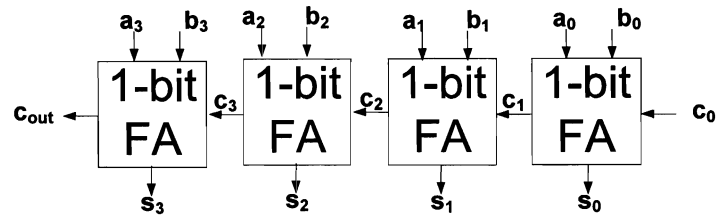


Figure 3.1 : Diagram of a 4-bit ripple carry adder

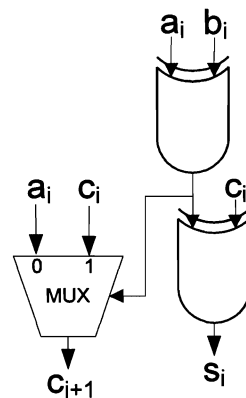


Figure 3.2 : Diagram of a 1-bit full adder (1-bit FA)

Table 3.2 : Maximum and minimum propagation delays of the XOR gate and the MUX in 90nm technology

Gate	delay _{min} (pico-sec)	delay _{max} (pico-sec)
XOR	33.3	55.2
MUX	30.5	51.2

series of blocks called full adders. Each full adder is a collection of logic gates which adds two input bits and a carry input to produce a sum output and a carry output. This means that the i^{th} full adder adds the two input bits a_i and b_i (a_i and b_i are the i^{th} bits in \mathbf{a} and \mathbf{b} ,

see Section 3.4) and the carry input c_i to produce s_i , the i^{th} sum bit, and c_{i+1} , the $(i + 1)^{\text{th}}$ carry bit. The sum of the adder is $\mathbf{s} = s_n s_{n-1} s_{n-2} \dots s_0$. This adder is called a ripple carry adder because the carry ripples through the adder one full adder at a time from the least significant bit to the most significant bit.

A diagram of a 4-bit ripple carry adder is shown in Fig. 3.1. In the diagram, each full adder is shown as a single block which is in fact a collection of logic gates. The logic diagram for each full adder is shown in Fig. 3.2. We admit that this might not be the best full adder design to optimize for area or speed, but we use Fig. 3.2 nonetheless for its simplicity. There are two types of gates present in the full adder design shown in Fig. 3.2, an XOR-gate and a multiplexer (MUX). To explain the concepts being introduced in this chapter, we will be using the delay values of these gates that were computed from HSPICE simulations. The minimum (for supply voltage of 0.8V) and maximum (for supply voltage of 1.2V) delay values of these gates in 90nm technology are shown in Table 3.5. The research presented at the beginning in this chapter will be extended later (starting from Section 3.13) to carry lookahead adders and can also be extended to other popular full adder circuits in, we believe, a straightforward manner as described later.

Note that an n -bit RCA based on Fig. 3.2 would have $N = 3 \times n$ gates. In general, we will use the following indexing scheme to refer to a gate in an RCA.

- The top XOR gate in the full adder in Fig. 3.2 that computes $a_k \oplus b_k$ at position k would be referred to using the index $3k + 1$ where the n full adders in the RCA are indexed by $0 \leq k \leq n - 1$ according to bit position.
- The bottom XOR gate in the full adder as per Fig. 3.2 that computes the sum output, s_k at position k would be referred to using the index $3k + 2$.
- The one and only multiplexer in the full adder at position k would be referred to using

the index $3k + 3$.

3.7.2 Modeling the error at the output of an approximate RCA

In this section, we will first present boolean logic functions we use to represent the outputs of an RCA. Then we will define and describe carry chains in ripple carry adders. We will next describe the effect of carry chains on error at the output of an approximate ripple carry adder. We also present the behavior of errors at the output of an overlocked RCA in the presence of a carry chain. We then describe the procedure that we follow to characterize the time, denoted as d_{pr} (see Section 3.4), it takes for a specific sum bit in an RCA to be correctly computed. We describe a mathematical formulation for the error function modeling the error at the output of an approximate RCA.

RCA logic

In this subsection we present one set of (from among many) boolean logic functions of binary addition with respect to an RCA. We will also discuss the effect of a carry chain on the outputs of an RCA.

We will consider n -bit RCAs for some n . Numbers to be added will be in the range $0, \dots, 2^n - 1$ and will have the standard binary representation. We will show our procedure for modeling and analysis of only unsigned ripple carry adders. We agree that many applications require handling of signed numbers as well, typically in 2's complement format. But because typical 2's complement adders utilize an unsigned adder to propagate the carry bits with additional circuitry around them to handle the sign, it appears reasonable to analyze errors due to carry propagation in an unsigned adder as an initial step.

Consider the addition of two n -bit numbers \mathbf{a} and \mathbf{b} in an n -bit RCA, resulting in an $(n + 1)$ -bit number, consisting of $\mathbf{s} = s_n s_{n-1} \dots s_0$. Thus sum \mathbf{s} is computed in an RCA as

INPUT A	0	0	1	0	1	1	0	0
INPUT B	0	0	0	1	0	1	0	0
SUM	0	1	0	0	0	0	0	0
POSITION	7	j=6	5	4	3	i=2	1	0

Figure 3.3 : An Example of a carry chain in a binary addition using an RCA

INPUT A	0	0	1	1	1	1	0	0
INPUT B	0	0	0	1	0	1	0	0
SUM	0	1	0	0	0	0	0	0
POSITION	7	6	5	4	3	2	1	0

Figure 3.4 : An Example of two contiguous carry chains in a binary addition using an RCA

$s_\ell = a_\ell \oplus b_\ell \oplus c_\ell$ for $0 \leq \ell \leq n-1$. The sequence of carries \mathbf{c} is computed as follows. c_0 is input (and is zero for addition) while $c_{\ell+1} = (a_\ell \oplus b_\ell) \cdot c_\ell + \overline{(a_\ell \oplus b_\ell)} \cdot a_\ell$ for $0 \leq \ell \leq n-1$. Note that we have yet to define s_n ; in fact, the last sum bit s_n is defined as being equal to the carry bit c_n .

Carry chains in ripple carry adders

In this section, we will discuss carry chains in the context of an RCA.

Consider an n -bit RCA and two specific n -bit binary numbers \mathbf{a} and \mathbf{b} as inputs to the RCA. To repeat the definition of a carry chain as described in Section 3.4, a carry chain is said to be starting from position i and ending at position j if and only if

- $a_i = b_i = 1$. This case is referred to as the generation of a carry.
- $a_w \neq b_w$. This case is referred to as the propagation of a carry.

- $a_j = b_j$. If $a_j = 0$, the carry is said to be killed, and if $a_j = 1$, another carry is said to be generated. In both the cases the carry chain that was generated at position i ends at position j .

where $0 \leq i < w < j \leq n - 1$.

In general, if there is a carry chain from i to j , we will set a boolean variable $C_{ij}(\mathbf{a}, \mathbf{b})$ to 1 as defined in Section 3.4. And for a carry chain from position i to position j , we have $s_i = 0 \oplus c_i$; $c_k = 1$ and $s_k = 0$, for $k \in \{i + 1, \dots, j - 1\}$; and $c_j = 1$, $s_j = 1$, and $c_{j+1} = a_j (= b_j)$. Thus, if we know that there is a carry chain from position i to position j , we can easily determine the correct sum bits from position $(i + 1)$ to position j .

In a ripple carry adder, e.g., with a design as described in Section 3.7.1, in the worst-case the carry propagates from the lowest significant bit position to the most significant bit position.

Example 3. An example of a carry chain in a binary addition using an RCA is shown in Fig. 3.3. As is denoted in the figure, the carry chain starts from position $i = 2$ and ends at position $j = 6$, and therefore we will say that there is a carry chain from 2 to 6 and will also denote $C_{26} = 1$. The outputs of the RCA, the sum bits, are shown in Fig. 3.3. As described earlier in this section, the correct sum bits between position 3 and 6 are fixed based on the fact that there is a carry chain from position 2 and 6. \square

Definition 1. Consider an n -bit addition with inputs \mathbf{a} and \mathbf{b} . If there exists two carry chains such that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$, then these carry chains are said to *overlap* if and only if $i \leq x < j \leq y$.

The case shown in Example 3 has a single carry chain across the entire 8-bit addition. While technically there can be more than one carry chain in a single addition, the carry

chains cannot overlap (as per Definition 1) over each other in any case. We show this observation to be true in Observation 1.

Observation 1. Consider an n -bit addition with inputs \mathbf{a} and \mathbf{b} . As per the definition of a carry chain in Section 3.4, two carry chains cannot overlap. Specifically there cannot exist $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$ such that $i \leq x < j \leq y$.

Proof. We will prove this using the method of contradiction.

From Section 3.4, $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ in an n -bit RCA if and only if

- $a_i = b_i = 1$.
- $a_w \neq b_w$.
- $a_j = b_j$.

where $0 \leq i < w < j \leq n - 1$.

Assume that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$ in the same n -bit addition such that $i \leq x < j \leq y$. From the definition, if $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ then for all $0 \leq i < w < j \leq n - 1$ it is known that $a_w \neq b_w$. But if $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$, then we know from the definition of the carry chain that $a_x = b_x = 1$. But from our assumption the carry chains are such that $i \leq x < j$. Consider the case where $w = x$. This results in a contradiction because if $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ then $a_w \neq b_w$ but if $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$ then $a_w = b_w$.

Hence the observation holds.

□

Also in Example 3, the carry chain that started at position $i = 2$ was killed at position $j = 6$ where $a_6 = b_6 = 0$. But as described in the definition of a carry chain, it could also have been killed if $a_j = b_j = 1$. We clarify these points in Example 4.

Input A	0	1	1	1	1	(15)		
Input B	0	1	0	0	1	(9)		
Time							$s^{(i)}$	Error
0	0	0	0	0	0	0	$s^{(0)} = 0$	24
1	0	0	0	1	1	0	$s^{(1)} = 6$	18
2	0	1	0	1	0	0	$s^{(2)} = 20$	4
3	0	1	0	0	0	0	$s^{(3)} = 16$	8
4	0	1	1	0	0	0	$s^{(4)} = 24$	0

Figure 3.5 : An example to demonstrate that in an RCA it is possible that by increasing the supply voltages the accuracy of the sum is decreased.

Example 4. An example of two contiguous carry chains in a binary addition using an RCA is shown in Fig. 3.4. In this case the first carry chain starts at position 2 and is killed at position 4. But because $a_4 = b_4 = 1$ another carry chain is generated at position 4 which is killed at position 6. Thus in this case both $C_{24}(\mathbf{a}, \mathbf{b}) = 1$ and $C_{46}(\mathbf{a}, \mathbf{b}) = 1$. This shows that even though $c_2 = 1, c_3 = 1, c_4 = 1$ and $c_5 = 1$, it actually consists of two separate carry chains. Also as described in the definition of a carry chain, even a position that generates a new carry chain can also kill a previous carry chain. Furthermore, as per the definition of a carry chain, multiple carry chains can exist in a single addition but cannot overlap. This is clearly shown by the two directed arrows in Fig. 3.3 and Fig. 3.4. \square

The relationship between RCA carry chains and overclocking errors

As mentioned in Section 3.5, we assume that the clock cycle time (D) of an adder is never lower than the worst-case propagation delay of a single full adder. Considering an approximate RCA, this would imply that there would be a possibility of error at the output of an RCA only if there is propagation of carry. Because if there is no propagation of carry, the clock cycle time is sufficient for the full adders to compute the sum outputs of the RCA. Stated in a different way, there may be an error at the output only if there are carry chains in

the approximate RCA. Note that this is true only if we assume that the temporary values of all the carry bits in the adder are zeros at the beginning of the addition. As explained in Section 3.5, we assume that all the carry bits are 0 when the inputs are provided to the adder for each addition.

Hence in developing an error model for an approximate RCA we will consider the behavior of error at the output of an RCA in the presence of carry chains.

We will now describe a point which is worth noting about RCA error with respect to a carry chain. The point is that in general [12, 47], it has been assumed that to increase the output accuracy of an overclocked RCA the supply voltage(s) of the gates in the circuit of the RCA should be increased which in turn results in increasing the energy consumption of the circuit of the RCA. However, due to overclocking in the presence of a carry chain, *increasing* the voltage provided to the components of the RCA may also result in *decreasing* the accuracy of the sum read from the RCA in some cases. An example of such a case is presented in Example 5.

Example 5. Consider a 5-bit ripple carry adder (RCA) with a design adding 01111 and 01001. The gate-level design of an RCA is described in Section 3.7.1. Consider the carry chain starting from position 0 till position 3 as shown in Fig. 3.5. For the sake of this example, we will assume that all the full adders in the RCA have the same propagation delays. We are excluding the cases where the propagation delays change because of random variations and also different input transitions and hence are assuming that the propagation delays are exact. Also define $s^{(t)}$, just for this example, to denote the binary sum output of the 5-bit RCA at the t^{th} instant, where t ranges from 0 (which denotes the output before the inputs were given to the adder) till the correct output is computed where each subsequent step denotes a change in the sum output. Then the computed value of s starts with the initial value of $s^{(0)} = 000000$ (output value 0), consecutively becomes

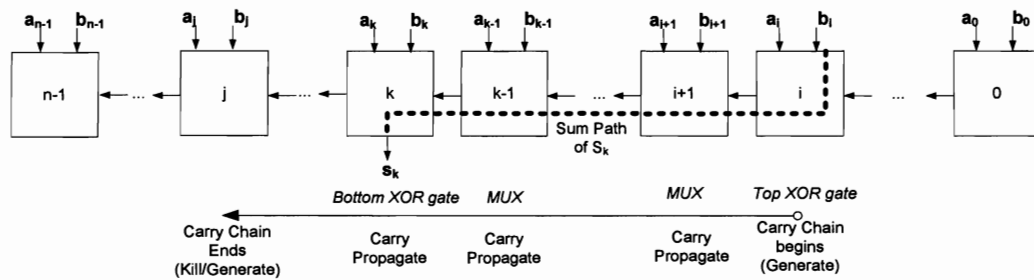


Figure 3.6 : Diagram of an n-bit ripple carry adder to describe the modeling of the critical path of a sum bit in a carry chain

$s^{(1)} = 000110$ (output value 6), $s^{(2)} = 010100$ (output value 20), $s^{(3)} = 010000$ (output value 16), and $s^{(4)} = 011000$ (output value 24). As the correct sum is 24, the consecutive errors at $t = 1$ to 4 are 18, 4, 8, and 0. So, assuming some specified overclocked clock cycle time, if the voltage supplied allowed the reading of $s^{(3)}$, the resulting error would be 8. In contrast, if the voltage supplied were lower, allowing only the reading of $s^{(2)}$ with the same overclocking, the resulting error would be only 4. \square

In this subsection we discussed the behavior of errors in an RCA due to overclocking in the presence of carry chains. We also show using an example the output error of an approximate RCA might increase when the supply voltage(s) of the RCA is(are) increased.

Critical path of a sum bit in a carry chain of an RCA

In this subsection we will define and discuss the critical path (referred to as the sum path) of a particular sum bit in the middle of a carry chain in an RCA.

We call the longest delay path with respect to a particular sum bit s_k as the "critical path for sum s_k ." In an RCA, the critical path for sum s_k is the series of gates in the RCA which constitutes the longest delay path, assuming worst-case gate delays. This path is essential in computing whether there is enough time to always compute correctly a particular sum

bit s_k . Of course, the true critical path for sum s_k depends on inputs \mathbf{a} and \mathbf{b} ; however, the result is that any true critical path—whose delay exceeds that of a single FA—comes from a prior bit position: for s_k , then, the true critical path given inputs \mathbf{a} and \mathbf{b} will come from bit position i where $i < k$. To capture all such possibilities, we define d_{ik} to be the time between the correct computation of sum bit s_k and the time when the inputs are provided to the RCA circuit thus triggering a true critical path for s_k starting from bit i . Inputs \mathbf{a} and \mathbf{b} are provided to the RCA at some time t_{in} . Let t_k be the time when the correct value of s_k is generated (assuming worst-case delays of all gates). Then $d_{ik} = t_k - t_{\text{in}}$. For the special case when the carry chain is from i to $j = n$, t_j is the time instant when carry c_n is generated. \mathbf{d} denotes the $n \times n$ matrix of all d_{ik} , $0 \leq i < k \leq n$. Properly speaking, \mathbf{d} for a particular RCA in a particular technology is a function of the critical paths of the sum bits of the RCA, v_i for each gate i in any critical path of any sum bit, and $\epsilon_i(v_i)$; however, for brevity, we will simply refer to \mathbf{d} without specifying all the input values on which \mathbf{d} depends.

Example 6. Consider an n -bit ripple carry adder, shown in Fig. 3.6, based on the gate level description described in Section 3.7.1. Assume that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$. In this adder, d_{ik} in the worst-case would be the sum of propagation delays of the top XOR gate in the full adder (referring to Fig. 3.2) at position i , MUX gates of the full adders from position $i + 1$ to position $k - 1$ and the bottom XOR gate in the full adder (referring to Fig. 3.2) at position k . This is the critical path of the sum bit s_k assuming that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$. \square

The critical path of a sum bit in the case of an RCA is further clarified through Example 7.

Example 7. Consider the 3-bit ripple carry adder shown in Fig. 3.7 based on the gate level model of an RCA described in Section 3.7.1. Assume that the inputs are such that there is a carry chain from position 0 to position 2. One instance of such inputs are $\mathbf{a} = 011$,

$\mathbf{b} = 001$ and $c_0 = 0$. For this instance, a carry bit of 1 is generated at position 0 and is propagated through position 1 and is killed at position 2. For this scenario, $d_{02} = t_2 - t_{in}$, where t_2 (listed in Section 3.4) is the time when the correct s_2 is generated. Assuming worst-case gate delays, d_{02} would be equal to the sum of worst-case propagation delays of GATE-1 (where GATE- k denotes the gate with the number k inside it), GATE-3, GATE-6 and GATE-8. The critical path corresponding to d_{02} is shown in Fig. 3.7 as a dotted line from bit position 0 to the sum output in bit position 2. That means in the worst case, $d_{02} = \epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_6(v_6) + \epsilon_8(v_8)$. For our target technology, considering maximum supply voltages for all the gates, i.e $v_1 = v_3 = v_6 = v_8 = 1.2V$, we get $d_{02} = 33.3 \text{ ps} + 30.5 \text{ ps} + 30.5 \text{ ps} + 33.3 \text{ ps} = 127.6 \text{ ps}$.

With calculations similar to d_{02} , we find that

$$\mathbf{d} = \begin{pmatrix} - & 97.1 \text{ ps} & 127.6 \text{ ps} \\ - & - & 97.1 \text{ ps} \\ - & - & - \end{pmatrix}$$

□

We presented a description of the critical path delay of a sum bit in a carry chain in terms of the propagation delays of the gates in an RCA.

RCA error based on an analysis of carry chain errors

In this subsection, we will develop a function for the error at the output of an overclocked RCA. We define the error at the output of the RCA as a function of \mathbf{a} , \mathbf{b} , the topology of the RCA, $\epsilon_k(v_k)$ and D .

The circuit of an RCA is built of gates. The RCA is given time D for each addition.

The sum outputs are read at time $t_{in} + D$, with D independent of the inputs. Due to overclocking, which we assume, the sum actually read, \mathbf{s}^a , may be different from $\mathbf{s} = \mathbf{a} + \mathbf{b}$.

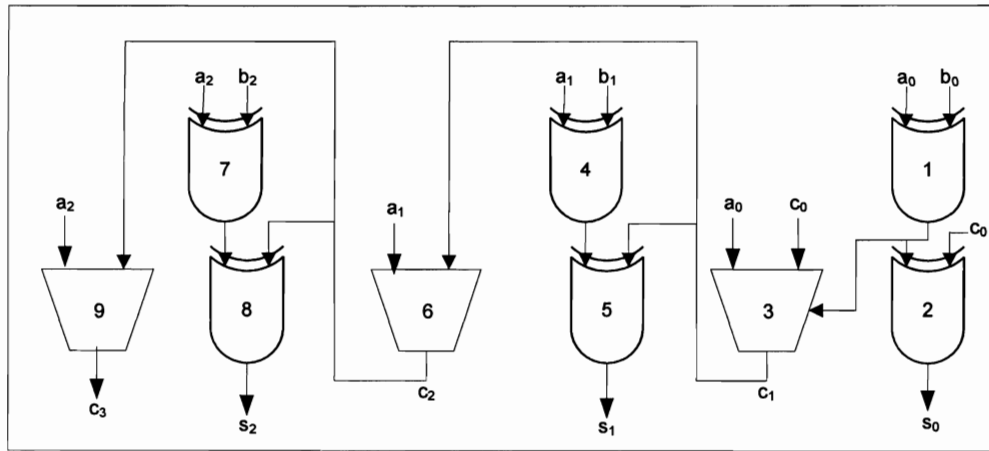


Figure 3.7 : Gate level diagram of a 3-bit ripple carry adder

We now proceed to characterize the absolute magnitude of the error $|s^a - s|$. We define an *indicator function* as follows:

$$I_k(\mathbf{a}, \mathbf{b}, d, D) = \begin{cases} 1 & \text{if } d_{ik} > D \text{ and } \exists i, j \text{ such that } C_{ij}(\mathbf{a}, \mathbf{b}) = 1 \text{ and } i < k < j \\ -1 & \text{if } d_{ik} > D \text{ and } \exists i, j \text{ such that } C_{ij}(\mathbf{a}, \mathbf{b}) = 1 \text{ and } i < k = j \\ 0 & \text{otherwise.} \end{cases} \quad (3.4)$$

If k is in some carry chain and the correct sum is 0 but a sum bit of 1 is read, then there is positive error and thus $I_k = 1$. If the correct sum of a bit in a carry chain is 1 and a 0 is read, then $I_k = -1$, indicating negative error. If the correct value of the sum bit in a carry chain is read or if the sum bit is not in a carry chain at all, then there is no error and thus $I_k = 0$. Our approach to apply the indicator function to indicate presence of a positive or negative error at a particular bit position is demonstrated using an example in Example 8.

