BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

DESIGNING ENERGY-EFFICIENT COMPUTING SYSTEMS USING EQUALIZATION AND MACHINE LEARNING

by

ZAFAR TAKHIROV

Specialist, Russian-Tajik (Slavonic) University, 2008
M.S., Boston University, 2012

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2018

© 2018 by
ZAFAR TAKHIROV
All rights reserved

Approved by

First Reader

Ajay J. Joshi, PhD
Associate Professor of Electrical and Computer Engineering

Second Reader

Venkatesh Saligrama, PhD
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Computer Science

Third Reader

Ayse K. Coskun, PhD
Associate Professor of Electrical and Computer Engineering

Fourth Reader

Joseph Wang, PhD
Research Scientist,
Amazon.com, Inc.

Fifth Reader

Michel Kinsy, PhD
Assistant Professor of Electrical and Computer Engineering

Duo parabolas me servavit:
Per aspera ad astra, et...
Lingua latina non verpa canina...

Acknowledgments

First and foremost, I am endlessly thankful to two people without whom this work would not have been possible – my advisor, professor Ajay Joshi, and my lovely wife, Alice. Their help and support, despite my constant resistance, was the only thing that brought me where I am right now. Ajay's guidance and help was crucial in every aspect of my academic work, while Alice's unconditional support will be the sole reason I managed to survive this confusing, head-bashing, excruciating path.

I would like to express gratitude to my parents, Gulnara and Mahmadzahir, as they were the ones who encouraged me to take on this path, and were very helpful both morally and financially throughout my life. Both of my brothers, Akbar and Askar, were there when I needed them the most, and thus should always be thanked for believing in me. On the same note, I would like to thank the Department of Electrical and Computer Engineering of Boston University, and especially the administration and the advisors of the Late Entry Accelerated Program (LEAP) for providing me with an opportunity to achieve something I never hoped to engage.

I would also like to use these pages to mention the people I collaborated with academically and professionally. It goes without saying that without numerous discussions with Joe Wang and Marcia S. Louis, the later part of this work would have been dull and unimpressive. My only regret is that I didn't meet them earlier in my career. Also, Yuhong Huang was incredibly supportive both as a manager and as a friend during my time at Analog Devices, Inc. She was an incredible mentor, and she is continuing being a great family friend. I would like to mention my colleagues in the infamous PHO 340 who I spent great amount of time with: discussing completely irrelevant topics (Boyou), being roommates and three-am-walk-home buddies (Chao), partying like it was the last thing in our life (Mahmoud), arguing till our noses started bleeding (Schuyler), learning how to curse in Chinese (Yenai), playing chess (Saiful

– I will still always win), and of course I would not forget Leila for keeping my awesomeness in check and bringing me down to earth. The same goes to the rest of the PHO 340, where I spent the last several years and made a lot of friends. This goes to all members of the PEAC and CAAD labs without exception.

Many more people played an indirect, but crucial role during my work. In particular without the help of Aksana, Sveta, Leo, and Vlad I would probably spend a lot more nights sleeping on Fenway park benches. Ato and his team were the motivation and inspiration for all those sleepless nights me working with them, and still having the energy to work on more projects in the morning.

DESIGNING ENERGY-EFFICIENT COMPUTING SYSTEMS USING EQUALIZATION AND MACHINE LEARNING

ZAFAR TAKHIROV

Boston University, College of Engineering, 2018

Major Professor: Ajay Joshi, PhD
Associate Professor of Electrical and Computer Engineering

ABSTRACT

As technology scaling slows down in the nanometer CMOS regime and mobile computing becomes more ubiquitous, designing energy-efficient hardware for mobile systems is becoming increasingly critical and challenging. Although various approaches like near-threshold computing (NTC), aggressive voltage scaling with shadow latches, etc. have been proposed to get the most out of limited battery life, there is still no "silver bullet" to increasing power-performance demands of the mobile systems. Moreover, given that a mobile system could operate in a variety of environmental conditions, like different temperatures, have varying performance requirements, etc., there is a growing need for designing tunable/reconfigurable systems in order to achieve energy-efficient operation. In this work we propose to address the energy-efficiency problem of mobile systems using two different approaches: circuit tunability and distributed adaptive algorithms.

Inspired by the communication systems, we developed feedback equalization based digital logic that changes the threshold of its gates based on the input pattern. We

showed that feedback equalization in static complementary CMOS logic enabled up to 20% reduction in energy dissipation while maintaining the performance metrics. We also achieved 30% reduction in energy dissipation for pass-transistor digital logic (PTL) with equalization while maintaining performance. In addition, we proposed a mechanism that leverages feedback equalization techniques to achieve near optimal operation of static complementary CMOS logic blocks over the entire voltage range from near threshold supply voltage to nominal supply voltage. Using energy-delay product (EDP) as a metric we analyzed the use of the feedback equalizer as part of various sequential computational blocks. Our analysis shows that for near-threshold voltage operation, when equalization was used, we can improve the operating frequency by up to 30%, while the energy increase was less than 15%, with an overall EDP reduction of ≈10%. We also observe an EDP reduction of close to 5% across entire above-threshold voltage range.

On the distributed adaptive algorithm front, we explored energy-efficient hardware implementation of machine learning algorithms. We proposed an adaptive classifier that leverages the wide variability in data complexity to enable energy-efficient data classification operations for mobile systems. Our approach takes advantage of varying classification hardness across data to dynamically allocate resources and improve energy efficiency. On average, our adaptive classifier is ≈100× more energy efficient but has ≈1% higher error rate than a complex radial basis function classifier and is ≈10× less energy efficient but has ≈40% lower error rate than a simple linear classifier across a wide range of classification data sets. We also developed a field of groves (FoG) implementation of random forests (RF) that achieves an accuracy comparable to Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) under tight energy budgets. The FoG architecture takes advantage of the fact that in random forests a small portion of the weak classifiers (decision trees) might be

sufficient to achieve high statistical performance. By dividing the random forest into smaller forests (Groves), and conditionally executing the rest of the forest, FoG is able to achieve much higher energy efficiency levels for comparable error rates. We also take advantage of the distributed nature of the FoG to achieve high level of parallelism. Our evaluation shows that at maximum achievable accuracies FoG consumes ≈1.48×, ≈24×, ≈2.5×, and ≈34.7× lower energy per classification compared to conventional RF, SVM_{RBF}, Multi-Layer Perceptron Network (MLP), and CNN, respectively. FoG is 6.5× less energy efficient than SVM_{LR}, but achieves 18% higher accuracy on average across all considered datasets.

Contents

1 Introduction 1

1.1 Background and Motivation 1

1.2 General Principles for Energy-Efficient System Design 4

1.3 Related Work 12

1.3.1 Circuits and Architectures that Trade-Off Performance, Energy, and Error Rate 12

1.3.2 Adaptive Systems and Machine Learning Hardware 15

1.4 Contributions 20

2 Error Mitigation in Digital Logic Using Feedback Equalization 25

2.1 Introduction 25

2.2 Error Manifestations in Digital Logic 26

2.3 Equalization Techniques 28

2.3.1 Feedback circuit 29

2.3.2 Schmitt trigger 30

2.3.3 Feedback Equalization with Schmitt Trigger (FEST) 31

2.4 Experimental Results 33

2.4.1 Kogge-Stone Adder 33

2.4.2 Finite Impulse Response Filter 36

2.5 Experimental Results for Near-Threshold Design 38

2.5.1 Experimental Setup 38

2.5.2 Experimental Results 39

2.6 Conclusion 41

3 **Energy Efficient Pass-Transistor Logic Using Decision Feedback Equalization** **45**

3.1 Introduction 45

3.2 Equalized Pass-Transistor Logic 47

3.3 Modeling and Design Automation of E-PTL 50

3.3.1 Power Modeling 50

3.3.2 Error Rate Modeling 51

3.3.3 Delay Modeling 56

3.3.4 Optimization Tool-flow 56

3.3.5 Modeling vs. Simulation 57

3.4 Evaluation 58

3.5 Conclusion 60

4 **Adaptive Classification: Energy-Efficient Design for Machine Learning** **61**

4.1 Introduction 61

4.2 Adaptive Classifier – Theory 64

4.2.1 Hardness of an Input and Penalty for Misclassification 64

4.2.2 Training the “chooser” for a fixed energy budget 67

4.2.3 Training the “chooser” for a changing energy budget 71

4.3 Adaptive Classifier – Microarchitecture Description 73

4.4 Evaluation 76

4.4.1 Experimental Setup 78

4.4.2 Experimental Results of an approach with static energy budget 81

4.4.3 Experimental Results of the approach with dynamic energy budget 85

4.5 Discussion 88
4.6 Conclusion 89

5 Random Forest and the Field of Groves 91

5.1 Introduction 91
5.2 Energy-Efficient Computing using Random Forests 93
  5.2.1 Field of Groves Algorithm 96
  5.2.2 Micro-architecture 100
5.3 Evaluation 106
  5.3.1 Experimental Setup 106
  5.3.2 Experimental Results 112
5.4 Conclusion 116

6 Conclusion and Future Work 118

6.1 Summary of Contributions 118
  6.1.1 Feedback Equalization 118
  6.1.2 Machine Learning Accelerators 120
6.2 Future Research Directions 122
  6.2.1 Feed-Forward Equalization in Pass-Transistor Logic 122
  6.2.2 Specialized Architectures for Machine Learning Acceleration . 123
  6.2.3 Bridging the gap between Hardware and Machine Learning . . 124

References 125

Curriculum Vitae 141

List of Tables

2.1 Inverter threshold values 29

2.2 Switching thresholds of different designs 32

2.3 Evaluation results of different designs 44

3.1 Minimum energy comparison of different designs 60

4.1 Notation and variables used in the current work 64

4.2 Pseudo-label assignment example: given different inputs “8”, we generate pseudo-labels depending on the correct predictions of the “core”. 65

4.3 Datasets used and the number of inputs misclassified during the training phase. 66

4.4 Energy and Delay for the adaptive classifier and conventional classifiers for different data sets. Each row corresponds to subplots in Figure 4·5. Energy results are shown in J/classification and delay is represented as μs. 77

4.5 Comparison of the proposed and SEC approaches (Venkataramani et al., 2015a) trained for low energy subject to error rate budget. The time is in seconds. 78

5.1 Notation used in section 5.2.2 100

5.2 Datasets used for FoG evaluation 107

5.3 Accuracy and Energy dissipation for FoG 110

5.4 CPU, GPU, and FPGA Results for FoG 117

List of Figures

1·1 Low-power parallel and pipelined computing 5
1·2 Order of Execution matters 6

2·1 DC response of the feedback equalizer circuit. 27
2·2 Variable threshold inverter circuit (Sridhara et al., 2008). 28
2·3 Inverting Schmitt trigger circuit (Baker, 2004). 30
2·4 Pipelined combinational logic with FEST circuit. 31
2·5 Transients of KSA 34
2·6 4-bit KSA 35
2·7 4-bit, 3-tap FIR 36
2·8 EDP results across entire above-threshold range 40
2·9 Distribution of 24-bit multiplier 41
2·10 Distribution of AES-256 42
2·11 Distribution of RS Decoder 43
2·12 Distribution of an FFT 43

3·1 Schematic diagram of a Equalized Pass-Transistor Logic (E-PTL) . . 47
3·2 Transients of equalized and non-equalized 47
3·3 PDF of noisy PTL 51
3·4 Model of the proposed E-PTL 53
3·5 Model vs. Simulation for 16-bit CLA 55
3·6 16-bit Carry-Look Ahead Adder 57
3·7 Monte-Carlo simulation results for CLA 59

4·1 EDP and Error rate vs. the budget of an adaptive classifier: (a) trained with EDP budget as a feature; (b,c,d) trained to maximize the accuracy while meeting an energy constraint of 10^{-20}, 10^{-18}, and 10^{-16} respectively. 72

4·2 Block diagram of the adaptive classifier. “Chooser” function is marked as ?⃝, and its microarchitecture is the same as the “Easy” block. Data paths are shown as solid lines, while control path is dashed. The linear classifier is implemented as a multiply-accumulate (MAC) block and is equivalent to equation (4.9). The polynomial function implementation uses a MAC to compute higher order components $\alpha^p_{(\cdot)} x(\cdot)^p$ as per equation (4.10). The RBF classifier also uses and MAC and its the exponential function is implemented as a look up table. 74

4·3 Energy delay products for different adaptive classifier designs generated by Aladdin (Shao et al., 2014). 78

4·4 Average overheads of different types of “chooser” functions. The EDP overhead is average EDP contribution of the “chooser” with respect to an adaptive classifier where this “chooser” is implemented. The error rate overhead is the error rate contributed by the “‘chooser” function itself. 82

4·5 Error rate vs. Energy Delay Product (EDP) for adaptive classifier and conventional classifiers. Adaptive classifier can be tuned to achieve any of the intermediate error rate values. 83

4·6 Energy dissipation distribution and the total energy dissipation vs. error rate for different classification problems. 85

4·7 Relative area distribution of subblocks in adaptive classifier (bars) as well as the total area (solid line). 86

4·8 Energy dissipation and accuracy for dynamic adaptive classifier normalized to the adaptive classifier trained for the static energy budget. 87

5·1 Example of a Random Forest 93
5·2 Common Machine Learning Algorithms 94
5·3 Accuracy degradation for MLP and RF 94
5·4 Logical view of RF and FoG. 99
5·5 Field of Groves implementation 101
5·6 FoG processing element (PE) 102
5·7 FoG system design flow diagram 108
5·8 Accuracy and Energy dissipation of different classifiers 111
5·9 Accuracy and EDP of FoG 114
5·10 Example of FoG run-time tuning using the “threshold” variable. . . . 115
5·11 Computational performance comparison between GPU, FPGA, and SIMD CPU 116

List of Abbreviations

AC	Alternating Current
ACU	Analog Computing Unit
AGC	Automatic gain control
AI	Artificial Intelligence
ALU	Arithmetic Logic Unit
ANN	Artificial Neural Network
API	Application Programming Interface
ASIC	Application-specific Integrated Circuit
BJT	Bipolar Junction Transistor
BLAS	Basic Linear Algebra Subprograms
BSD	Berkeley Software Distribution
BU	Boston University
CMOS	Complimentary Metal-oxide-semiconductor
CNN	Convolutional Neural Network
CORDIC	Coordinate Rotation Digital Computer
CPU	Central Processing Unit
CUDA	NVIDIA's Parallel Programming API for GPUs
DBN	Deep Belief Network
DC	Direct Current
DNN	Deep Neural Network
DSP	Digital Signal Processor
DVFS	Dynamic Voltage and Frequency Scaling
E-PTL	Equalized Pass-Transistor Logic
EDEP	Energy-delay-error Product
EDP	Energy-delay Product
EMC	Or Dell EMC^2 – Egan-Marino Corporation
EU	Arbitrary energy dissipation unit
FANN	Fast Artificial Neural Network library
FEST	Feedback Equalization with Schmitt Trigger
FFE	Feedforward Equalization
FoG	Field of Groves accelerator
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
HDL	Hardware Description Language
HW	Hardware

IBM	International Business Machines
ICSG	Boston U. Integrated Circuits and Systems Group
IEEE	The Institute of Electrical and Electronics Engineers
IDC	International Data Corporation
ISA	Instruction Set Architecture
IoT	Internet of Things
JPEG	Joint Photographic Experts Group
LEAP	Boston U. Late Entry Accelerated Program
LSTM	Long Short Term Memory
MAC	Multiply Accumulate
ML	Machine Learning
MLP	Multilayer Perceptron Neural Network
MOSFET	Metal-oxide-semiconductor Field-effect-transistor
MSE	Mean Squared Error
NASA	The National Aeronautics and Space Administration
NN	Neural Network
NTC	Near-Threshold Computing
NTV	Near-Threshold Voltage
OS	Operating System
PC	Personal Computer
PE	Processing Element
PTL	Pass Transistor Logic
PVT	Process, Voltage, Temperature (variations)
RBF	Radial Basis Function
RISC	Reduced Instruction Set Computer
RISC-V	Fifth Generation of RISC Instruction Sets
RNN	Recurrent Neural Network
RTL	Register-transfer Level
RoCC	Rocket Custom Coprocessor
SIMD	Single Instruction Multiple Data
SNR	Signal-to-Noise Ratio
SVM	Support Vector Machine
TPU	Tensor Processing Unit
UART	Universal Asynchronous Receiver/Transmitter
VCD	Value Change Dump
VLSI	Very Large Scale Integration

Chapter 1

Introduction

1.1 Background and Motivation

As we enter the 'Internet of Things' (IoT) regime, the computing industry is placing more and more emphasis on designing energy-efficient mobile systems (Wu, 2015). This increasing emphasis on energy-efficiency is driven by the need for the mobile systems to be able to process large quantities of data in real time (which requires proportional amount of energy) and also the slow development in battery technology. Moreover, these mobile systems are expected to operate in a variety of environments, and hence there is a need to develop tunable/adaptive systems that can dynamically adapt and maximize the system energy efficiency. The focus of this research is to address this energy efficiency problem by developing tunable digital logic circuits as well as developing hardware based on adaptive data processing algorithms.

In addition to IoT, emerging paradigm of "Big Data" is also driving research towards more energy-efficient designs. Big Data is a general name for data sets so large and complex that traditional approaches to computation become inadequate. A good example of "Big Data" is social media analysis – information whether someone "liking" company pages on Facebook or tweeting complaints about products needs to be harvested and processed in order to deliver the best customer interaction. Widespread access to information as well as ever increasing rate of information generation require new energy-efficient data-processing computation units (IDC and EMC2, 2012, Bell et al., 1972).

This problem of Big Data and IoT is non-trivial - devices are getting smaller 100 times every decade (TSensors, 2013), and every year the number of active nodes connected to the Web is increasing by hundreds of thousands (Hilbert and Lopez, 2011, IDC and EMC2, 2013, TSensors, 2013). Fundamentally, this creates a demand for a new design paradigm in the area of mobile computing. Miniaturization of energy sources and "intelligent" power management, although being of research interest, is not growing as fast as the demand for computation, making energy-efficiency an immediate requirement.

Currently, there are multiple parallel efforts in place to design energy-efficient computing systems. At the device level, Complementary Metal Oxide Semiconductor (CMOS) technology scaling enabled us to lower power and improve performance every two years. However, CMOS technology scaling has slowed down and is expected to end in the next 10 years, thus limiting the improvements in energy efficiency (Williams, 2017). Hence, alternate switching devices like Tunnel Field Effect Transistors (TFETs), Carbon Nanotube FETs (CNFETS) are being explored (Richter et al., 2014, Paul et al., 2006). Other emerging technologies such as graphene FETs (Iannazzo et al., 2015, Khan et al., 2014), silicon NanoWire FETs (Turkyilmaz et al., 2013, Yazeer et al., 2016), Spintronics (Bishnoi et al., 2017), and Memristor-based systems (Zangeneh and Joshi, 2014a) have also shown promising results in pushing the boundaries of the Moore's law.

At the circuit level, (Zhai et al., 2004) and (Calhoun and Chandrakasan, 2004) have shown that there exists a minimum energy supply voltage, which lies in the subthreshold region (Wang and Chandrakasan, 2004). Although subthreshold design enables ultra-low energy dissipation, the drawback of subthreshold design is that energy reduction comes at a price of significant performance degradation, which lowers the energy efficiency. A better approach is to operate the circuits near the threshold voltage. This operating point provides the best balance between energy dissipation and performance (Dreslinski et al., 2010). However, we would like to note that this balanced operating point is highly dependent on the application and environmental conditions. As a result, the solution to this energy efficiency problem should 1) have the ability to operate at-or-near the maximum energy efficiency point, 2) have the ability to adapt to changing environmental conditions while satisfying condition 1, and 3) have the mechanism to change the current operating figures-of-merit if the environment (i.e. battery state) change. To tackle these requirements, we propose a multilayer approach to energy-efficient computing. In particular, we laid the grounds for the development of circuit-, architecture-, and algorithm-level mechanisms as given below:

- **Circuit-level mechanism:** While designing a circuit, we use a lot of optimization parameters, such as technology node, device dimensions, architecture, routing topology, etc. These parameters can be seen as pre-fabrication knobs for "static optimization". However, it is difficult to predict the exact operating conditions of a circuit. Hence it is necessary to provide a tuning capability in the circuit. In this work we developed a feedback equalization mechanism for computation blocks. In particular we have shown that feedback equalization with Schmitt Trigger and feedback equalization in pass-transistor logic are viable solutions for energy-efficient computation. We expanded this work to enable tunability (details in Chapters 2 and 3), which enables a more robust operation in changing environmental conditions.
- **Algorithm-level mechanisms:** Generally algorithmic approaches are application specific, however, we showed that is it possible to develop an adaptive system that would utilize machine learning in order to achieve adaptability. For example a mobile processor, that is manufactured without prior knowledge of

operating conditions might utilize machine learning algorithms to adapt to the changing environment. In this case the machine learning approach creates a "tuning knob" for trading off energy, performance, as well as error rate, where applicable. As shown in Chapter 4, machine learning algorithms can be utilized to adapt to changing operating conditions, such as change in workload, change in performance requirements, energy budget, etc. In our work we show that using machine learning approaches, a more energy-efficient operating point for supply voltage is achievable. We also show the implementation of a reconfigurable machine learning approach to enable adaptability for changing input data-sets.

- **Architecture-level mechanisms:** Ideally, one would like to have sight of the entire computing stack, and without the discussion of the architecture-level mechanisms the picture of energy-efficient computing is incomplete. In Chapter 5 we propose the use of distributed architecture to achieve lower energy dissipation using decision forest. In general we show that machine learning systems that benefit from graceful degradation of accuracy are logical choice to operate in ever-changing energy-budget conditions in mobile systems.

1.2 General Principles for Energy-Efficient System Design

To provide a little more general background on energy efficiency, let us share some insights that we have gathered with regard to the energy-efficient and low-power hardware design. In this subsection we would like to summarize some high level ideas that one must always consider, as well as some general "tips" that might be useful as stated in (Sarpeshkar, 2010, Rabaey, 2009, Lee, 2004, Chandrakasan, 1996). Note, that there is no "silver bullet" in designing energy-efficient circuits, architectures, and algorithms. However, the following key ideas might be a good starting point towards

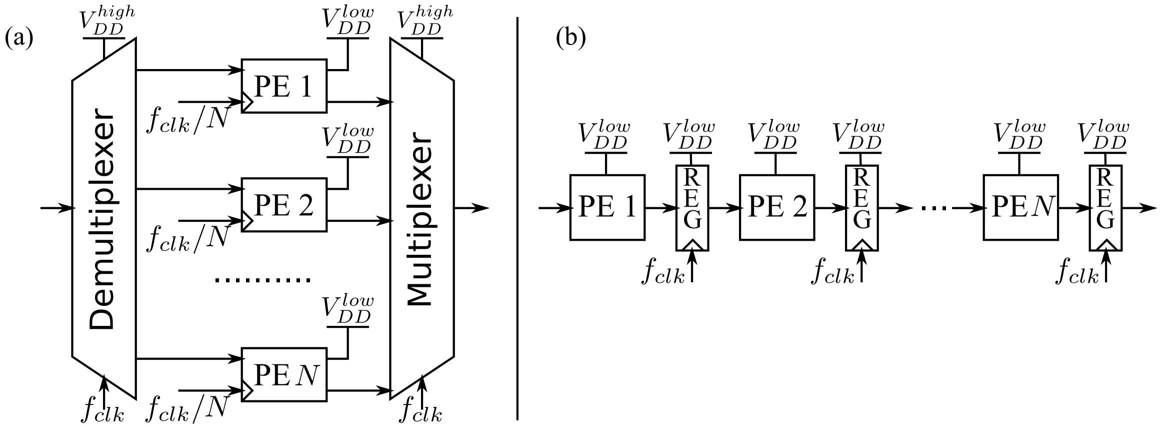


Figure 1·1: Conceptual diagrams of low-power computing architectures. (a) Parallel architecture with N low voltage slow processing elements (PEs); (b) Pipelined version of the computing unit with N slow elements operating at low voltage.

improving the energy related figures-of-merit (FoMs).

Two of the most popular ideas for energy-efficient circuit designs are **parallelism** and **pipelining**. Circuit level parallelism uses a demultiplexer to redirect a high-bandwidth serial input to many low-throughput identical slow-and-parallel computing units. The outputs are then sequentially sampled by a fast multiplexer to recreate the high-throughput serial output. Figure 1·1(a) shows an illustration of this concept as proposed in (Chandrakasan and Brodersen, 1995). N processing elements (PEs) are operating at lower voltage and frequency (V_{DD}^{low} and f_{clk}/N respectively). This architecture has a multiplexer and a demultiplexer that operate at high voltage V_{DD}^{high} in order to support higher frequency f_{clk}. Figure 1·1(b) shows the pipelined version of the computing system. This architecture does not need to serialize the inputs and deserialize the outputs, and thus has no need for fast, high-voltage components. The dynamic power dissipation for parallel and pipelined architectures can be written as

$$P_{nominal} = \alpha f_{clk} C_{big} {V_{DD}^{high}}^2 \tag{1.1}$$

$$P_{parallel} = \alpha f_{clk} \left(C_{PE} {V_{DD}^{low}}^2 + C_{MUX} {V_{DD}^{high}}^2 \right) \tag{1.2}$$

$$P_{pipeline} = N \cdot \alpha f_{clk} {V_{DD}^{low}}^2 \left(C_{PE} + C_{REG} \right), \tag{1.3}$$

where C_{big}, C_{PE}, C_{MUX}, and C_{REG} are the parasitic capacitances for a nominal (non-parallel, non-pipelined), small-and-slow PE, multiplexer/demultiplexer, and register designs respectively. Note that the pipelined version of the architecture introduces factor N in the dynamic power consumption, but does not include the high-voltage C_{MUX} as does the parallel architecture. The design choice between parallel and pipelined versions depends on how many slow PE elements are available and what the power dissipation of individual components in the system is.

The **order** in which the computations are done in an architecture also affects overall energy efficiency of a system. In general terms, computations that reduce the information bandwidth of the data should occur as early as possible. For example, spatial filtering or convolution that allows image information to be encoded in a compressed representation should occur early in the computational pipeline. One of the often overlooked principles of energy-efficient computing is the use of **intelligent-**

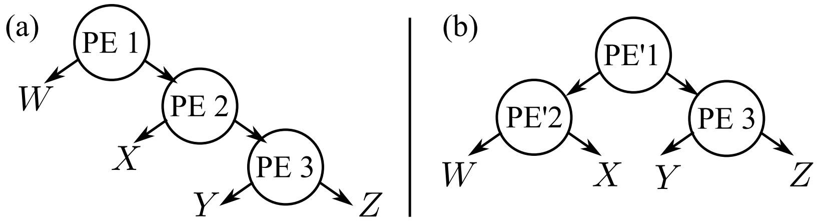


Figure 1·2: Order of execution is defined by the use of an applicable intelligent-and-efficient algorithm, especially in conditional execution: (a) Processing elements are arranged sequentially, and the results are identified conditionally after each PE; (b) Same results with modified conditional PEs arranged in a tree.

and-efficient algorithms with inherently low operation count. In general, this idea dictates the trade-offs that govern the choice of level of parallelism, pipelining, order of execution, and architecture topology. As a general example, algorithms with operations organized in a conditional tree may lead to only a logarithmic number of computing operations, which might be considered intelligent-and-efficient, and in a nutshell represents choice between pipelined vs. parallel execution. Figure 1·2 shows conceptual diagram of different ordering of the processing elements. In this example, if we assume conditional execution, we might choose either (a) sequential or (b) tree-like architecture. The average energy cost for executing architectures (a) and (b) could roughly be estimated as $\mathbb{E}_{(a)} = \mathbb{E}_{PE} \cdot (P_W \cdot 1 + P_X \cdot 2 + P_Y \cdot 3 + P_Z \cdot 3)$ and $\mathbb{E}_{(b)} = \mathbb{E}_{PE} \cdot (P_W \cdot 2 + P_X \cdot 2 + P_Y \cdot 2 + P_Z \cdot 2)$ respectively, where $\mathbb{E}_{PE}$ is the average energy dissipation per conditional PE, and $P_{\{W,X,Y,Z\}}$ is the probability of different conditionals.

To illustrate the choice between different architectures for conditional execution shown on figures 1·2 (a) and (b), consider an example when the cost of all PEs is 1 energy unit (EU), and conditions W, X, Y, and Z are equiprobable ($P_{\{W,X,Y,Z\}} = 0.25$). In that case the average energy cost for (a) would be $\mathbb{E}_{(a)} = 2.25$ EUs. However, the use of tree-like architecture (b) would yield $\mathbb{E}_{(b)} = 2.0$ EUs. As a counter example, let us assume that the conditions happen with probabilities $P_W = 0.5, P_X = 0.25, P_{\{Y,Z\}} = 0.125$. In that case, $\mathbb{E}_{(a)} = 1.75$ EU, while $\mathbb{E}_{(b)} = 2.0$ EU. This example, although over-simplified, shows how the choice of architectures and algorithms can improve the overall energy efficiency.

In addition to the general ideas mentioned above, we can identify several principles or "tips" for low-power design. Although there is some overlap and redundancy in these principles, we are hoping these principles will capture the essence of the underlying idea of energy efficiency. The principles, in no particular order, are

1. **Exploit preprocessing in an optimal way.** Preprocessing a high-throughput input often improves all FoMs in a low-power or energy-efficient system. The preprocessing was shown to be very efficient in analog domain (Sarpeshkar et al., 2005, Avestruz et al., 2008). In digital domain convolutional layers in CNNs effectively preprocess the input to extract more information (Simard et al., 2003), thus improving the accuracy. Chapter 4 describes an adaptive classifier that has a "chooser" function that can be seen as preprocessing the input by identifying its "hardness" and selectively enabling different computational paths.

2. **Find a technology that "encodes" your computation.** Some functions, such as simple integration or even RBF (Tsividis et al., 1990) is possible to implement using a single bypolar junction transistor (BJT), which could significantly improve energy-efficiency. Another example is the use of emerging technologies such as memristive crossbar networks to mitigate the processor-memory bottleneck (Velasquez and Jha, 2014). "Technology encodings" were shown to be efficient in both analog and digital domains (Rabaey, 2009, Sarpeshkar, 2010)

3. **Use parallel and pipelined architectures.** As described earlier, parallel and pipelined systems implement complex computational systems through divide-and-conquer approach. The speed, complexity, and accuracy of each PE can be significantly less than that needed in the overall computation. Chapter 5 describes an energy-efficient way of implementing random forest and field of groves algorithms using parallelization and pipelining.

4. **Balance computation and communication costs.** A frequent trade-off in many mobile and embedded systems is the balance between communication and computation energy costs. If one computes too little, there is too much information to transmit, increasing the communication cost. On the other hand,

if computing too much, one might be passing on the opportunity to save some energy by computing in the “cloud”. The optimal solution for this problem is still in its infancy, and requires extensive analyses.

5. **Use hybrid systems.** In the mixed-signal domain, the concept of hybrid (Sarpeshkar, 1998) and collective analog (Mandal et al., 2009) is widely used to combine the best of the analog and digital worlds to design ultra low-power systems. The main idea is to separate the computation into multiple simpler blocks that communicate with each other, thus using parallelization and rebalancing the computation vs. communication costs. This principle overlaps with the principle of parallelism but does not assume that the processing elements in the system are identical. Expanding the definition of the hybrid systems, we would also include the notion of specialized accelerators under this bullet. A complex system can be separated into two parts:

    (a) General purpose processor: high ignorance of what is important; limited adaptation; no learning; highly energy-efficient for general tasks;

    (b) Specialized accelerator: high knowledge of what is important; supporting adaptation and live reprogrammability; high use of analog and mixed-signal components in addition to digital; highly energy-efficient for specialized tasks.

    Chapters 4 and 5 describe accelerators that are specialized for machine learning classification, and achieve high energy-efficiency compared to general purpose processors.

6. **Reduce the amount of information to be processed.** Energy and information are naturally linked, and the overarching principle of information reduction could encompass all principles in low-power or energy-efficient design

listed here. This principle is rather abstract, and its concrete implementation requires the knowledge of the algorithms, signal processing, architectures, circuit topologies, as well as prior knowledge of the types of data that would be processed, and application for the design.Many examples could be placed under this principle, ranging from purely analog circuits such as automatic gain control (AGC) (Baker and Sarpeshkar, 2006) and ending with efficient digital algorithms like clock gating (Rabaey, 2009). Adaptive classifier described in chapter 4 can be loosely seen as utilizing this principle by adaptively selecting how much information should be processed by linear, polynomial, or infinite order RBF blocks.

7. **Use feedback and feedforward architectures.** Feedback is an important concept in improving both the efficiency and robustness of an energy-efficient system. The energy-efficiency of the feedback stems from three primary mechanisms (Sarpeshkar, 2010): 1) a lowpass or integrator transfer function in a feedback path provides differentiation, delta encoding, highpass filtering, or "adaptation" such that unwanted information (usually DC) is removed, while desirable information (usually AC) is propagated; 2) non-linear feedback implements automatic gain control or compression, which allows relatively constant average information (as current or SNR) to be mapped from input to output even if environmental conditions change; 3) feedback attenuation of the device noise statistics. Chapters 2 and 3 describe our view of how the feedback loop is incorporated in the computational channels.

8. **Separate speed and precision/accuracy.** Recent advances in the domain of machine learning (Wang et al., 2017) have shown drastic improvement in all FoMs of energy-efficient approximate computing (see chapter 1.3). However, energy-efficiency is still a major bottleneck in the mobile and embedded systems.

Thus, it is important to remember that it costs power to be simultaneously fast and accurate (whether we are performing analog/digital processing, sensing, or actuating). It is advantageous to design architectures (as far as possible) where speed and precision are not simultaneously needed. For example, in case of the machine learning, it is possible to design systems where only the latch is precise, while the rest of the architecture would utilize low-power approximate computing units. This principle can be seen in nature: neurons "discretize" continuous analog vector inputs into a set of digital outputs. The process of sampling is performed using current-integrating comparator (Sarpeshkar et al., 1992), and is equivalent to an approximate integrator combined with accurate comparator. A typical neuron in the human brain operates on a ≈6000-input vector with ≈0.5nW of power, such that ≈22 billion neurons in the brain consume ≈15W of power (Aiello, 1997). Nature inspired a lot of theoretical and practical work (see chapter 1.3).

9. **Use subthreshold operation if possible.** Although not the best tip for high-performance computing modes, operating in subthreshold regime is advantageous from the point of view of low-power operation. According to (Sarpeshkar, 2010, Mead, 1989, Enz and Vittoz, 2006) the main power dissipation reducing traits of operating in the subthreshold regime are 1) maximized speed per watt; 2) lower saturation voltage; 3) no velocity saturation; 4) low resistive and inductive drops dues to parasitics; 5) possibility of implementing machine learning computing blocks in this regime in a highly energy-efficient fashion – blocks such as exponential/radial basis function (RBF), tanh, cosh, sinh, etc. can profitably use "leakage" currents as "analog computing units" (ACU).

10. **Operate slowly and/or adiabatically.** Although not necessarily energy efficient, operating slowly might be power efficient. One of the reasons why natural

> neurons are extremely power efficient is because they operate relatively slowly, and do not dissipate power with the surrounding cells – so called adiabatic process. Operating far below nominal frequency f_{nom} was shown to be power efficient (Lee, 2004) in digital circuits. It is suggested that following this principle one must try and operate at a frequency the is at least five times below f_{nom} and because this principles assumes low computational performance, it is best suited to work well in subthreshold regimes as shown in (Sarpeshkar, 2010).

It is worth pointing out that the "No Free Lunch" theorem (Wolpert and Macready, 1997) is universal, and in our case robustness, flexibility, accuracy, and performance always trade-off with energy efficiency: the extra degrees of freedom necessary to maintain robustness or flexibility invariably hurt efficiency; increase in accuracy or performance almost always requires an increase in complexity, thus making the system less efficient. With that said, we employ the reader to accept the listed principles with a grain of salt – keeping the trade-off game in mind.

