Appropriateness of Imperfect CNFET Based Circuits for Error Resilient Computing Systems

by

Kaship Nabi Sheikh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Jie Han Associate Professor, Dept. of Electrical and Computer Engineering, University of Alberta
Supervisor(s):	Lan Wei Assistant Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	David Nairn Associate Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Youngki Yoon Associate Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal-External Member:	Armaghan Salehian Associate Professor, Dept. of Mechanical and Mechatronics Engineering, University of Waterloo

Abstract

With superior device performance consistently reported in extremely scaled dimensions, low dimensional materials (LDMs), including Carbon Nanotube Field Effect Transistor (CNFET) based technology, have shown the potential to outperform silicon for future transistors in advanced technology nodes. Studies have also demonstrated orders of magnitude improvement in energy efficiency possible with LDMs, in comparison to silicon at competing technology nodes. However, the current fabrication processes for these materials suffer from process imperfections and still appear to be inadequate to compete with silicon for the mainstream high volume manufacturing. Among the LDMs, CNFETs are the most widely studied and closest to high volume manufacturing. Recent works have shown a significant increase in the complexity of CNFET based systems, including demonstration of a 16-bit microprocessor. However, the design of such systems has involved significantly wider-than-usual transistors and avoidance of certain logic combinations. The resulting complexity of several thousand transistors in such systems is still far from the requirements of high-performance general-purpose computing systems having billions of transistors. With the current progress of the process to fabricate CNFETs, their introduction in mainstream manufacturing is expected to take several more years. For an earlier technology adoption, CNFETs appear to be suited for error-resilient computing systems where errors during computation can be tolerated to a certain degree. Such systems relax the need for precise circuits and a perfect process while leveraging the potential energy benefits of CNFET technology in comparison to conventional Si technology. In this thesis, we explore the potential applications using an imperfect CNFET process for error-resilient computing systems, including the impact of the process imperfections at the system level and methods to improve it.

The current most widely adopted fabrication process for CNFETs (separation and placement of solution-based CNTs) still suffers from process imperfections, mainly from open CNTs due to missing of CNTs (in trenches connecting source and drain of CNFET). A fair evaluation of the performance of CNFET based circuits should thus take into consideration the effect of open CNTs, resulting in reduced drive currents. At the circuit level, this leads to failures in meeting 1) the minimum frequency requirement (due to an increase in critical path delay), and 2) the noise suppression requirement. We present a methodology to accurately capture the effect of open CNT imperfection in the state-of-the-art CNFET model, for circuit-level performance evaluation (both delay and glitch vulnerability) of CNFET based circuits using SPICE. A Monte Carlo simulation framework is also provided to investigate the statistical effect of open CNT imperfection on circuit-level performance. We introduce essential metrics to evaluate glitch vulnerability and also provide an effective link between glitch vulnerability and circuit topology.

The past few years have observed significant growth of interest in approximate computing for a wide range of applications, including signal processing, data mining, machine learning, image, video processing, etc. In such applications, the result quality is not compromised appreciably, even in the presence of few errors during computation. The ability to tolerate few errors during computation relaxes the need to have precise circuits. Thus the approximate circuits can be designed, with lesser nodes, reduced stages, and reduced capacitance at few nodes. Consequently, the approximate circuits could reduce critical path delays and enhanced noise suppression in comparison to precise circuits. We present a systematic methodology utilizing Reduced Ordered Binary Decision Diagrams (ROBDD) for generating approximate circuits by taking an example of 16-bit parallel prefix CNFET adder. The approximate adder generated using the proposed algorithm has $\sim 5\times$ reduction in the average number of nodes failing glitch criteria (along paths to primary output) and 43.4% lesser Energy Delay Product (EDP) even at high open CNT imperfection, in comparison to the ideal case of no open CNT imperfection, at a mean relative error of 3.3%.

The recent boom of deep learning has been made possible by VLSI technology advancement resulting in hardware systems, which can support deep learning algorithms. These hardware systems intend to satisfy the high-energy efficiency requirement of such algorithms. The hardware supporting such algorithms adopts neuromorphic-computing architectures with significantly less energy compared to traditional Von Neumann architectures. Deep Neural Networks (DNNs) belonging to deep learning domain find its use in a wide range of applications such as image classification, speech recognition, etc. Recent hardware systems have demonstrated the implementation of complex neural networks at significantly less power. However, the complexity of applications and depths of DNNs are expected to drastically increase in the future, imposing a demanding requirement in terms of scalability and energy efficiency of hardware technology. CNFET technology can be an excellent alternative to meet the aggressive energy efficiency requirement for future DNNs. However, degradation in circuit-level performance due to open CNT imperfection can result in timing failure, thus distorting the shape of non-linear activation function, leading to a significant degradation in classification accuracy. We present a framework to obtain sigmoid activation function considering the effect of open CNT imperfection. A digital neuron is explored to generate the sigmoid activation function, which deviates from the ideal case under imperfect process and reduced time period (increased clock frequency). The inherent error resilience of DNNs, on the other hand, can be utilized to mitigate the impact of imperfect process and maintain the shape of the activation function. We use pruning of synaptic weights, which, combined with the proposed approximate neuron, significantly reduces the chance of timing failures and helps to maintain the activation function shape even at high process imperfection and higher clock frequencies. We also provide a framework to obtain classification accuracy of Deep Belief Networks (class of DNNs based on unsupervised learning) using the activation functions obtained from SPICE simulations. By using both approximate neurons and pruning of synaptic weights, we achieve excellent system accuracy (only $< 0.5\%$ accuracy drop) with 25% improvement in speed, significant EDP advantage (56.7% less) even at high process imperfection, in comparison to a base configuration of the precise neuron and no pruning with the ideal process, at no area penalty.

In conclusion, this thesis provides directions for the potential applicability of CNFET based technology for error-resilient computing systems. For this purpose, we present methodologies, which provide approaches to assess and design CNFET based circuits, considering process imperfections. We accomplish a DBN framework for digit recognition, considering activation functions from SPICE simulations incorporating process imperfections. We demonstrate the effectiveness of using approximate neuron and synaptic weight pruning to mitigate the impact of high process imperfection on system accuracy.

Acknowledgements

This dissertation would not have been possible without the support, guidance of my research supervisor Prof. Lan Wei. I am grateful to her for trusting and allowing me to explore exciting research directions. I am also thankful to her for her easy availability for discussing research ideas that helped in maintaining continued progress towards achieving the goals of this Ph.D. research. Prof. Wei is considerate, respectful to her students, and care about their long term growth. I owe my deepest gratitude to her for my success, achievements, and support in difficult times during my stay here.

I would also like to thank my committee members Prof. David Nairn, Prof. Youngki Yoon, Prof. Armaghan Salehian for serving on my dissertation committee and providing me with valuable feedback that has helped in making this work more comprehensive.

I would also like to thank my collaborator Dr. Shu Jen Han (currently at HFC Semiconductor Corp), for providing the important initial directions for my research. I am also thankful to Daniel Zhou and An Qi Zhang for their assistance while working as URA in our research group.

I am thankful to my colleagues Zongxian Yang, Hao Zhang, Hazem Elgabra, Xuesong Chen, Shubham Ranjan, Rubaya Absar, Yiju Zhao, Sid Zarabi, and Egon Fernandes, for all the discussions and feedback during group meetings. During my time at Waterloo, I have made some excellent friends. Special thanks to Aimal Khan for providing consistent help and valuable inputs from the time I arrived in Waterloo. I have spent countless hours discussing a wide variety of topics with Mohammed Zeeshan. The time spent with him has helped in my life outside work.

My deepest gratitude to my parents for their encouragement, love, and never-ending support. I am thankful to my parents-in-law for their patience and encouragement during my graduate studies. I would also like to thank my siblings and cousins for their love and support. Above all, this thesis would not have been possible without support, love, encouragement of my wonderful wife Mahvesh. During my graduate studies, I was faced with stress numerous times. She was always there with the kind support and listening ear. I thank her for the patience and sacrifices. I humbly dedicate this thesis to her.

Table of Contents

List of Tables

List of Figures

Abbreviations

BDD Binary Decision Diagram 52, 56

BP Black Phosphorous 2

CNFET Carbon Nanotube Field Effect Transistor xiv, xv, xviii, xix, xxi, 2–6, 8–10, 13, 15–26, 28–32, 37, 40, 41, 43, 47, 49, 68, 69, 71, 72, 75, 80, 82, 90, 94, 116–120, 140–145, 167

CNN Convolutional Neural Network 120

CNNs Convolutional Neural Networks 120

CNT Carbon Nanotube xiv, xv, xxi, 2, 9–13, 17, 20–26, 28–32, 37, 40, 47, 51, 71, 80, 90, 117–119, 139–141, 143–146

CVD Chemical Vapor Deposition 11

DBN Deep Belief Network xix, 69, 86, 88–90, 115, 118, 119

DNN Deep Neural Network 68, 69, 90

DNNs Deep Neural Networks 5, 6, 15–17, 67–69, 90, 93, 94, 118

EDP Energy Delay Product xviii, xxi, 3, 15, 64–66, 113–115, 117, 118

GAA Gate All Around 28, 29

ITRS International Technology Roadmap for Semiconductors 2

LBG Local Bottom Gate 25

LDMs	Low Dimensional Materials 2, 3
LUT	Lookup Table 19, 41, 42
MC	Monte Carlo 80
Pd	Palladium 25
PUF	Physical Unclonable Function 13
RCA	Ripple Carry Adder xvi, 37–40
ROBDD	Reduced Order Binary Decision Diagram 52, 53, 55, 56, 66, 117
RRAM	Resistive Random-Access Memory 15
SCE	Short Channel Effects 1, 2, 9
Si	Silicon 31, 167
STA	Static Timing Analysis 4, 20, 41
TG	Top Gate 29
TMDs	Transition Metal Dichalcogenides 2
TRNG	True Random Number Generator 13
VS	Virtual Source 25
VSCNFET	Virtual-Source Carbon Nanotube Field-Effect Transistor 19, 25, 26, 47, 117

Chapter 1

Introduction

1.1 Motivation

The scaling down of silicon transistor dimensions has driven the semiconductor industry over the past several decades. With every new technology generation, scaling results in more transistors (high packing density, reduced cost per transistor), with the tremendous benefit of high performance for the same or less power [1, 2, 3]. Transistor scaling has faced numerous challenges, including difficult gate control, increased Short Channel Effects (SCE), mobility degradation at reduced dimensions, non-scalable leakage, parasitic components, and so on [4, 1]. In addition to traditional geometrical dimensional scaling, several remarkable innovations including strain [5, 6], high-k [6, 7], metal gate [6, 7] were introduced over the years to provide sustained benefits from scaling. The introduction of FinFET in 22 nm [8, 9] was a major milestone for the semiconductor industry by deviating from the traditional planar devices. The FinFET transistor provides better gate control over the channel, reduced SCE, reduced leakage power [9]. With tremendous efforts, 7 nm FinFETs were also successfully fabricated [10], and 5 nm FinFETs with extreme ultraviolet lithography (EUV) have also entered risk production phase [11, 12]. However, continuing with FinFETs in more advanced technology nodes looks quite challenging [13].

The International Technology Roadmap for Semiconductors (ITRS) has predicted the need for novel materials as channel to address transistor scaling in coming years [14]. With superior properties over bulk silicon in various aspects, the family of Low Dimensional Materials (LDMs) including graphene [15, 16], Transition Metal Dichalcogenides (TMDs) [17, 18, 19, 20], Black Phosphorous (BP) [21, 22, 23], CNT [24, 25, 26] has been actively explored to replace or complement silicon for future technology nodes. The striking feature of LDMs is naturally thin body (free from dangling bonds) at ultra-scaled dimensions, providing high mobility (free from surface scattering) [27] and excellent gate control. Because of high carrier mobility $\sim$ 20,000 cm^{2}V^{-1}s^{-1} at room temperature, graphene has attracted significant interest for LDMs based FETs. However, graphene due to zero bandgap [28] is not suitable for digital applications. BP among LDMs is also actively pursued, since its demonstration of mobility $\sim$ 1000 cm^{2}V^{-1}s^{-1} [22, 23] with on/off current ratio $> 10^5$ [29]. TMDs with the demonstration of 1 nm gate length MOS_2 based FET [18], showed significant potential of TMDs as channel material for advanced technology nodes. Recently, 1 bit microprocessor based on MOS_2 FETs, consisting of 115 transistors was fabricated [20], demonstrating the potential for large scale manufacturing of TMDs based FETs. In comparison to other LDMs, CNT has been extensively studied for almost two decades and has shown the potential of being close to high-volume manufacturing [24, 25, 26]. CN-

FET technology has demonstrated strong electrostatics and excellent transport with the potential of operating CNFETs at low voltages; consequently providing the promise of achieving order of magnitude improvement in Energy Delay Product (EDP) over the competing silicon-based transistors in advanced technology nodes [30, 31, 32, 33]. Recently, CNFET based brain-inspired computing system was fabricated with the capability of providing $\sim 35\times$ EDP improvement (after place and route) over competing silicon technology [34]. Recent works have also experimentally shown the possibility of scaling both the channel and contact length in CNFET to < 10 nm, with remarkably low contact resistance [35]. However, large scale manufacturing of LDMs including CNFETs is facing significant challenges and still seem quite far from competing with the scale of silicon-based general-purpose computing systems. The issue with imperfect process is a major roadblock for large scale manufacturing. It is a challenge which requires improvements from materials, devices, circuits and systems. An early technology adoption of LDMs including CNFETs is still possible for error-resilient computing paradigms such as approximate and neuromorphic computing. These paradigms on one hand provide the necessary tolerance to process imperfections and on the other leverage from the tremendous energy efficiency benefit of CNFET based technology.

1.2 Approach and Scope

With extensive research and the potential of being closest to high volume manufacturing in comparison to other LDMs [36, 34, 37, 38], we have focused on CNFET based circuits in this work. The work in the dissertation can be divided into the following three main portions.

1.2.1 Quantifying impacts of CNFET process imperfection on circuit-level performance

In this dissertation, we first find out the process imperfections arising from the currently popular processes for CNFET fabrication. Specifically, we identify the process imperfections having a prominent effect on CNFET circuit-level performance. A methodology is provided to effectively include the effect process imperfections, in the state of art CNFET model. We then provide a simulation framework to evaluate circuit-level performance in terms of common digital VLSI performance metrics delay, noise tolerance. Moreover, a Monte Carlo simulation methodology is also provided to capture the statistical effect of CNFET process imperfections on circuit-level performance. This part of the dissertation achieves the goal of evaluating CNFET performance in the presence of imperfection, using the traditional SPICE simulators HSPICE, Ultrasim. Moreover, we also provide a look up table based methodology for the fast evaluation of CNFET circuit-level delay, which can be easily integrated with the industry standard Static Timing Analysis (STA) tools such as PrimeTime.

1.2.2 Explore potential applications of CNFETs for approximate computing

In the second section, we apply the circuit-level performance evaluation methodology (developed in the first section), to common conventional circuits. Specifically, we focus on adders, which form an integral part of a wide variety of approximate computing applications, including DSP, image processing [39]. We first investigate the potential of approximate circuits in increased tolerance to process imperfections, followed by systematic methodology of generating approximate circuits for reduced glitch and delay violations due to process imperfections. This part of the dissertation achieves the goal of 1) showing the potential of approximate circuits in reducing the violations due to process imperfections, with a slight compromise in logic accuracy; 2) developing a systematic methodology of obtaining approximate circuits with reduced process-induced glitch and timing violations.

1.2.3 Explore potential applications of CNFETs for neuromorphic computing

In the last section of the dissertation, we explore the appropriateness of CNFETs for neuromorphic computing. Neuromorphic computing architectures are inspired by the structure of the human brain and consume significantly less power, in comparison to traditional Von Neumann based architectures [40, 41]. Deep Neural Networks (DNNs) have attracted significant attention for a wide range of applications including image classification [42, 43], speech recognition [44], natural language processing [45], with their hardware implementation requiring energy-efficient neuromorphic computing architectures. CNFETs have the potential to meet the growing energy efficiency demands of neuromorphic computing applications. But a careful evaluation of performance impact due to imperfection in neuromorphic architectures is required. We first investigate the role of increased process imperfection on the accuracy of DNNs. We then show that the error resilience feature of DNNs can be utilized to overcome the substantial accuracy degradation due to process imperfections.

1.3 Thesis outline

The thesis is organized as follows, with the outline shown in Figure 1.1. In Chapter 2, we include the works from literature demonstrating the potential and challenges with CNFET based circuits and systems. Specifically, we focus on device imperfections with current immature processes for the CNFET fabrication and introduce potential applications for error-resilient computing. In Chapter 3, we present the framework to incorporate the effect of CNFET process imperfections on circuit-level performance. In Chapter 4, we investigate potential applications of CNFET based circuits for approximate computing and provide a systematic methodology to obtain approximate circuits with sustained performance even in the presence of process imperfections. In Chapter 5, we present the framework evaluating the effect of process imperfections on the accuracy of DNNs. In Chapter 6, we utilize the techniques of network pruning and approximate circuit to maintain accuracy even at high process imperfection. In Chapter 7, we summarize and conclude this dissertation.

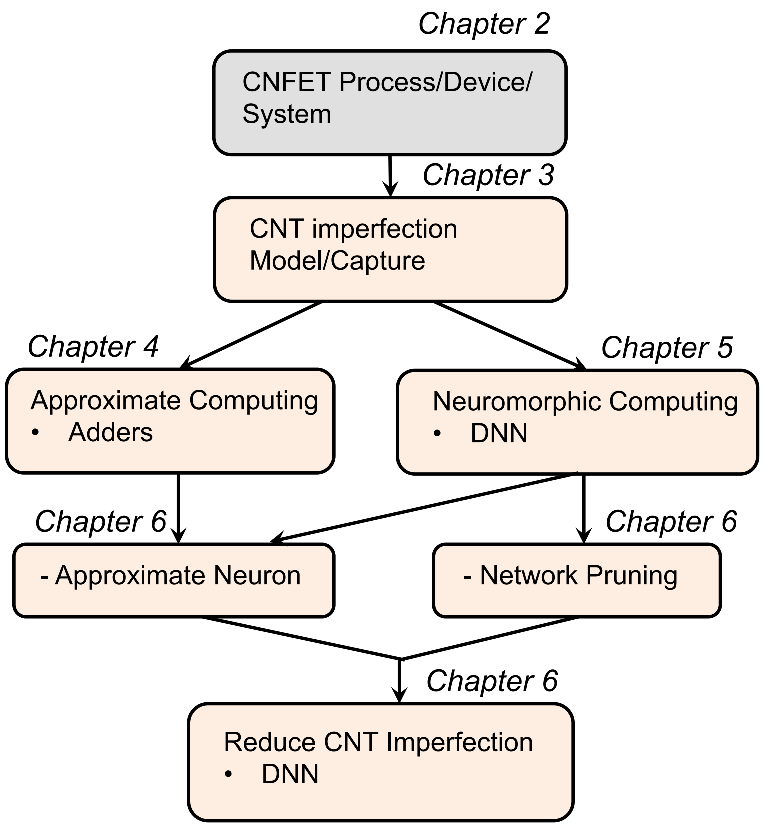


Figure 1.1: Organization of thesis.

Chapter 2

Literature Review of Carbon Nanotube FET: Process, Device, Circuits and Systems

In this chapter, we first introduce several challenges associated with fabricated CNFET devices, followed by a comparison of the two popular processes for CNFET fabrication. We then present the works in the literature related to CNFET based circuits and systems. We also briefly discuss the appropriateness of CNFETs for error-resilient computing. Eventually, we provide an overview of work in the remaining chapters of this thesis.

2.1 CNFET devices: advantages and challenges

Scaling in advanced technology nodes can be continued with CNFET devices having CNTs as conducting channel with an ultra-thin body ($\sim 1{-}2$ nm), providing excellent electrostatic gate control, reduced SCE in advanced technology nodes [30, 46]. The naturally thin body in CNT means reduced scattering, and consequently, high mobility/carrier velocity is guaranteed even at ultra-scaled dimensions [47, 35, 46]. Moreover, the strong electrostatic control and high mobility would facilitate low voltage of operation, thus enhancing energy efficiency of circuits based on CNFETs [30, 32, 33]. The experimental demonstration of CNFET scaled to 9 nm channel length was a great step showing potential of superior low voltage performance with CNFETs in sub -10 nm nodes [30]. Systems based on CNFETs have shown order of magnitude energy efficiency in comparison to competing silicon node [34, 33]. Also, NFETs and PFETs in CNFETs have nearly the same mobility resulting in similar current for the same transistor width [48], thus enabling efficient layout in comparison with silicon. Recent work also demonstrated CNFETs with a small footprint of 40 nm, indicating the potential scalability of CNFETs to advanced technology nodes [46].

Despite several advantages of CNFETs, the fabrication processes for CNFETs are still immature and face several challenges as listed below:

- **Alignment**: The CNTs in a CNFET can deviate by a certain angle (Figure 2.1). The angular deviation results in variation in CNT length and contact length, consequently causing variation in device performance [24].
- **Contact Length**: Reducing contact length to small dimensions without increasing

contact resistance is an important requirement for the scalability of CNFETs in the sub-10 nm regime [30, 35, 46].

- **Density**: The density of CNTs deposited per μm is another key factor for the performance of CNFET based circuits. Researchers at IBM presented a CNT density requirement of 125 CNTs/μm for realizing high performance general-purpose computing systems based on CNFETs [49].

- **Semiconducting purity**: The semiconducting purity is expressed as the percentage of semiconducting CNTs of the total CNTs. In general, CNT synthesis results in 33% of metallic CNTs and 66% of desired semiconducting CNTs. The metallic CNTs have to be reduced by a significant amount either pre-transfer [25, 50, 36] or post-transfer [51, 52, 53] to substrate, for realizing any meaningful logic functionality out of CNFET based circuits.

- **Missing CNTs**: The trenches connecting source and drain of CNFETs may not be covered by CNTs, leading to "open CNTs" which do not conduct current even in "on" state. The open (missing) CNTs is currently the major issue as explained in later sections (2.2.2 and 3.1).

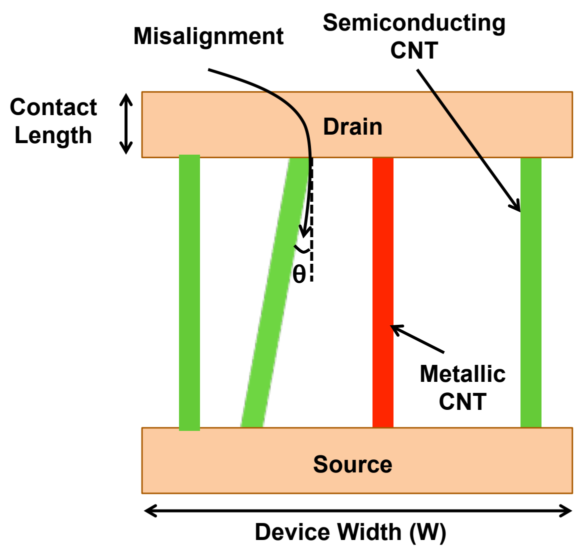


Figure 2.1: Schematic of CNFET with CNTs as conducting channels connecting source and drain.

2.2 CNFET Process

2.2.1 CVD process

Chemical Vapor Deposition (CVD) process has demonstrated the potential to grow aligned CNT arrays on crystalline substrates, such as quartz, sapphire. One of the well-known method is using pattern catalyst lines on a crystalline quartz substrate, resulting in aligned CNTs $\sim$ 99.5% [52]. However, the average density reported with single growth is low $\sim$ 5$-$10 CNTs/μm [54]. Hong et al. [55] presented a method of using multiple growth cycles, resulting in average and peak CNT density of $\sim 20-30$ CNTs/μm and 45 CNTs/μm respectively [55]. Patil et al. [56] demonstrated a CNT transfer technique to help effectively transfer CNTs to target silicon substrate using gold film, which was removed by etchant leaving behind CNTs on the silicon substrate. The CVD procedures were able to produce aligned CNTs and achieve CNT density $>$ 45 CNTs/μm. But the metallic CNT content is $\sim$ 33%, which needs to be reduced significantly using post CNT transfer technique. Joule heating is one such technique, employed by applying high voltage across the CNFETs, eventually leading to a breakdown of metallic CNTs due to high current and leaving the majority of the semiconducting CNTs nearly unaltered [56, 52]. High voltage can be detrimental as it can lead to a dielectric breakdown in a few devices, making them inoperable [57]. There are other techniques as well [58, 51] for elimination of metallic CNTs post CNT transfer. However, it has been repeatedly mentioned in ref. [57, 26] that metallic CNTs removal post CNT transfer, would result in unexpected number of semiconducting CNTs in devices, leading to undesired device to device variation. A recent technique of growing chirality specific CNTs has shown promise to avoid the problem of metallic CNTs, however the average CNT density reported is still poor $\sim$ 10 CNTs/μm [59].

2.2.2 Solution processed sorting and placement

The solution-processed sorting (purification) and placement is currently the most popular approach for high-density aligned CNT growth. As mentioned before, the raw CNT synthesis results in 33% of metallic CNTs. The first step of this method involves purification (reducing metallic CNT percentage) by dispersing CNTs in the solution and then extracting the semiconducting CNTs and leaving behind the metallic CNTs in the solution [24, 25]. Two popular approaches have been dispersing CNTs into aqueous solution using surfactant [24] and the other one utilizing conjugated polymers for CNT dispersion in solution [25]. Conjugated polymer approach is preferred because of its potential to achieve high purity and high density CNTs [25, 26]. A number of recent works have reported achieving semiconducting purity $> 99.99\%$ [50, 36]. The next step to follow after sorting is the placement of purified CNTs selectively into trenches, onto the substrate, to connect the source and drain region of CNFETs. The placement of surfactant wrapped CNTs in aqueous solution onto the patterned HfO_2/SiO_2 substrate (with HfO_2 as trenches) was carried using ion-exchange transfer [24]. In [25], the purified CNTs after sorting (purification) step were selectively placed by binding the polymer wrapped CNTs to HfO_2 trenches. In comparison to the placement of purified CNTs from the aqueous solution, the density achieved with the binding of polymer wrapped CNTs was at least two times higher [25]. With a trench width of 100 nm, the yield (trench coverage) achieved was reported to be $> 90\%$, but with scaled trench width of 50 nm, the reported yield was $> 70\%$ [25]. Figure 2.2 provides the summary of steps in sorting and placement process for CNFET fabrication.

2.3 Recent advances in CNFET based circuits and systems

With over 20 years of extensive research, significant progress has been made in digital circuits based on CNFETs, ranging from simple logic gates [56], flip flops [60], to small scale circuits such as Physical Unclonable Function (PUF) [61], True Random Number Generator (TRNG) [62] and first carbon nanotube computer (1-bit) with 178 p-type CNFETs [52], was a big step towards showing the potential of CNFETs for large scale integration. In recent years, the complexity of CNFET based systems, has witnessed tremendous growth

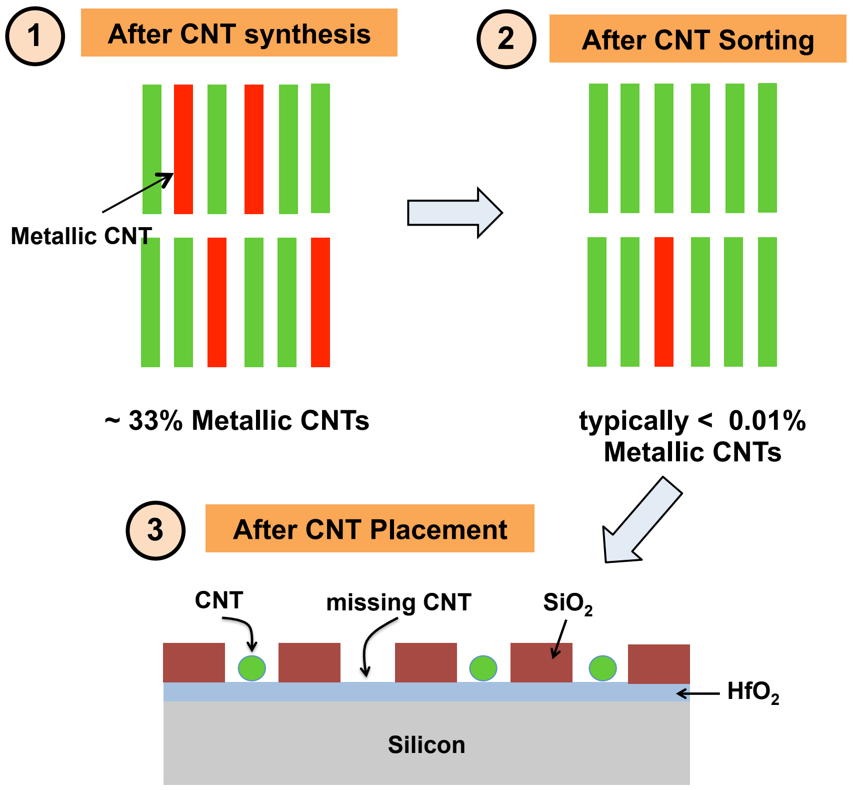


Figure 2.2: Overview of steps in sorting and placement process for CNFET fabrication.

even extending to heterogeneous systems. Some of recent heterogeneous systems include three dimensional (3D) imaging system with 2,784 CNFETs (distributed over two layers) integrated over top of silicon imager [63], 3D computing system involving > 2 million CNFETs (logic) integrated with Resistive Random-Access Memory (RRAM) (memory) and silicon [38]. A brain inspired computing system with $35\times$ EDP advantage (after place and route) in comparison to 28 nm silicon node and classification accuracy $> 98\%$ was also presented [34]. Recently, 1 Kbit SRAM with 1,024 (6,144 CNFETs in total) fully functional memory cells was also experimentally demonstrated [37]. The recent introduction of 16-bit microprocessor having $> 14,000$ CNFETs [36] is considered to be a milestone towards the potential large-scale integration with CNFET based technology.

Although the recent CNFET based systems have shown great promise, they are still far in comparison to the scale of a general-purpose computing system having billions of transistors. The reason is the immature process technology, imposing design constraint of limiting the number of stages [52, 36], avoiding some logic gate combinations [36], sizing constraint (using significantly wider transistors) [36] for ensuring no failure requirement of general-purpose computing systems. However, early technology adoption is still possible with error-resilient computing systems, having the excellent feature of inherent error tolerance. Neuromorphic computing-based architectures inspired by brain [64, 65], provides the implementation of complex Deep Neural Networks (DNNs) [66] at significantly low power. Neural networks, including DNNs, are inherently error-resilient and produce acceptable results even with slightly imprecise computations [67, 68]. The relaxation in exact computation can be utilized to have imperfection tolerant circuits and also preventing the occurrence of major failures with system-level modifications. With the complexity of

DNNs, expected to increase in future, CNFET technology (even with the immature process) can help in meeting the growing energy-efficiency requirements of the future hardware systems implementing DNNs.

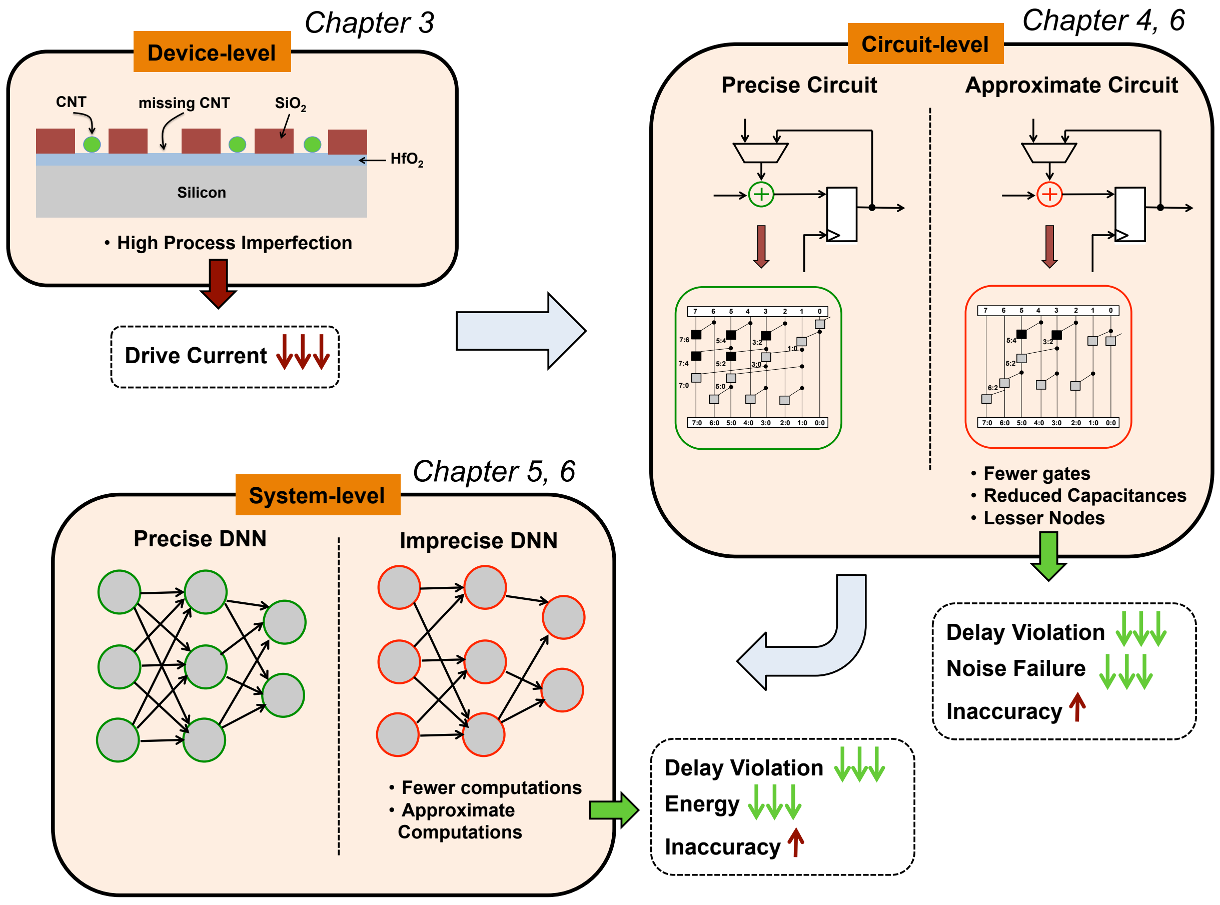


Figure 2.3: Overview of work in the thesis.

2.4 Overview of work in the thesis

In Section 2.3, we discussed several recent and prior works related to CNFET based circuits and systems. These works have focused on CNFET technology enablement for a wide variety of circuits and systems. But the work in this dissertation specifically investigates the appropriateness of CNFETs for error-resilient computing systems. Figure 2.3 provides an overview of the work in the thesis. At the device-level, we capture the effect of process imperfection for circuit-level performance evaluation. At the circuit and system level, we introduce approximate circuits and imprecise DNNs, respectively, to reduce the impact of CNFET process imperfections, with marginal degradation in accuracy. The major contributions of this dissertation are first, providing the set of methodologies for evaluating the circuit-level performance impact of process imperfections and secondly, providing the set of techniques at circuit and system level to reduce the impact of process imperfections. The techniques presented in the dissertation are not just restricted to CNFETs but can be effectively applied to other emerging materials based technologies suffering from the immature process.

2.5 Conclusions

The solution-based process of sorting and placement of CNTs is currently the most popular process for CNFET fabrication, because of high semiconducting purity $> 99.99\%$ [50, 36]. However, missing CNTs in the trenches is still $> 30\%$ for narrow trenches [25] and expected to increase further with the scaling of trench widths in the future. The recent works on CNFET based circuits and systems show great advancement in increasing complexity ($> 14,000$ CNFETs for recent reported microprocessor [36]). However, it is still quite far from the general-purpose computing systems, typically having billion of transistors. With the current process quality, CNFETs can still be appropriate for error-resilient computing systems, which can tolerate few imprecise computations and thus, can be modified to provide the necessary tolerance for increased process imperfection.

Chapter 3

Capture Effect of CNFET Process Imperfections on Circuit-Level Performance

In Chapter 2, we discussed about process imperfections arising from the immature process for CNFET fabrication. In this Chapter, we first provide the methodology to include the effect of process imperfection in Virtual-Source Carbon Nanotube Field-Effect Transistor (VSCNFET) model [69], used for SPICE simulations. The methodology is utilized to evaluate the impact of process imperfections on circuit-level performance in terms of common VLSI metrics delay, noise tolerance. A Monte Carlo simulation based framework is also proposed to accurately capture the statistical effect of process imperfections on circuit-level performance. Eventually, a Lookup Table (LUT) based methodology is introduced for fast evaluation of CNFET circuit-level delay, which can be easily integrated with industry-standard STA tools such as PrimeTime. Parts of this chapter are published in ref. [70, 71, 72].

3.1 CNFET Process Imperfections

As discussed previously, the 2-step process of sorting (purification) and placement of solution processed CNTs is currently the popular choice for CNFET fabrication. CNFETs fabricated from 2-step sorting and placement process can suffer from two major imperfections: (1) left over metallic CNTs in sorting (separation) step, resulting in "short CNTs" always conducting even under the bias of "off" state; (2) trenches connecting source and drain of CNFETs, not covered by CNTs during placement step leading to "open CNTs" which do not conduct current even in the "on" state. We now define shorthand notations $PCNT_{short}$ as percentage of short CNTs left in the solution after the completion of sorting step and $PCNT_{open}$ as probability of a trench not covered by CNT, during the placement step.

With process imperfections, a CNFET designed to have N CNTs under the gate has combination of short, open and semiconducting CNTs with

$$N_{short} + N_{open} + N_{nor} = N \tag{3.1}$$

Where N_{short}, N_{open}, N_{nor} are number of short, open and semiconducting CNTs respectively.

Given $PCNT_{open}$ and $PCNT_{short}$ from the 2-step sorting and placement process of solution-processed CNTs, we now determine the probability of a particular CNT in a CNFET to open, short or semiconducting. As explained before, the bulk of the metallic CNTs are removed in the first step(sorting). But there is still some percentage ($PCNT_{short}$) of metallic CNTs left in the solution. Thus, after the first step, among all the remaining CNTs in the solution, the percentage of metallic and semiconducting CNTs is $PCNT_{short}$ and $(1-PCNT_{short})$ respectively. In the second step(placement), the CNTs in the solution are to be placed along the trenches. But, some percentage ($PCNT_{open}$) of trenches have missing(open) CNT. Thus, among the trenches in a CNFET, the percentage of trenches having missing or filled with CNT (metallic or semiconducting) is given by $PCNT_{open}$ and $(1 - PCNT_{open})$ respectively. Hence, the probability of a particular CNT in a CNFET to be open, short, and semiconducting is given by $PCNT_{open}$, $(1 - PCNT_{open}) \cdot PCNT_{short}$, and $(1 - PCNT_{open}) \cdot (1 - PCNT_{short})$, respectively.

The probability of CNFET with N trenches to have N_{open} of open tubes and N_{short} of short tubes (and thus $N_{nor} = N - N_{open} - N_{short}$ of normal semiconducting tubes) can be expressed by trinomial distribution formula (Equation 3.2)

$$\begin{aligned} P_{N_{open},N_{short}} = \frac{N!}{N_{open}!\,N_{short}!\,N_{nor}!} \cdot PCNT_{open}^{N_{open}} \cdot \left[\,(1 - PCNT_{open}) \cdot PCNT_{short}\,\right]^{N_{short}} \\ \cdot \left[\,(1 - PCNT_{open}) \cdot (1 - PCNT_{short})\,\right]^{N_{nor}} \end{aligned} \tag{3.2}$$

which on further simplication leads to Equation 3.3 [73].

$$P_{N_{open},N_{short}} = (C_N^{N_{open}} \cdot PCNT_{open}^{N_{open}}) \cdot (1 - PCNT_{open})^{N-N_{open}} \cdot \\ (C_{N-N_{open}}^{N_{short}} \cdot PCNT_{short}^{N_{short}}) \cdot (1 - PCNT_{short})^{N_{nor}} \tag{3.3}$$

The current works report achieving semiconducting purity $> 99.99\%$ [37, 36, 74] means percentage of short CNTs $PCNT_{short} < 0.01\%$, while the percentage of open CNTs ($PCNT_{open}$) is still reported $> 30\%$ for a trench width of 50 nm [25].

As discussed in Section 2.1, a low density of placed CNTs directly limits the realization of high-performance CNFETs. From the past several years, there have been consistent efforts to reduce the width of trenches for increasing the density of placed CNTs [24, 25]. However, scaling the trench width increases the percentage of trenches (connecting source and drain of CNFETs) missing CNTs, meaning higher $PCNT_{open}$. There are other challenges of controlling CNT diameter, length variations. However, the performance of CNFETs is still primarily limited by the problem of scaling trench widths (affecting CNT density) [24, 25, 26]. With trench width expected to scale further in the future, $PCNT_{open}$ is expected to be even higher and continue to remain the major issue in the future CNFETs as well. In this work, we focus on only open CNT imperfection.

The probability of N_{open} ($P_{N_{open}}$) CNTs among N total trenches can be obtained from Equation 3.3, by ignoring the short CNT imperfection (with ($PCNT_{short} < 0.01\%$ consistently reported).

$$P_{N_{open}} = (C_N^{N_{open}} \cdot PCNT_{open}^{N_{open}}) \cdot (1 - PCNT_{open})^{N-N_{open}} \tag{3.4}$$

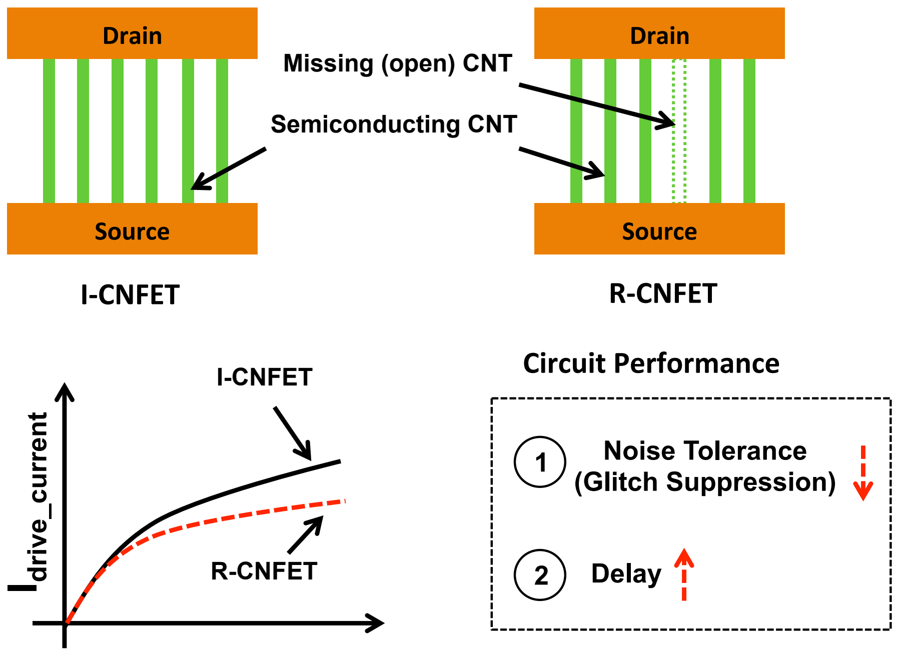


Figure 3.1: (Top Left) Ideal CNFET (I-CNFET) with no process imperfection (all semiconducting CNTs). (Top Right) Real CNFET (R-CNFET) with process imperfections (mixture of open and semiconducting CNTs). The presence of open CNTs would mean a reduced drive current for R-CNFET in comparison to its ideal case I-CNFET; consequently affecting the circuit performance with increased delay and reduced glitch suppression.