Example 8. Consider the addition being performed in Fig. 3.3. Let us apply the definition of the indicator function to this addition, assuming an 8-bit ripple carry adder based on the ripple carry adder described in Section 3.7.1. Considering maximum supply voltage to all the gates, the delays of the gates are given in Table 3.5 in Section 3.7.1. For this example, let us assume that the clock cycle time, D , is 130ps. We then calculate the values of \mathbf{d} , using the method described in Section 3.7.2. For the purpose of this addition, because there is only one carry chain, $C_{26}(\mathbf{a}, \mathbf{b})$, we will only compute the following: $d_{23} = 97.1\text{ps}$, $d_{24} = 127.6\text{ps}$, $d_{25} = 158.1\text{ps}$ and $d_{26} = 188.6\text{ps}$. Using the values of \mathbf{d} and D , we can compute $I_k(\mathbf{a}, \mathbf{b}, d, D)$ for $0 \leq k \leq n - 1$ as shown in Eq. 3.4.

- $k = 0$: There is no i, j such that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $0 \leq i < k \leq j$. Therefore $I_0(\mathbf{a}, \mathbf{b}, d, D) = 0$.
- $k = 1$: There is no i, j such that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $0 \leq i < k \leq j$. Therefore $I_1(\mathbf{a}, \mathbf{b}, d, D) = 0$.
- $k = 2$: From the definition of a carry chain in Section 3.4, we know that $C_{26}(\mathbf{a}, \mathbf{b}) = 1$, but $k = 2$ does not satisfy $0 \leq i < k \leq j$. Therefore $I_2(\mathbf{a}, \mathbf{b}, d, D) = 0$.
- $k = 3$: We know that $C_{26}(\mathbf{a}, \mathbf{b}) = 1$ and for $k = 3$ it satisfies $0 \leq i < k \leq j$. But $d_{23} \leq D$, therefore $I_3(\mathbf{a}, \mathbf{b}, d, D) = 0$.
- $k = 4$: We know that $C_{26}(\mathbf{a}, \mathbf{b}) = 1$ and for $k = 4$ it satisfies $0 \leq i < k \leq j$. But $d_{24} \leq D$, therefore $I_4(\mathbf{a}, \mathbf{b}, d, D) = 0$.
- $k = 5$: We know that $C_{26}(\mathbf{a}, \mathbf{b}) = 1$ and for $k = 5$ it satisfies $0 \leq i < k \leq j$. Also $d_{25} > D$, therefore as per the first line of the indicator function in Eq. 3.4, $I_5(\mathbf{a}, \mathbf{b}, d, D) = 1$.

- $k = 6$: We know that $C_{26}(\mathbf{a}, \mathbf{b}) = 1$ and for $k = 6$ it satisfies $0 \leq i < k \leq j$. Also $d_{26} > D$, therefore as per the second line of the indicator function in Eq. 3.4, $I_6(\mathbf{a}, \mathbf{b}, d, D) = -1$.
- $k = 7$: There is no i, j such that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $0 \leq i < k \leq j$. Therefore $I_7(\mathbf{a}, \mathbf{b}, d, D) = 0$.

□

The indicator function is now used to develop a function to compute the error at the output of an approximate RCA. $Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = |\sum_{k=0}^n (s_k^a - s_k)2^k|$ is the error introduced during the computation, assuming non-varying deterministic worst-case delays.

Theorem 1.

$$Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = \sum_{k=0}^n I_k(\mathbf{a}, \mathbf{b}, d, D)2^k. \quad (3.5)$$

Proof. To prove the theorem, we will describe the three cases of the definition of $I_k(\mathbf{a}, \mathbf{b}, d, D)$, as given in Eq. 3.4, which is used in the right hand side of Eq. 3.5.

1. Consider the first line in the right hand side of Eq. 3.4. This is the case where the correct c_k has not been computed by time D because $d_{ik} > D$. Here, $c_k = 1$ and $a_k \oplus b_k = 1$ because we know that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $i < k < j$. So, $s_k = 0$ but the correct c_k has not been computed by time D and thus $s'_k = 1$. Therefore, the error at bit position k contributed $(1 - 0)2^k = I_k(\mathbf{a}, \mathbf{b}, d, D)2^k$ to the total error.
2. Consider the second line in the right hand side of Eq. 3.4. In this situation the correct c_j (because $k = j$) has not been computed by time D because $d_{ij} > D$. Here $c_j = 1$ and either $a_j = b_j = 0$ or $a_j = b_j = 1$ because we know that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $k = j$. So, $s_j = 1$ but the correct c_j has not been computed by time D and thus

$s'_j = 0$. Therefore, the error at bit position j contributed $(0-1)2^j = I_j(\mathbf{a}, \mathbf{b}, d, D)2^j$ to the total error.

3. Consider the third line in the right hand side of Eq. 3.4. This is the case where there is no error at the output of the adder. We have two situations in which there is no error.*

(a) The first situation is when the particular bit-position is in a carry chain but the correct c_k has been computed by time D because $d_{ik} \leq D$. Therefore $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $i < k \leq j$. This means that $s'_k = s_k$. Therefore there was no error at k and $0 = I_k(\mathbf{a}, \mathbf{b}, d, D)2^k$.

(b) The second situation is when the particular bit position is not in a carry chain. As we mentioned in Section 3.5, the clock cycle time is at least greater than the delay of a single full adder. So if the particular full adder is not in a carry chain then there would not be any error at that position. Therefore, there are no i and j for which $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $i < k \leq j$. So, $c_k = 0$. Therefore there was no error at k and $0 = I_k(\mathbf{a}, \mathbf{b}, d, D)2^k$.

□

Summary of subsection 3.7.2

The goal of Subsection 3.7.2 is to develop a function for the error at the output of an approximate RCA. To do this, we first describe boolean logic functions for the bits computed in an RCA. We then describe the reason that, given our assumptions, errors at the RCA output occur only in the presence of carry chains. We also describe how the outputs of an RCA

*The case where $k = 0$ comes under this case because there is no error at this position since it is assumed that D is always greater than or equal to the worst-case delay of a single full adder. Hence there is no situation in which there could be an error at this position.

are affected in the presence of a carry chain. Also, we show using a couple of examples the behavior of RCA error due to overclocking in a carry chain. We then describe a sample computation of the critical path delay of a sum output of an RCA. The function for the error at the output of an approximate RCA is computed using the critical path delay of a sum output in the form of an indicator function.

3.8 Efficient Evaluation of Average Error of an Approximate RCA

In this section, we describe our approach to compute the average of the error at the output of an approximate RCA over a candidate set of inputs. We then present the technique that we follow to efficiently compute this average error. We also describe the constraints that we pose on our target problem.

Theorem 1 (Eq. 3.5) gives $Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d})$ which is the error at the output of the target RCA for two specific inputs. The average of this error over all possible inputs is

$$Er_{\text{avg}}(D, \mathbf{d}) = \text{avg}_{0 \leq \mathbf{a}, \mathbf{b} \leq 2^n - 1} Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}). \quad (3.6)$$

where \mathbf{d} and D are defined in Section 3.4.

This is a sum of 2^{2n} terms, which is not feasible to compute in a straightforward manner for large n . We will now transform the expression in Eq. 3.6 into a form that can be computed in $O(n^2)$ operations.

Recall that an error can occur only if there is a carry chain in the computation. We then note that the total error in a computation is the sum of the errors (if any) in the individual carry chains. The error introduced by the carry chain from i to j is

$$Er(D, i, j, \mathbf{d}) = \sum_{k=i+1}^j (s'_k - s_k)2^k = \sum_{k=i+1}^j I_k 2^k. \quad (3.7)$$

From Section 3.7.2, we know that the indicator function I_k depends on $D, \mathbf{a}, \mathbf{b}$ and \mathbf{d} . If we observe Eq. 3.4, we can see that in the definition of I_k we use the variables \mathbf{a}

and \mathbf{b} to determine whether there is a carry chain around position k . As in Eq. 3.7, we assume that $C_{ij} = 1$ for $i < k \leq j$. (note that this can be satisfied for many different input pairs). Therefore we do not need \mathbf{a}, \mathbf{b} as inputs to the indicator function. To simplify our explanation we will use the same terminology for the indicator function for $\text{Er}(D, i, j, \mathbf{d})$.

Theorem 2.

$$\text{Er}(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = \sum_{\text{all } i, j \text{ for which } C_{ij}(\mathbf{a}, \mathbf{b})=1} \text{Er}(D, i, j, \mathbf{d})$$

Proof. Consider an n -bit addition. Let there be α carry chains in the addition, where $0 \leq \alpha \leq n$.[†]

We know from Theorem 1 that there could be an error at the output of an addition only if there is a carry chain. Therefore for the case when $\alpha = 0$ there is no error. If $\alpha = 0$ that means there does not exist i, j such that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$. Hence the theorem holds for $\alpha = 0$.

Consider $\alpha \neq 0$. Let the x^{th} carry chain start from position i^x and be killed at position j^x . From Theorem 1, we know that

$$\text{Er}(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = \sum_{k=0}^n I_k 2^k \quad (3.8)$$

If we expand the right hand side of Eq. 3.8, we get

$$\text{Er}(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = I_0 2^0 + I_1 2^1 + \dots + I_{n-1} 2^{n-1} + I_n 2^n \quad (3.9)$$

As $I_k = 0$ if there is no carry chain, we can group the terms on the right hand side of Eq. 3.9 into groups of carry chains. This is possible since we have shown in Observation 1 that carry chains cannot overlap in the same addition.

$$\text{Er}(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = \left(I_{i^1} 2^{i^1} + \dots + I_{j^1} 2^{j^1} \right) + \dots + \left(I_{i^\alpha} 2^{i^\alpha} + \dots + I_{j^\alpha} 2^{j^\alpha} \right) \quad (3.10)$$

[†]For example, if $\alpha = 0$ then that means there is no carry being propagated in the entire addition. The other extreme case would be if $\alpha = n$ which would happen if $\mathbf{a} = \mathbf{b} = 2^n - 1$.

where carry chain 1 was generated at i^1 and killed at j^1 and carry chain α was generated at i^α and killed at j^α .

From Eq. 3.10 and Eq. 3.7

$$Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = Er(D, i^1, j^1, \mathbf{d}) + \cdots + Er(D, i^\alpha, j^\alpha, \mathbf{d}) \quad (3.11)$$

Hence the theorem holds. □

One way to compute the average total error at the output of an adder is by summing the errors of all possible carry chains weighted by the probability of their occurrence.

$$Er_{\text{avg}}(D, \mathbf{d}) = \sum_{0 \leq i < j \leq n-1} p_{ij} Er(D, i, j, \mathbf{d}), \quad (3.12)$$

where p_{ij} is the probability that there exists a carry chain from i to j . Thus, the average total error is evaluated by computing and adding $n(n-1)/2$ terms only.

The probabilities p_{ij} can be computed given the distributions of the inputs a and b . Here we will assume uniform distribution, that is, a_i and b_i , for all $0 \leq i < n$, are each 0 or 1 with probability $\frac{1}{2}$. Based on the definition of a carry chain from Section 3.4, for a carry chain to be present from position i to position j the following conditions have to be satisfied.

- $a_i = b_i = 1$. Probability of $a_i = b_i = 1$ is $\frac{1}{2} \times \frac{1}{2} = \left(\frac{1}{2}\right)^2 = \frac{1}{4}$
- $a_w \neq b_w$. Probability of $a_w \neq b_w$ is equal to $P(a_w = 0 \text{ and } b_w = 1) + P(a_w = 1 \text{ and } b_w = 0)$. This is equal to $\left(\frac{1}{2} \times \frac{1}{2}\right) + \left(\frac{1}{2} \times \frac{1}{2}\right) = \frac{1}{2}$.
- $a_j = b_j$. Probability of $a_j = b_j$ is equal to $P(a_j = 0 \text{ and } b_j = 0) + P(a_j = 1 \text{ and } b_j = 1)$. This is equal to $\left(\frac{1}{2} \times \frac{1}{2}\right) + \left(\frac{1}{2} \times \frac{1}{2}\right) = \frac{1}{2}$.

Therefore p_{ij} is the product of probabilities of all the three conditions i.e.,

$$\begin{aligned}
 p_{ij} &= P(a_i = b_i = 1) \times P(a_w \neq b_w) \times P(a_j = b_j) \\
 &= \left(\frac{1}{2}\right)^2 \times \left(\prod_{w=i+1}^{j-1} \frac{1}{2}\right) \times \frac{1}{2} \\
 &= \left(\frac{1}{2}\right)^{j-i+2}
 \end{aligned}$$

The above method calculates the probability that there exists a carry chain if the input distribution is uniform. In real world applications, this might not be true. Therefore, if the knowledge about the probability distribution of the actual inputs is known then that could be used instead of using $P(a_i = 0) = P(b_i = 0) = P(a_i = 1) = P(b_i = 1) = \frac{1}{2}$. If the case is such that instead of the probability distribution we have a candidate input benchmark, then the probability distribution could be computed using the benchmark. This would require only one pass through the entire candidate set of inputs which is an $O(n)$ operation. We will leave the case of non-uniform input bits for future work.

In this section we discussed our approach to efficiently computing the average error of an approximate RCA. We also presented the constraints that we pose on our target problem.

3.9 Energy Consumption Models

In this section, we first describe an energy model to estimate energy consumption for a CMOS circuit of an RCA. We then describe our approach to extend the energy model of an RCA to estimate the energy consumption of an approximate RCA for solving our target problem.

3.9.1 Energy model for an RCA

In this subsection we will discuss an energy model for an RCA.

The total energy consumption in an RCA consists of two separate components, the dynamic energy consumption and the static energy consumption. The dynamic energy consumption constitutes the energy spent during the charging and discharging of capacitive loads during logic changes. The average dynamic energy consumed by a CMOS circuit thus depends on the number of logic changes which is denoted by the switching activity of the adder circuit. The switching activity of gate ℓ , denoted as W_ℓ , is the average number of logic changes that gate undergoes in a single addition. W_ℓ is approximately estimated as the ratio of the number of logic changes of gate ℓ to the total number of additions (say \mathcal{A}).

To estimate the dynamic energy consumption at the gate level of a CMOS circuit of an RCA, we will use the following.

$$E^D = \sum_{\ell=1}^N E_\ell^D(v_\ell) W_\ell \quad (3.13)$$

where $E_\ell^D(v_\ell)$ is the dynamic energy consumption of the ℓ^{th} gate being operated at supply voltage v_ℓ and W_ℓ is the average switching activity of the ℓ^{th} gate in a single clock cycle (assuming a non-pipelined adder).

The total energy consumption also includes the static energy consumption. The static energy consumption in a CMOS circuit is due to the leakage current between different nodes in a transistor when the transistor is not switching. In general we assume that the leakage current does not change over time and hence static energy consumption of a CMOS circuit per clock cycle is estimated to be a linear function of the clock cycle time. To estimate static energy consumption we use the following model

$$E^S = \sum_{\ell=1}^N P_\ell^S(v_\ell) D \quad (3.14)$$

where $P_\ell^S(v_\ell)$ is the static power consumption of the ℓ^{th} gate being operated at supply voltage v_ℓ and D is the clock cycle time of the circuit.

Therefore, the total energy consumption is the sum of both the dynamic energy consumption and the static energy consumption given by

$$E = \sum_{\ell=1}^N (E_{\ell}^D(v_{\ell})W_{\ell} + P_{\ell}^S(v_{\ell})D) \quad (3.15)$$

where N is the total number of gates in the adder.

3.9.2 Energy model for an approximate RCA

In this subsection we will present our approach to model energy consumption of an approximate RCA by extending the energy model for an RCA discussed in Section 3.9.1.

The total energy consumption of an RCA as described in Eq. 3.15 is

$$E = \sum_{\ell=1}^N (E_{\ell}^D(v_{\ell})W_{\ell} + P_{\ell}^S(v_{\ell})D)$$

For an approximate RCA, Eq. 3.15 may be used if we find the switching activities for the gates under overclocking. As described in Section 3.7.2, due to overclocking the sum actually read might be different from the correct sum. The fact of whether at a given bit position the correct sum bit was computed in time or not was modeled using the indicator function in Eq. 3.4 in Section 3.7.2. We will use a similar model of an indicator function to check if a particular gate in the RCA had a logic change within the clock cycle time and, based on that, re-evaluate (reduce) the switching activity to reflect this.

Consider an n -bit approximate RCA. Assume a carry chain starting from position i and ending at position j . Define an indicator function for computing the energy consumption as follows

$$I_k^E(\mathbf{d}, \mathbf{a}, \mathbf{b}, D) = \begin{cases} 1 & \text{if } d_{ik} > D \text{ and } \exists i, j \text{ such that } C_{ij}(\mathbf{a}, \mathbf{b}) = 1 \text{ and } i < k \leq j \\ 0 & \text{otherwise.} \end{cases} \quad (3.16)$$

Based on the gate level design of the ripple carry adder described in Section 3.7.1, we assume that in correct computation, soon after the inputs are provided to the adder all the sum bits are initially computed with all the carry bits as 0. ‡ We discuss the effect of non-zero carry bits on the energy modeling in Appendix A.

As the adder is computing, the correct carry bits propagate through the ripple carry adder, and then the sum bits might switch again to reflect the correct values. If I_ℓ^E is 1 then it denotes that the sum bit at position ℓ is incorrect at the end of the clock cycle time. This denotes that the carry did not propagate to position ℓ in time which means that the MUX gate in the full adder at position ℓ might have switched at most once and the XOR gate that computes the sum in the full adder at position ℓ switched at most once. Here we only need to consider the effect of a single carry chain at a particular bit position because multiple carry chains cannot overlap as per Observation 1. A carry chain starting from position i to position j occurs with a probability of p_{ij} .

We also know from the definition of switching activities in Section 3.4 that

$$W_k = \frac{\text{Total number of toggles of gate } k}{\mathcal{A}}$$

where $0 \leq k \leq N - 1$. Here we are using the indexing scheme that we defined in Section 3.7.1.

But if $I_\ell^E = 1$ then the number of toggles at gate k , where $k = 3\ell + 2$ or $k = 3\ell + 3$

‡To estimate the energy consumption model for an approximate RCA we assume that all the carry bits are 0 when the inputs are provided for each addition. But in reality the carry bits typically retain values from the prior computations and are not reset to 0 every time.

(using the indexing scheme defined in Section 3.4), decreases by 1. Hence

$$\begin{aligned}
 W_k^a &= \frac{(\text{Total number of toggles of gate } k) - 1}{\mathcal{A}} \\
 &= \frac{\text{Total number of toggles of gate } k}{\mathcal{A}} - \frac{1}{\mathcal{A}} \\
 &= W_k - \frac{1}{\mathcal{A}}
 \end{aligned}$$

Thus the average switching activities at the gates computing the sum bit and the carry bit at position ℓ decrease by $\frac{1}{\mathcal{A}}$. But because the carry chain occurs with a probability of p_{ij} the switching activities decrease by $p_{ij} \times \frac{1}{\mathcal{A}} = \frac{p_{ij}}{\mathcal{A}}$. But if $I_\ell^E = 0$ then the switching activities do not decrease.

Therefore including the value of the indicator function, we see that the switching activity of gate $k = 3\ell + 2$ and gate $k = 3\ell + 3$ should be reduced by $\frac{p_{ij} I_\ell^E}{\mathcal{A}}$. The above analysis is for a given carry chain $C_{ij}(\mathbf{a}, \mathbf{b})$. Averaging over all possible carry chains the total reduction value is $\sum_{0 \leq i < j \leq n-1} \frac{p_{ij} I_\ell^E}{\mathcal{A}}$.

Therefore, the corresponding switching activities of these gates in an n -bit approximate RCA should be decreased accordingly based on the following equations.

$$W_{3\ell+2}^a = W_{3\ell+2} - \left(\sum_{0 \leq i < j \leq n-1} \frac{p_{ij} I_\ell^E}{\mathcal{A}} \right) \quad (3.17)$$

$$W_{3\ell+3}^a = W_{3\ell+3} - \left(\sum_{0 \leq i < j \leq n-1} \frac{p_{ij} I_\ell^E}{\mathcal{A}} \right) \quad (3.18)$$

where $0 \leq \ell < n$. As per the design of an RCA described in Section 3.7.1, each full adder has 3 gates. Therefore the number of gates is $N = 3 \times n$. As we compute the switching activity for each gate, there would be $N = 3n$ unique switching activities. The switching activities are indexed based on the indexing scheme for gates defined in Section 3.7.1.

Example 9. Consider the 3-bit RCA shown in Fig. 3.7. We will compute the approximate switching activities (W_ℓ^a for $0 \leq \ell < N$) for the gates in this adder. The first step in

Table 3.3 : Maximum and minimum propagation delays of the XOR gate and the MUX in 90nm technology

<i>Gate</i>	delay _{min} (pico-sec)	delay _{max} (pico-sec)
XOR	33.3	55.2
MUX	30.5	51.2

computing \mathbf{W}^a is to compute the switching activities (\mathbf{W}) for an RCA that is not being overclocked. The total number of input combinations to a 3-bit RCA is $\mathcal{A} = 2^3 \times 2^3 = 64$. Using simulations of the 3-bit RCA, we compute the average switching activity at the output of each gate when there is no overclocking, which is the ratio of the number of toggles at the output the gate to the number of additions. This gives the average switching of each gate per addition.

Now we reduce these switching activities to take into account the effect of overclocking. For the sake of this example, let us consider that $C_{13}(\mathbf{a}, \mathbf{b}) = 1$. From the description of the sum path in Section 3.7.2 and the propagation delay values shown in Table 3.3 (repeated from Table 3.5 in Section 3.7.1), we know that $d_{01} = 97.1\text{ps}$ and $d_{02} = 127.6\text{ps}$. Let the clock cycle time, D , be 120ps. This means that from Eq. 3.16, $I_0^E = 0$, $I_1^E = 0$ and $I_2^E = 1$. Therefore from Eq. 3.17 and Eq. 3.18, the switching activities of gate 8 ($3 \times 2 + 2 = 8$), the XOR gate that computes the sum in the last full adder in Fig. 3.7, and gate 9 ($3 \times 2 + 3 = 9$), the MUX that computes the carry out in the last full adder, have to be reduced. The amount of reduction to the total number of toggles of these two gates is 1. But this is weighted with the probability of the carry chain, in this case that would be p_{13} . Also because we are computing the average switching activity we have to divide this by the total number of additions also, in this case we assume that $\mathcal{A} = 2^3 \times 2^3 = 64$.[§]

[§]Since it is a simple 3-bit adder our benchmark consists of all possible input cases. But in general the

Therefore the approximate switching activities of the two gates are $W_8^a = W_8 - \frac{p_{13}I_2^E}{\mathcal{A}}$ and $W_9^a = W_9 - \frac{p_{13}I_2^E}{\mathcal{A}}$, where $I_2^E = 1$, $p_{13} = (\frac{1}{2})^4$ (assuming uniform input probabilities) and $\mathcal{A} = 64$. \square

Our algorithm to re-evaluate the switching activities for an approximate RCA is shown in Algorithm 1. This algorithm takes as input the size of the adder (n), the probability that a particular carry chain occurs (\mathbf{p}) and the switching activities (\mathbf{W}) of the gates in a RCA without overclocking and computes the switching activities (\mathbf{W}^a) for an approximate RCA.

