FINE-GRAINED POWER GATED MULTIPLIER WITH ONLINE CALIBRATION FOR MEDICAL IOT DEVICES

A Thesis

by

SWATHI CHANGALARAYAPPA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Eun Jung Kim
Co-Chair of Committee,	Peng Li
Committee Member,	Jeyavijayan Rajendran
Head of Department,	Miroslav M. Begovic

May 2019

Major Subject: Computer Engineering

Copyright 2019 Swathi Changalarayappa

ABSTRACT

With intensive research in the fields of machine learning and neural networks to improve its accuracy comes the responsibility to realize feasible hardware solutions on battery powered IoT devices. This work presents a study of analysis of power hungry computations and a fine-grained power gated multiplier design using approximation, that aims at energy optimization exploiting error resilience of these applications. We use truncation to reduce cycles and low power techniques to reduce power, thus achieving a 2-fold energy reduction. We use wearable IoT devices for medical purposes as our case study and show the generality of our work across applications. Our work performs similar to, or better than the latest work in the field and is a more generic implementation. We propose an online calibration mechanism to determine the approximation rate dynamically that maximizes energy optimization with very low accuracy loss. Our method uses a clustering solution to pre-determine the output label in a majority of cases, without having to need an inference model, thus further reducing energy. We achieve 78% energy improvement compared to a baseline implementation with just 0.46% accuracy loss across benchmarks.

DEDICATION

To my parents, for their unconditional love and support.

ACKNOWLEDGMENTS

I would like to express my gratitude to my committee chair and my advisor, Dr. E J Kim; Co-Chair, Dr. Peng Li and my committee members, Dr. Jeyavijayan Rajendran, for their guidance and support throughout the course of this research.

I would like to express my sincere thanks to Kyung Hoon for his guidance at every step. Thanks to all the members of HPC Lab for their constant support. Thanks to my department, Electrical and Computer Engineering faculty and staff for the immense knowledge I gained through the courses, that helped me with my research. Lastly, but with utmost importance, I am grateful to the support extended by my friends and family throughout the course of Master's degree.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by my thesis chair and advisor, Dr. E J Kim.

The main idea in chapter 5 was formalized by Kyung Hoon and results in section 6.3 was collected by Kyung Hoon. Data for section 3.3 was collected by Andrew Doolittle.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a Graduate Merit Scholarship from Department of Electrical and Computer Engineering, Texas A&M University.

NOMENCLATURE

IoT	Internet of Things
SVM	Support Vector Machine
LR	Logistic Regression
GBT	Gradient Boosting Tree
LOD	Leading One Detector
MSB	Most Significant Bit
LSB	Least Significant Bit
CPF	Common Power Format
RBF	Radiant basis Function

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Recent advancements in the field of machine learning and neural networks has reached human like accuracy, especially in the field of image recognition and classification. Increasing complexity of these algorithms [1] in search of better performance has concerned us with its energy consumption to realize them on hardware. Machine learning models such as Support Vector Machine(SVM) calculates dot product of every feature input with a huge number of support vectors. Neural networks are getting deeper and complex. Apart from the classification model, these algorithms have the extra burden of transforming raw data into other domains, extracting features that are characteristics of different classes. It is challenging to realize them on a battery operated device without having to recharge them frequently.

We identify the components that are the main energy culprits and focus on optimizing them. For our case study, we use medical Internet Of Things (IoT) devices for our analysis. We are especially interested in these devices as they challenge us the most because of 3 reasons: 1) Unlike many other applications, the dataset available for training these devices is limited due to patient confidentiality 2) Testing dataset differs from person to person and also changes per person over time due to improvement or degradation of their medical conditions [2] 3) These devices really need to be small and energy efficient, especially in the light of emerging in-vivo sensors and wireless charging [3][4]. We will then generalize our implementations for other domains to prove our generality.

1.1 Medical IoT Devices

The application of technology in the field of human health is, by far, one of the most interesting, advanced and sought out concepts that exists today. Supercomputers are used to model and analyze diseases [5]. Robotic surgery is the most minimal invasive surgery [6]. Smart pills can sense information about the body and dispense medicine based on current conditions [7]. The list goes on, but the recent application of the IoTs on human health gives rise to dedicated research towards its feasibility and implementation.

IoTs are various physical devices that are used in everyday lives to collect, process, analyze, transmit and draw conclusions on data. This technological field is an expanding area with new ideas and devices added each day [8]. A specific category of IoTs are wearable devices for medical applications that monitor health by tracking various activity signals such as heart rate, brain signals, pulses etc. These devices can be used to understand the general health condition of a person, record his physical activities or to trigger alarm in case of emergency for a patient [9][10][11]. However, such conclusions from data can only be made using advanced machine learning and neural network techniques. Recently, wearable systems have been actively researched and have become more prevalent commercially for data analytics with the recent development of machine learning technologies [12]. General classification methods are being adopted for bio signal analysis in intelligent wearable systems where sensors are typically used to wirelessly transmit bio signals captured from different parts of human body, and a data aggregator (e.g. smartphone) classifies meaningful events from the signals. Sensors have evolved to become more intelligent smart sensors and are able to provide some of the advanced real time applications to users (e.g. timely prediction or intervention for patients), so it is possible for the sensor to carry out the complex classification task locally and then report the analysis results to the data aggregator.

There are 3 major deployment scenarios for these IoT devices: (i) In sensor approach: A stand-alone device that would sense, compute and conclude, such as a smart watch, (ii) In aggregator approach: A sensor to capture and transmit the signals wirelessly which are captured and processed by a data aggregate device, such as a smart phone, (iii) A cross-end approach, as proposed in Xpro [13], where the sensor performs some kind of data analysis before sending it to the aggregator. In all the 3 deployment scenarios above, the energy optimization of both the sensor and aggregator are critical.

In this work, we attempt to achieve energy optimization by addressing two research problems. First, the low precision is often used in computations to achieve better energy-efficiency. However, a pre-determined precision rate can hinder the achievable energy optimization as well as effect accuracy when the test date can change according to subject or over time. Second, even if it is

desirable to control the precision per input data for better energy-efficiency and reliability, the traditional hardware design do not efficiently support run time dynamic precision control. Hence, we design a custom approximated hardware for our system and design an online calibration mechanism to decide on the approximation. We will generalize our work and compare it to the recent advancements in this field.

The reminder of this paper is organized as follows. In chapter 2, we present the application scope of this paper. We then present the motivation for approximation in chapter Chapter 3. Chapter 4 lists major advancements in approximate multipliers. We introduce a unique hardware solution to optimize energy in Chapter 5, and propose a calibration engine to determine the best approximation rate. In Chapter 6, we present our results. Finally, we draw conclusion in Chapter 7.

2. FRAMEWORK

2.1 Overall System

A general classification task involves two key processes: feature extraction and model classification. Figure 2.1 shows our framework that we use for analysis. We choose eight hardware friendly features as inputs to our model - Min, Max, Mean, Czero, Variance(Var), Standard deviation(Std), Skewness(Skew) and Kurtosis(Kurt). These features capture statistical moments and signal characterstics from time domain input raw data. To enhance the distinguish ability of the features, we also extract the corresponding features by taking as input wavelet domain coefficients which are decomposed from time domain biosignal data. The DWT technique is widely used in signal processing tasks due to its great time and frequency localization ability at multiple resolutions. We use SVM as classification models for our analysis. We train SVMs by using LIBSVM [14] and use 10-fold cross-validation technique to find the optimal hyperparameter of each SVM classifier. Figure 2.2 shows the dependencies and the flow of our design.

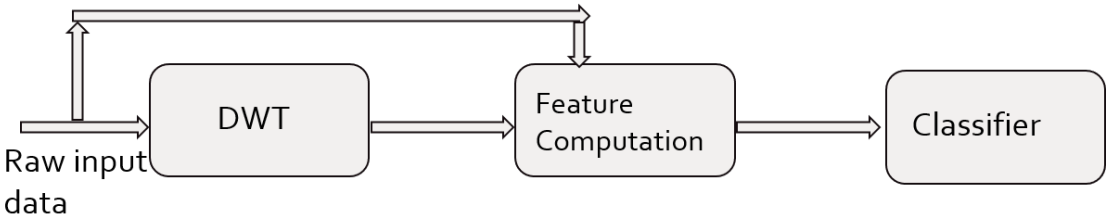


Figure 2.1: Framework for IoT devices

We design the feature extraction models in verilog and call them the functional cells. The functional cells are built using multiplication, division, cordic exponential sub-modules. The DWT module computes 6 levels of DWT. Features are extracted from time and all the 6 DWT levels of DWT, thus giving 8*7 features to choose from. To enhance its energy efficiency, each functional cell is in the idle state that powers off internal processing modules via power gating, if not all input

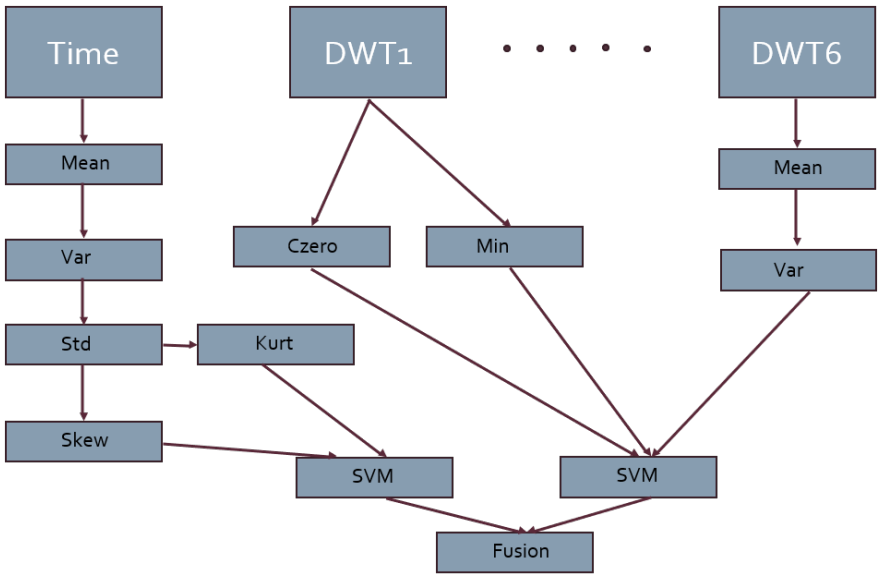


Figure 2.2: Computational dependencies for inference in IoT devices

data is ready. Otherwise, it is enabled for execution. Because we focus on biological signals for low-power sensor nodes, serial operation is the preferred choice.

2.2 Evaluation Method

Our goal is to have the framework based on the approximate computing mimic the computation behavior of the precise computing.

2.2.1 Benchmarks and Model

We evaluate our proposed approximate computing framework in representative healthcare applications where an intelligent sensor predicts the precarious situations such as seizure or heart attack by monitoring the health state of human-beings through their biosignals for a long period of time. In this work, we choose the applications that require a binary classification of the widely-used biosignal such as ECG, EEG and EMG. The evaluated benchmarks are summarized in Table 2.1.

We design the smart sensor to incorporate the full-fledged analytic engine with both feature extraction and inference computation by using functional cells in our framework. To comprehensively evaluate the reliability of our proposed approximation technique, we use 100 classifiers per application, so we evaluate 600 inference engines in total.

