Exploring the Accuracy vs Energy Efficiency Trade-offs in Error-Aware Low Voltage DNN Accelerators

A Dissertation Presented

by

Mallika Rathore

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

August 2021

Stony Brook University

The Graduate School

Mallika Rathore

We the dissertation committee for the above candidate for the
Doctor of Philosophy degree,
hereby recommend acceptance of this dissertation.

Dr. Emre Salman - Advisor of Dissertation
Associate Professor, Department of Electrical and Computer Engineering

Dr. Milutin Stanaćević - Chairperson of Defense
Associate Professor, Department of Electrical and Computer Engineering

Dr. Peter Milder - Advisor of Dissertation
Associate Professor, Department of Electrical and Computer Engineering

Dr. Emre Tuncer - Defense Committee Member
Hardware Engineer, Google

This dissertation is accepted by the Graduate School

Eric Wertheimer
Dean of the Graduate School

Abstract of the Dissertation

Exploring the Accuracy vs Energy Efficiency Trade-offs in Error-Aware Low Voltage DNN Accelerators

by

Mallika Rathore

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2021

Energy efficiency is a critical design objective in deep learning hardware, particularly for real time machine learning applications where the processing takes place on resource-constrained platforms. Deep neural networks (DNNs) have an inherent fault tolerance which can potentially provide opportunities to design low power, error-aware deep learning hardware. Due to this error resilience of the applications, voltage scaling is an attractive method to enhance efficiency. The power savings obtained at low voltage operation are at the expense of timing errors which can impact the inference accuracy of the application (based on the error resilience). The primary objective of this work is to exploit the inherent resilience of DNNs and improve the energy efficiency by developing low-voltage, error-aware accelerators. To achieve this objective, a methodology is proposed to quantify the inference accuracy

and energy consumption as a function of operating voltage. The related trade-offs and design overheads analyzed with this framework can be used to push the boundaries on low voltage operation and facilitate the implementation of efficient error detection and correction techniques in accelerator structures.

The framework proposed in this work is implemented in three steps. First, an analytical timing error probability model is developed to determine the per-bit timing error probability for the data paths in the systolic array, without relying on time consuming hardware simulations. Second, the error resilience of a neural network is determined to analyze the impact of bit-level datapath timing errors in the compute-intensive convolutional and fully-connected layers in quantized neural networks, at per-layer and per-bit granularity. The results obtained during the first two steps are used to explore the per-layer voltage scaling and efficient error correction techniques for sensitive bits where the primary objective is to improve the energy efficiency with negligible impact on inference accuracy. It is observed that the network-level voltage scaling in a 128×128 systolic array implementing EfficientNet-B4 DNN can provide 67% improvement in energy efficiency at 700 MHz operating clock frequency with minimal impact on inference accuracy. The proposed analytical framework can further push the boundaries on low voltage operation through per-layer voltage scaling and bit-level error correction techniques,

while also providing an insight into the related accuracy-energy trade-offs and design overhead, therefore facilitating smarter energy optimization and error corrections techniques in low voltage DNN accelerators.

This work is dedicated to my parents,

Devendra Singh Rathore and Nivedita Rathore, for their

never-ending support and love,

and my husband, Michael, for being my rock.

Table of Contents

List of Figures **xiv**

List of Tables **xvii**

Acknowledgements **xix**

1 Introduction **1**

2 Background **6**
2.1 Energy Efficiency in Deep Neural Networks 6
2.1.1 Brief Introduction of DNN 7
2.1.2 Energy Efficiency in DNN hardware 9
2.2 Low Voltage Operation 12
2.2.1 Voltage Scaling in FinFET Technology 12
2.2.2 Low Voltage TFET Technology 14
2.3 Accuracy and Energy Efficiency 18

3 Error Probability Modeling Methodology - Simple Sequential Path **22**
3.1 Timing Probability Distribution 23
3.2 Error Probability Computation 24
3.3 Results 28
3.3.1 Error Probability for Different Clock Periods 28
3.3.2 Error Probability at Same *ClockPeriod-to-PathDelay* Ratio . 29
3.3.3 Error Probability at Same Clock Frequency 32
3.3.4 Error Probability at Same Voltage 33
3.4 Discussion 34

4 Error Probability Modeling Methodology - Multiple Timing Paths and Pipeline Stages **35**
4.1 Error Probability for Multiple Timing Paths . . . 36
4.2 Error Probability for Multiple Pipeline Stages . . . 39
4.3 Results . . . 41

5 Case Study - 8-bit MAC **45**
5.1 Circuit Description and Implementation . . . 46
5.2 FinFET vs. TFET technologies . . . 49
5.3 Results . . . 52
5.3.1 Dependence on Timing Slack . . . 52
5.3.2 Dependence on Supply Voltage . . . 54
5.3.3 Error Probability vs. Power Consumption . . . 57
5.4 Discussion . . . 57

6 DNN Error Resilience Analysis Framework **60**
6.1 Related Work . . . 61
6.2 Error Resilience Analysis Framework . . . 63
6.2.1 Inputs . . . 63
6.2.2 Model Setup . . . 65
6.3 Results . . . 66
6.3.1 Neural Networks . . . 68
6.3.2 Experimental Setup . . . 69
6.3.3 Error Resilience Analysis . . . 71
6.3.3.1 Network-level error resilience analysis . . . 72
6.3.3.2 Per-layer error resilience analysis . . . 76
6.3.3.3 Per-bit error resilience analysis . . . 79
6.4 Conclusion . . . 82

7 Quantifying Accuracy and Energy Efficiency Trade-offs in Systolic Arrays **84**
7.1 DNN to Systolic Array Hardware Mapping . . . 86
7.1.1 Per-Layer Runtime Evaluation . . . 88
7.1.2 Array Utilization Computation . . . 92
7.2 Per-Layer Energy Modeling . . . 93
7.2.1 Per-Layer Energy Consumption . . . 94
7.2.2 DNN Energy Consumption . . . 95
7.3 Network-Level Voltage Scaling . . . 96

7.3.1 Timing Error Probability for Systolic Array 96
7.3.2 Quantifying DNN accuracy with respect to supply voltage . 100
7.3.2.1 Quantifying DNN energy consumption with respect to supply voltage 102
7.3.2.2 Accuracy-Energy trade-off 106
7.4 Per-Layer Voltage Scaling 107
7.5 Discussion 112

8 Error Detection and Correction in Systolic Arrays 114
8.1 Background 115
8.2 Quantifying Accuracy with Bit-Level Error Correction 118
8.3 Bit-Level Error Detection and Correction using iRazor 119
8.3.1 DNN Accuracy with EDAC 124
8.3.2 Latency Overhead 127
8.3.2.1 MSB error correction overhead 128
8.3.2.2 Multiple higher order bits error correction overhead 129
8.3.2.3 Latency overhead results 130
8.3.3 Energy Overhead 134
8.4 Limitations to supply voltage scaling with EDAC 141
8.5 Discussion 144

9 Conclusion 152

Bibliography 157

List of Figures

1.1 Primary steps of the proposed methodology to better understand the interactions among voltage scaling, timing error probabilities, and inference accuracy in deep neural networks. 2

2.1 Simple neural network example. 7
2.2 Basic CNN architecture. The input layer represents the RGB feature maps of an input image. 9
2.3 Schematic view and energy band diagram of (a) FinFET, and (b) TFET depicting basic differences between carrier injection mechanisms, thermionic emission and band to band tunneling, in the respective technologies. 15
2.4 ON-state current improvement from homojuntion TFET to heterojunction TFET by reducing the tunnel barrier, E_{beff}. 16
2.5 Comparison of I_{ON} vs I_{ON}/I_{OFF} ratio for different operating points on the $I_D - V_G$ curve for 0.3V operating voltage. 17
2.6 $I_D - V_G$ characteristics of 20 nm Heterojunction TFET and Si FinFET at 0.3 V operating voltage. 18
2.7 CNN accuracy as a function of dynamic variations. 19

3.1 Simple sequential data path. 24
3.2 Voltage *pdf* and delay *pdf* of the sequential data path shown in Fig. 3.1 at two σ values of 10% and 30%: (a) 20 nm FinFET technology, (b) 20 nm heterojunction TFET technology. 25
3.3 Illustration of timing error probability of a data path shown in Fig. 3.1 and designed in 20nm HetJ TFET technology. The error probability is computed from (a) timing *pdf* by using (3.4) and (b) voltage *pdf* by using (3.5). The shaded regions in both graphs are equivalent. . . 27

3.4 Dependence of error probability on clock period and power supply noise for the circuit shown in Fig. 3.1: (a) 20nm FinFET technology, (b) 20nm HetJ TFET technology. 30
3.5 Error probability as a function of noise at respective nominal supply voltages (0.9V for FinFET and 0.3V for TFET). The *ClockPeriod-to-PathDelay* ratio considered is 2. 31
3.6 Error probability as a function of noise at respective nominal supply voltages (0.9V for FinFET and 0.3V for TFET). The clock frequency considered for both the technologies is 4GHz. 32
3.7 Error probability as a function of noise at 0.4V supply voltage and 2GHz clock frequency. 33

4.1 1-bit MAC unit to illustrate multiple paths within a pipeline stage and sequentially adjacent pipeline stages. 36
4.2 Example to illustrate the timing error probability calculation of multiple timing paths within a sequential circuit: (a) Delay *pdfs* and (b) voltage *pdf* of Path A and Path B in Fig. 4.1, designed using 20nm HetJ TFET technology. The operating voltage and clock period are 0.3V ($\sigma = 30\%$) and 400ps, respectively. 38
4.3 Timing error probability as a function of power supply noise, σ_{vdd}, for the output node of a 1-bit MAC unit in 20nm HetJ TFET technology with 0.3V operating voltage and 1.8GHz clock frequency. . . 40
4.4 Timing error probability as a function of number of sequentially adjacent paths for different power supply noise at 0.3V operating voltage for 20nm HetJ TFET technology. 41
4.5 Schematic of a 2-bit multiplier with input and output pipeline stages. 41
4.6 Timing error probabilities as a function of power supply noise, σ_{VDD}, for the four inputs to the multiplier (Fig. 4.5) designed using (a) 20nm HP FinFET, and (b) 20nm HetJ TFET technologies. The respective operating voltages considered are 0.9V and 0.3V, and the clock frequency is 2.5GHz. 43
4.7 Timing error probabilities as a function of power supply noise, σ_{VDD}, for MSB output of the multiplier ($out3$). 44

5.1 Block diagram of 8-bit MAC used in the case study to characterize the trade-offs among voltage, error probability, and power consumption. 46
5.2 Detailed block diagram of (a) 4-bit multiplier, and (b) 8-bit multiplier circuits. 47

5.3 Timing error probability of each output bit of 8-bit MAC unit as a function of power supply noise, σ_{vdd}/μ_{vdd}: (a) 20nm HP FinFET technology with 0.9V operating voltage and 3.06GHz clock frequency and (b) 20nm HomJ TFET technology with 0.3V operating voltage and 1.81GHz clock frequency. Note that clock frequencies are chosen to ensure that *ClockPeriod-to-PathDelay* is the same for both cases. 48
5.4 Dependence of delay on voltage of the highest-delay path in 8-bit MAC unit for (a) 20nm HP FinFET, (b) 20nm LP FinFET, (c) 20nm HomJ TFET, and (d) 20nm HetJ TFET technologies. 50
5.5 Voltage *pdf* and the error probability of the highest-delay path in 8-bit MAC unit for (a) 20nm FinFET and (b) 20nm HetJ TFET technologies. The mean of the *pdf* corresponds to the nominal operating voltage and the σ is 10%. 51
5.6 Dependence of timing error probability on *ClockPeriod-to-PathDelay ratio* for the MSB of 8-bit MAC unit at 10% power supply noise at nominal operating voltages: (a) 20nm FinFET and (b) 20nm TFET technologies. Note that the clock frequencies corresponding to each *ClockPeriod-to-PathDelay ratio* are listed in Table 5.1. 53
5.7 Timing error probability vs. power consumption tradeoff at 5% power supply noise and a *ClockPeriod-to-PathDelay* ratio of 1.5: (a) 20nm HP FinFET, (b) 20nm LP FinFET, (c) 20nm HetJ TFET, and (d) 20nm HomJ TFET technology. 58

6.1 DNN error resilience analysis framework methodology. 64
6.2 Inference accuracy of different neural networks, considering ImageNet dataset. 71
6.3 Distribution of Top-1 inference accuracy obtained over 100 runs with random error injection at 0.001%. 73
6.4 Top-1 classification accuracy with respect to different error rates for (a) ResNet-18, MobileNetV2, and (b) EfficientNet networks. Note that EfficientNet-B0 and B2 networks are quantized to 12 bits, while the other networks are quantized to 10 bits. 74
6.5 $Err_{1\%}$ and $Err_{0.5\%}$ of different quantized, pre-trained neural networks. 75
6.6 Per-bit error resilience analysis for EfficientNet-B0 and B4 networks. It should be noted that for no errors in MSBs, the remaining bits have the same error rate. 81

7.1 Methodology to quantify the trade-off between DNN accuracy and energy consumption at different operating voltages. 85
7.2 Systolic array block diagram. 86
7.3 Steps to compute runtime for dataflows in systolic array based on. . 90
7.4 Basic block diagram of a processing element in a systolic array. . . 97
7.5 Top-1 classification accuracy of EfficientNet-B4 with respect to supply voltage scaling for 700 MHz and 1 GHz clock frequency (2.85 and 2 *ClockPeriod-to-PathDelay* ratio respectively). 101
7.6 Dynamic, leakage and total power and energy consumption of 256×256 systolic array implementing WS and OS dataflows at 700 MHz for EfficientNet-B4 neural network. 102
7.7 Dynamic and Leakage power consumption of systolic arrays of size 256×256, 128×128, 64×64, and 16×16, while considering 700 MHz clock frequency. Both the WS and OS dataflows are compared. 104
7.8 Total energy consumption of systolic arrays of size 256×256, 128×128, 64×64, and 16×16. Both the WS and OS dataflows are compared. . 104
7.9 EfficientNet-B4 Top-1 classification accuracy and percentage decrease in total energy consumption for systolic array implementing WS dataflow. The baseline Top-1 accuracy at 0.9V is 82.4%. 105
7.10 EfficientNet-B8 Top-1 classification accuracy and percentage decrease in total energy consumption for systolic array implementing WS dataflow. The baseline Top-1 accuracy at 0.9V is 85.59%. . . . 106
7.11 MobileNetV2 Top-1 classification accuracy and percentage decrease in total energy consumption of different systolic array sizes implementing WS dataflow. 108
7.12 MobileNetV2 per-layer *ranking_metric*. 109

8.1 Top-1 classification accuracy of EfficientNet-B4 with 1–3 higher order bits error-free for 0.55V operating voltage and 700 MHz clock frequency. Note that all the layers of the network are scaled to 0.55V. 118
8.2 Schematic of iRazor flip-flop. 120
8.3 Waveforms illustrating the iRazor flip-flop error detection functionality. 120
8.4 Basic schematic for error detection and correction based on. 122
8.5 Waveforms showing iRazor error detection and correction functionality. 122

8.6 EfficientNet-B4 and B8, and MobileNetV2 Top-1 classification accuracy with bit-level EDAC implementation when considering 0.55V and 0.65V for 700 MHz and 1 GHz clock frequencies, respectively. 125
8.7 EfficientNet-B4 inference accuracy improvement with per-bit error correction at 0.5V and 0.6V for 700 MHz and 1 GHz, respectively. Note that the highest MSB position for all the layers in this DNN is 35. 126
8.8 Energy consumption of MobileNetV2 DNN with and without error detection for different systolic array sizes considering WS and OS dataflow at 1 GHz clock frequency. 136
8.9 Energy consumption of EfficientNet-B4 DNN with and without error detection for different systolic array sizes considering WS and OS dataflow at 1 GHz clock frequency. 137
8.10 Energy consumption of EfficientNet-B8 DNN with and without error detection for different systolic array sizes considering WS and OS dataflow at 1 GHz clock frequency. 138

List of Tables

2.1 Noise Margin of 20nm TFET, FinFET and CMOS at Nominal and Scaled Supply Voltages. 20

3.1 Operating clock frequencies for different power supply noise in simple sequential path, designed using 20nm HP FinFET and 20nm HetJ TFET technologies, to obtain error probability ≈2%. Nominal supply voltage is used for both the technologies. 29
3.2 Timing characteristics for error probability computation at respective nominal supply voltages for 20nm HP FinFET and 20nm HetJ TFET technologies. The *ClockPeriod-to-PathDelay* ratio considered for both the technologies is 2. 31

5.1 Operating clock frequencies for FinFET and TFET technologies to obtain different *ClockPeriod-to-PathDelay* ratios. Nominal supply voltage is used for both technologies 54
5.2 Timing error probability for the MSB of 8-bit MAC unit designed in 20nm FinFET technology for different supply voltages and *ClockPeriod-to-PathDelay* ratios. Power supply noise is constant at 5%. 56
5.3 Timing error probability for the MSB of 8-bit MAC unit designed in 20nm TFET technology for different supply voltages and *ClockPeriod-to-PathDelay* ratios. Power supply noise is constant at 5%. 56

6.1 Impact of quantization on classification accuracy. 70
6.2 ResNet-18 per-layer error analysis 77
6.3 MobileNetV2 per-layer error analysis 78
6.4 $Err_{1\%}(\%)$ of different layers in EfficientNet networks. 80
6.5 $Err_{0.5\%}(\%)$ of different layers in EfficientNet networks. 80

7.1 Spatial-Temporal allocation of DNN dimensions. Note that N_{ofmap} is the number of OFMAP (output feature map) pixels generated by the filter, N_{filter} is the number of convolution filters and W_{conv} is the number of partial sums generated per output pixels. 89
7.2 Dynamic and leakage power consumption for a processing element at different voltages for 700 MHz clock frequency. 94
7.3 *ClockPeriod-to-PathDelay* ratio of different accumulator sizes. . . . 98
7.4 Timing error probability of MSB for different adder bits across all layers in EfficientNet-B8 with respect to different operating voltages at 700 MHz clock frequency. 99
7.5 Timing error probability of MSB for different adder bits across all layers in EfficientNet-B8 with respect to different operating voltages at 1 GHz clock frequency. 100
7.6 Total cycle count and maximum utilization of EfficientNet-B4 for different systolic array size. 103
7.7 Operating voltage scaling for six convolutional layers with the highest *ranking_metric* in 5 different scenarios (S1–S5). 111
7.8 MobileNetV2 Top-1 classification accuracy and percentage decrease in total energy consumption in different scenarios described in Table 7.7. The percentage decrease in energy is with respect to the energy consumption at 0.9V. 111

8.1 Comparison of the voltage at which the critical data path (of the respective bit positions) fails timing for D flip-flop and iRazor latch. 123
8.2 Top-1 accuracy and maximum timing error probability (p_{max}) for MobileNetV2, EfficientNet-B4 and B8 networks at different voltages and frequencies. 126
8.3 Latency overhead (lat_{over}) for MobileNetV2 DNN comparing different systolic array sizes with EDAC implemented at 0.55V, 700 MHz and 0.65V, 1 GHz. 130
8.4 Latency overhead (lat_{over}) for EfficientNet-B4 DNN comparing different systolic array sizes with EDAC implemented at 0.55V, 700 MHz and 0.65V, 1 GHz. 131
8.5 Latency overhead (lat_{over}) for EfficientNet-B8 DNN comparing different systolic array sizes with EDAC implemented at 0.55V, 700 MHz and 0.65V, 1 GHz. 131

8.6 Latency overhead for EfficientNet-B4 at 0.5V, 700 MHz and 0.6V, 1 GHz. The accuracy obtained with this EDAC implementation is 81.7% which is within 1% of the baseline accuracy. 133
8.7 Energy consumption of MobileNetV2 at 0.65V and 1 GHz frequency, with and without EDAC. 139
8.8 Energy consumption of EfficientNet-B4 at 0.65V and 0.6V, and 1 GHz frequency, with and without EDAC. 140
8.9 Energy consumption of EfficientNet-B8 at 0.65V and 1 GHz frequency, with and without EDAC. 141
8.10 MobileNet-V2 latency results summary for different voltage, frequency, systolic array sizes, and dataflows. 146
8.11 MobileNet-V2 energy results summary for different voltage, frequency, systolic array sizes, and dataflows. 147
8.12 EfficientNet-B4 latency results summary for different voltage, frequency, systolic array sizes, and dataflows. 148
8.13 EfficientNet-B4 energy results summary for different voltage, frequency, systolic array sizes, and dataflows. 149
8.14 EfficientNet-B8 latency results summary for different voltage, frequency, systolic array sizes, and dataflows. 150
8.15 EfficientNet-B8 energy results summary for different voltage, frequency, systolic array sizes, and dataflows. 151

ACKNOWLEDGEMENTS

I would like to take this opporunity to thank everyone who have helped me through this incredible journey as a PhD student in Stony Brook. First and foremost, I express my sincere thanks and gratitude to my advisors, Dr. Emre Salman and Dr. Peter Milder for their tremendous support and guidance throughout. They have been extremely patient with me and provided incredible ideas and direction as I stumbled through my research during the last five years. I am very grateful for all their valuable advice and feedback which have shaped this research, and which will eventually shape my career post graduation. I also thank Dr. Milutin Stanaćević and Dr. Emre Tuncer for taking out the time to be a part of my defense committee and for your valuable comments and feedback which gave an important direction to this work.

Special thank you to my friends in NanoCAS laboratory: Manav, Ivan, and Tutu, for being my sounding board when I had to offload ideas and for creating so many fun and wonderful memories with me during the last five years. I am thankful for your friendship, and wish you all great success in future.

I also want to thank my husband, Michael, for being there with me every step of the way. This would not have been possible without his support. Last but not

the least, I extend my thanks to my parents for always believing in me and for their selfless love and support. I dedicate this thesis to them.

Thank you.

Chapter 1

Introduction

Deep learning hardware such as accelerators for deep neural networks (DNNs) include tens of thousands of multipliers and adders, and can perform highly parallelized and high-performance computing. For instance, a specialized accelerator hardware called tensor processing unit (TPU) is used to accelerate the inference of DNNs and is deployed in Google's datacenters for complex neural network applications [1]. However, due to increasing computational complexity, the recent research works are focused on improving the energy efficiency without compromising on the performance and robustness of the design. Therefore, energy efficiency has become an important design consideration, particularly with escalating power and temperature/cooling issues in deeply scaled technologies.

Voltage scaling is an effective and well-known approach to enhance efficiency. However, the increasing susceptibility to supply voltage variations at low voltages has resulted in timing errors and degraded robustness. Moreover, higher leakage power consumption in scaled technologies has limited the supply voltage scaling. The recent works mitigate this limitation through novel design methodologies, and

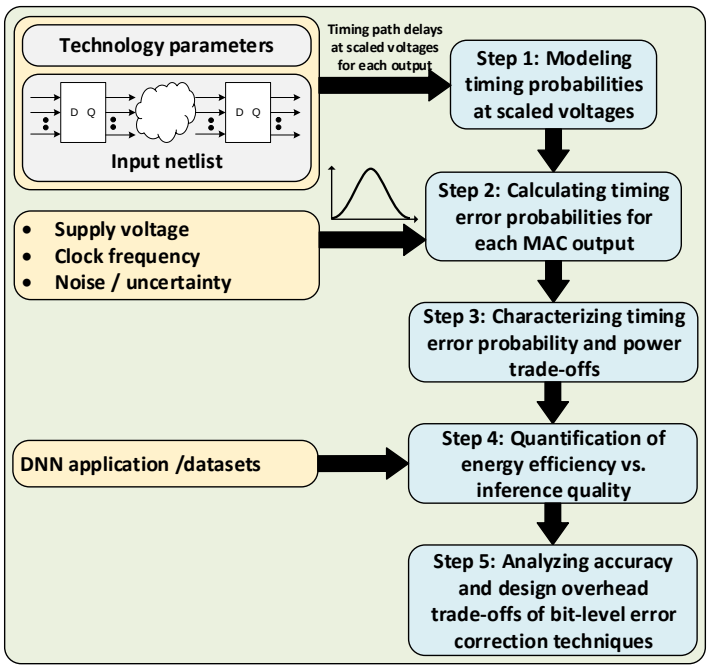


Figure 1.1: Primary steps of the proposed methodology to better understand the interactions among voltage scaling, timing error probabilities, and inference accuracy in deep neural networks.

other device and technology innovations. Deep learning applications have a unique characteristic attributing to the inherent resilience to errors [2–5]. Therefore, despite the strong trade-offs among accuracy, performance and power, the impact of voltage scaling on these parameters is particularly worth investigating for deep learning applications.

The primary objective of this work is to exploit the inherent resilience of DNNs and improve the energy efficiency by developing low-voltage, error-aware accelerators. This objective is achieved by the proposed framework, which quantifies the inference accuracy and energy consumption as a function of operating voltage, and

also analyzes the related trade-offs and design overheads of implementing efficient error detection and correction techniques in accelerator structures. The framework is implemented by first modeling the timing probabilities at scaled voltages for a given circuit netlist, technology parameters, operating frequency, and supply voltage (step 1 in Fig. 1.1). In step 2, timing error probabilities are calculated for each output of the neural network layer at a certain level of noise/uncertainty. These error probabilities enable us to quantify the trade-off between error probability and power consumption for voltage-scaled systolic arrays (basic computational unit in DNN accelerators) at given supply noise and operating frequency, as shown in step 3. These results are then used to quantify the trade-off between energy efficiency (obtained via voltage scaling) and inference accuracy for specific deep learning applications (step 4). Finally, the results obtained during the previous steps are used to explore the per-layer voltage scaling and efficient error correction techniques for sensitive bits (step 5) where the primary objective is to improve the energy efficiency with negligible impact on inference accuracy.

The proposed timing error modeling methodology, implemented in step 1, is analyzed for an 8-bit multiply accumulate unit implemented with both advanced FinFET and emerging tunneling field-effect transistor (TFET) technologies to evaluate the dependence of the results on technology [6]. Note that TFET devices have been proposed for ultra low voltage operation (0.1 to 0.5 V), facilitated by lower subthreshold swings where leakage current is significantly reduced as compared to modern FinFET technologies [7]. Thus, TFETs can be an interesting alternative for building ultra low voltage and highly parallel deep learning hardware. For DNN error resilience analysis, FinFET technology is used to analyze the per-bit output probability in systolic arrays. A PyTorch-based framework is implemented

to evaluate the DNN error resilience with respect to error rates injected in a given network. These bit-level errors can be injected (with per-layer and per-bit granularity) at a uniform error rate or the per-bit timing error rates obtained from the proposed probability model can be used to understand the impact of these hardware errors on the inference accuracy of the network. Existing error detection and correction techniques (such as iRazor [8]) are also explored to analyze the trade-offs between accuracy improvements and additional design overhead of implementing these techniques to correct bit-level errors for the sensitive bits in the network. Furthermore, the results and observations presented in this thesis can be leveraged to push the boundaries on low voltage operation and design energy efficient, error aware DNN accelerators.

The rest of the dissertation is organized as follows. Chapter 2 provides a background on the recent works exploring the energy efficiency in deep neural network hardware, low voltage operation in existing and emerging semiconductor device technologies, and the impact of voltage scaling on the energy efficiency and robustness of the hardware. Chapter 3 explains the formulation of the timing probability distribution from voltage distribution, which further depends on the voltage-delay characteristics. It also provides a proof-of-concept for the proposed error probability modeling methodology by considering the example of a simple timing path. Chapter 4 describes an extension of the proposed methodology to determine the timing error probability for circuits with multiple timing paths within one or multiple pipeline stages. Considering the example of three pipeline stage 8-bit MAC circuit, a case study is presented in chapter 5 to quantify the timing error probability results in different scenarios. Chapter 6 discusses the error resilience of DNNs at network-level, per-layer and per-bit granularity, analyzed through a framework

by injecting bit-level errors at uniform error rate in a quantized neural network. In chapter 7, the per-bit timing error probabilities obtained from the proposed error probability model for a processing element in a systolic array are used to quantify the DNN accuracy as a function of supply voltage. The trade-off between accuracy and energy is also quantified and discussed in the chapter. The energy and latency overhead of implementing bit-level error detection and correction techniques and the corresponding improvement in DNN inference accuracy are quantified in chapter 8. Finally, the work is concluded in chapter 9 with a discussion on future directions facilitated by the results obtained in this research.

Chapter 2

Background

It is challenging to obtain significant power reduction while maintaining performance, particularly in compute-intensive hardware blocks such as accelerators for deep neural networks (DNNs). As the operating voltage approaches the near- and sub-threshold regions, the susceptibility to noise increases, resulting in potential timing errors which can lead to design failures. The modeling methodology proposed in this work analyzes the power consumption and timing accuracy trade-offs in low voltage operation. Researchers have focused on improving the energy efficiency, especially in deep learning hardware, by exploring architectural, design, or device and technology innovations. This chapter provides a background for energy efficient neural network architectures, voltage scaling paradigm, and related issues.

2.1 Energy Efficiency in Deep Neural Networks

Deep neural networks (DNN) have revolutionized the artificial intelligence (AI) implementations with significant breakthroughs in a number of applications such

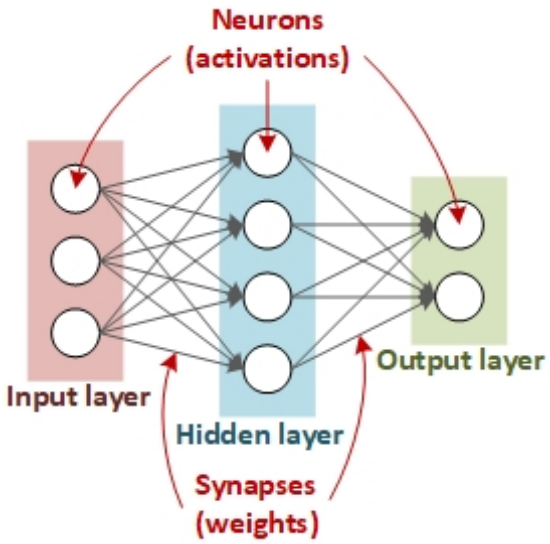


Figure 2.1: Simple neural network example [9].

as image classification, speech recognition and natural language processing. The massive parallelism and computational complexity of DNNs led to their hardware implementation in GPU, FPGA and custom ASICs [1]. This section provides an introduction to DNNs and discusses the recent works in improving the energy efficiency and performance of deep learning hardware.

2.1.1 Brief Introduction of DNN

Neural network is a brain-inspired computing paradigm which involves implementation of a non-linear function to a weighted summation of the input values. A computational neural network consists of an input and output layer, and a number of intermediate layers called “hidden layers”, which propagate the weighted sums to the output layer, as shown in Fig. 2.1 [9]. DNNs are the neural networks used in deep learning and can have tens of hidden layers. For example, DeepMind

AlphaGo [10], used as benchmark in [1], has 89 layers, and ResNet model, used for image classification, can have a maximum of 152 layers [11]. This makes the neural networks capable of learning more complex and abstract features, therefore, improving the classification accuracy.

The four basic stages of pattern recognition include - acquisition, preprocessing, feature extraction, and classification [12]. Acquisition generates raw inputs; preprocessing reduces the noise and performs geometric corrections; feature extraction identifies the attributes which differentiate between different classes of patterns; and classification assigns the input to one of the predefined classes. Deep convolutional neural networks (CNNs) automate the feature extraction by using a large training database (training sets) to learn the input features. CNN architecture consists of multiple layers of convolution and pooling, with or without an activation function. Fully connected layers (FCNs) take the output of the CNN to map a set of 2-D features into a class. The final output consists of prediction values for different objects (or classes) where the input image can be classified. Fig. 2.2 shows the basic architecture of a CNN [13].

In conclusion, the fundamental computation strategy in a neural network comprises of training (learning) and inference (prediction). Training a DNN involves using a large sample database to learn the features and determine the input weights, which are later used during inference for accurate classification. The basic operation in each layer is multiplication and accumulation. Some CNN architectures can have a large number of parameters which increases the complexity of the neural networks. For instance, VGGNET [14] has 138 million parameters (most of which are contributed by the fully connected layers), and GoogLeNet [15] (because of the implementation of smaller convolutions) has 4 million parameters [16]. Therefore,

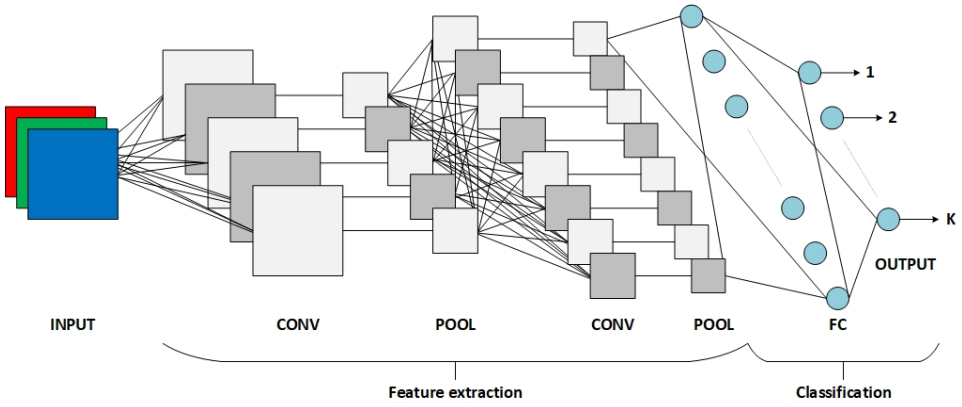


Figure 2.2: Basic CNN architecture [13]. The input layer represents the RGB feature maps of an input image.

the hardware implementation of DNNs consists of tens of thousands of multipliers and adders [1]. However, high computational complexity of the hardware for deep neural networks increases the power consumption, which is further exacerbated in scaled technologies, resulting in performance and accuracy limitations. Recent works in deep learning hardware have presented many optimization strategies to overcome these limitations, as summarized in the following section.

2.1.2 Energy Efficiency in DNN hardware

Due to high computational complexity, energy efficiency is a primary concern in the implementation of deep learning hardware. Prior work on improving the energy efficiency in deep learning hardware focuses on optimization at the architecture level, on the network structure, or of the numerical datatype used. At the architecture level, researchers have designed programmable accelerators (e.g. TPU [1], Eyeriss [17], DLAU [18]) and design/optimization tools (e.g. [19, 20]) to produce computational structures that are efficient for classes of neural networks

or optimized for specific networks. For instance, Google's custom ASIC, tensor processing unit (TPU) [1], implements a deterministic model which provides better match to the 99th-percentile response time requirement as compared to CPUs and GPUs. The lack of various microarchitectural features, which are implemented in CPUs and GPUs to improve the average throughput, results in comparatively lower power consumption and faster response time in TPUs. Eyeriss accelerator proposed in [17] provides an improvement in energy efficiency and throughput by using a proposed processing dataflow, which minimizes the expensive data movement through maximum data reuse (locally). An FPGA-based scalable deep learning accelerator unit (DLAU) is proposed in [18] which implements three pipelines processing units to improve throughput. Tile techniques, FIFO buffers, and pipelines are used to reuse computing units and minimize memory transfer operations. FPGA-based CNN acceleration is an interesting approach due to their programmable, parallel and power-efficient computing design [20]. However, due to limited resources and performance, various design frameworks are proposed to improve the performance and efficiency of FPGA-based accelerators. For instance, DNNWEAVER, a framework proposed in [19], automatically generates a synthesizable accelerator for a given (DNN, FPGA) pair. The generated accelerators provide better efficiency and performance as compared to multi-core CPUs and GPUs. Shen et al. propose a novel CNN accelerator design in [20] which partitions the FPGA resources into multiple processors, each of which operate on multiple images concurrently. This increases the computational efficiency and improves the overall throughput.

On the network level, researchers aim to simplify networks using techniques such as pruning [21], which can reduce the amount of weights and computation. In this technique, the network is first trained to identify the important connections.

The unimportant connections are then pruned before the network is trained again to fine tune the weights of the remaining connections. This results in a drastic reduction in the number of parameters (9× for AlexNet [22] and 13× for VGG-16 [14]) without any accuracy loss. The reduced number of parameters leads to smaller memory capacity and bandwidth requirements, making it suitable for implementation in mobile systems.

Lastly, simplifications to the numerical datatype by deeply quantizing learned parameters (e.g., [23, 24]) can be exploited in the hardware via simplified arithmetic and reduced data movement, thereby improving energy efficiency and performance. For instance, it is observed in [23] that fixed-point instructions provide a 3× improvement over an optimized floating-point baseline. Binarized neural networks (BNNs), introduced in [24], are DNNs with binary weights and activations at run-time and when computing the parameter gradients at train-time. BNNs replace the arithmetic operations with bit-wise operations, resulting in improved power-efficiency and reduced memory size.

Other approaches focus on data-sparsity in weights [25] or activations [26], or to compress the entire network to reduce cost [21]. Network quantization and weight sharing in [25] compresses the network by reducing the number of bits required to represent each weight. Through *Cnvlutin* [26], the computation is simplified by eliminating the ineffectual operations (e.g. multiplication with zero). This improves the performance and energy over the state-of-the-art accelerators with no accuracy loss.

2.2 Low Voltage Operation

Increasing power density on chip has made energy efficiency a key design objective, particularly in compute-intensive hardware. Voltage scaling is an effective approach to reduce the power consumption due to quadratic dependence of dynamic power on operating voltage. However, scaling supply voltage requires threshold voltage scaling to maintain sufficient on-current and performance. Threshold voltage scaling has posed additional challenges due to an exponential increase in off-current and therefore, the leakage power consumption [27]. With increasing demand for energy efficient designs during the past decade, researchers have focused on overcoming the voltage scaling limitations through novel design methodologies and devices.

2.2.1 Voltage Scaling in FinFET Technology

When the planar technology reached the limitations of gate length scaling due to increasing short channel effects, devices with enhanced gate control such as FinFETs mitigated this issue while providing an improvement in device performance as the technology continued to scale down to sub-32nm nodes [28]. Despite the superiority of FinFETs in suppressing the short channel effects, the fundamental limit on subthreshold swing limits the scaling of threshold and nominal supply voltages. Some of the recent works on low voltage design include novel design methodologies and topologies to enable low voltage operation, and near-threshold voltage scalability. Sitik et al. propose a design methodology for clock distribution networks with low-swing clocks, novel flip-flop topology, and clock buffers designed using 20 nm FinFET technology, demonstrating significant power savings as com-

pared to the full-swing implementation [29–31]. However, as the technology scales down to near- and sub-threshold operating regions, the significant power savings achieved are at the cost of highly degraded performance and robustness. Recent works in heterogenous architectures, near-threshold-voltage designs [32–34] and approximate computing [3, 35, 36] have facilitated ultra-low voltage operations for deeply scaled FinFET technologies.

