The Thesis Committee for Shirin Fathima
certifies that this is the approved version of the following Thesis:

All-Digital Time-Domain CNN Engine For Energy Efficient Edge Computing

APPROVED BY

SUPERVISING COMMITTEE:

Jaydeep Kulkarni, Supervisor

Mahesh Mehendale

All-Digital Time-Domain CNN Engine For Energy Efficient Edge Computing

by

Shirin Fathima

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2019

Acknowledgments

I would like to thank my supervisor, Prof. Jaydeep Kulkarni, Assistant Professor, Department of Electrical and Computer Engineering, The University of Texas at Austin for providing with the wonderful opportunity to work on this project. I would like to thank him for his guidance and motivation throughout the project.

I would also like to thank Aseem Sayal, Ph.D. Student, Department of Electrical and Computer Engineering, for working with me in this project. I am grateful for his contributions to the project, constant support and guidance as a senior student and colleague. I would like to convey my gratefulness to S.S. Teja Nibhanupudi for his contribution in the project and helping me understand Cadence tools.

I would like to express my gratitude towards my parents and friends for their encouragement and co-operation which has helped me in completing this project. And above all, to the Almighty God, who gave me the strength, continued guidance and protection.

All-Digital Time-Domain CNN Engine For Energy Efficient Edge Computing

Shirin Fathima, M.S.E.
The University of Texas at Austin, 2019

Supervisor: Jaydeep Kulkarni

Machine Learning is finding applications in a wide variety of areas ranging from autonomous cars to genomics. Machine learning tasks such as image classification, speech recognition and object detection are being used in most of the modern computing systems. In particular, Convolutional Neural Networks (CNNs, class of artificial neural networks) are extensively used for many such ML applications, due to their state of the art classification accuracy at a much lesser complexity compared to their fully connected network counterpart. However, the CNN inference process requires intensive compute and memory resources making it challenging to implement in energy constrained edge devices. The major operation of a CNN is the Multiplication and Accumulate (MAC) operation. These operations are traditionally performed by digital adders and multipliers, which dissipates large amount of power. In this 2-phase work, an energy efficient time-domain approach is used to perform the MAC operation using the concept of Memory Delay Line (MDL). Phase I of

this work implements LeNet-5 CNN to classify MNIST dataset (handwritten digits) and is demonstrated on a commercial 40nm CMOS Test-chip. Phase II of this work aims to scale-up this work for multi-bit weights and implements AlexNet CNN to classify 1000-class ImageNet dataset images.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine Learning (ML) approach is marching in every field and finding application areas limited only by human imagination. As such, all computing systems are being designed with a significant focus on a variety of ML tasks such as image classification, speech recognition, object localization, etc [1]-[2]. Convolutional Neural Networks (CNNs, class of artificial neural networks) are extensively used for many such ML applications, due to their state of the art classification accuracy at much lesser computation compared to their fully connected network counterpart. CNN parameters (called as filter weights) are determined by efficient training algorithms which are typically executed using high-performance cloud resources. The trained filter weights are transferred on to the edge compute device and are utilized during the CNN inference process for estimating various input features. However, CNNs still require intensive compute and memory resources making it challenging to deploy it in energy constrained edge devices. For example, well known convolutional neural networks like AlexNet, VGG comprises of 60 million and 138 million parameters respectively [3]-[4]. These parameters are used to compute 724 million and about 15.5 billion multiply-accumulate (MAC) operations for AlexNet and VGG networks respectively [5]. These high number of MAC operations con-

sume a significant amount of computing power. Hence, there is a critical need to research and develop efficient computing capabilities to perform machine learning tasks in an edge device for improved energy efficiency, security, and privacy.

Various analog compute techniques using charge manipulation schemes and A/D converters have been proposed to realize efficient MAC computations in a CNN accelerator (Fig. 1.1). Analog approaches [6]-[8] compute MAC operation in analog voltage domain using SRAM array, capacitors, and data converters. In this case, input data is encoded in the pulse-width modulated wordline [7] or pulse amplitude modulated wordline [8]. The MAC operation is computed as the bitline voltage which represents the sum of dot product (SRAM bit weight * Wordline Input). This design approach [6] is susceptible to process variations of the nano-scale SRAM bitcell transistors as well as functional failure due to bit-flips as a result of long duration wordline activation, large bitline voltage differential and possibility of corrupting a weak bit by connecting it to a strong bit storing opposite value.

In [9]-[10], MAC operation is performed in digital domain using multipliers and adders; resulting in high access energy cost due to data movement. In [11]-[13], MAC operations are computed in frequency domain using digital controlled oscillator (DCO) with either resistor or capacitor loading. Various nodes of a ring oscillator are loaded with different capacitor banks to alter the RC time constant of the oscillator. Capacitor value is controlled by the SRAM bitcells storing the neural network weight. This approach makes the

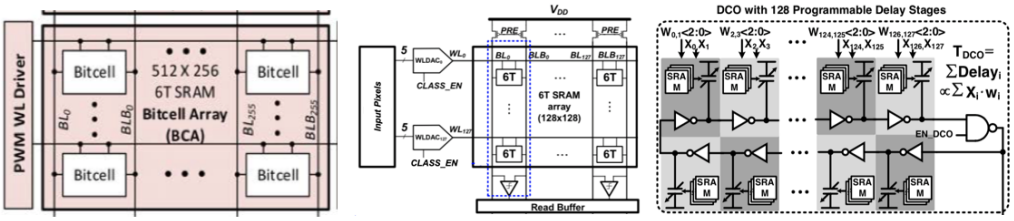


Figure 1.1: Prior Work: Analog voltage domain [7]-[8] and Frequency domain MAC [11] approach.

design sensitive to parasitic diffusion capacitances of the DCO and the added R, C components need to be larger than this diffusion capacitances to linearly modulate ring oscillator frequency in response to the weight change. In addition, adding capacitor at every node of a continuous running ring oscillator increases the total switching capacitance and consequently higher power dissipation. Furthermore, implementing such binary weighted capacitor banks consumes significant layout area degrading the area efficiency of the prior frequency domain approach.

In this work, a time domain MAC computing approach is proposed by leveraging the concept of a time accumulator. In phase I[1], an energy efficient CNN engine implemented in commercial 40nm CMOS process (Fig.

[1]Aseem Sayal, Shirin Fathima, S.S. Teja Nibhanupudi and Jaydeep P. Kulkarni, "All-Digital Time-Domain CNN Engine Using Bidirectional Memory Delay Lines for Energy-Efficient Edge Computing," *in Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pp.228-230, San Francisco, CA, USA, 2019. implementing time-domain based CNN Engine. In this work, my contributions include pre-silicon functional simulation/verification of all the circuit modules, post-silicon testing and debugging to obtain experimental results, and software development to implement LeNet-5 CNN in Keras/TensorFlow.

3.1) is demonstrated [14]. Key attributes of the proposed design are: (1) Bi-directional Memory Delay Lines (MDL) performing time domain signed MAC operations (2) multi-precision filter weight support (signed/unsigned 1-8bits) (3) 16 filters each supporting 2x2 sub-sampling (max. pooling) and averaging (4) all-digital, technology scalable design without requiring any capacitors, A/D converters, and/or frequency generators/modulators (5) near threshold voltage operation and 1x-16x speed-up with 4 input encoding modes. In phase II, the proposed work is scaled up to implement AlexNet CNN supporting multi-bit weights.

1.1 Background

This section first briefly discusses the Convolutional Neural Networks (CNNs), specifically LeNet-5 architecture. Next, data representation in different signal domains such as digital, analog voltage, frequency and time-domain are discussed. Finally, concept of time-domain MAC computation is presented.

1.1.1 Convolutional Neural Networks

As discussed briefly in the previous section, a Convolutional Neural Network (CNN) [15] is a class of deep learning artificial neural networks. Such networks typically consist of input layer, output layer, and hidden layers, where each hidden layer implements multiple filters. LeNet-5 CNN [16] architecture is shown in Fig. 1.2 which is used to classify the hand-written digits (MNIST database [16]). In the training phase, the weights of all the filters for every

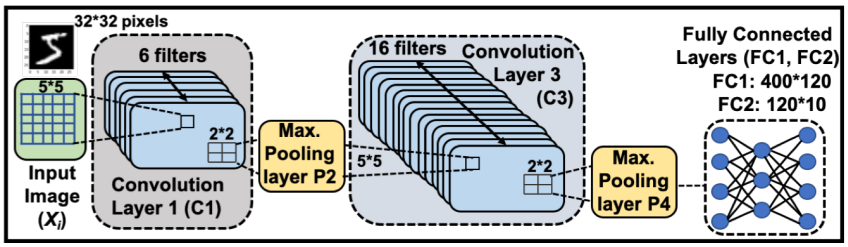


Figure 1.2: LeNet-5 CNN Architecture [14].

layer are determined using a training algorithm (such as back-propagation [17]-[18]) on training set of MNIST data. During inference phase, when a test data is presented to the CNN layers, key features (e.g. lines, orientation in a handwritten digits MNIST dataset) are extracted at each layer. The major operation in CNNs is the Multiple and Accumulate (MAC) operation given by (1.1) which is computed by adding of dot products of weight matrix and input image matrix [5]. The MAC value is averaged out to compute Multiply-Accumulate-Average (MAV) value (given by (1.2)) which ensures that output value doesn't go out of range.

$$MAC = \sum_{i=1}^{N} \left(X_i * w_i\right) \tag{1.1}$$

$$MAV = \frac{1}{N} \sum_{i=1}^{N} \left(X_i * w_i\right) \tag{1.2}$$

where N is the number of dot products per MAC, X_i is the i^{th} input pixel value and w_i is the i^{th} weight value.

Typically, an activation layer is used between two consecutive convolu-

tional layers. The Rectified Linear Unit (ReLU) is the most commonly used activation layer which returns 0 if it receives any negative input, and returns back the value, if it is positive. Pooling (sub-sampling) is performed to reduce the data size feeding into the next layer [19]-[20]. Once the data size of input feature maps is reduced after passing through convolution and pooling layers, fully connected convolutional network (FCN) layers are used. From the energy efficiency perspective, these networks compute millions/billions of Multiply-Accumulate (MAC) operations which consume 90% of the total CNN energy consumption [5]. Hence, it becomes of utmost importance to devise design solutions for energy efficient MAC computations.