Acronym	Biosignal
C1	ECG
C2	ECG
E1	EEG
E2	EEG
M1	EMG
M2	EMG

Table 2.1: Summary of Benchmarks

2.2.2 Simulator

To facilitate the evaluation of the classification quality for a number of applications, we design a in-house simulator that exactly models the computation behavior of the hardware components of our framework. We implement the primitive logic such as multiplier, divider and Cordic-based math functions dealing with fixed-point real values, all functional cells using the logic and the engine optimizer in the simulator. The simulator itself is designed in Matlab, however, we use Synopsys VCS for verilog simulations and Synopsys design vision for activity based power estimation using Synopsys 90nm generic library.

3. MOTIVATION

In this chapter, we present the motivation for our work.

3.1 The energy culprit

One of the specifications important to an IoT device's end user is not having to charge it frequently. Since the battery life of a device is determined by the energy consumed by the system, it is important to reduce it as much as possible. Both the power of the system and how long it runs, determines its total energy. If, let us say, a device is made of multiple components and P1, P2 and P3 are its power, it takes an overall C cycles to run, then its overall energy is E = (P1+P2+P3)*C. To reduce the energy, it is crucial to reduce both power and cycles. We look at various steps involved in the complete process - DWT transform of raw data, feature computation and classification model and plot the power consumed and cycles taken by various computations. Figure 3.1 breaks down the power consumed by computation of 1st level DWT from raw data, extraction of features using these DWT values and SVM classification using DWT features. Though presented for one of the benchmarks, our analysis shows similar trend among the rest as well.

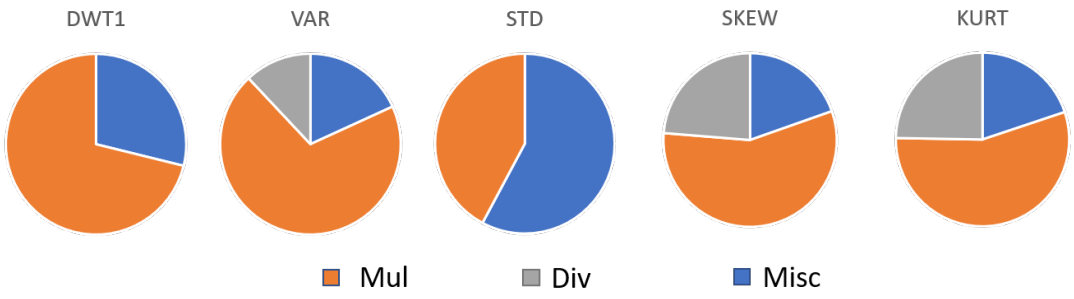


Figure 3.1: Distribution of Power

Figure 3.2 breaks down the cycles consumed by multiplier for these computations respectively. Misc corresponds to miscellaneous logic in the design around the multiplier, divider block,

required for the correct functionality of the design.

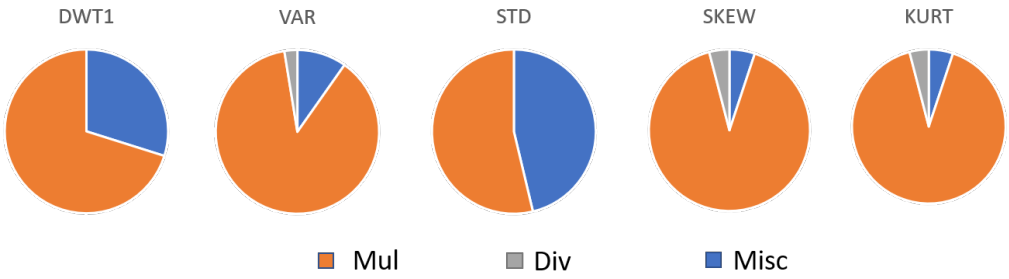


Figure 3.2: Distribution of Cycles

Clearly, the multiplier logic consumes majority of the time and power. To achieve energy optimization, it is important to optimize both the power as well as cycles of the multiplier logic.

3.2 Redundant bits in input values

We closely observe computational data from various machine learning and neural network applications. We examine the inputs of multipliers to identify the number of leading zeros, which could potentially be cut off from the input values, effectively reducing the hardware required to store and process these bits, without effecting accuracy. To our happy surprise, we realize that most of these values are very small numbers. Figure 3.3 shows the frequency of number of leading zeros in input values among 5 sets across 3 different applications: 1) Neural network for digit recognition using MNIST dataset 2) DWT computation for medical data using EEG, ECG and EMG datasets 3) Feature extractions for the above medical data 4) Non-linear SVM as a model classifier for the same medical data and 5) Face detection using linear SVM. Here, the maximum number of zeros possible is 15, and not 16, due to sign bit. Clearly, most of them have no values in their integer part. Multiplication with these input operands is a waste of energy. This calls for a unique design of a multiplier that takes advantage of this input pattern dynamically, not restricting the bits to any fixed number.

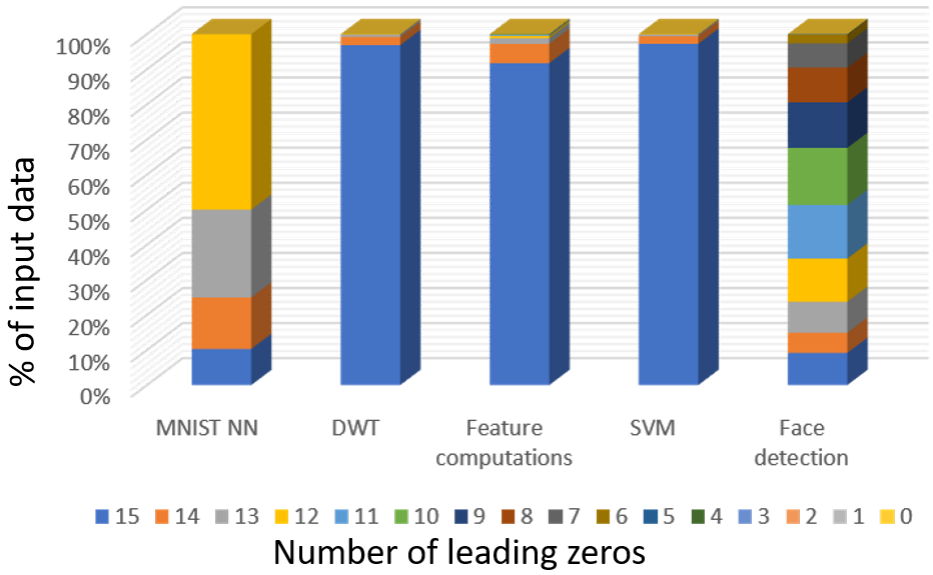


Figure 3.3: Leading Zeros in Biosignals

3.3 Subject dependencies of bio signals

When dealing with medical IoT devices that capture the patients EEG, ECG and EMG signals, it is crucial to note the sensitivity of data with respect to different subjects and their conditions to accurately predict any unusual activity in their medical state. We need to control fine tuning parameters, such as approximation rate, due to 2 reasons: 1) Lee et al. [2] show how the signals can vary with improving medical conditions of patient. This requires us to design our hardware in a way so that we can tune it overtime. 2) We observe how the precision required analyzing these signals vary with subjects. We consider EMG signals from 5 subjects and use each possible combination of 3 different subjects to make 10 different groups. For each group, SVM models were trained using top 3 models to determine the precision rate for accurate computation. For each model, precision was calculated using the group the model was trained on (offline calibration) and with the 2 subjects that were not part of that group (online calibration). Figure 3.4 shows the precision deduced while offline calibrating vs required calibration observed during online calibration.

The variability in the calibration result and the accuracy loss as presented in Figure 3.5 indicates

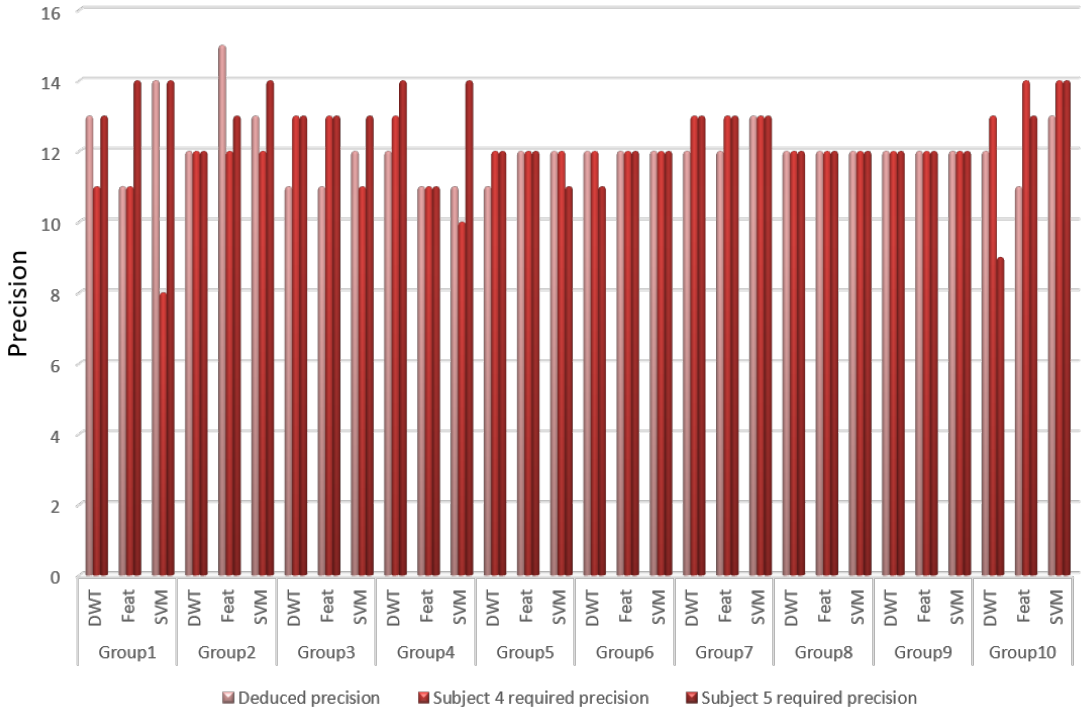


Figure 3.4: Deduced vs required precision across subjects

that offline calibration may not be adequate for the signals sampled from a particular subject. In cases where online calibration finds a lower needed precision, the offline calibration would pick the approximation degree too conservatively and we wouldn't see the full benefits of approximation. In cases where online calibration finds it needs a higher precision, the offline calibration will choose approximation too aggressively and will reduce reliability. This calls for an online calibration mechanism that can aggressively adjust the approximation rate without hurting performance.