Approximate computing is a promising approach to satisfy rising performance requirements while improving the energy efficiency by permitting certain amount of error in the results. This is achieved by selective approximation of computation based on the inherent resilience of applications. Operating at scaled supply voltages is one approach to approximate computing. In [36], Chippa et al. obtain $10\times$ energy-delay improvements at the cost of 5% loss in accuracy as compared to the conventional, fully accurate design. Despite the significant improvement, approximate computing approach requires careful design and approximation strategy to avoid unacceptable loss of accuracy [35]. Moreover, it is important to address the challenges and improve the scope of the programming frameworks for this approach before it can be integrated with the existing hardware design infrastructure.

Near-threshold computing relies on the operating region where the supply voltage is scaled close to the threshold voltage of the transistor ($VDD \sim V_{th}$). FinFET performance at near-threshold voltages is discussed in [33, 34]. It was observed that even though the performance is limited by increased latency, FinFETs exhibit better energy gains compared to planar technologies at such low voltages. Cui et al. analyze the impacts of gate-length biasing on circuit speed and leakage power consumption for 7 nm deeply-scaled FinFET devices through a device-circuit framework [37]. A 70% reduction in leakage power is observed at the cost of reduced

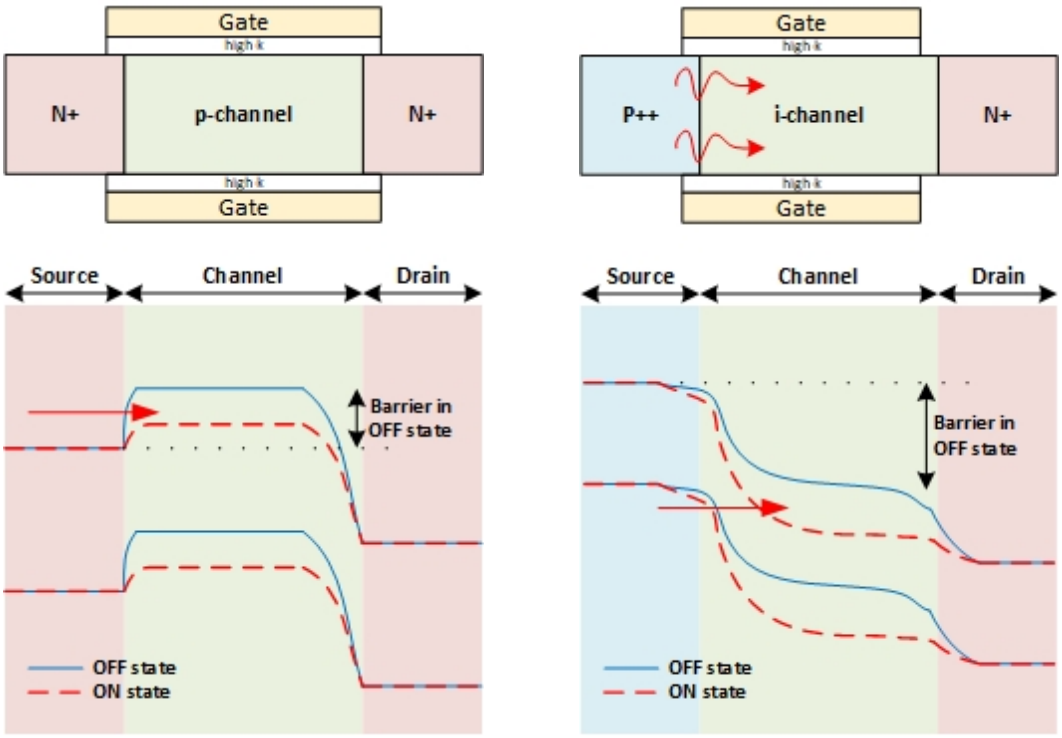
speed and increased dynamic energy consumption in near-V_{th} regimes.

Therefore, FinFETs can operate at near-threshold voltage and provide significant improvement in short channel effects and power savings in sub-20nm technology nodes as compared to the more conventional CMOS technologies. It is, however, important to consider the correlation between V_{th} and subthreshold slope to minimize the impact of process variations on energy efficiency and performance [38].

2.2.2 Low Voltage TFET Technology

In an effort to resolve the increasing leakage power due to the fundamental subthreshold slope limitation in FinFETs, various novel devices and materials, such as carbon nanotube FETs, nanowire FETs, III-V channel replacement devices, and tunnel field-effect transistors, are being explored and have shown promising results in near- and sub-V_{th} operation [39–41].

Tunnel field effect transistor (TFET) is designed as a reverse-biased *p-i-n* diode with asymmetrical source/drain doping. These devices use band-to-band tunneling (at the source-channel junction) as a carrier injection mechanism as opposed to thermal carrier injection in FinFETs. Fig. 2.3 [42,43] shows the difference between FinFET and TFET technologies with an energy-band diagram exhibiting the different carrier injection mechanisms. As shown in this figure, the carriers are injected thermionically over the barrier in FinFETs. Alternatively, for TFET technology, the alignment of the conduction band of the channel and the valence band of source enables the tunneling of carries at the source-channel interface. This characteristic enables the sub-60 mV/decade switching slope at room temperature, permitting aggressive voltage scaling (as low as 100 mV) [44]. However, due to the tunneling barrier at the source-channel interface, the tunneling probability is limited which



(a) FinFET (b) TFET

Figure 2.3: Schematic view and energy band diagram of (a) FinFET, and (b) TFET depicting basic differences between carrier injection mechanisms, thermionic emission and band to band tunneling, in the respective technologies.

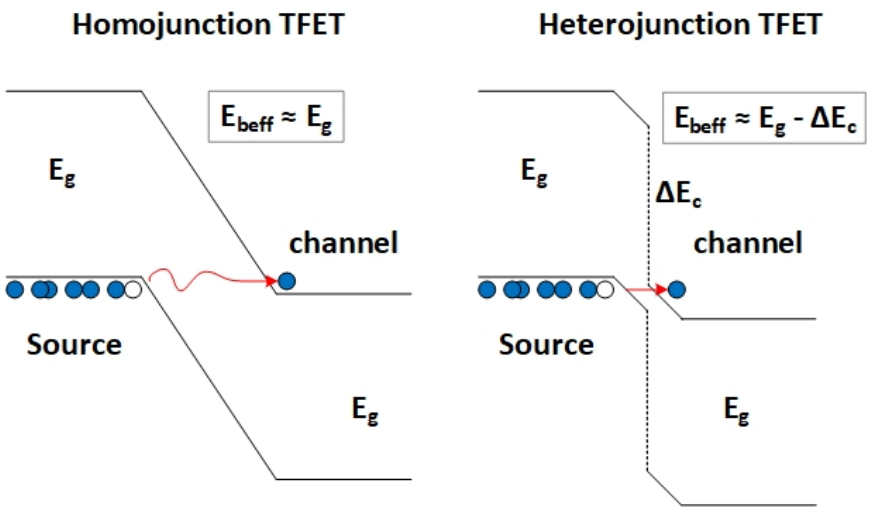


Figure 2.4: ON-state current improvement from homojuntion TFET to heterojunction TFET by reducing the tunnel barrier, E_{beff} [44].

limits the drive current when the device is on, particularly for large band-gap transistors.

Various semiconductor devices with lower band-gap have been explored and fabricated to improve the energy efficiency and performance of TFET-based designs, making it comparable to FinFETs at high frequency operation [7, 45–49]. For instance, III-V semiconductors enable hetero- band-gap alignment which improves the tunneling probability at low voltages. The improvement in the tunneling current in heterojunction TFET (as compared to homojunction TFET) is illustrated in Fig. 2.4 [44]. Saripalli et al. [50] compare the I_{on} vs I_{on}/I_{off} characteristics of 22 nm homojunction and heterojunction TFET at 0.3 V (Fig. 2.5), exhibiting an improved drive current and I_{on}/I_{off} due to sub-60 mV/dec subthreshold slope for the latter.

Comparing the $I_{ds} - V_{gs}$ characteristics of 20 nm heterojunction TFET and Si FinFET at 0.3 V (Fig. 2.6) [44], it was observed that TFET provides a steeper sub-

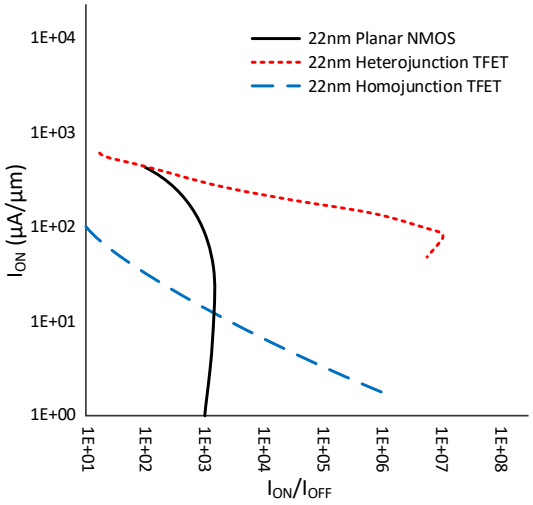


Figure 2.5: Comparison of I_{ON} vs I_{ON}/I_{OFF} ratio for different operating points on the $I_D - V_G$ curve for 0.3V operating voltage [50].

threshold slope and 7 times improvement in the drive current compared to FinFET, making it preferable for ultra-low voltage operation.

Taking into account the sub-60 mV/dec subthreshold slope and improved energy efficiency at low voltages obtained with III-V heterojunction TFETs, extensive voltage scaling can be achieved with significant power savings at ultra-low voltages while maintaining performance. Therefore, TFET technology is a promising candidate for highly parallelized applications where throughput can be maintained due to massive parallelism and power consumption can be reduced due to highly scaled voltages [51].

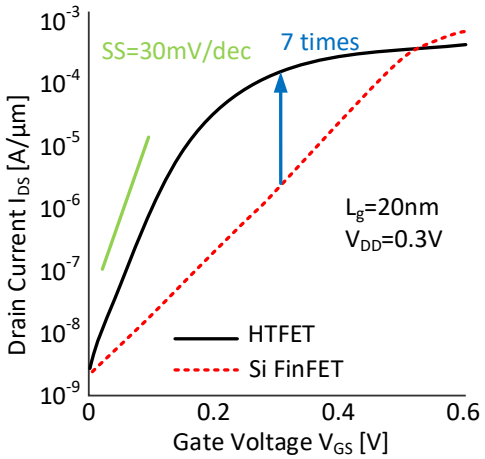


Figure 2.6: $I_D - V_G$ characteristics of 20 nm Heterojunction TFET and Si FinFET at 0.3 V operating voltage [44].

2.3 Accuracy and Energy Efficiency

Taking advantage of the inherent error resilience of deep learning applications [2–5], a number of timing error based methodologies have been explored recently to analyze the impact of voltage scaling on the inference accuracy. Timing speculation and voltage underscaling are related approaches [52, 53] which aim at running the chip at a lower voltage, therefore obtaining significant power savings at the cost of increased delay and timing errors in the design. These errors are then allowed to propagate or are corrected based on the error resilience of the application. For example, in [52], Zhang et al. propose a framework that enables aggressive voltage underscaling of DNN accelerators, demonstrating significant reduction in power consumption at no performance cost and less than 1% accuracy cost. A new timing error recovery technique is proposed which deals with the timing errors in MAC

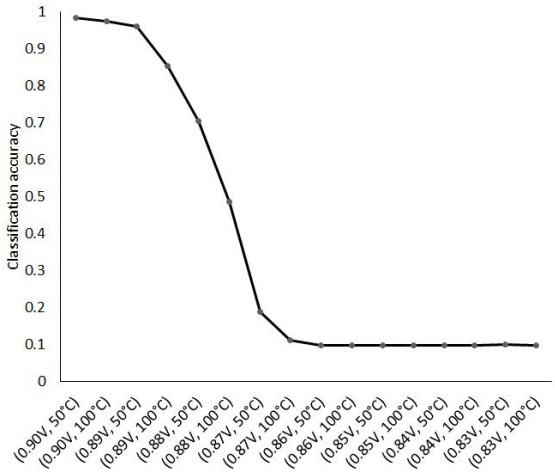


Figure 2.7: CNN accuracy as a function of dynamic variations [54].

units without re-executing erroneous MAC operations. Dynamic per-layer voltage underscaling is also proposed due to variations in timing error rate across layers in DNNs. Through these techniques, significant energy savings ($34\% - 57\%$) are achieved on state-of-the-art speech and image recognition benchmarks. In [53], the authors address the time-consuming gate-level timing simulations of large DNNs performed for exploration of timing speculation by proposing FATE. This methodology implements two complementary ideas: (a) DNN based acceleration of timing simulations to estimate the delay of a MAC unit as a function of its inputs, and (b) sampling based timing error estimation where timing simulations are sampled and performed on a subset of MAC units, and timing errors are probabilistically injected in the remaining MAC units at the same rate. With this methodology, $8 - 58\times$ speed-up in timing simulations is achieved with less than 2% error in classification accuracy.

Finally, in [54], authors propose a cross-layer approach to assess the accuracy of hardware neural networks (HNNs) to dynamic voltage and temperature varia-

Technology	Operating Voltage (V)	Noise Margin (mV)
20nm III-V TFET	0.3 (Nom)	92.90
	0.2	66.24
	0.1	34.02
20nm HP FinFET	0.9 (Nom)	249
	0.5	186.35
20nm HP CMOS	0.8 (Nom)	236.10

Table 2.1: Noise Margin of 20nm TFET, FinFET and CMOS at Nominal and Scaled Supply Voltages.

tions. In this approach, the timing errors are measured from hardware layer using gate-level simulations of a post layout circuit and injected back into neural network inference process to evaluate the accuracy at different operating conditions. As observed in Fig. 2.7, this approach shows significant reduction in classification accuracy of CNNs as the voltage is scaled. These recent works exploring the effect of voltage underscaling [52, 53] and dynamic variations [54] in deep learning hardware, have further highlighted the importance of understanding the effect of variations in operating voltage on the classification accuracy, particularly at scaled supply voltage in sub-20 nm technology nodes.

Operating at low voltages also increases the susceptibility to voltage variations due to reduced noise margins, as listed in Table 2.1. However, due to inherent error resilience of deep learning applications, researchers are exploring the accuracy and power trade-offs to explore the impact of voltage scaling in these applications. Most of these methodologies rely on time extensive hardware simulations. A timing error probability model is proposed in this work which can be used to determine the error probability at the output for a given operating voltage, clock frequency and percentage noise, without any time consuming simulations.

The proposed timing error methodology is demonstrated using both modern FinFET [55] and emerging TFET [56] technologies in 20 nm technology node. These technologies exhibit different error vs. energy characteristics at scaled voltages, as investigated in this work. Through this model, the error probability at the output of a given design can be evaluated for a certain supply voltage, and other parameters such as operating clock frequency, and the supply voltage variations. The impact of different operating conditions is discussed and analyzed through multiple scenarios for FinFET and TFET technology. The proposed error probability models are also used to analyze the bit-level error resilience of neural networks and quantify the inference accuracy as a function of supply voltage. This provides a new dimension for optimization and enable better understanding of the impact of voltage scaling on energy efficiency and accuracy trade-offs in neural networks, which can further be leveraged to enable the implementation of efficient error correction techniques and design low-power DNN accelerators.

Chapter 3

Error Probability Modeling Methodology - Simple Sequential Path

The modeling methodology proposed in this work uses a timing or voltage distribution (determined based on the operating supply voltage and power supply noise) to mathematically compute the timing error probability at the output of a given circuit (Step 2 in Fig. 1.1) [51]. Considering a simple sequential timing path (Fig. 3.1) as an example, this chapter presents the proof of concept for the computation of timing error probability. The rest of the chapter is organized as follows. Section 3.1 describes the computation of timing probability distribution from a given voltage distribution at certain operating voltage and power supply noise. The methodology for the computation of timing error probability from the given distribution for a simple sequential path is explained in section 3.2. The imple-

mentation results of the methodology for a simple timing path is analyzed in section 3.3 while considering different scenarios based on the operating voltage, clock frequency and power supply noise. Finally, the results are discussed in section 3.4.

3.1 Timing Probability Distribution

In highly parallel synchronous circuits, simultaneous switching of a large number of registers produce a significant current drawn from the power networks, therefore, causing voltage fluctuations [27, 57]. The fluctuations in the power supply voltage can be modeled as a random variable with a Gaussian distribution. The standard deviation (σ) and mean (μ) of the distribution is represented by the power supply noise and the operating voltage, respectively [51, 58]. Considering the unique voltage-delay characteristic of the technology, the timing probability distribution can be modeled as a function of the given voltage distribution. The probability density function (*pdf*) obatined is used in the proposed modeling methodology to determine the timing error probability.

The timing probability distribution is determined from the voltage distribution by applying the change of variable technique [59]. According to this technique, if X is a continuous random variable with the probability density function (*pdf*) f_X, then the probability density function of the random variable Y, defined as $Y = g(X)$, is given by,

$$f_Y(y) = f_X[g^{-1}(y)]\left|\frac{d}{dy}g^{-1}(y)\right|. \tag{3.1}$$

For a simple sequential circuit with a single input (Fig. 3.1), the overall delay of the timing path can be determined for different voltages and can be used to model

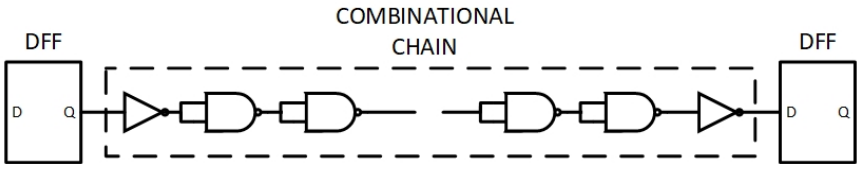


Figure 3.1: Simple sequential data path.

the voltage-delay relationship for the path. Given the voltage *pdf* $f_V(v)$, with the voltage random variable represented as a function of the timing random variable $v(t)$, the timing *pdf* $f_T(t)$ can be determined from (3.1) as,

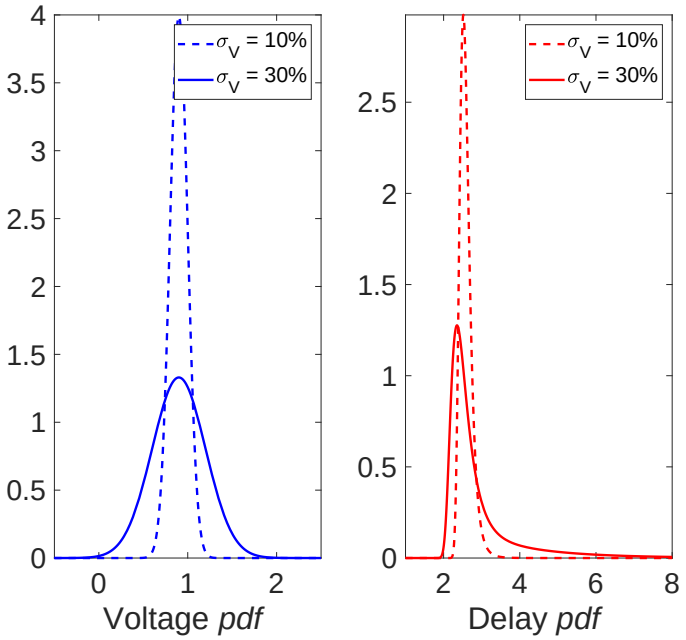
$$f_T(t) = f_V[v(t)]\left|\frac{d}{dt}v(t)\right|. \tag{3.2}$$

As an example, the voltage and timing *pdf* of the sequential data path shown in Fig. 3.1 are illustrated in Fig. 3.2(a) for 20nm high performance (HP) FinFET technology and in Fig. 3.2(b) for 20nm heterojunction (HetJ) TFET technology. The mean (μ) of the voltage distributions are the respective nominal voltages, 0.9V for FinFET and 0.3V for TFET, and the standard deviations (σ) considered are 10% and 30%. The timing *pdf* shown in these figures is obtained from (3.2). The skewed end in timing *pdf* is due to the nonlinear voltage-delay relationship, especially at lower supply voltages where an exponential increase in delay is observed.

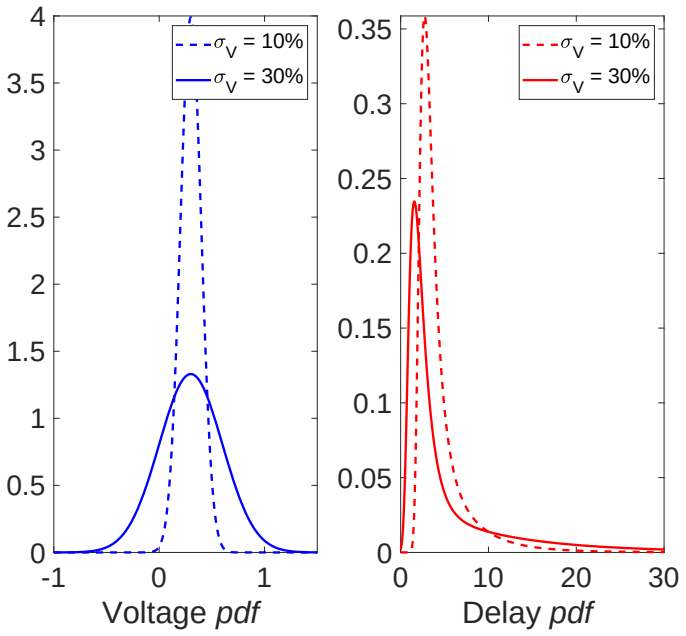
3.2 Error Probability Computation

In a synchronous digital circuit, a data path fails timing if the summation of the overall delay through the combinational logic (t_G) and the setup time (t_S) is greater than the clock period (T_{clk}),

$$t_G + t_S > T_{clk}. \tag{3.3}$$



(a) 20nm HP FinFET



(b) 20nm III-V HetJ TFET

Figure 3.2: Voltage *pdf* and delay *pdf* of the sequential data path shown in Fig. 3.1 at two σ values of 10% and 30%: (a) 20 nm FinFET technology, (b) 20 nm hetero-junction TFET technology.

Therefore, the timing error probability can be computed by the integration of the timing *pdf*, obtained from (3.2) for a simple datapath, where the lower integration limit is defined by the operating clock period. Thus, the timing error probability can be formulated as,

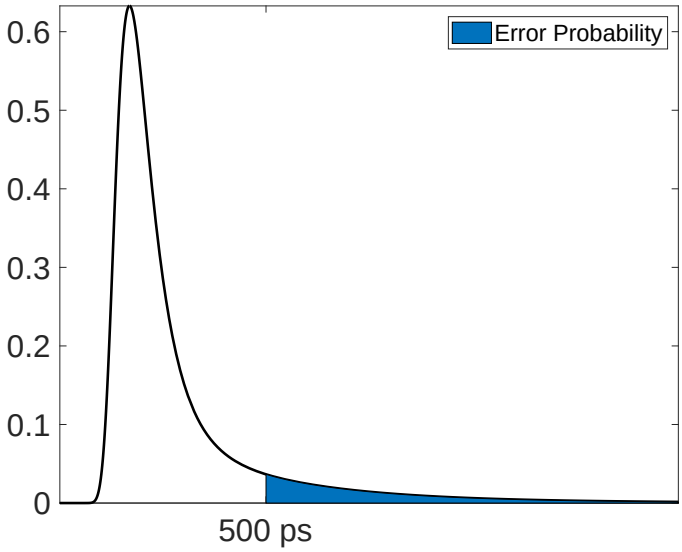
$$\text{Error Probability, } P = \int_{T_{clk}}^{\infty} p(\delta)\, d\delta, \tag{3.4}$$

where T_{clk} is the clock period and $p(\delta)$ is the timing *pdf*. Due to the voltage-delay relationship, the timing probability can also be computed from the voltage *pdf* by determining the voltage at which the delay of the timing path exceeds the operating clock period. The voltage at which the delay of the timing path is equal to the clock period can be determined by using the function, $v(t)$ (3.2). Therefore, the timing error probability can be computed as,

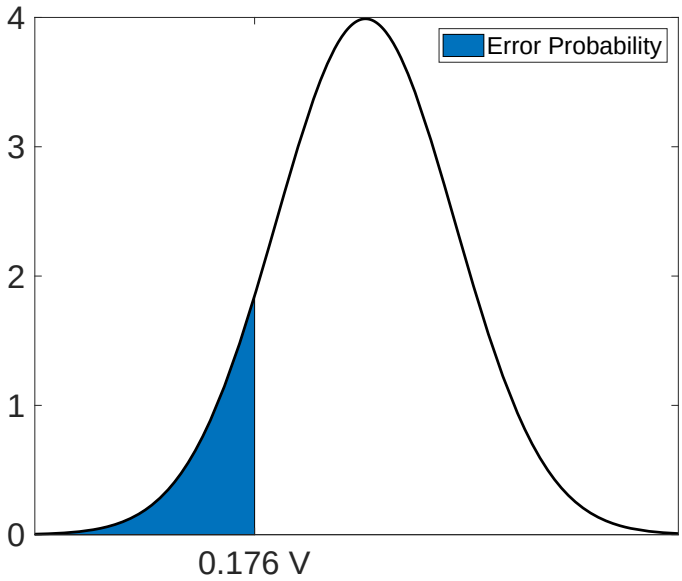
$$\text{Error Probability, } P = \int_{-\infty}^{V_{max}} p(V)\, dV, \tag{3.5}$$

where the upper integration limit V_{max} is the maximum operating voltage at which the path with the largest delay fails timing and $p(V)$ is the voltage *pdf*.

As an example, both (3.4) and (3.5) are illustrated in Fig. 3.3 where the circuit shown in Fig. 3.1 is implemented with 20nm HetJ TFET technology. The clock period is 500 ps and the operating voltage below which the path delay exceeds 500 ps is 0.178 V. The shaded regions in both graphs are equivalent and represent the error probability of approximately 11%. The proposed formulation of timing error probability is verified through Monte Carlo simulations. Specifically, the sequential data path shown in Fig. 3.1 is designed in 20nm FinFET technology and Monte



(a) Timing *pdf*



(b) Voltage *pdf*

Figure 3.3: Illustration of timing error probability of a data path shown in Fig. 3.1 and designed in 20nm HetJ TFET technology. The error probability is computed from (a) timing *pdf* by using (3.4) and (b) voltage *pdf* by using (3.5). The shaded regions in both graphs are equivalent.

Carlo simulations are performed at 0.9V nominal supply voltage with 10% σ. The clock frequency is in the range of 3 to 4 GHz. The difference between the timing error probability obtained via Monte Carlo simulations and obtained via (3.5) is negligible where the average error is approximately 0.12%.

3.3 Results

The simple datapath (Fig. 3.1), designed using 20nm HP FinFET [55] and 20nm HetJ TFET [56] technologies, is simulated in HSPICE to obtain voltage-dependent delay values. The proposed methodology can then be used to evaluate the dependency on the clock period and power supply noise (σ in voltage *pdf*). For this analysis, the power supply noise is varied from 1–30% while the mean of the voltage *pdf* is the operating supply voltage. The timing error probability results are analyzed while considering different scenarios, as described in the following subsections.

3.3.1 Error Probability for Different Clock Periods

The timing error probability is computed for different clock periods and power supply noise, as shown in Fig. 3.4, for the two technologies. Respective nominal voltages, 0.9V for FinFET and 0.3V for TFET, are considered as the operating supply voltage for this analysis. As illustrated in the figure, an increase in the clock period improves the timing error probability at the output, but it also depends on the uncertainty in the supply voltage. For instance, it is observed in Fig. 3.4(a) that for 1% supply noise, a 1.2% increase in the clock period reduces the error probability drastically ($\approx 96\%$). However, when considering 20% supply noise, a 75.4%

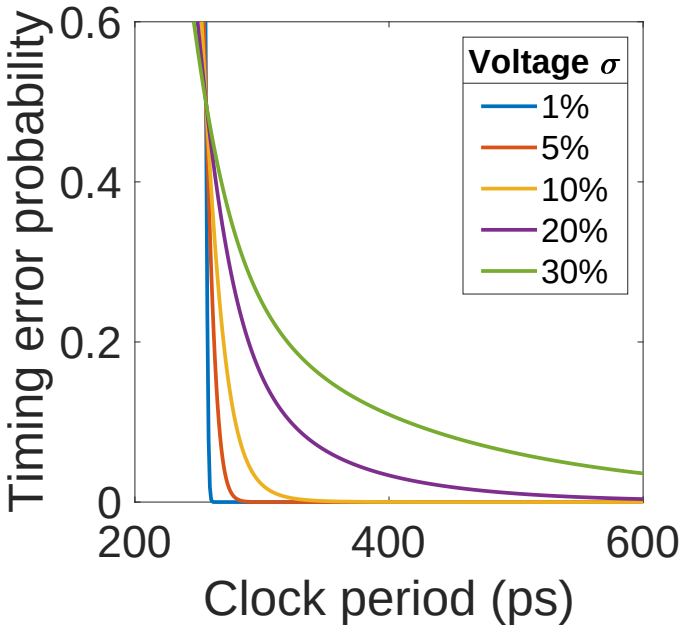
Technology	f_{clk} (GHz) for different σ_{VDD}		
	1%	5%	10%
HP FinFET	3.86	3.65	3.30
HetJ TFET	2.63	1.56	0.66

Table 3.1: Operating clock frequencies for different power supply noise in simple sequential path, designed using 20nm HP FinFET and 20nm HetJ TFET technologies, to obtain error probability ≈2%. Nominal supply voltage is used for both the technologies.

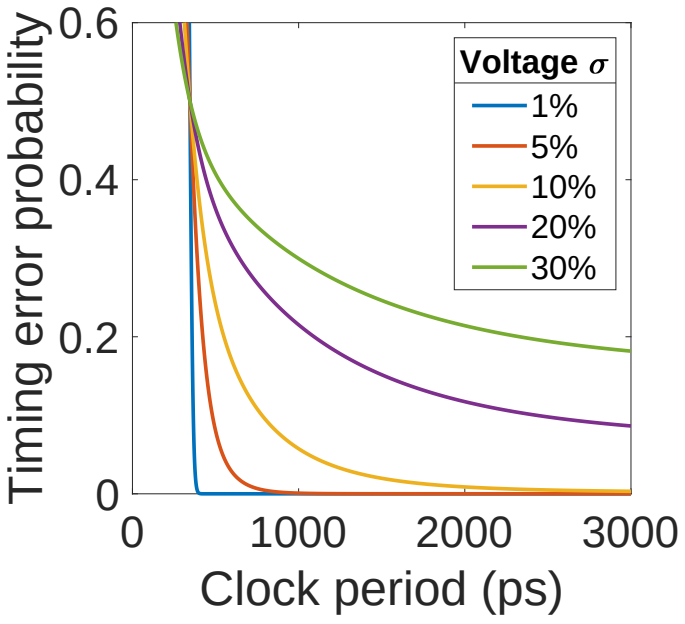
increase in the clock period is required for a similar reduction in error. TFET technology demonstrates a much higher error probability compared to FinFET technology, particularly at higher supply noise (>10%). This observation further implies that the circuit can operate at a higher frequency at nominal voltage for FinFET technology as compared to TFET technology, as listed in Table 3.1. It is also evident that the operating clock frequency required to obtain certain error probability, decreases with increasing power supply noise. For example, considering the circuit designed using TFET technology and operating at 0.3V, the maximum clock frequency to achieve ≈2% error probability is 2.6GHz at 1% supply noise as opposed to 655MHz at 10% supply noise.

3.3.2 Error Probability at Same *ClockPeriod-to-PathDelay* Ratio

In this scenario, 20nm HP FinFET and 20nm HetJ TFET technologies are compared while considering different frequencies for each technology in order to obtain same *ClockPeriod-to-PathDelay* ratio of 2. The power supply voltage considered for this analysis are the respective nominal voltages for the two technologies, i.e. 0.9V for FinFET and 0.3V for TFET. The operating clock frequency, path delay,



(a) 20nm HP FinFET



(b) 20nm III-V HetJ TFET

Figure 3.4: Dependence of error probability on clock period and power supply noise for the circuit shown in Fig. 3.1: (a) 20nm FinFET technology, (b) 20nm HetJ TFET technology.

Technology	Voltage (V)	Delay (ps)	f_{clk} (GHz)	$P_{10\%}$
HP FinFET	0.9	129.4	3.86	≈0
HetJ TFET	0.3	219	2.28	14.11%

Table 3.2: Timing characteristics for error probability computation at respective nominal supply voltages for 20nm HP FinFET and 20nm HetJ TFET technologies. The *ClockPeriod-to-PathDelay* ratio considered for both the technologies is 2.

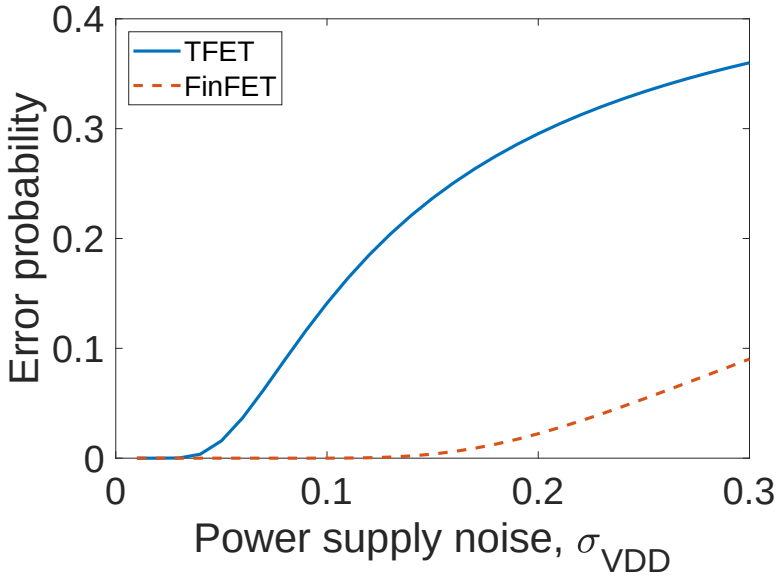


Figure 3.5: Error probability as a function of noise at respective nominal supply voltages (0.9V for FinFET and 0.3V for TFET). The *ClockPeriod-to-PathDelay* ratio considered is 2.

and the timing error probability at 10% noise are listed in Table 3.2 for the two technologies. The error probabilities of the two technologies are plotted in Fig. 3.5 as a function of power supply noise.

Even though FinFET circuit operates at a higher frequency compared to the TFET circuit, the latter exhibits a significantly higher error probability. This can be attributed to higher sensitivity to voltage fluctuations at the nominal voltage in

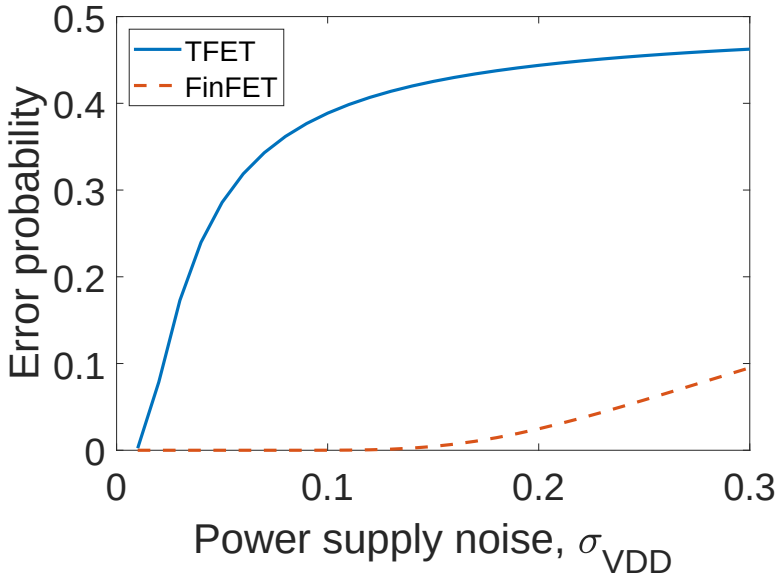


Figure 3.6: Error probability as a function of noise at respective nominal supply voltages (0.9V for FinFET and 0.3V for TFET). The clock frequency considered for both the technologies is 4GHz.

TFET, as compared to FinFET (discussed in detail in section 5.2). Therefore, even though TFET has higher available timing slack compared to FinFET, a drastic increase in the timing error probability is observed in TFET technology for power supply noise $>$5%.

3.3.3 Error Probability at Same Clock Frequency

The two technologies are compared at 4GHz operating clock frequency while considering nominal operating voltages. The *ClockPeriod-to-PathDelay* ratio for the two technologies at 4GHz clock frequency is 1.94 and 1.14 for FinFET and TFET technologies, respectively. TFET has a much higher delay as compared to FinFET at the nominal voltages due to which the timing error probability is significantly higher, as shown in Fig. 3.6.

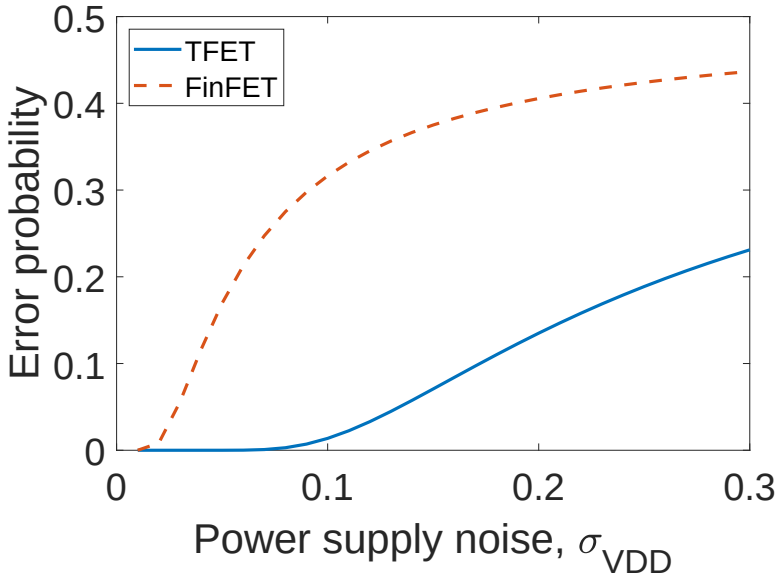


Figure 3.7: Error probability as a function of noise at 0.4V supply voltage and 2GHz clock frequency.

