

***PROBABILISTIC* BOOLEAN LOGIC, ARITHMETIC
AND ARCHITECTURES**

A Thesis
Presented to
The Academic Faculty

by

Lakshmi Narasimhan Barath Chakrapani

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science, College of Computing

Georgia Institute of Technology
December 2008

***PROBABILISTIC* BOOLEAN LOGIC, ARITHMETIC AND ARCHITECTURES**

Approved by:

Professor Krishna V. Palem, Advisor
School of Computer Science, College
of Computing
Georgia Institute of Technology

Professor Sung Kyu Lim
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Gabriel H. Loh
College of Computing
Georgia Institute of Technology

Professor Trevor Mudge
Department of Electrical Engineering
and Computer Science
University of Michigan, Ann Arbor

Professor Sudhakar Yalamanchili
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Date Approved: 24 March 2008

To my parents

The source of my existence, inspiration and strength.

ACKNOWLEDGEMENTS

आचार्यात् पादमादत्ते पादं शिष्यः स्वमेधया।
पादं स्रब्रह्मचारिभ्यः पादं कालक्रमेण च॥

*“One fourth (of knowledge) from the teacher, one fourth from self study, one fourth from fellow students and one fourth in due time”*¹

Many people have played a profound role in the successful completion of this dissertation and I first apologize to those whose help I might have failed to acknowledge. I express my sincere gratitude for everything you have done for me.

I express my gratitude to Professor Krisha V. Palem, for his energy, support and guidance throughout the course of my graduate studies. Several key results pertaining to the semantic model and the properties of probabilistic Boolean logic were due to his brilliant insights. In this sense, the results pertaining to the semantic model and the properties of probabilistic Boolean logic, were developed in a truly collaborative manner. I express my sincere gratitude to Professor Zvi Kedem for his enthusiasm and insightful feedback on the logic aspects of this work. I thank my dissertation committee member, Professor Sudhakar Yalamanchili for his patience, support and guidance during my stay at Georgia Tech. My dissertation committee member, Professor Gabriel Loh provided me invaluable feedback and advice on my research and writing. Given their standing in the microelectronics community and their busy schedules, I consider myself very fortunate to have had the guidance and encouragement of Professor James Meindl and Dr. Ralph Cavin. I thank my dissertation committee members, Professor Trevor Mudge and Professor Sung Kyu Lim for

¹Ancient *subhashita*, usually attributed to Haradatta’s commentary on Apastamba Dharmasutra

their encouragement and suggestions. I wish to thank Professor Vincent Mooney for his interest and encouragement. I wish to express my sincere and profound gratitude to Dr. Ranjani Parthasarathi for stimulating my interest in computer science and encouraging me to pursue graduate studies.

I would like to acknowledge the funding support from Defense Advanced Research Projects Agency (DARPA). Dr. Robert Graybill supported our research, right from its infancy, and enabled us to pursue it through a DARPA seedling in 2002. I wish to thank Dr. Shekhar Borkar, Dr. Vivek De and Dr. Keith Bowman for their enthusiastic support and guidance and acknowledge Intel Corporation for the funding which supported in part, the research reported in this dissertation.

I benefited greatly from the stimulating environment at the Georgia Institute of Technology, the California Institute of Technology and Rice University. I thank everyone who granted me an opportunity to work at these institutions.

I thank my colleagues Dr. Pinar Korkmaz and Suresh Cheemalavagu for their friendship and collaboration. A large part of this dissertation is synergistic with and complements Pinar's ideas and foundations. I thank Dr. Bilge Akgul, Jason George and Bo Marr for their friendship and collaboration on many joint papers. I wish to thank my friends Dr. Rodric Rabbah, Yogesh Chobe, Tushar Kumar and Dr. Balasubramanian Seshasayee for the many stimulating conversations and eclectic ideas. I thank my friends Romain Cledat, Jaswanth Sreeram and Rick Copeland for their friendship. I thank my friends Karthik Sankaranarayanan, Easwaran Raman, Madhusudanan Seshadri and Niranjana Venkatraman for their friendship over the many years, their standard of academic excellence and their personal integrity. I wish to thank Kiruthika for her friendship, patience, interesting conversations and company.

Words cannot express my gratitude for the sacrifices, patience, trust and support of my mother Saroja, my father Chakrapani, my sister Manjula and my brother-in-law Srivats.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	ix
LIST OF FIGURES	x
SUMMARY	xii
I INTRODUCTION	1
1.1 Reading Guide and Roadmap	9
II HISTORICAL PERSPECTIVE AND BACKGROUND	11
2.1 Logic, Computing and Probabilistic Algorithms	11
2.2 Frequentist Interpretation of Probability and Probability in Logics	13
2.2.1 The Frequentist Interpretation of Probability	13
2.2.2 Probability in Logics	15
2.3 Thermodynamics and the Energy Cost of Computing	16
2.3.1 The Energy Cost of Probabilistic Computing	20
2.4 Current Technology Challenges	22
2.4.1 Probabilistic Complementary Metal Oxide Semiconductor Devices	23
2.4.2 Techniques for Energy Efficient and Error-free Computing .	25
2.4.3 The Trade off between Energy, Error and Quality of Solution	26
2.4.4 Theoretical Approaches to Computing in the Presence of Faults	27
2.4.5 Practical Approaches to Computing In the Presence of Faults	28
III A PROBABILISTIC BOOLEAN LOGIC AND ITS MEANING	30
3.1 Probabilistic Boolean Logic and Well-Formed Formulae	31
3.1.1 Boolean Logic Preliminaries	32
3.1.2 The Operational Meaning of Probabilistic Boolean Operators	33

3.1.3	Probabilistic Boolean Formulae and their Truth Tables . . .	34
3.2	The Event Set Semantics of Probabilistic Boolean Logic	35
3.2.1	A Frequentist View of Probabilistic Boolean Logic	36
3.2.2	An Interpretation of a Probabilistic Boolean Formula for a Fixed Assignment Through Event Sets	39
3.3	A Formal Model for Probabilistic Boolean Logic	44
IV	PROPERTIES OF PROBABILISTIC BOOLEAN LOGIC	49
4.1	Classical Identities That are Preserved	49
4.2	Identities that are not Preserved	50
4.2.1	Associativity	50
4.2.2	Distributivity	51
4.3	Degree of Non-Associativity	52
4.3.1	Balanced Binary and Linear Probabilistic Boolean Formula	53
4.3.2	An Upper bound on the Probability of Unsatisfiability of a Balanced Binary Probabilistic Boolean Formula	55
4.3.3	A Lower bound on the Probability of Unsatisfiability of a Linear Probabilistic Boolean Formula	58
4.3.4	The Degree of Non-associativity of Probabilistic Boolean Logic	60
V	PROBABILISTIC BOOLEAN LOGIC AND MODELS OF COMPUTING	62
5.0.5	Thermodynamic Separation of Implicitly and Explicitly Probabilistic Gates and The Circuit Model of Computation	62
5.0.6	Energy Considerations For Realizing Probabilistic and Randomized Boolean Operators	66
5.0.7	Extending to Computational Model with State	66
VI	PROBABILISTIC ARCHITECTURES	69
6.1	Probabilistic System on a Chip Architectures	70
6.2	Energy and Performance Metrics for Probabilistic System on a Chip Architectures	72
6.2.1	Performance and Energy Modeling of Probabilistic System on a Chip Architectures	73
6.3	A Co-design framework	74

6.3.1	A Brief Description of the Applications of Interest	75
6.3.2	Application Level Gains	77
6.3.3	An Analysis of Gains	78
6.3.4	Implementation Independent Characteristics Influencing Co-design	79
6.3.5	Implementation Dependent Characteristics Influencing Co-design	84
6.4	A Comparison of Implicitly Probabilistic and Explicitly Random Circuits in the System on a Chip Context	88
6.5	The Suite of Applications, Partitioning, Optimization and PSOC Implementation	90
6.6	Some Practical Considerations	95
6.6.1	Reducing Multiple Voltage Levels	95
6.6.2	Quality of Randomization	98
VII	PROBABILISTIC ARITHMETIC	100
7.1	Abstracting a Mathematical Model	101
7.1.1	A Mathematical Model for Deterministic Addition	102
7.1.2	A Mathematical Model for Probabilistic Addition	105
7.2	Cost and Magnitude of Error of Probabilistic Addition	106
7.2.1	Error Magnitude of Probabilistic Addition	107
7.3	Relative Magnitude of Error	110
7.4	Some Practical Considerations	115
7.4.1	Truncation in Probabilistic Arithmetic	117
7.5	Case Study of Probabilistic Arithmetic in Digital Signal Processing	119
VIII	REMARKS AND FUTURE DIRECTIONS	121
	REFERENCES	127

LIST OF TABLES

1	Identities of PBL	50
2	The algorithms of interest, applications based on these algorithms and the core probabilistic step for each algorithm	76
3	Maximum and minimum EPP gains of PCMOS over the baseline implementation where the implementation \mathcal{I} has a StrongARM SA-1100 host and a PCMOS based co-processor	78
4	Application level flux, maximum and minimum EPP gains of PCMOS over the baseline implementation where the implementation \mathcal{I} has a StrongARM SA-1100 host and a PCMOS based co-processor	79
5	The EPP gain of PCMOS over SA-1100 and over CMOS for the core probabilistic step	84
6	The probability parameters required by the application, and the composition tree for generating them using two voltage levels	98

LIST OF FIGURES

1	Following Palem [139], (a) deterministic one bit switching functions and (b) their probabilistic counterparts with probability parameter (probability of correctness) p	21
2	The models from [2] of (a) a PCMOS switch and (b) representation of digital values 0 and 1 and the probability of error for a PCMOS switch	24
3	(a) The relationship between energy per switching step and probability of correctness for a NAND gate and an inverter at $90nm$ technology, and (b) the same relationship at $65nm$ technology (from [101])	25
4	A Boolean truth table for the formula $((x \wedge y) \vee (x \wedge z)) \vee (y \wedge z)$.	33
5	A probabilistic Boolean truth table for the PBF $((x \wedge_1 y) \vee_1 (x \wedge_1 z)) \vee_1 (y \wedge_{3/4} z)$	35
6	(a) A frequentist interpretation of a sentence $(1 \vee_{\frac{3}{4}} 0) \stackrel{r}{=} 1$ in PBL through an infinite sequence of events and (b) a succinct representation of this sequence as an event set	36
7	(a) The event set for the valid sentence $(0 \vee_{\frac{3}{4}} 0) \stackrel{\frac{3}{4}}{=} 0$ and $(0 \vee_{\frac{3}{4}} 0) \stackrel{\frac{1}{4}}{=} 1$ (b) three valid sentences and their event sets for the three remaining assignments to $(x \vee_{\frac{3}{4}} y)$	40
8	(a) Event set $\mathbf{E}_{\mathcal{S}', I'}$ of $(1 \vee_{\frac{3}{4}} 0) \stackrel{r'}{=} 1$ (b) event set $\mathbf{E}_{\mathcal{S}'', I''}$ of $(1) \stackrel{r''}{=} 1$ (c) $\tilde{\mathbf{E}} = \mathbf{E}_{\mathcal{S}', I'} \times \mathbf{E}_{\mathcal{S}'', I''}$ (d) constructing the event set for $((1 \vee_{\frac{3}{4}} 0) \vee_{\frac{5}{6}} 1) \stackrel{r}{=} 1$ from $\tilde{\mathbf{E}}$	41
9	(a) A linear PBF over n variables in syntactic form (b) as a tree structure illustrating the linear form (c) a reassociation of the same PBF (d) its balanced binary representation in tree form	54
10	(a) A transition function encoded as a transition truth table (b) a probabilistic circuit which computes this transition truth table (c) an equivalent randomized Boolean circuit which computes the transition truth table	67
11	The canonical PSOC architecture	71
12	The conventional implementation alternatives for an application . . .	72
13	The energy and performance modeling methodology for each component of a PSOC	75
14	Gain and flux for Bayesian network of various sizes	80

15	Variation of gain with respect to flux for Bayesian network	82
16	Variation of gain with respect to flux for randomized neural network .	83
17	For a fixed flux, variation of gain with respect to energy saved per unit flux and time saved per unit flux by using PCMOs	85
18	The co-processor architecture for a PSOC implementing Bayesian In- ference	91
19	The Custom ASIC host and PCMOs co-processor architecture for a PSOC implementing a probabilistic cellular automata algorithm	94
20	(a) Correct sum of A and B and incorrect sum when the carry at position 2 is computed incorrectly (b) correct sum of A and B and incorrect sum when the carry at position 3 is computed incorrectly (c) correct sum of A and B and incorrect sum when the carry at position 1 is computed incorrectly	102
21	The coefficients of $C = A \odot B$	104
22	Application level impact of probabilistic arithmetic on SAR (a) conven- tional error free operation, (b) uniform voltage scaling yielding 2.5x energy savings (c) BIVOS based probabilistic arithmetic yielding an ac- ceptable image and 5.6x energy savings (from [66])	119

SUMMARY

“Probabilistic behavior of computing elements can be tolerated as well as harnessed for low-energy and high-performance computation”

In this dissertation, we introduce a logic and arithmetic combined with probabilistic behaviors. First, we define models of computation rooted in the resulting *Probabilistic Boolean Logic* (PBL), and demonstrate a system-on-a-chip architecture based on these models. Next, we extend PBL to arithmetic and study the properties of the resulting arithmetic. In both cases (PBL gates as well as probabilistic arithmetic), the introduction of probabilistic behavior yields significant gains in the in the physical domain. These gains are in the energy consumed and the overall performance (speed) of computing. These developments collectively offer theoretical and empirical proof to support the thesis.

Parameter variations, noise susceptibility, and increasing energy dissipation of complementary metal oxide semiconductor (CMOS) devices (transistors) have been recognized as major challenges in circuit and architecture design in the nanometer regime. Among these, parametric variations and noise susceptibility increasingly cause CMOS devices to behave in an unreliable or “probabilistic” manner. This is true for novel non-CMOS materials as well, whose properties and manufacturing difficulties cause logic elements to behave in a probabilistic manner. To address these challenges, a shift in the design paradigm from current-day deterministic designs to statistical or probabilistic designs is deemed inevitable.

In this context, it should be noted that advances in Boolean logic, an understanding of its properties, and algorithms based on such properties have played a vital role

in the design and synthesis of digital circuits. If an analogous approach were to be adopted to theoretically characterize probabilistic logic elements, considerations of probability need to be injected into Boolean logic.

Motivated by these facts and considerations, a *Probabilistic Boolean Logic*, whose logical operators are by definition “correct” with a probability $\frac{1}{2} \leq p \leq 1$ is introduced. To characterize the meaning of a probabilistic Boolean formula (PBF) in this logic, we introduce and study the concept of *event sets*. Event sets serve as a basis for developing the laws of probabilistic Boolean logic. While most of the laws of Boolean logic can be naturally extended and shown to be valid in the case of probabilistic Boolean logic, there are some surprising differences. Based on probabilistic Boolean logic, we study two models of computation: the probabilistic Boolean circuit, and the probabilistic automaton whose transition function is computed by such a circuit.

To empirically demonstrate the utility and advantages of probabilistic Boolean circuits, we introduce and study a novel family of probabilistic architectures: the *probabilistic system-on-a-chip* (PSOC) architecture. These are based on CMOS devices rendered probabilistic due to noise, which are referred to as *probabilistic* CMOS or PCMOS devices. In addition to harnessing the probabilistic behavior of PCMOS devices, PSOC architectures yield significant improvements, in terms of energy as well as performance, in the context of probabilistic applications with broad utility. All of the application and architectural savings are quantified using the product of the energy and the performance denoted (energy \times performance): the PCMOS-based gains are as high as a substantial multiplicative factor of over 560 when compared to a competing energy-efficient realization.

Finally, we extend the consideration of probability of correctness from logic to arithmetic through *Probabilistic Arithmetic*, where the magnitude of correctness of an arithmetic operation may be traded for its energy; we can show that a relatively small amount of error in the arithmetic operators can be traded for significant energy

savings. This work provides the theoretical basis for the energy savings reported in the video decoding and radar processing applications, performed using digital filters realized using probabilistic arithmetic operations, that has been demonstrated by George et. al. [66].

CHAPTER I

INTRODUCTION

“Probabilistic behavior of computing elements can be tolerated as well as harnessed for low-energy and high-performance computation”

Automated computing, ranging from abstract machine models such as Turing machines [187] to practical programming languages, has its roots in the study and advances in logic¹. For example, two-valued Boolean logic is at the heart of the specification, automated construction and verification of silicon-based digital very large scale integrated (VLSI) circuits, which are the bedrock of the information technology revolution. The advances in logic and models of computation based on such logics, have had a significant impact on the design and construction of computing devices due to a correspondence between logical constructs and physical primitives that these computing devices are composed of. For example, there is a correspondence between Boolean operators and physically implemented logic gates in VLSI and by extension, between Boolean circuits and physically implemented circuits in VLSI. So far, the physical primitives such as transistors and logic gates used in building computing devices were (or for all practical purposes treated to be) deterministic at the macroscopic level and correspondingly, the logical constructs used to study them have been deterministic as well.

¹The reader is referred to Davis [43] for an excellent overview and a historical perspective, which relates advances in logic to the birth of modern computers and computer science in its present form.

However, the miniaturization of computing devices through technology scaling, popularly anticipated by Moore’s law [123], has challenged this assumption of determinism. Phenomena such as noise, parametric variations and other device perturbations [130, 169, 94] are increasingly introducing “statistical” or “probabilistic” behavior into transistors and logic gates. The current methodology of addressing these challenges by designing VLSI systems through conventional techniques rooted in deterministic logics and deterministic models of computation, and addressing probabilistic behavior in the realm of VLSI mainly in the form of rigorous test methodologies, are unlikely to be adequate for future technology generations [15, 16]. This is true for novel non-complementary metal oxide semiconductor (non-CMOS) materials such as molecular electronics [186] as well, where the material properties, perturbations and manufacturing difficulties cause physical primitives to behave in a probabilistic manner [79].

To accommodate this probabilistic behavior, it has been speculated that a shift in the design paradigm—from the current-day deterministic designs to statistical or probabilistic designs—would be necessary [82, 14]. For example, the international technology road-map for semiconductors (ITRS) forecasts [82] *“Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification, and test. Such a paradigm shift is likely forced in any case by technology scaling, which leads to more transient and permanent failures of signals, logic values, devices, and interconnects.”* This probabilistic behavior of building blocks in future technology generations, and its likely impact on computing devices has been recognized by leaders in industry: To cite an example, Borkar notes [14] *“We will shift from the deterministic designs of today to probabilistic and statistical designs of the future... So we now say, ‘If I do this in the design, the transistors and therefore the chip will perform in this way.’ In the future, we will say, ‘If I design with this logic depth or this transistor size, I will increase the probability that*

a given chip will perform in this way.’ ”

This prescription of “relaxing the requirement of 100% correctness” and “probabilistic and statistical designs” has several profound implications and challenges. The first challenge is rooted in the fracture of the correspondence between the probabilistic physical primitives and the deterministic logical constructs used to study and design computing devices based on these primitives. To remedy this fracture, a logic which incorporates probability, and whose constructs maintain a correspondence with physical primitives, needs to be developed and studied.

The notion of probability in logics and models of computation based on such logics is not new. While initial developments in logic and models of computation based on logic were deterministic, the notion of probability, when coupled with models of computation derived from logic, have proved to be very effective in realizing highly efficient algorithms for computing [157, 172]. Historically, probabilistic behavior was realized by adding an *external* source of randomness to conventional logic-based constructs, such as gates and automata, to *induce* randomness and hence probabilistic behavior [125]. To achieve this, pseudo-random bits are coupled with deterministic mechanisms. We refer to this as an *explicit* style of realizing probabilistic computing. Such an explicit style of realizing probabilistic computing has proved useful in the context of the design and implementation of efficient algorithms on deterministic computing devices. By contrast, to maintain a tight correspondence between probabilistic physical primitives and logic constructs, an *implicit* approach to realizing probabilistic computing needs to be introduced and studied. Characterizing this implicit approach to probabilistic computing and logic formally, and distinguishing it from its explicit counterpart, serve as the overarching philosophical themes of the first part of this work.

To this end, we introduce a novel *Probabilistic Boolean Logic* (PBL) as well as a model of computation—essentially a probabilistic automata (PA) in the Rabin

sense [156]—whose transition functions are realized through PBL. In PBL, the canonical operations—disjunction, conjunction, and negation, \vee_p, \wedge_q, \neg_r —have an associated probability p, q, r ($\frac{1}{2} \leq p, q, r \leq 1$) of being “correct”, and can be used to construct *probabilistic Boolean formulae* (PBF). Akin to formulae in classical Boolean logic, those in PBL can be constructed as compositions of probabilistic operators, variables, and the constants $\{0, 1\}$. Informally, for any input assignment to the deterministic variables in a probabilistic Boolean formula, its value is the outcome of a random experiment, whose *sample space* (for examples, see Feller [57]) is determined by the input assignment to the variables in the formula, its structure, *as well as* the associated probabilities of correctness of its constituent operators.

To formally characterize and “interpret” this informal notion of correctness of a PBF, we introduce the foundational concept of an *event set*: It consists of a set of events from a sample space, each of which is associated with a conventional deterministic Boolean formula. Given an input assignment I to a PBF, its event set can be used to characterize the possible set of events associated with this input assignment. This characterization helps us to unambiguously determine the correctness and truth of the PBF in a unified way. Thus, we note that in PBL, the assignment I is deterministic, and the probabilistic behavior is induced *entirely* by the implicitly probabilistic operators of the PBF.

This has to be contrasted with an approach to *explicitly* injecting probabilistic behavior into conventional Boolean formulae with deterministic operators, by considering some of the elements of the input assignment to be random variables (ranging over the set $\{0, 1\}$). Based on the event set semantics, we will distinguish the implicit and explicit approaches of melding logic with probability. Furthermore, we define the conditions under which two or more probabilistic Boolean formulae can be characterized as being equivalent using event sets. This formal notion of equivalence through event sets is used to characterize the significant identities or properties of PBL.

The properties of PBL for the most part correspond to those of classical Boolean logic. However, intriguingly, PBL does not preserve distributivity and associativity. In the latter context, a novel contribution of our work is to help quantify the “amount” by which a formula is non-associative. When we consider reassociations of the same formula, the probability with which it is satisfied varies. We use this variation as a basis for quantifying the *degree of non-associativity* of PBL. Specifically, we show that there exist formulae of length $n \rightarrow \infty$ such that the degree of non-associativity grows as $\Omega(n)$ where the probability of correctness of individual operations, $p = 1 - 1/n^c$. Conversely, the degree of non-associativity demonstrates how the probability of correctness of a given PBF F may be improved through considering reassociations of F , without compromising cost along other dimensions such as the size of the associated circuit.

To relate PBF to computing structures which can be implemented using probabilistic physical primitives, we introduce and study *Probabilistic Boolean Circuits*, a model of computing based on PBL, and characterize its relationship to conventional explicitly probabilistic circuit constructs from computer science that have randomness injected into them as “coin tosses”². It might seem natural to view these implicit and explicit formulations as being equivalent and consequently, the probabilistic Boolean circuit model based on PBL and the classical randomized circuit model as being interchangeable. While PBL and the associated constructs in the implicit context might be closely related to randomized circuits employing explicit randomness in terms of conventional complexity measures such as size or depth, we will infer that *the implicit*

²In this work, we distinguish between the terms *probabilistic* and *randomized* and hence the corresponding Boolean circuits. We use the terminology “probabilistic Boolean circuits” to refer to Boolean circuits whose gates correspond to one of the three probabilistic operators of PBL and hence are implicitly probabilistic. On the other hand, we use the terminology “randomized Boolean circuits” to refer to conventional Boolean circuits, some of whose inputs may be random variables and hence have probabilistic behavior explicitly injected into them.

variety is more efficient or less expensive, through the measure of energy consumption. Thus, physical energy consumption provides a novel approach to distinguishing explicit and implicit approaches beyond semantic and structural differences.

This characterization of the difference between physically realized implicitly probabilistic and explicitly random constructs based on energy considerations, builds on prior foundational work [138, 139] and work done in the context of CMOS devices rendered probabilistic by thermal noise, referred to as probabilistic CMOS or PC-MOS [35, 101, 102]. Finally, moving beyond circuit-based models and considering computational models with a notion of *state* in the form of a PA, we show that these gains, or energy advantages, persist. To demonstrate this, we consider the transition function of a PA and show that any transition function of such an automaton realized as an implicitly probabilistic circuit, consumes less energy than an equivalent explicitly realized circuit.

The second challenge is the construction of computing architectures using probabilistic physical primitives, based on the principles and properties of PBL. This includes defining a design and empirically demonstrating the usefulness as well as the advantages of such a design. In this context, the massive investments in legacy designs, design methodologies and tools dictate that such architectures, at least initially, cannot depart radically from current-day conventional architectures.

To respond to this critical challenge, we introduce and study a novel family of probabilistic architectures which we refer to as the *probabilistic system-on-a-chip* (or PSOC) architecture. In our current context, PSOCs are based on PCMOS devices. Our PSOC architecture, where an energy efficient (deterministic) general purpose processor like the StrongARM processor [174] is coupled to an application-specific probabilistic co-processor, resembles the ubiquitous host and co-processor (or accelerator) style of system-on-a-chip architectures [59].

We demonstrate that PSOC architectures yield significant improvements, both in the energy consumed *as well as* in the performance, in the context of probabilistic applications with broad utility. All of our application and architectural savings are quantified using the product of the energy and the performance denoted (energy \times performance): the PCMOs-based gains are as high as a substantial multiplicative factor of over 560 when compared to a competing energy-efficient realization. Since design considerations for PSOC architecture differ from those of conventional system-on-a-chip (SOC) architectures, we introduce and employ a novel algorithm-architecture-technology (A²T) co-design methodology to design efficient PSOC implementations. Our architectural design is application-specific and involves navigating the design space spanning the algorithm (the application), its architecture (the PSOC) and the probabilistic technology (PCMOs).

A third challenge in considering the role of probabilistic behavior in computing and architectures, is to reason about constructs based on logic, at a higher level of abstraction than logical operations. Specifically, in the deterministic context, arithmetic primitives, notably addition and multiplication have been widely studied and employed as architectural building blocks of algorithms [95]. Though all arithmetic operations are composition of logical operations, it is conceptually more tractable to reason about algorithms and computing structures directly at the granularity of arithmetic primitives. In the probabilistic context, we introduce a novel notion of *probabilistic arithmetic* and provide a rigorous framework to analyze the relationship between the (energy) cost and the *probability of correctness* for probabilistic addition.

Probabilistic addition operation of two n bit numbers is realized using n *probabilistic addition primitives* (or primitives for short), where each primitive is correct with a probability $\frac{1}{2} \leq p_i \leq 1$. In addition, a cost model which relates the probability of correctness of a primitive to its cost, is specified. The cost of probabilistic addition of two n bit numbers is the sum of the costs of its n constituent primitives. The cost

of a primitive (and hence its probability of correctness) may be different from that of another primitive within the same n bit probabilistic addition operation.

In this context, we mathematically characterize the trade-off between (energy) cost and magnitude of error of probabilistic arithmetic. Since the costs of individual probabilistic primitives may differ, we study the impact of variable *investment* in the individual primitives on the magnitude of correctness of probabilistic addition. We show that for an n bit ripple carry adder, if the energy is invested equally across all the primitives—irrespective of the value or the significance of the bit they compute—the expected magnitude of error of addition grows as $\Omega(\sqrt{2^n})$. Furthermore, we prove that in the *exponentially biased* (unequal) investment case—where primitives which compute bits of a higher significance have a higher probability of correctness—the expected magnitude of error grows as $O(n^3)$.

We continue by studying various concerns which relate to the practical implementation of probabilistic arithmetic, since the cost of the design can also grow with the number of distinct energy values being invested: (i) The effect of *binning*—where the number of different energy values being considered for investment does not grow as n but is limited ³ to some value $b < n$ —on the energy savings, (ii) the effect of *truncation*; where no energy is invested on any of the primitives which compute t least significant bits.

As an empirical evidence of the utility of probabilistic arithmetic, we revisit the work of George et al. [66], where probabilistic arithmetic operations realized through PCMOs, have been used to implement the fast Fourier transform (FFT) and hence a synthetic aperture radar (SAR) processing application [167]. The probabilistic arithmetic operations implemented in this work include adders and multipliers, whose constituent probabilistic primitives—the full adders—have varying probabilities of

³For example, b can be $\log(n)$ or a constant c , a natural number.

correctness. In particular, the full adders which compute bits of a higher significance have a higher probability of correctness and hence higher energy investment in the form of higher supply voltages. This scheme of non-uniform operating voltages (which corresponds to the exponential investment scheme in probabilistic arithmetic), is termed as the *biased* voltage scheme or BIVOS. George et. al. demonstrate an energy savings of over a factor of 5.6 in the context of the SAR application for a visually imperceptible degradation in the quality of solution [66].

1.1 *Reading Guide and Roadmap*

This dissertation is organized as follows: In Chapter 2, we provide a historical perspective and summarize prior work by others, as relevant background to the theoretical as well as the empirical aspects of the dissertation. In Section 2.1, we sketch the influence of logic on computing and remark on the role of probability in algorithms and models of computation. In Section 2.2.1, we outline the frequentist interpretation of probability which will serve as a background for defining the “meaning” of a PBF in probabilistic Boolean logic. In Section 2.3, we sketch a brief history of thermodynamics and explain the role of probability in thermodynamics. The statistical explanation for thermodynamic entropy led to Maxwell’s thought experiment, and thus provided an explanation, rooted in thermodynamics, for the energy cost of computing [104]. This was extended by Paley to quantify the energy cost of probabilistic computing and a brief summary is provided in Section 2.3.1. The physical manifestation of the energy cost of probabilistic computing can be studied in the context of probabilistic CMOS devices and as background, we summarize the main results in Section 2.4.1. We shall use these results to quantify the energy efficiency of probabilistic Boolean circuits as well as to implement PSOC architectures.

We summarize the background work related to the empirical parts of the dissertation in Section 2.4 under three categories: (i) Techniques for energy efficient

computing and techniques for the trade-off between energy and quality of solution in Section 2.4.2, (ii) the possible use of “statistical” computing elements and theoretical approaches to computing in the presence of faults in Section 2.4.4 and (iii) practical approaches to computing in the presence of faults in Section 2.4.5. The material in Chapter 2 is of a general interest to a broad audience.

In Chapters 3, 4 and 5 we define probabilistic Boolean logic, provide a formal model for this logic, study its properties and define a model of computation based on PBL. In Chapter 3, we introduce PBL, define the operational meaning of a PBF and define the equivalence of two probabilistic Boolean formulae. In Section 3.3, we define the formal model for PBL and this is primarily of interest to logicians and readers interested in mathematical logic. Based on the notion of equivalence of two PBF, in Chapter 4, we study some interesting properties of PBL. In Chapter 5 we describe the models of computation based on PBL. For readers interested only in the use of circuits based on PBL, Sections 3.1 and 5.0.5.1 are of interest and the rest of Chapters 3, 4 and 5 may be skipped.

In Chapter 6, we describe an architecture for implementing the models of computation rooted in PBL and show empirical results. This chapter is of interest to computer architects and circuit designers.

We define probabilistic arithmetic in Chapter 7 and report on its utility in Section 7.5. We expect this work to be of interest to computer scientists and electrical engineers considering the impact of nano-scale devices on future computing systems and their principles. Finally we remark and conclude in Chapter 8.

CHAPTER II

HISTORICAL PERSPECTIVE AND BACKGROUND

Our work is based on results from logic, probability and energy cost of computing, and has connections to three distinct areas: *mathematical logic*, *computer science*, and applications to *electrical engineering*. The developments in thermodynamics, probability theory, logic and computing are intimately related, with advances in each field spurring advances and developments in the others, with results derived in one field, frequently yielding deep insights into the others. In Section 2.1 and Section 2.3, we sketch relevant aspects of these relationships: The development of logic and probability and its influence on computing, and the developments in thermodynamics and the corresponding understanding of the energy cost of computing. Building on Section 2.3, in Section 2.3.1 we briefly summarize the results which quantify the energy cost of probabilistic computing in a theoretical context. We will frequently refer to this section in the subsequent chapters of this dissertation. Section 2.4 has an emphasis on electrical engineering, and provides a background for the empirical aspects of this dissertation.

2.1 Logic, Computing and Probabilistic Algorithms

Right from the birth of modern logic, its development was directed towards defining and characterizing automated computation. Gottfried Leibniz is regarded as the father of modern logic, though all of his work on logic was published posthumously [62]. Leibniz’s motivation was to perfectly represent “*the relationships between our thoughts*” (see [145]pp-105) in a symbolic framework. Leibniz further postulated that such a symbolic representation was necessary to produce the *calculus ratiocinator*, an algebra through which mechanical calculations could be performed

to automate the process of logical deduction. Leibniz praised the development of such a logic thus: *“How much better will it be to bring under mathematical laws human reasoning, which is the most excellent and useful thing to have”* (see [43]). However, Leibniz’s attempts at defining such a logic would prove futile and Leibniz himself noted *“After so many logics the logic I dream of has not yet been written”* (see [62]pp-5).

Boole successfully derived an algebra that could represent logical operations and principles [12]. For example, if x, y represent two sets of objects, xy would represent objects in both sets. Boole showed that if 0 were taken to denote the empty set and 1, the universal set, then $x(1 - x) = 0$, which perfectly represents the principle of contradiction: the set of statements which are true and untrue at the same time is empty [12]. Through the development of Boolean algebra, Boole demonstrated that logical deduction could be formalized as a system of algebra. Shannon demonstrated that Boolean logic operators could be implemented through electrical relays (electrical relays could implement the conjunction, disjunction and negation operators) and hence an electrical circuit could perform logical and numerical calculations [177]. Thus, the development of logic resulted in the development of computing.

The initial developments in the models of computing such as Turing machines [187] or the circuit model of computing were deterministic. Curiously and counter-intuitively, the notion of probability, when coupled with models of computing derived from logic, have proved to be very effective in realizing highly efficient algorithms for computing. Rabin’s introduction of probabilistic automata [156] and randomized algorithms are pioneering examples which introduced considerations of probability in models of computing. Their impact was eloquently anticipated by Schwartz [172]—*“The startling success of the Rabin-Solovay-Strassen algorithm (see Rabin [157]), together with the intriguing foundational possibility that axioms of randomness may constitute a useful fundamental source of mathematical truth independent of, but supplementary to, the*

standard axiomatic structure of mathematics (see Chaitin and Schwartz [23]) suggests that probabilistic algorithms ought to be sought vigorously.” These contributions have led to vast areas of study that explore the power that probability and randomness add to computing. Correspondingly, for philosophical and ontological reasons, probabilities have been associated with logics in the past, with *probable inference* as one of the main motivators [42].

2.2 *Frequentist Interpretation of Probability and Probability in Logics*

As a background to probable inference, we first consider the rule of inference in propositional logic. In propositional logic, if P and Q are sentences and if P implies Q (denoted $P \rightarrow Q$) then by the rule of Modus ponens [121] $((P \rightarrow Q), P)$ logically entails Q . Informally, by the rule of Modus ponens, the fact P implies Q and the fact P is true can be used to deduce that Q is true. Certain real-world situations merit the question, *If P is not known to be true with certainty, is Q true?* [41] For example, in several artificial intelligence applications, rules of inference and data are not known with certainty and only strongly indicated by evidence. With this as motivation, several researchers (see Cox [42], Nilsson [135], Fagin and Halpern [55], Fagin, Halpern and Megiddo [56], for example) have generalized logic to deal with uncertainties. In a dual sense, the relevance of the theory of probability to the theory of probabilistic inference, has had an influence on the interpretation of probability itself. In this section, we briefly summarize the frequentist interpretation of probability and logics which have incorporated probability.

2.2.1 The Frequentist Interpretation of Probability

The concept of probability has had differing *interpretations*, where the two important interpretations have been the *frequentist* approach, championed by Venn [190], von Mises [191], Reichenbach [165], and others, and the *Bayesian* interpretation, of which

de Finetti [45], Ramsey [161], Jaynes [85] and others are prominent proponents (for a detailed discussion, please see Cox [41] and Bergmann [9]). The word “frequentist” is used to refer to the proponents as well as to the *frequency theoretic* interpretation of probability, and is attributed to Kendall [91]. Efron [52] outlines the controversies between the Bayesian interpretation and the frequentist interpretation). The notion of probability under the frequentist interpretation, is outlined elegantly by von Mises [191] *“It is possible to speak about probabilities only in reference to a properly defined collective”* and Bergmann [9] *“Probability theory deals with mass phenomena and repetitive events”*. Whereas, the interpretation of probability according to the Bayesian approach, quoting Cox is *“A relation between a hypothesis and a conclusion, corresponding to the degree of rational belief and limited by the extreme relations of certainty and impossibility”* (see Cox [41] pp-4) .

The frequentist approach, broadly speaking, defines the probability of an event A in a sequence of trials as simply the ratio of the number of occurrences of A to the total number of trials, as the number of trials tends to infinity. For example, the probability of occurrence of heads in any toss of a coin would simply be the ratio of the number of occurrences of heads to the total number of trials in an infinite sequence of trials. This interpretation, while satisfying the requirement of ascertainability—in principle, probabilities can be assigned to each event—introduces paradoxes. One of the paradoxes is the paradox of finite sequences. Considering the extreme case of one trial of a fair coin, only a relative frequency of either 0 or 1 for the probability of occurrence of heads can be obtained. The possible number of distinct relative frequencies increase with the number of trials and is limited by the number of trials. As a result, frequentists define ascertainable probability ratios only on infinite sequences. This interpretation, in turn, introduces paradoxes. In particular, re-ordering countably infinite sequences could give rise to different relative frequencies. As an example, a countably infinite sequence of equal number of heads and tails can be re-ordered

to have heads as the outcome of every tenth trial—the rest of the trials being tails—thereby attributing a relative frequency of $\frac{1}{10}$ (instead of $\frac{1}{2}$) to heads.

von Mises addresses these concerns through the axiom of convergence and the axiom of randomness. In particular, the axiom of convergence states that the *limiting* relative frequency of any event exists in a sequence of infinite trials. The axiom of randomness states that this limiting relative frequency of any event in an infinite sequence and the limiting relative frequency in any infinite sub-sequence are the same, thereby attributing some property of uniform “randomness” to the infinite sequence under consideration. This notion of “similarity” of an infinite sequence to any infinite sub sequence was formalized by Church [38] and ultimately refined by Kolmogorov [100] and Chaitin [24].

Our motivation in choosing the frequentist approach is based on the fact that we wish to apply methods based on our interpretation of PBL, to derive techniques not only for designing and synthesizing integrated circuits, but also for verifying them. Here, measurement to ascertain the behavior of probabilistic Boolean circuits is crucial. Ascertaining the behavior would typically involve testing the circuit not only over a large number of inputs, but also over a large number of trials without using known priors¹, resulting in a sequence of outcomes which are elements of the “event set”.