Based on the revised estimates of the switching activities, the total energy consumption of an approximate RCA is as follows

$$E^a = \sum_{\ell=1}^N (E_{\ell}^D(v_{\ell})W_{\ell}^a + P_{\ell}^S(v_{\ell})D) \quad (3.19)$$

Substituting the relationship between average dynamic energy consumption and worst-case propagation delays from Eq. 3.2 as described in Section 3.6 in Eq. 3.19, we get

$$E^a = \sum_{\ell=1}^N \left(\gamma_{\ell} \frac{1}{\epsilon_{\ell}^2(v_{\ell})} W_{\ell}^a + P_{\ell}^S(v_{\ell})D \right) \quad (3.20)$$

3.9.3 Summary to Section 3.9

In this section we have described the energy model we use for a CMOS ripple carry adder. We also presented our approach to extend the energy model to an approximate RCA using carry chains.

number of additions considered to compute the switching activities could be lower than all possible input cases if the number of input cases is prohibitively large.

Algorithm 1 Approximate Switching Activities

```

1: procedure SWITCHING ACTIVITY( $n, \mathbf{p}, \mathbf{W}$ ) ▷ Calculates  $\mathbf{W}^a$ , switching activities of
   the gates in an approximate RCA
2:   for  $0 < j \leq n - 1$  do
3:     for  $0 \leq i < j$  do
4:       for  $i < k \leq j$  do
5:         Compute  $d_{ik}$                                 ▷ Discussed in Section 3.7.2
6:         Compute  $I_k^E$                                 ▷ As defined in Eq 3.16
7:         if  $I_k \neq 0$  then
8:            $W_{3k+2}^a \leftarrow W_{3k+2} - \frac{p_{ij} I_k^E}{A}$ 
9:            $W_{3k+3}^a \leftarrow W_{3k+3} - \frac{p_{ij} I_k^E}{A}$ 
10:        else
11:           $W_{3k+2}^a \leftarrow W_{3k+2}$ 
12:           $W_{3k+3}^a \leftarrow W_{3k+3}$ 
13:        end if
14:      end for
15:    end for
16:  end for
17: end procedure

```

3.10 Minimizing Average Error of an Approximate RCA Using Geometric Programming

In this section we describe our procedure to formulate our target problem, which is minimizing average error of an approximate RCA under a given energy budget, as a geometric program. Then we present our approach to perform supply voltage binning on the solution

obtained from the geometric program.

3.10.1 Formulation of an optimization problem

In this subsection we formulate our problem, as described in Section 3.3.1, of minimizing average error of an approximate RCA with a given energy budget.

We form an optimization problem consisting of an objective function and one or more constraint functions. The independent variables are called the decision variables whose values are the solution to the optimization problem.

In our case, the objective function is the average error of an approximate RCA as given in Eq. 3.12 in Section 3.8. The average error as shown in Eq. 3.12 is a function of D , \mathbf{d} and the circuit topology. The clock cycle time D is an independent variable, but \mathbf{d} (described in Section 3.7.2) is a matrix whose elements are a function of the adder topology, resulting critical path delays and gate supply voltages. We do not alter the adder topology but instead vary the adder supply voltage which directly alters \mathbf{d} . The authors found a formulation of error optimization in terms of \mathbf{d} (represented in terms of $\epsilon(\mathbf{v})$) to be much simpler than a direct formulation in terms of \mathbf{v} .

Therefore we consider the propagation delays of the gates as the decision variables. The RCA under consideration consists of N gates. We need to compute an optimized *supply voltage allocation scheme*, which is the exact assignment of supply voltages to the individual gates. To do that we need to compute delays $\epsilon_1(v_1), \epsilon_2(v_2), \dots, \epsilon_N(v_N)$ for which the average error is minimized under the constraint that the total energy consumption is below the total energy budget. These gate delays will in turn determine the supply voltage allocation scheme.

The optimization problem is to minimize Eq 3.12 which is

$$Er_{\text{avg}}(D, \mathbf{d}) = \sum_{0 \leq i < j \leq n-1} p_{ij} Er(D, i, j, \mathbf{d}), \quad (3.21)$$

subject to the following two constraints and assumptions.

1. For each gate k (as per the indexing scheme defined in Section 3.7.1), $k = 1, 2, \dots, N$

$$\epsilon_k(v_{\min}) \leq \epsilon_k(v_\ell) \leq \epsilon_k(v_{\max}), \quad (3.22)$$

where the lower and the upper bounds depend on the transistor technology, the type of component and fanout. Please refer to Section 3.5 for discussion of the assumptions behind Eq. 3.22.

2. The total energy consumption of all the gate is bounded from above by the given energy budget. Thus,

$$E^a = \sum_{\ell=1}^N \left(\gamma_\ell \frac{1}{\epsilon_\ell^2(v_\ell)} W_\ell^a + P_\ell^S(v_\ell) D \right) \leq \text{Energy Budget}. \quad (3.23)$$

The left hand side part of the above inequality has been obtained from Eq. 3.20 in Section 3.9. In Eq. 3.23, the proportionality constants (γ_ℓ) and the static power consumption values ($P_\ell^S(v_\ell)$) are constants which depend on the process technology and transistor-level designs of the gates used in the RCA. Since we alter neither the process technology nor the transistor-level designs of the gates, γ_ℓ and $P_\ell^S(v_\ell)$ are constant as far as the optimization problem is concerned. The clock cycle time D and the Energy Budget are variables which we can determine. The propagation delays of the gates, $\epsilon_\ell^2(v_\ell)$, are the variables that can change during the optimization process.

The assumptions under which the above optimization problem is framed are as follows.

- (i) The gate topology of the circuit of the adder is considered as a given and is not changed during the optimization.
- (ii) Potentially the optimization problem can result in a solution that allocates each gate a unique supply voltage. But as described in Section 3.2, with multiple supply voltages the need for voltage level converters arises. So to make the design more pragmatic we will bin the solution from the optimization problem to a fixed set of voltages. Currently we do not account for the additional energy consumption due to the voltage level converters in the optimization problem.
- (iii) As discussed in Section 3.5, the propagation delays of the gates are considered to be worst-case non-varying deterministic delays.

In this subsection we formulated our target problem as an optimization problem by describing the objective function, the constraints and the assumptions.

3.10.2 How to minimize average RCA error using geometric programming

In this section we present our approach using a technique called geometric programming to increasing the accuracy at the output of the RCA by minimizing the function given in Eq. 3.21 in Section 3.10.1.

In general the full class of optimization problems could be classified into two categories, linear optimization problems (LP) and non-linear optimization problems (NLP). The objective function of our optimization problem in this chapter, which is shown in Eq. 3.12, is not a linear function. To clarify this point, let us observe the objective function.

$$Er_{\text{avg}}(D, \mathbf{d}) = \sum_{0 \leq i < j \leq n-1} p_{ij} Er(D, i, j, \mathbf{d}),$$

Substituting Eq. 3.7 we get

$$Er_{\text{avg}}(D, \mathbf{d}) = \sum_{0 \leq i < j \leq n-1} p_{ij} \left(\sum_{k=i+1}^j I_k 2^k \right). \quad (3.24)$$

In Eq. 3.24 it can be observed that the indicator function I_k is not a linear function and in fact it is not even a continuous function.

The authors have not been able to come up with a linear function to estimate/represent Eq. 3.24; therefore, the authors have so far not been able to apply traditional linear programming optimization techniques to this problem. The alternative is to model our problem as a nonlinear optimization problem (NLP). Although modeling our problem as an NLP is trivial but solving a general NLP is computationally difficult. In contrast, a sub-class of NLP known as geometric programs (GP) are easy to solve, and also a global solution can be achieved efficiently. There are also very effective and reliable methods to solve a GP. Hence we chose to compute an approximation of our target problem as a GP.

Therefore, our solution is to formulate the problem of minimizing Eq. 3.21 subject to the constraints outlined in Section 3.10.1 as a geometric program and then solve it. To further explain our procedure we present the definition of a particular type of function called as a monomial in Definition 2 and an extension of monomials known as a posynomial (short for positive polynomial) in Definition 3 [56].

Definition 2. Let x_1, \dots, x_n denote n real positive variables, and $\mathbf{x} = (x_1, \dots, x_n)$ a vector with components x_i . A real valued function f of \mathbf{x} , with the form

$$f(\mathbf{x}) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n},$$

where $c > 0$ and $a_i \in \mathbf{R}$, is called a monomial. [56]

Definition 3. A posynomial is a sum of one or more monomial functions. [56]

To model our target problem as a geometric program, the objective function and all the constraints should be in the form of a posynomial. But as can be observed from Eq. 3.24, our objective function is not a posynomial. So we will compute a posynomial approximation of our objective function based on the methodology given in Section 8.2 of [56]. The approach of computing a posynomial approximation of a given function and then using geometric programming to solve it is referred to as signomial programming, discussed in detail in [56].

As can be seen from Eq. 3.12, our objective function is not a continuous function because of the indicator functions. According to [56], to compute the posynomial approximation of our objective function we have to first compute its continuous approximation. Then we have to use a feasible initial guess to find a posynomial approximation of the continuous approximation of our objective function. In our case, we start with a feasible uniform voltage allocation scheme as the initial guess and compute a posynomial approximation of the objective function in Eq. 3.26. The following is a mathematical description of the approximations and redefinitions that we use.

To simplify the continuous approximation of the discontinuous function $Er_{\text{avg}}(D, \mathbf{d})$ (shown in Eq. 3.24), we will redefine the indicator function I_k (given in Equation 3.4) which is a part of the right hand side of Eq. 3.24 as follows:

$$l_k = \begin{cases} 1 & \text{if } d_{ik} > D, i < k \leq j \\ 0 & \text{otherwise.} \end{cases} \quad (3.25)$$

Using this definition, $Er(D, i, j, \mathbf{d})$ which is used in Eq. 3.21 (previously defined in Eq. 3.7) is transformed as follows

$$Er(D, i, j, \mathbf{d}) = \sum_{k=i+1}^j I_k 2^k = \sum_{k=i+1}^{j-1} l_k 2^k - l_j 2^j \quad (3.26)$$

where I_k is shown in Equation 3.4. Because the new indicator function l_k is a non-negative function, the negative sign appears in the definition of $Er(D, i, j, \mathbf{d})$. Thus the combination

of the indicator function in Eq. 3.4 and the error function in Eq. 3.7 in Section 3.8 results in the same value as the redefined indicator function in Eq. 3.25 and transformed error function in Eq. 3.26.

This redefinition allows us to make the indicator function a non-negative function so that it can be represented in terms of the signum function. We do this because a signum function can be approximated to a smooth continuous analytical function which we will then approximate to a posynomial as required for a geometric program.

We now represent l_k , by

$$l_k(i, D) = \frac{1 + \text{sgn}(d_{ik} - D)}{2}$$

where $\text{sgn}(x)$ is the signum function. For $\kappa \gg 0$, $\text{sgn}(x) \approx \tanh(\kappa x)$, and we use $\kappa = 200$.[¶]

Therefore,

$$l_k(i, D) \approx \frac{1}{2} + \frac{\tanh(\kappa(d_{ik} - D))}{2} = \frac{1}{1 + e^{-2\kappa(d_{ik} - D)}}$$

Thus, our continuous and differentiable approximation of Eq. 3.26 is

$$\begin{aligned} \text{Er}(D, i, j, \mathbf{d}) &= \sum_{k=i+1}^{j-1} l_k(i, D)2^k - l_j(i, D)2^j \\ &\approx \sum_{k=i+1}^{j-1} \frac{1}{1 + e^{-2\kappa(d_{ik} - D)}} 2^k - \frac{1}{1 + e^{-2\kappa(d_{ij} - D)}} 2^j \end{aligned} \quad (3.27)$$

where d_{ij} are linear functions of $\epsilon_k(v_k)$. We use the monomial approximation technique (Section 8.2 of [56]) for this expression. This results in

$$\begin{aligned} &\text{Er}(D, i, j, \mathbf{d}) \\ &\approx \sum_{k=i+1}^{j-1} \frac{1}{1 + e^{-2\kappa(d_{ik} - D)}} 2^k - \frac{1}{1 + e^{-2\kappa(d_{ij} - D)}} 2^j \\ &\approx c \epsilon_1^{a^1} \epsilon_2^{a^2} \dots \epsilon_N^{a^N} \end{aligned} \quad (3.28)$$

[¶]This particular value of κ was chosen empirically by observing the plots of the two functions, $\text{sgn}(x)$ and $\tanh(\kappa x)$, and that the transition from -1 to 1 is fast enough.

where $c \in \mathbf{R}^+$ and $a^m \in \mathbf{R}$ for all $1 \leq m \leq N$. The expression in Eq. 3.28 is a monomial as per Definition 2.

We then construct the objective function $Er(D, \mathbf{d})$ as a posynomial (defined in Definition 3) from Eq. 3.24.

As $\epsilon(v)$ are the decision variables, we now express \mathbf{d} in terms of $\epsilon(v)$, and write the average error as $Er_{\text{avg}}(D, \epsilon)$. Then the problem is reduced to minimizing a posynomial subject to posynomial inequality constraints, giving us a geometric program in a standard form:

$$\begin{aligned} \text{Minimize } Er_{\text{avg}}(D, \epsilon) &= \sum_{j=1}^C c_j \epsilon_1^{a_j^1} \epsilon_2^{a_j^2} \dots \epsilon_N^{a_j^N} \\ \text{subject to } \epsilon_k(v_{\min}) &\leq \epsilon_k \leq \epsilon_k(v_{\max}), \quad k = 1, \dots, N \\ \text{and } \sum_{\ell=1}^N \left(\gamma_{\ell} \frac{1}{\epsilon_{\ell}^2} W_{\ell}^a + P_{\ell}^S(v_{\ell}) D \right) &\leq \text{Energy Budget} \end{aligned}$$

where C is the number of possible carry chains in a n -bit adder and N is the number of lower level components (such as gates) in the adder. In the case of a $n = 16$ -bit adder, $C = \frac{n(n-1)}{2} = 105$.

We use a standard geometric programming toolbox [56, 57] to solve this program. The solution of the first iteration is used to compute the posynomial approximation again, until the objective value starts to converge. This gives us the final allocation of delays to the components such that the average error is minimum for the given constraints. Using the delays allocated to the components, we can obtain the voltages to be supplied to them.

To clarify our approach to minimize error using geometric programming, we present an example of a 3-bit adder in Example 10.

Example 10. Consider a 3-bit RCA as shown in Fig. 3.7. We will compute the objective function for the 3-bit RCA. The possible carry chains are $C_{02}(\mathbf{a}, \mathbf{b})$, $C_{01}(\mathbf{a}, \mathbf{b})$ and $C_{12}(\mathbf{a}, \mathbf{b})$.

Table 3.4 : Propagation delay and supply voltage values from the geometric program and corresponding binned supply voltage values for the gates in Fig. 3.7

Gate Index	ϵ_ℓ (ps)	v_ℓ (volts)	Binned v_ℓ (volts)
1	44.6	0.84	0.8
2	46.9	0.8	0.8
3	34.0	1.16	1.2
4	44.6	0.84	0.8
5	40.8	0.92	0.9
6	33.3	1.2	1.2
7	39.3	0.96	1.0
8	38.5	0.98	1.0
9	33.3	1.2	1.2

From Eq. 3.24,

$$Er(D, \mathbf{d}) = p_{01}Er(D, 0, 1, \mathbf{d}) + p_{12}Er(D, 1, 2, \mathbf{d}) + p_{02}Er(D, 0, 2, \mathbf{d})$$

Following Eq. 3.27, the objective function becomes

$$Er(D, \mathbf{d}) = p_{01} \left(-\frac{1}{1 + e^{-2\kappa(d_{01}-D)}} 2^1 \right) + p_{12} \left(-\frac{1}{1 + e^{-2\kappa(d_{12}-D)}} 2^2 \right) + p_{02} \left(\frac{1}{1 + e^{-2\kappa(d_{01}-D)}} 2^1 - \frac{1}{1 + e^{-2\kappa(d_{02}-D)}} 2^2 \right)$$

From Section 3.7.2 and Fig. 3.7, we know that $d_{01} = \epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_5(v_5)$, $d_{12} = \epsilon_4(v_4) + \epsilon_6(v_6) + \epsilon_8(v_8)$ and $d_{02} = \epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_6(v_6) + \epsilon_8(v_8)$. Therefore,

the objective function becomes,

$$\begin{aligned}
 Er(D, \mathbf{d}) = & p_{01} \left(-\frac{1}{1 + e^{-2\kappa(\epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_5(v_5) - D)}} 2^1 \right) \\
 & + p_{12} \left(-\frac{1}{1 + e^{-2\kappa(\epsilon_4(v_4) + \epsilon_6(v_6) + \epsilon_8(v_8) - D)}} 2^2 \right) \\
 & + p_{02} \left(\frac{1}{1 + e^{-2\kappa(\epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_5(v_5) - D)}} 2^1 - \right. \\
 & \left. \frac{1}{1 + e^{-2\kappa(\epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_6(v_6) + \epsilon_8(v_8) - D)}} 2^2 \right)
 \end{aligned}$$

As described in Section 3.10.2, we first start with an initial guess of uniform voltage allocation. For this example, say that all the gates are provided with the highest supply voltage of 1.2V and the corresponding minimum propagation delay values are shown in Table 3.5. Let us say that the clock cycle time is $D = 120\text{ps}$. Hence if the values for the propagation delay and D are substituted then it results in $\epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_5(v_5) < D$ and $\epsilon_4(v_4) + \epsilon_6(v_6) + \epsilon_8(v_8) < D$. Therefore,

$$\begin{aligned}
 \frac{1}{1 + e^{-2\kappa(\epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_5(v_5) - D)}} & \approx 0 \\
 \frac{1}{1 + e^{-2\kappa(\epsilon_4(v_4) + \epsilon_6(v_6) + \epsilon_8(v_8) - D)}} & \approx 0
 \end{aligned}$$

Thus the objective function becomes

$$Er(D, \mathbf{d}) = \left| p_{02} \left(\frac{1}{1 + e^{-2\kappa(\epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_6(v_6) + \epsilon_8(v_8) - D)}} 2^2 \right) \right|$$

because we only consider the absolute magnitude of the error. We also know from Section 3.8 that $p_{02} = (\frac{1}{2})^4$.

Now we use the monomial approximation technique (Section 8.2 of [56]) for this expression. For keeping the constants at a reasonable exponent (for example, not have 2.5×10^{-49} as a constant), we use all the delay values in nanosecond units. This results

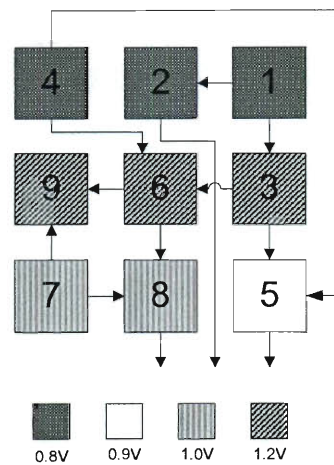


Figure 3.8 : A sample floorplan with 4 voltage islands based on the binned solution of a 3-bit RCA shown in Table 3.4

in the following posynomial approximation (in this case in fact it is a monomial) objective function:

$$Er(D, \mathbf{d}) = 0.2790\epsilon_1^{0.00833}\epsilon_3^{0.00762}\epsilon_6^{0.00762}\epsilon_8^{0.00833}$$

The final allocation of delays and their corresponding supply voltage values obtained through signomial programming [56] are shown in Table 3.4. The gate indices used in Table 3.4 are based on the indexing scheme defined in Section 3.7.1. The Energy Budget used to obtain this solution is 1.11×10^{-13} J.

□

In this section we present our solution approach which is a geometric problem based modeling of our target minimization problem. We also discuss about the specific toolbox that we use in this chapter.

3.10.3 Supply voltage binning

In this section we present our approach to binning the solution obtained from Section 3.10.2 to a specific set of voltages. We also compare the solutions of the geometric program and the binned solutions in the context of a ripple carry adder.

The solution from Section 3.10.2, in principle, can assign any voltage to any gate under the given constraints. For a practical application of the solution we need to limit the number of supply voltages and also the number of voltage islands.

We will first present our approach to limit the number of supply voltages. Let the possible set of supply voltages be \mathcal{V} and the maximum number of voltages be M_v . We then pick a M_v -combination of elements from \mathcal{V} . The voltages from this subset are then assigned to the gates in the RCA with gates having a higher voltage in the geometric program solution getting a higher voltage from this subset. This process is referred to as *binning*. We exhaustively search through all possible *binning* schemes. Using Eq. 3.24 and the relationship between propagation delay and supply voltage we can compute a closed form solution of the average error. Hence an exhaustive search is not computationally intensive. The algorithm that we use for supply voltage binning is shown in Algorithm 2. Let $P(\mathcal{V})$ be the power-set of the set \mathcal{V} and p an element of the power-set. Let m_i be the number of gates assigned with the i^{th} element of p . Without loss of generality assume that all the voltages in the sets are sorted in ascending order.

As an example, we show the binned voltages for the 3-bit RCA example in Table 3.4.