3.3.4 Error Probability at Same Voltage

The operating voltage and frequency considered in this analysis for both the technologies is 0.4V and 2GHz. The delay of the timing path at such low voltage is much lower in TFET as compared to FinFET. Therefore, the *ClockPeriod-to-PathDelay* ratio for FinFET and TFET technologies is 1.3 and 3.1, respectively. The higher ratio results is more available timing slack, due to which TFET exhibits a much lower timing error probability in this scenario as compared to FinFET. The respective error probabilities are shown in Fig. 3.7.

3.4 Discussion

From these preliminary results, it is observed that the timing accuracy of the datapath depends on the operating clock frequency and power supply noise, and can be improved by increasing the clock period and controlling the supply voltage uncertainties within certain range (based on the technology in consideration). Comparing the two technologies and considering same clock frequency, it is evident that if TFET and FinFET circuits operate at a sufficiently low voltage, TFET exhibits a much lower error probability since it outperforms FinFET at low voltages. However, at the respective nominal voltages, TFET has a much higher error probability due to comparatively higher delays.

Therefore, TFET technology is a promising approach to achieve lower subthreshold slope as compared to FinFET technologies, therefore enabling aggressive voltage scaling and energy efficient designs. However, due to lower noise margins at low voltages and higher sensitivity of delay to voltage variations, it is critical to analyze the accuracy of the circuit. Irrespective of the technology in consideration, the timing error probability computation using the proposed methodology can be used to further evaluate the timing accuracy, energy efficiency and performance trade-offs (discussed in detail later).

Chapter 4

Error Probability Modeling Methodology - Multiple Timing Paths and Pipeline Stages

Computation of the timing error probability for a simple single-input timing path is discussed in the previous chapter, providing a proof of concept of applying the proposed modeling methodology to analyze the effect of operating clock period and supply voltage fluctuations on the timing accuracy of the path. The modeling methodology is further extended in this chapter to incorporate the circuits with multiple inputs driving an output, and with multiple pipeline stages [6]. The rest of the chapter is organized as follows. Section 4.1 describes the error probability computation in circuits with multiple timing paths driving an output within the same pipeline stage. In section 4.2, the methodology is extended to circuits with multiple pipeline stages for determining the error probability at the output driven by sequen-

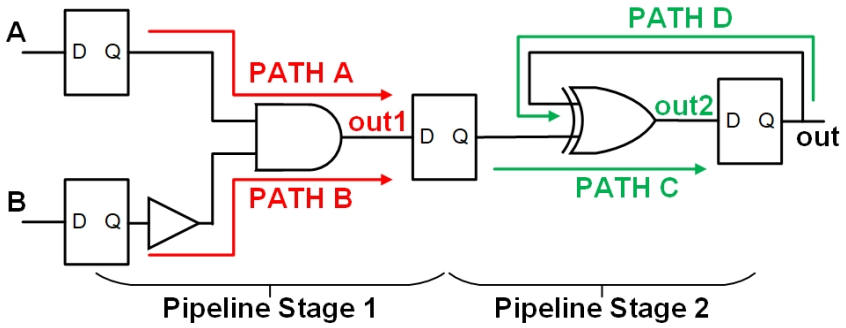


Figure 4.1: 1-bit MAC unit to illustrate multiple paths within a pipeline stage and sequentially adjacent pipeline stages.

tially adjacent paths. The proposed methodology is summarized in 4.3 through the results obtained when considering the example of a 2-bit multiplier circuit with input and output pipeline stages.

4.1 Error Probability for Multiple Timing Paths

This section discusses the error probability modeling methodology for multiple timing paths within the same pipeline stage, such as the error probability at node *out1* in Fig. 4.1. Here, the average supply voltage of the gates within the same pipeline stage is assumed to be the same. This assumption follows the observation that there is very high spatial correlation in the average supply voltage of the adjacent nodes [58]. It has also been demonstrated by [60] that the delay impact of the power supply noise is determined by the average supply voltage rather than peak voltage. Thus, the timing error probability of an output driven by multiple paths is determined by the integration of the voltage *pdf* where the upper integration limit is determined by the maximum voltage at which at least one of the timing paths fails. This path corresponds to the critical path.

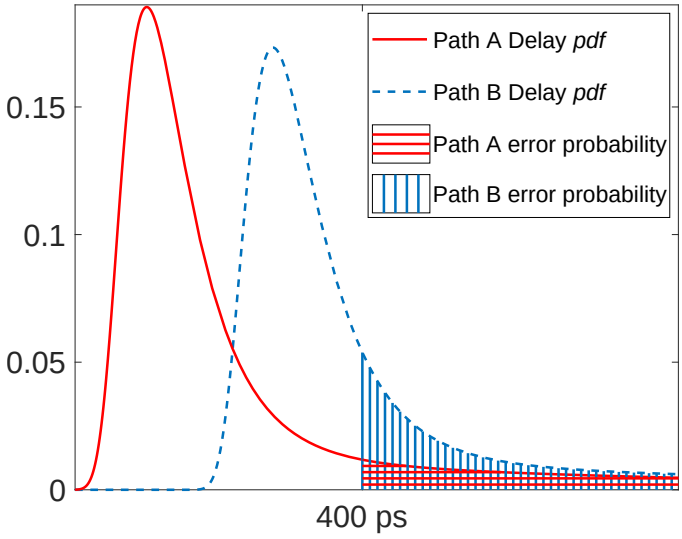
As an example, consider the timing paths Path A and Path B of the same pipeline stage in Fig. 4.1, where 20nm HetJ TFET technology is used. For 0.3V supply voltage with 30% σ, and 400ps clock period, the voltage *pdf* and delay *pdf* corresponding to these two timing paths are shown in Fig. 4.2. It is evident from Fig. 4.2(a) that the error probability for Path B is greater, since the delay of Path B is higher than the delay of Path A. Therefore, from the delay *pdf*s of Path A and Path B, the overall error probability can be determined by the summation of the individual error probabilities minus the probability that both paths fail (intersection probability). Equivalently, the overall error probability can be determined from the voltage *pdf* by identifying the voltage at which the delay of one of the timing paths is equal to the clock period (0.26V in this example). Since the delay of Path B is greater, Path B fails first as voltage is reduced. Thus, it is computationally more feasible to determine the overall error probability by integrating the voltage *pdf* with the appropriate upper limit.

Based on the above discussion, the formulation of the appropriate upper limit can be generalized for n paths driving an output as follows,

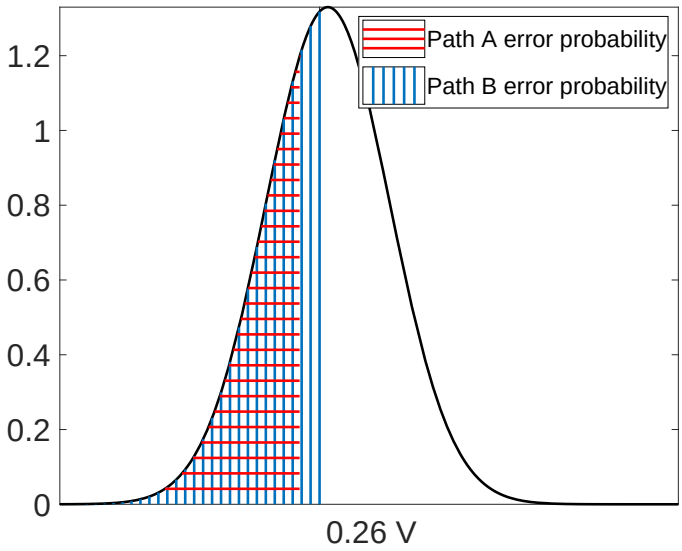
$$V_1, V_2, \ldots, V_n = v_1(t_{clk}), v_2(t_{clk}), \ldots, v_n(t_{clk}), \tag{4.1}$$

where t_{clk} is the clock period, and $V_1, V_2, \ldots, V_n$ are the supply voltages at which the corresponding timing path reaches the maximum delay to satisfy the timing requirement at a specific frequency. These voltages depend upon the data path depth as well as the specific gates used along the path. The maximum supply voltage, V_{max}, at which one of the timing paths fail can be determined from (4.1) as,

$$V_{max} = max(V_1, V_2, \ldots, V_n). \tag{4.2}$$



(a) Timing *pdf*



(b) Voltage *pdf*

Figure 4.2: Example to illustrate the timing error probability calculation of multiple timing paths within a sequential circuit: (a) Delay *pdfs* and (b) voltage *pdf* of Path A and Path B in Fig. 4.1, designed using 20nm HetJ TFET technology. The operating voltage and clock period are 0.3V ($\sigma = 30\%$) and 400ps, respectively.

Using (3.5), the overall output error probability can be determined by the integration of the voltage *pdf* where the upper integration limit is equal to (4.2).

4.2 Error Probability for Multiple Pipeline Stages

The previous section discusses the error probability computation within the same pipeline stage. In this section, the methodology is extended to determine the timing error probability between sequentially adjacent paths or timing paths between multiple pipeline stages.

Unlike previous section where timing paths within the same pipeline stage are assumed to have the same average supply voltage (due to spatial proximity), the supply voltage of different pipeline stages is assumed to be independent. This assumption is based on the observations in [58, 60], where an average supply voltage is considered for each switching interval to account for temporal and spatial correlation in the voltage of adjacent nodes. Therefore, referring to Fig. 4.1, the timing error probability of each pipeline stage can be independently determined from (3.5) and (4.2). The error probability at the final output node *out* can then be determined as the *union* of the error probabilities of each pipeline stage, as the error at the output can be caused by the failure of either stage or the failure of both stages at the same time. Thus, the error at node *out* is given by

$$P_{out} = P_{out1} \cup P_{out2} = P_{out1} + P_{out2} - P_{out1} \times P_{out2}. \tag{4.3}$$

The output probability, P_{out}, is plotted in Fig. 4.3 as a function of power supply noise at 0.3V operating voltage and 1.8GHz clock frequency.

This approach can be generalized for a circuit with n pipeline stages as follows,

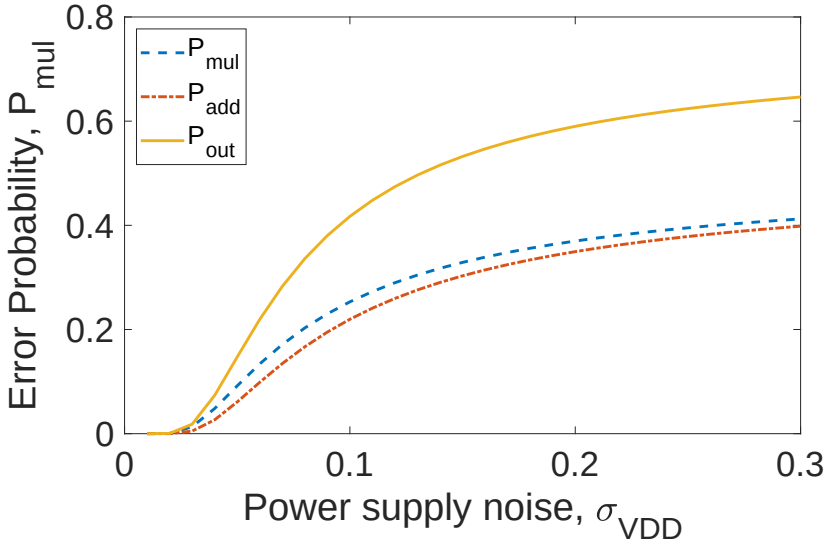


Figure 4.3: Timing error probability as a function of power supply noise, σ_{vdd}, for the output node of a 1-bit MAC unit in 20nm HetJ TFET technology with 0.3V operating voltage and 1.8GHz clock frequency.

$$P_{out} = P_{out1} \cup P_{out2} \ldots \cup P_{outn}, \tag{4.4}$$

where $P_{out1}, P_{out2}, \ldots, P_{outn}$ are the error probabilities at the output of each pipeline stage and P_{out} is the error probability for the final output. Furthermore, (4.4) can be applied when computing the timing error probability after n accumulations. For instance, if the feedback path D (Fig. 4.1) is simulated through multiple iterations, it is observed in Fig. 4.4, that the timing error probability approaches 1 after 40 iterations for 5% power supply noise when TFET technology is used. This is equivalent to the timing error probability at the output of 40 sequentially adjacent paths with the same delay as the feedback path D. The number of sequentially adjacent paths reduce to 18 for 10% power supply noise exhibiting a strong trade-off between the timing error probability and number of accumulations in the MAC.

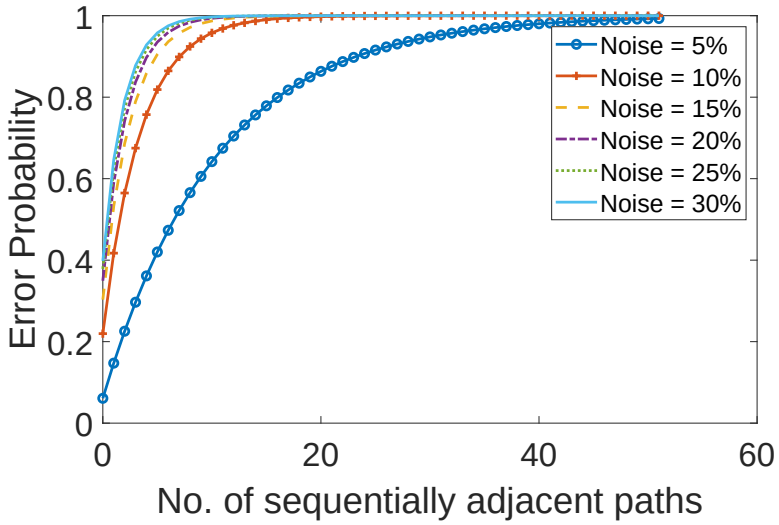


Figure 4.4: Timing error probability as a function of number of sequentially adjacent paths for different power supply noise at 0.3V operating voltage for 20nm HetJ TFET technology.

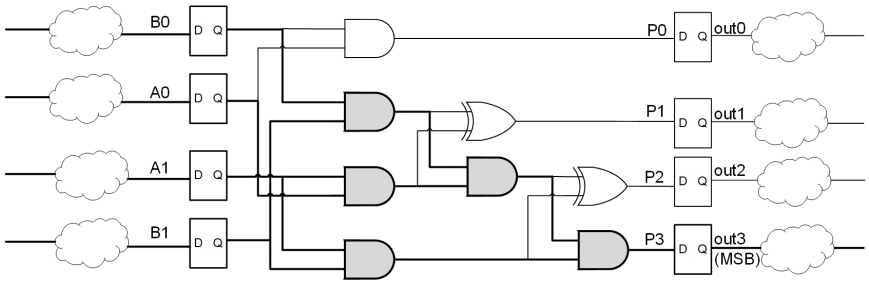


Figure 4.5: Schematic of a 2-bit multiplier with input and output pipeline stages.

4.3 Results

The proposed methodology to determine the error probability is summarized through a 2-bit multiplier as illustrated in Fig. 4.5. The most significant bit (MSB) *out*3 is used to clarify the methodology. Both FinFET (20nm HP) and TFET (20nm HetJ) technologies are considered. The clock frequency is 2.5GHz and the supply voltages are 0.9V and 0.3V (nominal voltages), respectively, for FinFET and TFET technologies. As observed in the figure, the four inputs to the multiplier are the

output of the first pipeline stage ($B0$, $A0$, $A1$, and $B1$) and therefore, they each have a corresponding timing error probability, as determined by (3.5) and (4.2). The timing error probabilities for the four inputs are shown in Fig. 4.6.

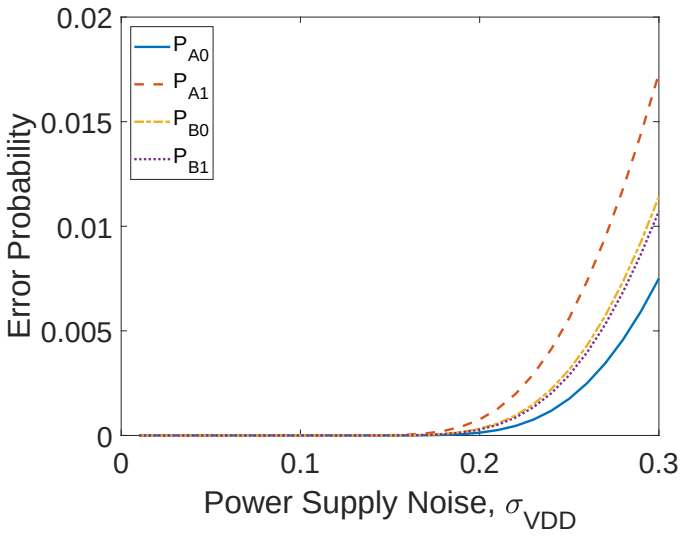
The timing error at node $out3$ can be due to timing failure within the first pipeline stage (e.g., at least one of the four outputs $B0$, $A0$, $A1$, and $B1$ fails) or due to timing error along the critical path within the second pipeline stage (e.g. node $P3$ fails). Note that when multiple inputs from the first pipeline stage drive the output within the second pipeline stage, the input with the highest timing error probability is considered as this input represents the highest voltage (V_{max}) at which one of the inputs fails. Thus, Eqn. (4.4) can be applied to determine the timing error probability at node $out3$ as,

$$P_{out3} = P_{P3} \cup max(P_{A0}, P_{B0}, P_{A1}, P_{B1}), \tag{4.5}$$

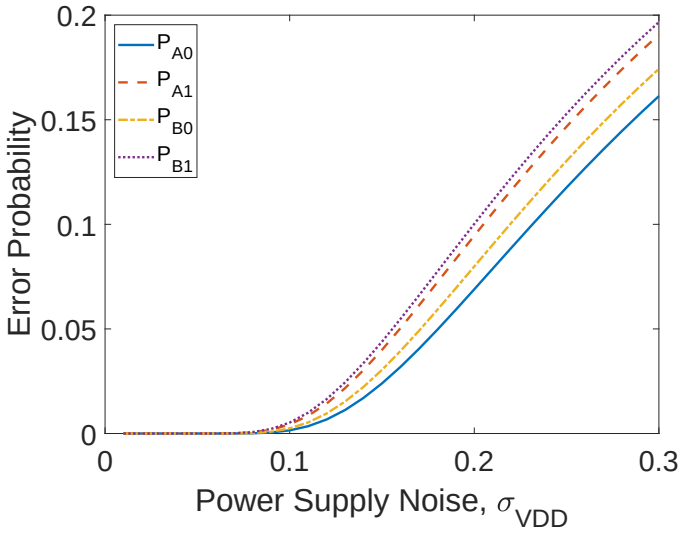
where P_{P3} is the timing error probability at the multiplier output P3 (second pipeline stage) and $P_{A0}, P_{B0}, P_{A1}, P_{B1}$ are the timing error probabilities from the first pipeline stage. The timing error probability at $out3$ is illustrated in Fig. 4.7.

This example illustrates the several scenarios described previously:

- Timing error probability of a single pipeline stage with one timing path, see (3.5).
- Timing error probability when there are more than one timing paths that drive an output within a single pipeline stage, see (4.1) and (4.2).
- Timing error probability when there are multiple pipeline stages between primary inputs and primary outputs, resulting in multiple sequentially adjacent timing paths, see (4.4).

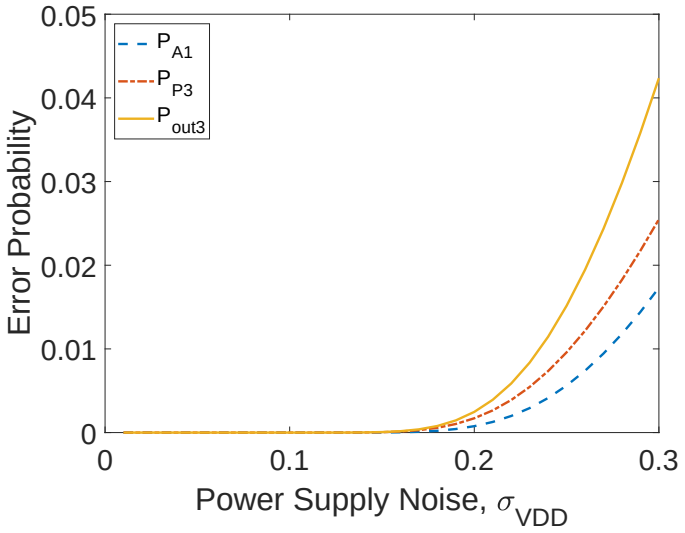


(a) 20nm HP FinFET

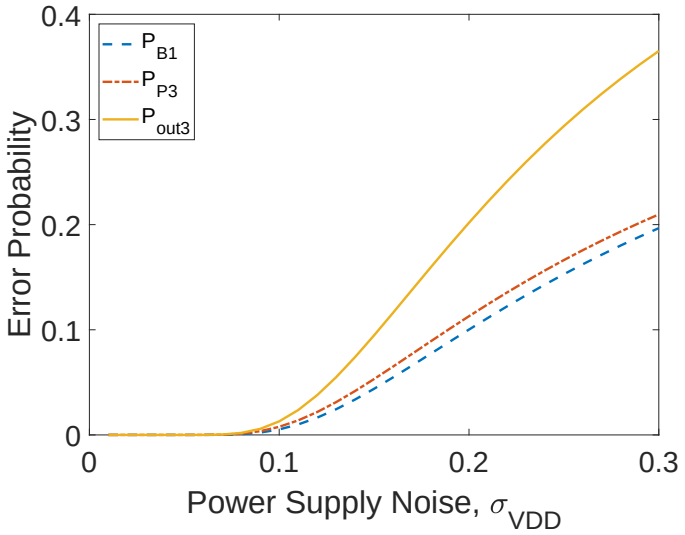


(b) 20nm HetJ TFET

Figure 4.6: Timing error probabilities as a function of power supply noise, σ_{VDD}, for the four inputs to the multiplier (Fig. 4.5) designed using (a) 20nm HP FinFET, and (b) 20nm HetJ TFET technologies. The respective operating voltages considered are 0.9V and 0.3V, and the clock frequency is 2.5GHz.



(a) 20nm HP FinFET



(b) 20nm HetJ TFET

Figure 4.7: Timing error probabilities as a function of power supply noise, σ_{VDD}, for MSB output of the multiplier (*out*3).

Chapter 5

Case Study - 8-bit MAC

A multiply-accumulate unit is the basic computation block in deep learning hardware. In the case study discussed in this chapter, the proposed error probability modeling methodology [6] is applied to an 8-bit MAC circuit designed using 20nm FinFET [55] and 20nm TFET [56] technologies. In FinFET, both high performance (HP) and low power (LP) technologies are considered, and for TFET, homojunction (HomJ) and heterojunction (HetJ) technologies are considered when evaluating the timing error probabilities. The rest of the chapter is organized as follows. Section 5.1 provides a brief description of the 8-bit MAC circuit. A comparative analysis of the timing error probability in TFET- and FinFET-based circuits is described in section 5.2. In section 5.3, the proposed methodology is implemented with different operating conditions to quantify the dependency of timing error probability on clock frequency (and therefore, available timing slack), and supply voltage. The trade-off between error probability and power consumption is also evaluated. Finally, the chapter is concluded with a brief discussion of the results obtained for 8-bit MAC in section 5.4.

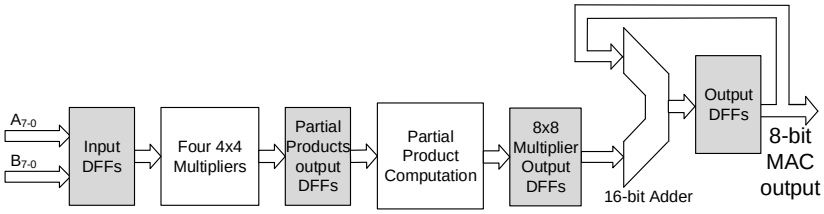


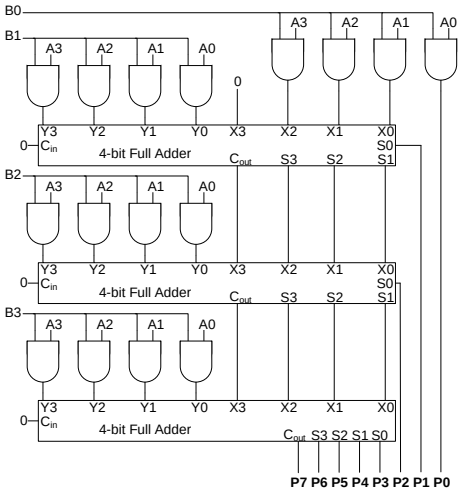
Figure 5.1: Block diagram of 8-bit MAC used in the case study to characterize the trade-offs among voltage, error probability, and power consumption.

5.1 Circuit Description and Implementation

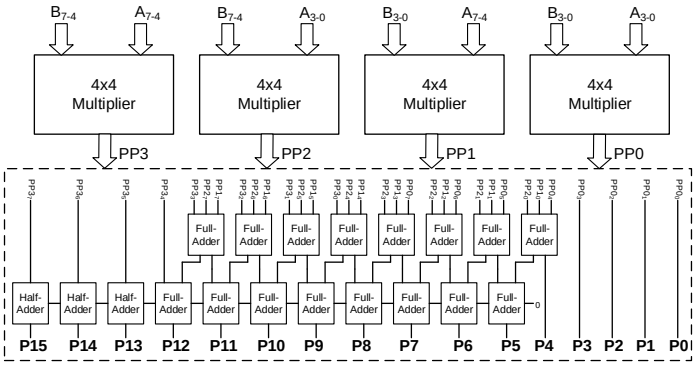
The 8-bit MAC circuit, considered in this case study, consists of three pipeline stages - 4-bit multiplier, partial product computation circuit, and 16-bit adder (as shown in Fig. 5.1). The first stage consists of four 4×4 multipliers. Each 4-bit multiplier (Fig. 5.2(a)) has 16 bitwise AND operations and three 4-bit adders. The partial products are computed within the second pipeline stage through 16 full adders and three half adders (Fig. 5.2(b)). The last stage consists of a 16-bit carry look-ahead adder for accumulation.

Since each pipeline stage consists of multiple timing paths, (3.5) and (4.2) are used to determine the path that results in the highest timing error probability, which corresponds to the critical path. Once the error probabilities at the outputs of each pipeline stage are determined, the error probabilities at the 16 primary outputs of the MAC unit are determined from (4.4). Note that this calculation assumes a single iteration of accumulation stage where the input of the 16-bit adder (that represents the feedback) has zero error probability. Eqn. (4.4) can be recursively applied to determine the error rate after any number of iterations, similar to the 1-bit MAC example illustrated in Fig. 4.4.

The timing error probability at each output of the 8-bit MAC unit is determined. For FinFET technology, the nominal supply voltage is 0.9V whereas for TFET tech-

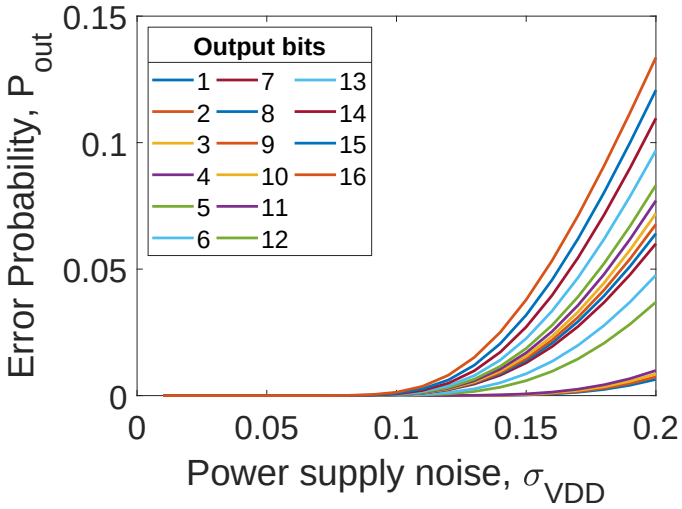


(a) 4-bit multiplier

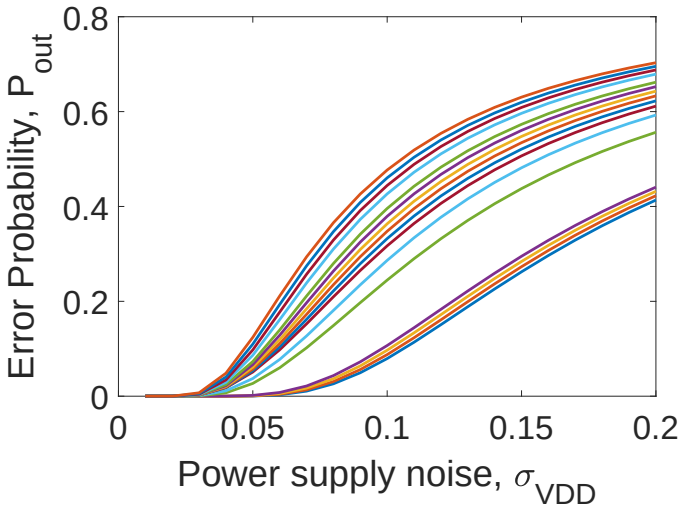


(b) 8-bit multiplier

Figure 5.2: Detailed block diagram of (a) 4-bit multiplier, and (b) 8-bit multiplier circuits.



(a) 20nm HP FinFET



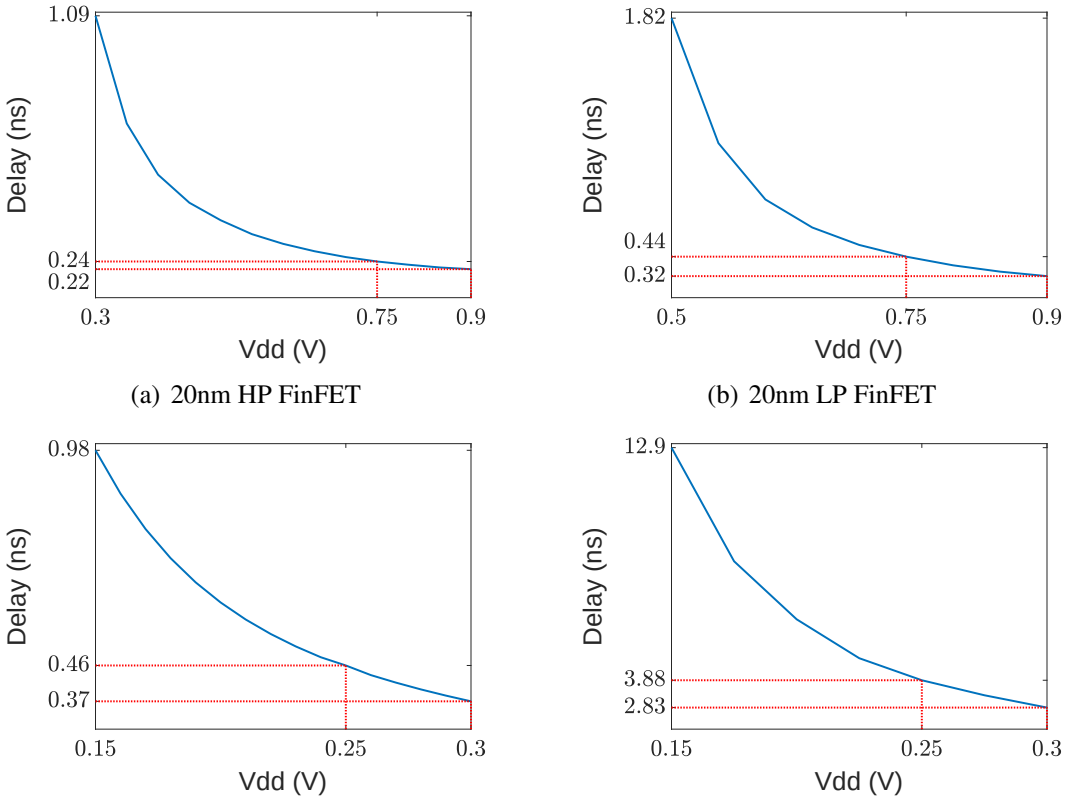
(b) 20nm III-V HetJ TFET

Figure 5.3: Timing error probability of each output bit of 8-bit MAC unit as a function of power supply noise, σ_{vdd}/μ_{vdd}: (a) 20nm HP FinFET technology with 0.9V operating voltage and 3.06GHz clock frequency and (b) 20nm HomJ TFET technology with 0.3V operating voltage and 1.81GHz clock frequency. Note that clock frequencies are chosen to ensure that *ClockPeriod-to-PathDelay* is the same for both cases.

nology, the voltage is 0.3V. The clock frequency for each technology is set to ensure that the ratio of the clock period to longest data path delay at nominal voltage, i.e. *ClockPeriod-to-PathDelay* ratio, is equal to 1.5 for both technologies. To satisfy this ratio, the clock frequency for FinFET-based MAC and TFET-based MAC are, respectively, 3.06GHz and 1.81GHz. The results are illustrated in Fig. 5.3 and are similar to the error probability vs. power supply noise results for a simple sequential path (Fig. 3.5).

5.2 FinFET vs. TFET technologies

It was observed in Fig. 3.4, Fig. 3.5, and Fig. 5.3, that for the same power supply noise, TFET technology has a much higher timing error probability compared to FinFET. The effect of power supply noise on the error probability in the two technologies is discussed in detail in this section. The voltage-delay characteristic of the highest delay path in 8-bit MAC is shown in Fig. 5.4 for FinFET and TFET technology. Comparing HP FinFET (Fig. 5.4(a)) and HetJ TFET (Fig. 5.4(c)) technologies, it is evident that the sensitivity of delay to voltage scaling below the nominal voltage is higher for TFET technology. For FinFET technology, this sensitivity increases only when the voltage is reduced to approximately the threshold voltage. Thus, the effect of the 10% σ (RMS supply noise) for the voltage *pdf* has a more significant impact on error probability for the TFET-based MAC unit. The voltage *pdf* for the highest-delay path within the 8-bit MAC unit is plotted in Fig. 5.5, where the maximum voltage at which this path fails is illustrated for both technologies (0.56 V for FinFET and 0.2 V for TFET). Since the shaded area gives the error probability, as determined by (3.5), TFET technology exhibits a greater error. This susceptibility



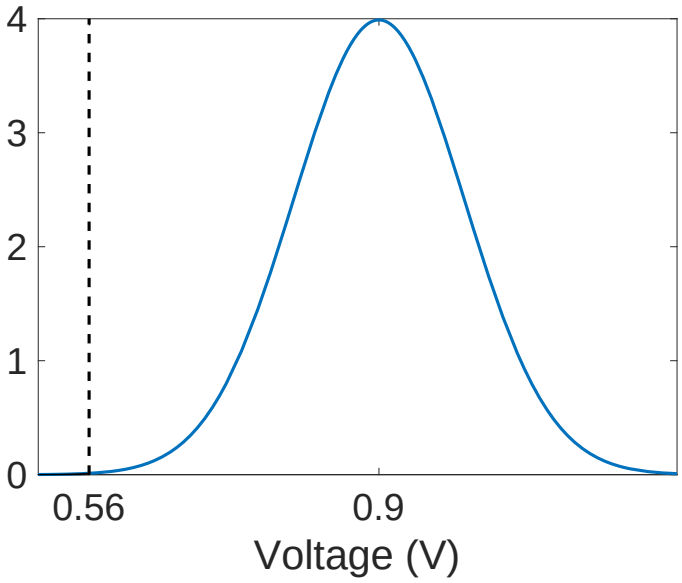
(a) 20nm HP FinFET

(b) 20nm LP FinFET

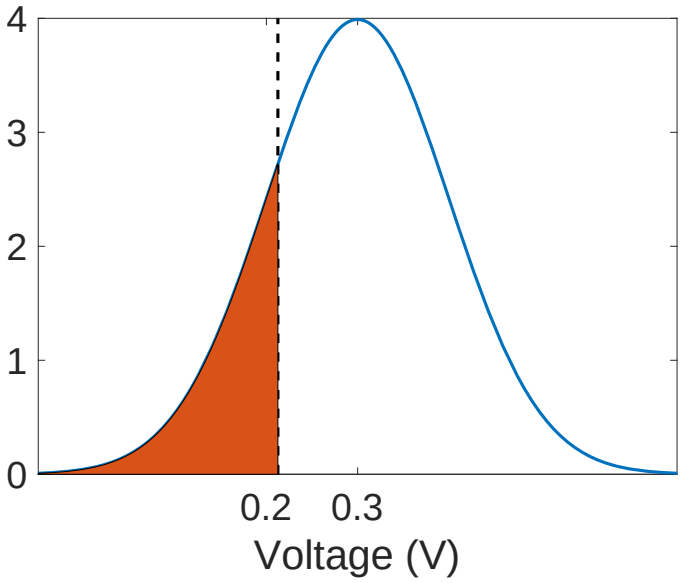
(c) 20nm III-V HetJ TFET

(d) 20nm III-V HomJ TFET

Figure 5.4: Dependence of delay on voltage of the highest-delay path in 8-bit MAC unit for (a) 20nm HP FinFET, (b) 20nm LP FinFET, (c) 20nm HomJ TFET, and (d) 20nm HetJ TFET technologies.



(a) 20nm HP FinFET



(b) 20nm III-V HetJ TFET

Figure 5.5: Voltage *pdf* and the error probability of the highest-delay path in 8-bit MAC unit for (a) 20nm FinFET and (b) 20nm HetJ TFET technologies. The mean of the *pdf* corresponds to the nominal operating voltage and the σ is 10%.

can be significantly reduced only when the noise is controlled within 2–3% of the supply voltage.

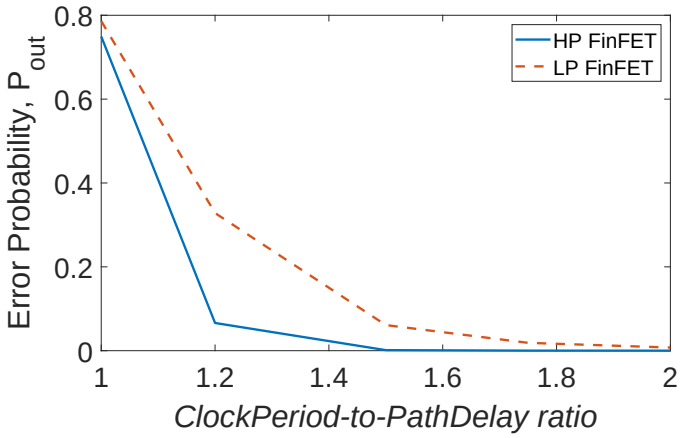
Similarly, comparing the LP FinFET technology (Fig. 5.4(b)) with HP FinFET and HomJ TFET technology (Fig. 5.4(d)) with HetJ TFET, the former technologies have comparatively higher sensitivity to voltage scaling (and higher delay), which would result in higher timing error probabilities, as observed in the results analyzed in the next section.