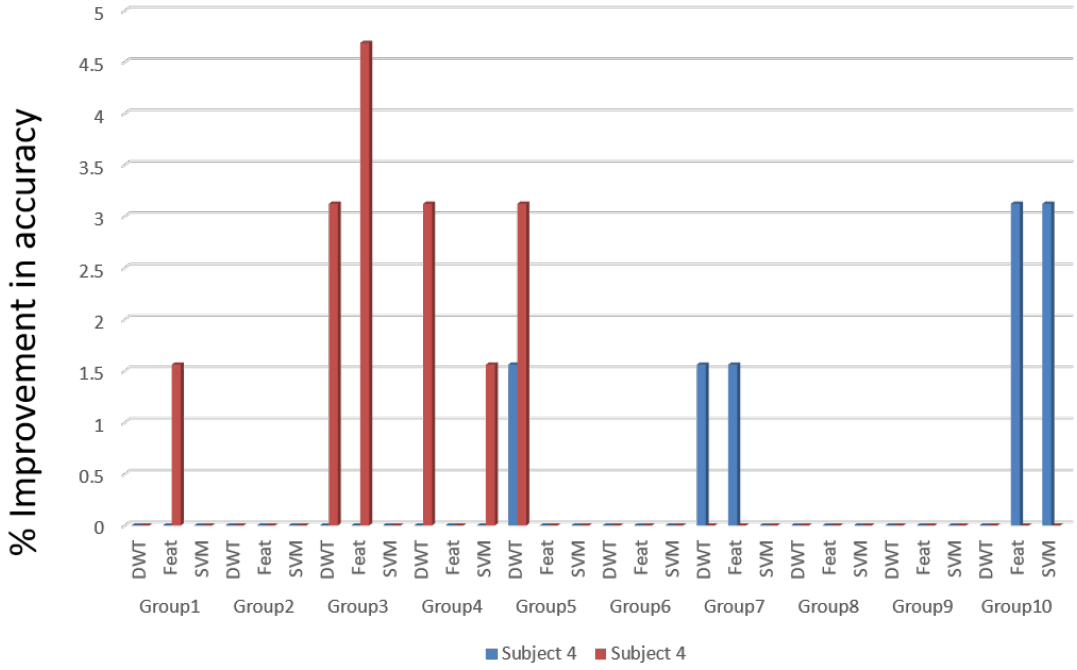


Figure 3.5: Precision loss with offline training

4. RELATED WORK

Neural network is one of the areas proven to be error resilient, and hence approximations and reduced precisions are extensively employed. Neural networks are computationally intensive, with majority of the energy consumed by multiplications. Various approaches are proposed that specifically identify layers to approximate, weights vs data approximations and identifying specific neurons that do not contribute to errors.

Du et al. [15] proposed a Neural network accelerator that uses a special kind of multiplier. This multiplier used is designed to perform inexact computation through logic minimization to reduce the multiplication energy. Mrazek et al. [16] later proposed a similar inexact multiplier of widths 7 and 11 but designed this approximate multiplier by mutating an accurate multiplier. The accurate multiplier is designed using Cartesian Genetic Programming (CGP) [17]. CGP is a method of genetic programming model, representing the Boolean function as a directed acyclic graph where the nodes are 2-input Boolean functions. CGP is used because of its ability to generate approximate hardware implementations [18]. The graph of accurate multiplier is recursively mutated until a graph is obtained with fewer gates and has an accuracy loss below the expected threshold. However, it is interesting to observe that these methods obtain a good reduction in energy without huge performance loss because the authors assume that it is possible to retrain the network after approximations, which readjusts the weights, and hence does not hurt performance.

Sarwar et al. [19] proposed a multiplier design to reduce the cycles and power consumed by a multiplier. In a multiplication operation, the product can be generated from smaller bit sequences, which are lower order multiples of multiplier input. Hence, if simple multiples of multiplier are available, the multiplication process is reduced to a few shifts and add operations. To gain performance and energy improvements, they proposed fewer multiples of multiplier input, thus bringing in the concept of approximation.

Sharma et al. [20] propose a new architecture design for approximate neural networks. It is composed of a collection of bit-level computational elements, that dynamically compose to

logically construct a complete computational engine. The number of bit-blocks used is dependent on the approximation degree, which is a pre-determined value, conveyed to the hardware through ISA (instruction set architecture).

Hashemi et al. [21] too use approximate multipliers to optimize energy by saving cycles. They use 2 leading one detectors (LODs) to identify the significant 1s from the MSB. For each operand, the location of the most significant 1 is used to capture the next k-2 elements, where k is the number of bits used in approximate multiplication. The remaining lower bits are either approximated to zeros or ones based on the closeness of approximated value to its actual value.

Shao et al [22] propose an error correction method to approximated multipliers. They propose a Approximate Arithmatic Computing (AAAC) model that controls approximation errors and present optimal error compensation schemes. The Error Compensation Unit (ECU) consists of signature generator and a K-to-1 mux. For a given input, signature generator produces a signature capturing essential information about the inputs. Based on this signature, input is classified into one of the K groups, each having a predetermined error compensation.

Lie et al [23] is the most recent work on multiplier design for machine learning applications. It is based on the idea that most of the machine learning algorithms are run on a pre-trained machine with fixed weights. So almost all the time, one of the inputs to the multiplier is fixed. To exploit this characteristic, the fixed weight is pre-processed and encoded with predetermined number of bits, according to acceptable levels of precision rather that storing them in binary format. The bit positions of "1"s are stored in the memory for quick shifts and computations. Here is an example explaining their idea: Lets say we intend to multiply A(= 011111012 (12510)) with predetermined number of iterations n(= 4). So, we store four leading 1s in A (= 6, 5, 4 and 3). In the first iteration, 6 is read from memory and B is left-shifted by that amount (010110102 « 6), and accumulate the shifted value to Sum that is initialized to 0s. In the second iteration, repeat the same procedure as the first iteration but now shift by 5 and add to accumulated sum, obtaining 864010 or accomplishing 77-percent accuracy. This process is repeated for the subsequent two leading 1s, providing 98-percent and about 100-percent accuracy after the third and fourth iterations.

5. DESIGN IDEA

Our aim is to redesign the multiplier hardware to reduce its power and cycles based on truncation, due to the error resilience of machine learning to variations in input. While doing so, we also show how to determine the approximation rate reliably, while maximizing energy optimization.

5.1 Multiplier design

We design a multiplier that multiplies two fixed-point numbers by controlling the computational precision to save the energy consumption. The key idea is to redesign a traditional multiplier to be able to dynamically control the number of bits used during computations by power gating. By turning off blocks related to unnecessary precision data, the multiplication is carried out energy-efficiently.

5.1.1 Low power design methodology

Power consumed by a system can be divided into 2 components - dynamic and static power. Dynamic power is associated with momentary current loss when both the NMOS and PMOS of CMOS transistors are ON during a 0 to 1 or a 1 to 0 transition. It is directly related to the clock frequency. On the other hand, static power is the power dissipated by a CMOS due to sub-threshold current when held at a constant value. We suffer static power loss whenever the CMOS is powered. Figure 5.1 explains the same in an inverter. While the major component of power consumption had been dynamic power for quite a long time, for submicron technology, static power is equal to or more than in proportion as compared to dynamic power and increases with decrease in technology. Low power design techniques have been well researched and implemented to reduce static power. These techniques aim at switching power off for unnecessary logic not in use. We use this method, called as sleep mode to switch off computational logic associated with unnecessary 0 valued bits on the integer and truncated bits on our fractional part of multiplier.

In order to switch off parts of circuit that do not contribute to the computations, or parts of circuits that can be dynamically be powered on and off, it is required to partition the circuit into

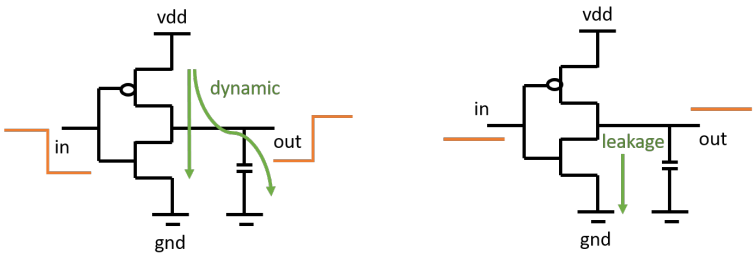


Figure 5.1: Dynamic and static power dissipation in a CMOS

power domains. Each power domain has its own header switch. A header power switch is just a wider PMOS, whose input is controlled by the power management circuit. We use a wider PMOS to limit the leakage associated with it. Since the outputs of a powered off logic domain would be corrupted due to no power, the circuits using them need to receive a constant logic to avoid forward logic corruption. This is done using isolation cells. Isolation cells are special logic elements that sit between "powered off" domain, X and another "powered on" domain, Y and drive a constant pre-programmed value when the domain X is turned off. This, is again controlled by the power management circuit. A system with 2 power domains, with domain X off and domain Y on is illustrated in Figure 5.2.

5.1.2 Main idea

Figure 5.3 illustrates the basic operation of an integer-based multiplication with an example of a multiplier operand, A (0101_2) and a multiplicand operand, B (0111_2). The multiplication is performed iteratively using two registers. At the beginning, register 1 stores a multiplicand. At each ith iteration, the ith bit of the multiplier is examined if it stores one or zero. If one, register 1 shifted left by i and is accumulated to register 2. After P (the bit-width of operands; 4 in this example) iterations, register 2 becomes the final product.

Based on our finding as presented in the motivation section, we find that most bits on the MSB are generally zeros and do not need computations. To exploit this, we use a leading 1-bit detector logic inside our multiplier. This is used to end accumulation when the multiplicand bit-scan reaches this leading 1 bit, as further zeros do not effect the result. We further propose to truncate a few

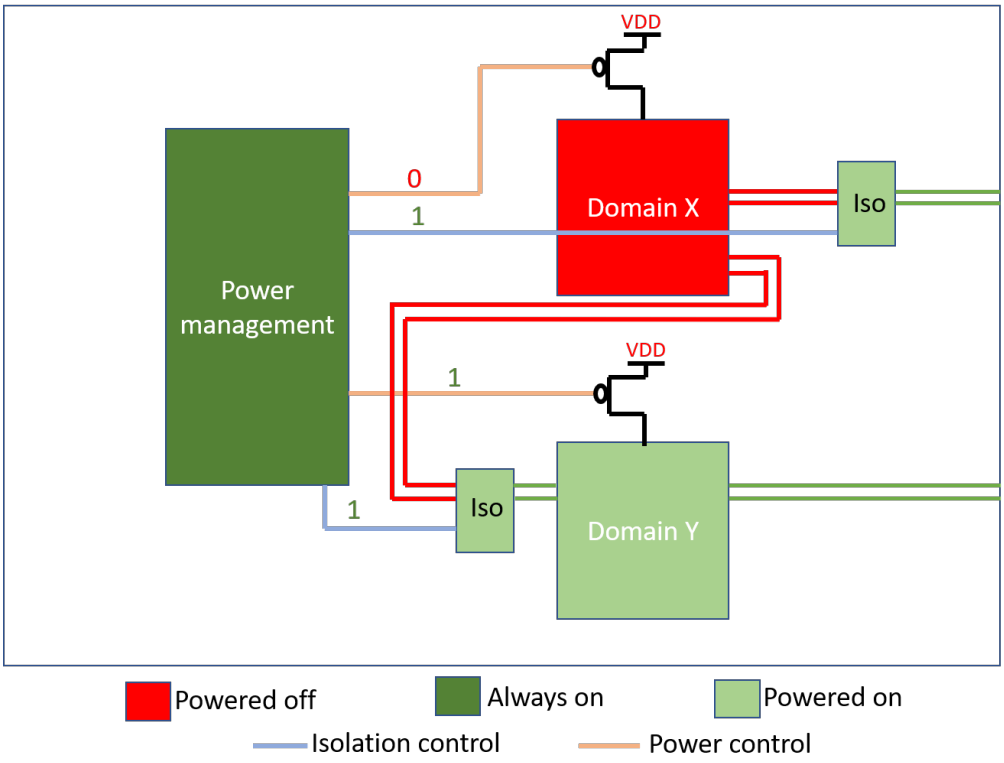


Figure 5.2: Header and isolation control of a low power system

bits on the fractional side to save power and cycles, since most of the machine learning and neural network algorithms are error resilient to imprecise data. Since the traditional multiplier circuits are tightly optimized to deal with fixed-point numbers with a certain precision (typically 32 bits), it is not trivial to reduce its power dissipation when the full-precision is not necessary. Thus, we redesign a multiplier as a combination of finer cells, called $MulCell$, each of which vertically fuses the multiplier circuit in a bit-level manner as illustrated in Figure 5.4. Each $MulCell$ is controlled by a distinct power subnet where the power source is controlled by a head switch and the output of the power-gated $MulCell$ is set to a fixed state by an isolation cell. When the full-precision is not necessary, unnecessary $MulCell$s are turned off to reduce the power consumption. How to automatically decide the necessary precision during runtime will be discussed in Section 5.2.