2.2.2 Probability in Logics

The two notable approaches towards incorporating probability in logics, involve associating confidences with sentences, and where the truth value of the sentence ranges over the interval $[0, 1]$ and is therefore many-valued. These approaches have a long and distinguished history (see Keynes [92] and Reichenbach [165] as good introductions). Relatively recently, considerations of probability in first order languages were

¹For an eloquent defense of the use of known priors, please see Jaynes [85], whose book is reviewed in a most stimulating manner by Diaconis [47].

treated by Scott and Kraus [173] who attribute Gaifman’s investigation of probability measures [63] on (finitary) first-order languages as an inspiration². Hailperin [73] and Nilsson [135] also consider variations of these notions, again with quantifiers and the confidence of the sentence associated with probability measures. The former author also offers an excellent historical analysis of this work. The work of Fagin and Halpern, and Fagin, Halpern and Megiddo continues in this rich tradition and represents a significant milestone [55, 56].

In contrast with all of this distinguished prior work, the individual variables in PBL are associated with truth values from the set $\{0, 1\}$, and are deterministic, while probability is incorporated into PBL through probabilistic operators. Our dual approach to the treatment of probability and logic stems in part from differing motivations. Whereas the former work has been motivated by inference in the presence of imprecise knowledge, our work has been motivated by the characterization of models of computing, (more specifically Boolean circuits) elements (such as gates) of which may exhibit probabilistic behavior.

2.3 Thermodynamics and the Energy Cost of Computing

Sadi Carnot is widely regarded as the father of modern thermodynamics. Carnot’s study [22] of heat engines was motivated by the desire to improve their efficiency. To model the problem, Carnot proposed an ideal heat engine as a thought experiment, and for the first time showed that the efficiency of any heat engine is limited by the temperatures of the source of heat and the “sink”—where the heat is eventually absorbed to keep the temperature of the engine under control—that any heat engine should possess. He identified that there was a difference between *heat-energy* and *temperature* and it is the heat-energy that is conserved in thermodynamic processes

²The Scott-Kraus development extends it to infinitary languages.

and converted to work, and not the temperature. Carnot's engine served as a foundation upon which Clayperon introduced the concept of thermodynamic reversibility. In thermodynamics, a reversible process is a process, where the initial state of the thermodynamic system can be restored by means of infinitesimal changes in its properties in the reverse direction, without loss or dissipation of energy. In 1848, Lord Kelvin developed the absolute scale of temperature which simplified the expression for the efficiency of Carnot engine [90]. In 1854, Clausius, based on Kelvin's and Carnot's work identified a quantity which he termed entropy, which quantified the amount of heat energy in any heat engine that could not be converted to useful work [39]. Clausius further postulated that the entropy of the universe monotonically increases. This is the famous second law of thermodynamics.

The development of atomic theory of matter, led to the development of kinetic theory of gases and attempts were made to explain the *macrostate* of a system—the macrostate is characterized by the observable properties of the system, such as temperature, volume and pressure—on the basis of its microstate (or the behavior and properties of constituent molecules). Properties of a thermodynamic system, such as pressure, volume and temperature could successfully be explained by the behavior of the constituent molecules. However, entropy defied explanation. In mechanics, the kinetic energy of any moving body could, in principle, be completely converted to work. According to the postulates of the kinetic theory, any ideal gas was a collection of moving molecules and hence, by extension, the collective kinetic energy of the molecules (or equivalently their heat energy) could completely be converted to work. However, this is an apparent contradiction to the law of entropy, which postulates that some of this heat energy could never be converted to work.

In a significant advance, Boltzmann related the amount of *randomness*, or *uncertainty* of the direction of motion and position of the constituent molecules, to the amount of entropy of the system [11]. Informally, considering a six-dimensional *phase*

space of any container of gas with n molecules—with three dimensions representing momentum of the constituent molecules and the other three dimensions representing their position—the position and momentum of all the molecules can simultaneously be represented by n points in this phase space. Given E , the energy of the system, several sets of n points—say W sets—are valid *microstates* of the system. Given the pressure, volume and temperature of a container of gas, the molecules in the container could be in any one of the W microstates, and hence there is an *uncertainty* about the microstate of the system. Boltzmann showed that the natural logarithm of this uncertainty, $\ln W$, is proportional to the entropy S of the system. Boltzmann’s famous equation $S = k \ln W$ where, k is the Boltzmann constant, formalized this relationship. Further contributions to this field of *statistical mechanics* were made by Gibbs [68], Maxwell and others.

Maxwell, in his book “Theory of Heat,” introduced the notion of the Maxwell’s demon [116], as a thought experiment to challenge the second law of thermodynamics. Even though entropy could be explained through the statistical uncertainty of the state of the constituent molecules, a violation of the second law of thermodynamics was seemingly possible by an intelligent being capable of observation and action at the molecular scale. In Maxwell’s own words [116] *“Let us suppose a vessel is divided into two portions A and B, by a division in which there is a small hole, and that a being, who can see the individual molecules opens and closes this hole, so as to allow only the swifter molecules to pass from A to B and only the slower one to pass from B to A. He will thus, without expenditure of work, raise the temperature of B and lower that of A, in contradiction to the second law of thermodynamics”*. The solution to this contradiction defied explanation for several years [105]. Notable contribution was made by Szilard, who showed how the intelligent being could be replaced by a mechanical device [105]. Interestingly, this mechanical device had two states, to indicate whether a molecule was in the left side of the container or in the right side.

Thus, Szilard ought to be widely credited for having inspired the modern notion of a “bit” and a machine with two “states”.

Landauer was the first to provide a satisfactory explanation of the Maxwell’s demon by relating computation to energy [104]. Landauer demonstrated that any mechanical (or a computer-based) Maxwell’s demon had to possess memory, which could be a container of gas, or more generally a bistable well, with a single molecule. The position of the molecule could encode either a 0 or a 1. Initially, if each cell in this memory associated with a single molecule were set to say, 0, subsequent observations of the random molecules by the demon, would encode the position of these molecules in the memory. Hence at the end of its operation, the memory would be filled with a random string—essentially transferring the randomness of the container of the gas into the memory—which needs to be erased to restore the memory to its original state.

The process of erasure, or the process of restoring the random states of the molecules which encode the bits in the memory to a known state, would consume $kT \ln 2$ joules of energy. Hence, in effect, the demon would have expended energy to bring order to the container of gas, and hence does not violate the second law of thermodynamics.

Landauer went on to conjecture that any elemental step of any computation would need to expend $kT \ln 2$ joules of energy. This was shown to be false by Bennett [8] who showed that only logically irreversible steps of computation—a NOT function is logically reversible, since the input can be deduced from the output, whereas an AND function is irreversible, since if the output is 0, the input could be (0, 0), (0, 1) or (1, 0)—need to consume energy. He further demonstrated how all functions could be computed reversibly, and hence, in principle, be implemented through reversible thermodynamic processes which consume no energy. Later Friedkin and Toffoli [60] showed how all logical operations may be realized through reversible logic gates.

Today, computing is implemented using complementary metal oxide semiconductor (CMOS) based very large scale integrated (VLSI) circuits through non-recovering and irreversible processes, where energy, once expended, is not recovered. Meindl and Davis [119] showed that $kT \ln 2$ joules of energy needs to be expended per switching step in CMOS transistors where T is the temperature of operations and k is the Boltzmann constant [11]. However, statistical phenomena such as the thermal noise in deeply scaled VLSI devices, further influence the energy consumption of practical implementation of computing.

Hence, the notion of probability and randomness formalized the notion of entropy in thermodynamics, inspired the Maxwell’s demon, and thus, thermodynamics serves as the basis for the energy cost of computing.

2.3.1 The Energy Cost of Probabilistic Computing

With the general development and the interplay between probability, computing and energy cost of computing as a background, we now detail an aspect that is extensively referred to in the rest of the dissertation: the theoretical relationship between probabilistic computing and energy consumption. The theoretical relationship between probabilistic computing and energy consumption is referred to in the context of the energy efficiency of the models of computing based on PBL. When such a model of computing—the probabilistic Boolean circuit—is implemented in VLSI, we utilize the relationship between probability of correctness and the energy cost of computing presented in Section 2.4.1 to reason about and to empirically demonstrate the energy efficiency of such circuits realized through VLSI.

Probabilistic switches, introduced by Palem [139], relate probabilistic behavior of switches to their energy consumption. A probabilistic switch is an object which realizes a *probabilistic one-bit switching function*. As illustrated in Figure 1(a), the four deterministic one bit switching functions—the four possible one bit input one

Input	output
0	0
1	1

Identity Function

Input	output
0	1
1	0

Complement Function

Input	output	
0	0 (p)	1 (1-p)
1	1 (p)	0 (1-p)

Identity Function

Input	output	
0	1 (p)	0 (1-p)
1	0 (p)	1 (1-p)

Complement Function

Input	output
0	0
1	0

Constant Function

Input	output
0	1
1	1

Constant Function

Input	output	
0	0 (p)	1 (1-p)
1	0 (p)	1 (1-p)

Constant Function

Input	output	
0	1 (p)	0 (1-p)
1	1 (p)	0 (1-p)

Constant Function

(a)

(b)

Figure 1: Following Palem [139], (a) deterministic one bit switching functions and (b) their probabilistic counterparts with probability parameter (probability of correctness) p

bit output functions—have a probabilistic counterpart (Figure 1(b)) with an *explicit* probability parameter (probability of correctness) p . He considered an abstract probabilistic switch sw to be the one which realizes one of these four probabilistic switching functions. Such elementary probabilistic switches may be composed to realize primitive Boolean functions, such as AND, OR, NOT functions [139].

While a switch that realizes the deterministic non-trivial switching function consumes at least $kT \ln 2$ Joules of energy [119], a probabilistic switch can realize a non-trivial switching function with $kT \ln(2p)$ Joules of energy in an idealized setting. Networks of such switches may be considered as a computational model and hence the energy consumption (or energy complexity) of a network of such switches may be studied.

While the network of switches provide a model of computation without state, as an analog to the parallel random access machine (PRAM) model of computing [67], Palem introduced the bit-level random access machine (BRAM) model of computing [138]. The probabilistic counterpart to BRAM model of computing is the randomized BRAM or the RABRAM model of computing. Palem showed [138] that probabilistic algorithms implemented on the RABRAM model of computing are more efficient than

their deterministic counterparts of identical time complexity. In this context, if T is the temperature at which switching takes place and k is the Boltzmann constant [11] and p is probability of correctness, independent of the implementation technology,

Theorem 1 ([139]) *The potential for saving through probabilistic switching over deterministic switching is $kT \ln \frac{1}{p}$ per switching step.*

This theorem relates the energy consumed to the probability of correctness p . Extending this to a full fledged model with state (and memory), if for a state s_i , the cardinality set of the possible next states are $F(s_i)$,

Theorem 2 ([138]) *The energy consumed in evaluating the transition function in the context a state pc_i of any BRAM program \mathcal{P} is at least $kT \ln(F(s_i))$ Joules, furthermore, the energy consumed in evaluating the transition function in the context a state pc_i of any RABRAM program $\mathcal{P}_{\mathcal{R}}$ can be as low as $kT \ln(F(s_i)p)$ Joules, where p is the probability parameter.*

2.4 Current Technology Challenges

With these historical developments as background, we survey current technology challenges [118, 136, 14, 44], with an emphasis on energy consumption and statistical behavior, and outline some approaches towards addressing these challenges.

While the Meindl and Davis [119] fundamental limit of CMOS energy consumption is $kT \ln 2$, Stein took into account the perturbation and error probability of CMOS based inverters, and showed that at least $165kT$ joules of energy needs to be expended per switching step to achieve a reliability of at most one error per 10^{19} switchings [184]. Correspondingly, in practical implementations, to keep the noise margins high, the supply voltage of transistors may not scale at a rate concomitant to their size [44]. Thus, in keeping with Moore’s law, as transistor sizes decrease (resulting in increased density of integration) and their frequency of operations increase (resulting in increased energy consumption per unit time), the *power density*

or the amount of energy consumed per unit area per unit time increases. This leads to increased thermal density with associated reliability and cooling problems, which impact the reliability, size, mobility and operating costs of computing platforms. This has a major impact on the design of microprocessors [14].

Variation in the behavior of highly scaled CMOS devices is another major challenge. The spatial variations in the behavior of CMOS devices (where two devices are non-identical) can be classified into systematic variations and random effects [185]. For example, lithographic lens aberration is a systematic variation while random dopant fluctuation is a random effect. In highly scaled CMOS devices, a few hundred dopant atoms control the electrical behavior of the transistor. In this scenario, a small change in the distribution and placement of these atoms causes a huge variability in the device behavior [136, 21]. The impact of such parametric variations on circuit design has been widely studied [15, 44, 16, 170]. While spatial variations are caused due to material properties and manufacturing difficulties, temporal perturbations in CMOS devices include various type of noise [130, 169, 94, 48].

With this and the energy cost of probabilistic computing presented in Section 2.3.1 as background, we now summarize the relationship between probability of correctness and energy cost of computing in the CMOS-based VLSI domain, studied by Korkmaz [101].

2.4.1 Probabilistic Complementary Metal Oxide Semiconductor Devices

Probabilistic complementary metal oxide semiconductor devices (PCMOS) devices are CMOS devices whose behavior is probabilistic. Of the several possible techniques for realizing PCMOS devices (some of which are described in [141]), it has been demonstrated that ambient thermal noise can be used to randomize the behavior of a conventional CMOS device [102]. The relationship between the probability of correctness of switching and the energy consumption of PCMOS devices was established through

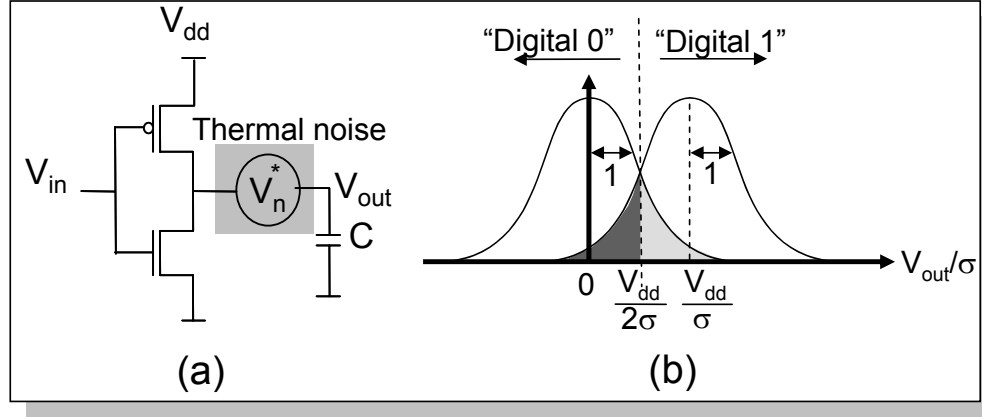


Figure 2: The models from [2] of (a) a PC MOS switch and (b) representation of digital values 0 and 1 and the probability of error for a PC MOS switch

analytical modeling and HSpice based simulations [35, 102, 28] as well as actual measurements of fabricated PC MOS based devices [101]. Assuming a supply voltage of V_{dd} , a noise RMS value of σ , and a capacitance C , if the thermal noise is modeled as an output coupled voltage source with a Gaussian distribution, as illustrated in Figure 2, errors occur—a digital 1 is treated as a 0 or vice versa—if the output voltage levels are in the gray shaded area of Figure 2(b). The relationship between noise magnitude σ , the switching voltage V_{dd} , and the probability of error can be obtained by calculating the area under the curve shaded by gray. Since the switching energy $E = CV_{dd}^2$

Law 1: Energy-probability Law: (from [2]) For any fixed technology generation (which determines the capacitance $C = \hat{C}$) and constant noise magnitude $\sigma = \hat{\sigma}$, the switching energy $\hat{E}_{\hat{C},\hat{\sigma}}$ consumed by a probabilistic switch grows with p . Furthermore, the order of growth of $\hat{E}_{\hat{C},\hat{\sigma}}$ in p is asymptotically bounded below by an exponential in p .

Extending this relationship to a Boolean *gate*, we illustrate in Figure 3(a) the relationship between the energy consumption per switching step to the probability of correctness for a NAND gate and an inverter at 90nm technology, whereas we compare the relationship between the energy consumption per switching step to the probability

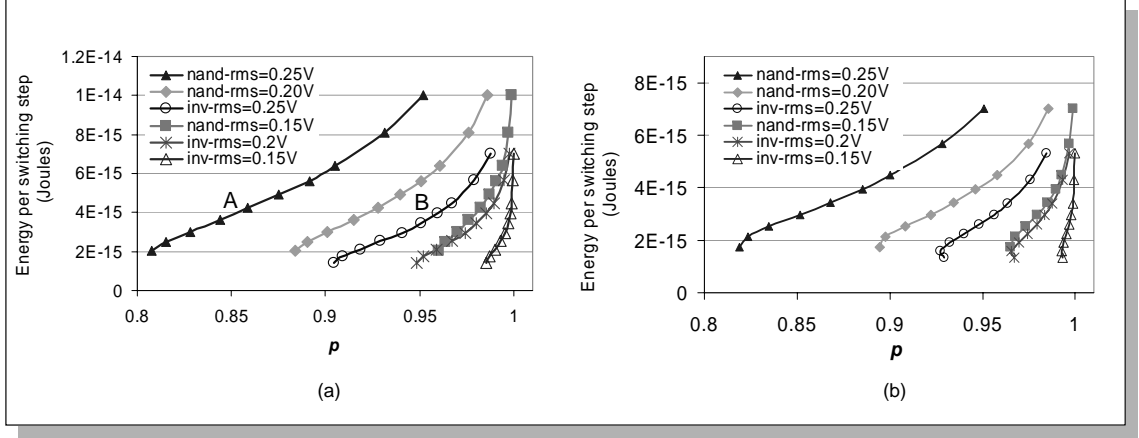


Figure 3: (a) The relationship between energy per switching step and probability of correctness for a NAND gate and an inverter at $90nm$ technology, and (b) the same relationship at $65nm$ technology (from [101])

of correctness for the NAND gate and the inverter at $65nm$ technology, in Figure 3(b). Both of these were obtained by analytical modeling. Further details of the modeling and HSpice based simulation results are presented elsewhere [101].

While we have presented a brief overview of the challenges posed by variability with a CMOS-centric view, the problems of reliability and variation are present in non-CMOS devices as well [79]. We now briefly survey the techniques adopted to overcome these challenges posed by energy consumption and variability.

2.4.2 Techniques for Energy Efficient and Error-free Computing

Energy efficient *error free* computation is a vast area of research. This problem has been studied at several levels: at the level of circuits, micro-architecture, memory, operating systems, algorithms, applications and compilers [147, 71, 154, 155]. The domain of digital signal processing (DSP) offers a good illustration for these techniques and hence we shall survey techniques relevant in the context of this dissertation, with an emphasis on DSP.

Energy efficient digital signal processing is a large area in its own right [32]. At the circuit level, the use of voltage scaling for reduced energy consumption has been

explored. In these techniques, increased propagation delay was considered to be the primary drawback to voltage overscaling. To maintain circuit performance and correctness while simultaneously realizing energy savings through voltage scaling, several researchers employ the use of multiple supply voltages by operating elements along the critical path at nominal voltage and reducing supply voltages along non-critical paths [33, 112, 200, 201]. Other techniques involve detecting and correcting switching errors due to voltage overscaling [54]. Supply voltage scheduling along with task scheduling to match application demand with the speed of operation has been studied as well [93].

Offering a contrasting approach, in [77, 178, 193], propagation delay errors are removed through error correction in a collection of techniques named “algorithmic noise-tolerance”. In [77], difference-based and prediction-based error correction approaches are investigated and in [193], adaptive error cancellation is employed using a technique similar to echo cancellation.

2.4.3 The Trade off between Energy, Error and Quality of Solution

As opposed to techniques described in Section 2.4.2, where application level correctness (or quality of solution) is maintained, techniques which trade-off energy for quality of solution have been investigated. Historically, techniques which do not trade quality, use better algorithms, replace complex operations such as multiplications with simpler operations such as additions and eliminate redundant computations. The techniques which trade quality of solution for energy efficiency, rely on the following observation: digital signal processing, by its very nature involves discretization of signals and coefficients, and quantization of signals. Such discretization and the limitation of the precision of the coefficients used, impact the quality [198] and afford an opportunity for optimizations that yield energy efficiency.

For example, the Poorman’s transform uses approximate values for the complex

exponential coefficients involved in multiplications in the discrete Fourier transform (DFT) algorithm. These approximate values allow for the replacement of multiplications with additions [103]. This ultimately produces errors, but yields energy efficiency. Similarly, the number of complex multiplications needed to perform a discrete Fourier transform may be reduced through varied techniques [164, 5, 17]. A complementary approach towards reducing the number of arithmetic operations is to apply coarse quantization to the signal values instead of approximating the coefficients, and by exploiting the overlap between signal frames [132]. Other adaptive and non-adaptive techniques utilize precision requirements along with incremental refinement (where more operations produce better results) [133, 3, 4, 181, 131], adjust the length of filter chain [107], adjust the values of the filter coefficients dynamically [195], or adjust the filter order dynamically based on input data [108, 146].

As a radical departure from these techniques, George et. al. [66], demonstrated how 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 [101]. Our probabilistic arithmetic provides the theoretical framework for this counter intuitive example, and the results of George et. al. are surveyed in Section 7.5.

2.4.4 Theoretical Approaches to Computing in the Presence of Faults

Utilizing implicitly probabilistic logic elements for reliable computing is a concept which dates back to von Neumann’s seminal work, where he studied techniques such as NAND multiplexing and majority voting to increase reliability of faulty logic gates [192]. von Neumann showed that if the failure probability of gates were statistically independent and low, computation can be performed reliably with a high

probability. Other researchers have improved upon von Neumann’s techniques to calculate the necessary and sufficient amount of redundancy to perform Boolean functions [50, 51]. These results were improved upon by Pippenger who showed how Boolean functions may be computed reliably (with constant multiplicative redundancy) by gates susceptible to noise [153, 150, 151].

In the context of parallel random access machines (PRAMS) [67], a different model of computation than circuits, algorithms and techniques for deterministic computation in the presence of processor and memory faults have been studied extensively in theory. Faults are modeled as events with some probability distribution. Most of these techniques involve fault correction based on redundancy or faults detection and re-execution [159, 109, 89, 36].

2.4.5 Practical Approaches to Computing In the Presence of Faults

Practical designs which exploit redundancy to achieve reliability at the circuit-level while utilizing faulty circuit elements (noise susceptible CMOS gates for example) have been demonstrated as well. In the domain of CMOS, the “probability of correctness” of a CMOS device originates from the probabilistic nature of charge transport as well as extraneous events like hits from energetic particles [127]. Bahar et al. demonstrate methods for improving the noise immunity of logic circuits by adopting design styles based on Markov Random Fields [6, 134]. Energy efficiency, performance and implementing probabilistic application are not the main considerations of this work. In addition, a large body of literature has covered circuit, architecture and software techniques for robustness in the presence of single event upset [204] caused by radiation interferences [203]. These techniques are analogous to those surveyed in a theoretical context in Section 2.4.4, which achieve reliable computation in the presence of faulty logic elements.

At the architecture level, the *architecture vulnerability factor* [129] (which is the

ratio of the number of bits required for the *architecturally correct execution* of the program to the total number of bits in the structure) quantifies the susceptibility of the architecture to perturbations. Architecture-level fault tolerance techniques range from duplication of instructions at the software and hardware levels [163, 166] to the duplication of threads [128] and the entire hardware [202]. These techniques are analogous to those surveyed in a theoretical context in the PRAM model of computation.

Broadly, these approaches can be divided into *fault tolerance* and *fault avoidance*. Whereas the former seeks to detect and rectify faults dynamically, the latter relies on apriori testing to eliminate defective elements. Conventional approaches to fault tolerance have included designing redundant systems with reliable arbitrators [179]. Fault tolerance approaches include techniques like speculative execution on faster (but less reliable) logic elements and verification by slower and more reliable logic elements [84]. Fault avoidance approaches have been studied in the context of reconfigurable architectures, where faulty blocks are not utilized for computing [69].

As a contrast, the empirical parts of this dissertation use *no redundancy* and *trade-off energy for quality of solution* to achieve computation in the presence of probabilistic behavior of logical and arithmetic primitives.

CHAPTER III

A PROBABILISTIC BOOLEAN LOGIC AND ITS MEANING

In this chapter, we incorporate considerations of probability into logic and introduce *Probabilistic Boolean Logic* (PBL). Probabilistic Boolean logic captures attributes of Boolean logic as well as probability in a unified model, through a probabilistic extension to Boolean logic where the three canonical operators—conjunction, disjunction and negation—have an associated probability p : ($\frac{1}{2} \leq p \leq 1$) of “correctness”.

The study of such a logic which incorporates probability in an “implicit” manner, is interesting in its own right. In this context, we define the meaning of any formula in PBL and study several interesting properties of PBL, some of which are analogous to those of Boolean logic, and unearth some interesting differences. Motivated by our desire to study computational models based on PBL, we introduce and study *Probabilistic Boolean Circuits*, and relate them to classical models of computation. We explicitly introduce state and relate this model of computation to the celebrated probabilistic automata model of Rabin [156]. We provide theoretical evidence, rooted in thermodynamics, that computation implemented using such implicitly probabilistic models of computing are likely to be more efficient than their deterministic counterparts as well as their counterparts implemented using explicitly probabilistic models of computing.

This *implicit* approach to realizing probabilistic computing could be based on using naturally probabilistic phenomena, and thus, there is *no* need for an external random source. Here, sources of randomness could include various types of

noise [35, 130, 169]—an increasingly perceptible phenomenon in physically deterministic devices [94]—and others. We note in passing that probabilistic behavior of the implicit type is anticipated increasingly in CMOS devices, gates and circuits—the building blocks of modern computers—and are caused by manufacturing deficiencies and noise susceptibility [82] as physical sizes of individual transistors approach $20nm$. This is viewed as an impediment to realizing deterministic switches and hence to Moore’s law [123]. Characterizing this implicit approach to probabilistic computing and logic formally, and providing an approach to distinguishing it from its explicit counterpart, serve as the themes of Chapter 3, Chapter 4 and Chapter 5.

3.1 *Probabilistic Boolean Logic and Well-Formed Formulae*

Informally, probabilistic Boolean formulae—like their deterministic counterparts—can be constructed from the Boolean constants $0, 1$, Boolean variables, and *probabilistic Boolean operators*: *probabilistic disjunction*, *probabilistic conjunction* and *probabilistic negation*. Probabilistic disjunction, conjunction and negation will be represented by the symbols \vee_p, \wedge_q and \neg_r respectively, where p, q, r are the corresponding probability parameters or *probabilities of correctness*. The probabilities of correctness associated with the disjunction, conjunction and negations operators are such that $\frac{1}{2} \leq p, q, r \leq 1$ and $p, q, r \in \mathbb{Q}$, the set of rationals. Initially, for clarity of exposition and for a model of finite cardinality, we consider only rational probabilities of correctness. We seek the indulgence of the reader and will defer a more detailed discussion of the justification underlying our choice of considering rational probabilities, to Section 3.2. A pair of probabilistic operators, say in the case of probabilistic disjunction, $\vee_p, \vee_{\hat{p}}$, will be deemed identical whenever $p = \hat{p}$. They will be considered to be *comparable* whenever $p \neq \hat{p}$; similarly for probabilistic conjunction and negation. Analogous to well-formed Boolean formulae, well-formed *probabilistic Boolean formulae* are defined as follows:

1. Any Boolean variable x, y, z, \dots and the constants 0,1 are well-formed probabilistic Boolean formulae¹.
2. If F, G are well-formed probabilistic Boolean formulae, $(F \vee_p G)$, $(F \wedge_p G)$ and $(\neg_p F)$ are well-formed probabilistic Boolean formulae.

Henceforth, we will use the term probabilistic Boolean formula, or PBF to refer to a well-formed probabilistic Boolean formula and the term Boolean formula (BF) to refer to a classical well-formed Boolean formula (which is deterministic). In addition, the length of a probabilistic Boolean formula is the number of operators n in the formula. Given a PBF F , we will use VAR_F to denote the set of variables in F . If $\text{VAR}_F = \phi$, that is if F is a formula over Boolean constants, F will be referred to as a *closed* well-formed probabilistic Boolean formula or a *closed* PBF.

3.1.1 Boolean Logic Preliminaries

For any Boolean formula or BF J consider the set of its constituent Boolean variables, $\{x_1, x_2, x_3, \dots, x_k\}$ denoted by BVAR_J where $|\text{BVAR}_J| = k$. Consider any assignment $I \in \langle 0, 1 \rangle^k$. Let J_I be the closed formula obtained by replacing each variable of J with the Boolean constant it is assigned. The value of the formula J , when x_i is assigned the i^{th} element (bit) of I , or equivalently, the value of the formula J_I , will be referred to as the *truth value* of J with (input) assignment I and will be denoted by $T(J_I)$. Given two Boolean formulae J, K , without loss of generality, let $\text{BVAR}_K \subseteq \text{BVAR}_J$. If I is an assignment to variables in J , I' is a *consistent assignment* to variables in K if and only if whenever $x_i \in \text{VAR}_K$, x_i is assigned to the same Boolean constant under the assignments I and I' .

Two Boolean formulae J and K where $|\text{BVAR}_J| = k$ are considered to be equivalent, whenever $T(B_I) = T(C_{I'})$ for all input assignments. We recall that one approach to specifying the truth value of Boolean formulae is through a Boolean truth

¹Typically we shall denote Boolean variables using lower case alphabets.

Input x y z	Truth Value
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

Figure 4: A Boolean truth table for the formula $((x \wedge y) \vee (x \wedge z)) \vee (y \wedge z)$

table. A truth table with 2^k , $k > 0$ rows and two columns is illustrated in Figure 4. Conventionally, the first column of each row contains the input assignment, where the n^{th} row, $0 \leq n < 2^k$, corresponds to the k bit binary representation of n , which we denote by N . The second column of each row contains an element of $\{0, 1\}$ where the symbols 1 and 0 denote the *true* and *false* values respectively. Referring to the example in Figure 4, the truth table corresponds to the Boolean formula $((x \wedge y) \vee (x \wedge z)) \vee (y \wedge z)$. The third row of the table with the input 010, is interpreted as the assignment $\langle x = 0, y = 1, z = 0 \rangle$, and yields the truth value of the formula to be 0 and hence the second column of this row contains a 0. In contrast, the fourth row which contains the input 011, with the symbol 1 in the second column, implying that the value of the formula for this assignment is 1.

3.1.2 The Operational Meaning of Probabilistic Boolean Operators

Let F, G, H denote $(x \vee_p y)$, $(x \wedge_q y)$ and $(\neg_r x)$ respectively, and let $T(F_\alpha), T(G_\beta)$ and $T(H_\gamma)$ denote their truth value under the assignments α, β and γ respectively. Then an informal operational approach to assigning or determining “truth” in the case of

a PBF is

$$\begin{aligned}
T(F_\alpha) &= \begin{cases} \text{Truth value of } (x \vee y) & \text{under the input assignment } \alpha \text{ with probability } p \\ \text{Truth value of } \neg(x \vee y) & \text{under } \alpha \text{ with probability } (1 - p) \end{cases} \\
T(G_\beta) &= \begin{cases} \text{Truth value of } (x \wedge y) & \text{under the input assignment } \beta \text{ with probability } q \\ \text{Truth value of } \neg(x \wedge y) & \text{under } \beta \text{ with probability } (1 - q) \end{cases} \\
T(H_\gamma) &= \begin{cases} \text{Truth value of } (\neg x) & \text{under the input assignment } \gamma \text{ with probability } r \\ \text{Truth value of } (x) & \text{under } \gamma \text{ with probability } (1 - r) \end{cases}
\end{aligned}$$

3.1.3 Probabilistic Boolean Formulae and their Truth Tables

Let us now extend this notion of truth with associated probability to arbitrary formulae in PBL. Our initial approach will be through a *probabilistic Boolean truth table*. As shown in Figure 5 and analogous to conventional truth tables, in a probabilistic truth table with $l = 2^k$ ($k > 0$) rows and three columns, the first column of the n^{th} row contains N , the k bit binary representation of n , $0 \leq n < 2^k$. The second and the third columns of the n^{th} row contain rational numbers $0 \leq p_n, q_n \leq 1$ where $p_n + q_n = 1$. The first column of the n^{th} row, which contains the binary representation N of n , is an assignment of Boolean constants to the variables in the formula as shown in the Figure 5. The second column of the n^{th} row, which is labeled p_n , represents the fact that the probability that value of the formula F_N is 1 is p_n for the assignment N , whereas the third column labeled q_n is the probability that the value of the *same formula* for the *same input assignment* is 0. For example, if F is a PBF over the variables x, y, z , and considering the row of the table with the assignment 010, the probability that the value of F is 1 for this assignment is $p_2 = 1/4$ whereas the probability that the value of F is 0 is $q_2 = 3/4$.

Input	Probabilities	
x y z	Truth Value=1	Truth Value=0
0 0 0	1/4	3/4
0 0 1	1/4	3/4
0 1 0	1/4	3/4
0 1 1	3/4	1/4
1 0 0	1/4	3/4
1 0 1	1	0
1 1 0	1	0
1 1 1	1	0

Figure 5: A probabilistic Boolean truth table for the PBF $((x \wedge_1 y) \vee_1 (x \wedge_1 z)) \vee_1 (y \wedge_{3/4} z)$

3.2 The Event Set Semantics of Probabilistic Boolean Logic

In Section 3.1.2, we have introduced an operational meaning of PBL and established the fact that probabilistic Boolean formulae in this logic can be represented by probabilistic Boolean truth tables. Given a PBF, intuitively, for any assignment of values to the variables in the PBF, the value of the PBF is determined by (i) the operators (probabilistic disjunction, conjunction or negation) in the PBF and (ii) the probabilities of correctness of each of the operators. Whereas the former captures the notion of the “underlying” deterministic Boolean formula, the latter characterizes the probability that the truth value of the PBF matches that of the underlying deterministic Boolean formula. Note that this probability might vary with the input assignments, and in general, indeed it does. Based on these two observations, we will formalize the meaning of PBF in PBL based on the meaning of Boolean logic, and the frequentist

interpretation of probability [191], for a given input I .

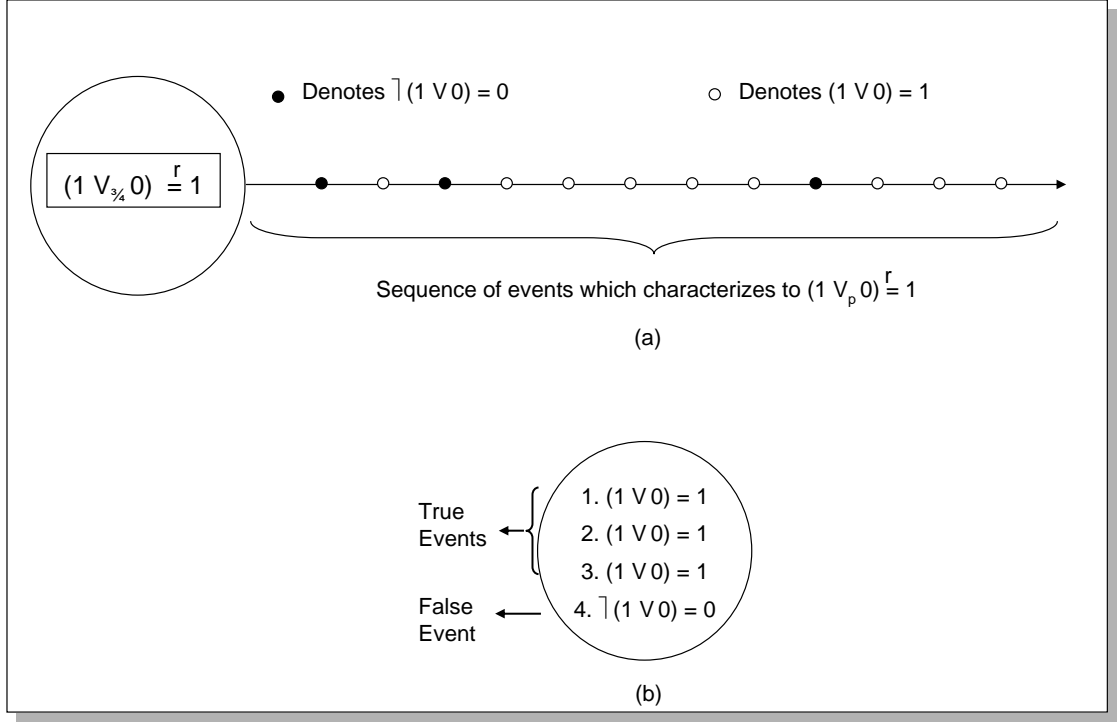


Figure 6: (a) A frequentist interpretation of a sentence $(1 \vee_{3/4} 0) \stackrel{r}{=} 1$ in PBL through an infinite sequence of events and (b) a succinct representation of this sequence as an event set

3.2.1 A Frequentist View of Probabilistic Boolean Logic

If F is any PBF and I is an assignment to variables in F , then F_I will be used to denote the closed PBF where every variable in F is replaced by the Boolean constant it is assigned. We will use the symbol $\stackrel{r}{=}$ to mean “is equal to with a probability r ”. Also, for any assignment I to the variables in F , we will use \mathcal{S}_I to denote the sentence $F_I \stackrel{r}{=} 1$ (and $\bar{\mathcal{S}}_I$ to denote the sentence $F_I \stackrel{\bar{r}}{=} 0$). Our goal is to provide a semantic framework that gives meaning to sentences formally. To this end, consider a closed PBF F_I of the form $(1 \vee_p 0)$ where $p = 3/4$. We recall that from the operational meaning given to the \vee_p operator, the probability that the truth value of F_I is equal to $T(1 \vee 0)$ is $3/4$, whereas the probability that the truth value of F_I is equal to $T(\neg(1 \vee 0))$ is $1/4$. Since the symbol $\stackrel{r}{=}$ means “is equal to with a probability r ”, the

sentence \mathcal{S}_I which denotes $(1 \vee_p 0) \stackrel{r}{=} 1$ is *valid* if and only if $p = r$; \mathcal{S}_I is an *invalid* sentence otherwise.

Considering \mathcal{S}_I , under the frequentist interpretation of probability, an infinite sequence Υ consists of two types of events, each associated with a sentence in classical (deterministic) Boolean logic as follows: in our example (Figure 6(b)), one type of event corresponds to those instances where F_I “behaves like” $(1 \vee 0)$ and hence the event is associated with the sentence in Boolean logic $(1 \vee 0) = 1$, whereas the latter corresponds to those instances where F_I “behaves like” $\neg(1 \vee 0)$ and hence the event is associated with the sentence $\neg(1 \vee 0) = 0$. This concept is illustrated in Figure 6(a) which shows the infinite sequence of events, each associated with a sentence. With $p = 3/4$, we note that the relative frequency of the events which correspond to sentences of the form $(1 \vee 0) = 1$ is $3/4$. Thus, our semantic interpretation of the validity of a sentence in our example, is based on the validity (and the ratio) of the two types of sentences in Boolean logic, $(1 \vee 0) = 1$ and $\neg(1 \vee 0) = 0$. The first type of event is characterized by the sentence $(1 \vee 0) = 1$ being *valid* whereas the second type of event is characterized by the *validity*² of the sentence $\neg(1 \vee 0) = 0$. The probability parameter p determines the relative frequency of these events as n , the number of events $\rightarrow \infty$.