5.3 Results

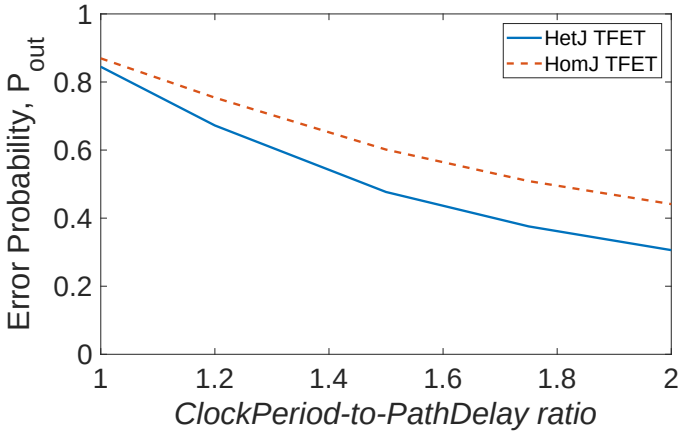
To evaluate the dependence of timing error probability on the operating clock frequency, and supply voltage, the most significant bit (MSB) output of the 8-bit MAC is considered. The trade-off between the timing error probability and power consumption is also evaluated. Both FinFET and TFET technologies at 20nm node are considered in this analysis. For FinFET, high performance (HP) and low power (LP) technologies are considered. For TFET, homojunction (HomJ) and heterojunction (HetJ) technologies are considered. The timing error probability computed by the implementation of the proposed methodology is analyzed as follows.

5.3.1 Dependence on Timing Slack

The operating voltage considered for the technologies are the respective nominal voltages, 0.9V for FinFET and 0.3V for TFET. Due to different voltage-delay characteristics of the technologies, the operating clock frequency is varied to obtain *ClockPeriod-to-PathDelay* ratio in the range of 1 to 2. The different clock frequencies are listed in Table 5.1. The corresponding timing error probabilities are shown in Fig. 5.6 for 10% power supply noise.



(a) 20nm FinFET



(b) 20nm III-V TFET

Figure 5.6: Dependence of timing error probability on *ClockPeriod-to-PathDelay ratio* for the MSB of 8-bit MAC unit at 10% power supply noise at nominal operating voltages: (a) 20nm FinFET and (b) 20nm TFET technologies. Note that the clock frequencies corresponding to each *ClockPeriod-to-PathDelay ratio* are listed in Table 5.1.

Technology	f_{clk} (GHz) for different T_{clk}/d_{path}				
	1	1.2	1.5	1.75	2
HP FinFET	4.49	3.82	3.06	2.62	2.29
LP FinFET	3.08	2.57	2.05	1.76	1.54
HetJ TFET	2.72	2.27	1.81	1.55	1.36
HomJ TFET	0.36	0.29	0.23	0.20	0.17

Table 5.1: Operating clock frequencies for FinFET and TFET technologies to obtain different *ClockPeriod-to-PathDelay* ratios. Nominal supply voltage is used for both technologies

As shown in this figure, for FinFET technologies, the error probability can be significantly reduced when sufficient timing slack is provided. For instance, at 10% supply noise, if the *ClockPeriod-to-PathDelay* ratio is greater than 1.5, the error probability is negligible for HP FinFET technology. Alternatively, for TFET technology, there is considerable error probability even when the clock period is twice the largest path delay. As described before, this behavior is due to the significantly higher sensitivity of delay to supply voltage for TFET technology. Comparing HP FinFET and LP FinFET, the timing error probability for the latter becomes negligible when the *ClockPeriod-to-PathDelay* ratio is approximately 2 as opposed to 1.5 for HP FinFET. This is also because of higher sensitivity of delay to voltage variation, as shown in Fig. 5.4(b).

5.3.2 Dependence on Supply Voltage

In this analysis, the operating voltage is scaled below the nominal voltage for both the technologies. For FinFET, the voltage is scaled from 0.9V to 0.55V while for TFET, the voltage is scaled from 0.3V to 0.175V. The power supply noise con-

sidered is 5%. For each supply voltage, the clock frequency is determined to ensure that the *ClockPeriod-to-PathDelay* ratio remains the same at nominal voltage. Thus, the MAC units operating at scaled voltages run at lower frequency (Table 5.1). *ClockPeriod-to-PathDelay* ratios of 1.2, 1.5, and 2 are considered in this analysis. The error probability results are listed in Tables 5.2 and 5.3 for, respectively, FinFET and TFET technologies. According to these tables, HP FinFET technology exhibits enhanced voltage scaling capability as the impact on error probability is relatively weaker compared to LP FinFET and TFET technologies. For example, at a *ClockPeriod-to-PathDelay* ratio of 1.5, supply voltage can be scaled down to 0.7V with an error probability of less than 2% for HP FinFET technology. Alternatively, for TFET technology, the error probabilities quickly increase as the supply voltage is reduced. To achieve similar error probabilities as FinFET technology, the power supply noise needs to be $\approx 3\%$ or less in TFET-based MAC units.

VDD (V)	LP FinFET			HP FinFET		
	$\frac{T_{clk}}{d_{path}} = 1.2$	$\frac{T_{clk}}{d_{path}} = 1.5$	$\frac{T_{clk}}{d_{path}} = 2$	$\frac{T_{clk}}{d_{path}} = 1.2$	$\frac{T_{clk}}{d_{path}} = 1.5$	$\frac{T_{clk}}{d_{path}} = 2$
0.90	3.65%	0.01%	≈0%	0.01%	≈0%	≈0%
0.85	27.21%	0.37%	≈0%	0.49%	≈0%	≈0%
0.80	76.40%	5.97%	0.05%	7.18%	≈0%	≈0%
0.75	98.52%	37.01%	1.42%	40.64%	0.05%	≈0%
0.70	100%	84.94%	15.33%	86.97%	1.34%	≈0%
0.65	100%	99.41%	60.85%	99.54%	14.07%	0.07%
0.60	100%	100%	95.59%	100%	57.44%	1.55%
0.55	100%	100%	99.94%	100%	94.43%	15.26%

Table 5.2: Timing error probability for the MSB of 8-bit MAC unit designed in 20nm FinFET technology for different supply voltages and *ClockPeriod-to-PathDelay* ratios. Power supply noise is constant at 5%.

VDD (V)	HomJ TFET			HetJ TFET		
	$\frac{T_{clk}}{d_{path}} = 1.2$	$\frac{T_{clk}}{d_{path}} = 1.5$	$\frac{T_{clk}}{d_{path}} = 2$	$\frac{T_{clk}}{d_{path}} = 1.2$	$\frac{T_{clk}}{d_{path}} = 1.5$	$\frac{T_{clk}}{d_{path}} = 2$
0.300	59.63%	28.14%	9.21%	41.36%	12.41%	2.53%
0.275	82.76%	53.26%	23.99%	67.72%	29.90%	8.53%
0.250	95.30%	78.12%	47.95%	87.91%	55.35%	22.49%
0.225	99.24%	93.33%	73.81%	97.19%	79.67%	45.73%
0.200	99.93%	98.78%	91.27%	99.62%	94.01%	71.96%
0.175	100%	99.87%	98.23%	99.97%	98.95%	90.16%

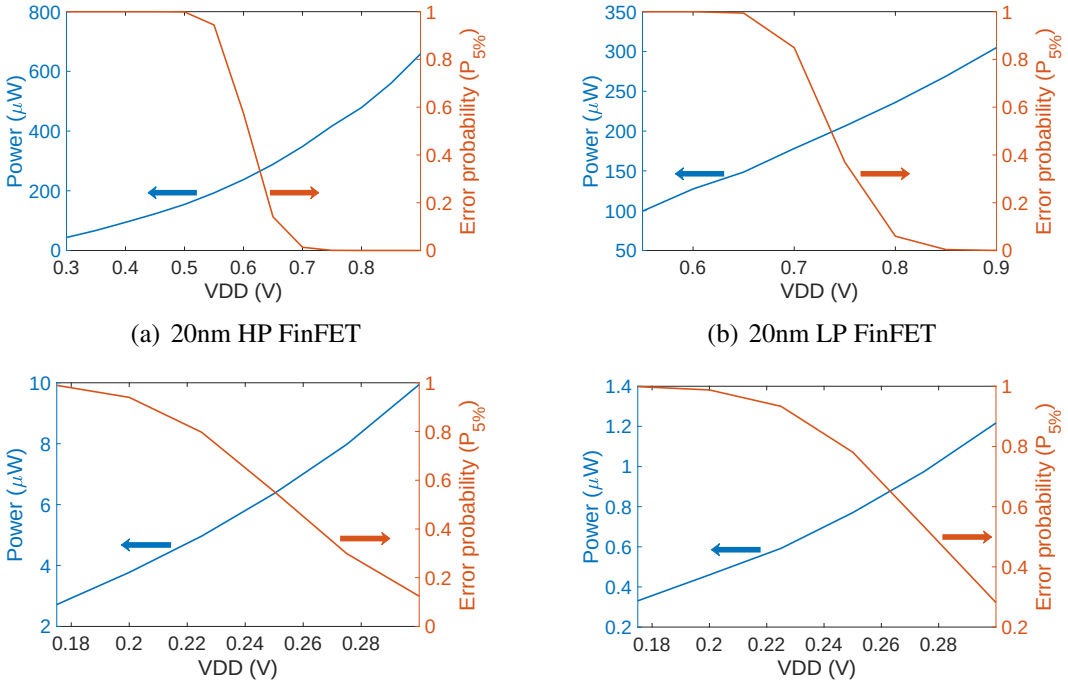
Table 5.3: Timing error probability for the MSB of 8-bit MAC unit designed in 20nm TFET technology for different supply voltages and *ClockPeriod-to-PathDelay* ratios. Power supply noise is constant at 5%.

5.3.3 Error Probability vs. Power Consumption

Considering 5% power supply noise and 1.5 *ClockPeriod-to-PathDelay* ratio, the trade-off between timing error probability and power savings (due to voltage scaling) is quantified for FinFET and TFET technology. The results are illustrated in Fig. 5.7. In HP FinFET technology, voltage scaling from 0.9 V to 0.7 V reduces the power consumption by 47.09% with negligible increase in error probability. The power consumption is much lower in LP FinFET technology, even though the voltage scaling capability is limited compared to HP technology. TFET-based MAC units consume significantly lower power, but the error probabilities are also much higher. For HetJ TFET technology, even a small reduction in supply voltage increases the error probability beyond 20% while achieving a marginal reduction in power consumption. Therefore, the significantly lower power consumption of TFET-based MAC unit over FinFET-based MAC (9.94μW for HetJ TFET vs 658.3μW for HP FinFET) is achieved at the expense of more susceptibility to timing errors.

5.4 Discussion

Several important design considerations are discussed in this section based on the timing error probability results presented in the previous section. An important conclusion is that in TFET-based MAC circuits, primary emphasis should be placed on achieving low noise. Due to the fundamentally different charge transfer mechanism between TFET and FinFET technologies (tunneling vs. diffusion), in FinFET-based circuits, there is relatively weaker dependence of delay to voltage at and around nominal supply voltage compared to TFET-based circuits. This depen-



(a) 20nm HP FinFET

(b) 20nm LP FinFET

(c) 20nm III-V HetJ TFET

(d) 20nm III-V HomJ TFET

Figure 5.7: Timing error probability vs. power consumption tradeoff at 5% power supply noise and a *ClockPeriod-to-PathDelay* ratio of 1.5: (a) 20nm HP FinFET, (b) 20nm LP FinFET, (c) 20nm HetJ TFET, and (d) 20nm HomJ TFET technology.

dence starts to be much stronger once the supply voltage is reduced to the levels of threshold voltage (due to exponential dependence of current on voltage). In TFET-based circuits, however, delay is more sensitive to voltage, even at nominal supply voltages. This characteristic makes voltage scaling highly challenging for TFET-based MAC units as the error rates rapidly rise. These relatively high timing errors can be partially mitigated by constraining the supply noise within 3% of the supply voltage (referring to Fig. 5.3). This requirement would make the power distribution design process more challenging. Also, referring to Fig. 5.6, at the same power supply noise, increasing clock period can be a highly effective method to reduce error rates in FinFET technologies. In TFET technology, however, a more drastic increase in clock period is required to achieve the same reduction in error rates.

Based on the observations, it can be concluded that the proposed model can be used to investigate the effect of power supply noise, clock frequency and supply voltage on the error rates, without relying on time consuming hardware-level simulations. Furthermore, these models can facilitate the quantification of quality-of-results vs. error rates at the application level, providing useful guidelines for several future directions such as error correction in MAC units.

Chapter 6

DNN Error Resilience Analysis Framework

The error probability methodology implemented on a 8-bit MAC in the previous chapter indicates that the timing error rate (due to reduced supply voltage) can significantly vary based on the bit position. Therefore, to understand the bit-level datapath errors in DNNs, it is important to evaluate the error resilience of a neural network at per-bit granularity with varying error rates (unlike prior works that focus on memory errors, which assume uniform error rates across all bits [61, 62]). In this chapter, a DNN error resilience analysis framework is developed to quantify DNN error resilience by inserting bit-level errors in the compute-intensive layers of the network (convolutional and fully-connected layers). This framework is implemented in PyTorch and applied to state-of-the-art neural networks such as ResNet-18 [63], MobileNetV2 [64], and EfficientNet [65]. These DNNs have complex network structures developed to improve the classification accuracy through the

implementation of either residual learning with shortcut connections [63], inverted residual structures with linear bottlenecks [64], or a compound scaling methodology [65]. Different networks considered in this analysis are pre-trained on the ImageNet dataset [66] and are quantized to a given fixed-point bit precision (as determined by the user) prior to error insertion. The rest of the chapter is organized as follows. Section 6.1 gives a brief overview of deep learning hardware and summarizes prior works which focus on the inherent fault tolerance of DNN hardware. Section 6.2 describes the methodology implemented to quantify the accuracy with respect to error rates. The results of the error resilience analysis of different neural networks are discussed at network-level, per-layer and per-bit granularity in section 6.3. Finally, section 6.4 concludes the chapter.

6.1 Related Work

Several prior works characterize the impact of errors in DNNs through fault injection, proposing error mitigation [2] or power optimization [67] techniques in DNN hardware. Li et al. [2] characterize the propagation of soft errors in DNN accelerators and propose techniques to mitigate these errors. The fault injection framework implemented in [2] considers single-event upsets in each datapath latch and buffer component to identify silent data corruption in the network, but they do not analyze the impact on DNN inference accuracy due to different bit-level error rates in the neural network. In [67], the authors analyze the filter/weight sensitivities (due to bit errors), obtain the robustness map of MAC units (by determining the timing error rates), and map the sensitive weights to robust MAC units in order to improve the error resilience of DNN hardware. However, they lack a comprehensive

study of DNN resilience to bit-level datapath errors (in the computational block) at per-layer and per-bit granularity. In addition, multiple research works focus on analyzing the impact on inference accuracy due to memory errors instead of datapath errors. For instance, the Ares framework [68] quantifies the inference accuracy with respect to static memory faults (in weights and activation) at the network-level and per-layer granularity. The per-layer error resilience differences of on-chip memory buffers are exploited in [61], where the error resilience is evaluated during design time to determine the buffer voltage for each layer.

The closest related work focusing on datapath faults is [69], where the authors leverage the results of bit-error resilience analysis to propose a novel error mitigation technique which reduces the impact of bit-errors before the execution of each layer. Even though the error injection methodology implemented in [69] investigates the impact of bit errors in network weights and activations on the accuracy, it does not quantify the DNN error resilience due to different error rates in different layers and bits in the network. Furthermore, the relationship between accuracy and different network and layer parameters (such as number of operations and the computation cost) is not evaluated in previous works. Investigating this relationship is important to enable systematic methods for designing efficient error-aware DNN accelerators.

Notably, the comprehensive analysis of the impact of bit-level datapath errors on inference accuracy, as presented in this work, can be leveraged to improve previously proposed error mitigation and energy optimization techniques. For example, the dynamic voltage scaling methodology proposed in [61] can be enhanced by considering per-layer error resilience due to timing errors in the computational blocks of neural networks. Additionally, our analysis of the high error sensitivity

of the higher order bits and the impact on inference accuracy observed through per-bit error resilience analysis can be considered when applying error detection and recovery techniques, such as TE-Drop in [52], and median feature selection in [69].

6.2 Error Resilience Analysis Framework

This section describes the error resilience analysis framework used to quantify the inference accuracy with respect to bit-level datapath errors in the compute-intensive convolutional and fully-connected layers. The framework, implemented in PyTorch, takes as inputs a pre-trained neural network model and the parameters which define the quantization and error rates applied to different layers (per-layer analysis) and bits (per-bit analysis) of the neural network. Based on the inputs, the neural network is quantized and the inference accuracy is evaluated by injecting the bit-level errors. With this analysis, the DNN accuracy can be evaluated with respect to different error rates and locations to determine the per-layer and per-bit error sensitivity of the network. The framework is summarized in Fig. 6.1 and discussed in detail in the following subsections.

6.2.1 Inputs

The following classes of inputs can be specified to configure the error resilience analysis framework and define different data precision and error rates considered in the analysis:

- *Quantization* - The framework allows the user to evaluate the inference accuracy by quantizing the weights and forward activations for each layer to different bit precisions, which are defined as an input to the framework.

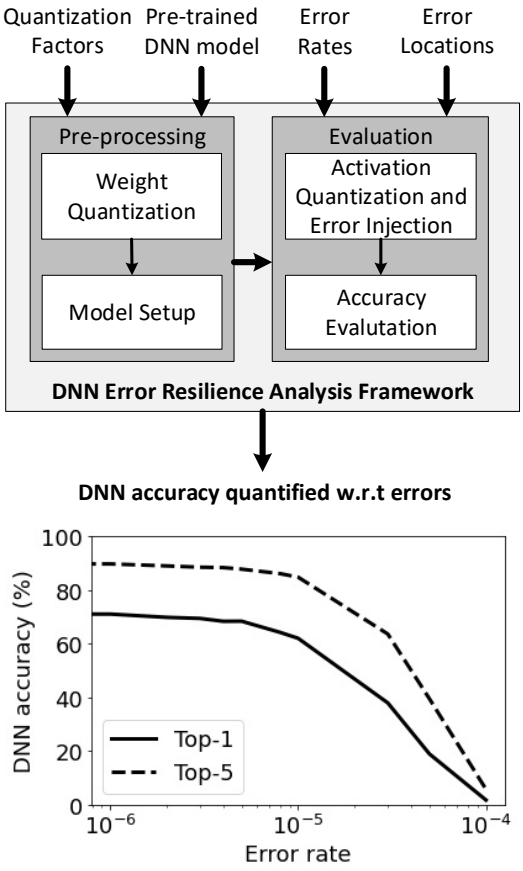


Figure 6.1: DNN error resilience analysis framework methodology.

- *Error rates* - The bit-wise errors injected in the convolutional and fully-connected layers of the network are based on the error rate specified as an input to the framework. Different error rates can be defined for different error locations, as described below.

- *Error locations* - The user can specify error location sites at per-layer or per-bit granularity. When performing per-layer error analysis, users can flag specific layers in the network to inject errors, or they can provide a set of layer

characteristics (such as input and output feature size, or group size) to inject errors in all matching layers. When performing per-bit error analysis, the user can define different error rates for each individual data bit.

6.2.2 Model Setup

Prior to error resilience evaluation, the PyTorch model of the DNN is setup for quantization and error injection. Our framework performs linear quantization in PyTorch based on [70], which quantizes the weights and activations (layer outputs) to the given bit precision. First, a preprocessing step quantizes the DNN's pre-trained weights and stores them in the desired fixed-point format. However, quantizing activations is more complex because the values to be quantized are only known during inference, so they cannot be similarly preprocessed. To allow flexible modeling of the quantization of forward activations during inference, the framework first computes the activations in floating point format using unmodified PyTorch operators, and then quantizes them to reduced precision fixed-point format. This is accomplished by adding a new layer type called *LinearQuant* after each convolutional and fully-connected layer. The LinearQuant layer takes as input floating point activation values, and it outputs these values quantized to the desired fixed-point format. In this way, the layer models bit-accurate fixed-point computation.

After quantization, the model is configured for error injection to allow the addition of errors at the desired rate in the specified locations. To model the errors in forward activations during inference evaluation, code must be added to the output of each layer to probabilistically inject errors (at per-bit granularity) to the quantized activations at the output of the convolutional and fully-connected layers. This is implemented in our work by replacing the *LinearQuant* layer described above with

a *QuantAndError* layer, which implements linear quantization (as before) followed by the injection of errors. Probabilistic error injection is realized by randomly flipping the original data bits based on the error rate provided by the user. To improve runtime, the error injection can be performed on an entire tensor of activation feature maps instead of one value at a time. For a given error rate, an error tensor is generated to identify the bit positions of the activations which will become error sites. The error tensor has the same dimensions as the tensor of the layer's output activations, but it is populated with 1s and 0s. For example, if a layer has a uniform error rate of 0.1, then 10% of the error tensor's values will be '1'. The respective bit positions in the original data tensor are then flipped to get the faulty data tensor. Once the inputs are defined and the neural network is setup to implement quantization and error injection, the accuracy can be evaluated through inference. The following section details the experiments that evaluate the effect of bit-level errors on different networks, applied at different granularities with different error rates.

6.3 Results

The methodology described in the previous section is used to quantify the classification accuracy with respect to errors for different DNNs (ResNet18 [63], MobileNetV2 [64], EfficientNet-B0, B2, B4, B6 and B8 [65]). As described previously, the classification accuracy of the neural networks is evaluated at reduced precision with random error injection in the convolutional and fully-connected layers. The key observations of this analysis are summarized below:

- For different networks, DNN error sensitivity does not depend on the number of parameters. For instance, MobileNetV2 has lower accuracy and approx-

imately 3–5× fewer parameters compared to EfficientNet-B2 and B4 networks, but the three networks show similar loss in accuracy with increasing error rates. However, when comparing MobileNetV2 and baseline EfficientNet (B0), the former is 1.62× more error tolerant, even though the classification accuracy of EfficientNet-B0 is approximately 3% better and has 1.5× more number of parameters. On the other hand, for EfficientNet networks, as the number of parameters and network size increase with scaling, the error resilience of the network improves along with the inference accuracy.

- The error resilience of the convolutional layers in a neural network can vary by orders of magnitude. The last convolutional layer is typically the most error resilient. For example, for EfficientNet-B6, the last convolutional layer is approximately 86× more error resilient compared to the first convolutional layer.

- Evaluating the error resilience of neural network layers and analyzing it with respect to the layer characteristics, such as the number of arithmetic operations (additions and multiplications) required to compute a layer output (*MOps*) and number of operations per output point (*Ops/output*), we observe that there is a lack of correlation between the computational cost and error sensitivity of a layer. Some layers with higher *MOps* are also more error tolerant. For instance, the first and last convolutional layer in ResNet-18 have the same *MOps* but the latter is 50× more error resilient. Through this analysis, the error sensitivity of the layers can be quantified with respect to *MOps* and can be used to identify the layers which are error-resilient as well as compute-intensive. These robust layers can be leveraged to design low-power DNN hardware.

- For all the networks evaluated in the per-layer error resilience analysis experiments, the fully connected layer is more error sensitive compared to the convolutional layers. For instance, the first convolutional layer in MobileNetV2 can tolerate 5.2× higher error rate as compared to the fully-connected layer.

- The framework is used to evaluate a subset of the networks at per-bit granularity to understand the improvement in the inference accuracy if some of the higher significant bits have no errors. For MobileNetV2 network, the error resilience of the network drastically improves 2–244× when 1–5 higher significant bits are error free and the same error rate is applied to the remaining bits. This framework can therefore be helpful to understand the trade-offs among error detection and correction, inference accuracy improvements, and energy efficiency.

The neural networks used in this work are summarized in section 6.3.1. These networks are setup and configured for error resilience analysis experiments, as described in section 6.3.2. The error resilience of the networks due to errors at network-level, per-layer and per-bit granularity are analyzed in section 6.3.3.

6.3.1 Neural Networks

The DNNs considered in this analysis have varying sizes and characteristics. The number of model parameters and the respective inference accuracies are summarized in Table 6.1. These networks are chosen because they employ a variety of architectural techniques to improve efficiency and accuracy. For instance, ResNet models [63] implement a shortcut connection to convert a plain network to the residual counterpart to improve the classification accuracy. In [64], Sandler, et

al. introduced MobileNetV2, which is based on an inverted residual structure with linear bottlenecks. In this network, the layer modules take a low-dimensional compressed input, which is first expanded to high dimension before applying depthwise convolutions. The features are then compressed back to low-dimensional representation with linear convolution. This network decreased the number of operations and memory needed, which improved the performance of the mobile models significantly. In [65], a compound scaling methodology was proposed, which provided a drastic improvement in accuracy over the existing convolutional neural networks. In this methodology, the depth, width and resolution of the network are scaled uniformly using an effective *compound coefficient*. Through this methodology, the authors propose a new baseline network which is scaled up to a family of models called *EfficientNets*. The EfficientNet architecture is similar to the MNASNet [71] and MobileNetV2 [64], where the features are expanded before applying depthwise convolutions and compressed back to low-dimensions in the bottleneck modules. Analyzing these different networks using the error resilience analysis framework provides better understanding of the impact of errors on the inference accuracy and also enables better convergence of the key observations through error resilience analysis experiments.

6.3.2 Experimental Setup

This section describes the network configuration and setup performed prior to error resilience analysis. First, the baseline accuracy is first determined for a given neural network, pre-trained on the ImageNet dataset [66]. The network is then quantized to a given fixed-point bit precision, such that the classification accuracy of the network is within $\sim$1% of the baseline accuracy. The baseline accuracy

DNN	Parameters	Baseline Top-1 acc.	Quantized network	
			Bits	Top-1 acc.
ResNet-18	11.5M	69.25%	10	69.23%
MobileNetV2	3.5M	72.59%	10	71.73%
EfficientNet-B0	5.3M	75.61%	12	75.60%
EfficientNet-B2	9.2M	80.12%	12	80.04%
EfficientNet-B4	19M	82.67%	10	82.15%
EfficientNet-B6	43M	84.64%	10	84.14%
EfficientNet-B8	81M	85.65%	10	85.51%

Table 6.1: Impact of quantization on classification accuracy.

of the original pre-trained networks and the quantized networks is summarized in Table 6.1.

Evaluating the DNN accuracy with quantization adds a small runtime[1] overhead of 1.08–1.97×. However, this runtime increases by an average of 37.28× when evaluating the inference accuracy with quantization and error injection. For a large network, such as EfficientNet-B4 (19M parameters), the runtime can increase from 2.4 hours (for evaluating the baseline accuracy) to 4.5 days (for evaluating the accuracy with respect to one error rate) when considering the entire validation dataset. To achieve manageable runtimes while measuring many error rates, the accuracy is evaluated for a random selection of 5000 samples (10%) from the validation dataset. The same random 5000 images are used for each evaluation to ensure consistency across experiments. To confirm that the subset of images yields representative results, Fig. 6.2 evaluates the difference between the inference accu-

[1] All runtimes described here are performed by measuring the time required for inference using the ImageNet validation dataset on a server with an Intel Xeon E5-2683v3 CPU with 28 cores, 2 GHz base operating frequency, and 128 GB DDR4 memory (133.25 GB/s bandwidth).

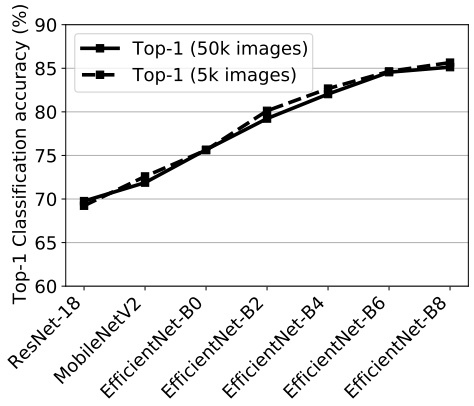


Figure 6.2: Inference accuracy of different neural networks, considering ImageNet dataset.

racy using the entire validation dataset and the selected random sample. The mean difference in the Top-1 inference accuracy is 0.32%, but evaluating the accuracy with the reduced dataset improves the runtime significantly: for EfficientNet-B4, using 10% of the dataset reduces the runtime without quantization and error injection from 2.4 hours to 15.6 minutes, and with quantization and error injection, the runtime decreases from 4.5 days to 11.6 hours.

6.3.3 Error Resilience Analysis

The following three scenarios are considered to evaluate the DNN error resilience:

- Network-level error analysis, where the same error rate is applied to all bits in all convolutional and fully-connected layers.
- Per-layer error analysis, where the same error rate is applied to select layers in the network.

- Per-bit error analysis, where some of the higher significant bits are error-free while the same error rate is applied to the remaining bits in all the convolutional and fully-connected layers.

This analysis uses two metrics to quantify network behavior in the presence of errors: Top-1 classification accuracy and error resilience. First, the Top-1 classification accuracy shows the overall accuracy of a network with the given error rate(s). Although this captures the application-level correctness of the algorithm, it does not allow a direct comparison of the resilience of different networks (or portions of networks) in the presence of errors. For example, one may observe that some layers can tolerate higher levels of errors than others while yielding the same overall accuracy. Second, error resilience can be quantified using the metric Err_T, where T is the loss in application level accuracy. Err_T represents the error rate at which the Top-1 inference accuracy drops by T. For instance, $T = 1\%$ indicates that the DNN inference accuracy is one percentage point lower than the accuracy of an error-free quantized network, and $Err_{1\%}$ gives the error rate that yields this accuracy. Thus, a higher Err_T implies a more error tolerant network or layer. The results presented below quantify the DNN error resilience using $T = 1\%$ and $T = 0.5\%$, but other values of T could be chosen to match other desired accuracy requirements.

6.3.3.1 Network-level error resilience analysis

Fig. 6.4 illustrates the Top-1 classification accuracy of the quantized, pre-trained networks with respect to different error rates. The same error rate (shown on the x-axis) is applied to all the layers of the network in this analysis. The neural networks exhibit a relatively weaker dependence on error until a certain error rate ($\leq$ 0.001%), beyond which the accuracy degrades drastically. For EfficientNet net-

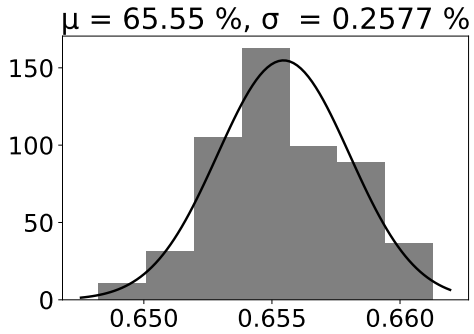
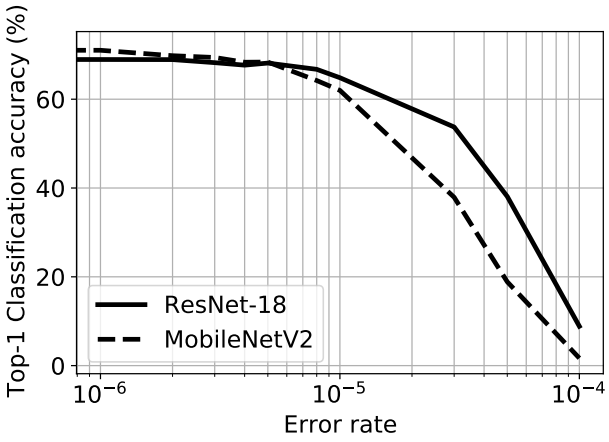


Figure 6.3: Distribution of Top-1 inference accuracy obtained over 100 runs with random error injection at 0.001%.

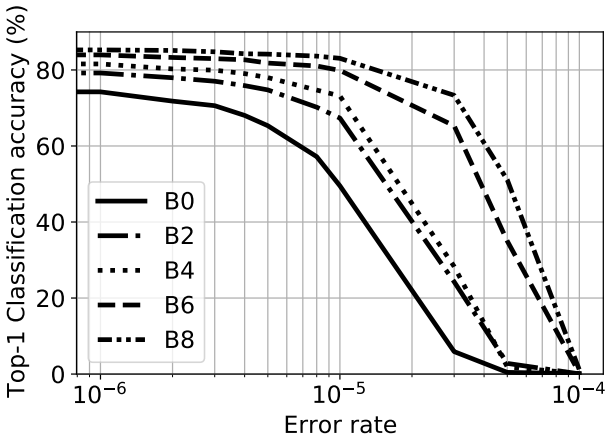
works, the accuracy with respect to errors improves as the network scales from B0 to B8 such that EfficientNet-B8 has higher inference accuracy at a given error rate compared to other DNNs considered in this analysis.

Variability in accuracy: Due to random injection of errors in this analysis, there is a variation in the inference accuracy if the same error rate is applied to the entire network in multiple runs. This randomness is evaluated in ResNet-18 network, by determining the accuracy at 0.001% error rate over 100 runs. The distribution of Top-1 accuracy obtained has a standard deviation of 0.2577% which is small compared to the mean inference accuracy of 65.55% (Fig. 6.3). The accuracy of ResNet-18 at 0.001% error rate in Fig. 6.4(a) is 64.84% which lies at –2.63σ in the distribution. Note that the variation reduces at lower error rates such that σ for 0.0001% and 0.00001% error rates is 0.2115% and 0.096% respectively.

Further evaluating the error resilience of the networks, we observe that even though a subset of the networks have comparable error resilience, one network can have better inference accuracy or lower number of parameters compared to the other. This trade-off between the network characteristics and error resilience is further investigated. For example, the error resilience of the networks, quantified as



(a) ResNet-18 and MobileNetV2



(b) EfficientNet architectures B0, B2, B4, B6, and B8

Figure 6.4: Top-1 classification accuracy with respect to different error rates for (a) ResNet-18, MobileNetV2, and (b) EfficientNet networks. Note that EfficientNet-B0 and B2 networks are quantized to 12 bits, while the other networks are quantized to 10 bits.

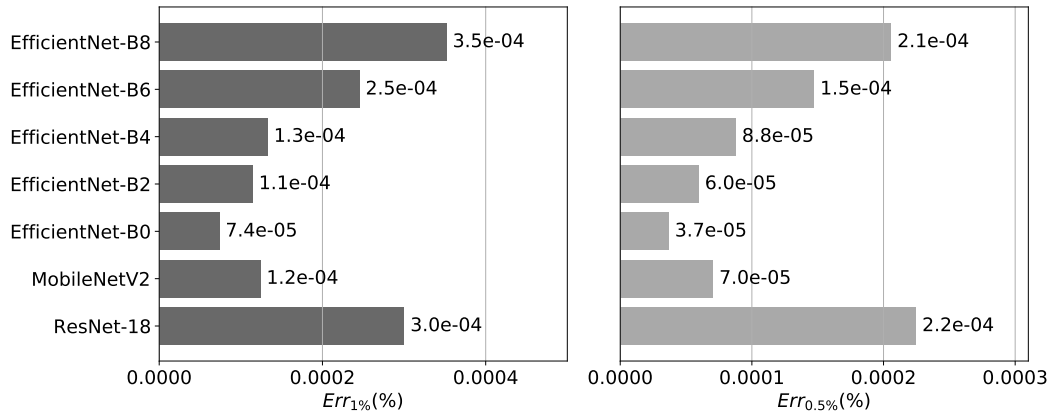


Figure 6.5: $Err_{1\%}$ and $Err_{0.5\%}$ of different quantized, pre-trained neural networks.

$Err_{1\%}$ and $Err_{0.5\%}$, is compared in Fig. 6.5. Overall, ResNet-18 and EfficientNet-B8 networks exhibit higher error resilience as compared to other networks, while EfficientNet-B0 has the lowest error resilience. ResNet-18 and EfficientNet-B8 show comparable $Err_{0.5\%}$, which is 5.7–1.47$\times$ higher than other networks. Comparing $Err_{1\%}$, ResNet-18 has 1.2$\times$ lower error tolerance compared to EfficientNet-B8, while it is 1.2–4.05$\times$ more error tolerant compared to other networks. Even though ResNet-18 has better error resilience, it has lower inference accuracy and 1.25–3.3$\times$ more parameters compared to MobileNetV2 and EfficientNet-B0 and B2 networks (Table 6.1). We observe that smaller EfficientNets (such as B0 and B2) are relatively more sensitive to errors, but the error resilience of the EfficientNet networks considerably improves with scaling, such that the $Err_{1\%}$ of B6 and B8 networks is 1.85–3.1$\times$ higher compared to B2 and B4 networks. MobileNetV2 and the EfficientNet-B2 and B4 networks have comparable $Err_{1\%}$ ($\approx 0.0001\%$), even though the EfficientNet networks have almost 2.6–5.4$\times$ more parameters and 7.5–10% better inference accuracy. These observations highlight the trade-offs between inference accuracy, network characteristics and error resilience, which are

complex and difficult to predict, and therefore, are important to analyze through these experiments.

6.3.3.2 Per-layer error resilience analysis

To understand the impact on classification accuracy due to errors in select layers of the neural network, the error resilience of individual layers is also evaluated in this work. Different layers in a neural network exhibit different computational complexity, which depends on the number of parameters and output feature size. This computational complexity of a network layer is represented as *MOps* (total number of arithmetic operations), while the computation complexity of an output point is represented as *Ops/output* (number of operations per output point). The following experiment highlights two key observations. First, there is a high degree of variability in the error resilience of different layers in a network. Second, there is no correlation between a layer's error resilience and computation cost of the network layer. Through this analysis, a network's layers that are most error resilient and compute-intensive can be identified; this observation can then be leveraged toward efficient low-power DNN hardware design. The per-layer error resilience analysis experiments for different networks are discussed below.

- *ResNet-18* - The ResNet-18 model uses four basic building blocks, with four convolutional layers in each block. The four convolutional layers in a given block x are represented as Conv$x$_1 to Conv$x$_4. Table 6.2 gives the error resilience of three representative convolutional layers and the fully-connected layer from ResNet-18. The three convolutional layers are chosen from different building blocks, and each represents a different layer depth and significantly different computation cost and complexity. Comparing the $Err_{1\%}$

Layer	MOps	Ops/output	$Err_{1\%}(\%)$	$Err_{0.5\%}(\%)$
conv1_1	231.21	1152	0.0073	0.0052
conv3_1	115.61	2304	0.0528	0.0225
conv4_4	231.21	9216	0.4216	0.2155
Fully-connected	1.02	1024	0.0014	0.0008

Table 6.2: ResNet-18 per-layer error analysis

metric, the last convolutional layer (conv4_4) is 57.7× and 8× more error resilient as compared to the conv1_1 and conv3_1 layers, respectively. In addition, this layer is also highly compute-intensive, with 8× more *Ops/output* compared to conv1_1 layer and 2× more *MOps* compared to conv3_1 layer. Additionally, we observe that the fully-connected layer is much more sensitive to errors as compared to the convolutional layers. The $Err_{1\%}$ and $Err_{0.5\%}$ of the fully-connected layer are only 0.0014% and 0.0008%, which is 5.2× and 6.5× lower than the respective error resilience of the conv1_1 layer.