Figure 5.5 and Figure 5.6 illustrates a walk-through example about how our proposed MulCell-based multiplier works without and with power gating.

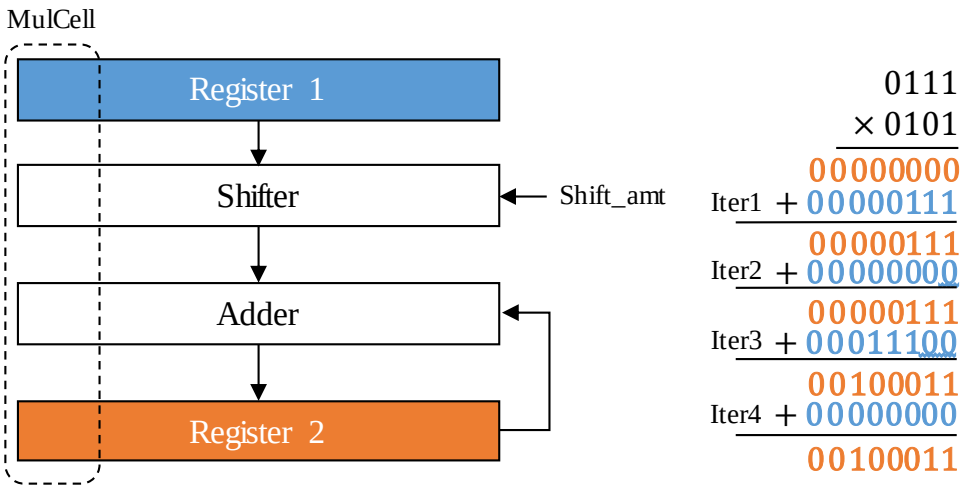


Figure 5.3: Illustration of an integer-based multiplier

5.1.3 Effectiveness of MulCell Design

Since we focus on reducing cycles too, we use $MulCell$ based approach on Booths multiplier. Here, instead of scanning 1 bit of multiplicand B, we scan three bits at a time, with 1-bit overlap. If the three bits are 000 or 111, we do nothing. If the bits are 001 or 010, our partial product is A, if it is 011, it is 2A, if it is 101 or 110 it is -A, else it is -2A. Since among the 3 bits scanned, one bit is overlapped, our cycles are halved. We modify this design to use our $MulCells$. We wire a given $MulCell$ with the previous two $MulCells$ that automatically computes shifted A and shifted 2A, thus further reducing power. Figure 5.7 shows our circuit with highlighted changes with respect to integer multiplier.

However, it is not effective to choose a fixed approximation degree due to 3 reasons: 1) The expected accuracy might need to be updated 2) Test data can differ greatly from train data, thus effecting performance 3) The required approximation degree can change over time or can differ by subject. We need an online mechanism to re-calibrate the approximation degree to be used by the system.

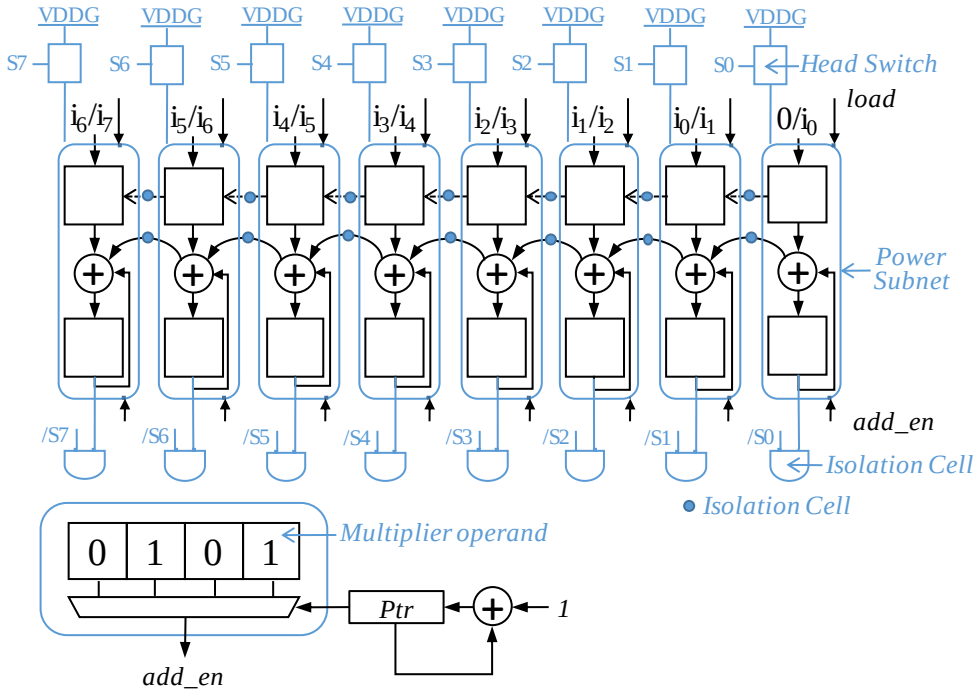


Figure 5.4: Proposed multiplier with the capability of bit-level power gating

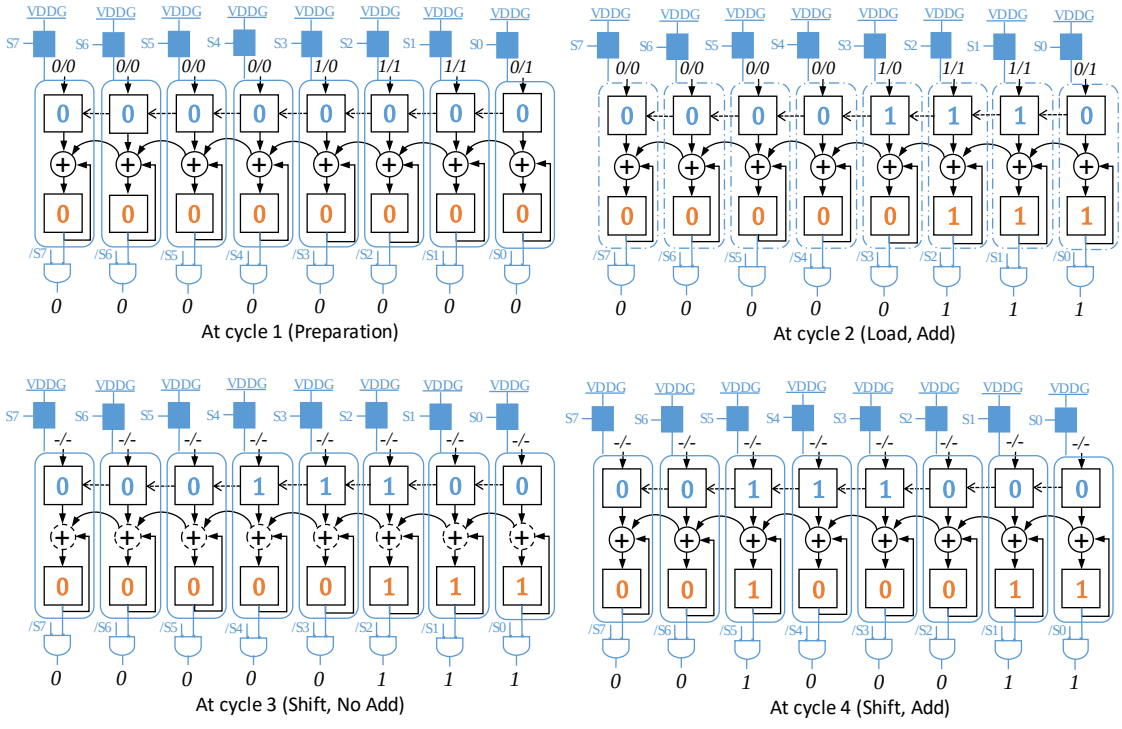


Figure 5.5: An execution example of our proposed multiplier without power gating

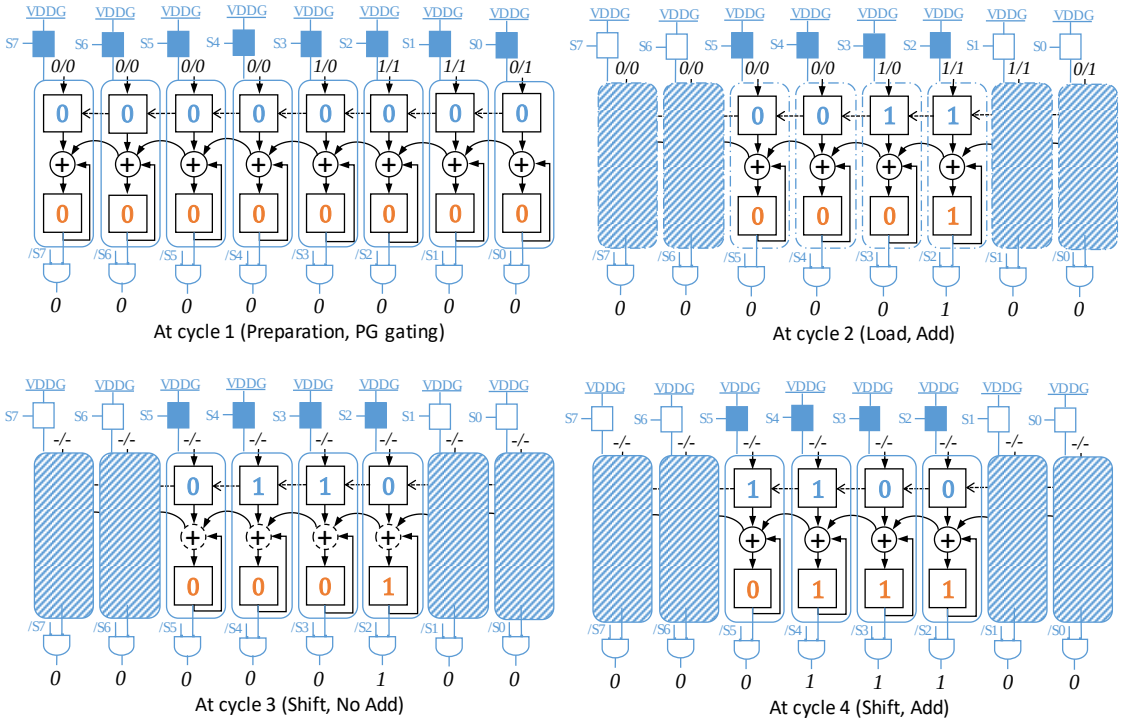


Figure 5.6: An execution example of our proposed multiplier with power gating

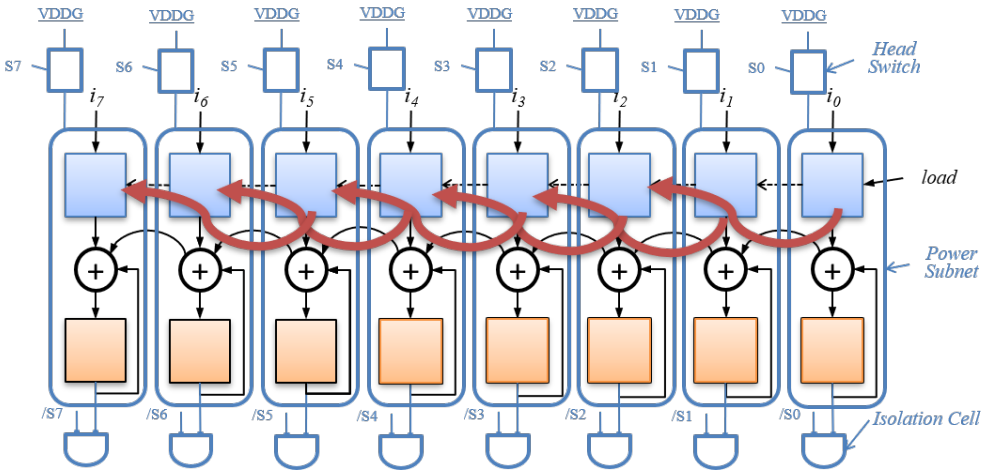


Figure 5.7: Architecture of Booths multiplier using $MulCell$

5.2 Calibration

We design an online calibration mechanism that searches through the possible approximation degree to be used further during the test phase. Here, we use 3 parameters for approximation degree, one each for DWT computation, feature extraction and SVM. All the degrees are independent of each other. Each of the multipliers in the respective computations use the approximation degree as set by the online calibrator.