1.1.2 Motivation for Time-Domain Computation

As MAC operations constitute a significant portion of the total CNN power budget, it is worthwhile to consider various methods for compact data representation to improve the energy efficiency (as shown in Fig. 1.3). The data in digital domain is represented as multi-bit digital vector. This approach incurs large number of signal toggling resulting in high dynamic switching capacitance and consequently higher power and area. This makes it challenging to implement such technique for energy constrained edge computing devices. In analog voltage domain, data is represented as a continuously varying analog signal [6]-[8]. However, finite voltage headroom and the sensitivity of circuit parameters to slight change in analog domain signals limit the voltage scalability; thereby degrading the MAC energy efficiency. In frequency domain,

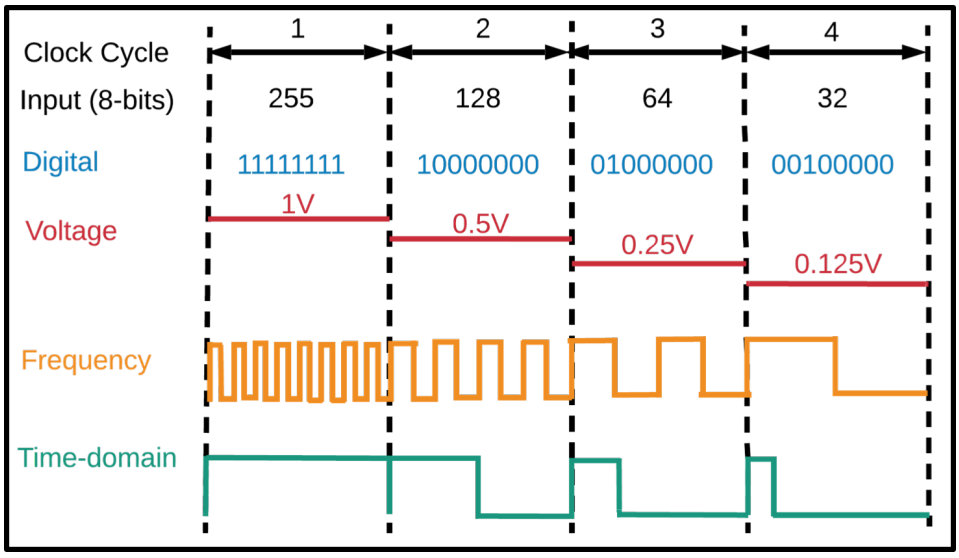


Figure 1.3: Data representation in digital bits, analog voltage, in frequency domain and in time domain.

data is represented with signals with varying frequency using Ring oscillators or RC loaded circuits [11]-[13]. However, accurate frequency generators and modulators limit the performance scalability of frequency domain approaches.

In time-domain computation, a multi-bit digital bit-stream is encoded as a single pulse-width modulated signal. It significantly reduces the toggle activity of various signals leading to a smaller dynamic switching capacitance (C_{DYN}) compared to the conventional digital data representation. Unlike the analog voltage domain approach, time domain approach is not susceptible to reduced voltage headroom issues and can support ultra-low voltage operation. Additionally, time domain approach doesn't require multiple clock sources unlike frequency domain approaches making it suitable for compact implementations. Thus, time domain processing although slow in operation, is a

promising approach for MAC computations especially in energy constrained edge devices.

1.1.3 Concept of Time-Domain MAC Computation

The concept of time domain MAC computation using Pulse Width Modulated (PWM) inputs is illustrated in Fig. 1.4. In this example, the MAC operation is performed by adding the dot-products of a 3x3 weights matrix and input pixel matrix (Fig. 1.4.a,c). Binary weights [21]-[23] are used to simplify the discussion. However, the time domain approach is scalable to multi-bit weight values. An input pixel value is encoded into a PWM signal using a Digital to Time Converter (DTC). Then, the PWM signal is connected to an

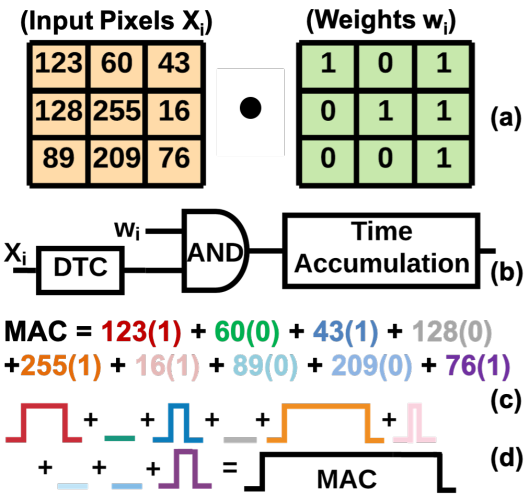


Figure 1.4: Concept of time-domain MAC operation, (a) Dot product operation of input pixel matrix and weight matrix, (b) Time-domain MAC circuit concept, (c) MAC operation in digital domain, and (d) MAC operation in time-domain [14].

AND gate with the other input of the AND gate connected to a weight bit to perform dot product multiplication. This time-encoded dot-product signal is then passed on time accumulator circuit (Fig. 1.4.b) to compute MAC operation in time-domain. Sequentially, all input pixel values are encoded as PWM signals and its dot product with respective weight bit is loaded on to the time accumulator circuit. Thus, all values are applied and the width of time pulse gets added to realize MAC operation in time domain (Fig. 1.4.d). The time accumulator circuit will be discussed in detail in the next section.

Chapter 2

Design and Architecture of Time Domain CNN Engine (Phase/Test-chip I)

This chapter deals with the design and architecture of our first test-chip.[1]

2.1 Architecture Overview

The proposed idea deals with the computation of multiply-accumulate (MAC) operation in time domain to perform energy efficient computing. Fig. 2.1 shows the high-level idea of the proposed architecture for 8-bit inputs and 1-bit weight neural network. Major components of this architecture along with the data flow path is outlined below.

First, pulse-width modulated (PWM) signals which are generated using a pulse generator, are selected based on the input 8-bit image pixel value. The

[1]Aseem Sayal, Shirin Fathima, S.S. Teja Nibhanupudi and Jaydeep P. Kulkarni, "All-Digital Time-Domain CNN Engine Using Bidirectional Memory Delay Lines for Energy-Efficient Edge Computing," *in Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pp.228-230, San Francisco, CA, USA, 2019. implementing time-domain based CNN Engine. In this work, my contributions include pre-silicon functional simulation/verification of all the circuit modules, post-silicon testing and debugging to obtain experimental results, and software development to implement LeNet-5 CNN in Keras/TensorFlow.

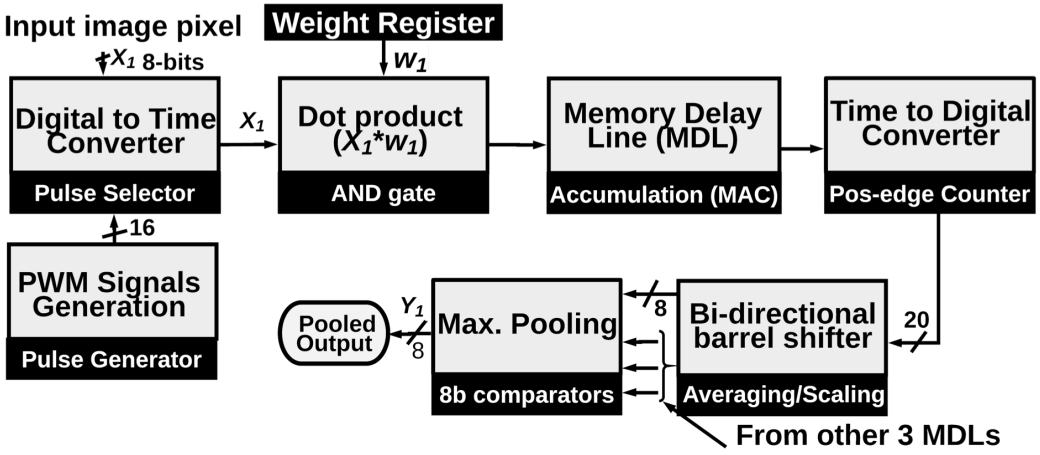


Figure 2.1: Time domain CNN architecture using the proposed Memory Delay Line (MDL).

pulse selector acts as Digital-to-Time (DTC) converter; converting the digital image pixel value into time encoded signal. Higher the pixel value; larger is the pulse-width of time encoded input. This free-running PWM signal from the pulse generator is passed through the pulse gating logic so that this time-encoded signal is utilized only once in the dot product calculation making sure no repetitive dot products are added to a given MAC output.

This gated PWM signal is then multiplied with the binary weight (0 or 1 or -1) using an AND gate; performing the dot product. Multiplication by 0 weight results in no toggling at the AND output while multiplying by weight ± 1 result in PWM output at the AND gate same as the input PWM signal. The time-encoded dot product signal is then added using a novel bi-directional memory delay line (MDL) to perform the signed addition of the dot products for each MAC operation. An up-down counter is followed by

the MDL to convert time-encoded MAC signal into digital value. It acts as a Time-to-Digital converter for performing post-processing in the digital domain. Using an up-down counter, a finite length MDL can be used to perform long duration time domain accumulation. This signal is right/left shifted using a bi-directional barrel shifter to perform averaging after MAC operation (MAV). This is also done to correctly scale the MAC output before feeding as the input to the next convolution layer. Once we obtain the shifter output value from 4 MDL units for each filter, the max. pooling operation (2x2 window) [14] is performed using 8-bit comparators. The pooled output from each filter is stored off-chip and reused as the input to the next convolution layer.

2.2 Design of Bi-directional Memory Delay Line

The time accumulation step in the proposed time domain MAC computation is based on the concept of time register and time adder.