3.2 Effect of CNT imperfection on circuit-level performance

In Section 3.1, we discussed about the major source of imperfection arising from the sorting and placement process of solution processed CNTs, is the open CNT imperfection. The presence of open CNTs ($N_{open} > 0$) would have a direct impact on the total number of semiconducting tubes in a CNFET. A higher percentage of open CNT ($PCNT_{open}$) would effectively mean reduced number of semiconducting tubes for CNFETs, consequently leading to reduction in drive currents. Figure 3.1 shows an example of Ideal CNFET (I-CNFET) with all semiconducting CNTs, and Real CNFET (R-CNFET) with a mixture of open and semiconducting CNTs. I-CNFET refers to the desired case of no process imperfection ($PCNT_{open} = 0\%$); however, from the actual process (suffering from open CNT imperfection) the expected CNFET is R-CNFET (having some of percentage of open CNTs). Ignoring diameter variation, the drive current in a CNFET can be assumed to be proportional to effective number of CNTs ($N - N_{open}$) (only semiconducting CNTs). With open CNTs present in some cases, R-CNFET would have reduced drive current in comparison to I-CNFET. The reduced driving current consequently leads to degradation in circuit-level performance. In this paper, we have focused on impact of open CNT imperfection on two major circuit-level performance metrics:

1. **Noise Tolerance**: We define "Noise Tolerance" in terms of glitch suppression, as the ability of the circuit to suppress the induced glitches. The presence of glitches of significant magnitude creates a risk of logic failure (glitches of high magnitude can propagate to flop inputs, and can lead to wrong data capture in latch/flop) [75, 76]. We thus put a limit on maximum allowable glitch magnitude in the circuit and count the number of nodes in the circuit or path to the primary output failing maximum allowable glitch value. The degraded drive current in CNFETs (due to presence of open CNTs) would reduce the ability of a driving stage to suppress the glitch occurring at its output node; consequently increasing the number of nodes failing

glitch criteria.

2. **Delay**: In a circuit, the delay of the critical path has to be kept below the target value to meet the minimum frequency requirement. The presence of open CNTs would reduce the driving current in CNFETs while parasitic capacitance remains nearly unaffected, hence consequently increasing the critical path delay, which might result in failure of the circuit to meet certain target frequency value.

3.3 VSCNFET model

We have used Stanford VSCNFET model [69] for our SPICE simulations. The VSCNFET model is a semi empirical model based on virtual source concept [77]. The model is extracted from data, obtained through experiments [47, 30] and numerical simulations [78, 79]. The important model parameter Virtual Source (VS) carrier velocity is extracted from experimental data of a device, fabricated for different channel lengths [47]. The device contains a single CNT (the CNT is grown on quartz substrate and transferred to silicon substrate) as channel, with Palladium (Pd) as Local Bottom Gate (LBG), HfO_2 as dielectric and Pd as drain/source contacts [47].

The VSCNFET model provides prediction for device behavior with scaling of dimensions for future sub-10nm nodes [78, 79]. For our analysis, we have used the default parameter values provided in the Table 3.1 [69], if not otherwise specified. The parameters in Table 3.1 are provided for a projected CNFET device (Figure 3.2) at 5 nm technology node, with contacted gate pitch of 31 nm [79].

It should be noted that the work in the thesis is not just applicable for the VSCNFET model but can be easily extended to other improved CNFET models, calibrated to experimental data in the future technology nodes.

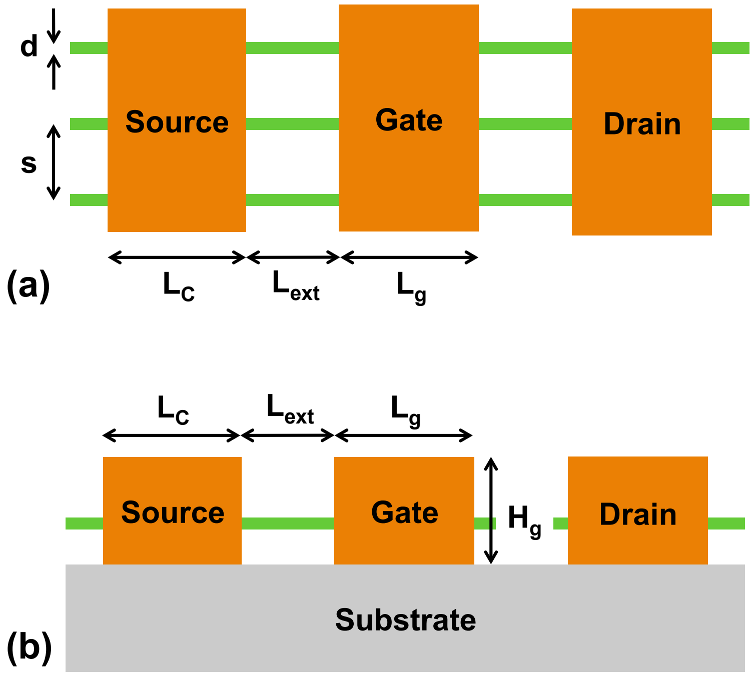


Figure 3.2: Schematic showing (a) top, (b) front view of a CNFET device with some of dimensions parameters used in VSCNFET model.

3.4 Capture open CNT imperfection in SPICE

VSCNFET model considers all CNTs as semiconducting CNTs. So, we use a modified version of VSCNFET model to capture the effect of open CNT imperfection. As discussed before, the effect of N_{open} is reduction in effective number of CNTs ($N - N_{open}$) in a CNFET. One way to model N_{open} in SPICE is to effectively change the number of CNTs for a given

Name	Description	Value
Vdd	Supply voltage	0.71 V
Lg	Physical gate length	11.7 nm
Lc	Contact length	12.9 nm
Lext	Source/drain extension length	3.2 nm
Hg	Gate height	20 nm
d	CNT diameter	1.2 nm
tox	Gate oxide thickness	3 nm
kox	Gate oxide dielectric constant	23
kcnt	CNT dielectric constant	1
ksub	Substrate dielectric constant	3.9
kspa	Spacer dielectric constant	7.5
Efsd	fermi level to band edge at source/drain related to doping density	0.258 eV
Vfb	flat band voltage (for threshold voltage adjustment)	0.015 V
Geomod	device geometry 1 : cylindrical gate-all-around 2 : top-gate with charge screening effect 3 : top-gate without charge screening effect	1

Table 3.1: VSCNFET parameter values [69].

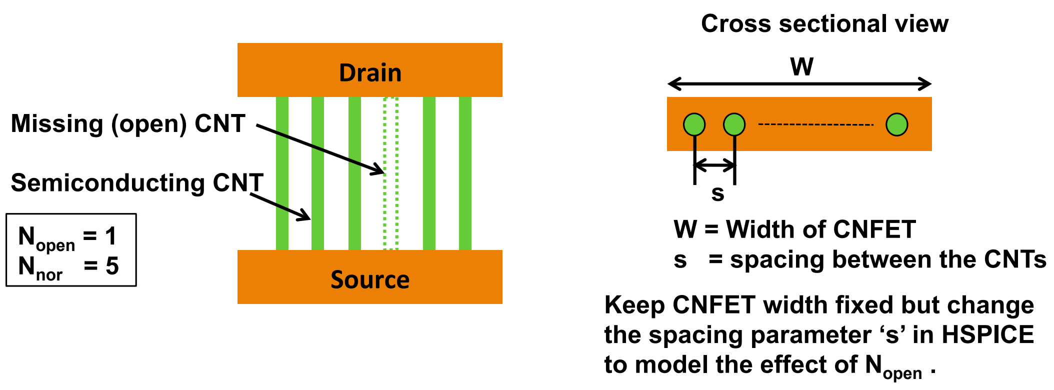

Drain
Missing (open) CNT
Semiconducting CNT
N_open = 1
N_nor = 5
Source
Cross sectional view
W
s
W = Width of CNFET
s = spacing between the CNTs
Keep CNFET width fixed but change the spacing parameter 's' in HSPICE to model the effect of N_open .

Figure 3.3: (Left) Schematic of a CNFET with process imperfections ($N = 6$, $N_{open} = 1$, $N_{nor} = 5$). (Right) Cross-sectional view of the CNFET with W = Width of the CNFET, s = spacing between the CNTs in CNFET. The presence of N_{open} can be effectively modeled in HSPICE by changing the spacing parameter 's', which effectively modifies the number of semiconducting CNTs under the CNFET for a given width 'W'. No modification in 'W' would mean the parasitic capacitance is nearly unaffected by this technique to model N_{open}.

width of the CNFET, by changing the spacing parameter 's' (spacing between CNTs for SPICE). Increasing 's' would effectively reduce the number of semiconducting CNTs ($N - N_{open}$), but keeping the width constant avoids changes to the parasitic capacitance. Figure 3.3 shows example of CNFET suffering from process imperfections with $N = 6$, $N_{open} = 1$. The CNFET can be modeled in SPICE by changing 's' such that number of semiconducting CNTs are $N - N_{open} = 5$.

The stated method to capture N_{open} would be accurate for the case where overall current of the CNFET is mainly affected by number of semiconducting CNTs ($N - N_{open}$) but least affected by the spacing between the CNTs. We define term 'I_{on}' as the on current of CNFET ($|Vgs| = Vdd$, $|Vds| = Vdd$) where Vgs, Vds, and Vdd are the gate to source, drain to source, and supply voltages, respectively. Figure 3.4(a) shows I_{on} for Gate

All Around (GAA) and Top Gate (TG) configuration each for three different CNFET widths with spacing in the ratio of (1 : 2 : 5) respectively for a given $N - N_{open}$. The main difference between GAA and TG is how gate controls the channel CNT, with gate completely surrounding CNT in GAA [49] and only from the top in TG [80]. GAA(s=1x), TG(s=1x) represent the base cases with the nominal spacing values while for the other cases GAA/TG(s=2x) and GAA/TG(s=5x), having spacing two and five times compared to the base case GAA/TG(s=1x) for a given $N - N_{open}$. For GAA configuration, we observe no change in I_{on} among the CNFETs for given $N - N_{open}$ (Figure 3.4(b)). However, for TG configuration, there is difference observed in I_{on} for different spacing CNFETs. The difference at a given spacing is expressed with respect to the base case (s=1x) as $RelativeI_{on}$ = [I_{on} - I_{on} (s=1x)]/I_{on}(s=1x), where I_{on} is the on current at the given spacing value. $RelativeI_{on}$ becomes more appreciable for higher value of $N - N_{open}$ (For a given CNFET, the charge screening effect becomes more prominent for higher $N - N_{open}$ or reduced spacing between CNTs). However, even for case TG(s=5x), the $RelativeI_{on} < 1.5\%$ is observed, revealing no significant change in I_{on} with spacing. Without loss in generality, the further simulation results shown are with GAA device configuration.

3.5 Monte carlo simulation for capturing statistical effect of open CNT imperfection

N_{open} can be different in CNFETs with same width, even for the same overall percentage of open CNTs ($PCNT_{open}$). Figure 3.5 provides an overview of Monte Carlo simulation based

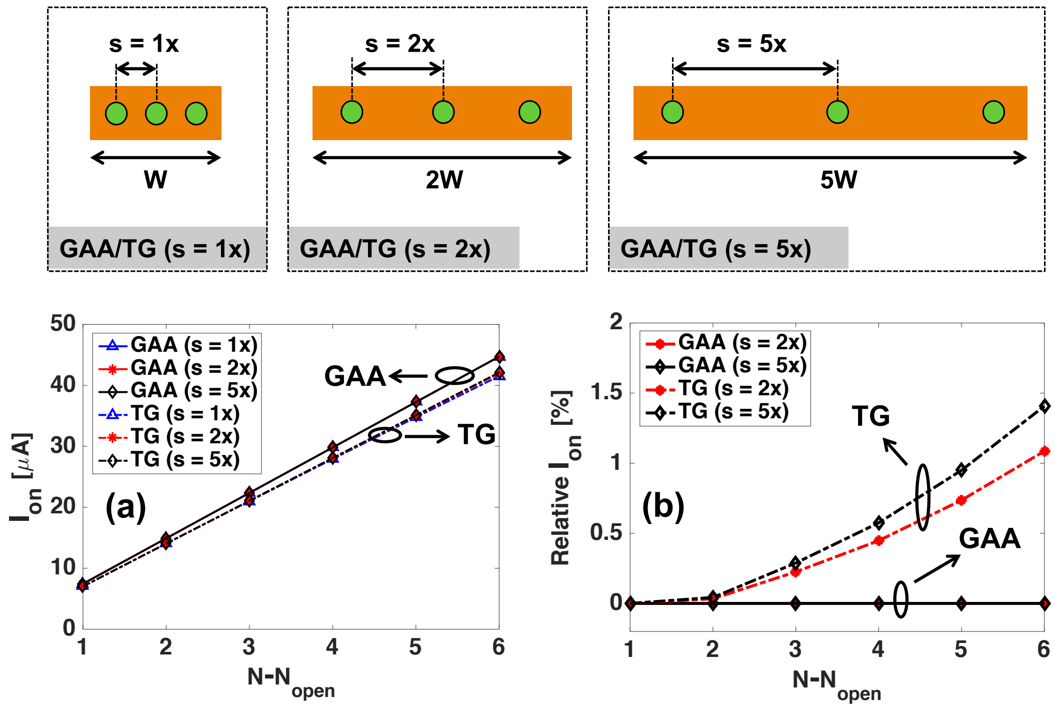


Figure 3.4: (Top) Cross-sectional view for three CNFETs with GAA or TG configuration, represented as GAA/TG(s=1x), GAA/TG(s=2x), GAA/TG(s=5x). Each of the three CNFETs have spacing 's' and width 'W' in the ratio of 1 : 2 : 5 respectively. The number of semiconducting CNTs shown is just for illustration. (Bottom) (a) I_{on} plotted as a function of $N - N_{open}$ for three different CNFETs (width and spacing both in the ratio of 1 : 2 : 5 respectively), for both GAA and TG configuration. At each $N - N_{open}$, I_{on} is the same across the three CNFETs with GAA configuration, but slightly different for CNFETs with TG configuration, (b) $Relative I_{on}$ plotted for each GAA and TG configuration for (s=2x) and (s=5x).

methodology to accurately capture the statistical effect of open CNTs. The methodology involves the generation of seeds for the Monte Carlo run (Each trial or sample of the Monte Carlo run is referred to as seed in this work), followed by SPICE simulation for each such seed.

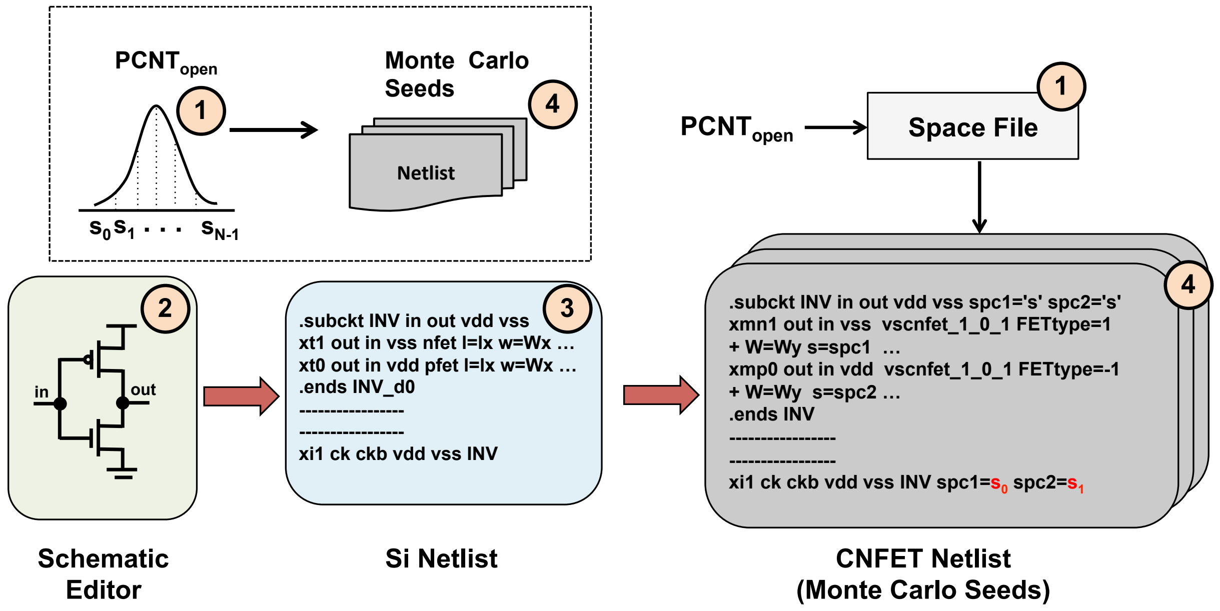
At a given $PCNT_{open}$, we first obtain list of possible spacing values (captured in $SpaceFile$ (Figure 3.5)) for each CNFET in the circuit, corresponding to the $PCNT_{open}$. $s_0, s_1, s_2, \ldots, s_{N-1}$ in Figure 3.5, are list of spacing values corresponding to given $PCNT_{open}$. Our simulation framework utilizes existing Silicon (Si) based library for the schematic generation in Cadence Virtuoso, followed by generating Si based netlist with connectivity information (Figure 3.5). We have provided codes in Section C.1 that can utilize the Si based netlist with the connectivity information and generate multiple seeds with each CNFET in the netlist being assigned from the obtained spacing values, to generate multiple copies of original netlist each acting as seed for Monte Carlo run (Figure 3.5).

Figure 3.6 shows the procedure for generation of $SpaceFile$ at given $PCNT_{open}$. The numbers obtained from a pseudo random number generator (uniform distribution between 0 to 1) are mapped onto the cumulative distribution C_k (Equation 3.5) and each corresponding spacing value is stored in $SpaceFile$ (Figure 3.6).

$$C_k = \sum_{i=0}^{k} P_{N_{open}=i} \tag{3.5}$$

where C_k represents the cumulative distribution for $N_{open} <= k$ (k = *0, 1, 2, ..., N-1*). The expression for $P_{N_{open}}$ is provided in Equation 3.4.

Each seed for the Monte Carlo run can be generated by assigning space values (from



```
.subckt INV in out vdd vss
xt1 out in vss nfet l=lx w=Wx …
xt0 out in vdd pfet l=lx w=Wx …
.ends INV_d0
-----------------
-----------------
xi1 ck ckb vdd vss INV
```

```
.subckt INV in out vdd vss spc1='s' spc2='s'
xmn1 out in vss  vscnfet_1_0_1 FETtype=1
+ W=Wy s=spc1  …
xmp0 out in vdd  vscnfet_1_0_1 FETtype=-1
+ W=Wy  s=spc2 …
.ends INV
-----------------
-----------------
xi1 ck ckb vdd vss INV spc1=s0 spc2=s1
```

Figure 3.5: Overview of Monte Carlo Seed generation. Multiple copies of the circuit netlist are created (each acting a seed/sample for Monte Carlo run), with the spacing (between CNTs in CNFET) of CNFETs having statistical distribution corresponding to the given $PCNT_{open}$.

$SpaceFile$ at given $PCNT_{open}$) to spacing parameter "s" of each transistor in the circuit netlist. Similarly, multiple copies of the netlist can be generated, differing in space values assigned to the transistors. The number of space values in $SpaceFile \geq \#seeds * \#transistors$, where $\#transistors$ are the total transistors in the netlist and $\#seeds$ are the total seeds for the Monte Carlo run. In this thesis, we have used 100 seeds for the Monte Carlo run if not specified.

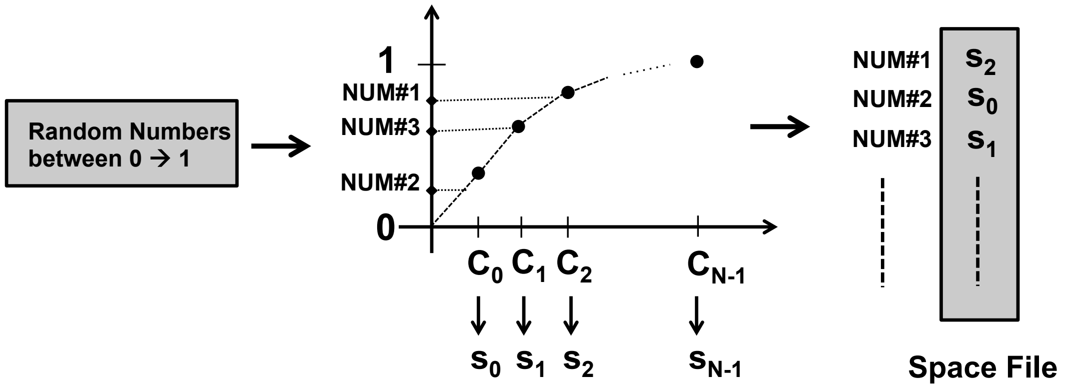


Figure 3.6: Generation of the $SpaceFile$ at given $PCNT_{open}$, involves mapping of the random numbers ($NUM\#1, NUM\#2, NUM\#3, \ldots$ etc) from a uniform random number generator on to y-axis of cumulative distribution function corresponding to the particular $PCNT_{open}$. The next highest cumulative distribution point closest to each mapped random number is chosen and corresponding space value is stored in $SpaceFile$ (e.g. $NUM\#1, NUM\#2, NUM\#3$ would result in selection of C_2, C_0, C_1 respectively, eventually leading to storing space values s_2, s_0, s_1 respectively in the $SpaceFile$). Cumulative distribution points $C_0, C_1, C_2, \ldots, C_{N-1}$ corresponds to spacing values $s_0, s_1, s_2, \ldots, s_{N-1}$ respectively.

3.6 Methodology for evaluating CNFET performance

3.6.1 Noise tolerance

Glitch circuit setup

We now explain our circuit set up for glitch simulation, by referring to example in Figure 3.7. Figure 3.7 shows the example where aggressors $Aggr1$, $Aggr2$, are attacking nodes $n1$, $n2$ (victim nodes) in the main circuit respectively. In the absence of aggressors, the node $n1$, $n2$ would have voltages close to VDD, GND respectively for the given input vector combination [00]. However, the rising and falling transitions at inputs of $Aggr1$, $Aggr2$ respectively would result in glitches at both $n1$, $n2$. The size of glitches at both nodes $n1$, $n2$ depends on multitude of factors including coupling capacitances, aggressors type/width, timing, transition, drive currents etc [81, 82]. In our circuit set up for glitch simulation, we assume our main circuit is under DC condition and aggressor (external to main circuit) of similar type/drive strength is attacking each node in the circuit. It is further assumed, that each aggressor is having an input transition (rise/fall) to result in glitches at the victim node. Practically, the coupling capacitances and aggressors depend upon the actual layout. However, without the layout information, it is still reasonable to say that the coupling capacitance at the output of victim driver is expected to be more with more number of gates connected to that output. Without the loss in generality, we assume coupling capacitance at each node to be $0.5fF/\mu m$. Moreover, if actual layout information is available, the rest of the simulation framework still remains the same.

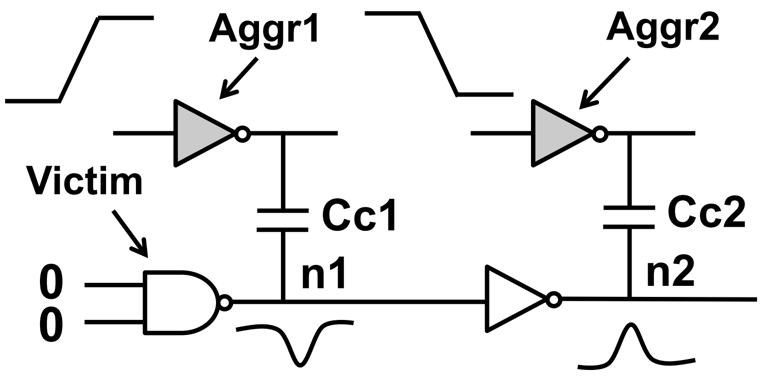


Figure 3.7: Schematic shows the case of aggressors ($Aggr1$, $Aggr2$) (shaded in grey), inducing glitches at victim nodes $n1$, $n2$ respectively. In the absence of aggressors, the node voltages at $n1$, $n2$ would be close to VDD, GND respectively. However, with the aggressors and given input transitions, the induced glitches at nodes $n1$, $n2$ would cause the voltage to go below VDD and above GND respectively.

Glitch Monte Carlo simulation setup

We now provide the simulation framework for obtaining peak glitch magnitude at each internal/output node of the circuit, at given $PCNT_{open}$ (Figure 3.8). In our current framework, each node in the main circuit (excluding primary inputs) is assumed to be attacked by aggressor (external to main circuit). Thus, each internal/output node is also termed as victim node. The simulation setup for finding peak glitch magnitude at each victim node, involves two main phases:

1. **DC sim phase**: This phase helps in finding out the transition type (rise/fall) at the input of aggressor to result in glitches at each internal/output (victim) node in the main circuit (Figure 3.8). e.g. In order to induce glitches at node $n1$ and $n2$, the transition type at input of $Aggr1$, $Aggr2$ should be rise, fall respectively, for input vector combination [00] (Figure 3.7).

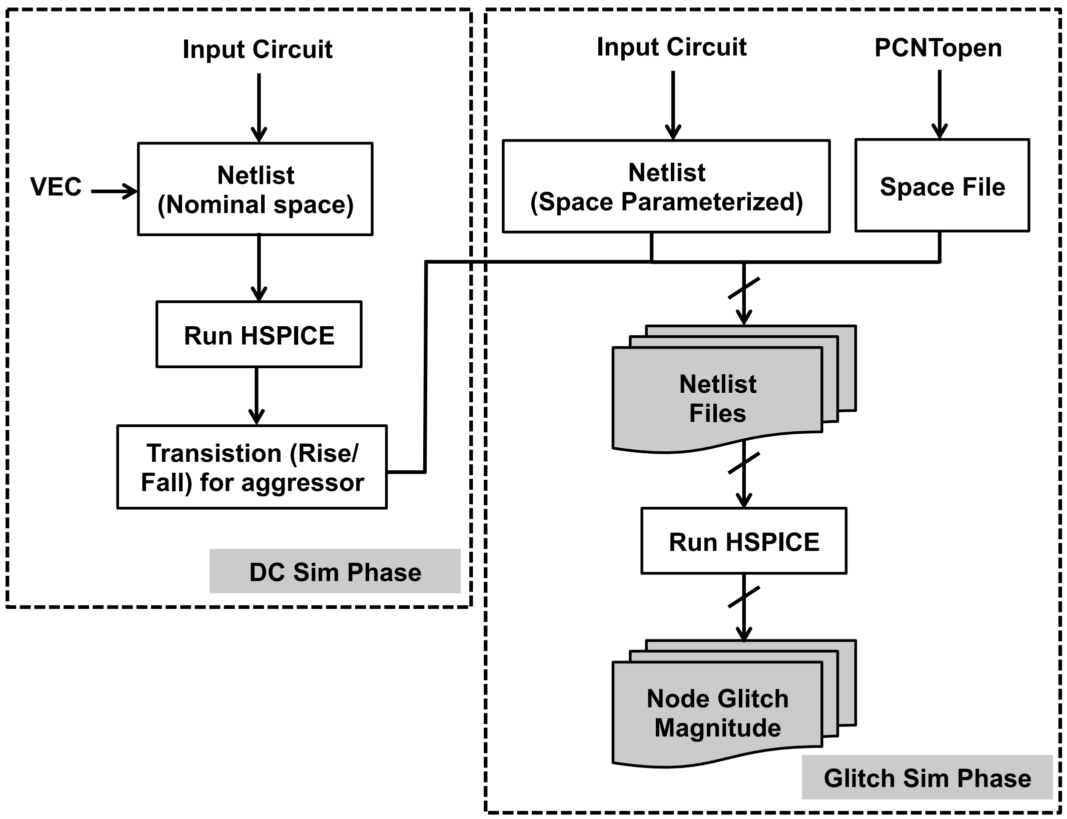


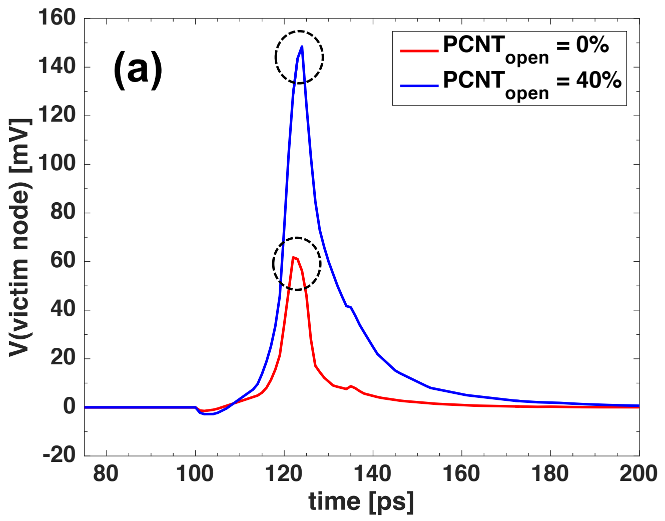
Figure 3.8: Steps to compute peak glitch magnitude for each internal/output (victim) node in the circuit involving transient Monte Carlo (MC) HSPICE simulation, at a given $PCNT_{open}$.

2. **Glitch sim phase**: In this phase, we first assign the spacing values from the $SpaceFile$ (at given $PCNT_{open}$), to "s" parameter of each CNFET in the netlist, for obtaining multiple copies of the netlist which act as seeds for the Monte Carlo run. Then we run transient HSPICE simulation to obtain peak glitch magnitude for each victim node in the circuit, for each seed (Figure 3.9). Figure 3.9 shows the instance of transient simulation by plotting voltage at victim node $S2$ of 4-bit RCA (Figure 3.10). At $PCNT_{open} = 40\%$, the peak glitch magnitude at node $S2$ varies between $\sim 70\,mV$ and $\sim 154\,mV$ over 100 seeds, showing the statistical effect of open CNT imperfection on glitch magnitude (Figure 3.9(b)).

Figures of merit

For our analysis, we set a criterion for maximum allowable glitch ($V_{peak\_glitch\_limit}$) of $0.175\,V$ ($= 0.25\,VDD$). For a given seed, if peak magnitude of glitch at a node exceeds $V_{peak\_glitch\_limit}$, we say that node failing glitch criteria for that seed. In order to compare circuit in terms of glitch vulnerability, we define few more terms: (i) $MeanFailNodes_{PATH}$ as the number of nodes (excluding primary inputs) along a path to the primary output in the circuit, failing glitch criteria, averaged over total number of seeds. (ii) $MeanFailNodes_{CKT}$ as the total number of nodes in the circuit, failing glitch criteria, averaged over total number of seeds.

$$MeanFailNodes_{PATH} = \frac{1}{S}\sum_{i=1}^{S} \#FailNodes_{PATH}(Seed_i) \tag{3.6}$$



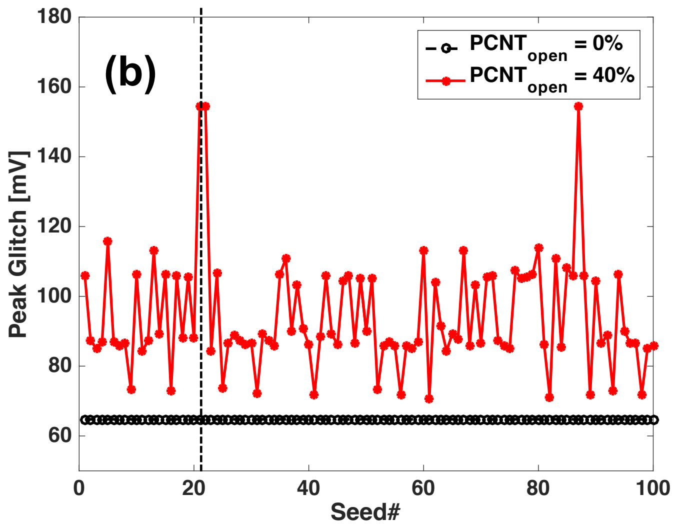


Figure 3.9: (a) Voltage at a victim node $S2$ of a 4-bit RCA circuit (Figure 3.10) resulting from the transient HSPICE simulation for seed 21 of Monte Carlo run for $PCNT_{open} = 0\%$ and 40%. The peak of the glitches (highlighted by dotted circles) at $PCNT_{open} = 0\%$ and 40%, have magnitude $\sim 64\,mV$ and $\sim 154\,mV$ respectively. (b) Peak glitch magnitude at the victim node $S2$ for 100 seeds ($Seed\#0$ to $Seed\#99$) for $PCNT_{open} = 0\%$ and 40%. At $PCNT_{open} = 0\%$, the peak glitch magnitude is the same across the seeds; however at $PCNT_{open} = 40\%$, the peak glitch magnitude vary between $\sim 70\,mV$ and $\sim 154\,mV$.

$$MeanFailNodes_{CKT} = \frac{1}{S}\sum_{i=1}^{S} \#FailNodes_{CKT}(Seed_i) \tag{3.7}$$

where S is the total number of seeds ($S = 100$ in this work), $\#FailNodes_{PATH}$, $\#FailNodes_{CKT}$ are the total number of failing nodes along a path and circuit respectively. $MeanFailNodes_{CKT}$ provides an estimate of average number of nodes in the circuit failing glitch criterion, without providing the specific details of which path or portions of the circuit are more probable to fail glitch criterion. However, a more important metric is $MeanFailNodes_{PATH}$, which provides an estimate of which primary output of the circuit is more vulnerable to effect of induced glitches. Often the primary outputs in the combinational circuit are being captured by flops. Glitch at the primary output can result in wrong data capture at a clock cycle and the correct value cannot be restored till the next clock cycle [75, 76].

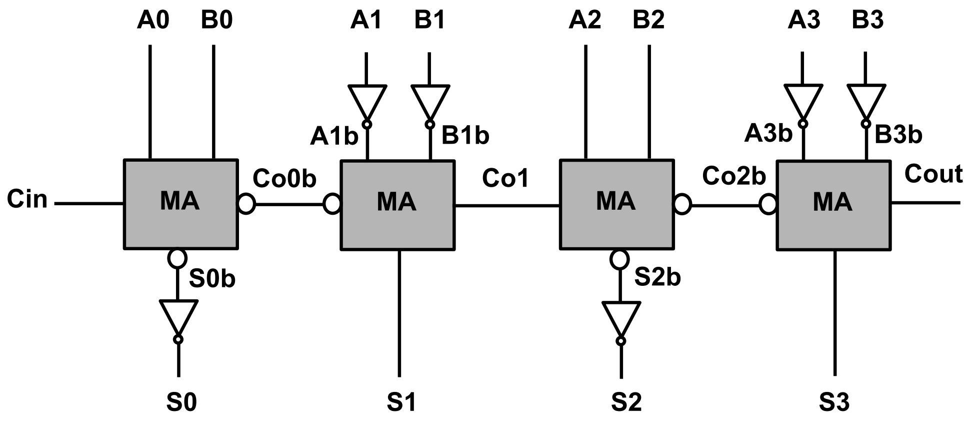


Figure 3.10: Schematic shows 4-bit RCA with each (primary, internal, output) node shown. $S3$ with highest $LN\#$ ($LN\# = 9$), is expected to have highest $MeanFailNodes_{PATH}$ among the primary outputs.

We define a term "Linked Node Number" ($LN\#$) as the total number of nodes (except primary inputs), along all the paths from the contributing primary inputs to that output. Figure 3.10 shows the schematic of 4-bit RCA with all nodes (primary, internal, output) nodes highlighted. Each of the internal nodes ($Co0b, A1b, B1b, Co1, Co2b, A3b, B3b, Cout$) and $S3$, can contribute to $\#FailNodes_{PATH}$ for $S3$ and their total count (9) correspond to $LN\#$ for $S3$. Similarly, the $LN\#$ for $S0, S1, S2$, and $Cout$ is 3, 5, 7 and 8 respectively. $MeanFailNodes_{PATH}$ holds a similar relation to $LN\#$ ($MeanFailNodes_{PATH} \leq LN\#$), as $MeanFailNodes_{CKT}$ to total nodes (except primary inputs) in the circuit. The relation simply means that $MeanFailNodes_{PATH}$ for a primary output cannot exceed $LN\#$ for that output.

3.6.2 Circuit-level Delay

In this section, we present the methodology to link open CNT imperfection ($PCNT_{open}$) with the circuit-level delay. The imperfect CNFETs ($N_{open} > 0$) would have reduced driving current, but as mentioned before, the parasitic capacitances would stay the same as ideal CNFETs ($N_{open} = 0$); resulting in overall increase in circuit-level delays with increase in $PCNT_{open}$ compared to CNFETs without process imperfection ($PCNT_{open} = 0$).

In general, the delay of the circuit depends on multitude of factors like input slopes, output loads, arc etc. In addition to this, different possible combinations of N_{open} for CNFETs make circuit-level delay computation for CNFET circuits even more computationally expensive, when all of the mentioned factors have to be considered. Hence, a methodology that could avoid running the dynamic HSPICE simulation for the whole circuit should be provided. STA has been used extensively for several years, to perform timing analysis of complex circuits, with significantly less run-time. STA is based on finding the cell libraries consisting of output Delay/Slope LUTs for the gates, as function of input slope and output load [83, 84]. However, compared to traditional STA, the delay methodology even need to consider N_{open} in gates while performing delay computation for CNFET based circuits. The delay for the CNFET circuits is computed in following steps:

1. **Pre-characterization**: HSPICE simulations are conducted to obtain realistic waveforms with certain input slope values. These waveforms are utilized in the characterization step for obtaining Delay/Slope LUTs for the different gates in the circuit.

2. **Gate arc capture**: The delay of a gate varies among the different possible input to output arcs for that gate. In this step, we identify all the possible input to output arcs of the gate e.g. for an inverter with 'a' input, 'o' output, both $a(0 \rightarrow 1)$, $o(1 \rightarrow 0)$ and $a(1 \rightarrow 0)$, $o(0 \rightarrow 1)$ is considered. In the characterization step, the Delay/Slope LUTs are to be obtained for each such possible arc.

3. **Characterization**: In this step, the Delay/Slope LUTs are obtained for each arc of the gate determined in step 2. For a given arc, HSPICE simulations are conducted considering each combination of (input slope, output load, N_{open}) where N_{open} range from 0 to $N-1$. Figure 3.11 shows the steps to obtain Delay/Slope LUTs for single arc of the input circuit/gate under consideration. The simulations provide $Delay_{N_{open}}$ $/Slope_{N_{open}}$ LUTs at different N_{open} ($Delay_{N_{open}}$, $Slope_{N_{open}}$ are the Delay, Slope LUTs respectively, at given N_{open}), which can be combined to obtain Delay/Slope LUTs for a particular $PCNT_{open}$. $Delay_{PCNTopen}$ (delay at particular $PCNT_{open}$)

is given by Equation 3.8. In a similar manner, an expression can be obtained for $Slope_{PCNTopen}$ (Slope at particular $PCNT_{open}$).

$$Delay_{PCNTopen} = \frac{\sum_{Nopen=0}^{N-1}(Delay_{N_{open}}.P_{N_{open}})}{\sum_{Nopen=0}^{N-1} P_{N_{open}}} \tag{3.8}$$

4. **Delay calculation**: Starting from the primary inputs, the Delay/Slope LUTs are utilized to obtain Delay/Slope values at each gate output. Delay at output of each gate is sum of delay of the late arriving input and the delay of the gate itself for the slope at that input and load at the gate output. This is continued till primary output is reached.

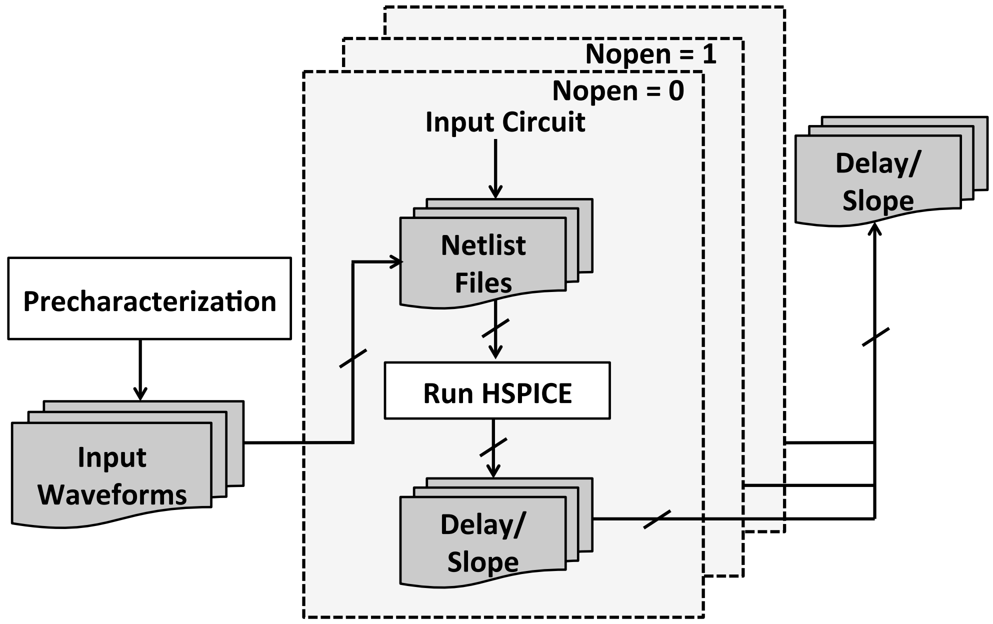


Figure 3.11: Steps to obtain Delay/Slope LUTs for single arc of an input circuit.

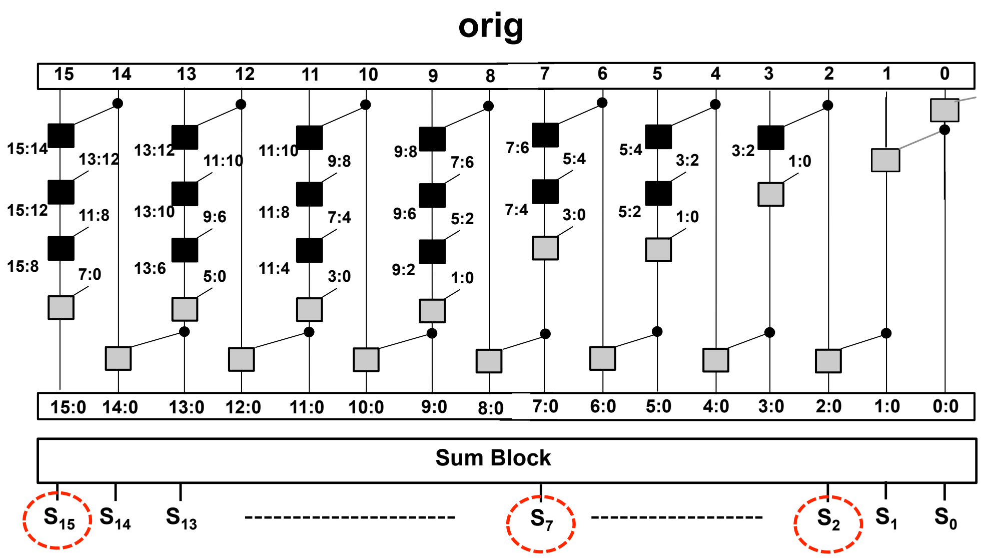


Figure 3.12: Schematic of 16 bit precise Han Carlson Tree Adder (*orig*) with Sum block formed using XOR gates to generate sum signals from S_0(LSB) to S_{15}(MSB). S_2, S_7, S_{15} are highlighted by dotted circles.

3.6.3 16-Bit Han Carlson CNFET adder

Next we apply the methodology to evaluate the performance degradation in CNFET circuits. Without the loss in generality, we have taken 16-bit Han Carlson Adder (Figure 3.12) for case study. For the purpose of fast addition, parallel prefix adders are commonly used; however they might suffer from increased energy and area compared to other adder topologies. We have chosen Han Carlson Adder [85] as it is one among the several prefix parallel adders with comparable performance with other prefix parallel adders, and at the same time reasonable power consumption. The precise 16-bit Han Carlson Adder has a bit-wise propagate/generate block (not shown in Figure 3.12), followed by Han Carlson Tree, and sum block providing the sum outputs from S_0(LSB) to S_{15}(MSB) (Figure 3.12). The adder schematic in (Figure 3.12) follow the conventional notation in [86], with black and grey (types of group generate/propagate) cells. The black cells in (Figure 3.12) compute both group generate and propagate signals. However, the gray cells only compute group generate signal. The circuits for group generate/propagate are the same as provided in [86].