1.3 Related Work

1.3.1 Circuits and Architectures that Trade-Off Performance, Energy, and Error Rate

The tunability and adaptability of a system is best described in the context of its operating condition. For example, in high performance computing it is rarely expected for performance to be tuned down, while in medical applications, implant systems are almost always expected to operate in low energy dissipation modes. Mobile systems, however, need and can take advantage of tunability and adaptability where we can trade off for low power dissipation or high performance. A mobile device that is running out of battery can reduce its energy requirements by turning off some of its components to save power. A more intelligent and efficient algorithm for a mobile device would be to collect run-time statistics in order to be able to predict which parts of the system it is more optimal to switch off.

Mobile computing makes an ideal candidate application for dynamically tunable and adaptable systems which could determine the optimal trade off between power dissipation and computational performance at run-time (Gautschi et al., 2017). One can switch the voltage anywhere between nominal voltage to achieve high computational performance and near-threshold voltage (NTC) to achieve the most energy savings (Dreslinski et al., 2010). Operating in the near-threshold regime provides the best energy-performance trade off. One of the main challenges to be addressed in the NTC region is the PVT variability as well as finding the right combination of circuit and architecture solutions that meet the energy-performance specifications. Seok et al. (Seok et al., 2011) have shown several established techniques in mitigating variability from different perspectives: logic, memory, and clock distribution. They have shown that techniques such as body biasing and soft edge clocking are good circuit techniques to minimize the effects of the variability. Karpuzcu et al. (Karpuzcu et al., 2013) proposed an algorithmic approach to NTC, while it was also shown that the energy could be traded-off for reliability and vice versa (Takhirov et al., 2013, Zangeneh and Joshi, 2014b). Although their work does not focus on variability, the algorithm is tunable to trade-off energy for reliability, thus making the trade-off problem three dimensional: while designing a system in near-threshold regime, it is possible to sacrifice one of the three FoMs (energy, performance, or error rate) while improving the other two. Dynamic voltage and frequency tuning (DVFS) is an important part of the run-time tunability, and its precision suffers from temperature variations. Kiamehr et al. have shown that ambient temperature has a huge impact on DVFS in NTC (Kiamehr et al., 2017). The have also proposed a low-cost, ambient temperature aware voltage scaling technique to reduce the unnecessary energy overhead caused by temperature variation. Some recent works provide analysis of process variation effect on the adiabatic logic in NTC (Lu and Kazmierski, 2016), as well as impact of FinFET on NTC scalability (Pinckney et al., 2017).

As the performance of a digital system is limited by error rates, some works are focused on mitigating or detecting and correcting errors due to timing faults in the system. Timing errors happen when some of the timing constraints, such as setup or hold time, are violated (Valadimas et al., 2013). In tunable systems and NTC that happens when the data path delay increases due to aggressive voltage scaling. A number of existing techniques were shown to manage timing errors at the **circuit level**. For example, Razor is a mechanism to tolerate PVT variation induced errors and soft errors by flagging spurious transitions followed by recovery at the architecture level (Ernst et al., 2004, Das et al., 2005, Das et al., 2009). he Razor based approach requires an extra memory element per flip-flop and suffers from high silicon area cost and power consumption. Moreover, in order to treat metastability phenomena in the main flip-flop, a metastability detector is required to guarantee high levels of reliability. Other approaches introduce the use of spatial or temporal computation redundancy at the circuit level (Naeimi and DeHon, 2008, Bowman et al., 2009), timing and delay redistribution (Kahng et al., 2010, Mohapatra et al., 2011), and even circuit level error correcting codes (Lala, 2001, Mathew et al., 2008, Poolakkaparambil et al., 2011). To monitor and mitigate degradation due to HCI, BTI, and dielectric breakdown (TDDB), numerous techniques, such as silicon odometer, in-situ sensors, etc. were proposed (Agarwal et al., 2007, Karl et al., 2008, Keane et al., 2010, Qi and Stan, 2008).

At the architectural level N-modular redundancy with voting mechanism has been widely used to tolerate error (Avirneni et al., 2009, Chen et al., 2011, Sartori et al., 2009). Other approaches used on the architecture level of hierarchy is the use of stochastic processing platform with multiple functional units (Leem et al., 2010, Narayanan et al., 2010) and instruction-level error correction used in NTC to achieve timing error resilience (Wang et al., 2017), while (Pan and Teodorescu, 2014) presented energy-efficient STT-RAM implementation. These approaches avoid the timing errors by utilizing error-resilient algorithms and their architectural implementations in many-core and many-processor systems. Given the multitude of the approaches to error mitigation and power-performance trade-off techniques, we can combine some of those techniques while using the error rate as a figure-of-merit rather than a constraint: consider an example, where the architecture has some level of error correction. In that case the circuit level constraints could be relaxed in order to achieve better power-specifications (Akturk et al., 2015). At the same time, some applications are error-tolerant by nature, which could allow for error rate to be another dimension in the "trade-off game" (Tagliavini et al., 2016). Our feedback equalization technique (described in chapters 2 and 3) uses variable threshold to mitigate timing errors while maintaining the goal power-performance specifications. In addition to that it enables the possibility of dynamic threshold voltage readjustment by biasing the feedback into the base of the CMOS transistors, which would widen the power-performance tunability range in digital circuits.

1.3.2 Adaptive Systems and Machine Learning Hardware

Apart from the circuit level tunable systems described in this section, it is also important to look into the research performed in the area of adaptive systems. In particular those related to energy efficient adaptive machine learning algorithms. This is motivated by the "Big Data" problem as combined with the problem of "Internet of Things" (IoT), where massive amount of data is being generated, and its processing is required in an energy-constrained environment. Machine Learning (ML) algorithms are one of the most useful tools in the age of the "Big Data" and IoT and are getting more pervasive in the mobile space (ComScore Report 2016, 2016, Abadi et al., 2015). However, because the operating conditions of the mobile devices constantly change during the run-time, the figures-of-merit also change. This means that it is very important to improve the run-time parameters adaptively, rather than statically.

There has been prior work in the implementation of the machine learning algorithms in all levels of abstraction of the hardware design. Some works have focused on application-specific solutions (Kaul et al., 2012, Lin et al., 2005). In general, approximate computing (Grigorian et al., 2015, Venkataramani et al., 2013, Narayanan et al., 2010, Sidiroglou-Douskos et al., 2011, Esmaeilzadeh et al., 2012) leverages the fact that computational effort can be scaled at different levels of abstraction by exploiting the resilience of applications to inexact solutions (Düben et al., 2015, Chippa et al., 2010, Chippa et al., 2011). However, generalizing such approaches to a broader spectrum of applications is a non-trivial task, that current research focuses on.

Recent works have shown that although GP-GPUs are still the most commonly used platform in accelerating machine learning algorithms (Cireşan et al., 2011, Yazdanbakhsh et al., 2015), other platforms are also a viable option. Specifically, neuromorphic approaches are getting more traction, and more works focus on accelerating neural networks. Emulation of the biological linear-leak integrate-and-fire (LLIF) spiking neurons was shown to be possible in basic digital circuits (Nere et al., 2013). It is possible to achieve similar neuromorphic functionality using analog computing with resistive crossbars (Liu et al., 2015), which adopts a rate-coding scheme where pre- and post-neuron signals are represented in digitized pulses. Unfortunately, due to their large computational requirements, purely digital hardware implementations of larger neuromorphic architectures prove to be challenging (Du et al., 2015).

Multiple works tackle the large-scale neuromorphic architectures from different angles. Rahman et al. (Rahman et al., 2016) proposed a flexible and highly efficient

3D neuron array architecture on an FPGA and is a natural fit for convolutional layers of a neural network. One of the main challenges with such an approach is the size of the network, and memory requirements for their acceleration. Techniques like factorization and pruning were proposed to compress the neural network models as well as maintain the form of the models for the execution of neuromorphic architectures (Chung and Shin, 2016). Another way to reduce the size of a model is to remove the near-zero weights, which was shown in (Kim et al., 2016) to improve the energy efficiency of the design without loss in accuracy. In their work, the weights in a neural network were also scaled, and the activation function was embedded in the accumulator, which makes this approach suitable for ultra low power accelerator design. Cambricorn-X (Zhang et al., 2016) exploits sparsity and irregularity in neural networks to achieve energy-efficient neural network acceleration. Their work utilizes an indexing module that efficiently selects and transfers needed neurons to appropriate processing elements. A more general purpose accelerators such as DaDianNao (Chen et al., 2014) uses multiple multi-stage processing elements to achieve energy-efficient acceleration of machine-learning algorithms in general. This approach inspired a whole family of DianNao accelerators (Chen et al., 2016a).

Another field that is of high interest in hardware design, is the application of the emerging technologies in neuromorphic computing and in machine learning in general (Panda et al., 2016b). Memristive devices have been proven to be useful in mimicking spiking neurons, and were used in memristive crossbar structures to speed up the executions of artificial neural networks (Xia et al., 2016, Liu et al., 2015). A more general work by Hu et al. (Hu et al., 2016) have shown that memristive devices are useful in accelerating matrix-vector multiplication in general. In their work they proposed a memristive crossbar array programming for arbitrary matrix values multiplication. This would allow for any machine learning algorithm to be accelerated. Other works focus more on emerging applications such as Resistive RAM (ReRAM or RRAM) to either implement supervised (Chi et al., 2016) or unsupervised (Bojnordi and Ipek, 2016) machine learning algorithms to achieve better energy-performance metrics. These works use metal-oxide ReRAM to store the synaptic weights and improve the efficiency of the matrix dot product, which is in the core of neural computing. ReRAM uses memristors as a memory unit, and recent works show that it could be used as an in-memory computing device (Tang et al., 2015, Song et al., 2017, Hamdioui et al., 2015).

Recent research has shown that adaptive machine learning acceleration is a positive step towards energy efficient computing. The classification system proposed by Venkataramani et al. (Venkataramani et al., 2015b) centers on using increasingly complex classifiers in a pipeline fashion on examples close to the decision boundary. Their approach assumes that complex classifiers will always correctly classify examples, potentially using multiple complex classification functions on inherently noisy examples. That means that if the input "hardness" is very high, all classifiers have to try to identify the labels, before sending the input to the complex classifier. Panda et al. (Panda et al., 2015, Panda et al., 2017) introduce an object detection mechanism using energy-efficient neural computing. The hierarchical framework of classifiers were set up in an increasing level of complexity, making the dynamic trade-off between classification performance and energy efficiency hard. Similarly, the classification system proposed by Park et al. (Park et al., 2015a) uses dynamic threshold adjustment in deep neural networks (DNN) to achieve low power operation. In this work, a "little" DNN is used, and if it fails, a "big" DNN is used, etc. Because of the little-big topology it is not possible to automatically enable the appropriate DNN without running all other DNNs (Sampson, 2015).

The approach proposed in the current work could be viewed as the energy-efficient learning problem, which in turn is a variation of the problem of learning under test-time budgets (Trapeznikov and Saligrama, 2013, Wang et al., 2014b, Wang et al., 2014a). Here, learning under test-time budgets is a family of approaches that focus on minimizing the costs at test time. In general, these approaches focus on selecting sensors as opposed to energy expenditure and ignore the energy usage associated with selection system. As the main obstacle in the test-time budget problem is the sequential revelation of information, the energy expenditure of the selection system is generally not accounted for. Additionally, many approaches to test-time budgeted learning require design of both the resource allocation as well as the classification functions, preventing the use of optimized modular systems (Xu et al., 2013, Kusner et al., 2014, Nan et al., 2015).

Our work could be contrasted by other works that focus on adaptive classification. In (Judd et al., 2016) authors propose an adaptive neural network that allows on-the-fly trade-off among accuracy, performance, and energy. Similarly Panda et al. (Panda et al., 2016a) focus on dynamic adjustment of computational efforts in neural networks. Unlike these works, our work is not restricted to using one type of machine learning algorithm such as neural networks. We use a notion of a "core" classifier, which we treat as a black box, and use another classifier ("chooser") to achieve the adaptive functionality.

Our work on adaptive systems is closely related to works on learning under test-time budgets (Wang et al., 2014b, Trapeznikov and Saligrama, 2013). Although our work falls in the same category as the works by Venkataramani et al., Panda et al., and Park et al., and we cast the energy efficiency learning problem in a similar fashion, our approach is fundamentally different as it does not ignore the energy usage associated with selecting a classification model. We also don't focus on the sequential learning problem as new information is revealed, but consider a non-sequential selection process and account for the energy consumed in selecting the most energy-efficient classifier.

The contribution of the current work also includes the use of distributed and parallel architectures. Algorithmically, our proposed approach is based on random forest (RF) classifiers (Breiman, 2001). Traditionally, energy efficiency has not been considered when designing random forests; however, recent work has studied learning of the RFs as a subject to test-time constraints (Nan et al., 2015). This approach centers on reducing feature/sensor acquisition cost, however it does not address the system energy constraints. Similar approaches to learning decision rules to minimize error subject to a budget constraint during prediction-time have been proposed (Kusner et al., 2014, Xu et al., 2012, Wang et al., 2015). Although closely related, these approaches also ignore the energy usage and disregard computational cost, making them of limited use in an energy constrained settings.

The state of the art machine learning techniques for image processing, video processing, object recognition, etc. are CNN and DNN, and custom hardware accelerator design have been proposed for the same (Chakradhar et al., 2010, Park et al., 2015b). In addition to that, a variety of techniques including stochastic computing (Kim et al., 2016), dataflow architecture (Chen et al., 2016b, Nowatzki et al., 2015), data reuse (Rahman et al., 2016), custom sparse matrix-vector multiplication (Han et al., 2016) and run-time adaptivity (Venkataramani et al., 2015a, Takhirov et al., 2016) have been adopted to achieve energy-efficient machine learning based operation.

1.4 Contributions

In the light of the current developments in the field of energy-efficient computing described in subsection 1.1, and the principles of energy-efficient system design rules laid out by us in subsection 1.2, we developed several mechanisms to achieve better energy-related figures-of-merit in digital design.

The following work was completed leading up to this dissertation:

Error Avoidance using Feedback Equalization and Schmitt Trigger

In this work we have implemented the feedback equalizer to improve the performance of digital CMOS logic. The improvement is achieved by eliminating the inter-symbol interference (ISI) between successive outputs of the digital logic. Our simulations have shown that at nominal frequency f_{nom} and 0% error rate the design with the feedback was able to operate at the voltages close to the threshold, while the baseline design (without feedback) saw higher degradation in error rates at voltages other than super-threshold. The results of the work were presented at the 13^{th} International Symposium on Quality Electronic Design 2012. The work has shown that a system with feedback equalization achieves lower energy dissipation while maintaining the error rate and computational performance. For example, in complex circuit blocks like FIR, the use of feedback equalization reduced the energy dissipation by up to 40% when operating at threshold voltage $V_T = 0.55V$ and nominal frequency of $f_{nom} = 500MHz$. This work was extended to evaluate the use of equalization for entire voltage range above threshold voltage V_T. The use of equalized FF at above threshold voltage enables an increase in operating frequency (up to 30%) and decrease in the overall energy-delay product (up to 10%) making it possible to operate the digital block at its near-optimal operating point from the EDP perspective over the entire above-threshold voltage range.

Decision Feedback Equalization in Pass-Transistor Logic

We proposed a novel differential “equalized pass-transistor-logic” (E-PTL) circuit that enables aggressive voltage scaling to lower energy consumption. Conversely the same approach could be used to enable aggressive overclocking to improve performance at the cost of error rate and/or energy dissipation. In our approach a differential Flip-Flop based Strong-Arm register was modified to accommodate for equalization. For a 16-bit Kogge Stone Adder (KSA), using convex optimization toolbox (CVX Research, Inc., 2012, Grant and Boyd, 2008) the minimum energy design point for a given performance and error rate goal was found and implemented. It was shown that the proposed design reduces energy consumption by up to 30% while operating close to the threshold voltage (a.k.a. near-threshold). The results of the work were published in the IEEE International Symposium on Low Power Electronics and Design in 2013.

Machine Learning on a Budget using Adaptive Effort Classifiers

In this work we have investigated adaptive machine learning classification systems and their implementation in digital hardware. The results of this work have shown that adaptive systems can be tuned along the energy efficiency vs. accuracy curve at run-time. Here, the data throughput (number of classifications per unit time) of adaptive system could be readjusted to be anywhere between 0.5x to 3x as compared to non-adaptive approaches. The same way the energy dissipation is adjustable (post-fabrication) from 0.01x to 10x as compared to the non-adaptive classifiers. The readjustment in adaptive classifier is as simple as changing a bias variable to enable more frequent use of one classifier vs. another. This allows for machine learning techniques to be used to adapt to changing environment while maintaining optimal power-performance metrics. This work was released for public in International Symposium on Low Power Electronics and Design 2016, and was further extended by allowing the host CPU report the current environmental conditions which is used as extra information in deciding the best operating mode. This enabled the adaptive classifier to be aware of the current energy reserves, which further optimized the trade-off between energy-dissipation and accuracy rates. This work is being prepared to be submitted to the Transactions on VLSI systems.

Distributed Computing using the Field of Groves approach

We also introduced a field of groves (FoG) implementation of random forests (RF) that achieves an accuracy comparable to Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) under tight energy budgets. The FoG architecture takes advantage of a property of random forests that a small portion of the weak classifiers (decision trees) might be sufficient to achieve high statistical performance. By dividing the random forest into smaller forests (Groves), and conditionally executing the rest of the forest, FoG is able to achieve much higher energy efficiency levels for comparable error rates. We also take advantage of the distributed nature of the FoG to achieve high level of parallelism. Our evaluation shows that at maximum achievable accuracies[1] FoG consumes ≈1.48×, ≈24×, ≈2.5×, and ≈34.7× lower energy per classification compared to conventional RF, SVM_{RBF}, Multi-Layer Perceptron (MLP), and CNN, respectively. FoG is 6.5× less energy efficient than SVM_{LR}, but achieves 18% higher accuracy on average across all considered datasets.

The remainder of the current dissertation is outlined as follows: Chapter 2 provides a detailed analysis of the feedback equalization in CMOS circuits. It also provides a comparison of sequential logic with feedback equalization with Schmitt trigger (FEST) circuit and sequential logic without FEST circuit. Chapter 3 introduces the feedback mechanisms in pass-transistor logic (PTL). It includes the modeling of the energy, performance, and error rate of an equalized system. We also present a comparison of

---

[1] Maximum achievable accuracies are different for all classifiers. Although highly dependent on the datasets, we can assume (with great caution) that RF, SVM_{RBF}, MLP, and CNN have comparable maximum accuracies; the accuracy of SVM_{LR} is generally much lower than the other presented algorithms.

static CMOS logic with equalized and non-equalized PTL circuits. Chapter 4 and 5 present novel machine learning architectures and algorithms: an adaptive classification algorithm and evaluation of its implementation in digital hardware and Field of Groves implementation of RF algorithms and the detailed evaluation of its hardware implementation. Chapter 6.2 concludes the current dissertation and discusses the future directions for the completed research.

Chapter 2

Error Mitigation in Digital Logic Using Feedback Equalization

2.1 Introduction

Inter-symbol Interference (ISI) is a common problem in the long-distance[1] wired digital communication systems, which emerges due to finite rise and fall times of the logical *HIGH* and *LOW*. Parasitic resistances and capacitances in a long wire reshape the square-wave signals of the digital logic to have long "tails". This causes increased delays in the circuit, and conversely limits the minimum operating voltage, as delay in digital CMOS is a function of the supply voltage. A commonly used approach to the timing problems caused by ISI in communication systems is to use equalization.

Getting an inspiration from the communication system, we proposed a feedback-based technique for mitigating timing errors in computational blocks. Many of the timing errors observed after voltage scaling are due to residual effects from the previous computation, which in effect is the same as ISI problem in the long-wire communication. We developed an equalization system that could be used in the critical paths of the computational blocks. The system uses feedback equalization to eliminate the "ISI" as well as a Schmitt Trigger to clean the signal of any artifacts and glitches. Feedback Equalization with Schmitt Trigger (FEST) adjusts the switching thresholds in the critical path based on the prior outputs. Specifically, if the input to

---

[1] For example core-to-core bus

a flip-flop is switching, the FEST circuit provides a faster charging (or discharging) path for the input capacitance of the flip-flop to ensure correct sampling of data. Overall, this makes the system more robust to timing errors and enables more power savings through voltage scaling. It should be noted that the FEST circuit can also be used to provide fast charging/discharging paths for the one or more gates in the critical path preceding the flip-flop to further reduce the power dissipation.

In this chapter I will describe the FEST circuit as well as its analysis. In order to analyze the proposed circuit, several designs were considered: as a first step 4-bit Kogge-Stone Adder (KSA) and a 3-tap 4-bit finite impulse response (FIR) filter were designed. After achieving promising results, more complex functions were implemented in a more aggressive technology node: 24-bit multiplier, AES-256, Reed-Solomon decoder, and Fast-Fourier Transform blocks.

2.2 Error Manifestations in Digital Logic

A CMOS circuit can have permanent faults (due to irreversible physical changes such as a gate output stuck at one or zero, short, or electromigration-induced open circuits), transient faults (due to temporary changes in supply voltage and temperature) and/or intermittent faults (due to unstable or marginal hardware) (Constantinescu, 2003). These faults manifest themselves into bit errors and might cause catastrophic system failures. We focused exclusively on timing errors due to supply voltage scaling. These timing errors can be traced to the fact that voltage scaling increases the switching time of the transistors, which in turn leads to timing constraint violations that result in incorrect sampling of the data by the flip-flop/latch. We proposed circuit-level technique that can mitigate these errors, thus permitting further voltage scaling and a reduction in energy consumption. For the purposes of this work, we assumed that the application layer is able to tolerate some errors, so long as the probability is very low – this assumption was made with a more complex future systems in mind. Understanding the interaction between our circuit technique and higher layer error-resiliency is an interesting subject for future study. Given a complex digital logic block, its critical paths are the ones that define the timing violations. The errors are expected to arise when the transitioning outputs of a gate do not have enough time to charge/discharge the parasitic capacitances due to reduced supply voltage (in case of NTC). To verify this simple hypothesis, we simulated a 4-bit Kogge-Stone Adder (KSA) in HSPICE. We designed the adder to run (without errors) at $2GHz$ at a nominal supply voltage ($800mV$ for $22nm$ PTM). The supply voltage was then scaled to $490mV$ and the KSA was tested by providing 1000 randomly generated input patterns and observing the timing errors. These observed error manifestations closely mirror the inter-symbol interference (ISI) errors that arise in communication channels. In this setting, a decision feedback equalization (DFE) mechanism is commonly used to combat ISI at the receiver end. The basic idea is that the receiver subtracts a weighted estimate of the prior bit before making a decision on the current bit. In the next section, we explore a circuit that mimics this technique for use in arithmetic logic.

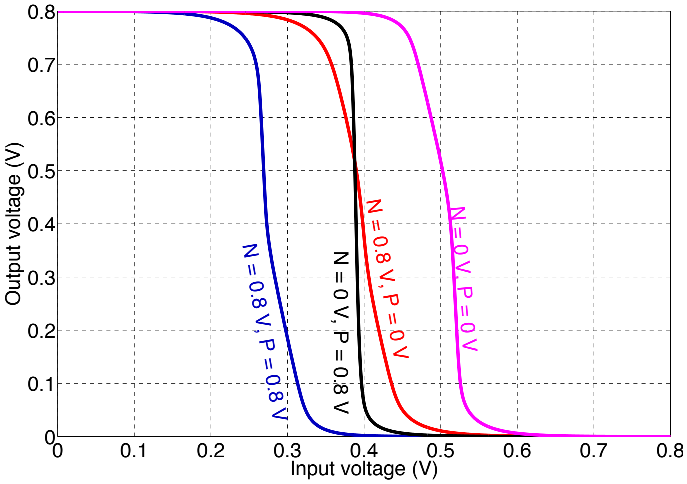


Figure 2·1: DC response of the feedback equalizer circuit.

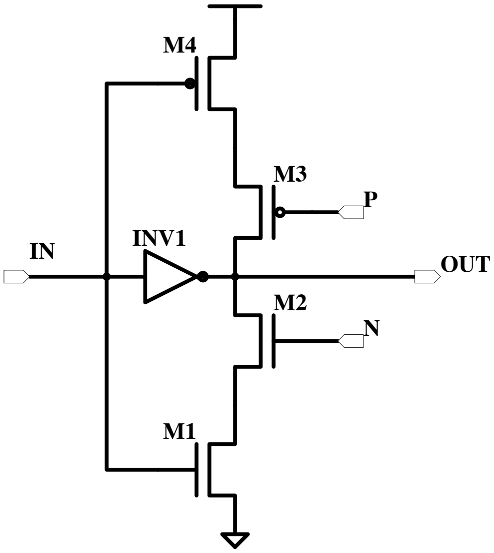


Figure 2·2: Variable threshold inverter circuit (Sridhara et al., 2008).

2.3 Equalization Techniques

In a communications system, DFE is often implemented by multiplying the estimate of the previous bit by an appropriate scaling factor and subtracting this quantity from the channel output. In our considerations, the multiply-accumulate circuits required for DFE far exceed the complexity of the digital arithmetic logic we are trying to protect. From a digital circuit design perspective, DFE can be viewed as a mechanism to vary the switching threshold of a logic gate based on the prior gate output. If the prior output is a zero, then the switching threshold is decreased to facilitate a transition to a one. Conversely, if the prior output is a one, then the switching threshold is increased. Below, we describe a variable threshold inverter that can be used as a basic feedback equalizer. Afterwards, we argue that coupling this circuit with a Schmitt trigger leads to a robust equalizer that is well-suited for over-scaling arithmetic logic to save energy.

2.3.1 Feedback circuit

Sridhara *et al.* employed feedback equalization to overcome the ISI encountered in on-chip communication channels (Sridhara et al., 2008). They used a variable threshold inverter (depicted in Figure 2·2) to implement feedback equalization on a bus. The switching threshold of this inverter depends upon the values of the inputs at nodes P and N as detailed in Table 2.1 with $V_{th}^{-} < V_{th}^{0} < V_{th}^{+}$. The sizes of transistors M1, M2, M3, and M4 are chosen depending on the desired values for V_{th}^{+} and V_{th}^{-}. This circuit can be used as a feedback equalizer in digital logic by simply connecting the P and N nodes to the previous output sampled by the flip-flop (see Figure 2·4). In this configuration, the circuit adjusts the switching threshold and facilitates faster high-to-low and low-to-high transitions. Note that the inverter (INV1) should be fairly weak to ensure that the output does not float (Sridhara et al., 2008). The DC response of the variable threshold inverter is shown in Figure 2·1.

P	N	V_{th}
LOW	$HIGH$	V_{th}^{0}
LOW	LOW	V_{th}^{+}
$HIGH$	$HIGH$	V_{th}^{-}

Table 2.1: The inverter threshold depends on the inputs P and N. Our system varies the threshold using feedback.

One drawback of using this circuit in low power digital logic is that, in the deep sub-micron regime, it becomes susceptible to glitches after voltage scaling. This is mainly due to the difference in switching times of the different nodes in the circuit. At nominal voltage, these glitches can be tolerated and the correct output will be sampled by the succeeding flip-flop. However, these glitches widen with voltage scaling and eventually get incorrectly interpreted as valid signals.

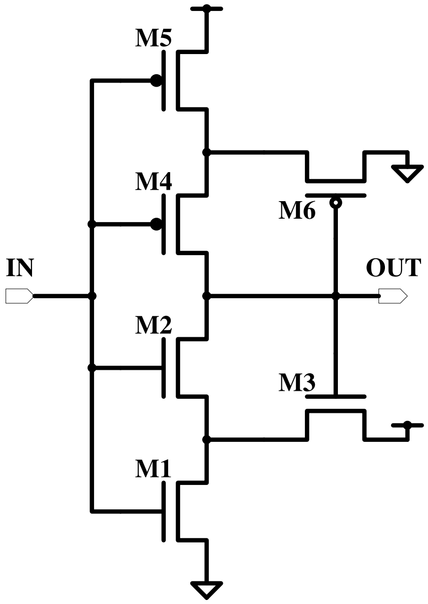


Figure 2·3: Inverting Schmitt trigger circuit (Baker, 2004).

2.3.2 Schmitt trigger

The deficiencies of the feedback circuit can be overcome by using a Schmitt trigger. A Schmitt trigger is a dual-threshold buffer or an inverter with positive feedback (i.e. with loop gain greater than one) that can mitigate the impact of glitches. It is typically used to smooth a ringing pulse, as well as in voltage controlled oscillators. It also can be useful in generating a clean pulse from a noisy input signal. An inverting Schmitt trigger described in (Baker, 2004) is shown in Figure 2·3. The Schmitt trigger transfer function exhibits a hysteresis loop with two switching thresholds – lower switching point V_{SPL} and higher switching point V_{SPH}. (In a conventional inverter, $V_{SPL} = V_{SPH}$.)

The threshold voltages of the Schmitt trigger can be controlled using the following equations

$$\frac{\beta_1}{\beta_3} = \left[\frac{V_{DD} - V_{SPH}}{V_{SPH} - V_{THN}}\right]^2 \tag{2.1}$$

where V_{THN} is the threshold voltage of the NMOS and the device transconductances satisfy $\beta_2 \geq (\beta_1 \text{ or } \beta_3)$ (M2 is a switch), and

$$\frac{\beta_5}{\beta_6} = \left[\frac{V_{SPL}}{V_{DD} - V_{SPL} - V_{THP}}\right]^2 \tag{2.2}$$

where V_{THP} is the threshold voltage of the PMOS and the device transconductances satisfy $\beta_4 \geq (\beta_5 \text{ or } \beta_6)$ (M4 is a switch) (Baker, 2004).

Although the Schmitt trigger is an effective way to eliminate glitches, it exhibits slight additional delay between the time when the input signal changes and the output signal changes. Thus, despite the fact that the Schmitt trigger is useful in eliminating artifacts, it can still introduce errors due to a possible shift of the gate output (which would be the input to the flip-flop) transition closer to the clock edge. This effect is mitigated by carefully sizing the feedback equalizer and Schmitt trigger, such that Schmitt trigger and following flip flop input capacitances would be small, while the feedback equalizers output strength would be large.

2.3.3 Feedback Equalization with Schmitt Trigger (FEST)

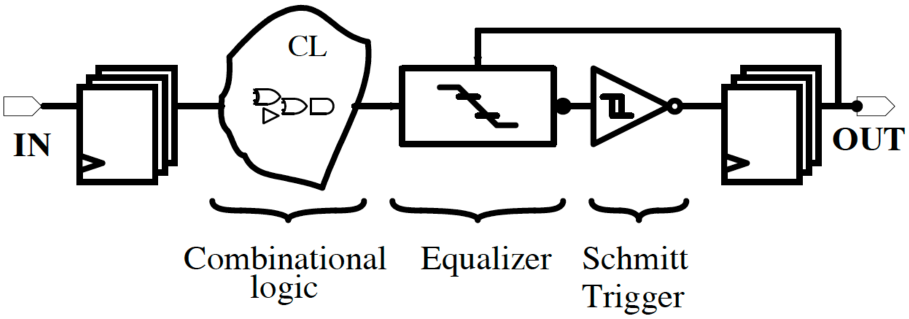


Figure 2·4: Pipelined combinational logic with FEST circuit.

The feedback equalizer described in Section 2.3.1 and the Schmitt trigger in Section 2.3.2 exhibit different output switching behavior for the same input transition. The feedback circuit, due to the altered threshold voltages, starts switching earlier, but the switching time might be larger if sized incorrectly (meaning that more time is needed to charge up the parasitics). On the other hand, the Schmitt trigger has a smaller switching time (when the switching starts, $|V_{GS}|$ is already high, and most of the parasitics are charged/discharged), but it starts switching later. It follows that in the feedback circuit there is a high chance of falling into meta-stability due to a timing violation. In the case of the Schmitt trigger, the chance of falling into meta-stability is relatively low (due to fast transitions), but there is a high chance of being sampled at the wrong cycle. This effect is easily mitigated if certain sizing rules are applied.

For our proposed FEST circuit we connect the output of the feedback circuit to the input of the Schmitt trigger. The feedback circuit stage reduces the ISI caused by voltage over-scaling while the Schmitt trigger stage smooths out any glitches created by the feedback circuit. On their own, the feedback circuit and the Schmitt trigger invert their inputs so the FEST circuit is non-inverting. Overall, the FEST circuit can be viewed as a variable threshold buffer that is robust to ISI and glitches. Figure 2·4 illustrates how the FEST circuit is connected to enhance pipelined combinational logic. A comparison of the switching thresholds of the four different designs – nominal circuit design, design with feedback equalizer circuit, design with Schmitt trigger circuit and design with FEST circuit, is shown in Table 2.2. Though the switching threshold of the design with the FEST circuit is higher (lower) than that of the design with feedback equalizer circuit for 0→1 (1→0) input transition, the design with FEST circuit takes advantage of the fast switching of the Schmitt trigger and reaches V_{DD} earlier than the remaining three designs.

Input Transition	Nominal design	Feedback Equalizer only	Schmitt Trigger only	FEST
$0 \rightarrow 1$	$400\ mV$	$280\ mV$	$600\ mV$	$300\ mV$
$0 \leftarrow 1$	$400\ mV$	$500\ mV$	$200\ mV$	$480\ mV$

Table 2.2: Comparison of switching threshold voltages for various designs.

2.4 Experimental Results

We examine two arithmetic circuits – a 4-bit Kogge-Stone adder (KSA) and a 4-bit 3-tap finite impulse response (FIR) filter to determine the effectiveness of our proposed FEST circuit. We compare three designs – nominal arithmetic circuit design, arithmetic circuit design with only the feedback equalizer circuit, and arithmetic circuit design with the FEST circuit. We do not consider the arithmetic circuit design with Schmitt trigger circuit as it does not exhibit graceful degradation in the word error rate (WER) with voltage scaling. For both the KSA and the FIR filter, we add the feedback circuit and the FEST circuit along the critical path(s). We simulate all three designs for the two circuits using the 22 nm predictive technology model (PTM) (Cao, 2009) with a nominal voltage of 800 mV. We scale the supply voltage to determine the trade-off between WER and energy consumption for all three designs. For each voltage value, we simulate 1000 random operations. In the energy consumption results, clock energy is not included, as the proposed design is independent of the clock energy. To determine computation errors in the KSA and FIR filter, we compare the simulated output with the output of error-free arithmetic logic. The WER is then calculated by dividing the total number of erroneous outputs with the total number of operations.

2.4.1 Kogge-Stone Adder

KSA is a parallelized version of a carry look-ahead adder (see Figure 2·6a). It computes its carry signals by calculating propagate (P) and generate (G) signals concurrently in each vertical stage. The result of the adder is generated in the last stage by XORing the output of the PG-calculator. The nominal design of our 4-bit KSA operates at 2 GHz. The analysis of the KSA has shown that the most errors happen in the higher order bits, which lie in the critical path. Conversely, the FEST circuit

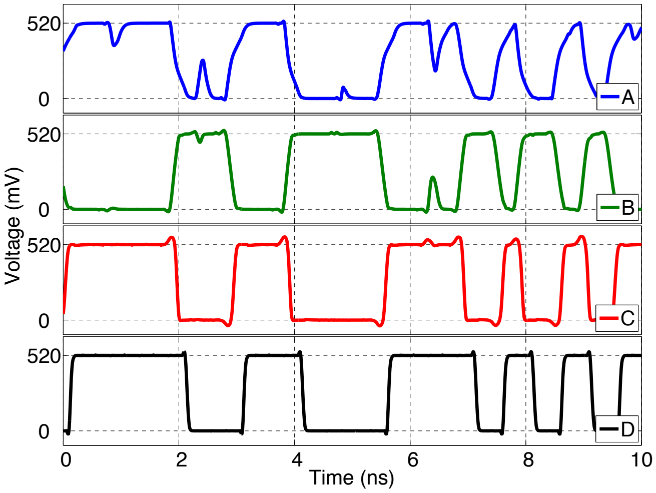
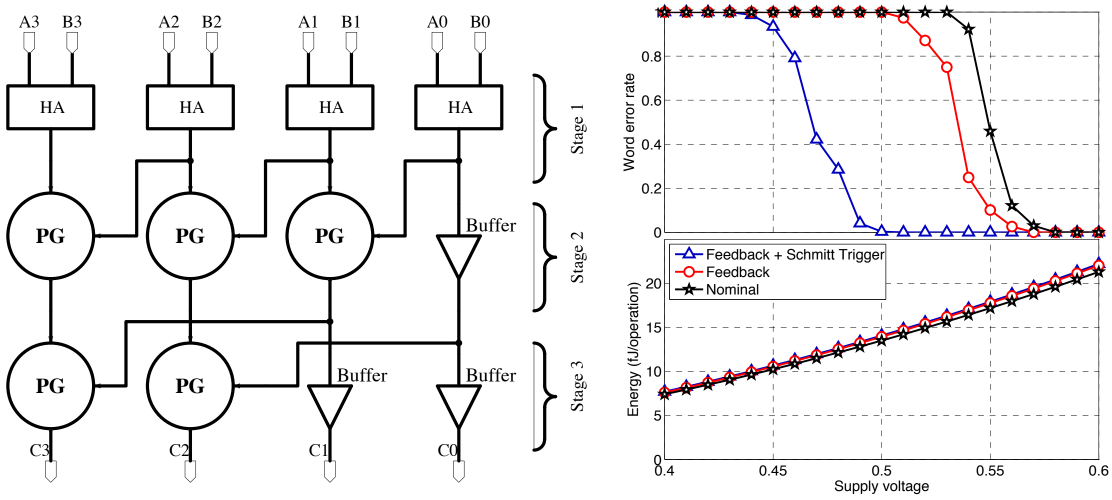


Figure 2·5: Transient response of the outputs of KSA (A), feedback equalization (B), Schmitt trigger (C), and latched signal (D).

errors are roughly uniform.