Rather than considering the infinite sequence of events Υ , we will use its finite representation or encoding of probability parameter, as follows: in our example, we consider a set (an “event set”) of 4 distinct events, three of which correspond to the sentence in Boolean logic, $(1 \vee 0) = 1$ and one event which corresponds to $\neg(1 \vee 0) = 0$. Such a succinct representation for the infinite sequence in Figure 6(a) is shown in Figure 6(b). To reinforce this point further, consider longer formulae, say H , of the form $((x \vee_p y) \vee_q z)$ where $p = 3/4$ and $q = 5/6$. Again, we will consider the sequence

²For a notion of validity of sentences and the semantics of Boolean logic—in fact the whole of predicate calculus—please see Mendelson [121].

which corresponds to the sentence \mathcal{S}'_I which denotes $H \stackrel{r}{=} 1$ where I denotes the assignment $\langle x = 1, y = 0, z = 1 \rangle$. The sequence Υ' associated with \mathcal{S}'_I would consist of events $((1 \vee 0) \vee 1) = 1$, $(\neg(1 \vee 0) \vee 1) = 1$, $\neg((1 \vee 0) \vee 1) = 0$ or $\neg(\neg(1 \vee 0) \vee 1) = 0$ with relative frequencies of $15/24$, $5/24$, $3/24$ and $1/24$ respectively. This infinite sequence may be represented in a succinct manner with a set of 24 elements, 15 of which are copies³ of the sentence $((0 \vee 1) \vee 0) = 1$, 5 elements being copies of $(\neg(0 \vee 1) \vee 0) = 0$, 3 elements being copies of $\neg((0 \vee 1) \vee 0) = 0$ and a single element of the form $(\neg(0 \vee 1) \vee 0) = 0$. From such a succinct representation, the sequence Υ' may be generated by picking elements uniformly at random and constructing an infinite sequence of such trials. Since events are picked at random, the sequence Υ' satisfies both the axiom of convergence and the axiom of randomness (please see Reichenbach [165] and Section 2.2.1) in the frequentist interpretation of probability.

A motivation towards developing PBL is to design efficient algorithms to synthesize implicitly probabilistic circuits and the computational efficiency of such algorithms is dependent on the size of the event sets. Therefore, we expect that it is advantageous to represent the sequence Υ' as a finite set, which is the basis for restricting the probability parameter of the operators of PBL to be the member of the set of rationals \mathbb{Q} . We note that if probabilities are drawn from the unit interval $[0, 1]$, the cardinality of the event set will not be finite and a notion of probability measure [99] has to be introduced. However, we note that the subsequent development of the semantics of PBL can be extended naturally to the case where the probability parameters of the operators are chosen from the interval $[0, 1]$.

³Since our intention is to characterize the elements as a set, for element distinctness, we ensure that the copies of each sentence is indexed uniquely from the set of naturals $\{0, 1, 2, \dots\}$, and thus individual copies can be distinguished from each other through this index. For ease of exposition, we will omit these indices in the body of this and subsequent chapters, but will include it in a rigorous formulation of these concepts in Section 3.3.

3.2.2 An Interpretation of a Probabilistic Boolean Formula for a Fixed Assignment Through Event Sets

With the frequentist interpretation of probability as a background, we will define the succinct representation of the infinite sequence of trials which characterizes a sentence in PBF. Revisiting the example in Figure 6, let \mathcal{S}_I denote $(1 \vee_p 0) \stackrel{r}{=} 1$ and Υ is the sequence which characterizes \mathcal{S}_I . We will refer to $\mathbf{E}_{\mathcal{S},I}$, the succinct representation of Υ as an *event set* of \mathcal{S}_I . In our example, any *event* $E \in \mathbf{E}_{\mathcal{S},I}$ will be associated with either the sentence $(1 \vee 0) = 1$ in Boolean logic, or with the sentence $\neg(1 \vee 0) = 0$. If $p = m/n$ ($p \in \mathbb{Q}$), $\mathbf{E}_{\mathcal{S},I}$ is a set of n elements (each element referred to as an event), m of which correspond to $(1 \vee 0) = 1$ and the rest to $\neg(1 \vee 0) = 0$. We will refer to the former type of events as being true whereas the latter type of events will be deemed to be false. Intuitively, the true events are witnesses to the formula under assignment I yielding a value of 1 whereas the false events correspond to those which yield a value of 0. Let $\psi(\mathbf{E}_{\mathcal{S},I})$ represent the fraction of the event set made up of copies of true events, the sentence $(1 \vee 0) = 1$.

Revisiting Figure 6, if $r = 3/4$, \mathcal{S}_I is a valid sentence and it is invalid otherwise. We can either say “ $r = 3/4$ is the value for which the sentence $F_I \stackrel{r}{=} 1$ is valid”, or this fact can be stated as “ F is *satisfied with probability* $r = 3/4$ *for the assignment* I ”. Given the event set $\mathbf{E}_{\mathcal{S},I}$ the rational number r and the Boolean constant 1, they are said to be in a relationship R , that is $(1, r, \mathbf{E}_{\mathcal{S},I}) \in R$, if and only if $\psi(\mathbf{E}_{\mathcal{S},I}) = r$. If $(1, r, \mathbf{E}_{\mathcal{S},I}) \in R$, then the sentence $(1 \vee_p 0) \stackrel{r}{=} 1$ is said to be *valid* under our interpretation; it is *invalid* otherwise.

Now consider the assignment \bar{I} which denotes $\langle x = 0, y = 0 \rangle$. As shown in Figure 7(a), a majority of the events in the event set are false events. In this context, it is more natural to reason about the validity of the sentence $\bar{\mathcal{S}}_{\bar{I}}$, which denotes $F_{\bar{I}} \stackrel{\bar{r}}{=} 0$ or $(0 \vee_p 0) \stackrel{\bar{r}}{=} 0$. If $\bar{\psi}(\mathbf{E}_{\bar{\mathcal{S}},\bar{I}})$ is the fraction of events in $\mathbf{E}_{\bar{\mathcal{S}},\bar{I}}$ which are copies of false events, $\bar{\mathcal{S}}_{\bar{I}}$ is a valid sentence if and only if $\bar{r} = \bar{\psi}(\mathbf{E}_{\bar{\mathcal{S}},\bar{I}})$. In this case,

$\bar{r} = 3/4$ is the value for which the sentence $F_{\bar{I}} \stackrel{\bar{r}}{=} 0$ is valid. Equivalently, we can say that F is *unsatisfied with probability $\bar{r} = 3/4$ for the assignment \bar{I}* . We note that $\bar{\psi}(\mathbf{E}_{\mathcal{S},I}) = 1 - \psi(\mathbf{E}_{\mathcal{S},I})$ and therefore, a sentence $F_I \stackrel{r}{=} 1$ is a valid sentence if and only if $F_I \stackrel{\bar{r}}{=} 0$ is a valid sentence, where $\bar{r} = (1 - r)$. For ease of exposition, and unless specified otherwise, we consider only sentences of the form $F_I \stackrel{r}{=} 1$, and reason about the probabilities with which F is satisfied. A rigorous formulation of validity of sentences in each case—sentences of the form $F_I \stackrel{r}{=} 1$ as well as those of the form $F_{\bar{I}} \stackrel{\bar{r}}{=} 0$ —is treated in a complete manner in Section 3.3.

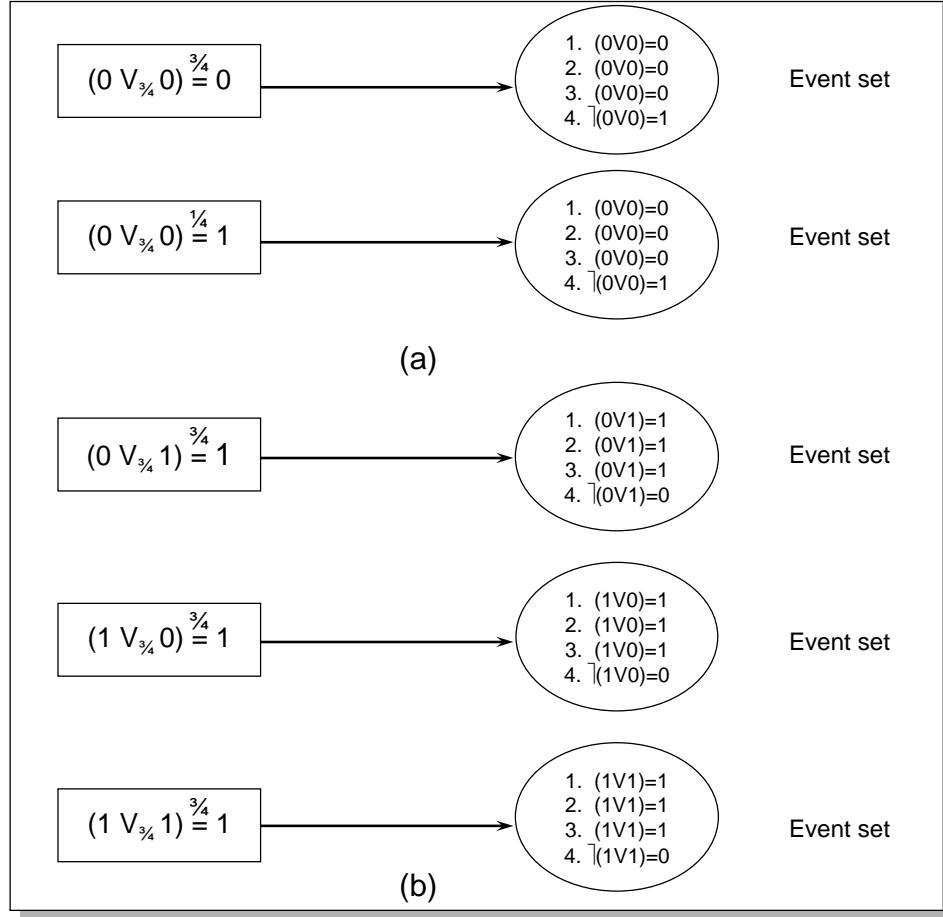


Figure 7: (a) The event set for the valid sentence $(0 \vee_{\frac{3}{4}} 0) \stackrel{\frac{3}{4}}{=} 0$ and $(0 \vee_{\frac{3}{4}} 0) \stackrel{\frac{1}{4}}{=} 1$ (b) three valid sentences and their event sets for the three remaining assignments to $(x \vee_{\frac{3}{4}} y)$

We observe that, as illustrated in Figure 7(b), for a formula F , for each of the

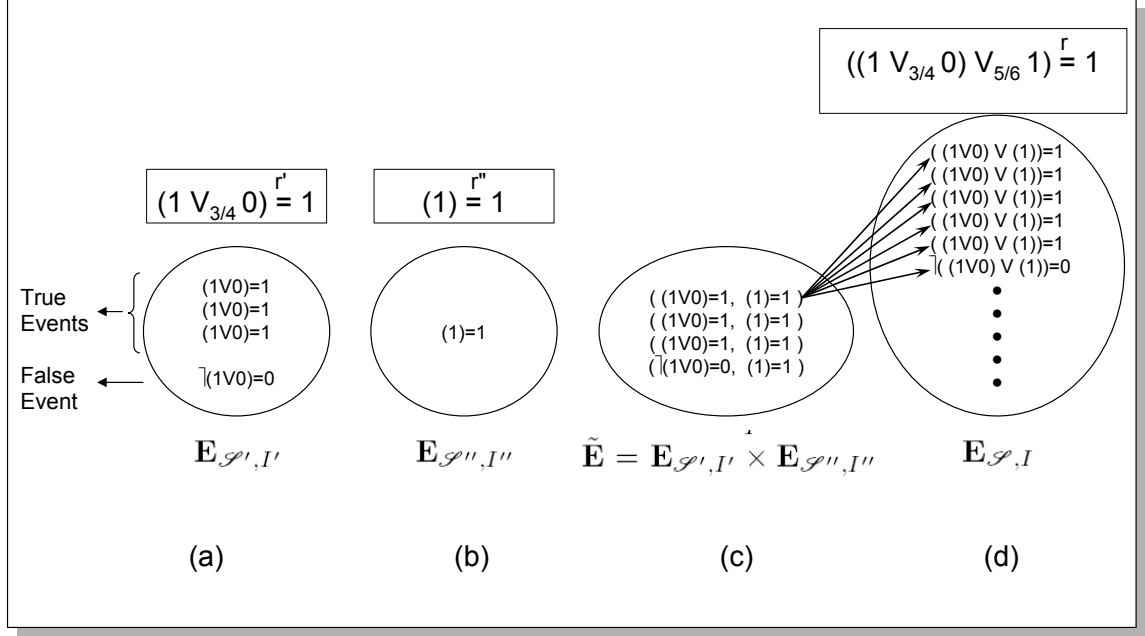


Figure 8: (a) Event set $\mathbf{E}_{\mathcal{S}', I'}$ of $(1 \vee_{3/4} 0) \stackrel{r'}{=} 1$ (b) event set $\mathbf{E}_{\mathcal{S}'', I''}$ of $(1) \stackrel{r''}{=} 1$ (c) $\tilde{\mathbf{E}} = \mathbf{E}_{\mathcal{S}', I'} \times \mathbf{E}_{\mathcal{S}'', I''}$ (d) constructing the event set for $((1 \vee_{3/4} 0) \vee_{5/6} 1) \stackrel{r}{=} 1$ from $\tilde{\mathbf{E}}$.

three remaining assignments $I \in \{\langle x = 0, y = 1 \rangle, \langle x = 1, y = 0 \rangle, \langle x = 1, y = 1 \rangle\}$, three valid sentences, each of the form $F_I \stackrel{r}{=} 1$ can be constructed, and each sentence is associated with its own event set. The collection of events sets and the notion of validity provides a *model* [121] in the sense of symbolic logic.

Consider any PBF G of the form (z) where z is a Boolean variable. For the assignment I which assigns 0 to z , if \mathcal{S}_I is the sentence $G_I \stackrel{r}{=} 1$, the event set $\mathbf{E}_{\mathcal{S}, I}$ consists of one event determined by the sentence in Boolean logic, $(0) = 0$. Similarly, for the assignment I' which is $\langle z = 1 \rangle$, the event set $\mathbf{E}_{\mathcal{S}, I'}$ consists of one event determined by the sentence $(1) = 1$.

We will now consider the event set of a PBF H of length $k + 1$ where $k \geq 0$. To illustrate the way in which event sets of sub-formulae combine, we consider an example where F and G are the formulae $(x \vee_q y)$ and (z) respectively, where H is of the form $(F \vee_p G)$, $q = 3/4$ and $p = 5/6$. We will consider the assignment $I = \langle x = 1, y = 0, z = 1 \rangle$ to the variables in H , where $I' = \langle x = 1, y = 0 \rangle$ and $I'' = \langle z = 1 \rangle$ are

the corresponding consistent assignments to F and G . Consider the valid sentences $\mathcal{S}_I, \mathcal{S}'_{I'}, \mathcal{S}''_{I''}$ which denote $H_I \stackrel{r}{=} 1$, $F_{I'} \stackrel{r'}{=} 1$ and $G_{I''} \stackrel{r''}{=} 1$ respectively, where $\mathbf{E}_{\mathcal{S},I}$, $\mathbf{E}_{\mathcal{S}',I'}$ and $\mathbf{E}_{\mathcal{S}'',I''}$ are the event sets of \mathcal{S}_I , $\mathcal{S}'_{I'}$ and $\mathcal{S}''_{I''}$ respectively. Referring to Figure 7, the event set of $\mathcal{S}'_{I'}$ consists of 4 events, 3 of which are true Boolean sentences $(0 \vee 1) = 1$ and one false Boolean sentence $\neg(0 \vee 1) = 0$. This is shown in Figure 8(a), where for the ease of exposition, we omit the indices of the events. With $z = 1$, as shown in Figure 8(b), the event set of $\mathcal{S}''_{I''}$ has one true event associated with the Boolean sentence $(1) = 1$. Let $\tilde{\mathbf{E}} = \mathbf{E}_{\mathcal{S}',I'} \times \mathbf{E}_{\mathcal{S}'',I''}$. As shown in Figure 8(c), we note that $|\tilde{\mathbf{E}}| = 4 \times 1 = 4$, and any element of $\tilde{\mathbf{E}}$ is of the form $(B = c, \hat{B} = \hat{c})$, where B, \hat{B} are closed BF and $c, \hat{c} \in \{0, 1\}$. For each element of $\tilde{\mathbf{E}}$, as shown in Figure 8(d), we create 5 copies (since $p = 5/6$) each of the form $(B \vee \hat{B}) = T(c \vee \hat{c})$ and 1 element of the form $\neg(B \vee \hat{B}) = T(\neg(c \vee \hat{c}))$ to get $\mathbf{E}_{\mathcal{S},I}$. Hence it follows that $|\mathbf{E}_{\mathcal{S},I}| = 6 \times |\tilde{\mathbf{E}}| = 24$, of which 20 events are true and the rest are false. Therefore, whenever $r = 5/6$, \mathcal{S}_I is a valid sentence, since $P_H = \psi(\mathbf{E}_{\mathcal{S},I}) = 20/24 = 5/6$. A rigorous formulation can be found in Section 3.3. We will however describe some attributes of the event sets for sentences which correspond to arbitrary formulae and assignments. These attributes will be used in Chapter 4 to characterize some of the properties of PBL.

In general, let H be of the form $(F \vee_p G)$ where $p = m/n$. To reiterate, for any assignment I to H , let I' and I'' denote the corresponding consistent assignment to variables in F and G respectively. Let the number of events in $\mathbf{E}_{\mathcal{S}',I'}$ be denoted by the symbol a , and let $|\mathbf{E}_{\mathcal{S}',I'}| = b$. Similarly, let the number of true events in $\mathbf{E}_{\mathcal{S}'',I''}$ be denoted by the symbol c , and let $|\mathbf{E}_{\mathcal{S}'',I''}| = d$.

Observation 3.2.2.1 *Under assignment I , $|\mathbf{E}_{\mathcal{S},I}|$ is (bdn) where $\mathbf{E}_{\mathcal{S},I}$ has $(acm + a(d - c)m + (b - a)cm + (b - a)(d - c)(n - m))$ true events. Therefore, if P_F, P_G and P_H denote the probabilities with which $F_{I'}, G_{I''}$ and H_I are respectively satisfied, $P_H = (P_F)(P_G)p + (1 - P_F)(P_G)p + (P_F)(1 - P_G)p + (1 - P_F)(1 - P_G)(1 - p)$.*

PROOF. Based on the frequentist interpretation of PBL and the event set semantics, we know that the probability that H is satisfied for the assignment I , is the ratio of the number of true events to the total number of events in $\mathbf{E}_{\mathcal{S},I}$. Hence $P_H = \psi(\mathbf{E}_{\mathcal{S},I})$. Similarly, $P_F = \psi(\mathbf{E}_{\mathcal{S}',I'}) = a/b$, $P_G = \psi(\mathbf{E}_{\mathcal{S}'',I'') = c/d$. The number of true events in $\mathbf{E}_{\mathcal{S},I}$ is $(acm + a(d-c)m + (b-a)cm + (b-a)(d-c)(n-m))$ and $|\mathbf{E}_{\mathcal{S},I}| = (bdn)$ (from Observation 3.3.0.4 in Section 3.3). Hence,

$$\begin{aligned} \psi(\mathbf{E}_{\mathcal{S},I}) &= \frac{(acm + a(d-c)m + (b-a)cm + (b-a)(d-c)(n-m))}{bdn} && \text{or} \\ P_H &= \psi(\mathbf{E}_{\mathcal{S},I}) = (P_F)(P_G)p + (1 - P_F)(P_G)p + (P_F)(1 - P_G)p \\ &\quad + (1 - P_F)(1 - P_G)(1 - p) \end{aligned}$$

□

Note: Again, we note that there might exist an assignment I , such that a majority of events in $\mathbf{E}_{\mathcal{S},I}$ may be false events (and hence $P_H < 1/2$). In this context, it is more natural to reason about the validity of the sentence $\bar{\mathcal{S}}_I$ which denotes $H_I \stackrel{\bar{r}}{=} 0$, and the probability with which H_I is unsatisfied rather than P_H , the probability with which it is satisfied. However, since Observation 3.2.2.1 is only a combinatorial relation between the event sets of $\mathcal{S}'_I, \mathcal{S}''_{I''}$, the probability parameter p , and the event set of \mathcal{S}_I , we have derived a relation using the function ψ . In combinatorial arguments such as in Observation 3.2.2.1, it is sufficient to use the function ψ without having to explicitly invoke $\bar{\psi}$ keeping in mind that for any event set \mathbf{E} , $\psi(\mathbf{E}) = (1 - \bar{\psi}(\mathbf{E}))$.

Akin to Observation 3.2.2.1, similar relationships between the event sets can be established for PBF of the form $H = (F \wedge_p G)$ and $H = \neg F$ as follows:

Observation 3.2.2.2 *If H denotes $(F \wedge_p G)$, $|\mathbf{E}_{\mathcal{S},I}| = (bdn)$ where $acm + (b-a)c(n-m) + (b-a)(d-c)(n-m) + (a)(d-c)(n-m)$ events in $\mathbf{E}_{\mathcal{S},I}$ are correct events. Furthermore, with $P_F = \psi(\mathbf{E}_{\mathcal{S}',I'}) = a/b$ and $P_G = \psi(\mathbf{E}_{\mathcal{S}'',I'') = c/d$, $P_H = \psi(\mathbf{E}_{\mathcal{S},I}) = (P_F)(P_G)p + (1 - P_F)(P_G)(1 - p) + (P_F)(1 - P_G)(1 - p) + (1 - P_F)(1 - P_G)(1 - p)$.*

Observation 3.2.2.3 *If H denotes $(\neg_p F)$, $|\mathbf{E}_{\mathcal{S},I}| = bn$ where $a(n-m) + (b-a)(m)$ events in $\mathbf{E}_{\mathcal{S},I}$ are correct events. Furthermore with $P_F = \psi(\mathbf{E}_{\mathcal{S}',I'}) = a/b$, $P_H = \psi(\mathbf{E}_{\mathcal{S},I}) = (P_F)(1-p) + (1-P_F)p$.*

3.2.2.1 Equivalence of PBF Through Event Sets

Consider two formulae H and H' where $\text{VAR}_H \subseteq \text{VAR}_{H'}$ (or vice-versa). Then H and H' are equivalent under the assignment I (where I' is the corresponding consistent assignment) if and only if

$$\psi(\mathbf{E}_{\mathcal{S},I}) = \psi(\hat{\mathbf{E}}_{\mathcal{S}',I'})$$

Finally PBF H and H' are equivalent, denoted $H \equiv H'$, if they are equivalent for every assignment $I \in \mathbf{I}$ (we claim without proof that the individual event sets $\mathbf{E}_{\mathcal{S},I}$ for a sentence \mathcal{S} and its input $I \in \mathbf{I}$ can be combined across all the inputs to yield a single finite representation common to all inputs).

3.3 A Formal Model for Probabilistic Boolean Logic

We now present a formal model for the language of probabilistic Boolean logic. Let \mathcal{L} denote the *language* of PBL, which is a set of well formed *sentences* in PBL. The *signature* of \mathcal{L} consists of

- A countable set VAR of variables.
- A countable set P of probability parameters.
- The connectives $\vee_p, \wedge_{p'}, \neg_{p''}$ where $p, p', p'' \in P$.
- The punctuation symbols (and).
- The set of constants $\{c^0, c^1\}$.
- Denumerable set of predicate letters $\stackrel{r}{=}$ where $r \in P$.

Any well formed *sentence* $\mathcal{S}[I]$ in this language is of the form $F_I \stackrel{r}{=} c^1$ or $F_I \stackrel{\bar{r}}{=} c^0$ where F is a well formed PBF, $r, \bar{r} \in P$, and I is an assignment which assigns one of $\{c^0, c^1\}$ to any variable $x \in \text{VAR}_F \subseteq \text{VAR}$.

The model \mathbf{M} for this language consists of

- The punctuation symbols (and).
- The set $\mathbb{N} = \{0, 1, 2, \dots\}$, of natural numbers.
- The set $C = \{0, 1\}$ of Boolean constants.
- A set \mathbb{B} of valid closed sentences from classical Boolean logic of the form $B = 1$ or $B = 0$, where B is a closed well formed formula in Boolean logic. Conventionally, the former sentences will be called *true* sentences and the latter are called *false* sentences.
- the set \mathbb{Q} , of non-negative rationals.
- A set \mathbb{E} where any $\mathbf{E}_{\mathcal{S}, I} \in \mathbb{E}$ is referred to as an *event set* where $\mathbf{E} \subseteq \mathbb{N} \times \mathbb{B}$, and any $(i, \mathcal{B}) \in \mathbf{E}_{\mathcal{S}, I}$ will be called an *event* (the index $i \in \mathbb{N}$ and Boolean sentence $\mathcal{B} \in \mathbb{B}$). Furthermore, if the classical Boolean sentence \mathcal{B} is true, the event (i, \mathcal{B}) will be referred to as a *true event*; it is a *false event* otherwise.
- Let \mathcal{S}_I denote $H_I \stackrel{r}{=} \hat{c}$ where H is a well formed PBF and $\hat{c} \in \{c^0, c^1\}$. If H is of length 0, H is of the form (x) where x is a Boolean variable. For the assignment I which denotes $\langle x = c^1 \rangle$, $\mathbf{E}_{\mathcal{S}, I}$ consists of one event of the form $(0, (1) = 1)$. Similarly for the assignment \hat{I} which denotes $\langle x = c^0 \rangle$, $\mathbf{E}_{\mathcal{S}, \hat{I}}$ consists of one event of the form $(0, (0) = 0)$.

Let H be a PBF of length $k \geq 1$, and let H be of the form $(F \vee_p G)$ where F and G are PBF of length $k - 1$ or less. For an assignment I to H and the corresponding consistent assignments I', I'' to F and G respectively, let $\mathcal{S}'_{I'}, \mathcal{S}''_{I''}$

respectively denote $F_{I'} \stackrel{r'}{=} c'$ and $G_{I''} \stackrel{r''}{=} c''$, $c', c'' \in \{c^0, c^1\}$. Let $\mathbf{E}_{\mathcal{S}', I'}$, $\mathbf{E}_{\mathcal{S}'', I''}$ be the event sets of $\mathcal{S}'_{I'}$ and $\mathcal{S}''_{I''}$ respectively. Let $p^M = m/n$ where m, n are relatively prime and $\tilde{\mathbf{E}} = (\mathbf{E}_{\mathcal{S}', I'} \times \mathbf{E}_{\mathcal{S}'', I''})$. For any $((i, \mathcal{B}'), (j, \mathcal{B}'')) \in \tilde{\mathbf{E}}$ let \mathcal{B}' denote $B' = t'$ and let \mathcal{B}'' denote $B'' = t''$, where B', B'' are well formed closed Boolean formulae and $t', t'' \in \{0, 1\}$. Let the number of true events in $\mathbf{E}_{\mathcal{S}', I'}$ be denoted by the symbol a , $|\mathbf{E}_{\mathcal{S}', I'}| = b$. Similarly, the number of true events in $\mathbf{E}_{\mathcal{S}'', I''}$ is c and $|\mathbf{E}_{\mathcal{S}'', I''}| = d$. Then,

$$\hat{\mathbf{E}}_{\mathcal{S}, I} = \{ \text{for } 0 \leq k < m, (f, (B' \vee B'') = T(t' \vee t'')) : ((i, \mathcal{B}'), (j, \mathcal{B}'')) \in \tilde{\mathbf{E}} \} \quad (1)$$

$$\text{where } f = (di + j) * n + k$$

$$\hat{\tilde{\mathbf{E}}}_{\mathcal{S}, I} = \{ \text{for } m \leq k < n, (g, (B' \vee B'') = T(\neg(t' \vee t''))) : ((i, \mathcal{B}'), (j, \mathcal{B}'')) \in \tilde{\mathbf{E}} \}$$

$$\text{where } g = (di + j) * n + k$$

$$\mathbf{E}_{\mathcal{S}, I} = \hat{\mathbf{E}}_{\mathcal{S}, I} \cup \hat{\tilde{\mathbf{E}}}_{\mathcal{S}, I}$$

- A function $\psi : \mathbb{E} \rightarrow \mathbb{Q}$ such that $\psi(\mathbf{E}_{\mathcal{S}, I})$ is the ratio of the number of true events in $\mathbf{E}_{\mathcal{S}, I}$ to $|\mathbf{E}_{\mathcal{S}, I}|$. A function $\bar{\psi} : \mathbb{E} \rightarrow \mathbb{Q}$ where $\bar{\psi}(\mathbf{E}_{\mathcal{S}, I})$ is the ratio of the number of false events in $\mathbf{E}_{\mathcal{S}, I}$ to $|\mathbf{E}_{\mathcal{S}, I}|$.
- A relationship $R \subseteq C \times \mathbb{Q} \times \mathbb{E}$ where $(1, r, \mathbf{E}_{\mathcal{S}, I}) \in R$ if and only if $\psi(\mathbf{E}_{\mathcal{S}, I}) = r$ and $(0, \bar{r}, \mathbf{E}_{\mathcal{S}, I}) \in R$ if and only if $\bar{\psi}(\mathbf{E}_{\mathcal{S}, I}) = \bar{r}$.

Observation 3.3.0.4 *Under the assignment I , $|\mathbf{E}_{\mathcal{S}, I}| = bdn$ where the number of true events in $\mathbf{E}_{\mathcal{S}, I}$ is $(acm + a(d - c)m + (b - a)cm + (b - a)(d - c)(n - m))$.*

PROOF. We recall that the number of true events in $\mathbf{E}_{\mathcal{S}', I'}$ is a , $|\mathbf{E}_{\mathcal{S}', I'}| = b$, the number of true events in $\mathbf{E}_{\mathcal{S}'', I''}$ is c and $|\mathbf{E}_{\mathcal{S}'', I''}| = d$. We know that $T(1 \vee 0) = T(1 \vee 1) = T(0 \vee 1) = 1$. From this, and from (1), $(ad + (b - a)c)m$ events in $\hat{\mathbf{E}}_{\mathcal{S}, I}$ are true events. Furthermore $T(\neg(0 \vee 0)) = 1$, and hence from (2), $(b - a)(d - c)(n - m)$

events in $\hat{\mathbf{E}}_{\mathcal{S},I}$ are true events. Hence the number of true events in $\mathbf{E}_{\mathcal{S},I}$ is $(ad + (b - a)c)m + (b - a)(d - c)(n - m) = (acm + a(d - c)m + (b - a)cm + (b - a)(d - c)(n - m))$. Furthermore, from (1), the number of events in $\hat{\mathbf{E}}_{\mathcal{S},I}$ is bdm and from (2), the number of events in $\hat{\mathbf{E}}_{\mathcal{S},I}$ is $bd(n - m)$. Hence the total number of events in $\mathbf{E}_{\mathcal{S},I}$ is $bdm + bd(n - m) = (bdn)$. \square

Given any well formed sentence $\mathcal{S}[I] \in \mathcal{L}$ of the form $F_I \stackrel{r}{=} c$, the *interpretation* of the sentence $\mathcal{S}[I]$ in the model \mathbf{M} , maps

- The constants c^0 to 0, c^1 to 1, c to $c^{\mathbf{M}} \in \{0, 1\}$.
- The probability parameters p, q, \dots to $p^{\mathbf{M}}, q^{\mathbf{M}}, \dots \in \mathbb{Q}$ where $1/2 \leq p^{\mathbf{M}}, q^{\mathbf{M}}, \dots \leq 1$.
- The probability parameter r of the predicate symbol to $r^{\mathbf{M}} \in Q$ such that $0 \leq r^{\mathbf{M}} \leq 1$.
- The sentence $\mathcal{S}[I]$ to an event set $\mathbf{E}_{\mathcal{S},I}$.
- The sentence $\mathcal{S}[I]$ is *valid* under this interpretation if and only if $(c^{\mathbf{M}}, r^{\mathbf{M}}, \mathbf{E}_{\mathcal{S},I}) \in R$.

As an example consider a sentence $\mathcal{S}[I] \in \mathcal{L}$ of the form $(x \vee_p y) \stackrel{r}{=} c^1$ where the assignment I denotes $\langle x = c^0, y = c^1 \rangle$. Then under the interpretation \mathbf{M} , c^0 is mapped to 0, c^1 to 1, p to some $p^{\mathbf{M}} \in \mathbb{Q}$, where $1/2 \leq p^{\mathbf{M}} \leq 1$ and r to $r^{\mathbf{M}} \in Q$ such that $0 \leq r^{\mathbf{M}} \leq 1$. Let $p^{\mathbf{M}} = m/n$ for positive, relatively prime integers m, n . Then the number of true events in the event set $\mathbf{E}_{\mathcal{S},I}$ of $\mathcal{S}[I]$ is m and these elements are $(0, (0 \vee 1) = 1), (1, (0 \vee 1) = 1), \dots, (m - 1, (0 \vee 1) = 1)$ and the number of false events in $\mathbf{E}_{\mathcal{S},I}$ is $(n - m)$ and these events are $(m, \neg(0 \vee 1) = 0), (m + 1, \neg(0 \vee 1) = 0), \dots, (n - 1, \neg(0 \vee 1) = 0)$. The sentence $\mathcal{S}[I]$ is valid under this interpretation if and only if $(1, r^{\mathbf{M}}, \mathbf{E}_{\mathcal{S},I}) \in R$, or equivalently, if and only if $\psi(\mathbf{E}_{\mathcal{S},I}) = r^{\mathbf{M}}$.

Similarly if H is of the form $(F \wedge_p G)$ and as before $p^M = m/n$,

$$\hat{\mathbf{E}}_{\mathcal{S},I} = \{ \text{for } 0 \leq k < m, (f, (B' \wedge B'') = T(t' \wedge t'')) : ((i, \mathcal{B}'), (j, \mathcal{B}'')) \in \tilde{\mathbf{E}} \}$$

$$\text{where } f = (di + j) * n + k$$

$$\hat{\hat{\mathbf{E}}}_{\mathcal{S},I} = \{ \text{for } m \leq k < n, (g, (B' \wedge B'') = T(\neg(t' \wedge t''))) : ((i, \mathcal{B}'), (j, \mathcal{B}'')) \in \tilde{\mathbf{E}} \}$$

$$\text{where } g = (di + j) * n + k$$

$$\mathbf{E}_{\mathcal{S},I} = \hat{\mathbf{E}}_{\mathcal{S},I} \cup \hat{\hat{\mathbf{E}}}_{\mathcal{S},I}$$

Similarly if H is of the form $\neg_p(F)$,

$$\hat{\mathbf{E}}_{\mathcal{S},I} = \{ \text{for } 0 \leq k < m, (i * n + k, \neg(B') = T(\neg(t'))) : (i, (B' = t')) \in \mathbf{E}_{\mathcal{S}',I'} \}$$

$$\hat{\hat{\mathbf{E}}}_{\mathcal{S},I} = \{ \text{for } m \leq k < n, (i * n + k, (B' = t')) : (i, (B' = t')) \in \mathbf{E}_{\mathcal{S}',I'} \}$$

$$\mathbf{E}_{\mathcal{S},I} = \hat{\mathbf{E}}_{\mathcal{S},I} \cup \hat{\hat{\mathbf{E}}}_{\mathcal{S},I}$$

CHAPTER IV

PROPERTIES OF PROBABILISTIC BOOLEAN LOGIC

Through the construct of event sets and the accompanying notion of equivalence of PBF, we will now characterize some identities of PBL in Section 4.1. Specifically, we show that several of the identities of conventional Boolean logic, such as commutativity, are preserved in PBL. Also, identities such as that introduced by DeMorgan [194], which relate pairs of dual logical operators— \vee and \wedge in conventional Boolean logic for example—are preserved in a suitably modified manner as described below. Properties such as distributivity and associativity are *not* preserved. We will use the letters, p, q, r, a, b, c to denote probabilities where as before, $1/2 \leq p, q, r, a, b, c \leq 1$ and $p, q, r, a, b, c \in \mathbb{Q}$.

4.1 *Classical Identities That are Preserved*

We have enumerated the significant identities of PBL in Table 4.1. As an illustrative example, let us consider commutativity (identity (1) in Table 4.1). Now, consider F and G which denote $(x \vee_p y)$ and $(y \vee_p x)$ respectively, where $p = m/n$. For any assignment I , in particular $\langle x = 1, y = 0 \rangle$, let $\mathbf{E}_{F,I}$ be the event set of F . In $\mathbf{E}_{F,I}$, m events are associated with $(1 \vee 0) = 1$ and hence associated with $(0 \vee 1) = 1$ since $(1 \vee 0) \equiv (0 \vee 1)$ in classical Boolean logic. Similarly, $n - m$ events in $\mathbf{E}_{F,I}$ are associated with the $\neg(1 \vee 0) = 1$ and hence $\neg(0 \vee 1) = 1$. Similarly for each possible input assignment $I \in \{\langle x = 0, y = 0 \rangle, \langle x = 0, y = 1 \rangle, \langle x = 1, y = 0 \rangle, \langle x = 1, y = 1 \rangle\}$. Hence, from the definition of equivalence of PBF, $(x \vee_p y) \equiv (y \vee_p x)$, or the operator \vee_p is commutative¹.

¹A straight forward induction will allow us to extend this to PBF of arbitrary length.

1. Commutativity	$(x \vee_p y) \equiv (y \vee_p x)$ $(x \wedge_p y) \equiv (y \wedge_p x)$
2. Double Complementation	$\neg_q(\neg_p x) \equiv \neg_p(\neg_q x)$ $\neg_p 0 \equiv \neg_1(\neg_p 1)$ $\neg_p 1 \equiv \neg_1(\neg_p 0)$
3. Operations with 0 and 1	$(0 \wedge_p x) \equiv (\neg_p 1)$ $(1 \wedge_p x) \equiv \neg_1(\neg_p x)$ $(0 \vee_p x) \equiv \neg_1(\neg_p x)$ $(1 \vee_p x) \equiv (\neg_p 0)$
4. Identity	$(x \vee_p x) \equiv \neg_1(\neg_p x)$ $(x \wedge_p x) \equiv \neg_1(\neg_p x)$
5. Probabilistic Tautology	$(x \vee_p (\neg_1 x)) \equiv \neg_p 0$ $(x \wedge_p (\neg_1 x)) \equiv \neg_p 1$
6. Probabilistic DeMorgan Identity	$\neg_p(x \vee_q y) \equiv (\neg_1 x) \wedge_r (\neg_1 y)$ $\neg_p(x \wedge_q y) \equiv (\neg_1 x) \vee_r (\neg_1 y)$ where $r = pq + (1 - p)(1 - q)$

Table 1: Identities of PBL

4.2 Identities that are not Preserved

Surprisingly, not all properties from conventional Boolean logic can be extended to the probabilistic case. In particular, associativity, distributivity and absorption as stated in Boolean logic are not preserved in PBL.