- *MobileNetV2* - For MobileNetV2, six convolutional layers with different characteristics are evaluated out of the 52 convolutional layers in the network. The layer depth, output shape and convolutional type (pointwise or depthwise) are described in the first column of Table 6.3. Note that convolutional layers 11, 25, 35 and 45 are part of the linear bottleneck modules in the network. Layer 25 is the expanding convolutional layer in the respective bottleneck while layer 45 is the layer where features are compressed back to low-dimensional representation. These layers represent varied characteristics of the convolutional layers and show variability in the error resilience across different layers in the neural network. For instance, layer 25 is more error resilient

Layer Description	MOps	Ops/output	$Err_{1\%}(\%)$	$Err_{0.5\%}(\%)$
Layer 1 $112 \times 112 \times 32$	21.68	54	0.0091	0.0021
Layer 11 $28 \times 28 \times 144$ Depthwise Conv	2.03	18	0.0146	0.0058
Layer 25 (Pointwise Conv) $14 \times 14 \times 384$	9.63	128	0.1679	0.0583
Layer 35 (Depthwise Conv) $14 \times 14 \times 576$	2.03	18	0.0153	0.0097
Layer 45 (Pointwise Conv) $7 \times 7 \times 160$	15.06	1921	0.0067	0.0027
Layer 52 (Pointwise Conv) $7 \times 7 \times 1280$	40.14	640	3	0.9182
Fully-connected	2.56	2561	0.0014	0.0008

Table 6.3: MobileNetV2 per-layer error analysis

compared to layer 11 (11.5× for $Err_{1\%}$ and 10× for $Err_{0.5\%}$) and is also more compute intensive with 4.7× more *MOps* and 7.1× more *Ops/output*. Similar to ResNet-18, the last convolutional layer has significantly higher error resilience (17.8–448.5×), and is also more compute intensive with higher number of operations (2–20×) compared to the other convolutional layers in the network. The fully-connected layer has the same error resilience as ResNet-18 and is more sensitive to errors compared to the convolutional layers. The $Err_{1\%}$ of the fully-connected layer is 4.8–2142.86× lower than the convolutional layers.

- *EfficientNet* - The error resilience of different layer structures is compared in Table 6.4 ($Err_{1\%}$) and Table 6.5 ($Err_{0.5\%}$) for EfficientNet networks B0, B2,

B4, B6, and B8. Note that when computing the error resilience of pointwise (or depthwise) convolutional layers, the same error rate is applied to *all* the pointwise (or depthwise) convolutional layers.

Similar to MobileNetV2 and ResNet-18, the last convolutional layer of EfficientNet is significantly more error resilient and computationally intensive than the other layers in the network. Compared to the first convolutional layer, the *MOps* and *Ops/output* of the last convolutional layer is 1.85–3.98× and 11.85–26.07× higher, respectively. Similar behavior is observed when comparing the pointwise and depthwise convolutional layers. The pointwise convolutional layers have higher *MOps* (9.86–25.74×) and *Ops/output* (3.92–10.75×) as compared to all the depthwise convolutional layers in the network, and are also 1.5–4× more error resilient.

Lastly, we evaluate how the error resilience of the EfficientNet networks changes as the networks scale up (from the smallest B0 to the largest B8). Comparing the $Err_{1\%}$ and $Err_{0.5\%}$ of these networks, the error resilience of the convolutional layers improves from B0 to B8. The fully-connected layer however has approximately the same error resilience across all EfficientNet networks i.e. the $Err_{1\%}$ is $\approx 0.002\%$ and $Err_{0.5\%}$ is $\approx 0.001\%$. Relative to the first convolutional layer, the fully-connected layer has approximately the same error resilience for B0–B4 networks, while it is more sensitive to errors for B6 and B8.

6.3.3.3 Per-bit error resilience analysis

The per-bit error resilience of three DNNs is analyzed to better understand the error sensitivity of the higher order bits. In this analysis, a given error rate is applied to the lower order bits while a number of the higher order bits are held error-free

Layer	B0	B2	B4	B6	B8
First Conv layer	0.0018	0.0026	0.0023	0.0124	0.0068
Last Conv layer	0.1746	0.1908	1.09	1.067	1.1623
All Pointwise Conv layers	0.0003	0.0006	0.0005	0.0011	0.0012
All Depthwise Conv layers	0.0001	0.0002	0.0002	0.0003	0.0009
Fully-connected layer	0.0018	0.002	0.002	0.0022	0.0019

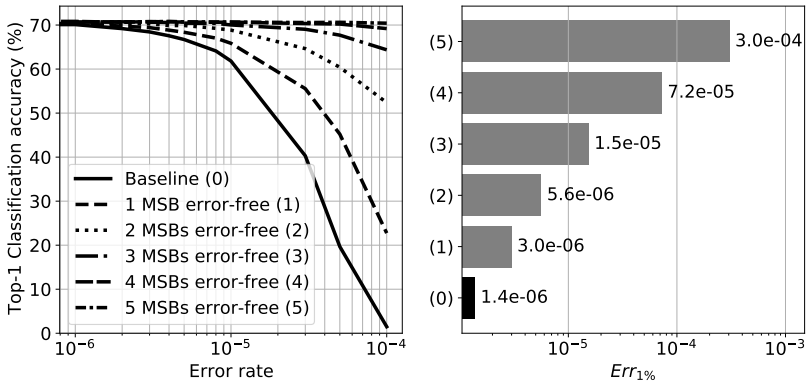
Table 6.4: $Err_{1\%}(\%)$ of different layers in EfficientNet networks.

Layer	B0	B2	B4	B6	B8
First Conv layer	0.0012	0.0009	0.0008	0.006	0.0032
Last Conv layer	0.0466	0.0804	0.3113	0.898	0.9314
All Pointwise Conv layers	0.0002	0.0005	0.0002	0.0002	0.001
All Depthwise Conv layers	0.0001	0.0001	0.0001	0.0002	0.0002
Fully-connected layer	0.001	0.0009	0.0008	0.0013	0.001

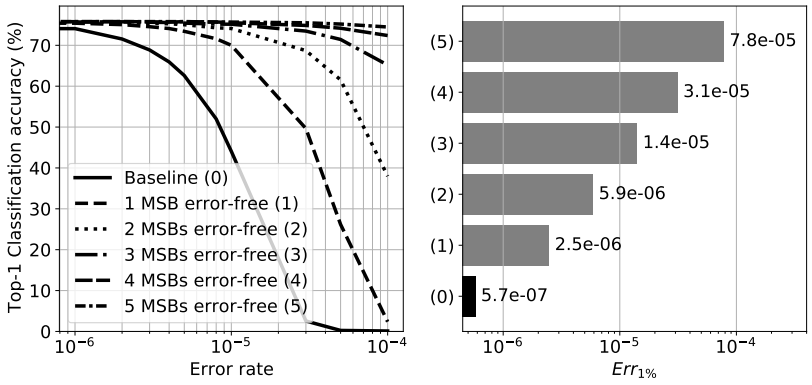
Table 6.5: $Err_{0.5\%}(\%)$ of different layers in EfficientNet networks.

across all layers. This experiment is conducted to investigate the potential benefit of using error correction mechanisms on select higher order bits. Fig. 6.6 illustrates the accuracy versus error rate and $Err_{1\%}$ for MobileNetV2, EfficientNet-B0, and EfficientNet-B4, when 0 to 5 most significant bits are error-free. For example, as shown in Fig. 6.6(a), MobileNetV2 (quantized to 10-bits) exhibits 2.14$\times$ higher error tolerance when a single bit is error-free, and 51.43$\times$ higher when four bits are error-free (with respect to $Err_{1\%}$). Beyond this point, the improvement in accuracy relatively saturates.

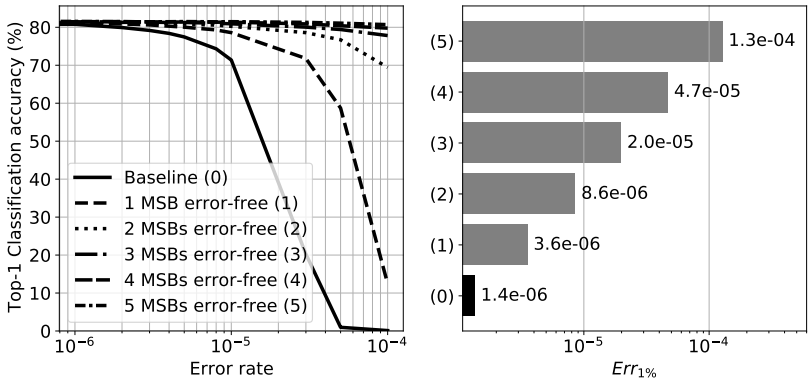
The per-bit analysis results for EfficientNet networks B0 and B4 are also illustrated in Fig. 6.6(b) and Fig. 6.6(c), respectively. The error resilience of both



(a) MobileNetV2 (10 bit quantization)



(b) EfficientNet-B0 (12 bit quantization)



(c) EfficientNet-B4 (10 bit quantization)

Figure 6.6: Per-bit error resilience analysis for EfficientNet-B0 and B4 networks. It should be noted that for no errors in MSBs, the remaining bits have the same error rate.

networks improves faster than MobileNetV2 when 1 to 3 higher significant bits are error-free. For instance, the $Err_{1\%}$ increases by 2.1× for MobileNetV2, and by 4.4× and 2.6× for EfficientNet B0 and B4, respectively, when the most significant bit has no error. According to the accuracy vs. error rate graph, the improvement in the accuracy of EfficientNet B4 network saturates relatively faster. When five higher order bits are made error-free, the improvement in the error sensitivity of MobileNetV2 is the highest. For instance, $Err_{1\%}$ of MobileNetV2 increases by 214.3×, while the error resilience of EfficientNet B0 and B4 increase by 136.8× and 92.8×, respectively. Overall, with the per-bit error resilience experiments, we observe that correcting the errors in 3 to 5 higher order bits (out of a total of 10 to 12 bits) improves the inference accuracy of the neural network, making it comparable to the baseline accuracy with no errors, particularly at low error rates.

6.4 Conclusion

Neural networks have an inherent tolerance to errors which can be exploited to develop efficient, low-power DNN accelerators. It is however important to understand the error resilience of the neural network at different levels of granularity, such as per-layer and per-bit error resilience analysis. This work develops a framework to quantify the inference accuracy of DNNs with respect to bit-wise datapath errors. The per-layer error resilience analysis demonstrates some layers (e.g. the fully connected layer) are very error sensitive, but that other layers (e.g. the last convolutional layer) tends to be more error resilient while also requiring a high number of *MOps*. The identification of convolutional layers that are both compute intensive and robust can enable smarter energy optimization methods, such as per-

layer voltage scaling that is more aware of each layer's error tolerance. Similarly, per-bit error resilience analysis can be helpful in (1) determining the number of most significant bits that need to be corrected to ensure the desired accuracy, and (2) understanding the trade-offs among error correction overhead, accuracy, and efficiency. These observations can be leveraged to develop efficient error-aware, low-power DNN hardware accelerators.

Chapter 7

Quantifying Accuracy and Energy Efficiency Trade-offs in Systolic Arrays

The DNN error resilience analysis, discussed in the previous chapter, highlights the varying degree of error sensitivity within a neural network. It is observed that some layers in the network are more error resilient and have high computation cost (higher *MOps*) as compared to other layers. These error resilient and compute intensive layers can be the key to implement efficient energy optimization techniques in deep learning hardware. The objective of this chapter is to quantify the trade-offs between inference accuracy of the neural network and energy consumption at different operating voltages. The methodology implemented to achieve this objective is summarized in Fig. 7.1. As shown in the figure, the inference accuracy with respect to different voltages is computed in two steps. First, the per-bit timing error rates of a processing element are computed for different operating voltages using

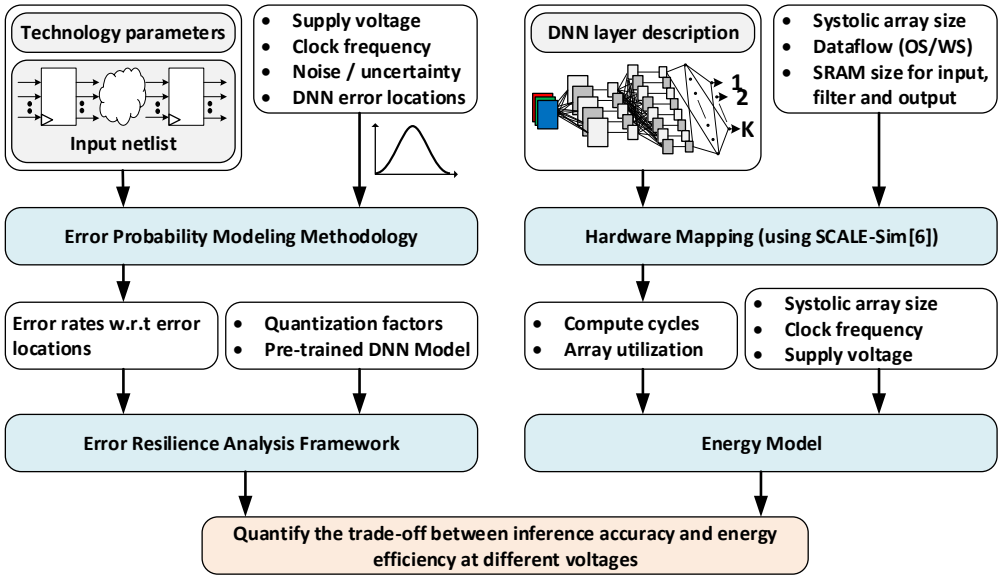


Figure 7.1: Methodology to quantify the trade-off between DNN accuracy and energy consumption at different operating voltages.

the methodology proposed in chapter 4. These error rates are then injected in a given neural network using the error resilience framework described in chapter 6 to evaluate the inference accuracy as a function of supply voltage. Concurrently, the per-layer energy consumption of the neural network is determined by first mapping the network onto a systolic array (using SCALE-Sim [72]) to determine the compute cycles and hardware utilization when computing the activations in the convolutional and fully-connected layers. This utilization is then used to evaluate per-layer energy cost at different operating voltages. Finally, the inference accuracy and energy consumption determined at different voltages are used to quantify the respective trade-offs.

The rest of the chapter is organized as follows. Section 7.1 describes the mapping of a neural network onto a systolic array to determine the per-layer array uti-

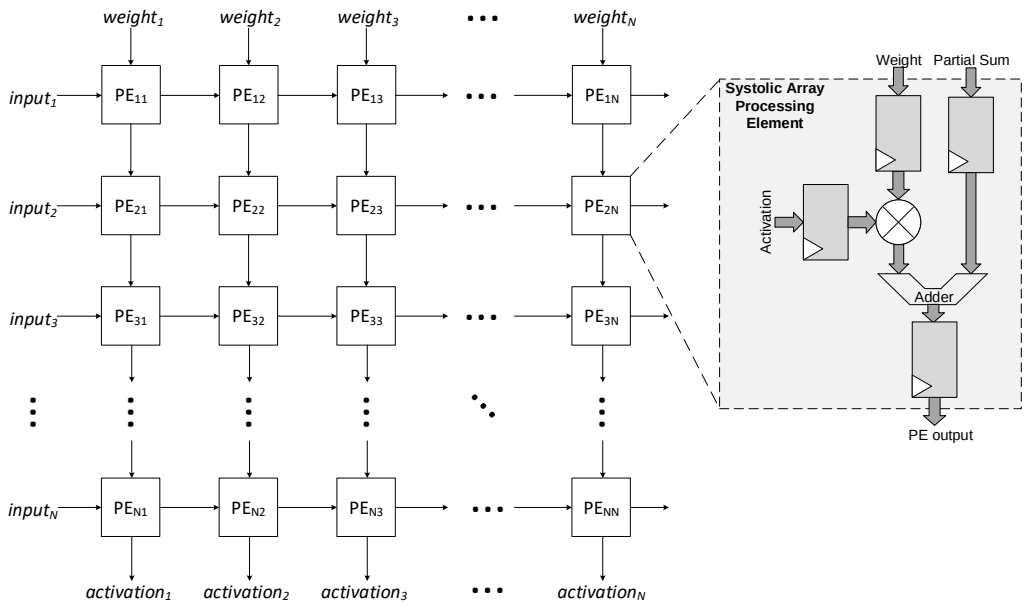


Figure 7.2: Systolic array block diagram.

lization and compute cycles using SCALE-Sim. The utilization percentage and runtime is used to compute the per-layer energy and power consumption by using an energy model, as described in section 7.2. The trade-off between inference accuracy and energy consumption at different operating voltages is evaluated through network-level voltage scaling (section 7.3) and per-layer voltage scaling (section 7.4). Finally, the results are discussed in section 7.5.

7.1 DNN to Systolic Array Hardware Mapping

A systolic array comprises of a number of multiply-add units (also called processing elements or PEs) which perform the multiplications and accumulations to compute the activations at the output of a neural network layer. A basic block

diagram of the layout of the systolic array and basic architecture of the PE is illustrated in Fig. 7.2. As shown in the figure, the inputs and weights traverse through the systolic array while the PEs compute the activations. There are a number of ways these computations can be mapped onto a systolic array, as described in [17]. These different mappings, called dataflows, can be categorized into weight stationary, output stationary and input stationary, depending on the data reuse patterns. These dataflows, as described below, determine the order in which input feature maps and weights are fed in the array, and the intermediate partial sums are stored and reused:

- *Output stationary* (OS) dataflow refers to a mapping where each processing element computes each pixel of the output feature map. Each input and weight required to compute an output pixel are streamed per cycle and accumulated in place in the respective PE. Once an activation is generated, the result is transferred to the memory and the respective PE starts computing the next output pixel.
- *Weight stationary* (WS) dataflow represents a mapping in which weights are mapped onto a systolic array and held in place while the inputs are fed into the array in every cycle to be multiplied with the respective weights. The partial sums are accumulated across multiple processing elements in a given systolic array column over multiple cycles. Once all the computations for a given set of weights are done, the mapping is repeated for the next set of weights and so on.
- *Input stationary* (IS) dataflow is similar to the WS dataflow except the input feature map is mapped onto a systolic array and kept in place while the weights are streamed into the systolic array.

Samajdar et al. [72] developed a cycle-accurate, systolic array based CNN accelerator simulator called SCALE-Sim (Systolic CNN Accelerator Simulator) which maps a neural network onto a systolic array based on the dataflow and array size specified by the user. The authors also validated SCALE-Sim by comparing the number of cycles obtained from the tool with RTL implementation of the systolic array with OS dataflow. The hardware mapping of a neural network layer onto systolic array is important to determine the number of compute cycles and array utilization for every layer in the DNN, which is later used in this work to estimate the energy consumption of systolic array. This analytical tool is used in this work to map a given neural network and determine the array utilization for both WS and OS dataflows. IS dataflow is not implemented in this work since it is similar to WS dataflow (input is held in place instead of weights) and it is observed that the OS and WS dataflows are more common in DNN accelerator structures [1, 17, 73–75]. Moreover, it is observed in [76] that the energy savings of IS dataflow are small compared to OS and WS dataflows. The methodology implemented to evaluate the compute cycles and utilization is described in the following sections.

7.1.1 Per-Layer Runtime Evaluation

When computing the convolution of two matrices in a systolic array, SCALE-Sim distributes the original operand matrices through spatial and temporal allocations, based on the dataflow specified by the user. For instance, the original operand matrices of size $M \times K$ and $K \times N$ are mapped to $S_R \times T$ and $T \times S_C$ respectively, where S_R and S_C are the spatial rows and columns along which computation is mapped and T denotes the temporal dimension. The tool then determines the number of cycles required to compute the output feature map (OFMAP) by considering

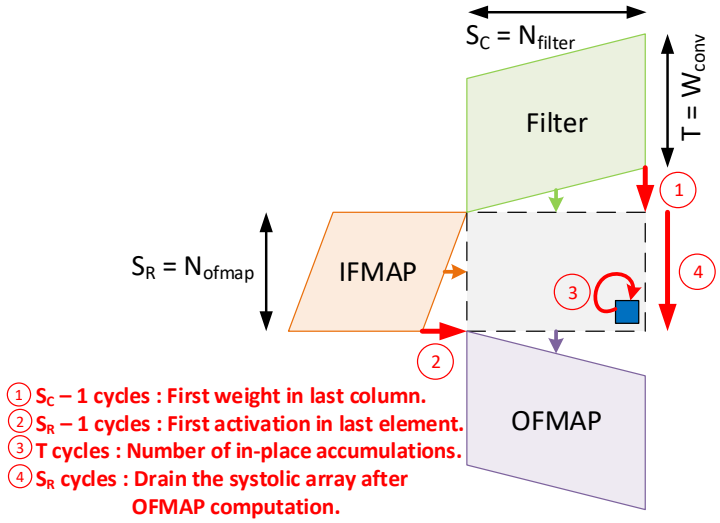
	Spatial rows (S_R)	**Spatial columns (S_C)**	**Temporal (T)**
OS	N_{ofmap}	N_{filter}	W_{conv}
WS	W_{conv}	N_{filter}	N_{ofmap}

Table 7.1: Spatial-Temporal allocation of DNN dimensions [72]. Note that N_{ofmap} is the number of OFMAP (output feature map) pixels generated by the filter, N_{filter} is the number of convolution filters and W_{conv} is the number of partial sums generated per output pixels.

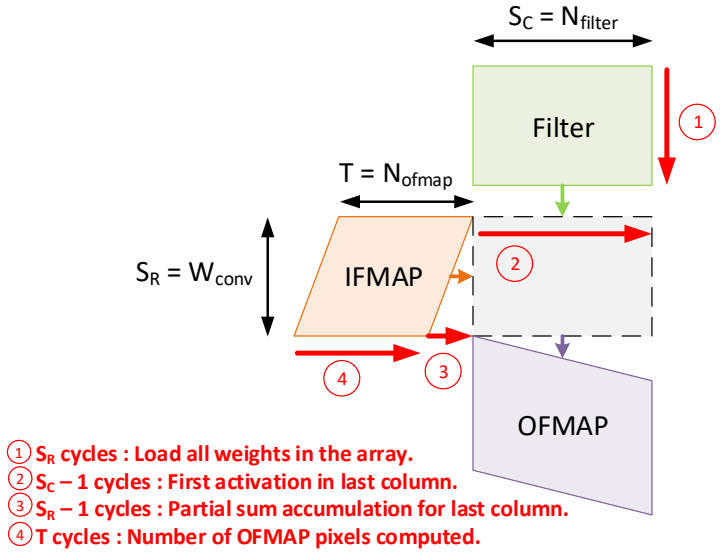
the two input operand matrices of size $S_R \times T$ and $T \times S_C$ (after spatial and temporal mapping). These spatial-temporal dimensions for *output stationary* (OS) and *weight stationary* (WS) dataflows are summarized in Table 7.1. Fig. 7.3 provides an overview of the runtime computation for the OS and WS dataflows. The input feature map (IFMAP) matrix and the filter matrix are fed from the left and top edge of the array, respectively, and the values propagate through the systolic array in a store and forward manner.

- In ***Output stationary (OS)*** (Fig. 7.3(a)) dataflow, each multiply-add unit in the systolic array is responsible for all multiplications and accumulations for a given OFMAP pixel. As observed in the figure, the last PE in the array (bottom right corner) receives first two operands in $S_R + S_C - 2$ cycles. It takes additional T cycles (same as the number of elements in a convolution window) to accumulate all the partial sums and compute the respective OFMAP pixel. After all the OFMAP pixels are computed, the array is drained from the bottom edge in S_R cycles.

- In ***Weight stationary (WS)*** (Fig. 7.3(b)) dataflow, filters are loaded in the array and kept in place while the IFMAP matrix is fed from the left edge of the array. It takes S_R cycles to load the filters in the array. Each PE multiplies



(a) Output stationary (OS) dataflow



(b) Weight stationary (WS) dataflow

Figure 7.3: Steps to compute runtime for dataflows in systolic array based on [72].

the IFMAP operand before forwarding the partial sum to the neighboring row for accumulation, while the IFMAP operand is forwarded to the neighboring column to compute the next OFMAP pixel. The first IFMAP operand arrives in the last column of the array in $S_C - 1$ cycles after which it takes $S_R - 1$ cycles to accumulate the partial sums. Each column of the array computes one OFMAP pixel and receives total of T operands.

Therefore, the total runtime for each layer for both OS and WS dataflow is computed as

$$2S_R + S_C + T - 2. \tag{7.1}$$

To validate the number of compute cycles obtained from SCALE-Sim and further understand the implementation, a processing element is designed in Verilog and implemented in a simple 4×4 systolic array. The array is simulated for both the weight stationary and output stationary dataflows to compute the convolution of a 4×4 IFMAP and 2×2 filter. The number of cycles to get the convolution output obtained from SCALE-Sim and Verilog simulation is observed to be the same.

The runtime computation described above [using (7.1)] considers the scenario when the array size can accomodate the maximum array size, $S_R \times S_C$. However, practically the array size is limited (e.g. 256×256 in [1]) and when mapping the computation for large neural network layers, the input operand matrices are divided into multiple *folds* by slicing along the S_R and S_C dimensions. In the scenario when $S_R > R$, where S_R is the number of OFMAP pixels (for OS dataflow) or the number of partial sums in the convolution window (for WS dataflow) and R is the number of rows in the systolic array, the OFMAP pixels are computed in multiple horizontal folds. In WS dataflow, each horizontal fold computes the partial sums

which are accumulated in an accumulator outside the systolic array to generate the corresponding OFMAP pixel. Similarly, if $S_C > C$, where S_C is the number of filters and C is the number of columns in the array, all output channels are computed in multiple vertical folds. When computing the total runtime for each vertical and/or horizontal fold, (7.1) is applied by replacing S_R and S_C with the number of rows and columns used in the respective fold of the array.

7.1.2 Array Utilization Computation

It is important to estimate the array utilization of a systolic array to determine the total power consumption when computing the activations in a neural network layer. The array utilization in SCALE-Sim is modeled by calculating the number of rows and columns used in the systolic array out of maximum available rows (array height) and columns (array width), across the total compute cycles. This is assuming that all the processing elements in a given row and column are used in each cycle [72]. However, practically due to store and forward nature of the array and the skewed data distribution, not all PEs are used for computation in a given cycle. The PEs used for multiplication and accumulation in a given cycle consume dynamic energy while the inactive PEs consume leakage energy. Note that this does not take into account power gating or other techniques implemented to reduce leakage. Based on the total number of cycles required to perform the total multiply-add operations in a given layer (obtained from SCALE-Sim), the array utilization for active PEs can be calculated as,

$$util = \frac{filter\_size \times OFMAP\_size}{array\_size \times total\_cycles}, \tag{7.2}$$

where total_cycles is the total cycle count for computing a layer output.

Considering the example of the first convolutional layer in MobileNetV2 with 3×3×3 filter matrix and OFMAP size 112×112×32, the total number of multiply-add operations in the systolic array required to compute all the outputs is $filter\_size \times OFMAP\_size = 10.838M$. The total number of multiply-add units available in the array for the total compute cycles is $array\_size \times total\_cycles$. Since it takes 12630 cycles to compute the output of a layer in a 256×256 systolic array, the utilization obtained from (7.2) is 1.309%.

Comparing the WS and OS dataflows, the array utilization increases (or decreases) at the same rate as the decrease (or increase) in the number of compute cycles. For instance, if an array has M multipliers and the array is 100% utilized, the number of cycles to compute N multiply-add operations is N/M. If the array is 50% utilized, the number of cycles to do the same computation is $2N/M$. To validate the utilization computed for WS and OS dataflows, the rate of increase in the compute cycles is compared with the rate of decrease in utilization and they are observed to be the same. These utilizations are then used to compute the power and energy consumption of the systolic array, as described in the following section.

7.2 Per-Layer Energy Modeling

The power and energy consumption in a systolic array, when performing inference of an input image, can be computed using the compute cycles and array utilization obtained for each neural network layer (as described in the previous section). The processing element is designed in HSPICE and simulated for a given technology to compute the average power consumption at different voltages (Table 7.2). Note that when computing the leakage power consumption of a process-

ing element, the clock signal is gated. The per-layer and neural network energy and power consumption are further described below.

Voltage (V)	Dynamic Power $P_{dyn(PE)}(\mu W)$	Leakage Power $P_{lkg(PE)}(\mu W)$
0.90	369.7	13.0
0.85	317.9	10.8
0.80	273.6	8.9
0.75	232.6	7.4
0.70	194.5	6.0
0.65	161.2	4.9
0.60	132.5	4.0
0.55	107.7	3.2
0.50	86.3	2.5
0.45	68.1	2.0
0.40	52.8	1.5

Table 7.2: Dynamic and leakage power consumption for a processing element at different voltages for 700 MHz clock frequency.

7.2.1 Per-Layer Energy Consumption

Given the power consumption of a processing element (Table 7.2), layer utilization ($util$) and array size, the per-layer dynamic and leakage power consumption can be evaluated as,

$$\begin{aligned} &\text{Dynamic Power, } P_{dyn} = P_{dyn(PE)} \times array\_size \times util, \\ &\text{Leakage Power, } P_{lkg} = P_{lkg(PE)} \times array\_size \times (1 - util). \end{aligned} \tag{7.3}$$

Given the total number of compute cycles for a layer, the energy consumption of a layer can be determined using the following equation,

$$E_{dyn}(\text{or } E_{lkg}) = \frac{P_{dyn}(\text{or } P_{lkg}) \times compute\_cycles}{f_{clk}}, \tag{7.4}$$

where f_{clk} is the operating clock frequency.

7.2.2 DNN Energy Consumption

The total energy consumption for a neural network when performing inference for an input image can be determined by summation of the per-layer energy consumption:

$$E_{DNN} = \sum_{n=1}^{N} E_n, \tag{7.5}$$

where N is the number of layers in the DNN and E_n is the energy consumption of layer n. The average power consumption of the entire network can be computed from the total number of compute cycles (obtained from summation of per-layer compute cycles) and the operating clock frequency:

$$P_{DNN} = E_{DNN} \times \frac{f_{clk}}{\sum_{n=1}^{N} cycles_n}. \tag{7.6}$$

The DNN energy and power consumption obtained analytically from this modeling methodology can be used to evaluate the trade-off between DNN accuracy and energy at different operating voltages.

7.3 Network-Level Voltage Scaling

In this section the DNN accuracy and energy consumption during inference are evaluated with respect to supply voltage scaling, where all layers in the neural network operate at the same voltage. For this analysis, the per-bit errors injected using the error resilience analysis framework are determined from the timing error probability methodology proposed in this work (chapter 3 and 4), as described in section 7.3.1. The inference accuracy is quantified with respect to different operating voltages and clock frequency in section 7.3.2.

7.3.1 Timing Error Probability for Systolic Array

The timing error probability for each bit at the output of an 8-bit MAC is described in chapter 5. Considering the example of EfficientNet-B8, since the network is quantized to 10 bits, a 10-bit multiplier is implemented in the systolic array processing element (PE), as shown in Fig. 7.4. Note that the 36-bit adder implemented in the processing element handles the additional guard bits required to avoid overflow when accumulating the partial sums. The number of guard bits depends on the number of accumulations (n_{acc}) required to compute each output of a DNN layer and can be calculated as $\log_2 n_{acc}$. The number of guard bits required for computation in all the layers is determined and the maximum value is considered to decide the adder bit-width. Note that for all the networks considered in chapter 6, only EfficientNet-B0 and B2 networks are quantized to 12 bits, so the processing element in Fig. 7.4 can be implemented for all the networks.

The error probability of the PE is computed in three steps. First, the timing delay for all the datapaths is obtained for 20nm HP FinFET technology through HSPICE simulations. Second, the error probability of different bits at the output of the PE

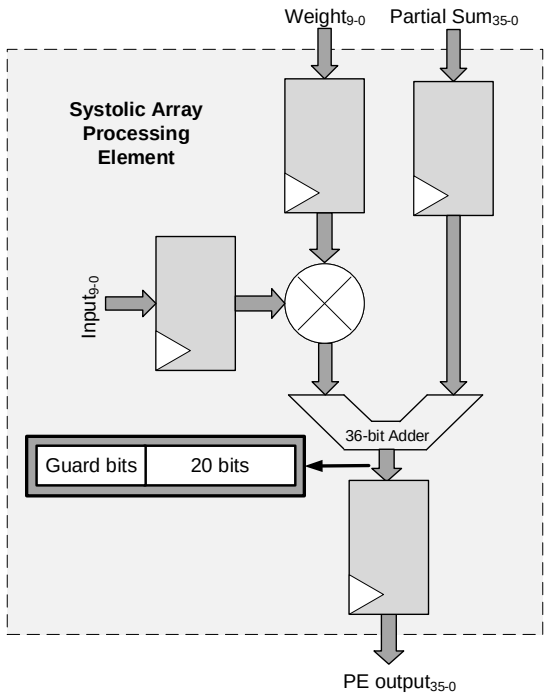


Figure 7.4: Basic block diagram of a processing element in a systolic array.

is evaluated from the voltage distribution by determining the voltage at which one or more datapaths fail timing. Finally, the per-bit accumulated timing error probability is obtained based on the number of multiply-add operations in the respective convolution window, which depends on the filter size for a given layer. Note that the timing error probability for both WS and OS dataflow is evaluated with respect to the total multiply-add operations to compute each output and is observed to be the same. Since the probability is independent of the dataflow implemented for the systolic array, the inference accuracy is evaluated with the same error probabilities for both dataflows.

The additional parameters required to compute the timing error probability is the voltage RMS noise, σ and *ClockPeriod-to-PathDelay* ratio. Based on the number of

Maximum bit-width	Delay (ps)	T_{clk}/d_{path}	
		$f_{clk} = 1GHz$	$f_{clk} = 700MHz$
23	454.47	2.200	3.143
24	461.94	2.165	3.093
25	469.40	2.130	3.043
26	476.87	2.097	2.996
27	484.33	2.065	2.950
28	491.80	2.033	2.905
29	499.27	2.003	2.861
30	506.73	1.973	2.819
31	514.20	1.945	2.778
32	521.66	1.917	2.738
33	528.47	1.892	2.703
34	535.94	1.866	2.666
35	543.40	1.840	2.629
36	550.87	1.815	2.593

Table 7.3: *ClockPeriod-to-PathDelay* ratio of different accumulator sizes.

accumulations for each output of a layer, the bit width of the adder output (including the guard bits) varies between 23 to 36 bits. The *ClockPeriod-to-PathDelay* ratio of different adder bit-widths are summarized in Table 7.3. The average *ClockPeriod-to-PathDelay* ratio for the processing element at 700 MHz and 1 GHz operating clock frequency is 2.851 and 1.996, respectively, while the noise (σ) considered for the voltage distribution when computing the timing error probability is 5%.

The per-layer, per-bit timing error probability computed from the model is pro-

Accumulator bit-width	0.9V	0.8V	0.7V	0.6V	0.5V
23	≈0%	≈0%	≈0%	≈0%	0.0004%
24	≈0%	≈0%	≈0%	≈0%	0.0017%
25	≈0%	≈0%	≈0%	≈0%	0.0047%
26	≈0%	≈0%	≈0%	≈0%	0.0222%
27	≈0%	≈0%	≈0%	≈0%	0.0959%
28	≈0%	≈0%	≈0%	0.0000001%	0.2430%
29	≈0%	≈0%	≈0%	0.0000001%	0.4183%
30	≈0%	≈0%	≈0%	0.0000003%	0.7080%
31	≈0%	≈0%	≈0%	0.0000006%	1.1775%
32	≈0%	≈0%	≈0%	0.0000011%	1.9246%
33	≈0%	≈0%	≈0%	0.0000021%	3.09%
34	≈0%	≈0%	≈0%	0.0000041%	4.8699%
35	≈0%	≈0%	≈0%	0.0000078%	7.5255%
36	≈0%	≈0%	≈0%	0.0000145%	11.3789%

Table 7.4: Timing error probability of MSB for different adder bits across all layers in EfficientNet-B8 with respect to different operating voltages at 700 MHz clock frequency.

vided as an input to the error resilience analysis framework (chapter 6) to quantify the DNN inference accuracy with respect to different operating voltages. The timing error probability for different most significant bit positions in different layers of EfficientNet-B8 is summarized in Table 7.4 and Table 7.5 for 700 MHz and 1 GHz clock frequencies, respectively. From the results, it is evident that higher operating frequency and lower operating voltages increase the timing error probability.

Accumulator bit-width	0.9V	0.8V	0.7V	0.6V	0.5V
23	≈0%	≈0%	≈0%	0.00005%	7.4564%
24	≈0%	≈0%	≈0%	0.0002%	23.089%
25	≈0%	≈0%	≈0%	0.0007%	45.4497%
26	≈0%	≈0%	≈0%	0.0041%	91.079%
27	≈0%	≈0%	≈0%	0.0211%	99.9842%
28	≈0%	≈0%	0.0000002%	0.0639%	100%
29	≈0%	≈0%	0.0000004%	0.1319%	100%
30	≈0%	≈0%	0.0000010%	0.2674%	100%
31	≈0%	≈0%	0.0000024%	0.5324%	100%
32	≈0%	≈0%	0.0000058%	1.0397%	100%
33	≈0%	≈0%	0.0000138%	1.9889%	100%
34	≈0%	≈0%	0.0000325%	3.7185%	100%
35	≈0%	≈0%	0.0000755%	6.7714%	100%
36	≈0%	≈0%	0.0001732%	11.9436%	100%

Table 7.5: Timing error probability of MSB for different adder bits across all layers in EfficientNet-B8 with respect to different operating voltages at 1 GHz clock frequency.

7.3.2 Quantifying DNN accuracy with respect to supply voltage

The per-bit timing error probabilities computed for all the layers in the network are injected in the DNN using the error resilience analysis framework and the classification accuracy is evaluated as a function of supply voltage. EfficientNet-B4 network is considered in this section to quantify the accuracy and evaluate the accuracy-energy trade-off at scaled supply voltages. This neural network has 160 convolutional layers and 1 fully-connected layer, and it evaluates 19M parameters.