Figure 2·5 shows a 10 ns transient snippet of the most significant bit of the Kogge-Stone Adder, showing the output of the combinational logic (graph A), output of the equalizer (graph B), output of the Schmitt trigger (graph C), and the latched result (graph D). Observe that the output of the KSA combinational logic contains glitches which are the result of different data propagation times. If not filtered out, these glitches cause setup and hold time violations in the output registers. At the output of the feedback equalizer (B), the rise-fall time improves, however some of the glitches are amplified. The Schmitt trigger filters the glitches (C) but slightly delays the signal. However, due to a very low rise-fall time of the Schmitt trigger, it completes its transition earlier then the original signal A, enabling the register to record the results correctly (D). At 520 mV, the average delay and rise/fall times of the FEST circuit are 40ps and 70ps, respectively. The KSA without FEST has a rise/fall time of 190ps. If we assume that the rise and fall is linear, the KSA with FEST reaches the 90% point 20ps earlier than the KSA without FEST.

Figure 2·6b shows the WER of the KSA for the three designs. For the nominal

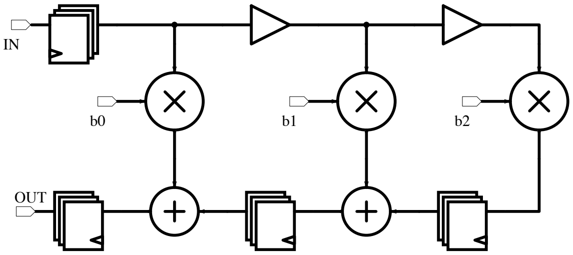


(a) **(b)**

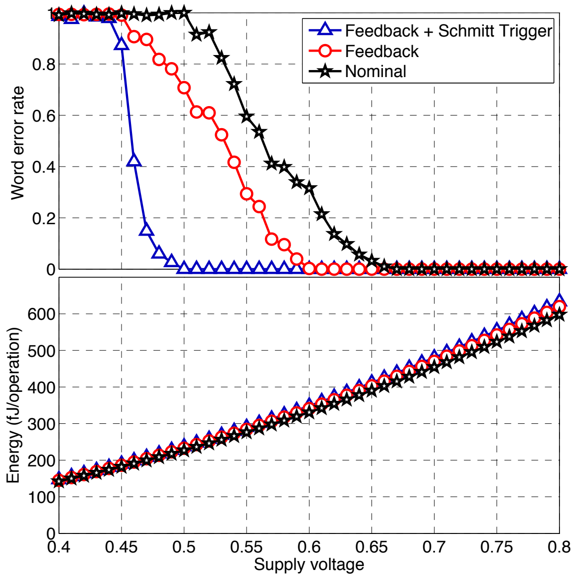
Figure 2·6: 4-bit Kogge-Stone Adder: a) Block diagram – HA = Half adder, PG = Propagate-Generate unit b) Word error rate and energy consumption for the nominal design, design with only the feedback circuit, and design with the FEST circuit.

design, errors start to appear below 580 mV. At this point, the charge and discharge times of the transistor parasitic capacitors are large enough to cause interference between computations. The feedback circuit provides comparatively faster charging and discharging paths so it is possible to scale the voltage slightly lower to 570 mV before we start seeing errors due to both glitches and timing errors. The FEST circuit is able to suppress glitches and timing errors until the voltage falls below 510 mV, which leads to significant energy savings.

Figure 2·6b also shows the energy consumption (data-dependent and fixed) per operation for each design. Note that the energy overhead of the feedback equalizer and FEST circuits is fairly small. To keep the WER near zero, nominal design and design with the feedback equalizer circuit require $\approx 19.5\ fJ/op$. For the same performance, design with the FEST circuit consumes $\approx 15.5\ fJ/op$ providing a 20% savings in energy. If an application is error-resilient at the algorithm level and can tolerate a relatively large fraction of errors, the proposed FEST circuit can offer even larger



(a) Block Diagram



(b) Word error rate and energy consumption

Figure 2·7: 4-bit 3-tap finite impulse response (FIR) filter: a) Block diagram; b) Word error rate and energy consumption for the nominal design, design with only the feedback circuit, and design with the FEST circuit.

energy savings. For example, at a WER of 0.1, the FEST circuit uses 25% less energy than the other designs.

2.4.2 Finite Impulse Response Filter

To determine the scalability of the proposed FEST circuit, we consider a 4-bit 3-tap FIR filter. Finite impulse response (FIR) filters are widely used in data communications and audio processing. The output of the FIR filter is defined as the convolution of the input signal x with an impulse response h. Specifically, the output is given by

$$y[n] = \sum_{i=0}^{N} h_i x[n-i], \tag{2.3}$$

where $x[n]$ is the input signal, $y[n]$ is the output, h_i is the weight of the i-th tap, and N is the total number of taps (or filter order).

For our case study, we designed and simulated an FIR filter using Wallace tree multipliers and Kogge-Stone adders as building blocks (see Figure 2·7a for a block diagram). The critical path, as seen in the block diagram passes through a buffer, multiplier, and an adder. At the nominal supply voltage of 800 mV, the 3-tap FIR filter reliably operates at 500 MHz.

One of the main sources of errors for this filter implementation is that the input signals to the adders arrive at different times – out of the two input signals, one of the signals passes through a multiplier before arriving at the input of the adder, while the second signal has no such delay. This causes the output signal of the adder to have a glitch. At nominal voltage, this glitch has a small width and does not get sampled by the flip-flop following the adder. However, as the supply voltage is scaled down, the width of the glitch increases, and at a critical voltage it is incorrectly sampled and interpreted as an independent bit value by the flip-flop following the adder. The FEST circuit enables us to further scale down the supply voltage while maintaining reliable operation by mitigating the errors resulting from glitches and slow transitions.

In Figure 2·7b, we have plotted the WER for each of the three designs operating at 500 MHz. For the nominal design, design with only the feedback equalizer circuit, and design with FEST circuit, the errors start to appear below 680 mV, 610 mV, and 510 mV, respectively. Figure 2·7b also shows the energy consumption of the 3-tap FIR filter for the three designs. To keep the WER near zero, the energy consumed by the nominal design, design with only the feedback equalizer circuit, and design with FEST circuit is 420 fJ/op, 350 fJ/op, and 250 fJ/op, respectively. Thus, the design with only the feedback equalizer circuit and design with the FEST circuit can provide 16% and 40% energy savings. If the overlying algorithm is error resilient up to a 10% error rate, then we get 5.7% energy savings with only the feedback circuit and 34.3% energy savings. The area overhead for the FEST circuit is 15.8%.

2.5 Experimental Results for Near-Threshold Design

The results presented here are an extension to the previous results. This work was done after "equalized pass-transistor logic" (E-PTL), which is described in Chapter 3. The reason E-PTL project was stalled is that implementation of PTL logic is extremely hard to automate, which makes it less appealing due to very high design costs. In addition to that PTL synthesis is outside of the scope of this research.

2.5.1 Experimental Setup

To evaluate our approach of using feedback equalization, several digital blocks were designed using Cadence Encounter and 40 nm GF process technology. The digital blocks were synthesized, place-and-routed (PAR) and RC-extracted using two different standard cell libraries - the baseline provided by Synopsys, and custom library that we developed specifically for operating under multiple voltage settings. The following computational blocks were designed for evaluation: 24-bit Wallace Tree multiplier, AES-256, Reed-Solomon Decoder (204, 188), and 256-point, radix-8 FFT using Verilog HDL.

In order to achieve the best average energy-delay operation in the DVFS-enabled custom systems, transistor sizing was subject to an iterative sizing approach, where the gates were first designed to operate optimally at near-threshold voltages using the approaches described in (Roy et al., 2015, Kwong and Chandrakasan, 2006), and then sized-up until a better **average** operation point in the DVFS-enabled system is achieved. This approach does not guarantee the best FoMs at any given voltage setting, however it allows for the minimization of the average EDP over the entire voltage range, which is beneficial during dynamic voltage scaling.

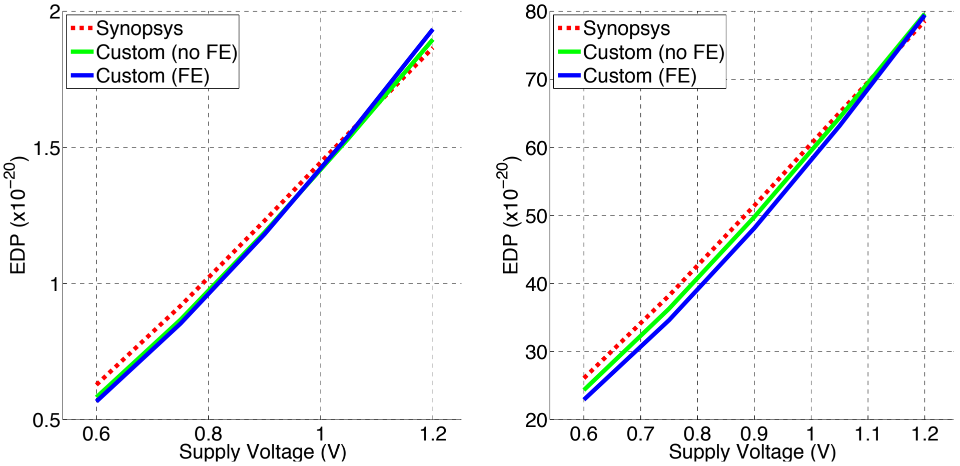
Baseline designs were PARed using standard cells provided by Synopsys, while custom designs followed the upsizing approach described earlier. Custom designed systems were sized such that they would provide reliable operation in the presence of process, voltage, and temperature variations (PVT). Voltage and process variations were assumed to be 10% of the nominal, while temperature was assumed to vary from $-25\,^{\circ}\mathrm{C}$ to $125\,^{\circ}\mathrm{C}$.

The maximum operating frequency was determined by stress testing the design. Essentially, we operated the design over a range of frequencies and over a million random inputs to determine the highest frequency at which no timing errors were observed. The operating energy was determined by keeping track of the current drawn by the voltage supply throughout the simulation.

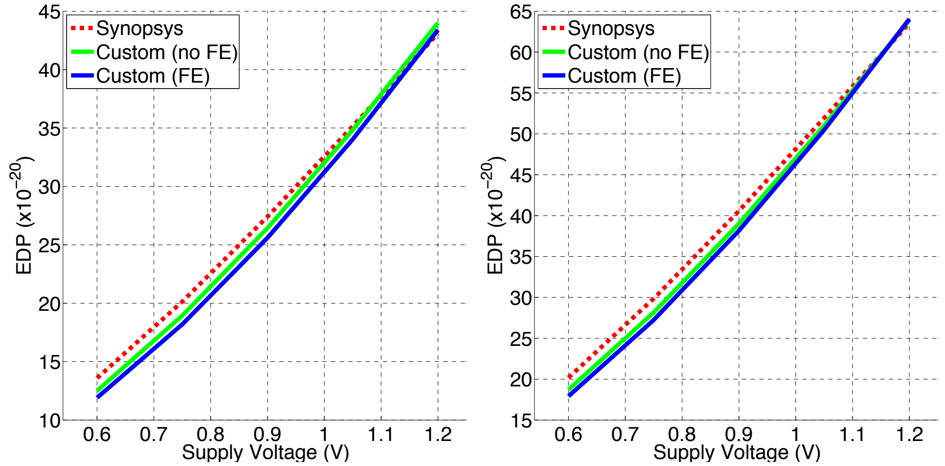
2.5.2 Experimental Results

Table 2.3 shows the maximum operating frequency and the corresponding energy dissipation of the evaluated designs. Because the delay is being traded-off for energy dissipation, we have also included the energy-delay product (EDP) metric in the table. At near-threshold voltage, our proposed equalized design has lower EDP than the other two designs for all digital blocks. At nominal supply voltage, our proposed equalized design has marginally higher EDP. Figure 2·8 shows the comparison of EDP across multiple voltage ranges. From the plots we can see that the EDP for E-FF designs is lower at near-threshold voltages, and only slightly higher in the super-threshold. The average improvement in EDP close 5% across the entire voltage ranges for all designs.

Figure 2·9 compares the equalized approach of the 24-bit multiplier to non-equalized



(a) 24-bit Multiplier **(b)** AES-256



(c) Reed-Solomon Decoder **(d)** Fast Fourier Transform

Figure 2·8: EDP results across entire above-threshold range

in the presence of PVT variations. The Monte-Carlo simulation results indicate that for equalized design we have $3\% - 5\%$ lower spread in operating frequency (σ^2/μ). In the best case scenario (the highest error-free operating frequency), at near-threshold voltage, the non-equalized design operates at $\approx 1.6GHz$, while equalized multiplier operates at $\approx 2GHz$. This translates to $\approx 25\%$ increase in operating frequency, while consuming $\approx 15\%$ more energy, which manifests itself as lower EDP. Figures 2·10, 2·11, and 2·12 show similar results in the AES, Reed-Solomon, and FFT respectively.

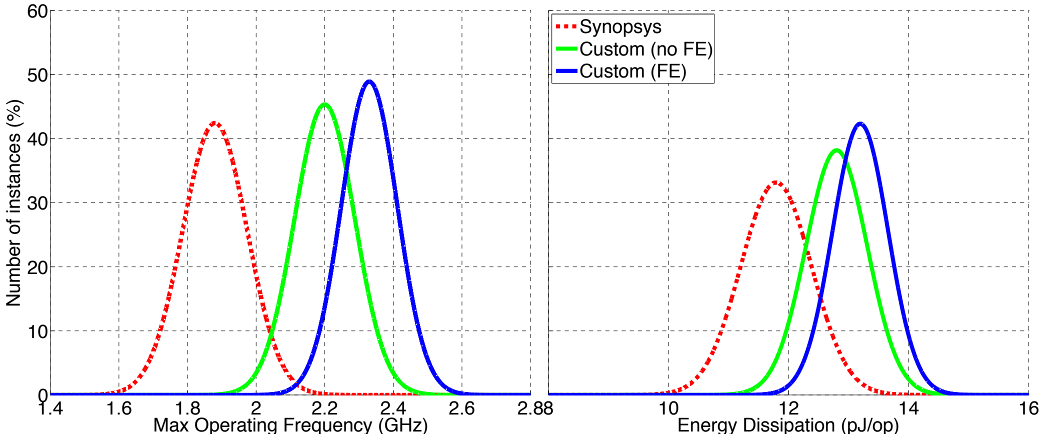


Figure 2·9: Distribution of the maximum operating frequency and energy dissipation in a 24-bit Multiplier under PVT variations. The dashed line represents the standard cells provided by Synopsys; solid lines represent the custom standard cell library designed by us. Equalized design shows improvement in performance by ≈ 25%, while consuming ≈ 15% more energy.

2.6 Conclusion

In this chapter we presented a feedback equalizer with Schmitt trigger (FEST) circuit for use in error mitigation in digital CMOS logic. This work is one of the first to propose the use of equalization in computational circuits as a tool for error mitigation. The results of the work have shown that FEST circuit enables rapid switching of the logic gates thus decreasing the worst case propagation delay in sequential blocks. This allows for reducing energy dissipation in digital logic through provided opportunity for further scaling of supply voltage.

As a case study, a 4-bit Kogge-Stone Adder (KSA) and a 3-tap 4-bit finite impulse response (FIR) filter were implemented using 22nm CMOS technology. In case of KSA, 20% reduction in energy dissipation was observed, while maintaining an operating frequency of $2GHz$ with no errors. Similar results were observed in the FIR filter simulations: up to 40% reduction in energy dissipation at the operating frequency of $500MHz$.

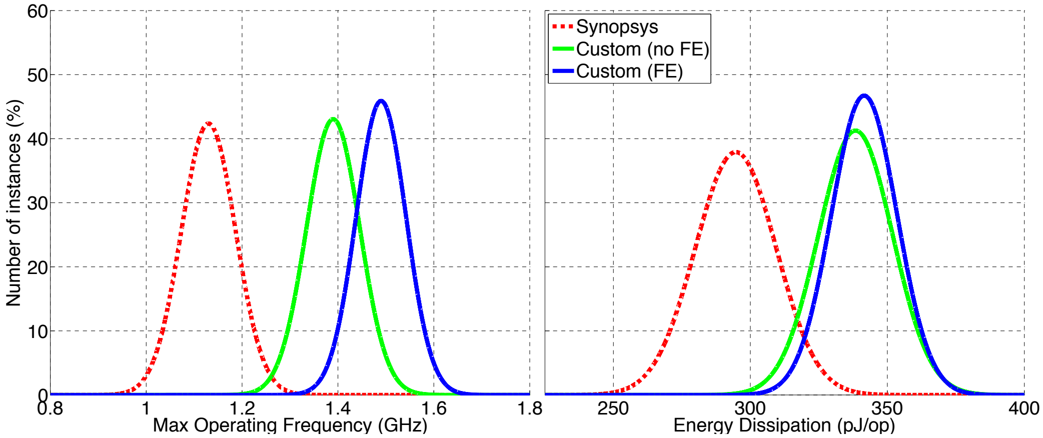


Figure 2·10: Distribution of the maximum operating frequency and energy dissipation in a AES-256 core under PVT variations. The dashed line represents the standard cells provided by Synopsys; solid lines represent the custom standard cell library designed by us. Equalized design shows improvement in performance by ≈ 40%, while consuming ≈ 20% more energy.

As part of the extended evaluation process we have designed the non-equalized systems (baseline) using standard cells provided by Synopsys for the 40nm Global Foundries process node. We have developed a library of custom standard cells for equalized designs using the same process technology. We have designed and analyzed a 24-bit multiplier, AES-256, Reed-Solomon Decoder (204, 188), and 256-point Fast-Fourier Transform blocks. The systems where the equalization was used, we saw improvement in operating frequency of up to 30%, while the energy increase was lower than 15%. We also showed the reduction of average EDP across entire above-threshold voltage range (of up to 10%). The Monte-Carlo analysis of digital blocks have shown that in the worst case scenario the equalized system is capable of operating at much higher performance while maintaining the energy dissipation.

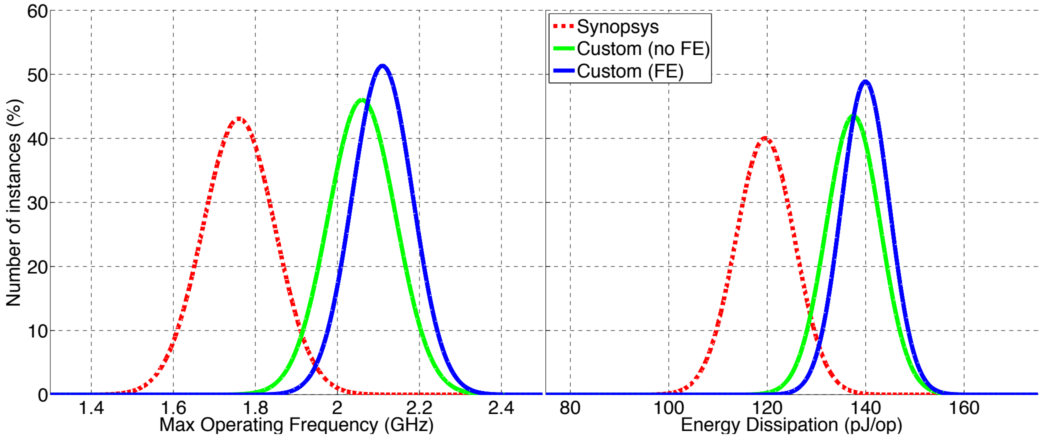


Figure 2·11: Distribution of the maximum operating frequency and energy dissipation in a Reed-Solomon Decoder core under PVT variations. The dashed line represents the standard cells provided by Synopsys; solid lines represent the custom standard cell library designed by us. Equalized design shows improvement in performance by ≈ 20%, while consuming ≈ 15% more energy.

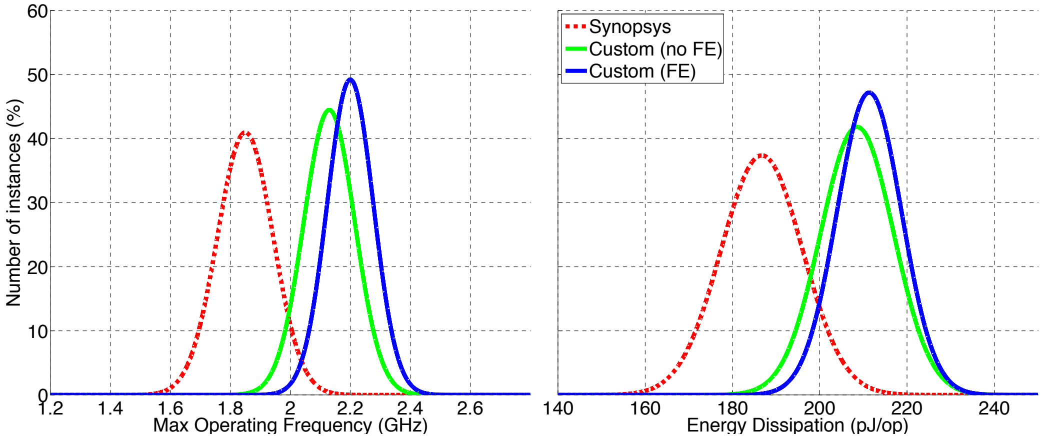


Figure 2·12: Distribution of the maximum operating frequency and energy dissipation in a Fast-Fourier Transform (FFT) core under PVT variations. The dashed line represents the standard cells provided by Synopsys; solid lines represent the custom standard cell library designed by us. Equalized design shows improvement in performance by ≈ 20%, while consuming ≈ 15% more energy.

Design	Operating Freq. (GHz)	Energy Dissip. (pJ/op.)	EDP
Multiplier			
Standard	1.88	11.8	6.3
Custom (no FE)	2.2	12.8	5.8
Custom (FE)	2.33	13.2	5.7
AES-256			
Standard	1.13	294.7	260.8
Custom (no FE)	1.39	338.4	243.5
Custom (FE)	1.49	341.6	229.2
Reed-Solomon Dec.			
Standard	1.76	239.2	135.9
Custom (no FE)	2.06	275.0	133.5
Custom (FE)	2.11	279.9	132.7
FFT			
Standard	1.85	373.4	100.9
Custom (no FE)	2.13	417.1	97.9
Custom (FE)	2.2	422.7	96.1

(a) $V_{DD} = 0.6V$

Design	Operating Freq. (GHz)	Energy Dissip. (pJ/op.)	EDP
Multiplier			
Standard	2.53	47.2	18.6
Custom (no FE)	2.70	51.8	19.2
Custom (FE)	2.73	52.2	19.3
AES-256			
Standard	1.51	1178.7	785.8
Custom (no FE)	1.70	1353.6	796.2
Custom (FE)	1.73	1366.2	803.7
Reed-Solomon Dec.			
Standard	2.42	429.1	429.2
Custom (no FE)	2.56	449.6	449.6
Custom (FE)	2.58	458.3	458.3
FFT			
Standard	2.49	644.6	644.6
Custom (no FE)	2.61	657.6	657.5
Custom (FE)	2.64	663.8	663.8

(b) $V_{DD} = 1.1V$

Table 2.3: Evaluation results for variaous computation blocks at 0.6V and 1.1V. *Standard* represents the design PARed with standard cells provided by Synopsys; *Custom* corresponds to the custom cells the we designed using up-/down- sizing approach. *no FE* and *FE* indicate if design included the feedback equalizer. EDP FoM is normalized to $1{\times}10^{-21}$

Chapter 3

Energy Efficient Pass-Transistor Logic Using Decision Feedback Equalization

3.1 Introduction

In the previous chapter (2) we introduced the FEST circuit to be used for mitigating the timing errors caused by aggressive voltage scaling. Elaborating on this work we have developed an equalized system for use in differential signal systems, and in particular for use in pass transistor logic. In this chapter we will review the proposed system as well as the analysis of performance, energy efficiency, as well as modeling approach for (equalized) pass-transistor logic (PTL).

As mentioned previously, the performance of the CMOS-based computing systems is being increasingly constrained by limiting power budgets (Fuller and Millett, 2011). Moreover, the continuing scaling of CMOS devices has increased the probability of occurrence of faults/defects, which has made the need for the novel designs (Borkar et al., 2004, Gielen et al., 2008). For lowering power dissipation, several techniques including the supply voltage scaling, the use of sleep transistors, the use of pass-transistor logic (PTL), the use of multiple threshold voltage (V_{th}) devices and the dynamic scaling of V_{th} are being widely deployed. To mitigate/detect-and-correct the errors manifesting due to the faults/defects in the CMOS devices various techniques including redundant latches/paths, slack redistribution, and confidence-driven computation have been proposed.

As with the FEST circuit described in Chapter 2, we took inspiration from communication theory and proposed a novel circuit design technique that uses equalization to lower energy consumption, improve performance and manage errors in digital logic circuits. The use of the FEST circuit described earlier, although showed improvement was focusing only on improving the last gate of the combinational logic which limited the potential savings in terms of energy and performance. The use of PTL with equalization takes care of the problem due to the RC nature of the PTL. We propose a novel differential equalized pass-transistor logic (E-PTL) that dynamically adjusts the strength of the currents in its internal paths to ease the logic circuit output transitions, and in turn mitigates timing errors and creates opportunities for lowering power dissipation and/or improving performance. Our proposed E-PTL can be readily incorporated into the digital flow for designing both low-power custom ASIC and general-purpose processors. The main contributions of this work are as follows:

- We propose a novel differential E-PTL circuit design technique that enables aggressive voltage scaling to lower energy consumption and/or enables aggressive over-clocking to improve performance, while mitigating the occurrence of timing errors by dynamically adjusting the strength of the current in its internal paths.

- We present detailed circuit-level power, error and delay models for the E-PTL circuit. We present the formulation of a convex optimization problem using these models to determine the minimum energy design for a given performance and error constraint. We solve the optimization problem using the CVX toolbox (CVX Research, Inc., 2012, Grant and Boyd, 2008) and validate our model-based design against HSPICE simulations for an example 16-bit adder.

- We compare E-PTL, conventional PTL and static complementary CMOS logic (SCL)-based designs of four different arithmetic blocks. Our proposed technique

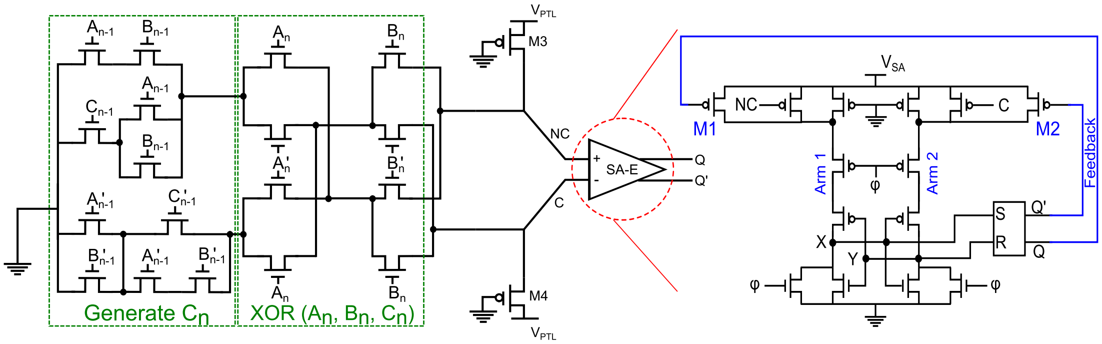


Figure 3·1: Schematic diagram of a Equalized Pass-Transistor Logic (E-PTL) for $Sum_n = A_n \oplus B_n \oplus C_n$, where $C_n = A_{n-1} \cdot B_{n-1} + A_{n-1} \cdot C_{n-1} + B_{n-1} \cdot C_{n-1}$.

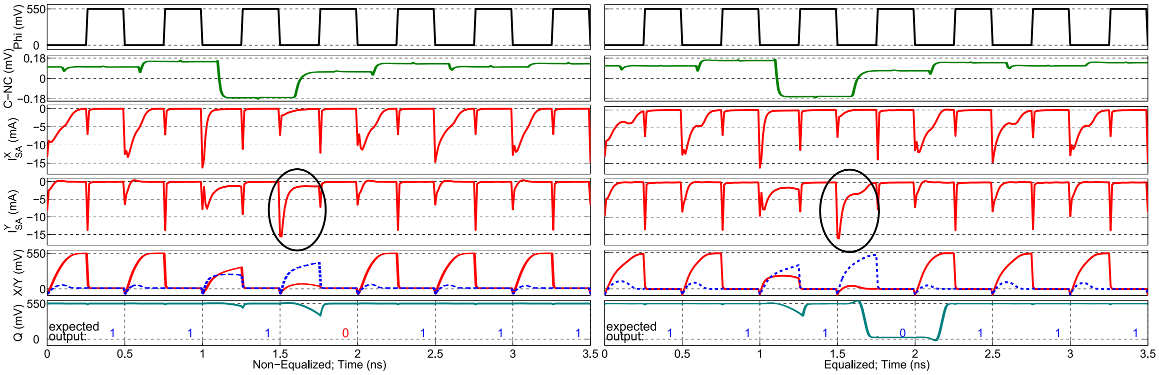


Figure 3·2: Timing diagram of a non-equalized (left) and equalized (right) system. The highlighted waveforms in row 4 show the boosted current in equalized system compared to the non-equalized system.

reduces energy consumption by up to 30% on average while sustaining the circuit throughput and maintaining target error rates. We also evaluate the variability tolerance of our proposed design technique.

3.2 Equalized Pass-Transistor Logic

We propose the Equalized Pass-Transistor-Logic (E-PTL) as a low-power alternative to conventional static CMOS logic (SCL).The choice of PTL for designing equalized digital logic circuits is driven by the fact that the equivalent resistance-capacitance (RC) model of a PTL design closely resembles the RC model of on-chip communication links, which makes it more amenable to the application of equalization techniques.

Figure 3·1 shows the circuit topology for our proposed E-PTL design technique. The circuit consists of two stages - PTL network and sense-amplifier + DFE (SA-E). The non-equalized PTL design is same as E-PTL except that the SA does not have any DFE. The PTL circuit is based on DCVSL family, and has PMOS transistors with gates connected to logic 0 acting as pull-up transistors. The PTL network consists of two sub-networks, one each for the complemented and non-complemented implementation of the minimized sum-of-products (SOP) form of the logic function. The product term (AND) is implemented using pass transistors in series, while the sum operation (OR) is implemented by connecting the product implementations in parallel. The gate inputs of the NMOS devices in the PTL sub-networks are controlled by the outputs of the sense amplifiers in the previous pipeline stages. Both PTL and SA-E have their own dedicated supply voltages – V_{PTL} and V_{SA}. In the circuit shown in Figure 3·1, one sub-network of the PTL stage has been designed to perform $A_n \oplus B_n \oplus C_n$ operation, while the other sub-network performs $\overline{A_n \oplus B_n \oplus C_n}$ operation. These sub-networks complete their operation during the positive half of the clock cycle, and the outputs are fed to the differential input NC and C of the SA-E. In the negative half of every clock cycle, the DFE in E-PTL is used to dynamically adjust the strength of the current in each arm of the SA based on the data sampled in the previous clock cycle.

Figure 3·2 shows the timing waveforms of the non-equalized and equalized design of our sample circuit, where the expected output bit stream is 1110111. At 1.5 ns the $V_C - V_{NC}$ value is negative, which corresponds to an expected output of logic 0. Note that the rise/fall times of the $V_C - V_{NC}$ are very steep. The reason for this is that due to tight error rate constraints ($BER \approx 0$ in the worst case scenario) the optimization process tends to converge to larger channel width transistors, thus in average cases making the rise/fall times steep. In the non-equalized case, the strength of current in Arm 2 (i.e., $I_0 + I_C$) is not sufficient to trip the cross-coupled inverters in a timely manner. Here I_0 is the current through the two arms when $V_C = V_{NC} = V_{SA}$, I_C is the current due to the voltages at nodes C. This lower current strength is due to the fact that $V_C - V_{NC}$ has not completely reversed within the allocated half clock period. This partial reversal is due to the slow switching of the transistors in the PTL. This phenomenon leads to the latch maintaining its previous output of logic 1. In the equalized case, the logic 1 output from previous cycle switches ON transistor M2, which provides a boost to the current in arm 2 ($I_0 + I_C + I_{FB}^{Q'}$), and trips the cross-coupled inverter at an earlier time than that in the non-equalized case (in spite of the transistors in the PTL switching equally slowly). The SR latch is then able to correctly sample the data. Here, $I_{FB}^{Q'}$ is the current through transistor M2. The current boost provided in the SA through equalization provides opportunities for aggressive voltage scaling to lower energy consumption and/or over-clocking of the circuit to improve performance. It should be noted that the transistors M1 and M2 need to be sized carefully in order to avoid under- and over-equalization. Under-equalization can lead to a situation where the amount of feedback current is not sufficient to ensure correct operation. On the other hand, over-equalization can lead to larger than required boost to the current leading to incorrect tripping of the cross-coupled inverters. In addition, we also compared a single-ended E-PTL (i.e., sense amplifier receives the one input from PTL and the other input is a threshold voltage) with our differential E-PTL. Unlike the differential E-PTL, in the single-ended E-PTL approach different threshold voltages were required for different PTL networks, which led to a non-trivial overhead. On the other hand, using same threshold voltage for all PTL networks led to a sub-optimal design.

3.3 Modeling and Design Automation of E-PTL

In this section, we present detailed models for the power dissipation, bit error rate, and performance of E-PTL logic. We also present an automated tool-flow that uses these models to generate an energy-efficient design that meets error rate constraints.

3.3.1 Power Modeling

Our E-PTL circuit consists of two stages: the PTL stage and the SA-E stage. The different components of power dissipation in the PTL stage can calculated as

$$P_{PTL}^{dynamic} = \left(V_{SA}^2 \sum_i C_{g,PTL}^i + V_{PTL}^2 \sum_i C_{d,PTL}^i \right) \cdot f \cdot \alpha \tag{3.1}$$

$$P_{PTL}^{static} = \frac{V_{PTL}^2}{\sum_i (R_{on}^i + R) \| \sum_i (R_{off}^i + R)} \tag{3.2}$$

$$P_{PTL}^{leak} = V_{PTL} \mu_0 C_{ox} \sum_i \frac{W_i}{L_i} (n-1) V_T^2 \cdot e^{\frac{-V_{th}}{nV_T}} \tag{3.3}$$

where V_{SA} is the sense amplifier supply voltage, V_{PTL} is the PTL supply voltage, V_{th} is the threshold voltage of the transistors, C_g^i is the gate capacitance of the transistor, C_d^i is the diffusion capacitance of the transistor, R_{on}^i is the resistance of the transistor in saturation, R_{off}^i is the resistance of the transistor in cutoff, R is the resistance of the pull up transistors M3 and M4 (see Figure 3·1), f is the operating frequency, α is the activity factor, μ_0 is the carrier mobility, C_{ox} is the oxide capacitance of a transistor, n is a technology dependent parameter, V_T is the thermal voltage, and W_i and L_i are the width and length of the transistor i.