4.2.1 Associativity

Let F and G denote $(x \vee_p (y \vee_p z))$ and $((x \vee_p y) \vee_p z)$ respectively, where $\text{VAR} = \{x, y, z\}$ is the set of variables in F as well as in G .

Theorem 3 *There exists an assignment I to VAR such that $\psi(\mathbf{E}_{F,I}) \neq \psi(\mathbf{E}_{G,I})$ and therefore $F \not\equiv G$. Hence PBL is not associative.*

PROOF. Consider the assignment I which denotes $\langle x = 1, y = 0, z = 0 \rangle$. If $\mathbf{E}_{F,I}$ and $\mathbf{E}_{G,I}$ are the event sets of F_I and G_I respectively, it follows from the definition of event sets, that $\psi(\mathbf{E}_{F,I}) = p^2$ whereas $\psi(\mathbf{E}_{G,I}) = p^2 + (1 - p)^2$ (from Observation 3.2.2.1). Hence there exist values of p , $1/2 \leq p \leq 1$ such that $\mathbf{E}_{F,I} \not\equiv \mathbf{E}_{G,I}$, and therefore $F \not\equiv G$. \square

4.2.2 Distributivity

Consider as a natural extension of distributivity in the PBL context, expressed as

$$(x \vee_p (y \wedge_q z)) \equiv ((x \vee_a y) \wedge_b (x \vee_c z))$$

We shall now show that this identity does not hold for PBL.

Theorem 4 *There exist p, q , $1/2 < p, q < 1$ such that $(x \vee_p (y \wedge_q z)) \not\equiv ((x \vee_a y) \wedge_b (x \vee_c z))$ for any $1/2 \leq a, b, c \leq 1$, and therefore \vee_p does not distribute over \wedge_q .*

PROOF. Without loss of generality, let F represent $(F' \vee_p F'')$ where F', F'' respectively denote (x) , $(y \wedge_q z)$, and G denotes the formula $((x \vee_a y) \wedge_b (x \vee_c z))$. In particular, let $1/2 < p, q < 1$. Also, let I, J , the input assignments to F , represent $\langle x = 1, y = 0, z = 0 \rangle, \langle x = 0, y = 1, z = 1 \rangle$ respectively where I'', J'' are the corresponding consistent assignments to F'' .

We will first show that $\psi(\mathbf{E}_{F,I}) \neq \psi(\mathbf{E}_{F,J})$. Suppose $\psi(\mathbf{E}_{F,I}) = \psi(\mathbf{E}_{F,J})$. Since $\langle x = 1 \rangle$ in I , from the definition of probabilistic disjunction operator, $\psi(\mathbf{E}_{F,I}) = p$. Furthermore, since $\langle y = 1, z = 1 \rangle$ in J , from the definition of the probabilistic conjunction operator, $\psi(\mathbf{E}_{F'',J''}) = q$ and from Observation 3.2.2.1, $\psi(\mathbf{E}_{F,J}) = pq + (1 - p)(1 - q)$. Since, $\psi(\mathbf{E}_{F,J}) = \psi(\mathbf{E}_{F,I})$,

$$\begin{aligned} pq + (1 - p)(1 - q) &= p & \text{or} \\ (1 - 2p)(1 - q) &= 0 \end{aligned}$$

Then, $(1 - 2p) = 0$ or $(1 - q) = 0$ or both, which contradicts the fact that $1/2 < p, q < 1$.

Now, let $F \equiv G$. Then from the definition of equivalence of PBF, it must be the case that $\psi(\mathbf{E}_{F,I}) = \psi(\mathbf{E}_{G,I})$ and $\psi(\mathbf{E}_{F,J}) = \psi(\mathbf{E}_{G,J})$. Furthermore, we have shown that $\psi(\mathbf{E}_{F,I}) \neq \psi(\mathbf{E}_{F,J})$ and hence $\psi(\mathbf{E}_{G,I}) \neq \psi(\mathbf{E}_{G,J})$.

For the assignments I and J , and from the definition of a probabilistic disjunction and Observation 3.2.2.2,

$$\psi(\mathbf{E}_{G,I}) = \psi(\mathbf{E}_{G,J}) = 1 - b - ac + 2abc$$

which is a contradiction \square

4.3 Degree of Non-Associativity

We know from Section 4.2 and Theorem 4 that formulae in PBL are not associative. We will now quantify the *degree* to which a PBF is non-associative. Besides inherent intellectual interest, such a characterization is of interest from a pragmatic perspective, since tools for synthesizing logic circuits from formulaic specifications (logic synthesis tools), use “reassociation” as a ubiquitous transformation for optimizing digital logic circuits [122]. This transformation is legal or valid in the Boolean logic context, since associativity is truth preserving. Typically, this transformation is applied to improve the performance (time) while preserving the cost (size) of a Boolean circuit. In contrast to Boolean logic, in the case of PBL, a reassociation can result in a significant change to the probability with which the formula is satisfied, depending on the input assignment. As a simple example, consider Figure 9(a), where we illustrate a PBF F and its reassociation F' in Figure 9(c). For those who are computationally minded, F and F' are depicted as trees, explicitly indicating the order in which their constituent operators would be evaluated. Continuing, for an input assignment $\langle x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1 \rangle$, it is easy to verify that the probability that F is satisfied is p whereas the probability that F' is satisfied is $p^2 + p^2(1 - p) + (1 - p)^3$; very different probability values for, $1/2 < p < 1$.

More generally, let \mathbf{F} be a maximal set of formulae where $F, F' \in \mathbf{F}$ if and only if they are reassociations of each other. For $F, F' \in \mathbf{F}$ and for a particular input

assignment² I to F as well as to F' , let the probabilities that F_I and F'_I are unsatisfied be q_I and q'_I respectively. If \mathbf{I} is the set of all input assignments to F (and F'), we can quantify the *amount* by which F and F' are non-associative as,

$$NA(F, F') = \max_{\mathbf{I} \in \mathbf{I}} \left\{ \frac{q'_I}{q_I}, \frac{q_I}{q'_I} \right\} \quad (3)$$

Building on this, we can quantify the non-associativity of the set \mathbf{F} to be

$$\eta_{\mathbf{F}} = \max_{\mathbf{F} \in \mathcal{F}} \{NA(F, F')\} \quad (4)$$

The *degree of non-associativity* of PBL with formulae of length no greater than n , Δ_n is

$$\Delta_n = \max_{\mathbf{F} \in \mathcal{F}_n} \{\eta_{\mathbf{F}}\} \quad (5)$$

where $\mathbf{F} \in \mathcal{F}_n$ if and only if the length of F is at most n for any $F \in \mathbf{F}$.

4.3.1 Balanced Binary and Linear Probabilistic Boolean Formula

We will now consider two associations of the same base formula F , a “linear” formula L (Figure 9(a)) and a “balanced binary” formula B (Figure 9(c)). In order to bound Δ_n from below, we will bound the probability Q_L that L is not satisfied from below, and the probability Q_B that B is not satisfied from above. Then we will use the fact that $\Delta_n \geq Q_L/Q_B$.

Consider n PBF, $C^1, C^2, C^3, \dots, C^n$ where $C^i = (x_i)$ and without loss of generality let $n = 2^m$ for some positive integer m . For $1 \leq i \leq n/2$, H^i is $(C^{2i-1} \vee_p C^{2i})$ and for $n/2 \leq i \leq n-1$, H^i is of the form $(H^j \vee_p H^{j+1})$ where $j = (2i - n - 1)$. For example, with four variables $\{x_1, x_2, x_3, x_4\}$, C^1, C^2, C^3, C^4 would be $(x_1), (x_2), (x_3), (x_4)$ respectively H^1 would denote $(x_1 \vee_p x_2)$, H^2 would denote $(x_3 \vee_p x_4)$, and H^3 or B

²Since F and F' are defined on (exactly) the same set of Boolean variables, the same assignment I is valid in both cases.

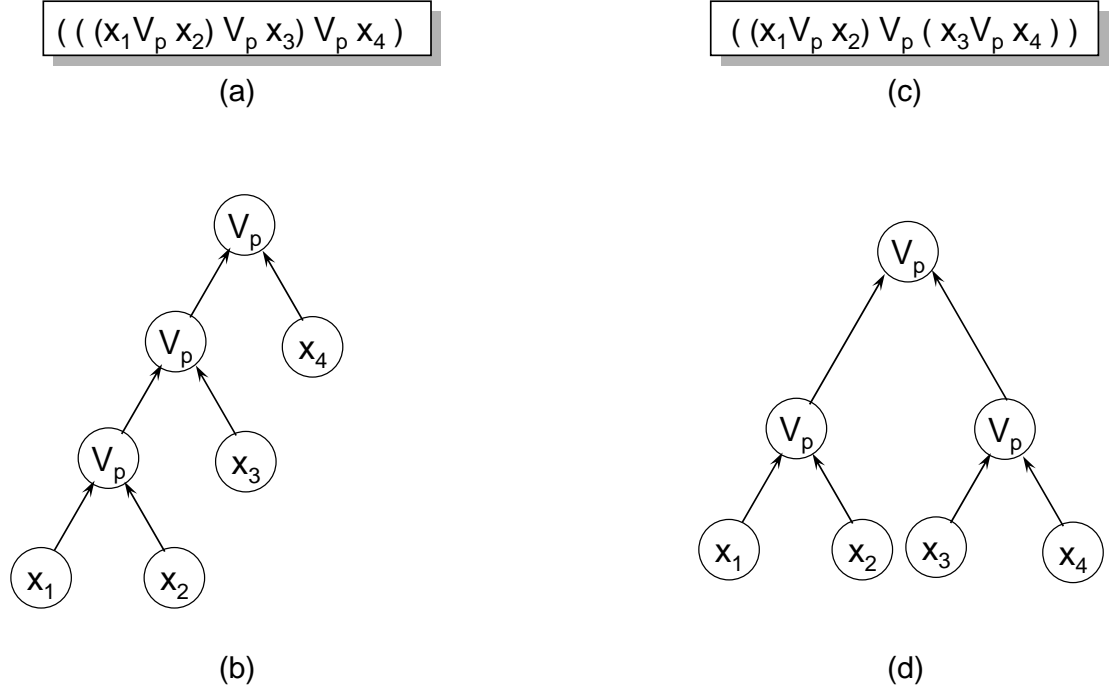


Figure 9: (a) A linear PBF over n variables in syntactic form (b) as a tree structure illustrating the linear form (c) a reassociation of the same PBF (d) its balanced binary representation in tree form

would be $(H^1 \vee_p H^2)$ which is $((x_1 \vee_p x_2) \vee_p (x_3 \vee_p x_4))$ as shown in Figure 9(c),(d). Thus, PBF B is of length 3 and height 2. For convenience, B denotes H^{n-1} . We shall refer to B as a *balanced binary* probabilistic Boolean formula of height m and length $(n - 1)$, since, as illustrated in Figure 9(d), B is a balanced binary tree.

For the same set of n variables, we can construct the probabilistic Boolean formula L , a reassociation of B , as follows: For some $1/2 \leq p < \leq 1$ (and $q = (1 - p)$ as before) construct the probabilistic Boolean formula L where L is defined as follows: $G^2 = (C^1 \vee_p C^2)$ and for $2 < i \leq n$, $G^i = (G^{i-1} \vee_p C^i)$ and for notational convenience, we will use the symbol L to represent G^n , where L is a *linear* probabilistic Boolean formula of length $(n - 1)$, since topologically L is a linear structure with $(n - 1)$ probabilistic disjunction operators (as in Figure 9(b)).

We will now state a useful fact to be used subsequently in multiple contexts as we estimate Δ_n .

Lemma 4.3.1 *Given any PBF F of the form $(F' \vee_p F'')$ with an assignment I and corresponding consistent assignments I', I'' to F' and F'' , if $Q_F, Q_{F'}$ and $Q_{F''}$ are the probabilities that they are unsatisfied, $Q_F = q + Q_{F'}Q_{F''}(1 - 2q)$.*

PROOF. This Lemma follows from the fact that $P_F = (1 - Q_F)$, $P_{F'} = (1 - Q_{F'})$ and $P_{F''} = (1 - Q_{F''})$, and therefore from Observation 3.2.2.1,

$$\begin{aligned} Q_F &= (q)(1 - Q_{F'})(1 - Q_{F''}) + (q)(1 - Q_{F'})(Q_{F''}) \\ &\quad + (q)(Q_{F'})(1 - Q_{F''}) + (1 - q)(Q_{F'})(Q_{F''}) \end{aligned} \quad (6)$$

$$= q + Q_{F'}Q_{F''}(1 - 2q) \quad (7)$$

□

Now, consider a balanced binary PBF of length n and consider H_i of length k where $n/2 \leq i \leq n - 1$. From the definition of a Balanced binary PBF, H^i is of the form $(H^j \vee_p H^k)$, where $j = (2i - n - 1), k = (2i - n)$. If H^j is satisfied with a probability P_j , we observe from Lemma 4.3.1 that

Observation 4.3.1.1 *The probability with which H^i is satisfied is at least pP_j*

4.3.2 An Upper bound on the Probability of Unsatisfiability of a Balanced Binary Probabilistic Boolean Formula

Lemma 4.3.2 *Let Q_B be the probability that a PBF B of length $(n - 1)$, where $n = 2^k$ for some integer $k \geq 2$, is unsatisfied with an input assignment α , where $\alpha(x) = 1$ for all $x \in \text{VAR}_B$. Then $Q_B < \sum_{i=1}^{\log(n)} q^i$ with $q = (1 - p)$.*

PROOF. We will prove the lemma by induction on the length of B . For the basis, consider a balanced binary PBF \hat{B} of length $2^2 - 1 = 3$ with 4 variables, where $\hat{B} = (\hat{B}' \vee_p \hat{B}'')$. Now consider an input assignment $\hat{\alpha}(x_i) = 1$ for $1 \leq i \leq 4$. Since B' and B'' are identical in a balanced binary PBF, we have $Q_{\hat{B}'} = Q_{\hat{B}''} = q$ and therefore from Lemma 4.3.1,

$$Q_B = q + q^2(1 - 2q)$$

and since $q > 0$

$$Q_B < q + q^2 \tag{8}$$

Now Consider B of the form $(B' \vee_p B'')$, where B' and B'' are balanced binary PBF of length $(2^{k-1} - 1)$, $k \geq 3$ and B is of length $(2^k - 1)$. By definition of α , an identical value (of 1) is assigned to all the variables of B' and B'' and $Q_{B'} = Q_{B''}$. As an induction hypothesis, let $Q_{B'} = Q_{B''} < \sum_{i=1}^{k-1} q^i$. From this hypothesis and Lemma 4.3.1, we have

$$\begin{aligned} Q_B &\leq q + \left(\sum_{i=1}^{k-1} q^i \right) \left(\sum_{i=1}^{k-1} q^i \right) (1 - q) = q + \left(\sum_{i=1}^{k-1} q^i \right) (q - q^k) \\ &\quad \text{hence} \\ Q_B &< \sum_{i=1}^k q^i \quad \text{for } q > 0 \end{aligned}$$

With $k = \log(n)$, we have the proof. \square

Building on this lemma, we will now determine an upper-bound on the probability Q_B that a (balanced binary) PBF is not satisfied, when a constant fraction $\lambda = n\epsilon$ for $0 < \epsilon < 1$ of its variables are assigned a value of 1 (and the rest are assigned a value of 0) through an assignment³ α . We will continue to consider the case where all of the probabilistic disjunction operators have the same associated probability parameter p where $n \geq 4$.

Theorem 5 *Let Q_B be the probability that a balanced binary PBF B of length $n - 1$ is unsatisfied for an assignment α , where $\alpha(x_i) = 1$ for $0 < i \leq \lambda$, $\alpha(x_i) = 0$ for*

³The symbol α is reused with varying constraints throughout this chapter, which entails some abuse of notation

$\lambda < i \leq n$, and $q \log(n/\lambda) \leq 1$. Then, $Q_B < (1 + \log(\frac{n}{\lambda}))q$ for $n \geq 4$ whenever $n = 2^k$, $\lambda = 2^l$ for $l < k$.

PROOF. Let B be a balanced binary PBF of length $n \geq 4$. Consider an assignment α such that $\alpha(x_i) = 1$ for $0 < i \leq \lambda$, and $\alpha(x_i) = 0$ for $\lambda < i \leq n$. Consider the sub-formula H^m of B , with variables $\text{VAR}_{H^m} = \{x_1, x_2, x_3, \dots, x_\lambda\}$. Since $\lambda = 2^l$, from the definition of a balanced binary PBF, H^m is a balanced binary PBF and $m = (n+1-2n/\lambda)$. Let P_m be the probability that H^m is satisfied for the assignment α .

Since $\lambda \leq n/2$, there exists a sub formula H^o of B , which is of length $2\lambda - 1$, such that $H^o = (H^m \vee_p H^{m+1})$ and $o = (n+1-n/\lambda)$. The probability that H^o is satisfied (from Observation 4.3.1.1) is at least pP_m . Continuing, a straight forward induction will show that P_B , the probability that $B = H^{n-1}$ is satisfied, is (at least) $p^{\log(n/\lambda)}P_m$.

If Q_m is the probability that H^m is unsatisfied, from Lemma 4.3.2, $Q_m < \sum_{i=1}^{\log(\lambda)} q^i$. Since $P_m = 1 - Q_m$, $P_m > 1 - \sum_{i=1}^{\log(\lambda)} q^i = 1 - \frac{(q - q^{\log(\lambda)+1})}{(1-q)}$,

$$P_B > p^{\log(\frac{n}{\lambda})}P_m = (1-q)^s \left[1 - \frac{(q - q^t)}{(1-q)} \right] = (1-q)^s - (1-q)^{s-1}(q - q^t)$$

where $s = \log(n/\lambda)$ and $t = \log(\lambda) + 1$

$$(1-q)^s - (1-q)^{s-1}(q - q^t) > (1-q)^s - (1-q)^{s-1}(q)$$

since $0 < q < 1/2$, and therefore

$$P_B > (1-q)^{s-1}(1-2q) \quad (9)$$

We get $(1-q)^{s-1} = [\sum_{k=0}^{s-1} \binom{s-1}{k} (-q)^k]$ by using the binomial theorem⁴ to expand $(1-q)^{s-1}$. There are s terms in the expansion and where we refer to 1 as the first term, $(s-1)(-q)$ as the second term and so on. For convenience, the j^{th} term when $j > s$ will be taken to be 0. Since $\lambda \leq n/2$, $s \geq 1$, and whenever $1 \leq s \leq 2$,

⁴The interested reader is referred to [83] (page 86) where the binomial theorem is derived for $(a+b)^n$ where a, b are elements of a commutative ring and n is any positive integer

$(1 - q)^{s-1} = 1 - (s - 1)q$. Consider the case when $s > 2$, and let j be odd and $2 < j \leq s$, then the sum of j^{th} and $j + 1^{th}$ term of the binomial expansion of $(1 - q)^{s-1}$ is $u_j = \frac{(s-1)!q^{j-1}}{(j-1)!(s-j)!}(1 - (s - j)q/j)$. Since $sq \leq 1$, $u_j \geq 0$ and therefore $(1 - q)^{(s-1)} \geq (1 - (s - 1)q)$. Therefore, from (9),

$$P_B > (1 - (s - 1)q)(1 - 2q)$$

$$Q_B = 1 - P_B < (1 - (1 - (s - 1)q)(1 - 2q))$$

or

$$Q_B < (s + 1)q$$

and hence

$$Q_B < \left(1 + \log\left(\frac{n}{\lambda}\right)\right) q$$

□

We note in passing that due to symmetry, it is easy to see that the result derived in Theorem 5 holds even if the last λ variables are set to 1 and the rest of the variables to zero. In fact, it can be shown that the result derived in Theorem 5 holds irrespective of the position of the “runs” of variables assigned a value 1, due to the inherent symmetry in a balanced binary PBF.

4.3.3 A Lower bound on the Probability of Unsatisfiability of a Linear Probabilistic Boolean Formula

We will now consider the case of a linear PBF L . Recall that L is of the form $((x_1 \vee_p x_2) \vee_p x_3) \cdots \vee_p x_n$ where, again, the value 1 is assigned to x_i if and only if $1 \leq i \leq \lambda$ and the value 0 to x_i whenever $\lambda < i \leq n$.

Theorem 6 *Given a linear PBF L , of length $n - 1$, where Q_L is the probability that L is unsatisfied with the input assignment α where $\alpha(x_i) = 1$ if $1 \leq i \leq \lambda < n$ and 0*

otherwise,

$$Q_L \geq \max\{0, (n - \lambda + 1)q - (n - \lambda)(n - \lambda + 1)q^2\}$$

PROOF. Let $\lambda + k = n$. Since $\lambda < n$, it follows that $k \geq 1$. Consider the case when $k = 1$. Then the PBF L is of the form $(L' \vee_p x_{\lambda+1})$, where L' is a linear PBF of length $\lambda - 1$ with $\text{VAR}_{L'} = \{x_1, x_2, x_3, \dots, x_\lambda\}$. If Q_L is the probability that L is unsatisfied by the assignment α , using Lemma 4.3.1 and recalling that $x_i = 1$ for $1 \leq x_i \leq \lambda$ and 0 otherwise,

$$\begin{aligned} Q_L &= q + Q_{L'}(1 - 2q) \\ &= q + q(1 - 2q) \\ &= 2q - 2q^2 \end{aligned}$$

Hence the theorem is true whenever $k = 1$ (since $n - \lambda = k = 1$).

Let $k \geq 2$ and let us suppose that the theorem is false. Furthermore, let \hat{L} be the shortest sub-formula of L , for which the theorem is false and therefore $Q_{\hat{L}} < \max\{0, (\hat{k} + 1)q - (\hat{k} + 1)(\hat{k})q^2\}$. If the length of \hat{L} is $\lambda + \hat{k}$, where the set $\text{VAR}_{\hat{L}}$ of variables of \hat{L} is $\{x_1, x_2, x_3, \dots, x_{\lambda+\hat{k}+1}\}$, it must be the case that $\hat{k} > 1$ (since we have shown the theorem to be true for $\hat{k} = 1$). From the definition of a linear PBF, \hat{L} is of the form $(\hat{\hat{L}} \vee_p x_{\lambda+\hat{k}+1})$ where $\hat{\hat{L}}$ is of length $\lambda + \hat{k} - 1$. From the hypothesis, the theorem is true for $\hat{\hat{L}}$, or equivalently $Q_{\hat{\hat{L}}} \geq \max\{0, q + [(\hat{k})q - (\hat{k})(\hat{k} - 1)q^2]\}$. From Lemma 4.3.1, it follows that

$$Q_{\hat{L}} = q + Q_{\hat{\hat{L}}}(1 - 2q) \geq \max\{0, q + [(\hat{k})q - (\hat{k})(\hat{k} - 1)q^2](1 - 2q)\}$$

and hence

$$Q_{\hat{L}} \geq \max\{0, (\hat{k} + 1)q - (\hat{k} + 1)(\hat{k})q^2\}$$

A contradiction. \square

4.3.4 The Degree of Non-associativity of Probabilistic Boolean Logic

Theorem 7 *There exist two probabilistic Boolean formulae B and L , both of length $(n-1) \rightarrow \infty$ and $n \geq 4$ such that B is a reassociation of L and furthermore $NA(B, L)$ grows as $\Omega(n)$.*

PROOF. Consider $n = 2^m$, $m \geq 2$ variables $\{x_1, x_2, x_3, \dots, x_n\}$ where B and L are respectively the balanced binary Boolean formula and the linear probabilistic Boolean formula over this set of variables. From Theorem 5, for the assignment α and $1/2 \leq p < 1$ and $q = (1 - p)$, a λ exists such that

$$Q_B \leq \left(1 + \log\left(\frac{n}{\lambda}\right)\right) q \quad (10)$$

And furthermore, from Theorem 6 also for the same assignment α , and the value λ ,

$$Q_L \geq \max\{0, (n - \lambda + 1)q - (n - \lambda)(n - \lambda + 1)q^2\} \quad (11)$$

Consider

$$\begin{aligned} \mathbf{Q} &= \frac{(n - \lambda + 1)q - (n - \lambda)(n - \lambda + 1)q^2}{(1 + \log(\frac{n}{\lambda}))q} \\ &= \frac{(n - \lambda + 1) - (n - \lambda)(n - \lambda + 1)q}{(1 + \log(\frac{n}{\lambda}))} \quad \text{since } q \neq 0 \end{aligned}$$

For all $n \in \mathbb{N}^+$, $n \geq 4$, $q = \frac{1}{n^c}$ for $c \geq 2$, and $\lambda = n/2$,

$$\mathbf{Q} = \frac{n}{4} + \frac{1}{2} - \frac{1}{4n^{c-1}} - \frac{1}{8n^{c-2}} > 0$$

Recall from the definition of NA , the amount of non-associativity that

$$NA(B, L) = \max_{\forall I \in \mathbf{I}} \left\{ \frac{Q'_I}{Q''_I}, \frac{Q''_I}{Q'_I} \right\}$$

where Q'_I, Q''_I are respectively the probabilities that B and L are unsatisfied with an input assignment I . Whenever $\mathbf{Q} > 0$, it follows that

$$NA(B, L) \geq \frac{Q_L}{Q_B} \geq \mathbf{Q} = \frac{(n - \lambda + 1)q - (n - \lambda)(n - \lambda + 1)q^2}{(1 + \log(\frac{n}{\lambda}))q}$$

Therefore, for any $n \in \mathbb{N}^+$, $n \geq 4$, $q = \frac{1}{n^c}$ for $c \geq 2$, and $\lambda = n/2$,

$$NA(B, L) \geq \frac{n}{4} + \frac{1}{2} - \frac{1}{4n^{c-1}} - \frac{1}{8n^{c-2}} \geq \frac{n}{4} = \Omega(n)$$

□

Therefore, it immediately follows that

Corollary 8 *The degree of non-associativity, Δ_n of PBL grows as $\Omega(n)$*

CHAPTER V

PROBABILISTIC BOOLEAN LOGIC AND MODELS OF COMPUTING

We will now define models of computation based on PBL and distinguish the implicitly and explicitly realized probabilistic behaviors—the latter referred to as randomized for terminological clarity—using a measure based on the *energy consumed* in computing the result by a computational step. We will use the background from Section 5.0.6 to separate probabilistic and randomized (implicit and explicit) Boolean circuits. Building on this, in Section 5.0.7, we will extend this concept beyond combinational (Boolean) logic to a model of computation with state. Here we distinguish implicitly realized PA with PBL as a foundation, from their explicitly realized counterparts through explicit coin tosses, using the energy consumed by each state transition.

5.0.5 Thermodynamic Separation of Implicitly and Explicitly Probabilistic Gates and The Circuit Model of Computation

We will define *probabilistic Boolean circuits*, a model of computing, based on PBL and then distinguish them from their explicit counterpart, the randomized Boolean circuit with coin tosses.

5.0.5.1 PBF and Probabilistic Boolean Circuits

Analogous to conventional Boolean circuits, a *probabilistic Boolean circuit* is defined as follows: a directed acyclic connected graph $\hat{\mathbb{C}} = (\hat{V}, \hat{E})$, where \hat{V} is the set of vertices and \hat{E} the set of directed edges. The vertices are of three kinds. *Input* vertices, of in-degree 0 associated with Boolean variables (called input variables of the circuit) or Boolean constants $\{0, 1\}$, *internal* vertices associated with one of three

operators \vee_p, \wedge_q, \neg_r where $1/2 \leq p, q, r \leq 1$ and one distinguished output vertex of in-degree 1 and out-degree 0. Internal vertices associated \vee_p and \wedge_q have in-degree 2 and out-degree 1, whereas those associated with \neg_r have in-degree and out-degree 1. For any assignment of Boolean constants 0 or 1 to the input variables of the circuit, the value of the input vertex is either the Boolean constant assigned to the corresponding Boolean variable, or the Boolean constant directly associated with the vertex. The value of any internal vertex u , is the value obtained by applying the probabilistic Boolean operator associated with the vertex, to values associated with its input edges. The value of a directed edge $(u, v) \in \hat{E}$ is the value associated with the vertex u . Finally, the value *computed* by the probabilistic Boolean circuit is the value associated with the output vertex. If the cardinality of the set of input vertices is k , $\hat{\mathbb{C}}$ *computes* a probabilistic Boolean truth table \mathcal{T} with no more than 2^k rows.

Observation 5.0.5.1 *For any PBF F and the probabilistic truth table \mathcal{T} it represents, there exists a probabilistic Boolean circuit $\hat{\mathbb{C}}_F$ which computes \mathcal{T} .*

This observation is straightforward since a well formed PBF is obtained by the application of the rules outlined in Section 3.1. An equivalent probabilistic Boolean circuit can be constructed by creating input vertices for every Boolean variable and constant in the PBF, and an internal vertex for every Boolean operator.

5.0.5.2 *Randomized Boolean Circuits and Their Relationship to Probabilistic Boolean Circuits*

Randomized Boolean circuits have been used as a computational model to study randomized algorithms [1, 125]. Analogous to conventional Boolean circuits, a randomized Boolean circuit is a directed acyclic connected graph $\mathbb{C} = (V, E)$. As before, V can be partitioned into subsets, where the input vertices are associated with Boolean variables (called input variables of the circuit), Boolean constants or Boolean random variables. The internal vertices are associated with one of three operators or labels

\vee, \wedge, \neg from Boolean logic. Any internal vertex $v \in V$ has the property that there is at most one edge (u, v) such that $u \in V$ is an input vertex associated with a Boolean random variable. As before, there is one distinguished output vertex of in-degree 1 and out-degree 0. Notions of values associated with vertices and edges correspond to those introduced in Section 5.0.5.1 above.

Observation 5.0.5.2 *For probabilistic truth table \mathcal{T} , there exists a randomized Boolean circuit which computes it.*

We will now establish the fact that *any* randomized Boolean circuit (or more specifically its truth table) can be realized by a probabilistic Boolean circuit. Let $U \subseteq V$ denote input vertices associated with Boolean random variables in \mathbb{C} . Consider vertex $u \in U$ and a set of internal vertices V' such that whenever $v \in V'$, $(u, v) \in \mathbb{C}$. Let u be associated with Boolean random variable x_u such that probability that $x_u = 1$ is $p_u \in \mathbb{Q}$. The source of randomness in this case, which as part of an assignment binding values to the variables labeling the vertices in U , is explicit. By this, we mean that (informally) these bits are pseudo random and are produced by a suitable combination of deterministic gates. We formalize this as an “hypothesis” as follows.

HYPOTHESIS 1. *Each input bit bound to the random variable x_u where $u \in U$ is produced by a pseudo random source¹ constituted of gates all with a probability of correctness $p = 1$.*

We will predicate the development in the sequel on Hypothesis 1 being valid.

Returning to the goal of relating randomized Boolean circuits to its probabilistic counterpart, for any vertex $u \in \mathbb{C}$ as described above, let $p_u \geq 1/2$. We replace u with a new input vertex u'' associated with Boolean constant 0, a new internal vertex

¹There is a rich body of work, which seeks to address the cost for producing a (pseudo) random bit through techniques ranging from recycling of random bits [80], to techniques which extract randomness from weak random sources [37] and methods to “amplify” randomness through pseudo-random number generators [10, 199]. While Hypothesis 1 is claimed only for pseudo random generators, we opine that it is also valid for alternate sources of (pseudo) randomness.

u' associated with $\neg_{\hat{p}}$ where $\hat{p} = p_u$, and a new edge (u'', u') . Now for all edges (u, v) where $v \in V$, we replace it with edge (u', v) (when $p_u < 1/2$, u'' is associated with 1 and $p = 1 - p_u$). We shall refer to this circuit as $\mathbb{C}/\{u\}$.

Lemma 5.0.3 *The Boolean random variable x_u representing the value of any edge (u, v) in \mathbb{C} , where $v \in V$, is equivalent to the Boolean random variable $\hat{x}_{u'}$ representing the value of the edge (u', v) in $\mathbb{C}/\{u\}$.*

PROOF. Immediate from the definition of a probabilistic negation operator and the equivalence of random variables. \square

Let $\hat{\mathbb{C}} = \mathbb{C}/U$ denote the probabilistic Boolean circuit derived from \mathbb{C} by applying the above transformation for all vertices $u \in U$.

Theorem 9 *Given a randomized Boolean circuit \mathbb{C} , there exists a probabilistic Boolean circuit $\hat{\mathbb{C}}$ such that \mathbb{C} and $\hat{\mathbb{C}}$ compute identical truth tables.*

PROOF. For any $u \in U$, from Lemma 5.0.3 and a straightforward induction on the elements of U , it can be shown that \mathbb{C} and \mathbb{C}/U compute identical probabilistic Boolean truth tables. \square

5.0.5.3 Energy Advantages of Probabilistic Boolean Circuits

Based on Theorem 9 and the manner in which $\hat{\mathbb{C}}$ is constructed from \mathbb{C} , we can claim

Claim 5.0.5.1 *The energy consumed by the implicitly probabilistic circuit $\hat{\mathbb{C}} = \mathbb{C}/U$, is less than that consumed by \mathbb{C} which is explicitly randomized whenever the energy cost for producing each (pseudo) random bit x_u as an input to \mathbb{C} is higher than that of a probabilistic inverter realizing the probabilistic operation \neg_{p_u} .*

We will subsequently see (in Section 5.0.6) that the energy cost of producing a random (or pseudo random) bit is indeed higher than that of realizing a PBL operation

$\neg \hat{p}$. This is true based both on thermodynamic principles and through empirical studies based on physical realization of gates through randomness, thereby converting the conditional claim 5.0.5.1 above into an unconditional claim in these two contexts.

5.0.6 Energy Considerations For Realizing Probabilistic and Randomized Boolean Operators

The central result of Section 5.0.5 above, was to distinguish randomized and probabilistic Boolean circuits of identical size and depth through a metric which quantifies the *energy consumed* by these circuits. Referring back to Section 2.3.1, we call that in the physical domain, probabilistic switches [139] serve as a foundational model relating the thermodynamic (energy) cost of computing, to the probability of correctness of computing.

We recall from Section 2.4.1 that this theoretical evidence was substantiated empirically, in the domain of switches implemented using complementary metal oxide semiconductor (CMOS) technology, where the relationship between the probability of correctness of switching and its energy consumption was established through analytical modeling, as well as actual measurements of manufactured *probabilistic* CMOS (PCMOs) based devices [34]. To reiterate, whenever Law 1 holds, given any randomized Boolean circuit \mathbb{C} and its equivalent probabilistic Boolean circuit \mathbb{C} , the energy consumed by the latter is less than the energy consumed by the former.

5.0.7 Extending to Computational Model with State

PA in the Rabin sense [156], with incorporate probabilistic transition functions. A PA over an alphabet Σ is a system $\langle S, M, s_0, Q \rangle$ where $S = \{s_0, \dots, s_n\}$ is a finite set (of states), M is a function from $(S \times \Sigma)$ into the interval $[0, 1]^{n+1}$ (the transition probabilities table) such that for $(s, \sigma) \in (S \times \Sigma)$, the transition function $M(s, \sigma) = (p_0(s, \sigma), \dots, p_n(s, \sigma))$ where $0 \leq p_i(s, \sigma)$ and $\sum p_i(s, \sigma) = 1$. The initial state is denoted by s_0 where $s_0 \in S$ and $Q \subseteq S$ is the set of designated final states.

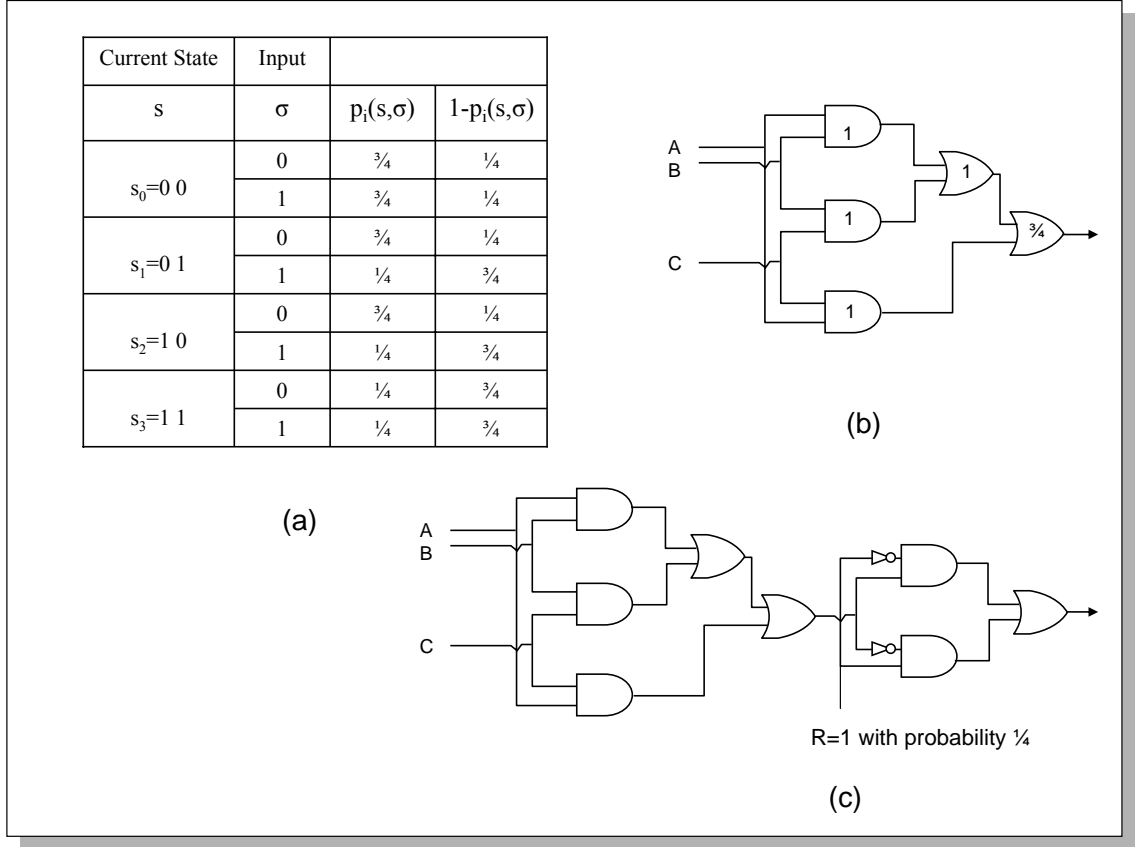


Figure 10: (a) A transition function encoded as a transition truth table (b) a probabilistic circuit which computes this transition truth table (c) an equivalent randomized Boolean circuit which computes the transition truth table

To establish that the distinction between the implicitly probabilistic and explicitly randomized variants established in Section 5.0.5 persists, we consider a *restricted probabilistic automaton* \mathcal{P} over an alphabet $\hat{\Sigma} = \{0, 1\}$. Given a state $\hat{s} \in \hat{S}$ and an input $\hat{\sigma} \in \hat{\Sigma}$, the cardinality of the set of possible successor states (with non zero transition probability) is at most two. That is for $(\hat{s}, \hat{\sigma}) \in (\hat{S} \times \hat{\Sigma})$, where $\hat{M}(\hat{s}, \hat{\sigma}) = (\hat{p}_0(\hat{s}, \hat{\sigma}), \dots, \hat{p}_n(\hat{s}, \hat{\sigma}))$, there exist distinct indices i and j , $0 \leq i, j \leq n$ such that $\hat{p}_i(\hat{s}, \hat{\sigma}) + \hat{p}_j(\hat{s}, \hat{\sigma}) = 1$ and for $0 \leq k \leq n$, $k \neq i$ and $k \neq j$, $\hat{p}_k(\hat{s}, \hat{\sigma}) = 0$. Furthermore, $\hat{p}_i(\hat{s}, \hat{\sigma}), \hat{p}_j(\hat{s}, \hat{\sigma}) \in \mathbb{Q}$; Rabin's formulation of PA is not restricted to rational probabilities since $p_i(s, \sigma)$ can be any value in the unit interval.