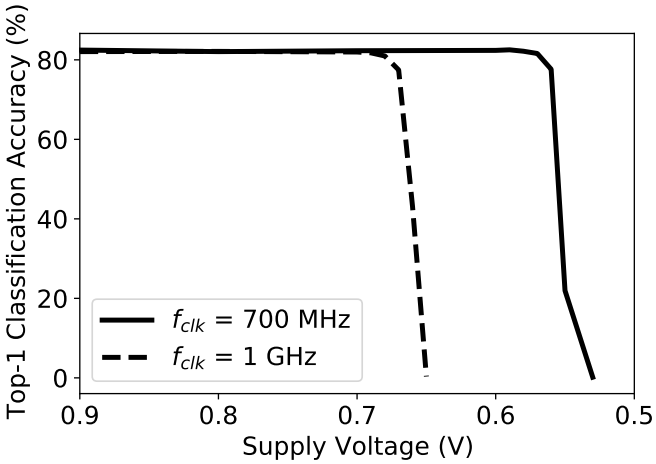


Figure 7.5: Top-1 classification accuracy of EfficientNet-B4 with respect to supply voltage scaling for 700 MHz and 1 GHz clock frequency (2.85 and 2 *ClockPeriod-to-PathDelay* ratio respectively).

It is observed in the previous chapter that the network can be quantized to 10-bits with 0.6% degradation in accuracy. The adder bit-width considered for the systolic array processing element to avoid overflow is 35 bits. For ImageNet, the Top-1 accuracy of the 10-bit quantized, pre-trained network is 82.15%.

The accuracy of EfficientNet-B4 with respect to operating voltage is shown in Fig. 7.5. As observed in the figure, the inference accuracy starts degrading beyond 0.6V and 0.7V when the operating clock frequency is 700 MHz and 1 GHz, respectively. At 0.57V and 700 MHz, the highest error probability in the network is 0.0006% and the accuracy drops by 1.04%, while at 0.68V and 1 GHz, the accuracy drops by 1.33% with 0.0008% error probability observed in the network. These results are consistent with the results obtained when uniform error rates are applied in the network during the network-level error resilience analysis (Fig. 6.4).

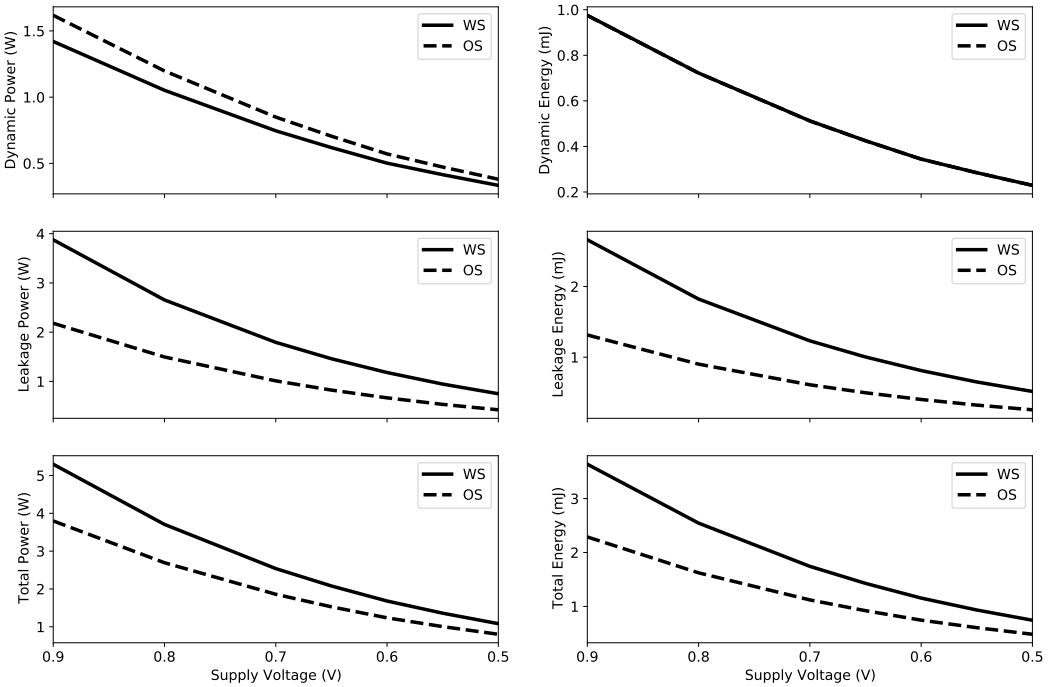


Figure 7.6: Dynamic, leakage and total power and energy consumption of 256×256 systolic array implementing WS and OS dataflows at 700 MHz for EfficientNet-B4 neural network.

7.3.2.1 Quantifying DNN energy consumption with respect to supply voltage

Considering a 256×256 systolic array with the same number of compute units as Google TPUv1 [1], SCALE-Sim is used to determine the number of compute cycles and array utilization is evaluated using the methodology described in section 7.1.2 for every layer in the EfficientNet-B4 network. Fig. 7.6 illustrates the power and energy consumption of the systolic array when performing inference of an input image, implementing both WS and OS dataflows at 700 MHz clock frequency. The maximum utilization of the systolic array is 18.63% and 38.31% for WS and OS dataflows respectively, which results in an average 2.45× higher leakage power consumption as compared to the dynamic power consumption. It

Array size	WS		OS	
	$util_{max}$ (%)	Cycles	$util_{max}$ (%)	Cycles
256×256	18.63	480664	38.31	421839
128×128	43.80	729449	69.94	681721
64×64	71.72	1219798	81.13	1190813
16×16	98.77	5380120	99.99	5210810

Table 7.6: Total cycle count and maximum utilization of EfficientNet-B4 for different systolic array size.

is observed that the dynamic energy is same for both the dataflows due to direct dependence of the dynamic energy on the utilization and compute cycles:

$$Energy \propto Utilization \times Cycles. \tag{7.7}$$

The rate of increase in utilization is the same as the rate of decrease in the compute cycles when comparing the WS and OS dataflows (section 7.1.2), which results in the approximately same dynamic energy consumption. Comparing the total power consumption, WS dataflow is better than output stationary, however, the 1.14× higher number of cycles results in higher energy consumption.

Energy Consumption w.r.t array size -

The power and energy consumption in the systolic array are compared for different array sizes. It is observed that as the array size reduces, there is an increase in utilization. The maximum utilization and total cycle count for different systolic array sizes are summarized in Table 7.6. The dynamic and leakage power consumption is compared in Fig. 7.7. It is observed that the leakage power becomes comparable or lower than the dynamic power as the array size reduces. The total

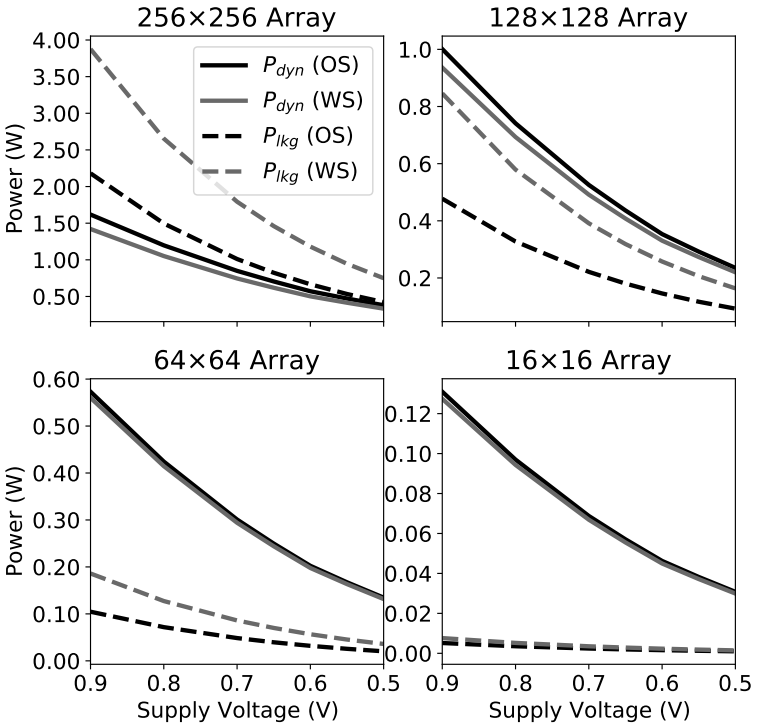


Figure 7.7: Dynamic and Leakage power consumption of systolic arrays of size 256×256, 128×128, 64×64, and 16×16, while considering 700 MHz clock frequency. Both the WS and OS dataflows are compared.

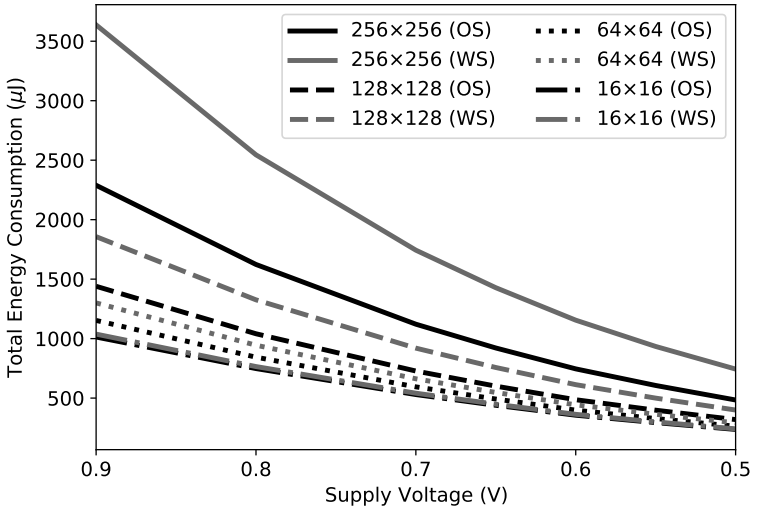
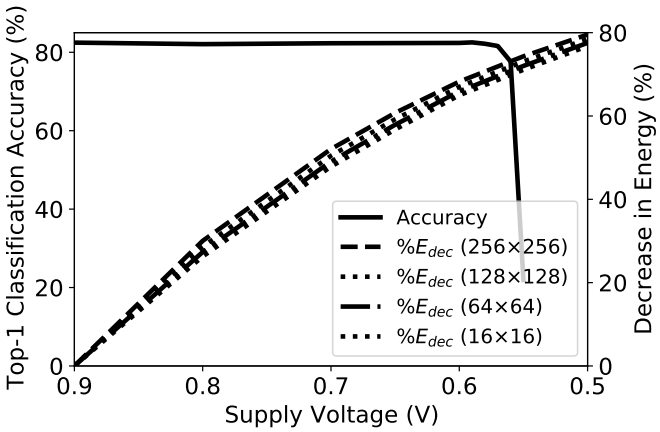
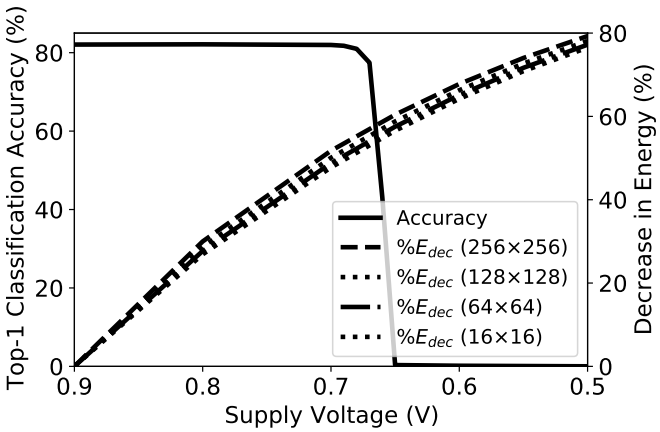


Figure 7.8: Total energy consumption of systolic arrays of size 256×256, 128×128, 64×64, and 16×16. Both the WS and OS dataflows are compared.



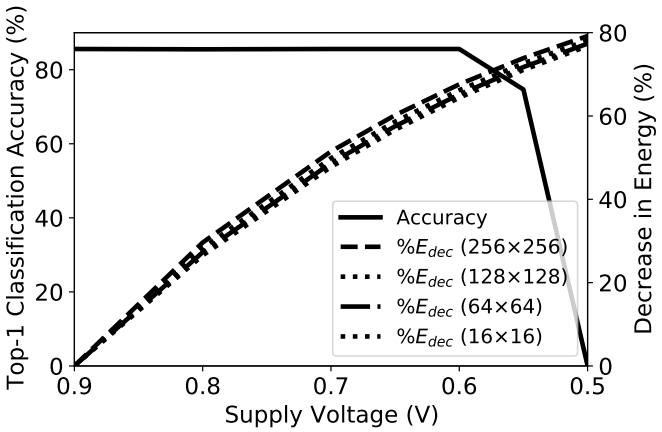
(a) f_{clk} =700 MHz



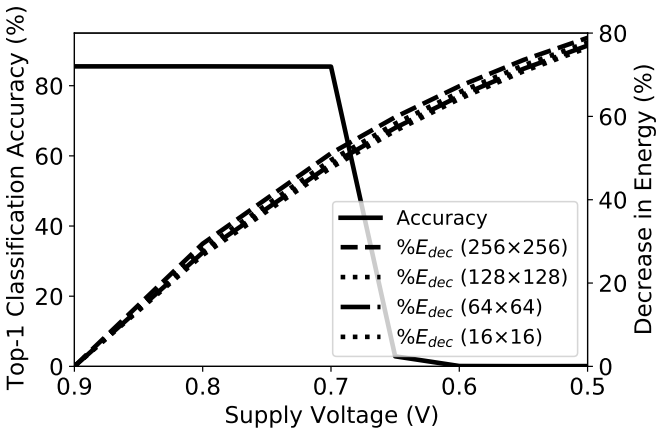
(b) f_{clk} =1 GHz

Figure 7.9: EfficientNet-B4 Top-1 classification accuracy and percentage decrease in total energy consumption for systolic array implementing WS dataflow. The baseline Top-1 accuracy at 0.9V is 82.4%.

energy consumption also decreases with array size (Fig. 7.8). Even though the total energy consumption reduces with array size, there is a latency trade-off due to additional overhead of the increased cycle count for both WS and OS implementation (Table 7.6).



(a) f_{clk} =700 MHz



(b) f_{clk} =1 GHz

Figure 7.10: EfficientNet-B8 Top-1 classification accuracy and percentage decrease in total energy consumption for systolic array implementing WS dataflow. The baseline Top-1 accuracy at 0.9V is 85.59%.

7.3.2.2 Accuracy-Energy trade-off

The trade-off between inference accuracy and improvement in energy efficiency is illustrated in Fig. 7.9 for EfficientNet-B4, considering 700 MHz and 1 GHz frequencies. The percentage decrease in energy is calculated with respect to the total

energy consumption at nominal voltage (0.9V). For 700 MHz operating clock frequency, the inherent resilience of the neural network allows 36.67% reduction in supply voltage (0.9V to 0.57V) without an impact in the inference accuracy. This voltage scaling results in a significant reduction in energy consumption (71.89%). Scaling the supply voltage further to 0.55V reduces the energy, but it is at the cost of 73.4% lower inference accuracy. The supply voltage scaling at 1 GHz is limited to 0.68V at which the inference accuracy reduces by 1.3%, while the total energy reduces by 53.08%. Similarly for EfficientNet-B8 (Fig. 7.10), at 0.6V (700 MHz) and 0.7V (1 GHz), there is a negligible loss in inference accuracy (85.5%) with an average 65.84% and 57.95% energy savings, respectively. Depending on the error tolerance of the neural network application, these results can be used to achieve significant energy savings with low voltage operation.

7.4 Per-Layer Voltage Scaling

It is observed in the previous section that by scaling the supply voltage for all the layers in the neural network, significant energy savings can be obtained without impacting the inference accuracy. As shown in Fig. 7.11, MobileNetV2 Top-1 accuracy reduces by 4.49% at 0.55V, but there is an average 72.68% reduction in total energy consumption at 700MHz. It is also observed in section 6.3.3 that based on the per-layer error resilience of the network, some layers can tolerate higher error rates as compared to others and some of these error resilient layers (e.g. the last convolutional layer) are also more compute intensive (higher *MOps*). Based on the per-layer error resilience, the operating voltage of some layers can be further reduced to improve the energy efficiency of the neural network. When scaling

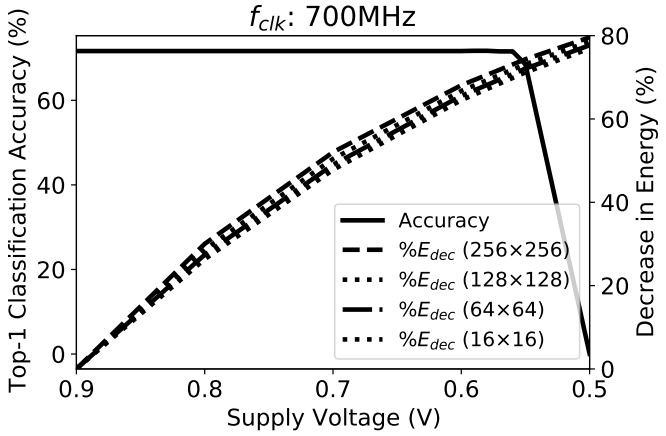


Figure 7.11: MobileNetV2 Top-1 classification accuracy and percentage decrease in total energy consumption of different systolic array sizes implementing WS dataflow.

the operating voltage for each layer, it is important to identify the layers which are more error resilient and also provide significant improvement in power and energy. The error resilience of each layer can be quantified by injecting per-layer errors at the given rate (determined from the error probability modeling methodology) using the framework described in the previous chapter. The layer consuming the highest power and energy is the layer which also has maximum array utilization over the total compute cycles (for the respective layer). In other words, higher array utilization across the compute cycles in a given layer results in higher energy consumption, and the effect of voltage scaling is significant for these layers.

Therefore, using the per-layer array utilization, compute cycles and $Err_{1\%}$ (error rate at which DNN accuracy is one percent point lower), the *ranking_metric* of a layer n can be quantified as:

$$ranking\_metric_n = Err_{1\%_n} \times util_n \times cycles_n. \tag{7.8}$$

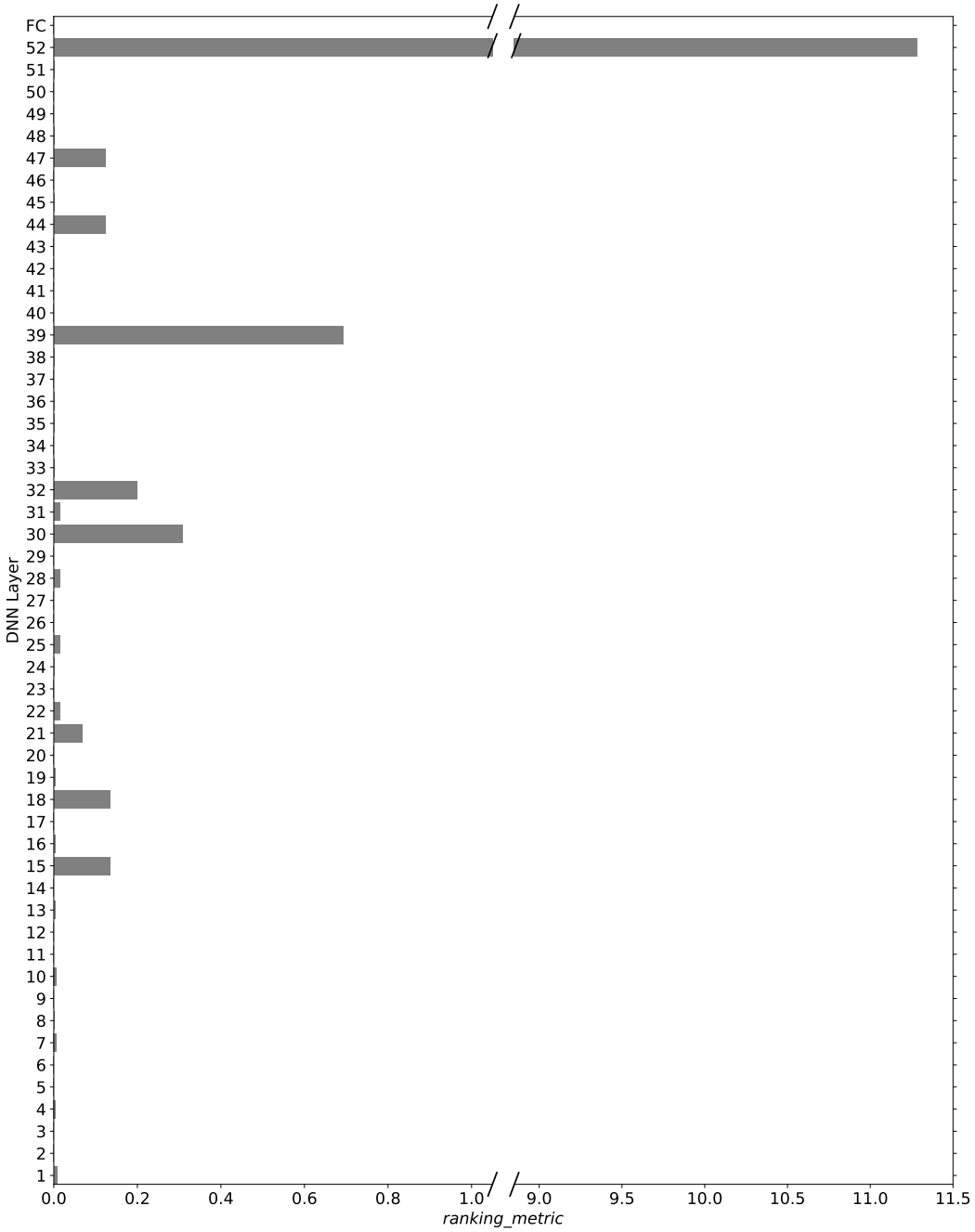


Figure 7.12: MobileNetV2 per-layer *ranking_metric* [eq. (7.8)].

Note that for $ERR_{T\%}$ metric, $T = 1$ is chosen in this work as the desired loss in application level accuracy, though any value of T can be chosen based on the accuracy requirements. The $ranking_metric$s evaluated for all the MobileNetV2 DNN layers are shown in Fig. 7.12. As observed in the figure, the last convolutional layer has the highest ranking which is expected since the layer is most error resilient and has highest *MOps*, based on the per-layer error resilience analysis results in Table 6.3.

Quantifying the per-layer $ranking\_metric$ provides an organized approach to per-layer voltage scaling as the layer rankings prioritize the voltage scaling of the error resilient and compute intensive layers in the network. When the operating clock frequency is 700 MHz, the supply voltage can be scaled for the entire network to 0.6V. Based on the layer rankings, the per-layer operating voltage is further scaled to understand the impact on inference accuracy and energy consumption of the network. The results of per-layer voltage scaling are analyzed in different scenarios (S1–S5), each considering a scaled voltage for a given layer while all other layers are at 0.6V. In these scenarios, the supply voltage of six convolutional layers with highest $ranking\_metric$ are scaled, as summarized in Table 7.7. The variation in inference accuracy and percentage reduction in energy consumption is illustrated in Table 7.8. It is observed that the inference accuracy reduces from 71.65% to 70.63%, however, the reduction in total energy consumption as compared to the nominal voltage (0.9V) is 68.82% and 68.11% for WS and OS dataflows, respectively, as opposed to 68.10% (WS) and 67.36% (OS) when all layers are scaled to the same voltage (0.6V). Therefore, the per-layer supply voltage scaling results in 1.1% lower accuracy (compared to the 10-bit quantized, error-free network) with 1.06% additional energy savings (compared to the scenario when all the layers are at 0.6V). The energy consumption of each layer considered in the S1–S5 scenar-

	DNN layers					
	15	18	30	32	39	52
$ranking\_metric_n$	0.135	0.135	0.307	0.199	0.692	11.29
V_{layer} (S0)	**All layers at 0.6V**					
V_{layer} (S1)	0.6	0.6	0.6	0.6	0.6	**0.5**
V_{layer} (S2)	0.6	0.6	0.6	0.6	**0.55**	**0.5**
V_{layer} (S3)	0.6	0.6	**0.55**	0.6	**0.55**	**0.5**
V_{layer} (S4)	0.6	0.6	**0.55**	**0.55**	**0.55**	**0.5**
V_{layer} (S5)	**0.55**	**0.55**	**0.55**	**0.55**	**0.55**	**0.5**

Table 7.7: Operating voltage scaling for six convolutional layers with the highest *ranking_metric* in 5 different scenarios (S1–S5).

Scenario	Top-1 accuracy (%)	% Decrease in Energy	
		WS	OS
S0	71.65	68.10	67.36
S1	71.45	68.46	67.76
S2	71.61	68.55	67.86
S3	71.24	68.60	67.88
S4	71.32	68.72	67.98
S5	70.63	68.82	68.11

Table 7.8: MobileNetV2 Top-1 classification accuracy and percentage decrease in total energy consumption in different scenarios described in Table 7.7. The percentage decrease in energy is with respect to the energy consumption at 0.9V.

ios (Table 7.7) is a small percentage (0.7–3.2%) of the total energy consumption, which results in smaller energy savings with per-layer voltage scaling as compared to network-level voltage scaling.

7.5 Discussion

The timing error modeling methodology and the error resilience framework described previously are used to analyze the DNN inference accuracy with respect to supply voltage. Scaling the operating voltage increases the per-bit timing error probability of a processing element in a neural network, which further affects the inference accuracy. However, scaling the operating voltage for a neural network depends on the clock frequency. It is observed that the supply voltage can be scaled more at reduced operating clock frequency. For instance, the supply voltage for the entire EfficientNet-B4 network can be scaled to 0.68V and 0.57V when considering 1 GHz and 700 MHz operating clock frequency, respectively. The inference accuracy at the respective voltages reduces by $\approx$1%, while the energy consumption when performing inference of an input image reduces by 53.08% and 71.89%. Furthermore, the error resilient and compute intensive layer in the network can be identified by quantifying the *ranking_metric* of each layer with respect to the per-layer $Err_{1\%}$ and array utilization over the compute cycles. The ranking of each layer guides the per-layer voltage scaling by targeting the resilient layers in the network. It is observed that scaling six convolutional layers in MobileNetV2 reduces the inference accuracy of the network from 71.73% to 70.63% while a 68.82% (WS) and 68.11% (OS) improvement in energy is observed compared to the nominal voltage. However, compared to 0.6V, the energy can be reduced by only 0.98$\times$, which is a

very small improvement at the cost of 1.1% lower inference accuracy. The accuracy and energy consumption improvements with bit-level error detection and correction methodologies are discussed in the next chapter.

Chapter 8

Error Detection and Correction in Systolic Arrays

The previous chapter quantifies the inference accuracy of a neural network with respect to the timing errors in the systolic arrays at scaled voltages. The per-layer analysis highlights the increase in energy savings when the operating voltage of error resilient and compute-intensive layers are futher scaled compared to other layers. Furthermore, the per-bit error resilience analysis discussed in section 6.3.3 highlights the significant improvement in inference accuracy when one or more higher order bits are error-free. These results can be leveraged to scale the supply voltage further as the inference accuracy improves with bit-level error correction, therefore, enabling aggressive voltage scaling in DNN accelerators. However, it is important to analyze the trade-off between the design overhead and improvement in the quality-of-results that can be obtained through per-bit error detection and correction techniques. An existing error detection and correction technique (iRa-

zor [77]) is implemented in this chapter to analyze the energy and latency overhead of performing inference of an input image, while also evaluating the improvement in inference accuracy at scaled voltages.

The rest of the chapter is organized as follows. Section 8.1 provides a brief overview of the existing error detection and correction techniques. The per-bit error resilience of the neural network is analyzed in section 8.2. Section 8.3 discusses the implementation of iRazor flip-flop in systolic array and analyzes the design overhead and related accuracy trade-offs. The limitations to the supply voltage scaling at higher frequency is evaluated in section 8.4 by determining the lowest voltage at which the accuracy loss can be recovered with bit-level error correction. The results and observations are discussed in section 8.5.

8.1 Background

The process, voltage and temperature (PVT) variations can drastically impact the circuit performance [27]. In addition to providing additional design margins to account for these variations, various techniques have been proposed to mitigate this issue. Some of these variation-aware techniques include dynamic scaling of the voltage and frequency based on the performance requirements (while also improving the overall energy efficiency of the circuit) [78–85], or implementing an adaptive architecture to mitigate critical path timing failures in digital design through *in situ* error detection and correction (EDAC) [8, 86–88]. Shan et al. [85] propose a timing error prediction based adaptive voltage frequency scaling (AVFS) with on-chip detection window tuning (to overcome the challenge of varying detection window due to PVT variations). An overview of the differences between error de-

tection, prediction and masking is provided in [89]. Error detection is done *after* the clock edge while error prediction is based on monitoring the datapath signals for transitions for a specified period of time *before* the clock edge. Error masking technique (e.g. TIMBER [89]) combines both the aspects, where the errors are detected after the original clock edge, but propagates the correct values by borrowing time from the next pipeline stage [89,90]. A timing error-masking aware microcontroller system is presented in [90], where the error borrowing from the next pipeline stage is achieved through soft edge flip-flops. The error-aware microcontroller realizes ultra-low voltage operation resulting in significant energy savings.

Additional EDAC techniques implemented at the circuit and architectural level detect errors in a given datapath and correct the errors by gating the clock for one cycle giving the pipeline an opportunity to correct the error. For instance, Razor flip-flop [86] implements a shadow latch operating on a delayed clock with the regular flip-flop to detect setup time failures in the critical datapaths. The mismatch in the output of the shadow latch and regular flip-flop raises an error signal which prompts restoration of the correct data from the shadow latch, while one cycle overhead is incurred to ensure that correct data is processed in the pipeline. An alternative approach to error correction is proposed in [87], where the razor flip-flop circuit is modified (with 38.15% fewer transistors) to only detect the errors, while the correction is done through architectural replay. These techniques have been implemented in error resilient DNN hardware design to mitigate the impact of timing errors with voltage underscaling [52,53]. In these works, the timing error resilience is improved by implementing Razor flip-flops to detect errors and bypass the subsequent MAC, therefore, dropping the erroneous partial sum. Another recent EDAC approach is iRazor [77] flip-flop, which suppresses the time margin added to tol-

erate process, voltage and temperature (PVT) variations. Significant performance gains and energy reductions are achieved across 0.6–1V voltage range, at the cost of area overhead.

As opposed to these prior works, the objective of this work is to analyze the overhead of implementing bit-level error detection and correction and evaluate the trade-off between accuracy, energy, and latency overhead. With the error resilience analysis framework proposed in this thesis, the number of bits which need to be corrected to obtain the desired accuracy can be identified and the related trade-offs of bit-level error correction can be analytically evaluated to enable smarter and efficient implementation of EDAC techniques. To achieve this objective, the EDAC technique proposed in [77] is implemented in this work. The error detection in [77] is implemented by substituting a regular flip-flop with an iRazor flip-flop while the error correction is implemented by gating the clock whenever there is an error. Since the probability of error is considerably lower (as observed in the previous chapters) the latency overhead of gating the clock is smaller compared to the overall computation time. The accuracy, energy and latency trade-offs are further explored in this chapter. Note that this framework is not limited to the iRazor EDAC technique and other error correction methodologies can be adapted to analyze the overheads.

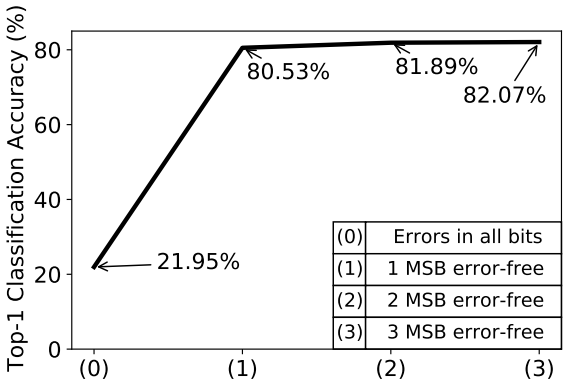


Figure 8.1: Top-1 classification accuracy of EfficientNet-B4 with 1–3 higher order bits error-free for 0.55V operating voltage and 700 MHz clock frequency. Note that all the layers of the network are scaled to 0.55V.

8.2 Quantifying Accuracy with Bit-Level Error Correction

In this section, the per-bit error resilience of the EfficientNet-B4 DNN is analyzed by quantifying the inference accuracy with respect to bit-level error correction. Considering the network-level voltage scaling results obtained at 700 MHz clock frequency, the Top-1 accuracy reduces from 82.46% at 0.9V to 21.95% at 0.55V. The per-bit error resilience of the network is analyzed by considering one or more higher order bits error-free before evaluating the inference accuracy. The accuracy results for 0.55V operating voltage and 700 MHz clock frequency are illustrated in Fig. 8.1. As observed in the figure, the inference accuracy improves by 3.67–3.74$\times$ when 1–3 higher order bits in all layers of the network are error-free. In addition, it is observed in Fig. 7.6 that scaling the operating voltage of the neural network from nominal voltage to 0.55V reduces the total energy consumption by 74.32%. The per-bit error correction of specific higher order bits can enable signif-

icant improvement in inference accuracy, therefore, enabling low voltage operation and significant energy savings in DNNs. It is important to analyze the overhead of correcting bit-level errors in the circuit to understand the actual trade-off between accuracy and energy efficiency of neural network when implementing EDAC techniques to mitigate the effect of timing errors in specific higher order bits. Note that each layer in the network utilizes different accumulator bit-widths, which depends on the number of accumulations per output in a given layer. Therefore, when implementing error correction in hardware, it is not feasible to correct only the most significant bit in all the layers in the network. For example, in EfficientNet-B4, implementing error correction in 35th bit position corrects the errors only in 14 layers (out of 161 layers). The implementation of iRazor flip-flop to detect and correct errors in the systolic array is discussed in the following section.

8.3 Bit-Level Error Detection and Correction using iRazor

As discussed previously, there are a number of existing EDAC techniques such as Razor [86], error masking using soft-edge flip-flops [90] and iRazor [77]. In this work, iRazor flip-flop is implemented to correct bit-level timing errors in the systolic array. Zhang et al. proposed a flip-flop design which adds a light-weight error detection circuit to a latch (with asynchronous *reset*), as shown in Fig. 8.2. Fig. 8.3 shows the error detection operation of iRazor flip-flop when the input transitions after the clock edge. Timing violations are detected if the input transitions in the error detection window, which is defined by setting CTL as low. When the input transitions before the rising edge of *CLK* or before the falling edge of *CTL* sig-

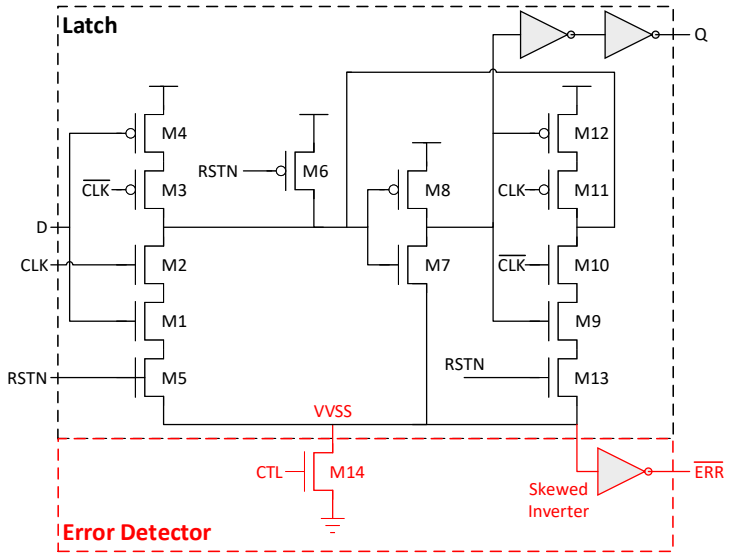
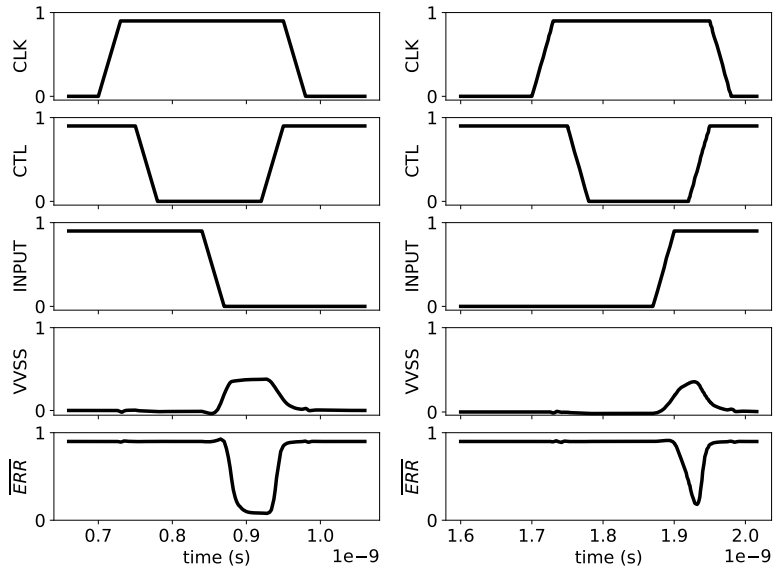


Figure 8.2: Schematic of iRazor flip-flop [77].



(a) High-to-low input transition (b) Low-to-high input transition

Figure 8.3: Waveforms illustrating the iRazor flip-flop error detection functionality.

nal, no error is flagged and the virtual ground (*VVSS* in Fig. 8.2 and 8.3) is held at ground. The input is considered valid before the falling *CTL* edge due to time borrowing functionality of iRazor latch. However, when the input *D* transitions within the error detection window, the virtual ground is raised due to capacitor discharge through the first tristate inverter (M1–M5 in Fig. 8.2) or the subsequent inverter (M7–M8), pulling the $\overline{ERR}$ signal down through the HI-skewed inverter.

The falling $\overline{ERR}$ triggers the clock gating logic, as shown in Fig. 8.4. To ensure an even distribution, 16 $\overline{ERR}$ signals drive the dynamic OR latch stage (as opposed to 10 $\overline{ERR}$ signals in [77]). Multiple latch stages drive the OR propagation stages to generate the global *razor_error* signal which drives the clock gating logic. For a $N \times N$ systolic array, the error correction circuit includes $\frac{N \times N}{16}$ dynamic OR latches. Considering n input OR gates, the number of propagation stages to get the global *razor_error* signal can be evaluated as $\log_n \frac{N \times N}{16}$. This error detection and correction logic is implemented in 20nm HP FinFET technology in HSPICE. The waveforms in Fig. 8.5 illustrate the proof-of-concept of this logic. As shown in the figure, an erroneous data transition triggers the *en* signal for clock gating. The clock gating logic gates the clock allowing the pipeline an additional cycle to propagate correct data signal, therefore, adding a latency overhead of one clock cycle.