5.2.1 Basic optimization mechanism

We propose a batch optimization mechanism that finds a minimum precision for an individual input data and aggregates a set of optimal precision into a final precision by conservatively choosing maximum one. The model is trained online and then implemented in hardware. Before actually using the model on test data, we run a calibration phase on a subset of test data. During this phase, the model is predicting in precise mode, with no loss of accuracy. During the same time, it performs various iterations on the same data to deduce the approximation rate. Once the calibration phase is complete, the model inference is based on approximate data as decided by the calibration engine. The calibration phase can be re-triggered as necessary. The calibration engine is a simple state machine and functions as follows: As shown in Figure 5.8, a 2-bit $Mode$ register controls the mode the system is currently operating in. When this $Mode$ is "00", the analytical engine is operating is precise mode. When the $Mode$ is "01", the analytical engine is operating in approx mode and for "10", calibration is complete and the analytic engine is predicting in approx mode with the approximation rate as decided. The control for $Mode$ register is driven by an energy optimizer. The engine optimizer first sets the $Mode$ in "00", records the predicted values. It then sets it to "01", and the analytic engine starts to predict for increasing approximate rates and compares it against precise values. The approximate computing stops when the predicted values no more match the precise values. At this point, the optimizer engine records the last best approximation rates into a table and moves the $Mode$ into its next state, "10" if no more calibration data is available, indicating completion of calibration, else to "00" to calibrate next data. The calibration

is run for B input data values and the maximum of all the minimum precision values as recorded in the table is chosen and written into $PrecClassResults$, as the final approximation degree. The analytic engine then continues to operate with recorded precision rate from $PrecClassResult$. To avoid excessive number of computations, the optimization engine starts with values recorded in $PrecClassResult$, rather than from zeros. Figure 5.9 shows the timeline of calibration and actual prediction.

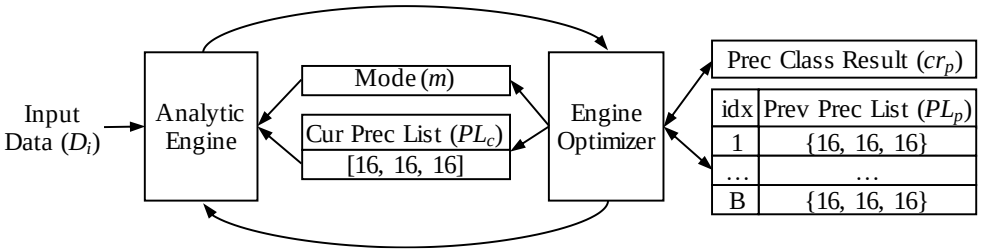


Figure 5.8: Calibration engine design

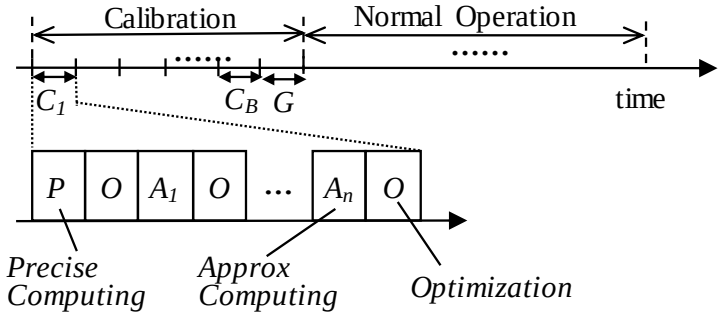


Figure 5.9: System execution timeline

5.2.2 Precision bound problem

We observe that the majority of input data compute precise classification results with low precision, but the final precision needs to be bounded by high precision to reliably deal with some input

data sensitive to approximate computing. Figure 5.10 shows a histogram of minimum precision per input data collected from the calibration training phase when a SVM cell carries out its inference computation approximately by using precise features as input. In this example, the benchmarks, C1, C2, E1, E2,M1 and M2 choose 11, 7, 7, 12, 11 and 13 out of 16-bit precision, respectively due to 1-3% input cases. Otherwise, it is statistically probable that the accuracy loss incurs by such percentages. Interestingly, 44-62% of input data produce precise classification results with zero precision.

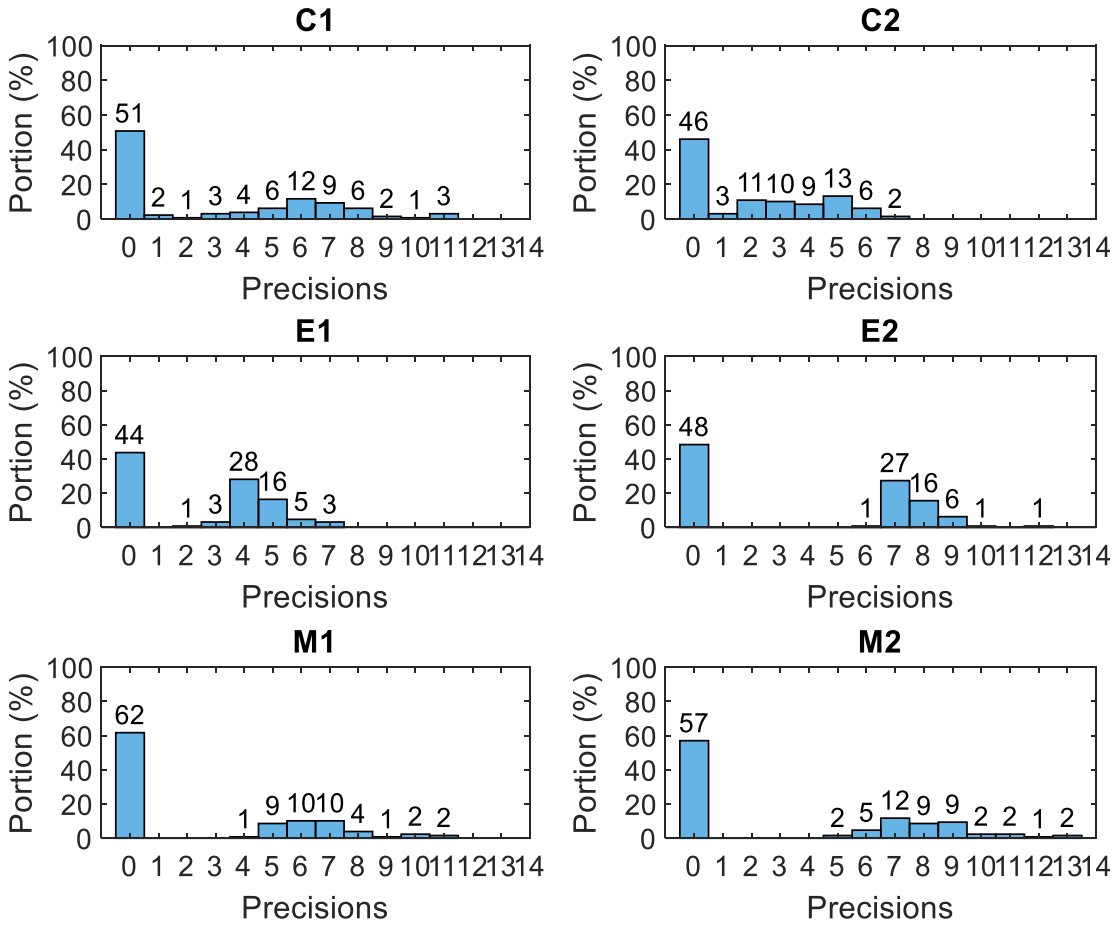


Figure 5.10: Min precision per input data

To further explore if the precision bound problem is specific to SVM, we perform experiments on 3 models SVM, GBT and NN. For SVM, we choose precise data and approximate model. We

use approximated features and precise model for GBT and approximated features and model for NN, so as to cover all cases. We build 10 different models of each by varying the input features and plot a graph of maximum precision required in each case for all the benchmarks, as shown in Figure 5.11. Clearly, the maximum precision is 11-15 across models. The maximum bound problem exists across models, limiting the maximum energy optimization that can be achieved, thus motivating us to build a calibration system that can adjust the precision according to the data type.

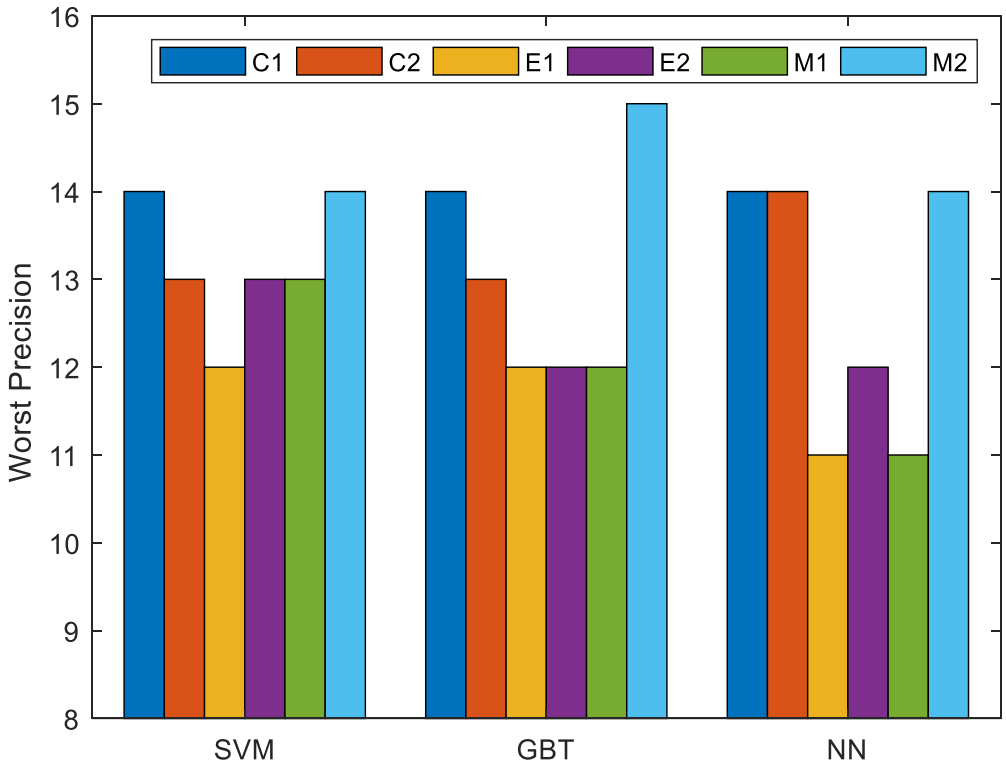


Figure 5.11: Minimum precision required across models

5.2.3 Solution

Since our main problem of higher precision rate is the bound caused by a small percentage of sensitive data, we intend to identify this sensitive data, so as to distinguish it from others. Figure 5.12 shows the distribution of datapoints in a 2D plane. Blue data points corresponds to positive

labels, and red data points correspond to negative labels, that require low precisions. The light blue stars correspond to data points requiring highest precisions and are at the boundary that separates the 2 regions.