2.2.1 Concept of Time Register

The basic goal of the time-register is to be able to store time and retrieve it when needed. The time-register can be implemented by a series of gated delay cells, called gated delay-line (GDL), and an OR-gate as shown in Fig. 2.2 [24]. Unlike a conventional delay-line, a GDL has a gating property using an enable pin (EN) connected to header/footer transistors. The EN signal controls the propagation of the SET signal in the GDL and can be high by either the input or the trigger signal. The phase of the GDL is increased

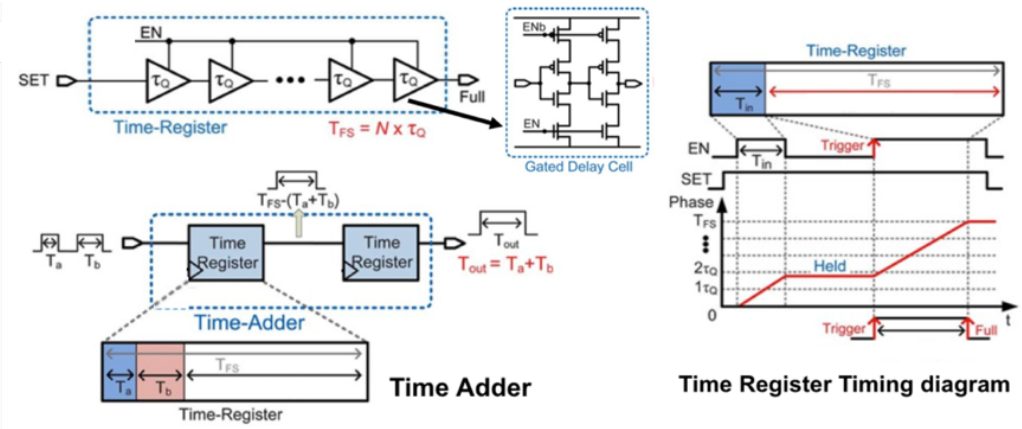
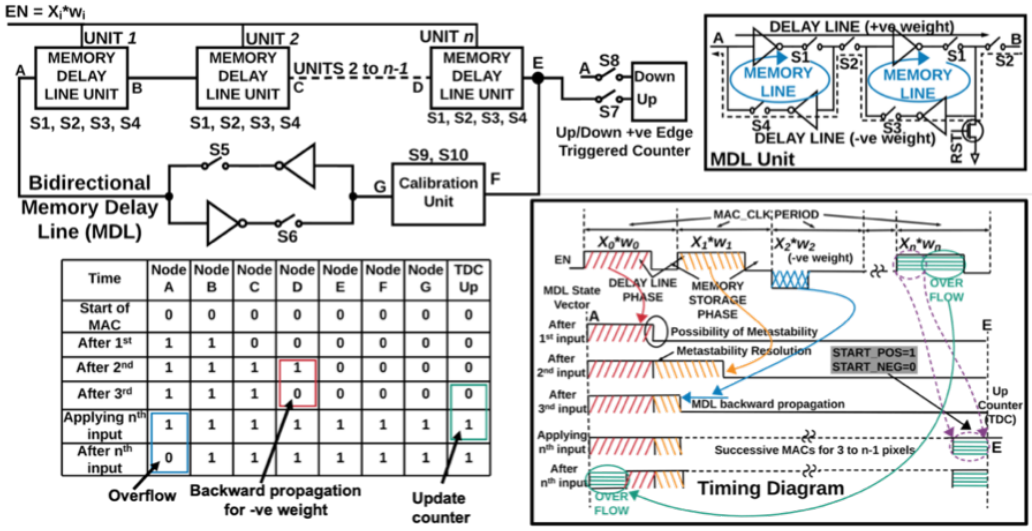


Figure 2.2: Time Register and Time Adder concept [24].

when EN is high and held to its previous value when it is low. Hence, when an input pulse is received, the phase of the GDL is advanced by the amount of the input pulse-width and held when input pulse goes low. When the SET signal edge reaches the end of the GDL, a FULL signal is asserted. The capacity of time-storage can be enlarged by simply increasing the number of delay cells. An interesting property of the time-register is that it can also be used for time addition & subtraction. Suppose two inputs of pulse-widths are sequentially fed to the time-register (Fig. 2.2). The phase of the time register is advanced by the sum of the two pulse-widths performing a time domain addition.

2.2.2 Memory Delay Line

The time-domain MAC computation in the proposed work is derived from the time-register concept. The property of time addition of two different input pulses is utilized to perform the MAC operations in a CNN layer.



Time	Node A	Node B	Node C	Node D	Node E	Node F	Node G	TDC Up
Start of MAC	0	0	0	0	0	0	0	0
After 1st	1	1	0	0	0	0	0	0
After 2nd	1	1	1	1	0	0	0	0
After 3rd	1	1	1	0	0	0	0	0
Applying nth input	1	1	1	1	1	1	1	1
After nth input	0	1	1	1	1	1	1	1

Figure 2.3: Illustration of the proposed bi-directional MDL concept [14].

The time domain MAC computations are realized using a novel Bi-directional Memory Delay Line (MDL) which accumulates the dot product of weight bit and the time encoded input pulse-width (Fig. 2.3). Each MDL unit comprises of two cross coupled inverter pairs, S1-S4 switches, and a Reset logic. The time encoded dot product of input (X_i) and 1-bit weight (w_i) acts as EN pulse and controls MDL operating mode. During time-accumulation phase, EN=1 and MDL acts either as a forward delay line (for positive dot product) or a backward delay line (for negative dot product) thus enabling bi-directional data flow emulating signed dot products. When EN goes low, the MDL acts as a memory storage line and retains the MDL state vector using cross-coupled inverters. The metastability risk during EN falling transition is resolved by the next incoming EN pulse as the MDL is transformed into a chain of cascaded delay cells. When the MDL state vector string progresses towards the

either end of MDL (node A or node E), an up- down counter is triggered which translates time domain dot product accumulation information into digital bits acting as a time-to-digital converter. If the accumulated dot product pulse-width exceeds the full-scale MDL delay, an overflow condition is detected, and propagating edge is inverted (using S5-S6) and applied at the beginning of MDL (node A). Thus, a finite length MDL can be used to perform long duration time domain accumulation using up-down counters. The calibration unit consists of additional delay cells which can be added to original MDL to mitigate MDL mismatch in the presence of process variations.

2.3 Design Verification and Software Implementation

The specific contributions of this thesis work are detailed in this section. The design involved a culmination of several modules such as the pulse generator and selector, weight shift registers, MDL, scan chain, counter, shifter and comparators. Each of them was functionally tested at several states individually as well as a top level integrated design. The simulation graphs for different levels of hierarchical testing are shown in this section. Fig. 2.4 shows the functional simulation graph for the pulse generator at the 4 speed-up modes, Fig.2.6 shows a filter-level testing and Fig.2.7 shows the top level testing for C1 layer. Major modules worth noting are as follows:

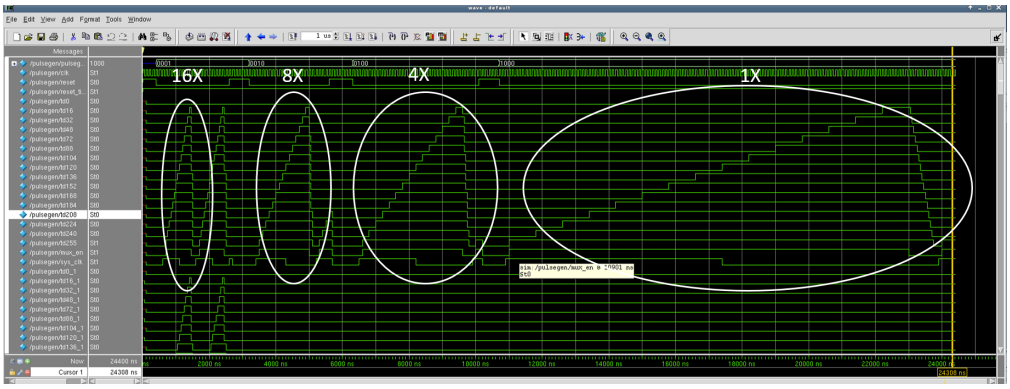


Figure 2.4: Functional Simulation of Pulse Generator at the 4 speed up modes - 16x, 8x, 4x and 1x.

2.3.1 Pulse generator and Selector:

In this design, the 8-bit inputs are encoded as PWM signals. 16 free-running PWM signals denoted as td0 to td255 in Fig. 2.4 and td0 to td16 in Fig. 2.5 are generated in pulse generator module such that the pulse width of each signal increments by 17*t_o and varies in the range of 0*t_o to 255*t_o. The minimum possible pulse width equals half period of input clock. These pulses are generated in two phases: MSB phase and LSB phase. In MSB phase, pulse width is incremented by 16*t_o whereas in LSB phase, it gets incremented by to. The pulse selector module acts a Digital to time converter which converts the input into time encoded PWM signal. The appropriate pulse is selected by the pulse selector as shown in Fig. 2.5.b. A functional simulation to test all 4 speedup modes of the pulse generator has been done pre-tapeout as shown in Fig. 2.4.