To determine the number of voltage islands we have to design the floorplan of the actual circuit. We agree that designing the floorplan and supply voltage binning concurrently could lead to a more efficient solution but we will target this problem in our future work. With regard to this work, we first bin the solution obtained from the geometric program and then design the floorplan accordingly to minimize the number of different voltage islands. This

Algorithm 2 Supply Voltage Binning

```

1: procedure BINNING( $P(\mathcal{V}), M_v$ )
2:   Sort all the gates in ascending order with respect to the supply voltage assigned by
   the geometric program
3:   for  $p \in P(\mathcal{V})$  do
4:     if  $|p| = M_v$  then
5:       for  $\forall m_i : \sum_{i=1}^{M_v} m_i = N, m_i \geq 1$  do
6:         Assign the first  $m_i$  unassigned gates with the  $i^{\text{th}}$  element ( $i^{\text{th}}$  voltage) of
            $p$ 
7:       end for
8:       if  $\sum_{\ell=1}^N \left( \gamma_{\ell} \frac{1}{\epsilon_{\ell}^2} W_{\ell}^a + P_{\ell}^S(v_{\ell}) D \right) \leq \text{Energy Budget}$  then
9:         Evaluate  $Er_{\text{avg}}(D, \epsilon(v))$  from Eq. 3.24
10:      end if
11:    end if
12:  end for
13:  Pick the  $p$  and  $m_i$ 's with the minimum  $Er_{\text{avg}}(D, \epsilon(v))$ 
14: end procedure

```

might result in some extended interconnects. But for this work, we currently assume that the overheads due to a few rare elongated interconnects is not very significant. Based on the supply voltage binning and acceptable overheads of multiple voltage lines we can fix the number of voltage islands. In Fig. 3.8 we present an example of a floorplan with four different voltage islands for the 3-bit RCA solution shown in Table 3.4. Each box in the figure represents a single gate. The digit in each box corresponds to the index of the gate as indicated in Fig. 3.7.

3.11 Simulation Framework for RCA Experimentation

In this section we describe the simulation framework that we used for our experiments through which we aim to show that the voltage assignments generated by our geometric program described in Section 3.10.2 after voltage binning have lower error when compared to corresponding uniform voltage scaling or a naive biased voltage scaling assignment.

We use Synopsys HSPICE Version B-2008.09 with Synopsys 90nm technology to design and simulate our RCAs. We discuss our models for energy consumption and average error in Section. 3.7. The range of voltages with which the gates in the circuit are operated is 0.8V to 1.2V. As discussed in Section 3.2 we realize that using multiple voltages may necessitate voltage level converters. Currently, we do not include voltage level converters in our simulations because our voltage shifts are usually very small (in the 0.1V – 0.2V range).

We simulate the circuits with different supply voltage configurations and obtain the average error magnitude and average energy consumption values. The average error magnitude for the experiments is computed by taking an average, over the number of additions performed in the experiment, of the absolute magnitude between the correct output of the approximate adder and the actual output of the approximate adder which might be different due to overclocking. The average energy consumption is measured from our HSPICE simulations by taking an average, over the number of additions, of the total energy consumption. The total energy consumption is computed as the sum of the integrals of the product of the current drawn and the magnitude of the supply voltage over the entire period of simulation.

The input data set for the experiments is drawn from a uniform distribution. To simulate the behavior of the adder we used 10,000 input combinations. We admit that the number of test cases is not very high, but we chose this number so that we could explore across multiple cases of voltage allocation schemes.

We designed transistor level models in HSPICE for the RCA based on the gate level

description from Section 3.7.1. We apply the procedure described in Sec. 3.10.2 to obtain a globally optimized supply voltage allocation scheme for the RCA. This scheme is then binned to specific supply voltage values using the approach in Section 3.10.3. For our experiments, to perform supply voltage binning we used specific voltage levels which are 0.8V, 0.9V, 1.0V, 1.1V and 1.2V.

The critical path delay of an adder is computed in HSPICE by providing the adder with worst-case inputs. For an adder a worst-case input is one in which there is a carry chain from position 0 to position $n-1$.

We use three cases for comparing the advantage of our approach. The three cases are

- Case 1: Uniform voltage scaling (UVoS) : All the gates in the RCA are assigned the same voltage. The voltage levels can vary from 0.8V to 1.2V at the granularity of 0.01V. Therefore there are 41 different voltage allocations.
- Case 2: Naive biased voltage scaling (n-BiVoS): For comparison, we will consider the biased voltage scaling (BiVoS) approach of George et al. [11] modified as follows. First, we split the number of bits equally into four sets: for 16 bits, there are four sets of four bits each, while for 32 bits, there are four sets of 8 bits each. Then, we tried the following possible combinations of four distinct voltages assuming a step size of 0.1V from 0.8V to 1.2V: (i) {0.8V, 0.9V, 1.0V, 1.1V}, (ii) {0.8V, 0.9V, 1.0V, 1.2V}, (iii) {0.8V, 0.9V, 1.1V, 1.2V}, (iv) {0.8V, 1.0V, 1.1V, 1.2V} and (v) {0.9V, 1.0V, 1.1V, 1.2V} where the voltages are assigned from lowest to highest from the LSB to the MSB. For example, 0.8V is the supply voltage for the least significant four bits (in the case of a 16-bit RCA) or eight bits (for the 32-bit RCA) in four out of five of the cases above. We call this approach "naive-BiVoS" or n-BiVoS for short.
- Case 3: Binned geometric programming solution (BGPS): The solution generated

Table 3.5 : Maximum, minimum propagation delays and proportionality constants of the XOR gate and the MUX in 90nm technology

<i>Gate</i>	delay _{min} (pico-sec)	delay _{max} (pico-sec)	γ_k (Jsec ²)
XOR	33.3	55.2	2.0E-35
MUX	30.5	51.2	1.6E-35

from the geometric program which is binned to limit the number of supply voltages.

The metrics that we compare are the average energy consumption and the average error magnitude. We also use the metric of relative error which is the ratio of error to the correct output, indicating how much the error affects the output. From these results we picked a few data points that illustrate the savings yielded by our methodology.

3.12 RCA Experimental Results

We first present the results for individual 16-bit and 32-bit ripple carry adders. Then we show the energy impact of a 16-bit globally optimized RCA in the context of an FFT.

3.12.1 Simulation results of 16-bit and 32-bit approximate ripple carry adders

In this section, we present the results for 16-bit and 32-bit addition using approximate RCAs.

Based on the circuit description of the RCA in subsection 3.7.1, the number of gates in a 16-bit RCA is 48 and in a 32-bit RCA is 96. We model these adders using the error model described in Section 3.7.2. The objective function for optimization, presented in Eq. 3.12, is computed efficiently in Section 3.8. The constraints for the geometric program, which are dependent on the technology are also described in Section 3.8. The first constraint

Table 3.6 : Summary of results of 16-bit and 32-bit approximate ripple carry adders

n -bit	D (ns)	Average Error Magnitude					Energy Consumption (fJ)	Energy Savings
		UVoS	n-BiVoS	BGPS	UVoS /BGPS	n-BiVoS /BGPS		
16-bit	0.4	36.83	50.06	21.66	1.70	2.31	110.78	1.41
16-bit	0.4	36.83	45.97	17.96	2.05	2.56	128.14	1.22
16-bit	0.4	36.83	38.55	26.31	1.40	1.47	132.9	1.18
16-bit	0.4	40.92	36.83	23.25	1.76	1.58	139.92	1.12
32-bit	0.6	18171	33704	13978	1.30	2.41	132.74	2.07
32-bit	0.6	15634	24637	11412	1.37	2.16	139.41	1.97
32-bit	0.6	14892	15215	11142	1.34	1.37	152.54	1.8
32-bit	0.6	14199	10931	8931	1.59	1.22	159.3	1.73

in Eq. 3.22 limits $\epsilon_k(v_k)$, the propagation delay of a single gate, between the maximum worst case delay (based on the lowest supply voltage allowed which is 0.8V for our target technology) and the minimum worst case delay (based on the highest supply voltage allowed which is 1.2V). The RCA circuit that we are using is built using two types of gates, the Exclusive OR (XOR) gate and a multiplexer (MUX). The minimum and maximum delays for these gates in our target technology, computed in HSPICE, are provided in Table 3.5. The second constraint in Eq. 3.23 limits the sum of energy consumption of all the gates. For our target technology, the constant γ_k in Eq. 3.23 has been computed for the two types of gates and is shown in Table 3.5.

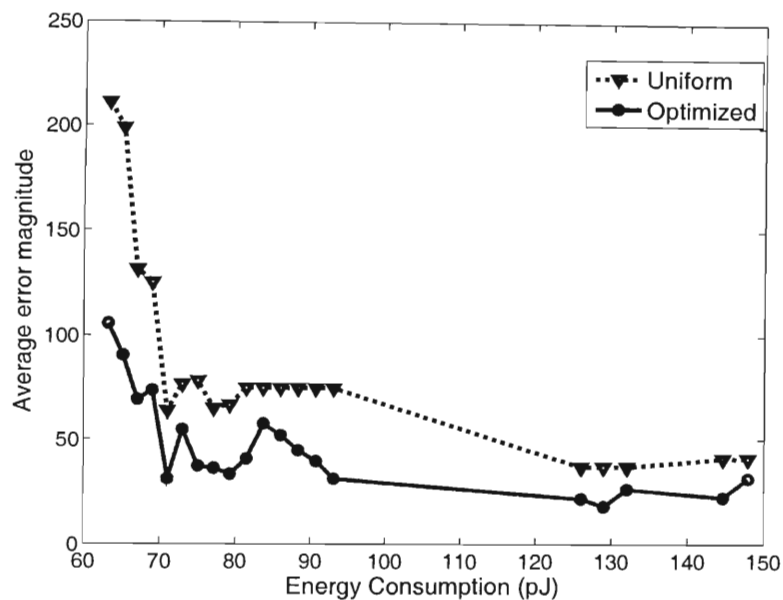


Figure 3.9 : Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 16-bit ripple carry adder

For the experiments that we present here for the 16-bit RCA we chose $4E-10$ sec as the clock cycle time (D). To describe the results, we chose an instance where 110 fJ is the Energy Budget in the geometric program for all the 48 gates. We present the results of our comparison for the three cases described in Section 3.11 in Table 3.6. The solution for Case 3 (BGPS) is where 18 gates had 0.8V, 11 gates with 0.9V, 8 gates with 1.0V and 11 gates with 1.2V.

Furthermore, we observed that a 16-bit RCA would have an energy consumption of 157pJ (by increasing the supply voltages to 1.12V) to have no overclocking errors and thus have 100% accuracy at the same frequency of 2.5Ghz and uniform voltage supply (which is the current design methodology). This shows that we achieve a 1.4X reduction in energy consumption by overclocking the adder and biasing the supply voltage, albeit at some loss of accuracy. These results are also shown in Table 3.6, where “Energy Savings” refer to the

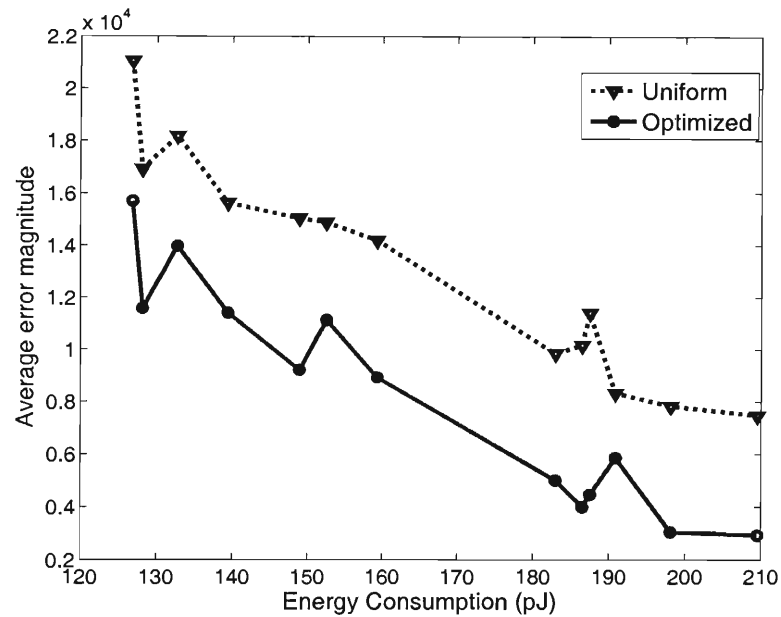


Figure 3.10 : Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 32-bit ripple carry adder

reduction in energy consumption of the globally optimized approximate adder with respect to the conventional correct adder operating with the same clock cycle time (D). Simulation results of a 16-bit RCA being overclocked at 2.5Ghz but for different values of Energy Budget in HSPICE are shown in Fig. 3.9.

The clock cycle time (D) for the 32-bit RCA has been chosen to be 6E-10 sec. To demonstrate the results, let us choose one instance of the 32-bit approximate RCA with an Energy Budget (in the geometric program) of 132 fJ for all the 96 gates. The solution that resulted after supply voltage binning of the geometric program solution is a 32-bit RCA where 39 gates with 0.8V, 5 gates with 0.9V, 24 gates with 1.1V and 28 gates with 1.2V. The results for a 32-bit RCA are shown in Table 3.6. Also, average error magnitude versus average energy consumption results for a 32-bit approximate RCA are shown in Fig. 3.10.

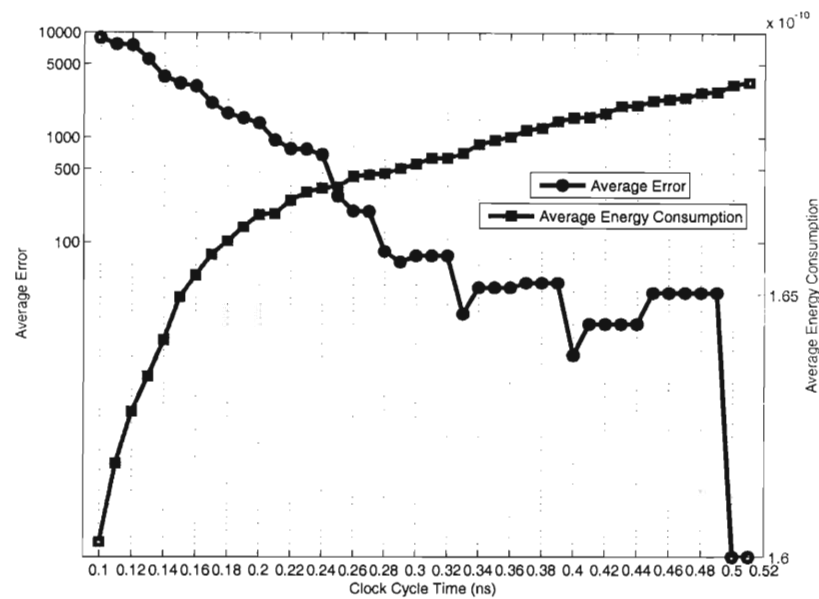


Figure 3.11 : Average error and average energy consumption versus clock cycle time for a 16-bit approximate RCA with constant and uniform supply voltage of 1.2V

3.12.2 Non-monotonic error rates

In this section, we will present some results to demonstrate the non-monotonicity of the average error magnitude with respect to the average energy consumption and clock cycle time of an approximate adder.

Examining Fig. 3.9 and 3.10 we see that the accuracy of the adder typically increases with increase energy consumption. But there are several deviations in the trend. Therefore accuracy does not monotonically increase with increasing energy consumption. This happens because the average error magnitude of an approximate adder does not always decrease with increasing the supply voltages of the gates in the circuit of the adder. This phenomena has been explained in Section 3.7.2.

To demonstrate the non-monotonicity of the average error at the output of an approxi-

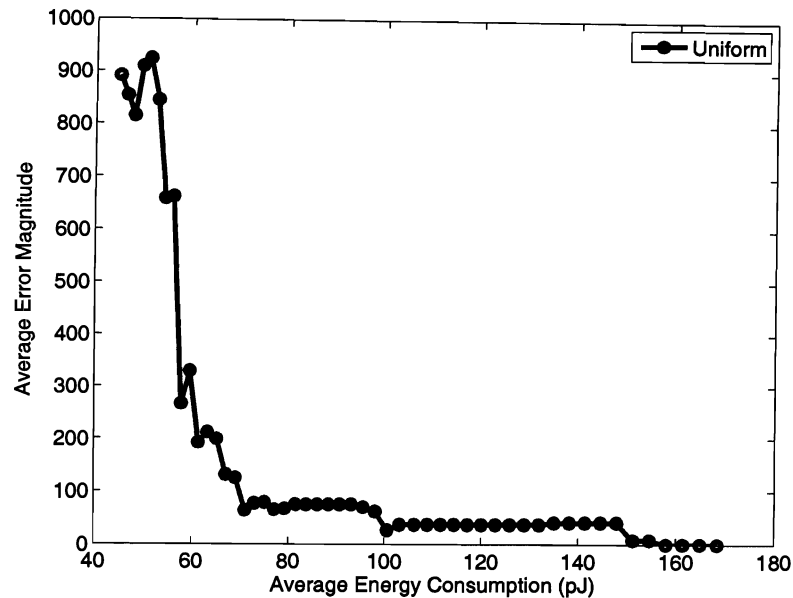


Figure 3.12 : Average error versus energy consumption for a 16-bit approximate RCA with constant clock cycle time of $4E-10$ sec

mate 16-bit RCA the behavior of average error versus clock cycle time is shown in Fig. 3.11. The results in in Fig. 3.11 were generated by keeping a constant and uniform supply voltage of 1.2V for the entire approximate RCA.

Also, as another example, we present the behavior of average error versus energy consumption for an approximate 16-bit RCA in Fig. 3.12 while keeping the clock cycle time constant at $4E-10$ sec. This was computed by simulating a 16-bit approximate RCA with different supply voltages starting from 0.8V till 1.2V with a granularity if 0.01V.

3.12.3 Approximate RCA FFT example

In this section we first describe the architecture of the 8-point FFT that we use for these experiments. Then we present simulation results of an 8-point FFT designed using conventional adders, uniform voltage scaled adders and globally optimized adders.

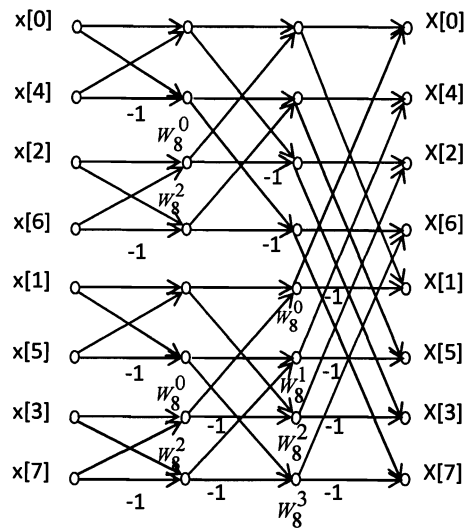


Figure 3.13 : Flow graph of a complete decimation-in-time decomposition of a 8-point FFT

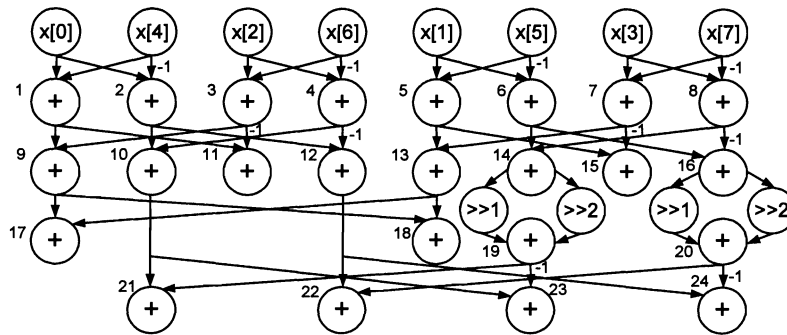


Figure 3.14 : Graph-theoretic representation of an 8-point FFT

Architecture of an 8-point FFT

In this subsection we will definition of a discrete Fourier transform and the architecture of the 8-point FFT that we use.

The fast Fourier transform (FFT) is a discrete Fourier transform (DFT) algorithm which reduces the number of computations needed for P points from $2P^2$ to $2P \lg P$. The DFT

can be represented as follows

$$X[k] = \begin{cases} \sum_{n=0}^{N-1} x[n] W_N^{kn} & \text{if } 0 \leq k \leq N - 1 \\ 0 & \text{otherwise.} \end{cases}$$

where x is the input which is a finite duration sequence (of length N) of complex numbers, X is the DFT of x , and $W_N = e^{\frac{-2\pi i}{N}}$.

We use a complete decimation-in-time decomposition [58] of an 8-point FFT. A flow graph of this 8-point FFT is shown in Fig. 4.6. We use techniques [59] to transform multiplications in the 8-point FFT to additions of shifted inputs. We do this transform because in this chapter we target only approximate adders and do not optimize approximate multipliers. Also, it has been shown by Zhou et al. [60] that multiplier-less FFTs have lower number of computations than the FFTs with both multipliers and adders. The transformed 8-point FFT with 24 fixed point real 16-bit ripple carry adders is shown in Fig. 4.7.

We described the mathematical formulation of the DFT. We also discussed about the design of the 8-point FFT and the transformation we used to convert the multipliers into adders.

Experimental results of an approximate 8-point FFT

The approximate FFT was simulated in MATLAB by artificially introducing errors at the outputs of adders in the FFT based on the average error values from the simulations described in Section 3.12.1. As input to the FFT, we used the image shown in Fig. 3.15(a) with a resolution of 100X100 pixels.

We use a 16-bit approximate RCA but using the image data as input to the adder. Then we collect average error for the three types of approximate adders through simulations in HSPICE using the same framework as described in Section 3.11 except for the input data.

We use a Gaussian noise source at the output of every adder in MATLAB to simulate the effect of overclocking with the mean and variance collected from HSPICE simulations of the RCA. This will result in an approximately computed FFT of the input image. We then perform a correct inverse-FFT in MATLAB of this approximate FFT of the input image. Our goal is to see the extent to which the data has been preserved in this experiment. We would expect, if both the FFT and the inverse-FFT were correct, that the final image would be an exact copy of the original image.

The different cases that we compare are

- Case 1: Conventional design where all the adders are being operated at 100% accuracy.
- Case 2: BGPS: The voltage allocation scheme generated by mathematical formulation and the optimization scheme presented in this chapter.
- Case 3: n-BiVoS: The voltage allocation scheme where the adder is divided into 4 equal bins with voltages assigned from the set 0.8, 0.9, 1.0, 1.1, 1.2V.
- Case 4: UVoS: Approximate adders in the circuit have uniform voltage allocation but still being operated at the same frequency.

The resultant images from the four cases that are discussed above are shown in Fig. 3.15. The image generated by the FFT using BGPS adders has a PSNR that is 15db lower than the image generated by the FFT using UVoS adders with a similar energy consumption. Also the image generated by the n-BiVoS adders is 8.5 dB lower than the image generated by the BGPS adders. All these adders have a similar energy consumption.