The power consumed in SA-E stage can be calculated as

$$P_{SA} = \alpha \cdot 2V_{PTL}^2 C_{g,in}^{SA} \cdot f \ . + V_{SA} I_{min} \beta \ + P_{latch}. \tag{3.4}$$

The first term corresponds to the dynamic power consumed in charging/discharging the gate capacitance of the PMOS transistors that receive C or NC as inputs. The

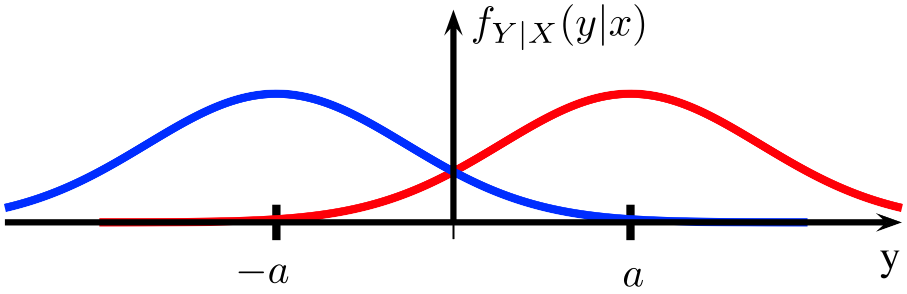


Figure 3·3: Conditional probability density function for the noisy PTL output $y = x + z$ where x is either a or $-a$ and z is zero-mean Gaussian noise.

second component is the static power consumed in the sense-amplifier. Here, I_{min} is the total current passing through the sense-amplifier when all transistors are minimum sized and β is the scaling factor corresponding to the sizing up of all the transistors (by the same scale) to scale the current. P_{latch} corresponds to the (dynamic and static) power consumed by the latch. We have ignored the power consumed in the wires and clock as we expect the equalized PTL and non-equalized PTL designs to have minimal difference in these two power components.

3.3.2 Error Rate Modeling

Timing errors in a circuit are caused by inter-symbol interference (ISI) due to variations in circuit RC delay. The change in the delay of a circuit can be due to voltage scaling, process-voltage-temperature variations, negative bias temperature instability, cosmic radiation, noise, etc, which change the RC properties of a system, and thus affect the transition time of the various nodes of the circuit. Below, we model the probability of error, first in the absence of ISI and then in its presence.

Noise Model

Our model focuses on the impact of noise at the SA stage, where the differential output of the PTL stage is thresholded into a logical output and latched. For now, assume that the observation at the input of the SA stage, sampled once per clock period, can be written as $y_i = x_i + z_i$ where x_i is the output of the PTL stage and z_i is noise. For a logical 1, the differential PTL output is $x_i = a$ and, for a logical 0, it is $x_i = -a$ for some positive value a. The noise z_i is assumed to be independent of x_i and Gaussian with mean zero and variance σ^2 (see Figure 3·3).

The SA stage simply thresholds its observation: it latches a logical 1 if $y_i \geq 0$ and a 0 if $y_i < 0$. Clearly, the probability of error will decrease if the strength a of the differential PTL output increases. The probability of making an incorrect decision is given by the probability that the noise pushes the PTL output across the decision threshold $p_{\text{error, no ISI}} = Q(a/\sigma)$ where $Q(v) \triangleq \int_v^\infty \frac{1}{\sqrt{2\pi}} \exp(-\frac{u^2}{2}) du$.

ISI Model

The error model above ignores the possibility of ISI. That is, it assumes that the previous PTL differential output has been completely dissipated when the SA thresholds the current output. However, in our considerations, we assume that the supply voltage is scaled to the point that ISI is a significant factor. To quantify the effect of ISI, consider a simple low-pass RC filter (see Figure 3·4). The voltage across the resistor R is defined by $i(t)R = V_{in}(t) - V_{out}(t)$, where $i(t) = \frac{dQ_c}{dt}$ is the current flowing through the resistor, and $Q_c = CV_{out}(t)$ is the charge at the capacitor. Rearranging the equations, we get

$$V_{in}(t) - V_{out}(t) = RC \frac{dV_{out}(t)}{dt} \tag{3.5}$$

In discrete time (with a sampling period of Δt), this becomes

$$V_{in,i} - V_{out,i} = RC \frac{V_{out,i} - V_{out,i-1}}{\Delta t}. \tag{3.6}$$

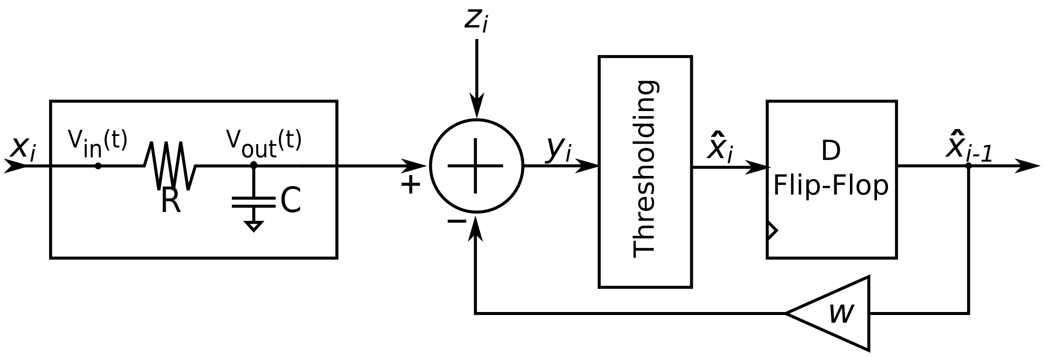


Figure 3·4: Model of our proposed E-PTL. R and C represent equivalent parasitics in the PTL. Thresholding and latching is performed using the sense-amplifier.

Equivalently,

$$V_{out,i} = (1-\omega)V_{in,i} + \omega\ V_{out,i-1} \tag{3.7}$$

$$\omega = \frac{RC}{RC+\Delta t}\ . \tag{3.8}$$

As the RC-delay increases (or the clock period decreases), ω approaches 1, and the previous PTL output starts affecting the current output. However, if $\Delta t \gg RC$, the effect of the ISI approaches zero. Notice that the input-output relationship in (3.7) has an infinite impulse response. For our purposes, we can safely assume that RC and Δt are such that all but the first-order ISI term have a negligible effect,

$$V_{out,i} \approx (1-\omega)V_{in,i} + \omega V_{in,i-1}\ . \tag{3.9}$$

Probability of Error

Combining our ISI and noise models, we arrive at the following model of the input to the SA stage

$$y_i = (1-\omega)x_i + \omega x_{i-1} + z_i\ , \tag{3.10}$$

where $x_i \in \{-a, a\}$ is the differential output of the PTL stage at clock period i and z_i is independent zero-mean Gaussian noise with variance σ^2. The SA thresholds its observation y_i and latches it. If we make no attempt to mitigate the ISI, it can significantly increase the probability of error. In the worst case, the current and previous PTL outputs have opposite signs. For instance, if $x_i = a$ and $x_{i-1} = -a$, then the SA observation will be

$$y_i = (1 - 2\omega)a + z_i. \tag{3.11}$$

Thus, probability of error with ISI can be upper bounded as

$$p_{\text{error, ISI}} \leq Q\left(\frac{(1 - 2\omega)a}{\sigma}\right). \tag{3.12}$$

The differential PTL output a is a function of

$$\begin{aligned} a &= |I_C - I_{NC}| && (3.13)\\ &= \mu_0 C_{ox}\frac{W}{L}(n-1)V_T^2\left(e^{\frac{V_{GS}^{+}-V_{th}}{nV_T}} - e^{\frac{V_{GS}^{-}-V_{th}}{nV_T}}\right) && (3.14)\end{aligned}$$

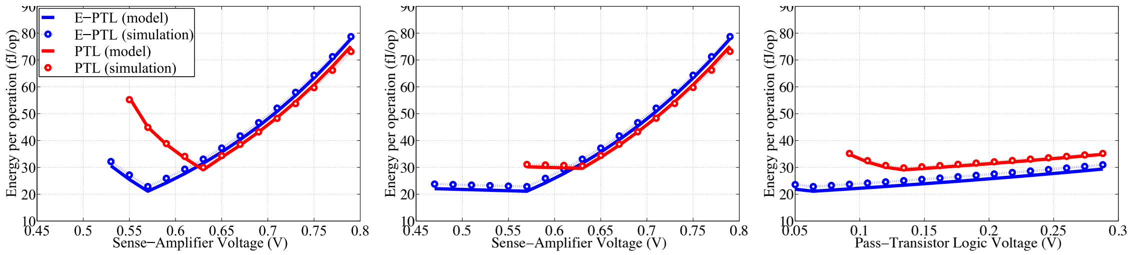
where,

$$V_{GS}^{+} = V_{PTL}\cdot\frac{\sum_i R_{on}^i}{\sum_i R_{on}^i + R}, \qquad V_{GS}^{-} = V_{PTL}\cdot\frac{\sum_i R_{off}^i}{\sum_i R_{off}^i + R} \tag{3.15}$$

Here the values for $\{R_{on}, R_{off}, R\}$ are dependent on the corresponding widths and lengths of the transistors in the PTL.

Impact of Decision Feedback Equalization

To counter the effects of ISI, our E-PTL architecture employs decision feedback equalization (DFE) prior to the SA threshold. Specifically, the circuit uses its estimate $\hat{x}_{i-1}$ of the previous PTL output x_{i-1} to remove the ISI from the SA observation.



(a) **(b)** **(c)**

Figure 3·5: Model vs. Simulation for 16-bit CLA. (a) The PTL voltage is fixed at its optimal value, while the sizing of the transistors is free for optimization. (b) The sizing of the transistors is fixed at its optimal value, while the PTL voltage is free for optimization. (c) The SA voltage is fixed at its optimal value, while the sizing of the transistors is free for optimization.

This results in the following new observation at the SA

$$\tilde{y}_i = (1-\omega)x_i + \omega x_{i-1} + z_i - \omega \hat{x}_{i-1} \ . \tag{3.16}$$

If $\hat{x}_{i-1} = x_{i-1}$ (meaning the DFE prediction was correct), then the SA will observe the current PTL output free of ISI, which significantly decreases the likelihood of an error. However, since $\hat{x}_{i-1}$ is the result of thresholding the previous noisy observation $\tilde{y}_{i-1}$, it is definitely possible that it is in error. If this is the case, the likelihood of an error will increase as the signal strength will be further diminished by DFE. The key point is that *on average* the error probability will decrease. This is well-established in the communication theory literature (Belfiore and Park, 1979) and this analysis can be carried out for our system model as well. However, owing to the non-linearity of the thresholding step used to produce $\hat{x}_{i-1}$, it is not possible to write down the error probability in closed form, although it can be accurately characterized using numerical methods.

3.3.3 Delay Modeling

The overall delay of the E-PTL circuit can be written as the sum of the PTL delay τ_{PTL} and the SA delay τ_{SA}. The PTL can be modeled as a simple RC network and its delay can be calculated using the Elmore delay technique. The SA delay consists of the delay from the falling edge of the clock until the input of one of the inverters increases above V_{th} as well as the setup time of the latch. It can be written as

$$\tau_{SA} \quad = \quad \frac{V_{th}^{P} C}{I_0 + I_{C/NC} + I_{FB}^{Q'/Q}} + t_{RS} \tag{3.17}$$

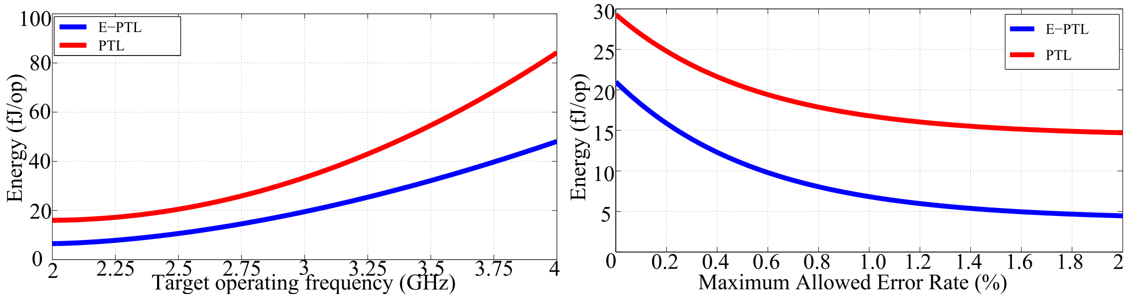
where, $I_{C/NC}$ is the current contributed by the input, $I_{FB}^{Q'/Q}$ current contributed by the feedback, and I_0 is the default current offset in the modified Strong-arm SA. The switch time of the RS latch is t_{RS} and $I_{C/NC}+I_{FB}^{Q'/Q}$ defines how fast the cross-coupled inverters switch. Note that whichever inverter reaches V_{th}^{P} at its source terminal first will dominate the cycle.

3.3.4 Optimization Tool-flow

The optimization tool-flow consists of several steps. The first step is converting a combinational function into a minimized sum-of-products (SOP) form. We used the Quine-McCluskey algorithm to generate minimized expressions. The second step is the formulation of the $\min \|\mathbf{Ax} - \mathbf{b}\|_1$ problem where the matrix $\mathbf{A}$ represents equations governing the energy dissipation, critical delays, error rates, etc. of the circuit, vector $\mathbf{b}$ represents the design goals and circuit constraints and vector $\mathbf{x}$ has the free parameters. We used the CVX optimization toolbox (CVX Research, Inc., 2012, Grant and Boyd, 2008) that solves $\min \|\mathbf{Ax} - \mathbf{b}\|_1$ and returns the optimal transistor parameters $\mathbf{x}$. The results of the CVX optimization toolbox as well as the previously generated minimized SOP can be fed into the subcircuit netlister which generates a SPICE netlist for further verification.

3.3.5 Modeling vs. Simulation

To validate our modeling approach, we designed a 16-bit carry-lookahead adder (CLA) via the optimization approach above and compared the energy usage predicted by our model to that obtained from HSPICE simulations for a 22nm PTM (Cao, 2009) technology model. Figure 3·5a shows the model-based optimization and simulation results for the 16-bit CLA when the V_{PTL} is kept fixed at the optimal value, and we sweep the transistor sizes for each value of V_{SA} to determine the minimum energy per operation. This figure shows that V_{SA} obtained from our model-based optimization matches that obtained by exhaustively searching the parameter space. Similarly in Figure 3·5b, we held the transistor sizing fixed and swept the V_{PTL} for each value of V_{SA} to determine the minimum energy point. Finally, for Figure 3·5c, we kept V_{SA} fixed at the optimal value and swept transistor sizing for each value of V_{PTL} to determine the minimum energy point. Overall, the design parameters obtained using our model-based optimization approach closely match the design parameters obtained through sweeping the design space via simulation.



(a) Energy vs Operating frequency with 1% word error rate

(b) Energy dissipation vs word error rate at 2 GHz.

Figure 3·6: 16-bit Carry-Look Ahead Adder

3.4 Evaluation

In this section, we first compare the E-PTL design with the non-equalized PTL design for different target frequencies and error rates using a 16-bit adder example. Then we compare E-PTL, PTL and SCL designs a 16-bit CLA circuit, an 8-bit multiplier circuit, an 8-bit 3-tap FIR filter circuit, and a 64-bit CRC circuit. Figure 3·6a shows the energy versus target operating frequency plot for both PTL and E-PTL design approaches for a 16-bit CLA. For each target frequency, we determined the PTL and E-PTL designs that consumed the least amount of energy per operation and had 1% word error rate. These designs were generated using the automated tool flow described in Section 3.3. On average, the E-PTL design consumes 20% less energy than the PTL design. This lower energy consumption is due to the fact that more aggressive voltage scaling is possible in E-PTL. Figure 3·6b shows a plot of energy per operation versus word error rate for the PTL and E-PTL designs for a 16-bit CLA operating at 2 GHz frequency. As the target word error rate increases, the two plots diverge because the E-PTL design has more flexibility in sizing the transistors in the PTL stage. On average, the E-PTL design consumes 45% less energy than the PTL design over a target word error rate range of 0% to 2%.

Table 3.1 shows a comparison of the energy per operation for SCL, PTL and E-PTL designs of a 16-bit CLA circuit, an 8-bit multiplier circuit, an 8-bit 3-tap FIR filter circuit, and a 64-bit CRC circuit. The adder and multiplier circuits were designed for 2 GHz, while the FIR filter and CRC circuits were designed for 500 MHz. All circuits were designed to have zero error rate, and the optimization problem objective was set to minimize energy-per-bit figure of merit. Compared to the SCL designs, the E-PTL designs have 30% lower energy per operation on average due to lower supply voltage in the computational part of the circuit ($V_{PTL} \ll$ nominal voltage). Similarly, compared to the PTL designs, the E-PTL designs have 15% lower energy per operation on average. The lower energy per operation in E-PTL is mainly due to the fact that the equalization technique enables more aggressive scaling of the supply voltage.

We have also compared the robustness of a 16-bit CLA designed using PTL and E-PTL. A Monte Carlo simulation was performed with 22 nm PTM models and a ±10% variation in supply voltage, channel length and temperature. Figure 3·7 shows the variation in delay and energy. The PTL design was optimized to have a delay of 500 psec. We considered two different E-PTL designs. E-PTL 1 design was optimized to have the same energy consumption (29.6 fJ/op) as the optimized PTL design. The mean delay for E-PTL 1 design was ≈ 365 ps. The E-PTL 2 design was optimized such that its worst-case delay (under variations) was less than 500 psec. The energy consumption in the E-PTL 2 design was approximately 27.5 fJ/op. Thus, the E-PTL 2 design creates a win-win situation which can tolerate variations in delay (i.e., it meets target performance) and simultaneously provides 7% lower energy consumption than PTL design.

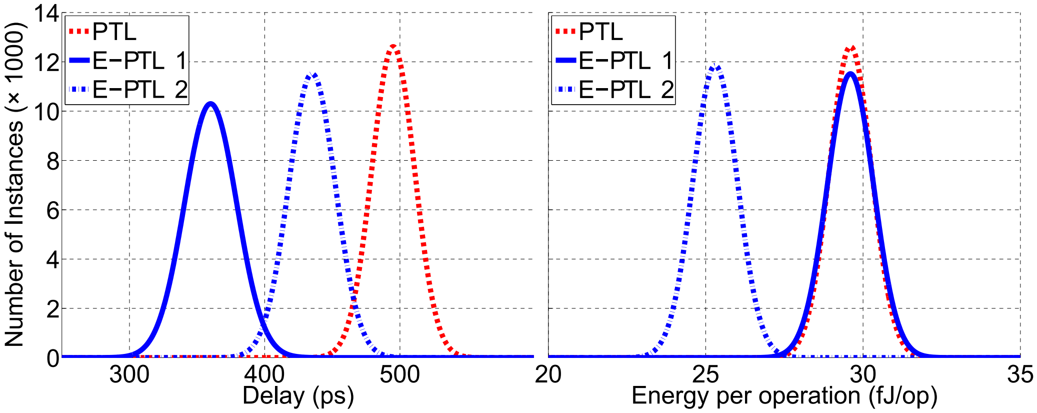


Figure 3·7: Monte-Carlo simulation results for delay and energy a 16-bit CLA. PTL and E-PTL 1 are the designed to operate at a fixed energy budget. E-PTL 2 is designed to have $\sim$ 0% failure rate in presence of variations at 2 GHz.

Goal: *f*	**Digital block**	**SCL**	**PTL**	**E-PTL**
2 GHz	16-bit CLA	45.1 fJ/op	29.6 fJ/op	21.1 fJ/op
2 GHz	8-bit Multiplier	285.1 fJ/op	219.6 fJ/op	204.3 fJ/op
500 MHz	8-bit 3-tap FIR	1750 fJ/op	1590 fJ/op	1360 fJ/op
500 MHz	64-bit CRC	259.2 fJ/bit	237.1 fJ/bit	217.0 fJ/bit

Table 3.1: Comparison of the minimum energy in SCL, PTL and E-PTL designs of various digital logic blocks. Word Error Rate is set to 0.

3.5 Conclusion

In this chapter proposed a novel equalized PTL design. We have shown that when PTL is combined with equalization techniques, we can lower energy dissipation while maintaining the goal frequency (performance). In addition to that computation in PTL logic is current based, thus the supply voltage could be scaled to extremely low values. We have shown that depending on the circuit topology, E-PTL allows for up to 30% reduction in energy dissipation, while keeping the performance and error rate metrics the same.

Chapter 4

Adaptive Classification: Energy-Efficient Design for Machine Learning

4.1 Introduction

Over the last decade, there has been a shift in the consumer electronics market towards mobile computing. Mobile workloads have become more diverse, and are increasingly trending towards utilizing computationally expensive machine learning approaches to process large amounts of data (Lane et al., 2010). Some examples of these problems are live translation (Wu et al., 2016), fingerprint matching (Unar et al., 2014), and diabetes testing (Sowjanya et al., 2015, Wu, 2015). Other applications gaining traction in the mobile domain involve processing time- series data using machine learning algorithms. These applications include live audio processing (Lee et al., 2009), fitness assessment (Kranz et al., 2013), and smart homes that can learn user's schedule (Rashidi and Cook, 2009). One of the challenges in these applications is that some inputs within time-series are easy to classify, while others require complex machine learning algorithms (Cinar et al., 2017). For example, the problem of recognizing human activities was shown to be a very complex and energy-consuming problem, currently tackled using recurrent neural networks (RNN) (Godfrey and Gashler, 2017, Ordóñez and Roggen, 2016, Olah, 2015). The main challenge in such applications is that the time-series input data usually requires signal pre-processing for feature extraction of the data to achieve high accuracy. Essentially, executing expensive machine learning algorithms on battery powered mobile systems with limited energy budget is challenging. It requires minimizing energy consumption of machine learning algorithms while achieving the desired accuracy.

Although most machine learning algorithms are designed to achieve high accuracy, the choice of a particular machine learning algorithm often involves a three-dimensional trade-off between error rate, energy dissipation, and computational performance. In order to achieve high accuracy, data scientists often choose to use complex non-linear classification functions such as ensemble methods, radial basis function (RBF), and neural networks that allow for flexible boundaries to be learned yielding strong classification accuracy. These non-linear classification functions, however, are computationally demanding resulting in lower energy efficiency (Venkataramani et al., 2015b). In contrast, simple decision functions such as sparse linear functions can be used for applications that can tolerate errors. The sparse linear functions have low complexity, and hence are very energy efficient. However, they lack the flexibility to learn complex decision boundaries necessary for high classification accuracy.

Typically, the same machine learning algorithm is used for classification of all data inputs within the same dataset. A problem with this approach is that it does not take into account the variability across individual input data. Some inputs within a dataset are "easy" and could be classified using a less complex linear function, while other inputs are "hard" and require non-linear classification functions. For most "easy" inputs, using a complex classifier results in a similiar classification accuracy as a simple classifier, but has higher engergy dissipation. Conversely, for "hard" inputs, using a simple classifier lowers engergy dissipation, but it results in a lower classification accuracy. In this section, we introduce an adaptive classifier approach that has a"chooser" function which identifies the "hardness" of the incoming data and then assigns it to the appropriate "core" classifier[1], taking both classification accuracy and energy dissipation into account. In our approach, the chooser function dynamically picks a classifier for each new input to maximize classification performance subject to an energy budget constraint. This approach also enables one to incorporate both existing and new machine learning approaches into constrained learning problems. Essentially, one can seamlessly incorporate optimized, complex classifiers into a budgeted system. Moreover, the budget can be dynamically changed during run-time by retraining the chooser function, which avoids the need to retrain complex classifiers. The main contributions of our work are as follows:

– We propose a novel adaptive classifier that uses a linear regression-based chooser function to classify the incoming data as "easy", "harder", and "hardest", and based on this classification the chooser function assigns the data to the most energy-efficient classifier that meets the target accuracy.

– We propose an energy budget-aware approach for training the "chooser" function. In this approach, the "chooser" function takes the current energy capacity of the battery and the "hardness" of data as input features to decide the "core" classifier. Here we assume that the adaptive classifier will receive the current energy budget as an input from the host processor or micro-controller connected to the adaptive classifier.

We implemented our adaptive classifier in hardware with Chisel (Bachrach et al., 2012) and Global Foundries 40nm technology node. The implementation includes the design of the chooser function and three different "core" classifiers: logistic regression based classifier (linear), SVM with a polynomial and RBF kernels. The evaluation results show that on average the proposed approach consumes up to 10× more energy as compared to a linear classifier, while achieving $\approx$ 40% higher accuracy. When compared to RBF SVM, our adaptive classifier consumes 2 orders of magnitude less energy, while introducing $\approx$ 0.5% higher error rate on average. The second proposed approach, which is budget-aware, is an improvement on the adaptive classifier, which

---

[1] Only one "core" classifier at a time is enabled by the "chooser"

achieves up to 11% lower energy dissipation on average as compared to the approach that is not budget-aware.

The rest of this section is organized as follows. We open up with discussion of the formal definition of "hard" and "easy" problems and a description of the adaptive classifier in Section 4.2. Section 4.3 introduces a detailed overview of the microarchitecture of the proposed approaches. We describe the evaluation of the current work in Section 4.4, with discussion and concluding remarks in Sections 4.5 and 4.6.

4.2 Adaptive Classifier – Theory

Table 4.1: Notation and variables used in the current work

Term	Description
$x_i(d) \in \mathcal{X}^{m \times D}$	Value of training input feature $i = \{1, ..., m\}$ and dimension $d = \{1, ..., D\}$ in a training set $\mathcal{X}$
$y_i \in \mathcal{Y}^{m \times 1}$	Value of training input label $i = \{1, ..., m\}$ in label space $\mathcal{Y}$
$(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$	Training input pair (x_i, y_i) in space $\mathcal{X} \times \mathcal{Y}$
$f_1, \ldots, f_k : \mathcal{X} \to \mathcal{Y}$	Set of k "core" classifiers f that map feature space $\mathcal{X}$ to label space $\mathcal{Y}$
$c_1, \ldots, c_k \in \mathbb{R}$	Cost of "core" classifiers*
$g : \mathcal{X} \to 1, \ldots, k$	"Chooser" classifiers g that maps input $\mathcal{X}$ to the "core" classifiers
$\beta \in \mathcal{B}$	Current energy budget β
ε_f	Error rate of classifier f
$\mathrm{Acc}_f = 1/\varepsilon_f$	Accuracy of classifier f
$\mathrm{EE}_f = 1/\mathrm{EDP}$	Energy efficiency of classifier f defined as inverse of EDP

* The cost of a classifier is expressed in either energy, delay, energy-delay product (EDP), or any other minimizable metric. In our case, the cost is EDP.

In this section, we first present the motivation for the adaptive classifier by describing the variability in hardness within the same dataset. We also present a detailed description of the algorithm and architecture of our adaptive classifier later in the current section.

4.2.1 Hardness of an Input and Penalty for Misclassification

As different machine learning algorithms perform differently in terms of energy-efficiency, statistical performance (error rate), and computational performance (de-

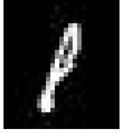
Input Example	"Core" Classifiers Linear	Polynomial	RBF	Assigned Pseudo-label
	✓	✓	✓	"easy"
	✗	✓	✓	"hard"
	✗	✗	✓	"harder"
	✗	✗	✗	"easy"

Table 4.2: Pseudo-label assignment example: given different inputs "8", we generate pseudo-labels depending on the correct predictions of the "core".

lay), depending on the properties of a dataset, one or another approach might be more appropriate. In addition to that, within the same dataset there exists variation in complexity. For example, a wide variety of classification approaches ranging from sparse linear classifiers to deep neural networks exist today, with each approach presenting a trade-off between classification accuracy and energy efficiency. Generally, the input dataset is treated as a "homogeneous" set of inputs $\mathcal{X}$ that require some classifier g in order to maximize the accuracy. If several such classifiers exist, the one with the lowest evaluation cost c is chosen. One way to optimize this approach with respect to the cost (energy-efficiency), one can treat the input set as non-homogeneous in terms of "hardness". In the current work we identify the "hardness" of an input depending on the "core" classifiers used. Consider an example of creating pseudo-labels shown in table 4.2. We used three "core" SVM classifiers with different kernels: linear, third degree polynomial, and RBF. From the table we see that given some inputs, if an input is classified correctly by all classifiers, it is labeled as "easy", while if it is only classified by more complex classifiers, we label the input as "harder" or

Table 4.3: Datasets used and the number of inputs misclassified during the training phase.

Dataset	Linear	Polynomial	RBF
Image Segmentation	168	20	0
ISOLET	543	69	4
Letter Recognition	239	34	2
MNIST	481	98	9
Penbase Recognition	109	14	0
Spam filter	39	8	0
Vowel Recognition	29	9	1

"hardest". In our work, because the main optimization goal is energy-efficiency, we label as "easy" all the inputs that cannot be classified by any classifier. This allows us to avoid expensive computation on the inputs that are incorrectly identified by the "core" classifiers anyway.

The "hardness" distribution varies between the datasets. Table 4.3 shows the datasets that we have considered and it also shows number of misclassified inputs of our "core" classifiers on the said datasets. Notice that the more complex classifiers tend to have less misclassified data inputs. Using one classifier for the whole dataset involves penalty either in energy efficiency or in accuracy[2].

To explain the penalty incurred, let us consider a synthetic example with distributed "hardness" when using an SVM with linear, polynomial, and RBF kernels, having 60% of all inputs as "easy", 30% as "harder", and 10% as the "hardest". If we use only the linear classifier, we will have no energy-efficiency penalty (as linear kernel is more energy efficient than both polynomial and RBF kernels), but we will take a hit in accuracy. The accuracy penalty can be calculated case-by-case: if the input is "easy", there is no accuracy penalty; if the input is "harder", the accuracy penalty is the potential accuracy gain that we missed multiplied by the fraction of the "harder" inputs, that is $(\text{Acc}_{\text{poly}} - \text{Acc}_{\text{linear}}) \times 0.3$. Similarly, if the input is "hardest", the penalty is $(\text{Acc}_{\text{RBF}} - \text{Acc}_{\text{linear}}) \times 0.1$. The total accuracy penalty of choosing

---

[2] The penalty is the loss of potential gain from other alternatives when one alternative is chosen, also known as "Opportunity Cost".

linear classifier is $0.1\text{Acc}_{\text{RBF}} + 0.3\text{Acc}_{\text{poly}} - 0.4\text{Acc}_{\text{linear}}$. Conversely, if only an RBF is used, all of the inputs will be identified using the most complex classifier, so no accuracy penalty will be incurred. The energy efficiency penalty in such a case could be expressed as $0.6\text{EE}_{\text{linear}} + 0.3\text{EE}_{\text{poly}} - 0.9\text{EE}_{\text{RBF}}$. Similar analysis can be performed if only the polynomial classifier is used.

The above example is known in the optimization community as “No Free Lunch” theorem (Wolpert and Macready, 1997, Gómez and Rojas, 2016), which states that “if the performance of one approach is superior to that of another approach, than the reverse must be true over the set of all other optimization problems”, which in simpler terms says “there is no free lunch in optimization”. However, although there is no way of super-optimizing for both accuracy and energy efficiency, in our work we focus on balancing the two for an optimal trade-off between the error rate and energy-efficiency. To achieve that we introduce a “chooser” function, which identifies the “hardness” of the problem and enables either high-accuracy or high-energy-efficiency algorithm.

4.2.2 Training the “chooser” for a fixed energy budget

During the offline training of the adaptive classifier, all the “core” classifiers are first trained to achieve the minimum error rate possible. After that pseudo-labels for the “chooser” are created depending on correct predictions of the “core” classifiers. Earlier in this section we have shown table 4.2 which shows an example for creating pseudo labels for a handwritten digit recognition. Once the pseud-labels are identified, the “chooser” is trained (offline) to classify the “hardness” of the inputs based on the “hardness” pseudo-labels. The “chooser” function is trained for a given error rate constraint, which allows the adaptive classifier system to dynamically trade off classification accuracy versus energy efficiency.

As shown in table 4.3, in any given dataset the number of “hardest” inputs is not

$$\mathcal{L}_{ee} = \begin{array}{cc} & \begin{array}{ccc} \text{lin} & \text{poly} & \text{RBF} \end{array} \\ \begin{array}{l} \text{Easy} \\ \text{Harder} \\ \text{Hardest} \end{array} & \begin{pmatrix} 0. & 0.6(\mathrm{EE}_{\mathrm{lin}} - \mathrm{EE}_{\mathrm{poly}}) & 0.6\mathrm{EE}_{\mathrm{lin}} + 0.3\mathrm{EE}_{\mathrm{poly}} - 0.9\mathrm{EE}_{\mathrm{RBF}} \\ 0. & 0. & 0.3(\mathrm{EE}_{\mathrm{poly}} - \mathrm{EE}_{\mathrm{RBF}}) \\ 0. & 0. & 0. \end{pmatrix} \end{array} \tag{4.1}$$

$$\mathcal{L}_{ac} = \begin{array}{cc} & \begin{array}{ccc} \text{lin} & \text{poly} & \text{RBF} \end{array} \\ \begin{array}{l} \text{Easy} \\ \text{Harder} \\ \text{Hardest} \end{array} & \begin{pmatrix} 0. & 0. & 0. \\ 0.3(\mathrm{Acc}_{\mathrm{poly}} - \mathrm{Acc}_{\mathrm{lin}}) & 0. & 0. \\ 0.1\mathrm{Acc}_{\mathrm{RBF}} + 0.3\mathrm{Acc}_{\mathrm{poly}} - \mathrm{Acc}_{\mathrm{lin}} & 0.3(\mathrm{Acc}_{\mathrm{RBF}} - \mathrm{Acc}_{\mathrm{poly}}) & 0. \end{pmatrix} \end{array} \tag{4.2}$$

the same as the number of “easy” inputs. That means, that the “chooser” function most of the time is trained on imbalanced data[3] biased towards one of the pseudo-labels. To avoid the situation where all of the inputs are treated as “easy” we use a loss matrix which is defined by the ratios of the number of “easy”, “harder”, and “hardest” inputs. To construct the loss matrix, we consider the penalty incurred by choosing one of the “core” classifiers, and the train the “chooser” to minimize that penalty. Considering the same example we used in the previous subsection, let us construct a loss matrix $L = \alpha_{ee}\mathcal{L}_{ee} + \alpha_{ac}\mathcal{L}_{ac}$, where $\mathcal{L}_{ee}$ and $\mathcal{L}_{ac}$ are energy efficiency and accuracy loss respectively, and α is a scaling coefficient. Given the “hardness” distributions shown earlier, we want to penalize the incorrect decision of the “chooser” function, and the penalty should be proportional to both the distribution of the classes (to avoid imbalanced classes) as well as the energy efficiency and accuracy loss. The equations (4.1) and (4.2) show the matrices constructed using the distributions between “easy”, “harder”, and “hardest” being 60%, 30%, and 10%. This example shows that misclassification in the “chooser” function will increase the total loss either in energy dissipation or in the accuracy, and the training of the “chooser” becomes a

---

[3] The term “imbalanced” in this context refers to the imbalanced classes – a situation when classes are not represented equally within the same dataset.

minimization problem.

To formalize the training of the "chooser" function, the behavior described earlier arises from the optimization problem when learning over training data. Given a set of n training data/label pairs, $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \{1, \ldots, C\}$, where $\mathcal{X}$ is the input space, and $\{1, \ldots, C\}$ is the collection of output labels, and given a collection of k classifiers $f_1, \ldots, f_k : \mathcal{X} \rightarrow \{1, \ldots, C\}$ with associated energy efficiency budget $c_1, \ldots, c_k \in \mathbb{R}$, the goal is to learn a "chooser" function $g : \mathcal{X} \rightarrow \{1, \ldots, k\}$ that maps each example to one of the k models in input space $\mathcal{X}$ such that the average error rate is minimized subject to energy efficiency budget constraint. This optimization problem can be formulated as:

$$\min_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \mathbb{1}_{f_j(x_i) \neq y_i} \mathbb{1}_{g(x_i)=j}, \tag{4.3}$$

$$\text{Subject to: } \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} c_j \mathbb{1}_{g(x_i)=j} \leq \beta$$

where β is the chosen average EDP budget and $\mathbb{1}_z$ is the indicator function, with a value of 1 when the logical expression z is true and a value of 0 when the logical expression z is false. By representing the constraint as a Lagrange multiplier, the problem can be represented as an unconstrained optimization problem:

$$\min_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} L_j(x_i, y_i) \mathbb{1}_{g(x_i)=j}, \tag{4.4}$$

where $L_j(x_i, y_i) = \mathbb{1}_{f_j(x_i) \neq y_i} + \lambda c_j$ is defined as the loss associated with using the classification function f_j on example x_i, incorporating both the classification error of the prediction function as well as the EDP modulated by the Lagrangian multiplier λ (in practice, the value λ is swept over in order to find a feasible system that satisfies the budget constraint[4]). This problem is equivalent to the well-studied cost-sensitive

---

[4] Note that we derive this problem for the EDP/energy/power constraint, however we could pose

learning problem, allowing for existing learning techniques to be applied.