$MeanFailNodes_{PATH}$ for 16-bit precise Han Carlson Adder ($orig$) is plotted as function of $PCNT_{open}$ (Figure 3.13 (a)). Figure 3.13 (a) shows that $MeanFailNodes_{PATH}$ for each of the outputs S_2, S_7, S_{15} of $orig$ increases with increase in $PCNT_{open}$. $MeanFailNodes_{PATH}$ is quite small (< 0.5) for $PCNT_{open} \leq 10\%$ but significant at $PCNT_{open} = 40\%$. Except at lower $PCNT_{open}$ where $MeanFailNodes_{PATH}$ is negligible, S_{15} has the highest $MeanFailNodes_{PATH}$ (Figure 3.13 (a)). Figure 3.13 (b) shows that the $LN\#$ increases for the sum outputs LSB towards MSB, with S_{15} having the highest $LN\# = 64$. A direct relation is observed between $MeanFailNodes_{PATH}$ and $LN\#$; S_{15} having the highest $LN\#$ has the highest $MeanFailNodes_{PATH}$ at given $PCNT_{open}$ in comparison to S_7 ($LN\# = 34$) and S_2 ($LN\# = 14$). At $PCNT_{open} = 40\%$, the $MeanFailNodes_{PATH}$ for S_{15} is 36.1, in comparison to $MeanFailNodes_{PATH}$ for S_7, S_2 which is 19.8, 7.9 respectively (Figure 3.13 (a)). For $orig$, $MeanFailNodes_{PATH}$ of S_{15} is highest and S_{15} is also the one with highest $LN\#$ ($LN\# = 64$). We thus define term critical primary output as the primary output providing highest $MeanFailNodes_{PATH}$ or probably the one with highest $LN\#$.

We have also plotted the worst-case Delay (among a set of 100 random vectors) for S_2, S_7 and S_{15} of $orig$ (Figure 3.14). The Delay values are normalized to worst-case Delay of S_{15} (at $PCNT_{open} = 0\%$). With increase in $PCNT_{open}$, Delay for each of the output S_2, S_7 and S_{15} increases, which was expected because of reduced drive currents at high

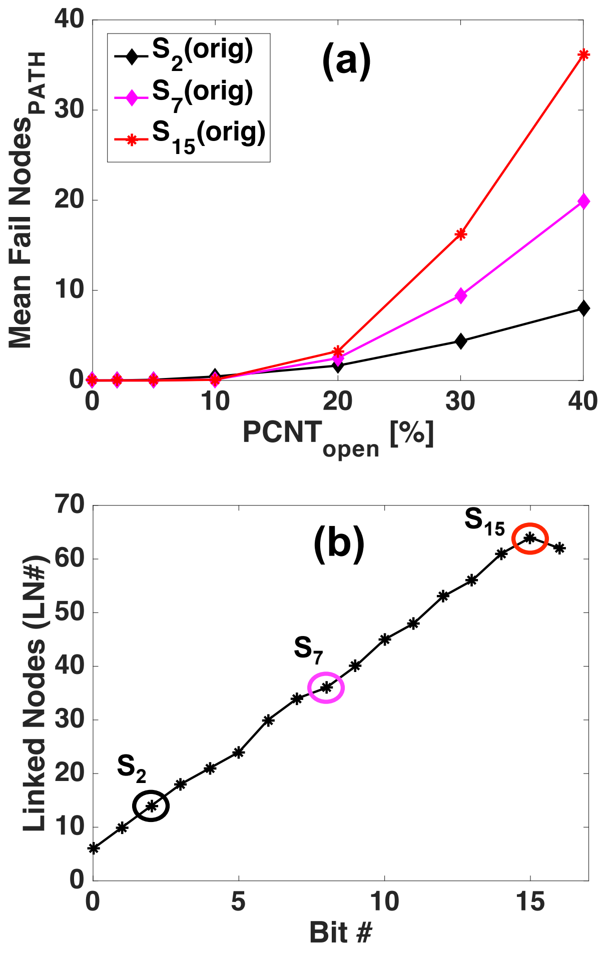


Figure 3.13: (a) $MeanFailNodes_{PATH}$ plotted for $S_2(orig)$, $S_7(orig)$, $S_{15}(orig)$ as function of $PCNT_{open}$. (b) $LN\#$ of $orig$ plotted for each sum output from S_0(LSB) to S_{15}(MSB) including C_{out}. The x-axis is the bit position with $Bit\#(0, 1, \ldots, 15, 16)$ representing $S_0, S_1, \ldots, S_{15}, C_{out}$ respectively. S_{15} with the highest $LN\#$ has expectedly highest $MeanFailNodes_{PATH}$ at a given $PCNT_{open}$.

$PCNT_{open}$. S_{15} encounters more stages (in critical path) in comparison to S_2 and S_7, and expectedly has the worst delay among S_2, S_7 and S_{15}. Among S_2, S_7 and S_{15}, S_2 encounters least number of stages and has the least delay (39.7% less in comparison to S_{15} at $PCNT_{open} = 0\%$). But at $PCNT_{open} = 40\%$, the Delay for S_2 is even 8.4% more than Delay of S_{15} at $PCNT_{open} = 0\%$. S_7 Delay is only 5.5% less in comparison to S_{15} at $PCNT_{open} = 0\%$, which is expected as S_7 and S_{15} encounters nearly similar number of stages from the primary inputs.

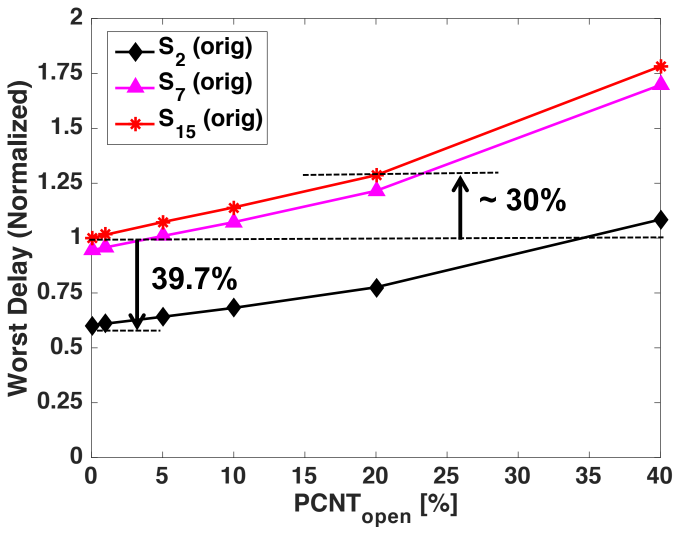


Figure 3.14: Worst Delay (among 100 random input vectors) for $S_2(orig)$, $S_7(orig)$ and $S_{15}(orig)$ plotted as function of $PCNT_{open}$. The Delay values are normalized to that of $S_{15}(orig)$ with $PCNT_{open} = 0\%$. Among S_2, S_7 and S_{15}, S_2 has the least delay (39.7% less in comparison to S_{15} at $PCNT_{open} = 0\%$). S_{15} has worst delay with delay degradation of $\sim 30\%$ at $PCNT_{open} = 20\%$ in comparison to delay at $PCNT_{open} = 0\%$.

3.7 Conclusions

We present modifying spacing between CNTs in a CNFET as an effective way to model the effect of open CNT imperfection in circuit-level performance evaluation, using SPICE simulations with VSCNFET model [69]. A simulation framework is also provided to create CNFET based Monte Carlo seeds, from netlist containing circuit connectivity. Linked Nodes ($LN\#$) associated with each primary output provides an effective link between noise tolerance and open CNT imperfection. For a 16-bit Han Carlson adder, the primary output towards the MSB (S_{15}) has the highest number of nodes failing glitch criteria in comparison to other primary outputs. S_{15} has 36.1 nodes failing glitch criteria, in comparison to 19.8, 7.9 nodes failing glitch criteria in S_7, S_2 respectively. This is directly linked to $LN\#$, with S_{15} having $LN\# = 64$, in comparison to $LN\# = 34$, 14 in S_7, S_2 respectively. Thus, high $LN\#$ with a primary output is linked to the probability of more failing nodes along the path to the output.

Chapter 4

Carbon Nanotube FET — Appropriateness for Approximate Computing

4.1 Introduction

Recently, the area of approximate computing has attracted widespread interest among the research community. Approximate computing targets a wide range of applications having inherent error resilience, including signal processing, data mining, machine learning, image, video processing, etc [87, 88]. These applications can tolerate errors during computation because they can involve computations in multiple iterations [89] or utilize system architecture that has inherent error tolerance [90] or target limited perceiving ability of humans, etc. This can significantly relax the computation requirement; thus can be utilized for the benefits of energy efficiency, performance, area, etc [91, 39]. For the digital circuits, the approximate computing can be realized mainly by two approaches 1) voltage over scaling, 2) functional approximation. In voltage over scaling, the circuits are operated at reduced voltage resulting in timing violations of some paths but get power saving with reduced voltage [92, 93]. Functional approximation, on the other hand, refers to realizing an approximate logic function instead of exact function [91, 39, 94]. The approximate logic function, however, should vary only slightly compared to original function in terms of logic equivalence. But it still provides tremendous opportunity to reduce circuit complexity. The approximate logic function can thus be realized using fewer logic gates, simpler circuits, etc, eventually providing the benefits of reduced area, energy, etc. For emerging technologies, including CNFETs, the functional approximation technique can provide the opportunity for obtaining circuits less affected by the process. With a careful consideration of each primary output, the approximate circuit can be designed encountering fewer gates, lesser nodes along path, lesser nodes in entire circuit, reduced capacitances; consequently providing greater process imperfection tolerance compared to precise circuits.

In this chapter, we first discuss about how approximate circuits can be used to reduce process-induced degradation in CNFET circuits, followed by the methodology to generate approximate circuits for that purpose. As discussed before, Approximate circuits obtained using functional approximation technique do not need to exactly match the functionality of the original circuit. Thus approximate circuits can have the luxury of lesser gates, simpler circuits; bringing in the obvious benefits of reduced area and energy efficiency [91, 94]. Generally, approximate circuits are obtained for the purpose of energy efficiency and reduced area at a minimal logic penalty [39]. But, the approximate circuits with lesser gates, simpler circuits can be less affected by process induced degradation and thus can be effectively obtained to provide additional benefit of process imperfection tolerance for low yield technologies (suffering from process imperfections) (Figure 4.1). Thus, compared to existing methods for obtaining approximate circuits, the approximate circuits for reduced process-induced degradation would have to be obtained/designed in a different way as the main focus now is reduced process-induced degradation; not just reduction in energy efficiency and area. Parts of this chapter are published in ref. [95].

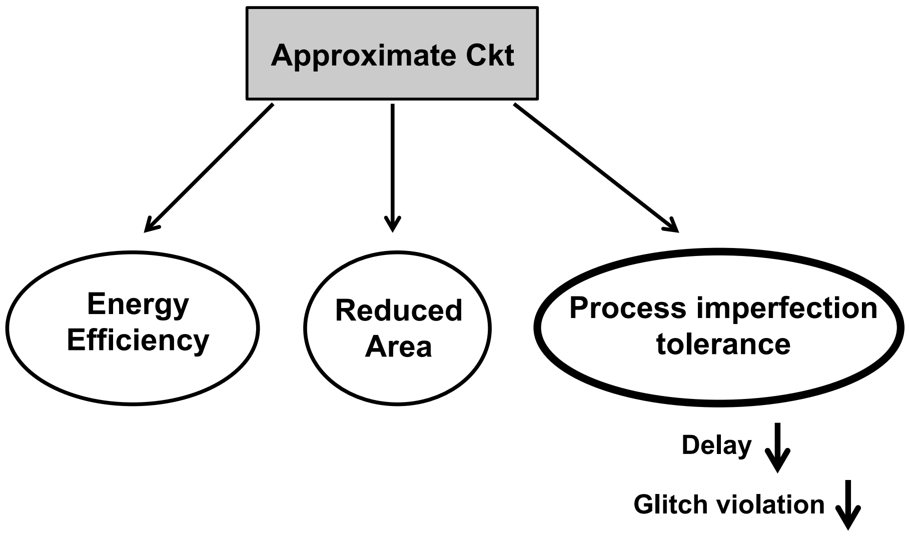


Figure 4.1: Approximate circuit helps to improve process imperfection tolerance for low yield emerging technologies, by reducing the delay and reduced glitch violations.

4.2 Reduce process induced violations with approximate circuits

In this section, we discuss in little more detail about how the approximate circuit can be utilized to reduce process-induced degradation. In Section 3.2, we discussed two aspects of circuit-level degradation due to open CNT imperfection ($N_{open} > 0$) 1) Increased delay, 2) Increased glitch violations. We thus restrict our discussion to how approximate circuits lead to reduced delay and reduced glitch violations (reduced $Mean\,Fail\,Nodes_{PATH}$, $MeanFailNodes_{CKT}$). Figure 4.2 shows an example where original circuit (Figure 4.2 (a)) consists of 6 gates and 6 $LN\#$. If the gates $g1$,$g2$, $g4$, $g5$ in the original circuit (highlighted by dotted rectangle in Figure 4.2 (a)) are being replaced by gate $a1$ to obtain approximate circuit (Figure 4.2 (b)); the $LN\#$ for the primary output ($g6$ output) with approximate circuit are being reduced by 50% (from 6 to 3). The reduction in $LN\#$ is expected to reduce the $Mean\,Fail\,Nodes_{PATH}$ for primary output and thus reducing the chance of glitch violations. Moreover, the critical path delay from primary inputs to primary output will have 2 stages instead of 4 stages (Figure 4.2 (b)); thus reducing the circuit delay. Additionally, in most of the cases, there is an additional reduction in capacitances (because of reduced connections), leading to a further reduction in delay with approximate circuits. From the perspective of process imperfection tolerance, the overall impact is reduced delay and reduced chances of glitch violation. Assuming equal probability for all inputs, the approximate circuit (Figure 4.2 (b)) has logic inaccuracy of $< 5\%$ compared to the original circuit (Figure 4.2 (a)).

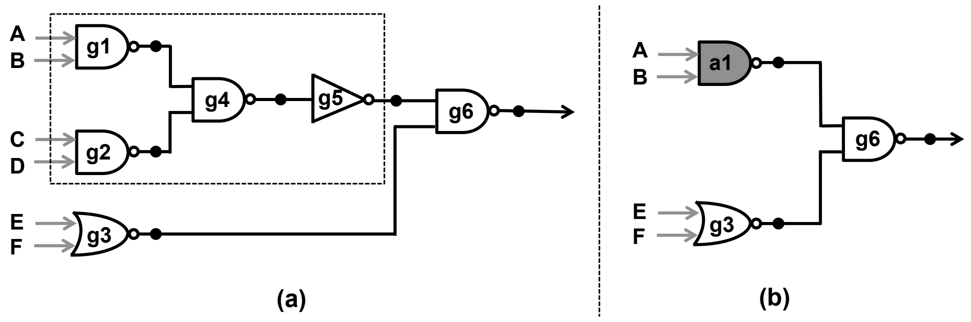


Figure 4.2: (a) Example shows a circuit to realize Boolean function $AB + CD + E + F$. (b) Approximate circuit obtained for circuit in (a).

4.3 ROBDD for obtaining approximate circuit

The logic function of a particular node in the circuit would be a function of contributing primary inputs in its fan-in cone. e.g. In Figure 4.2 (a), the logic function of node (output $g4$) is a function of A, B, C, D. Similarly, the node (output $g6$) is function of all 6 primary inputs A, B, C, D, E, F. If we take the case of node ($g6$ output),an approximate logic function for node ($g6$ output), would lead to reduced logic expression, consequently lesser number of gates; however, the approximate logic expression should be very similar in functionality to the original logic function (i.e. very less logic error). One-way of efficiently obtaining approximate logic expression are Reduced Order Binary Decision Diagram (ROBDD)s [96]. We have utilized *Cudd* package [97] to obtain ROBDD representation for the original logic function and then utilized *Cudd_SubSetShortPaths* to obtain ROBDD with fewer nodes [97]. The new ROBDD obtained would have reduced minterms and would consequently be realized with fewer gates. Figure 4.3 (a) shows ROBDD of circuit in Figure 4.2 (a) for variable ordering $E > F > A > B > C > D$. Assuming equal probability for all the inputs, the short paths (encountering few nodes) in the Binary Decision Diagram (BDD) (e.g. Removing path $E = 1$ will hurt logic accuracy by 50%) would affect the logic accuracy to a great extent; however, removing longer paths has minimal effect on the logic accuracy. *Cudd_SubSetShortPaths* operates with the principle of retaining the short paths; while removing the longer paths. Applying *Cudd_SubSetShortPaths* to ROBDD in Figure 4.3 (a), can to lead to ROBDD (Figure 4.3 (b)) with fewer nodes (nodes C, D removed). The small size ROBDD (Figure 4.3 (b)) can provide approximate logic function; which can be realized with fewer gates (Figure 4.2 (b)) in comparison to original circuit and also at mean logic error $< 5\%$.

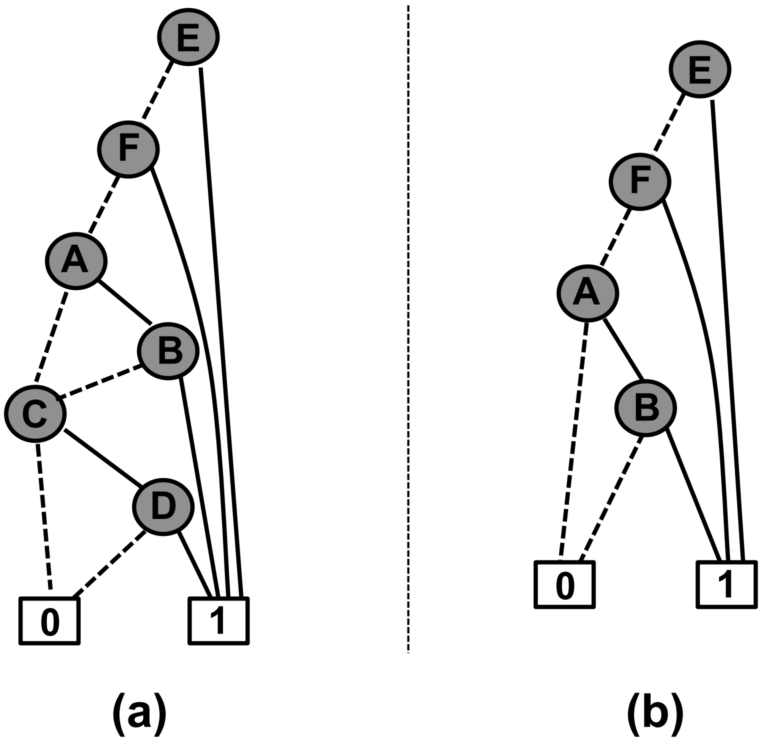


Figure 4.3: (a) ROBDD for example circuit in Figure 4.2 (a) for variable ordering $E > F > A > B > C > D$. (b) ROBDD for the circuit in Figure 4.2 for variable ordering $E > F > A > B$. ROBDD in (b) is derived from (a) using *Cudd_SubSetShortPaths* [97].

4.4 Methodology to generate approximate circuit for reduced process induced degradation

As discussed in Section 3.6.3, $Mean\,Fail\,Nodes_{PATH}$ of primary output is closely dependent on $LN\#$. If the $LN\#$ associated with critical primary output are reduced, the $Mean\,Fail\,Nodes_{PATH}$ of the critical primary output is expected to reduce, meaning lesser nodes vulnerable to glitch failure. It is usually expected that the critical primary output (one with highest $LN\#$) also has the worst Delay among the primary outputs; so the reduction in $LN\#$ can also reduce the number of stages in critical path or reduce interconnections that effectively reduce capacitances at few nodes in the critical path; improving the critical path Delay.

Approximate circuits can help to reduce the $LN\#$ and worst-case Delay but with logic error penalty. A systematic methodology is thus required to obtain an approximate circuit for an original circuit, which reduces the glitch violations and reduce critical path Delay but with a minimal logic error. The approximate circuit should be obtained in such a manner wherein overall reduction in $LN\#$ for the primary outputs is obtained such that the critical primary output for the approximate circuit should have significantly less $LN\#$ in comparison to critical primary output of the precise circuit.

Figure 4.4 shows the procedure of generating an approximate circuit for reduced process-induced degradation. The first step in the procedure is to identify the node in the circuit, which is to be approximated. e.g. Figure 4.2(a) shows the example of a circuit with single primary output ($g6$ output). If the node selected for approximation is output of $g6$, then the

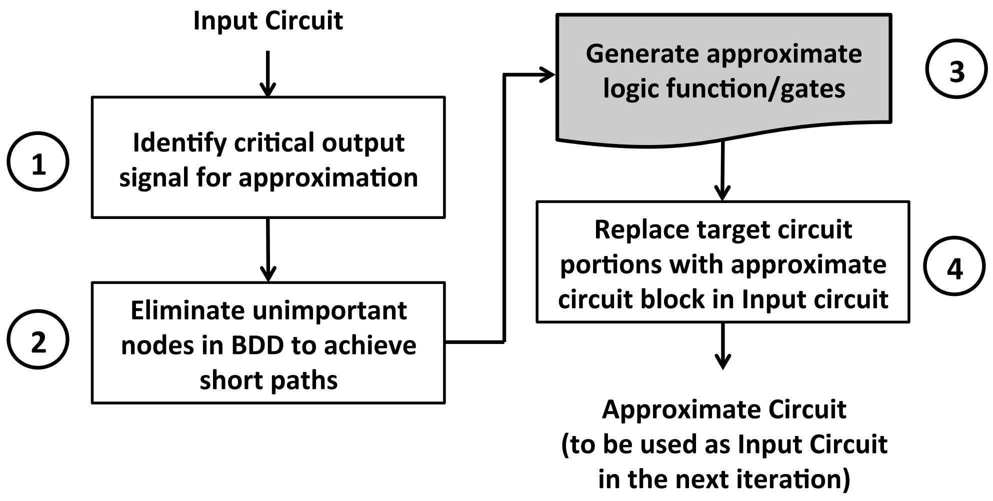

Input Circuit
1
Identify critical output signal for approximation
2
Eliminate unimportant nodes in BDD to achieve short paths
Generate approximate logic function/gates
3
Replace target circuit portions with approximate circuit block in Input circuit
4
Approximate Circuit (to be used as Input Circuit in the next iteration)

Figure 4.4: Steps to obtain approximate circuit by replacing the circuit portions, which contribute to the critical output. The approximate circuit obtained will act as input circuit for the next iteration and each of the steps $1 - 4$ are to be repeated to obtain the circuit for next iteration till final approximate circuit is obtained.

entire circuit consisting of $g1$, $g2$, $g3$, $g4$, $g5$, $g6$ enter into the approximation procedure. However, if the output of $g4$ is selected for approximation, then only $g1, g2, g4$ enter into the approximation procedure, and so on. There is more opportunity for approximation if node $g6$ ($LN\# = 6$) is selected for approximation instead of node $g4$ ($LN\# = 3$); however more approximation can lead to increased logic error.

The procedure in Figure 4.4 is explained by taking 16 Bit Han Carlson Tree adder as case study. We first explain the procedure for a relatively simpler case, where the entire circuit is considered for the approximation. In the first iteration, the whole precise circuit act as input circuit (Figure 4.4). (1) The procedure starts with first identification of the critical output (The primary output with highest $LN\#$, e.g. S_{15} in precise adder in Figure 4.5). (2) ROBDD is obtained for the critical output using $Cudd\_SubSetShortPaths$ to eliminate unimportant nodes while retaining short paths in BDD, which holds importance for logic accuracy consideration (explained in Section 4.3). (3) The ROBDD with fewer nodes will consequently provide an approximate logic function realized using fewer gates. (4) The circuit portions in the input circuit exclusively contributing to the critical output are being replaced by the approximate circuit block (from step 3) to obtain overall approximate circuit. The approximate circuit obtained now is used as input circuit in the next iteration. In the next iteration, the next critical output (e.g. S_{14} will be the critical output after circuit portions contributing to S_{15} are approximated) is selected for approximation and the steps from 2 to 4 (Figure 4.4) are followed to obtain the approximate circuit for next iteration, this procedure continues till we reach the primary outputs in the circuit having equal or less $LN\#$ in comparison to approximated primary outputs.

Next we discuss about the steps to obtain approximate circuit where only "partial circuit" is affected by approximation procedure. In this case, an intermediate node is chosen as the critical output for step 1 (Figure 4.4) along the path to the primary output, instead of the primary output itself. The circuitry following the selected intermediate node to the primary output node is not affected by the approximation (e.g. At the start, S_{15} is having highest $LN\#$ and if $G_{13:0}$ is the intermediate node selected as critical output, then the circuitry following $G_{13:0}$ to S_{15} is not affected by the approximation). Steps 2 to 4 (Figure 4.4) are being followed in a similar manner to the case of "whole circuit" discussed previously. The procedure from step 1 to 4 (Figure 4.4) is followed iteratively, with the similar terminating condition that the primary outputs having equal or less $LN\#$ in comparison to approximate primary outputs (the primary outputs whose intermediate nodes were approximated).

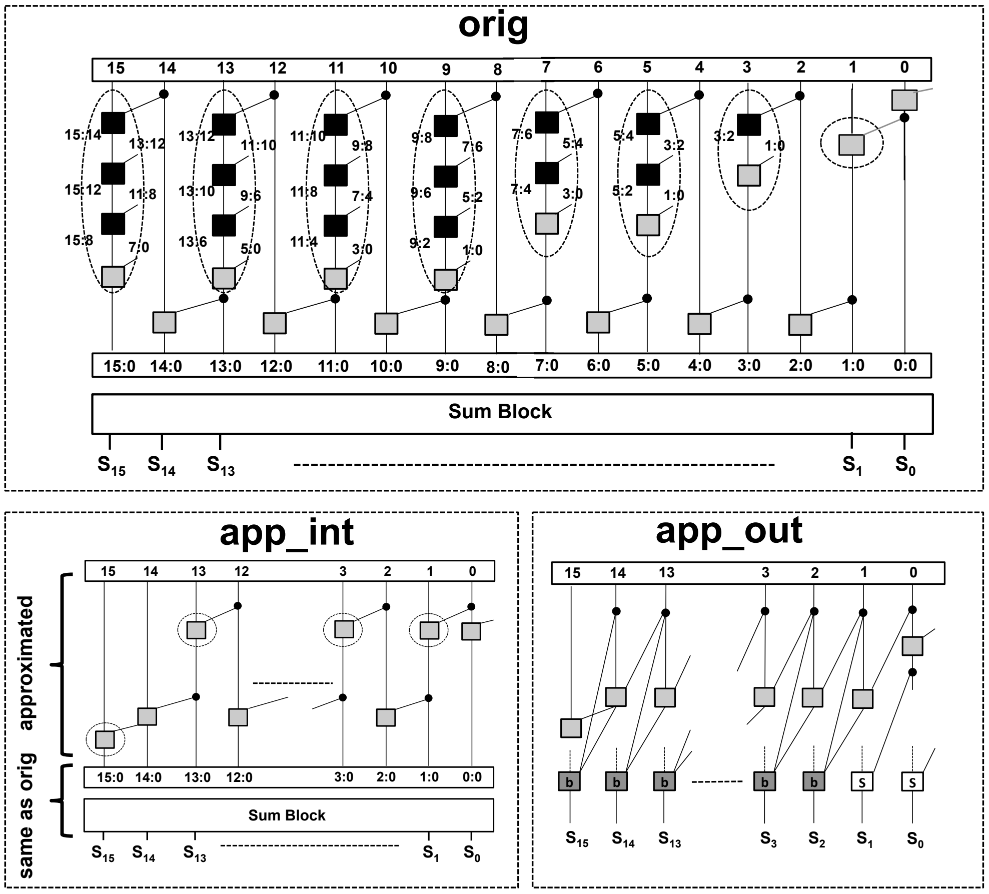
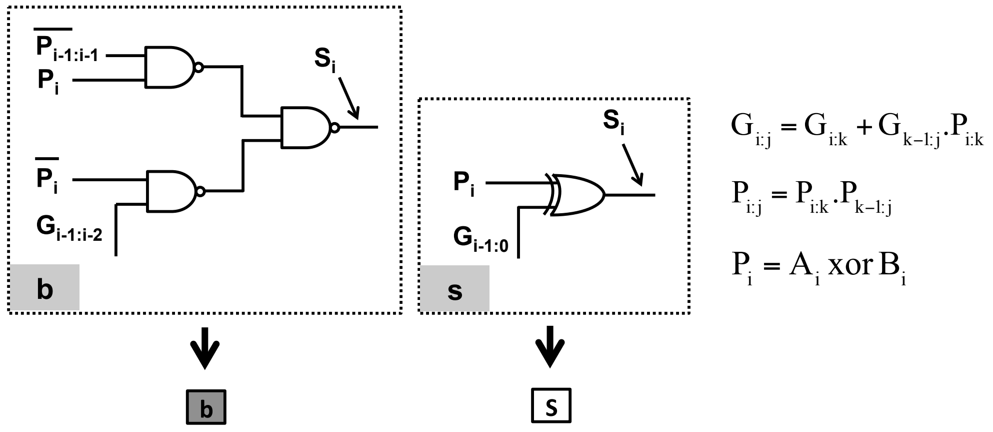


Figure 4.5: Schematics of 16 bit precise Han Carlson Tree adder (*orig*), approximate circuits with partial circuit approximated (*app_int*) and whole circuit approximated (*app_out*). The broken line on top of '*b*' and '*s*' in *app_out* (full connection not shown to avoid congestion) represents bit wise propagate signal (P_i). There is Sum block consisting of XOR gates at the output of *orig* and *app_int* for generating sum signals ($S_0, S_1, \ldots, S_{15}$). The portions highlighted by dotted circles in *orig* represents the circuit blocks which are replaced by approximate circuit block to obtain *app_int*.



$\overline{P_{i-1:i-1}}$
P_i
$\overline{P_i}$
$G_{i-1:i-2}$
S_i
b
P_i
$G_{i-1:0}$
S_i
s
$G_{i:j} = G_{i:k} + G_{k-1:j}.P_{i:k}$
$P_{i:j} = P_{i:k}.P_{k-1:j}$
$P_i = A_i \text{ xor } B_i$
b
s

Figure 4.6: Schematics of 1-bit adder modules '*b*', '*s*' utilized to generate sum outputs $(S_0, S_1, \ldots, S_{15})$ of *app_out*. $G_{i:j}$, $P_{i:j}$ and P_i refer to group generate, group propagate and bit wise propagate signal respectively.

In the case of "partial circuit", the intermediate node chosen as critical output, might affect more than 1 primary output (e.g. $G_{13:0}$ affect both S_{14} and S_{15} (Figure 4.5)). This would result in logically more accurate circuit in comparison to case of "whole circuit". However, there can be increased $LN\#$ or increased capacitances at nodes for the "partial circuit" in comparison to "whole circuit" case. But, irrespective of whether "whole" or "partial" case consideration, the approximate circuit would have reduced $LN\#$, reduced number of stages, and reduced capacitances at some nodes; consequently leading to lesser process induced degradation with approximate circuit in comparison to precise circuit.

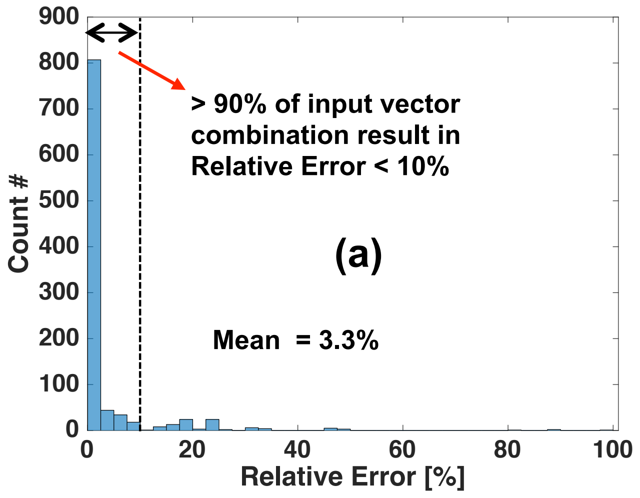
4.5 Approximate CNFET adders for reduced process induced degradation

The methodology to generate approximate circuits (discussed in Section 4.4) is applied to 16-bit precise Han Carlson Adder *orig* to construct two approximate 16-bit Adders *app_int*, *app_out* (Figure 4.5) based on whether the partial or whole precise circuit is entered into the approximation procedure. In comparison to the precise circuit, *app_out* has approximations being done for all the sum outputs ranging from S_2 to S_{15}. Each of the sum outputs from S_2 to S_{15} in *app_out* (Figure 4.5) are composed of 1 bit modules '*b*' (Figure 4.6), while S_0 and S_1 are composed of '*s*' block (similar to precise version). In comparison to *app_out*, *app_int* was obtained by having only partial circuit considered for approximation. Thus, for *app_int* the approximations are done only in the internal tree structure while the sum block providing S_0 to S_{15} outputs remains the same as precise adder (Figure 4.5). In comparison to precise adder, *app_int* has lesser number of gray/black blocks (required for generation of group generate signal). Both *app_int* and *app_out* are obtained with the purpose of reduced $LN\#$ in comparison to precise circuit for reduced process induced degradation; however at the same time it is important to compare the approximate adders in terms of logic accuracy which is critical while considering approximate circuits.

We define a term $\%RelativeError$ to represent the logic error

$$\%RelativeError = \left| \frac{S_{approx} - S_{orig}}{S_{orig}} \right| * 100 \tag{4.1}$$

where S_{orig} is the original sum value based on input vector combination and S_{approx} is



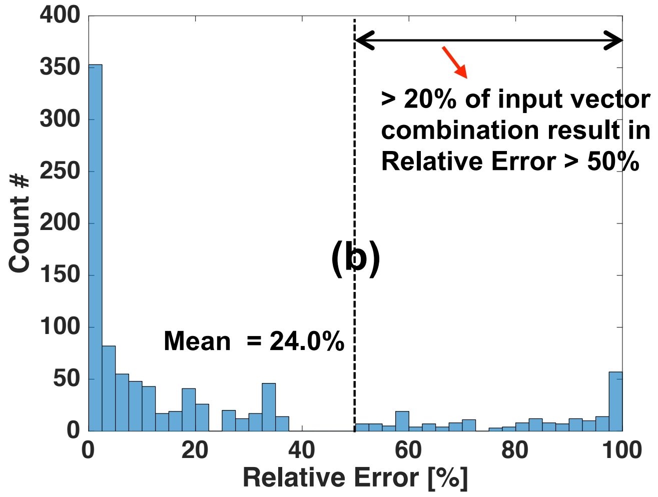


Figure 4.7: Histogram shows $\% RelativeError$ for approximate circuit (a) *app_int* and (b) *app_out*. Approximate circuit *app_int* achieves significantly low mean $RelativeError$ 3.3% with $> 90\%$ of the input vector combination having $RelativeError < 10\%$.

the sum value computed based on the outputs from the approximate adders. Figure 4.7 shows the $\%RelativeError$ for both *app_int* and *app_out* for 1000 random input vector combination. *app_out* shows high mean $\%RelativeError = 24\%$, with considerable proportion of vectors $> 20\%$ of input vector combination resulting in $\%RelativeError > 50\%$ (Figure 4.7(b)). *app_int* provides significantly better performance with mean $\%RelativeError$ only 3.3% and majority of input vectors ($> 90\%$) having $\%RelativeError < 10\%$ (Figure 4.7(a)).

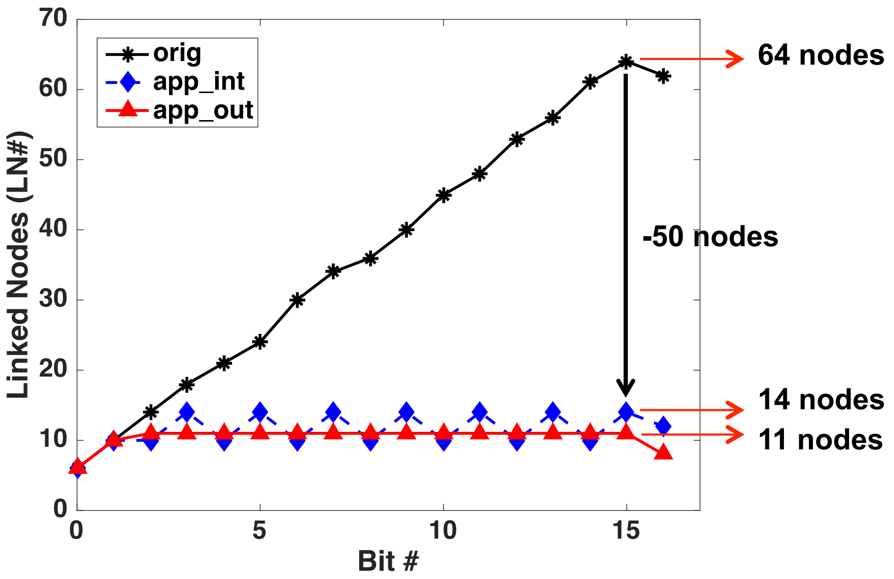


Figure 4.8: Linked Nodes ($LN\#$) plotted for each sum output S_0(LSB) to S_{15}(MSB) including C_{out}, with x-axis representing the bit position. $Bit\#(0, 1, \ldots, 15, 16)$ represent outputs (S_0, S_1,. . . ., S_{15}, C_{out}) respectively. There is significant reduction in $LN\#$ using approximate circuits (*app_int* and *app_out*). The output with highest $LN\#$ of *orig*, *app_int* and *app_out* have $LN\#$ 64, 14 and 11 respectively.

As discussed before, $Mean\,Fail\,Nodes_{PATH}$ is heavily dependent on $LN\#$. Reducing the $LN\#$ associated with the critical primary output can significantly improve the glitch suppression. Figure 4.8 shows that $LN\#$ for the precise adder (*orig*) increases monotonically for sum outputs from S_0 to S_{15} with S_{15} having the highest $LN\#$ (64 nodes). For both *app_int* and *app_out* there are significantly lesser $LN\#$. $LN\# = 14, 11$ is achieved for the critical primary output for *app_int* and *app_out* respectively. The significantly less $LN\#$ for *app_int* and *app_out* would mean a tremendous reduction in process induced degradation (both in terms of $Mean\,Fail\,Nodes_{PATH}$ and critical path Delay).

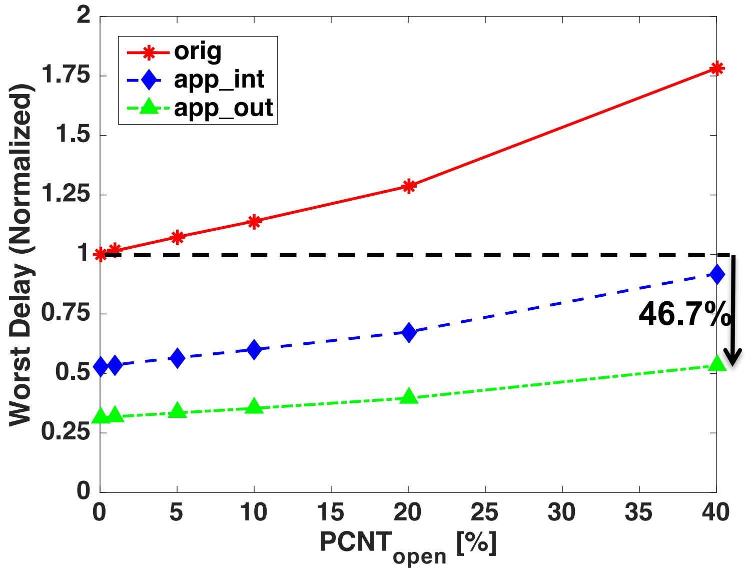


Figure 4.9: Worst Delay (normalized to that of *orig* at $PCNT_{open} = 0\%$) as a function of $PCNT_{open}$. The worst Delay for approximate circuits *app_out*, *app_int* are lower by 46.7%, 8.1% respectively in comparison to precise circuit (*orig*) at $PCNT_{open} = 0\%$.

Figure 4.9 shows the Worst Delay (among a set of 100 vectors) for the critical path in each *orig*, *app_int* and *app_out*. The Delay values are normalized to Worst Delay in *orig* at $PCNT_{open} = 0\%$. Significantly lesser Delays are observed for both *app_int* and *app_out* in comparison to precise version. This is again due the reason that significantly lesser $LN\#$ for both *app_int* and *app_out* would result in lesser number of stages in critical path compared to *orig*. Moreover, there will be reduced number of interconnections at few nodes resulting in reduced capacitances, which also reduce the critical path Delay. Even at $PCNT_{open} = 40\%$, the Delay for *app_int* and *app_out* are lower by 8.1% and 46.7% respectively in comparison to Delay of *orig* at $PCNT_{open} = 0\%$ (Figure 4.9). The result shows that with both *app_int* and *app_out*, frequency target (achieved by precise circuit *orig* at $PCNT_{open} = 0\%$) can be met even at process degradation ($PCNT_{open} = 40\%$).

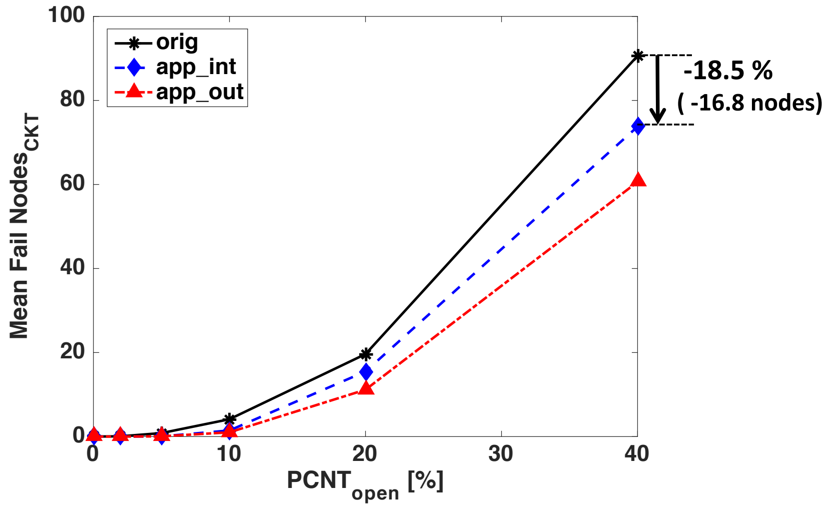


Figure 4.10: $MeanFailNodes_{CKT}$ for precise '*orig*' and approximate adders (*app_int*, *app_out*) as a function of $PCNT_{open}$. At $PCNT_{open} = 40\%$, Using approximate circuits *app_int*, *app_out* reduces the $MeanFailNodes_{CKT}$ by 18.5%, 33.1% respectively, in comparison to precise circuit (*orig*).

Figure 4.10 shows $MeanFailNodes_{CKT}$ for precise '*orig*' and approximate adders (*app_int*, *app_out*) as a function of $PCNT_{open}$. At $PCNT_{open} = 40\%$, Using approximate circuits *app_int*, *app_out* reduces the $MeanFailNodes_{CKT}$ by 18.5%, 33.1% respectively, in comparison to precise circuit (*orig*). With significantly reduced number of $LN\#$ for the critical primary output, the approximate circuits *app_int*, *app_out* yield even more significant reduction in terms of $MeanFailNodes_{PATH}$. Figure 4.11 shows the plot of $MeanFailNodes_{PATH}$ for critical primary output of precise '*orig*' and approximate adders (*app_int*, *app_out*) as a function of $PCNT_{open}$. At $PCNT_{open} = 40\%$, the $MeanFailNodes_{PATH}$ reduces significantly by 80.6% and 84.9% with *app_int* and *app_out* respectively in comparison to *orig*. The $MeanFailNodes_{PATH}$ is reduced to just 7 and 5.4 with *app_int* and *app_out* respectively in comparison to 36.1 with *orig* (Figure 4.11).