We observe here without proof, illustrated for completeness through an example in

Figure 10 that the transition function of any (restricted) PA \mathcal{P} can be represented as a probabilistic truth table. An example PA is illustrated in Figure 10 whose (transition) truth table is shown in Figure 10(a), where Figure 10(b) is a probabilistic Boolean circuit which computes this transition truth table, and Figure 10(c) is a randomized Boolean circuit which computes the transition truth table (with the random source labeled R). If each element of \hat{S} is encoded in binary, any $K \in (\hat{S} \times \hat{\Sigma})$ can be represented by a binary string (with the state concatenated to the input alphabet). For any state \hat{s} and an input alphabet $\hat{\sigma}$, the two possible successor states \hat{s}_i, \hat{s}_j (with non zero transition probabilities) can be represented by 0 and 1 respectively. Then, the transition function \hat{M} can be represented by a probabilistic Boolean truth table, with $2|\hat{S}|$ rows and 3 columns, where the first column of the k^{th} row contains K , the binary representation of k where K is an element of $(\hat{S} \times \hat{\Sigma})$. The second column contains $\hat{p}_{\hat{s}_j, \hat{\sigma}}$. From Observation 5.0.5.2 and Theorem 9, the (transition) truth table of \mathcal{P} can be computed using a probabilistic or randomized Boolean circuit respectively. This construction immediately allows us to extend the separation between probabilistic and randomized Boolean circuits to be applicable to the PA \mathcal{P} . Let $\hat{\mathbb{C}}_{\mathcal{P}}$ and $\mathbb{C}_{\mathcal{P}}$ respectively be the probabilistic and randomized Boolean circuit implementations of the transition function of \mathcal{P} . Then

Observation 5.0.7.1 *The energy consumed by $\hat{\mathbb{C}}_{\mathcal{P}}$ is less than that consumed by $\mathbb{C}_{\mathcal{P}}$ whenever the energy cost for producing each (pseudo) random bit x_u as an input to $\mathbb{C}_{\mathcal{P}}$ is higher than that of a probabilistic inverter realizing the probabilistic operation \neg_{p_u} .*

Again, based on the discussion in Section 5.0.6, we conclude that Claim 5.0.7.1 can be made unconditionally in the contexts when Theorem 1 or Law 1 are valid, in conjunction with Hypothesis 1.

CHAPTER VI

PROBABILISTIC ARCHITECTURES

As a key result of Chapter 5, we showed that the energy cost of a probabilistic Boolean circuit which implements a probabilistic Boolean function, is less than that of a randomized Boolean circuit of equivalent functionality. In this chapter, as an empirical demonstration and as a vehicle to realize the energy benefits in a practical context, we propose a *system on a chip* architecture, which we refer to as a *probabilistic* system on a chip architecture. The central idea behind probabilistic system on a chip (PSOC) architectures is to harness the probabilistic behavior of PCMOs devices and logic gates based on such devices—these are the physical implementation of logical operators of PBL—to design architectural primitives with well defined statistical behaviors ¹. These primitives, in turn, implement key (probabilistic) steps of probabilistic algorithms. Probabilistic algorithms, by definition are those which “toss coins” or execute steps whose outcomes have probabilities associated with them. Examples of such algorithms include the celebrated test for *primality* [157, 183], used as a key building block in RSA public-key cryptosystems. As we demonstrate in this chapter, PSOC implementations yield impressive energy and performance benefits at the application level. These energy and performance benefits arise from two sources: (i) The low voltage (and hence low energy) characteristics of PCMOs technology. This we characterized as the energy-probability relationship in the context of the logical operators of PBL (ii) harnessing the *implicit* probabilistic behavior of PCMOs devices and PBL operators *directly* to perform useful computation rather than to overcome this statistical behavior to achieve determinism—the conventional approaches toward

¹In this context, the reader is referred back to Section 2.4.1, to recall the key characteristics of PCMOs technology.

this end are rooted in redundancy or high voltage operation and inevitably lead to energy and possibly performance penalties.

The rest of the chapter is organized as follows, we first describe the probabilistic system on a chip architecture in Section 6.1. We then describe our metrics (Section 6.2) which we use to study the performance of PSOC architectures. The energy and performance modeling methodology which we adopt to evaluate PSOC architectures, is described in Section 6.2.1. Our PSOC co-design methodology—the application-architecture-technology (A²T) co-design methodology—differs from conventional co-design methodologies and is central to achieving the energy and performance benefits reported here. This co-design methodology, the main technology and algorithm characteristics which influence this methodology and the application characteristics of PSOC designs are elaborated in Section 6.3. We present results (Section 6.3.2 and analyze the results to account for and explain the energy and performance gains observed in PSOC implementations in Section 6.3.3. We discuss application optimization and PSOC implementation in detail in Section 6.5.

6.1 Probabilistic System on a Chip Architectures

We envision PSOC architectures are to consist of two parts as illustrated in Figure 11: a *host* processor which consists of a conventional low energy embedded processor such as the StrongARM SA-1100 [81] coupled to a *co-processor* which utilizes PCMOs technology. The co-processor is the practical implementation of the probabilistic Boolean circuit based on PBL. As we shall see in the subsequent sections, such a host - co-processor architecture affords several benefits. For a comparative study of the benefits of PSOC architectures with current designs, we consider three choices to be competitors for a PSOC.

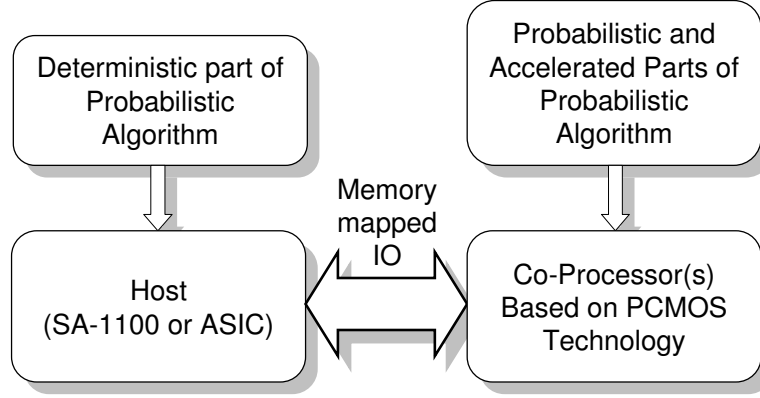


Figure 11: The canonical PSOC architecture

- As shown in Figure 12(a), a conventional “host-only” architecture which executes a deterministic algorithm; where a deterministic counter part of the probabilistic algorithm executes completely on the host processor. This is illustrated in Figure 12(a).
- A conventional “host-only” architecture which executes a probabilistic algorithm, (case (b)) where the probabilistic algorithm of interest executes completely on the host processor. The probabilistic component utilizes well known pseudo-random number generators implemented in software [144]. This style of implementation is shown in Figure 12(b).
- As shown in Figure 12(c), A conventional SOC, where a CMOS based co-processor implements the probabilistic parts of the application whereas the deterministic parts are executed as software on the host processor. The CMOS based co-processor forms the randomized Boolean circuit considered in Chapter 5.

These cases encompass alternate implementations of the application. Throughout this study, the co-processors illustrated in Figure 11 and Figure 12(c) are realizations using PCMOS and CMOS respectively that are application specific.

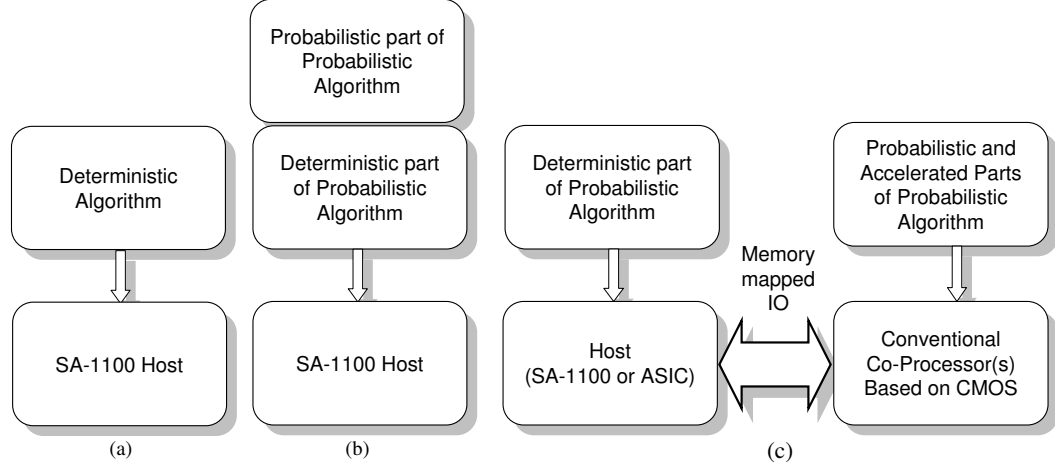


Figure 12: The conventional implementation alternatives for an application

6.2 *Energy and Performance Metrics for Probabilistic System on a Chip Architectures*

To highlight and to analyze the benefits of PCMOs technology, we now introduce several metrics to study the gains possible from PSOC implementations. In particular we will consider the **energy performance product** or EPP for short, as the chief metric of interest. The EPP metric has been chosen due to several considerations. It captures the chief characteristics of interest, namely the *energy* as well as the *time* needed for the execution of an application. In addition, given an architectural design to implement an application, the application execution could potentially be accelerated by replicating architectural blocks to exploit parallelism. In addition, techniques like voltage scaling could be used to trade performance for energy efficiency. It is our intention that the EPP metric would remain invariant under replication as well as voltage scaling as improvements in time would be off set by increase in energy and vice-versa. Hence EPP is a valuable metric to compare architectural implementations across differing technologies. Given the EPP of two alternate realizations, they can be compared by computing the *energy performance product gain*

Energy performance product gain: $\Gamma_{\mathcal{I}}$ is the ratio of the EPP of the baseline denoted by β to the EPP of a particular architectural implementation \mathcal{I} . $\Gamma_{\mathcal{I}}$ is

calculated as follows:

$$\Gamma_{\mathcal{I}} = \frac{Energy_{\beta} \times Time_{\beta}}{Energy_{\mathcal{I}} \times Time_{\mathcal{I}}} \quad (12)$$

Initially, to highlight the benefits of PSOC over the case where there is no co-processor, the baseline will correspond to the case where the entire computation is executed on the SA-1100 host. For example, in the case of the randomized neural network application which solves the vertex cover problem, the baseline will be the case where the SA-1100 host computes both the probabilistic and deterministic parts of the application (as illustrated in case (b) in Section 6.1) and \mathcal{I} corresponds to the case where the core probabilistic step is computed using a PCMOS based co-processor and the rest of the computation is performed using a SA-1100 host (as illustrated in Figure 11). Later, to quantify the benefits of PSOC implementations over conventional CMOS based SOC implementations, the baseline will correspond to the case where the SA-1100 host is coupled to a functionally identical CMOS based co-processor (case (c) in Section 6.1), where the co-processor computes the core probabilistic step. Wherever we present the EPP gain results, we will explicitly mention the baseline.

6.2.1 Performance and Energy Modeling of Probabilistic System on a Chip Architectures

Energy consumed (in joules) and performance (in terms of running time in seconds) as the application executes on a particular architecture, will be the chief attributes of interest. Our energy and performance modeling is *simulation based*. However, the energy consumed by the PCMOS devices are derived from actual measurements from a PCMOS test chip. As shown in Figure 11 in a PSOC architecture, the co-processors are memory mapped and the communication is modeled through LOAD and STORE instructions executed by the host. A special instruction triggers the execution of the application-specific PCMOS co-processor.

To model the performance of an application executing on such a PSOC, we have

modified the Trimaran [78, 27] compiler and simulator to reflect the ISA of StrongARM SA-1100 processor. The simulator records the trace of activity in the SA-1100 host processor, and access to the co-processors. This simulation is combined with the performance models of the co-processor, typically obtained through HSpice simulations, to yield the performance of the application in terms of the execution time.

The energy consumed by an application executing on such a PSOC is the sum of the energy consumed by the host, the energy consumed by the co-processor and the energy consumed due to communication between these components. To measure the energy of an application executing on such an architecture, we have incorporated the analytical model of Jouletrack [180] into the Trimaran simulator. This model is reported by its authors to be within 3% of the energy consumed by the actual SA-1100 processor. Thus, apart from estimating the performance of an application, the simulator is also used to estimate the energy consumed by the StrongARM host. The latencies caused by the slower PCMOS co-processor is accounted for as well. To estimate the energy consumed by the co-processors, the co-processors were designed and synthesized using and the associated energy consumption estimated using HSpice. In addition, actual measurement data of fabricated devices also using TSMC $0.25\mu m$ technology and their results are used as well. This, combined with the trace of the activity in the co-processor (recorded by the simulator) yields the energy consumed in the co-processor. Our performance and energy modeling techniques for a PSOC are illustrated in Figure 13. Since the applications of interest are probabilistic, at least fifty distinct executions are used to calculate the energy and performance of an application of various alternate realizations (listed in Section 6.1).

6.3 A Co-design framework

The gains obtained by leveraging PCMOS technology is due to the inherent energy advantages of probabilistic Boolean circuits over randomized Boolean circuits. These

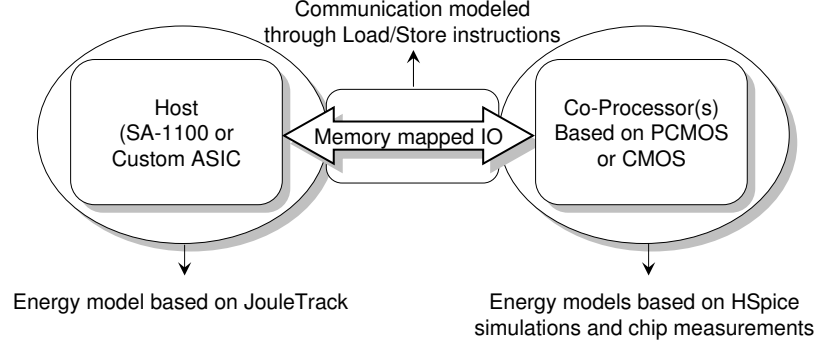


Figure 13: The energy and performance modeling methodology for each component of a PSOC

gains are realized at an application level using a unique co-design methodology that exploits technology characteristics of PCMOs as well as the algorithm characteristics of the application drivers, to provide a “good fit” implementation in the form of a PSOC architecture. Since the co-design methodology is of a greater interest than a detailed description of the application drivers and their implementation details, we briefly introduce the applications of interest and their characteristics that play an important role in co-design (a detailed description of each of the algorithms, the specific partitioning strategy for each of these applications and the corresponding PSOC implementation details are presented in Section 6.5). We then present the energy and performance results obtained from the PSOC implementation and a comparative study using metrics introduced in Section 6.2. We analyze these gains and then describe the algorithm and technology characteristics that influence the co-design.

6.3.1 A Brief Description of the Applications of Interest

We consider applications based on probabilistic algorithms, drawn from the cognitive and security domain. The algorithms include *Bayesian inference* [110], *Probabilistic Cellular Automata* [61], *Random Neural Networks* [64] and *Hyper Encryption* [49]. These algorithms would be referred to as BN, PCA, RNN and HE respectively. The applications in which each of these algorithms are utilized and the applications studied in this work are summarized in Table 2.

Algorithm	Application Scenarios	Implemented Application(s)	Core Probabilistic Step
Bayesian Inference [110]	SPAM Filters, Cognitive applications, Battlefield Planning [149]	Windows printer trouble shooting, Hospital Patient Management [7]	Choose a value for a variable from a set of values based on its conditional probability
Random Neural Network [64]	Image and pattern classification, Optimization of NP-hard problems	Vertex cover of a graph	Neuron firing modeled as a Poisson process
Probabilistic Cellular Automata [197]	Pattern classification	String classification [61]	Evaluating the probabilistic transition rule
Hyper-Encryption [49]	Security	Message encryption	Generation of a random string and encryption pad generation from this string

Table 2: The algorithms of interest, applications based on these algorithms and the core probabilistic step for each algorithm

The PSOC implementation for each of the algorithms consists of a StrongARM SA-1100 host and an application-specific co-processor as mentioned in Section 6.1. The co-processor design for each of these applications involves the partitioning of each of these applications between the host and the application specific PCMOS based co-processor. Once partitioned, PCMOS based co-processors are designed by hand. Though the specific manner in which these applications are partitioned vary and is not (currently) automated, they follow a common theme. Common to these applications (and to almost all probabilistic algorithms) is the notion of a *core probabilistic step* with its associated probability parameter p . For example, in probabilistic cellular automata application that has been considered [61], this is the probabilistic transition of an automaton which decides its next state based on the current state and a probability parameter p associated with the transition rule. The core probabilistic step for each of the application of interest is presented in Table 2. For each of the candidate applications, this core probabilistic step is identified by hand and PCMOS based co-processors designed for it. The deterministic parts of the application (for example, choosing which transition rule to apply in the context of probabilistic cellular automata) is implemented as software executing on the host processor.

6.3.2 Application Level Gains

Table 3 summarizes the application level EPP gains of PSOC over the baseline, for each of the applications of interest. Gains at the scope of an entire application range from a factor of about 80 for the PCA application, to a factor of about 300 in the context of the RNN application. As mentioned earlier, the baseline implementation for BN, HE, PCA and RNN applications is the StrongARM SA-1100 computing the deterministic as well as the probabilistic content and \mathcal{I} is a PSOC executing an identical probabilistic algorithm.

As seen from Table 3, the application level gains of each of the application vary.

Algorithm	$\Gamma_{\mathcal{I}}$	
	Min	Max
BN	3	7.43
RNN	226.5	300
PCA	61	82
HE	1.12	1.12

Table 3: Maximum and minimum EPP gains of PCMOs over the baseline implementation where the implementation \mathcal{I} has a StrongARM SA-1100 host and a PCMOs based co-processor

For example in the RNN case, a range of EPP gains are observed whenever multiple data points are available. This is attributed to the probabilistic nature of the applications: their execution characteristics differ yielding different gains for different input sets and sizes. In the sequel, we analyze the factors affecting gains in a systematic way.

6.3.3 An Analysis of Gains

Intuitively, the application level gain in energy and performance depend on two factors: (i) the “amount of opportunity” in the application to leverage the PCMOs based co-processor and (ii) the amount of gains afforded “per unit of opportunity”. Broadly, the factors which influence gain can be studied under two categories *Implementation independent* characteristics (which include *algorithmic* characteristics like the “amount of opportunity” inherent in an algorithm) and *implementation dependent* characteristics (which includes *technology and architecture* characteristics which influence the amount of gains afforded “per unit of opportunity”). These algorithmic, architecture and technology characteristics in turn, influence the co-design methodology (hence the name A²T co-design methodology); these considerations are outlined in the sequel and the specific effect on the PSOC design for each of the applications of interest will be described in Section 6.5

6.3.4 Implementation Independent Characteristics Influencing Co-design

As mentioned before, the *core probabilistic step* of each application is implemented in the PCMOS based co-processor and one core probabilistic step will be regarded as one “unit of opportunity”. The core probabilistic step for each of the application has been presented in Table 2. Given this, it is natural to expect that higher the opportunity to exploit PCMOS technology for efficient implementations, higher will be the gains. The “amount of opportunity” is formalized through the notion of *Probabilistic Flux* \mathcal{F} (or flux for short) where \mathcal{F} of an algorithm is defined as the *ratio of the core probabilistic steps to the total number of operations of an algorithm during a typical execution of the algorithm*. The “total number of operations” in this context refers to the total number of cycles consumed by the deterministic instructions executing on the StrongARM processor. Informally, \mathcal{F} can be regarded as ratio of the number of times a PSOC co-processor would be invoked to the number of times the host processor is “invoked” (cycles executed by the host processor). Flux for various algorithms will be presented in either the ratio form or in the form of a percentage.

With this as background and revisiting Table 3, we observe that the application level gains of each of the application vary. For example in the BN case, a range of EPP gains are observed whenever multiple data points are available. The Table 4 presents the flux as well as the Min and Max gains for each of the applications.

Algorithm	Flux \mathcal{F} (as percentage of total operations)	$\Gamma_{\mathcal{I}}$	
		Min	Max
BN	0.25%-0.75%	3	7.43
RNN	1.64%-1.97%	226.5	300
PCA	4.19%-5.29%	61	82
HE	12.5%	1.12	1.12

Table 4: Application level flux, maximum and minimum EPP gains of PCMOS over the baseline implementation where the implementation \mathcal{I} has a StrongARM SA-1100 host and a PCMOS based co-processor

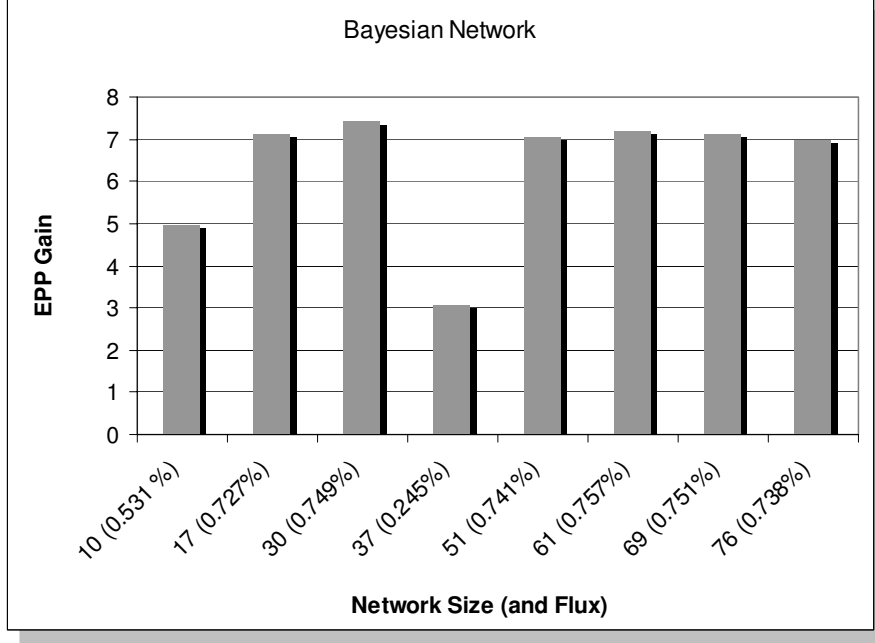


Figure 14: Gain and flux for Bayesian network of various sizes

The variation in gain is attributed to the probabilistic nature of the applications under consideration. Since these applications are probabilistic, their execution characteristics (and hence the flux) depend on the input size and the actual inputs. To understand the effect of flux, let us consider the BN application in detail. Figure 14 shows the gain for each network size and the corresponding flux \mathcal{F} .

As is to be expected, as the flux increases from 0.25 % (for a Bayesian Network size of 37) to 0.75 % (for a Bayesian Network size of 69), The corresponding gain increases from a factor of 3 to a factor of 7.14. In general, for a specific application, consider the energy consumed by the baseline implementation. This is a sum of the energy consumed at the StrongARM host for executing the deterministic parts of the application ($Energy_{det,\beta}$) and the energy consumed at the StrongARM host for executing the probabilistic part of the application ($Energy_{prob,\beta}$).

$$Energy_{\beta} = Energy_{det,\beta} + Energy_{prob,\beta}$$

Consider $Energy_{det,\beta}$ the energy consumed by the baseline for executing the deterministic part of the application. If the average energy consumed per “invocation” of the host processor (per cycle of the host processor) is $Energy_{cycle,host}$ and the number of invocations of the host processor is $Cycles_{det,host}$

$$\begin{aligned} Energy_{\beta} &= Energy_{det,\beta} + Energy_{prob,\beta} \\ &= Cycles_{det,host} \times Energy_{cycle,host} + Energy_{prob,\beta} \end{aligned}$$

Consider $Energy_{prob,\beta}$, the energy consumed by the baseline for executing the probabilistic part of the application. Let the energy consumed per “invocation” of the core probabilistic step be $Energy_{flux,\beta}$. From the definition of Flux (\mathcal{F}), the number of invocations of the core probabilistic step is $\mathcal{F} \times Cycles_{det,host}$ and

$$\begin{aligned} Energy_{\beta} &= Energy_{det,\beta} + Energy_{prob,\beta} \\ &= Cycles_{det,host} \times Energy_{cycle,host} + Energy_{prob,\beta} \\ &= Cycles_{det,host} \times Energy_{cycle,host} + \mathcal{F} \times Cycles_{det,host} \times Energy_{flux,\beta} \end{aligned}$$

Similarly the energy consumed by the PSOC implementation $Energy_{\mathcal{I}}$ can be written as

$$\begin{aligned} Energy_{\mathcal{I}} &= Cycles_{det,host} \times Energy_{cycle,host} + \mathcal{F} \times Cycles_{det,host} \times Energy_{flux,\mathcal{I}} \\ &\approx Cycles_{det,host} \times Energy_{cycle,host} \end{aligned}$$

The approximation arises due to the fact that the PCMOs based co-processor consumes almost negligible energy (this can be seen from Table 5; However, the actual gains presented here consider the energy of the co-processor as well, the approximation is used purely for explanation purposes). Similarly, we can derive an expression for

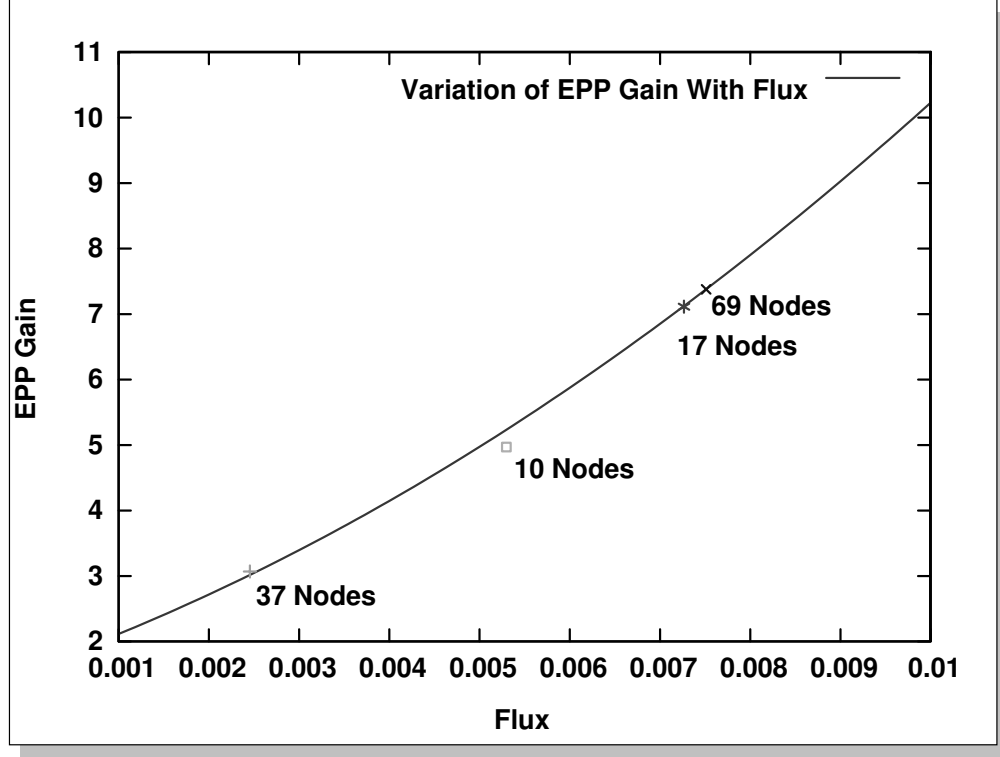


Figure 15: Variation of gain with respect to flux for Bayesian network

performance as well and for a specific application, the gain $\Gamma_{\mathcal{I}}$ can be characterized as

$$\begin{aligned}
 \Gamma_{\mathcal{I}} &= \frac{Energy_{\beta} \times Time_{\beta}}{Energy_{\mathcal{I}} \times Time_{\mathcal{I}}} \\
 &= \left(1 + \frac{\mathcal{F} \times Energy_{flux,\beta}}{Energy_{cycle,host}}\right) \times \left(1 + \frac{\mathcal{F} \times Time_{flux,\beta}}{Time_{cycle,host}}\right) \quad (13)
 \end{aligned}$$

Reverting back to Section 6.3.3, we notice that the gains depend on Flux \mathcal{F} , an implementation independent algorithmic characteristic which determines the “amount of opportunity”. Also the gains depend on $Energy_{flux,\beta}$ and $Time_{flux,\beta}$ which are implementation dependent technology and architecture characteristics. Counter intuitively, the gains also depend on $Energy_{cycle,host}$ and $Time_{cycle,host}$, which capture the *efficiency of the host processor*! This will be further explored in Section 6.4

For the Bayesian Network application, Figure 15 shows how $\Gamma_{\mathcal{I}}$ varies with the

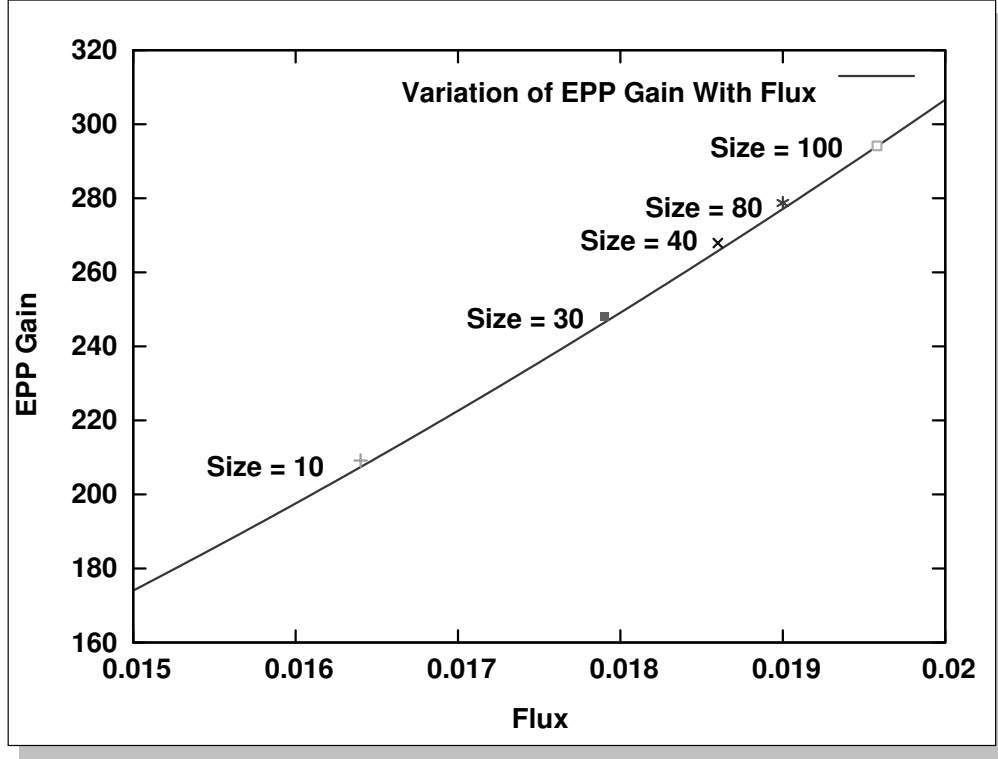


Figure 16: Variation of gain with respect to flux for randomized neural network

flux. The line is analytically calculated using Equation 13, and the points correspond to actual values measured using the simulations. Two particular points of interest, whose flux correspond to Bayesian network sizes of 37 nodes and 69 nodes respectively are also shown in the figure. It can be seen that the simulation result matches closely with that of the analytical model. Similarly, Figure 16 shows the variation of $\Gamma_{\mathcal{I}}$ with the flux for the randomized neural network application. Again, the line is calculated analytically and the points correspond to gains obtained from simulation. Thus, the flux \mathcal{F} of an algorithm is an important characteristic that determines the gains derived from a PSOC implementation. Hence, given an algorithm, it is advantageous to maximize opportunity (in this context increase the amount of probabilistic steps whenever possible) and given an application, to leverage higher gains, it is advantageous to leverage an algorithm with the highest “probabilistic content” or flux. These considerations influence the selection and optimization of the algorithm used

Application	gain over SA-1100	gain over CMOS
BN	9.99×10^7	2.71×10^6
RNN	1.25×10^6	2.32×10^4
PCA	4.17×10^4	7.7×10^2
HE	1.56×10^5	2.03×10^3

Table 5: The EPP gain of PCMOs over SA-1100 and over CMOS for the core probabilistic step

for a particular application in our A²T co-design methodology.

6.3.5 Implementation Dependent Characteristics Influencing Co-design

The application level gains not only depends on the flux of an application but on the energy and performance gains afforded per “unit of opportunity”. Table 5 presents the EPP gain of PCMOs based co-processor for the core probabilistic step of each of the applications of interest. The second column in the table corresponds to the case where β is the SA-1100 host without any co-processor and the third column corresponds to the case where β is a SA-1100 host coupled to a conventional CMOS based co-processor. As can be seen from the table, a PCMOs based co-processor is over five orders of magnitude better in terms of EPP when compared to a SA-1100 processor, and over three orders of magnitude when compared to a CMOS based co-processor while executing the core probabilistic step of the HE application.

For a given flux, the application level gain would increase with increase in the energy as well as performance gain per unit flux. To illustrate this, let us revisit the Bayesian Network application, and the gain $\Gamma_{\mathcal{I}}$ where \mathcal{I} is a PSOC and the baseline is a StrongARM SA-1100 host without a co-processor. In particular, let us consider the case where the size of the Bayesian Network is 37 nodes and the corresponding flux is 0.25%. Now, higher the efficiency of PCMOs (in the form of lesser energy and faster execution) per invocation of the core probabilistic step, higher would be the gain. That is, higher the energy and time *saved* per invocation of the core probabilistic

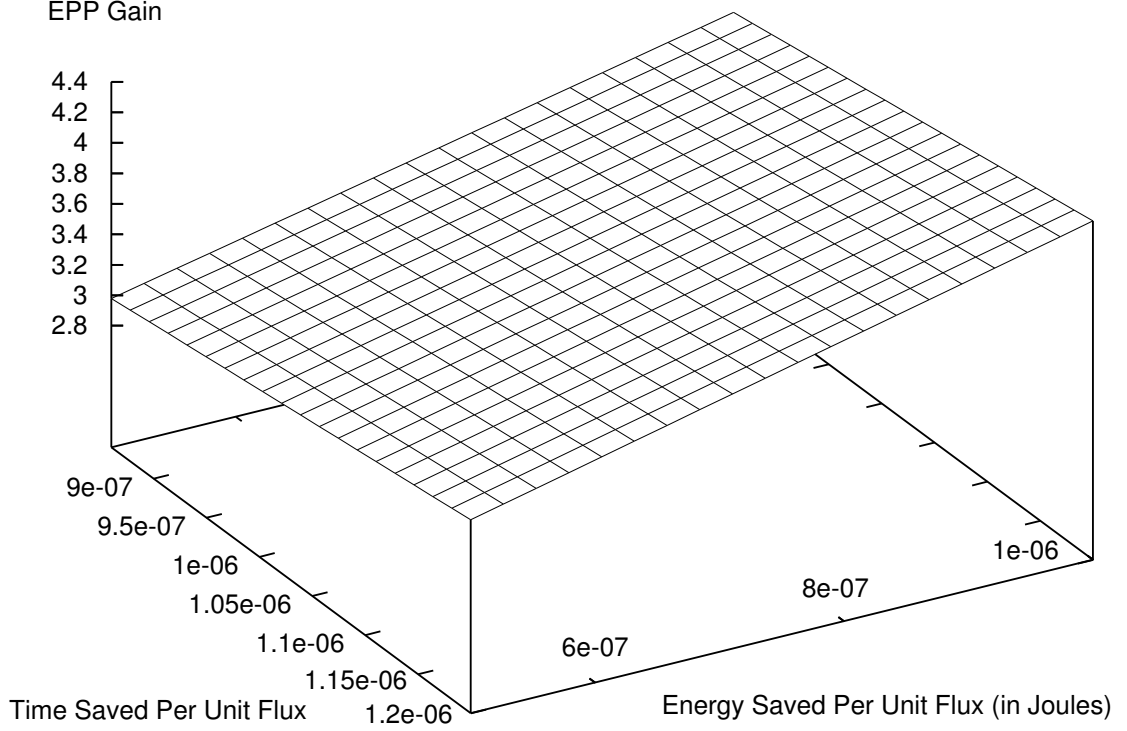


Figure 17: For a fixed flux, variation of gain with respect to energy saved per unit flux and time saved per unit flux by using PCMOS

step, higher is the gain afforded by the PSOC implementation. Figure 17 illustrates the variation of $\Gamma_{\mathcal{I}}$ with respect to the cycles per unit flux and energy per unit flux expended by the baseline implementation. The point where the surface intersects the z axis, presents the performance and energy consumption per unit flux which corresponds to a gain of 3 and the point plots the performance and energy consumption per unit flux for the Bayesian Network of size 37.

With that as background and revisiting Table 5, we observe that the energy and performance gain afforded per unit of flux varies across applications. This is an artifact of the functionality of the core probabilistic step as well as the characteristics of PCMOS technology. The technology characteristics of PCMOS technology, which influence the energy and performance gains per core probabilistic step is enumerated below:

- **PCMOS Energy Efficiency** - PCMOS based switches are extremely efficient for

implementing logical operations with probabilities associated with their outcomes. For example, the energy consumed for one *probabilistic inversion* (a logical NOT operation with a probability of correctness p associated with it), operation is 0.4 pico-joules [35] whereas emulating similar functionality using a hardware based implementation of the Park-Miller algorithm consumes 2025 *times* this much energy. As would be expected, more complex core probabilistic operations afford high gains per unit flux.

- **PC MOS Specialization** Apart from efficient operation, PC MOS devices can be “tuned” to the desired probability parameter of any probabilistic step S . For example, PC MOS based primitives could be built for probabilistic inversion with a probability of correctness $p = 0.75$. Further details as to how the probability of correctness can be controlled is presented in [35].