In the processing element implemented using 20nm HP FinFET technology in HSPICE, the regular flip-flop is substituted with the iRazor latch for the output bit position where the timing error is corrected. When computing the output error probability for a given bit position with iRazor latch, the data path delay which will result in timing failure in the latch can be determined from the following equation:

$$t_G + t_S > 1.5 \times T_{clk}, \tag{8.1}$$

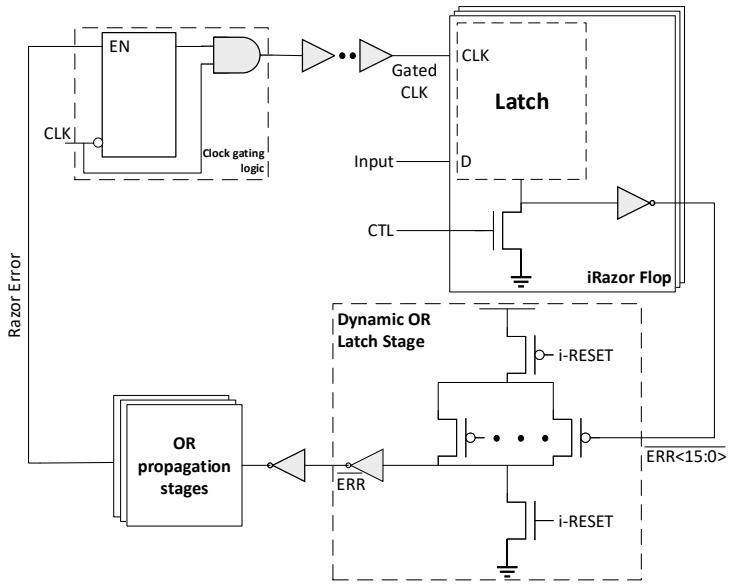


Figure 8.4: Basic schematic for error detection and correction based on [77].

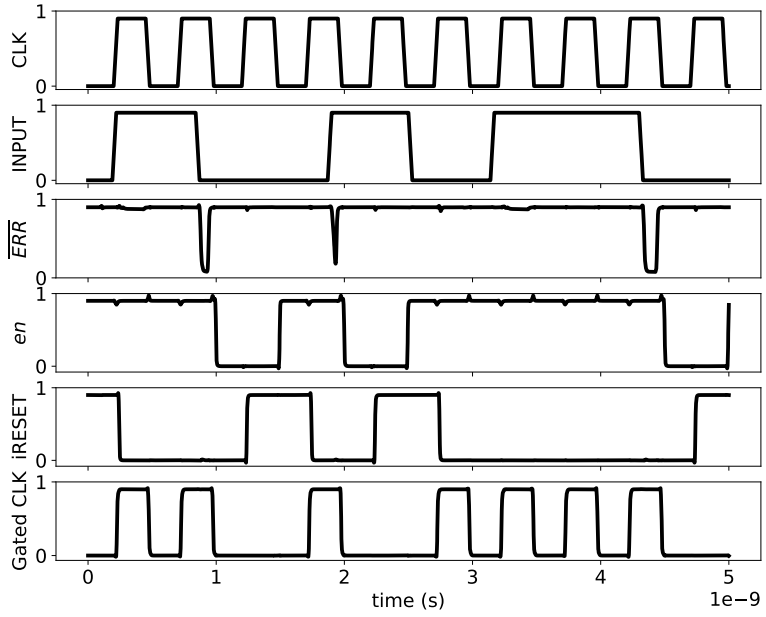


Figure 8.5: Waveforms showing iRazor error detection and correction functionality.

Bit position	f_{clk} =700 MHz		f_{clk} =1 GHz	
	DFF	iRazor	DFF	iRazor
25	0.3838	0.261	0.4494	0.3279
26	0.3869	0.2638	0.4539	0.3308
27	0.39	0.2666	0.4585	0.3337
28	0.3931	0.2694	0.463	0.3366
29	0.3962	0.2721	0.4677	0.3394
30	0.3992	0.2749	0.4724	0.3423
31	0.4023	0.2776	0.4771	0.3450
32	0.4053	0.2804	0.4818	0.3478
33	0.4083	0.2831	0.4866	0.3505
34	0.4113	0.2857	0.4914	0.3533
35	0.4143	0.2884	0.4963	0.356
36	0.4173	0.2911	0.5012	0.3586

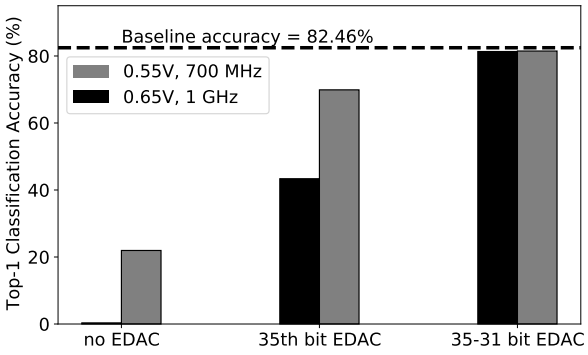
Table 8.1: Comparison of the voltage at which the critical data path (of the respective bit positions) fails timing for D flip-flop and iRazor latch.

where t_G is the data path delay, t_S is the setup time of the latch and T_{clk} is the operating clock period (since the latch has an additional half a clock period due to transparency). Considering this clock period, the timing error probability is computed for the respective bit positions with iRazor latch using the methodology discussed in chapters 3 and 4. The voltage at which one or more paths driving a given output bit position fail timing is compared for regular flip-flop and iRazor latch in Table 8.1. The voltage at which timing error occurs is 0.69× (700 MHz) and 0.72× (1 GHz) lower for the iRazor latch. The timing error probability computed from this voltage represents the errors which cannot be detected by the iRazor latch (the data

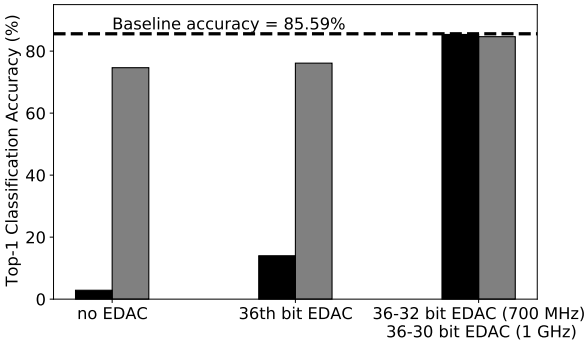
transitions outside the error detection window). This probability is orders of magnitude lower than the error probablity obtained without EDAC. For instance, at 0.55V and 700 MHz frequency, the timing error probability of the 35th bit in layer 86 in EfficientNet-B4 DNN is 0.0103% without error detection and correction. Substituting the regular flip-flop with iRazor in the 35th bit position of the PE reduces the timing error probability for the layer to 4.9×10^{-12}, which improves the inference accuracy in the network significantly. The improvement in inference accuracy, and the corresponding latency and energy overhead of implementing EDAC in systolic array are discussed in the following sections.

8.3.1 DNN Accuracy with EDAC

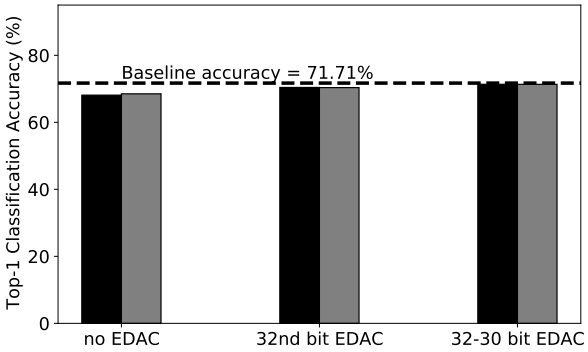
The inference accuracy with bit-level error correction is evaluated for 10-bit quantized MobileNetV2, EfficientNet-B4 and B8 DNNs. It is observed that based on the operating frequency (and available timing slack for critical data path), the supply voltage for EfficientNet-B4 and MobileNetV2 can be scaled to 0.57V and 0.68V respectively for 700 MHz and 1 GHz respectively with $<1\%$ degradation in Top-1 accuracy. For EfficientNet-B8, the respective voltages are 0.6V and 0.7V. Scaling the voltage further results in a significant degradation in inference accuracy (Table 8.2). When implementing bit-level error correction using iRazor, it is observed that correcting only select higher order bits in the network can improve the inference accuracy significantly (Fig. 8.6). For B4 and B8 networks, the inference accuracy obtained is within 1% of the baseline accuracy when the timing errors in 5 higher order bits are corrected, whereas for MobileNetV2, correcting only 3 higher order bits gives the desired accuracy. Note that correcting 3 higher order bits in MobileNetV2 corrects timing errors in 1–3 bits in 20 layers (37.74% of 53 layers),



(a) EfficientNet-B4



(b) EfficientNet-B8



(c) MobileNetV2

Figure 8.6: EfficientNet-B4 and B8, and MobileNetV2 Top-1 classification accuracy with bit-level EDAC implementation when considering 0.55V and 0.65V for 700 MHz and 1 GHz clock frequencies, respectively.

	MobileNetV2	EfficientNet-B4	EfficientNet-B8
0.9V	71.65%	82.46%	85.59%
0.55V (f_{clk}=700 MHz)	68.51% p_{max} = 0.003%	21.95% p_{max} = 0.01%	74.65% p_{max} = 0.029%
0.65V (f_{clk}=1 GHz)	68.12% p_{max} = 0.006%	0.33% p_{max} = 0.018%	2.87% p_{max} = 0.06%

Table 8.2: Top-1 accuracy and maximum timing error probability (p_{max}) for MobileNetV2, EfficientNet-B4 and B8 networks at different voltages and frequencies.

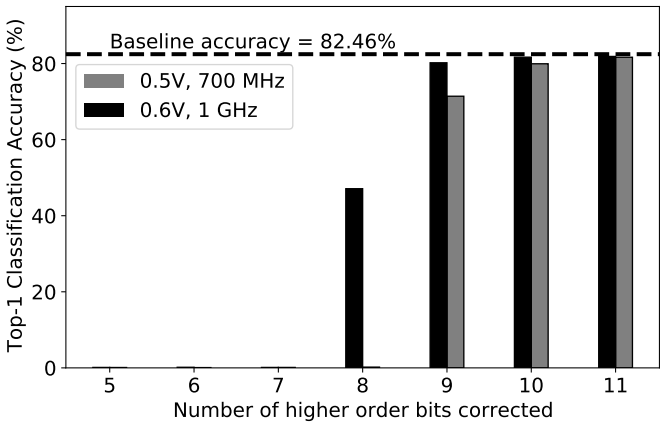


Figure 8.7: EfficientNet-B4 inference accuracy improvement with per-bit error correction at 0.5V and 0.6V for 700 MHz and 1 GHz, respectively. Note that the highest MSB position for all the layers in this DNN is 35.

and correcting 5 higher order bits corrects the timing errors in 1–5 bits in 48 layers (29.81% of 161 layers) in EfficientNet-B4 and 62 layers (20.39% of 304 layers) in EfficientNet-B8.

To further push the boundaries on low voltage operation, EfficientNet-B4 network is evaluated for 0.5V and 0.6V at 700 MHz and 1 GHz clock frequencies, respectively. The Top-1 accuracy of the network without EDAC at both the volt-

ages is 0.16% and the maximum timing error probability in the network is 7.53% (700 MHz) and 6.77% (1 GHz). As observed in Fig. 8.6(a), correcting 5 higher order bits recovers the accuracy loss in the network at 0.55V (700 MHz) and 0.65V (1 GHz). Due to higher error probability at reduced voltages, the number of bits to be corrected at 0.5V and 0.6V is evaluated by implementing EDAC in 5 to 11 higher order bits and the accuracy is evaluated with each increment in bit correction. The improvement in inference accuracy with per-bit error correction is illustrated in Fig. 8.7, and it is observed that the desired accuracy (within 1% of the baseline accuracy) is obtained by correcting 35–25 bits (700 MHz) and 35–26 bits (1 GHz). In this case, the timing errors are corrected in 150 layers (93.17%) and 140 layers (86.96%) of the EfficientNet-B4 network at 700 MHz and 1 GHz frequencies, respectively. The Top-1 accuracy obtained with EDAC is 81.7%, which is within 1% of the baseline accuracy of 82.49%.

8.3.2 Latency Overhead

In the previous section, it is observed that correcting bit-level errors in systolic array can provide a significant improvement in inference accuracy, even at low operating voltages. However, the error correction implemented with iRazor flip-flop gates the clock for one cycle when an error is detected, which adds a latency overhead to the total output computation in the systolic array, when performing inference of an input image. This latency overhead can be analytically quantified with respect to the timing error probability for every layer in the network.

8.3.2.1 MSB error correction overhead

A timing error probability of p for a single PE indicates that an error is expected to occur every $1/p$ cycles. If N PEs are used for computing the activation in a given layer, the probability that at least one PE has an error in a given clock cycle can be evaluated as,

$$P = 1 - (1-p)^N. \tag{8.2}$$

The average number of active PEs in the systolic array for a given layer for each cycle can be computed based on the utilization obtained from (7.2). The expected number of errors in a given layer which requires n cycles to compute all the activations can be analytically determined as: $n \times P$. This is equivalent to the number of cycles that the clock is gated to prevent erroneous signals to be propagated in the pipeline during computation. Therefore, the average latency overhead can be evaluated as the additional timing overhead of gating $n \times P$ clock cycles:

$$\text{Latency overhead, } lat_{over} = n \times [1 - (1-p)^N] \times \frac{1}{f_{clk}}, \tag{8.3}$$

where p is the error probability of the MSB in the PE, N is the average number of active PEs in the systolic array for a given layer in a given cycle, n is the total number of compute cycles for the layer and f_{clk} is the operating clock frequency. Considering the example of the first convolutional layer in EfficientNet-B4 which computes all the activations in 12646 cycles for WS dataflow, if the final timing error probability for the MSB at the output of a PE is 7.52×10^{-10} and the utilization of a 256×256 array is 1.96%, the average number of active PEs per cycle is 1286. Therefore, the probability that there is an error in at least one of the PEs per cycle is 0.0001%, which adds an average latency overhead of 17.5ps.

8.3.2.2 Multiple higher order bits error correction overhead

The most significant bit position in different layers of the network depends on the number of accumulations per output in that layer. When correcting multiple bits in the network, the number of bits corrected depends on the MSB position of the layer. For instance, if layer x MSB position is 31 and layer y is 35, and the error correction is implemented for 31–35 bits; 5 higher order bits are corrected for layer x while only the MSB is corrected for layer y. Therefore, different layers in the network have different latency overhead when correcting multiple bits. Given the error probability of the ith bit position of the PE (p_i) and the average number of active PEs per cycle for a given layer (N), the probability that at least one of the active PEs have an error in one of the m bits corrected can be computed as,

$$P = 1 - \prod_{i=1}^{m} (1 - p_i)^N. \tag{8.4}$$

Therefore, similar to (8.3), the average latency overhead of correcting m bits in a layer can be calculated as:

$$\text{Latency overhead, } lat_{over} = n \times [1 - \prod_{i=1}^{m} (1 - p_i)^N] \times \frac{1}{f_{clk}}, \tag{8.5}$$

where n is the number of compute cycles for the respective layer and f_{clk} is the operating clock frequency. It is observed that the latency overhead increases as more errors are corrected due to an increase in the total error probability. Considering the example of layer 156 in EfficientNet-B4, when only MSB is corrected (bit 35), the error probability P is 0.0009% and the latency overhead is 0.69ns. However, when 5 higher order bits are corrected for this layer, the error probability increases to 0.002% which adds an average latency overhead of 1.48ns.

Data-flow	Array size	0.55V, 700 MHz			0.65V, 1 GHz		
		$t_{compute}$[a] (ms)	lat_{over}[b] (ns)	lat_{over}[c] (ns)	$t_{compute}$[a] (ms)	lat_{over}[b] (ns)	lat_{over}[c] (ns)
WS	256×256	0.49	0.644	2.616	0.34	0.721	2.322
	128×128	0.68	0.644	2.616	0.48	0.721	2.322
	64×64	1.09	0.644	2.616	0.76	0.721	2.322
	16×16	4.60	0.644	2.616	3.22	0.721	2.322
OS	256×256	0.29	0.638	2.605	0.20	0.714	2.311
	128×128	0.44	0.638	2.605	0.31	0.714	2.311
	64×64	0.77	0.638	2.605	0.54	0.714	2.311
	16×16	3.79	0.638	2.605	2.65	1.020	2.311

[a] No EDAC, [b] 32nd bit EDAC, [c] 32–30 bit EDAC

Table 8.3: Latency overhead (lat_{over}) for MobileNetV2 DNN comparing different systolic array sizes with EDAC implemented at 0.55V, 700 MHz and 0.65V, 1 GHz.

8.3.2.3 Latency overhead results

The latency overhead of MobileNet-V2, EfficientNet-B4 and B8 when performing inference of an input image are evaluated at 0.55V and 0.65V operating voltage for 700 MHz and 1 GHz clock frequency, respectively. Considering the three scenarios described in Fig. 8.6, the total latency overhead is evaluated for different array sizes implementing WS and OS dataflow, and summarized in Table 8.3 (MobileNetV2), Table 8.4 (EfficientNet-B4) and Table 8.5 (EfficientNet-B8). It is observed that even though the latency overhead increases as more higher order bits are corrected, it is still orders of magnitude lower than the total compute time in the systolic array. The results observed in the table are discussed below:

- **Different DNNs and Array sizes:** The latency overhead is higher for larger DNNs as the total compute time and MSB error probability increases. For

Data-flow	Array size	0.55V, 700 MHz			0.65V, 1 GHz		
		$t_{compute}$[a] (ms)	lat_{over}[b] (ns)	lat_{over}[c] (ns)	$t_{compute}$[a] (ms)	lat_{over}[b] (ns)	lat_{over}[c] (ns)
WS	256×256	2.95	23.03	79.85	2.06	45.02	114.64
	128×128	4.01	23.03	79.85	2.81	45.01	114.63
	64×64	6.28	23.03	79.87	4.40	45.02	114.65
	16×16	26.08	23.05	79.91	18.25	45.05	114.72
OS	256×256	1.54	23.03	79.84	1.08	45.01	114.62
	128×128	2.28	23.03	79.85	1.60	45.01	114.63
	64×64	3.89	23.03	79.85	2.72	45.01	114.63
	16×16	21.38	23.03	79.85	14.96	45.02	114.63

[a] No EDAC, [b] 35th bit EDAC, [c] 35–31 bit EDAC

Table 8.4: Latency overhead (lat_{over}) for EfficientNet-B4 DNN comparing different systolic array sizes with EDAC implemented at 0.55V, 700 MHz and 0.65V, 1 GHz.

Data-flow	Array size	0.55V, 700 MHz			0.65V, 1 GHz		
		$t_{compute}$[a] (ms)	lat_{over}[b] (ns)	lat_{over}[c] (ns)	$t_{compute}$[a] (ms)	lat_{over}[b] (ns)	lat_{over}[d] (ns)
WS	256×256	9.14	43.76	312.52	6.40	102.62	654.26
	128×128	13.07	43.77	312.54	9.15	102.63	654.32
	64×64	21.77	43.77	312.58	15.24	102.64	654.40
	16×16	105.07	43.82	312.76	73.55	102.74	654.82
OS	256×256	4.61	43.76	312.49	3.23	102.61	654.10
	128×128	7.13	43.76	312.50	4.99	102.61	654.19
	64×64	12.81	43.76	312.51	8.97	102.61	654.23
	16×16	84.70	43.76	312.52	59.29	102.61	654.28

[a] No EDAC, [b] 36th bit EDAC, [c] 36–32 bit EDAC, [d] 36–31 bit EDAC

Table 8.5: Latency overhead (lat_{over}) for EfficientNet-B8 DNN comparing different systolic array sizes with EDAC implemented at 0.55V, 700 MHz and 0.65V, 1 GHz.

instance, the MSB position for all the layers in MobileNetV2 is in the range of 24–32 while that for EfficientNet-B4 is 23–35. In addition, the total computation cycles for 161 layers in EfficientNet-B4 is 5.9× more than the computation cycles for 53 layers in MobileNetV2. For all the networks, it is observed that the latency overhead is approximately the same across different array sizes when only the MSB is corrected. From (8.4), it is evident that as the average number of active PEs per cycle reduces with array size, the probability that at least one of the active PEs has an error in one of the bits corrected (P) also reduces. Since latency overhead is directly proportional to the total number of compute cycles (n) and the probability P : $lat_{over} \propto n \times P$ (eq. (8.5)), as the number of bits corrected increases (P increases) and the number of compute cycles increase (as the array size reduces), there is an increase in latency overhead. However, this increase is negligible for smaller DNNs.

- **Different dataflows:** Comparing the WS and OS dataflows, the average utilization for OS is higher which increases the number of active PEs and therefore, the probability that one of the active PEs has an error. However, the total number of compute cycles in OS is comparatively lower, due to which the latency overhead is the same or less than the latency overhead of WS dataflow. The relative overhead (ratio of latency overhead and $t_{compute}$) is higher for OS due to lower $t_{compute}$, i.e., more errors are detected per cycle. For instance, the relative overhead for EfficientNet-B4 is 1.91–1.22× higher for OS dataflow, when considering 0.55V and 700 MHz.

The supply voltage can be further scaled to achieve more energy savings by correcting more higher order bits in the network, but there is a latency and energy

Dataflow	Array size	700 MHz		1 GHz	
		$t_{compute}$ (ms)	lat_{over} (μs) @0.5V EC: 35–25 bits	$t_{compute}$ (ms)	lat_{over} (μs) @0.6V EC: 35–26 bits
WS	256×256	2.95	144.73	2.06	48.86
	128×128	4.01	149.98	2.81	49.38
	64×64	6.28	153.37	4.40	49.74
	16×16	26.08	156.60	18.25	50.13
OS	256×256	1.54	129.05	1.08	46.78
	128×128	2.28	139.87	1.60	48.21
	64×64	3.89	147.59	2.72	49.08
	16×16	21.38	156.22	14.96	50.05

Table 8.6: Latency overhead for EfficientNet-B4 at 0.5V, 700 MHz and 0.6V, 1 GHz. The accuracy obtained with this EDAC implementation is 81.7% which is within 1% of the baseline accuracy.

overhead of implementing bit-level error correction. It is observed in Fig. 8.6(a) and 8.7 that the loss in inference accuracy due to high error probabilities at scaled voltages can be recovered by correcting 5 bits (at 0.55V) or 11 bits (at 0.5V) at 700 MHz clock frequency. To evaluate the latency overhead of correcting more bits in the network at scaled voltages, EfficientNet-B4 is scaled to 0.5V and 0.6V for 700 MHz and 1 GHz frequencies, respectively. The bit positions corrected for the network are 35–25 bits for 700 MHz and 35–26 bits for 1 GHz to achieve 81.7% Top-1 accuracy, which is within 1% of the baseline accuracy (82.5%). The latency overhead evaluated for this scenario is summarized in Table 8.6. Note that at 0.5V, the timing error probability of the PE is 1.12–6.51× higher for 35–25 bit positions which results in higher latency overhead compared to 0.6V. Considering the

128×128 systolic array, relative to the total compute time without error detection and correction, the overhead is 3.74% and 6.13% for WS and OS dataflows at 700 MHz. At 1 GHz, the respective overheads are 1.75% and 3.01%. This overhead is significantly higher compared to that observed in Table 8.4 due to low voltage operation and more bit-level error correction, but it is still orders of magnitude lower than the total compute time.

8.3.3 Energy Overhead

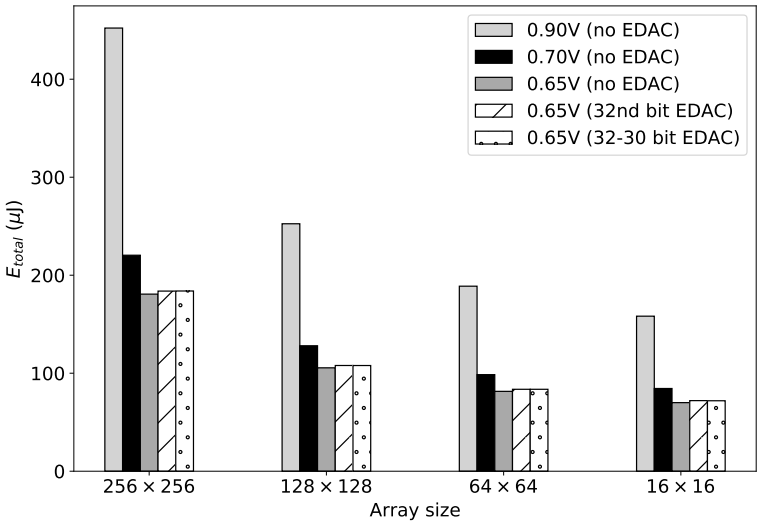
The error detection and correction in the PE is implemented by replacing the selected flip-flops with the iRazor flip-flop. The additional energy consumption of the EDAC logic (Fig. 8.4) implemented to correct bit-level errors results in an energy overhead, which is evaluated in this section. The energy overhead is computed by first determining the energy consumption of the PE with iRazor flip-flop (number of iRazor flip-flops depends on the number of bits corrected) and the clock gating logic through HSPICE simulations. The total energy consumption with EDAC is then analytically computed using the energy model described in section 7.2. As described in the previous chapter, this energy consumption is the energy consumed by the systolic array during inference of an input image. Note that the energy consumption of the array with EDAC discussed in this section does not take into account the energy cost of routing additional signals (e.g. *CTL* which is similar to a phase-shifted clock signal, as shown in Fig. 8.3, and controls the error detection logic in iRazor flip-flop) for EDAC in iRazor.

It is observed in the previous chapter that the operating voltage of the DNNs (considered in this analysis) can be scaled to 0.7V at 1 GHz frequency without a significant loss in accuracy. With bit-level EDAC, the supply voltage can be further

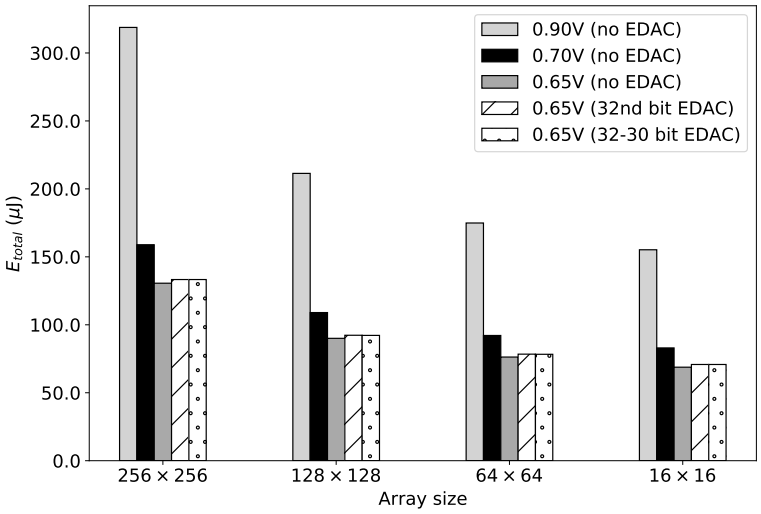
scaled to 0.65V and 0.6V while maintaining the inference accuracy within 1% of the baseline accuracy. However, this is at the cost of energy and latency overhead. For instance, the latency overhead of EfficientNet-B4 can range between 45.01ns–156.6μs (Table 8.4 and Table 8.6) based on the array size and dataflow.

The energy consumption of the systolic array with and without error detection and correction is illustrated in Figs. 8.8, 8.9 and 8.10 for MobileNetV2, EfficientNet-B4 and B8 DNNs, respectively. The clock frequency considered for all the networks is 1 GHz and the operating voltage is scaled to 0.65V and 0.6V (for EfficientNet-B4). It is observed in the figure that even though there is a small increase in the total energy consumption with EDAC implementation, the total energy consumption is comparatively less than that at 0.7V without EDAC. Considering different systolic array sizes, when only the MSB is corrected, the energy consumption at 0.65V reduces by 14.7–16.6% and 10.4–15.2% compared to the energy consumption at 0.7V for MobileNetV2 and EfficientNet-B4. For EfficientNet-B4, the supply voltage is scaled further to 0.6V (with 35–26 higher order bits corrected) and the total energy consumption reduces by 31.52–33.22% compared to 0.7V, resulting in significant energy savings. Since the bit-width of the accumulator increases for larger DNNs, the timing error probability of the critical path increases. The high timing error probability causes a larger latency and energy overhead, which results in lower energy savings at scaled voltages. For instance, the total energy consumption at 0.65V with EDAC is observed to be only 2.94–10.98% lower than the energy consumption at 0.7V without EDAC in EfficientNet-B8 DNN.

The total energy consumption at 0.65V and 0.6V (for EfficientNet-B4), and the percentage energy overhead of implementing EDAC are also summarized in Tables 8.7, 8.8 and 8.9 for MobileNetV2, EfficientNet-B4 and B8, respectively. For

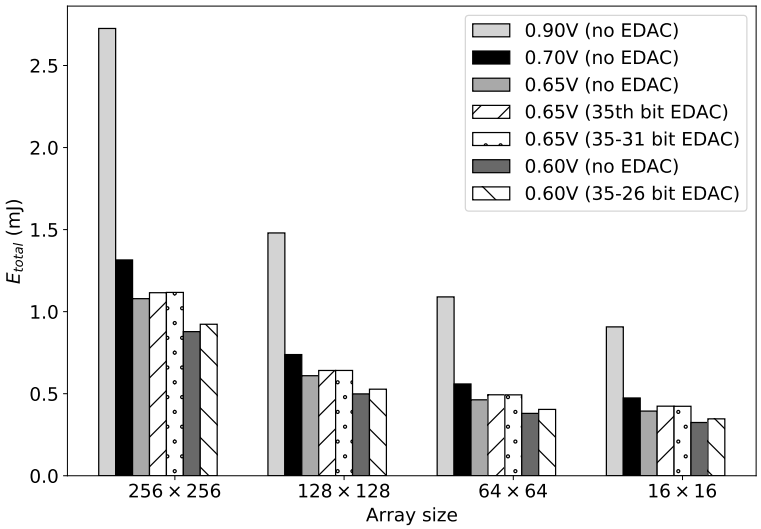


(a) WS

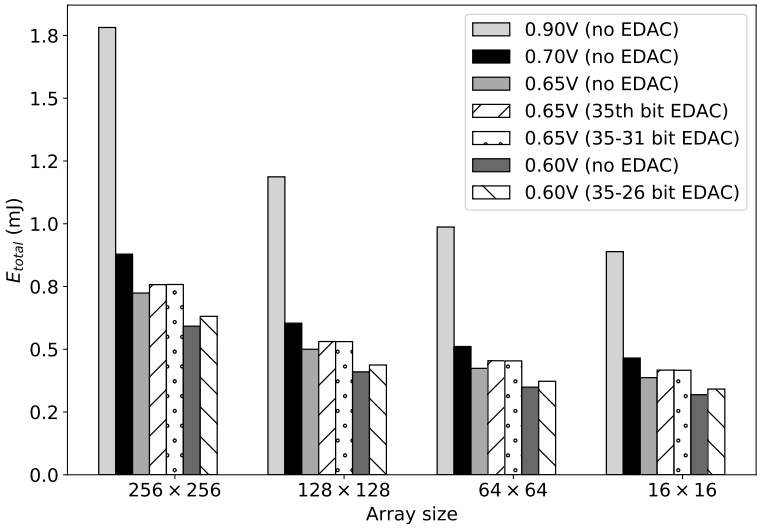


(b) OS

Figure 8.8: Energy consumption of MobileNetV2 DNN with and without error detection for different systolic array sizes considering WS and OS dataflow at 1 GHz clock frequency.

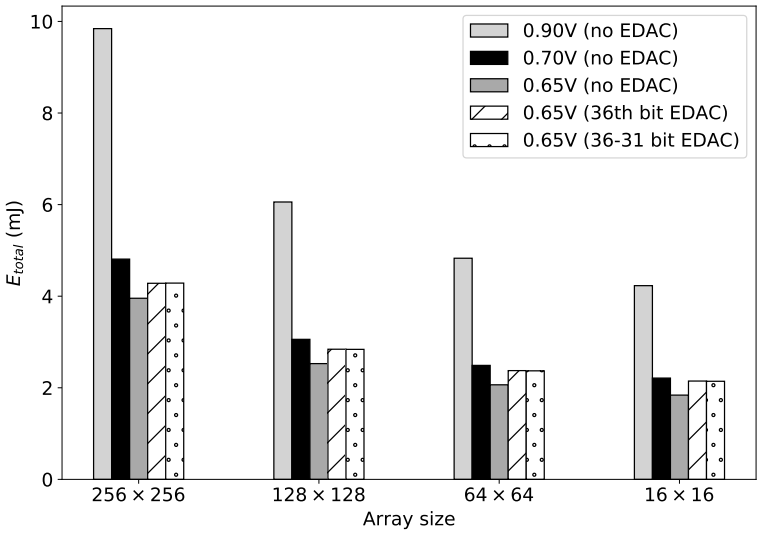


(a) WS

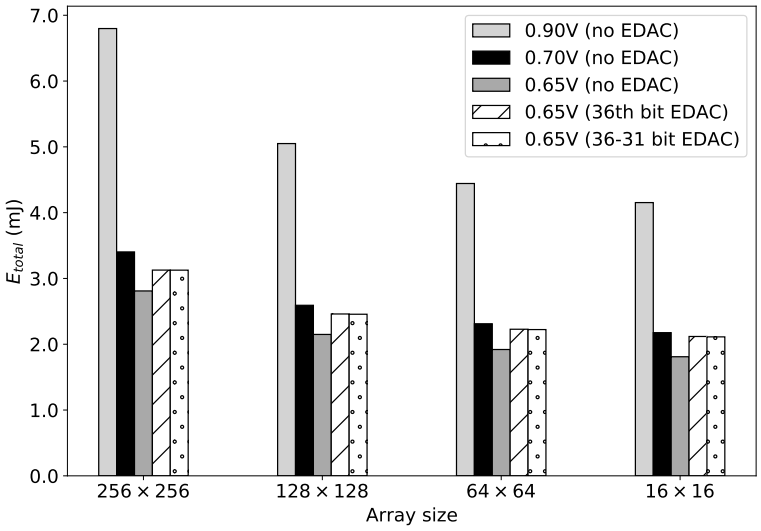


(b) OS

Figure 8.9: Energy consumption of EfficientNet-B4 DNN with and without error detection for different systolic array sizes considering WS and OS dataflow at 1 GHz clock frequency.



(a) WS



(b) OS

Figure 8.10: Energy consumption of EfficientNet-B8 DNN with and without error detection for different systolic array sizes considering WS and OS dataflow at 1 GHz clock frequency.

	Array size	E_{total} (μJ) at 0.65V		
		no EDAC	32nd bit	32–30 bits
Top-1 Acc.		68.12%	70.42%	71.36%
WS	256×256	180.7	183.80 (+1.72%)	183.91 (+1.78%)
	128×128	105.5	107.88 (+2.26%)	107.85 (+2.23%)
	64×64	81.49	83.64 (+2.64%)	83.57 (+2.55%)
	16×16	69.98	72.03 (+2.93%)	71.95 (+2.82%)
OS	256×256	130.6	133.27 (+2.04%)	133.30 (+2.06%)
	128×128	90.04	92.28 (+2.49%)	92.23 (+2.43%)
	64×64	76.27	78.38 (+2.77%)	78.30 (+2.66%)
	16×16	68.81	70.84 (+2.95%)	70.76 (+2.83%)

Table 8.7: Energy consumption of MobileNetV2 at 0.65V and 1 GHz frequency, with and without EDAC.

smaller arrays, the total energy consumption reduces due to lower leakage (compared to larger arrays). However, the lower energy consumption for smaller arrays increases the relative energy overhead (ratio of energy overhead and energy consumption without EDAC). For instance, the increase in total energy consumption for different array sizes is 0.036–0.03mJ (WS) or 0.317–0.308mJ (OS) for EfficientNet-B4 at 0.65V (with EDAC implementation), but as the total energy consumption reduces for smaller arrays, the relative energy overhead increases. Overall, there is a reduction in total energy consumption with EDAC and with this framework, it is observed that based on the DNN, further energy savings are possible by implementing bit-level error detection and correction in systolic array.

	Array size	E_{total} (mJ) at 0.65V			E_{total} (mJ) at 0.6V	
		no EDAC	35th bit	35–31 bits	no EDAC	35–26 bits
Top-1 Acc.		0.33%	43.36%	81.33%	0.16%	81.68%
WS	256×256	1.079	1.115 (+3.3%)	1.117 (+3.5%)	0.878	0.923 (+5.1%)
	128×128	0.610	0.642 (+5.2%)	0.642 (+5.2%)	0.499	0.528 (+5.8%)
	64×64	0.463	0.493 (+6.5%)	0.493 (+6.5%)	0.380	0.404 (+6.3%)
	16×16	0.394	0.424 (+7.6%)	0.423 (+7.4%)	0.325	0.347 (+6.8%)
OS	256×256	0.724	0.758 (+4.7%)	0.758 (+4.7%)	0.592	0.631 (+6.6%)
	128×128	0.500	0.531 (+6.2%)	0.531 (+6.2%)	0.410	0.437 (+6.6%)
	64×64	0.424	0.454 (+7.1%)	0.454 (+7.1%)	0.349	0.373 (+6.9%)
	16×16	0.387	0.417 (+7.8%)	0.416 (+7.5%)	0.319	0.341 (+6.9%)

Table 8.8: Energy consumption of EfficientNet-B4 at 0.65V and 0.6V, and 1 GHz frequency, with and without EDAC.

	Array size	E_{total} (mJ) at 0.65V		
		no EDAC	36th bit	36–31 bits
Top-1 Acc.		2.87%	13.99%	84.78%
WS	256×256	3.96	4.282 (+8.27%)	4.286 (+8.37%)
	128×128	2.53	2.842 (+12.42%)	2.839 (+12.30%)
	64×64	2.07	2.375 (+15.01%)	2.370 (+14.77%)
	16×16	1.84	2.147 (+16.68%)	2.142 (+16.41%)
OS	256×256	2.81	3.127 (+11.28%)	3.126 (+11.25%)
	128×128	2.15	2.460 (+14.47%)	2.456 (+14.29%)
	64×64	1.92	2.228 (+16.04%)	2.223 (+15.78%)
	16×16	1.81	2.118 (+17.02%)	2.112 (+16.69%)

Table 8.9: Energy consumption of EfficientNet-B8 at 0.65V and 1 GHz frequency, with and without EDAC.