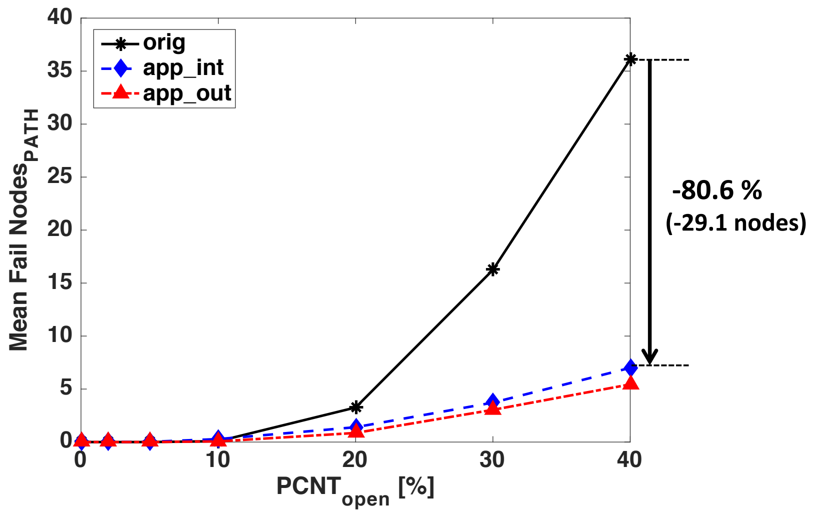


Figure 4.11: $MeanFailNodes_{PATH}$ for critical primary output of precise '*orig*' and approximate adders (*app_int*, *app_out*) as a function of $PCNT_{open}$. At $PCNT_{open} = 40\%$, the $MeanFailNodes_{PATH}$ reduces significantly by 80.6% and 84.9% with *app_int* and *app_out* respectively in comparison to *orig*.

Both delay and energy should be considered while comparing digital circuits. One important metric to hold such comparison is energy delay product (EDP). For a digital circuit, a minimum value for EDP is always desired. With the increase in $PCNT_{open}$, the Delay is expected to increase while Energy is not expected to vary significantly (With increase in $PCNT_{open}$, there would be minimal change in overall capacitances, consequently leading to nearly similar dynamic energy values even at different $PCNT_{open}$); thus EDP is expected to increase with $PCNT_{open}$. Figure 4.12 shows the EDP for *orig*, *app_int* and *app_out* at different $PCNT_{open}$ with each EDP normalized to EDP for *orig* at $PCNT_{open} = 0\%$. EDP for *orig* at $PCNT_{open} = 40\%$ is increased by 55.4% in comparison to EDP for *orig* at $PCNT_{open} = 0\%$. However, with *app_int* and *app_out* the EDP at $PCNT_{open} = 40\%$ is still lesser by 43.4% and 69.5% respectively, in comparison to EDP for *orig* at $PCNT_{open} = 0\%$, showing a significant EDP advantage with approximate circuits even at high process imperfection ($PCNT_{open} = 40\%$).

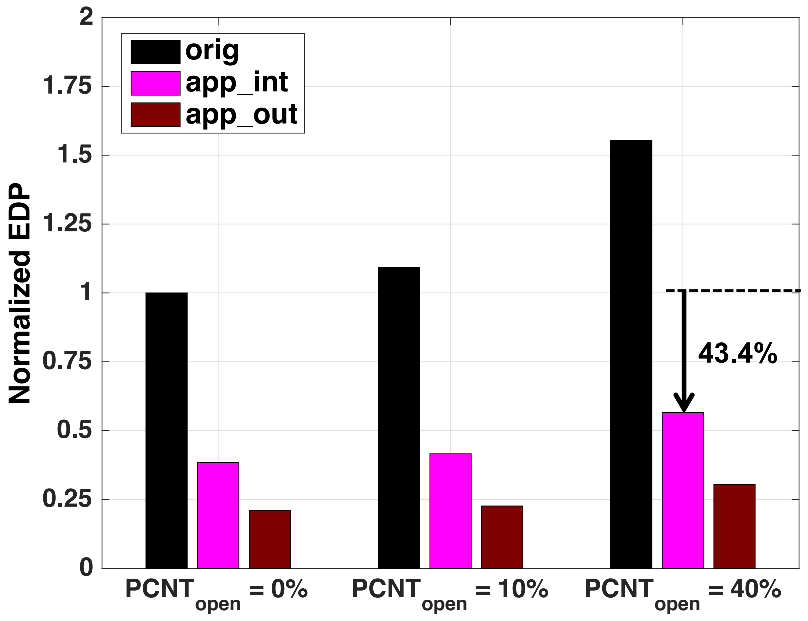


Figure 4.12: EDP (normalized to EDP of *orig* at $PCNT_{open} = 0\%$) for *orig*, *app_int* and *app_out*, at different $PCNT_{open}$. At $PCNT_{open} = 40\%$, the EDP for *app_int* and *app_out* is still lesser by 43.4% and 69.5% respectively in comparison to EDP for *orig* at $PCNT_{open} = 0\%$.

Table 4.1 provides a summary for comparison of 16 bit precise (*orig*) and approximate

CKT	$MeanRelative$ $Error$ [%]	$Normalized\,Delay$† [$PCNT_{open}$ = 40%]	MFN_{PATH}o [$PCNT_{open}$ = 40%]	EDP¶ [$PCNT_{open}$ = 40%]	$Normalized$ $Area$
orig	0%	1.78X	36.1	1.55X	1X
app_out	24.0%	0.53X	5.4	0.30X	0.78X
app_int	3.3%	0.92X	7.0	0.56X	0.78X

† Normalized to Delay of *orig* at $PCNT_{open}$ = 0%
¶ Normalized to EDP of *orig* at $PCNT_{open}$ = 0%
o MFN_{PATH} is the $MeanFailNodes_{PATH}$ of the critical primary output

Table 4.1: Comparison of precise(*orig*) and approximate(*app_int*, *app_out*) 16-bit adders.

(*app_int*, *app_out*) adders.

4.6 Conclusions

This chapter presents a systematic methodology using ROBDD [96], to obtain approximate circuit. By taking an example of 16-bit Han Carlson tree adder (one of the popular parallel prefix adders) as a reference; the 16-bit approximate adder obtained by performing approximations in internal tree structure (*app_int*) has manageable $MeanRelativeError$ of 3.3% but significant benefits in providing tolerance to process imperfection. *app_int* has less delay in comparison to the precise adder, even at high process imperfection. Moreover, there is significant reduction of nodes failing glitch criteria ($\sim$ 5$\times$ reduction) and significantly lesser EDP ($\sim$ 43.4% less EDP), even at high process imperfection.

Chapter 5

Carbon Nanotube FET — Appropriateness for Neuromorphic Computing

5.1 Introduction

In recent years, DNNs have achieved tremendous success for a wide range of applications including image classification [42, 43], speech recognition [44], etc. Advancement in hardware with the adoption of neuromorphic architectures inspired by brain [64, 65, 98] and improvement in technology, have drastically improved energy efficiency and thus made possible the implementation of systems supporting DNNs, requiring tremendous energy efficiency. TrueNorth chip by IBM demonstrated implementation of complex neural networks with real time power $< 70\,mW$ [64]. Intel Lohihi chip on $14\,nm$ [65] has demonstrated capability of achieving 100 times lower power consumption for a Deep Neural Network (DNN) implemented benchmark, compared to conventional GPU [99]. With the scalability [13] becoming increasingly difficult for silicon based systems, a possible option to keep up with the growing complexity of datasets, applications, and growing depth/size of DNNs [100, 68]; is to leverage the scaling capability and potential for tremendous energy efficiency possible with emerging technologies like CNFETs. Neural networks including DNNs, being inherently error resilient, [67, 68] can produce acceptable results even in the presence of some errors during computation. The relaxation in the requirement of precise computation is extensively used to improve the energy efficiency of hardware systems implementing DNNs [67, 68, 98, 101]. The flexibility of marginal imprecise computation in DNNs can be further utilized to reduce the unpredictable and high magnitude errors due to imperfect fabrication process for CNFETs and still leverage the tremendous energy efficiency benefit of CNFETs. With all these reasons in mind, we take CNFET based DNNs for further study in this work.

The shape of the non-linear activation function plays a vital role in the accuracy of DNNs. The imperfect process can lead to timing failures, thus distorting the shape of activation function and, consequently, degradation in classification accuracy. In this chapter, we have considered the sigmoid activation function [102] as an example for non-linear activation function. We first explore CNFET based digital neuron for sigmoid generation. We then provide simulation framework to capture the effect of process imperfection and frequency on the shape of the activation function. Towards the end of the chapter, we also utilize the activation functions to obtain classification accuracy for simple digit recognition using DBN (class of DNNs based on unsupervised learning using sigmoid activation function).

In this chapter, we have only considered the sigmoid activation function. However, the simulation framework in this chapter can easily be adapted for the generation of other activation functions, including ReLU, hyperbolic tangent, etc. At high process imperfection and increased frequency, the other activation functions would also observe distortion in the activation function shape and, consequently, degradation in classification accuracy, similar to the case of the sigmoid activation function. Parts of this chapter are published in ref. [103].

5.2 Basics of Deep Neural Network

DNN is a multilayer neural network with at least one hidden layer [104, 68]. Figure 5.1 shows an example of feed-forward DNN. The output of a neuron 'j' in the current layer (Figure 5.1) is provided in Equation 5.1.

$$y_j = f\left(\sum_i w_{ij} y_i + b_j\right) \tag{5.1}$$

Where b_j is the bias of neuron 'j', w_{ij} is the weight between neuron 'j' of current layer and neuron 'i' of previous layer, y_i is the neuron from lower layer, $f(x)$ is the non-linear activation function. Among the non-linear activation functions, sigmoid activation function [102] is quite popular and commonly used. We have considered sigmoid activation function realized using CNFET digital neuron for further study in this work. Sigmoid activation

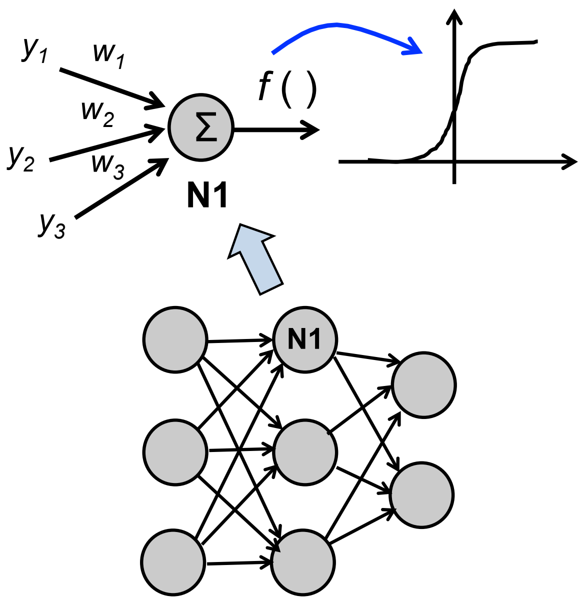


Figure 5.1: DNN with example neuron $N1$ shown. $N1$ receives inputs y_1, y_2, y_3 from the previous layer through synaptic connections w_1, w_2, w_3 respectively. f is the non-linear activation function (sigmoid considered in this work).

function is given by following equation.

$$f_x = \frac{1}{(1 + e^{-x})} \tag{5.2}$$

5.3 Sigmoid generation using digital neuron

5.3.1 Effect of open CNT imperfection on activation function

We now discuss the link between an increase in $PCNT_{open}$ to shape of the sigmoid activation function. As discussed in the previous section, the open CNT imperfection reduces the drive current of the CNFETs but parasitic capacitance remains nearly unaffected, thus in the presence of high $PCNT_{open}$, the circuit-level delays including critical path delay can increase (Figure 5.2), resulting in failure to meet given frequency target. The digital neuron can experience timing failures, and thus the shape of the activation function can deviate from the ideal sigmoid shape.

5.3.2 Digital neuron circuit

We have used crossbar architecture (Figure 5.3(b)) for neurosynaptic core, similar to TrueNorth core architecture, with axons, dendrites, and neurons represented by horizontal, vertical lines and blue boxes respectively. The connection between axon and dendrite is synapse represented by black dot ($C_{ij} = 1$). In the absence of synapse, $C_{ij} = 0$ between axon and dendrite. Figure 5.3(a) shows the circuit of our digital CNFET neuron. The neu-

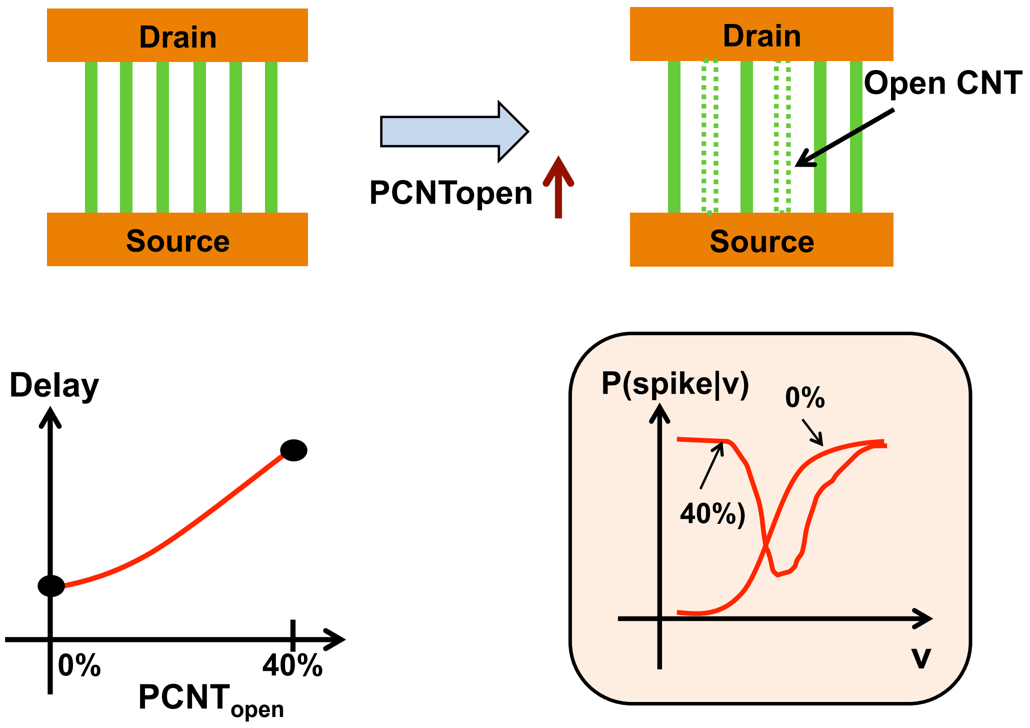


Figure 5.2: Comparison of CNFETs under no process imperfection and with high percentage of open CNTs. With increase in open CNTs [%] ($PCNT_{open}$), the effective drive current reduces, for similar parasitic capacitance, consequently resulting in higher circuit delay. The activation function '$P(spike|v)$' for neuron circuit can observe distortion in shape from ideal sigmoid, at high $PCNT_{open}$.

ron circuit comprises of 8-bit Han Carlson adder, 8-bit comparator, flops and muxes. The membrane potential of 'j^{th}' neuron, at time step 't' (without leak) is given by Equation 5.3.

$$V_j(t) = V_j(t-1) + \sum A_i * C_{ij} * w_{ij} \tag{5.3}$$

$V_j(t)$ is the membrane potential value (at the output of store flop in Figure 5.3(a)) at time step 't', to be stored in the memory. Here A_i represents the spike at i^{th} axon, $V_j(t-1)$ is the membrane potential at previous time step (also represented as $V_j(t-1)$) retrieved from memory; w_{ij} is the weight of the synapse, applied through 's_j' (Figure 5.3(a)). The neuron is event driven with signal E (Figure 5.3(a)) only activates when both A_i and C_{ij} are 1 or else 0.

Figure 5.3(c) shows the timing diagram of digital neuron circuit, explaining sequence of signals for spike generation at neuron output. V_X denotes the membrane potential (at the output of capture flop in Figure 5.3(a)) internal to the neuron circuit. The spike generation typically involves following steps:

1. **Capture** $V_j(t-1)$: The neuron captures the membrane potential from the previous time step.

2. **Synaptic addition**: The neuron integrates the contribution of active synapses at the given time step.

3. **Leak**: After synaptic addition is completed, the neuron apply the leak value to the membrane potential of the neuron.

4. **Threshold**: In this step, the membrane potential V_X is compared with the threshold value. The generation of spike and resulting $V_j(t)$ value can be summarized as follows.

```
if $V_X > V_+$ then
    $Spike = 1$
    $V_j(t) = V_{reset}$
else if $V_- \leq V_X \leq V_+$ then
    $Spike = 0$
    $V_j(t) = V_X$
else
    $Spike = 0$
    $V_j(t) = V_{reset}$
end if
```

where V_{reset} is the reset value; V_+, V_- are the positive and negative threshold values respectively. The generation of spike is represented by $Spike = 1$ in Figure 5.3(c).

We use a shorthand notation v to represent the expected V_X value (after synaptic addition is completed (Figure 5.3(c)) and without applying leak). v is given by following equation [105] .

$$v = V_{init} + \sum A_i * C_{ij} * w_{ij} \tag{5.4}$$

Here V_{init} is the membrane potential at previous time step (also represented as $V_j(t-1)$).

We use the methodology of stochastic leak and threshold developed by [106] to realize

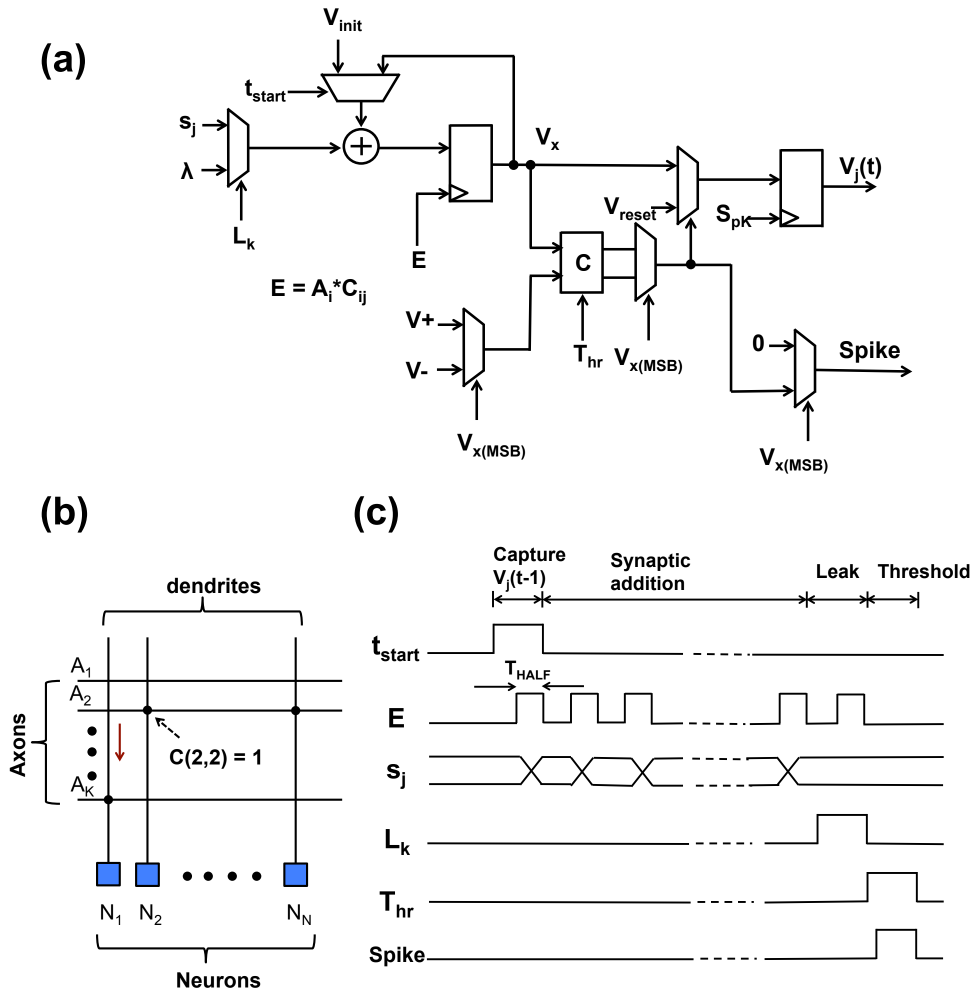


Figure 5.3: (a) CNFET based neuron circuit. (b) Crossbar architecture with axons, dendrites, neurons as horizontal, vertical lines and blue boxes respectively. Presence of dot ($C_{ij} = 1$) at axon and dendrite intersection, represent synapse. Neuron adds contribution of each synapse in serial manner (indicated by red arrow). (c) Timing diagram explaining the sequence of important signals of neuron circuit for spike generation. T_{HALF} is half of the time period between consecutive synaptic events.

sigmoid generation using digital neuron (Figure 5.3(a)). After the synaptic addition is completed, the neuron can be applied successive stochastic leak and threshold steps (Figure 5.4). We define term $P(spike|v)$ to refer to activation function as the probability of spike given v (Figure 5.4). We define another term $v_{circuit}$ as the actual V_X value in the circuit, after synaptic addition is completed. In the event of timing failure, $v_{circuit}$ can be different from v (Figure 5.4), consequently resulting in $P(spike|v)$ different from ideal sigmoid $\frac{1}{1+e^{\frac{-v}{scale}}}$ (where $scale > 1$ is used to increase the precision of v in the linear region of sigmoid [106]). $Syn\#j$ (Figure 5.4) denote the effective synaptic weight combination encountered by neuron 'j' during the synaptic addition step $Syn\#j = \sum A_i * C_{ij} * w_{ij}$.

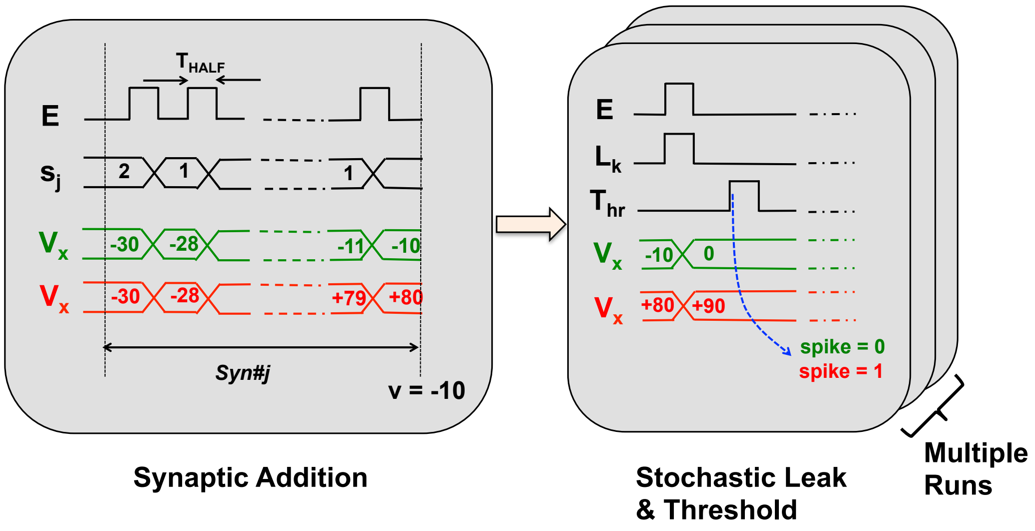
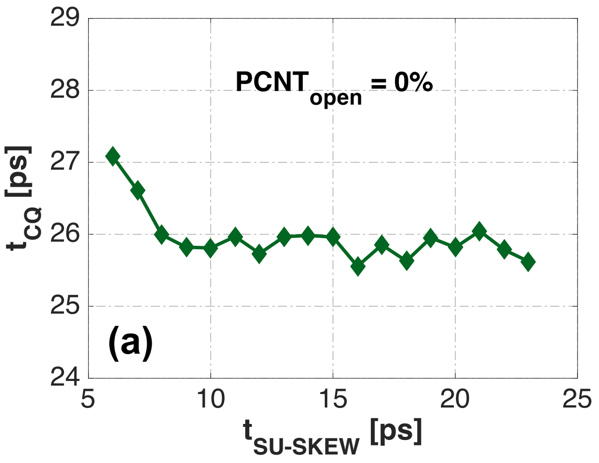


Figure 5.4: Timing diagram explaining the sequence of important signals during synaptic addition with $Syn\#j$, followed by successive stochastic leak and threshold steps over multiple runs. The case of no timing failure and timing failure encountered are represented in form of V_X with green, red color respectively. Under the event of timing failure (red color), $v_{circuit} = 80 \neq v$, generates false spike ($spike = 1$), which otherwise is not generated for normal case of no timing failure (green color).



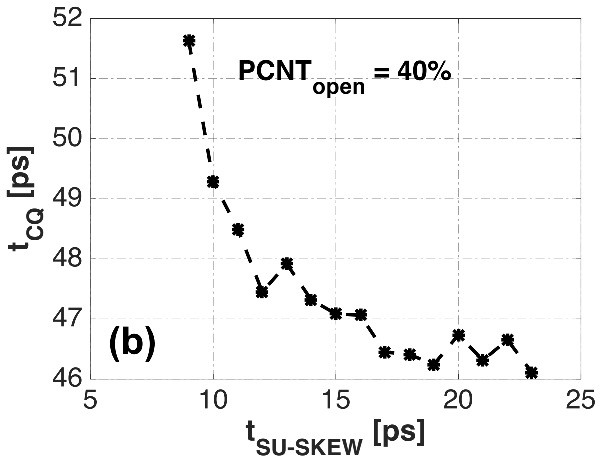


Figure 5.5: t_{CQ} as function of setup skew (termed as $t_{SU-SKEW}$) of flop at (a) $PCNT_{open}$ = 0%, (b) $PCNT_{open}$ = 40% respectively. t_{SU} is defined as $t_{SU-SKEW}$ where t_{CQ} degradation is $\sim$ 10% of the nominal value obtained for high $t_{SU-SKEW}$.

5.3.3 Factors affecting timing failure for digital neuron

The neuron adds contribution from each synapse connected in a serial manner (Figure 5.3(b), (c)). A typical neurosynaptic core contains 256 axons per neuron (means a maximum of 256 synapses) [105]. So, the majority of time neuron spends is in the synaptic addition ($\sum A_i * C_{ij} * w_{ij}$) step. Thus, time period or frequency of operation of the whole neuromorphic system is mainly determined by time period between consecutive synaptic events. We define a term T_{HALF} as half of the time period between consecutive synaptic events (Figure 5.3(c)). The circuit frequency $f = 1/(2T_{HALF})$. The choice of T_{HALF} has significant impact on shape of sigmoid activation function. T_{HALF} should satisfy the timing requirement of the critical path encountered with synaptic addition. The launch and capture of data at flip flop (following adder circuit in Figure 5.3(a)) should complete within the time period (2*T_{HALF}). The following condition for T_{HALF} should thus be satisfied to avoid timing violation.

$$T_{HALF} \geq \frac{1}{2}\left(t_{SU} + t_{CQ} + t_{Adder} + t_{comb}\right) \tag{5.5}$$

Where t_{CQ}, t_{SU} is the clock to Q, setup time respectively, of the D-flip flop for synaptic addition (Figure 5.3); t_{Adder}, t_{comb} is the delay of adder, combinational circuit (apart from adder and flop) respectively. Figure 5.5(a), (b) shows the plots of t_{CQ} as function of setup skew (termed as $t_{SU-SKEW}$) of flop at $PCNT_{open} = 0\%$, 40% respectively. Both t_{CQ} and t_{SU} increases by 80.9% and 66.7% respectively for increase in $PCNT_{open}$ from 0% to 40% (Figure 5.5(a), (b)). A similar trend is observed for adder with significant increase in worst-case delay (t_{Adder}) by 73.5% at $PCNT_{open} = 40\%$ in comparison to

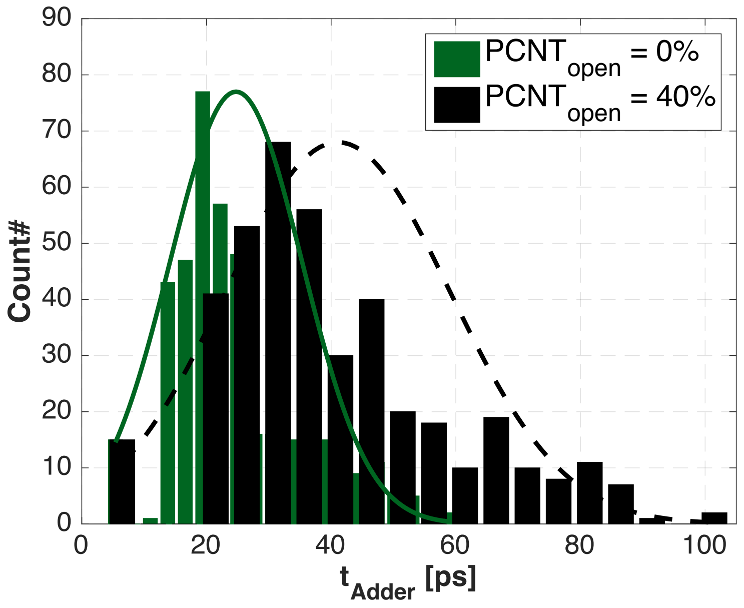


Figure 5.6: Histogram showing delay of adder (t_{Adder}) at $PCNT_{open} = 0\%$ and $PCNT_{open} = 40\%$.

$PCNT_{open} = 0\%$ (Figure 5.6). A higher T_{HALF} value is thus required to avoid timing violations at high $PCNT_{open}$. Hence, both $PCNT_{open}$ and T_{HALF} play a role in the shape of the activation function and results for activation function in this paper, are presented as a function of both $PCNT_{open}$ and T_{HALF}.

5.3.4 Simulation framework for sigmoid generation

Figure 5.7 provides the procedure of obtaining sigmoids from the simulation of actual neuron circuit, at the given T_{HALF}. $Synapse_{comb}$ in Figure 5.7 is a generic term to refer to the effective synaptic weight combination. Each synaptic weight combination ($Syn\#j$) at the given T_{HALF} considered is first used to generate signals E and s_j (Figure 5.3(a)).

Neuron netlist (at a given $PCNT_{open}$) containing circuit connectivity, along with signals (E, s_j), different V_{init} values considered are used to generate netlist files for the SPICE run. After SPICE run, multiple $v_{circuit}$ ($\#Synapse_{comb}$*$\#V_{init}$ in total) values are obtained, each corresponding to particular V_{init}, $Syn\#j$ considered. After obtaining $v_{circuit}$, the next step to follow is applying stochastic leak and threshold [106] steps, to obtain sigmoids. In order to complete simulation within feasible run time, we implement stochastic leak (λ) and threshold ($V+$) steps in software (MATLAB), to avoid the excessive circuit simulation time to implement multiple runs of stochastic leak and threshold, needed for sigmoid generation [106].

Neuron netlist

One should conduct Monte Carlo simulation in order to accurately capture the statistical effect of open CNTs (N_{open}). At a given $PCNT_{open}$, 50 Monte Carlo (MC) seeds are generated to assign a N_{open} to each transistor in the circuit following the methodology in [70]. In each MC seed, average N_{open}/N (averaging over all transistors) equals to $PCNT_{open}$. However, for each individual CNFET, N_{open}/N could be anywhere between 0 to 1, and is different from seed to seed. E.g. For a sample seed ($seed\#0$) of a circuit with only two CNFETs CN1, CN2 in Figure 5.8, N_{open}/N for CN1, CN2 is 0.33. 0.17 respectively, for an overall $PCNT_{open} = 25\%$.

Running circuit simulations for sigmoid generation for large set of seeds ($\#Seeds = 50$) of the neuron circuit would be very time consuming. A simple alternative to reduce run time is to select limited number of seeds out of the large set of seeds generated. Figure 5.8

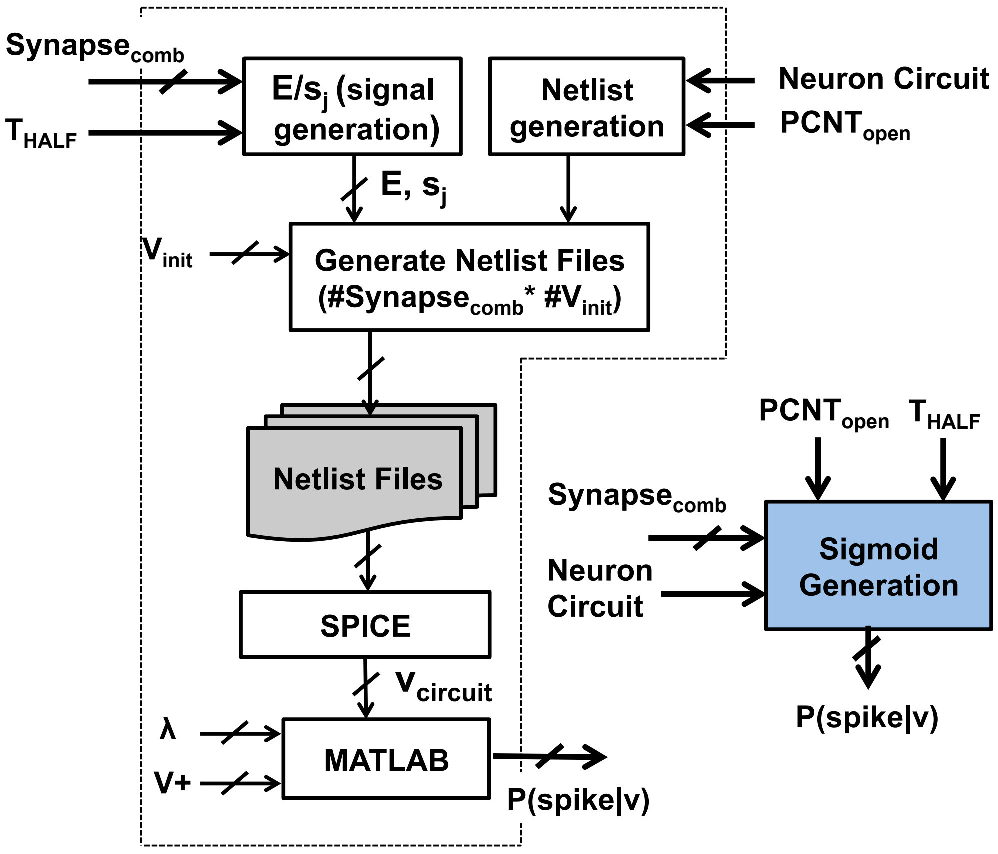


Figure 5.7: Simulation framework for sigmoid (activation function) generation using actual neuron circuit.

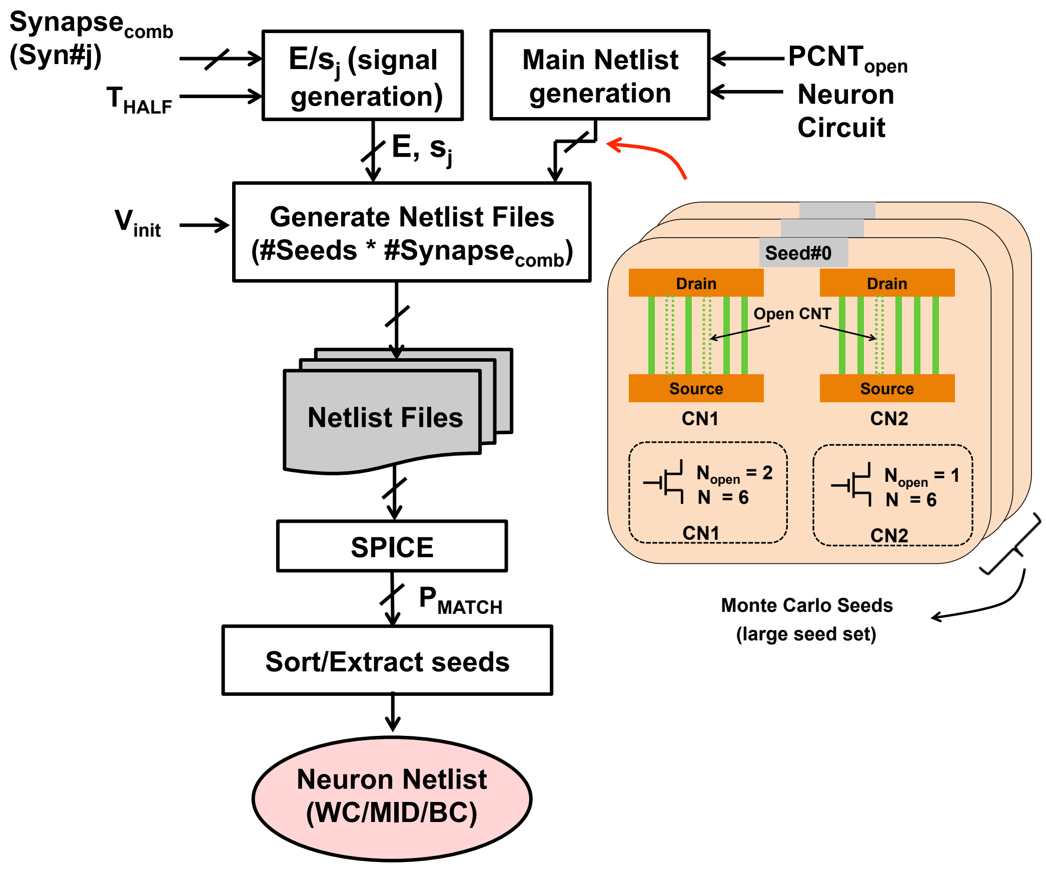


Figure 5.8: Steps to extract small number of seeds from large seed set at a given $PCNT_{open}$. The output 3 seeds (WC, BC, MID) shown correspond to neuron netlists having worst, best and median probability of observing timing failure due to CNFET process imperfection.

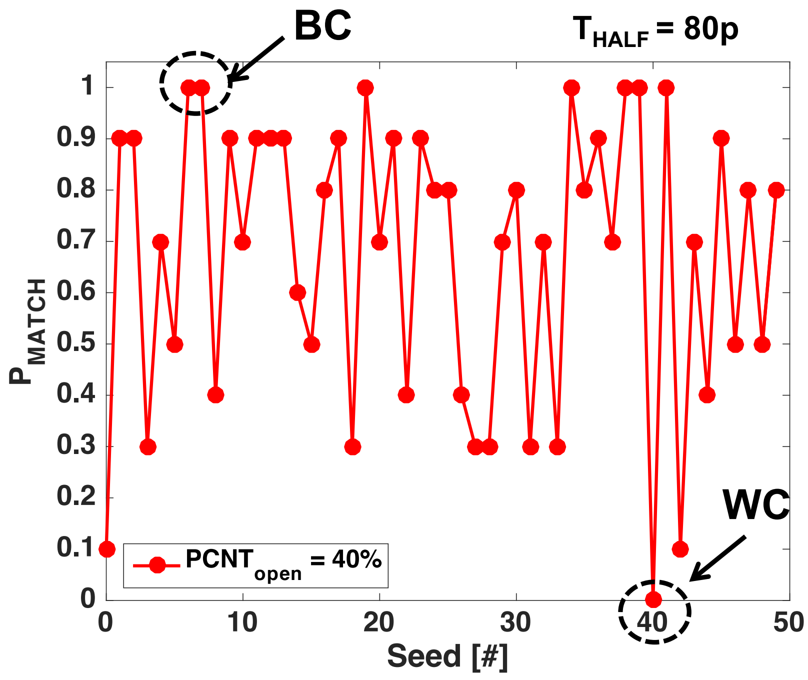
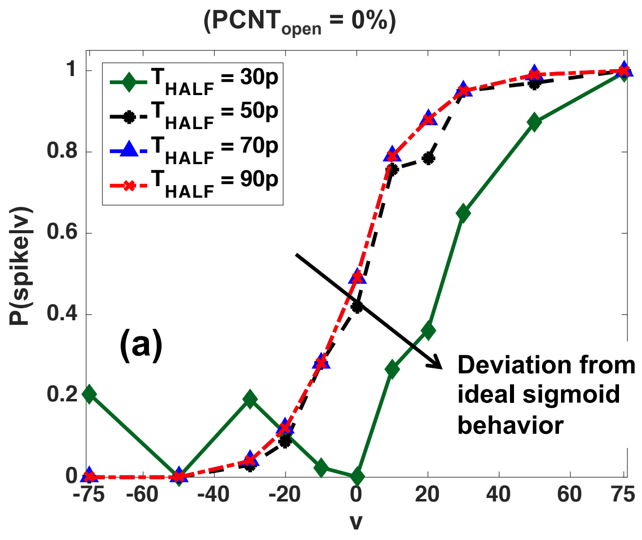


Figure 5.9: P_{MATCH} (probability of $v_{circuit} = 0$) for different seeds at $PCNT_{open} = 40\%$ ($T_{HALF} = 80\,ps$). WC, BC corresponds to the seeds with minimum and maximum P_{MATCH} value respectively. $Seed\#40$ is the WC seed with $P_{MATCH} = 0$.

provides the series of steps to extract small number of seeds from a large seed set. At a given $PCNT_{open}$, we first generate a large set of seeds ($\#Seeds = 50$) for the neuron circuit, followed by obtaining signals E, s_j in a similar manner as discussed earlier. Once we have the large set of seeds available, along with the signals (E, s_j), we can combine them to generate netlist files ($\#Seeds$*$\#Synapse_{comb}$ in total) for SPICE run. We now define another term P_{MATCH} as the probability or average number of synaptic combination ($Syn\#j$) for which $v_{circuit} = v$. Here we conduct SPICE simulations for only $v = 0$, in order to keep reasonable simulation time. We then evaluate P_{MATCH} for the condition $v_{circuit} = 0$. After obtaining P_{MATCH} (with condition $v_{circuit} = 0$) for each seed, we sort the seeds and choose the worst case (WC), best case (BC), and median (MID) seeds based on the value of P_{MATCH}. P_{MATCH} is an indicator of timing failure observed. e.g. $P_{MATCH} = 1$ would mean no timing failure observed for any $Syn\#j$ at the given conditions. Thus, BC, WC, MID seeds intended to replicate best, worst and median timing failure scenarios, should correspond to seeds with highest, lowest and median P_{MATCH} values respectively. Figure 5.9 shows the Monte Carlo simulation results of P_{MATCH} for 50 seeds at $PCNT_{open} = 40\%$, $T_{HALF} = 80\ ps$, where P_{MATCH} is the probability of ($v_{circuit} = 0$) obtained over different $Syn\#j$. For the seeds with close P_{MATCH} values, the choice of exact seed would have only a marginal impact on the actual sigmoids and consequently minimal effect on system classification accuracy.

Effect of increased process imperfection and frequency

We now study the effect of both increased process imperfection (high $PCNT_{open}$) and increased frequency (reduced T_{HALF}) on $P(spike|v)$. Instead of showing $P(spike|v)$ curves



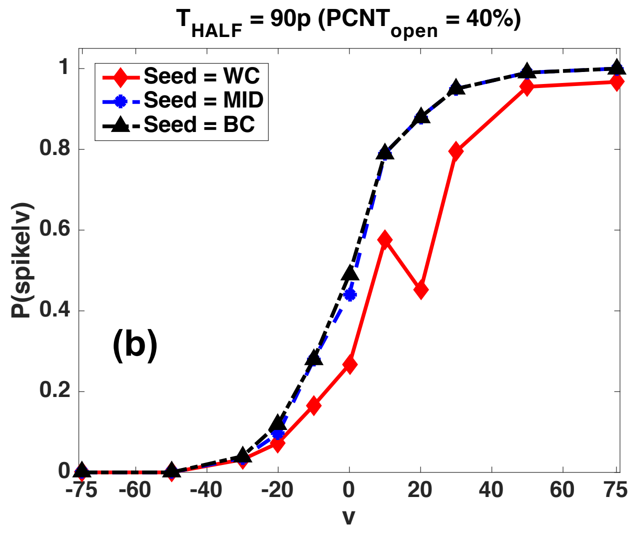


Figure 5.10: $P(spike|v)$ variation with (a) T_{HALF} at $PCNT_{open} = 0\%$, (b) Different seeds at $T_{HALF} = 90ps$, $PCNT_{open} = 40\%$.

for each $Syn\#j$ we show average $P(spike|v)$ curves over all $Syn\#j$ for rest of the paper. Figure 5.10(a), shows the effect of reduced time period resulting in timing violations on average $P(spike|v)$. At $PCNT_{open}$ = 0% (no process imperfections and thus all seeds are identical), a significant deviation from ideal sigmoid behavior is only observed for T_{HALF} = 30ps and below (Figure 5.10(a)). Figure 5.10(b) compares average $P(spike|v)$ for three different seeds (WC, BC, and MID seeds), at $PCNT_{open}$ = 40%, T_{HALF} = 90 ps. Only $P(spike|v)$ for the WC seed, deviates from the ideal sigmoid behavior, showing difference in $P(spike|v)$ curves with different seeds. For rest of the paper, the result displayed will be with MID seed unless specified.