Corresponding implementations in software or conventional CMOS incurs a penalty for non trivial probabilities ($p \neq 0.5$). This is because, say to achieve a probability parameter $p = 0.75$, typical implementations would generate a number uniformly at random, say between 0 and 2^{16} and compare it with $2^{16} \times 0.75$. This involves *dilation* of one bit to 16 bits captured by the notion of the *Dilation Factor* \mathcal{D} . Hence core probabilistic step with non-trivial probabilities afford higher gains per unit flux.

- **PC MOS Replication Due to Specialization** Whereas *specialization* to a particular probability parameter p has the advantage of avoiding penalty associated with tuning and dilation, separate PC MOS building blocks need to be implemented for probabilistic operations that are similar but differ only in their probability parameter. For example, two different PC MOS based primitives need to be built for two probabilistic inversion operations with probability $p = 0.75$ and $p = 0.80$ respectively. This replication of PC MOS primitives due to specialization

is captured by the metric *Spread factor* denoted by \mathcal{S} and is a count of such distinct probability parameters used by an application. Spread factor guides application optimization by reducing the distinct probability parameters used by an application, and architecture optimization by choosing a non-specialized implementation if the spread factor is too high.

- **PCMOS Operating Frequency** - Though PCMOS devices are extremely (energy) efficient, the operating frequencies of our current implementations are low [35], at about $1MHz$. This acts as a potential limitation to the peak rate with which probabilistic steps can be executed on the PCMOS based co-processor. Given this limitation, the peak rate with which a probabilistic step S needs to execute on the co-processor so that the host processor is not stalled, is a characteristic of interest. This peak rate henceforth be referred to as the *application demand rate for the probabilistic step S* . Intuitively, the application demand rate is dependent on algorithm characteristics and the operating frequency of the host processor. If the application demand rate for a probabilistic step S is higher than the operating frequency of the PCMOS building block which executes the step S , the host processor would need to *stall* till the probabilistic steps finish execution. This is analogous to *memory stall cycles* in modern microprocessors where there is a mismatch between the frequency of operation of the data path and the memory subsystem. This limitation can be remedied through parallelism; by replicating PCMOS building block which executes the step S . The number of replications is captured through the *replication factor \mathcal{R}* . The replication factor is a characteristic that guides application as well as architecture optimization. On the application side, *program transformations* could be performed to better interleave the probabilistic steps with the deterministic steps (which execute on the host processor) so that the peak application demand rate is reduced. In addition, since the current implementation

of PCMOS devices do not allow them to be switched off when not needed (akin to clock-gating techniques in conventional micro architecture design), increased replication, which decreasing the time consumed to execute an application might increase the energy consumption. This trade off needs to be taken into account while replication PCMOS building blocks.

- **PSOC Communication Costs** There is an inherent cost of communication between the host processor and the PCMOS based co-processor which can potentially reduce the gains. While partitioning the application, this should be considered as well.

6.4 *A Comparison of Implicitly Probabilistic and Explicitly Random Circuits in the System on a Chip Context*

We have demonstrated the utility of PCMOS technology and PSOC implementations of selected applications by presenting the energy and performance gains of PCMOS based PSOC designs over a conventional host only style of implementation. A more ambitious and interesting comparison would be with that of a conventional SOC design where a functionally identical co-processor is designed with conventional CMOS technology. These form an empirical comparison of the energy efficiency of system on a chip architectures in the implicit and explicit contexts. With the conventional CMOS based SOC (explicit context) as the baseline, the gain $\Gamma_{\mathcal{I}}$ where \mathcal{I} is a PCMOS based PSOC for HE as well as PCA application is 1.

This is in spite of high flux and gains per core probabilistic step in the corresponding applications. To explain this, let us revisit Equation 13. We note that the gains depend on the Flux \mathcal{F} , the gains per core probabilistic step (approximately $Energy_{flux,\beta}$ and $Time_{flux,\beta}$) which were studied and analyzed in the preceding sections. More importantly the gains depend on $Energy_{cycle,host}$ and $Time_{cycle,host}$ as well, which indicates that *if the computation that is executed on the SA-1100 host*

dominates the energy and time consumption of the entire application, then the gains from PCMOS based PSOC would be low. Hence, even though the proportion of the core probabilistic steps in the entire application is high and the gains per core probabilistic step is high, using a PCMOS based co-processor has almost no impact at the application level time and energy consumption. Thus gains through PCMOS—the limits being substantial as shown in Table 5—can be truly achieved only if the amount of effort spend in the co-processor is comparable in terms of EPP units to that spent in the host.

To verify this hypothesis, a baseline SOC architecture in which the host processor and the co-processor are both custom ASIC architectures is considered. With this notion, moving away from a StrongARM host processor to one realized from custom ASIC logic, amount of energy and running time spent in the host is considerably lower. Thus and perhaps counter intuitively, increasing the efficiency of the competing approach enhances the value of PCMOS gains at the application level. In the context of the HE application, and with this change to the baseline, the gain $\Gamma_{\mathcal{I}}$ increases to 9.38 - almost an order of magnitude. Similarly when a baseline with a custom ASIC host is used, the $\Gamma_{\mathcal{I}}$ value in the context of the probabilistic cellular automata application increases to 561. In all of these comparisons, the CMOS based co-processor has been operated at an optimal frequency, that is the frequency which yield the lowest energy consumption without degrading application performance. In addition, the CMOS based co-processors are assumed to leverage techniques like clock-gating with no overhead. In this respect the gain estimates are conservative. We view this fact as being extremely favorable for PSOC based designs, as host processors become more efficient with future technology generations, the gains of PSOC architectures over conventional SOC architectures increase.

6.5 *The Suite of Applications, Partitioning, Optimization and PSOC Implementation*

In this section, we describe in detail the applications, their partitioning, optimization and PSOC implementation.

Bayesian Networks (BN) - Bayesian inference [110] is statistical inference technique. *Hypotheses*, their corresponding *probability weights* and *evidences* are central characteristics of this technique. The probability weight p associated with a hypothesis H is interpreted as the *degree of belief* in the hypothesis. Evidences support (or discount) a hypothesis, thereby increasing (or decreasing) the associated probability weight and hence the degree of belief in the hypothesis. Hypotheses whose probability weights approach 1 are most likely and those whose probability weights approach 0 are very unlikely. A Bayesian network can be used to perform Bayesian inference. A Bayesian network is a directed acyclic graph G of nodes V which represent variables and edges $E \subseteq V \times V$ which represent dependence relations between the variables. Each node $v_x \in V$ can be associated with a value from a finite set of values Σ_x . The set Σ_x will be referred to as the *set of possible values associated with v_x* .

Without loss of generality, let $v_1, v_2, v_3, \dots, v_k$ be the k parents of v_x . Let Σ_1 be the set of possible values associated with v_1 ; similarly let $\Sigma_2, \Sigma_3, \dots, \Sigma_k$ be associated with v_2, v_3, \dots, v_k respectively. Each value $\sigma \in \Sigma_x$, has a conditional probability $p(\sigma/\sigma' \in \Sigma'_x)$ associated with it; where $\Sigma'_x = \Sigma_1 \times \Sigma_2 \times \Sigma_3 \cdots \Sigma_k$. In essence σ' is the string of values of the variables represented by the k parents of the node v_x and Σ'_x is the set of all possible strings. Variables whose values are known apriori are called as *evidences* and based on evidences, other variables can be *inferred*. Based on network topology and conditional probabilities associated with the variables, various cognitive tasks can be performed. The particular Bayesian networks considered in this study are a part of the following applications: Hospital patient monitoring system [7] and

a printer trouble shooting application for the Windows operating system.

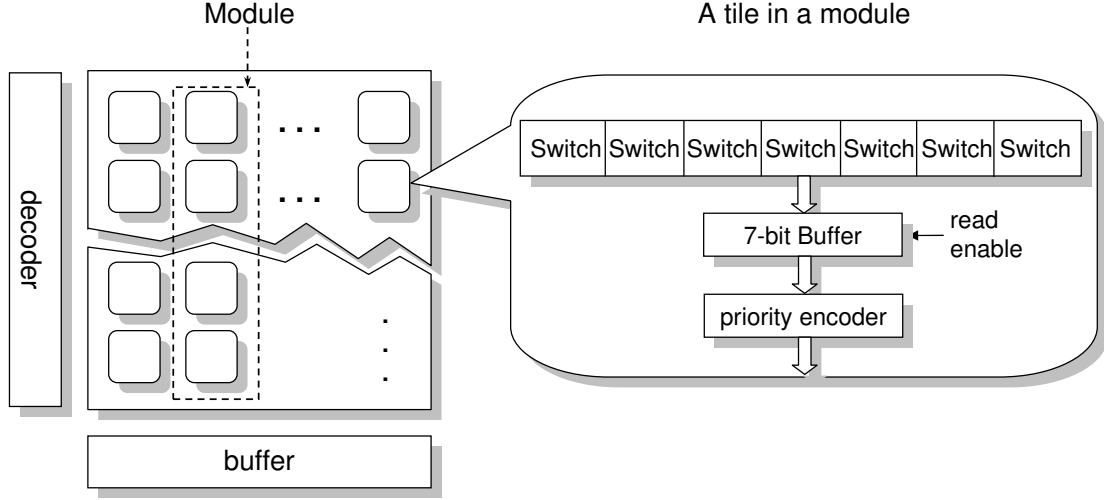


Figure 18: The co-processor architecture for a PSOC implementing Bayesian Inference

Partitioning, Optimization and PSOC Implementation We choose the likelihood weighting algorithm [149] for Bayesian inference. To illustrate, consider a node $v_x \in V$ with $\Sigma_x = \{0, 1, 2\}$. As before, let Σ'_x be the set of all possible strings of values associated with the parents of x . Let $0 \leq p(0/\sigma'), p(1/\sigma'), p(2/\sigma') \leq 1$ where $p(0/\sigma') + p(1/\sigma') + p(2/\sigma') = 1$, be the conditional probabilities associated with $0, 1, 2 \in \Sigma_x$ respectively, given that σ' is the string formed by the outputs of the parents of the node v_x . The inference process performs a random experiment with three possible outcomes 0, 1 or 2 with the associated probability $p(0/\sigma'), p(1/\sigma')$ and $p(2/\sigma')$ respectively.

In our PSOC architecture, Bayesian inference will be performed by three PC MOS switches A, B and C which corresponds to 0, 1, 2 respectively. The inputs for these switches are fixed at 0 and the probability of correctness associated with A, B, C is by design, $p(0/\sigma'), \frac{p(1/\sigma')}{1-p(0/\sigma')}$ and 1 respectively. Thus, when the switches are inspected in the order $\langle A, B, C \rangle$, the value which corresponds to the leftmost switch whose output is the value 1 is the value inferred by the node. In the PSOC design, the set of switches $\{A, B, C\}$ will be referred to as a *row*. A row of switches is associated

with each member of the set Σ'_x , hence at least $|\Sigma'_x|$ rows are required for a node v_x . These set of rows associated with a node v_x will be referred to as a *module* which corresponds to the node v_x .

As shown in Figure 18, the PCMOS module which corresponds to a node v_x implements a table. Rows in this module are indexed by a particular string σ' of values associated with the parents of v_x . The number of columns in the module is $|\Sigma_x|$, where each column corresponds to a value from the set Σ_x ; in our example, $|\Sigma_x| = 3$ (and in the figure, it is 7). A switch in the module, identified by $\langle \text{row}, \text{column} \rangle$ is a specialized PCMOS switch whose probability of correctness is computed as indicated above. Finally a conventional priority encoder is connected to the outputs of a row to determine the final result of the random experiment; it performs the function of inspecting the values of a row and choosing the final output associated with v_x . The random experiment (used for inference) in this probabilistic algorithm, is implemented in the PCMOS co-processor (which consists of several modules), with the remainder implemented as software executing on the host.

Random Neural Network (RNN) Following Gelenbe [64], a random neural network consists of *neurons* and *connections* between the neurons. Information is exchanged between the neurons in the form of *bipolar signal trains*. Neurons have potentials associated with them, which are defined to be the sums of incoming signals. This potential in turn, influences the rate of firing. A random neural network can be modeled as an undirected graph G of nodes (neurons) V and directed edges (connections) $E \subseteq V \times V$. Each node has an associated potential ψ which is incremented (decremented) by incoming (outgoing) firings. The firings occur with a constant rate with exponentially distributed intervals. When a node fires, its potential is decremented by one and the polarity and destination of the firing are determined by probability parameters p_i and p_d respectively. Through a suitable combination of

network topology, probability parameters and firing rates, several optimization problems can be solved. The particular neural network considered in this study is used to heuristically determine the vertex-cover of a graph due to Gelenbe and Batty [65].

Partitioning, Optimization and PSOC Implementation The Poisson process which models the “firing” of a neuron is implemented in the PCMOS co-processor, with the rest of the computation (distributing the firings, updating the potentials) implemented to execute on the host processor. To realize the Poisson process characterizing a neuron firing, the Bernoulli approximation of a Poisson process [57] is used. As an example of a methodological step in our A²T co-design approach, since the rate at which neuron firings need to be modeled exceeds the rate at which PCMOS based switches can compute, the PCMOS based devices which model the Poisson process are replicated to match the required rate. In the interests of efficiency, and as another example of our A²T methodology, the application is restructured to reduce the replication factor \mathcal{R} , by interleaving the modeling of neuron firings (in the PCMOS based co-processor) and the processing of these firings (in the host processor)—distributing the firings more evenly over the course of the entire application’s execution. This has the effect of reducing the *peak* application demand bandwidth.

Probabilistic Cellular Automata are a class of cellular automata used to model stochastic processes. Cellular automata consist of *cells* with local (typically nearest neighbor) communication. Each cell is associated with a *state* and a simple *transition rule* which specifies the next state of a state transition based on its current state and the states of its neighbors. In the *probabilistic* string classification algorithm [61], the state of each cell is either 0 or 1. The next state of a cell depends on its current state and the current states of its *left* and *right* neighbors. Thus there are 8 possible transition rules where each rule has two possible outcomes 0 or 1. In addition, the transition rules are probabilistic: for a transition rule τ_i ($0 \leq i \leq 7$) probability that the output state of the rule is 0 is denoted by $p_{i,0}$ and probability

that the output state is 1 is denoted by $p_{i,1}$.

Partitioning, Optimization and PSOC Implementation Each transition rule is implemented by a PCMOS inverter whose input is a 0. The i^{th} inverter corresponds to the i^{th} transition rule and the probability of correctness associated with the i^{th} inverter is $p_{i,1}$. The control-intensive part of choosing transition rule (based on the state of a cell and the states of its neighbors) and updating the states is implemented on the host processor. Since the rate at which the transition rules need to be evaluated exceeds the frequency of operation of the PCMOS devices (choosing between the transition rule and updating the current state can be executed very fast on the SA-1100 host), this structure is replicated many times.

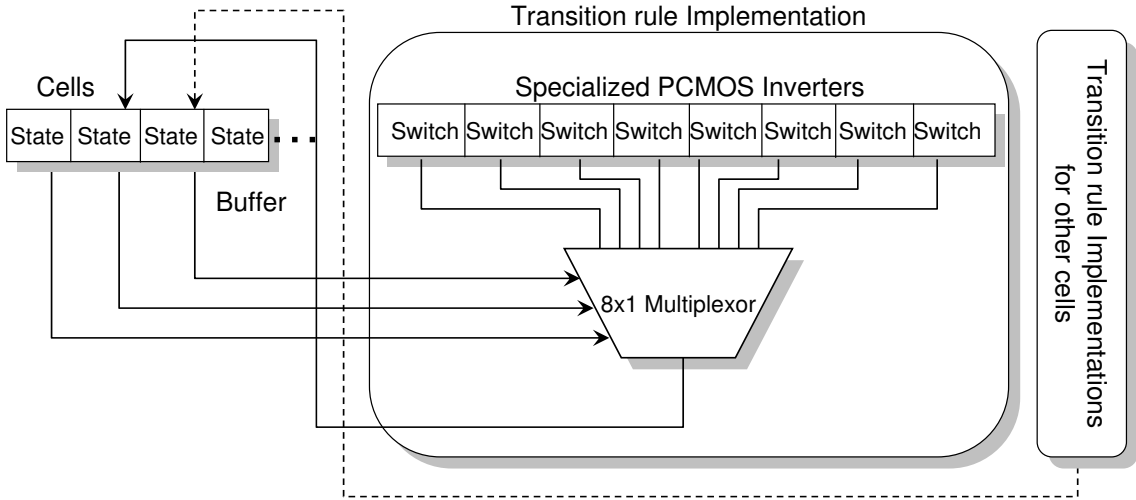


Figure 19: The Custom ASIC host and PCMOS co-processor architecture for a PSOC implementing a probabilistic cellular automata algorithm

In addition, a custom CMOS based ASIC can be designed to implement the deterministic part of the algorithm. As shown in Figure 19, the ASIC consists of an n bit buffer and n 8×1 multiplexers, one for each cell. The “select” input of the j^{th} multiplexer $0 < j < n + 1$ is the current state of the j^{th} cell and the states of its left and right neighbors. The inputs of the multiplexers are from PCMOS inverters specialized to the corresponding probability parameters. The transitions are stopped as soon as all the cells have identical states. This is detected by an n input OR and an n input

AND gate. The energy of this custom asic is obtained through HSpice simulations with the energy of PCMOS inverters obtained from actual chip measurements.

6.6 *Some Practical Considerations*

We now address certain practical considerations towards implementing the PSOC designs described in this chapter. We first consider the problem of multiple voltage levels and then remark on the effect of the quality of randomization on the applications implemented using PSOCs.

6.6.1 Reducing Multiple Voltage Levels

In the designs described in this work, the probability p of correctness need to be varied on an application specific basis. In addition, an application may use several distinct probability parameters. This, as described in Section 6.3.5, increases the *Spread factor* due to replication caused by specialization. In addition, since the probability parameter p is controlled by varying the voltage, a high spread factor implies that several distinct voltage levels are needed to operate the PCMOS devices in the chip. Supplying distinct voltage levels on a chip requires voltage regulators which are costly in terms of area as well as energy. We make two observations towards addressing this problem (i) The distinct probability parameters are a requirement of the application and the application *sensitivity* to probability parameters is an important aspect. That is, if an application, uses probability parameters p_1, p_2, p_3 , it might be the case that the application level quality of solution is not degraded much when only p_1, p_2 are used. This, however can be determined only experimentally. (ii) Given a probability parameter p_1 and p_2 through logical operations, other probability parameters might be derived. For example, if the probability of obtaining a 1 from one PCMOS device is p and the probability of obtaining a 1 from a second PCMOS device is q , a logical AND of the output of the two PCMOS devices produces a 1 with a probability $p.q$. Using this technique, in the context of an application (the case of Bayesian inference is used

here), the spread factor may be reduced by producing several probability parameters using a few probability parameters. The formulation of such an optimization problem is described below.

Consider a tree (the *composition tree*) with directed edges $G \equiv (V, E)$, where V is a set of vertices and $E \subseteq (V \times V)$ a set of directed edges. This tree describes the set of logical operations required to derive the probability parameters required by an application.

Let $V = I \cup C \cup O$ where $I \cap C = C \cap O = I \cap O = \phi$ and $|O| = 1$. Let I be the set of *input vertices*, C be the set of *computing vertices* and $o \in O$ the *output vertex*. The input vertices are PCMOS devices, the computing vertices are the logical operations and the output of the output vertices yield the probability parameters required by the application. Given an edge $e \equiv (u, v) \in E$ where $u, v \in V$, the *value associated with the edge*, $val(e)$ is the value associated with the vertex u . Or, in other words, $val(e) = val(u)$ where $e = (u, v)$.

- **Set I :** For any vertex $v \in I$, $val(v) \in \mathbb{R}^+$.
- **Set O :** For the vertex $o \in O$, $val(v) = val(e)$ where e is the incoming edge incident on o .
- **Set C :** Any vertex $v \in C$ is of one of three types AND, OR, NOT. For all vertices of type AND, OR, the in-degree is two and out-degree is one. For all vertices of type NOT, the in-degree is one and out degree is one.
 - For any vertex $v \in C$ and v of type AND with incoming edges e_i, e_j , the value associated with v , $val(v) = val(e_i) \times val(e_j)$.
 - For any vertex $v \in C$ and v of type OR with incoming edges e_i, e_j , the value associated with v , $val(v) = 1 - (1 - val(e_i)) \times (1 - val(e_j))$.
 - For any vertex $v \in C$ and v of type NOT with incoming edge e_i , the value associated with v , $val(v) = 1 - val(e_i)$.

Consider a set P such that $P \equiv \{p_1, p_2, p_3, \dots, p_k\}$ where $p_i \in \mathbb{R}^+$ and Q such that $Q \equiv \{q_1, q_2, q_3, \dots, q_l\}$ where $q_i \in \mathbb{R}^+$. Let P be called as the set of *input probabilities* and Q be called as the set of *application-required probabilities*.

A composition tree G_i is said to compute $q_i \in Q$ with input probabilities P , if for each input vertex v of G , $val(v) \in P$ and the value of the output vertex of G , $val(o) \approx q_i$ where $x \approx y$ if for some ϵ , $y - \epsilon \leq x \leq y + \epsilon$. That is, when elements from set P are input to the composition tree G_i , the value of the output vertex $\approx q_i$

For a composition tree G_i which computes q_i given P , G_i is defined to be the minimal composition tree, if $\nexists G'_i$ such that G'_i computes q_i given P and the number of vertices in G'_i is less than the number of vertices in G_i . Henceforth, “composition tree” would refer to the minimal composition tree.

To compute the application required probability parameters from a set of input probability parameters, the cost includes the cost of the PCMOS devices, the cost of the logic in the composition tree and the cost introduced due to multiple (though reduced) probability parameters of the input.

The cost of computing q_i given a set of input probabilities P , denoted by $C_P(q_i)$ is the number of vertices in composition tree G_i which computes q_i given P . The cost of computing the set Q given P is denoted by $C_P(Q)$ is $\sum_{q_i \in Q} C_P(q_i)$. The cost of the set of input probabilities, denoted by \bar{C}_P is $\bar{C}_P = k \times |P|$ where k is some constant.

Question: Given a Q , compute P and the composition trees such that $C_P + \bar{C}_P$ is minimum over all possible P .

This optimization problem might be solved using a combination of linear programming and heuristics. As an illustration, an (unoptimized) hand implementation of deriving twenty probability parameters from two input probability parameters is described below in the Table 6 (note that the other ten probability parameters can be obtained by the NOT of those in the table).

Application-required Probability Parameters	Composition Tree
0.05	$[[(0.4) \text{AND} (0.5)] \text{AND} (0.5)] \text{AND} (0.5)$
0.10	$[(0.4) \text{AND} (0.5)] \text{AND} (0.5)$
0.15	$[(0.5) \text{AND} [\text{NOT} (0.4)]] \text{AND} (0.5)$
0.20	$(0.4) \text{AND} (0.5)$
0.25	$(0.5) \text{AND} (0.5)$
0.30	$(0.5) \text{AND} [\text{NOT} (0.4)]$
0.35	$[\text{NOT} [(0.5) \text{AND} [\text{NOT} (0.4)]] \text{AND} 0.5$
0.40	0.40
0.45	$[\text{NOT} [[(0.4) \text{AND} (0.5)] \text{AND} (0.4)]]$
0.50	0.50

Table 6: The probability parameters required by the application, and the composition tree for generating them using two voltage levels

6.6.2 Quality of Randomization

In any implementation of applications which leverage probabilistic algorithms the *quality* of the implementation is an important aspect apart from the energy and running time. In conventional implementations of probabilistic algorithms—which usually leverage hardware or software based implementations of pseudo random number generators to supply pseudo random bits which serve as “coin tosses”—it is a well known fact that random bits of a “low quality” affect application behavior, from the correctness of Monte Carlo simulations [58] to the strength of encryption schemes. To ensure that application behavior is not affected by low quality randomization, the quality of random bits produced by a particular strategy should be assessed rigorously. The problem of “quality assessment” of random sequences has been well studied and is rooted in the very concept of “randomness”. Kolmogorov considers a finite sequence to be random if there is no appreciably shorter sequence that describes it fully, in some unambiguous mathematical notation (from [70]). However the problem of determining the shortest sequence which describes a finite sequence of numbers, is in general, undecidable [24]. A more practical definition of “pseudo-randomness”

was introduced by Yao, where informally, a sequence is pseudo-random if there is no *polynomial time* algorithm which can distinguish that sequence from a truly random one [199]. However, it is impractical to test for pseudo-randomness since there are an infinite number of polynomial time algorithms. Hence the current strategy is to leverage statistical tests to test for the quality of randomness. To study the statistical properties of PCMOs devices in a preliminary way, we have utilized the randomness tests from the NIST Suite [162] to assess the quality of random bits generated by PCMOs devices. Preliminary results indicate that PCMOs affords a higher quality of randomization; A future direction of study is to quantify the impact of this quality on the application level quality of solution.

CHAPTER VII

PROBABILISTIC ARITHMETIC

Efficient execution of arithmetic operations is of paramount importance for high performance and low energy implementation of algorithms, where arithmetic operations dominate. Examples of such algorithms include those in the domain of digital signal processing (DSP). In the context of VLSI-based implementation of arithmetic operations, efficient implementations, including those which perform various trade-offs—time of execution for the size of the implementation for example—have been studied extensively (see [46] for example). Typically, these techniques exploit properties at the algorithmic level (the relationship between size of implementation and speed of implementation for example) and the VLSI level (like the traditional energy-switching time relationship). The principles of PBL and the relationship between energy and probability of correctness, introduces a new trade-off which may be studied in the context of arithmetic.

In this chapter we extend the constructs rooted in PBL to reason about *probabilistic arithmetic*. To do this, we first consider a technique of addition, and in Section 7.1.1 define a model which characterizes deterministic addition of two polynomials. In Section 7.1.2, we define probabilistic addition and *error vectors*, the elements of which correspond to the probability of correctness of individual probabilistic primitives. The relationship between the cost and the probability of correctness of primitive operations which constitute addition, will be based on the analytical model of CMOS energy consumption described in [35] and summarized in Section 2.4.1. In Section 7.2, we relate the cost of a probabilistic addition to the expected magnitude of error of such an addition. In Section 7.3, we present our main result.

We show that polynomials exist, such that for the same cost of addition under two *error vectors* P, \hat{P} , the relative magnitude of error—the ratio of the expected magnitudes of errors for addition under P, \hat{P} —can grow as $\Omega(2^{n/(2+\epsilon)})$ for some positive $\epsilon \ll 1$. We relate this result to the case when the polynomials are chosen uniformly at random. We study certain interesting contexts, relevant to practical implementation of probabilistic arithmetic in Section 7.4: (i) that of *binning* where the cost of m successive probabilistic primitives are equal and (ii) truncation, where the cost of the probabilistic primitives which compute the t least significant bits is set to 0.

7.1 *Abstracting a Mathematical Model*

We first define a mathematical model for relating the magnitude of error to a particular scheme of *energy investment* and total energy consumption based on ripple-carry of addition. Given the probability of correctness of individual bit-level addition, the *magnitude of error* of the addition of two n bit binary numbers is an attribute of interest.

As an informal example, consider the 8 bit binary addition of two binary numbers A and B , where $A = 01001010$ and $B = 01000110$. The least significant bit (the “first” bit) is written on the right and the most significant bit (the “eighth bit”) on the left. As illustrated in Figure 20(a), we notice that the addition of the second bit *generates* a carry, which is added to the third bit, which in turn produces a carry bit of 1 and hence *propagates* the carry bit, and when added to the fourth bit, produces a 1 as the carry-bit and sets the result of the addition of the fifth bit to 1. We shall call this a *carry chain* of length 3 originating at *position* 2. In this example, there is also a carry chain of length 1, originating at position 7. If this carry originating at position 2 were to be computed incorrectly (say, by implementing the carry generation circuit using operators from PBL), due to the carry chain, the 3rd, 4th and 5th bits would be computed incorrectly, but the error magnitude $2^2 = 4$ is dependent only the position

	$\begin{array}{ccccccc} & \nearrow^1 & & \nearrow^1 & \nearrow^1 & \nearrow^1 & \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & A \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & B \end{array}$		$\begin{array}{ccccccc} & \nearrow^1 & \nearrow^1 & \nearrow^1 & \nearrow^1 & \nearrow^1 & \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & A \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & B \end{array}$		$\begin{array}{ccccccc} & & & \nearrow^1 & \nearrow^1 & \nearrow^1 & \nearrow^1 & \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & A \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & B \end{array}$	
Sum (correct)	1 0 0 1 0 0 0 0		1 0 0 0 0 0 0 0		0 0 1 0 0 0 0 0	
Sum (incorrect)	1 0 0 0 1 1 0 0		0 1 1 1 1 0 0 0		0 0 0 1 1 1 1 0	
Error magnitude	4		8		2	
	(a)		(b)		(c)	

Figure 20: (a) Correct sum of A and B and incorrect sum when the carry at position 2 is computed incorrectly (b) correct sum of A and B and incorrect sum when the carry at position 3 is computed incorrectly (c) correct sum of A and B and incorrect sum when the carry at position 1 is computed incorrectly

at which this carry originated. This illustrates the first attribute of our mathematical model: *The magnitude of error is independent of the length of the carry chain.*

Consider Figure 20 (b) where the length of carry chain is 5, and the case where the carry originating at position 3 is computed incorrectly. The error magnitude is 8. Similarly, consider the case when the inputs are 00010101 and 00001011 (Figure 20(c)). Even though the length of the carry chain is 5—the same length as the case described in Figure 20(b)—since the carry chain originates in a less significant position, the error magnitude is 2 and is lower in this case. This illustrates a second attribute characterized by our mathematical model: *Errors in the carry produced by bits of a higher significance give rise to higher magnitude of error when compared to errors in the carry produced by bits of a lower significance.*

7.1.1 A Mathematical Model for Deterministic Addition

We now define our mathematical model for deterministic addition. Consider a variable x and a polynomial of degree n whose coefficients are chosen from the set $\{0, 1\}$. For example, let A denote a polynomial $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 x^0$ such that $a_i \in \{0, 1\}$. The index of any coefficient (or equivalently, the degree of the corresponding

monomial) will be referred to as its *position*. The evaluation of this polynomial at $x = 2$ denoted by $A(2)$ will be considered to be the *value* of this polynomial. The polynomial A is a binary representation of the integer $A(2)$ and this integer will be referred to as *the integer represented by A* . Given two polynomials A, B , the distance between A and B is defined to be $|A(2) - B(2)|$, which is the absolute value difference between the integers represented by A and B .

Consider two polynomials A, B , of degree n where A represents $a_n x^n + a_{n-1} x^{n-1} + \dots + a_0 x^0$ and B represents $b_n x^n + b_{n-1} x^{n-1} + \dots + b_0 x^0$ where $a_i, b_i \in \{0, 1\}$. Let \mathcal{C} be a Boolean function $\mathcal{C} : \{0, 1\}^3 \rightarrow \{0, 1\}$ where $\mathcal{C}(a, b, c)$ is defined to be $(a \wedge b) \vee (a \wedge c) \vee (b \wedge c)$ where \vee, \wedge, \neg are the Boolean conjunction, disjunction and negation operators respectively, and $a, b, c \in \{0, 1\}$. Then the operator \odot will be defined as follows: Let $C = A \odot B$ where C denotes the polynomial $c_{n+1} x^{n+1} + c_n x^n + \dots + c_1 x^1 + c_0 x^0$, and

$$c_i = \begin{cases} 0 & \text{if } i = 0 \\ \mathcal{C}(a_j, b_j, c_j) & \text{for } 1 \leq i \leq n+1, \text{ where } j = i-1 \end{cases}$$

Here c_i will be referred to as the carry bit computed at position $i-1$. Informally, the coefficients of the polynomial C represent the carry bits produced by the binary addition of the coefficients of the polynomials A and B . Figure 21 illustrates the $A \odot B$ where the coefficients of A , $\langle a_n, a_{n-1}, \dots, a_0 \rangle = \langle 0, 1, 0, 0, 1, 0, 1, 0 \rangle$ and the coefficients of B , $\langle b_n, b_{n-1}, \dots, b_0 \rangle = \langle 0, 1, 0, 0, 0, 1, 1, 0 \rangle$. As a slight variation of Pippenger [152], we define a position i to *generate* a carry if $a_i = b_i = 1$ and a position i *propagates* a carry if exactly one of a_i, b_i equals 1. A carry chain of length k is said to originate at position i if the i^{th} position generates a carry, and $k-1$ subsequent positions propagate a carry and the $k+1^{th}$ subsequent position does not propagate a carry. A set of k consecutive positions $\{i-1, i-2, \dots, i-k\}$ will be referred to as an active-block of size k at position i , if the $(i-k)^{th}$ position does not propagate a carry and all of the remaining $(k-1)$ positions and the i^{th} position propagate a carry. For

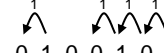
		Carrys
$A = 0x^7 + 1x^6 + 0x^5 + 0x^4 + 1x^3 + 0x^2 + 1x^1 + 0x^0$	0 1 0 0 1 0 1 0	Coefficients of A
$B = 0x^7 + 1x^6 + 0x^5 + 0x^4 + 0x^3 + 1x^2 + 1x^1 + 0x^0$	0 1 0 0 0 1 1 0	Coefficients of B
$C = A \odot B = 0x^8 + 1x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 1x^2 + 0x^1 + 0x^0$	0 1 0 0 1 1 1 0 0	Coefficients of C

Figure 21: The coefficients of $C = A \odot B$

any position j that does not propagate a carry, the active block at position j is the set $\{\}$ (in the special case under which the i^{th} position propagates a carry and the positions $i - 1, i - 2, \dots, 0$ propagate a carry as well, the active block at position i would be $\{i - 1, i - 2, i - 3, \dots, 0\}$).

Given two polynomials A, B , the vector $K = \langle k_n, k_{n-1}, \dots, k_0 \rangle$, where k_i denotes the length of the active block at position i , will be referred to as the “chain vector of A, B ”. Given the polynomials A, B and a position j , $0 \leq j \leq n$, let A_j^0 denote the polynomial whose coefficient at the j^{th} position is 0 and the rest of the coefficients are identical to those of A . Similarly let A_j^1 denote the polynomial whose coefficient at the j^{th} position is 1 and the rest of the coefficients are identical to those of A . The polynomials B_j^0 and B_j^1 are defined similarly. If $C' = A_j^0 \odot B_j^0$ and $C'' = A_j^1 \odot B_j^1$, then the carry produced at position j is said to affect the carry produced at position i , $j < i$ if and only if $c''_{i+1} \neq c'_{i+1}$.

Observation 7.1.1.1 *If A and B are two polynomials of degree n , whose chain vector is $K = \langle k_n, k_{n-1}, \dots, k_0 \rangle$, and if $C = A \odot B$, then c_{i+1} is not affected by the carries generated by the positions $i - k_i - 1, \dots, 0$.*

PROOF. Suppose the carry bit c_{i+1} generated by the i^{th} position is affected by a carry generated by a position $0 \leq j \leq i - k_i - 1$. Then it must be the case that a_m, b_m , where $m = i - k_i$, propagates a carry. Hence, it must be the case that the length of active block at i is greater than k_i . Which is a contradiction. \square

We have characterized the notion of carry coefficients, carry chains and active blocks. Based on these, we will now develop a mathematical model for the sum

of two polynomials. Let \mathcal{S} be a Boolean function $\mathcal{S} : \{0, 1\}^3 \rightarrow \{0, 1\}$ such that $\mathcal{S}(a, b, c) = (a \wedge (\neg b) \wedge (\neg c)) \vee (c \wedge (\neg a) \wedge (\neg b)) \vee (b \wedge (\neg c) \wedge (\neg a)) \vee (a \wedge b \wedge c)$ for $a, b, c \in \{0, 1\}$. If $C = A \odot B$ and C denotes $c_0x^0 + c_1x + c_2x^2 + \cdots + c_{n+1}x^{n+1}$, the sum of the polynomials A and B will be denoted $D = A \oplus B$, where D denotes the polynomial $d_{n+1}x^{n+1} + d_nx^n + \cdots + d_0x^0$, and

$$d_i = \begin{cases} c_i & \text{if } i = n + 1 \\ \mathcal{S}(a_i, b_i, c_i) & \text{for } 0 \leq i < n + 1 \end{cases}$$

We shall refer to d_i as the sum bit computed at the i^{th} position.

7.1.2 A Mathematical Model for Probabilistic Addition

We shall now define a mathematical model for *probabilistic addition*, where the sum of two polynomials may be computed incorrectly. We consider the case where the carry bit computed at any position i is correct with a probability p_i , $1/2 \leq p_i \leq 1$. First, we define a function $\hat{\mathcal{C}}$ such that $\hat{\mathcal{C}}(a, b, c, p) = \mathcal{C}(a, b, c)$ with probability p , and $\hat{\mathcal{C}}(a, b, c, p) = \neg \mathcal{C}(a, b, c)$ with probability $(1 - p)$ where $a, b, c \in \{0, 1\}$ and $1/2 \leq p \leq 1$. Since carry bits at different positions may be computed with different probabilities of correctness, we consider P , the “error vector” of length $n + 1$, which denotes $\langle p_n, p_{n-1}, \dots, p_1, p_0 \rangle$, $1/2 \leq p_i \leq 1$. We will refer to p_i as the local-error at position i . Then the probabilistic carry operator \odot_P is defined as follows: If $\hat{C} = A \odot_P B$, and as before, A, B, \hat{C} represent the polynomials $a_nx^n + a_{n-1}x^{n-1} + \cdots + a_0x^0$, $b_nx^n + b_{n-1}x^{n-1} + \cdots + b_0x^0$ and $\hat{c}_{n+1}x^{n+1} + \hat{c}_nx^n + \cdots + \hat{c}_1x^1 + \hat{c}_0x^0$, respectively

$$\hat{c}_i = \begin{cases} 0 & \text{if } i = 0 \\ \hat{\mathcal{C}}(a_j, b_j, \hat{c}_j, p_j) & \text{for } 1 \leq i \leq n + 1, \text{ where } j = i - 1 \end{cases}$$

The probabilistic sum of the polynomials A and B will be denoted $\hat{D} = A \oplus_P B$ where \hat{D} denotes the polynomial $\hat{d}_{n+1}x^{n+1} + \hat{d}_nx^n + \cdots + \hat{d}_0x^0$, and

$$\hat{d}_i = \begin{cases} \hat{c}_i & \text{if } i = n + 1 \\ \mathcal{S}(a_i, b_i, \hat{c}_i) & \text{for } 0 \leq i < n + 1 \end{cases}$$

Observation 7.1.2.1 *If the error vector P of length $n + 1$ denotes $\langle 1, 1, 1, \dots, 1 \rangle$, then for arbitrary polynomials A and B of degree n , if $D = A \oplus B$ and $\hat{D} = A \oplus_P B$, $D(2) = \hat{D}(2)$.*

The error vector, where the local-error at each position is the same, will be referred to as a uniform error vector. That is, a uniform error vector \hat{P} is a vector $\langle \hat{p}_n, \hat{p}_{n-1}, \dots, \hat{p}_1, \hat{p}_0 \rangle$, where $1/2 \leq \hat{p}_i < 1$ and $\hat{p}_i = \hat{p}_j$ for all $\hat{p}_i, \hat{p}_j \in \hat{P}$.