In practice, we upper bound the objective as an importance weighted supervised learning problem of the form

$$\min_{g \in \mathcal{G}} \sum_{i=1}^{n} W_i \mathbb{1}_{g(x_i) \neq l_i}, \tag{4.5}$$

where the pseudo-label l_i and importance weight W_i are defined as

$$\begin{aligned} W_i &= \max_{p,q \in \{1,\ldots,k\}} L_p(x_i, y_i) - L_q(x_i, y_i) \\ l_i &= \min_{p \in \{1,\ldots,k\}} L_p(x_i, y_i). \end{aligned} \tag{4.6}$$

This allows for efficient training and implementation using standard supervised learning techniques such as logistic regression and support vector machines.

The pseudo-label l_i and importance weight W_i define if an example is "easy", "harder", "hardest", etc. We consider an example to be "harder" or "hardest" if the pseudo-label l_i (representing the optimal classification system to route example i) points to a complex, energy-inefficient classifier. Conversely, we consider an example to be "easy" if the pseudo-label l_i points to an energy-efficient classifier. Note that for examples with no variation in error rate across classifiers, the importance weight W_i is generally small, as a difference in losses simply represents the difference in energy efficiency between classifiers. The over-fitting of the classifiers (including the chooser) should be addressed during the training period, however, no special steps are required to avoid such problems, as the "chooser" function is not prone to over-fitting due to the fact that it is expected to be much simpler than the most complex classifier in the ensemble. For example in our approach, the "chooser" function is a linear classifier,

---

the problem as a minimum energy dissipation system given an average error constraint. Introducing a Lagrange multiplier for the average error constraint yields an identical optimization problem (modulated by a constant) as (4.4), and therefore solving the EDP constrained problem is equivalent to solving the error constrained problem.

while the classifier for the "hardest" examples is an RBF kernel function.

4.2.3 Training the "chooser" for a changing energy budget

From equations (4.3) and (4.4) we notice, that the training of the "chooser" function is performed with minimum EDP in mind. However, if such a constraint is very tight, the average accuracy of the adaptive classifier system might suffer. To mitigate such a problem, we propose using a system where the adaptive classifier is aware of the current energy reserves, and would change its behavior accordingly. In that case, the equation (4.3) changes its form as follows

$$\min_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \mathbb{1}_{f_j(x_i) \neq y_i} \mathbb{1}_{g(x_i, \beta)=j}, \tag{4.7}$$
$$\text{Subject to: } \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} c_j \mathbb{1}_{g(x_i, \beta)=j} \leq \beta, \forall \beta \in \mathcal{B}$$

where $\mathcal{B}$ is a set of energy budgets that in which the adaptive classifier is expected to operate. In this case, the current energy budget β is an input to the "chooser" function as a feature. The equation (4.7) could be rewritten as an unconstrained minimization problem:

$$\min_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} L_{j,b}\left(x_i, y_i\right) \mathbb{1}_{g(x_i, \beta_b)=j}, \tag{4.8}$$

where the loss is defined as $L_{j,b}\left(x_i, y_i\right) = \mathbb{1}_{f_j(x_i) \neq y_i} + \lambda(c_j - \beta_b)$.

This change in the training assumes that the adaptive classifier has information about the current energy reserves[5], which allows the adaptive classifier to change its behavior in order to meet the imposed constraints. To visualize how this approach differs from the one described in subsection 4.2.2, refer to figure 4·1. In this figure, (a) shows the training approach that takes changing energy budgets into considerations,

---

[5] The overhead of acquiring the current energy budget is not considered in the current work. Refer to section 4.5 for discussion on this matter.

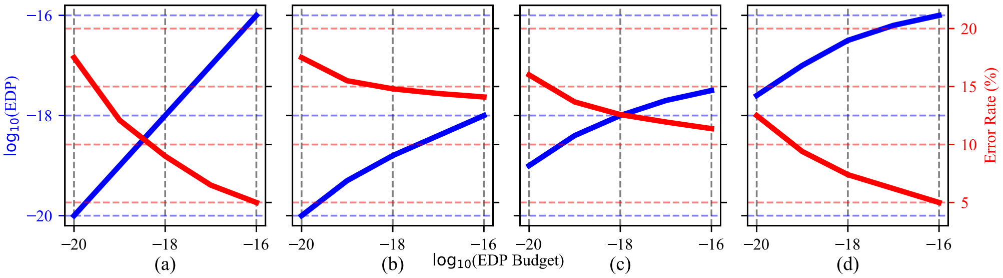


Figure 4·1: EDP and Error rate vs. the budget of an adaptive classifier: (a) trained with EDP budget as a feature; (b,c,d) trained to maximize the accuracy while meeting an energy constraint of 10^{-20}, 10^{-18}, and 10^{-16} respectively.

while (b, c, d) are trained to maximize the accuracy, while being constrained by the EDP of 10^{-20}, 10^{-18}, and 10^{-16} respectively. Note that subfigures (b, c, d) meet their respective EDP constraints, however, when the energy budget changes, they become suboptimal: if the adaptive classifier is trained only for 10^{-20} EDP constraint (figure 4·1(b)), the EDP is the lowest across all EDP budgets, but the error rate is higher on average. The error rate is lower for training for 10^{-18}, and 10^{-16} EDP (figures 4·1(c, d)), but the average EDP of the classifier is very high. At the same time, figure 4·1(a) shows that the modified training routine manages to keep the average energy dissipation within the imposed constraints, while maintaining high accuracy.

The systems proposed in this section present many beneficial properties rarely found in energy-efficient systems. By treating each classification model as a “black box” as opposed to a known, modifiable object, existing energy-efficient classification approaches can be directly used when constructing the system. Furthermore, multiple complex classifiers can be easily integrated into the system by providing the capability to upload the data to a server to apply an extremely complex classifier. Due to the modularity of our design, the system is even able to integrate humans into the loop for cases where humans have lower error rates than machines in classifying objects.[6]

Additionally, changing the energy constraint of the proposed system does not require that the classifiers be retrained. This is particularly valuable for mobile systems, where adaptation is required to account for the difference in usage patterns of users, and energy budgets need to be updated due to changes in battery usage settings (Lane et al., 2010). Rather than changing the classifiers, the system only needs to re-train the chooser function g to adapt to a new budget constraint. Furthermore, the usage of a convex supervised approach such as logistic regression or a Support Vector Machine (SVM) to train the chooser function g allows for streaming online updates to be applied during runtime using stochastic gradient updates.

4.3 Adaptive Classifier – Microarchitecture Description

In this section, we present a detailed description of our adaptive classifier approach. We present the theory and the implementation details of three different classifiers and a chooser function that chooses one of the three classifiers based on the "hardness" of the data.

Microarchitecture

Figure 4·2 shows the microarchitecture of our proposed adaptive classifier system. It consists of the "Chooser" function (marked as (?)), and several "core" classifiers with various complexities. In our case, the "chooser" function uses a multi-class logistic regression classifier. Once trained the "chooser" function serves as a MUX/DEMUX controller which selects an appropriate classifier. In the current work the "chooser" function is implemented as a logistic regression based multi-class one-vs-all classifier

---

[6] In this case, a second budget constraint may exist regarding the number of times a human is queried. Due to lack of space, we do not discuss this model here, however the exact same reduction to a cost-sensitive learning problem can be made by incorporating a second Lagrange multiplier for this new constraint.

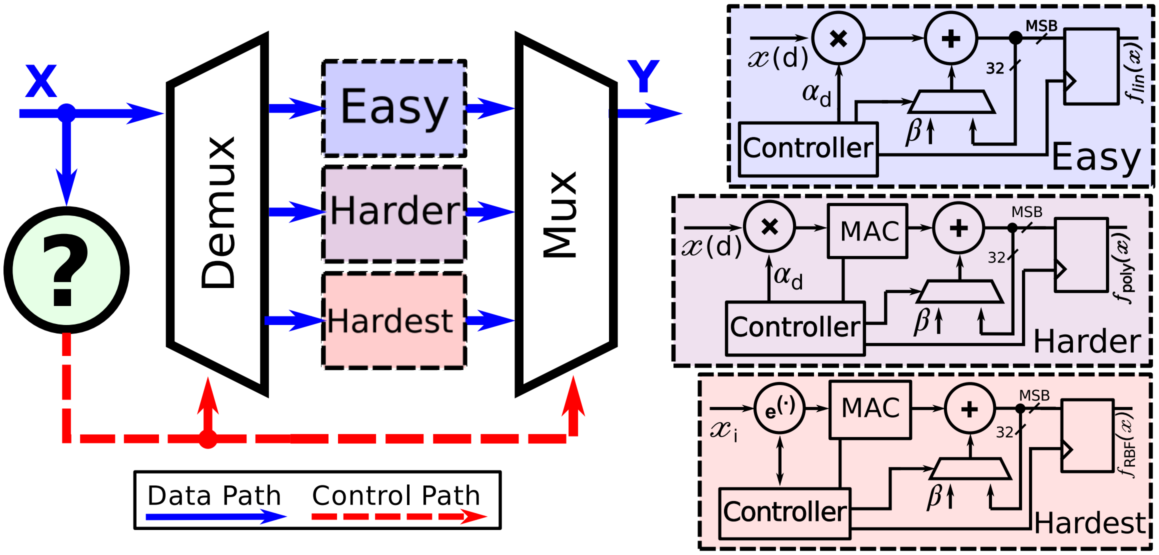


Figure 4·2: Block diagram of the adaptive classifier. "Chooser" function is marked as $\textcircled{?}$, and its microarchitecture is the same as the "Easy" block. Data paths are shown as solid lines, while control path is dashed. The linear classifier is implemented as a multiply-accumulate (MAC) block and is equivalent to equation (4.9). The polynomial function implementation uses a MAC to compute higher order components $\alpha^p_{(\cdot)} x(\cdot)^p$ as per equation (4.10). The RBF classifier also uses and MAC and its the exponential function is implemented as a look up table.

with three labels. Note that the adaptive classifier design proposed in this paper doesn't include the training hardware, and all the classifiers, including the "chooser" are trained offline.

During the offline training of the system, all the "core" classifiers are trained to achieve the minimum error rate possible. The "chooser" is then trained (offline) to classify the "hardness" of the inputs by analyzing the results of the "core" classifiers using the procedures described in sections 4.2.2 and 4.2.3. The "easy", "hard" and "hardest" class labels identified by the "chooser" function are used to route the input to the "core" classifier of appropriate complexity. The "chooser" function is trained for a given error rate constraint, which allows the adaptive classifier system to dynamically trade off classification accuracy versus energy efficiency. The computations are performed using fixed point, 2's complement representations, thus only the most significant bit (MSB) is required to identify the label of a binary class.

Below, we describe the functional form of three "core" classification functions – linear function, polynomial function, and RBF kernels, which we used in our evaluation. The binary functional forms are presented, with multi-class prediction accomplished using a one-versus-all max-margin coding to convert from multi-class to binary classification.

We consider linear classifiers of the form

$$f_{lin}(x) = \text{sign}\left(\sum_{d=1}^{D} \alpha_d x(d) + \beta\right), \tag{4.9}$$

where D is the dimensionality of the data and $x(d)$ corresponds to the value of the d^{th}-dimensional element of example x. The classifier is parametrized by the weights $\alpha_1, \ldots, \alpha_D$ and the offset β, with these parameters learned using training data. The "easy" inputs are routed to this low complexity classifier. This classifier is of similar complexity as the "chooser" function, and is the simplest of the "core" functions. It is implemented as a multiply-accumulate block as shown in Figure 4·2(Easy). Depending on the design topology chosen after design space exploration, the MAC block is parallelized accordingly. It has the highest energy efficiency, but it also has a high error rate.

Similarly, we utilize homogeneous polynomial classifiers of the form

$$f_{poly}(x) = \text{sign}\left(\sum_{d=1}^{D} \sum_{p=1}^{P} \alpha_d^p x(d)^p + \beta\right), \tag{4.10}$$

where P is the power of the polynomial classifier. As in the linear case, the classifier is parametrized by the weight parameters $\alpha_1^1, \ldots, \alpha_D^P$ and the offset β. The "harder" input is routed to this higher complexity classifier block. This block is implemented as a 5^{th} order polynomial evaluation function. The "harder" block reuses the multiply-accumulate block from the "easy" block. This classifier provides a moderate compromise between error rate, energy efficiency and execution time. The polynomial is computed iteratively in five clock cycles using a multiply-and-accumulate (MAC) block (see Figure 4·2(Harder)). The MAC block is used to compute the higher order components $\alpha_{\cdot}^{p} x(\cdot)^{p}$ by feeding the multiplier output as the input.

Finally, we consider radial basis function kernel classifiers of the form

$$f_{rbf}(x) = \operatorname{sign}\left(\sum_{i=1}^{n} \gamma_i e^{\frac{-\|x-x_i\|_2^2}{\sigma}}\right), \tag{4.11}$$

where σ is a user-specified kernel parameter and the points $x_1, \ldots, x_n$ are the training data points. The classifier is parametrized by the kernel weights $\gamma_1, \ldots, \gamma_n$. The "hardest" input is routed to this highest complexity classifier block. Because RBF includes an exponential function, this block is the most complex compared to the previous two. However, an SRAM is used as a look-up table (LUT) in order to avoid lengthy computations of the exponential function. The RBF function computation partially reuses the blocks in the other two classifiers in order to reduce area and increase hardware utilization. The block diagram shown in figure 4·2(Hardest) includes an exponential function that is implemented as a look-up table. The MAC block than computes $\gamma_i e^{(\cdot)}$.

4.4 Evaluation

In this section we first discuss the evaluation setup and the evaluation of our proposed adaptive classifier. Later in this section we will describe the modified training approach results, as well as the results on large datasets.

	Adaptive Effort Classifier System						Conventional Classifiers					
	High Error Rate		Low Error Rate		Average		Linear		Polynomial		Radial-Basis	
Data Set	Energy	Delay	Energy	Delay	Energy	Delay	Energy	Delay	Energy	Delay	Energy	Delay
Synthetic	52.6e-15	0.1	29e-12	0.7	903.9e-15	0.2	15.8e-15	0.03	92.3e-15	0.3	27e-12	0.4
Image Seg.	4.9e-12	1.0	2.7e-12	4.9	9.3e-12	1.4	5.4e-12	1	79.8e-12	2.6	2.6e-12	4.2
ISOLET	4.5e-9	34.6	2.9e-9	153.4	130.0e-9	69.1	5.9e-9	35.1	79.4e-9	71.4	2.7e-9	142.6
Letter Rec.	4.1e-12	1.0	2.3e-12	4.3	29.7e-12	1.1	4.4e-12	0.8	6.5e-12	2.5	1.9e-12	3.7
MNIST	8.6e-9	43.3	3.9e-9	184.1	186.4e-9	94.8	10.5e-9	44.5	14e-9	86.8	3.93e-9	181.3
Penbase	3.3e-12	0.7	2.0e-12	3.7	21.5e-12	1.5	4e-12	0.9	5.3e-12	1.6	1.8e-12	3.7
Spam filter	50.1e-12	2.9	21.6e-12	14.5	1.1e-12	4.3	53.1e-12	3.2	67.6e-12	5.4	20.5e-12	13.
Vowel Rec.	1.4e-12	0.5	73.2e-11	2.8	15.9e-12	1.2	1.7e-12	0.5	2.4e-12	1.3	0.7e-12	2.3

Table 4.4: Energy and Delay for the adaptive classifier and conventional classifiers for different data sets. Each row corresponds to subplots in Figure 4·5. Energy results are shown in J/classification and delay is represented as μs.

	Current Work		SEC	
	Time	EDP	Time	EDP
Synthetic	1.67e-04	1.81e-19	1.72e-04	3.86e-19
Image Segm.	2.60e-03	1.30e-17	3.22e-03	3.99e-17
ISOLET	2.51e-02	8.98e-12	2.50e-02	1.78e-11
Letter Rec.	4.86e-03	3.27e-17	5.42e-03	8.13e-17
MNIST w/ bkgnd	3.69e-02	1.77e-11	3.67e-02	3.48e-11
Penbase Rec.	5.15e-03	3.23e-17	5.57e-03	7.54e-17
Spam	2.83e-03	4.73e-15	3.39e-03	1.36e-14
Vowel Rec.	4.03e-03	1.91e-17	4.05e-03	3.85e-17

Table 4.5: Comparison of the proposed and SEC approaches (Venkataramani et al., 2015a) trained for low energy subject to error rate budget. The time is in seconds.

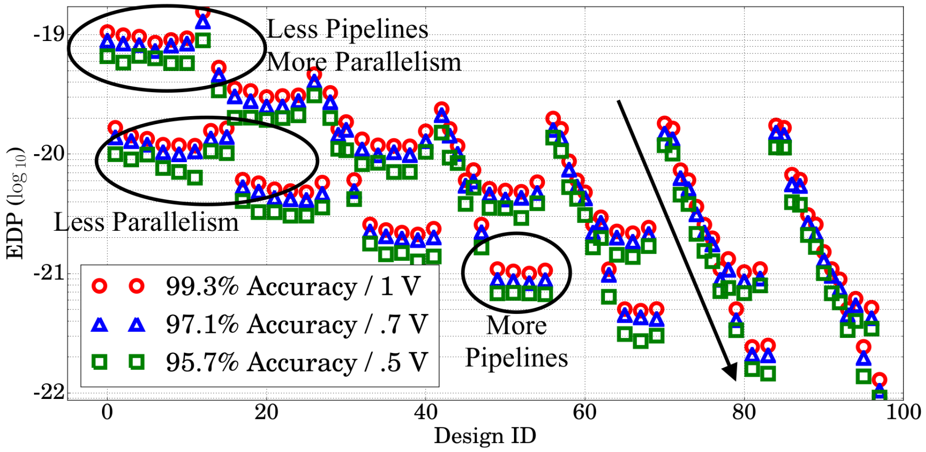


Figure 4·3: Energy delay products for different adaptive classifier designs generated by Aladdin (Shao et al., 2014).

4.4.1 Experimental Setup

Before exploring the hardware design space of our adaptive classifier, the training of the adaptive classifier system was performed offline using Python for several different constraints scenarios[7], where we set the maximum acceptable error rate independent of the application, but we assume linear decrease in energy budget over a period of time. Our goal was to minimize the average energy dissipation for a given error

---

[7] Scenarios such as high accuracy, high energy-efficiency, or linear switch from high accuracy to high energy-efficiency.

rate. In software models[8], the minimum EDP point is equivalent to the use of the simplest[9] model possible for the given constraints. The "chooser" function was trained to achieve minimum energy dissipation by optimizing the "core" classifiers utilization for a given error rate, i.e. if two classifiers with different energy dissipations have comparable error rates, the "chooser" function would be biased towards the simpler classifier. The data sets were separated into training and evaluation sets using an 80/20 scheme, where 80% of the inputs were used for training, and 20% for evaluation. All figures of merit in the evaluation section were acquired using the "evaluation" data sets, thus guaranteeing that inputs were never seen before by the systems under test. The main figures of merit that were considered for evaluation were computational performance (defined as average computation time per input), energy dissipation, and statistical performance, defined as misclassification error rate.

The training phase of the adaptive classifier could be performed in two different ways: 1) training for static energy budget (Takhirov et al., 2016); 2) training for energy-budgets that change over time. The first approach assumes that the energy budgets don't change over time while the second training approach assumes that energy budgets change over time, thus the behavior of the "chooser" should be more dynamic. As part of the current evaluation we assumed that the second approach was performed in an environment with uniform and almost linear battery discharge rate (thus decreasing the available energy budget). This means that the system had to be trained without being biased to any particular classifier.

After the training, in order to explore and narrow down the hardware design space of our adaptive classifier, we used the Aladdin toolset (Shao et al., 2014). The

---

[8] We used sklearn (Pedregosa et al., 2011) models as well as custom designed Tensorflow (Abadi et al., 2015) models

[9] Simplest model means the one that has the smallest number of computations per example. For example, if a linear and a polynomial function give the same error rate for a given input, the linear function is considered to be the simplest.

different design choices that we considered include level of MAC parallelization, size of SRAM for lookup table, level of computational parallelism, and the number of pipeline stages. Figure 4·3 shows the results obtained using the Aladdin tool for one of the input data sets (`Letter Recognition`). Here, all the EDP points are computed for the maximum statistical classifier accuracy and are acquired by varying the size of the computational units, memory size, level of computational parallelism, and number of pipeline stages across the different designs. The scatter plot shows how the various design choices affect the EDP of the system. The figure also shows the EDP at different voltages and the corresponding accuracy levels. The goal for the current work was minimizing the EDP for the given error rate constraint. Hence we selected the designs with the minimum EDP.

The chosen design was then implemented using Chisel (Bachrach et al., 2012). This framework was chosen, as it generates both hardware description code (Verilog) as well as C++ functional model. That allows us to perform a rapid verification of the design. The Verilog HDL code was then synthesized and placed-and-routed using 40nm Global Foundries technology with Synopsys standard cell libraries. The delay and energy dissipation were acquired from placed-and-routed, RC-extracted designs using Cadence toolset. We used seven different data sets from the UCI library (UC Irvine Machine Leanring Repository, 2017) and a synthetic data set for our evaluation. The list of the used data sets is shown on Table 4.4. In addition to the adaptive system, individual classifiers were implemented for comparison and validation: SVM (linear, polynomial, RBF kernels), RF, and CNN. The training of the system was performed offline using MATLAB for several different constraints scenarios, where the maximum acceptable error rate was set, while the minimum energy dissipation point was the goal.

Choice of the "Chooser" function

Although any type of classifier can serve as a "chooser" function, in this work we use a linear regression-based SVM classifier as a "chooser" (see equation 4.9). This choice was motivated by the tight EDP requirements for the "chooser", as well as minor average error drop when a more complex classifier is used as a "chooser" function. During the analysis, we considered three different classifiers: logistic regression-based SVM (linear classifier), third order polynomial SVM, and a fifth order polynomial SVM classifiers. Figure 4·4 shows the average EDP and error rate overheads when using different types of "chooser" functions. The EDP overhead is the EDP contribution of the "chooser" relative to the total EDP of the adaptive classifier. The error rate overhead is the average increase in error rate of the adaptive classifier with the specified "chooser" function as compared to the error rate of an ideal "chooser"[10]. As we can see from the figure, the more complex "chooser" functions don't contribute much in terms of error rate, while the EDP overhead is quite high. Third-order and fifth-order polynomials introduce 13% and 26% higher EDP than the linear classifier, while the error rate contributions are 0.5% and 1% lower respectively.

4.4.2 Experimental Results of an approach with static energy budget

Results presented in this subsection are for the adaptive classifier using a naïve training approach[11]. Figure 4·5 shows the Error Rate vs. EDP of the proposed approach and that of the linear, polynomial, and RBF classifier when used individually. We can see from the figure that depending on the input data set the proposed approach behaves differently. Adaptive classifier system performs best when the input data set has uniformly distributed "hardness", meaning the input has an equal mix of "easy",

---

[10] Ideal "chooser" function is a function that has 100% accuracy in identifying the "hardness" of an input. Although such classifier does not exist in practice, it is useful to acquire comparison metrics for different "chooser" functions.

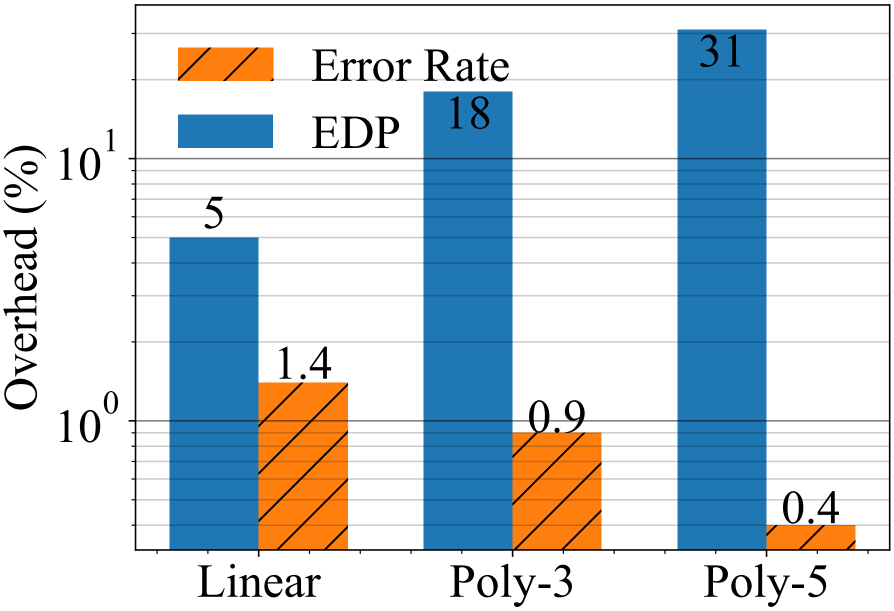


Figure 4·4: Average overheads of different types of "chooser" functions. The EDP overhead is average EDP contribution of the "chooser" with respect to an adaptive classifier where this "chooser" is implemented. The error rate overhead is the error rate contributed by the "'chooser" function itself.

"harder" and "hardest" examples. If the input data set is mostly "easy" or mostly "hard", adaptive classifier would waste energy because the overhead introduced by the "chooser" would not justify a limited number of switchings between different classifiers required. As seen from the data plots, linear classifier cannot achieve low error rates, while RBF, although showing extremely low error rates, has high EDP. For example, `Letter recognition` data set shown in Figure 4·5 shows that linear classifier cannot achieve an error rate below 65%, while RBF, which has error rate close to 0%, has very high delay and energy dissipation.

Unlike the conventional classifiers, the adaptive classifier system enables us to trade-off error rate and EDP. For example, in case of the `Letter Recognition` data

---

[11] The approach is considered to be "naïve" because it naïvely assumes static, never-changing energy budgets.

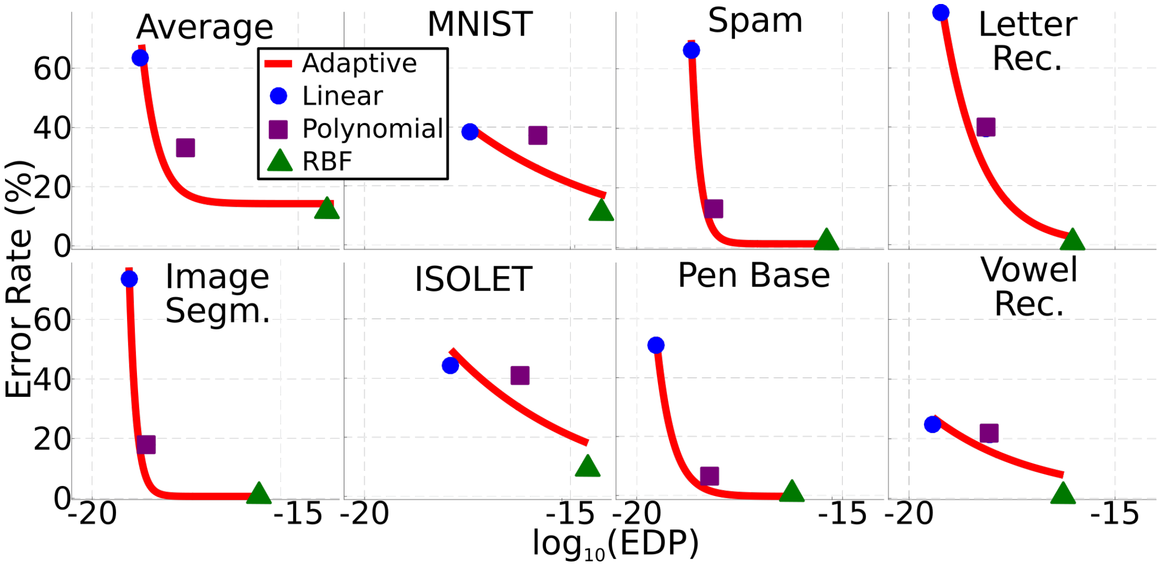


Figure 4·5: Error rate vs. Energy Delay Product (EDP) for adaptive classifier and conventional classifiers. Adaptive classifier can be tuned to achieve any of the intermediate error rate values.

set, if the EDP constraint falls between the EDP of the linear and polynomial classifiers, then in the conventional approach one has to choose the linear classifier which results in a high error rate. However, if one were to use an adaptive classifier, then the "chooser" function could be tuned to the EDP constraint and an error rate lower than that of a linear classifier can be achieved. This allows for dynamic selection of polynomial or linear classifier when lower error rates are required. Similar approach can be adopted for EDP constraints lying between polynomial and RBF classifier for the `Letter Recognition` data set as well as for other data sets to achieve lower error rates compared to conventional classifiers. Figure 4·6 shows one of the possible usage ratio of different classifiers in our adaptive classifier which shows one of the possible energy dissipation distribution of different classifiers in our adaptive classifiers. These plots are for a training for minimum EDP scheme, while the "chooser" function was swept for different error rates. From the plots we can see that for lower error rate requirements, the system becomes biased towards RBF and polynomial, while for cases where a higher error rate is acceptable the system chooses to use linear and polynomial classifiers. Note that the utilization correspondingly affects the total energy dissipation of the adaptive system, thus a dynamic trade-off between error rate vs. EDP is achievable.

Table 4.4 shows the energy dissipation and delay values of the adaptive classifier when running the various data sets. The delays represent the minimum achievable delay without change in the achieved error rate. “High error rate” represents the training scheme where we do not constrain the error rate (due to which the linear classifier is always preferred) resulting in lowest EDP. On the other hand, the “Low error rate” column shows the training scenario where the target error rate budget is set to be low (the target error rate is set at the error rate of the RBF in this example) resulting in a higher EDP. On average the proposed adaptive classifier approach consumes 10x more energy as compared to the linear classifier and 100x less energy than RBF. The average delay of the adaptive system across all examples is ≈2x of the linear approach, and ≈0.33x times of that of RBF. At the same time the error rate is on average 0.5% higher than RBF, but ≈ 40% lower than linear.

From Table 4.4 we can see that the “High Error Rate” and “Low Error Rate” results (both energy and delay) are comparable to linear and RBF classifiers, respectively. This behavior is expected, because if the training is biased towards one of the error rate extremes, the adaptive classifier tends to choose only linear or only RBF. This is also confirmed by the energy distribution plots shown on Figure 4·6. This figure shows the total energy consumed by the adaptive classifier system at different error budgets as well as the distribution of energy consumption in the different parts of the adaptive classifier. Note that the “chooser” function tends to use linear classifier more often when a high error rate is acceptable, while being biased towards using RBF when the error rate constraints are tight. That means that energy dissipation

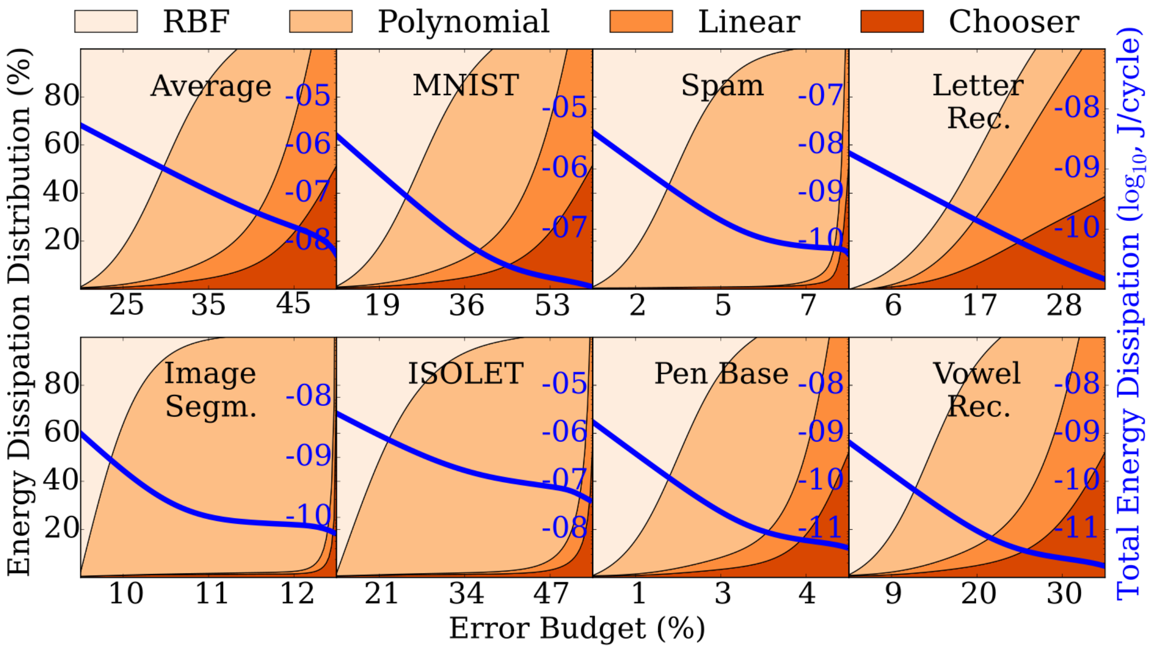


Figure 4·6: Energy dissipation distribution and the total energy dissipation vs. error rate for different classification problems.

of the linear classifier dominates at high error rate budgets.

Figure 4·7 shows the area of post-placement-and-routing design of the adaptive classifier. The bars show the relative areas of different blocks, while the solid line shows the total area. The results include the SRAM LUT as part of the RBF block. This LUT occupies 43900 μm^2 and is used to evaluate the exponentiation function. Note that the area distributions are different across different data sets as we generated a unique classifier design for every data set separately.

4.4.3 Experimental Results of the approach with dynamic energy budget

The training routine described in section 4.2, is significantly different from the work presented in the (Takhirov et al., 2016). The main difference is that by changing the training routine and the microarchitecture of the "chooser" function, we have managed to reduce the average energy-dissipation by up to 12% and improve the

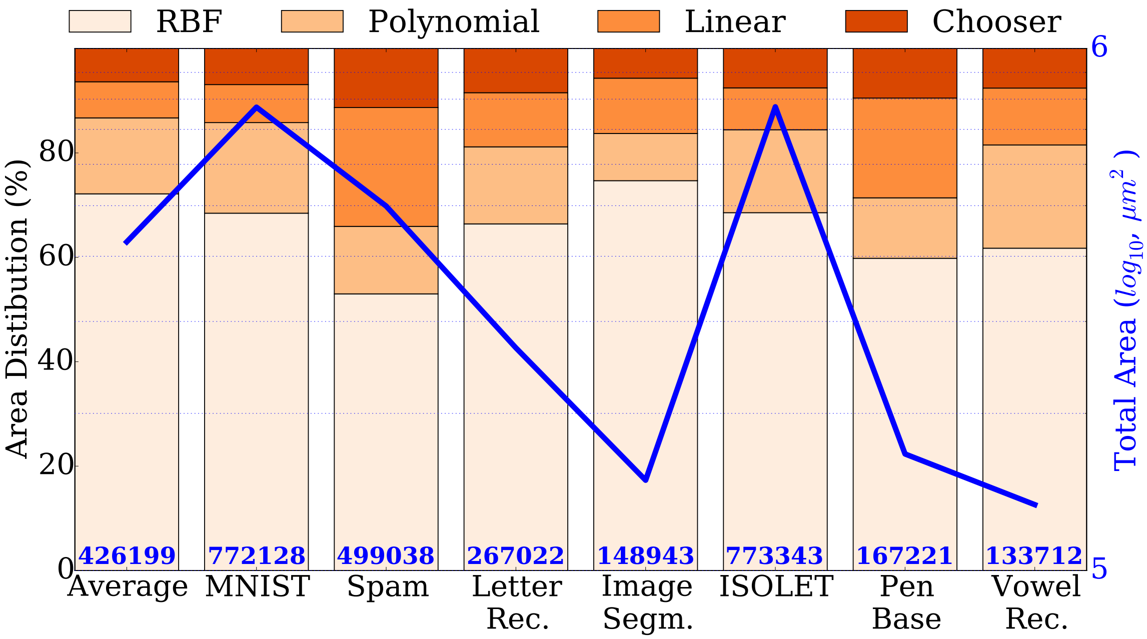


Figure 4·7: Relative area distribution of subblocks in adaptive classifier (bars) as well as the total area (solid line).

computational performance by 7% as compared to (Takhirov et al., 2016) (see figure 4·8). At the same time, the statistical performance is left almost unchanged (within 0.5% drop in accuracy on average). This effect is caused by the the "chooser" awareness of the change in energy budgets. That way it can utilize the computational resources more efficiently: in the static training approach, if the energy budgets are very low, the "chooser" sends all of the examples to the less complex "core" classifiers, which drastically reduces the accuracy. The main goal of the "chooser" in this case is to meet the energy constraints no matter what accuracy is. In the dynamic tuning approach, the "chooser" takes into account the impact of individual "core" classifiers on the average energy dissipation, and sends only a subset of the inputs to the simpler classifiers as long as the energy constraints are met. The main goal of the "chooser in this case is to maximize the accuracy while meeting the energy constraints.