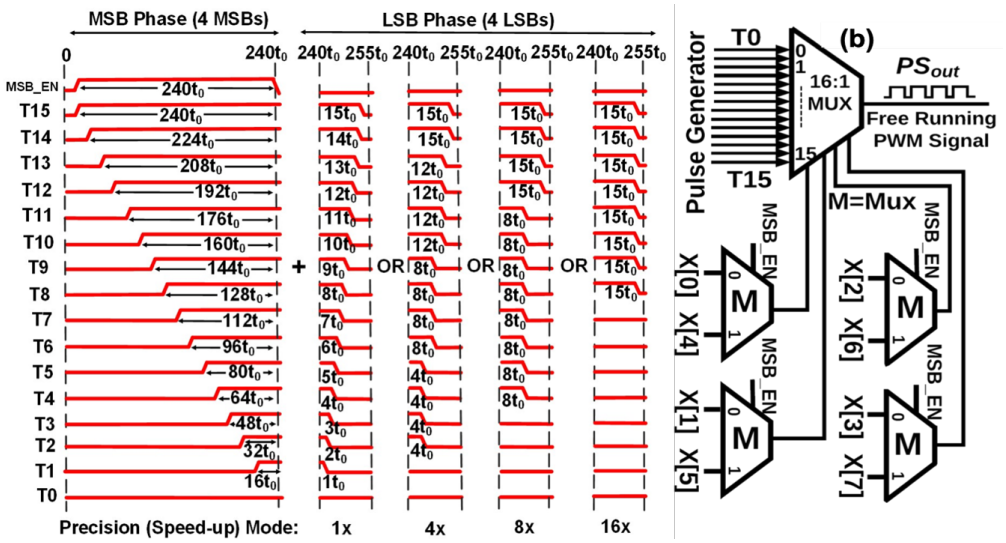


Figure 2.5: (a) PWM signals generated by Pulse Generator module in 1x-16x speedup modes, (b) Pulse Selector Logic [14].

2.3.2 Time to Digital Converter

A 20-bit positive edge triggered, up-down counter is used to convert the time-encoded MAC value into digital domain. The counter value is incremented when 0→1 transition occurs at node E (if switch S7 is turned on and SIGN=1), and is decremented when 0→1 transition occurs at node A (if switch S8 is turned on and SIGN=0). Fig. 2.6 shows the 4 MAC counter values (20-bit) computed for one filter.

2.3.3 Bi-directional Barrel Shifter

The counter output value is fed to the 20-bit bi-directional barrel shifter (supporting left shift and right shift up to 7 bits) to compute multiply-accumulate-average (MAV) and scaling operations. The time-encoded input-pixel value

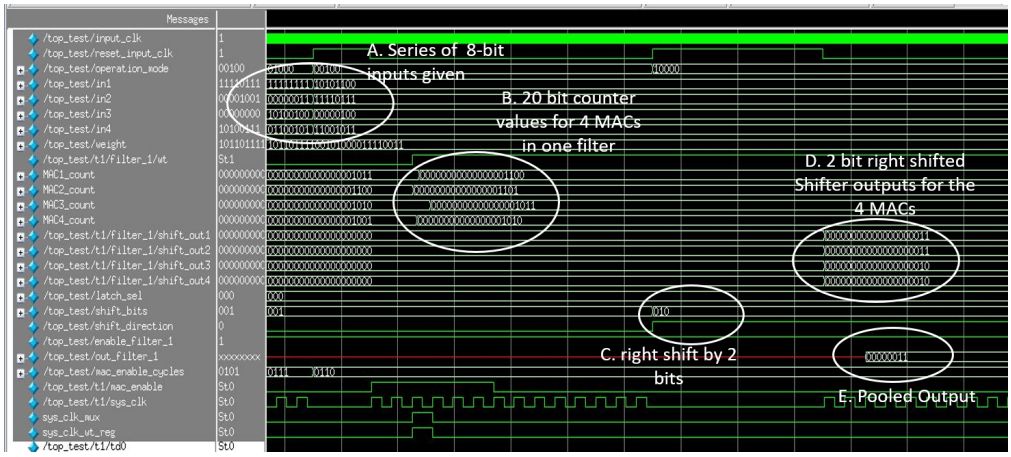


Figure 2.6: Filter level functional testing

and counter output values are represented in different time scales. Since the counter output value does not represent the correct MAC value, a scaling operation is essential at this point to restore back the correct MAC value. This will ensure a correct MAC output in terms of counter values to be applied as inputs to the next convolutional layer. The counter output needs to be multiplied by a scaling factor depending on the MDL full-scale value to reflect the actual MAC output. The shifter gets this scaling factor into the values.

To perform the averaging operation, right shift by appropriate number of bits is performed. It is worth mentioning that the averaging factor in the MAV computation operation is a multiple of 2^m, such that $2^m \geq N$ where N is the number of dot-products in a MAC. The CNN is trained considering 2^m as an averaging factor to observe no loss in the classification accuracy during the inference process. The shifter operation can be seen in Fig. 2.6.c-d which shows a right shift by 2 bits.

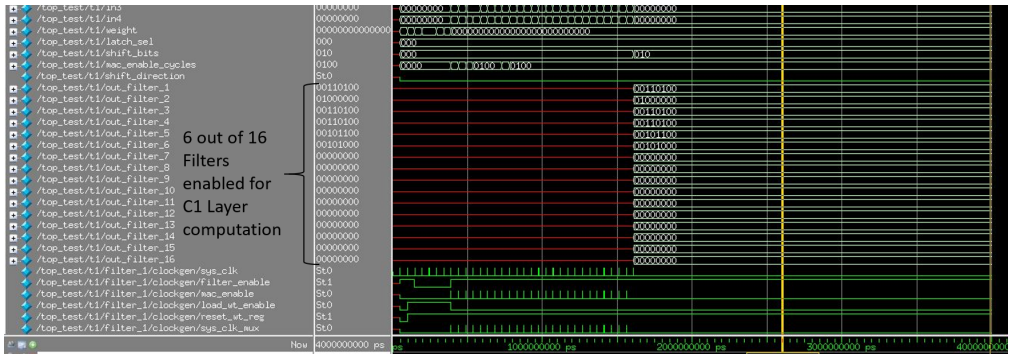


Figure 2.7: Top level testing for C1 layer

2.3.4 Pooling using 8-bit comparators

The sub-sampling operation using max-pooling across 2x2 window is implemented to reduce the intermediate layer output memory footprint by 75%. This is achieved by 4 concurrent MDL operations and feeding the MDL shifter outputs to three 8-bit comparators. The pooled output from each filter is stored off-chip and reused as the input to the next convolution layer. Fig.2.6.e shows the pooled output for the filter which is the maximum of the four shifter outputs.

2.3.5 Top level

The complete flow of the design as shown in Fig. 2.1 was functionally tested. Both C1 and C3 layers were functionally tested in the sequence the design was expected to run. An example simulation graph is shown in Fig. 2.7 which shows the C1 layer computations with 6 filters enabled.

2.3.6 Software Implementation

Fig. 2.8 shows the overall data flow and measurement setup of the test-chip implemented using commercial 40nm CMOS process technology [14]. The initial training of the MNIST data using the LeNet-5 architecture was done using the Keras Tensor Flow [25]. These training scripts were generated for 2 different cases - pure software and hardware based. In the hardware based simulations, the effect of residue in MDL, input quantization and bit precision have been taken into account while training in all the 4 speed-up modes. 8-bit fixed point input and weights were used in convolution and pooling layers, whereas 16-bit floating point inputs and weights were used in fully connected network (FCN) and soft-max layers. The training dumps out the required weights that needs to be used for inference. The accuracy results obtained through the Tensor Flow for pure software results are shown in Fig. 3.4 as S/W. The hardware based simulations gave an accuracy exactly as the actual experimental accuracy in all the 4 speed-up modes.

Similarly, a Keras Tensor flow approach was taken to analyse the AlexNet architecture. The scalability analysis of the proposed time-domain MAC approach was performed for AlexNet CNN by incorporating the MDL residual delay effects and input quantization effects. 2-class subset of ImageNet database (cats vs. dogs) was used as the dataset. 8-bit fixed point input and weights were used in convolution and pooling layers, whereas 16-bit floating point inputs and weights were used in fully connected network (FCN) and soft-max layers. 8 MDLs are used to perform time-accumulation of dot-products of

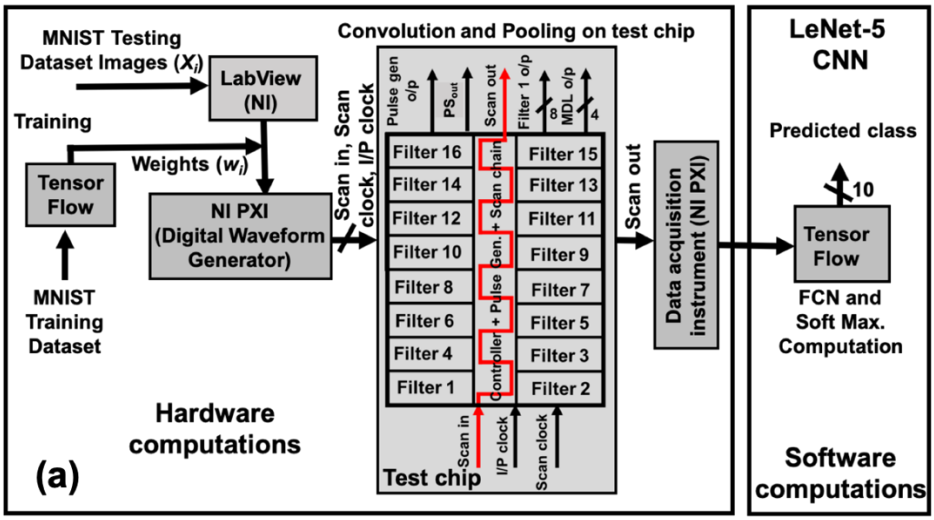


Figure 2.8: (a) Data flow of the proposed time-domain CNN engine [14].

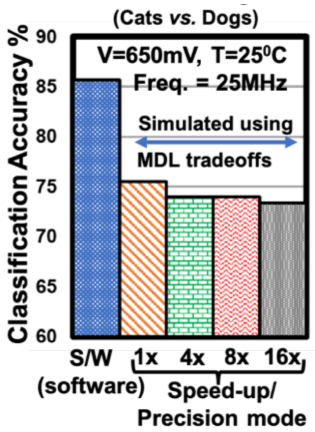


Figure 2.9: Simulated Classification Accuracy on AlexNet over 2-class ImageNet dataset for different speed-up modes [14].

8-bit weight and time-encoded input pixel value. 13% classification accuracy loss is observed in the simulations when compared with software implementation (Fig. 2.9). This can be attributed to the fact that residue loss in timing accumulation occurs for each weight bit, and gets added since 8 independent

MDL operations. Moreover, residue loss is significantly high for MSB bits of weight signal.