The energy cost of computing a carry bit is related to its probability of correctness in CMOS based implementations through its operating voltage. In practical implementations, it is infeasible (or very expensive) to implement many distinct levels of supply voltages and hence probabilities of correctness. Hence in practical implementations, we may consider “binning” of error vectors. That is, for any error vector $P = \langle p_n, p_{n-1}, \dots, p_1, p_0 \rangle$ with m bits in each bin, m successive local-errors are equal. That is $p_i = p_j$ if $\lfloor i/m \rfloor = \lfloor j/m \rfloor$.

Another interesting case is addition with *decreased precision*. That is, for probabilistic addition of two polynomials A, B of length n , the least significant t coefficients of the carry polynomial could be guessed uniformly at random from the set $\{0, 1\}$ and rest of the coefficients could be added under a error vector \hat{P} . A error vector $\hat{P} = \langle \hat{p}_n, \hat{p}_{n-1}, \dots, \hat{p}_0 \rangle$, is truncated at t if $\hat{p}_0 = \hat{p}_1 = \dots = \hat{p}_{t-1} = 1/2$ and $\hat{p}_t = \hat{p}_{t+1} = \dots = \hat{p}_n > 1/2$.

7.2 Cost and Magnitude of Error of Probabilistic Addition

The cost of performing an arithmetic operation, like the ripple-carry addition, is an attribute of interest. Typically, the cost of arithmetic operations have been studied

in terms of their implementation costs. In the context of VLSI-based implementations, the area cost of the circuit and the time required to perform the corresponding arithmetic operations (usually taken to be proportional to the “depth” of the circuit) have been the chief cost metrics [18]. However, we depart from these traditional cost metrics and consider the *energy consumption* of the CMOS based implementation as an attribute of interest. Our model is based on the relationship between the energy cost of an operation and its probability of correctness [35, 28, 101].

Definition 10 Energy Cost of Probabilistic Addition: *For any error vector P of length $(n + 1)$ which denotes $\langle p_n, p_{n-1}, \dots, p_1, p_0 \rangle$, where $1/2 \leq p_i < 1$, the energy cost $\mathcal{E}(P)$ of an addition operation \oplus_P is defined to be*

$$\mathcal{E}(P) = \sum_{i=0}^n \log \left(\frac{1}{2 - 2p_i} \right)$$

Two error vectors P and P' are said to be of equal energy (or a *re-investment* of each other) if $\mathcal{E}(P) = \mathcal{E}(P')$.

7.2.1 Error Magnitude of Probabilistic Addition

Given two polynomials A and B of degree n , and an error vector P , if $D = A \oplus B$ is taken to be the correct result of the addition of A and B , then $\hat{D} = A \oplus_P B$ is likely to be incorrect or erroneous. We now seek to quantify this magnitude of error, defined as the distance between \hat{D} and D . That is, if $D = A \oplus B$, and $\hat{D} = A \oplus_P B$ the magnitude of error, $\text{Err}(\hat{D}) = |D(2) - \hat{D}(2)|$. Since the operator \oplus_P is probabilistic, $A \oplus_P B$ is \hat{D}_1 with probability q_1 , \hat{D}_2 with probability q_2 and so on, where $\sum q_i = 1$. The *expected magnitude of error* of $A \oplus_P B$ is the expectation of $\text{Err}(A \oplus_P B)$ which is $q_1|D(2) - \hat{D}_1(2)| + q_2|D(2) - \hat{D}_2(2)| + \dots$ and is denoted by $\text{Experr}(A \oplus_P B)$. We will now bound the expected magnitude of error of $A \oplus_P B$ from above and below.

Lemma 7.2.1 *For any two polynomials A and B and the error vector P , if $K = \langle k_n, k_{n-1}, \dots, k_0 \rangle$ is the chain vector of A, B , the expected magnitude of error*

$$\text{Err}(A \oplus_P B) \leq \sum_{i=0}^n 2^{i+1} \left(1 - \prod_{j=i}^{i-k_i} p_j \right)$$

Furthermore, if for all i , $k_i \leq \lambda$, and $p_i \geq p_j$ whenever $i > j$,

$$\text{Err}(A \oplus_P B) \leq 2^{\lambda+1} + \sum_{i=\lambda}^n 2^{i+1} (1 - (p_{i-\lambda})^{\lambda+1})$$

PROOF. Let $\hat{D} = A \oplus_P B$, $\hat{C} = A \odot_P B$, $D = A \oplus B$ and $C = A \odot B$. Then the magnitude of error, $\text{Err}(A \oplus_P B)$, is

$$\text{Err}(A \oplus_P B) = \left| D(2) - \hat{D}(2) \right| = \left| \sum_{i=0}^{n+1} d_i 2^i - \sum_{i=0}^{n+1} \hat{d}_i 2^i \right| = \left| \sum_{i=0}^{n+1} 2^i (d_i - \hat{d}_i) \right|$$

and hence

$$\text{Err}(A \oplus_P B) \leq \sum_{i=0}^{n+1} 2^i |d_i - \hat{d}_i|$$

Since $d_i = \mathcal{S}(a_i, b_i, c_i)$ and $\hat{d}_i = \mathcal{S}(a_i, b_i, \hat{c}_i)$, it follows that $d_i \neq \hat{d}_i$ if and only if $c_i \neq \hat{c}_i$. Hence $\sum_{i=0}^{n+1} 2^i |d_i - \hat{d}_i| = \sum_{i=0}^{n+1} 2^i |c_i - \hat{c}_i|$. Therefore,

$$\text{Err}(A \oplus_P B) \leq \sum_{i=0}^{n+1} |2^i (c_i - \hat{c}_i)| = \sum_{i=1}^{n+1} 2^i |c_i - \hat{c}_i| \quad \text{since } c_0 = \hat{c}_0$$

The expected magnitude of error $\text{Experr}(A \oplus_P B)$ over many probabilistic additions of A and B is

$$\text{Experr}(A \oplus_P B) = \text{Exp}[\text{Err}(P)] \leq \text{Exp} \left[\sum_{i=1}^{n+1} 2^i |c_i - \hat{c}_i| \right] = \sum_{i=1}^{n+1} (2^i \text{Exp} [|c_i - \hat{c}_i|])$$

Whenever $c_i = \hat{c}_i$, $|(c_i - \hat{c}_i)| = 0$ and whenever $c_i \neq \hat{c}_i$, $|(c_i - \hat{c}_i)| = 1$. Therefore, if r_i is the probability that $c_i \neq \hat{c}_i$, $\text{Exp}[|c_i - \hat{c}_i|] = r_i$. Therefore,

$$\text{Experr}(A \oplus_P B) \leq \sum_{i=1}^{n+1} 2^i r_i \quad (14)$$

From Observation 7.1.1.1, it follows that $\hat{c}_{i+1} = c_{i+1}$ if for all $i \leq j \leq i - k_i$, $\hat{\mathcal{C}}(a_j, b_j, \hat{c}_j, p_j) = \mathcal{C}(a_j, b_j, c_j)$. Hence, the probability that $c_{i+1} = \hat{c}_{i+1}$ is at least $\prod_{j=i}^{i-k_i} p_j$. Therefore, it follows that $r_{i+1} \leq 1 - \prod_{j=i}^{i-k_i} p_j$, and from (14)

$$\text{Experr}(A \oplus_P B) \leq \sum_{i=0}^n 2^{i+1} \left(1 - \prod_{j=i}^{i-k_i} p_j \right) \quad (15)$$

since $k_i \leq \lambda$

$$\text{Experr}(A \oplus_P B) \leq \sum_{i=0}^{\lambda-1} 2^{i+1} \left(1 - \prod_{j=i}^0 p_j \right) + \sum_{i=\lambda}^n 2^{i+1} \left(1 - \prod_{j=i}^{i-\lambda} p_j \right)$$

since $p_i \geq p_j$ whenever $i > j$

$$\text{Experr}(A \oplus_P B) \leq 2^{\lambda+1} + \sum_{i=\lambda}^n 2^{i+1} (1 - (p_{i-\lambda})^{\lambda+1}) \quad (16)$$

□

Lemma 7.2.2 *For any two polynomials A and B and the error vector P , the expected magnitude of error of \oplus_P is at least*

$$2^{n+1} (1 - p_n) \prod_{i=0}^{n-1} p_i$$

PROOF. Let $C = A \odot B$, $D = A \oplus B$, $\hat{C} = A \odot_P B$ and $\hat{D} = A \oplus_P B$. We know from Markov's inequality [57] that

$$\text{Experr}(A \oplus_P B) = \text{Exp}[\text{Err}(A \oplus_P B)] \geq 2^{n+1} \Pr[\text{Err}(A \oplus_P B) \geq 2^{n+1}]$$

where $\Pr[\text{Err}(A \oplus_P B) \geq 2^{n+1}]$ is the probability that $|D(2) - \hat{D}(2)|$ is greater than or equal to 2^{n+1} . Let E be the event such that, for all $0 \leq i \leq n$, $\hat{d}_i = d_i$ and $\hat{d}_{n+1} \neq d_{n+1}$ and let $\Pr[E]$ be the probability that event E occurs. Then trivially,

$$\Pr [\text{Err}(A \oplus_P B) \geq 2^{n+1}] \geq \Pr[E]$$

Since $d_i = \mathcal{S}(a_i, b_i, c_i)$ and $\hat{d}_i = \mathcal{S}(a_i, b_i, \hat{c}_i)$, for $0 \leq i \leq n$, $d_i \neq \hat{d}_i$ if and only if $c_i \neq \hat{c}_i$. Hence, if E' is the event that for all $0 \leq i \leq n$, $\hat{c}_i = c_i$ and $\hat{c}_{n+1} \neq c_{n+1}$ and if $\Pr[E']$ is the probability that event E' occurs, then $\Pr[E] = \Pr[E']$. Therefore,

$$\Pr [\text{Err}(A \oplus_P B) \geq 2^{n+1}] \geq \Pr[E']$$

Since $\hat{c}_{i+1} = \hat{\mathcal{C}}(a_i, b_i, \hat{c}_i, p_i)$, and since whenever $\hat{c}_i = c_i$, the probability that $\hat{\mathcal{C}}(a_i, b_i, \hat{c}_i, p_i) = \hat{\mathcal{C}}(a_i, b_i, c_i)$ is p_i ,

$$\Pr[E'] = (1 - p_n) \prod_{i=0}^{n-1} p_i$$

Therefore, $\text{Experr}(A \oplus_P B) \geq 2^{n+1}(1 - p_n) \prod_{i=0}^{n-1} p_i \quad \square$

From Lemma 7.2.2 it is immediate that

Corollary 11 *For any two polynomials A and B and the uniform error vector \hat{P} , the expected magnitude of error of $\oplus_{\hat{P}}$ is at least*

$$2^{n+1}(1 - \hat{p}_0)\hat{p}_0^n$$

7.3 Relative Magnitude of Error

From Definition 10 we note that the energy cost of a probabilistic addition is determined by the error vector P . In addition, from Lemma 7.2.1 and Lemma 7.2.2, it is evident that the expected magnitude of error of any probabilistic addition of two polynomials A, B , is determined by the error vector P and the chain vector K . We show that given two error vectors P, P' such that they are re-investments of each

other, the expected magnitude of error of the addition of two polynomials differs. We quantify this difference as the *relative magnitude of error*. That is, if A, B are two polynomials of degree n and P, P' are error vectors of length $n + 1$ such that $\mathcal{E}(P) = \mathcal{E}(P')$, the relative magnitude of error is

$$\text{RE}(P, P') = \max \left\{ \frac{\text{Experr}(A \oplus_P B)}{\text{Experr}(A \oplus_{P'} B)}, \frac{\text{Experr}(A \oplus_{P'} B)}{\text{Experr}(A \oplus_P B)} \right\}$$

If \mathcal{P} is a set of error vectors of length $n + 1$ such that for all $P, P' \in \mathcal{P}$, $\mathcal{E}(P) = \mathcal{E}(P')$, then the maximum relative magnitude of error is defined to be

$$\Gamma_n = \max_{P, P' \in \mathcal{P}} \{\text{RE}(P, P')\}$$

We show that there exists polynomials A, B of degree n such that as $\Gamma_n = \Omega(2^{n/(2+\epsilon)})$ for some positive $\epsilon \ll 1$. We consider a pair of polynomials A, B such that each element of their chain vector is of constant length, and a pair of error vectors P, P' of equal energy such that P' is the uniform error vector. We use Lemma 7.2.1 and Lemma 7.2.2 to bound the expected magnitude of error from below and above to get an estimate of Γ_n . We define an *exponential error vector* P to be a error vector such that $P = \langle p_n, p_{n-1}, \dots, p_0 \rangle$ where $p_i = 1 - (1/2)^{i+1}$. Then

Lemma 7.3.1 *If P is an exponential error vector of length $n+1$ and $\hat{P} = \langle \hat{p}_n, \hat{p}_{n-1}, \dots, \hat{p}_0 \rangle$ is a uniform error vector of length $n+1$ such that $\mathcal{E}(P) = \mathcal{E}(\hat{P})$, then for all $0 \leq i \leq n$, $\hat{p}_i = 1 - 1/2^{(1+n/2)}$*

PROOF. From the definition of an exponential error vector, we know that P denotes the vector $\langle p_n, p_{n-1}, \dots, p_0 \rangle$ where $p_i = 1 - (1/2)^{i+1}$. Hence, $\mathcal{E}(P) = \sum_{i=0}^n \log(1/(2 - 2p_i))$ and therefore,

$$\mathcal{E}(P) = \sum_{i=0}^n \log \left(\frac{1}{2 - 2 \left(1 - \frac{1}{2^{i+1}}\right)} \right) = \sum_{i=0}^n \log(2^i)$$

hence

$$\mathcal{E}(P) = \frac{(n)(n+1)}{2} \tag{17}$$

Since in a uniform error vector \hat{P} , $\hat{p}_i = \hat{p}_j$ for all $\hat{p}_i, \hat{p}_j \in \hat{P}$, the energy cost of probabilistic addition under a uniform error vector is

$$\mathcal{E}(\hat{P}) = \sum_{i=0}^n \log \left(\frac{1}{2 - 2\hat{p}_i} \right) = (n+1) \log \left(\frac{1}{2 - 2\hat{p}_0} \right) \quad (18)$$

From the fact that $\mathcal{E}(\hat{P}) = \mathcal{E}(P)$, (17) and (18),

$$(n+1) \log \left(\frac{1}{2 - 2\hat{p}_0} \right) = \frac{(n)(n+1)}{2}$$

or

$$\frac{1}{2 - 2\hat{p}_0} = 2^{\frac{n}{2}}$$

and hence

$$\hat{p}_0 = 1 - \left(\frac{1}{2} \right)^{(1+\frac{n}{2})}$$

therefore, for all $0 \leq i \leq n$, $\hat{p}_i = 1 - (1/2)^{1+(n/2)} \quad \square$

Theorem 12 *There exist polynomials A, B of degree n such that the relative magnitude of error $\Gamma_n = \Omega(2^{n/(2+\epsilon)})$ for some positive $\epsilon \ll 1$.*

PROOF. Consider two polynomials A, B of degree n such that their chain vector K is of the form $\langle k_n, k_{n-1}, k_{n-2}, \dots, k_0 \rangle$ such that for all $0 \leq i \leq n$, $k_i \leq 3$. Such polynomials exist, $a_i = 1, b_i = 0$ for all $0 \leq i \leq n$ is a trivial example. Let P be an exponential error vector of length $n+1$. Then from Lemma 7.2.1 Equation (16),

$$\text{Experr}(A \oplus_P B) \leq 16 + \sum_{i=3}^n 2^{i+1} (1 - (p_{i-3})^4)$$

Since $p_i = 1 - (1/2)^{(i+1)}$, it follows that for $3 \leq i \leq n$, $p_{i-3} = 1 - (1/2)^{(i-2)}$. Expanding $(p_{i-3})^4$ using the Taylor series and approximating,

$$\text{Experr}(A \oplus_P B) < 16 + \sum_{i=3}^n 2^{i+1} \left(1 - \left(1 - \frac{4}{2^{i-2}} \right) \right) = 16 + 32(n-2) \quad (19)$$

If $\hat{P} = \langle \hat{p}_n, \hat{p}_{n-1}, \dots, \hat{p}_0 \rangle$ is a uniform error vector such that $\mathcal{E}(\hat{P}) = \mathcal{E}(P)$, from Lemma 7.3.1, for all $0 \leq i \leq n$, $\hat{p}_i = 1 - 1/2^{(1+n/2)}$. Furthermore, from Corollary 11, $\text{Experr}(A \oplus_{\hat{P}} B) > 2^{n+1}(1 - \hat{p}_0)\hat{p}_0^n$ and therefore,

$$\text{Experr}(A \oplus_{\hat{P}} B) \geq 2^{n+1} \left(\frac{1}{2} \right)^{1+\frac{n}{2}} \left(1 - \frac{1}{2^{(1+\frac{n}{2})}} \right)^n$$

expanding using Taylor series

$$\text{Experr}(A \oplus_{\hat{P}} B) > 2^{\frac{n}{2}} \left(1 - \frac{n}{2^{(1+\frac{n}{2})}} \right)$$

and hence

$$\text{Experr}(A \oplus_{\hat{P}} B) > \sqrt{2^n} - \frac{n}{2} \quad (20)$$

Recall that the relative magnitude of error

$$\Gamma_n \geq \max \left\{ \frac{\text{Experr}(A \oplus_P B)}{\text{Experr}(A \oplus_{\hat{P}} B)}, \frac{\text{Experr}(A \oplus_{\hat{P}} B)}{\text{Experr}(A \oplus_P B)} \right\}$$

hence

$$\Gamma_n \geq \frac{\text{Experr}(A \oplus_{\hat{P}} B)}{\text{Experr}(A \oplus_P B)} > \frac{\sqrt{2^n} - \frac{n}{2}}{16 + 32(n-2)}$$

Hence $\Gamma_n = \Omega(2^{n/(2+\epsilon)})$ \square

It is immediate from the theorem that

Corollary 13 *If A and B are polynomials and $K = \langle k_n, k_{n-1}, k_{n-2}, \dots, k_0 \rangle$ is the chain vector of A, B such that $k_i \leq \log(n)$, and P is the exponential error vector, $\text{Experr}(A \oplus_{\hat{P}} B) = O(n^2 \log(n))$.*

We note that if the coefficients of A and B are chosen uniformly at random from the set $\{0, 1\}$, and if K is the chain vector of A, B , then $\Pr[k_i = c]$, the probability that the length of active block at position i is c is $1/2^c$. Hence the expected length of an active block, or equivalently, the expected value of k_i , is 2.

Lemma 7.3.2 *If P is an exponential error vector of length n and E_P is the expected magnitude of error of the probabilistic sum of polynomials A and B of length n , whose chosen uniformly at random from the set $\{0, 1\}$, then $E_P = O(n^3)$.*

PROOF. Given polynomials A, B , if K is the chain vector of A, B , we know from Lemma 7.2.1 that

$$\text{Err}(A \oplus_P B) \leq \sum_{i=0}^n 2^{i+1} \left(1 - \prod_{j=i}^{i-k_i} p_j \right)$$

Hence, it follows that

$$E_P \leq \sum_{i=0}^n 2^{i+1} \left(\sum_{m=0}^i \Pr[k_i = m] \left(1 - \prod_{j=i}^{i-m} p_j \right) \right)$$

since $\Pr[k_i = c] = 1/2^c$,

$$\begin{aligned} E_P &\leq \sum_{i=0}^n 2^{i+1} \left(\sum_{m=0}^i \frac{1}{2^m} \left(1 - \prod_{j=i}^{i-m} p_j \right) \right) \\ E_P &< \sum_{i=0}^n 2^{i+1} \left(\sum_{m=0}^i \frac{1}{2^m} (1 - (p_{i-m})^{m+1}) \right) \end{aligned}$$

expanding using Taylor series

$$E_P < \sum_{i=0}^n 2^{i+1} \left(\sum_{m=0}^i \frac{1}{2^m} \left(\frac{m+1}{2^{i-m+1}} \right) \right)$$

hence

$$E_P < \frac{(n+1)^2(n+2)}{2} = O(n^3)$$

□

7.4 Some Practical Considerations

We now consider the case of “binned” error vectors. We recall that any error vector $P = \langle p_n, p_{n-1}, \dots, p_1, p_0 \rangle$ is defined to be binned with n/m bins, if m successive local-errors are equal. A exponential error vector of length n with n/m bins is defined to be an error vector $P = \langle p_n, p_{n-1}, \dots, p_0 \rangle$ such that m divides $n + 1$ and $p_i = 1 - 1/(2^{m\lfloor i/m \rfloor + 1})$.

We analyze binning and show that for addition under the exponential error vector P with $n/\log(n)$ bins, the expected magnitude of error is $O(n^2 \log(n))$. Furthermore, if \hat{P} is a uniform error vector such that $\mathcal{E}(P) = \mathcal{E}(\hat{P})$, we show that the expected magnitude of error for addition under \hat{P} is $\Omega(2^{(n/(2+\epsilon))})$ for some positive $\epsilon < 1$.

Claim 7.4.0.1 *If P is an exponential error vector of length n with n/m bins whereas $\hat{P} = \langle \hat{p}_n, \hat{p}_{n-1}, \dots, \hat{p}_0 \rangle$ is a uniform error vector of length n such that $\mathcal{E}(P) = \mathcal{E}(\hat{P})$, then for all $0 \leq i \leq n$, $\hat{p}_i = 1 - 1/2^{(1+t/2)}$ where $t = n(n+m)/(n+1)$.*

PROOF. We know that $\mathcal{E}(P) = \sum_{i=0}^n \log(1/(2 - 2p_i))$ and $p_i = 1 - 1/(2^{m\lfloor i/m \rfloor + 1})$. Hence,

$$\mathcal{E}(P) = \sum_{i=0}^n \log \left(\frac{1}{2 - 2 \left(1 - \frac{1}{2^{m\lfloor \frac{i}{m} \rfloor + 1}} \right)} \right) = \sum_{j=0}^{n/m} m \log(2^{mj})$$

hence,

$$\mathcal{E}(P) = \frac{n(n+m)}{2} \tag{21}$$

From the fact that $\mathcal{E}(\hat{P}) = \mathcal{E}(P)$, (21) and (18),

$$(n+1) \log \left(\frac{1}{2 - 2\hat{p}_0} \right) = \frac{(n)(n+m)}{2}$$

or

$$\hat{p}_0 = 1 - \left(\frac{1}{2} \right)^{(1+\frac{t}{2})}$$

where $t = n(n + m)/(n + 1)$. Therefore, for all $0 \leq i \leq n$, $\hat{p}_i = 1 - (1/2)^{1+(t/2)}$ \square

Lemma 7.4.1 *If P is a exponential error vector of length n with n/m bins, A and B are polynomials of degree $n - 1$ and $K = \langle k_n, k_{n-1}, k_{n-2}, \dots, k_0 \rangle$ is the chain vector of A, B such that $k_i \leq m$, then $\text{Experr}(A \oplus_{\hat{P}} B)$ is $O(nm2^m)$. Furthermore, if $m = \log(n)$, $\text{Experr}(A \oplus_{\hat{P}} B)$ is $O(n^2 \log(n))$.*

PROOF. From Lemma 7.2.1, (16) the expected magnitude of error

$$\text{Experr}(A \oplus_P B) < 2^{m+1} + \sum_{i=m}^n 2^{i+1} (1 - (p_{i-m})^{m+1})$$

since $p_i = 1 - 1/(2^{m\lfloor i/m \rfloor + 1})$

$$\begin{aligned} \text{Experr}(A \oplus_P B) &< 2^{m+1} + m \sum_{j=1}^{n/m} 2^{mj+1} \left(1 - \left(1 - \frac{1}{2^{mj-m+1}} \right)^{m+1} \right) \\ &< 2^{m+1} + m \sum_{j=1}^{n/m} 2^{mj+1} \left(1 - \left(1 - \frac{m+1}{2^{mj-m+1}} \right) \right) \end{aligned}$$

hence,

$$\text{Experr}(A \oplus_P B) < 2^{m+1} + (n)(m+1)2^m$$

Hence, it is immediate that for $m = \log(n)$, $\text{Experr}(A \oplus_P B)$ is $O(n^2 \log(n))$. \square

Lemma 7.4.2 *If P is a exponential error vector of length n with n/m bins, whereas $\hat{P} = \langle \hat{p}_n, \hat{p}_{n-1}, \dots, \hat{p}_0 \rangle$ is a uniform error vector of length n , such that $\mathcal{E}(P) = \mathcal{E}(\hat{P})$, and if A and B are polynomials of degree $n - 1$, $\text{Experr}(A \oplus_{\hat{P}} B)$ is $\Omega(2^{(n-m)/2})$. Furthermore if $m = \log(n)$, $\text{Experr}(A \oplus_{\hat{P}} B)$ is $\Omega(2^{(n/(2+\epsilon))})$ for some positive $\epsilon < 1$.*

PROOF. From Claim 7.4.0.1, for all $0 \leq i \leq n$, $\hat{p}_i = 1 - 1/2^{(1+t/2)}$ where $t =$

$n(n+m)/(n+1)$. From Corollary 11

$$\text{Experr}(A \oplus_{\hat{P}} B) > 2^{n+1} \left(\frac{1}{2^{1+\frac{t}{2}}} \right) \left(1 - \frac{1}{2^{1+\frac{t}{2}}} \right)^n$$

since $(n+m) > n(n+m)/(n+1)$,

$$\text{Experr}(A \oplus_{\hat{P}} B) > 2^{n+1} \left(\frac{1}{2^{1+\frac{(n+m)}{2}}} \right) \left(1 - \frac{n}{2^{1+\frac{t}{2}}} \right)$$

hence,

$$\text{Experr}(A \oplus_{\hat{P}} B) > 2^{\frac{n-m}{2}} \left(1 - \frac{n}{2^{1+\frac{t}{2}}} \right)$$

Since $m \geq 1$, $(n-m)/2 < n(n+m)/(2(n+1))$. Hence,

$$\text{Experr}(A \oplus_{\hat{P}} B) > 2^{\frac{n-m}{2}} - n$$

Hence, it is immediate that when $m = \log(n)$, $\text{Experr}(A \oplus_{\hat{P}} B)$ is $\Omega(2^{(n/(2+\epsilon))})$. \square

7.4.1 Truncation in Probabilistic Arithmetic

Another implementation alternative is addition with *decreased precision*. That is for probabilistic addition of two polynomials A, B of length n , the least significant t coefficients of the sum could be guessed uniformly at random from the set $\{0, 1\}$ and rest of the coefficients could be added under a uniform error vector \hat{P} , thereby investing more energy in bits of a higher significance when compared to bits of a lower significance.

Consider a exponential error vector P of length $n+1$ and a uniform error vector $\hat{P} = \langle \hat{p}_n, \hat{p}_{n-1}, \dots, \hat{p}_0 \rangle$, of length $n+1$, where $\hat{p}_0 = \hat{p}_1 = \dots = \hat{p}_{t-1} = 1/2$, $\hat{p}_t = \hat{p}_{t+1} = \dots = \hat{p}_n$ and $\mathcal{E}(P) = \mathcal{E}(\hat{P})$. In the uniform error vector case, the least significant t coefficients of $A \oplus_P B$ are correct with a probability $1/2$ and hence, can be guessed from the set $\{0, 1\}$. Hence, we will refer to \hat{P} as a uniform error vector *truncated* at t . We show that for a truncated uniform error vector where constant number of elements are truncated, the expected magnitude of error is $\Omega(2^{n/c})$ for some positive constant c .

We know from (17) that $\mathcal{E}(P) = \frac{(n)(n+1)}{2}$. Since $\mathcal{E}(P) = \mathcal{E}(\hat{P})$,

Claim 7.4.1.1 For $t \leq i \leq n$, $\hat{p}_i = 1 - (1/2)^s$ where $s = 1 + (n)(n+1)/(2(n-t+1))$.

PROOF. We know that

$$\begin{aligned}\mathcal{E}(\hat{P}) &= \sum_{i=0}^n \log \left(\frac{1}{2 - 2\hat{p}_i} \right) \\ &= \sum_{i=t}^n \log \left(\frac{1}{2 - 2\hat{p}_i} \right) \\ &= (n - t + 1) \log \left(\frac{1}{2 - 2\hat{p}_i} \right)\end{aligned}$$

Since $\mathcal{E}(P) = \mathcal{E}(\hat{P})$, for $t \leq i \leq n$, $\hat{p}_i = 1 - (1/2)^s$ where $s = 1 + (n)(n+1)/(2(n-t+1))$. \square

Lemma 7.4.3 For any two polynomials A, B of degree $n-1$, and a exponential error vector P , if \hat{P} is a uniform error vector truncated at t such that $\mathcal{E}(P) = \mathcal{E}(\hat{P})$, then $\text{Experr}(A \oplus_{\hat{P}} B)$ is $\Omega(2^{n/c})$ where t, c are positive constants.

PROOF. From Lemma 7.2.2,

$$\begin{aligned}\text{Experr}(A \oplus_P B) &\geq 2^{n+1} (1 - \hat{p}_n) \prod_{i=0}^{n-1} \hat{p}_i \\ \text{Experr}(A \oplus_P B) &\geq 2^{n+1} \frac{1}{2^{s+t}} \left(1 - \frac{1}{2^s} \right)^{n-t}\end{aligned}$$

where $s = 1 + (n)(n+1)/(2(n-t+1))$. Hence,

$$\begin{aligned}\text{Experr}(A \oplus_P B) &\geq 2^{n+1-s-t} \left(1 - \frac{1}{2^s} \right)^{n-t} \\ \text{Experr}(A \oplus_P B) &\geq 2^{u/(2w)} - \frac{n-t}{2^{v/w-1}}\end{aligned}$$

where $u = n^2 + n - 4nt + 2t^2$, $v = t(n+1) - t^2$, $w = (n-t+1)$. Hence, $\text{Experr}(A \oplus_P B) = \Omega(2^{n/c})$, for some positive constant c . \square

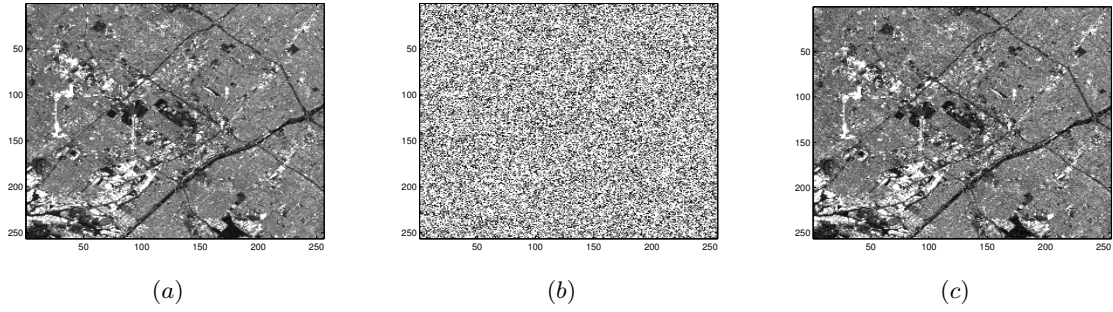


Figure 22: Application level impact of probabilistic arithmetic on SAR (a) conventional error free operation, (b) uniform voltage scaling yielding 2.5x energy savings (c) BIVOS based probabilistic arithmetic yielding an acceptable image and 5.6x energy savings (from [66])

7.5 *Case Study of Probabilistic Arithmetic in Digital Signal Processing*

The theoretical foundations developed in prior sections provide an alternate approach towards realizing energy efficient digital signal processing DSP by implementing DSP primitives through probabilistic arithmetic. In particular, the result in Lemma 7.3.2 shows that for polynomials of length n , whose coefficients are chosen uniformly at random from the set $\{0, 1\}$, the expected magnitude of error for probabilistic addition using the exponential error vector is $O(n^3)$. In this context, George et. al. [66] have demonstrated significant energy savings in the synthetic aperture radar processing algorithm by utilizing probabilistic arithmetic implemented using *probabilistic* CMOS (or PCMOs) technology. In this section, we briefly summarize the results reported by George et. al. [66].

In the domain of CMOS, the energy consumption of a primitive switch is related to its supply voltage. When these switches are used as building blocks to implement (Boolean) logic primitives, the probability of correct switching is determined by the supply voltage of the constituent switches. This provides the basis for the trade-off between energy and probability of correctness of CMOS based implementation of Boolean logic primitives (gates). This relationship between the switching

energy and the probability of correct switching in CMOS is derived from first principles by Cheemalavagu et al. [34]. When these probabilistic gates are used to implement arithmetic primitives, such as a ripple carry adder, based on the insight from Theorem 12, we know that probabilistic addition under an exponential error vector yields a lesser expected magnitude of error when compared to probabilistic addition under a uniform error vector. Such an implementation of a ripple carry adder in CMOS, where bits of higher significance are operated at higher supply voltages, (and hence, have a higher probability of correctness) will be referred to as a *biased voltage scaled implementation* or a BIVOS implementation. On the other hand, those implementations where the probability of correctness of full adders are the same—irrespective of the significance of the bit they compute—will be referred to as the conventional uniformly scaled voltage implementation.

To demonstrate the value of probabilistic arithmetic, George et. al. have considered the synthetic aperture radar (SAR) application [167] using a satellite image of Los Angeles County for experimentation. In this context BIVOS based probabilistic arithmetic implementation results in significant energy savings with minimal impact on application quality. This is illustrated in Figure 22 where Figure 22(a) is the image derived from conventional (correct) processing. Figure 22(c) is the image obtained by processing with probabilistic arithmetic operations with BIVOS scheme, which yields 5.6x in energy saving with no perceptible degradation. The image in Figure 22(b) is obtained by processing with probabilistic arithmetic operations with uniformly scaled voltage implementation for 2.5x in energy savings.

Thus, this empirical study of probabilistic arithmetic shows that (i) probabilistic arithmetic can be an effective way for energy efficient computing and (ii) the theoretical study of investment techniques can help improve the quality of solution for a fixed amount of energy savings.

CHAPTER VIII

REMARKS AND FUTURE DIRECTIONS

We wish to note that PBL was developed as a logic throughout this work, thus diverging from the classical approach of treating Boole’s work on two-valued logic as an algebra with a concomitant—often unspecified—axiomatization. This choice was deliberate since we wished to introduce simple and explicit semantics to our particular approach to introducing probability into logic on the one hand, and furthermore to cast it in a form that is natural to the two application domains of interest: computer science and electrical engineering. Recall that our own interest stemmed significantly from the generally expected trend that gates and switches used to design circuits and computing architectures are going to be probabilistic, since deterministic designs are unlikely to be feasible as device (transistor) sizes approach ten nanometers.

We note that PBL is a significantly simple logic since it does not admit quantification. So, a reasonable approach is to try and compare PBL to a suitable subset of the richer logics which use the predicate calculus as a basis. The essence of the difference between the previous approaches (which can be broadly referred to as *sentential probability* logics) on the one hand and PBL on the other, can be understood through the event set semantics (Section 3.2). In particular, we draw the reader’s attention to Observation 3.2.2.1 which clearly identifies the effect of the probability parameter p in an identity of the form $F \equiv (F' \vee_p F'')$. The main point worth noting here is that the event set of F is dependent on the parameter p associated with the operator \vee_p , *in addition* to the event sets associated with its constituent probabilistic formulae F' and F'' . *It is important to note that this is not true of the previous approaches—in these cases, the operators are always deterministic.* Thus, based on previous approaches,

the probability associated with a formula of the form $G \equiv (G' \vee G'')$ would entirely depend on the probabilities associated with the two sub-formulae G', G'' and *not* on the operator \vee .

Our work on PBL can be extended in the intellectual and practical contexts and specific directions for future inquiry are expanded upon in [31]. In particular, the case of a logic wherein each operator is associated with a probability interval, as opposed to a definite probability value, would be of interest. We note that this extension is also of considerable interest in the context of integrated circuit (IC) design. Currently, logic synthesis is an extremely successful technology, where, given an input specification as a formula, a (heuristically) optimized circuit is produced, based on VLSI cost considerations [122]. Extending this to PBL to enable automated circuit synthesis would be of great value. While circuit synthesis is interesting in its own right, advances in the theory and practice of verification of probabilistic circuits is indispensable for their large scale adoption.

The event set semantics of PBL suggest a probability attribute for each operator (or gate) based on a set of trials associated with it. This implicitly connotes an interpretation where the set of trials resulting in the events occur over time. However, in the context of VLSI circuits, the observed statistical variations may occur spatially across the transistors or gates on the surface of the chip, whereas individual transistors or gates, once manufactured, need not exhibit randomness. While it is straightforward to reinterpret the concept of an event set and the associated semantics to the case of spatial variations, given its importance to the design of integrated circuits, detailing this extension will be of immense value.

We also extended the notion of implicitly probabilistic operations from the domain of logic to arithmetic, by incorporating considerations of probability into arithmetic operations and demonstrated that the energy or more generally, cost advantages persist. In this context, we wish to recall two important results from our development of

PBL and draw analogies: (i) For an input assignment I to any PBF F , its probability of correctness is an attribute of interest and can be quantified through the model (or event set) of F_I . The “correct” truth value of F is taken to be the truth value of the “underlying classical Boolean formula” (or the deterministic restriction) of F for the input assignment I . In the context of probabilistic arithmetic—specifically addition—the *magnitude of correctness* (rather than the probability of correctness) is an attribute of interest. For example, if the least significant bit of 8 bit addition were to be computed incorrectly, the magnitude of error would be 1, whereas if the most significant bit were to be computed incorrectly, the magnitude of error would be 128. (ii) Since PBL does not preserve associativity, for any PBF F , reassociations of F may alter the probability of correctness of F_I . Analogous to the non-associativity result in PBL, for the same energy investment, we have shown that if the energy is invested uniformly across all primitive logical operations of an addition, the expected magnitude of error grows as $\Omega(\sqrt{2^n})$ whereas in the non-uniform investment case it grows as $O(n^2)$. It is thereby indicated that *re-investment* of energy in a circuit that realizes arithmetic operations (analogous to reassociation of probabilistic Boolean formulae which realize logical operations in the PBL context) is likely to yield substantial improvements in the expected magnitude of error.

While we have considered probabilistic primitives and studied the case of ripple carry adders, the impact of these primitives on alternate architectures for addition, subtraction over conventional and alternate number representations, such as the Kogge-Stone adder [96], various other forms of carry look ahead adders and carry save adders [143] could be studied. In this context, new adder structures which optimize area and speed in the presence of our energy-correctness trade-offs could be investigated. Finally, based on probabilistic primitives and adders based on these primitives, structures which implement multiplication operations ought to be investigated.