Figure 5.11 compares average $P(spike|v)$ for $PCNT_{open}$ = 0%/10%/20%/40% at different T_{HALF}. At T_{HALF} = 90 ps, $P(spike|v)$ for each $PCNT_{open}$ match nearly with ideal sigmoid; however, at T_{HALF} = 70 ps, $P(spike|v)$ show significant deviation from ideal sigmoid for $PCNT_{open}$ = 40%, indicating timing failure at high $PCNT_{open}$. At T_{HALF} = 50 ps, significant deviation from ideal sigmoid is observed for all $PCNT_{open}$ except $PCNT_{open}$ = 0% (Figure 5.11(c)). For T_{HALF} = 40 ps, even $PCNT_{open}$ = 0% deviates appreciably from ideal sigmoid behavior (Figure 5.11(d)).

5.4 Classification accuracy methodology

We use the $P(spike|v)$ curves and test the classification accuracy of DBN for a simple MNIST dataset classification. Since the accuracy is always linked to shape of activation function $P(spike|v)$, more complicated datasets would yield similar noticeable drop in accuracy as MNIST dataset (considered in this work), once $P(spike|v)$ starts to deviate

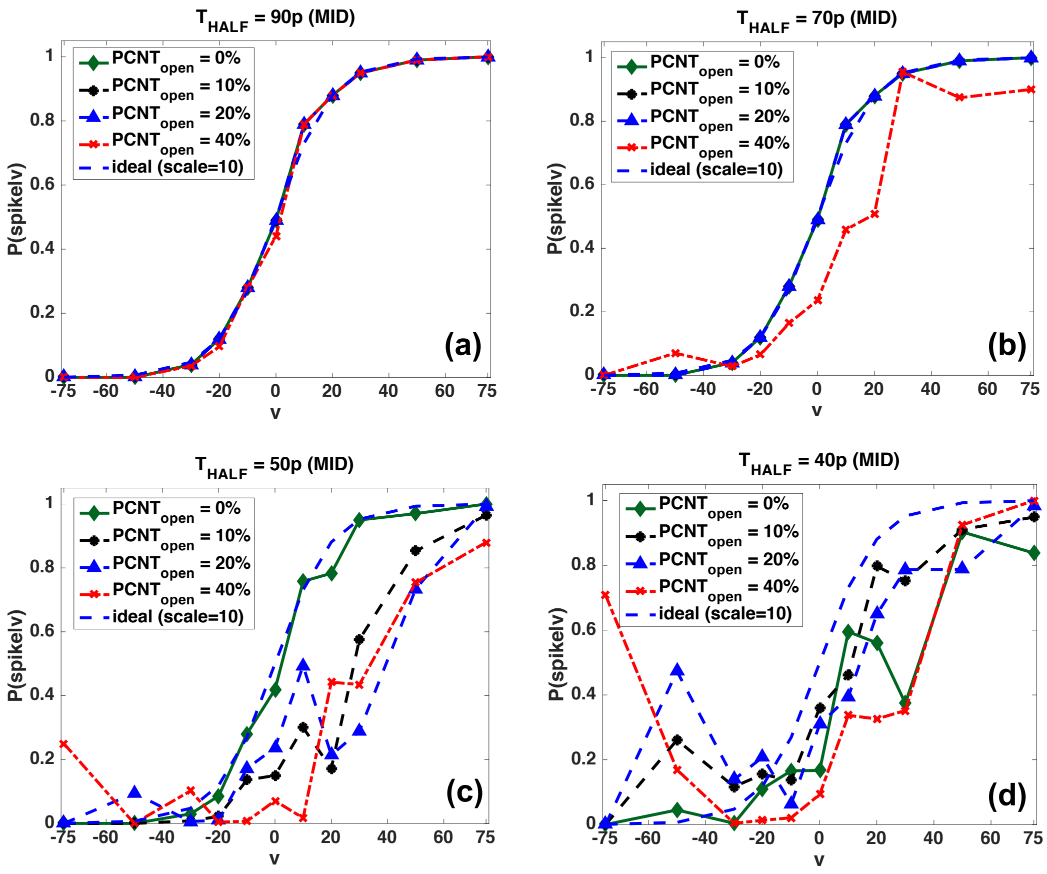
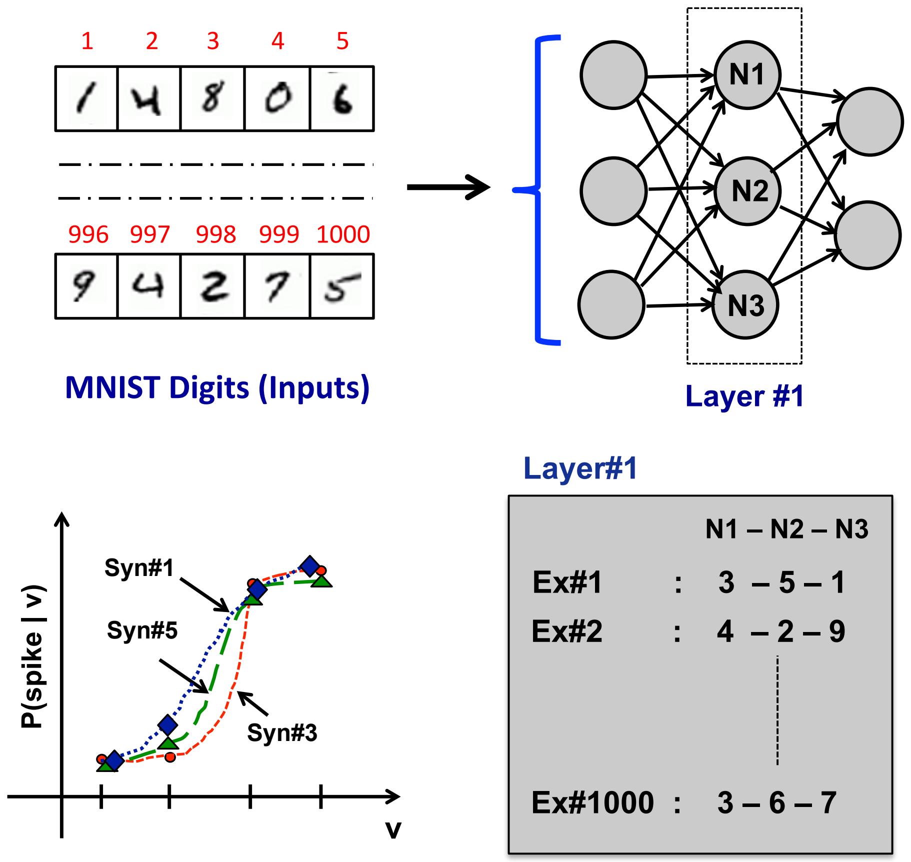


Figure 5.11: Comparison of $P(spike|v)$ at different $PCNT_{open}$, for MID seed at a) T_{HALF} = 90 ps, b) T_{HALF} = 70 ps, c) T_{HALF} = 50 ps, d) T_{HALF} = 40 ps. At T_{HALF} = 90 ps, only small deviation is observed across different $PCNT_{open}$ in comparison with ideal sigmoid ($scale = 10$). $PCNT_{open}$ = 40%, significant deviation is observed from ideal sigmoid ($scale$ = 10) at T_{HALF} = 70 ps. At T_{HALF} = 40 ps, significant deviation from ideal sigmoid is observed at each considered $PCNT_{open}$ (0%/10%/20%/40%).



	N1 – N2 – N3
Ex#1 :	3 – 5 – 1
Ex#2 :	4 – 2 – 9
Ex#1000 :	3 – 6 – 7

Figure 5.12: Schematic showing setup for tagging sigmoids, for classification accuracy of a DBN with 3 Neurons ($N1$, $N2$, $N3$) in $Layer\#1$. $N1$, $N2$, $N3$ are each being assigned $P(spike|v)$ (sigmoid) curves corresponding to a $Syn\#j$. e.g. For $Ex\#1$ of dataset: $N1$, $N2$, $N3$ are assigned sigmoids pertaining to $Syn\#3$, $Syn\#5$, $Syn\#1$ respectively.

from the ideal case. During training of the DBN, ideal sigmoid ($scale = 10$) is used to obtain the weights and bias of the considered DBN using the MATLAB code in [107]. In the testing phase, the $P(spike|v)$ curves (obtained for different $Synapse_{comb}$) are randomly assigned (with equal probability) to neurons of the DBN for each testing example. E.g. In Figure 5.12, $Layer\#1$ in DBN contains neurons $N1, N2, N3$ which are assigned $P(spike|v)$ (sigmoid) corresponding to $Syn\#3$, $Syn\#5$, $Syn\#1$ respectively for example #1 ($Ex\#1$). Similar procedure is followed for rest of the examples. Once the tagging of $P(spike|v)$ to DBN neurons is completed, the MATLAB code (adapted from [107]) utilizes the weights (w_{ij}) and bias (b_j) from training to determine classification accuracy over entire 10,000 test images of MNIST dataset.

5.5 Effect of increased process imperfection and frequency on $P(spike|v)$ and classification accuracy

In this section, we first present the results of $P(spike|v)$ obtained from circuit simulation (using methodology in Section 5.3.4) at different $PCNT_{open}$ and T_{HALF}, followed by classification accuracy (obtained using the testing methodology in Section 5.4) utilizing the $P(spike|v)$ from circuit simulation. Figure 5.13(a), (b) show the effect of increased frequency (reduced T_{HALF}) on average $P(spike|v)$ (average over all $Synapse_{comb}$ considered) at $PCNT_{open} = 0\%$, 40% respectively. In general, we see a significant deviation of $P(spike|v)$ for all listed T_{HALF} except $T_{HALF} = 90ps$ at $PCNT_{open} = 40\%$ (Figure 5.13(b)). However for $PCNT_{open} = 0\%$, appreciable deviation is observed only for

$T_{HALF} = 40\ ps$ (Figure 5.13(a)), confirming increased presence of timing violations at high $PCNT_{open}$ at given T_{HALF}, resulting in significant deviation from ideal sigmoid shape. Figure 5.13(c) shows classification accuracy (shorthand notation 'accuracy') of DBN with size 784-500-500-10. As expected, the accuracy at given T_{HALF}, $PCNT_{open}$ is driven by shape of $P(spike|v)$, with accuracy degrading to 82.4% at $T_{HALF} = 40\ ps$ for $PCNT_{open} = 0\%$ (highlighted in red circle in Figure 5.13(c)), in accordance to $P(spike|v)$ which deviates appreciably from ideal sigmoid at $T_{HALF} = 40ps$ (Figure 5.13(a)). With $PCNT_{open} = 40\%$, significant decrease in accuracy (with difference of 9.2%) in comparison to $PCNT_{open} = 0\%$ is observed at $T_{HALF} = 70ps$ and even more for lower T_{HALF} values, relating to significant deviation observed for $P(spike|v)$ at high $PCNT_{open}$ ($PCNT_{open} = 40\%$) in comparison to $PCNT_{open} = 0\%$ (Figure 5.13(a), (b)).

5.6 Conclusions

In this chapter, we investigate the effect of CNFET process imperfections on classification accuracy of DNNs. Specifically, we focus on distortion in activation function shape with increased process imperfection. The simulation framework also considers the statistical effect of open CNT imperfection during activation function generation from CNFET based digital neuron. Both reduced time period between synaptic events and high open CNT imperfection is observed to result in timing failures, thus distortion in the shape of the activation function. The framework for DNN classification accuracy implemented in MATLAB, utilizes the activation functions generated from SPICE simulations. Expectedly, the classification accuracy is linked to the shape of the activation function, with a

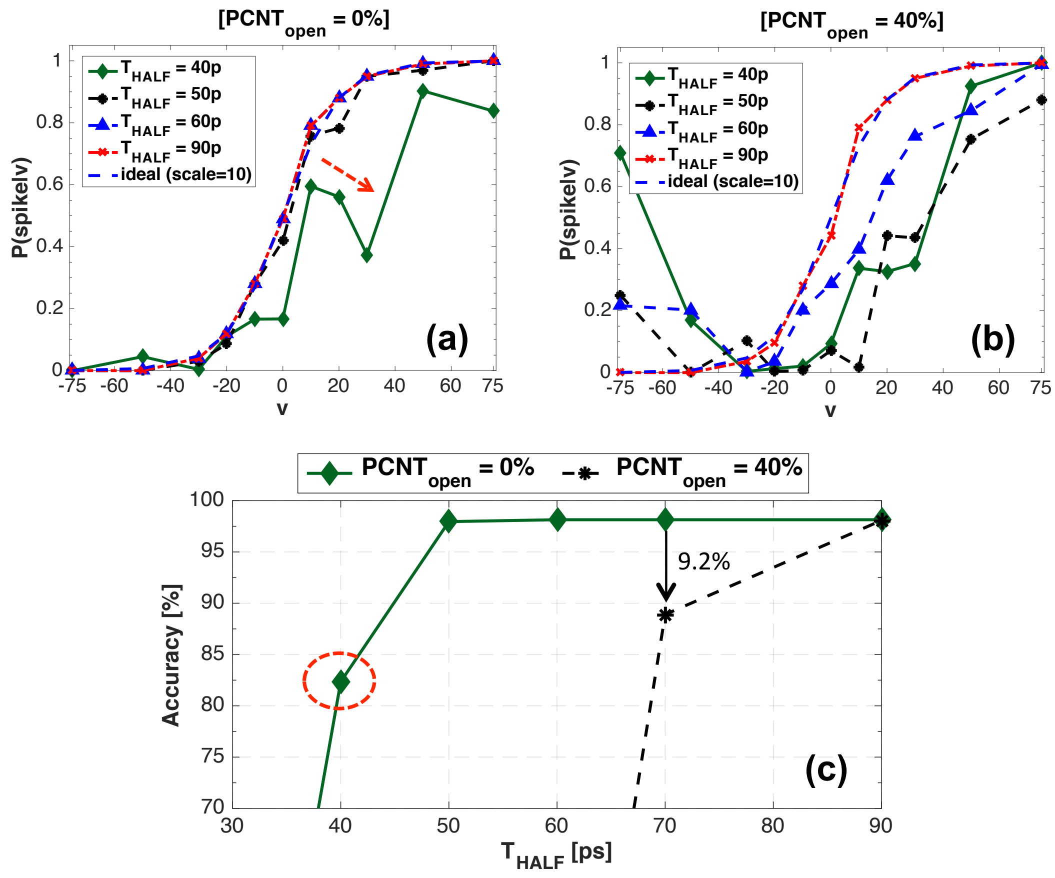


Figure 5.13: Comparison of $P(spike|v)$ at different T_{HALF} with (a) $PCNT_{open} = 0\%$, (b) $PCNT_{open} = 40\%$. (c) Comparison of accuracy (%) as function of T_{HALF} at $PCNT_{open} = 0\%$ and $PCNT_{open} = 40\%$.

noticeable drop in accuracy once the activation function deviates from the ideal case.

Chapter 6

Techniques to Mitigate Impact of CNFET Process Imperfections

6.1 Introduction

Neural networks, including DNNs, are inherently error resilient [67, 68], and can produce acceptable results even in the presence of some errors during computation. The relaxation in the requirement of precise computation is extensively used to improve the energy efficiency of hardware systems implementing DNNs. Pruning of synaptic connections is widely used to reduce the size of DNNs, computation, and improvement in energy efficiency [67, 68]. Another way is to utilize approximate circuits [108, 109] realized using the functional approximation for computation, which itself are energy efficient compared to their precision counterparts. Utilizing pruning and approximate circuit can affect the accuracy of DNNs, but the improvement in energy efficiency with pruning and approximate circuits outweighs the marginal degradation in accuracy. Moreover, the flexibility of pruning few synaptic connections and using approximate circuit components can significantly reduce the unpredictable and high accuracy degradation due to CNFET process imperfection and still leveraging from energy efficiency benefit (order of magnitude improvement compared to silicon [33]) of CNFETs.

In this chapter, we first present the modified simulation framework, with the option of including both synaptic weights after pruning and approximate neuron for sigmoid generation. We then explain in detail the way pruning can be utilized to reduce timing violations. We then propose an approximate neuron (realized using approximate adder with significantly lesser critical path delay) to further reduce timing violations. Towards the end of the chapter, we compare activation functions and classification accuracy achieved with different configurations, obtained using precise or approximate neuron and with or without synaptic weight pruning. Parts of this chapter are published in ref. [103].

6.2 Modified simulation framework for sigmoid generation

In previous chapter, we observed the effect of $PCNT_{open}$ and T_{HALF} on $P(spike|v)$. High $PCNT_{open}$ is linked with reduced drive current of CNFETs (but same parasitic capacitance), resulting in increased circuit-level delays, thus requiring high T_{HALF} to maintain the shape of $P(spike|v)$. One efficient way to mitigate the effect of high $PCNT_{open}$ is pruning of synaptic weights. Pruning can reduce the number of synaptic events, and its effective use to suppress certain synaptic events can provide option to reduce T_{HALF} (Figure 6.1(a)), even without encountering timing violation at the same high $PCNT_{open}$. However, pruning of synaptic weights contribute to inaccuracy, but the magnitude of synaptic weights is usually small [67, 68] and result in relatively small inaccuracy impact compared to the case of timing violations. The inaccuracy impact can further be compensated by adjusting the bias (b_j) to reduce the inaccuracy over a set of $Synapse_{comb}$, encountered by the neuron.

The neuron circuit (Figure 5.3(a)) can be modified by replacing the precise adder with an approximate adder. The approximate neuron circuit thus obtained can have reduced adder (t_{Adder}) delay, consequently reduced critical path delay (t_{Delay}) in Figure 6.1(b)) and hence, reducing the probability of occurrence of timing violation at given T_{HALF} and $PCNT_{open}$. The pruning and approximate neuron can be effectively utilized to reduce the timing violations even at high $PCNT_{open}$, thus maintaining the shape of $P(spike|v)$ (Figure 6.1(c)) and help to achieve or surpass frequency requirement even at high $PCNT_{open}$.

We introduce a term $Syn\#j(mod)$ to refer to the modified $Synapse_{comb}$ (with pruning and bias compensation applied, explained in Section 6.3), in comparison to $Syn\#j$. We now discuss the framework to obtain sigmoid $P(spike|v)$ curves, including the effect of pruning and approximate neuron. Figure 6.2 provides overview of steps to obtain $P(spike|v)$ curves (equal to M) at given $PCNT_{open}$ and T_{HALF}, where M refers to number of synaptic weight combination ($Synapse_{comb}$) considered. Additional steps of "Circuit Modification" and "Pruning + Bias Compensation" are included to obtain approximate neuron and modified $Synapse_{comb}$ ($Syn\#j(mod)$) as input to the "Sigmoid Generation" block. The rest of the procedure remains the same as the sigmoid generation discussed in Section 5.3.4. The

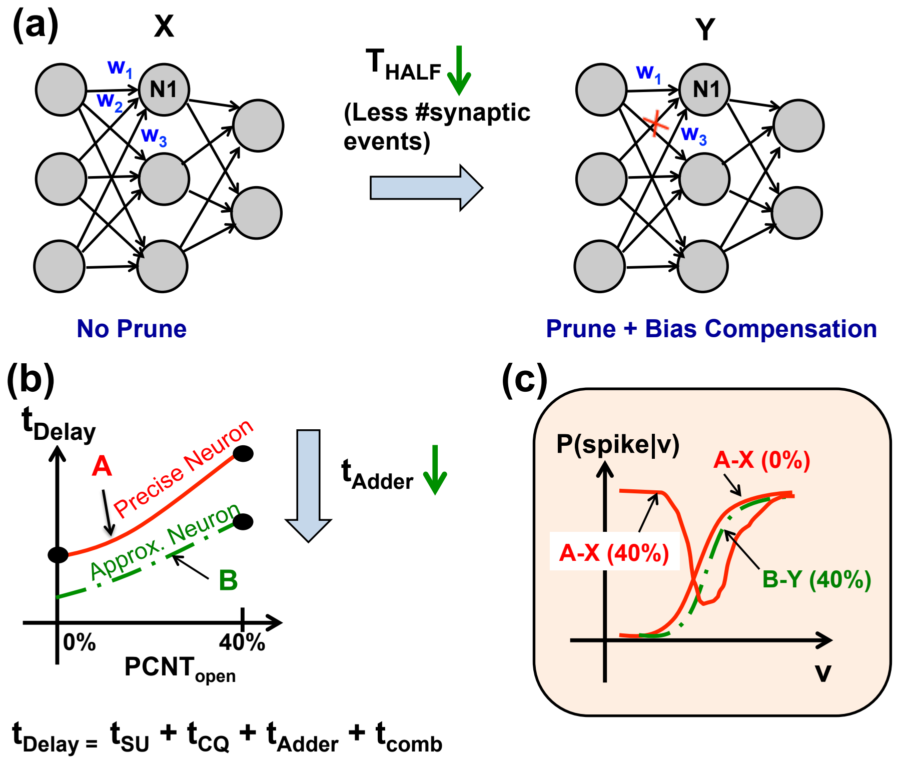


Figure 6.1: DNN with (a) no pruning (X), pruning of synaptic weights and neuron bias compensation (Y) (b) Comparison of critical path Delay versus open CNTs [%] ($PCNT_{open}$) with precise (A) and approximate (B) neuron. (c) Comparison of activation function '$P(spike|v)$' for different configurations (based on combination of precise (red) /approximate(green) neuron and no pruning/pruning of synaptic weights) at different $PCNT_{open}$.

details of “Circuit Modification” and “Pruning + Bias Compensation” are provided in later subsections.

Later in the paper, we also compare different configurations. Each configuration is denoted by generic notation $Neuron\text{-}Synapse_{comb}$, where $Neuron$ can be precise or approximate neuron circuit and $Synapse_{comb}$ can be $Syn\#j$ or $Syn\#j(mod)$ (Figure 6.2). We use shorthand notations A, B to represent precise, approximate neuron circuit respectively and X, Y to represent $Syn\#j$, $Syn\#j(mod)$ respectively (Figure 6.2). $P(spike|v)$ curves for any configuration can be obtained by having required neuron circuit and $Synapse_{comb}$ type as input to “Sigmoid Generation” block e.g. Configuration $A-Y$ will have precise neuron, $Syn\#j(mod)$ as inputs to “Sigmoid Generation”, with “Circuit Modification” being absent in comparison to sigmoid generation for configuration $B-Y$ in Figure 6.2. After choosing the neuron circuit (A or B) and the $Synapse_{comb}$ (X or Y), circuit simulation using VSCNFET model is conducted at different levels of $PCNT_{open}$, to obtain $P(spike|v)$ curves (Figure 6.2). For each $PCNT_{open}$, multiple Monte Carlo seeds are fed into the circuit to mimic the random distribution of open CNTs.

6.3 Pruning with bias compensation to reduce timing violations

We now explain the pruning method with the help of timing diagram. As explained before, the neuron is event driven with $E = A_i * C_{ij}$. In the absence of synaptic connection between axon and dendrite, $C_{ij} = 0$ implies $E = 0$. Zero skipping technique is extensively pursued

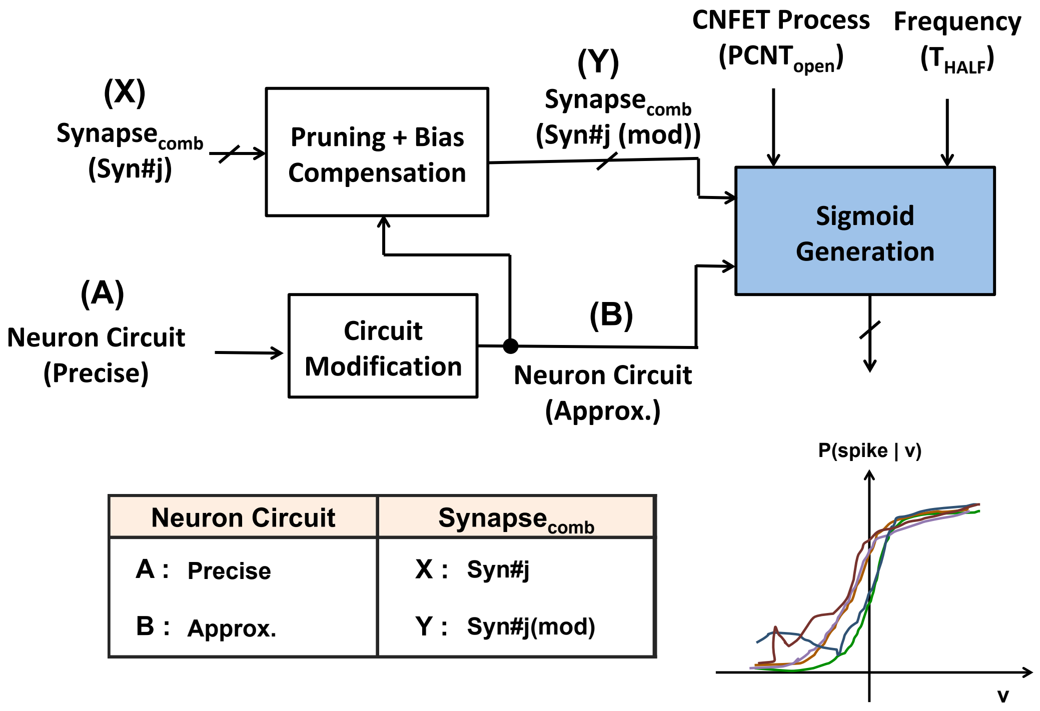


Figure 6.2: Overview of steps for $P(spike|v)$ (sigmoid) curves generation, at given CNFET process quality ($PCNT_{open}$) and particular frequency (T_{HALF}) for configuration $B - Y$.

to skip the zero weights [67, 68, 98] to reduce power. In our base case X ($Syn\#j$), we employ zero skipping by having $C_{ij} = 0$ meaning $E = 0$. Figure 6.3(a), (b) shows that when $w_{ij} = 0$, that value is not applied to s_j and s_j continue to hold the previous value. Moreover, $E = 0$ because $C_{ij} = 0$ when $w_{ij} = 0$. With pruning (Y), we avoid having events of $E = 1$ in consecutive cycles (Figure 6.3(b)). Consequently, the circuit does not need to perform any calculation in the next cycle following an event of $E = 1$, which relaxes timing and reduction in timing errors. It should be noted that s_j and E have been used as input signals for the neuron block. w_{ij} (weight stored in memory) is included in Figure 6.3(a), (b) for illustration purposes, explaining the case how w_{ij} value would affect s_j and E, when whole of neuromorphic system is implemented.

Pruning of synaptic weights however, can result in deviation of $v_{circuit}$ from v. Nevertheless, the deviation is relatively small in comparison to the case of timing violations [98, 110], which generally happens for the higher order bits. The higher order bits towards MSB are more likely to form critical path in adder of neuron circuit (Figure 5.3(a)), thus leading to more deviation of $v_{circuit}$ from v and consequently significant and unpredictable deviation of $P(spike|v)$ from ideal sigmoid. Further, deviation of $v_{circuit}$ from v (with pruning) can be reduced, by tuning the bias (b_j) associated with neuron. We define a term $Bias_{COMP}$ (Figure 6.3(b)) as additional value added to b_j of neuron to reduce the deviation. We choose optimum value of $Bias_{COMP}$ (denoted as $Bias_{COMP}(opt)$) resulting in minimum value of average root mean square error ($RMSE$) given by following equation

$$RMSE(avg) = \frac{1}{M} \sum_{k=1}^{M} RMSE(Syn\#j(mod)_k) \tag{6.1}$$

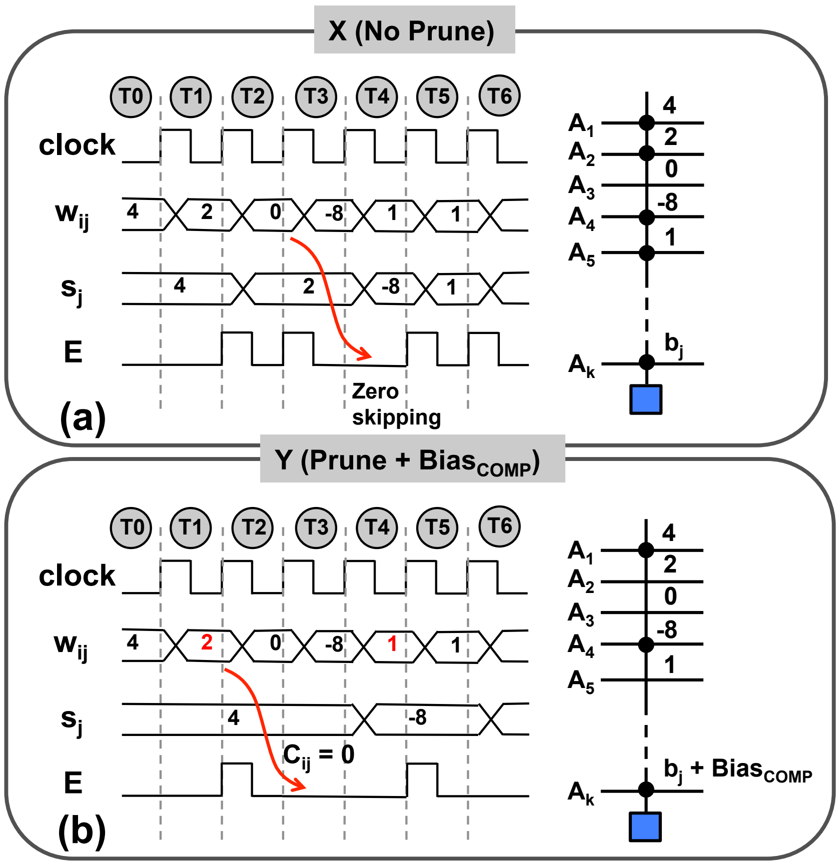
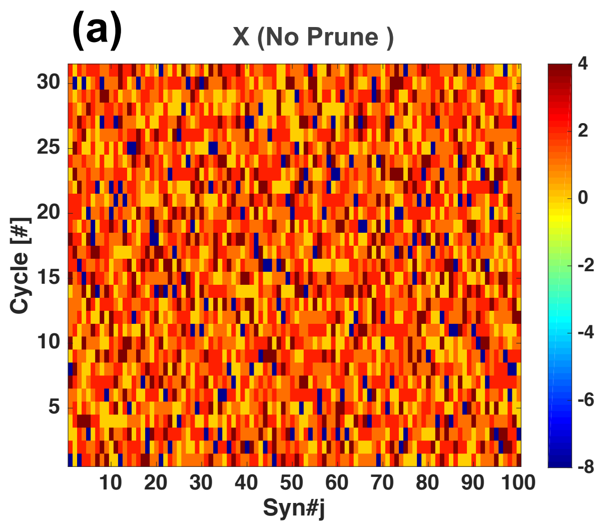


Figure 6.3: Timing Diagram for (a) X $(No\,Prune)$, (b) Y $(Prune + Bias_{COMP})$ case.

where $RMSE(Syn\#j(mod)_k)$ is the $RMSE$ of $P(spike|v)$ (generated using k^{th} $Syn\#j(mod)$) with respect to ideal sigmoid and M is the total number of $Syn\#j(mod)$ considered.

Figure 6.4(a), (b) shows multiple $Synapse_{comb}$ ($M = 100$) for X ($No\,Prune$), Y ($Prune + Bias_{COMP}$) case respectively. Figure 6.5(a) shows $RMSE(avg)$ as a function of $Bias_{COMP}$ for configuration $A - Y$ ($Precise\,Neuron–Prune + Bias_{COMP}$), for $M = 10$ and $M = 100$. For $A - Y$ ($M = 100$), we observe a minimum value of $RMSE(avg)$ =0.084, corresponding to $Bias_{COMP}(opt) = 12$ (Figure 6.5(a)). At the $Bias_{COMP}(opt)$ value, average of $P(spike|v)$ curves over all $Synapse_{comb}$, for $A - Y$ ($M = 100$) is pretty close to ideal sigmoid (inset of Figure 6.5(b)). Moreover, Figure 6.5(b) shows system accuracy which is obtained by using $P(spike|v)$ curves, explained in detail in Section 5.4 and plotted as a function of $Bias_{COMP}$. The system yield a high value of system accuracy (97.91%) at $Bias_{COMP}(opt)$ value for $A-Y$ ($M = 100$) (Figure 6.5(b)). We also obtain the results for $A - Y$ ($M = 10$), by considering first 10 $Synapse_{comb}$ of $Syn\#j(mod)$ in Figure 6.4(b). $RMSE(avg)$ for $A - Y$ ($M = 10$) (Figure 6.5(a)) follow a similar trend with $Bias_{COMP}$ as $A - Y$ ($M = 100$), yielding nearly identical minimum $RMSE(avg)$ of 0.065 (close to 0.084 for $A - Y$ ($M = 100$)), resulting in $Bias_{COMP}(opt)$ of 15 (Figure 6.5(a)). Again, average $P(spike|v)$ curves for $A - Y$ ($M = 10$) at $Bias_{COMP}(opt)$, match close to ideal sigmoid, yielding in system accuracy of 97.98% (just slightly higher than for $A - Y$ ($M = 100$)) (Figure 6.5(b)). Without loss in generality, we have considered $M = 10$ number of $Synapse_{comb}$ for each X ($No\,Prune$), Y ($Prune + Bias_{COMP}$) cases for rest of the paper, resulting in reduced compute time for circuit simulations but still efficient to mimic case with higher values for M.



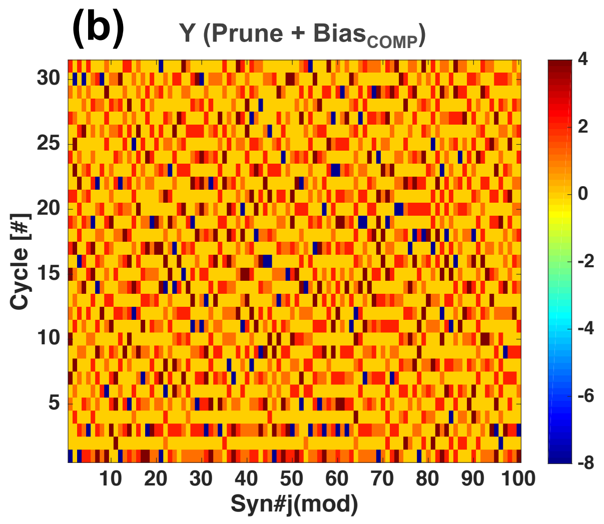
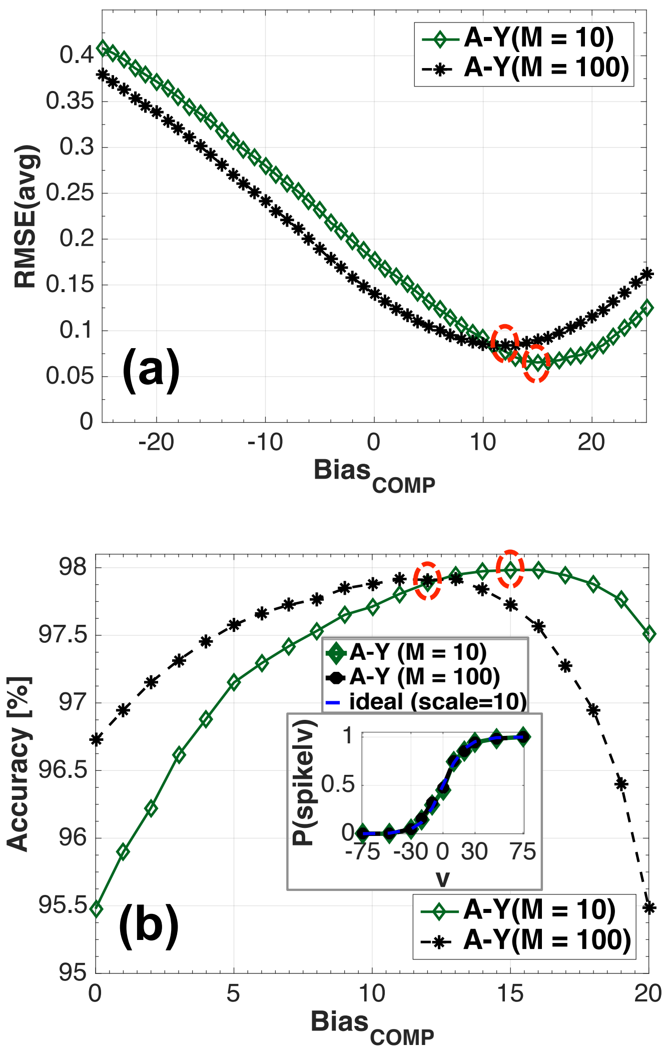
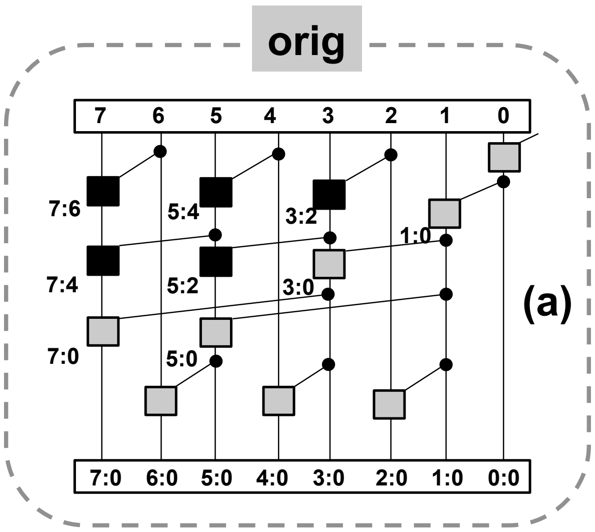


Figure 6.4: Multiple synaptic combinations ($M = 100$) for (a) X ($No\,Prune$), (b) Y ($Prune$ + $Bias_{COMP}$) case shown, with effective synaptic weight at each clock cycle [#] of synaptic addition step.



A : Precise Neuron **X** : No Prune **Y** : Prune + Bias$_{COMP}$

Figure 6.5: (a) $RMSE(avg)$ as function of $Bias_{COMP}$, (b) System accuracy as a function of $Bias_{COMP}$, for configuration $A-Y$ ($Precise\,Neuron$ – $Prune$ + $Bias_{COMP}$), for $M =$ 10, $M = 100$ $Synapse_{comb}$. Inset of (b) shows comparison of $P(spike|v)$ with corresponding $Bias_{COMP}(opt)$ values, for configuration $A-Y$ with $M = 10$, $M = 100$ $Synapse_{comb}$.



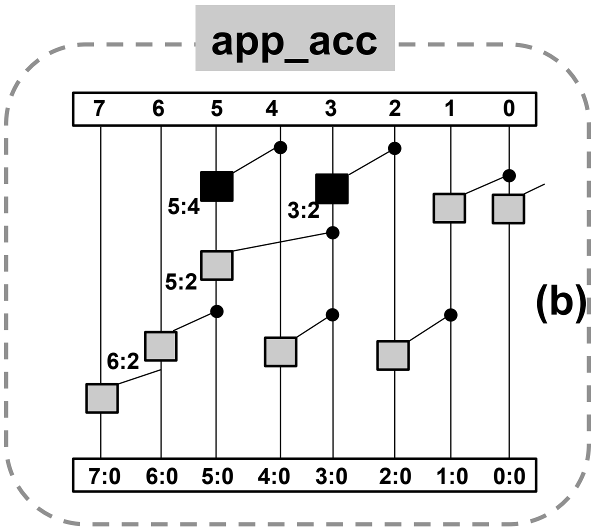
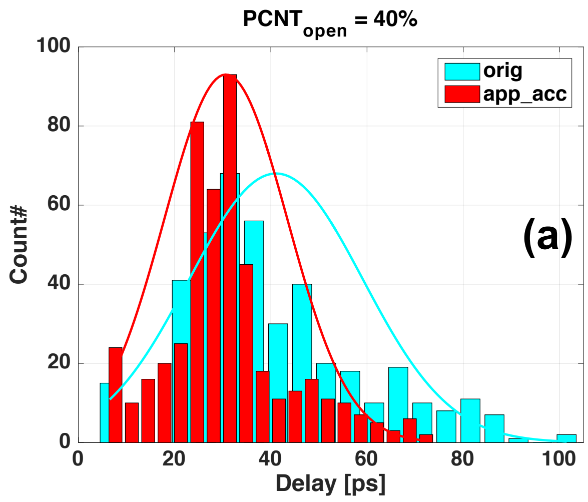


Figure 6.6: Schematics of 8-bit (a) precise Han Carlson Tree adder (*orig*), (b) approximate adder (*app_acc*).

6.4 Approximate neuron

Since the delay of adder is the most significant contributor to critical path delay limiting frequency, we generate our approximate neuron circuit by replacing precise adder with approximate adder in neuron circuit (Figure 5.3(a)). Approximate neuron obtained by using approximate adder has been used in the past for enhanced energy efficiency [111, 109]. Since our focus is towards reducing timing failures due to open CNT imperfection, we plan to obtain approximate adder with focus on speed. The approximate adder should have reduced critical path delay even compared to that in fast parallel prefix adders. For fair comparison, we have chosen 8-bit precise Han Carlson tree adder '*orig*' (Figure 6.6(a)) as adder circuit in precise neuron. We obtain 8-bit approximate adder '*app_acc*' (Figure 6.6(b)), which has fewer stages in critical path in comparison to even the fast adder topology '*orig*' chosen in this work. Delay for sum outputs ($S_0(LSB)$ to $S_7(MSB)$) for both *orig* and *app_acc* is plotted in Figure 6.7(a) for $PCNT_{open} = 40\%$. Significant improvement in critical path delay is obtained with *app_acc* having 67 *ps* (34.9% lower) worst-case delay in comparison to 103 *ps* in *orig*. However, there are some errors due to the approximate computation (denoted as $Error$, $Error = |S_{orig} - S_{app\_acc}|$, where S_{app_acc}, S_{orig} as the sum value based upon the output of *app_acc*, *orig* respectively) shown in Figure 6.7(b). To compensate the inaccuracy due to approximate adder, signal 'C_{OMP}' (Figure 6.8(c)) is generated to detect the error. Part of the neuron circuit is modified accordingly (Figure 6.8(a), (b)) to compensate for the error. The timing diagram explaining important signals for error compensation is provided in Figure 6.8(d), (e). Circuit related with C_{OMP} added some area while the approximate adder is smaller than the precision counterpart. The overall approximate neuron area (including circuitry for error compensation) is slightly lesser (0.56% below) in comparison to the original neuron area.



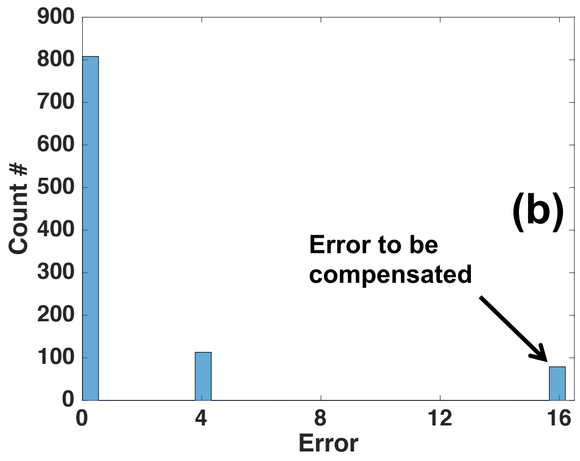


Figure 6.7: a) Delay comparison of *orig* and *app_acc* at $PCNT_{open} = 40\%$, (b) *Error* for *app_acc*.