Chapter 3

Experimental Results (Phase/Test-chip I)

This chapter presents the experimental results of our phase I 40nm test-chip[1] implementing time-domain CNN Engine. First, measurement setup and test-chip summary is described. Next, experimental demonstration of MDL, pulse-generation and gating logic modules is presented. Then, experimental accuracy, throughput and energy efficiency results for LeNet-5 CNN over MNIST dataset images are described. Finally, the comparison of the proposed time-domain approach with state-of-the-art analog, digital, frequency and time domain approaches is presented.

3.1 Measurement Setup and Test-chip Summary

Fig. 3.1 shows the measurement setup and die-micrograph of our test-chip implemented using commercial 40nm CMOS process technology [14]. The

[1]Aseem Sayal, Shirin Fathima, S.S. Teja Nibhanupudi and Jaydeep P. Kulkarni, "All-Digital Time-Domain CNN Engine Using Bidirectional Memory Delay Lines for Energy-Efficient Edge Computing," *in Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 228-230, San Francisco, CA, USA, 2019. implementing time-domain based CNN Engine. In this work, my contributions include pre-silicon functional simulation/verification of all the circuit modules, post-silicon testing and debugging to obtain experimental results, and software development to implement LeNet-5 CNN in Keras/TensorFlow.

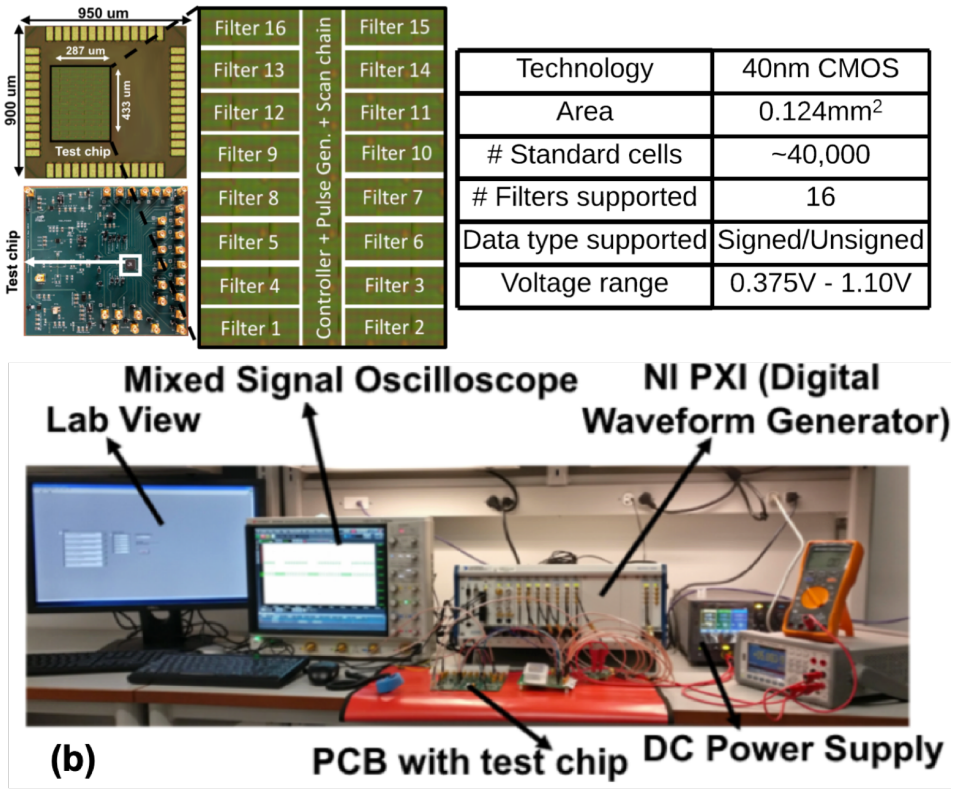


Figure 3.1: (a) Die micrograph, Summary and (b) Lab measurement setup of 40nm test-chip implementing time-domain CNN engine [14].

test-chip implements the LeNet-5 CNN architecture and occupies total area of $0.124mm^2$. The trained filter weights and validation set MNIST database image pixel values are fed to the test-chip using LabVIEW-PXIe data acquisition instruments [26].

3.2 Test-Chip Characterization

The experimental demonstration of MDL behavior under delay phase and memory phase for different MDL lengths (32, 48 and 64 units) is shown in Fig. 3.2.a. The oscilloscope capture confirms successful operation of the proposed MDL concept. During the MDL delay phase (when *EN* is held high), the state vector on MDL is advanced; thereby resulting in transitions, whereas MDL state is held constant during the memory phase (when *EN* is low). The pulse gating logic waveforms are shown in Fig. 3.2.b. confirming

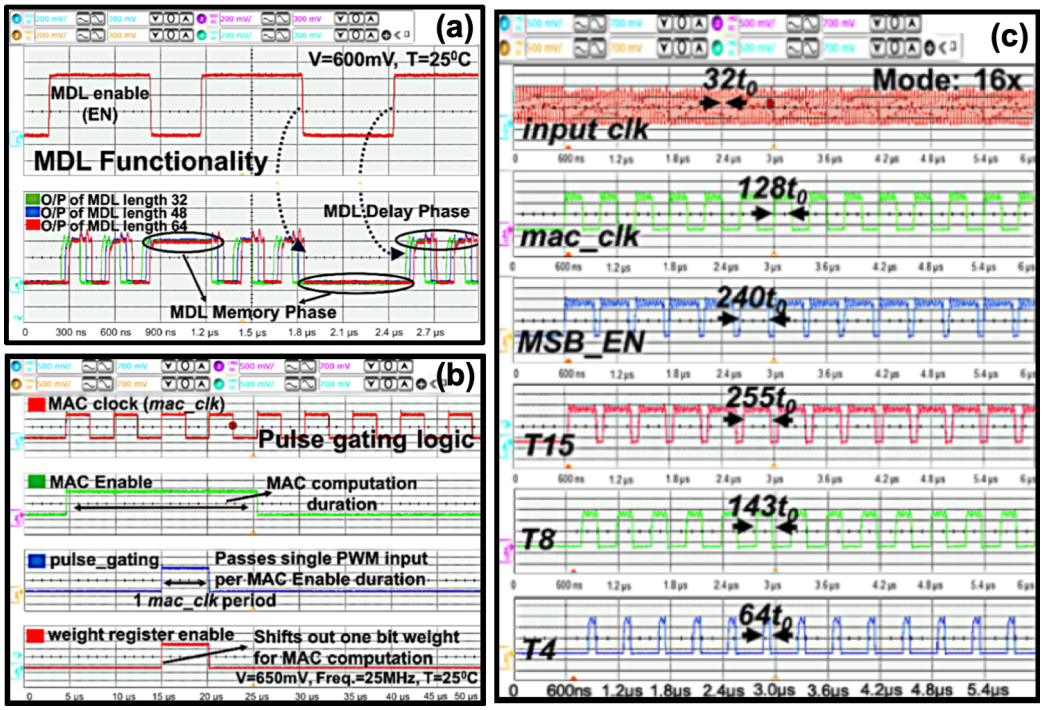


Figure 3.2: Measured (a) MDL waveforms demonstrating delay and memory phases, (b) Pulse gating logic module waveforms, and (c) PWM signals generated from Pulse Generator module [14].

Table 3.1: Parameters of LeNet-5 Convolution layer C1 and C3 layers [14].

Parameters for LeNet-5	C1	C3
Filter Size	5*5*1*6	5*5*6*16
Input/Filter bit width	8bits/ 1bit	8bits/ 1bit
Input Size	32*32*1	14*14*6
Output Size	14*14*6	5*5*16
#Filters	6	16
#Operations/convolution*	(25*4*6)*2	(150*4* 16)*2

time-encoded input pixel value is used only once in the dot product calculation. The pulse generator module functionality is verified with the correct toggling of *MSB_EN*, *T15*, *T8*, and *T4* outputs in 16x speed-up mode, as shown by Fig. 3.2.c. 1x-16x speed-up in PWM input representation is validated with multiple speed-up modes for a test-case input of 214 (Fig. 3.3). In 1x speed-up mode, 128 input clock cycles are used to time-encode input pixel value of 214 accurately. However, pixel value 214 is encoded 4 times in 4x speed-up mode for same number of 128 input clock cycles, thereby increasing the throughput by 4x but with a quantization error of 2. Similarly, 16 and 8 cycles of input clock are required in 8x and 16x speed-up modes to time-encode 214-pixel value as 216 and 208 respectively. Thus, the PWM throughput increases with higher speed-up mode at the expense of higher quantization error.

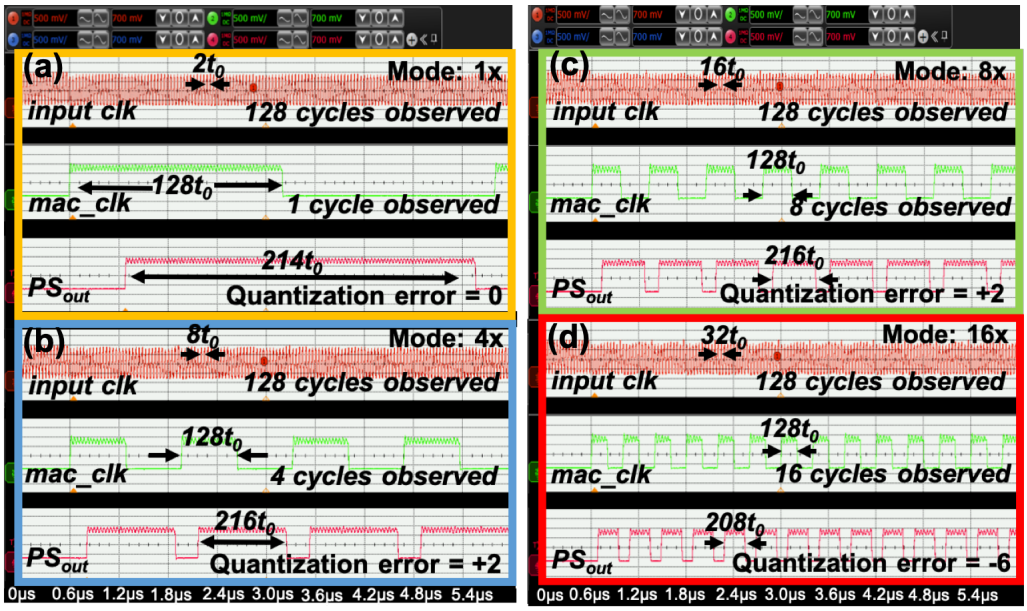


Figure 3.3: Experimental demonstration of (a) 1x, (b) 4x, (c) 8x, and (d) 16x speed-up modes [14].