Figure 3.15 : Images generated by (a) Case 1 (b) Case 2 (c) Case 3 and (d) Case 4

3.13 Additional Terminology for Carry Lookahead Adders

In this section, we will redefine some terms that have been defined in Section 3.4 specifically in the context of an RCA. Most of the terms can be trivially applied to the case of a carry look ahead adder by changing the context from an RCA to a CLA. But some terms which require specialization in the context of a CLA are presented in this section. These redefined terms are relevant as a reference from Section 3.14 to Section 3.18.

1. Carry chain : Given an n -bit CLA and two specific n -bit binary numbers \mathbf{a} and \mathbf{b} as inputs to the CLA, a carry chain is said to be present from position i to position j if and only if

- $a_i = b_i = 1$.
- $a_w \neq b_w$.
- $a_j = b_j$.

where $0 \leq i < w < j \leq n - 1$. We admit that in a CLA there are multiple paths through a carry bit is computed and hence analogous to the case of an RCA a physical instantiation of a chain of carry bits does not take place in a carry lookahead adder. And hence we will define a carry chain independent of the actual circuit and computation because it is used to identify the possibility of an error in our model.

2. C_{ij} : Consider an n -bit CLA and two specific n -bit binary numbers \mathbf{a} and \mathbf{b} as inputs to the RCA. A boolean variable C_{ij} is defined on whether there is a carry chain starting from position i and ending at position j such that

$$C_{ij} = \begin{cases} 1 & \text{if there is a carry chain from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases} \quad (3.29)$$

where $0 \leq i < j \leq n - 1$.

3. N : Represents the number of gates in the CLA. For simplicity we will consider a carry lookahead block in a carry lookahead adder as a single gate.
4. d_{pr} : Given an n -bit CLA and two specific n -bit binary numbers \mathbf{a} and \mathbf{b} . Assume that $C_{pq} = 1$ for $p < r < q$. Then we define $d_{pr} = t_r - t_{in}$. d_{pr} is the time elapsed from the instant when the inputs are provided to the CLA till the correct sum bit, s_r , is generated.
5. \mathbf{d} : Represents an $n \times n$ matrix where the r^{th} element in the p^{th} row is d_{pr} . Elements d_{pr} for which $p \geq r$ are invalid and so are not considered.

The terms that we have defined in this section serve as a reference for the models and procedure described from Section 3.14 till Section 3.18.

3.14 CLA Delay Assumptions

The basic assumptions that we discuss in the case of RCA in Section 3.5 still hold in the case of a CLA.

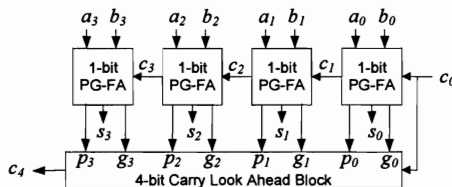


Figure 3.16 : Diagram of a 4-bit carry lookahead adder

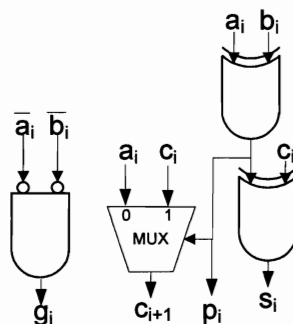


Figure 3.17 : Diagram of a 1-bit Propagate-Generate full adder (1-bit PG-FA)

3.15 Energy and Error Models for an Approximate CLA Adder

In this section we first present a brief discussion of a carry lookahead adder. This is followed by a description of the mathematical framework and the models that we use to characterize both the energy consumption as well as the average error at the output of an approximate CLA.

3.15.1 Description of a carry lookahead adder

In this section a brief description of a carry look ahead is provided.

Weinberger and Smith [61] proposed the “carry lookahead” scheme in 1958. The carry lookahead scheme in contrast to the carry-rippling technique speeds up the propagation

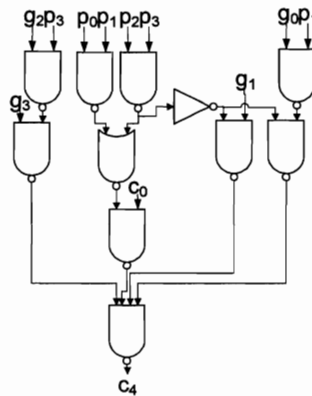


Figure 3.18 : Diagram of a 4-bit Carry Look Ahead Block

of carry by additional logic. An example of a 4-bit binary adder with 4-bit carry lookahead logic is shown in Fig. 3.16. In a carry lookahead adder, 1-bit propagate-generate full adders (1-bit PG-FAs), shown in Fig. 3.17, are used to generate the additional bits required by the carry lookahead logic, as described in Fig. 3.18. These additional bits are called the propagate ($p_i = a_i \oplus b_i$) and generate bits ($g_i = a_i \cdot b_i$) and hence the name, 1-bit propagate-generate full adder. For more information on propagate bits, generate bits and carry lookahead logic please refer to [62].

3.15.2 Energy consumption model for an approximate CLA

The energy model that we use for a CLA is similar to our energy model for an RCA with one exception. We do compute the reduced switching activities for the XOR and MUX gates that compute the sum and carry bits, as shown in the 1-bit PG FA shown in Fig. 3.17, using the same algorithm as in the case of an RCA. But at this time we do not consider the reduced switching activities of the gates in the carry lookahead block because that would mean considering the time taken from the inputs to each gate in the carry lookahead block.

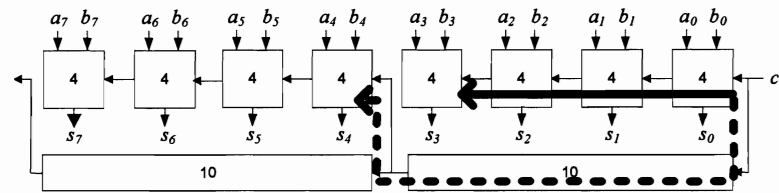


Figure 3.19 : Illustration of the paths and delay of the carry bit across a carry-lookahead adder

Also from our simulations we have observed that the reduction in switching activities is less than 8% and hence can be neglected. Also our target problem is not affected since we only specify an Energy Budget which is an upper bound and hence an overestimation does not violate the constraint.

3.15.3 Modeling the error at the output of an approximate CLA

In this section, we will first present boolean logic functions that we use in our design of the CLA to compute the sum output. We will then describe carry chains in a carry look ahead adder. We specifically concentrate on the differences from our analysis of the RCA. We present the primary difference in our modeling of the CLA which is the characterization of the critical path for each sum bit, also referred to as the “sum path”. We describe the mathematical formulation for the error function which is the error at the output of an approximate adder.

CLA logic

In this subsection we describe one set of (from among many) boolean logic functions of addition in the context of a CLA.

We will consider n -bit CLA for some n . Numbers to be added will be in the range

$0, \dots, 2^n - 1$ and will have the standard binary representation.

Consider the addition of two n -bit numbers \mathbf{a} and \mathbf{b} , resulting in an $(n + 1)$ -bit number, consisting of $\mathbf{s} = s_n \dots s_0$. The main difference in a CLA from an RCA is that group propagate and group generate bits are computed from the individual propagate and generate bits. The boolean equations are as follows.

$$p_i = a_i \oplus b_i$$

$$g_i = a_i \cdot b_i$$

$$P_{[i:j]} = P_{[i:k]} \cdot P_{[k+1:j]} = p_i \cdot p_{i+1} \dots p_j$$

$$G_{[i:j]} = G_{[i:k]} \cdot P_{[k+1:j]} + G_{[k+1:j]}$$

Please refer [62] for comprehensive description of boolean logic functions of addition in a carry lookahead adder.

Carry chains in carry lookahead adders

In this section, we will discuss about carry chains in the context of a carry lookahead adder.

In an n -bit carry look ahead adder, there are carry look ahead blocks as described in Section 3.15.1. Therefore in a CLA the carry does not “ripple” through the adder in the way the carry does in a RCA. Every carry look ahead block acts as a look ahead for a specific set of positions. Let us say there is a carry look ahead block for bit positions starting from position x to position y , where $0 \leq x < y \leq n$. This carry look ahead block computes the carry input for the full adder at position $y + 1$ much quicker than the time it would usually take the carry to propagate through all the full adders between position x and position y . Thus the carry takes different paths to reach full adders at different bit positions. This is described using Example 11 where instances of two different paths taken by a carry bit is described in a carry look ahead adder.

Example 11. To illustrate the concept of a carry chain in a carry look ahead adder, consider the example of a 8-bit carry look ahead adder shown in Fig. 3.19. Assume that there is carry chain from position 0 to position 6. In this case, the carry input for the full adder at position 3 is generated at position 0 and has to propagate through position 1 and then through position 2 arriving at position 3. But the carry input for the full adder at position 5 is computed by the carry look ahead block and is directly propagated to the full adder at position 4 which then propagates it to the full adder at position 5. \square

If there is a carry chain from i to j , we will set a boolean variable C_{ij} as defined in Section 3.13. And if $C_{ij} = 1$, similar to the case of an RCA, we have $s_i = 0 \oplus c_i$; $c_k = 1$ and $s_k = 0$, for $k \in \{i + 1, \dots, j - 1\}$; and $c_j = 1$, $s_j = 1$, and $c_{j+1} = a_j (= b_j)$.

The relationship between CLA carry chains and overclocking errors

In this section, we will discuss the behavior of error at the output of an approximate CLA in the context of a carry chain. We will describe this behavior using an example of a CLA.

The description in Section 3.7.2 about RCA carry chains is also valid in the case of a CLA. This means that even in the case of a CLA as the supply voltage(s) to the gates in the circuit of a CLA are increased there is a possibility that the error at the output of a CLA might also increase.

But there is one other consideration which is specific to a CLA and does not apply to an RCA. Consider a carry chain from i to j , where $0 \leq i < j < n$. The point that we wish to note is that there are cases where the carry bit may propagate to position k_2 before it propagates to k_1 for $i < k_1 < k_2 \leq j$. An example of such behavior, which seems to be counter-intuitive, is presented in Example 12.

Example 12. For example, consider a 8-bit carry lookahead adder (CLA), as depicted in Fig. 3.19, where the number in each box corresponds to the time it takes for it to produce

the output from the time it gets the input. We add 01010101 and 10101010 and we assume that $c_0 = 1$. Then, this carry propagates throughout the adder, but it propagates to $k_1 = 3$ in 12 time units from c_0 (solid black line in Fig. 3.19) and to $k_2 = 4$ in only 10 time units from c_0 (broken black line in Fig. 3.19). \square

Critical path of a sum bit in a carry chain of a CLA

The definition of a “sum path” in the case of an RCA can be applied to a CLA with some changes. As defined in Section 3.4, d_{ik} is the time between the correct computation of the sum bit s_k and the time when the inputs are provided to the circuit, provided $C_{ij} = 1$ for $i < k < j$. To determine d_{ik} as sum of propagation delays of the gates in an adder, ϵ_ℓ , we choose a specific design of a CLA as described in Section 3.15.1.

In an n -bit carry look ahead adder with l -bit look ahead blocks, assume that $C_{ij} = 1$ for $0 \leq i < j \leq n$. In this case, we will again compute d_{ik} . As described earlier in this section, the path that the carry takes for each full adder might be different. But based on our assumption that $C_{ij} = 1$, we can specifically determine the exact path as shown earlier in this section. The carry path might contain one or more carry look ahead blocks. If the carry path has a carry look ahead block, then the critical path for the correct computation of the sum bit s_k would include the computation of the propagate and generate bits from the 1-bit PG-FAs, as explained in Section 3.15.1. For simplicity we will assume a carry look ahead block as a single unit for the sake of simplicity in the description of a sum path. In such a scenario, there are two types of carry paths that are possible to the k^{th} full adder as described in Example 11. The two types of sum paths with reference to Section 3.15.3 are described below.

1. Carry Path Type 1: There is no lookahead block in the path. The path path consists of

- (a) The top XOR gate in the full adder (referring to Fig. 3.17) at position i .
 - (b) The MUX gates in full adders at positions from $i + 1$ till $k - 1$.
 - (c) The bottom XOR gate that computes s_k in full adder at position k .
2. Carry Path Type 2: There is at least one lookahead block in the path. In this case the sum path consists of
- (a) Maximum of (i) AND gate (referring to Fig. 3.17) in the full adder at position i and (ii) XOR gate in the full adders at position $t > i$ and are connected to the first lookahead block.
 - (b) Carry lookahead blocks
 - (c) The MUX gates in full adders from the position where the last carry lookahead block provides input to a full adder till the full adder at position $k - 1$. If the last lookahead block connects to the full adder at position k then there is no MUX gate in the carry path.
 - (d) The bottom XOR gate that computes s_k (referring to Fig. 3.17) in full adder at position k .

Based on the particular sum path type, d_{ik} is defined as the sum of the worst-case propagation delays of the gates (and lookahead blocks) in the sum path.

Let us model the adder as a graph $G(V, E)$ where V is the set of vertices (in this case they are gates) and E is the set of edges (or interconnects in the circuit).

Definition 4. Given a directed acyclic graph $G(V, E)$, the inverted graph $G'(V', E')$ is defined as follows:

$$V' = V \quad (3.30)$$

$$\text{Let } v_i, v_j \in V \text{ then } (v_i, v_j) \in E' \text{ iff } (v_j, v_i) \in E \quad (3.31)$$

In general for a given adder design, $G(V, E)$, all possible sum paths for position k , at the gate level, can be computed by performing a Breadth First Search (BFS) algorithm originating from the particular gate that computes the sum bit s_k on the “inverted” graph $G'(V', E')$.

The result of the BFS would contain all the paths that influence the output. The path(s) that has(have) the longest length is(are) assumed to be the most demanding and hence the critical path for that sum bit. And thus only the longest path(s) is(are) considered in the computation of error at the output of the adder. This analysis is performed only once and thus is computationally not very expensive. Thus d_{ik} would be the *max* of the sum(s) of the propagation delays of the the gates (vertices) in the longest path(s).

Adder error based on an analysis of carry chain errors

The procedure to develop the function for the error at the output of a CLA given \mathbf{a} , \mathbf{b} , the topology of the CLA, ϵ_k and D is exactly similar to the procedure described for an RCA in Section 3.7.2.

Adder error at the output of a CLA is

$$Er(D, \mathbf{a}, \mathbf{b}, d) = \sum_{k=0}^{n+1} I_k 2^k. \quad (3.32)$$

where

$$I_k(\mathbf{a}, \mathbf{b}, d, D) = \begin{cases} 1 & \text{if } d_{ik} > D \text{ and } \exists i, j \text{ such that } C_{ij}(\mathbf{a}, \mathbf{b}) = 1 \text{ and } i < k < j \\ -1 & \text{if } d_{ik} > D \text{ and } \exists i, j \text{ such that } C_{ij}(\mathbf{a}, \mathbf{b}) = 1 \text{ and } i < k = j \\ 0 & \text{otherwise.} \end{cases}$$

3.16 Efficient Evaluation and Minimization of Average Error in an Approximate CLA

Our approach to compute the average of the error at the output of an approximate CLA over all possible inputs and the technique that we follow to efficiently compute this average error is completely identical to that of an RCA. The constraints are also the same.

Our approach using geometric programming to increasing the accuracy at the output of the adder by minimizing the function given in Eq. 3.32 is identical to our approach for an RCA.

3.17 Simulation Framework for CLA Experimentation

In this section we describe the simulation framework that we used for our experiments on a CLA through which we aim to establish the validity of the solutions generated by geometric program described in Section 3.16.

The framework that we use for the experiments for a CLA is similar to the framework, described in Section 3.11, for an RCA.

The only difference in the case of a CLA is the case of multiple sum paths. But in any case, once all the unique sum paths are computed after that at every iteration of the geometric program we just need to pick the longest path.

3.18 Simulation Results for CLAs

We first present the results for individual 16-bit and 32-bit carry lookahead adders.

Table 3.7 : Summary of results of 16-bit and 32-bit approximate carry look ahead adders

n -bit	D (sec)	Average Error Magnitude		Energy Consumption (pJ)	Energy savings
		BGPS	UVoS		
16-bit	4E-10	559	1229.8	77.74	2.55
32-bit	8E-10	18300.3	10964.7	218.8	2.96

3.18.1 Simulation results of 16-bit and 32-bit approximate carry lookahead adders

In this section, we compare the average energy consumption and the average error magnitude for 16-bit and 32-bit addition using approximate CLAs with globally optimized voltage allocation versus a uniform voltage allocation scheme. To demonstrate the generality of our methodology, we applied the approach that was applied to a 16-bit and 32-bit approximate RCAs to 16-bit and 32-bit CLA.

The 16-bit CLA has 48 gates and 4 lookahead blocks. The CLA contains gates other than the XOR gate and the MUX and hence we compute $\epsilon_k(v_{\min})$ and $\epsilon_k(v_{\max})$ were computed for the carry look ahead block through HSPICE simulations. With an Energy Budget of 6.7E-13 J we note that up to 2.25X reduction in average error magnitude for the same average energy consumption of 77.74 pJ is realized through our method. A CLA would have an energy consumption of 175.99 pJ to have 100% accuracy at the same frequency and uniform voltage supply, indicating a savings in average energy consumption of 2.26X for our design, when compared to the conventional design. The results of the 16-bit approximate CLA with a clock cycle time of 4E-10 sec are shown in Fig. 3.20. Also the results of 32-bit approximate CLA with a clock cycle time of 8E-10 sec are shown in Fig. 3.21.

The best results among the data points that we simulated for 16-bit and 32-bit approximate CLAs are summarized in Table 3.7.

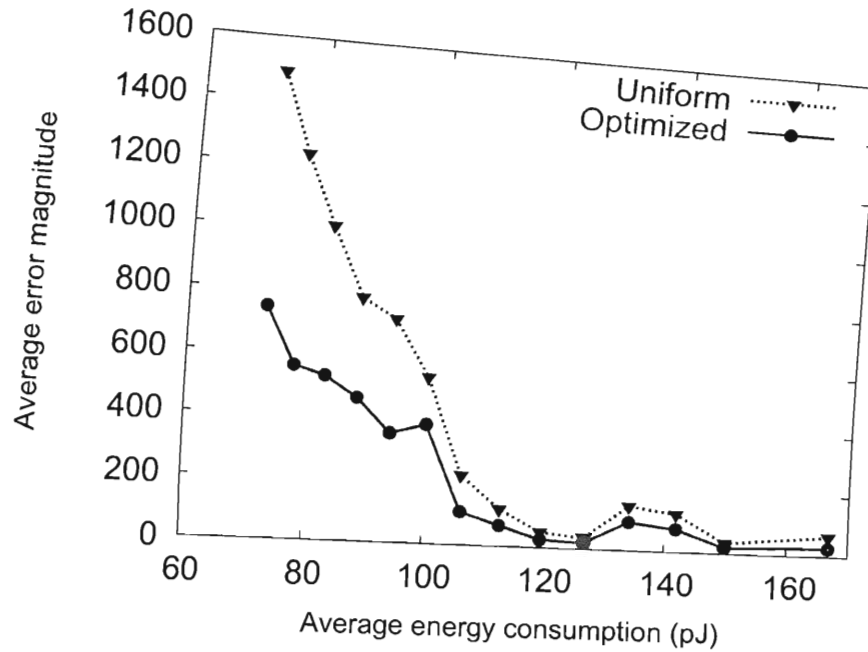


Figure 3.20 : Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 16-bit carry lookahead adder

3.19 Optimizing General Circuits

In the previous sections, we have modeled the propagation of errors in an approximate adder and based on that optimized the allocation of multiple supply voltages to various gates in the circuit.

Now, we present a technique to model the propagation and effect of errors in a general circuit. Based on this, we extend the geometric programming approach to optimally identify supply voltages to various components (which could be gates or a collection of multiple gates).

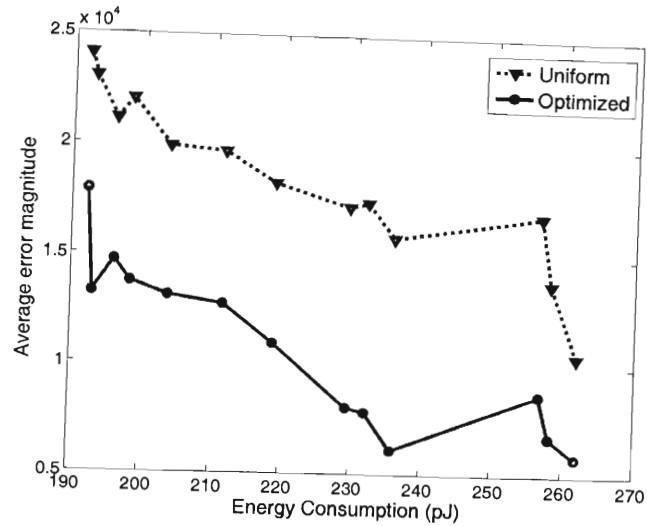


Figure 3.21 : Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 32-bit carry lookahead adder

3.19.1 Model of a General Circuit

For arbitrary circuits, we will characterize the error based on propagation of errors in the presence of overclocking. The circuit is built of some lower level components, such as gates. The average transition delay of component ℓ will be denoted by ϵ_ℓ . (We assume that the average transition delay of a component is inversely proportional to its supply voltage). The circuit is given time D for each addition. Inputs are provided to the circuit at some time t and the result is read at time $t + D$.

For a given circuit and a particular input (i) to it, define

$$d_l(\epsilon, i) = \{ \text{sum of } \epsilon_k | k^{\text{th}} \text{ gate is in the critical path of computation of output } l \}$$

The output o is read at time $t + D$ with D independent of the inputs. Due to overclocking, which we assume, the output actually read, o' , may be different from o . We now proceed to

characterize the error $o' - o$. We define an *indicator function* as follows:

$$I_l(\epsilon, i, D) = \begin{cases} 1 & \text{if } d_l(\epsilon, i) > D \\ 0 & \text{otherwise.} \end{cases}$$

Essentially, $I_k \neq 0$ if critical path delay of output l is more than D which means that the output present might not be the correct output.

The upper bound on the estimated error in the output for this particular input (i) is

$$Er(d, i, D) = \sum_{l=1}^{\theta} I_l(\epsilon, i, D) \gamma_l \quad (3.33)$$

where θ is the total number of outputs of the given circuit.

Here γ_l is the weight associated with the output. If the output is a binary number then $\gamma_l = 2^l$. If the error at the output of the circuit is measured by hamming distance, then $\gamma_l = 1$.

There are a couple of assumptions that were made while formulating Eq. 3.33. Basically we assume that the error is worst-case in two possible ways.