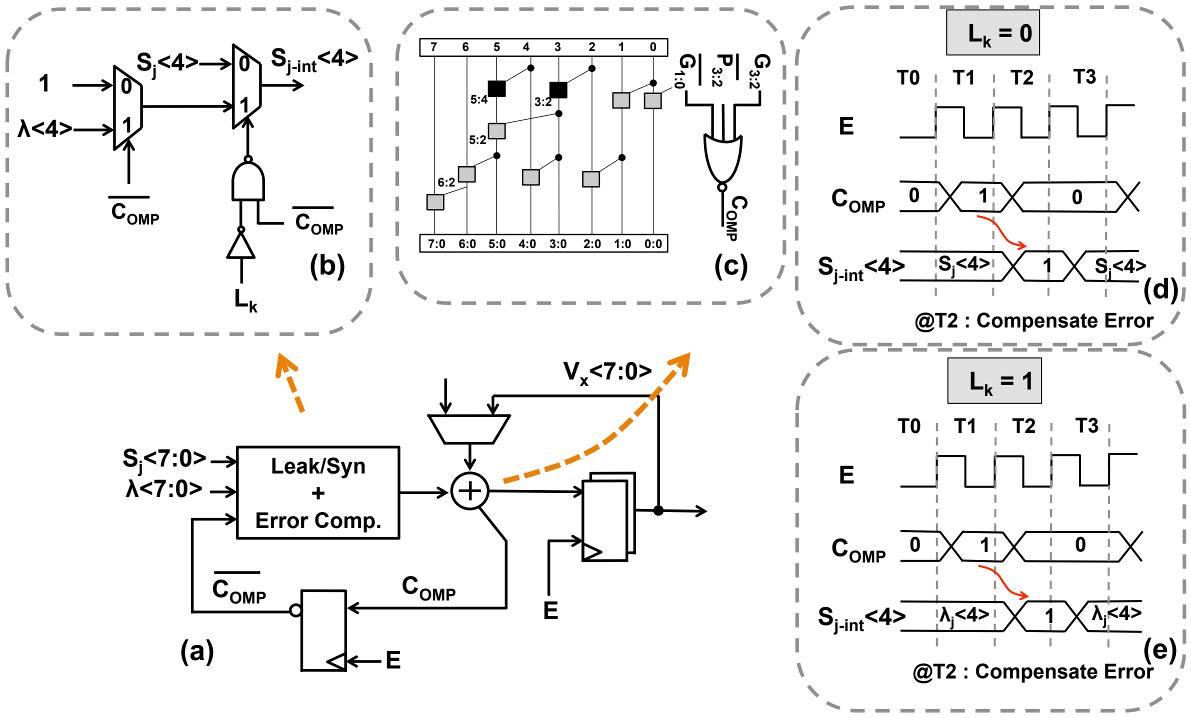


Figure 6.8: (a) Part of digital neuron circuit shown with approximate adder *app_acc* and circuitry for error compensation. (b) Modified circuit for 4th bit position of neuron circuit. (c) Schematic of *app_acc* with NOR3 gate for C_{OMP} signal. Timing diagram showing important signals for error compensation for (d) $L_k = 0$, (e) $L_k = 1$.

6.5 Results and discussion

In this section, we first compare the $P(spike|v)$ obtained (refer to section 6.2) with different configurations ($A - X$, $A - Y$, $B - Y$ introduced before) for different CNFET process ($PCNT_{open}$) and frequency (T_{HALF}) conditions. Next, we compare classification accuracy (using testing methodology in Section 5.4) with different configurations for a standard network size (784-500-500-10). The classification accuracy analysis is even extended to other smaller network sizes as well. Eventually, we provide best configuration in terms of key metrics comparison, towards end of the section.

6.5.1 Best configuration for maintaining $P(spike|v)$ shape at high $PCNT_{open}$

As discussed earlier, $P(spike|v)$ with base configuration $A - X$ deviates significantly from ideal sigmoid at high $PCNT_{open}$ ($PCNT_{open} = 40\%$), with deviation observed for all listed T_{HALF} except $T_{HALF} = 90\ ps$ (Figure 6.9(b)). With pruning and approximate neuron, the deviation is expected to happen only at lower T_{HALF} values. Even at $PCNT_{open} = 40\%$, $P(spike|v)$ with $A-Y$ ($Precise\,Neuron$, $Prune + Bias_{COMP}$) deviates from ideal sigmoid only for $T_{HALF} = 40ps$ (Figure 6.9(c)). But even better results are observed for $P(spike|v)$ with $B - Y$ ($Approximate\,Neuron$, $Prune + Bias_{COMP}$), with only negligible deviation observed at $T_{HALF} = 40ps$ (Figure 6.9(d)). $B-Y$ performs best in maintaining $P(spike|v)$ shape at high $PCNT_{open} = 40\%$, even better compared to base case $A-X$ at $PCNT_{open} = 0\%$ (no process imperfection) (Figure 6.9(a)).

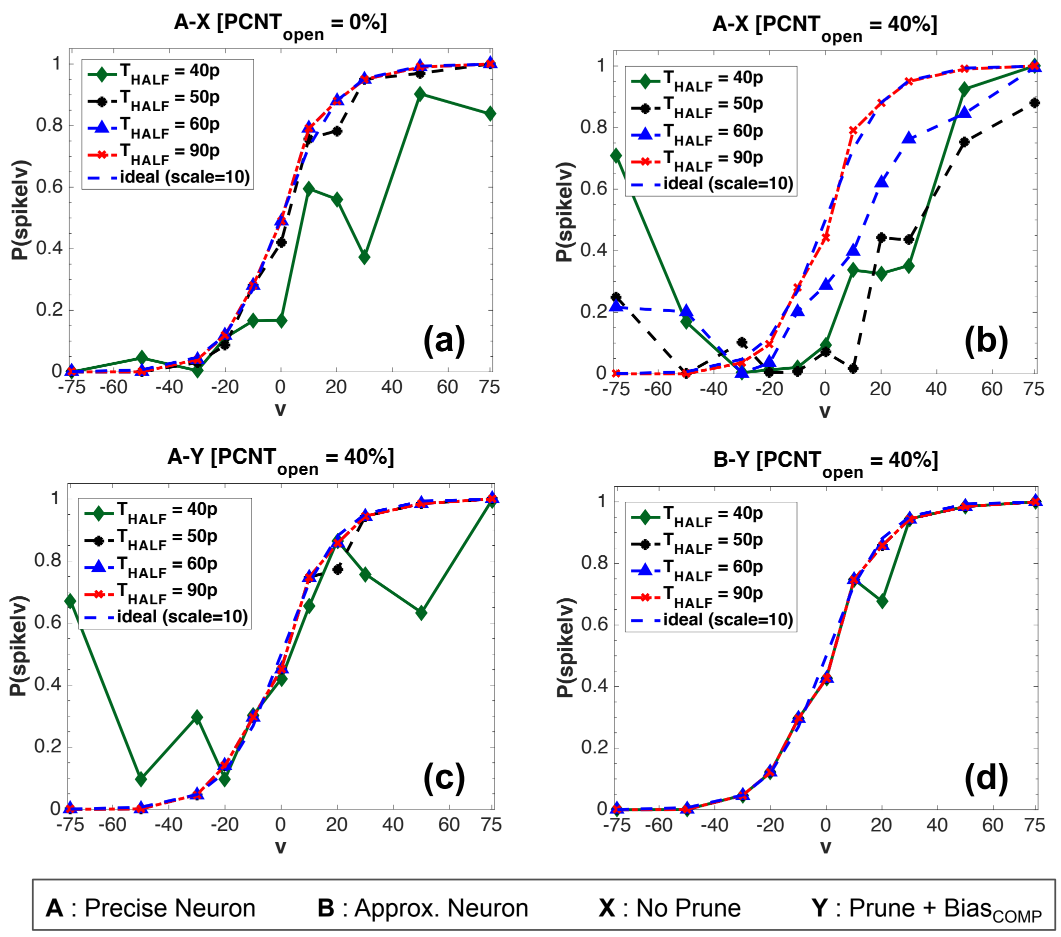


Figure 6.9: Comparison of $P(spike|v)$ at different T_{HALF} for different configuration (a) $A - X$ at $PCNT_{open} = 0\%$, (b) $A - X$ at $PCNT_{open} = 40\%$, (c) $A - Y$ at $PCNT_{open} = 40\%$, (d) $B - Y$ at $PCNT_{open} = 40\%$.

6.5.2 Classification accuracy using $P(spike|v)$ curves

In section 5.5, we observed significant degradation in accuracy (9.2% less) even for much higher T_{HALF} values ($T_{HALF} = 70ps$) for $A - X(PCNT_{open} = 40\%)$, in comparison to accuracy for base case $A - X(PCNT_{open} = 0\%)$. However with configuration $A - Y$, $B - Y$, high value of accuracy is maintained even up to low T_{HALF} values at high $PCNT_{open}$ ($PCNT_{open} = 40\%$) (Figure 6.10). At $T_{HALF} = 40ps$, accuracy with $B - Y(PCNT_{open} = 40\%)$ is even better (15.32% higher) than base case $A - X(PCNT_{open} = 0\%)$, demonstrating potential of high frequency of operation with $B - Y$ even at high $PCNT_{open}$ ($PCNT_{open} = 40\%$) (Figure 6.10). But peak accuracy with $B - Y$ is lesser, but only by 0.19% in comparison to 98.14% for base case $A - X(PCNT_{open} = 0\%)$.

Accuracy for different configurations is even compared for different network sizes. We have considered four network sizes 784-30-30-10, 784-70-70-10, 784-300-300-10 and 784-500-500-10 also represented as $NN1$, $NN2$, $NN3$, and $NN4$ respectively. For each configuration, we observe nearly similar trend as function of T_{HALF} for each of the different network sizes (Figure 6.11(a), (b), (c), (d)). However as expected, the minimum and maximum peak accuracy among the considered network sizes is observed with smallest ($NN1$) and largest ($NN4$) network sizes respectively. At $PCNT_{open} = 40\%$, $B - Y$ is still best among different configurations, in achieving high accuracy at low T_{HALF} ($T_{HALF} = 40ps$) (Figure 6.11(d)). For a given network, peak accuracy with $B - Y(PCNT_{open} = 40\%)$ is lowest, but only by small amount 0.39%, 0.19% for $NN1$, $NN4$ respectively in comparison to base case $A - X(PCNT_{open} = 0\%)$ (Table 6.1).

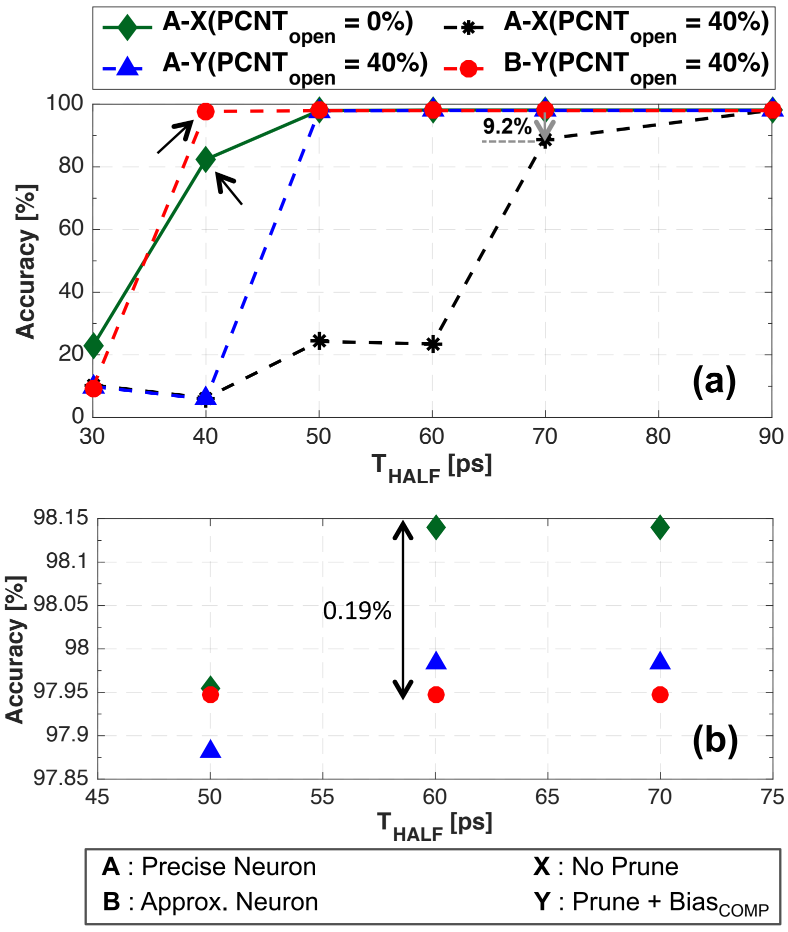


Figure 6.10: (a) Comparison of accuracy (%) as function of T_{HALF} for different configuration $A-X$ at $PCNT_{open} = 0\%$, $A-X$, $A-Y$, $B-Y$ at $PCNT_{open} = 40\%$. Accuracy at $PCNT_{open} = 0\%$ and 40% is represented by solid and dotted curves respectively. At $T_{HALF} = 40ps$, Accuracy for $B-Y$ even at $PCNT_{open} = 40\%$ is better (15.32% higher) than accuracy for $A-X$ (at $PCNT_{open} = 0\%$) (also shown by arrows). (b) Accuracy (%) as function of T_{HALF} (limited T_{HALF} range 50 ps to 70 ps) shown for different configurations $A-X$ at $PCNT_{open} = 0\%$, $A-Y$, $B-Y$ at $PCNT_{open} = 40\%$ (Only data points without lines shown for better clarity). The figure in (b) is included for purpose of showing peak accuracy.

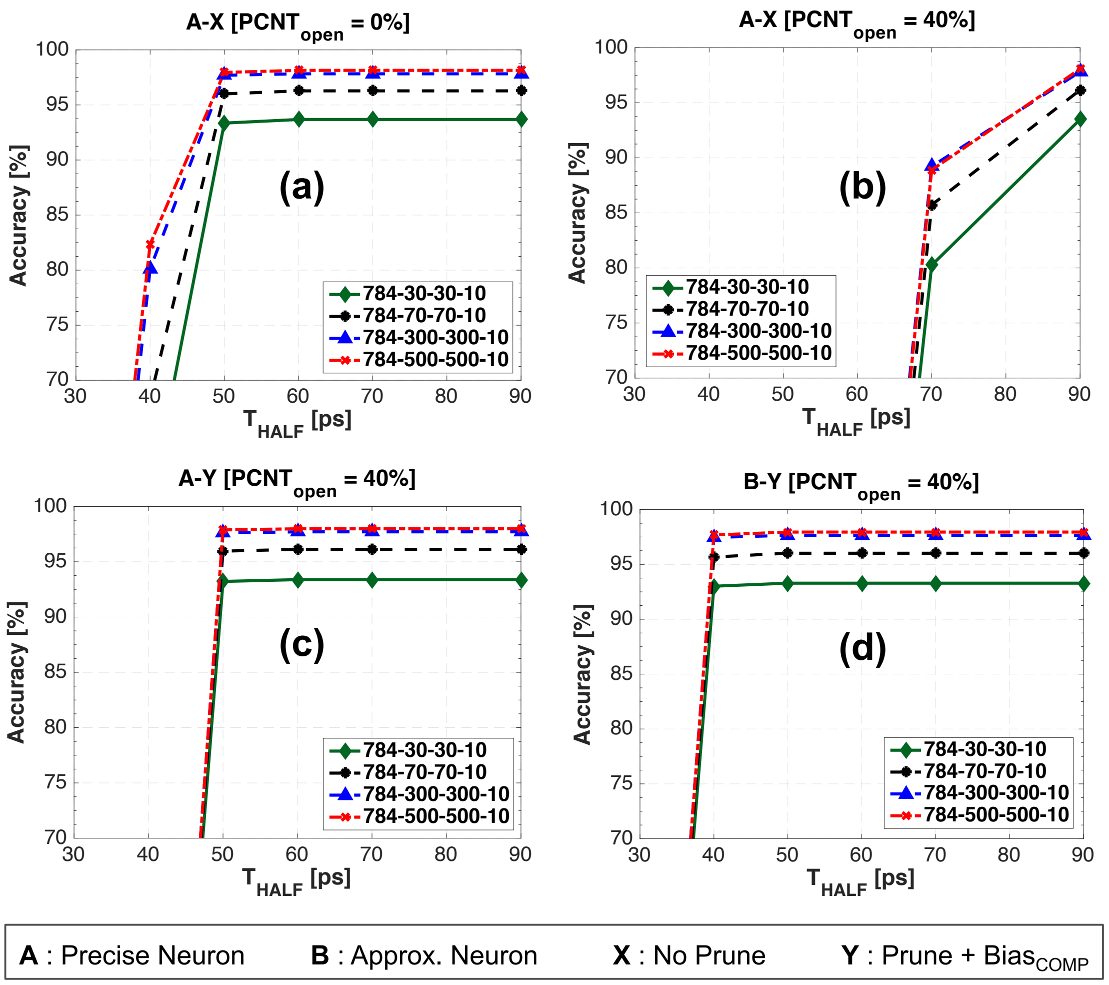


Figure 6.11: Comparison of accuracy (%) as function of T_{HALF} with different network sizes, for different configuration (a) $A-X$ at $PCNT_{open} = 0\%$, (b) $A-X$ at $PCNT_{open} = 40\%$, (c) $A-Y$ at $PCNT_{open} = 40\%$, (d) $B-Y$ at $PCNT_{open} = 40\%$.

A : Precise Neuron **B** : Approx. Neuron **X** : No Prune **Y** : Prune + Bias$_{COMP}$

Configuration (CNFET Process)	NN1	NN2	NN3	NN4
A-X (PCNT$_{open}$ = 0%)	93.68	96.28	97.83	98.14
A-Y (PCNT$_{open}$ = 40%)	93.39	96.12	97.71	97.98
B-Y (PCNT$_{open}$ = 40%)	93.29	96.02	97.65	97.95

NN1 : 784-30-30-10 **NN2** : 784-70-70-10 **NN3** : 784-300-300-10 **NN4** : 784-500-500-10

Table 6.1: Comparison of peak accuracy (%) with $A - X(PCNT_{open} = 0\%)$, $A - Y(PCNT_{open} = 40\%)$ and $B - Y(PCNT_{open} = 40\%)$ for different network sizes.

6.5.3 Key comparison for choosing best configuration

EDP is also computed to compare different configurations. Figure 6.12 shows EDP for different configurations, normalized to EDP with base case $A-X$ ($PCNT_{open} = 0\%$). For base configuration $A-X$, EDP at $PCNT_{open} = 40\%$ is increased by 69.3% in comparison to EDP at $PCNT_{open} = 0\%$. But even at $PCNT_{open} = 40\%$, EDP with configurations $A-Y$, $B-Y$ is less by 38.0% and 56.7% respectively in comparison to EDP of $A-X$ ($PCNT_{open} = 0\%$) (Figure 6.12), showing the tremendous EDP advantage even at high $PCNT_{open}$, by using pruning and approximate neuron.

Table 6.2 shows the key comparison of three configurations, with $B-Y$ even at high $PCNT_{open}$ ($PCNT_{open} = 40\%$) showing the best results for all the listed metrics except the peak accuracy, where it is only 0.19% less than peak accuracy in base case $A-X$ ($PCNT_{open} = 0\%$).

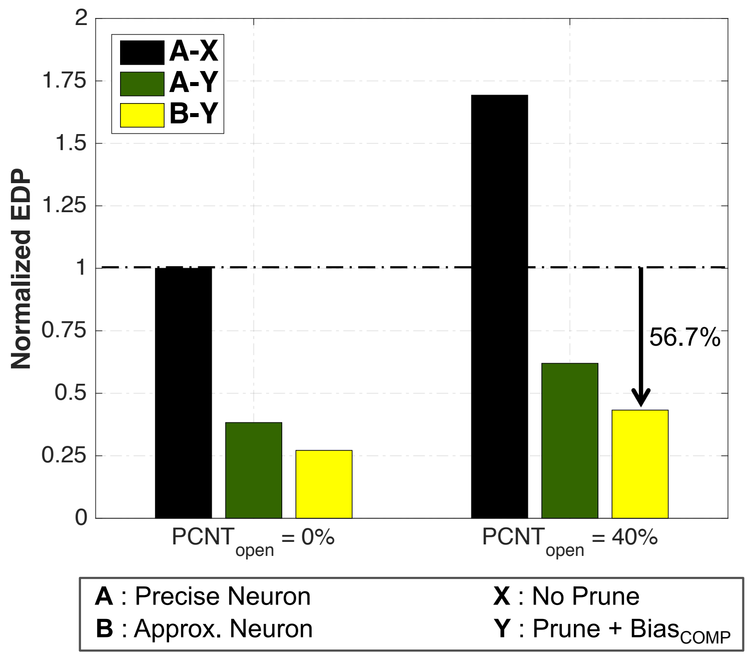


Figure 6.12: EDP normalized to base configuration $A - X$ at $PCNT_{open} = 0\%$.

A : Precise Neuron **B** : Approx. Neuron **X** : No Prune **Y** : Prune + Bias$_{COMP}$

Configuration (CNFET Process)	**Normalized EDP**	**Normalized Area**	**Min T$_{HALF}$ *(ps) (Accuracy$_{drop}$ < 0.5%)**	**Peak Accuracy (%)**
A-X (PCNT$_{open}$ = 0%)	1x	1x	50	**98.14**
A-Y (PCNT$_{open}$ = 40%)	0.62x	1x	50	97.98
B-Y (PCNT$_{open}$ = 40%)	**0.43x**	**0.99x**	**40**	97.95

$*$ Min T_{HALF} $(Accuracy_{drop} < 0.5\%)$ is the minimum T_{HALF} for which accuracy is within 0.5% of the peak accuracy (98.14%) for base configuration $A - X$.

Table 6.2: Comparison of $A - X(PCNT_{open} = 0\%)$, $A - Y(PCNT_{open} = 40\%)$ and $B - Y(PCNT_{open} = 40\%)$. Best results for each metric are highlighted in green.

6.6 Conclusions

In this chapter, we compare the activation functions for different configurations (obtained by combination of precise/approximate neuron and no pruning/pruning of synaptic weights). For a fair comparison of activation functions obtained with different configurations, we also compare accuracy of DBN for digit recognition application (taken as example) utilizing the activation functions for each considered configuration. The proposed configuration obtained by combination of approximate neuron and pruning of synaptic weights, even at high process imperfection ($PCNT_{open} = 40\%$), achieves several advantages compared with the base case with perfect process, including (1) excellent system accuracy at a higher speed (only $< 0.5\%$ accuracy drop with 25% improvement in speed) (2) significant EDP advantage (56.7% less), and (3) marginally smaller area (0.56% less).

Chapter 7

Conclusions and Future Work

This dissertation aims to investigate the possible adoption of CNFET based circuits suffering from process imperfections, for error resilient computing systems. This chapter summarizes the key contributions towards that target and also provides future research goals.

7.1 Conclusion and summary

7.1.1 Methodologies for effective capture of CNFET process imperfection on circuit-level performance

The dissertation initially discusses the major source of imperfection arising from the currently popular process, affecting the circuit-level performance of CNFET based circuits.

With $> 30\%$ of missing CNTs (in trenches connecting source and drain of CNFET) for currently reported trench widths and further scaling of trench widths expected in future, the open CNT imperfection is currently and probably will remain a major source of imperfection affecting CNFET performance. In Chapter 3, we showed the use of modified version of VSCNFET model to capture open CNT imperfection. A Monte Carlo simulation framework is provided to accurately capture the statistical effect of open CNT imperfection on circuit-level performance. In Chapter 3, we also present a link between noise tolerance and circuit topology in terms of $LN\#$ associated with each primary output. Generally, high $LN\#$ associated with a primary output is linked to more nodes failing glitch criteria, along paths to the primary output.

7.1.2 CNFET based circuits for approximate computing

In Chapter 4, we investigate the appropriateness of CNFET based circuits for approximate computing by taking example of 16-bit Han Carlson adder [85] (high speed parallel prefix adders). We present a systematic methodology using ROBDD to obtain an approximate adder with a reduced $LN\#$, consequently less impact of the imperfect process on circuit-level performance. Approximate adder obtained using the methodology has less delay, significant reduction in nodes failing glitch criteria ($\sim 5\times$ reduction) and significantly less EDP ($\sim 43.4\%$ less EDP) even at high process imperfection, with $MeanRelativeError$ of 3.3%.

7.1.3 CNFET based circuits for neuromorphic computing

In Chapter 5, we provide simulation framework to capture the effect of process imperfection and frequency on shape of activation function generated using digital CNFET neuron. Timing failures arising due to increased open CNT imperfection and frequency is shown to distort the activation function shape, and consequently, significant degradation in classification accuracy is observed for DBN, using the activation functions obtained from SPICE simulations. Neural networks, including DBNs, are inherently error resilient and several works in literature have demonstrated pruning of synaptic weights to reduce the size of DNNs for the reduction in consumed energy [112, 68]. But pruning of specific synaptic weights can also significantly reduce the probability of timing failure with slight expected degradation in accuracy. Also, approximate circuits can be used in place of precise circuits to have reduced stages, reduced capacitance at nodes, and consequently reduced critical path delay, which further reduces the probability of timing failure. In Chapter 6, we propose an approximate neuron circuit, which combined with the pruning of synaptic weights, is demonstrated to maintain the shape of activation function even at high process imperfection and higher frequency. For comparison, the activation functions are obtained for different configurations (obtained by the combination of precise/approximate neuron and no pruning/pruning of synaptic weights). By using both approximate neuron and pruning of synaptic weights, we achieve excellent system accuracy (only $< 0.5\%$ accuracy drop) with 25% improvement in speed, significant EDP advantage (56.7% less) even at high process imperfection, in comparison to base configuration of precise neuron and no pruning with ideal process, at no area penalty.

In conclusion, this dissertation provides directions for potential applicability of CNFET based technology for error resilient computing systems. For this purpose, we present methodologies, which provide an assessment of the circuit-level performance of CNFET based circuits, considering process imperfections. We accomplish DBN framework for digit recognition, considering activation functions from SPICE simulations incorporating process imperfections. We demonstrate the effectiveness of approximate neuron and synaptic weight pruning to mitigate the impact of high process imperfection on system accuracy.

7.2 Future research directions

This dissertation provides circuit and system solutions to address major source of process imperfection arising from immature process, for CNFET based error resilient computing systems. However, successful adoption of CNFET technology for the implementation of advanced computing systems would still require work across the device, circuit, and system level. Our future work would thus focus on the following goals:

- **Device-level modifications**: In our current analysis, we have focused on open CNT imperfection, which is the major source of process imperfection affecting CNFET circuit-level performance. For more precise results and comparison with other competing technologies, we plan to incorporate other sources of process imperfections in our future analysis, including diameter variations, length variations, etc.

- **Complicated neural networks and datasets**: The complexity of neural networks has increased significantly over the past few years. We plan to implement these complex neural networks as part of our future work. Current state of art Convolutional Neural Networks (CNNs) [42] including ResNet [43], VGGNet [113] have shown significantly low error rates even for complicated dataset ImageNet [114]. For implementing these networks, we would first need to have a simulation framework to implement the convolution, pooling, and softmax layers of Convolutional Neural Network (CNN). The simulation results could then be utilized to get the system accuracy of these networks for complicated datasets like ImageNet [114], CIFAR-10 [115] etc.

- **Integration with standard frameworks**: We further plan to integrate our work with some of the popular deep learning frameworks like Caffe [116], TensorFlow [117] etc. This integration would be necessary to have reasonable computation time for complicated DNNs like ResNet [43], VGGNet [113]. Moreover, it would provide the common platform, thus increasing the possibility of more people to contribute towards this work.

- **Standard cell design**: The design of the entire complicated CNFET based computing system in hardware would require the use of automatic place and route tools, for which a library of standard cells should be designed. The standard cell library should possess different drive strengths even for the same functionality, with circuit design effort involving a special emphasis on reducing the impact of process imperfection.

References

[1] M. T. Bohr and I. A. Young, "CMOS scaling trends and beyond," *IEEE Micro*, vol. 37, no. 6, pp. 20–29, Nov. 2017.

[2] W. M. Holt, "1.1 moore's law: A path going forward," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Jan. 2016, pp. 8–13.

[3] T. N. Theis and H.-. P. Wong, "The end of moore's law: A new beginning for information technology," *Computing in Science Engineering*, vol. 19, no. 2, pp. 41–50, Mar. 2017.

[4] K. Kuhn, *CMOS and Beyond CMOS: Scaling Challenges.* Elsevier, 2018.

[5] T. Ghani, M. Armstrong, C. Auth, M. Bost, P. Charvat, G. Glass, T. Hoffmann, K. Johnson, C. Kenyon, J. Klaus *et al.*, "A 90nm high volume manufacturing logic technology featuring novel 45nm gate length strained silicon cmos transistors," in *IEEE International Electron Devices Meeting 2003.* IEEE, 2003, pp. 11–6.

[6] K. Mistry, C. Allen, C. Auth, B. Beattie, D. Bergstrom, M. Bost, M. Brazier, M. Buehler, A. Cappellani, R. Chau *et al.*, "A 45nm logic technology with high-

k+ metal gate transistors, strained silicon, 9 cu interconnect layers, 193nm dry patterning, and 100% pb-free packaging," in *2007 IEEE International Electron Devices Meeting*. IEEE, 2007, pp. 247–250.

[7] P. Packan, S. Akbar, M. Armstrong, D. Bergstrom, M. Brazier, H. Deshpande *et al.*, "High performance 32nm logic technology featuring 2¡ sup¿ nd¡/sup¿ generation high-k+ metal gate transistors," in *Electron Devices Meeting (IEDM), 2009 IEEE International*, 2009.

[8] J. Cartwright, "Intel enters the third dimension," *nature news*, 2011.

[9] C. Auth, C. Allen, A. Blattner, D. Bergstrom, M. Brazier, M. Bost, M. Buehler, V. Chikarmane, T. Ghani, T. Glassman *et al.*, "A 22nm high performance and low-power cmos technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density mim capacitors," in *2012 Symposium on VLSI Technology (VLSIT)*. IEEE, 2012, pp. 131–132.

[10] S.-Y. Wu, C. Lin, M. Chiang, J. Liaw, J. Cheng, S. Yang, C. Tsai, P. Chen, T. Miyashita, C. Chang *et al.*, "A 7nm cmos platform technology featuring 4 th generation finfet transistors with a 0.027 um 2 high density 6-t sram cell for mobile soc applications," in *2016 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2016, pp. 2–6.

[11] "", "Tsmc's 5nm fin field-effect transistor (finfet) process technology," 2019. [Online]. Available: https://www.tsmc.com/english/dedicatedFoundry/technology/5nm.htm

[12] J. Hruska, "Tsmc completes 5nm node design, node in risk production," 2019. [Online]. Available: https://www.extremetech.com/computing/289157-tsmc-completes-5nmnode-design-node-in-risk-production

[13] T. B. Hook, "Power and technology scaling into the 5 nm node with stacked nanosheets," *Joule*, vol. 2, pp. 1–4, 2018.

[14] I. Roadmap, "International technology roadmap for semiconductors 2.0 (itrs2. 0)," *Semiconductor Industry Association*, 2015.

[15] Y.-M. Lin, A. Valdes-Garcia, S.-J. Han, D. B. Farmer, I. Meric, Y. Sun, Y. Wu, C. Dimitrakopoulos, A. Grill, P. Avouris *et al.*, "Wafer-scale graphene integrated circuit," *Science*, vol. 332, no. 6035, pp. 1294–1297, 2011.

[16] S.-J. Han, A. V. Garcia, S. Oida, K. A. Jenkins, and W. Haensch, "Graphene radio frequency receiver integrated circuit," *Nature communications*, vol. 5, 2014.

[17] B. Radisavljevic, A. Radenovic, J. Brivio, V. Giacometti, and A. Kis, "Single-layer mos2 transistors," *Nature nanotechnology*, vol. 6, pp. 147–150, 2011.

[18] S. B. Desai, S. R. Madhvapathy, A. B. Sachid, J. P. Llinas, Q. Wang, G. H. Ahn, G. Pitner, M. J. Kim, J. Bokor, C. Hu, H.-S. P. Wong, and A. Javey, "Mos2 transistors with 1-nanometer gate lengths," *Science*, vol. 354, pp. 99–102, 2016.

[19] A. Nourbakhsh, A. Zubair, R. N. Sajjad, A. Tavakkoli KG, W. Chen, S. Fang, X. Ling, J. Kong, M. S. Dresselhaus, E. Kaxiras *et al.*, "Mos2 field-effect transistor with sub-10 nm channel length," *Nano letters*, vol. 16, no. 12, pp. 7798–7806, 2016.

[20] S. Wachter, D. K. Polyushkin, O. Bethge, and T. Mueller, “A microprocessor based on a two-dimensional semiconductor,” *Nature communications*, vol. 8, 2017.

[21] D. J. Perello, S. H. Chae, S. Song, and Y. H. Lee, “High-performance n-type black phosphorus transistors with type control via thickness and contact-metal engineering,” *Nature communications*, vol. 6, 2015.

[22] L. Li, M. Engel, D. B. Farmer, S. jen Han, and H.-S. P. Wong, “High-performance p-type black phosphorus transistor with scandium contact,” *ACS nano*, vol. 10, pp. 4672–4677, 2016.

[23] T. Li, Z. Zhang, X. Li, M. Huang, S. Li, S. Li, and Y. Wu, “High field transport of high performance black phosphorus transistors,” *Applied Physics Letters*, vol. 110, p. 163507, 2017.

[24] H. Park, A. Afzali, S.-J. Han, G. S. Tulevski, A. D. Franklin, J. Tersoff, J. B. Hannon, and W. Haensch, “High-density integration of carbon nanotubes via chemical self-assembly,” *Nature nanotechnology*, vol. 7, pp. 787–791, 2012.

[25] B. Kumar, A. L. Falk, A. Afzali, G. S. Tulevski, S. Oida, S.-J. Han, and J. B. Hannon, “Spatially selective, high-density placement of polyfluorene-sorted semiconducting carbon nanotubes in organic solvents,” *ACS nano*, vol. 11, pp. 7697–7701, 2017.

[26] S.-J. Han, J. Tang, B. Kumar, A. Falk, D. Farmer, G. Tulevski, K. Jenkins, A. Afzali, S. Oida, J. Ott, J. Hannon, and W. Haensch, “High-speed logic integrated circuits with solution-processed self-assembled carbon nanotubes,” *Nature nanotechnology*, vol. 12, pp. 861–865, 2017.

[27] W. Cao, J. Kang, D. Sarkar, W. Liu, and K. Banerjee, "2D semiconductor fets—projections and design for sub-10 nm VLSI," *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3459–3469, Nov. 2015.

[28] A. C. Neto, F. Guinea, N. M. Peres, K. S. Novoselov, and A. K. Geim, "The electronic properties of graphene," *Reviews of modern physics*, vol. 81, no. 1, p. 109, 2009.

[29] Y. Xu, Z. Shi, X. Shi, K. Zhang, and H. Zhang, "Recent progress in black phosphorus and black-phosphorus-analogue materials: properties, synthesis and applications," *Nanoscale*, vol. 11, no. 31, pp. 14 491–14 527, 2019.

[30] A. D. Franklin, M. Luisier, S.-J. Han, G. Tulevski, C. M. Breslin, L. Gignac, M. S. Lundstrom, and W. Haensch, "Sub-10 nm carbon nanotube transistor," *Nano letters*, vol. 12, pp. 758–762, 2012.

[31] N. Patil, J. Deng, S. Mitra, and H.-. P. Wong, "Circuit-level performance benchmarking and scalability analysis of carbon nanotube transistor circuits," *IEEE Transactions on Nanotechnology*, vol. 8, no. 1, pp. 37–45, Jan. 2009.

[32] L. Wei, D. J. Frank, L. Chang, and H.-. P. Wong, "A non-iterative compact model for carbon nanotube fets incorporating source exhaustion effects," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Dec. 2009, pp. 1–4.

[33] G. Hills, M. G. Bardon, G. Doornbos, D. Yakimets, P. Schuddinck, R. Baert, D. Jang, L. Mattii, S. M. Y. Sherazi, D. Rodopoulos *et al.*, "Understanding energy efficiency benefits of carbon nanotube field-effect transistors for digital vlsi," *IEEE Transactions on Nanotechnology*, vol. 17, no. 6, pp. 1259–1269, 2018.

[34] T. F. Wu, H. Li, P. Huang, A. Rahimi, J. M. Rabaey, H.-. P. Wong, M. M. Shulaker, and S. Mitra, “Brain-inspired computing exploiting carbon nanotube fets and resistive RAM: Hyperdimensional computing case study,” in *Proc. IEEE Int. Solid - State Circuits Conf. - (ISSCC)*, Feb. 2018, pp. 492–494.

[35] Q. Cao, S.-J. Han, J. Tersoff, A. D. Franklin, Y. Zhu, Z. Zhang, G. S. Tulevski, J. Tang, and W. Haensch, “End-bonded contacts for carbon nanotube transistors with low, size-independent resistance,” *Science*, vol. 350, pp. 68–72, 2015.

[36] G. Hills, C. Lau, A. Wright, S. Fuller, M. D. Bishop, T. Srimani, P. Kanhaiya, R. Ho, A. Amer, Y. Stein, D. Murphy, Arvind, A. Chandrakasan, and M. M. Shulaker, “Modern microprocessor built from complementary carbon nanotube transistors,” *Nature*, vol. 572, pp. 595–602, 2019.

[37] P. S. Kanhaiya, C. Lau, G. Hills, M. Bishop, and M. M. Shulaker, “1 Kbit 6T SRAM arrays in carbon nanotube FET CMOS,” in *Proc. Symp. VLSI Technology*, Jun. 2019, pp. T54–T55.

[38] M. M. Shulaker, G. Hills, R. S. Park, R. T. Howe, K. Saraswat, H.-S. P. Wong, and S. Mitra, “Three-dimensional integration of nanotechnologies for computing and data storage on a single chip,” *Nature*, vol. 547, no. 7661, p. 74, 2017.

[39] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, “Low-power digital signal processing using approximate adders,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[40] D. Monroe, "Neuromorphic computing gets ready for the (really) big time," *Communications of the ACM*, vol. 57, pp. 13–15, 2014.

[41] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[44] L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," in *Proc. Speech and Signal Processing 2013 IEEE Int. Conf. Acoustics*, May 2013, pp. 8604–8608.

[45] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *and Language Processing IEEE/ACM Transactions on Audio, Speech*, vol. 22, no. 4, pp. 778–784, Apr. 2014.

[46] Q. Cao, J. Tersoff, D. B. Farmer, Y. Zhu, and S.-J. Han, "Carbon nanotube transistors scaled to a 40-nanometer footprint," *Science*, vol. 356, no. 6345, pp. 1369–1372, 2017.

[47] A. D. Franklin and Z. Chen, "Length scaling of carbon nanotube transistors," *Nature nanotechnology*, vol. 5, no. 12, p. 858, 2010.

[48] C. Qiu, Z. Zhang, M. Xiao, Y. Yang, D. Zhong, and L.-M. Peng, "Scaling carbon nanotube complementary transistors to 5-nm gate lengths," *Science*, vol. 355, no. 6322, pp. 271–276, 2017.

[49] A. D. Franklin, S. O. Koswatta, D. B. Farmer, J. T. Smith, L. Gignac, C. M. Breslin, S.-J. Han, G. S. Tulevski, H. Miyazoe, W. Haensch, and J. Tersoff, "Carbon nanotube complementary wrap-gate transistors," *Nano letters*, vol. 13, pp. 2490–2495, 2013.

[50] D. Zhong, Z. Zhang, L. Ding, J. Han, M. Xiao, J. Si, L. Xu, C. Qiu, and L.-M. Peng, "Gigahertz integrated circuits based on carbon nanotube films," *Nature Electronics*, vol. 1, no. 1, p. 40, 2018.

[51] S. H. Jin, S. N. Dunham, J. Song, X. Xie, J.-h. Kim, C. Lu, A. Islam, F. Du, J. Kim, J. Felts *et al.*, "Using nanoscale thermocapillary flows to create arrays of purely semiconducting single-walled carbon nanotubes," *Nature nanotechnology*, vol. 8, no. 5, p. 347, 2013.

[52] M. M. Shulaker, G. Hills, N. Patil, H. Wei, H.-Y. Chen, H.-S. P. Wong, and S. Mitra, "Carbon nanotube computer," *Nature*, vol. 501, no. 7468, p. 526, 2013.

[53] M. M. Shulaker, G. Hills, T. F. Wu, Z. Bao, H.-. P. Wong, and S. Mitra, "Efficient metallic carbon nanotube removal for highly-scaled technologies," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Dec. 2015, pp. 32.4.1–32.4.4.

[54] S. J. Kang, C. Kocabas, T. Ozel, M. Shim, N. Pimparkar, M. A. Alam, S. V. Rotkin, and J. A. Rogers, "High-performance electronics using dense, perfectly aligned arrays of single-walled carbon nanotubes," *Nature nanotechnology*, vol. 2, no. 4, p. 230, 2007.

[55] S. W. Hong, T. Banks, and J. A. Rogers, "Improved density in aligned arrays of single-walled carbon nanotubes by sequential chemical vapor deposition on quartz," *Advanced materials*, vol. 22, no. 16, pp. 1826–1830, 2010.

[56] N. Patil, A. Lin, J. Zhang, Hai Wei, K. Anderson, H.-. P. Wong, and S. Mitra, "Vmr: VLSI-compatible metallic carbon nanotube removal for imperfection-immune cascaded multi-stage digital logic circuits using carbon nanotube fets," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Dec. 2009, pp. 1–4.

[57] G. S. Tulevski, A. D. Franklin, D. Frank, J. M. Lobez, Q. Cao, H. Park, A. Afzali, S.-J. Han, J. B. Hannon, and W. Haensch, "Toward high-performance digital logic technology with carbon nanotubes," *ACS nano*, vol. 8, no. 9, pp. 8730–8745, 2014.

[58] G. Zhang, P. Qi, X. Wang, Y. Lu, X. Li, R. Tu, S. Bangsaruntip, D. Mann, L. Zhang, and H. Dai, "Selective etching of metallic carbon nanotubes by gas-phase reaction," *Science*, vol. 314, no. 5801, pp. 974–977, 2006.

[59] S. Zhang, L. Kang, X. Wang, L. Tong, L. Yang, Z. Wang, K. Qi, S. Deng, Q. Li, X. Bai *et al.*, "Arrays of horizontal carbon nanotubes of controlled chirality grown using designed catalysts," *Nature*, vol. 543, no. 7644, p. 234, 2017.

[60] D.-m. Sun, M. Y. Timmermans, Y. Tian, A. G. Nasibulin, E. I. Kauppinen, S. Kishimoto, T. Mizutani, and Y. Ohno, “Flexible high-performance carbon nanotube integrated circuits,” *Nature nanotechnology*, vol. 6, no. 3, p. 156, 2011.

[61] Z. Hu, J. M. M. L. Comeras, H. Park, J. Tang, A. Afzali, G. S. Tulevski, J. B. Hannon, M. Liehr, and S.-J. Han, “Physically unclonable cryptographic primitives using self-assembled carbon nanotubes,” *Nature nanotechnology*, vol. 11, no. 6, p. 559, 2016.

[62] W. A. Gaviria Rojas, J. J. McMorrow, M. L. Geier, Q. Tang, C. H. Kim, T. J. Marks, and M. C. Hersam, “Solution-processed carbon nanotube true random number generator,” *Nano letters*, vol. 17, no. 8, pp. 4976–4981, 2017.