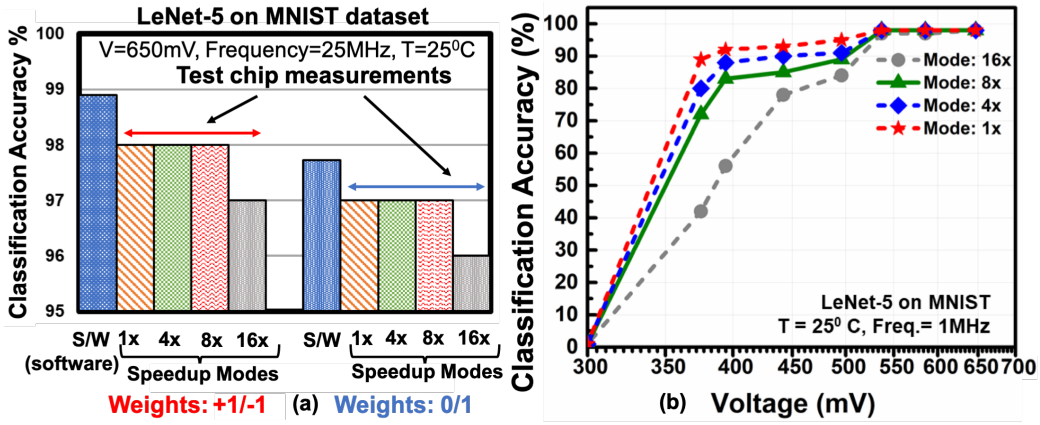


Figure 3.4: Measured classification accuracy on LeNet-5 CNN over 100 MNIST dataset test-images for 4 speed-up modes and: (a) signed/unsigned weights at 650mV, and (b) signed weights for 300mV-700mV range [14].

3.3 Accuracy, Throughput and Energy Efficiency

The classification accuracy is measured for LeNet-5 CNN over 100 MNIST dataset images. Table 3.1 lists down key LeNet-5 network parameters. 8-bit fixed point input pixel values and binary signed/unsigned weights are used in the convolution layers, whereas 16-bit floating point inputs and weights are used in fully connected layer and software implementation. For signed binary weights (± 1), 98% classification accuracy is obtained in 1x-8x speed-up modes whereas it gets dropped by 1% in 16x speed-up mode (Fig. 3.4.a). 16x speed-up mode resulted in lower accuracy because of input quantization and increased sensitivity of MDL residue. For unsigned weights (0 and 1), 1% drop with respect to signed weights case is observed. The measured accuracy values are close to state-of-the-art floating point software implementation which confirms the overall functionality of the proposed MDL based time domain MAC computing approach. Moreover, MDL supports ultra-low voltage operation; working down till 375mV with more than 90% accuracy in 1x speed-up mode. 97% classification accuracy is observed at voltages down till 537mV in 16x speed-up mode (Fig. 3.4.b).

For LeNet-5 CNN, both convolution layer C1 and C3 layer measured throughput increases with higher speed-up mode and with increasing supply voltage achieving a peak throughput of 0.38(0.128) GOPS for C3(C1) layer at 585mV (Fig. 3.5). The maximum frequency is limited to 25MHz due to the test equipment limitation. Thus, no increase in throughput is observed above 550mV. The measured energy efficiency peaks with supply voltage scaling and

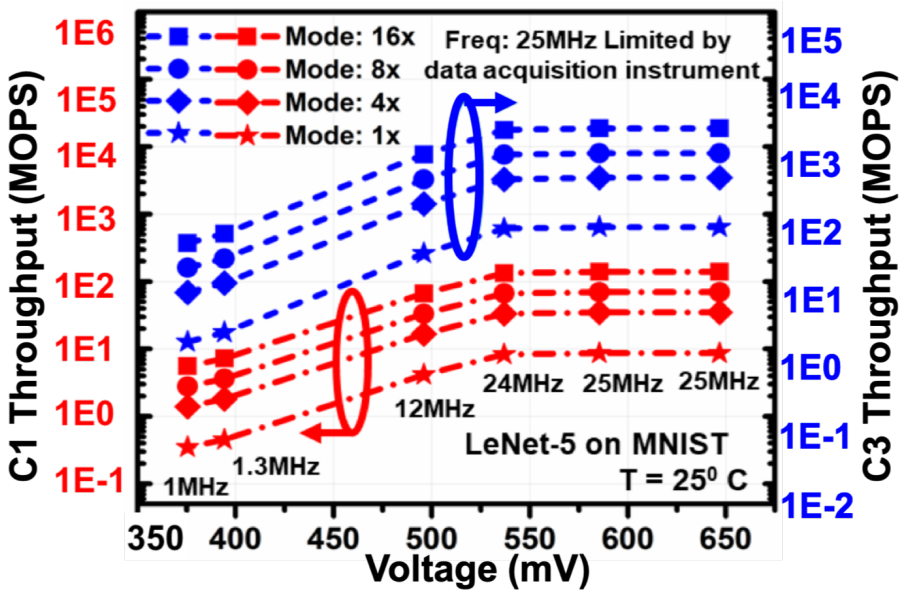


Figure 3.5: Measured Throughput for Convolution layers C1 and C3 of LeNet-5 CNN for different voltages and speed-up modes [14].

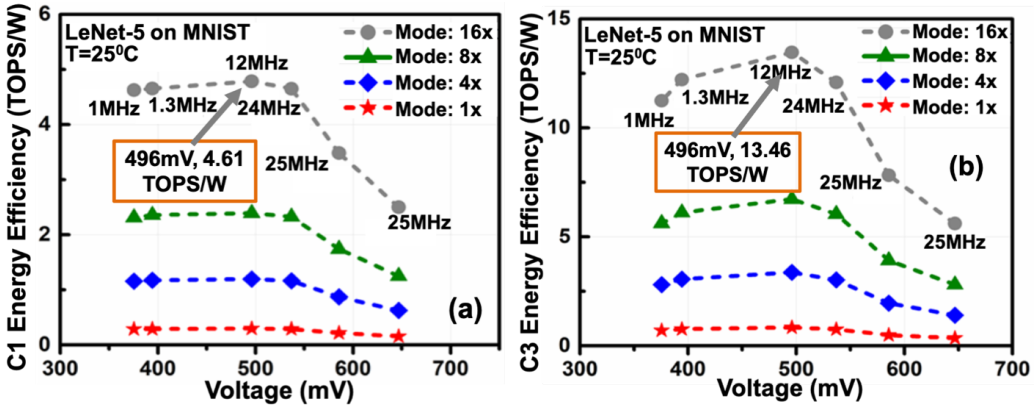


Figure 3.6: Measured Energy Efficiency for Convolution layers C1 and C3 of LeNet-5 CNN for different voltages and speed-up modes [14].

reaches maximum of 13.46(4.61) TOPS/W for C3(C1) layer at 496mV (Fig. 3.6). With increase in voltage, energy efficiency first increases till 550mV and

Table 3.2: Performance summary of proposed time-domain CNN engine implementing convolution layers C1 and C3 of LeNet-5 [14].

LeNet-5 Results/Metrics	Convolution Layer – C1				Convolution Layer – C3			
Speedup Mode	1x	4x	8x	16x	1x	4x	8x	16x
Input clock frequency (MHz)	24.0	24.0	24.0	24.0	24.0	24.0	24.0	24.0
MAC clock frequency (MHz)	0.19	0.75	1.50	3.00	0.19	0.75	1.50	3.00
Convolution Cycle Time (µs)	149.3	37.33	18.67	9.33	842.67	210.67	105.33	52.67
Operating Voltage (V)	0.537	0.537	0.537	0.537	0.537	0.537	0.537	0.537
Power (µW)	28.67	28.67	28.67	28.67	30.17	30.17	30.17	30.17
Throughput (GOPS)	0.008	0.032	0.064	0.128	0.023	0.091	0.183	0.365
Energy Efficiency (TOPS/W)	0.29	1.16	2.33	4.65	0.76	3.02	6.04	12.08

then decreases. This trend is observed since test equipment supports maximum input clock frequency of 25MHz, and thus power increases while throughput remains constant above 550mV.

Table 3.2 summarizes the test-chip performance. For a supply voltage of 537mV and input clock frequency of 24MHz, energy efficiency of 12.08 TOPS/W and throughput of 0.365 GOPS for convolution layer C3 is observed while achieving classification accuracy of 97% in 16x speed-up mode.

3.4 Comparison with prior approaches

Table 3.3 compares the proposed time-domain approach with earlier proposed analog [6], [8], digital [9]-[10] and time-domain [11], [13] approaches. The comparison is made for different metrics such as technology node, input/weight bit precision, chip size, low Vcc operation support, throughput, power and energy efficiency. The proposed design is compact; occupies $0.124mm^2$,

Table 3.3: Comparison of proposed time-domain CNN engine with prior Energy Efficient ML Accelerators [14].