8.4 Limitations to supply voltage scaling with EDAC

It is observed that scaling the supply voltage increases the timing error probability in the DNNs and therefore, reduces the inference accuracy. The results discussed in the previous section illustrate that the loss in accuracy in the DNNs can be recovered through bit-level EDAC, such that the voltage for MobileNetV2 and EfficientNet-B4 can be scaled to 0.5V and 0.6V at 700 MHz and 1 GHz clock frequency, respectively. In this section, the primary objective is to determine the breaking point in voltage scaling at which the accuracy loss cannot be recovered due to significantly high timing error probability with and without EDAC. To achieve this objective, a DNN is analyzed at lower voltages (and higher clock frequency) to determine the minimum voltage at which the accuracy loss in the network can be recovered with bit-level error detection and correction using iRazor.

For MobileNetV2, it is observed in Fig. 8.8 that the supply voltage can be scaled to 0.65V at 1 GHz clock frequency, by correcting 3 higher order bits in the network. To evaluate the effect of reducing the voltage further, the voltage is scaled to 0.55V and 0.5V. At 0.55V, the maximum error probability (p_{max}) observed by the network is 93.4% which results in ≈0% inference accuracy. An accuracy of 71.08% is obtained (within 1% of the baseline accuracy of 71.73%) by correcting 32–22 bits at which p_{max} is 0.213%, which is the timing error probability of bit 21. The latency and energy overhead of correcting 32–22 bits is 236.3μs and 28.01μJ (w.r.t. to energy consumption with no EDAC at 0.55V, 1 GHz) for 128×128 array (WS dataflow).

Scaling the voltage further to 0.5V increases the timing error probability due to which more higher order bits need to be corrected. However, it is observed that as more bits are corrected at this low voltage, the loss in inference accuracy is due to the highest timing error in the uncorrected bit position or the corrected bit position. Note that the corrected bit position has a non-zero timing error probability because the iRazor flip-flop is not ideal and it cannot detect every timing error (the data transitions outside the error detection window) in the system. To illustrate the different timing error probabilities with different number of bit corrections at low voltages, two scenarios are considered. In the first scenario, 32–21 bits are corrected and the value of p_{max} observed is 36.93%, which is the error probability of the uncorrected bit position 20. In the second scenario, 32–14 bits are corrected and the value of p_{max} observed is 1.26%, which is the error probability of corrected bit position 32. Therefore, even with EDAC, the highest error probability of 1.26% at 0.5V and 1 GHz results in 0.1% inference accuracy, as opposed to 71.08% accuracy at 0.55V (with 32–22 bits corrected). The timing error probability of iRazor is

dominant when more bits are corrected, and because both the probabilities are high, the inference accuracy cannot be recovered at 0.5V and 1 GHz clock frequency.

As the accumulator size increases for larger DNNs, the timing error probability at low voltage and high frequency operation also increases, therefore, limiting the supply voltage scaling. It is observed in the previous section that the accuracy of EfficientNet-B4 is 81.68% (within 1% of the baseline accuracy) at 0.6V and 1 GHz clock frequency with 35–26 bit error correction (p_{max} = 0.0066%), and the latency and energy overhead is 49.38μs and 0.029mJ (w.r.t. the energy consumption with no EDAC at 0.6V, 1 GHz), respectively. Scaling the voltage to 0.55V at 1 GHz frequency increases the p_{max} to 99.86% resulting in $\approx$0% accuracy. Similar to MobileNetV2, the highest error probability in the network at 0.55V (with EDAC) can be the error probability of the uncorrected or the corrected bit position. This is illustrated by considering two scenarios where bit positions 35–20 and 35–18 are corrected by replacing the regular flip-flop with iRazor. The accuracy evaluated with EDAC (for both the scenarios) is 0.5% due to the high error probability with and without error correction. When correcting 35–20 bits, p_{max} is 0.0484%, which is the error probability of bit 19. When 35–18 bits are corrected, p_{max} is 0.047%, which is the error probability of bit 35 with iRazor flip-flop. In both the scenarios, the timing error probability is significantly high for EfficientNet-B4 due to which the accuracy loss cannot be recovered. Therefore, EfficientNet-B4 can be scaled to only 0.6V as opposed to 0.55V for MobileNetV2.

8.5 Discussion

The error probability model and the resilience analysis framework proposed in this work provide an opportunity to quantify the DNN quality-of-results with respect to supply voltage and analyze the performance, energy trade-offs. An existing EDAC technique [77] is implemented in the processing element of the systolic array to evaluate the accuracy, energy trade-offs and understand the impact of implementing bit-level error correction techniques in the systolic array. The results discussed in this chapter are summarized in Tables 8.10 to 8.15. It is observed that as the supply voltage is scaled, more errors need to be corrected to recover the loss in inference accuracy, but there is a trade-off between accuracy, latency and energy consumption with bit-level error correction. Depending on the DNN considered, the systolic array size and the dataflow, the overhead can be significantly high. For instance, the latency overhead of correcting 11 higher order bits in MobileNetV2 to recover the accuracy loss at 0.55V, 1 GHz is 0.18–0.54ms (16.65–53.57% of the total latency) and the energy overhead is in the range of 3.1–81.7μJ (6.64–69% of total energy). The trade-off between latency and total energy consumption of systolic array with EDAC implementation is analyzed and it is observed that reducing the systolic array size reduces the energy consumption but at the cost of higher latency. This overhead increases as more higher order bits are corrected in the processing element. It is important to note that the supply voltage scaling is limited by the high timing error probability of the systolic array with and without EDAC at a given frequency. For EfficientNet-B4, if the supply voltage is scaled to 0.55V at 1 GHz frequency, the accuracy of the network cannot be recovered even when all the bits are corrected. This is due to the non-zero timing error probability of the iRazor latch which increases with supply voltage scaling (even though it is

less than the regular flip-flop). Therefore, the framework presented in this dissertation can be used to evaluate this trade-off and implement smarter and more efficient bit-level error mitigation techniques. In the future, the area overhead can also be analyzed through hardware implementation to compare the power, performance and area trade-off of low voltage DNN accelerators.

DNN	f_{clk} (GHz)	VDD (V)	EDAC bit(s)	p_{max}	Top-1 Acc. (%)	Latency (ms)							
						WS				OS			
						256	128	64	16	256	128	64	16
MB-V2	0.7	0.90	N/A	1.11×10^{-25}	71.65	0.4902	0.699	1.0852	4.6015	0.2893	0.4362	0.7669	3.7899
		0.60	N/A	2.13×10^{-08}	71.65	0.4902	0.699	1.0852	4.6015	0.2893	0.4362	0.7669	3.7899
		0.55	N/A	3.30×10^{-05}	68.51	0.4902	0.699	1.0852	4.6015	0.2893	0.4362	0.7669	3.7899
			32	9.96×10^{-06}	70.36	0.4902	0.699	1.0852	4.6015	0.2893	0.4362	0.7669	3.7899
			32–30	2.84×10^{-06}	71.34	0.4902	0.699	1.0852	4.6015	0.2893	0.4362	0.7669	3.7899
		0.50	N/A	0.0309	0.06	0.4902	0.699	1.0852	4.6015	0.2893	0.4362	0.7669	3.7899
			32–24	0.0001	71.36	0.4902	0.699	1.0852	4.6015	0.2893	0.4362	0.7669	3.7899
	1.0	0.90	N/A	5.19×10^{-18}	71.73	0.3431	0.4759	0.7596	3.2211	0.2025	0.3053	0.5368	2.6530
		0.70	N/A	1.38×10^{-07}	71.73	0.3431	0.4759	0.7596	3.2211	0.2025	0.3053	0.5368	2.6530
		0.65	N/A	6.36×10^{-05}	68.12	0.3431	0.4759	0.7596	3.2211	0.2025	0.3053	0.5368	2.6530
			32	1.32×10^{-05}	70.42	0.3431	0.4759	0.7596	3.2211	0.2025	0.3053	0.5368	2.6530
			32–30	2.59×10^{-06}	71.36	0.3431	0.4759	0.7596	3.2211	0.2025	0.3053	0.5368	2.6530
		0.55	N/A	0.9343	0.05	0.3431	0.4759	0.7596	3.2211	0.2025	0.3053	0.5368	2.6530
			32–22	0.0021	71.08	0.5269	0.7122	1.0639	3.7575	0.3299	0.4718	0.7636	3.1614

Table 8.10: MobileNet-V2 latency results summary for different voltage, frequency, systolic array sizes, and dataflows.

DNN	f_{clk} (GHz)	VDD (V)	EDAC bit(s)	p_{max}	Top-1 Acc. (%)	Energy Consumption (mJ)							
						WS				OS			
						256	128	64	16	256	128	64	16
MB-V2	0.7	0.90	N/A	1.11×10^{-25}	71.65	0.5902	0.3050	0.2139	0.1700	0.3997	0.2462	0.1751	0.1659
		0.60	N/A	2.13×10^{-08}	71.65	0.1883	0.1014	0.0737	0.0603	0.1305	0.0836	0.0653	0.0591
		0.55	N/A	3.30×10^{-05}	68.51	0.1514	0.0818	0.0596	0.0489	0.1051	0.0676	0.0542	0.0479
			32	9.96×10^{-06}	70.36	0.152	0.082	0.0597	0.0489	0.1055	0.0676	0.0548	0.0479
			32–30	2.84×10^{-06}	71.34	0.1526	0.0822	0.0597	0.0488	0.1058	0.0677	0.0548	0.0478
		0.50	N/A	0.0309	0.06	0.1205	0.0653	0.0477	0.0392	0.0838	0.054	0.0439	0.0384
			32–24	0.0001	71.36	0.1213	0.0655	0.0477	0.0391	0.0842	0.0541	0.0438	0.0383
	1.0	0.90	N/A	5.19×10^{-18}	71.73	0.4522	0.2522	0.1888	0.1582	0.3188	0.2114	0.1749	0.1552
		0.70	N/A	1.38×10^{-07}	71.73	0.2204	0.128	0.0985	0.0844	0.1589	0.109	0.0921	0.083
		0.65	N/A	6.36×10^{-05}	68.12	0.1807	0.1055	0.0815	0.07	0.1306	0.09	0.0762	0.0688
			32	1.32×10^{-05}	70.42	0.1838	0.1079	0.0836	0.072	0.1333	0.0923	0.0784	0.0708
			32–30	2.59×10^{-06}	71.36	0.1839	0.1079	0.0836	0.072	0.1333	0.0922	0.0783	0.0708
		0.55	N/A	0.9343	0.05	0.1184	0.0696	0.0541	0.0467	0.086	0.0597	0.0507	0.0459
			32–22	0.0021	71.08	0.2001	0.0976	0.0649	0.0498	0.1393	0.0789	0.0829	0.0488

Table 8.11: MobileNet-V2 energy results summary for different voltage, frequency, systolic array sizes, and dataflows.

DNN	f_{clk} (GHz)	VDD (V)	EDAC bit(s)	p_{max}	Top-1 Acc. (%)	Latency (ms)							
						WS				OS			
						256	128	64	16	256	128	64	16
B4	0.7	0.90	N/A	4.78×10^{-25}	82.46	2.9495	4.0084	6.2793	26.0771	1.5416	2.2844	3.8889	21.3768
		0.60	N/A	7.76×10^{-08}	82.38	2.9495	4.0084	6.2793	26.0771	1.5416	2.2844	3.8889	21.3768
		0.55	N/A	0.0001	21.95	2.9495	4.0084	6.2793	26.0771	1.5416	2.2844	3.8889	21.3768
			35	5.88×10^{-05}	69.89	2.9496	4.0085	6.2793	26.0771	1.5416	2.2844	3.8889	21.3769
			35–31	5.36×10^{-06}	81.48	2.9496	4.0085	6.2794	26.0771	1.5417	2.2845	3.889	21.3769
		0.50	N/A	0.0753	0.16	2.9495	4.0084	6.2793	26.0771	1.5416	2.2844	3.8889	21.3768
			35–25	0.0002	81.66	3.0943	4.1584	6.4327	26.2337	1.6707	2.4243	4.0365	21.5331
	1.0	0.90	N/A	3.77×10^{-17}	82.4	2.0647	2.8059	4.3955	18.2539	1.0791	1.5991	2.7222	14.9638
		0.70	N/A	7.55×10^{-07}	81.97	2.0647	2.8059	4.3955	18.2539	1.0791	1.5991	2.7222	14.9638
		0.65	N/A	0.0003	0.33	2.0647	2.8059	4.3955	18.2539	1.0791	1.5991	2.7222	14.9638
			35	0.0001	77.52	2.0647	2.8059	4.3956	18.254	1.0792	1.5991	2.7223	14.9638
			35–31	5.89×10^{-06}	81.78	2.0648	2.806	4.3956	18.2541	1.0792	1.5992	2.7223	14.9639
		0.60	N/A	0.0677	0.16	2.0647	2.8059	4.3955	18.2539	1.0791	1.5991	2.7222	14.9638
			35–26	6.57×10^{-05}	81.68	2.1135	2.8553	4.4453	18.3041	1.1259	1.6473	2.7713	15.0138

Table 8.12: EfficientNet-B4 latency results summary for different voltage, frequency, systolic array sizes, and dataflows.

DNN	f_{clk} (GHz)	VDD (V)	EDAC bit(s)	p_{max}	Top-1 Acc. (%)	Energy Consumption (mJ)							
						WS				OS			
						256	128	64	16	256	128	64	16
B4	0.7	0.90	N/A	4.78×10^{-25}	82.46	3.6372	1.8571	1.3001	1.0378	2.289	1.4399	1.1529	1.0133
		0.60	N/A	7.76×10^{-08}	82.38	1.155	0.6129	0.4433	0.3634	0.7456	0.4863	0.3986	0.356
		0.55	N/A	0.0001	21.95	0.9341	0.4998	0.3639	0.3	0.6061	0.3983	0.3281	0.2939
			35	5.88×10^{-05}	69.89	0.9345	0.4976	0.361	0.2967	0.6047	0.3956	0.325	0.2906
			35–31	5.36×10^{-06}	81.48	0.936	0.4976	0.3605	0.296	0.6051	0.3953	0.3243	0.2899
		0.50	N/A	0.0753	0.16	0.7439	0.3997	0.292	0.2414	0.484	0.3193	0.2636	0.2365
			35–25	0.0002	81.66	0.7718	0.4037	0.2896	0.2366	0.5049	0.3216	0.2608	0.2318
	1.0	0.90	N/A	3.77×10^{-17}	82.4	2.7257	1.4796	1.0899	0.9072	1.7818	1.1875	0.9865	0.8888
		0.70	N/A	7.55×10^{-07}	81.97	1.3151	0.7385	0.5594	0.474	0.8794	0.6038	0.5107	0.4654
		0.65	N/A	0.0003	0.33	1.0791	0.6096	0.4628	0.3941	0.7244	0.4998	0.4239	0.387
			35	0.0001	77.52	1.1154	0.6416	0.4934	0.424	0.7575	0.5308	0.4542	0.417
			35–31	5.89×10^{-06}	81.78	1.1172	0.6417	0.493	0.4234	0.7581	0.5306	0.4537	0.4163
		0.60	N/A	0.0677	0.16	0.8783	0.4988	0.3801	0.3246	0.5916	0.4101	0.3487	0.3189
			35–26	6.57×10^{-05}	81.68	0.9232	0.5275	0.4043	0.3467	0.6312	0.4373	0.3725	0.3414

Table 8.13: EfficientNet-B4 energy results summary for different voltage, frequency, systolic array sizes, and dataflows.

DNN	f_{clk} (GHz)	VDD (V)	EDAC bit(s)	p_{max}	Top-1 Acc. (%)	Latency (ms)							
						WS				OS			
						256	128	64	16	256	128	64	16
B8	0.7	0.90	N/A	9.83×10^{-25}	85.59	9.1366	13.0655	21.7665	105.0692	4.6149	7.1337	12.8118	84.7014
		0.60	N/A	1.45×10^{-07}	85.59	9.1366	13.0655	21.7665	105.0692	4.6149	7.1337	12.8118	84.7014
		0.55	N/A	0.0002	74.65	9.1366	13.0655	21.7665	105.0692	4.6149	7.1337	12.8118	84.7014
			36	0.0001	76.12	9.1366	13.0656	21.7665	105.0693	4.615	7.1338	12.8119	84.7014
			36–32	9.96×10^{-06}	84.66	9.1369	13.0658	21.7668	105.0695	4.6152	7.134	12.8121	84.7017
	1.0	0.90	N/A	1.01×10^{-16}	85.51	6.3956	9.1459	15.2365	73.5485	3.2305	4.9936	8.9683	59.2909
		0.70	N/A	1.73×10^{-06}	85.45	6.3956	9.1459	15.2365	73.5485	3.2305	4.9936	8.9683	59.2909
		0.65	N/A	0.0006	2.87	6.3956	9.1459	15.2365	73.5485	3.2305	4.9936	8.9683	59.2909
			36	0.0003	13.99	6.3957	9.146	15.2366	73.5486	3.2306	4.9937	8.9684	59.2911
			36–31	5.89×10^{-06}	84.78	6.3963	9.1465	15.2372	73.5491	3.2311	4.9943	8.9689	59.2916

Table 8.14: EfficientNet-B8 latency results summary for different voltage, frequency, systolic array sizes, and dataflows.

DNN	f_{clk} (GHz)	VDD (V)	EDAC bit(s)	p_{max}	Top-1 Acc. (%)	Energy Consumption (mJ)							
						WS				OS			
						256	128	64	16	256	128	64	16
B8	0.7	0.90	N/A	9.83×10^{-25}	85.59	12.8492	7.441	5.6863	4.8251	8.5	6.0023	5.1354	4.7209
		0.60	N/A	1.45×10^{-07}	85.59	4.1626	2.515	1.9804	1.7181	2.841	2.078	1.8132	1.6865
		0.55	N/A	0.0002	74.65	3.3547	2.0348	1.6065	1.3969	2.2961	1.6847	1.4725	1.371
			36	0.0001	76.12	4.1303	2.373	1.8027	1.5223	2.7348	1.9124	1.627	1.4905
			36–32	9.96×10^{-06}	84.66	4.1338	2.3717	1.8	1.5188	2.7347	1.91	1.6238	1.3648
	1.0	0.90	N/A	1.01×10^{-16}	85.51	9.8429	6.0573	4.8293	4.2298	6.7981	5.0497	4.4429	4.1528
		0.70	N/A	1.73×10^{-06}	85.45	4.8096	3.0575	2.4893	2.2124	3.4033	2.5923	2.3108	2.1762
		0.65	N/A	0.0006	2.87	3.9552	2.5281	2.0653	1.8399	2.81	2.1493	1.9199	1.8103
			36	0.0003	13.99	4.2819	2.8419	2.3748	2.1473	3.1268	2.4597	2.2282	2.1175
			36–31	5.89×10^{-06}	84.78	4.2859	2.8393	2.3701	2.1416	3.1258	2.4555	2.2229	2.1117

Table 8.15: EfficientNet-B8 energy results summary for different voltage, frequency, systolic array sizes, and dataflows.

Chapter 9

Conclusion

As demonstrated in previous chapters, analyzing the impact of voltage scaling on the classification accuracy is important in error resilient deep learning applications. This analysis helps designers understand the power consumption and accuracy trade-off, and explore the implementation of bit-level error detection and correction (EDAC) techniques in DNN accelerators. A framework is proposed in this work to quantify the DNN inference accuracy as a function of supply voltage and understand the accuracy and energy efficiency trade-offs in low-voltage systolic arrays. This framework is implemented in three steps. First, an error probability modeling methodology is proposed to facilitate the quantification of quality-of-results vs. error rates at the application level, without relying on time-consuming simulations. Since multiplication and accumulation is the basic computational unit in deep learning hardware, the proposed methodology is implemented on 8-bit MAC to evaluate the timing error probability in different operating conditions (clock frequency, supply voltage and noise). Due to different characteristics of the two technologies explored in this work (FinFET and TFET), several important design con-

siderations are highlighted. An important observation is the impact of power supply noise on the timing error probability. The dependence on noise is stronger for TFET technology as the voltage is scaled to near-threshold levels due to exponential dependence of delay on voltage, which increases the sensitivity to voltage fluctuations at such low voltages.

Second, an error resilience framework is designed in PyTorch to investigate the error resilience of a neural network at per-layer and per-bit granularity. In this framework, bit-level errors are injected in all or select layers in a quantized neural network. The error resilience is analyzed for several state-of-the-art DNNs such as ResNet-18 [63], MobileNetV2 [64] and EfficientNet [65]. The key observation from per-layer analysis is that the error resilience of different layers in the network is independent of the computational complexity of the network, such that some layers (such as the last convolutional layer) are more resilient and implement higher number of *MOps*. These robust layers can enable significant improvements in the energy efficiency of DNN hardware. The per-bit error resilience analysis indicates that the inference accuracy can be improved significantly if 1–5 higher order bits (out of 10–12 bits) are error-free. These results can be leveraged to implement efficient error detection and correction techniques and understand the trade-offs between accuracy improvement and design overheads.

In the last step, the timing probability model and the error resilience framework are combined to quantify the DNN accuracy as a function of supply voltage. The per-bit error rate at the output of a processing element in a systolic array is determined from the proposed probability model while considering 5% supply noise and different operating voltages and clock frequency. The error rate for a layer output is determined by accumulating the error rate based on the number of multiply-add op-

erations for each output in the respective layer. These per-bit, per-layer error rates are then used as an input to the error resilience framework to evaluate the inference accuracy. For MobileNetV2, EfficientNet-B4 and B8 DNNs, it is observed that the operating voltage for entire network can be scaled to 0.6V and 0.7V for 700 MHz and 1 GHz clock frequency, respectively, without affecting the DNN accuracy. This voltage reduction provides a 64.28–68.09% and 47.6–51.26% improvement for 700 MHz and 1 GHz frequency, respectively. The per-layer voltage scaling is also explored to exploit the error resilience of compute-intensive layers in the network. In addition, different systolic array sizes are compared to analyze the dynamic and leakage power consumption in the systolic array, which can be quantified based on the array utilization and per-layer compute cycle count (obtained from SCALE-Sim [72]).

The observations and conclusions drawn from this framework are used to further explore the bit-level error detection and correction (EDAC) techniques and quantify the trade-offs between accuracy improvement and design overhead of additional hardware for EDAC implementation. In this work, iRazor [77] is used to correct the timing errors in the higher order bits within each layer output in the systolic array. At lower voltages (e.g. $<$0.7V at 1 GHz frequency), the higher timing error probabilities increase the loss in accuracy which can be recovered through per-bit error correction at the cost of latency and energy overhead. For instance, EDAC in a systolic array performing inference of an input image for EfficientNet-B4 at 0.6V and 1 GHz frequency, adds a latency overhead of $\approx$50μs, while the total energy consumption of the systolic array is 26.64–29.8% lower than the energy consumption at 0.7V (no EDAC). It is also observed that voltage scaling at higher frequency is limited by the significant increase in timing error probability, with and

without EDAC. To understand this limitation, the minimum voltage at which a DNN can recover the loss in accuracy with EDAC is evaluated. This minimum voltage varies for different networks and it depends on the accumulator size since the error probability is higher for accumulators with higher bit-widths. For instance, at 1 GHz clock frequency, MobileNetV2 can be scaled to 0.55V while EfficientNet-B4 can only be scaled to 0.6V for 1 GHz frequency. Below these supply voltages, the loss in accuracy cannot be recovered through per-bit error correction. Therefore, the work presented in this dissertation highlights the importance of analyzing the impact of low voltage operation in DNN accelerator structures and helps designers understand the complex trade-offs between accuracy degradation and improvement in energy efficiency with supply voltage scaling, particularly in the context of bit-level error detection and correction.

In future work, this framework can be used to explore the overhead and trade-offs of other error compensation techniques, such [91] in which Ji et al. propose a compensation-MAC that includes additional adders in existing MAC to compensate errors in multiplier data-path. This is an interesting approach as it removes latency overhead in the systolic array due to timing errors, and can be implemented to correct bit-level errors. Though Li et al. explore the accuracy and energy trade-offs with this error compensation technique, the framework proposed in this dissertation can further push the boundaries on low voltage operation by exploiting the error resilience of different layers. Moreover, the designers can use it to explore the overhead and trade-offs of different systolic array sizes and neural network structures. The observations and conclusions drawn from the analysis can also facilitate the design and optimization of low voltage deep learning hardware. In conclusion, this dissertation exploits the inherent resilience of neural networks to achieve aggressive

voltage scaling with negligible impact on inference accuracy in error-aware, energy efficient DNN accelerators.

Bibliography

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, and et al., "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, June 2017, pp. 1–12.

[2] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 8:1–8:12.

[3] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, May 2013, pp. 1–9.

[4] C. Chen, J. Choi, K. Gopalakrishnan, V. Srinivasan, and S. Venkataramani, "Exploiting approximate computing for deep learning acceleration," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 821–826.

[5] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[6] M. Rathore, P. Milder, and E. Salman, "Error probability models for voltage-scaled multiply-accumulate units," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 7, pp. 1665–1675, 2020.

[7] S. Datta, R. Bijesh, H. Liu, D. Mohata, and V. Narayanan, "Tunnel Transistors for Low Power Logic," in *2013 IEEE Compound Semiconductor Integrated Circuit Symposium (CSICS)*, Oct 2013, pp. 1–4.

[8] Y. Zhang, M. Khayatzadeh, K. Yang, M. Saligane, N. Pinckney, M. Alioto, D. Blaauw, and D. Sylvester, "irazor: Current-based error detection and correction scheme for pvt variation in 40-nm arm cortex-r4 processor," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 619–631, 2017.

[9] 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.

[10] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[11] 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.

[12] R. C. Gonzalez, "Deep Convolutional Neural Networks [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 35, no. 6, pp. 79–87, Nov 2018.

[13] A. Hidaka and T. Kurita, "Consecutive Dimensionality Reduction by Canonical Correlation Analysis for Visualization of Convolutional Neural Networks," in *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications*, vol. 2017, 12 2017, pp. 160–167.

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

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[16] F. Shahbaz, "Five Powerful CNN Architectures," https://medium.com/datadriveninvestor/five-powerful-cnn-architectures-b939c9ddd57b, Nov 2018.

[17] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, Jan 2017, pp. 127–138.

[18] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, "DLAU: A Scalable Deep Learning Accelerator Unit on FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2016.

[19] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh, "From high-level deep neural models to FPGAs," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2016, pp. 1–12.

[20] Y. Shen, M. Ferdman, and P. Milder, "Maximizing CNN accelerator efficiency through resource partitioning," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, June 2017, pp. 535–547.

[21] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv*, Oct 2015.

[22] 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.

[23] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[24] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," *arXiv*, Feb 2016.

[25] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both Weights and Connections for Efficient Neural Network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[26] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 1–13.

[27] E. Salman and E. G. Friedman, *High Performance Integrated Circuit Design*. McGraw Hill Professional, 2012.

[28] W. P. Maszara and M. R. Lin, "FinFETs - Technology and circuit design challenges," in *2013 Proceedings of the ESSCIRC (ESSCIRC)*, Sep 2013, pp. 3–8.

[29] C. Sitik, W. Liu, B. Taskin, and E. Salman, "Design Methodology for Voltage-Scaled Clock Distribution Networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3080–3093, Oct 2016.

[30] C. Sitik, E. Salman, L. Filippini, S. J. Yoon, and B. Taskin, "FinFET-Based Low-Swing Clocking," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 2, pp. 1–13, 2015.

[31] M. Rathore, W. Liu, E. Salman, C. Sitik, and B. Taskin, "A Novel Static D-Flip-Flop Topology for Low Swing Clocking," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, 2015, pp. 301–306.

[32] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold voltage (NTV) design Opportunities and challenges," in *DAC Design Automation Conference 2012*, June 2012, pp. 1149–1154.

[33] N. Pinckney, S. Jeloka, R. Dreslinski, T. Mudge, D. Sylvester, D. Blaauw, L. Shifren, B. Cline, and S. Sinha, "Impact of FinFET on Near-Threshold Voltage Scalability," *IEEE Design & Test*, vol. 34, no. 2, pp. 31–38, April 2017.

[34] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, Feb 2010.

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

[36] V. K. Chippa, S. Venkataramani, K. Roy, and A. Raghunathan, "StoRM: A Stochastic Recognition and Mining processor," in *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Aug 2014, pp. 39–44.

[37] T. Cui, J. Li, Y. Wang, S. Nazarian, and M. Pedram, "An Exploration of Applying Gate-Length-Biasing Techniques to Deeply-Scaled FinFETs Operating in Multiple Voltage Regimes," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 2, pp. 172–183, April 2018.

[38] R. Wang, X. Jiang, S. Guo, and R. Huang, "How close to the CMOS voltage scaling limit for FinFET technology? Near-threshold computing and stochastic computing," in *2017 IEEE 12th International Conference on ASIC (ASICON)*. IEEE, Oct 2017, pp. 56–59.

[39] "International Technology Roadmap for Semiconductors (ITRS)," 2015.

[40] F. Balestra, "Nanoscale FETs for high performance and ultra low power operation at the end of the Roadmap," in *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, Oct 2018, pp. 1–4.

[41] N. Collaert, A. Alian, H. Arimura, G. Boccardi, G. Eneman, J. Franco, T. Ivanov, D. Lin, R. Loo, C. Merckling, J. Mitard, M. Pourghaderi, R. Rooyackers, S. Sioncke, J. Sun, A. Vandooren, A. Veloso, A. Verhulst, N. Waldron, L. Witters, D. Zhou, K. Barla, and A.-Y. Thean, "Ultimate nano-electronics: New materials and device concepts for scaling nano-electronics beyond the Si roadmap," *Microelectronic Engineering*, vol. 132, pp. 218–225, Jan 2015.

[42] N. K. Kranthi and M. Shrivastava, "ESD Behavior of Tunnel FET Devices," *IEEE Transactions on Electron Devices*, vol. 64, no. 1, pp. 28–36, Jan 2017.

[43] S. Datta, R. Pandey, M. Barth, R. Bijesh, and H. Liu, "Tunnel FETs - the Next Switch in the Post FinFET Era?" July 2016.

[44] S. Datta, H. Liu, and V. Narayanan, "Tunnel FET technology: A reliability perspective," *Microelectronics Reliability*, vol. 54, no. 5, pp. 861–874, May 2014.

[45] M. Sanaullah and M. H. Chowdhury, "Subthreshold swing characteristics of multilayer MoS2 tunnel FET," in *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2015, pp. 1–4.

[46] H. Liu, S. Datta, and V. Narayanan, "Steep switching tunnel FET: A promise to extend the energy efficient roadmap for post-CMOS digital and analog/RF applications," in *International Symposium on Low Power Electronics and Design (ISLPED)*, Sep 2013, pp. 145–150.

[47] G. Dewey, B. Chu-Kung, J. Boardman, J. M. Fastenau, J. Kavalieros, R. Kotlyar, W. K. Liu, D. Lubyshev, M. Metz, N. Mukherjee, P. Oakey, R. Pillarisetty, M. Radosavljevic, H. W. Then, and R. Chau, "Fabrication, characterization, and physics of IIIV heterojunction tunneling Field Effect Transistors (H-TFET) for steep sub-threshold swing," in *2011 International Electron Devices Meeting*, Dec 2011, pp. 1–33.

[48] T. A. Ameen, H. Ilatikhameneh, P. Fay, A. Seabaugh, R. Rahman, and G. Klimeck, "Alloy Engineered Nitride Tunneling Field-Effect Transistor: A Solution for the Challenge of Heterojunction TFETs," *IEEE Transactions on Electron Devices*, pp. 1–7, 2018.

[49] E. Memisevic, J. Svensson, E. Lind, and L. Wernersson, "Vertical Nanowire TFETs With Channel Diameter Down to 10 nm and Point SMIN of 35 mV/Decade," *IEEE Electron Device Letters*, vol. 39, no. 7, pp. 1089–1091, July 2018.

[50] V. Saripalli, G. Sun, A. Mishra, Y. Xie, S. Datta, and V. Narayanan, "Exploiting Heterogeneity for Energy Efficiency in Chip Multiprocessors," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 2, pp. 109–119, June 2011.

[51] M. Rathore and E. Salman, "Error Probability Models to Facilitate Approximate Computing in TFET based Circuits," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[52] 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 *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, June 2018, pp. 1–6.

[53] J. J. Zhang and S. Garg, "FATE: Fast and Accurate Timing Error Prediction Framework for Low Power DNN Accelerator Design," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2018, pp. 1–8.

[54] X. Jiao, M. Luo, J. Lin, and R. K. Gupta, "An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2017, pp. 945–950.

[55] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "Predictive technology model," 2002.

[56] H. Liu, V. Saripalli, V. Narayanan, and S. Datta, "III-V Tunnel FET Model," 2015.

[57] H. Wang and E. Salman, "Closed-Form Expressions for I/O Simultaneous Switching Noise Revisited," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 2, pp. 769–773, Feb 2017.

[58] T. Enami, S. Ninomiya, and M. Hashimoto, "Statistical Timing Analysis Considering Spatially and Temporally Correlated Dynamic Power Supply Noise," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 4, pp. 541–553, April 2009.

[59] S. M. Ross, *A first course in probability*. Pearson Prentice Hall, 2010.

[60] M. Saint-Laurent and M. Swaminathan, "Impact of power-supply noise on timing in high-frequency microprocessors," *IEEE Transactions on Advanced Packaging*, vol. 27, no. 1, pp. 135–144, Feb 2004.

[61] M. Ha, Y. Byun, S. Moon, Y. Lee, and S. Lee, "Layerwise Buffer Voltage Scaling for Energy-Efficient Convolutional Neural Network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 1, pp. 1–10, 2021.

[62] Y. Chen, Y. Zhu, F. Qiao, J. Han, Y. Liu, and H. Yang, "Evaluating data resilience in CNNs from an approximate memory perspective," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, 2017, pp. 89–94.

[63] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[64] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[65] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *CoRR*, vol. abs/1905.11946, 2019. [Online]. Available: http://arxiv.org/abs/1905.11946

[66] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[67] W. Choi, D. Shin, J. Park, and S. Ghosh, "Sensitivity based error resilient techniques for energy efficient deep neural network accelerators," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[68] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, "Ares: A Framework for Quantifying the Resilience of Deep Neural Networks," in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18. New York, NY, USA: ACM, 2018, pp. 17:1–17:6. [Online]. Available: http://doi.acm.org/10.1145/3195970.3195997

[69] E. Ozen and A. Orailoglu, "Boosting Bit-Error Resilience of DNN Accelerators Through Median Feature Selection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3250–3262, 2020.

[70] A. Xichen, "pytorch-playground."

[71] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le, "MnasNet: Platform-Aware Neural Architecture Search for Mobile," *CoRR*, vol. abs/1807.11626, 2018. [Online]. Available: http://arxiv.org/abs/1807.11626

[72] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "SCALE-Sim: Systolic CNN Accelerator Simulator," 2019.

[73] Y.-H. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, 2017.

[74] Y. Chen, T. Chen, Z. Xu, N. Sun, and O. Temam, "Diannao family: energy-efficient hardware accelerators for machine learning," *Communications of the ACM*, vol. 59, no. 11, pp. 105–112, 2016.

[75] M. Sankaradas, V. Jakkula, S. Cadambi, S. Chakradhar, I. Durdanovic, E. Cosatto, and H. P. Graf, "A massively parallel coprocessor for convolutional neural networks," in *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*. IEEE, 2009, pp. 53–60.

[76] R. Venkatesan, Y. S. Shao, M. Wang, J. Clemons, S. Dai, M. Fojtik, B. Keller, A. Klinefelter, N. Pinckney, P. Raina *et al.*, "Magnet: A modular accelerator generator for neural networks," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.

[77] Y. Zhang, M. Khayatzadeh, K. Yang, M. Saligane, N. Pinckney, M. Alioto, D. Blaauw, and D. Sylvester, "irazor: Current-based error detection and correction scheme for pvt variation in 40-nm arm cortex-r4 processor," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 619–631, Feb 2018.

[78] P. Macken, M. Degrauwe, M. Van Paemel, and H. Oguey, "A voltage reduction technique for digital systems," in *1990 37th IEEE International Conference on Solid-State Circuits*. IEEE, 1990, pp. 238–239.

[79] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proceedings Eighth International Symposium on High Performance Computer Architecture*. IEEE, 2002, pp. 29–40.

[80] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark, "Voltage and frequency control with adaptive reaction time in multiple-clock-domain processors," in *11th International Symposium on High-Performance Computer Architecture*. IEEE, 2005, pp. 178–189.

[81] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core dvfs using on-chip switching regulators," in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*. IEEE, 2008, pp. 123–134.

[82] L. Wang, G. Von Laszewski, J. Dayal, and F. Wang, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE, 2010, pp. 368–377.

[83] C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the dvfs technique for cloud datacenters," *Future Generation Computer Systems*, vol. 37, pp. 141–147, 2014.

[84] C. Albea, D. Puschini, S. Lesecq, and Y. Akgul, "Advanced coupled voltage-frequency control for power efficient dvfs management," in *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2012, pp. 2168–2173.

[85] W. Shan, X. Shang, L. Shi, W. Dai, and J. Yang, "Timing error prediction avfs with detection window tuning for wide-operating-range ics," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 7, pp. 933–937, July 2018.

[86] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner, "Razor: circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, no. 6, pp. 10–20, 2004.

[87] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, "Razor ii: In situ error detection and correction for pvt and ser tolerance," in *2008 IEEE International Solid-State Circuits Conference-Digest of Technical Papers*. IEEE, 2008, pp. 400–622.

[88] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. Harris, D. Blaauw, and D. Sylvester, "Bubble razor: An architecture-independent approach to timing-error detection and correction," in *2012 IEEE International Solid-State Circuits Conference*. IEEE, 2012, pp. 488–490.

[89] M. Choudhury, V. Chandra, K. Mohanram, and R. Aitken, "TIMBER: Time borrowing and error relaying for online timing error resilience," in *2010 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2010, pp. 1554–1559.

[90] H. Reyserhove and W. Dehaene, "Design margin elimination in a near-threshold timing error masking-aware 32-bit ARM Cortex M0 in 40nm CMOS," in *ESSCIRC 2017 - 43rd IEEE European Solid State Circuits Conference*, Sep. 2017, pp. 155–158.

[91] D. Ji, D. Shin, and J. Park, "An error compensation technique for low-voltage dnn accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2020.