1. It is being assumed that if $d > D$ there will be an error, because even if the critical path is higher than the computation time the output might be correct.
2. The formulation of error in Eq. 3.33 considers that all bits that are different between the output and the correct answer will contribute positively to the error (which is the case in case of hamming distance) but might not be the case when the output is a binary number.

In retrospect, both of the above conditions are taken into consideration in the formulation of the average error of an approximate adder.

Now we compute the average of the output error over all possible input cases. Let \mathcal{J} denote the set that contains all possible inputs. Then

$$Er = \sum_{i \in \mathcal{J}} Er(d, i, D) \quad (3.34)$$

3.19.2 The Average Error Optimization Problem

Hence the optimization problem that we need to solve is as follows. It has constraints which are similar to the ones explained in the Section 3.10.1.

Minimize Er

such that $\epsilon_k(v_{\min}) \leq \epsilon_k \leq \epsilon_k(v_{\max})$

$$\sum_{\ell=1}^N \left(\gamma_{\ell} \frac{1}{\epsilon_{\ell}^2} W_{\ell}^a + P_{\ell}^S(v_{\ell}) D \right) \leq \text{Energy Budget}$$

To solve the above optimization problem, we propose to use geometric programming [56] which is an extended version of linear programming.

Hence we have to compute a posynomial approximation of Er . To do this, we approximate I_l as a continuous and differentiable function,

$$I_l = \frac{1 + \text{sgn}(d_l - D)}{2}$$

where $\text{sgn}(x)$ is the signum function. Therefore,

$$I_l \approx \frac{1}{2} + \frac{\tanh(\kappa(d_l - D))}{2} = \frac{1}{1 + e^{-2\kappa(d_l - D)}}.$$

We use the monomial approximation technique (Section 8.2 of [56]) for the above expression. This leads to

$$I_l \approx \frac{1}{2} \left(\frac{d_l}{D} \right)^{\frac{\kappa}{2}} = \frac{1}{c} \left(\frac{d_l}{D} \right)^{\alpha}$$

where α and c are some constants.

Thus a continuous and differentiable form of $Er(d, i, D)$ which is in the form of a posynomial is

$$Er(d, i, D) = \sum_{l=1}^{\theta} \frac{1}{c} \left(\frac{d_l}{D} \right)^{\alpha} \gamma_l$$

Thus the problem has been reduced to minimizing a posynomial subject to posynomial inequality constraints, giving us a geometric program in standard form. We use a standard geometric programming toolbox [56, 57] to solve this program. In contrast to the case of an approximate adder, this program is a straightforward geometric program and hence using signomial programming, which is an iterative geometric programming solution, is not necessary.

Thus the solution of the above program gives the optimal delays for the components under the physical limits that we impose. The respective magnitudes of the supply voltages can be inferred from these values because there is a very standard understood dependence between the supply voltage and the delay of standard logic gates.

3.20 Impact on Circuit Design

The novel methodology provides the circuit designer an efficient way to decide on the optimal design parameters in approximate adders. As mentioned at the beginning of Chapter 3, there has been no definitive method for allocating supply voltages to approximate arithmetic circuits. This means that the designer has had to exhaustively explore across all possible voltage allocation schemes, which is not practical for other than modest size circuits.

Our design automation solution, through geometric programming, very quickly gives a *quantitative* comparison of the relative importance of the components in an adder and assign supply voltages that can be selected by the designer.

Thus we enable the improvement of the output quality of low power approximate arithmetic adders by intelligently allocating voltages and thus make an attempt to tackle the issues of reliability and process variations.

3.21 Principal Thesis

Our primary contribution is a tractable solution methodology to automatically assigning supply voltages, while obeying design constraints, to components in approximate adders, so that the average error of the adder is minimized. The work is applicable to any current adder design of any size.

The primary thesis from this work is that brute-force assignment of multiple voltage levels in an adder might lead to an inefficient design. Hence an intelligent way of assigning voltage levels based on the relative importance of the gates is necessary. Also this approach shows that there exists an assignment that is theoretically the best but might not be very practical. So the theoretical solution has to be adjusted based on the specific constraints a designer might have.

Chapter 4

Modeling and Optimization of Dataflow Graph of Approximate Adders

To improve the accuracy of the output of the circuit for the same energy consumption, as opposed to uniform voltage scaling, a novel *biased* voltage scaling approach or BIVOS was proposed in [12], where *more important* data is computed more accurately and accuracy is less for *less important* data. For a single adder, this is done by having a higher supply voltage for the most significant bits and a lower supply voltage for the less significant bits.

First, though energy savings were obtained in [11, 12, 47] by biased investment at the level of an adder, there was no definitive methodology to *optimally* supply the voltage across the circuit. Second, there was no previous attempt to optimize a circuit that consisted of multiple components where each of these components could be an n -bit adder. In such circuits, analogous to the case of the computed bits in an adder [11, 12], the relative importance of these components has to be taken into account in order to optimize energy consumption. For example, it is the case that in some circuits, modeled in this chapter, the data produced by a particular adder is more important than other adders.

4.1 Primary Contributions

In this chapter, we address the latter problem of optimizing a circuit with multiple adders by showing a very efficient method to relatively invest energy across different adders in a circuit based on their *importance*. For supplying voltages for components inside an adder,

we use the previous BIVOS approach. Our method is general enough to model any adder design or combination of different adder designs, though we present our specific results for the special case of the ripple carry adder. Our primary contributions in this chapter are as follows:

- We provide a strong mathematical foundation capable of modeling the propagation of errors in a circuit with multiple shifters and approximate adders. Because of the generality of our result, this is applicable to any type of inexact adders, probabilistic [11] or approximate [12].
- We present a very fast algorithm to *quantitatively* compute the relative importance of each adder in any graph as defined in the model.
- We present a theorem that *optimally* distributes energy based on the relative importance of each adder, applicable to any directed acyclic graph structure.
- We demonstrate this approach on two example circuits, a *finite impulse response filter* (FIR) and a *Fast Fourier Transform* (FFT), and through HSPICE simulations we show that dramatic savings in energy consumption can be achieved when using our approach even when compared to the best existing prior art, BIVOS.

Keeping in mind the domain of DSP, we developed our approach to encompass popularly used circuits such as an FIR or an FFT which can be implemented using only adders and constant-number multipliers. Also, the standard implementation of a constant-number multiplier uses a set of adders and implicit shifting. Hence, in this chapter we will consider circuits which consist only of adders and shifters. We only consider optimization of energy and errors in dataflow graphs and do not model memory and feedback elements.

In Section 4.2, we present our target circuit model and state the associated optimization problem. In Section 4.3, we develop our solution to minimizing error for a given energy

budget. We summarize the method and describe the extension to other adders in Section 4.4. In Sections 4.5 and 4.6, we show the impact of the solution on two applications of interest, an FIR and an FFT. Section 4.7 discusses the impact our method has on a conventional circuit design framework.

4.2 The Model

In this section we define the model we use and specify the problem we address. The concept of trading error for energy savings in circuits is entirely at an early stage and thus, we felt a need for a foundational model and formal mathematical results, leaving detailed simulation of large systems for later.

As discussed, we will consider circuits of adders only. Some of these circuits have been obtained by reducing constant-number multipliers to a set of adders and shifters. This is advantageous because in many circuits used in DSP, values are multiplied by constants and hence general multipliers are not needed. They can be replaced with a set of adders and shifters using a variety of methods such as those used in [59], thereby reducing total area, energy consumed, and delay. In the design of such a circuit, shifting of a number is done implicitly by routing the interconnects to shift the bits appropriately to another position.

But in modeling we will explicitly consider and show shifters as they influence the size of the errors. The implicit implementation of the shifter S_1 shown in Fig. 4.1 which shifts a number to the left by 2 positions is shown in Fig. 4.2, where a rectangle represents a full adder and the number inside the rectangle denotes its position in the adder.

4.2.1 The graph based model

In the context of modeling, it is useful to consider shifters explicitly though they are not actually implemented. A circuit we target consists of the following components: inputs,

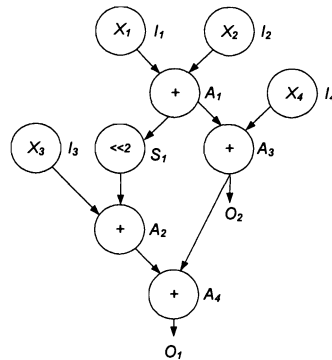


Figure 4.1 : Example of a graph-theoretical representation of a circuit

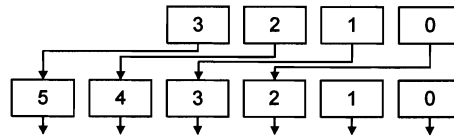


Figure 4.2 : An example of an implicit shifter

shifters and adders. Some adders or shifters are also labeled as outputs. It is convenient to model such a circuit using a directed acyclic graph (DAG).

The DAG will have some $N_I + N_A + N_S$ vertices: *inputs* $I_1, I_2, \dots, \text{and } I_{N_I}$; *adders* $A_1, A_2, \dots, \text{and } A_{N_A}$; and *shifters* $S_1, S_2, \dots, \text{and } S_{N_S}$. Some of the adders or shifters are also labeled as *outputs* O_1, O_2, \dots, O_{N_O} . Thus each A_i or S_j is either not an output or a unique O_k , so O_k is just an alias for A_i or S_j . Each O_k may have any number of bits but this will typically be a power of two. We will write n for N_I . For a simple example, see Fig. 4.1.

A vertex I_i has in-degree 0, and an input x_i to the circuit is supplied at I_i . An input x_i may have any number of bits but will typically be a power of two. A vertex A_j has in-degree 2 and is an adder, adding the two numbers on its incoming arcs. A vertex S_j has in-degree 1 and shifts the number on its incoming arc either left or right by the specified amount. We

will use the term s -shifter for a shifter vertex that shifts the input left with magnitude s (where s is a positive or negative integer).

At each adder or shifter, a linear function of x_1, x_2, \dots, x_n is computed. Thus the value of the output at O_k (corresponding to some adder or shifter) is a function $F_k(\mathbf{x}) = \sum_{i=1}^n w_{k,i} x_i$ for some $w_{k,1}, w_{k,2}, \dots, w_{k,n}$, which can also be written as $\mathbf{w}_k \cdot \mathbf{x}^T$ where $\mathbf{w}_k = (w_{k,1}, w_{k,2}, \dots, w_{k,n})$, $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and “ \cdot ” is the scalar product. \mathbf{x} is drawn from a set \mathbf{X} of allowable inputs. For example, each x_i could be an integer in the range $[0, 2^{10})$.

For Fig. 4.1, we see that $F_1(\mathbf{x}) = (5, 5, 1, 1) \cdot \mathbf{x}^T$.

4.2.2 The energy optimization problem for our target dataflow graph of adders

Let E be the *total energy budget* to be invested in the circuit, and let E_{Add} be the *energy required to run an adder correctly* for all possible inputs. We assume that $E < N_A E_{\text{Add}}$, and therefore we cannot assure that all adders run correctly for all inputs, and for different inputs \mathbf{x} , different error values may appear. Let E_j be the energy actually supplied to adder A_j . Given some choice of E_1, E_2, \dots , and E_{N_A} , such that $\sum_{j=1}^{N_A} E_j = E$, and some input \mathbf{x} , the resulting expected error for $F_k(\mathbf{x})$ is denoted by $Er(O_k, \mathbf{x})$. We use the term expected error here to consider the effect of temperature and process variations in the parameters of the circuit, but we will refer to it just as error in the later sections. We rely on [12] for calculation of expected error of a single adder. Our task in this chapter is to distribute E among the adders so as to

$$\text{minimize avg}_{\mathbf{x} \in \mathbf{X}} \sum_{k=1}^{N_O} Er(O_k, \mathbf{x}). \quad (4.1)$$

We refer to this problem as the *single resource dataflow energy-error optimization* problem, and our methodology to solve this problem is presented in Section 4.3.

4.3 The Single Resource Dataflow Energy-Error Optimization Method

To model the interactions and effect of multiple adders producing errors in the outputs of a circuit, we first consider just a single adder producing error in the circuit to define some metrics that we use later in the optimization.

4.3.1 A single approximate adder

We consider the implications of a single adder that produces an error (for at least some inputs). We will refer to such an adder as an *approximate adder*. We will see that the same errors in different adders may have different implications for errors at an output. Consider some input \mathbf{x} and assume that only one adder, A_{Er} , produces an “approximate” result since the inputs were supplied. This error propagates, and can cause an error of at an output O_k for $k \in \{1, 2, \dots, N_O\}$. We look at the value of the outputs after time t_O units since the inputs were supplied. Let $t(A_{Er}, k)$ be the time it takes for the output of A_{Er} to propagate to O_k . Let $Er(A_{Er}, \mathbf{x}, t)$ and $Er(O_k, \mathbf{x}, t)$ be the errors at the output of A_{Er} and output O_k after t units of time since the inputs were supplied with $Er(A_{Er}, \mathbf{x}, t) \neq 0$. Then we define the *significance* of A_{Er} , $\sigma(A_{Er}, \mathbf{x}, t_O, t(A_{Er}, k))$ as follows:

$$\sigma(A_{Er}, \mathbf{x}, t_O, t(A_{Er}, k)) = \frac{\sum_{k=1}^{N_O} Er(O_k, \mathbf{x}, t_O)}{Er(A_{Er}, \mathbf{x}, t_O - t(A_{Er}, k))}. \quad (4.2)$$

Let us refer to Fig. 4.1 again. Let the adders A_1, A_2, A_3 and A_4 be 8-bit ripple carry adders (RCAs) and the inputs be $x_1 = 15, x_2 = 1, x_3 = 0$, and $x_4 = 0$. Let the worst case delay of a full adder be 10 units of time, and therefore an 8-bit RCA has a critical path delay of 80 units. With this specific input, A_1 would take 50 units to compute the correct output in the worst case because the carry has to propagate across 5 full adders. For this output to propagate till O_1 , it would take about 70 units to propagate through the other adders as Fig. 4.1 describes a combinational circuit. But if we overclock the circuit such that we

sample O_1 after 50 units, the answer would only be *approximate*.

To compute the error at the output after 50 units of time, consider A_1 . After 30 units, the output of adding $00001111 = 15$ and $00000001 = 1$, assuming exact worst-case delays, is $00000100 = 8$. The output value of A_1 propagates to the output of A_2 by 40 units and it has a value of $8 \times 2^2 = 32$ when it reaches the left input to A_4 . Also the output value of A_1 propagates to the output of A_3 by 40 units and has a value of 8 when it reaches on the other input of A_4 . Thus, the value at the output of O_1 would be $32 + 8 = 40$ after 50 units of time whereas the correct output is $16 \times 2^2 + 16 = 80$. As a result, there is an error of 40 and the significance of A_1 to O_1 is equal to $40/8 = 5$ (where 8 is the error of the approximate adder after 30 units). This example shows that though the first adder had enough time to compute the correct value (by the end of 50 units of time A_1 would have computed correctly), this value did not have enough time to propagate through the rest of the circuit and thus to impact the final output.

Note that the exact magnitude of error is dependent on the time at which the outputs are sampled and the propagation time from the approximate adder to the particular output. But the quantity that we are interested is the significance of the approximate adder which is the amount by which the error at the approximate adder is amplified (or reduced) when it has propagated to the output. As is evident from the above example, this significance value is dependent only on the circuit topology and not the exact magnitude of errors. We strengthen this concept in Section 4.3.2 where we evaluate the significance of an adder based solely on the circuit topology independent of the individual errors. So we will omit the time parameter in the following discussions.

4.3.2 Computing the significance of an adder

Consider some circuit C , some energy budget E , and some input \mathbf{x} , such that there is exactly one approximate adder (all other adders compute correctly), and denote that adder by A_{Er} . This adder may cause errors in various vertices, and for vertex v and input \mathbf{x} , the error will be denoted by $Er(v, \mathbf{x})$. Then, the *significance* of A_{Er} to v under \mathbf{x} is defined by

$$\sigma(A_{Er}, v, \mathbf{x}) = \frac{Er(v, \mathbf{x})}{Er(A_{Er}, \mathbf{x})}. \quad (4.3)$$

Of course, if there is no path from A_{Er} to v , then $\sigma(A_{Er}, v, \mathbf{x}) = 0$. We will now prove some useful relations. First we note, that,

$$\sigma(A_{Er}, A_{Er}, \mathbf{x}) = 1. \quad (4.4)$$

Let $v \neq A_{Er}$. No errors can propagate to a circuit's inputs, so we will consider only shifters and adders. Assume that v is an s -shifter with an immediate predecessor u . Then as a shifter does not introduce errors but may *amplify* (or reduce) them, $Er(v, \mathbf{x}) = 2^s Er(u, \mathbf{x})$, and therefore

$$\sigma(A_{Er}, v, \mathbf{x}) = \frac{Er(v, \mathbf{x})}{Er(A_{Er}, \mathbf{x})} = 2^s \frac{Er(u, \mathbf{x})}{Er(A_{Er}, \mathbf{x})} = 2^s \sigma(A_{Er}, u, \mathbf{x}). \quad (4.5)$$

and if $\sigma(A_{Er}, u, \mathbf{x})$ does not depend on \mathbf{x} , neither does $\sigma(A_{Er}, v, \mathbf{x})$.

Assume that v is an adder with immediate predecessors u and w . Adder v is not A_{Er} and therefore does not introduce errors. Then,

$$\begin{aligned} \sigma(A_{Er}, v, \mathbf{x}) &= \frac{Er(v, \mathbf{x})}{Er(A_{Er}, \mathbf{x})} = \frac{Er(u, \mathbf{x}) + Er(w, \mathbf{x})}{Er(A_{Er}, \mathbf{x})} \\ &= \frac{Er(u, \mathbf{x})}{Er(A_{Er}, \mathbf{x})} + \frac{Er(w, \mathbf{x})}{Er(A_{Er}, \mathbf{x})} \\ &= \sigma(A_{Er}, u, \mathbf{x}) + \sigma(A_{Er}, w, \mathbf{x}). \end{aligned} \quad (4.6)$$

and if $\sigma(A_{Er}, u, \mathbf{x})$ and $\sigma(A_{Er}, w, \mathbf{x})$ do not depend on \mathbf{x} , neither does $\sigma(A_{Er}, v, \mathbf{x})$.

From Eqs. 4.4–4.6, by simple inductive argument, it follows also that the significance of a vertex is always greater or equal to 0 and *it does not depend on the value of the input \mathbf{x} that caused the error at A_{Er}* ! It is purely a property of the circuit's structure. Therefore we can write just $\sigma(A_{Er}, v)$ for the significance of A_{Er} to v no matter what \mathbf{x} is (though sometimes it may be convenient to write it explicitly).

Similarly, it is easy to see, that the $\sigma(A_{Er}, \mathbf{x})$ does not depend on \mathbf{x} , so we can just write $\sigma(A_{Er})$.

By referring to Fig. 4.1, we can provide intuition for significance and its properties. Assume that for some energy budget E_1 and input \mathbf{x}_1 , A_1 produced an error of δ_1 , and adders A_2 , A_3 , and A_4 processed their summands correctly. Then A_2 produced an error of $4\delta_1$, A_3 of δ_1 , and A_4 of $5\delta_1$. So, e.g., $\sigma(A_1, A_4, \mathbf{x}_1) = 5\delta_1/\delta_1 = 5$. But similarly, if for E_2 and input \mathbf{x}_2 , A_1 produced an error of δ_2 , and adders A_2 , A_3 , and A_4 processed their summands correctly, still $\sigma(A_1, A_4, \mathbf{x}_2) = 5\delta_2/\delta_2 = 5$.

We can also discuss the relative importance of the correctness of adders. If for some energy budget A_3 produced an error of δ , and adders A_1 , A_2 , and A_4 processed their summands correctly, $\sigma(A_3, A_4, \mathbf{x}) = \delta/\delta = 1$, so for A_4 correctness of A_1 is more important than that of A_3 .

For each v we define an *amplification factor*, $AF(v)$, to help produce a simple algorithm for computing significances, as

$$AF(v) = \begin{cases} 1, & \text{if } v \text{ is an input or an adder} \\ 2^s, & \text{if it is an } s\text{-shifter.} \end{cases} \quad (4.7)$$

We extend the definition to paths of vertices, by

$$AF(v_{j_1}, v_{j_2}, \dots, v_{j_k}) = \prod_{i=1}^k AF(v_{j_i}). \quad (4.8)$$

For vertices u and v , we will denote by $P(u, v)$ the set of all the paths from u to v .

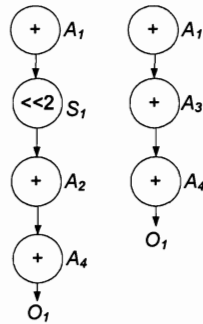


Figure 4.3 : The two paths from A_1 to $A_4 (= O_1)$. Error of δ at A_1 , while propagating through the left path contributes an error of 4δ to error at O_1 and an error of 1δ while propagating through the right path. Thus the total error at O_1 is 5δ and the significance of $A_1 = 5\delta/\delta$.

We next give a very easily computable, explicit formula for computing $\sigma(A_{\text{Er}}, v)$.

Lemma 4.3.1. *Assume that under some energy budget E and input \mathbf{x} a circuit has exactly one approximate adder, A_{Er} . Let v be any vertex. Then*

$$\sigma(A_{\text{Er}}, v) = \sum_{p \in P(A_{\text{Er}}, v)} AF(p).$$

EXAMPLE. Before starting the formal proof, we look at an example. Let us consider the graph in Fig. 4.1 with $A_{\text{Er}} = A_1$ and $v = O_1$. There are two paths from A_1 to O_1 , the “left” path $p_L = (A_1, S_1, A_2, O_1)$ and the “right” path $p_R = (A_1, A_3, O_1)$ as shown in Fig. 4.3.

Assume that an error of δ is generated by A_1 . Consider p_R first. On this path δ is passed through A_3 and the incoming error at O_1 is δ . Consider p_L now. On this path δ is converted to $2^2\delta$ by S_1 , the error of $2^2\delta$ is passed through A_2 and the incoming error at O_1 is $2^2\delta$. The total error at O_1 is $(2^2 + 1)\delta = 5\delta$.

By the definition in Eq. 4.8, we have $AF(p_L) = 1 \cdot 2^2 \cdot 1 \cdot 1 = 2^2$ and $AF(p_R) = 1 \cdot 1 \cdot 1 = 1$ and therefore $\sum_{p \in P(A_1, O_1)} AF(p) = 4 + 1 = 5$.