Reference	Tech. (nm)	Circuit Type	Input/ Weight Size	Chip Size (mm^2)	Pooling	Low Vcc Op.	Cap. or ADCs	Accuracy	Throughput (GOPS)	Power (μW)	Energy Efficiency (TOPS/W)
ISSCC'18 [6]	65	Analog	6/1bits	0.067	No	No	Yes	96.0%	10.70	380.7	28.10
ISSCC'18 [8]	65	Analog	8bits	1.44	No	No	Yes	96.0%	-	-	3.125
ISSCC'16 [9]	65	Digital	16bits	16.00	Yes	No	No	98.3%	64	4.51E+4	1.42
VLSI'16 [10]	40	Digital	6/4bits	2.40	No	Yes	No	98.0%	102	3.90E+4	2.60
CICC'17 [11]	65	Time	8/3bits	0.24	No	Yes	Yes	91.0%	0.396	2.05E+4	0.019
ISSCC'18 [13]	55	Time	6/6bits	3.125	No	Yes	No	-	2.152	690	3.12
This work [14]	40	Time	8bits/ 1bit*	0.124	Yes	Yes	No	97.0%	0.365	30.17	12.08

supports near-threshold voltage operation, and consumes 30μW of power.

Chapter 4

Scaling up Time Domain CNN approach to larger networks (Phase/Test-chip II)

In this chapter, the motivation for phase II work addressing key limitations of phase I work are listed. Then, the design and architecture overview of the proposed phase II time-domain CNN engine is presented. Finally, the design features for all the major modules are discussed.

4.1 Motivation for Phase II Time-Domain CNN Engine

4.1.1 Supporting Multi-bit weight values

Deep neural networks like AlexNet, VGG require multi-bit inputs and weights for state-of-art classification accuracy. In the phase I of the design, only binary weights have been supported. In phase II of this work, the MDL, counter and shifter design architectures have been investigated to perform dot product of multi-bit weights and multi-bit pixel inputs at minimal MDL residue, minimal area and power overheads, and maximizing throughput and energy efficiency. This proposed concept to support multi-bit input pixel and multi-bit weights will be implemented in our second 65nm test-chip.

4.1.2 Improving throughput by asynchronous MDL operation

In the phase I MDL design, the dot product of each time-encoded input signal with corresponding weight bit is applied for 1 full *MAC_CLK* period (synchronous operation); thereby slowing down the accumulation process especially when the input and/or weight is '0'. To improve the throughput, MDL operation can be performed asynchronously. This means that each dot product is not applied for fixed *MAC_CLK* time period, but for different time duration depending on the width of time-encoded input pixel value. Also, if larger bit-width inputs are time encoded similar to the previous approach, each *MAC_CLK* needs to be $2^{(n-1)}$ unit clock periods which makes time domain approach very slow for wider bit widths. Since most of the inputs, especially in intermediate stages, are very low values, encoding them using proposed dead time removal approach, makes the time domain representation dependent on input magnitude rather than the input dynamic range. Fig. 4.1 shows the

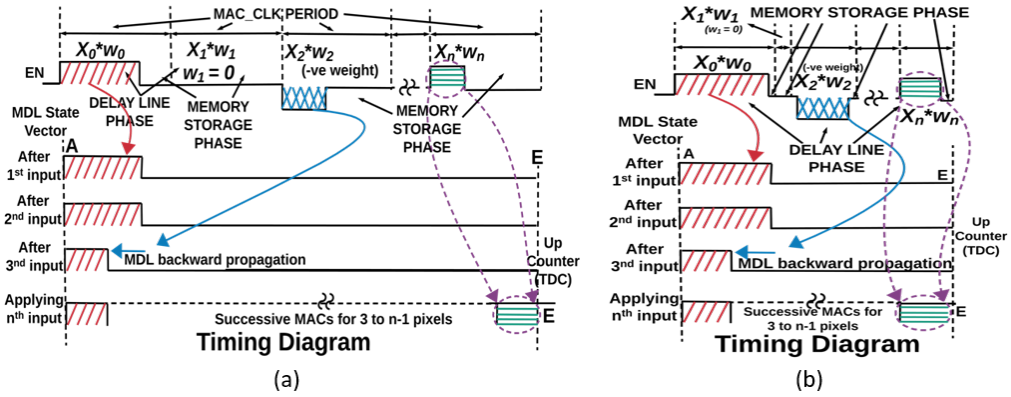


Figure 4.1: (a) Timing Diagram of (a) Phase I MDL Approach (b) Phase II MDL Approach Dead time Removal.

timing diagrams of phase I synchronous MDL operation and phase II asynchronous MDL operation. As shown in Fig. 4.1(a), all the inputs take 1 *MAC_CLK* period irrespective of input pulse-width. For X_1*w_1 case where w_1 equals 0, thus making zero pulse width of dot-product, or for X_n*w_n case where width of X_n is small, the phase I approach still consumes full *MAC_CLK* period. This wastage in time due to longer memory delay phase for such cases can be reduced by asynchronous operation as shown in Fig. 4.1(b). In this case, the width of memory delay phase is kept constant. Here, duration of time-encoded dot-product signal applied on MDL is proportional to its width (when signal is high); thereby speeding up process significantly.

4.2 Architecture Overview

Fig. 4.1 shows the high-level idea of the proposed architecture for 8-bit inputs and 8-bit weight neural network. Major components of this architecture along with the data flow path is outlined below.

First, pulse-width modulated (PWM) signals which are generated using a pulse generator, are selected based on the 4-bit input image pixel value. Here, 8-bit input pixel value is split into 2 4-bit values (4 MSB and 4 LSB bits), and dot-products of each MSB and LSB bits are computed separately. The pulse selector acts as Digital-to-Time (DTC) converter; converting the 4-bit pixel value into time encoded signal. Higher the pixel value; larger is the pulse-width of time encoded input. To improve the throughput, free-running PWM signal generation from the pulse generator is terminated as soon as time encoding of

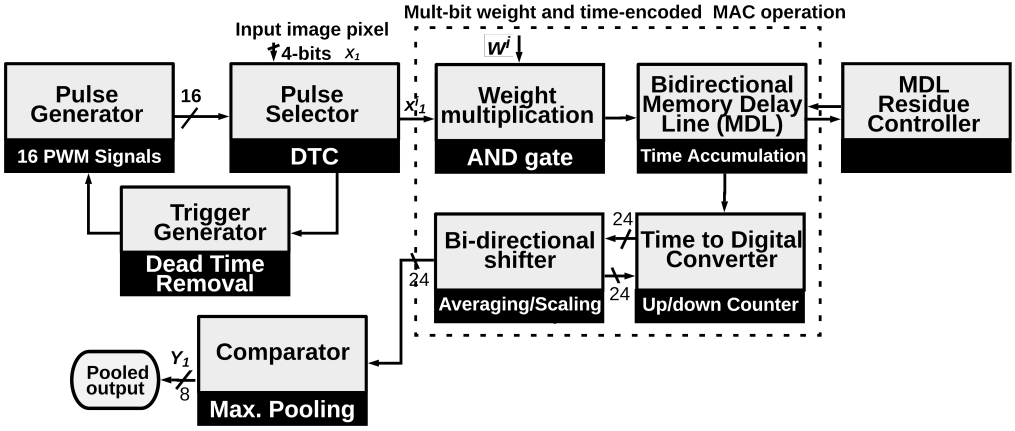


Figure 4.2: Architecture Overview of Phase II Time Domain CNN Engine.

the 4-bit input pixel value has been completed. For example, if the input pixel value is 8, then the pulse generation is stopped after 4 clock cycles (since pulse width of input pixel value is 8^*t_o, where 2^*t_o is input clock period). Once the time-encoding for a given input value is performed, next input value (stored in input shift register) is taken out to perform its time-encoding and dot-product computation. This asynchronous operation improves the overall throughput by eliminating all the dead-time (time when time-encoded input pixel value is held low in MAC_CLK period) in phase I of this work.

This PWM signal is then multiplied with the multi-bit weights sequentially. Each weight bit is multiplied by this time-encoded input pixel value using an AND gate; performing the dot product. The time-encoded dot product signal is then added using a bi-directional memory delay line (MDL) to perform the signed addition of the dot products for each MAC operation. An up-down counter is followed by the MDL to convert time-encoded MAC signal

into digital value. It acts as a Time-to-Digital converter for performing post-processing in the digital domain. Using an up-down counter, a finite length MDL can be used to perform long duration time domain accumulation. Once dot-products of all the MSB bit weight values with time-encoded input pixel values is performed, same time-encoded input pixel values are multiplied by next to most significant weight bit. To ensure correct MAC operation, counter MAC values for most significant weight bit are multiplied by 2 (left shift by 1 in shifter), such that time-accumulation and counter increments for next to most significant weight bit results in correct overall MAC value. This process is repeated till dot-products with all the 8 bits of weight are accumulated. The final MAC value from counter is right/left shifted using a bi-directional barrel shifter to perform averaging after MAC operation (MAV). This is also done to correctly scale the MAC output before feeding as the input to the next convolution layer. Once we obtain the shifter output value from 4 MDL units for each filter, the max. pooling operation (2x2 window) is performed using 24-bit comparators. The pooled output from each filter is stored off-chip and reused as the input to the next convolution layer of AlexNet CNN.

4.3 Design of Major Modules

In this section, the specific contributions of this thesis work have been described.

4.3.1 Asynchronous Pulse Generation and Selection

As mentioned earlier, 8-bit input pixel value is taken as 2 separate 4-bit values. The dot-product of 8-bit input pixel value with 8-bit weight value is computed in 2 phases - one with 4 MSB bits of input pixel value and another with 4 LSB bits. Thus, only 4 bits of the input data is represented as a PWM at a time, which consumes 8 clock cycles (for a n-bit input, 2^{n-1} clock cycles are required). 16 PWM signals (Fig. 4.3) are generated to represent 16 digital values (0 to 15), where pulse width of each PWM signal increments by t_o, where 2^*t_o is the clock period. Thus, 8-bit input pixel value can be time-encoded in 16 clock cycles (2 phases, each consuming 8 clock cycles) unlike phase I pulse generation approach which consumes 128 clock cycles (for 8-bit input pixel value, 2^{8-1} clock cycles are required); thereby resulting in 8x throughput improvement.