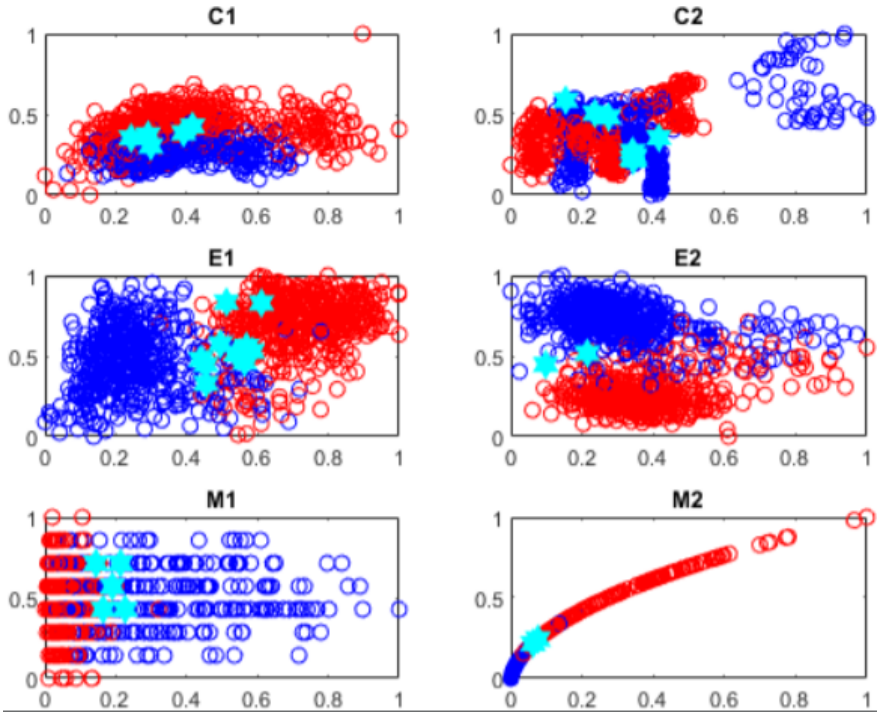


Figure 5.12: Data distribution requiring high and low precision

As our case study, we use SVM for this analysis, with precise features. A SVM model tries to classify input data into 2 labels: positive and negative, separating them by a hyperplane. Values far away from the hyperplane can be predicted with high confidence, and those close to the plane are the main causes of errors. In our case, these sensitive input data are the ones requiring high precision. To distinguish them from others, we divide the input dataset into several clusters - as shown in Figure 5.13. When the data is farthest from hyperplane, we can predict them with utmost certainty, and thus, require only a very few bits of them. For weakly certain data, we maintain high precision.

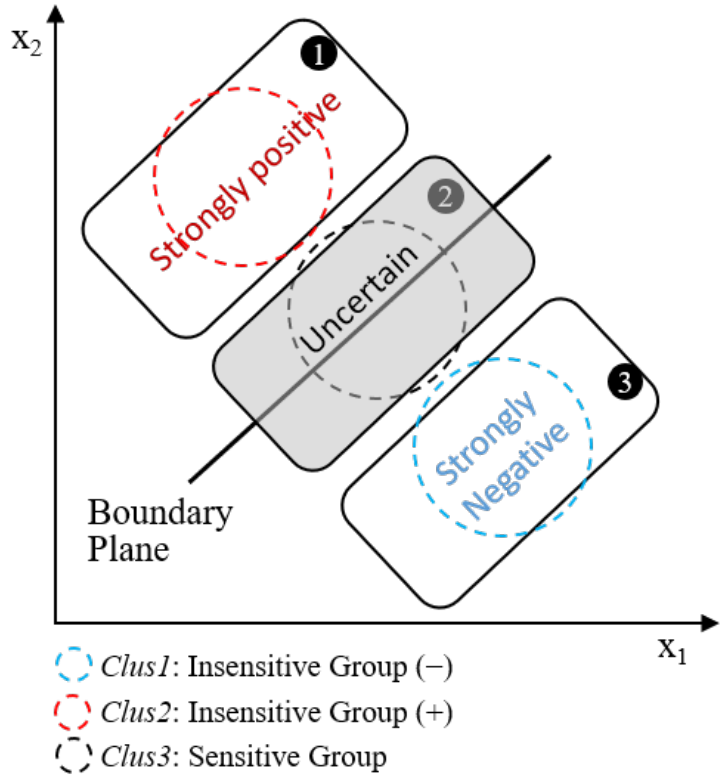


Figure 5.13: SVM clustering solution

5.2.3.1 Cluster formation

In order to form clusters, during calibration on test data, we record the feature values, their required precision, and the inference they belong to in a table as shown in Figure 5.14a. Just after calibration is complete, we re-arrange the table in ascending order of precision requirement. We pick the max 2 precision values as belonging to the uncertain group as shown in Figure 5.14b. We collect the data points that correspond to these precisions and are the representatives of the uncertain group.

To decide the representatives of other groups, all other feature points corresponding to positive label are representatives of strongly positive group and all others corresponding to negative labels correspond to strongly negative group. Once we have the representatives of these groups, we compute the centroid of each of the clusters as shown in Figure 5.14c. We have now devised 3

clusters.

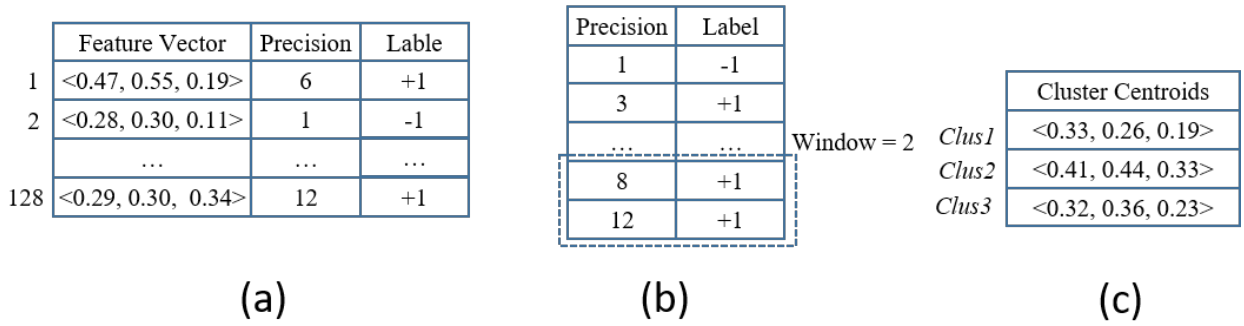


Figure 5.14: Cluster formation a) Collect data b) Decide on max precision that can be categorized as unceratin c) Compute centroid for each clusters

5.2.3.2 Cluster classification of new data

For a new incoming data, in order to determine the cluster it belongs to, we determine the Euclidian distance of the new data point with respect to the centroids of three groups. Based on this information, we compute the 3 scores using Radial Basis Function (RBF) kernel. The new point is said to belong to the cluster corresponding to maximum RBF score. If the new data corresponds to strongly positive or strongly negative group, then we skip the SVM model inference and make a prediction about it based on the cluster it belongs to. If the data corresponds to uncertain group, we use SVM with full precision to make the prediction.

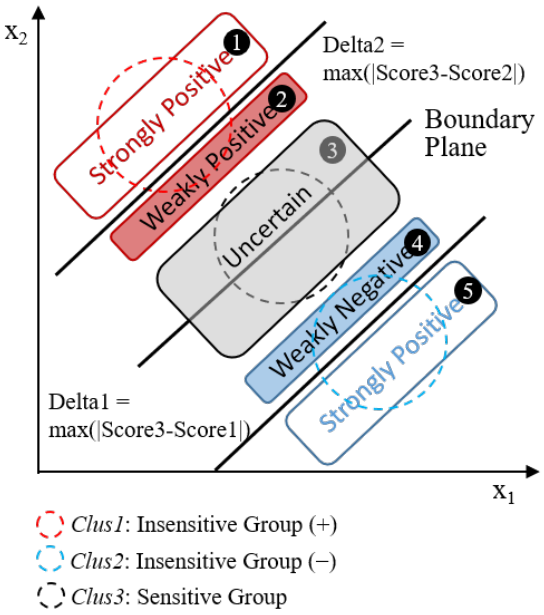
5.2.3.3 Outliers

It is highly possible that the sample data used to form clusters during calibration might not be an accurate representative of the entire data in general. For a new incoming data, its distance from all the clusters can be large, resulting in very poor score with respect to all clusters, as highlighted in Figure 5.15a. Picking the maximum score and grouping it in a confident cluster results in high accuracy loss. Hence, we form 2 more clusters: weakly positive and weakly negative clusters, belonging to those outliers that have similar scores with respect to 2 clusters, as shown in Figure

5.15b. Now, if the data belongs to uncertain cluster, weakly positive and weakly negative clusters, we use SVM to make classification decision with full precision. For strongly certain groups, we skip SVM, thus saving considerable energy.

	Score1	Score2	Score3	Label
1	0.07	0.41	0.23	+1
2	0.78	0.13	0.61	-1
3	0.09	0.10	0.06	-1
4	0.0158	0.0001	0.0033	1
	…	…	…	…
	0.50	0.44	0.64	0

(a)



(b)

Figure 5.15: Cluster formation a) Outliers with low scores b) Introduction of weakly positive and weakly negative clusters

6.1 Multiplier results

We evaluate energy savings from our multiplier design for the complete truncation range on 5 scenarios: 1) DWT computations on raw data for E1, E2, C1, C2, M1 and M2 2) All feature computations using time and DWT1-DWT6 input data 3) SVM inference using above features 4) SVM inference for face detection[24] and 5) Neural network for MNIST digit recognition. The results are presented in figures below. We present our results with the work of SiMul by Liu[23].