To evaluate the approach with training on dynamic energy budgets, we trained

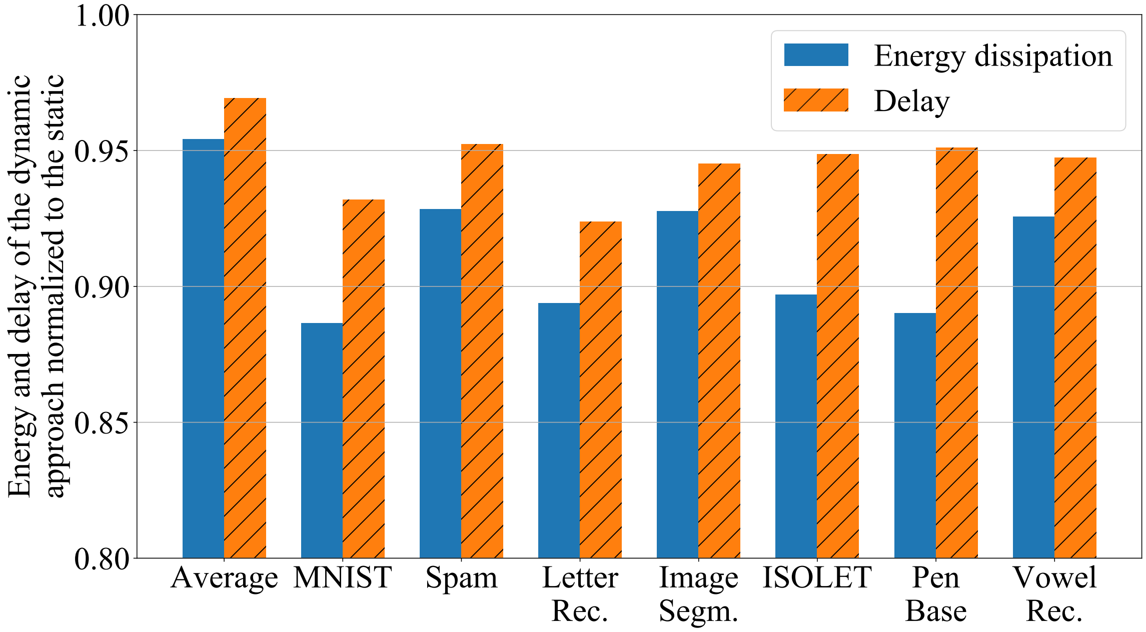


Figure 4·8: Energy dissipation and accuracy for dynamic adaptive classifier normalized to the adaptive classifier trained for the static energy budget.

the adaptive classifier, and assumed that the energy budgets change over time. The adaptive classifier takes an immediate energy budget constraint as a feature, and adjusts its behavior in accordance to the training routine described in by equations (4.7) and (4.8). Direct behavior comparison between different training routines is described by figure 4·1. In figure 4·1(a) we show the behavior of an adaptive classifier that was trained with `Penbase Recognition` dataset and assuming three different energy dissipation budgets: 10e-12, 1e-9, and 0.1e-6 J/classification with the delay fixed at 1GHz, which translates to EDP constraints of 1e-20, 1e-18, and 1e-16 respectively. Notice on subfigure (a), which corresponds to the dynamic training approach, the EDP constraint is met across all budgets, and the error rate curve shows a smooth trade-off between low and high accuracy modes. On subfigure (b), the adaptive classifier was trained for very tight energy budget, which prevents the classifier to improve on accuracy when the energy budget constraints are loosened. Conversely, on subfigure (d) we see that although the accuracy is the lowest, the classifier fails to meet the energy (or EDP) constraints. Figure 4·1(c) shows the behavior which is a combination of the previous two: it fails to meet the energy constraints when budgets are tight, and fails to reduce the error rate when the budgets are loosened.

Figure 4·8 shows the experimental results of the dynamic approach relative to the approach is to assume that the energy budgets are static. We see that in the best case we reduce the energy dissipation by $> 11\%$ and reduce the delay by $> 6\%$ (`MNIST`). In the worst case, dynamic approach reduces the energy dissipation and delay by $\approx 4.5\%$ and $\approx 3\%$ respectively (`Average`). The dynamic approach has insignificant area impact on the design, which varies from 0.6% (`Average`) to 1% (`Spam`) overhead as compared to the static approach. The variation on the area depends on the number of features of the dataset that the classifier was designed to operate for.

4.5 Discussion

Although in the current work we provide implementation details of a classifier, in practice any machine learning problem could be handled in a similar fashion. Adaptive classifier allows replacing the "core" classifiers with any machine learning estimator in addition to classification: clustering, regression, and even dimensionality reduction. For example, an adaptive clustering algorithm could have a "chooser" classifier that identifies the "hardness" of the current input, and the "core" clustering algorithms of different complexity would be used to achieve energy-efficient behavior.

In addition to that, adaptive classifier is well suited to work with remote or "cloud" computing in mind. The "chooser" function can be trained to identify if a problem is too expensive to compute locally, such that it would be computed remotely. This idea has some challenges in itself: it requires more research in the area of computation vs. communication energy-delay trade offs. The problem is that offloading large amount of data to the “cloud” might be expensive both in terms of energy and in terms of delay, and such an approach would require careful consideration.

As mentioned before, the approach that takes into account change in the current energy-budgets requires information on the current energy reserves. Although not a part of the current work, we would like to point out that acquiring the information about the energy reserves might have some energy penalty as well. For example, the energy/delay cost of acquiring current battery reserves could be very high. In such a case, the host CPU could retrieve and buffer the current energy budget infrequently in order to avoid a drop in EDP.

Both static and dynamic adaptive classifiers also have one major problem: the “hardness” of the input data is usually very unbalanced, with “easy” classes dominating, and the “hardest” classes being only a small fraction of all the inputs. That makes the training of the “chooser” prone to having “false” high accuracy, where every input is identified as “easy” because the absolute majority of the inputs were “easy”. There are several different ways to avoid such problem. One of the solutions is constructing loss matrices described by example equations (4.1) and (4.2). However, this approach requires a detailed analysis of the input datasets. Another way to resolve this issue is to change the loss metric from accuracy to F_1 score, which is a harmonic mean of precision and recall. This solution is not straightforward, and also requires detailed prior knowledge of the input dataset.

4.6 Conclusion

As part of our evaluation, we have performed a detailed comparison of our approach to the state-of-the-art scalable effort classifier (SEC) approach described in (Venkataramani et al., 2015a) (see Table 4.5). To make the comparison as fair as possible, SEC system was designed using the same process as the proposed approach. We used our linear, polynomial, and RBF classifiers to implement the three stages of the SEC, and the SEC design space exploration was performed using Aladdin tool. However, the energy results for the SEC approach were from just synthesized designs, while the adaptive classifier approach energy results were using netlists extracted from placed-and-routed designs. The “Time” column shown in Table 4.5 shows the results of C++ simulations. Results show that the execution time of for both approaches is comparable. However, the adaptive classifiers can achieve almost 2× lower EDP while operating at the same error rate. The reason for this is that when facing “harder” problems, the SEC approach still utilizes “easy” stages of the classification system before switching to more complex classifiers, thus wasting energy. Adaptive classifier approach chooses only one (appropriate) classifier depending on the “hardness”.

Chapter 5

Random Forest and the Field of Groves

5.1 Introduction

Over the last couple of decades the consumer market has gradually moved towards mobile computing. According to the comScore report, people of age 18-64 spend ≈70% of their digital time using a mobile device (ComScore Report 2016, 2016). Mobile workloads are increasingly data intensive, and hence machine learning (ML) algorithms are commonly used in these applications (Lane et al., 2010). The data-intensive nature of these applications does not allow us to run these applications purely on our mobile systems. For example, applications like speech recognition, although used quite often, are still evaluated remotely because limited energy budgets prohibit running a powerful ML algorithm on a mobile device.

Mobile systems are energy constrained, and hence while designing machine learning algorithms for mobile applications we need to manage two conflicting requirements – high accuracy and low energy dissipation. Over the past few years several architecture-level, circuit-level and algorithm-level optimizations have been proposed for improving the power, performance, area, and accuracy of machine learning accelerator designs (Du et al., 2015, Takhirov et al., 2016, Kusner et al., 2014, Gao and Koller, 2011, Park et al., 2012, Xu et al., 2012, Trapeznikov and Saligrama, 2013, Wang et al., 2015, Kusner et al., 2014, Nan et al., 2015, Al Maashri et al., 2012, Albericio et al., 2016, Han et al., 2016, Chen et al., 2016b, Shafiee et al., 2016, Chi et al., 2016, Rahman et al., 2016).

The machine learning community has shown that we do not always require complex classifiers such as convolutional neural networks (CNN) or kernel support vector machines (SVM) for classifying data, and that low-complexity classifiers such as random forests (RFs) are an adequate substitute for applications where high accuracy with low energy dissipation is required (Nan et al., 2015). In this paper, we propose an alternative implementation of RF classifiers. We divide the RF into groups of trees called groves for budget-constrained environments, where the budget is accuracy, energy, delay or energy-delay product. In general, RFs are a collection of decision trees (DTs) that independently predict the classification result, with the final decision made by combining the decisions of individual trees. Our approach uses the confidence of groves within the RF about their decision to optimize the resource utilization. In this work:

– We first evaluate the use of RF algorithm as an alternative to CNN, SVM with linear (SVM_{LR}) and with radial-basis function (SVM_{RBF}) as the kernels, and Multi-layer Perceptron (MLP) algorithms. Our analysis shows that the RF accuracy is comparable to the accuracy of CNN, SVM_{LR}, SVM_{RBF} and MLP for all evaluated datasets, and on average RF consumes ≈10× lower energy per classification.

– We also propose a novel implementation of RF called Field of Groves (FoG). FoG is composed of multiple groves, where every grove is a subset of decision trees. During the evaluation period, the groves start the class probability estimations in parallel, with every grove receiving different inputs. If the probability threshold (confidence level) is not met, the "partially computed" result is issued to the next grove. That way more computational resources are dynamically allocated to examples with higher uncertainty, thus reducing the average cost of estimation. Our evaluation shows that at comparable accuracy FoG consumes ≈1.48×, ≈24×, ≈2.5×, and ≈34.7× lower energy per classification compared to conventional RF, SVM_{RBF}, MLP, and CNN, respectively. FoG is 6.5× less energy efficient than SVM_{LR}, but achieves 18% higher accuracy on average across all considered datasets.

5.2 Energy-Efficient Computing using Random Forests

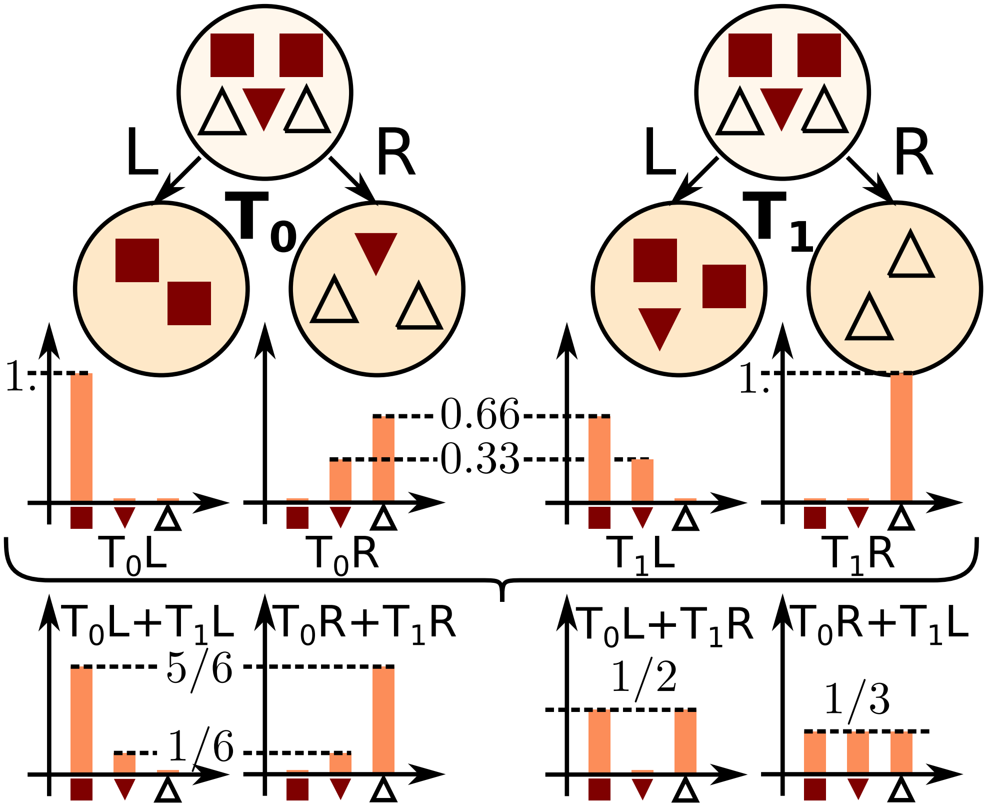
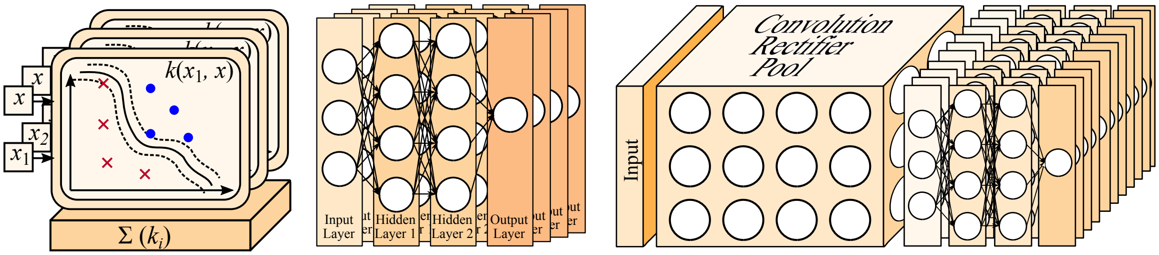


Figure 5·1: Example of decision making within a random forest or a Grove with two decision trees $\mathbf{T}_0$ and $\mathbf{T}_1$. "L" and "R" represent the decision paths within the trees. Symbols within the decision nodes represent the distribution during the training phase. The bottom bar plots illustrate the decision combinations. For example, "T_0L+T_1R" shows what the resulting distribution would look like if tree $\mathbf{T}_0$ chooses path "L", and tree $\mathbf{T}_1$ chooses path "R".

As mentioned in Section 5.1, we commonly use SVM as well as traditional neural network-based algorithms like CNN or MLP for classifying data sets with large number of features. Figure 5·2 shows high-level logical view of SVM, MLP, and CNN.



(a) Support Vector Machine (b) Multi Layer Perceptron (c) Convolutional Neural Network

Figure 5·2: Conceptual diagrams of Support Vector Machine, Multi-Layer Perceptron (fully connected), and Convolutional Neural Network.

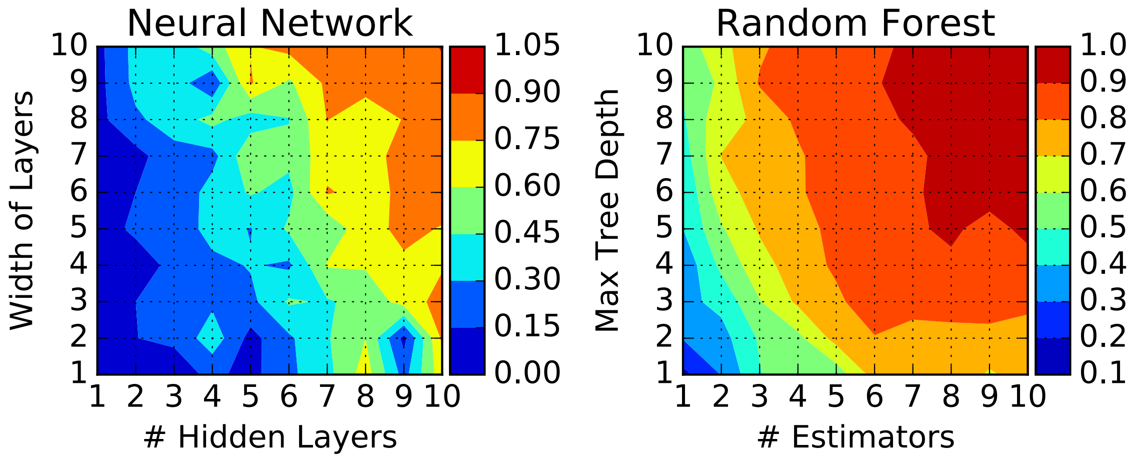


Figure 5·3: Accuracy degradation for MLP and RF. For simplicity, a reduced `Digits` dataset was used. In the RF every estimator is a DT.

In this section we analyze the use of RFs as compared to the popular classification algorithms.

RF is composed of binary decision trees (DT) (see Figure 5·4a), and although the entire RF is composed of $O(t2^d)$ decisions, where t is the number of trees, and d is the upper bound on the tree depth, evaluation during testing requires only $O(td)$ computations. Every DT_i receives some input features X_{Ri}, where X_{Ri} is a random subset of input X. A "Majority Vote" across trees is then used to identify the label. Such an approach avoids overfitting and ensures high accuracy. To illustrate the operation of a random forest, consider the equation 5.1 and 5.2 below

$$\hat{y} = \text{argmax}\left(F_T(x^{(i)})\right) \tag{5.1}$$

$$F_T(x^{(i)}) = \gamma \sum_{t=1}^{T} h_t(x^{(i)}), \tag{5.2}$$

where $\hat{y}$ is the class prediction, F_T is the aggregated decision of the forest with T trees, and h_t is the decision of tree t. γ in this context is a normalization factor, and is set to $1/T$ for random forest (as opposed to boosting methods where γ is a per-tree normalization factor). To illustrate the equation, let us look at the example on figure 5·1, which shows a forest with 2 decision trees of depth 2. When combining the results of individual trees, the results are aggregated using a majority vote shown on equation 5.2. For example, if tree T_0 goes through path "L", and tree T_1 goes through decision path "L", than the example decision is obvious, as there is a clear maximum in the decision leaf. However, if T_0 goes through decision path "L", and T_1 through "R", there is no obvious maximum.

Note that during the random forest training (shown on algorithm 1, the trees are generated depending on their validation cost, where the cost could be energy, delay, energy-delay product, or accuracy. Because the DTs within RF work independently on random subsets of input features, those decision trees can be seen as having additive properties, which means that even if several of the DT blocks are turned off, the total accuracy degrades gracefully (Figure 5·3). This "scalability" property, where the accuracy of RF scales with the number of DTs makes the RF a suitable candidate for budget-limited environments, with budget being energy, delay, accuracy, or a combination of them. For example, depending on the budget requirements, some of the trees could be turned off to improve energy efficiency, but at the cost of statistical performance. The reason why turning off DT blocks in an RF generally leads to a graceful degradation of accuracy, is that the predicted label for a new test example within a sub-tree is independent of other trees and the majority vote is used to reduce variance. This type of "scalability" is also observed in CNNs and MLPs but the scalability is less forgiving and hence, those algorithms are hard to adapt to environments with changing budgets. The reasoning is that inontrast to RFs, CNN and MLP network nodes are connected to many other nodes, which usually makes it difficult to predict how each node affects the accuracy of the the neural network at run-time (it is possible to achieve that offline however). In general, when operating at unlimited energy budgets, CNNs and MLPs generally provide higher accuracy than RF, but RF provides us an opportunity to game accuracy for energy-efficiency in energy-constrained environments. In Section 5.3, we show more detailed comparison of the classifiers.

Algorithm 1 Constructing Feature Budgeted Random Forest. **Validation** cost is the energy dissipation cost of classifying "unseen" examples.

```
Require: Training feature set $X$; Training label set $y$; Budget $B$ - budget could be
    accuracy, energy, delay, energy-delay-product, etc.
Ensure: Set of trained decision trees $T$
procedure BudgetRF($X, y, B$)
    $T \leftarrow \emptyset$                    ▷ Assume that empty set has a cost of 0
    while Average validation cost on $T \leq B$ do
        $T \leftarrow T \cup$ Tree($X, y$)
    end while
    return $T$
end procedure
```

5.2.1 Field of Groves Algorithm

As described in section 5.2, RF is suitable for environments where accuracy could be traded-off for energy efficiency. The main advantage of the RF approach stems from the fact that the accuracy of RF tends to improve with an increase in the number of DTs. Moreover, DTs have few active computational nodes during prediction, and the nodes are generally of very low computational complexity. One of the disadvantages

Algorithm 2 Constructing Field of Groves Classifier

```
Require: Number of estimators $n > 0$; Maximum size of a grove $k \leq n$; Training set $X, y$
procedure GCTrain($n, k, X, y$)
    Train $RF \leftarrow$ RandomForestTrain($n, X, y$)
    return Split(RF, k)
end procedure
```

Subroutine - Splitting a Random Forest

```
Require: Pretrained random forest $RF$; Maximum size of a grove $k > 0$
Ensure: Split grove ensemble $GC$
procedure Split($RF, k$)
    $i \leftarrow 0$
    $GC \leftarrow \emptyset$
    while $i <$ Length($RF.estimators$) do
        $G \leftarrow$ new Random Forest of size $k$
        $G.estimators \leftarrow RF.estimators[i..i + k]$
        $GC \leftarrow GC \cup G$
        $i \leftarrow i + k$
    end while
    return GC
end procedure
```

that RF experiences is "over-utilization" of the computational resources. Previous works have shown that large portion of input samples within datasets are far enough from the decision boundaries, and do not require complex classifiers (Takhirov et al., 2016, Venkataramani et al., 2015a). Conventional RFs, however, lack the ability to allocate less computational resources for the inputs if desired. This problem could be solved by using only a limited number of trees, depending on the current confidence level.

In this section we propose a novel RF implementation called Field of Groves (FoG), which avoids any unnecessary expending of energy on inputs with low uncertainty. Each grove is composed of a random, non-overlapping subset of the trees from the "original" RF. Figure 5·4(b) shows the logical view of our proposed FoG

Algorithm 3 Evaluating Field of Groves Classifier

```
Require: Stopping threshold $0 < thresh < 1$; Maximum number of hops $max\_hops \leq$ Number of groves $n\_groves$; Input set $X$
procedure GCEval($X, thresh, max\_hops$)
    parallel for every $x$ in $X$
        $start \leftarrow$ Random(from 0 to $n\_groves$)    ▷ Start at random grove to avoid bias
        prob $\leftarrow \{0\}^{\#labels}$
        for $j \leftarrow$ from 0 to $max\_hops$ do
            $index \leftarrow (start + j) \mod n\_groves$
            prob $\leftarrow$ prob + $Grove(index).predict\_prob(x)$
            prob_norm $\leftarrow$ prob$/(j + 1)$
            if MaxDiff(prob_norm) $\geq thresh$ then
                return prob_norm
            end if
        end for
        return prob_norm
    end parallel for
end procedure
```

Subroutine - Minimum Difference of Maximum Values

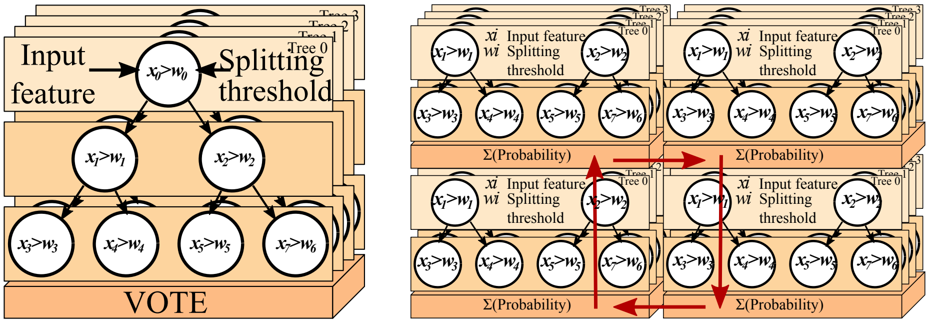
```
Require: Array $ar$
procedure MaxDiff($ar$)
    max1, max2 $\leftarrow$ TwoMaximumValues($ar$)
    return abs(max1 - max2)    ▷ In case of "Multi-output classification", min($\cdot$) function is called prior to returning
end procedure
```

implementation of RF. Equation 5.4 shows the decision making process of the FoG.

$$\hat{y} = \text{argmax}\left(F'_T(x^{(i)})\right) \tag{5.3}$$

$$\begin{aligned} F'_{T+1}(x^{(i)}) &= \mathbb{1}_{\mathbb{M}(F'_T \geq \tau)} F'_T \\ &+ \mathbb{1}_{\mathbb{M}(F'_T < \tau)}(F'_T + h_{T+1}(x^{(i)})), \end{aligned} \tag{5.4}$$

where $\nVdash_z$ is an indicator function of subset z, T is the current Grove, initially randomly chosen from all the available Groves[1]. $\mathbb{M}(\mho)$ is a `minmax` function that returns the difference of the two maximum values in an array $\mho$. If $\mho$ is a 2-dimensional array, it



(a) Random Forest (b) Field of Groves

Figure 5·4: Logical view of RF and FoG.

returns the minimum of the maximum differences. τ is the threshold value. Equation 5.4 redefines the equation 5.2 by including a conditional execution of the next subset of trees depending on the confidence level $\mathbb{M}(F_{T-1}) \geq \tau$. Also, in the context of FoG, $h_T(x^{(i)})$ is a T'th Grove (or a collection of decision trees), rather than a single decision tree.

Training and Testing

The training of the FoG is described in Algorithm 2 and is done offline. During this training phase, a RF is first pre-trained (using Algorithm from (Nan et al., 2015)), and the DTs are randomly split into groves. The splitting involves a simple division of the forest into sets with k DTs, where k is the size of the grove.

The label evaluation algorithm for the approach is shown in Algorithm 3. Here, for every input $x \in X$, we compute the confidence score using one of the randomly selected grove. Confidence in this context is defined as the difference between the most probable and second most probable labels[2] If the confidence is higher than the goal threshold, the computation for x is complete. Otherwise, x and the current probability distribution is sent to the next grove, where the probability array is recomputed again

---

[1] Incrementing and decrementing the Grove indexes follows the rules of modular arithmetic

[2] In case the classification problem is "multi-label" or "multi-output", the `MaxDiff` returns the `Min` of the differences within the label. That means that confidence level is defined as the "minimum difference of the maximum values"

form x and combined with the one received from the previous grove. That way more groves contribute to the inputs with high uncertainty. This process is repeated until either the threshold is exceeded or the the entire forest is evaluated. Note the contrast between FoG and conventional RF evaluation: in FoG the groves return probability distributions which are averaged out across groves; in the conventional RF however the DTs return class predictions, which are later put to a majority vote.

5.2.2 Micro-architecture

The high-level architecture of the FoG implementation of RF is shown on Figure 5·5. Here, the groves are connected in a circular fashion, with each grove being able to send its current inference to the next grove. To understand the operation of the system, let's consider an example with a 3-class (class A, class B and class C) problem and the threshold value set to 0.1, which means that in order to come to a decision, the probability difference between two most probable classes should be at least 0.1. Let us assume that the processor sends an input X that has 5 features. When FoG receives this input, it is assigned an *id*, and is sent to one of the groves (say grove G0 in Figure 5·5) through the accelerator Input Queue.

Notation	Description
a	Value of variable a
$\$a$	Memory address of variable a (pointer)
$\$a + c$	Memory offset by c
a_j	j-th element in vector $a \in \mathbb{R}^m$
$a^{(i)}$	i-th vector
fr_G, bk_G	Front and back of a queue in Grove G

Table 5.1: Notation used in section 5.2.2

Data Queue

Once $G0$ receives a new input, it places it into the local memory, which serves as a data queue. The queue is controlled using two pointers: $\$fr_{G0}$ and $\$bk_{G0}$ for front

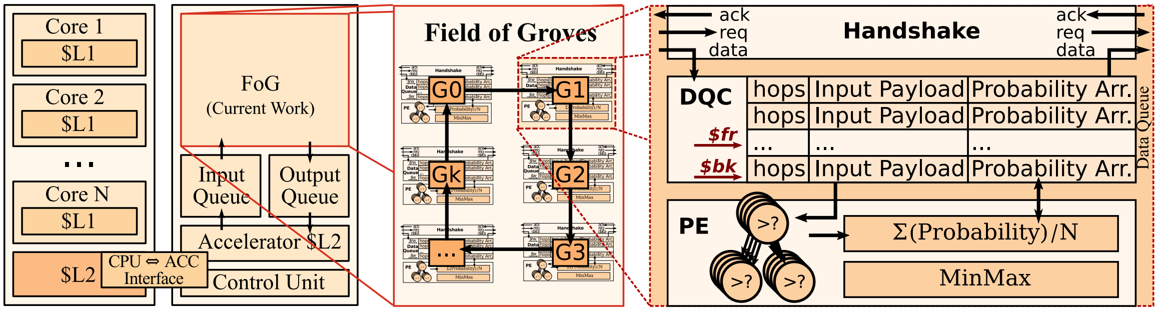


Figure 5·5: Random Forest implemented as Field of Groves and the microarchitecture of a grove. Notice the grove $G0$ can communicate to grove $G1$ through the “Handshake” block without going out of the FoG. The “Data Queue” includes a controller to maintain pointers $\$fr$ and $\$bk$.

and back of the queue respectively. $\$fr_{G0}$ always points to location that contains the input that is currently being processed and $\$bk_{G0}$ points to the first empty location at the back of the queue. For each input we store `Input Payload`, which holds the received input features + id; `Probability Array`, which contains the current prediction probabilities; `hops` which is a count of groves that have so far processed the current Input Payload. Whenever a new input is received: if the input is received from the processor, it is placed at the back of the queue. The `Input Payload` and `Probability array` values are set based on the information sent by the processor and the `hop count` is set to 0. If the input is received from the neighboring grove, it is placed at the front of the queue. The `Input Payload` and `Probability array` values are set based on the information sent by the neighboring grove and the `hop count` is incremented by 1. This ensures that the input that were partially computed have higher priority. In our example, because the input is from the processor, it is placed into the $\$bk_{G0}$ location of the queue. For the new input: {`hops = 0, Input Payload =` X`, Probability = `{`0,0,0`}}

Data queue is controlled by the queue controller (DQC) which is responsible for maintaining the $\$fr$ and $\$bk$ pointers. For each received input, DQC routes $\$fr$ to

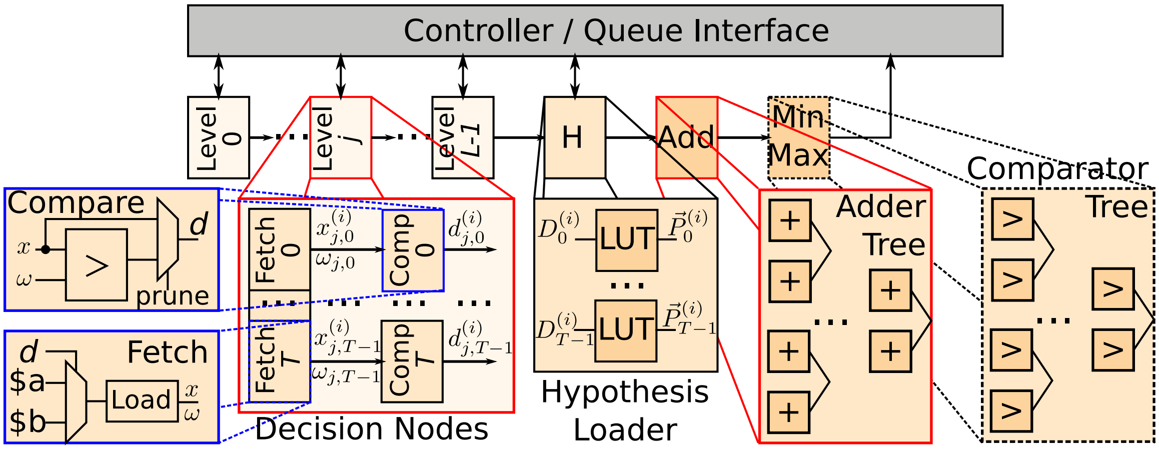


Figure 5·6: Grove Processing Element (PE) as shown on figure 5·5. The decision making process goes in a pipeline through different levels of the decision trees using "Fetch-and-Compare" blocks. The results are aggregated using fixed-point adder tree and a maximum finder (comparator tree).

the processing element, the processing element reads the entries corresponding to $\$fr$ and once the computation is complete, it writes the results back to $\$fr$. $\$fr$ and $\$bk$ are incremented by Γ, which is the length of queue word and represents number of rows in physical memory that are required to store the hop count, `Input Payload` and `Probability Array`. Γ is a variable, and it depends on the number of features and number of classes in dataset. In our example, $\Gamma = 1 + 5 + 1 + 3 = 10$ (1 byte for `hops`, 5 bytes for features in `Input Payload` + 1 byte `id`, and 3 bytes to store the current label prediction in the `Probability Array`). Our current implemetation of the FoG has a data queue of 6kB, and can store 8 `MNIST` examples per grove. Note that the memory can be easily increased to support datasets with larger feature counts and label counts.

Processing Element (PE)

The PE in every grove is represented by a set of decision trees and its operation is described in algorithm 3. The implementation block diagram is shown on figure

5·6. `Controller / Queue Interface` is responsible for routing the values from DQC to the pipelined PE. As shown on the figure, first L stages are decision trees being executed in parallel. The level of parallelism and number of pipeline stages are design-time optimization parameters. Every decision node in a tree consists of fetch-and-compare blocks, which load the input and weights from the PE buffer (not illustrated), and compare the values to identify the next stage node variables. Note that both blocks are subject to pruning – a process of "early stopping" of the decision process in a tree to improve generalization (Hastie et al., 2001). After the decision levels, the PE employs a "Hypothesis Loader" – a pretrained set of probability vectors per decision tree leaf. Every leaf has its own probability vector. Because there is a one-to-one mapping from the decision leaf to the precomputed probability, this block is implemented as a simple look-up table (a tree with 5 levels will have $2^5 = 32$ entries in the LUT). After that, the probabilities across all trees are summed up in the "Adder Tree", and the two maximum values in the resulting probability array is found using a "Comparator Tree".

The `Input Payload` (X) is processed by all the trees within the grove to determine the probability distribution of the labels. This result is then averaged with `Probability Array` received from previous grove or just written back in case of a new input and the current confidence level is computed (as the difference between the two largest values in the `Probability Array`). The latency of the PE depends on the number of trees per grove, the maximum depth of each tree and degree of parallelism.

Once PE finishes the computation, a decision is made if the current confidence level is adequate. If so, the DQC is notified that the classification of the current `Input Payload` is complete and the computed result needs to be sent back to the processor. However, if the confidence level is lower than a threshold $thresh$, a request is sent to the next grove for further processing. Here the entire entry (`Hop Count`, `Input Payload` and `Probability Array`) for the current input is copied to the next grove.