But by Eq. 4.3 (recall that \mathbf{x} can be omitted), $\sigma(A_1, O_1) = \frac{5\delta}{\delta} = 5$. Therefore the

Lemma holds for this example.

Proof. To shorten the proof, we skip over simple “pathological” cases, such as the case of one vertex connected by two outgoing arcs to a single adder.

As already noted, if there is no path from A_{Er} to v , i.e., $P(A_{Er}, v) = \emptyset$ then $\sigma(A_{Er}, v) = 0$. Therefore the claim holds in such cases.

We prove the lemma by induction on N , the number of vertices in the circuit.

The smallest circuit of interest has $N = 3$ vertices, two inputs feeding one adder. So this will be our base case. Here $v = A_{Er}$ and $P(A_{Er}, A_{Er})$ consist of only one path of length 1, namely (A_{Er}) . From Eq. 4.4, $\sigma(A_{Er}, A_{Er}) = 1$ and as by Eq. 4.7, $AF(A_{Er}) = 1$, the claim holds.

Let now $N > 3$ and assume that the lemma holds for all circuits with at most $N - 1$ vertices. Consider any circuit C of N vertices and remove from it any vertex v of out-degree 0 together the arcs incoming to it, resulting in a new circuit D . We will also use C and D as subscripts to indicate to which of the two circuits we are referring.

Note that, in general, if A_{Er} has out-degree 0, then again any v of interest is just A_{Er} itself, and similarly to the base case, $\sigma(A_{Er}, A_{Er}) = AF(A_{Er}) = 1$ and the claim holds. Therefore, if the removed vertex was A_{Er} , we already know that the claim holds for C , so consider the case where A_{Er} was not the removed vertex, and therefore it is also in D .

There are two cases for v . If v is an s -shifter, it has one predecessor, say u . There is a one-to-one correspondence between paths in $P_D(A_{Er}, u)$ and those in $P_C(A_{Er}, v)$. Every path p in the latter set is obtained by extending exactly one path q in the former set by v . By Eqs. 4.7–4.8, $AF(p) = 2^s AF(q)$ and therefore $\sum_{p \in P_C(A_{Er}, v)} AF(p) = 2^s \sum_{q \in P_D(A_{Er}, u)} AF(q)$. From Eq. 4.5, $\sigma_C(A_{Er}, v) = 2^s \sigma_D(A_{Er}, u)$, and the claim follows.

If v is an adder, it has two predecessors, say u and w . Note that as a path cannot end both with u and w , $P_D(A_{Er}, u) \cap P_D(A_{Er}, w) = \emptyset$. There is a one-to one correspondence between

paths in $P_D(A_{Er}, u) \cup P_D(A_{Er}, w)$ and those in $P_C(A_{Er}, v)$. Every path p in the latter set is obtained by extending exactly one path q in the former set by v . By Eqs. 4.7–4.8, $AF(p) = AF(q)$, and therefore $\sum_{p \in P_C(A_{Er}, v)} AF(p) = \sum_{q \in P_D(A_{Er}, u)} AF(q) + \sum_{q \in P_D(A_{Er}, w)} AF(q)$. From Eq. 4.6, $\sigma_C(A_{Er}, v) = \sigma_D(A_{Er}, u) + \sigma_D(A_{Er}, w)$, and the claim follows.

By induction, for any vertex z in D the claim holds. Since v was of out-degree 0, no path from A_{Er} to z in C can pass through v , v cannot “impact” any other vertex in C , and therefore, $Er_C(z, \mathbf{x}) = Er_D(z, \mathbf{x})$. As, of course $Er_C(A_{Er}, \mathbf{x}) = Er_D(A_{Er}, \mathbf{x})$, it follows that $\sigma_C(z) = \sigma_D(z)$. By induction $\sigma_D(A_{Er}, z) = \sum_{p \in P_D(A_{Er}, z)} AF(p)$, and as $P_C(A_{Er}, z) = P_D(A_{Er}, z)$ the claim holds for all such vertices z . \square

Theorem 1. *Assume that under some energy budget E and input \mathbf{x} a circuit has exactly one approximate adder, A_{Er} . Then,*

$$\sigma(A_{Er}) = \sum_{k=1}^{N_o} \sum_{p \in P(A_{Er}, O_k)} AF(p). \quad (4.9)$$

Proof. Immediate from Eq. 4.2 and Lemma 4.3.1. \square

To explicitly demonstrate the application of Theorem 1 to compute the significance (relative importance) of each adder we use the graph in Fig. 4.1. The relative significance values from Eq. 4.9 for each output from each vertex is shown in Table 4.1.

The above method to compute the significance of an adder, $\sigma(A_{Er})$, by Eq. 4.9, is very fast. A simple method of computing $AF(p)$ is to do a breadth-first search [63] from each vertex and count all paths from the vertex to the outputs. This would be a $O((V + E)V)$ operation where V is the number of vertices and E is the number of arcs in the graph.

Corollary 1. *Assume that under some energy budget E and input \mathbf{x} a circuit that contains no shifters (explicit or implicit) has exactly one approximate adder, A_{Er} . Then, using $||$ to*

Table 4.1 : Significance values of the adders in the graph shown in Fig. 4.1

A_{Er}	$\sigma(A_{Er}, O_1)$	$\sigma(A_{Er}, O_2)$	$\sigma(A_{Er})$
A_1	5	1	6
A_2	1	0	1
A_3	1	1	2
A_4	1	0	1

denote cardinality,

$$\sigma(A_{Er}) = \sum_{k=1}^{N_o} |P(A_{Er}, O_k)|.$$

Proof. If there are no shifters, $AF(p) = 1$ for any path p . □

4.3.3 Multiple approximate adders

Until now, we have considered only the case when one adder produces an error while adding its summands. In general, a set of adders can produce errors. The cumulative effect of this set of adders produces an error in an output whose absolute value is between 0 and the sum of the absolute values of the individual errors, as they may partially (or fully) cancel each other out.

Modeling this phenomenon of errors canceling out is complex and also partially depends on the particular input. Hence to simplify the analysis we will target to minimize the *the worst possible case*, in which the errors do not cancel each other to any extent. Thus, we want to minimize the sum of the absolute values of the errors.

4.3.4 Case study: Ripple carry adder

The discussion in Sections 4.2 and 4.3 holds for any type of adder. To proceed, however we will need to choose specific designs of adders, whose error production under various energy investment has been studied. This leads to considering the Ripple Carry Adder (RCA) for which such results exist.

It has been shown in [12], through simulations over a large number of input cases, that the average expected error $Er(A_i)$ for a RCA A_i with BIVOS, is roughly proportional to its delay D , i.e. $Er(A_i) \propto D$. In a conventional CMOS transistor, the delay of a transistor D_T is inversely proportional to its supply voltage VDD , i.e. $D_T \propto VDD^{-1}$.

The dynamic energy consumed during a transition of a transistor switch E_T is proportional to the square of its supply voltage, i.e., $E_T \propto VDD^2$. Also, the delay of adder A_i , D , is proportional to the delay of a transistor D_T , and the dynamic energy E_i consumed by adder A_i is proportional to the energy of a transistor E_T . Thus $E_i \propto E_T \propto VDD^2 \propto D_T^{-2} \propto D^{-2} \propto Er(A_i)^{-2}$ and for some constant r , we can write

$$E_A \cong r \frac{1}{Er^2(A_i)}. \quad (4.10)$$

4.3.5 Optimizing energy distribution

We will make use of a solution to an optimization problem which we present next.

Lemma 4.3.2. *Let integer $n > 0$ and $c, a_1, a_2, \dots, a_n > 0$. Then, the function $\sum_{j=1}^n a_j x_j$ subject to constraints $x_j > 0$ for $j = 1, \dots, n$ and $\sum_{j=1}^n x_j^{-2} = c$ is minimized at*

$$x_i = \frac{\left(\sum_{j=1}^n a_j^{2/3}\right)^{1/2}}{c^{1/2} a_i^{1/3}} \text{ for } i = 1, \dots, n$$

Proof. By using Lagrange multipliers. □

It will actually be more useful to write the solution as

$$x_i^2 = \frac{\sum_{j=1}^n a_j^{2/3}}{c a_i^{2/3}} \text{ for } i = 1, \dots, n \quad (4.11)$$

We now have

Theorem 2. *Given a circuit with significance σ_i computed for each adder A_i , the optimal distribution of a given energy budget E to minimize the average sum of worst case errors is given by*

$$E_i = E \frac{\sigma_i^{2/3}}{\sum_{j=1}^{N_A} \sigma_j^{2/3}} \quad (4.12)$$

where E_i is the energy devoted to A_i .

Proof. Let $Er(A_i, \mathbf{x})$ be the error produced at approximate adder A_i in the given circuit for input \mathbf{x} . From Eq. 4.2 and Theorem 1 the sum of errors at the outputs due to A_i is $\sigma(A_i)Er(A_i, \mathbf{x})$. Thus, the worst case error that we want to minimize is the average over all \mathbf{x} 's of $\sum_{i=1}^{N_A} |\sigma(A_i)Er(A_i, \mathbf{x})|$. Since we have a fixed energy budget E , $\sum_{i=1}^{N_A} Er^{-2}(A_i, \mathbf{x})$ is constant, see Eq. 4.10. Applying Eq. 4.11, we obtain Eq. 4.12. \square

It is to be noted that we only determine through Theorem 2 the energy of each adder in the graph. The method through which the supply voltages for the components inside a single adder are allocated is similar to the method in [12]. We pick a set of voltages and then using simulations of the single adder for different binning schemes (binning [12] is a technique in which each component is assigned a particular voltage bin) we pick the best.

4.4 Summary of the Method and Extension to Other Adders

Our method of minimizing energy usage across multiple adders in a circuit depends on the following two key elements.

1. Quantitative characterization of the relative significance of each adder in the circuit

We model the circuit as a DAG consisting of adders and shifters, and compute the significance of each adder based on the topology of the circuit as formalized in Eq. 4.9

2. The computation of the best distribution of the given energy budget, based on the relative importance of all the adders, which have been computed above. To do this, we rely on the relationship between the voltage supplied to an adder (and thus the energy devoted to it) and the average error produced by the adder

This computation is done for the case of ripple carry adder, for which the relationship between the energy and the error is known, and we obtain the energy to be devoted to each adder in Eq. 4.12

Our method to implement the first element is adder-independent. It does not matter which adder circuit is used, as the relative importance of an adder *depends only on the graph topology*. The second element, though, depends on the way error produced at the output of an adder is affected with respect to the energy invested on that adder, which we did for ripple carry adders.

For adders with different designs (such as a carry lookahead adder or a carry skip adder) the error-energy relationship might be different. In this case, the result that will be different is of Lemma 4.3.2 (which will influence Theorem 2) where the constraint varies based on the error-energy relationship. For example, if for some adder design the energy of the adder was proportional to the cube of the inverse of the error (instead of the inverse of a square in the case of dynamic energy usage of a ripple carry adder), then the constraint would be $\sum_{i=1}^n \frac{1}{x_i^3} - c$. In a general case, let us consider that the error-energy relationship is $E_A \cong f(Er(A_i))$, then the constraint would be $\sum_{i=1}^n f(x_i) - c$.

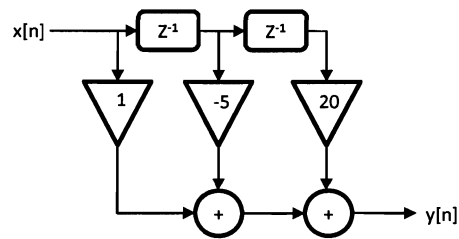


Figure 4.4 : A finite impulse response filter

4.5 Optimizing Designs of DSP Primitives

We apply the *single resource dataflow energy-error optimization* method of Theorem 2 to two specific cases, a *finite impulse response filter* and the *Fast Fourier Transform* which are ubiquitous in embedded signal processing. These signal processing elements are used in a variety of applications such as hearing aids or media players where the output of the devices is evaluated by human perception (which can tolerate errors).

4.5.1 Converting constant-number multipliers to adders and shifters

As discussed, every constant-number multiplier can be converted to a set of adders and shifters. For example, $x \times 5 = (x \ll 2) + x$ and $x \times 15 = (x \ll 4) - x$. Also, if we want both $x \times 21$ and $x \times 13$, then we can compute $x \times 5 = (x \ll 2) + x$ just once and use it to compute both $x \times 21 = (x \ll 4) + x \times 5$ and $x \times 13 = (x \ll 3) + x \times 5$.

To summarize, we try to use canonical signed digit (CSD) coding and sub-expression sharing techniques [59] to transform constant-number multiplications to additions and shifting most efficiently.

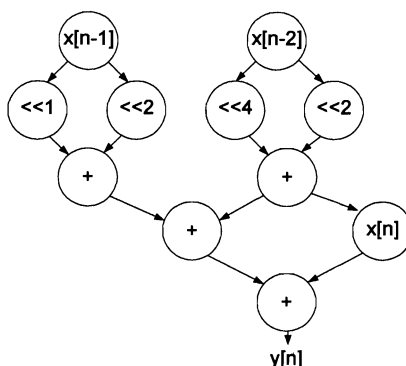


Figure 4.5 : Graph theoretical representation of a finite impulse response filter

4.5.2 Approximate finite impulse response filter

We will consider digital filters with a finite-duration impulse response (FIR). The output is

$$y[n] = \sum_{m=0}^{N-1} h[m] x[n - m].$$

where n and m are integers representing samples in time, x is the input sequence, y the output sequence and h is the impulse response of length N .

Any FIR can be represented in the graph-theoretic framework of Section 4.2. Consider the FIR shown in Fig. 4.4 which computes $y[n] = x[n] - 5x[n - 1] + 20x[n - 2]$. Fig. 4.5 shows it in the form of a DAG where we use a 16-bit 2's complement RCA for each adder.

According to Theorem 2, the optimal distribution of energy across adders depends on the significance of each adder. As there are no shifters and there is exactly one path from each adder to the output, $\sigma(A_i) = 1$ for $1 \leq i \leq N_A$. Thus, every adder in this circuit is “equally important” and distribution of energy investment equally across all adders produces the minimum error magnitude in the output. This holds for all FIRs.

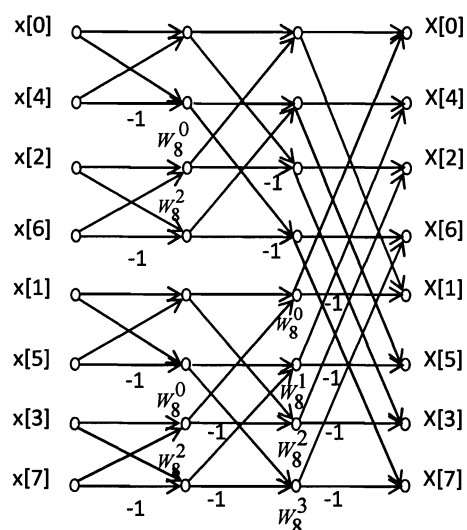


Figure 4.6 : Flow graph of a complete decimation-in-time decomposition of a 8-point FFT

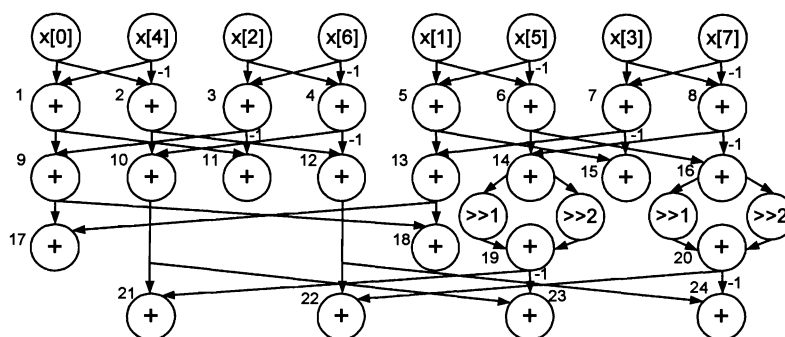


Figure 4.7 : Graph-theoretic representation of an 8-point FFT

4.5.3 Approximate fast Fourier transform

For a finite duration sequence of complex numbers, the *Discrete Fourier Transform* (DFT) is used to transform a sequence from its original representation (often in the time domain) to

the frequency domain representation. It is

$$X[k] = \begin{cases} \sum_{n=0}^{N-1} x[n]W_N^{kn} & \text{if } 0 \leq k \leq N - 1 \\ 0 & \text{otherwise.} \end{cases}$$

where x is the input sequence, X is the DFT and $W_N = e^{-\frac{2\pi i}{N}}$.

We use the *Fast Fourier Transform* (FFT) to compute the DFT. A flow graph of a complete decimation-in-time decomposition of an 8-point FFT computation is shown in Fig. 4.6, with its graph-theoretic description shown in Fig. 4.7.

To distribute the energy budget, we use Eq. 4.12, after computing σ_i 's for all adders A_i 's. For this, we use Eq. 4.9, where each A_i in turn plays the role of A_{Er} . From Fig. 4.7, $N_A = 24$. We do not show all the significance values for all the adders, but for example, $\sigma_1 = 2$ and $\sigma_6 = 3$. Given all σ_i 's, we apply Eq. 4.12.

This method is easily applicable to higher order FFTs and we applied it to a 16-point FFT, whose diagram we do not show, as it is quite large. We use the 16-point FFT design presented in [60], which is a hardware-efficient architecture based on the phase-amplitude splitting technique which converts a DFT to cyclic convolutions and additions. In the design, all the multipliers are converted into a set of shifters and adders.

We convert the original systolic implementation into a parallel implementation by replicating the circuit after removing memory elements and feedback loops. This results in a circuit consisting of 207 adders, which processes all 16 data words simultaneously. We used 16-bit 2's complement adders to design both the 8-point and 16-point FFTs. We show the results of applying the energy optimization technique on these circuits in the next section.

4.6 Simulation Framework and Results

In this section we present the framework for validation of this global optimization scheme. We will also present the simulation results which show the savings in energy consumption that can be obtained by applying this methodology.

4.6.1 Simulation framework

To validate our claims we use Synopsys HSPICE Version B-2008.09 and explore across different supply voltage configurations. The number of configurations is too large for all of them to be explored in HSPICE. So we developed a very fast, C++ based simulator framework for approximate circuits, which we use as first step in narrowing down the number of candidate configurations of interest. We feed this simulator the energy consumption and transition delay values of basic gates simulated in HSPICE. The simulator uses this data to simulate the behavior of approximate circuits. From this simulation we obtain the average error at the output and the average dynamic energy consumption for the circuit over a large set of input data.

Once candidate configurations are obtained by this simulator, we simulate the entire circuit with these configurations in HSPICE. All the simulations are performed in Synopsys 90 nm technology. The technology chosen has approximately 0.1% static leakage, so we have not yet needed to model static energy consumption. The range of voltages in which the circuit components are operated is 0.7 V to 1.2 V. To limit the overhead of supplying and transmitting these voltages, we picked only four specific ones for all our circuits, namely 0.7 V, 0.9 V, 1.0 V, and 1.2 V. We looked at all possible combinations of supply voltages considering 0.7 V, 0.8 V, 0.9 V, 1.0 V, 1.1 V and 1.2 V, but we picked these four specific voltages to present the results as they seemed to perform the best given that we wanted at most four distinct voltages (and hence four voltage domains/islands).

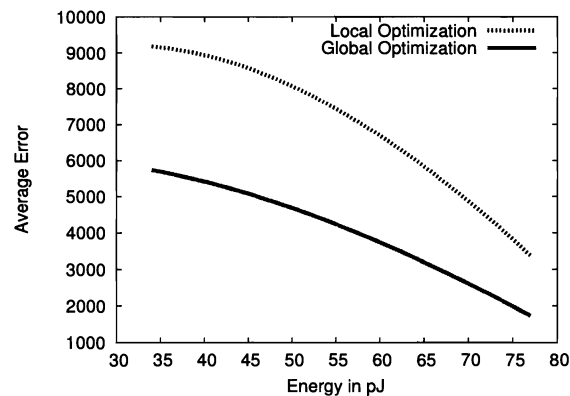


Figure 4.8 : Energy consumption vs. Average error for a 8-point FFT with local and global optimization

Although our custom simulator is only used to propose candidates to HSPICE for the 8-point FFT, we validated the simulator with respect to the energy consumption and average error to be within a margin of 12% by complete HSPICE simulations of an 8-point FFT for the four supply voltage levels.

These circuits are built using 16-bit 2's complement ripple carry adders. We picked the 16-bit ripple carry adder for the sake of simplicity of implementation and ease of understanding. The circuit that has been simulated is a combinational circuit with no pipelining and sampling only at the final outputs. Similar to the technique used in [12], we overclock the circuits so that the frequency of operation is higher than that permitted by the critical path of the circuit.

4.6.2 Results and comparisons

We start by briefly summarizing previous results, which are directly relevant to our work. Let E_{con} denote the the energy consumption of a conventional implementation of a circuit, in which the circuit is being operated at a frequency that is equal to what is permitted by the

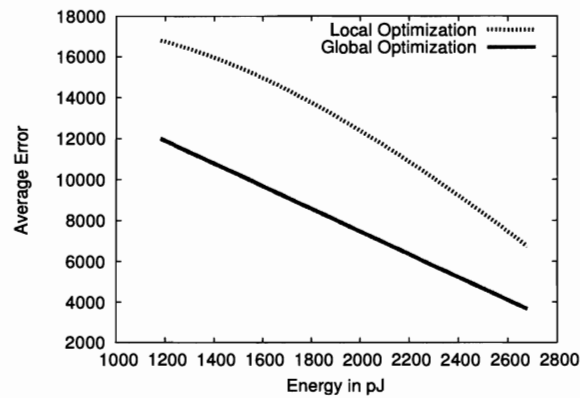


Figure 4.9 : Energy consumption vs. Average error for a 16-point FFT with local and global optimization

critical path delay, and hence has no *approximate* answers.

Now we consider circuits which are overclocked, that is, they are operated at frequencies higher than that permitted by conventional implementation (limited by critical path delay). Therefore, these circuits are approximate in their computation, as not enough energy (energy less than E_{con}) is supplied to them, so that they cannot run correctly at the higher frequencies.