[63] T. Srimani, G. Hills, C. Lau, and M. Shulaker, “Monolithic three-dimensional imaging system: Carbon nanotube computing circuitry integrated directly over silicon imager,” in *Proc. Symp. VLSI Technology*, Jun. 2019, pp. T24–T25.

[64] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[65] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, “Loihi: A neuromorphic manycore processor with on-chip learning,” *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[66] S. K. Essera, P. A. Merollaa, J. V. Arthura, A. S. Cassidya, R. Appuswamya, A. Andreopoulosa, D. J. Berga, J. L. McKinstrya, T. Melanoa, D. R. Barcha *et al.*, "Convolutional networks for fast energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 41, pp. 11 441–11 446, 2016.

[67] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. ACM/IEEE 43rd Annual Int. Symp. Computer Architecture (ISCA)*, Jun. 2016, pp. 267–278.

[68] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[69] C.-S. Lee and H.-S. P. Wong, "Stanford virtual-source carbon nanotube field-effect transistors model," Apr 2015. [Online]. Available: https://nanohub.org/publications/42/2

[70] K. Sheikh and L. Wei, "Methodology to capture statistical effect of process imperfections on glitch suppression in CNFET circuits and to improve by using approximate circuits," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI, GLSVLSI 2018, Chicago, IL, USA, May 23-25, 2018*, D. Chen, H. Homayoun, and B. Taskin, Eds. ACM, 2018, pp. 27–32.

[71] K. Sheikh and L. Wei, "Evaluation of circuit performance degradation due to cnt process imperfection," in *2018 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA).* IEEE, 2018, pp. 1–2.

[72] K. Sheikh and L. Wei, "Using approximate circuits to counter process imperfections in cnfet based circuits," in *Proc. Automation and Test (VLSI-DAT) 2018 Int. Symp. VLSI Design*, Apr. 2018, pp. 1–4.

[73] K. Sheikh, S. Han, and L. Wei, "Cnfet with process imperfection: Impact on circuit-level yield and device optimization," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2209–2221, Dec. 2016.

[74] G. J. Brady, A. J. Way, N. S. Safron, H. T. Evensen, P. Gopalan, and M. S. Arnold, "Quasi-ballistic carbon nanotube array transistors with current density exceeding si and gaas," *Science advances*, vol. 2, p. e1601240, 2016.

[75] R. Levy, D. Blaauw, G. Braca, A. Dasgupta, A. Grinshpon, Chanhee Oh, B. Orshav, S. Sirichotiyakul, and V. Zolotov, "Clarinet: a noise analysis tool for deep submicron design," in *Proc. 37th Design Automation Conf*, Jun. 2000, pp. 233–238.

[76] K. L. Shepard, "Design methodologies for noise in digital integrated circuits," in *Proc. Design and Automation Conf.. 35th DAC. (Cat. No.98CH36175)*, Jun. 1998, pp. 94–99.

[77] A. Khakifirooz, O. M. Nayfeh, and D. Antoniadis, "A simple semiempirical short-channel MOSFET current–voltage model continuous across all regions of operation

and employing only physical parameters," *IEEE Transactions on Electron Devices*, vol. 56, no. 8, pp. 1674–1680, Aug. 2009.

[78] C.-. Lee, E. Pop, A. D. Franklin, W. Haensch, and H.-. P. Wong, "A compact virtual-source model for carbon nanotube fets in the sub-10-nm regime—part i: Intrinsic elements," *IEEE Transactions on Electron Devices*, vol. 62, no. 9, pp. 3061–3069, Sep. 2015.

[79] C. Lee, E. Pop, A. D. Franklin, W. Haensch, and H. P. Wong, "A compact virtual-source model for carbon nanotube fets in the sub-10-nm regime—part ii: Extrinsic elements, performance assessment, and design optimization," *IEEE Transactions on Electron Devices*, vol. 62, no. 9, pp. 3070–3078, Sep. 2015.

[80] S. J. Wind, J. Appenzeller, R. Martel, V. Derycke, and P. Avouris, "Vertical scaling of carbon nanotube field-effect transistors using top gate electrodes," *Applied Physics Letters*, vol. 80, pp. 3817–3819, 2002.

[81] P. Heydari and M. Pedram, "Capacitive coupling noise in high-speed VLSI circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 3, pp. 478–488, Mar. 2005.

[82] M. Nanua and D. Blaauw, "Investigating crosstalk in sub-threshold circuits," in *Proc. 8th Int. Symp. Quality Electronic Design (ISQED'07)*, Mar. 2007, pp. 639–646.

[83] F. Dartu, N. Menezes, J. Qian, and L. T. Pillage, "A gate-delay model for high-speed CMOS circuits," in *Proc. 31st Design Automation Conf*, Jun. 1994, pp. 576–580.

[84] J. Qian, S. Pullela, and L. Pillage, “Modeling the ”effective capacitance” for the RC interconnect of CMOS gates,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 12, pp. 1526–1535, Dec. 1994.

[85] T. Han and D. A. Carlson, “Fast area-efficient VLSI adders,” in *Proc. IEEE 8th Symp. Computer Arithmetic (ARITH)*, May 1987, pp. 49–56.

[86] D. Harris and I. Sutherland, “Logical effort of carry propagate adders,” in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 1. IEEE, 2003, pp. 873–878.

[87] J. Han and M. Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design,” in *2013 18th IEEE European Test Symposium (ETS)*. IEEE, 2013, pp. 1–6.

[88] Q. Xu, T. Mytkowicz, and N. S. Kim, “Approximate computing: A survey,” *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2015.

[89] S. T. Chakradhar and A. Raghunathan, “Best-effort computing: re-thinking parallel software and hardware,” in *Design Automation Conference*. IEEE, 2010, pp. 865–870.

[90] B. Moons, B. De Brabandere, L. Van Gool, and M. Verhelst, “Energy-efficient convnets through approximate computing,” in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–8.

[91] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "Salsa: systematic logic synthesis of approximate circuits," in *DAC Design Automation Conference 2012*. IEEE, 2012, pp. 796–801.

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

[93] S. G. Ramasubramanian, S. Venkataramani, A. Parandhaman, and A. Raghunathan, "Relax-and-retime: A methodology for energy-efficient recovery based design," in *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013, p. 111.

[94] M. Soeken, D. Große, A. Chandrasekharan, and R. Drechsler, "Bdd minimization for approximate computing," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2016, pp. 474–479.

[95] K. Sheikh and L. Wei, "Methodology to generate approximate circuits to reduce process induced degradation in cnfet based circuits," in *Proc. Int. Conf. Simulation of Semiconductor Processes and Devices (SISPAD)*, Sep. 2018, pp. 360–363.

[96] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *Computers, IEEE Transactions on*, vol. 100, no. 8, pp. 677–691, 1986.

[97] F. Somenzi, "Cudd: Cu decision diagram package release 3.0. 0 (2015)," *URL: http://vlsi. colorado. edu/˜ fabio/CUDD*.

[98] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. Wei, "14.3 a 28nm soc with a 1.2 ghz 568nj/prediction sparse deep-neural-network engine with¿

0.1 timing error rate tolerance for iot applications," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2017, pp. 242–243.

[99] P. Clarke, "Intel scales up self-learning neuromorphic computing," 2019. [Online]. Available: https://www.eenewsanalog.com/news/intel-scales-self-learning-neuromorphic-computing

[100] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, p. 92, 2019.

[101] S. Koppula, L. Orosa, A. G. Yağlıkçı, R. Azizi, T. Shahroodi, K. Kanellopoulos, and O. Mutlu, "Eden: Enabling energy-efficient, high-performance deep neural network inference using approximate dram," *arXiv preprint arXiv:1910.05340*, 2019.

[102] M. M. Lau and K. H. Lim, "Investigation of activation functions in deep belief network," in *2017 2nd international conference on control and robotics engineering (ICCRE)*. IEEE, 2017, pp. 201–206.

[103] K. Sheikh and L. Wei, "Reducing impact of cnfet process imperfections on shape of activation function by using connection pruning and approximate neuron circuit," in *International Symposium on Quality Electronic Design (ISQED)*, 2020 (accepted).

[104] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[105] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, "Truenorth: Design and tool flow of

a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[106] S. Das, B. U. Pedroni, P. Merolla, J. Arthur, A. S. Cassidy, B. L. Jackson, D. Modha, G. Cauwenberghs, and K. Kreutz-Delgado, "Gibbs sampling with low-power spiking digital neurons," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2015, pp. 2704–2707.

[107] M. Tanaka and M. Okutomi, "A novel inference of a restricted boltzmann machine," in *2014 22nd International Conference on Pattern Recognition*. IEEE, 2014, pp. 1526–1531.

[108] Z. Du, A. Lingamneni, Y. Chen, K. V. Palem, O. Temam, and C. Wu, "Leveraging the error resilience of neural networks for designing highly energy efficient accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1223–1235, 2015.

[109] Y. Kim, Y. Zhang, and P. Li, "Energy efficient approximate arithmetic for error resilient neuromorphic computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 11, pp. 2733–2737, 2014.

[110] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 19.

[111] Y. Kim, Y. Zhang, and P. Li, "An energy efficient approximate adder with carry skip for error resilient neuromorphic vlsi systems," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD).* IEEE, 2013, pp. 130–137.

[112] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[113] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[114] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[115] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[116] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia.* ACM, 2014, pp. 675–678.

[117] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems, 2015," *Software available from tensorflow. org*, vol. 1, no. 2, 2015.

Appendix A

Circuit-Level Yield Analysis with Short CNT Imperfection

In Chapter 3, we discussed about the two types of process imperfections introduced from the solution processed sorting and placement of CNTs: (1) the left-over metallic tubes in the separation step resulting in "short CNTs", always conducting even under the bias of the "off state"; (2) the trenches not fully covered by the CNTs during the placement step lead to "open CNTs", where CNTs are missing or unable to cover the entire trench. Thus the channel does not conduct current even under the bias of "on" state. In this chapter, we provide a methodology that links the two process imperfections short and open CNTs to the circuit-level yield. With the proposed methodology, we demonstrate that the open CNTs (with high percentage $> 30\%$) is of significant concern for the circuit-level yield and the effect of short CNTs (with low percentage $< 0.01\%$) on the circuit-level yield, can be safely ignored.

For expressing circuit-level yield, we consider both the drive current offered by the conducting branch, and the leakage current through the non-conducting branch. Since the purpose is to provide first-order estimation of the circuit-level yield for process guidelines and early technology assessment for digital applications, we will focus on the "on" and "off" states rather than the detailed DC/transient behavior of the transistor or circuit.

We first calculate the on and off currents of every single CNFET with a given number of semiconducting (N_{nor}), open (N_{open}), and short (N_{short}) CNTs. The currents of these CNFETs are then combined to obtain the currents of pull-up and pull-down branches in a CMOS circuit given the input. Finally, the probability of generating correct output under this input is evaluated by enumerating all possible combination of N_{nor}, N_{short}, N_{open} in each CNFET for single-stage and cascade circuits. Various contributions discussed in this chapter are published in ref. [73].

A.1 On and Off currents of a single CNFET

We use an ideal CNFET with N CNTs under the gate as the baseline device, where "ideal" means all of the N CNTs are semiconducting tubes (i.e. $N_{nor} = N$, $N_{open} = N_{short} = 0$). For this baseline CNFET, the current at "on" state (where $|V_{GS}| = |V_{DS}| = V_{DD}$) and "off" state (where $|V_{GS}| = 0$ and $|V_{DS}| = V_{DD}$) are calculated as $I_{on\_normal\_FET} = N.I_{on}$ and $I_{off\_normal\_FET} = N.I_{off}$, where I_{on} and I_{off} are the on current and off current of a single semiconducting CNT at on and off states, respectively. For the baseline reference, we choose $I_{on}/I_{off} = 10^4$ for the nominal devices based on practical design considerations and state-of-the-art CNT technology [30]. V_{GS}, V_{DS}, and V_{DD} are the gate to source, drain to source, and supply voltages, respectively.

With process imperfections, a CNFET designed to have N tubes under the gate has a combination of semiconducting, open and short CNTs, with $N_{short} + N_{open} + N_{nor} = N$. We ignore charge-screening effect and diameter variation; thus, each semiconducting CNT in a CNFET is assumed to have equal current (I_{on} during on state and I_{off} during off state). We also ignore the diameter and contact resistance differences between metallic and semiconducting tubes, therefore, the short CNT carries the current equal to I_{on}, irrespective of whether the CNFET is biased at on or off states. For the open CNT, the current is considered zero for both on and off modes. Hence, the conducting capability of CNFET is largely affected by the presence of open/short CNTs. For similar transistor geometry, the currents for CNT based nFET and pFET can be considered to be the same. The on and off currents for both nFET/pFET CNFETs are thus expressed as $I_{on\_FET} = N_{nor}.I_{on} + N_{short}.I_{on}$ and $I_{off\_FET} = N_{nor}.I_{off} + N_{short}.I_{on}$, respectively (Figure A.1).

A.2 Conducting/Non-Conducting Criteria for Pull-Up/Pull-Down Branches in CMOS Circuits

CNFETs have symmetric bandstructure for nFETs and pFETs, thus n-type and p-type CNFETs in an inverter can be sized with an equal number of CNTs for the same driving current [36]. In CMOS circuits, the transistors are usually sized to achieve certain driving capability equivalent to that of a reference inverter with N_{inv} tubes in both nFET and pFET. For example, a 2-input NAND is sized to have nFETs with $N_N = 12$ and pFETs

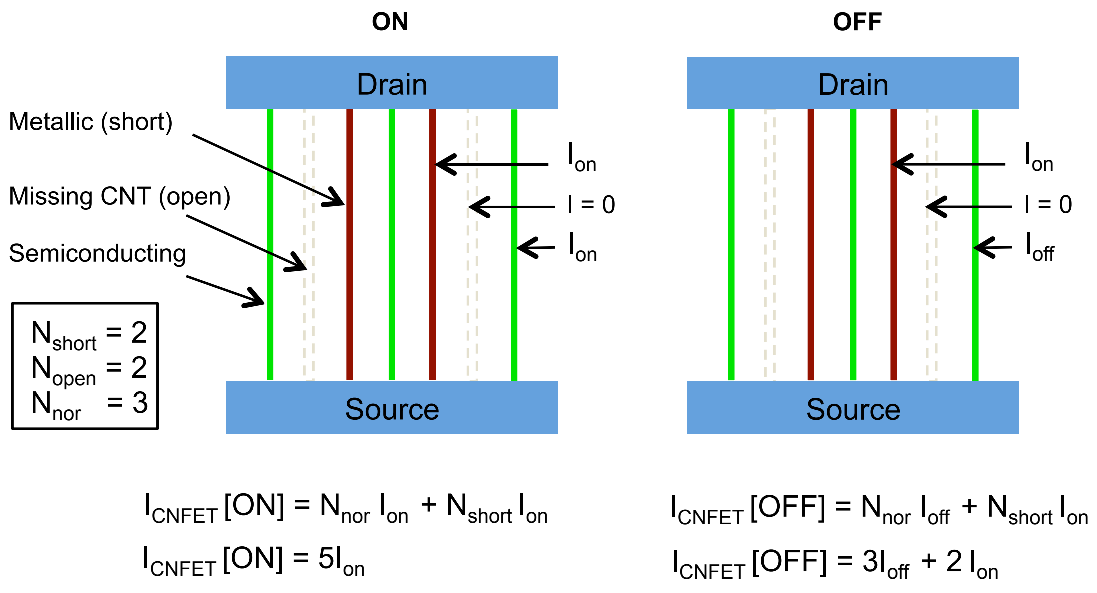

ON
OFF
Drain
Drain
Metallic (short)
Missing CNT (open)
Semiconducting
$N_{short} = 2$
$N_{open} = 2$
$N_{nor} = 3$
I_{on}
$I = 0$
I_{on}
I_{on}
$I = 0$
I_{off}
Source
Source
$I_{CNFET}[ON] = N_{nor} I_{on} + N_{short} I_{on}$
$I_{CNFET}[ON] = 5I_{on}$
$I_{CNFET}[OFF] = N_{nor} I_{off} + N_{short} I_{on}$
$I_{CNFET}[OFF] = 3I_{off} + 2 I_{on}$

Figure A.1: CNFET with process imperfections with $N = 7$, $N_{nor} = 3$, $N_{open} = 2$, $N_{short} = 2$. The current computation for device is shown for both on and off modes.

with $N_P = 6$ for a reference inverter with $N_{inv} = 6$ for both nFET and pFET. Assuming ideal transistors, N_{inv} is chosen to obtain the on and off currents (I_{on_inv} and I_{off_inv}, respectively) satisfying the speed, static power and noise margin requirements of the circuit.

Every single stage CMOS gate consists of pull-up (PU, for charging the output node to V_{DD}) and pull-down (PD, for pulling the output node to ground) branches. We first consider the cases where the output is expected to be 1 with given input vectors.

For output = 1, the PU branch is the conducting branch responsible to provide a current satisfying the speed requirement and PD branch is non-conducting with a leakage current not exceeding the static power requirement. For determining whether the stage of circuit with imperfect CNFETs is functional or not, the currents through PU (I_{PU}) and PD (I_{PD}) branches of the circuit, are compared with currents of reference inverter (I_{on_inv},

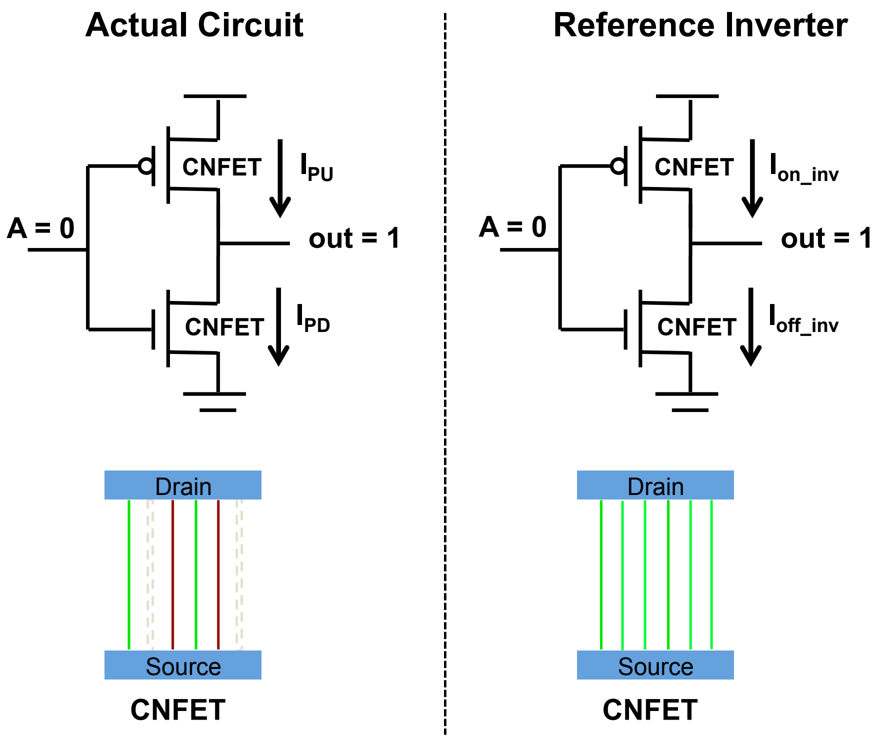


Figure A.2: (Left) Actual inverter circuit with imperfect CNFET for input A = 0, expected out = 1. (Right) Reference inverter circuit with ideal CNFET (all semiconducting CNTs) for input A = 0, out = 1.

I_{off_inv}). For an inverter circuit with imperfect CNFETs, I_{PU} and I_{PD} are simply equal to I_{on_FET} or I_{off_FET} depending on the input vector, where I_{on_FET} or I_{off_FET} are functions of N_{short}, N_{open}, and N_{nor} of each transistor as described in Section A.1. Figure A.2 shows the example of actual inverter circuit with imperfect CNFETs, and the reference inverter with ideal CNFETs for the expected output = 1. Both inverters have $N_{inv} = 6$; however, not all CNTs in the actual circuit are semiconducting tubes. For inverter circuit with imperfect CNFETs to be qualified as "functional" while the expected output is 1, two criteria must be satisfied:

1. **Conducting criterion**: The conducting branch (PU) has sufficient current to drive the load and meet certain speed requirement. We use $I_{PU} > 0.7 I_{on\_inv}$ in this analysis, assuming a 30% speed design margin.

2. **Non-conducting criterion**: The non-conducting branch (PD) has a reasonably low leakage current not to exceed the static power and noise margin requirement. We use $I_{PD} < 100 I_{off\_inv}$ for devices with nominal $I_{on}/I_{off} = 10^4$ in this analysis.

A similar approach is used to determine whether the circuit is functional when the output is expected to be 0. With output = 0, PU branch is non-conducting and PD branch is conducting.

A similar treatment can be extended to the case of having multiple transistors in PU or PD branch of the circuit. For such circuits again, each transistor is considered to contribute either I_{on_FET} or I_{off_FET} depending on the specific input vector. The total currents in the PU and PD branches (I_{PU} and I_{PD}, respectively) are calculated using these on and off currents from all the transistors in the branches according to the circuit configuration (i.e. transistors in series or in parallel).

A.3 Circuit-level Pass Rate

We define a term "pass rate" as the probability of the circuit generating the correct output. Criteria in previous section is used to determine whether the circuit output is considered "correct" with given N_{short}, N_{open}, and N_{nor} of each CNFET. Pass rate is calculated by summing the probabilities of all (N_{short}, N_{open}, N_{nor}) combinations that produce the correct output. It is expected that pass rate depends on process imperfection parameters ($PCNT_{open}$ and $PCNT_{short}$), as well as the circuit topology and the input vector.

We take the case of PU branch with expected output =1 as an example. For each combination of N_{short}, N_{open} and N_{nor}, I_{PU} is calculated. If $I_{PU} > 0.7 I_{on\_inv}$, PU branch is considered a qualified conducting branch and the variable IFI_{c_PU} in Equation A.1 is set to 1. Otherwise, IFI_{c_PU} is set to 0. $P_{N_{open},N_{short}}$ in Equation A.1 refers to the probability of N_{open}, N_{short} CNTs in CNFET provided by Equation 3.3. The overall probability of PU to be conducting (P_{onPU}) is calculated by summing the probability of (N_{open}, N_{short}) combination which satisfies $I_{PU} > 0.7 I_{on\_inv}$.

$$P_{onPU} = \sum_{N_{open}=0}^{N} \sum_{N_{short}=0}^{N-N_{open}} IFI_{c\_PU} . P_{N_{open},N_{short}} \tag{A.1}$$

The conducting probability (P_{onPD}) for PD is defined in a similar manner. The same method is used for finding non-conducting probability (P_{offPU}) for PU and non-conducting probability (P_{offPD}) for PD with conditions of $I_{PU} < 100I_{off\_inv}$ and $I_{PD} < 100I_{off\_inv}$, respectively.

Both conducting and non-conducting criteria must be satisfied to qualify a stage as functioning properly. Based on the input vector combination, the pass rate of expected output being 0 or 1 is given by Equation A.2 and Equation A.3, respectively.

$$passrate(output = 0) = P_{offPU}.P_{onPD} \tag{A.2}$$

$$passrate(output = 1) = P_{onPU}.P_{offPD} \tag{A.3}$$

Figure A.3 shows the block diagram for the pass rate computation of single stage CMOS circuits. The pass rate computation takes into account process parameters ($PCNT_{short}$, $PCNT_{open}$) as well as the input vector combination (VEC) in addition to the circuit configuration. Figure A.3 also shows the pass rate computation for single stage CMOS circuits for expected out = 1. For expected out = 1, we expect PU to be conducting (ON) and PD to be non-conducting (OFF). Thus, pass rate for expected out = 1 is calculated as Equation A.3.

The pass rate thus gives a measure of functionality of a stage, for a particular set of process imperfection parameters and circuit configuration under given input vectors. This methodology can also be used to determine the requirements for the material process quality in order to achieve target pass rate at the circuit level. The method of relating CNT process parameters with conducting/non-conducting branches can be revised according to specific processes of other channel materials, and the methodology can be extended for

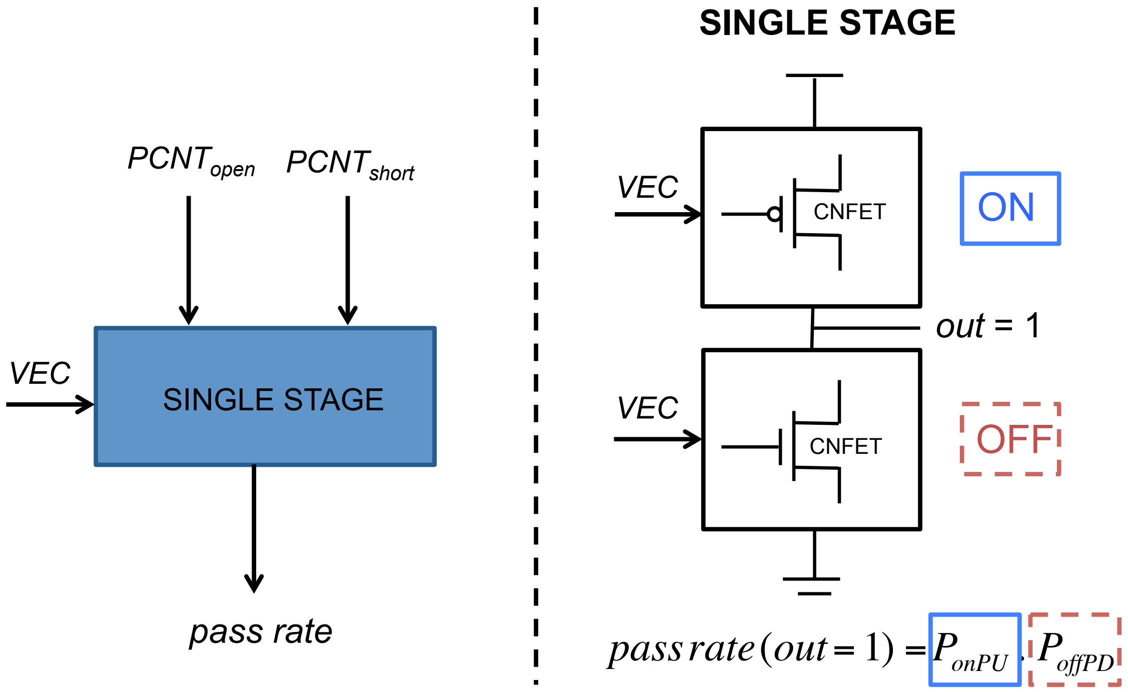


Figure A.3: (Left) Block diagram shows pass rate computation of single stage CMOS circuits incorporates process parameters ($PCNT_{open}$, $PCNT_{short}$) and input vector combination (VEC). (Right) Pass rate computation for single stage for expected output out = 1 following Equation A.3.

yield analysis of those materials as well. The acceptable circuit-level pass rate varies with different error-resilient applications. For further analysis, we use 80% as the acceptable pass rate as the reference.

A.4 Single Stage CMOS circuits

Pass rate of various commonly used CMOS circuit topologies are examined, including NAND, NOR and mirror adder (MA). All circuits mentioned in Figure A.4 are sized to have an equivalent driving capability as a reference inverter with $N_{inv} = 6$. For circuits with multiple transistors in PU and PD branches, the circuit pass rates of these topologies have a clear dependence on the input vectors, as illustrated in Figure A.4.

For most of the cases we discuss, the pass rate is dominated by the probability of having a qualified conducting branch. Thus, the more number of parallel transistors are on and the larger numbers of tubes in these transistors, the higher is the pass rate. With the same driving capability, an inverter, with only one transistor in PU and PD branches, gives the lower bound of the pass rate among all circuits and input.

Figure A.4 shows that 2-input and 3-input NAND gates achieve the best pass rates for the input vector combination [00] and [000] respectively. With input of [00] and [000] for 2- and 3- input NAND gates, respectively, all pFETs in the PU path are on, boosting up the drive current in the PU path, hence increasing the P_{onPU}. With symmetric PU and PD branches (Figure A.5), pass rate of MA is the same for vector combinations complement to each other (e.g ABC_{in} = [000] and [111]). Moreover, the pass rate for $(!C_{out})$ is the

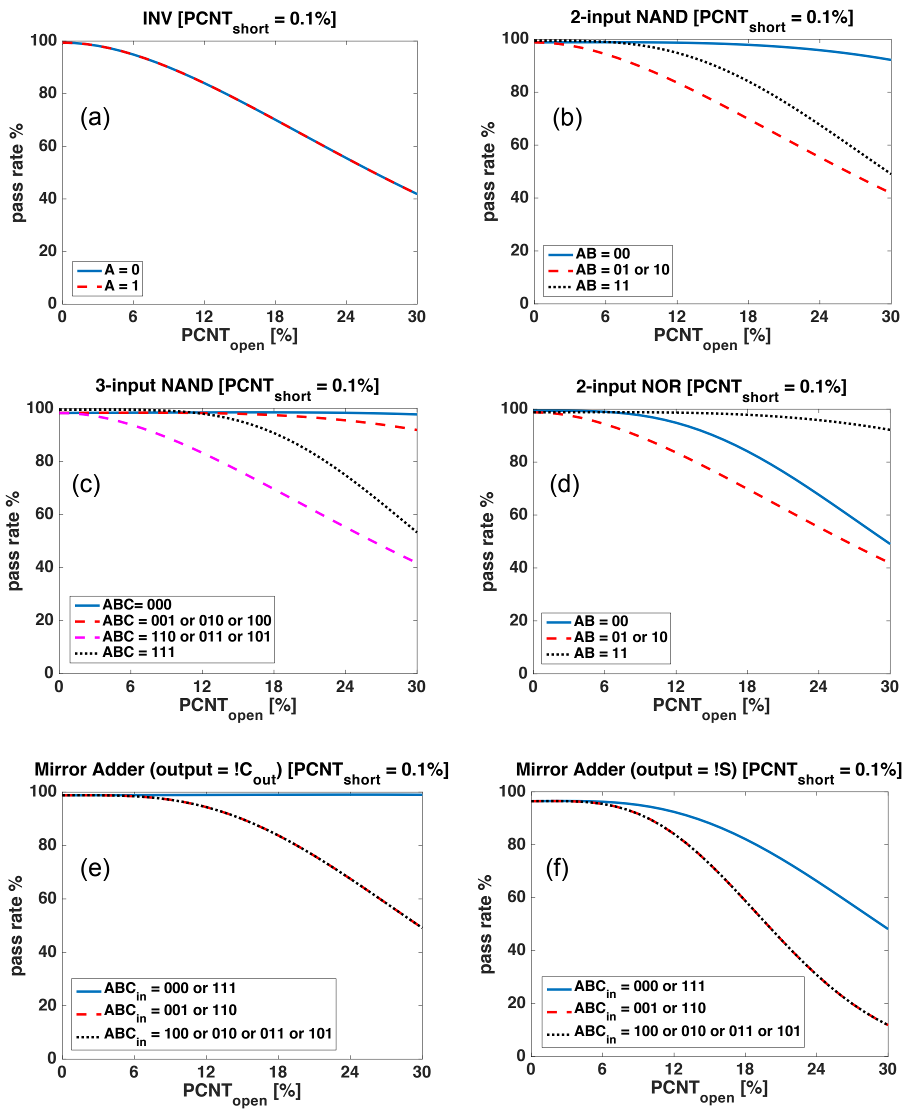


Figure A.4: Pass rate for single stage circuits for $PCNT_{short}$ = 0.1% (a) Inverter, (b) 2-input NAND, (c) 3-input NAND, (d) 2-input NOR, (e) MA (output = $!C_{out}$), (f) MA (output = $!S$). All other topologies are sized in accordance to the reference inverter (N_{inv} = 6). The inverter shows the lowest pass rate compared to all other topologies.

best with ABC_{in} = [000 or 111]. This is again due to more number of parallel transistors that are on for these vector combinations. For the 2-input and 3-input NAND gates shown in Figure A.4, the pass rate is the worst for the vectors having a single 0 in the vector combination. This is because only one of the pFETs in PU path is on under such vector combination. With input of [11] and [111] for 2- and 3- input NAND gates, respectively, only one PD path is on, however, the size of the nFETs is larger than that of the pFETs, hence the pass rate is higher than the cases with input of a single 0. For all the cases examined, even at high $PCNT_{short}$ ($PCNT_{short} = 0.1\%$), an 80% pass rate can be achieved with $PCNT_{open} \leq 13.5\%$.

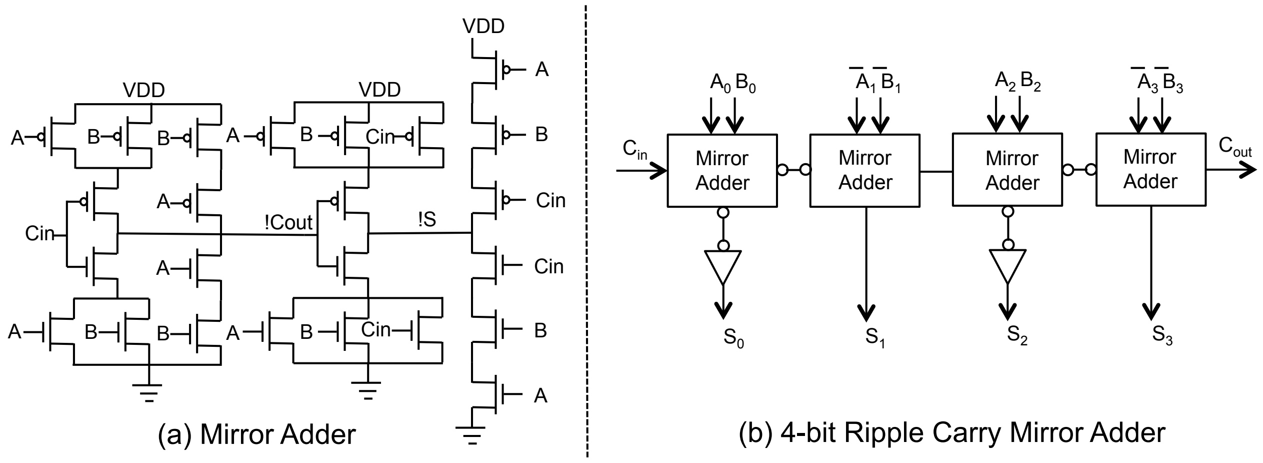


Figure A.5: (a) Mirror Adder schematic, (b) Block diagram of 4-bit RCA. Assume both true and inverted inputs, available for 4-bit RCA.

A.5 Cascaded Stages

The pass rate for the cascaded circuits is computed by traversing from input to output, by computing the pass rate after each stage. The pass rate after 1st stage is computed by calculating the probability of 1st stage output to be 0 (defined as PR_1^0) or 1 (defined as PR_1^1). The probability calculation is done using the single stage conducting/non-conducting criteria described in Section A.3, assuming perfect input to stage 1. The pass rate after 2nd stage is computed using two steps (Figure A.6). In the first step, the probability of 2nd stage output to be 0 (defined as PR_2^0) is calculated for two separate cases: (1) the output of 1st stage (i.e. input of 2nd stage) is 0 with a probability of PR_1^0 (the component of PR_2^0 obtained from this case is represented as $[PR_2^0]_0$) and (2) the output of 1st stage is 1 with a probability of PR_1^1 (resulting in $[PR_2^0]_1$).

In the second step, the results from the two cases ($[PR_2^0]_0$ and $[PR_2^0]_1$) are summed up to obtain the overall probability for 2nd stage output to 0. These two steps are repeated to obtain the probability of 2nd stage output to be 1 (defined as PR_2^1). In a similar manner, the probability of output to be 0 or 1 for each subsequent stage is computed till the final output is reached.

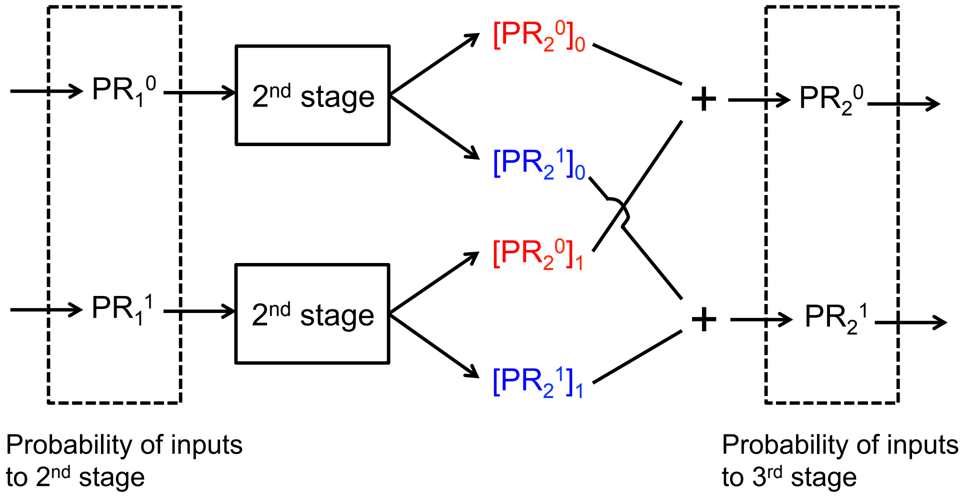


Figure A.6: Computation of pass rate for cascaded stages. The pass rate (probability) computation for 2nd stage output (input to 3rd stage) is shown for expected 2nd stage output to be 0 or 1.

A.6 Determine Process Requirement Based on Circuit-Level Pass Rate Target

The requirements of material process quality ($PCNT_{short}$ and $PCNT_{open}$) to achieve a specific pass rate target can be determined using the same methodology, given the circuit topology as well as the input vectors, if applicable.

The requirements of $PCNT_{short}$ and $PCNT_{open}$ to meet an 80% pass rate for different number of stages of FO1 inverter chain is shown in Figure A.7. For a 4-stage inverter chain, $PCNT_{open} \leq 2.5\%$ is required with $PCNT_{short} = 0.1\%$ (Figure A.7 (a)).

Figure A.7 (b) shows the process requirements for C_{out} of different bits of RCA. It can be seen from Figure A.7 (b), $PCNT_{open} \leq 5.6\%$ for C_{out} of a 4-stage RCA is required to achieve 80% pass rate with $PCNT_{short} = 0.1\%$.

For the same number of stages or the same pass rate, an inverter chain has much tighter process requirements compared to other circuits sized to have the same driving strength at $N_{inv} = 6$. The pass rate is dominated by $PCNT_{open}$ if $PCNT_{short}$ is lower than 0.01%. The benefit of further reducing $PCNT_{short}$ below 0.01% is marginal.

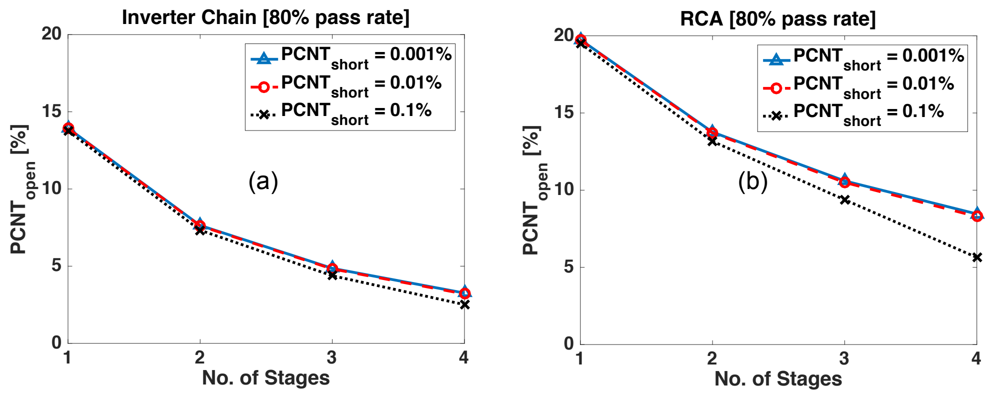


Figure A.7: (a) Process requirements ($PCNT_{open}$, $PCNT_{short}$) to achieve an 80% pass rate for FO1 inverter chain with N_{inv}=6 with different number of stages. (b) Process requirements ($PCNT_{open}$, $PCNT_{short}$) to achieve 80% pass rate for C_{out} of different numbers of bits of RCA under worst case input.

Appendix B

Glitch Analysis for Additional Approximate Circuits

In Chapter 4, we discussed that comparing with precise circuits, using approximate circuits is an effective way to reduce the number of glitches exceeding glitch criteria due to process imperfections. In this chapter, we provide glitch analysis for additional circuits, including multiplier (4-bit Wallace Multiplier) circuit. We take the case of 4-bit Ripple Carry Adder (RCA) and 4-bit Wallace Multiplier (WM) as reference. 4-bit Ripple Carry Adder (RCA) is a simpler and small circuit (in total 14 internal/output nodes), which can help to easily correlate the result to the circuit design. Additionally, we consider 4-bit Wallace Multiplier (WM), which would show the applicability of the methodology to slightly complex circuits ($>$ 50 internal/output nodes). Further, in this chapter, we will introduce different approximate designs for 4-bit adder and 4-bit multiplier, which utilize the blocks in Figure B.1. This comes with a penalty of logic accuracy. However, the significant reduction in the number of nodes failing glitch criteria justifies the logic accuracy degradation in particular for high $PCNT_{open}$ ($> 10\%$). Moreover, as explained before, given the other performance benefits such as delay, power, the improvement in glitch vulnerability to a tolerable level enables the emerging technology to be used for circuit design targeting error resilient applications. To account for the logic error in approximate designs, we define the term $\%Error_{Logic}$ in a standard way,

$$\%Error_{Logic} = mean\left(\left|\frac{S_{approx} - S_{orig}}{S_{orig}}\right| * 100\right) \tag{B.1}$$

where S_{orig} is the original sum/multiplier value based on input vector combination, S_{approx} is the sum/multiplier value computed based on output of the adder/multiplier. The term 'mean' refers to the average over all possible input vector combination for the 4-bit adder/multiplier. Various contributions discussed in this chapter are published in ref. [70].