Continuing with the previous example, let us say that after $G0$ completes processing the input X, it returns the probability distribution of $\{0.32, 0.35, 0.33\}$. This is used to compute the confidence. In this example the confidence is $0.35 - 0.33 = 0.02$. Because the threshold was set at 0.1, the classification of input X is considered incomplete. It is written back to the location $\$fr_{G0}$, and a `req` flag in the handshake is raised. At this point, the fr_{G0} is incremented, and grove $G0$ is ready for the next input. The value stored at $\$fr_{G0} - 1$ is {`hops = 1, Input Payload =` X, `Probability =` $\{0.32, 0.35, 0.33\}$}

Handshaking Protocol

Groves use a simple handshaking protocol to talk with each other. After $G0$ computes the output probabilities, it checks its confidence and if the confidence is low it sets a `req` flag to signal the neighboring grove $G1$ to copy the current input as well as computed probabilities. Once the copy is complete, an acknowledgment flag `ack` is raised by $G1$ for one cycle to notify that the copy procedure is complete. At that time $G0$ pulls the `req` line down, completing the handshake.

Because $G1$ receives its input from another grove (in our case $G0$), it places it at $\$fr_{G1}$ of its queue. Assume that $G1$ computed the probability distribution as $\{0.28, 0.45, 0.27\}$. These values are averaged with the values computed by $G0$. The entries corresponding to the current input are now {`hops` $= 2$, `Input Payload` $= X$, `Probability` $= \{0.3, 0.4, 0.3\}$} and the predicted label is argmax (`Probability`) $= 1$. At this point, the threshold value constraint is met $0.4 - 0.3 \geq 0.1$, which indicates that this input does not require any further processing, and should be sent to the accelerator output queue. Note that in the example discussed above, the value of `hops` was increasing with every new grove.

Run-time Tunability

In our proposed FoG implementation of RF the energy-efficiency and accuracy could be easily tuned by changing the `probability threshold` and `maximum hops` parameters. The `threshold` parameter indirectly controls the number of groves that process the input. The `maximum hops` parameter places an upper limit on the number of groves that process the input (based on EDP or accuracy constraints). A detailed evaluation of how the probability threshold parameter and the maximum hop count parameter affects the energy efficiency and accuracy of our FoG implementation is presented in Section 5.3.

Reprogrammability

To support various trained RFs corresponding to various datasets, the DTs were implemented to be reprogrammable. For a given dataset, every node is populated with the weights ω_i, as well as memory address offsets for the respective features x_j. In addition to that the DQC is programmable to support variable step for the queue pointer. For example, if a node N checks the conditional $x_N > \omega_N$, then this node will store the constant ω_N, as well as offset $OFFx_N$. This indicates that the location of the input x_N is at $\$fr + OFFx_N$. At the same time the DQC stores a value Γ and the next entry in the queue has an address $\$fr_{next} = \$fr + \Gamma$. The reasoning behind having a variable step size Γ is that we want to support different number of features as well as different number of labels for different datasets. For example, MNIST dataset has 784 features and 10 labels, while Penbase Digits dataset has only 16 features and 10 labels[3].

---

[3] Physically each entry of the data queue is spread over several rows. Here $OFFx_N$ is the offset within an Input

One of the aspects of the programmability of the decision trees in the RF and a FoG. This is done by reprogramming the hypothesis look-up tables (LUT) shown on figure 5·6. The LUTs hold information about the probability distributions at every leaf of a decision tree. The number of entries in every lookup is 2^D, where D is the depth of a decision tree (D is capped at 5 in our implementation). Note that the current work does not support changing the number of trees at run-time.

Let us consider a simple reprogramming example to illustrate the above process. Suppose that FoG was setup to process handwriting digit recognition, and was configured for MNIST dataset. That means that $\Gamma = 784$, and the features for input i are stored at $\$x_0^{(i)} = \$fr + 0, ..., \$x_{783}^{(i)} = \$fr + 783$. The next input in the queue will be stored at $\$x^{(i+1)} = \$fr_{next} = \$fr + \Gamma = \$fr + 784$, which means that the input $i + 1$ is queued in the memory address $\$x_0^{(i+1)} = \$fr + 784, ..., \$x_{783}^{(i+1)} = \$fr + 1567$. Suppose that we have decided to reduce the resolution of the handwriting recognition dataset, and switch to `Penbase Digits` dataset, which has only 16 features. This makes $\Gamma = 16$, and the consecutive inputs in the memory would be stored at $\$x_0^{(i)} = \$fr + i\Gamma + 0, ..., \$x_{15}^{(i)} = \$fr + i\Gamma + 15$, $\$x_0^{(i+1)} = \$fr + (i + 1)\Gamma + 0, ...$

Although our approach requires mechanism for queue management and needs an additional mechanism to "dereference" the offset pointer (which slightly increases the complexity per tree), this design choice is a necessary trade-off to support programming of hardware required to support datasets having different feature counts.

5.3 Evaluation

5.3.1 Experimental Setup

General Design Flow for Custom Accelerator

We designed SVM with linear regression kernel (SVM_{LR}), SVM with Radial-Basis Function kernel (SVM_{RBF}), MLP, CNN, RF and FoG classifiers for our analysis. To compare the different classifiers we used five different datasets from the UCI library (UC Irvine Machine Leanring Repository, 2017), and the list of these datasets is shown in table 5.2 under the “Dataset” column. These datasets were chosen because they represent a diverse set of workloads typically seen on a mobile device. It is worth making a side note that there are datasets which would require much more complex classifiers such as deep neural networks (i.e. ImageNet dataset). However, because the main gaol of this project is energy-efficiency and it is not expected an embedded system to work with extremely large datasets, we have decided to exclude them from the evaluation. All the results shown in this section are acquired using the inputs never seen before by the systems under test.

Data Set	Description	Features	Size
ISOLET	Spoken letters	617	7797
Penbase	Hand written digits	16	10992
MNIST	Hand written digits	784	62000
Letter	Hand written letters	16	20000
Segment.	Image patches	19	2310

Table 5.2: Datasets used for evaluation.

We used the following design flow for performing a detailed power (shown on figure 5·7), performance and area comparison of our proposed FoG with other ML classifier algorithms:

Step 1: First, basic computational blocks, such as adders, multipliers, multiply-accumulate (MAC), sigmoid, etc. that are required by all the classifiers are designed considering trade-offs between energy and delay by sweeping through architectural and circuit level parameters, such as bitwidth precision, parallelization, pipelining, memory ports, memory and buffer size. We used Aladdin tool (Shao et al., 2014) to explore the architectural design space, and Cadence tools to extract Power-Performance-Area (PPA) values for each block in this step.

Step 2: Once the library of computational units is generated, it is used in the

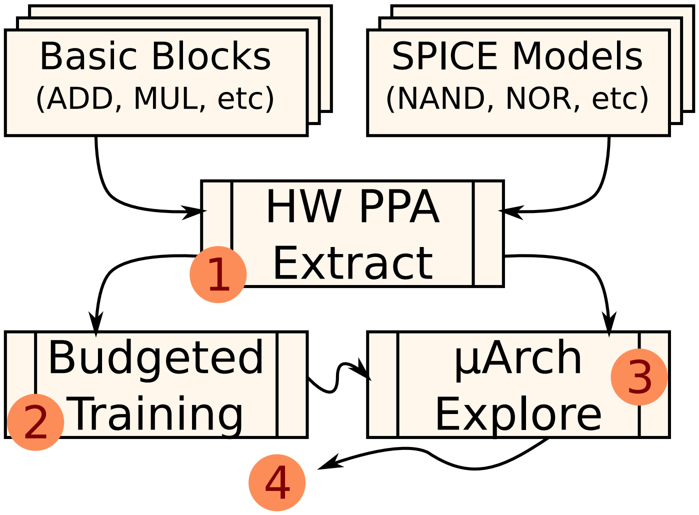


Figure 5·7: System design flow diagram. At steps ① basic building blocks are designed and simulated to acquire the Power-Performance-Area (PPA) numbers. The PPA is then fed into the step ② to get the accelerator configuration given energy constraints. The PPA numbers are also fed into the μArchitecture exploration ③, which generates a suitable chip layout for simulation ④

offline budgeted training described in algorithm 1 (Nan et al., 2015) and algorithm 2. We used energy-delay product (EDP) as budget metric during this phase. If there are several designs that meet the energy constraints, we choose the one with the maximum accuracy. Budgeted training requires information about the costs of building blocks which is provided by the PPA models[4]. We use SciKit-Learn (Pedregosa et al., 2011) for the training (and exploration of logical structure) of the classifiers.

Step 3: At this step, the detailed hardware microarchitecture of the accelerator is designed. Microarchitecture exploration is independent of training, and is done using Aladdin toolset (Shao et al., 2014). The PPA models from the previous step are used during this step to determine Pareto optimal frontier and select the most energy-efficient design. The parameters explored for Pareto optimality are the same

---

[4] Note that the cost could be defined as either energy, delay, area, accuracy or any combination of them. The PPA library has information about delay, energy, and area, while the accuracy cost is determined using cross-validation data.

as the ones used in step 1, but in this step individual blocks (adders, multipliers, etc.) are seen as black-boxes.

Step 4: In this final step we design the whole architecture using Chisel HDL (Bachrach et al., 2012). This design environment was chosen, as it generates both hardware description code (Verilog) as well as C++ functional model. That allows for the functionality of the hardware to be verified against software implementation for correctness. The Verilog code was synthesized using 40 *nm* Global Foundries technology with Synopsys standard cells for detailed power-performance analysis.

FoG Design Considerations

In our implementation, all the classifiers were designed for minimum EDP at maximum accuracy. The FoG classifier was designed from the RF classifier by extracting the pre-trained DTs and re-assembling them into groves. As described above, the number of decision trees per grove and the number of groves is decided during the design time. During the design time we analyzed the EDP and accuracy of different FoG topologies and the minimum EDP design point was selected (while maintaining the accuracy). Figure 5·9 shows the accuracy and EDP results across different combinations of sizes of groves and total numbers of groves in the FoG.

To illustrate the choice of design time parameters while considering run-time tunability, let us discuss an example with 16 decision trees and `ISOLET` dataset. After examining the accuracy and EDP of different topologies (see Figure 5·9a), we isolated two candidate topologies: 8x2 and 4x4[5]. At this point we can use the "run-time tunability" as a deciding factor between these two roughly equivalent candidate topologies. Figure 5·10 shows the accuracy and EDP across all datasets as a function of threshold. Figure 5·10a shows that 8x2 topology is more energy-efficient, but the accuracy is lower for lower threshold settings. Figure 5·10b shows, in contrast, that the accuracy penalty for lower thresholds is not as drastic, but the energy-efficiency

		SVM					FoG	
	Dataset	lr	rbf	MLP	CNN	RF	max	opt
Maximum Accuracy (%)	`ISOLET`	69	93	87	94	92	91	90
	`Penbase`	86	95	91	96	96	93	93
	`MNIST`	82	95	87	96	96	94	93
	`Letter`	78	93	93	96	95	85	85
	`Segment.`	67	91	91	96	95	94	92
Energy (nJ/class.)	`ISOLET`	5.9	980	82.5	1150	41	49	30
	`Penbase`	0.4	18	13.3	186	16	14	7.1
	`MNIST`	6.1	1020	93	1300	43	47	38
	`Letter`	0.5	19	13.7	192	16	12.9	7.6
	`Segment.`	0.6	26	14.5	203	13	9	4.7
	Area (mm^2)	0.13	0.53	0.93	2.1	1.38	1.9	1.9

Table 5.3: Accuracy (top) and Energy dissipation (bottom) in nJ per classification for different datasets (UC Irvine Machine Leanring Repository, 2017). Frequency is fixed at 1 GHz for all datasets. SVM_{lr} and SVM_{rbf} show the results for SVM with linear and RBF kernels; FoG_{max} and FoG_{opt} show the results for FoG with its threshold set to maximum and to the optimal accuracy tuning point, respectively. The area results are in mm^2

.

penalty is higher. In our case we go with 8x2 topology as minimum EDP is our primary goal. Note that once the physical topology is selected, the "threshold" variable could be changed during run-time to achieve a different operating point.

Design for use on GPU and Other Accelerators

GPUs have been widely used to support ML due to their speed advantage over SIMD CPU. To evaluate the scalability of different algorithms, we implemented the baseline algorithms as well as RF and FoG on a modern GPU card (NVIDIA GTX 660M, 2GB GDDR5). We compare the results against a CPU implementation (Intel Core i7-3610QM, 8 cores @ 2.3GHz, with compilation flags "`-O3 -ftree-vectorize -march=native`"). As shown on figure 5·11 GPU achieves an average speedup of 23x-90x with respect to SIMD CPU implementation. The results are similar to the pre-

---

[5] We use a x b to describe a FoG topology with a number of groves with b decision trees in each grove.

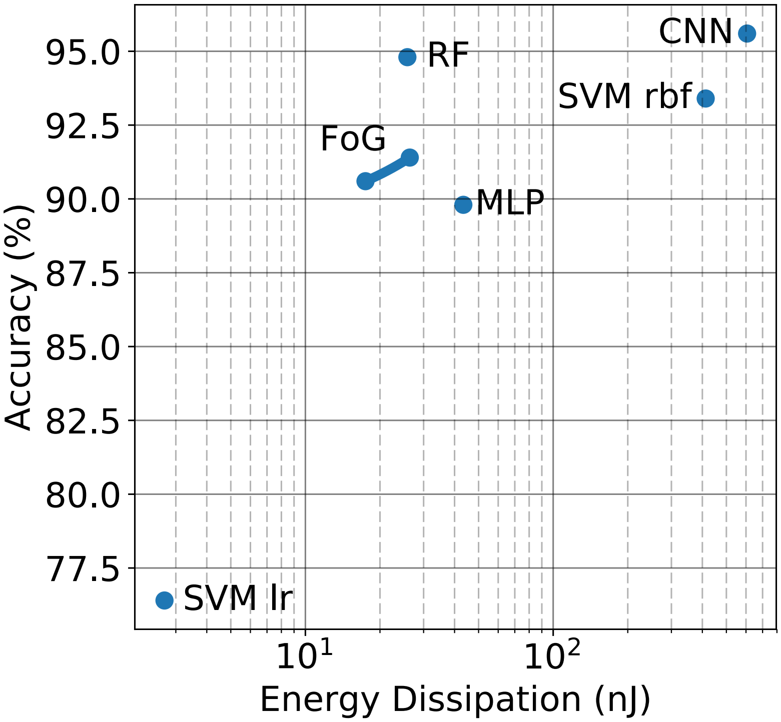


Figure 5·8: Average Accuracy vs. Average Energy dissipation scatter plot for different classifiers. CNN achieves the highest accuracy but orders of magnitude more expensive in terms of energy dissipation. Linear SVM is the most energy efficient, but the accuracy is the lowest across different classifiers. Random Forest and Field of Groves provide a reasonable tradeoff between accuracy and energy dissipation.

viously reported speed comparisons for ML applications for GPU vs. CPU (Cireşan et al., 2011, Teodoro et al., 2009). In order to ensure a fair comparison, both GPU and CPU implementations were designed to guarantee functional equivalence. This also makes the accuracy to be the same for both GPU and CPU implementations.

The proposed approach can also be used as part of other accelerators. For example, FoG and RF algorithms would benefit from parallel architectures of DianNao-family processors (Chen et al., 2016a), as well as Cambricon (Liu et al., 2016). The functional units in these accelerators are suitable to execute the weak classifiers (decision trees and groves) effectively.

5.3.2 Experimental Results

To perform a comparison of the classifier algorithms listed in section 5.3.1, we first trained all the algorithms for their maximum accuracy without worrying about energy efficiency. Table 5.3 shows the comparison of accuracy between different classifiers. Two different numbers for the FoG are reported: FoG_{max} and FoG_{opt}. FoG_{max} shows the results for the FoG with its "threshold" parameter set to maximum. This forces the FoG to behave like an RF because every input will have to go through every decision tree of every grove. FoG_{opt} shows the results for the case when confidence threshold was set to accuracy optimal point – a threshold point above which accuracy does not increase with threshold but below which accuracy decreases with decrease in threshold.

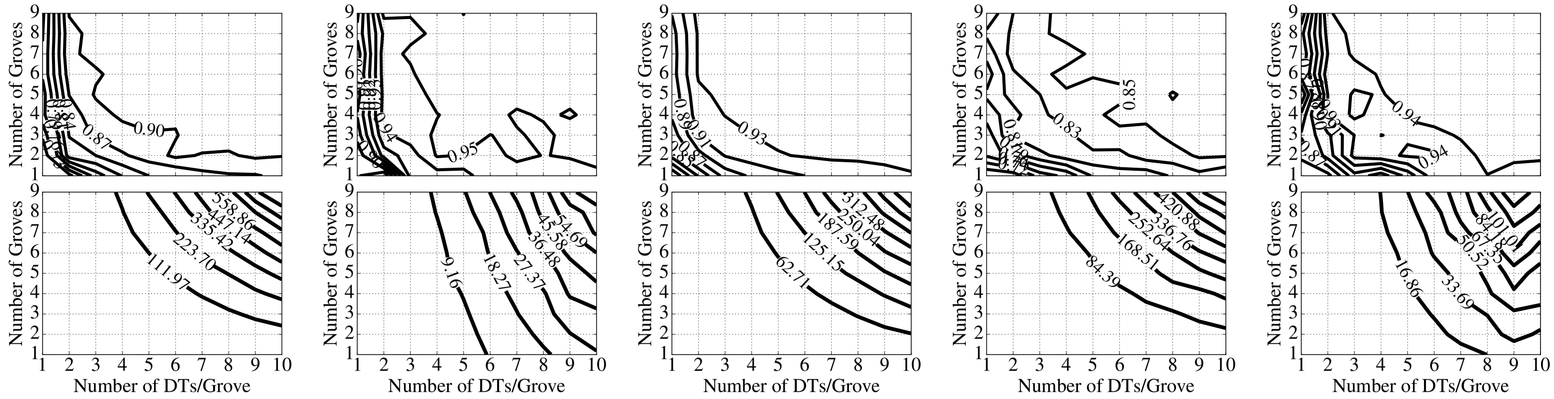
From the table we can see that CNN has the highest accuracy for all datasets. The accuracy of the traditional RF classifier is comparable to CNN for all datasets. In terms of energy per classification, RF consumes $\approx$ 95.4%, 91.5%, and 99.4% less energy than SVM_{RBF}, MLP and CNN, respectively. The RF energy consumption is $\approx$22.4% higher than that of SVM_{LR}, but RF on average provides 20% higher accuracy than linear SVM. The very low energy dissipation in RF is due to the fact that the basic computational unit in a DT is very simple (a basic comparator).

Table 5.3 also shows the accuracy and energy dissipation of our proposed FoG implementation of the RF classifier. Here all classifiers have been designed to operate at 1 GHz. The maximum achievable accuracy of the FoG (both *max* and *opt*) implementation is lower than RF and CNN by 3.2% and 4%, respectively, but FoG_{opt} classifier consumes 42% and 99.7% lower energy than RF and CNN, respectively. The FoG_{max} on average consumes 6% lower energy than RF, and 99.5% lower than CNN. When comparing to the SVMs, the accuracy of the FoG classifier outperforms the linear support vector machine SVM_{LR} by $\approx$15% on average, and achieves comparable statistical performance when compared to the SVM_{RBF}. In terms of energy SVM_{LR} is ≈20% more efficient on average, while SVM_{RBF} is more expensive (97.9% higher energy consumption when compared to FoG_{opt}).

The main advantage of the FoG is that while achieving statistical performance comparable to the performance of the RF, it also allows easy run-time change in the energy-accuracy trade-off. Figure 5·10 shows how accuracy could be traded off for energy for 8x2 and 4x4 FoG topologies. Figure 5·10a shows that energy-efficiency could be easily improved by an order of magnitude without sacrificing much accuracy by tuning the confidence threshold from 1.0 to 0.5 for most datasets. After that a "trade-off" region of tunability starts – one can improve energy-efficiency by trading off accuracy. This run-time tuning opportunity will prove beneficial in environments with constraint energy. The figure shows that for 8x2 design, two orders of magnitude improvement in energy efficiency could be achieved by tuning the confidence threshold from 0.5 to < 0.1. The accuracy drop in case of aggressive confidence tuning is anywhere between 10% to 30% depending on the dataset. The story is similar for 4x4 topology (figure 5·10a), however, the "trade-off" region of tunability starts at confidence threshold of ≈0.3. Although the accuracy drop is not as drastic, the EDP for 4x4 topology is much higher – an order of magnitude higher for low accuracy, and equivalent for high accuracy points.

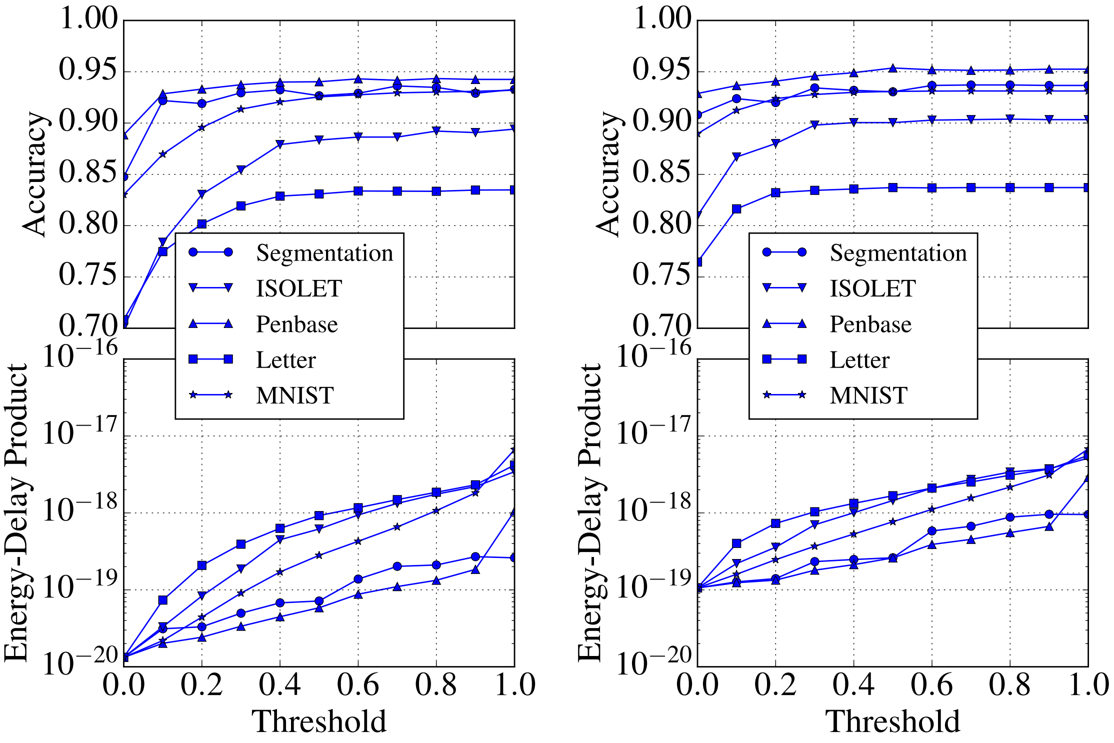
Table 5.3 also shows the area comparison between different classifiers. It must be noted that most classifiers' area changes drastically with the internal parameters – e.g. convolutional layers sometimes implemented as having "volume" activation, and changing the size of one layer, might contribute to the total area change cubically. Overall, the area of our FoG implementation is larger than all classifiers except CNN.

Comparison of the GPU vs. CPU implementation is also shown on figure 5·11. GPU's provide 23x speedup over the CPU implementation on average, and RF ar-



(a) ISOLET **(b)** Penbase **(c)** MNIST **(d)** Letter **(e)** Segmentation

Figure 5·9: Accuracy and EDP as a function of “Number of Groves” and the “Number of Decision Trees per Grove”. The product of the two variables shows the total number of decision trees in the FoG.



(a) 8 Groves, 2 DTs/Grove **(b)** 4 Groves, 4 DTs/Grove

Figure 5·10: Example of FoG run-time tuning using the "threshold" variable.

chitecture achieves 3.95× and 2.1× higher speedup over speedups of MLP and CNN respectively. FoG architecture does not achieve speedups as high as RF, due to interaction between Groves, that is hard to synchronize on GPUs. The figure also shows the speedup when different designs implemented on FPGA. Notice that FoG achieves higher speedups on FPGA, because Groves are implemented as independent cores with their own memory, which improved the computational performance. Note, that CPU and GPU are implemented using floating point precision using Caffe and BLAS (Jia et al., 2014, Whaley and Petitet, 2005), while FPGA is implemented with fixed point precision. Table 5.4 summarizes the implementation and the results.

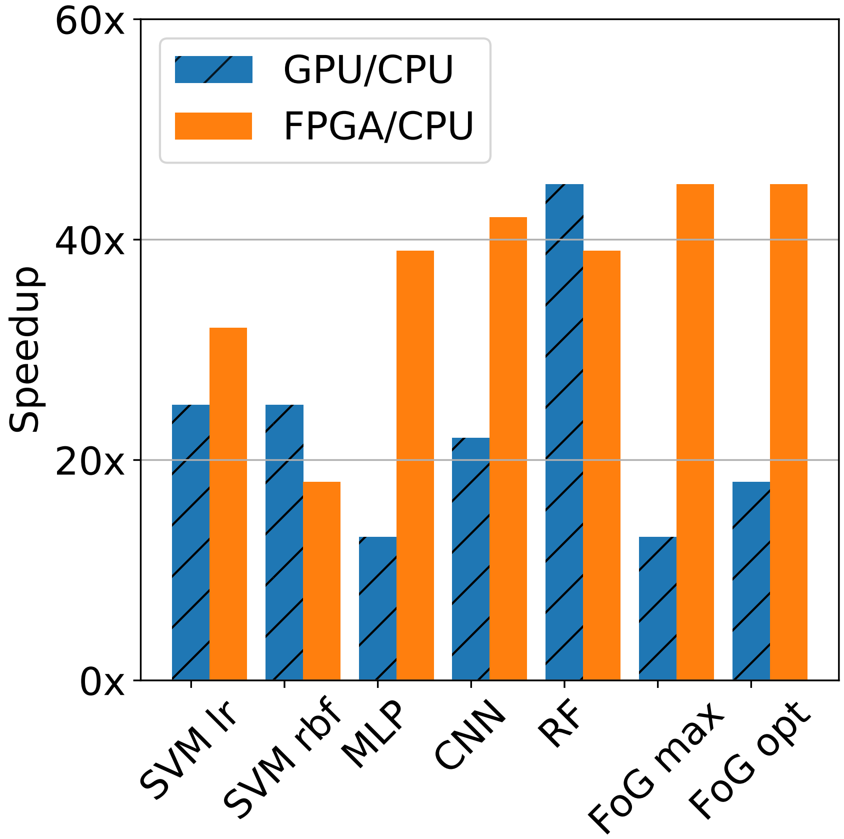


Figure 5·11: Computational performance comparison between GPU, FPGA, and SIMD CPU

5.4 Conclusion

In this work we have compared the lightweight RF classification algorithm with heavyweight classification algorithms like CNN, MLP, and SVM in terms of accuracy and energy efficiency. We proposed a novel FoGs approach to RF implementation that can dynamically trade-off accuracy for energy efficiency at run time, while achieving accuracy comparable to traditional RFs. The proposed FoG approach examines decision confidence for each input, and allocates the computational resources depending on the input's uncertainty levels. We implemented the FoG using a 40 nm technol-

	CPU	GPU	FPGA
Device	i7-3630QM	GTX660M	XC7020
Power (Peak)	45 W	50 W	36 W
Precision	float	float	fixed
Frequency	2.3 GHz	835 MHz	667 MHz
RAM	8 GB	2GB VRAM	512 MB
Cores	4	384	N/A
Process Node	22nm	28nm	N/A
Average speedup over CPU	1x	23x	37x

Table 5.4: CPU, GPU, and FPGA used in the evaluation. The specs of the devices are acquired from their respective data sheets. The speedup is normalized to CPU evaluation and training times.

ogy, and tested it using the datasets provided by the UCI repository. The evaluation results show that the accuracy of the traditional RF classifier is comparable (if not larger) to CNN for all datasets that we considered and at the same time RF consumes $\approx$ 95.4%, 91.5%, and 99.4% less energy than SVM$_{RBF}$, MLP and CNN, respectively. The maximum achievable accuracy of the FoG implementation is lower than RF and CNN by 3.2% and 4%, respectively, but FoG classifiers have 42% and 99.7% lower energy than RF and CNN, respectively.

Chapter 6

Conclusion and Future Work

With this work we open a discussion of multi-layer approach to designing energy-efficient hardware and present two major contributions: using equalization to design energy-efficient digital circuits, and algorithmic and architectural approach to the same problem. In this section, we summarize these contributions and provide some commentary on the limitations of our proposed approach and directions for future work.

6.1 Summary of Contributions

6.1.1 Feedback Equalization

Chapter 2 introduced the application of the feedback mechanisms in the conventional digital circuits. In order to enable the feedback equalization we proposed the FEST – a novel gate that consists of a single tap decision feedback equalizer and a Schmitt trigger. We have observed that at nominal frequency we can reduce the energy dissipation by 20% when using FEST. This approach is a step to dynamically tunable circuits, where the higher level system such as instruction-level application (firmware or OS) could potentially control the way individual signal paths (or even gates) behave. Chapter 3 extended chapter 2 by covering the application of feedback mechanisms in the non-conventional circuit topologies, namely pass-transistor logic (PTL). Equalized PTL (E-PTL) uses a differential computational path and modified sense-amplifier to allow ultra low-power, high performance computation. We showed that PTL circuits, as well as communication-inspired design techniques, can provide significant gains in the ultra-low power reliable design and can achieve on average 30% lower EDP for a given frequency. The use of feedback equalization to improve the reliability of digital circuits operating at NTV was also explored. This operating mode, suffering from process and temperature variability, benefits from the feedback equalization. Results show that by using equalization it is possible to reduce the energy dissipation by up to 30%, and improve process variability (σ^2/μ) by up to 5%. This reduces stress placed on the circuit designers who often choose to "over-design" to meet the timing and energy budgets. By relaxing the constraints, figures of merit could be improved at design time.

Limitations

Feedback equalization often requires complete redesign of the system at the circuit level. For example, E-PTL approach is feasible only in the context of pass-transistor logic. The digital circuits today are designed using standard cells and automated tools during the place-and-routing. However, synthesis of PTL circuits is extremely hard to automate or even impossible[1]. This creates a huge limitation for the use of E-PTL, which could be mitigated if such an approach is only used for time/energy critical paths only. FEST circuit also requires custom designed libraries to operate at its peak efficiency. Although this limitation is not as drastic as with the E-PTL, there is some design-time cost associated with the process. In addition to that, FEST approach is a "last-gate" optimizer, which means it can relax the sizing requirements for the last gate only. Although the relaxation propagates back through the logic path, the amount of saving per gate (time or energy) diminishes as the logic path becomes

---

[1] Optimal Pass-Transistor logic circuits from the point of view of the place-and-route are closer in nature to the analog circuits. Although there is currently some effort in developing analog synthesizers, so far those attempts are not very successful (del Mar Hershenson et al., 1998, Moreto et al., 2015).

longer. That means that the best application for the proposed approach is the highly-pipelined, high-frequency, low voltage system, where the individual pipeline stages are extremely short, and at the same time operating voltage is low, thus creating a long rise-/fall-times that FEST could battle with. Note that equalization was shown to be suitable for sub-threshold operation (Zangeneh and Joshi, 2014b). In this regime, the rise- and fall-times of the circuits are extremely slow, and equalization allows much lower EDP with minimal circuit modifications.

6.1.2 Machine Learning Accelerators

In Chapter 4 we opened a discussion of the energy-efficient architectures and algorithms by introducing adaptive classifiers and their energy-efficient implementation by using on-the-fly adaptivity for use in energy-constrained mobile devices. The approach described takes advantage of the "hardness" of the problems and examples. By utilizing multiple classifiers of different complexities (and thus different FoMs), adaptive classifier routes the data path to the most accurate, yet most energy-efficient classifier. Such an approach can sacrifice up to 0.5% accuracy to achieve up to 100× lower power dissipation.

Chapter 5 discussed, in contrast, the possibility to achieve lower energy dissipation through utilization of the additive properties of the ensemble systems by intelligently choosing how many components in the ensemble will participate in the computation.This allows for machine learning algorithms, such as random forest (or any other bagging or boosting algorithm), to become "tunable" and "adaptive". This is done by separating the decision trees into "Groves" that are functionally identical and that conditionally execute the given input. Groves are arranged in a "field" (thus the name Field-of-Groves - FoG), and every Grove can request more computational resources from its neighbors. In this algorithm as more groves process an input, the higher is the certainty in the result is achieved, thus allowing early stopping. Because the Groves are functionally identical, the data path can start at any Grove, thus allowing high level of parallelism in data processing. The tunability of the accelerators is a critical requirement for modern mobile computing, and by utilizing the FoG architecture, a mobile system can trade-off up to 5% accuracy for up to 20× lower energy dissipation when compared to a deep CNN.

Adaptive classifier and FoG can be seen as algorithms with dynamic resource reallocation. In this context computational resources are reallocated by changing the datapath and the use of conditional execution.

Limitations

We would like to note that neither adaptive classifier, nor the FoG architecture claim to have better FoMs than state of the art classifiers such as CNN or even SVM. This means that proposed algorithms cannot in principle achieve neither better accuracies nor better energy-dissipation FoMs than state of the art. However, our proposed algorithms enable higher level of robustness and flexibility by being adaptable at run-time. We assume that it is allowed to sacrifice one of the FoMs in order to win in another (i.e.: trade-off accuracy for energy efficiency).

One of the bigger challenges of the adaptive classifier is the area requirements. The nature of the adaptive classifiers – utilization of several different classifiers in parallel – would require much larger area as compared to a conventional approach. In addition to that, in case the “hardness” in the input data is highly non-linear, the “chooser” function in the proposed approach would have to be more complex, thus negating any possible reduction in energy dissipation. This limitation is avoidable if the proposed architecture is used for specific datasets, where the “hardness” is well defined. Another approach is designing the adaptive system as a system with incrementally increasing complexity (i.e. incrementally expanding Taylor series as an ML kernel) as proposed by other works.

Field of Groves has a slightly different problem – it is most beneficial when large amount of data is being processed in parallel. However, if the input is sequential, the FoG under-utilizes its resources, thus increasing the energy dissipation per computation[2]. Because FoG by definition has lower-or-equal maximum accuracy than RF, usage of FoG in this context becomes a "lose-lose" choice. However, it is worth mentioning that it is possible to design the FoG system to be able to disable some of its parts on demand[3], in case the workload is low.

6.2 Future Research Directions

In this section we provide an overview of the future directions of the completed research.

6.2.1 Feed-Forward Equalization in Pass-Transistor Logic

Because PTL circuits could be very well modeled as communication channels, we can utilize other advancements in the communications theory to further improve the energy-efficiency of the E-PTL. Feed-forward adaptive equalization (FFE) reduces noise and echo, while opening the "eye"[4] of the signal. FFE can encode or predistort the input signal, which reduces the activity factor of a PTL, thus reducing the dynamic power dissipation. FFE deemphasizes low frequency components in order to flatten the channel response. Without it, the input driver would transmit logical 1 as a single pulse, which is dispersed by the circuit loss and distortion. By predistorting the input, the pre-/post-curse ISI could be mitigated thus increasing the operating frequency, reducing power envelope of the system, and reducing the error rate. However, FFE comes with its own limitations – it is extremely hard to design a feed-forward system even with the prior knowledge of the computational channel,

---

[2] Idle parts of the system still dissipate power due to leakage.

[3] Clock and power gating

and should be studied in the future work.

6.2.2 Specialized Architectures for Machine Learning Acceleration

As GPUs serve as specialized processors for image and video processing, machine learning algorithms will require a specialized generic accelerator. Recent advancements such as DianNao family (Chen et al., 2016a) and TPUs (Jouppi et al., 2017) have shown that there is a great potential in the area of specialized processors. In particular, it is possible to identify the set of instructions that are frequently used in the machine learning algorithms, and accelerate them. In correlation to the FoG architecture, such a mechanism would be composed of a highly distributed set of cores, with each core representing a "generic" Grove. In this context the cores could be on the same die, same package, or same board – the effects of long communications between cores must be extensively studied.