Considering a circuit of an adder, the question studied in [12] was how to choose voltages so as to minimize the energy for some acceptable level of the average error. In the first method, all the full adders within the adder were given the same voltage level (uniformly voltage scaled), with the resulting required energy denoted by E_{UVOS} . In the second method, various full adders were given one of four voltage levels (biased voltage scaled), with resulting required energy denoted by E_{BIVOS} . It was shown that for the same level of overclocking and average error, $E_{\text{BIVOS}} < E_{\text{UVOS}}$. This shows that in an adder, some bit positions are more important than others. But this previous work only addresses *local optimization*. Using this approach, design of circuits with more than one n -bit adder implies using BIVOS in each adder without considering the relative significance of the various adders. We will refer to

this as *locally optimized*.

We propose to optimize the energy consumption of an approximate circuit with the optimization done using the *single resource dataflow energy-error optimization* method presented in this chapter, with the resulting energy denoted by E_{global} . Thus the relative significance of an entire adder with other adders in the circuit are taken into account to invest energy in an efficient way by finding the solution to Eqn. 4.1. But for supply voltages internal to a single adder we use the BIVOS scheme presented in [12]. Thus, supply voltages are assigned to minimize energy considering both entire n -bit adders and components inside a single adder. We will refer to this as *globally optimized*. The results we present show that globally optimized yields better energy savings for the same expected error than locally optimized.

We do not present results for the FIR because by using the global optimization scheme we conclude that all the adders are equally important and thus energy has to be distributed uniformly across all the adders. Hence the conventional locally optimized BIVOS scheme is the best we can do.

For the case of an 8-point FFT (shown in Fig. 4.7), we assume a given energy budget for the entire circuit and also find the critical path delay. For each energy budget that we pick, we obtain a point that has been used to interpolate the curve in Fig. 4.7. The 8-point FFT has a critical path delay of ≈ 10 ns when operated at 1.2 V but it is overclocked to a frequency of 0.2 GHz (essentially the inputs and outputs are provided with an interval of 5 ns) which is faster than permitted by a conventional design methodology.

Applying Theorem 2 (Eq. 4.12), we find an energy budget per adder from the total energy budget of the circuit. Based on this individual energy budget, we apply the previously published BIVOS scheme for each of the 24 adders using the four supply voltages (0.7 V, 0.9 V, 1.0 V, and 1.2 V) assumed. The result for the overall 8-point FFT in HSPICE with

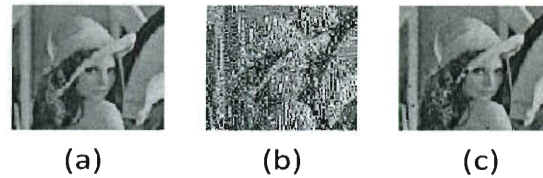


Figure 4.10 : Reconstructed images obtained after processing through a (a) conventional correct 8-point FFT (b) locally optimized approximate 8-point FFT (c) globally optimized approximate 8-point FFT

uniformly distributed random data as input is shown in Fig. 4.8. For the same energy investment of 77 pJ, the globally optimized 8-point FFT has 1.95X lower error than the “locally” optimized FFT operating at the same speed. Also, global optimization gives the designer at the least 1.44X lower energy investment for the same amount of quality trade off in the FFT. Overall, the globally optimized FFT has 2.8X lower *energy-delay product* (EDP), when compared to a conventional FFT for the same error.

A similar comparison using results from our custom simulator, which has been validated with HSPICE, is presented for a 16-point FFT in Fig. 4.9 to show that the analysis is scalable. In this case, we achieved 2.05X lower error for the same investment of 2700 pJ in both the globally optimized and only locally optimized FFT when they are overclocked to a frequency of 0.1 GHz (the critical path delay of the 16-point FFT at 1.2 V is ≈ 20 ns).

This phenomenon is also demonstrated in Fig. 4.10 which consists of reconstructed images after processing them through an approximate 8-point FFT and an inverse FFT. The three images in Fig. 4.10 are the original image, the image processed through FFT with only local optimization, and the image processed through FFT with global optimization, respectively from left to right. Fig. 4.10(c) has an SNR which is 1.42X times higher than the Fig. 4.10(b) for a similar energy of 62 pJ and operating frequency of 0.125 GHz. Also Fig. 4.10(c) has 1.7X lower energy consumption when compared to Fig. 4.10(a).

4.7 Impact on Circuit Design

The generality of our approach allows a circuit designer who is attempting to optimize the design of any circuit that can be represented as a graph of adders to automatically compute the *optimal* investment of energy. The design automation tool will invest energy such that the quality is computed as the “best”.

Moreover, the algorithm to compute the relative importance of the adders is extremely fast. So the approach scales very easily with the size and complexity of the circuit.

4.8 Principal Thesis

The primary thesis from this chapter is that local optimization of individual elements only can get so far in efficiency. To extract the full potential of any design methodology a global optimization is always necessary. Granted that global optimization with all variables is usually computationally prohibitive. That is one the reasons in designing efficient approximate circuits we split the optimization into two phases. The first phase is designing a good datapath element, actually a library of efficient datapath elements. The second phase is to use the elements in this library for a dataflow circuit.

Chapter 5

Remarks and Future Directions

In this thesis we have shown that current circuit design methodology would soon require drastic changes in terms of relaxing the requirement of absolute correctness. We have demonstrated one such opportunity for trading accuracy to lower energy consumption in arithmetic circuits and then extended to primitive digital signal processing circuits. We also presented a methodology through which multiple voltages can be distributed in an approximate arithmetic adder such that the loss of accuracy is minimum keeping under a given energy budget. We then present an approach to efficiently design an approximate network of adders which essentially models most popular digital signal processing primitives such as filters.

The research presented in this thesis can be extended in multiple areas. Some of our initial thoughts in those directions are as follows.

- **Library of Inexact circuits** : The current practice in circuit design framework (especially data-path synthesis) is to have a library of arithmetic operators from which a circuit designer/tool can choose a particular design of an operator (such as adder/multiplier or DSP primitives) based on the requirements and constraints posed. The optimization problems that have been framed in this thesis can be solved for various levels of energy budgets which would result in different solutions of supply voltage allocations. This would result in a library of approximate designs. Building such a library of approximate arithmetic operators or DSP primitives would be enormously useful for any future design purposes.

- **A minimal error energy-constrained resource-constrained concurrent scheduling and binding of inexact datapath circuits** : Currently the modeling and analysis of inexact datapath elements assumes a brute-force implementation of the entire algorithm. But for large algorithms, especially in digital signal processing, a brute-force implementation of a dedicated resource for each operation in the entire algorithm is futile. Some sort of architecture synthesis is required in such scenarios. The optimization framework described in this thesis for a datapath sequencing graph can provide a “priority” function dependent on the energy constraint. Then based on the resource constraints a concurrent scheduling and binding can be performed such that the error at the output of the datapath is minimal. This is assuming that there is a library of inexact arithmetic blocks with varying levels of energy-accuracy tradeoffs. This entire algorithm can be modeled as a combination of a conventional ILP formulation and the theoretical optimization presented in this thesis. This algorithm would be very useful in designing the datapath for very large circuits. The control block is assumed to be designed using correct logic so as to preserve the logical correctness of the entire algorithm. For example, a 1024-point FFT could be designed this way.
- **Dual Optimization Problem**: The optimization framework that we have shown for a single arithmetic adder considers one approach of framing the problem. There is a dual problem, mentioned earlier in Chapter 3, which is reducing the energy consumption while keeping the average error under an upper bound. The same framework could be used to frame the dual problem.
- **Extending the datapath network analysis to other arithmetic primitives**: The framework for optimization of a network of datapath elements currently assumes that all the elements compute linear functions (like adders). The motivation for such

an assumption is that many digital signal processing elements are constructed using a network of adders and shifters because the multiplications that are computed are constant-number multiplications. But there are many other applications which use normal multiplication and hence modeling the effect of multipliers in the dataflow graph is very important. And if possible optimizing the distribution of energy consumption across multipliers and adders together is very interesting.

- **Other energy-accuracy tradeoff methods:** This thesis only presents one possible approach for implementing a tradeoff between energy consumption and accuracy. There is a definite possibility that there are many different ways of doing the same which could be more efficient also. Based on other requirement and constraints a circuit designer might have a different implementation could be more ideal. Analysis and modeling of other tradeoff implementations is very interesting.

Appendix A

Effect of non-zero carry bits on error and energy models

In this section we discuss the effect of non-zero carry bits on the error and energy models presented in this paper.

Consider the indicator function given in Eq. 3.4 in Chapter 3.

$$I_k = \begin{cases} 1 & \text{if } d_{ik} > D \text{ and } \exists i, j \text{ such that } C_{ij} = 1 \text{ and } i < k < j \\ -1 & \text{if } d_{ik} > D \text{ and } \exists i, j \text{ such that } C_{ij} = 1 \text{ and } i < k = j \\ 0 & \text{otherwise.} \end{cases} \quad (\text{A.1})$$

Eq. A.1 is useful in modeling the error at the output of an adder because it denotes whether there is a possibility of error at the sum output of a given bit position provided the adder topology, worst-case propagation delays of the gates and the clock cycle time.

As per Theorem 1, this is based on the fact that when $C_{ij} = 1$, then unless $d_{ik} \leq D$

Bit position	5	4	3	2	1	0			
Intermediate carry bits at Time =0		0	0	1	0	0			
Input A		0	1	1	1	1	(15)		
Input B		0	1	0	0	1	(9)		
Time								$s^{(i)}$	
0		0	0	0	0	0	0	$s^{(0)} = 0$	24
1		0	0	0	0	1	0	$s^{(1)} = 6$	18
2		0	1	1	1	0	0	$s^{(2)} = 28$	-4
3		0	1	1	0	0	0	$s^{(3)} = 16$	8
4		0	1	1	0	0	0	$s^{(4)} = 24$	0

Figure A.1 : An example of a 5-bit binary addition using an RCA when the intermediate carry bits are assumed to non-zero at the beginning of the addition.

the sum output at bit position k is not computed correctly. If observed carefully, this is assuming that a carry cannot begin in the middle of a carry chain which could happen if the intermediate carry bits were not all zero. For example, if an intermediate carry bit at a particular bit position is 1 before the addition begins then a “rogue” carry chain could be generated. This is illustrated in Example 13.

Example 13. Consider the 5-bit addition of the two numbers 01001 and 01111. This is a duplication of Example 5 but instead of assuming that all the intermediate carry bits are 0, we will assume the intermediate carry bits as shown in Fig. A.1. As we can see from the figure, due to the fact that the intermediate carry at bit position 2 is 1 before the addition started there is a “rogue” carry chain that begins from bit position 2 which, in this case, produces the correct sum output at bit position 3 earlier (shown in Example 5) than it would have been computed without the rogue carry chain. \square

To model the effect of “rogue” carry chains using the procedure of indicator functions is very difficult. Since now we have to take into account the probability that an intermediate carry bit is 1 and a rogue carry chain is generated. For example, consider that $C_{ij} = 1$, then to evaluate whether there is a possibility of error at bit position k , where $i < k \leq j$, then we have to take into account all the cases where a rogue carry chain could have begun from the between bit position i and bit position k .

We also need to model the effect of multiple rogue carry chains. Our optimization of evaluating the average error at the output of the adder by computing an average over all possible carry chains is valid because of the fact that carry chains do not overlap (as shown in Observation 1. But as explained above, “rogue” carry chains can overlap and hence our error model would not hold.

Similarly our energy model is based on a similar indicator function which again would not be valid in the presence of these “rogue” carry chains.

Thus a major modification of error and energy modeling would be needed to take the effect of non-zero intermediate carry bits into account. Modeling this effect would probably result in a closer estimate to the reality but will significantly increase the complexity of the algorithm.

Bibliography

- [1] T. Forester, *The Information Technology Revolution*. The MIT press, 1985.
- [2] G. E. Moore, “Cramming more components onto integrated circuits,” *Electronics Magazine*, vol. 38, no. 8, 1965.
- [3] THE CLIMATE GROUP, “Smart2020: Enabling the low carbon economy in the information age,” 2008.
- [4] ITRS, “International technology roadmap for semiconductors update,” 2008.
- [5] K. Natori and N. Sano, “Scaling limit of digital circuits due to thermal noise,” *Journal of Applied Physics*, vol. 83, pp. 5019–5024, 1998.
- [6] N. Sano, “Increasing importance of electronic thermal noise in sub-0.1mm Si-MOSFETs,” *The IEICE Transactions on Electronics*, vol. E83-C, pp. 1203–1211, 2000.
- [7] L. B. Kish, “End of Moore’s law: Thermal (noise) death of integration in micro and nano electronics,” *Physics Letters A*, vol. 305, pp. 144–149, 2002.
- [8] J. von Neumann, “Probabilistic logics and synthesis of reliable organisms from unreliable components,” in *Automata Studies*, 1956.
- [9] K. V. Palem, “Proof as experiment: Probabilistic algorithms from a thermodynamic perspective,” in *Proceedings of the International Symposium on Verification (Theory and Practice)*, June 2003.

- [10] K. Palem, "Energy aware computing through probabilistic switching: a study of limits," *Computers, IEEE Transactions on*, vol. 54, pp. 1123 – 1137, sep. 2005.
- [11] J. George, B. Marr, B. E. S. Akgul, and K. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," in *Proceedings of the The IEEE/ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, pp. 158–168, 2006.
- [12] L. N. Chakrapani, K. K. Muntimadugu, A. Lingamneni, J. George, and K. V. Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: a mathematical foundation and preliminary experimental validation," in *CASES '08: Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems*, (New York, NY, USA), pp. 187–196, ACM, 2008.
- [13] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet, "Energy Parsimonious Circuit Design through Probabilistic Pruning," *Proceedings of the Design Automation and Test in Europe*, 2011.
- [14] P. Korkmaz, B. E. S. Akgul, L. N. Chakrapani, and K. V. Palem, "Advocating noise as an agent for ultra low-energy computing: Probabilistic CMOS devices and their characteristics," *Japanese Journal of Applied Physics*, vol. 45, pp. 3307–3316, Apr. 2006.
- [15] L. N. Chakrapani and K. V. Palem, "A probabilistic boolean logic for energy efficient circuit and system design," in *Proceedings of Asia South Pacific Design Automation Conference*, pp. 628–635, 2010.
- [16] J. M. Tour and D. K. James, "Molecular electronic computing architectures: A review,"

- in *Handbook of Nanoscience, Engineering and Technology, Second Edition* (I. Goddard, W. A., D. W. Brenner, S. E. Lyshevski, and G. J. Iafrate, eds.), pp. 5.1–5.28, New York: CRC Press, 2007.
- [17] H.-K. Hsieh, “Chinese tally mark,” *The American Statistician*, vol. 35, no. 3, p. 174, 1981.
- [18] G. Ifrah, *The Universal History of Numbers : From Prehistory to the Invention of the Computer*. Wiley, October 2000.
- [19] V. S. Steckline, “Zermelo, boltzmann, and the recurrence paradox,” *American Journal of Physics*, vol. 51, no. 10, pp. 894–897, 1983.
- [20] J. Simpson, *Simpson’s Contemporary Quotations*. Boston: Houghton Mifflin, 1988.
- [21] S. Borkar, “Designing reliable systems from unreliable components: The challenges of transistor variability and degradation,” *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.
- [22] B. Shekhar, “Exponential challenges, exponential rewards - the future of moore’s law,” in *Proceedings of the IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*, p. 2, 2003.
- [23] V. De and S. Borkar, “Low power and high performance design challenges in future technologies,” in *Proceedings of the 10th Great Lakes symposium on VLSI*, (New York, NY, USA), pp. 1–6, ACM, 2000.
- [24] S. Borkar, “Probabilistic & statistical design – the wave of the future,” in *VLSI-SoC: Research Trends in VLSI and Systems on Chip*, pp. 69–79, 2008.
- [25] N. Pippenger, “Reliable computation by formulas in the presence of noise,” *Information Theory, IEEE Transactions on*, vol. 34, pp. 194 –197, Mar. 1988.

- [26] T. Feder, "Reliable computation by networks in the presence of noise," *Information Theory, IEEE Transactions on*, vol. 35, no. 3, pp. 569–571, 2002.
- [27] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Proceedings of the International Conference on Computer Aided Design*, November 2002.
- [28] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 76–81, 1998.
- [29] J. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, pp. 436–443, Dec. 1997.
- [30] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy/power," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, pp. 270–276, Apr. 2003.
- [31] Y. Yeh, S. Kuo, and J. Jou, "Converter-free multiple-voltage scaling techniques for low-power CMOS digital design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, pp. 172–176, Jan. 2001.
- [32] Y. Yeh and S. Kuo, "An optimization-based low-power voltage scaling technique using multiple supply voltages," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 535–538, May 2001.
- [33] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, and T. M. Krisztian Flautner,

- “Self-tuning dvs processor using delay-error detection and correction,” in *IEEE Journal of Solid-State Circuits(JSSC)*, 2006.
- [34] D. Ernst, N. S. Kim, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, and K. Flautner, “Razor: Circuit-level correction of timing errors for low-power operation,” in *IEEE MICRO special issue on Top Picks From Microarchitecture Conferences of 2004*, 2005.
- [35] K. Bourzac, “Intel prototypes low-power circuits,” *Technology Review, Published by MIT*, 2010. <http://www.technologyreview.com/computing/24843/>.
- [36] J. Tschanz, K. Bowman, S. Lu, P. Aseron, M. Khellah, A. Raychowdhury, B. Geuskens, C. Tokunaga, C. Wilkerson, T. Karnik, and V. De, “A 45nm resilient and adaptive microprocessor core for dynamic variation tolerance,” in *Proc. of IEEE Intl. Solid-State Circuits Conf.*, pp. 282–283, 2010.
- [37] B. Shim, S. R. Sridhara, and N. R. Shanbhag, “Reliable low-power digital signal processing via reduced precision redundancy,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, pp. 497–510, May 2004.
- [38] R. Hegde and N. R. Shanbhag, “Soft digital signal processing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, pp. 813–823, Dec. 2001.
- [39] L. Wang and N. R. Shanbhag, “Low-power filtering via adaptive error-cancellation,” *IEEE Transactions on Signal Processing*, vol. 51, pp. 575–583, Feb. 2003.
- [40] K. V. Palem, “Energy aware algorithm design via probabilistic computing: from algorithms and models to Moore’s law and novel (semiconductor) devices,” in *Proceedings of the IEEE/ACM International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pp. 113–117, 2003.

- [41] K. V. Palem, "Energy aware computing through probabilistic switching: A study of limits," *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1123–1137, 2005.
- [42] L. N. Chakrapani and K. V. Palem, "A probabilistic boolean logic and its meaning," in *Rice University, Department of Computer Science Technical Report*, pp. TR08–05, 2008.
- [43] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power dct architecture," in *Proc. of the Conf. on Design, Automation and Test in Europe*, pp. 630–635, 2007.
- [44] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low power methodology manual: for system-on-chip design*. Springer Verlag, 2007.
- [45] L. N. B. Chakrapani, P. Korkmaz, B. E. S. Akgul, and K. V. Palem, "Probabilistic system-on-a-chip architectures," *ACM Transactions on Design Automation of Electronic Systems*, vol. 12, no. 3, pp. 1–28, 2007.
- [46] A. Hayes, "Apparent position governs contour-element binding by the visual system." *Proceedings of the Royal Society B: Biological Sciences*, vol. 267, no. 1450, p. 1341, 2000.
- [47] K. Palem, L. Chakrapani, Z. Kedem, A. Lingamneni, and K. Muntimadugu, "Sustaining Moore's law in embedded computing through probabilistic and approximate design: Retrospects and prospects," in *Proc. of the IEEE/ACM Intl. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, pp. 1–10, 2009.
- [48] Z. M. Kedem, V. J. Mooney, K. K. Muntimadugu, K. V. Palem, A. Devarasetty, and P. D. Parasuramuni, "Optimizing energy to minimize errors in dataflow graphs using approximate adders," in *CASES*, pp. 177–186, 2010.

- [49] T. Kamei, T. Yamada, T. Koike, M. Ito, T. Irita, K. Nitta, T. Hattori, and S. Yoshioka, "A 65nm dual-mode baseband and multimedia application processor soc with advanced power and memory management," in *Proceedings of the 2009 Asia and South Pacific Design Automation Conference, ASP-DAC '09*, (Piscataway, NJ, USA), pp. 535–539, IEEE Press, 2009.
- [50] AMD, "AMD Turion™ X2 Ultra and Turion™ X2 Key Architecture Features." <http://www.amd.com/uk/products/notebook/processors/turion-x2/Pages/turion-x2-mobile-features.aspx>, 2010.
- [51] J.-M. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, pp. 436–443, Dec. 1997.
- [52] B. Sahu and G. Rincon-Mora, "A low voltage, dynamic, noninverting, synchronous buck-boost converter for portable applications," *Power Electronics, IEEE Transactions on*, vol. 19, pp. 443 – 452, March 2004.
- [53] N. Pippenger, "Analysis of carry propagation in addition: An elementary approach," tech. rep., University of British Columbia, Vancouver, BC, Canada, Canada, 2001.
- [54] M. Ditzel, W. Serdijn, and R. Otten, *Power-aware architecting for data-dominated applications*. Springer Verlag, 2007.
- [55] A. Matsuzawa, "Low-voltage and low-power circuit design for mixed analog/digital systems in portable equipment," *Solid-State Circuits, IEEE Journal of*, vol. 29, no. 4, pp. 470–480, 2002.
- [56] S. Boyd, S. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and Engineering*, vol. 8, no. 1, pp. 67–127, 2007.

- [57] A. Mutapcic, K. Koh, S. Kim, and S. Boyd, “GGPLAB: A MATLAB toolbox for geometric programming,” 2006.
- [58] J. Cooley and J. Tukey, “An algorithm for the machine calculation of complex fourier series,” *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.
- [59] N. Boullis and A. Tisserand, “Some optimizations of hardware multiplication by constant matrices,” *IEEE Transactions on Computers*, vol. 54, pp. 1271–1282, 2005.
- [60] Y. Zhou, J. Noras, and S. J. Shepherd, “Novel design of multiplier-less FFT processors,” *Signal Process.*, vol. 87, no. 6, pp. 1402–1407, 2007.
- [61] A. Weinberger and J. L. Smith, “A logic for high-speed addition,” in *National Bureau of Standards Circular 591*, pp. 3–12, 1958.
- [62] I. Koren, *Computer arithmetic algorithms*. AK Peters, Ltd., 2002.
- [63] T. Cormen, C. Stein, R. Rivest, and C. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, third ed., 2009.