6.1.1 DWT computations of biosignal data

In DWT, the transform matrices are fixed. The computations involve multiplication with these fixed values, and hence, SiMul shows good performance. But since we power gate along with saving cycles, we see better energy savings with our multiplier for lower truncation bits than SiMul. The results for DWT energy savings are shown in Figure 6.1

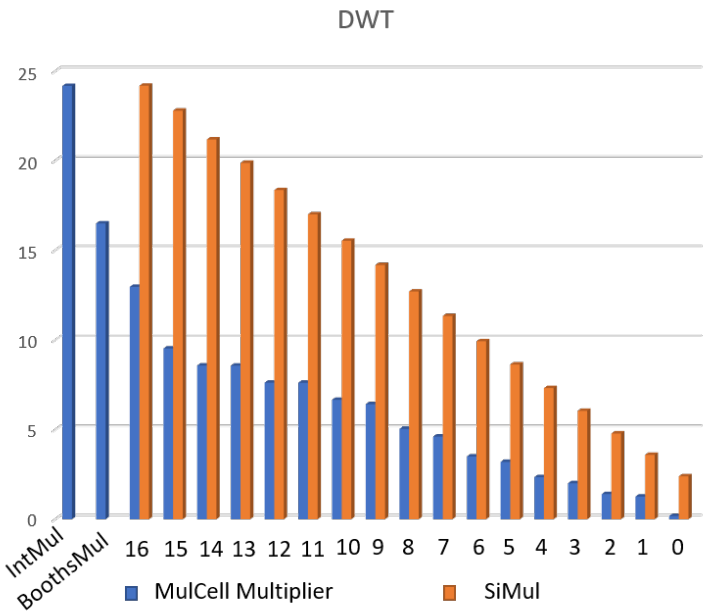


Figure 6.1: Energy reduction for DWT computations on biosignals

6.1.2 Feature computation of biosignal data

For feature computations, since none of the inputs are ever fixed, Simul performs same as a baseline integer multiplication. SiMul is not suitable for computations involving varying input values. Our multiplier performs very well in this case as shown in Figure 6.2

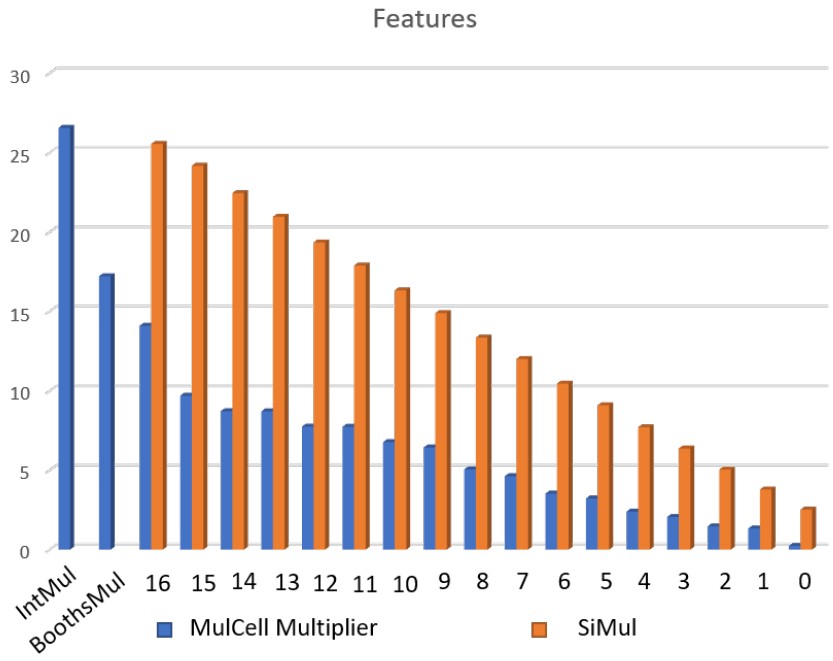


Figure 6.2: Energy reduction for feature extraction on bio signal

6.1.3 SVM for inference of extracted features from biosignal

SVM involves multiplication of input with support vectors, but computation of exponential involves multiplications where both the input values change. In such case, SiMul cannot be used, and hence is replaced with an integer multiplier. Our multiplier reduces energy significantly with increasing truncation bits as shown in Figure 6.3

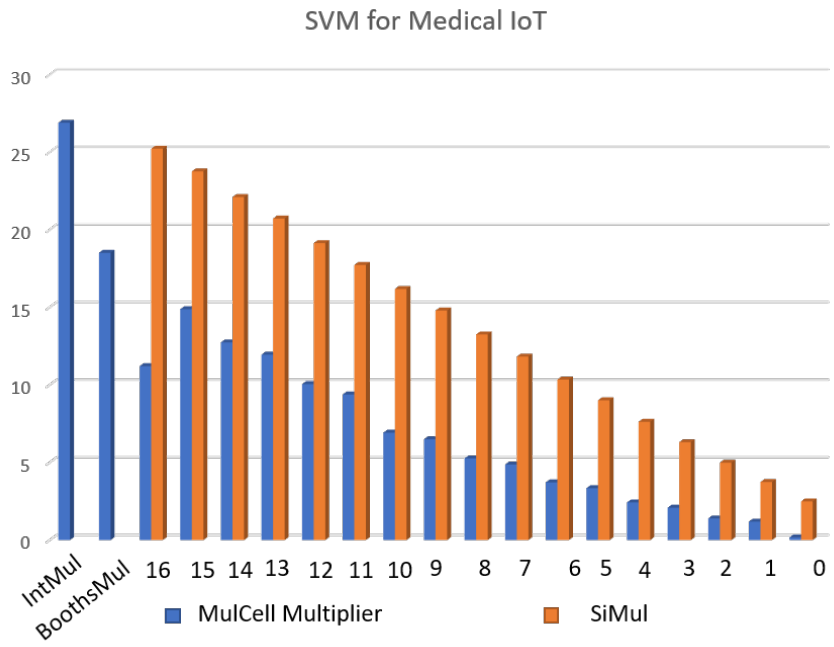


Figure 6.3: Energy reduction for SVM inference on biosignals

6.1.4 SVM for Face Detection

Figure 6.4 shows energy optimization on SVM for Face detection. All the multipliers used in SVM for face detection have one input fixed, and hence, SiMul seems to perform better.

6.1.5 Neural Network for MNIST digit recognition

NN involves multiplication of input with weights, that are fixed. It also has exponential calculations, where SiMul is replaced with Integer multiplication. Our multiplier performs similar to SiMul, as shown in Figure 6.5.

6.2 Other concerns

6.2.1 Overhead

For 90nm technology, the turning on and off of power switch takes about 200-300ps, thus enabling to run our multiplier even at 2GHz clock cycle with good margin without any overhead, because this can be overlapped with the first clock cycle used to load the data. We have an area

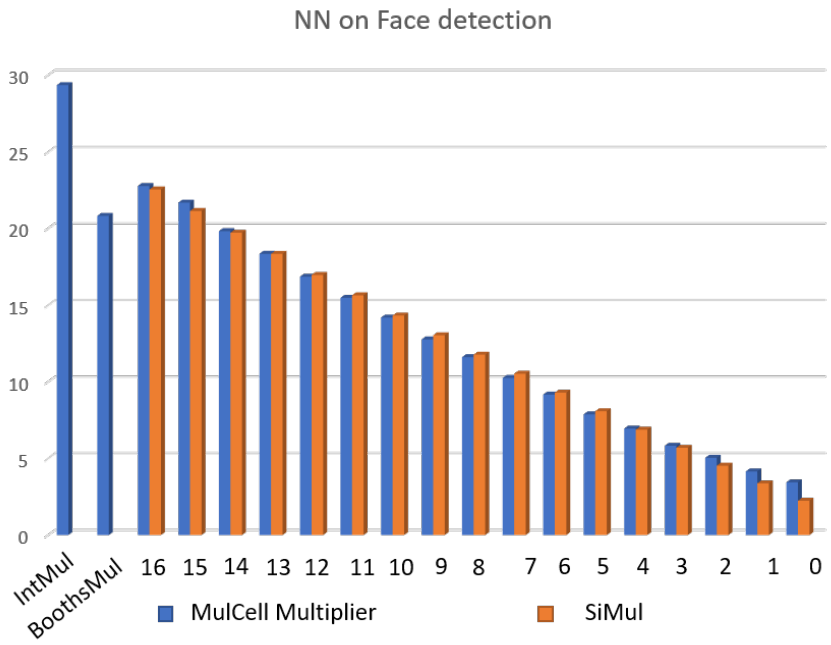


Figure 6.4: Energy reduction for Face Detection

overhead of 39% and power overhead of 13%. The overheads are higher than expected due to 3 reasons: 1) The overhead logic used to detect leading one, 2) Header and isolation cells required for low power design and 3) Due to partitioning the design into 32 power domains, the synthesis tool can no-longer optimize logic across power domain boundaries, a constrain not applicable to general multipliers.

6.2.2 Reliability

Though the isolation cells ensure the forward logic does not get corrupted due to switching off of a power domain, care must be taken during design that the control signals from power management follow a power cycles sequence such that isolation condition is turned on before cutting power supply and is later turned off only after the power has been fully restored. The power cycle may sometimes need a reset, but in our case, since only a few bits from multiplier are turned off, we do not see the need for a reset. We do not even retain any of the logic values during sleep mode, as the output of multiplier are considered zero for switched off $MulCells$. We perform

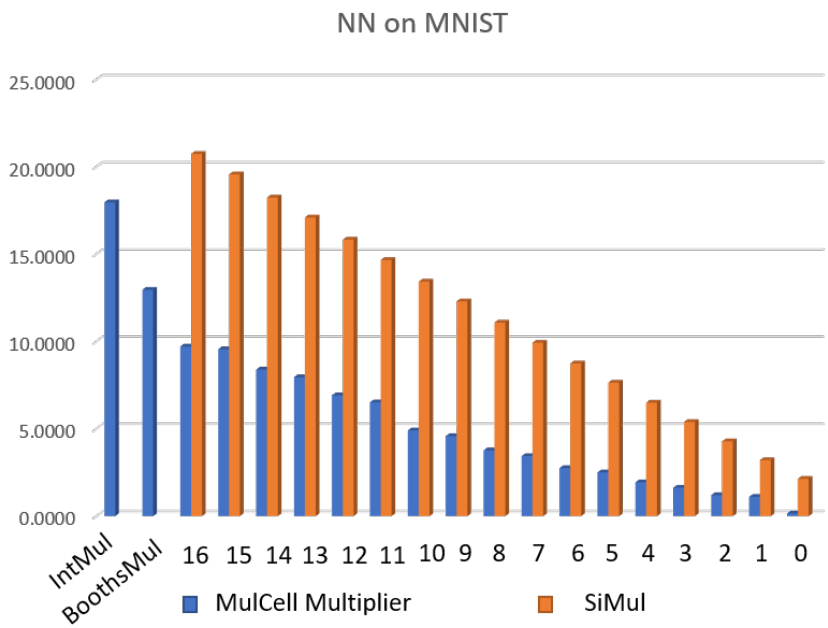


Figure 6.5: Energy reduction for NN MNIST digit recognition

low power simulation using Synopsis VCS using CPF files to ensure expected functionality.

6.3 Calibration results

We evaluate the clustering method by skipping SVM when certain about inference and show that we achieve an energy savings of about 66%. We further save 12% energy by using our $MulCell$ based multiplier. We achieve an overall 78% energy savings just on SVM across benchmarks, as shown in Figure 6.6.

For DWT and Feature computations, we use our basic calibration mechanism to choose the required precision and use this value as input to our approximated multiplier. We see an overall 46% energy savings across benchmarks, as shown in Figure 6.7.

We also analyze the final output error caused by using clustering method for inference and by using approximation for DWT and SVM and plot them for all benchmarks, each using 10 different models of SVM (each using 3 different features from the available features) as shown in the Figure 6.8. We observe only 0.46% accuracy loss on an average and a maximum error of 4.69% for 2 of the models.