Given such a characterization of arithmetic operations such as addition and multiplication, the trade-off between energy and magnitude of correctness may be extended to the algorithmic level. Classically, for algorithms which involve extensive arithmetic operations—the domain of digital signal processing (DSP) is a good example—and those which are synthesized into physical computational structures as application-specific integrated circuits (ASICs), the notion of Winograd’s *arithmetic complexity* is of interest [196]. Arithmetic complexity seeks to quantify the number of arithmetic operations performed by a particular algorithm and hence is a good indicator of the size, execution time and energy consumption of circuits which implement arithmetic algorithms. While characterizing the arithmetic complexity of an algorithm, the fast Fourier transform (FFT) for example, all arithmetic operations of the same type—say k bit multiplication—are considered to cost the same. Furthermore, there is no notion of a probability of correctness associated with the constituent arithmetic operations.

In the context of probabilistic arithmetic, which has two attributes (i) a non uniform measure of (energy) cost across operations of the same type and (ii) a novel trade-off between energy cost and magnitude of correctness, an extension of arithmetic complexity which incorporates these additional considerations would enable entirely new designs by exposing this additional trade-off. Such a complexity measure could characterize the energy consumption as well as the magnitude of correctness at the algorithmic level, and could enable the investigation of energy efficient algorithms which trade energy for quality of solution.

The principles of PBL, the PSOC architecture based on PBL and probabilistic arithmetic, we believe, have a broad intellectual appeal and in a practical context, demonstrate the utility of probabilistic primitives in the design and implementation of computing systems. Though our empirical demonstrations were based on noise-susceptible CMOS devices, these principles are applicable in the context of probabilistic physical

primitives, in non-CMOS materials as well as in CMOS devices where erroneous behavior is not due to noise susceptibility. To illustrate this trade-off between energy and quality of solution in current-day technology generations, probabilistic arithmetic in the context of voltage-overscaled implementations of arithmetic structures has been studied [30]. The erroneous behavior in this context is induced by aggressive voltage scaling, and hence the errors are induced due to propagation delays. A higher investment in energy implies operating the corresponding circuit elements at a higher voltage, and would translate into faster propagation of signal values and hence lower probability of error.

A PSOC architecture can be thought of as the physical implementation of a probabilistic automaton, where the deterministic bookkeeping operations and state is maintained in the deterministic host processor, and the probabilistic state transition functions are computed in the probabilistic co-processor. Thus, the energy efficiency demonstrated through PSOC architectures, when compared to designs that implement these probabilistic applications in an explicitly probabilistic manner (by employing pseudo-random bits), serves as empirical evidence for the energy efficiency characterized in the theoretical setting of the probabilistic automata.

To demonstrate the utility of PSOC and probabilistic arithmetic, we have considered instances from the domain of embedded and probabilistic applications. Applications, in general, can be classified into three categories: (i) applications which benefit from (or harness) probabilistic behavior at the device level naturally (ii) applications that can tolerate (and trade-off) probabilistic behavior at the device level (but do not need such behavior naturally) and (iii) applications which cannot tolerate probabilistic behavior at all. It is interesting to note that logic elements which exhibit probabilistic behavior can be utilized for the third category of applications as well. It is conceivable that they would include either temporal or spatial redundancy with error correcting techniques [115]. In this context, redundancy and error correcting

techniques would prove impractical if the amount of redundancy negates advantages gained due to technology scaling. Quantifying the overheads imposed by such techniques and delineating the contexts under which they would prove useful is a direction for future inquiry as a way of sustaining Moore's law.

REFERENCES

- [1] ADLEMAN, L. M., “Two theorems on random polynomial time,” in *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, pp. 75–83, 1978.
- [2] AKGUL, B., CHAKRAPANI, L., KORKMAZ, P., and PALEM, K., “Probabilistic CMOS technology: A survey and future directions,” in *Proceedings of the IFIP International Conference on Very Large Scale Integration*, pp. 1–6, Oct. 2006.
- [3] AMIRTHARAJAH, R. and CHANDRAKASAN, A., “A micropower programmable DSP using approximate signal processing based on distributed arithmetic,” *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 337–347, Feb. 2004.
- [4] AMIRTHARAJAH, R., XANTHOPOULOS, T., and CHANDRAKASAN, A., “Power scalable processing using distributed arithmetic,” in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 170–175, 1999.
- [5] ANANTHASHAYANA, V., “Comments on ‘Fourier analysis and signal processing by use of the mobius inversion formula’ by I.S. Reed et. al.,” *IEEE Transactions on Signal Processing*, vol. 40, p. 676, Mar. 1992.
- [6] BAHAR, R. I., MUNDY, J., and CHEN, J., “A probabilistic-based design methodology for nanoscale computation,” in *Proceedings of the 2003 IEEE/ACM International Conference on Computer-aided Design*, pp. 480–486, 2003.
- [7] BEINLICH, I., SUERMONDT, G., CHAVEZ, R., and COOPER, G., “The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks,” in *Proceedings of the Second European Conference on AI and Medicine*, pp. 247–256, 1989.
- [8] BENNETT, C. H., “Logical reversibility of computation,” *IBM Journal of Research and Development*, vol. 17, pp. 525–532, Nov. 1973.
- [9] BERGMANN, G., “The logic of probability,” *American Journal of Physics*, vol. 9, pp. 263–272, 1941.
- [10] BLUM, M. and MICALI, S., “How to generate cryptographically strong sequences of pseudo-random bits,” *SIAM Journal on Computing*, vol. 13, no. 4, pp. 850–864, 1984.
- [11] BOLTZMANN, L., *Lectures on gas theory*. Berkeley: University of California Press, 1964.

- [12] BOOLE, G., *An Investigation of the Laws of Thought*. Dover Publications, 1958.
- [13] BOOLOS, G. S. and JEFFREY, R. C., *Computability and Logic: 3rd ed.* New York, NY, USA: Cambridge University Press, 1989.
- [14] BORKAR, S., “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.
- [15] BORKAR, S., “Designing reliable systems from unreliable components: The challenges of transistor variability and degradation,” *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.
- [16] BORKAR, S., KARNIK, T., NARENDRA, S., TSCHANZ, J., KESHAVARZI, A., and DE, V., “Parameter variations and impact on circuits and microarchitecture,” in *Proceedings of the 40th Annual Conference on Design Automation (DAC)*, pp. 338–342, 2003.
- [17] BOUDREAUX-BARTELS, G. and PARKS, T., “Discrete Fourier transform using summation by parts,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 12, pp. 1827–1830, Apr. 1987.
- [18] BRENT, R. P. and KUNG, H. T., “The area-time complexity of binary multiplication,” *Journal of The ACM*, vol. 28, no. 3, pp. 521–534, 1981.
- [19] BRENT, R. P., “The parallel evaluation of general arithmetic expressions,” *Journal of the ACM*, vol. 21, no. 2, pp. 201–206, 1974.
- [20] BRYANT, R. E., “Symbolic Boolean manipulation with ordered binary-decision diagrams,” *ACM Computing Surveys*, vol. 24, no. 3, pp. 293–318, 1992.
- [21] BURNETT, D., HIGMAN, J., HOEFLER, A., LI, C.-N., and KUHN, P., “Variation in natural threshold voltage of NVM circuits due to dopant fluctuations and its impact on reliability,” *IEEE International Electron Devices Meeting Technical Digest*, pp. 529–532, 2002.
- [22] CARNOT, S., *Reflections On The Motive Power Of Fire - 1824 (Translated by Robert Fox)*. Manchester University Press, 1986.
- [23] CHAITIN, G. J. and SCHWARTZ, J. T., “A note on monte carlo primality tests and algorithmic information theory,” *Communications on Pure and Applied Mathematics*, vol. 31, pp. 521–527, 1978.
- [24] CHAITIN, G., “Algorithmic information theory,” *IBM Journal of Research and Development*, vol. 21, pp. 350–359, 1977.
- [25] CHAKRAPANI, L. N., KORKMAZ, P., MOONEY, V. J., PALEM, K. V., PUTTASWAMY, K., and WONG, W. F., “The emerging power crisis in embedded

- processors: what can a compiler do?,” in *Proceedings of the IEEE/ACM International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pp. 176–180, 2001.
- [26] CHAKRAPANI, L. N., AKGUL, B. E. S., CHEEMALAVAGU, S., KORKMAZ, P., PALEM, K. V., and SESHASAYEE, B., “Ultra efficient embedded SoC architectures based on probabilistic CMOS technology,” in *Proceedings of the 9th Design Automation and Test in Europe*, pp. 1110–1115, Mar. 2006.
 - [27] CHAKRAPANI, L. N., GYLLENHAAL, J., MEI W. HWU, W., MAHLKE, S. A., PALEM, K. V., and RABBAH, R. M., “Trimaran: An infrastructure for research in instruction-level parallelism,” in *Proceedings of the 17th International Workshop on Languages and Compilers for Parallel Computing; Lecture Notes in Computer Science*, vol. 3602, pp. 32–41, Springer-Verlag Berlin Heidelberg, Aug. 2005.
 - [28] CHAKRAPANI, L. N. B., GEORGE, J., MARR, B., AKGUL, B. E. S., and PALEM, K. V., “Probabilistic design: A survey of probabilistic CMOS technology and future directions for terascale IC design,” in *VLSI-SoC: Research Trends in VLSI and Systems on Chip*, vol. 249, pp. 101–118, Springer Boston, 2008.
 - [29] CHAKRAPANI, L. N. B., KORKMAZ, P., AKGUL, B. E. S., and PALEM, K. V., “Probabilistic system-on-a-chip architectures,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 12, no. 3, pp. 1–28, 2007.
 - [30] CHAKRAPANI, L. N. B., MUNTIMADUGU, K. K., AVINASH, L., GEORGE, J., and PALEM, K. V., “Highly energy and performance efficient embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation,” in *Proceedings of the IEEE/ACM International Conference on Compilers, Architecture, and Synthesis of Embedded Systems (to appear)*, 2008.
 - [31] CHAKRAPANI, L. N. B. and PALEM, K. V., “A probabilistic Boolean logic and its meaning,” *Rice University, Department of Computer Science Technical Report*, June 2008.
 - [32] CHANDRAKASAN, A., AMIRTHARAJAH, R., GOODMAN, J., and RABINER, W., “Trends in low power digital signal processing,” *IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 604–607, 1998.
 - [33] CHANG, J. and PEDRAM, M., “Energy minimization using multiple supply voltages,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, pp. 436–443, Dec. 1997.
 - [34] CHEEMALAVAGU, S., KORKMAZ, P., and PALEM, K. V., “Ultra low-energy computing via probabilistic algorithms and devices: CMOS device primitives

- and the energy-probability relationship,” in *Proceedings of the International Conference on Solid State Devices and Materials*, pp. 402–403, Sept. 2004.
- [35] CHEEMALAVAGU, S., KORKMAZ, P., PALEM, K. V., AKGUL, B. E. S., and CHAKRAPANI, L. N., “A probabilistic CMOS switch and its realization by exploiting noise,” in *Proceedings of the IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 452–457, 2005.
 - [36] CHLEBUS, B. S., GASIENIEC, L., and PELC, A., “Deterministic computations on a PRAM with static processor and memory faults,” *Fundamenta Informaticae*, vol. 55, no. 3-4, pp. 285–306, 2003.
 - [37] CHOR, B. and GOLDREICH, O., “Unbiased bits from sources of weak randomness and probabilistic communication complexity (extended abstract),” in *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pp. 429–442, 1985.
 - [38] CHURCH, A., “On the concept of a random sequence,” *Bulletin of the American Mathematical Society*, vol. 46, pp. 130–135, 1940.
 - [39] CLAUSIUS, R., “On the moving force of heat and the laws of heat which may be deduced therefrom,” *Philosophical Magazine (Translation of the original German text *Über die Bewegende Kraft der Wärme* published in 1850 in *Annalen der Physik*)*, 1851.
 - [40] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., and STEIN, C., *Introduction to Algorithms, Second Edition*. The MIT Press, Sept. 2001.
 - [41] COX, R. T., “Probability, frequency and reasonable expectation,” *American Journal of Physics*, vol. 14, pp. 1–13, 1946.
 - [42] COX, R. T., *Algebra of Probable Inference*. Johns Hopkins University Press, 2002.
 - [43] DAVIS, M., *Engines of Logic: Mathematicians and the Origin of the Computer*. New York, USA: W. W. Norton and Company, 2001.
 - [44] DE, V. and BORKAR, S., “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.
 - [45] DE FINETTI, B., “Foresight, its logical laws, its subjective sources,” in *Studies in Subjective Probability (Translated and reprinted)* (KYBURG, H. and SMOLKA, H., eds.), pp. 93–159, New York: Wiley, 1964.
 - [46] DESCHAMPS, J.-P., BIOUL, G. J. A., and SUTTER, G. D., *Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems*. Wiley-Interscience, 2006.

- [47] DIACONIS, P., “A Frequentist does this, a Bayesian that,” *SIAM News*, Mar. 2004.
- [48] DING, L. and MAZUMDER, P., “On circuit techniques to improve noise immunity of CMOS dynamic logic,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 9, pp. 910–925, 2004.
- [49] DING, Y. Z. and RABIN, M. O., “Hyper-Encryption and everlasting security,” in *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science; Lecture Notes In Computer Science*, vol. 2285, pp. 1–26, 2002.
- [50] DOBRUSHIN, R. L. and ORTYUKOV, S. I., “Lower bound for the redundancy of self-correcting arrangements of unreliable functional elements,” *Problems of Information Transmission*, vol. 13, no. 3, pp. 59–65, 1977.
- [51] DOBRUSHIN, R. L. and ORTYUKOV, S. I., “Upper bound on the redundancy of self-correcting arrangements of unreliable elements,” *Problems of Information Transmission*, vol. 13, no. 3, pp. 201–20, 1977.
- [52] EFRON, B., “Controversies in the foundations of statistics,” *The American Mathematical Monthly*, vol. 85, no. 4, pp. 231–246, 1978.
- [53] ERMOLOVA, N. and HAGGMAN, S., “Simplified bounds for the complementary error function; application to the performance evaluation of signal-processing systems,” in *Proceedings of the 12th European Signal Processing Conference*, pp. 1087–1090, Sept. 2004.
- [54] ERNST, D., KIM, N. S., DAS, S., PANT, S., PHAM, T., RAO, R., ZIESLER, C., BLAAUW, D., AUSTIN, T., and MUDGE, T., “Razor: A low-power pipeline based on circuit-level timing speculation,” in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 7–18, Oct. 2003.
- [55] FAGIN, R. and HALPERN, J. Y., “Reasoning about knowledge and probability,” *Journal of the ACM*, vol. 41, no. 2, pp. 340–367, 1994.
- [56] FAGIN, R., HALPERN, J. Y., and MEGIDDO, N., “A logic for reasoning about probabilities,” *Information and Computation*, vol. 87, no. 1, pp. 78–128, 1990.
- [57] FELLER, W., *An Introduction to Probability Theory and its Applications*. Wiley Eastern Limited, 1984.
- [58] FERRENBURG, A. M., LANDAU, D. P., and WONG, Y. J., “Monte carlo simulations: Hidden errors from “good” random number generators,” *Physical Review Letters*, vol. 69, pp. 3382–3384, 1992.

- [59] FISHER, J. A., FARABOSCHI, P., and YOUNG, C., *Embedded Computing : A VLIW Approach to Architecture, Compilers and Tools*. Morgan Kaufmann, Dec. 2004.
- [60] FREDKIN, E. and TOFFOLI, T., “Conservative logic,” *International Journal of Theoretical Physics*, vol. 21, pp. 219–253, 1982.
- [61] FUKS, H., “Non-deterministic density classification with diffusive probabilistic cellular automata,” *Physical Review E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 66, pp. 066106.1–066106.4, 2002.
- [62] GABBAY, D. M. and WOODS, J., *The Rise of Modern Logic: From Leibniz to Frege, (Handbook of the History of Logic)*, vol. 3. North Holland, 2004.
- [63] GAIFMAN, H., “Concerning measures in first order calculi,” *Israel Journal of Mathematics*, vol. 2, no. 1, pp. 1–18, 1964.
- [64] GELENBE, E., “Random neural networks with negative and positive signals and product form solution,” *Neural Computation*, vol. 1, no. 4, pp. 502–511, 1989.
- [65] GELENBE, E. and BATTY, F., “Minimum graph covering with the random neural network model,” *Neural Networks: Advances and Applications*, vol. 2, 1992.
- [66] GEORGE, J., MARR, B., AKGUL, B. E. S., and PALEM, K., “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.
- [67] GIBBONS, P. B., “A more practical PRAM model,” in *Proceedings of the First Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 158–168, 1989.
- [68] GIBBS, W., “On the equilibrium of heterogeneous substances,” *Transactions of the Connecticut Academy*, Oct. 1875, May, 1876, May, 1877, July, 1878.
- [69] GOLDSTEIN, S., BUDI, M., MISHRA, M., and VENKATARAMANI, G., “Re-configurable computing and electronic nanotechnology,” in *Proceedings of the 14th International Conference on Application-specific Systems, Architectures and Processors*, pp. 132–143, 2003.
- [70] GOOD, I. J., “Random thoughts about randomness,” in *Proceedings of the Biennial Meeting of the Philosophy of Science Association*, pp. 117–135, 1972.
- [71] GRAYBILL, R. and MELHEM, R., eds., *Power aware computing*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

- [72] GURUSWAMI, V., “Better extractors for better codes?,” in *Proceedings of the The thirty-sixth annual ACM symposium on Theory of computing*, pp. 436–444, 2004.
- [73] HAILPERIN, T., *Sentential Probability Logic: Origins, Development, Current Status, and Technical Applications*. Lehigh University Press, 1996.
- [74] HAJEK, A., “Interpretations of probability,” in *Stanford Encyclopedia of Philosophy* (Ed. Edward N. Zalta), 2007.
- [75] HALPERN, J., “Reasoning about knowledge: an overview,” in *Proceedings of the 1986 Conference on Theoretical aspects of reasoning about knowledge*, (San Francisco, CA, USA), pp. 1–17, Morgan Kaufmann Publishers Inc., 1986.
- [76] HARTMANIS, J. and STEARNS, R. E., “On the computational complexity of algorithms,” *Transactions of the American Mathematical Society*, vol. 117, pp. 285–306, 1965.
- [77] HEGDE, R. and SHANBHAG, N. R., “Soft digital signal processing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, pp. 813–823, Dec. 2001.
- [78] [HTTP://WWW.TRIMARAN.ORG](http://www.trimaran.org), “Trimaran: An infrastructure for research in instruction-level parallelism.”
- [79] HUANG, J., MOMENZADEH, M., and LOMBARDI, F., “An overview of nanoscale devices and circuits,” *IEEE Design and Test*, vol. 24, no. 4, pp. 304–311, 2007.
- [80] IMPAGLIAZZO, R. and ZUCKERMAN, D., “How to recycle random bits,” in *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pp. 248–253, 1989.
- [81] INTEL CORP., “SA-1100 microprocessor technical reference manual,” Sept. 1998.
- [82] ITRS, “International technology roadmap for semiconductors,” 2007.
- [83] JACOBSON, N., *Basic Algebra I*. W H Freeman and Company, 1974.
- [84] JACOME, M., HE, C., DE VECIANA, G., and BIJANSKY, S., “Defect tolerant probabilistic design paradigm for nanotechnologies,” in *Proceedings of the 41st Annual Conference on Design Automation (DAC)*, pp. 596–601, 2004.
- [85] JAYNES, E., *Probability Theory: The Logic of Science*. Cambridge, UK: Cambridge University Press, 2003.
- [86] KAMATH, A., MOTWANI, R., PALEM, K. V., and SPIRAKIS, P. G., “Tail bounds for occupancy and the satisfiability threshold conjecture,” *Random Structures and Algorithms*, vol. 7, no. 1, pp. 59–80, 1995.

- [87] KARP, R. M., “The probabilistic analysis of some combinatorial search algorithms,” in *Algorithms and complexity: New directions and recent results* (Traub, J. P., ed.), pp. 1–19, New York, USA: Academic Press, 1976.
- [88] KARP, R., “Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane,” *Mathematics of Operations Research*, vol. 2, no. 3, pp. 209–224, 1977.
- [89] KEDEM, Z. M., PALEM, K. V., RAGHUNATHAN, A., and SPIRAKIS, P. G., “Combining tentative and definite executions for very fast dependable parallel computing,” in *Proceedings of the Twenty-third Annual ACM Symposium on the Theory of Computing*, (New York, NY, USA), pp. 381–390, ACM, 1991.
- [90] LORD KELVIN, “On an absolute thermometric scale: Founded on Carnot’s theory of the motive power of heat, and calculated from Regnault’s observations,” in *Sir William Thomson, Mathematical and Physical Papers, vol. 1*, pp. 100–106, Cambridge University Press, 1882.
- [91] KENDALL, M. G., “On the reconciliation of theories of probability,” *Biometrika*, vol. 36, no. 1-2, pp. 101–116, 1949.
- [92] KEYNES, J. M., *A Treatise on Probability*. London: Macmillan, 1921.
- [93] KIM, T., “Application-driven low-power techniques using dynamic voltage scaling,” in *Proceedings of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 199–206, 2006.
- [94] KISH, L. B., “End of Moore’s law: Thermal (noise) death of integration in micro and nano electronics,” *Physics Letters A*, vol. 305, pp. 144–149, 2002.
- [95] KNUTH, D. E., *Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd Edition)*. Addison-Wesley Professional, Nov. 1997.
- [96] KOGGE, P. M. and STONE, H., “A parallel algorithm for the efficient solution of a general class of recurrence equations,” *IEEE Transactions on Computers*, vol. 22, pp. 786–793, Aug. 1973.
- [97] KOLAITIS, P. G. and VARDI, M. Y., “0-1 laws and decision problems for fragments of second-order logic,” *Information and Computation*, vol. 87, no. 1-2, pp. 302–338, 1990.
- [98] KOLAITIS, P. G. and VARDI, M. Y., “0-1 laws for fragments of existential second-order logic: A survey,” *Mathematical Foundations of Computer Science*, pp. 84–98, 2000.
- [99] KOLMOGOROV, A. N., *Foundations of the Theory of Probability* (Trans. Nathan Morrison). New York: Chelsea Publishing Company, 1956.

- [100] KOLMOGOROV, A. N., “Three approaches to the quantitative definition of information,” *Problems of Information Transmission*, vol. 1, no. 1, pp. 1–7, 1965.
- [101] KORKMAZ, P., *Probabilistic CMOS (PCMOS) in the Nanoelectronics Regime*. PhD thesis, Georgia Institute of Technology, 2007.
- [102] KORKMAZ, P., AKGUL, B. E. S., CHAKRAPANI, L. N., and PALEM, K. V., “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.
- [103] LAMOUREUX, M., “The Poorman’s transform: approximating the Fourier transform without multiplication,” *IEEE Transactions on Signal Processing*, vol. 41, no. 3, pp. 1413–1415, Mar 1993.
- [104] LANDAUER, R., “Irreversibility and heat generation in the computing process,” *IBM Journal of Research and Development*, vol. 3, pp. 183–191, July 1961.
- [105] LEFF, H. and REX, A., eds., *Maxwell’s Demon: Entropy, Information, Computing*. Princeton University Press, 1990.
- [106] LU, M., *Arithmetic and Logic in Computer Systems*. Hoboken, NJ: John Wiley & Sons, Inc., 2004.
- [107] LUDWIG, J., NAWAB, S., and CHANDRAKASAN, A., “Low power filtering using approximate processing for DSP applications,” *The IEEE Custom Integrated Circuits Conference*, pp. 185–188, May 1995.
- [108] LUDWIG, J., NAWAB, S., and CHANDRAKASAN, A., “Low-power digital filtering using approximate processing,” *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 395–400, Mar. 1996.
- [109] LYUU, Y.-D., *Information dispersal and parallel computation*. New York, NY, USA: Cambridge University Press, 1992.
- [110] MACKAY, D., “Bayesian interpolation,” *Neural Computation*, vol. 4, no. 3, 1992.
- [111] MANO, M. M., *Digital Design*. Prentice Hall, 2001.
- [112] MANZAK, A. and CHAKTRABARTI, C., “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.
- [113] MARPE, D., WIEGAND, T., and SULLIVAN, G. J., “The H.264/MPEG4-AVC standard and its fidelity range extensions,” *IEEE Communications Magazine*, Sept. 2005.

- [114] MARTIN, S. M., FLAUTNER, K., MUDGE, T., and BLAAUW, D., “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*, Nov. 2002.
- [115] MARTORELL, F., COTOFANA, S., and RUBIO, A., “An analysis of internal parameter variations effects on nanoscaled gates,” *IEEE Transactions on Nanotechnology*, vol. 7, pp. 24–33, Jan. 2008.
- [116] MAXWELL, J. C., *Theory of Heat (1871)*. Dover Books on Physics, 2001.
- [117] MEINDL, J. D., “Low power microelectronics: retrospect and prospect,” *Proceedings of The IEEE*, vol. 83, pp. 619–635, Apr. 1995.
- [118] MEINDL, J. D., CHEN, Q., and DAVIS, J. A., “Limits on silicon nanoelectronics for terascale integration,” *Science*, vol. 293, no. 5537, pp. 2044–2049, 2001.
- [119] MEINDL, J. D. and DAVIS, J. A., “The fundamental limit on binary switching energy for terascale integration (TSI),” *IEEE Journal of Solid State Circuits*, vol. 35, pp. 1515–1516, Oct. 2000.
- [120] MEINDL, J., “Theoretical, practical and analogical limits in ULSI,” *IEEE International Electron Devices Meeting Technical Digest*, pp. 8–13, 1983.
- [121] MENDELSON, E., *Introduction to Mathematical Logic*. Chapman and Hall, 1997.
- [122] MICHELI, G. D., *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1994.
- [123] MOORE, G. E., “Cramming more components onto integrated circuits,” *Electronics Magazine*, vol. 38, no. 8, 1965.
- [124] MOORE, G., “No exponential is forever: But forever can be delayed!,” in *Proceedings of the IEEE International Solid-State Circuits Conference*, pp. 20–23, 2003.
- [125] MOTWANI, R. and RAGHAVAN, P., *Randomized Algorithms*. Cambridge University Press, 1995.
- [126] MOTWANI, R., “Average-case analysis of algorithms for matching and related problems,” *Journal of the ACM*, vol. 41, no. 6, pp. 1329–1356, 1994.
- [127] MUKHERJEE, S. S., EMER, J., and REINHARDT, S. K., “The soft error problem: An architectural perspective,” in *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, pp. 243–247, 2005.

- [128] MUKHERJEE, S. S., KONTZ, M., and REINHARDT, S. K., “Detailed design and evaluation of redundant multithreading alternatives,” *SIGARCH Computer Architecture News*, vol. 30, no. 2, pp. 99–110, 2002.
- [129] MUKHERJEE, S. S., WEAVER, C., EMER, J., REINHARDT, S. K., and AUSTIN, T., “A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor,” in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2003.
- [130] NATORI, K. and SANO, N., “Scaling limit of digital circuits due to thermal noise,” *Journal of Applied Physics*, vol. 83, pp. 5019–5024, 1998.
- [131] NAWAB, S. H., OPPENHEIM, A. V., CHANDRAKASAN, A. P., M. WINOGRAD, J., and T. LUDWIG, J., “Approximate signal processing,” *The Journal of VLSI Signal Processing*, vol. 15, pp. 177–200, 1997.
- [132] NAWAB, S. and DORKEN, E., “A framework for quality versus efficiency trade-offs in STFT analysis,” *IEEE Transactions on Signal Processing*, vol. 43, no. 4, pp. 998–1001, 1995.
- [133] NAWAB, S. and WINOGRAD, J., “Approximate signal processing using incremental refinement and deadline-based algorithms,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 2857–2860, May 1995.
- [134] NEPAL, K., BAHAR, R. I., MUNDY, J., PATTERSON, W. R., and ZASLAVSKY, A., “Designing logic circuits for probabilistic computation in the presence of noise,” in *Proceedings of the 42nd Annual Conference on Design Automation (DAC)*, pp. 485–490, 2005.
- [135] NILSSON, N. J., “Probabilistic logic,” *Artificial Intelligence*, vol. 28, no. 1, 1986.
- [136] PACKAN, P. A., “Device physics: Pushing the limits,” *Science*, vol. 285, no. 5436, pp. 2079–2081, 1999.
- [137] PALEM, K. V., “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.
- [138] PALEM, K. V., “Proof as experiment: Probabilistic algorithms from a thermodynamic perspective,” in *Proceedings of the International Symposium on Verification (Theory and Practice)*, June 2003.
- [139] PALEM, K. V., “Energy aware computing through probabilistic switching: A study of limits,” *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1123–1137, 2005.

- [140] PALEM, K. V., AKGUL, B. E. S., and GEORGE, J., “Variable scaling for computing elements,” *Invention Disclosure*, Feb. 2006.
- [141] PALEM, K. V., CHEEMALAVAGU, S., KORKMAZ, P., and AKGUL, B. E., “Probabilistic and introverted switching to conserve energy in a digital system,” *US Patent*, no. 20050240787, 2005.
- [142] PAPADIMITRIOU, C. H., *Computational Complexity*. Addison Wesley, 1993.
- [143] PARHAMI, B., *Computer arithmetic: Algorithms and hardware designs*. Oxford, UK: Oxford University Press, 2000.
- [144] PARK, S. and MILLER, K. W., “Random number generators: good ones are hard to find,” *Communications of the ACM*, vol. 31, 1988.
- [145] PARKINSON, G., *Leibniz-Logical Papers*. Oxford University Press, 1966.
- [146] PAU, C., “A stereo audio chip using approximate processing for decimation and interpolation filters,” *IEEE Journal of Solid-State Circuits*, vol. 35, no. 1, pp. 45–55, Jan 2000.
- [147] PEDRAM, M., “Power minimization in IC design: principles and applications,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 1, no. 1, pp. 3–56, 1996.
- [148] PERING, T., BURD, T., and BRODERSEN, R., “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.
- [149] PFEFFER, A., *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford Univeristy, 2000.
- [150] PIPPENGER, N., “On networks of noisy gates,” in *Proceedings of the 26th Annual IEEE Symposim on Foundations of Computer Science*, pp. 30–38, 1985.
- [151] PIPPENGER, N., “Invariance of complexity measures for networks with unreliable gates,” *Journal of the ACM*, vol. 36, pp. 531–539, 1989.
- [152] PIPPENGER, N., “Analysis of carry propagation in addition: An elementary approach,” tech. rep., University of British Columbia, Vancouver, BC, Canada, Canada, 2001.
- [153] PIPPENGER, N., STAMOULIS, G. D., and TSITSIKLIS, J. N., “On a lower bound for the redundancy of reliable networks with noisy gates,” *IEEE Transactions on Information Theory*, vol. 37, no. 3, pp. 639–643, 1991.
- [154] PUTTASWAMY, K., CHAKRAPANI, L. N., CHOI, K.-W., DHILLON, Y. S., DIRIL, U., KORKMAZ, P., LEE, K.-K., PARK, J. C., CHATTERJEE, A.,

- ELLERVEE, P., VINCENT JOHN MOONEY, I., PALEM, K. V., and WONG, W.-F., "Power-performance trade-offs in second level memory used by an arm-like risc architecture," in *Power aware computing* (GRAYBILL, R. and MELHEM, R., eds.), pp. 211–224, Norwell, MA, USA: Kluwer Academic Publishers, 2002.
- [155] RABBAH, R. M. and PALEM, K. V., "Data remapping for design space optimization of embedded memory systems," *ACM Transactions on Embedded Computing Systems*, vol. 2, no. 2, pp. 186–218, 2003.
- [156] RABIN, M. O., "Probabilistic automata," *Information and Control*, vol. 6, pp. 230–245, 1963.
- [157] RABIN, M. O., "Probabilistic algorithms," in *Algorithms and Complexity, New Directions and Recent Trends* (TRAUB, J. F., ed.), pp. 29–39, Academic Press, 1976.
- [158] RABIN, M. O., "Complexity of computations," *Communications of the ACM*, vol. 20, no. 9, pp. 625–633, 1977.
- [159] RABIN, M. O., "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM*, vol. 36, no. 2, pp. 335–348, 1989.
- [160] RABIN, M. O. and SCOTT, D. S., "Finite automata and their decision problems," *IBM Journal of Research and Development*, vol. 3, pp. 114–125, 1959.
- [161] RAMSEY, F. P., "Truth and probability (reprinted 1990)," in *Philosophical Papers*, D. H. Mellor (ed.), Cambridge: Cambridge University Press, 1926.
- [162] RANDOM NUMBER GENERATION AND TESTING, "<http://csrc.nist.gov/rng/>."
- [163] RAY, J., HOE, J. C., and FALSAFI, B., "Dual use of superscalar datapath for transient-fault detection and recovery," in *Proceedings of the 34th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 214–224, 2001.
- [164] REED, I., TUFTS, D., YU, X., TRUONG, T., SHIH, M.-T., and YIN, X., "Fourier analysis and signal processing by use of the mobius inversion formula," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, pp. 458–470, Mar. 1990.
- [165] REICHENBACH, H., *The Theory of Probability*. Berkeley, USA: University of California Press, 1949.
- [166] REIS, G., CHANG, J., VACHHARAJANI, N., RANGAN, R., and AUGUST, D., "SWIFT: Software implemented fault tolerance," in *Proceedings of the International Symposium on Code Generation and Optimization*, pp. 243–254, 2005.
- [167] RICHARDS, M., *Fundamentals of Radar Signal Processing*. McGraw-Hill, 2005.

- [168] ROSENBLOOM, P. C., *The Elements of Mathematical Logic*. Dover Publications, 2005.
- [169] SANO, N., “Increasing importance of electronic thermal noise in sub-0.1mm Si-MOSFETs,” *The IEICE Transactions on Electronics*, vol. E83-C, pp. 1203–1211, 2000.
- [170] SARANGI, S., GRESKAMP, B., TEODORESCU, R., NAKANO, J., TIWARI, A., and TORRELLAS, J., “VARIUS: A model of process variation and resulting timing errors for microarchitects,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 21, pp. 3–13, Feb. 2008.
- [171] SAVAGE, J. E., *The Complexity of Computing*. Melbourne, FL, USA: Krieger Publishing Co., Inc., 1987.
- [172] SCHWARTZ, J. T., “Fast probabilistic algorithms for verification of polynomial identities,” *Journal of the ACM*, vol. 27, no. 4, pp. 701–717, 1980.
- [173] SCOTT, D. and KRAUSS, P., “Assigning probabilities to logical formulas,” in *Aspects of Inductive Logic* (HINTIKKA, J. and SUPPES, P., eds.), pp. 219–264, Duke University Press, 1966.
- [174] SEAL, D., *ARM Architecture Reference Manual*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.
- [175] SHALTIEL, R., “Recent developments in explicit constructions of extractors,” *Bulletin of the EATCS*, pp. 67–95, 2002.
- [176] SHANNON, C. E., “A mathematical theory of communication,” *Bell system technical journal*, vol. 27, 1948.
- [177] SHANNON, C. E., “A symbolic analysis of relay and switching circuits,” Master’s thesis, Massachusetts Institute of Technology, 1937.
- [178] SHIM, B., SRIDHARA, S. R., and SHANBHAG, N. R., “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.
- [179] SIEWIOREK, D. P. and SWARZ, R. S., *Reliable Computer Systems: Design and Evaluation*. AK Peters, Ltd., 1998.
- [180] SINHA, A. and CHANDRAKASAN, A., “JouleTrack a web based tool for software energy profiling,” in *Proceedings of the 38th Annual Conference on Design Automation (DAC)*, pp. 220–225, 2001.
- [181] SINHA, A., WANG, A., and CHANDRAKASAN, A., “Energy scalable system design,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, pp. 135–145, Apr. 2002.

- [182] SIPSER, M., *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.
- [183] SOLOVAY, R. and STRASSEN, V., “A fast monte-carlo test for primality,” *SIAM Journal on Computing*, pp. 84–85, 1977.
- [184] STEIN, K.-U., “Noise-induced error rate as limiting factor for energy per operation in digital ICs,” *IEEE Journal of Solid-State Circuits*, 1977.
- [185] STINE, B. E., MEMBER, S., BONING, D. S., and CHUNG, J. E., “Analysis and decomposition of spatial variation in integrated circuit processes and devices,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 10, pp. 24–41, 1997.
- [186] TOUR, J. M. and JAMES, D. K., “Molecular electronic computing architectures: A review,” in *Handbook of Nanoscience, Engineering and Technology, Second Edition* (GODDARD, W. A., I., BRENNER, D. W., LYSHEVSKI, S. E., and IAFRATE, G. J., eds.), pp. 5.1–5.28, New York: CRC Press, 2007.
- [187] TURING, A. M., “On computable numbers, with an application to the entscheidungsproblem,” *Proceedings of the London Mathematical Society*, vol. 2, no. 42, pp. 230–265, 1936.
- [188] UMANS, C., “Pseudo-random generators for all hardnesses,” *Journal of Computer and System Sciences*, vol. 67, no. 2, pp. 419–440, 2003.
- [189] VALLURI, M. and JOHN, L., “Is compiling for performance == compiling for power ?,” in *Proceedings of the 5th Annual Workshop on Interaction between Compilers and Computer Architectures*, 2001.
- [190] VENN, J., *The Logic of Chance (reprinted 1962)*. New York, USA: Macmillan and co, 1876.
- [191] VON MISES R., *Probability, Statistics and Truth*. New York, USA: Macmillan and Company, 1957.
- [192] VON NEUMANN, J., “Probabilistic logics and the synthesis of reliable organisms from unreliable components,” *Automata Studies*, pp. 43–98, 1956.
- [193] WANG, L. and SHANBHAG, N. R., “Low-power filtering via adaptive error-cancellation,” *IEEE Transactions on Signal Processing*, vol. 51, pp. 575–583, Feb. 2003.
- [194] WHITESITT, J. E., *Boolean Algebra and Its Applications*. Dover Publications, 1995.
- [195] WIDROW, B. and STEARNS, S. D., *Adaptive signal processing*. Prentice-Hall, Englewood Cliffs, NJ, 1985.
- [196] WINOGRAD, S., *Arithmetic Complexity of Computations*. Society for Industrial and Applied Mathematics, 1980.

- [197] WOLFRAM, S., *A New Kind of Science*. Wolfram Media, 2002.
- [198] XANTHOPOULOS, T. and CHANDRAKASAN, A., “A low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization,” *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 740–750, May 2000.
- [199] YAO, A., “Theory and application of trapdoor functions,” in *Proceedings of the 23rd Symposium on The Foundations of Computer Science*, pp. 80–91, 1982.
- [200] YEH, Y. and KUO, S., “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.
- [201] YEH, Y., KUO, S., and JOU, J., “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.
- [202] YEH, Y., “Triple-triple redundant 777 primary flight computer,” *IEEE Aerospace Applications Conference*, vol. 1, pp. 293–307, Feb. 1996.
- [203] ZIEGLER, J. F., CURTIS, H. W., MUHLFELD, H. P., MONTROSE, C. J., and CHIN, B., “IBM experiments in soft fails in computer electronics (1978–1994),” *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 3–18, 1996.
- [204] ZIEGLER, J. F. and LANFORD, W. A., “Effect of cosmic rays on computer memories,” *Science*, vol. 206, pp. 776–788, Nov. 1979.