B.1 4-bit CNFET RCA Precise/Approximate Circuits

We have constructed 2 approximate 4-bit adder circuits using combination of (MA, MA2 introduced in Figure B.1) and buffer circuits (Figure B.2). The idea of using MA2, buffer circuits is taken from [39], which explains the advantage of using them over the precise MA counterpart in terms of speed, power, area, with some compromise in logic accuracy. The 2 approximate adder circuits represented as [RCA (app-a), RCA (app-b)] differ in logic accuracy and area (Figure B.2). The introduction of MA2 in place of MA replaces the two

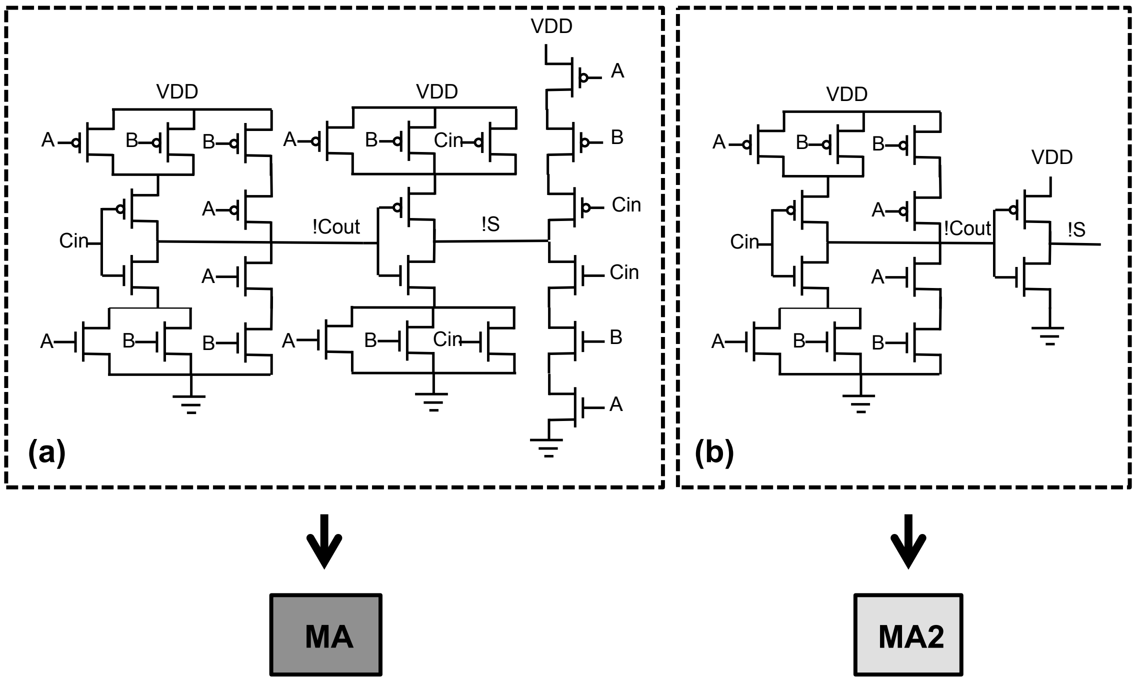
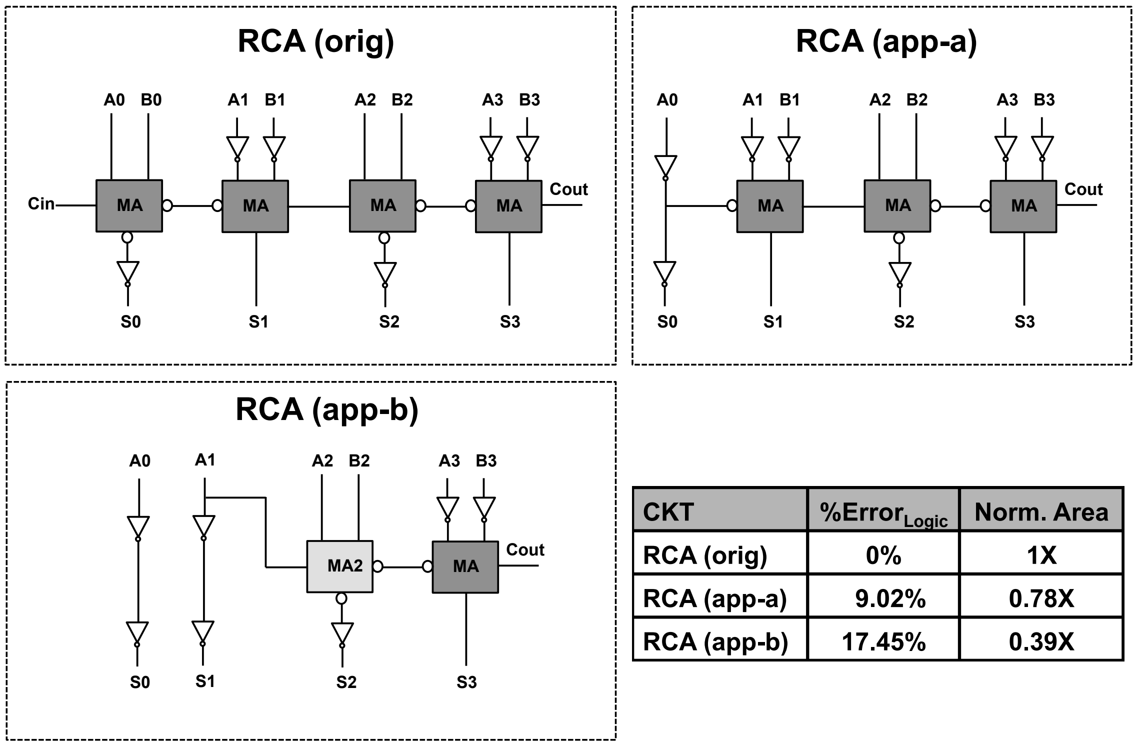


Figure B.1: (From Left to Right) (a) Precise Mirror Adder 'MA'. (b) Approximate Mirror Adder schematic 'MA2' taken from [39]. 'MA2' has the accurate circuitry for carry ($!C_{out}$), however, the sum ($!S$) is approximated. Both circuits 'MA', 'MA2' would be represented by their box symbol or by letters 'MA', 'MA2' for rest of this chapter.



CKT	%Error$_{Logic}$	Norm. Area
RCA (orig)	0%	1X
RCA (app-a)	9.02%	0.78X
RCA (app-b)	17.45%	0.39X

Figure B.2: Shows the schematic of 4-bit precise Ripple Carry adder 'RCA (orig)', 4-bit approximate adders 'RCA (app-a)', 'RCA (app-b)'. $\%Error_{Logic}$ is computed over all the possible 512 input vectors. The area of RCA (app-a) and RCA (app-b) are 0.78X and 0.39X that of the precise adder RCA (orig) respectively.

stacked branches by single transistor in each PU/PD at sum (!S) node (Figure B.1). This helps to improve glitch tolerance for a couple of reasons. 1) The single transistor provides required driving current with a smaller size instead of stacked transistors in MA which need increased sizes to provide the drive current, more number of transistors connected to node !S, and thus suffers from increased coupling capacitor at the node !S in comparison to MA2. 2) In MA2, there are fewer transistors connected to each input A, B, C_{in} in comparison to MA; this reduces the number of connections and thus results in a reduction in coupling capacitor for nodes driving each of the input A, B, C_{in} of MA2. So, both the node !S and nodes driving A, B, C_{in} benefits from reduced coupling capacitor and thus become more glitch tolerant for MA2 in place of MA. Similar explanation can be applied for buffer circuits where both carry (!C_{out}) and !S are approximated, and the benefits are even more in terms of glitch tolerance. An additional benefit with approximate circuits is the reduced number of internal/output nodes. In comparison to 14 internal/output nodes of precise RCA (orig), approximate circuits RCA (app-a), RCA (app-b) have 13, 11 internal/output nodes, respectively.

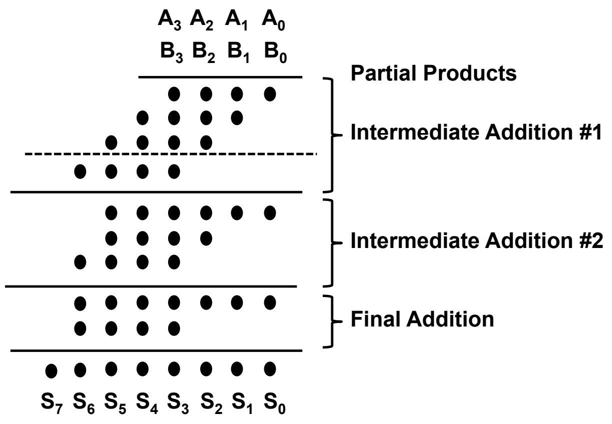
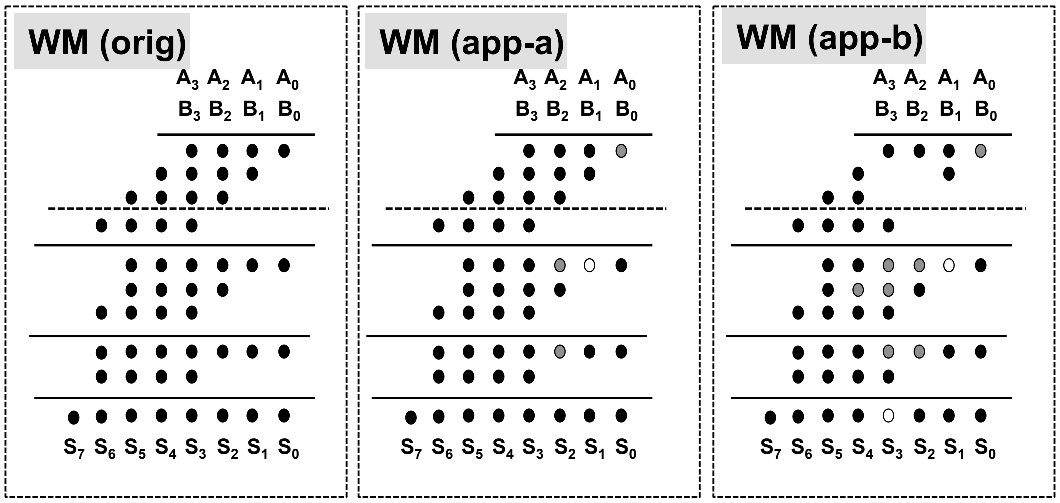


Figure B.3: Schematic shows different stages of precise 4-bit Wallace Multiplier.



CKT	%Error$_{Logic}$	Norm.Area
WM (orig)	0%	1X
WM (app-a)	8.78%	0.90X
WM (app-b)	21.42%	0.69X

Figure B.4: (Starting from left) shows the schematic of 4-bit precise Wallace multiplier 'WM (orig)', 4-bit approximate multiplier 'WM (app-a)', 'WM (app-b)'.$\%Error_{Logic}$ is computed over all the possible 256 input vectors. The area of WM (app-a) and WM (app-b) are 0.90X and 0.69X that of the precise multiplier WM (orig) respectively.

B.2 4-bit CNFET Wallace Multiplier Precise/Approximate Circuits

Wallace Multiplier (WM) is commonly used for the purpose of fast multiplication. Figure B.3) shows the schematic of precise 4-bit Wallace Multiplier. The partial products in the first stage are followed by intermediate addition stages consisting of full adder and half adder blocks, followed by a 4-bit adder in the final stage. The full adder blocks can be approximated using MA2, buffer circuits (as discussed in the previous section). We have proposed two approximate 4-bit multiplier circuits using MA2, buffer circuits for full adder blocks (Figure B.4). The two approximate multiplier circuits represented as [WM (app-a), WM (app-b)] differ in logic accuracy and area (Figure B.4)). Gray dots in 'WM (app-a)' and 'WM (app-b)' (Figure B.4)) represent the approximate sum ($!S$)/carry ($!C_{out}$) signals from MA2 or buffer circuits. White dots represent approximate sum signal of half adder where NAND2/NOR2 gates are used instead of XOR2 gates (this simplification also reduces the number of nodes). Some of the partial products are not required as part of the approximation approach and hence omitted in 'WM (app-b)'. As explained before, the overall effect of the simplifications in approximate circuits would be the enhancement of glitch tolerance at some nodes and also the reduction in the number of nodes. In comparison to 56 internal/output nodes of precise WM (orig), approximate circuits WM (app-a), WM (app-b) have 52, 48 internal/output nodes respectively.

B.3 RESULTS AND DISCUSSION

B.3.1 Fail Nodes in the whole Circuit

$MeanFailNodes_{CKT}$ provides average estimate of total number of failing nodes of a circuit at given $PCNT_{open}$. Figure B.5(a) show that $MeanFailNodes_{CKT}$ for precise and all approximate 4-bit RCA is quite small (< 2) for $PCNT_{open} \leq 10\%$. However, for higher $PCNT_{open}$ range, $MeanFailNodes_{CKT}$ for RCA (orig) cannot be ignored. At $PCNT_{open}$ $= 40\%$, $MeanFailNodes_{CKT}$ for RCA (orig) becomes a significant percentage (70.2%)

of its total number of nodes (9.8 out of 14 nodes). However, with RCA (app-b) the $MeanFailNodes_{CKT}$ at $PCNT_{open} = 40\%$ is lower by 59.4% down to 4.0 nodes in comparison to precise RCA (orig). Similarly in Figure B.5(b), approximate 4-bit multiplier WM (app-b) has significantly lower $MeanFailNodes_{CKT}$ over its precise WM (orig) counterpart for $PCNT_{open} \geq 5\%$. At $PCNT_{open} = 20\%$, with WM (app-b) the $MeanFailNodes_{CKT}$ is only 5.6 compared with 20.6 of WM (orig) (lower by 72.7%) (Figure B.5(b)). For a circuit with about 50 nodes, that is a significant reduction in the number of vulnerable nodes, hence a prominent improvement in glitch tolerance. The reasons for the improvement in glitch tolerance in both RCA (app-b) and WM (app-b) in comparison to their precise counterparts are the reduction in number of nodes and improvement in glitch tolerance of some of the nodes, as explained before.

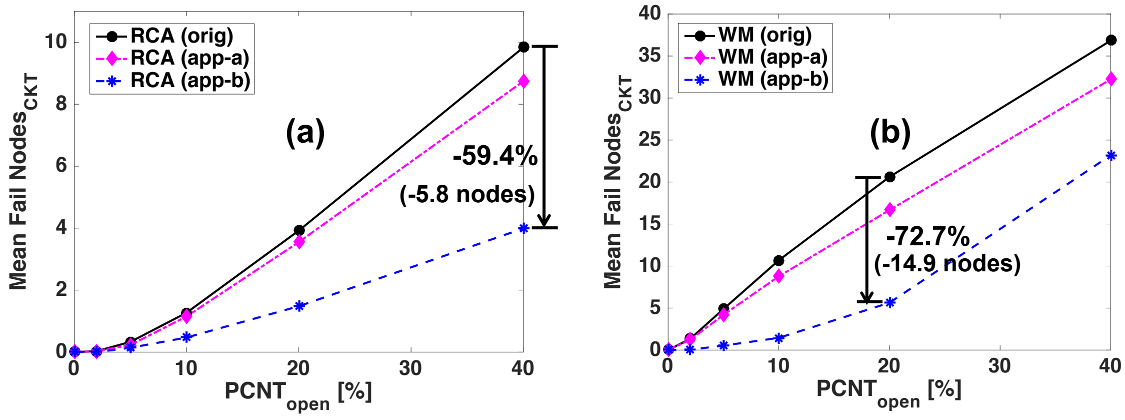


Figure B.5: $MeanFailNodes_{CKT}$ for different precise and approximate circuits of (a) 4-bit adder, (b) 4-bit multiplier. Using approximate circuits significantly reduces the number of vulnerable nodes which fail the glitch criteria.

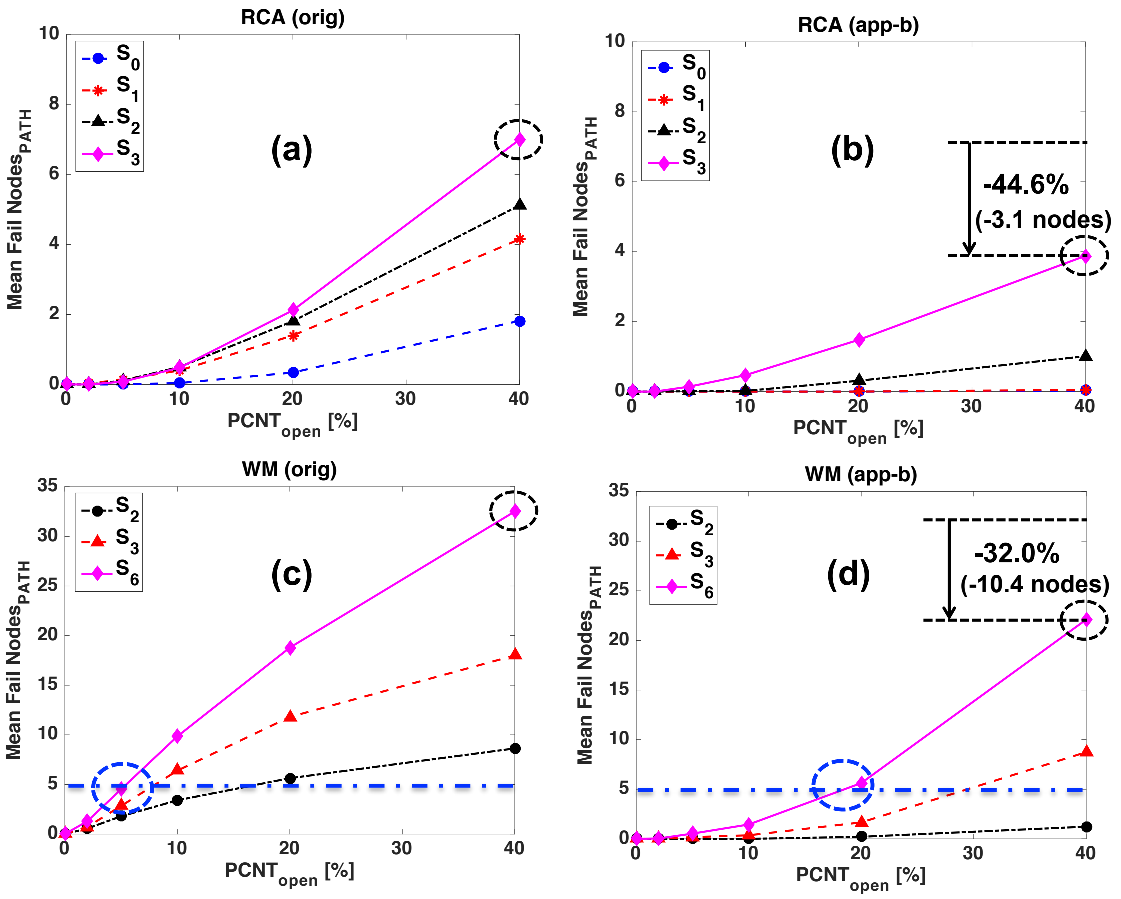


Figure B.6: $MeanFailNodes_{PATH}$ for (a) RCA (orig). (b) RCA (app-b), (c) WM (orig), (d) WM (app-b). Black dotted circle indicate the critical output (worst $MeanFailNodes_{PATH}$) at $PCNT_{open}$ =40%. Highlighted blue line and blue dotted circle indicate that using WM (app-b) instead of WM (orig), the requirement of $PCNT_{open}$ is largely reduced to achieve $MeanFailNodes_{PATH}$ below certain target value (5 in this case).

B.3.2 Fail Nodes along a path

Figure B.6(a), (b) show that $MeanFailNodes_{PATH}$ for both precise and approximate 4-bit RCA is negligible for outputs S_0, S_1, S_2, S_3 for $PCNT_{open} \leq 10\%$. S_3 with highest number of internal nodes connected to its path from the primary inputs, has the expected worst $MeanFailNodes_{PATH}$ over the entire $PCNT_{open}$ range for both RCA (orig) and RCA (app-b) (Figure B.6 (a), (b)). As towards the most significant bit, S_3 is considered as a critical output for both RCA (orig) and RCA (app-b). At $PCNT_{open} = 40\%$, using RCA (app-b) $MeanFailNodes_{PATH}$ of S_3 is only 3.8 compared with 7 of RCA (orig) (lower by 44.6%). For both WM (orig) and WM (app-b), S_6 is the critical output with highest $MeanFailNodes_{PATH}$ over the entire $PCNT_{open}$ range. At $PCNT_{open} = 40\%$, S_6 of WM (app-b) is 22.1 compared with 32.5 of WM (orig) (lower by 32.0%) (Figure B.6 (c), (d)). Assuming a design target of $MeanFailNodes_{PATH} < 5$ is set for this application, we can observe that using WM (app-b) can largely relax the requirement of $PCNT_{open}$ (from $\sim$ 5% to $\sim$ 20%) in comparison to WM (orig) (Figure B.6 (c), (d)).

B.3.3 Choosing Optimum Circuit

The improvement of glitch tolerance, and imperfect process induced failure in general, by using approximate circuits, comes with the penalty of logic inaccuracy. Tradeoff between the two must be considered. Figure B.7(a) show that at $PCNT_{open} = 40\%$, $MeanFailNodes_{CKT}$ with RCA (app-b) reduces by 5.8 nodes (59.4%) in comparison to its precise counterpart RCA (orig) with a penalty of logic error 17%. However, this reduction of 5.8 nodes is significant considering the fact that RCA (orig) in total has only 14 internal/output nodes. Similar trend is observed in Figure B.7(b), where $MeanFailNodes_{PATH}$ of S_3 with RCA (app-b) are reduced by 44.6% at $PCNT_{open} = 40\%$ in comparison to RCA (orig). In comparison to RCA (app-b), RCA (app-a) has lesser logic error of 9% but the reduction in $MeanFailNodes_{CKT}$ and $MeanFailNodes_{PATH}$ of S_3 is only 1.1 and 0.2 nodes respectively compared to RCA (orig) at $PCNT_{open} = 40\%$. Hence, RCA (app-b) is considered as a better option at $PCNT_{open} = 40\%$. Figure B.7(c) show that in comparison to WM (orig) $MeanFailNodes_{CKT}$ with WM (app-b) is reduced by 14.9 nodes and 13.6 nodes at $PCNT_{open} = 20\%/PCNT_{open} = 40\%$ respectively, with a logic error of 21%. Figure B.7(d) show a similar trend for $MeanFailNodes_{PATH}$ for S_6, wherein using WM (app-b) reduces the $MeanFailNodes_{PATH}$ by 13.1 nodes and 10.4 nodes at $PCNT_{open} = 20\%/PCNT_{open} = 40\%$ respectively. With WM (app-a), the reduction in $MeanFailNodes_{CKT}$ in comparison to WM (orig) is 3.9 nodes and 4.6 nodes at $PCNT_{open} = 20\%/PCNT_{open} = 40\%$ respectively (Figure B.7(c)), which is much lesser compared to improvement seen with WM (app-b). So, for applications where we want to reduce logic error below 10%, WM (app-a) will be considered in place of WM (app-b). However, for error resilient applications, WM (app-b) will be the optimum choice.

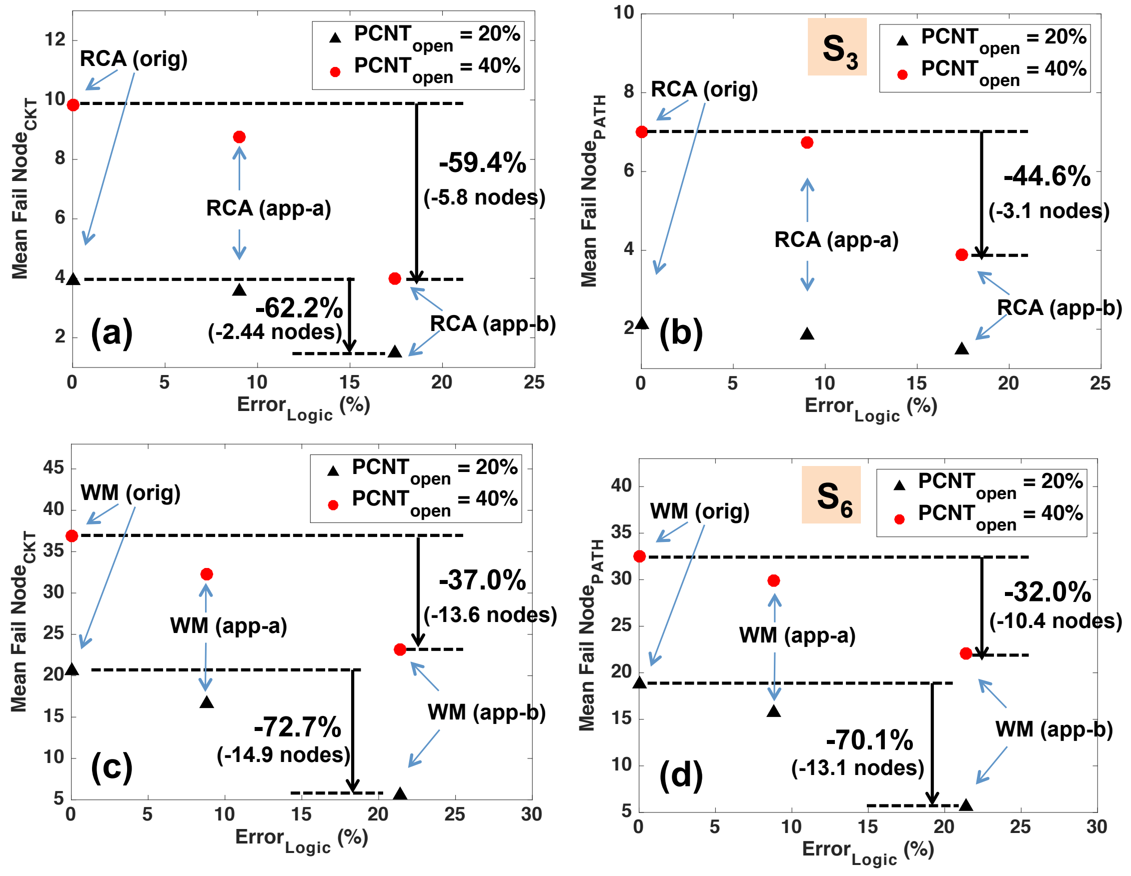


Figure B.7: Plot for 4-bit adder between (a) $MeanFailNodes_{CKT}$ and $\%Error_{Logic}$, (b) $MeanFailNodes_{PATH}$ for S_3 and $\%Error_{Logic}$. Plot for 4-bit multiplier between (c) $MeanFailNodes_{CKT}$ and $\%Error_{Logic}$, (d) $MeanFailNodes_{PATH}$ for S_6 and $\%Error_{Logic}$.

Appendix C

Framework for CNFET Monte Carlo Seed Generation

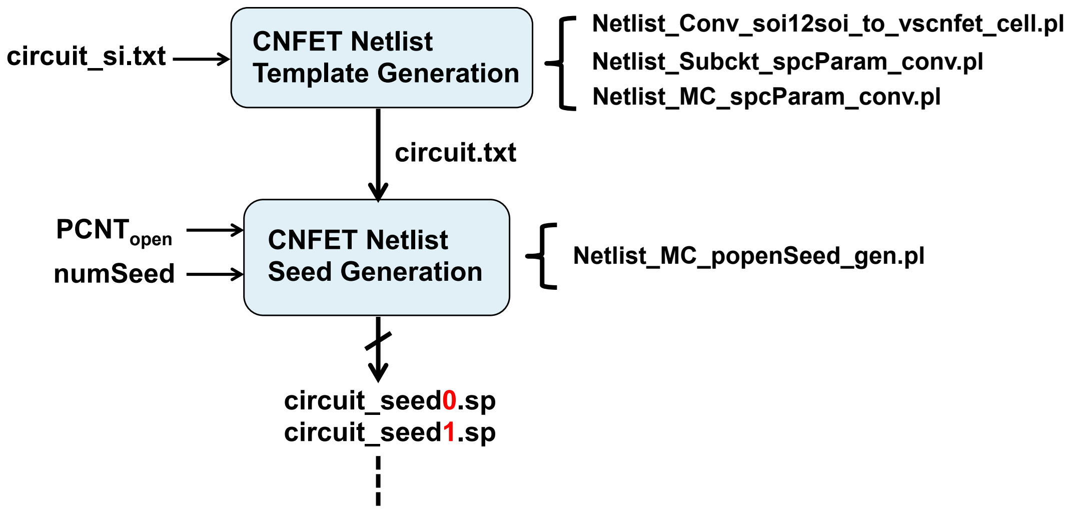

circuit_si.txt
CNFET Netlist Template Generation
Netlist_Conv_soi12soi_to_vscnfet_cell.pl
Netlist_Subckt_spcParam_conv.pl
Netlist_MC_spcParam_conv.pl
circuit.txt
PCNT_open
numSeed
CNFET Netlist Seed Generation
Netlist_MC_popenSeed_gen.pl
circuit_seed0.sp
circuit_seed1.sp

Figure C.1: (a) Steps (along with list of codes) for generation of multiple copies of CNFET based netlist (each acting as seed/sample for Monte Carlo run).

In Figure C.1, we provide further details of the framework (discussed briefly in section 3.5) for CNFET based Monte Carlo Seed generation. As discussed before, our simulation framework (Figure 3.5) utilizes existing Si based library for the schematic generation in Cadence Virtuoso, followed by generating Si based netlist ($circuit\_si.txt$) with connectivity information. The framework (Figure C.1) utilizes connectivity information in $circuit\_si.txt$, converts it into CNFET based netlist ($circuit.txt$), followed by generation of multiple copies (each acting as seed/sample for Monte Carlo run) of $circuit.txt$, by assigning values from $SpaceFile$ corresponding to given $PCNT_{open}$. Figure C.2(a) provides the set of commands (utilizing the codes listed in Figure C.1) for generation of CNFET based Monte Carlo seeds.

```
perl Netlist_Conv_soi12soi_to_vscnfet_cell.pl -netlistFile  TmpFiles/circuit_si.txt
-cellFile TmpFiles/circuit_cell.txt > TmpFiles/circuit_SPACE.txt

perl Netlist_Subckt_spcParam_conv.pl -netlistFile TmpFiles/circuit_SPACE.txt
-cellFile TmpFiles/circuit_cell.txt > TmpFiles/circuit_spc.txt

perl Netlist_MC_spcParam_conv.pl -inFile  TmpFiles/circuit_spc.txt
cat VSCNFET_HeaderFile.txt TmpFiles/circuit_spc.txt >  TmpFiles/circuit_post_spc.txt
mv TmpFiles/circuit_post_spc.txt circuit.txt

perl Netlist_MC_popenSeed_gen.pl -inFile circuit.txt -spcDir spaceFiles
-poFile dir_popen.txt -runDir . -numseed 50
```

(a)

circuit-si.txt	Silicon (Si) based circuit netlist
VSCNFET_Header.txt	File containing list of parameter values used for VSCNFET model
dir_popen.txt	File containing list of $PCNT_{open}$
spaceFiles	Directory containing spaceFiles
TmpFiles	Directory for temporary files generated

(b)

Figure C.2: (a) Set of commands for generation of multiple copies of CNFET based netlist (each acting as seed/sample for Monte Carlo run) (b) Description of files/directories referred in (a).

C.1 Codes for CNFET Monte Carlo Seed Generation

```
################################################################
### Netlist_Conv_soi12soi_to_vscnfet_cell.pl  ##########
################################################################
## Usage text ##
$USAGE  = "\n";
$USAGE .= "Usage:Netlist_Conv_soi12soi_to_vscnfet_cell.pl -netlistFile <NETLISTFILE>\
$USAGE .= " -cellFile <CELLFILE>\n";

## Get switch information ##
ParseArgs();
my %num_xtor;
$cnfet_param = "+Lg=Lg Lc=Lc Lext=Lext Hg=Hg Geomod=Geomod Vfb=Vfbp d=Dia\n";
$cnfet_param .= "+SDTmod=SDTmod BTBTmod=BTBTmod Rcmod=Rcmod Rs0=Rs0";
CreateDictNumXtor();

open(FNET, "$netlistFile") or die "Cannot open $netlistFile";
while($line_net = <FNET>)
{
  chop($line_net);
  # Modify FET description
  if ($line_net =~ /\s*xt([0-9].*)\s+(\S+.*)d_([a-z]*)\s+l=(\S+)\s+w=(\S+).*/)
  {
    $inst = $1; $ports4 = $2; $model = $3; $len = $4; $width = $5;
    if($ports4 =~ /\s*(\S+.*)\s+0/)
    {
      $ports3 = $1;
    }
    if ($model eq 'nfet')
    {
      print "xmn$inst $ports3 vscnfet_1_0_1 FETtype=1 W=$width s=SPACE\n$cnfet_param\
    } elsif ($model eq 'pfet') {
      print "xmp$inst $ports3 vscnfet_1_0_1 FETtype=-1 W=$width s=SPACE\n$cnfet_paran
    }
  }
  # Modify Instance description
  elsif ($line_net =~ /\s*x[ig](\S+)\s+(.*)\s+(\S+)\s*$/)
  {
    $inst = $1; $ports = $2; $cell = $3;
    my $numXtor = GetNumXtorPerCell($cell);
    my $spaceExpr = GetSpaceParamlist($numXtor);
    print "xi$inst $ports $cell $spaceExpr\n";
  }
  # Modify subckt description
  elsif ($line_net =~ /\s*.subckt\s+(\S+)\s+(.*)\s*$/)
  {
    $cell = $1; $ports = $2;
    my $numXtor = GetNumXtorPerCell($cell);
    my $spaceExpr = GetSpacelistForSubckt($numXtor);
    print ".subckt $cell $ports $spaceExpr\n";
  }
  else
  {
    print "$line_net\n";
  }
}
close FNET;

## Parse Args ##
sub GetSpacelistForSubckt()
{
  my $numXtor = $_[0];
  my $spaceVar = 1; # Initialize (spc1)
  my $spaceExpr = '';
  while ($spaceVar < $numXtor)
  {
    $spacetemp = "spc${spaceVar}='s'";
    $spaceExpr = $spaceExpr . $spacetemp . ' ';
    $spaceVar++;
  }
  $spacetemp = "spc${spaceVar}='s'";
  $spaceExpr = $spaceExpr . $spacetemp;
  return $spaceExpr;
```

```
}
sub GetSpaceParamlist()
{
  my $numXtor = $_[0];
  my $spaceVar = 1; # Initialize (spc1)
  my $spaceExpr = '';
  while ($spaceVar < $numXtor)
  {
    $spacetemp = "spc${spaceVar}=SPACE";
    $spaceExpr = $spaceExpr . $spacetemp . ' ';
    $spaceVar++;
  }
  $spacetemp = "spc${spaceVar}=SPACE";
  $spaceExpr = $spaceExpr . $spacetemp;
  return $spaceExpr;
}
sub GetNumXtorPerCell()
{
  my $cell = $_[0]; my $numXtor;
  open(FNET_cell, "$cellFile") or die "Cannot open $cellFile";
  while(my $line_cell = <FNET_cell>)
  {
    chop($line_cell);
    if ($line_cell =~ /\s*${cell}\s*:\s*(\S+)\s*$/)
    {
      $numXtor = $1;
      return $numXtor;
    }
  }
  close FNET_cell;
}
sub CreateDictNumXtor()
{
  my $num_xtor_cell = 0;
  my $flag_cell = 0; # $flag_cell = 1 when inside the subckt
  my $num_xtor_inst = 0;
  my $inst;
  open(FNET_local, "$netlistFile") or die "Cannot open $netlistFile";
  while(my $line_net = <FNET_local>)
  {
    chop($line_net);
    if ($line_net =~ /\s*\.subckt\s+(\S+)\s+.*/)
    {
      $curr_cell = $1; $flag_cell = 1;
    }
    if ($line_net =~ /\s*\.ends\s+${curr_cell}.*/)
    {
      $num_xtor_cell = $num_xtor_cell + $num_xtor_inst;
      $num_xtor{$curr_cell} = $num_xtor_cell;
      $flag_cell = 0;
      $num_xtor_cell = 0;
      $num_xtor_inst = 0;
    }
    if (($flag_cell == 1) && ($line_net =~ /\s*xt[0-9].*fet.*/))
    {
      $num_xtor_cell++;
    }
      if (($flag_cell == 1) && ($line_net =~ /\s*xi[0-9].*\s+(\S+)\s*$/))
      {
        $inst = $1;
        $num_xtor_inst = $num_xtor_inst + $num_xtor{$inst};
      }
  }
  close FNET_local;
}
sub ParseArgs
{  local $arg;
   while (defined($arg = shift(@ARGV)))
   {
      if ($arg eq "-h")
      {  print("$USAGE\n");
         exit 1;
      }
      elsif ($arg eq "-netlistFile")
      {  $netlistFile= shift(@ARGV);
      }
```

```
        elsif ($arg eq "-cellFile")
        {  $cellFile= shift(@ARGV);
        }
      }
  }

################################################################
### Netlist_Subckt_spcParam_conv.pl  #####################
################################################################
## Usage text ##
$USAGE  = "\n";
$USAGE .= "Usage:Netlist_Subckt_spcParam_conv.pl -netlistFile <NETLISTFILE>";
$USAGE .= " -cellFile <CELLFILE>\n";

## Get switch information ##
ParseArgs();
my $in_subckt = 0; # Switch for denoting line with subckt
my $cell;
my $spaceParCount = 1;
my @newSpaceList = qw//;

open(FNET, "$netlistFile") or die "Cannot open $netlistFile";
while($line_net = <FNET>)
{
  chop($line_net);
  if ($in_subckt == 1) {
    # print "$cell\n";
    if ($line_net =~ /(\s*xm.*)\s+(s=SPACE)\s*$/) {
      my $exprBeforeSpace = $1; my $spaceExpr = $2;
      $spaceExpr =~ s/SPACE/spc${spaceParCount}/g;
      $spaceParCount++;
      print "$exprBeforeSpace $spaceExpr\n";
    } elsif ($line_net =~ /(\s*xi.*)\s+(\S+)\s+(spc1=.*)$/) {
      my $exprBeforeCurrCell = $1; my $currCell = $2; my $spaceList = $3;
      my $numXtor = GetNumXtorPerCell($currCell);
      my @spaceArray = split /\s+/, $spaceList;
      foreach (@spaceArray) {
        my $currSpace = $_;
        my $newcurrSpace = $currSpace;
        $newcurrSpace =~ s/SPACE/spc${spaceParCount}/g;
        push @newSpaceList, $newcurrSpace;
        # print "$newcurrSpace\n";
        $spaceParCount++;
      }
      my $exprAfterCurrCell = GetSpacelistInsideSubckt(@newSpaceList);
      print "$exprBeforeCurrCell $currCell $exprAfterCurrCell\n";
      @newSpaceList = qw//;
    } else {
      print "$line_net\n";
    }
  } else {
    print "$line_net\n";
  }
  if ($line_net =~ /\s*.subckt\s+(\S+).*$/)
  {
    $cell = $1;
    $in_subckt = 1;
  }
  if ($line_net =~ /\s*.ends\s+($cell).*$/)
  {
    $in_subckt = 0;
    $spaceParCount = 1;
  }
  # print "$in_subckt\n";
}
close FNET;

## Parse Args ##
sub GetSpacelistInsideSubckt()
{
  my @newSpaceList = @_;
  my $spaceExpr = '';
  # print "$newSpaceList[0] $newSpaceList[1]\n";
  my $spaceVar = 0;
  foreach (@newSpaceList) {
    $spaceExpr = $spaceExpr . $newSpaceList[$spaceVar] . ' ';
    $spaceVar++;
  }
  return $spaceExpr;
}
sub GetNumXtorPerCell()
{
  my $cell = $_[0]; my $numXtor;
  open(FNET_cell, "$cellFile") or die "Cannot open $cellFile";
  while(my $line_cell = <FNET_cell>)
  {
    chop($line_cell);
    if ($line_cell =~ /\s*${cell}\s*:\s*(\S+)\s*$/)
    {
      $numXtor = $1;
      return $numXtor;
    }
  }
  close FNET_cell;
}
sub ParseArgs
{  local $arg;
   while (defined($arg = shift(@ARGV)))
   {
      if ($arg eq "-h")
      {  print("$USAGE\n");
         exit 1;
      }
      elsif ($arg eq "-netlistFile")
      {  $netlistFile= shift(@ARGV);
      }
      elsif ($arg eq "-cellFile")
      {  $cellFile= shift(@ARGV);
      }
    }
}

################################################################
### Netlist_MC_spcParam_conv.pl  #####################
```

```
################################################################
## Usage text ##
$USAGE  = "\n";
$USAGE .= "Usage:Netlist_MC_spcParam_conv.pl -inFile <INFILE>\n";
$USAGE .= "\n";

## Get switch information ##
ParseArgs();
my $numspace = `grep -o SPACE ${inFile} | wc -l`;
print $numspace;
my $i_var = 1;
while ($i_var <= $numspace)
{
  $bash_cmd = "perl -0777 -p -i -e \"s/SPACE/s${i_var}/\"  ${inFile}";
  system(${bash_cmd});
  $i_var++;
}
## Parse Args ##
sub ParseArgs
{  local $arg;
   while (defined($arg = shift(@ARGV)))
   {
      if ($arg eq "-h")
      {  print("$USAGE\n");
         exit 1;
      }
      elsif ($arg eq "-inFile")
      {  $inFile= shift(@ARGV);
      }
    }
}

################################################################
### Netlist_MC_popenSeed_gen.pl  #####################
################################################################
## Usage text ##
$USAGE  = "\n";
$USAGE .= "Usage:Netlist_MC_popenSeed_gen.pl -inFile <INFILE> -spcDir <SPCDIR>";
```

```
$USAGE .= "-poFile <POFILE> -runDir <RUNDIR> -numseed <NUMSEED>\n";
$USAGE .= "\n";

## Get switch information ##
ParseArgs();
#my $numseed = 1;
my $fstLineNo = 1;
my $lstLineNo = ‘grep -o ’=s[0-9]\\+’ ${inFile} | wc -l‘;
$inFile =˜ /(\S+).txt/;
my $outFilePrefix = $1;
print "${outFilePrefix}\n";
open(FPO, "$poFile") or die "Cannot open $poFile";
while($line = <FPO>)
{
  chop($line);
  my $spaceFile = "${spcDir}/CNFET_MC_RandomSpace_100000_PCNTopen_${line}pct.txt";
  print "${spaceFile}\n";
  my $idx = 0;
  while ($idx < $numseed)
  {
    my $seedFile = "${runDir}/PCNTopen_${line}pct/netlists/${outFilePrefix}_seed${idx}
    system("cp $inFile $seedFile");
    my $u_var = ${fstLineNo} + ${idx} * ${lstLineNo};
    my $v_var = ${u_var} + ${lstLineNo} - 1;
    print "$u_var $v_var\n";
    @spc_list = ‘awk ’NR>=${u_var} && NR<=${v_var}’ $spaceFile‘;
    my $jj = 1;
    foreach $spc (@spc_list)
    {
      chop($spc);
      system("perl -p -i -e \"s/s${jj}([ˆ0-9])/${spc}\\1/g\" ${seedFile}");
      $jj++;
      # print "Space = $spc\n";
    }
    $idx++;
  }
}
close FPO;
```

```
## Parse Args ##
sub ParseArgs
{  local $arg;
   while (defined($arg = shift(@ARGV)))
   {
      if ($arg eq "-h")
      {  print("$USAGE\n");
         exit 1;
      }
      elsif ($arg eq "-inFile")
      {  $inFile= shift(@ARGV);
      }
      elsif ($arg eq "-spcDir")
      {  $spcDir= shift(@ARGV);
      }
      elsif ($arg eq "-poFile")
      {  $poFile= shift(@ARGV);
      }
      elsif ($arg eq "-runDir")
      {  $runDir= shift(@ARGV);
      }
      elsif ($arg eq "-numseed")
      {  $numseed= shift(@ARGV);
      }
    }
}
```

Appendix D

Publications from this Work

1. **Kaship Sheikh**, Lan Wei, "Reducing Impact of CNFET Process Imperfections on Shape of Activation Function by Using Connection Pruning and Approximate Neuron Circuit," *International Symposium on Quality Electronic Design, (ISQED)*, 2020 (accepted). (Chapter 5, 6)

2. **Kaship Sheikh**, Lan Wei, "Methodology to Generate Approximate Circuits to Reduce Process Induced Degradation in CNFET Based Circuits," *International Conference on Simulation of Semiconductor Processes and Devices 2018 (SISPAD)*, pp. 360-363, 2018. (Chapter 4)

3. **Kaship Sheikh**, Lan Wei, " Methodology to Capture Statistical Effect of Process Imperfections on Glitch Suppression in CNFET circuits and Counter using Approximate Circuits", *IEEE/ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 27-32, 2018. (Chapter 3, B)

4. **Kaship Sheikh**, Lan Wei, " Evaluation of Circuit Performance Degradation due to CNT Process Imperfection", *IEEE International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)*, 2018. (Chapter 3)

5. **Kaship Sheikh**, Lan Wei, " Using Approximate Circuits to Counter Process Imperfections in CNFET based Circuits", *IEEE International Symposium on VLSI Design, Automation, and Test (VLSI-DAT)*, pp. 1-4, 2018. (Chapter 3)

6. **Kaship Sheikh**, Shu-Jen. Han, and Lan Wei, "CNFET With Process Imperfection: Impact on Circuit-Level Yield and Device Optimization," *IEEE Transactions on Circuits and Systems I: Regular Papers, (TCAS-I)* vol. 63, no. 12, pp. 2209-2221, 2016. (Chapter 3, A)

7. **Kaship Sheikh**, Shu-jen Han, Lan Wei, "Impact of CNT Process Imperfection on Circuitlevel Functionality and Yield", *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 401-404, 2016.(Invited paper) (Chapter 3, A)