We also believe that specialized architectures should take advantage of the modern advancements in the field of neuromorphic devices such as "memristors". Memristors are one of the four fundamental circuit elements. They are memory resistors in that their resistance can be altered depending on the magnitude of the voltage applied. Likewise, when no voltage is applied across a memristor, the most recent resistance value is retained. Memristors have similar behavior to biological synapses, and as such, have been frequently utilized to implement neuromorphic systems. The design of specialized architectures using such devices would greatly improve both computational performance as well as energy efficiency. We propose exploring the use of memristors in the design of the adaptive classifiers and FoG.

---

[4] An "Eye" of a signal, "eye pattern" or "eye diagram", is an oscilloscope view of a digital signal which is repetitively scanned to evaluate the noise and inter-symbol interference. The opening of the eye in this context is effectively horizontal and vertical noise margins.

6.2.3 Bridging the gap between Hardware and Machine Learning

The disparity and lack of effective communication between machine learning (ML) and hardware (HW) communities, caused ML researchers to often ignore the hardware constraints such as energy or computational performance. At the same time, HW professionals have limited view of the advancements in the ML field. We believe that both communities need to join the efforts in developing specialized hardware for machine learning.

One of the main problems that needs to be jointly resolved in the nearest future is finding the balance between computation and communication in the mobile systems. Processing the data locally is expensive in terms of energy dissipation, but communication with the “cloud” is not free either. Depending on the accuracy requirements, battery status, available resources, etc. an intelligent decision is supposed to be made on how much data to compute locally. Both adaptive classifier and FoG are suitable candidates to be the basis for this research direction.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Agarwal, M., Paul, B., Zhang, M., and Mitra, S. (2007). Circuit Failure Prediction and Its Application to Transistor Aging. In *25th IEEE VLSI Test Symposium*, pages 277–286.

Aiello, L. C. (1997). Brains and guts in human evolution: the expensive tissue hypothesis. *Brazilian Journal of Genetics*, 20.

Akturk, I., Kim, N. S., and Karpuzcu, U. R. (2015). Decoupled Control and Data Processing for Approximate Near-Threshold Voltage Computing. *IEEE Journal of Microarchitectures*, 35(4):70–78.

Al Maashri, A., Debole, M., Cotter, M., Chandramoorthy, N., Xiao, Y., Narayanan, V., and Chakrabarti, C. (2012). Accelerating neuromorphic vision algorithms for recognition. In *Proceedings of the 49th Annual Design Automation Conference (DAC), 2012.*, pages 579–584. IEEE/ACM.

Albericio, J., Judd, P., Hetherington, T., Aamodt, T., Jerger, N. E., and Moshovos, A. (2016). Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing. In *Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 1–13.

Avestruz, A. T., Santa, W., Carlson, D., Jensen, R., Stanslaski, S., Helfenstine, A., and Denison, T. (2008). A 5 *mu* W/Channel Spectral Analysis IC for Chronic Bidirectional Brain-Machine Interfaces. *IEEE Journal of Solid-State Circuits (JSSC)*, 43(12):3006–3024.

Avirneni, N., Subramanian, V., and Somani, A. (2009). Low overhead Soft Error Mitigation techniques for high-performance and aggressive systems. In *IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*, pages 185–194.

Bachrach, J., Vo, H., Richards, B., Lee, Y., Waterman, A., Avižienis, R., Wawrzynek, J., and Asanović, K. (2012). Chisel: constructing hardware in a scala embedded language. In *Proceeding of the 49th ACM Design Automation Conference (DAC), 2012.*, pages 1216–1225. IEEE/ACM.

Baker, M. W. and Sarpeshkar, R. (2006). Low-Power Single-Loop and Dual-Loop AGCs for Bionic Ears. *IEEE Journal of Solid-State Circuits (JSSC)*, 41(9):1983–1996.

Baker, R. J. (2004). *CMOS Circuit Design, Layout, and Simulation, Second Edition.* Wiley-IEEE Press.

Belfiore, C. and Park, J.H., J. (1979). Decision feedback equalization. *Proceedings of the IEEE*, 67(8):1143–1156.

Bell, C. G., Chen, R., and Rege, S. (1972). Effect of technology on near term computer structures. *Computer*, 5(2):29–38.

Bishnoi, R., Oboril, F., and Tahoori, M. B. (2017). Design of Defect and Fault-Tolerant Nonvolatile Spintronic Flip-Flops. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(4):1421–1432.

Bojnordi, M. N. and Ipek, E. (2016). Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–13.

Borkar, S., Karnik, T., and De, V. (2004). Design and Reliability Challenges in Nanometer Technologies. In *Proceedings of the 41st Annual Design Automation Conference (DAC)*, pages 75–75, New York, NY, USA. ACM.

Bowman, K., Tschanz, J., Kim, N. S., Lee, J., Wilkerson, C., Lu, S., Karnik, T., and De, V. (2009). Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance. *IEEE Journal of Solid-State Circuits (JSSC)*, 44(1):49–63.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Calhoun, B. and Chandrakasan, A. (2004). Characterizing and modeling minimum energy operation for subthreshold circuits. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, pages 90–95.

Cao, Y. K. (2009). What is Predictive Technology Model (PTM)? *SIGDA Newsletter*, 39(3):1–1.

Chakradhar, S., Sankaradas, M., Jakkula, V., and Cadambi, S. (2010). A Dynamically Configurable Coprocessor for Convolutional Neural Networks. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)*, pages 247–257, New York, NY, USA. ACM.

Chandrakasan, A. P. (1996). Ultra low power digital signal processing. In *Proceedings of 9th International Conference on VLSI Design*, pages 352–357.

Chandrakasan, A. P. and Brodersen, R. W. (1995). Minimizing power consumption in digital CMOS circuits. *Proceedings of the IEEE*, 83(4):498–523.

Chen, C.-H., Kim, Y., Zhang, Z., Blaauw, D., Sylvester, D., Naeimi, H., and Sandhu, S. (2011). A confidence-driven model for error-resilient computing. In *Proceedings of the Design, Automation, and Test in Europe Conference Exhibition (DATE)*, pages 1–6.

Chen, Y., Chen, T., Xu, Z., Sun, N., and Temam, O. (2016a). DianNao Family: Energy-Efficient Hardware Accelerators for Machine Learning. *Communications of the ACM*, 59(11):105–112.

Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N., and Temam, O. (2014). Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual International Symposium on Microarchitecture (MICRO)*, pages 609–622. IEEE/ACM.

Chen, Y. H., Emer, J., and Sze, V. (2016b). Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. In *2016 ACM / IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 367–379.

Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y., and Xie, Y. (2016). PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory. In *43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 27–39. IEEE/ACM.

Chippa, V., Raghunathan, A., Roy, K., and Chakradhar, S. (2011). Dynamic effort scaling: Managing the quality-efficiency tradeoff. In *Proceedings of the 48th Annual Design Automation Conference (DAC)*, pages 603–608. IEEE/ACM.

Chippa, V. K., Mohapatra, D., Raghunathan, A., Roy, K., and Chakradhar, S. T. (2010). Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In *Proceedings of the 47th Annual Design Automation Conference (DAC)*, pages 555–560. IEEE.

Chung, J. and Shin, T. (2016). Simplifying deep neural networks for neuromorphic architectures. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*, pages 1–6. IEEE/ACM.

Cinar, Y. G., Mirisaee, H., Goswami, P., Gaussier, É., Aït-Bachir, A., and Strijov, V. (2017). Time Series Forecasting using RNNs: an Extended Attention Mechanism to Model Periods and Handle Missing Values. *CoRR*, abs/1703.10089.

Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, High Performance Convolutional Neural Networks for Image Classification. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1237–1242. AAAI Press.

ComScore Report 2016 (2016). comScore: The 2016 U.S. Mobile App Report. http://www.comscore.com/Insights/Presentations-and-Whitepapers/2016/The-2016-US-Mobile-App-Report.

Constantinescu, C. (2003). Trends and challenges in VLSI circuit reliability. *Proceedings of the Microarchitecture (MICRO)*, 23(4):14–19.

CVX Research, Inc. (2012). CVX: Matlab software for disciplined convex programming, version 2.0 beta. http://cvxr.com/cvx.

Das, S., Pant, S., Roberts, D., Lee, S., Blaauw, D., Austin, T., Mudge, T., and Flautner, K. (2005). A self-tuning DVS processor using delay-error detection and correction. In *Symposium on VLSI Circuits, Digest of Technical Papers*, pages 258–261.

Das, S., Tokunaga, C., Pant, S., Ma, W.-H., Kalaiselvan, S., Lai, K., Bull, D., and Blaauw, D. (2009). RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance. *Journal of Solid-State Circuits*, 44(1):32–48.

del Mar Hershenson, M., Boyd, S. P., and Lee, T. H. (1998). GPCAD: a tool for CMOS op-amp synthesis. In *International Conference on Computer-Aided Design (ICCAD), Digest of Technical Papers*, pages 296–303.

Dreslinski, R., Wieckowski, M., Blaauw, D., Sylvester, D., and Mudge, T. (2010). Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits. *Proceedings of the IEEE*, 98(2):253–266.

Du, Z., Ben-Dayan Rubin, D. D., Chen, Y., He, L., Chen, T., Zhang, L., Wu, C., and Temam, O. (2015). Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches. In *Proceedings of the 48th International Symposium on Microarchitecture (MICRO)*, pages 494–507. IEEE/ACM.

Düben, P., Schlachter, J., Parishkrati, Yenugula, S., Augustine, J., Enz, C., Palem, K., and Palmer, T. N. (2015). Opportunities for Energy Efficient Computing: A Study of Inexact General Purpose Processors for High-performance and Big-data Applications. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 764–769, San Jose, CA, USA. EDA Consortium.

Enz, C. C. and Vittoz, E. A. (2006). *Charge-Based MOS Transistor Modeling: The EKV Model for Low-Power and RF IC Design*. Wiley.

Ernst, D., Das, S., Lee, S., Blaauw, D., Austin, T., Mudge, T., Kim, N. S., and Flautner, K. (2004). Razor: circuit-level correction of timing errors for low-power operation. *24th Annual International Symposium on Microarchitecture (MICRO)*, 24(6):10–20.

Esmaeilzadeh, H., Sampson, A., Ceze, L., and Burger, D. (2012). Architecture support for disciplined approximate programming. In *Journal of SIGPLAN Notices*, volume 47, pages 301–312. ACM.

Fuller, S. and Millett, L. (2011). Computing Performance: Game Over or Next Level? *Computer*, 44(1):31–38.

Gao, T. and Koller, D. (2011). Active Classification based on Value of Classifier. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1062–1070. Curran Associates, Inc.

Gautschi, M., Schiavone, P. D., Traber, A., Loi, I., Pullini, A., Rossi, D., Flamand, E., Gürkaynak, F. K., and Benini, L. (2017). Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–14.

Gielen, G., De Wit, P., Maricau, E., Loeckx, J., Martín-Martínez, J., Kaczer, B., Groeseneken, G., Rodríguez, R., and Nafría, M. (2008). Emerging Yield and Reliability Challenges in Nanometer CMOS Technologies. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, pages 1322–1327, New York, NY, USA. ACM.

Godfrey, L. B. and Gashler, M. S. (2017). Neural Decomposition of Time-Series Data for Effective Generalization. *CoRR*, abs/1705.09137.

Gómez, D. and Rojas, A. (2016). An Empirical Overview of the No Free Lunch Theorem and Its Effect on Real-World Machine Learning Classification. *Neural Computation*, 28(1):216–228.

Grant, M. and Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, pages 95–110. Springer-Verlag Limited.

Grigorian, B., Farahpour, N., and Reinman, G. (2015). BRAINIAC: Bringing reliable accuracy into neurally-implemented approximate computing. In *21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 615–626. IEEE.

Hamdioui, S., Xie, L., Nguyen, H. A. D., Taouil, M., Bertels, K., Corporaal, H., Jiao, H., Catthoor, F., Wouters, D., Eike, L., and van Lunteren, J. (2015). Memristor based computation-in-memory architecture for data-intensive applications. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 1718–1725. EDA Consortium.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., and Dally, W. J. (2016). EIE: Efficient Inference Engine on Compressed Deep Neural Network. *CoRR*, abs/1602.01528.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer Inc., New York, NY, USA.

Hilbert, M. and Lopez, P. (2011). The world's technological capacity to store, communicate, and compute information. *Science*, 332(6025):60–65.

Hu, M., Strachan, J. P., Li, Z., Grafals, E. M., Davila, N., Graves, C., Lam, S., Ge, N., Yang, J. J., and Williams, R. S. (2016). Dot-product engine for neuromorphic computing: programming 1T1M crossbar to accelerate matrix-vector multiplication. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*, pages 1–6. IEEE/ACM.

Iannazzo, M., Muzzo, V. L., Rodriguez, S., Rusu, A., Lemme, M., and Alarcón, E. (2015). Design exploration of graphene-FET based ring-oscillator circuits: A test-bench for large-signal compact models. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2716–2719.

IDC and EMC2 (2012). The Digital Universe in 2020. *7th annual study of the digital universe.*

IDC and EMC2 (2013). The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. *7th Annual Study of the Digital Universe.*

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093.*

Joshi, A., Chen, C., Takhirov, Z., and Nazer, B. (2012). A multi-layer approach to green computing: Designing energy-efficient digital circuits and manycore architectures. In *International Green Computing Conference (IGCC)*, pages 1–3.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-l., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T. V., Gottipati, R., Gulland, W., Hagmann, R., Ho, C. R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., and Yoon, D. H. (2017). In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12, New York, NY, USA. ACM.

Judd, P., Albericio, J., Hetherington, T., Aamodt, T. M., and Moshovos, A. (2016). Stripes: Bit-serial deep neural network computing. In *49th Annual International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE/ACM.

Kahng, A., Kang, S., Kumar, R., and Sartori, J. (2010). Slack redistribution for graceful degradation under voltage overscaling. In *15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 825–831.

Karl, E., Singh, P., Blaauw, D., and Sylvester, D. (2008). Compact In-Situ Sensors for Monitoring Negative-Bias-Temperature-Instability Effect and Oxide Degradation. In *IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers.*, pages 410–623.

Karpuzcu, U., Sinkar, A., Kim, N. S., and Torrellas, J. (2013). EnergySmart: Toward energy-efficient manycores for Near-Threshold Computing. In *19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 542–553.

Kaul, H., Anders, M., Mathew, S., Hsu, S., Agarwal, A., Sheikh, F., Krishnamurthy, R., and Borkar, S. (2012). A 1.45 ghz 52-to-162gflops/w variable-precision floating-point fused multiply-add unit with certainty tracking in 32nm cmos. In *International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 182–184. IEEE.

Keane, J., Wang, X., Persaud, D., and Kim, C. (2010). An All-In-One Silicon Odometer for Separately Monitoring HCI, BTI, and TDDB. *Journal of Solid-State Circuits (JSSC)*, 45(4):817–829.

Khan, M. A., Mohanty, S. P., and Kougianos, E. (2014). Statistical process variation analysis of a graphene FET based LC-VCO for WLAN applications. In *15th International Symposium on Quality Electronic Design (ISQED)*, pages 569–574.

Kiamehr, S., Ebrahimi, M., Golanbari, M. S., and Tahoori, M. B. (2017). Temperature-Aware Dynamic Voltage Scaling to Improve Energy Efficiency of Near-Threshold Computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–10.

Kim, K., Kim, J., Yu, J., Seo, J., Lee, J., and Choi, K. (2016). Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*, page 124. IEEE/ACM.

Kranz, M., Möller, A., Hammerla, N., Diewald, S., Plötz, T., Olivier, P., and Roalter, L. (2013). The mobile fitness coach: Towards individualized skill assessment using personalized mobile devices. *Pervasive and Mobile Computing*, 9(2):203 – 215. Special Section: Mobile Interactions with the Real World.

Kusner, M., Chen, W., Zhou, Q., Xu, Z. E., Weinberger, K., and Chen, Y. (2014). Feature-Cost Sensitive Learning with Submodular Trees of Classifiers. In *Association for the Advancement of Artificial Intelligence Conference (AAAI)*.

Kwong, J. and Chandrakasan, A. (2006). Variation-Driven Device Sizing for Minimum Energy Sub-threshold Circuits. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, pages 8–13.

Lala, P. K. (2001). *Self-Checking and Fault-Tolerant Digital Design*. Academic Press, San Diego, CA.

Lane, N., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., and Campbell, A. (2010). A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150.

Lee, H., Pham, P., Largman, Y., and Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1096–1104. Curran Associates, Inc.

Lee, T. (2004). *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge University Press.

Leem, L., Cho, H., Bau, J., Jacobson, Q., and Mitra, S. (2010). ERSA: Error Resilient System Architecture for probabilistic applications. In *Proceedings of the*

Design, Automation, and Test in Europe Conference and Exhibition (DATE), pages 1560–1565.

Lin, Y.-T., Tsai, P.-Y., and Chiueh, T.-D. (2005). Low-power variable-length fast Fourier transform processor. *IEE Proceedings-Computers and Digital Techniques*, 152(4):499–506.

Liu, S., Du, Z., Tao, J., Han, D., Luo, T., Xie, Y., Chen, Y., and Chen, T. (2016). Cambricon: An Instruction Set Architecture for Neural Networks. In *Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 393–405.

Liu, X., Mao, M., Liu, B., Li, H., Chen, Y., Li, B., Wang, Y., Jiang, H., Barnell, M., Wu, Q., and Yang, J. (2015). RENO: A High-efficient Reconfigurable Neuromorphic Computing Accelerator Design. In *Proceedings of the 52nd Annual Design Automation Conference (DAC), 2015*, pages 66:1–66:6, New York, NY, USA. IEEE/ACM.

Lu, Y. and Kazmierski, T. J. (2016). Error-free near-threshold adiabatic CMOS logic in presence of process variation. In *2016 Forum on Specification and Design Languages (FDL)*, pages 1–5.

Mandal, S., Zhak, S. M., and Sarpeshkar, R. (2009). A Bio-Inspired Active Radio-Frequency Silicon Cochlea. *IEEE Journal of Solid-State Circuits (JSSC)*, 44(6):1814–1828.

Mathew, J., Singh, J., Taleb, A., and Pradhan, D. (2008). Fault Tolerant Reversible Finite Field Arithmetic Circuits. In *14th IEEE International On-Line Testing Symposium (IOLTS)*, pages 188–189.

Mead, C. (1989). *Analog VLSI and Neural Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Mohapatra, D., Chippa, V., Raghunathan, A., and Roy, K. (2011). Design of voltage-scalable meta-functions for approximate computing. In *Proceedings of the Design, Automation, and Test in Europe Conference Exhibition (DATE)*, pages 1–6.

Moreto, R. A. L., Thomaz, C. E., Gimenez, S. P., and Rotondaro, A. L. P. (2015). From architecture to manufacturing: An accurate framework for optimal OTA design. In *Latin America Congress on Computational Intelligence (LA-CCI)*, pages 1–6.

Naeimi, H. and DeHon, A. (2008). Fault-tolerant sub-lithographic design with rollback recovery. *Nanotechnology*, 19(11):115708.

Nan, F., Wang, J., and Saligrama, V. (2015). Feature-budgeted Random Forest. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*, volume 37, pages 1983–1991. JMLR.org.

Narayanan, S., Sartori, J., Kumar, R., and Jones, D. (2010). Scalable stochastic processors. In *Proceedings of the Design, Automation, and Test in Europe Conference Exhibition (DATE)*, pages 335–338.

Nere, A., Hashmi, A., Lipasti, M., and Tononi, G. (2013). Bridging the semantic gap: Emulating biological neuronal behaviors with simple digital neurons. In *19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 472–483. IEEE.

Nowatzki, T., Gangadhar, V., and Sankaralingam, K. (2015). Exploring the Potential of Heterogeneous Von Neumann/Dataflow Execution Models. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)*, pages 298–310, New York, NY, USA. ACM.

Olah, C. (2015). Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Ordóñez, F. and Roggen, D. (2016). Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors*, 16(1):115.

Pan, X. and Teodorescu, R. (2014). Using STT-RAM to enable energy-efficient near-threshold chip multiprocessors. In *2014 23rd International Conference on Parallel Architecture and Compilation Techniques (PACT)*, pages 485–486.

Panda, P., Sengupta, A., and Roy, K. (2016a). Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2017 Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 475–480. IEEE.

Panda, P., Sengupta, A., Sarwar, S. S., Srinivasan, G., Venkataramani, S., Raghunathan, A., and Roy, K. (2016b). Cross-layer approximations for neuromorphic computing: from devices to circuits and systems. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*, page 98. ACM.

Panda, P., Sengupta, A., Venkataramani, S., Raghunathan, A., and Roy, K. (2015). Object Detection using Semantic Decomposition for Energy-Efficient Neural Computing. *arXiv preprint arXiv:1509.08970*.

Panda, P., Venkataramani, S., Sengupta, A., Raghunathan, A., and Roy, K. (2017). Energy-Efficient Object Detection Using Semantic Decomposition. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–5.

Park, E., Kim, D., Kim, S., Kim, Y.-D., Kim, G., Yoon, S., and Yoo, S. (2015a). Big/little deep neural network for ultra low power inference. In *Proceedings of the 10th International Conference on Hardware/Software Co-Design and System Synthesis*, pages 124–132. IEEE Press.

Park, M. S., Kestur, S., Sabarad, J., Narayanan, V., and Irwin, M. J. (2012). An FPGA-based accelerator for cortical object classification. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 691–696. IEEE.

Park, S., Bong, K., Shin, D., Lee, J., Choi, S., and Yoo, H. J. (2015b). A 1.93TOPS/W scalable deep learning/inference processor with tetra-parallel MIMD architecture for big-data applications. In *IEEE International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers*, pages 1–3.

Paul, B., Fujita, S., Okajima, M., and Lee, T. (2006). Modeling and analysis of circuit performance of ballistic CNFET. In *Proceedings of the 43rd Annual Design Automation Conference (DAC)*, pages 717–722.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pinckney, N., Jeloka, S., Dreslinski, R., Mudge, T., Sylvester, D., Blaauw, D., Shifren, L., Cline, B., and Sinha, S. (2017). Impact of FinFET on Near-Threshold Voltage Scalability. *IEEE Design Test*, 34(2):31–38.

Poolakkaparambil, M., Mathew, J., Jabir, A., Pradhan, D., and Mohanty, S. (2011). BCH code based multiple bit error correction in finite field multiplier circuits. In *12th International Symposium on Quality Electronic Design (ISQED)*, pages 1–6.

Qi, Z. and Stan, M. R. (2008). NBTI Resilient Circuits Using Adaptive Body Biasing. In *Proceedings of the 18th ACM Great Lakes Symposium on VLSI*, pages 285–290, New York, NY, USA. ACM.

Rabaey, J. (2009). *Low Power Design Essentials*. Springer Publishing Company, Incorporated.

Rahman, A., Lee, J., and Choi, K. (2016). Efficient FPGA acceleration of Convolutional Neural Networks using logical-3D compute array. In *Proceedings of the Design, Automation Test in Europe Conference & Exhibition (DATE)*, pages 1393–1398.

Rashidi, P. and Cook, D. J. (2009). Keeping the Resident in the Loop: Adapting the Smart Home to the User. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(5):949–959.

Richter, S., Schulte-Braucks, C., Knoll, L., Luong, G., Schafer, A., Trellenkamp, S., Zhao, Q.-T., and Mantl, S. (2014). Experimental demonstration of inverter and NAND operation in p-TFET logic at ultra-low supply voltages down to VDD = 0.15 V. In *72nd Annual Device Research Conference (DRC)*, pages 23–24.

Roy, S., Liu, D., Um, J., and Pan, D. (2015). OSFA: A new paradigm of gate-sizing for power/performance optimizations under multiple operating conditions. In *Proceedings of the 52nd Design Automation Conference (DAC)*, pages 1–6.

Sampson, A. (2015). *Hardware and software for approximate computing.* PhD thesis, University of Washington.

Sarpeshkar, R. (1998). Analog Versus Digital: Extrapolating from Electronics to Neurobiology. *Neural Computation*, 10(7):1601–1638.

Sarpeshkar, R. (2010). *Ultra Low Power Bioelectronics: Fundamentals, Biomedical Applications, and Bio-Inspired Systems.* Cambridge University Press.

Sarpeshkar, R., Salthouse, C., Sit, J.-J., Baker, M. W., Zhak, S. M., Lu, T. K. T., Turicchia, L., and Balster, S. (2005). An ultra-low-power programmable analog bionic ear processor. *IEEE Transactions on Biomedical Engineering*, 52(4):711–727.

Sarpeshkar, R., Watts, L., and Mead, C. (1992). Refractory neuron circuits. In *Computation Neural Systems Technical Reports (CNS-TR-92-8).* California Institute of Technology, Pasadena, CA.

Sartori, J., Sloan, J., and Kumar, R. (2009). Fluid NMR – Performing power/reliability tradeoffs for applications with error tolerance. *SOSP Workshop on Power Aware Computing and Systems.*

Seok, M., Chen, G., Hanson, S., Wieckowski, M., Blaauw, D., and Sylvester, D. (2011). CAS-FEST 2010: Mitigating Variability in Near-Threshold Computing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1(1):42–49.

Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., Williams, R. S., and Srikumar, V. (2016). ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*, pages 14–26. IEEE Press.

Shao, Y. S., Reagen, B., Wei, G.-Y., and Brooks, D. (2014). Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures. In *Proceedings of the 41st International Symposium on Computer Architecture (ISCA)*, pages 97–108. IEEE.

Sidiroglou-Douskos, S., Misailovic, S., Hoffmann, H., and Rinard, M. (2011). Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 124–134. ACM.

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, pages 958–, Washington, DC, USA. IEEE Computer Society.

Song, L., Qian, X., Li, H., and Chen, Y. (2017). PipeLayer: A pipelined ReRAM-based accelerator for deep learning. In *International Symposium on High Performance Computer Architecture (HPCA)*, pages 541–552. IEEE.

Sowjanya, K., Singhal, A., and Choudhary, C. (2015). MobDBTest: A machine learning based system for predicting diabetes risk using mobile devices. In *IEEE International Advance Computing Conference (IACC)*, pages 397–402.

Sridhara, S., Balamurugan, G., and Shanbhag, N. (2008). Joint Equalization and Coding for On-Chip Bus Communication. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(3):314 –318.

Tagliavini, G., Marongiu, A., Rossi, D., and Benini, L. (2016). Always-on motion detection with application-level error control on a near-threshold approximate computing platform. In *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 552–555.

Takhirov, Z. and Huang, Y. (2014). Mixed-Signal Modeling and Verification. *Analog Devices, Inc. (Internal Tutorials).*

Takhirov, Z., Nazer, B., and Joshi, A. (2011). A preliminary look at error avoidance in digital logic via feedback equalization. In *49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1390–1391. IEEE.

Takhirov, Z., Nazer, B., and Joshi, A. (2012). Error mitigation in digital logic using a feedback equalization with schmitt trigger (FEST) circuit. In *13th International Symposium on Quality Electronic Design (ISQED)*, pages 312–319.

Takhirov, Z., Nazer, B., and Joshi, A. (2013). Energy-efficient pass-transistor-logic using decision feedback equalization. In *IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pages 335–340.

Takhirov, Z., Wang, J., Louis, M. S., Saligrama, V., and Joshi, A. (2017). Field of Groves: An Energy-Efficient Random Forest. *arXiv preprint arXiv:1704.02978*.

Takhirov, Z., Wang, J., Saligrama, V., and Joshi, A. (2016). Energy-Efficient Adaptive Classifier Design for Mobile Systems. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design (ISLPED)*, pages 52–57, New York, NY, USA. ACM.

Tang, T., Xia, L., Li, B., Luo, R., Chen, Y., Wang, Y., and Yang, H. (2015). Spiking neural network with rram: Can we use it for real-world application? In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*, pages 860–865. EDA Consortium.

Teodoro, G., Sachetto, R., Sertel, O., Gurcan, M. N., Meira, W., Catalyurek, U., and Ferreira, R. (2009). Coordinating the use of GPU and CPU for improving performance of compute intensive applications. In *IEEE International Conference on Cluster Computing and Workshops*, pages 1–10.

Trapeznikov, K. and Saligrama, V. (2013). Supervised Sequential Classification Under Budget Constraints. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 581–589.

TSensors, S. (2013). TSensors Roadmap v1.1. *TSensors Summit*.

Tsividis, Y. P., Gopinathan, V., and Toth, L. (1990). Companding in signal processing. *Electronics Letters*, 26(17):1331–1332.

Turkyilmaz, O., Clermidy, F., Amarù, L. G., Gaillardon, P. E., and Micheli, G. D. (2013). Self-checking ripple-carry adder with Ambipolar Silicon NanoWire FET. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2127–2130.

UC Irvine Machine Leanring Repository (2017). Machine Learning Repository. http://archive.ics.uci.edu/ml/. Accessed: 2015-11-23.

Unar, J., Seng, W. C., and Abbasi, A. (2014). A review of biometric technology along with trends and prospects. *Pattern Recognition*, 47(8):2673 – 2688.

Valadimas, S., Tsiatouhas, Y., Arapoyanni, A., and Xarchakos, P. (2013). Effective Timing Error Tolerance in Flip-Flop Based Core Designs. *Journal of Electronic Testing*, 29(6):795–804.

Velasquez, A. and Jha, S. K. (2014). Parallel computing using memristive crossbar networks: Nullifying the processor-memory bottleneck. In *9th International Design and Test Symposium (IDT)*, pages 147–152.

Venkataramani, S., Chippa, V. K., Chakradhar, S. T., Roy, K., and Raghunathan, A. (2013). Quality programmable vector processors for approximate computing. In *Proceedings of the 46th Annual International Symposium on Microarchitecture (MICRO)*, pages 1–12. ACM.

Venkataramani, S., Raghunathan, A., Liu, J., and Shoaib, M. (2015a). Scalable-effort Classifiers for Energy-efficient Machine Learning. In *Proceedings of the 52nd Annual Design Automation Conference (DAC)*, pages 67:1–67:6, New York, NY, USA. ACM.

Venkataramani, S., Raghunathan, A., Liu, J., and Shoaib, M. (2015b). Scalable-effort classifiers for energy-efficient machine learning. In *Proceedings of the 52nd Annual Design Automation Conference (DAC)*, page 67. ACM.

Wang, A. and Chandrakasan, A. (2004). A 180mV FFT processor using subthreshold circuit techniques. In *IEEE International Digest of Technical Papers on Solid-State Circuits Conference (ISSCC)*, pages 292–529 Vol.1.

Wang, J., Bolukbasi, T., Trapeznikov, K., and Saligrama, V. (2014a). Model selection by linear programming. In *European Conference on Computer Vision*, pages 647–662. Springer.

Wang, J., Trapeznikov, K., and Saligrama, V. (2014b). An LP for Sequential Learning Under Budgets. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 987–995.

Wang, J., Trapeznikov, K., and Saligrama, V. (2015). Efficient Learning by Directed Acyclic Graph For Resource Constrained Prediction. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 2152–2160.

Wang, S., Chen, C., Xiang, X. Y., and Meng, J. Y. (2017). A Variation-Tolerant Near-Threshold Processor With Instruction-Level Error Correction. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–14.

Whaley, R. C. and Petitet, A. (2005). Minimizing development and maintenance costs in supporting persistently optimized BLAS. *Software: Practice and Experience*, 35(2):101–121. `http://www.cs.utsa.edu/~whaley/papers/spercw04.ps`.

Williams, R. S. (2017). What's Next? [The end of Moore's law]. *Computing in Science Engineering*, 19(2):7–13.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Wu, G. (2015). Always Connected: Billions of Connected Nanosystems. In *Workshop on Rebooting the IT Revolution.*

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144.

Xia, L., Tang, T., Huangfu, W., Cheng, M., Yin, X., Li, B., Wang, Y., and Yang, H. (2016). Switched by input: Power efficient structure for RRAM-based convolutional neural network. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*, page 125. IEEE/ACM.

Xu, Z., Kusner, M., Weinberger, K., and Chen, M. (2013). Cost-sensitive tree of classifiers. In *International Conference on Machine Learning*, pages 133–141.

Xu, Z., Weinberger, K., and Chapelle, O. (2012). The greedy miser: Learning under test-time budgets. *arXiv preprint arXiv:1206.6451.*

Yazdanbakhsh, A., Park, J., Sharma, H., Lotfi-Kamran, P., and Esmaeilzadeh, H. (2015). Neural acceleration for gpu throughput processors. In *48th International Symposium on Microarchitecture (MICRO)*, pages 482–493. IEEE/ACM.

Yazeer, M. J., Za'bah, N. F., and Alam, A. H. M. Z. (2016). Triangular Shaped Silicon Nanowire FET Characterization Using COMSOL Multiphysics. In *International Conference on Computer and Communication Engineering (ICCCE)*, pages 494–498.

Zangeneh, M. and Joshi, A. (2014a). Design and Optimization of Nonvolatile Multibit 1T1R Resistive RAM. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(8):1815–1828.

Zangeneh, M. and Joshi, A. (2014b). Sub-threshold logic circuit design using feedback equalization. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1–6.

Zhai, B., Blaauw, D., Sylvester, D., and Flautner, K. (2004). Theoretical and practical limits of dynamic voltage scaling. In *Design Automation Conference, 2004. Proceedings. 41st*, pages 868–873.

Zhang, S., Du, Z., Zhang, L., Lan, H., Liu, S., Li, L., Guo, Q., Chen, T., and Chen, Y. (2016). Cambricon-X: An accelerator for sparse neural networks. In *49th Annual International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE/ACM.

CURRICULUM VITAE

Zafar Takhirov

Email: cc.rafaz@zafar.cc
Phone: 617-826-9666

Education

– **Ph.D., Boston University** **01/2018**
Computer Engineering
Advisor: Professor Ajay J. Joshi
Dissertation Title: Designing Energy-efficient Computing Systems Using Equalization And Machine Learning
– **M.Sc., Boston University** **05/2012**
Computer Engineering
Thesis: Equalization in on-Chip Many Core Interconnects
– **Specialist, Russian-Tajik Slavic University** **05/2008**
Linguistics
Thesis: Slang, Neologisms, and Profanity in German Languages

Work and Research Experience

– **Didi Research America, LLC.** **Mountain View, CA**
Software Developer II / Research Scientist 05/2017 – Present
– **Boston University** **Boston, MA**
Research Assistant 09/2011 – 05/2017
– **Zentist.IO** (formerly Avicennas Group, Inc.) **New York, NY**
Chief Technology Officer 05/2015 – 02/2016
– **Analog Devices, Inc.** **San Jose, CA**
Mixed-Signal Verification Engineer 02/2014 – 02/2015
– **Analog Devices, Inc.** **San Jose, CA**
Mixed-Signal Design Engineering Intern 05/2013 – 12/2013

Publications

– Takhirov, Z., Wang, J., Louis, M. S., Saligrama, V. and Joshi, A. “Field of Groves: An Energy-Efficient Random Forest”, *Design Automation Conference 2017 Work in Progress Section*, *arXiV preprint*, 04/11/2017
– Takhirov, Z., Wang, J., Saligrama, V. and Joshi, A. “Energy-Efficient Adaptive Classifier Design for Mobile Systems”, *2016 Proceedings of the 2016 International Symposium on Low Power Electronics and Design*
– Takhirov, Z., Nazer, B., and Joshi, A. “Energy-efficient pass-transistor-logic using decision feedback equalization”, *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on*
– Joshi, A., Chen, C., Takhirov, Z., Nazer, B. “A multi-layer approach to green computing: Designing energy-efficient digital circuits and manycore architectures”, *Green Computing Conference (IGCC), 2012 International*
– Takhirov, Z., Nazer, B., and Joshi, A. “Error mitigation in digital logic using a feedback equalization with schmitt trigger (FEST) circuit”, *Quality Electronic Design (ISQED), 2012 13th International Symposium on*
– Takhirov, Z., Nazer, B., and Joshi, A. “A preliminary look at error avoidance in digital logic via feedback equalization”, *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*