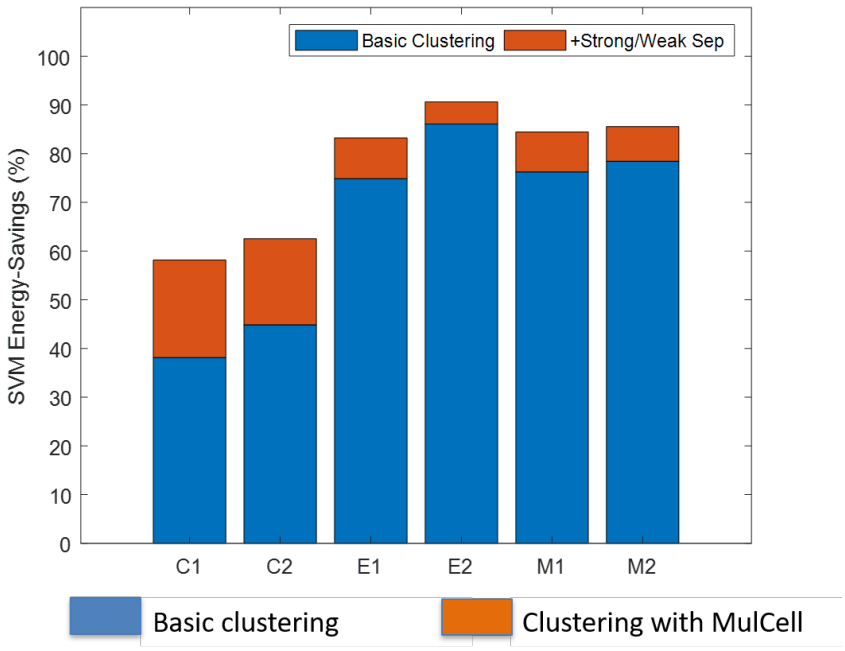


Figure 6.6: Energy reduction for SVM using clustering and $MulCell$ multiplier

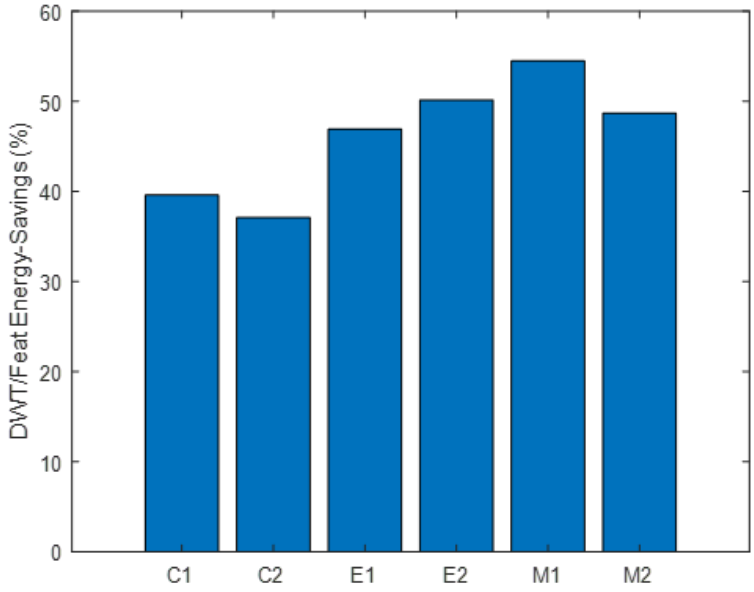


Figure 6.7: Energy reduction for DWT and Feature computation using $MulCell$ multiplier and precision chosen using basic optimization

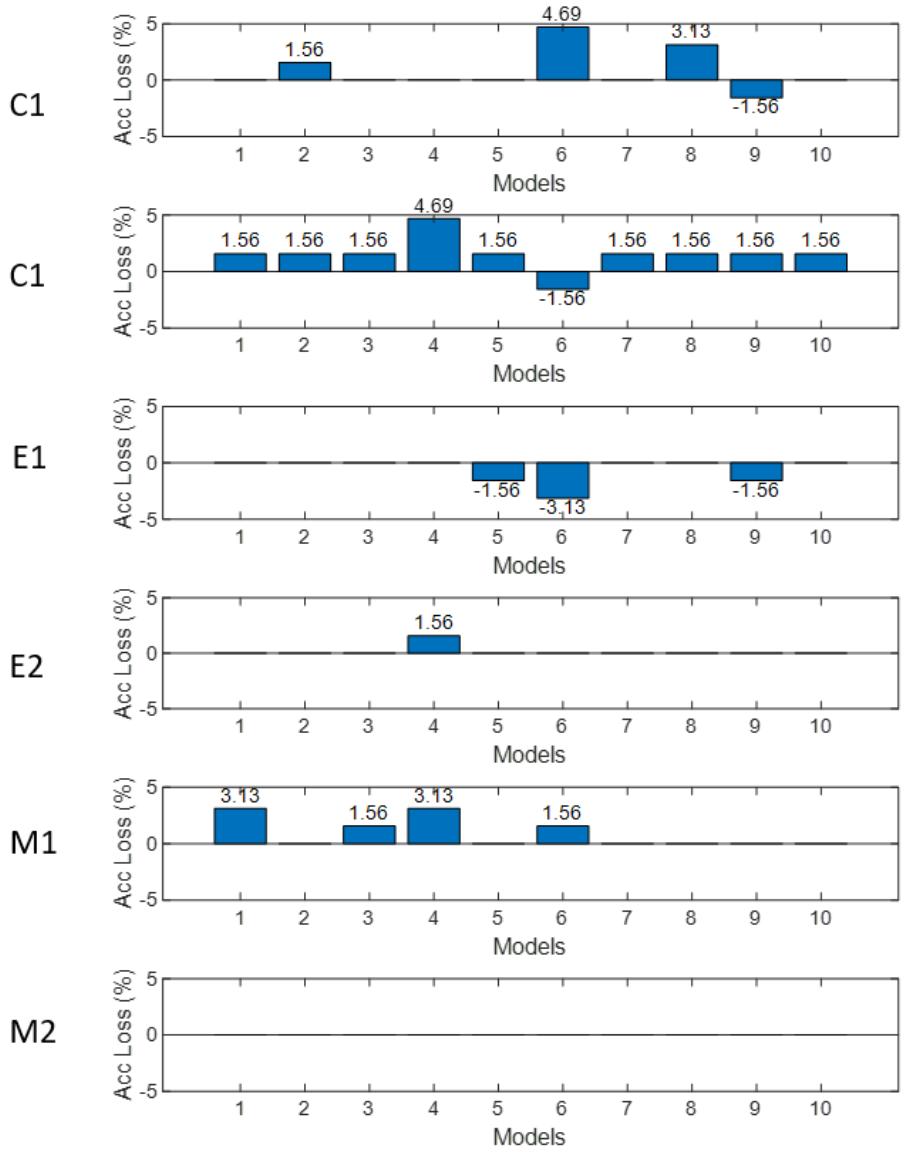


Figure 6.8: Accuracy loss as compared to actual labels for 10 different models

7. FUTURE WORK

We would like to extend the boundary problem to NN to see its effectiveness. We believe that a much simpler model to all of the machine learning models can be used as an estimate about where the data can belong to. We would like to extend the idea of deducing approximation with online calibration to applications outside machine learning too.

8. CONCLUSION

As energy consumption becomes a major issue for battery powered devices, we adopt approximation by truncating bits of multipliers to save power and energy. We show how our hardware design is generic and can be used across applications and is better that the most recent work. We devise a mechanism for self-calibration to determine approximation degree with no loss of accuracy. We adopt the cluster method, wherein we divide the input data into various clusters based on the data sensitivity. We skip inference model and use clustering method to make inference where possible. We show an overall energy savings of 78% for SVM and 46% for DWT and feature computation, with just 0.46% accuracy loss across benchmarks.

REFERENCES

[1] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial intelligence*, vol. 97, no. 1-2, pp. 245–271, 1997.

[2] K. H. Lee and N. Verma, "A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 7, pp. 1625–1637, 2013.

[3] Y. Peng, Z. Li, W. Zhang, and D. Qiao, "Prolonging sensor network lifetime through wireless charging," in *2010 31st IEEE Real-Time Systems Symposium*, pp. 129–139, IEEE, 2010.

[4] Y. Ma, Z. Luo, C. Steiger, G. Traverso, and F. Adib, "Enabling deep-tissue networking for miniature medical devices," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 417–431, ACM, 2018.

[5] D. W. Onstad, "Population-dynamics theory: the roles of analytical, simulation, and supercomputer models," *Ecological Modelling*, vol. 43, no. 1-2, pp. 111–124, 1988.

[6] M. J. Mack, "Minimally invasive and robotic surgery," *Jama*, vol. 285, no. 5, pp. 568–572, 2001.

[7] R. Goffredo, D. Accoto, and E. Guglielmelli, "Swallowable smart pills for local drug delivery: present status and future perspectives," *Expert review of medical devices*, vol. 12, no. 5, pp. 585–599, 2015.

[8] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley, "A brief survey of machine learning methods and their sensor and iot applications," in *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pp. 1–8, IEEE, 2017.

[9] Y. Khan, A. E. Ostfeld, C. M. Lochner, A. Pierre, and A. C. Arias, "Monitoring of vital signs with flexible and wearable medical devices," *Advanced Materials*, vol. 28, no. 22, pp. 4373–4395, 2016.

[10] M. A. Case, H. A. Burwick, K. G. Volpp, and M. S. Patel, "Accuracy of smartphone applications and wearable devices for tracking physical activity data," *Jama*, vol. 313, no. 6, pp. 625–626, 2015.

[11] P. Lukowicz, U. Anliker, J. Ward, G. Troster, E. Hirt, and C. Neufelt, "Amon: a wearable medical computer for high risk patients," in *Proceedings. Sixth International Symposium on Wearable Computers,*, pp. 133–134, IEEE, 2002.

[12] D. Azariadi, V. Tsoutsouras, S. Xydis, and D. Soudris, "Ecg signal analysis and arrhythmia detection on iot wearable medical devices," in *2016 5th International conference on modern circuits and systems technologies (MOCAST)*, pp. 1–4, IEEE, 2016.

[13] A. Wang, L. Chen, and W. Xu, "Xpro: A cross-end processing architecture for data analytics in wearables," in *ACM SIGARCH Computer Architecture News*, vol. 45, pp. 69–80, ACM, 2017.

[14] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[15] Z. Du, K. Palem, A. Lingamneni, O. Temam, Y. Chen, and C. Wu, "Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 201–206, IEEE, 2014.

[16] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *Proceedings of the 35th International Conference on Computer-Aided Design*, p. 81, ACM, 2016.

[17] J. F. Miller, *Cartesian Genetic Programming*, pp. 17–34. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[18] Z. Vasicek and L. Sekanina, "Evolutionary approach to approximate digital circuits design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 432–444, 2015.

[19] S. S. Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy, "Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing," in *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pp. 145–150, EDA Consortium, 2016.

[20] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, pp. 764–775, IEEE Press, 2018.

[21] S. Hashemi, R. Bahar, and S. Reda, "Drum: A dynamic range unbiased multiplier for approximate applications," in *Proceedings of the IEEE/ACM international conference on computer-aided design*, pp. 418–425, IEEE Press, 2015.

[22] B. Shao and P. Li, "Array-based approximate arithmetic computing: A general model and applications to multiplier and squarer design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 4, pp. 1081–1090, 2015.

[23] Z. Liu, A. Yazdanbakhsh, T. Park, H. Esmaeilzadeh, and N. S. Kim, "Simul: An algorithm-driven approximate multiplier design for machine learning," *IEEE Micro*, vol. 38, no. 4, pp. 50–59, 2018.

[24] O. Sakhi, "Face detection using support vector machine (svm)," 2012.