Moreover, throughput can be further improved by performing asynchronous MDL operation (Fig. 4.1). By asynchronous operation, it is meant that time-encoding operation for each input pixel value is not performed for fixed time duration - *MAC_CLK* clock period, but for different time duration which is proportional to the 4-bit input pixel value. Fig. 4.1 shows the timing diagram for the phase I and phase II approaches. As shown in Fig. 4.1.a, each input is applied for *MAC_CLK* clock period irrespective of input pulse-width. For $X_1{}^*w_1$ case where w_1 equals 0, thus making zero pulse width of dot-product, or for $X_n{}^*w_n$ case where width of X_n is small, the phase I approach still consumes full *MAC_CLK* period. This wastage in time due to

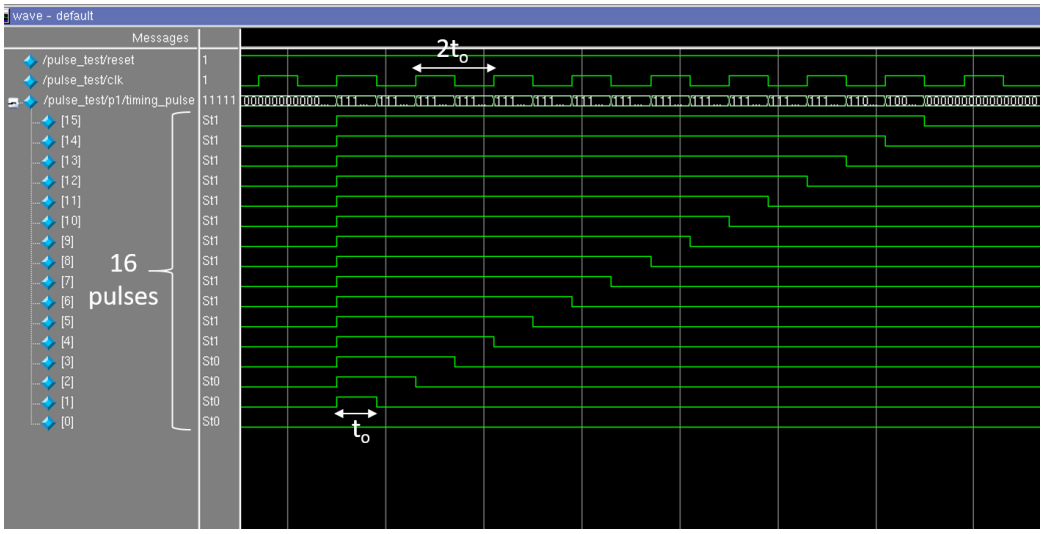


Figure 4.3: PWM signals generated by the Pulse Generator.

longer memory delay phase for such cases can be reduced by asynchronous operation as shown in Fig. 4.1.b. In this case, the width of memory delay phase is kept constant. Here, duration of time-encoded dot-product signal applied on MDL is proportional to its width (when signal is high); thereby speeding up process significantly.

A trigger generator module is designed which notifies PWM signal generation module that previous input value is time-encoded, and start generating the PWM signals again to perform time-encoding of next input pixel value (Fig. 4.4). 16 PWM timing pulses are generated in pulse generation module, one of these is selected in the pulse selector module depending of 4-bit input pixel value. In the proposed idea, 4 input values (Fig. 4.4.a) are passed simultaneously (to support 2x2 pooling operation as discussed in section 2). These 4 selected pulses (Fig. 4.4.b) are then ORed together(Fig. 4.4.c) to determine

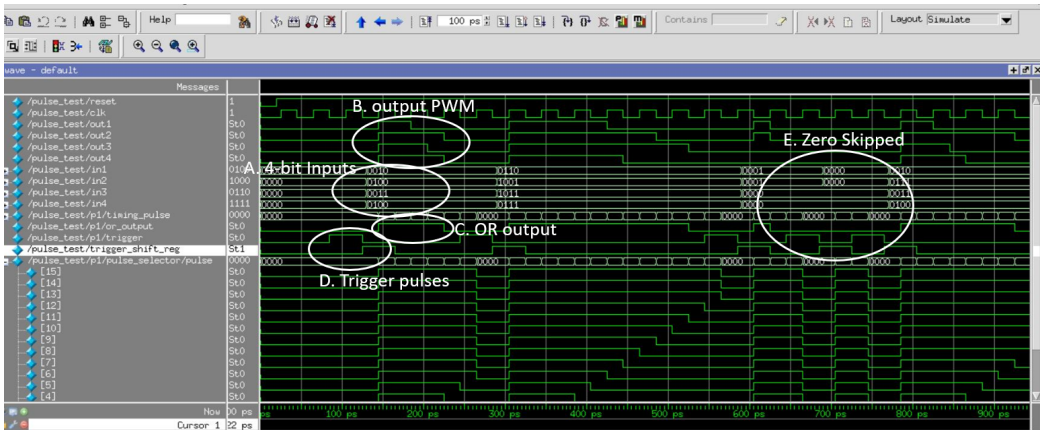


Figure 4.4: Simulation of Pulse and Trigger Generation modules illustrating dead time removal and zero-skipping in time-encoding of input pixel value.

the pulse with maximum pulse-width. Once the negative edge of ORed pulse signal is detected, the trigger generator sends a trigger signal (Fig. 4.4.d) to the pulse generator and the input/weight shift registers to start new PWM generation, and send the next input/weight bit values respectively as shown by simulation in Fig. 4.4. Thus, eliminating the time duration when the time-encoded input-pixel signal is held low helps in achieving higher throughput and energy efficiency. When all the 4 input-pixel values are zero (Fig. 4.4.e), the dead time removal logic works in a similar way, which we term as "Zero skipping". In this scenario, next input pixel value is processed after waiting for a single clock cycle.

4.3.2 Bidirectional MDL with Multi-bit Weight Support and Residue Loss Control

In phase II of this work, time-domain MAC computation is performed for multi-bit weights and multi-bit input pixel values. To support the multi-bit operation, bi-directional barrel shifter is used along with the MDL and counter. The signed time accumulation of dot products of multi-bit input and weight values is discussed as follows. First, the time-encoded dot product of multi-bit input-pixel value and MSB bit of weight is applied on on a bi-directional memory delay line (MDL) to perform the signed addition for each MAC operation. An up-down counter follows the MDL to convert time-encoded MAC signal into a digital value. Once all the dot-products of MSB bit weight values with respective time-encoded input pixel values are performed, the same time-encoded input pixel values are multiplied by the next to most significant weight bit. Since, the significance of this weight bit is lesser (2 times smaller), the counter value computed with most-significant weight value is multiplied by 2. This multiplication operation is performed by shifting left by 1 bit in the barrel shifter, and initializing the counter value with this shifted value. Thus, the scale of counter value is now changed to next to most significant weight bit value. Then, dot-products for next to most significant weight bit are accumulated and counter value is incremented/decremented. The shifting operation ensures the correct MAC value is computed for each weight bit value. This process is repeated till dot-products with all the 8 bits of weight are accumulated. Fig. 4.5.a shows the shifter operation. The 24-bit signed

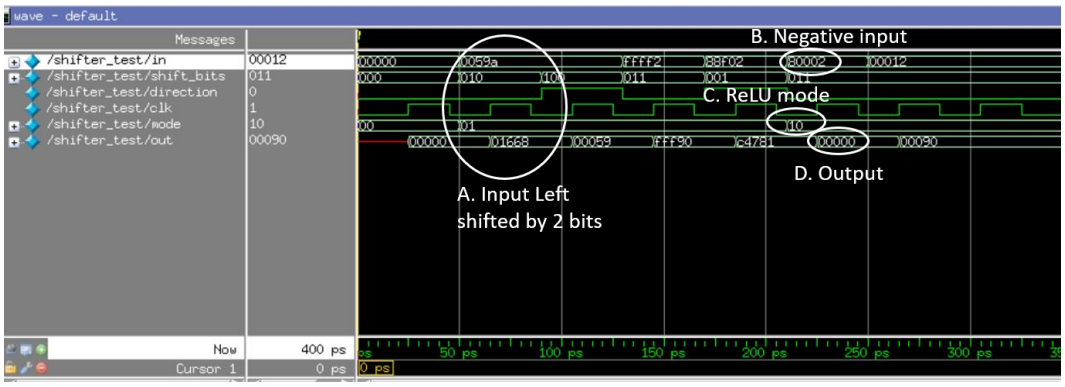


Figure 4.5: Functional Simulation of Shifter and ReLU modules.

input 0x0059a is being left shifted by 2 bits to obtain 0x1668.

One of the issues with the above mentioned approach of sequentially computing MAC operation with each weight bit of multi-bit value is high residue loss. The shifter operation ensures that counter value is properly scaled to accumulate subsequent dot-products with lower bit position of weight. However, the state-vector of MDL remains unchanged, and thus the residue time value is not scaled 2 times. This results in residue loss during each shifter operation; this can increase the classification accuracy loss. To minimize the residue loss, MDL unit with set/reset logic is proposed (Fig. 4.6). The residue controller is designed which provides the SET and RST signal values to change the state of MDL vector. This operation can be seen as multiplying the residue operation by appropriate scaling factor. To simplify the residue controller design, state of MDL vector is observed at start, mid and end positions of MDL and SET/RST values are generated which reduces the residue error loss by 75%. Fig. 4.7 shows the functional simulation of this controller where the

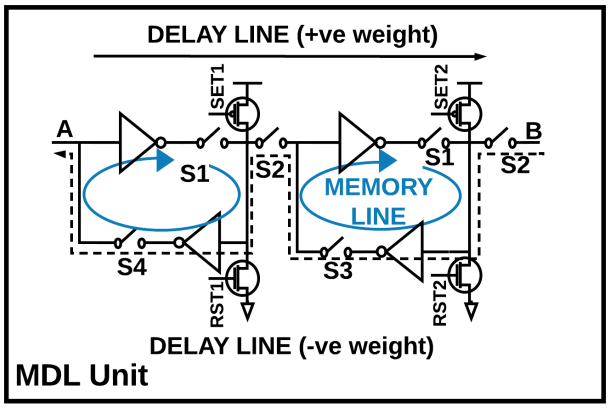


Figure 4.6: Proposed MDL Unit with Set/Reset Logic.

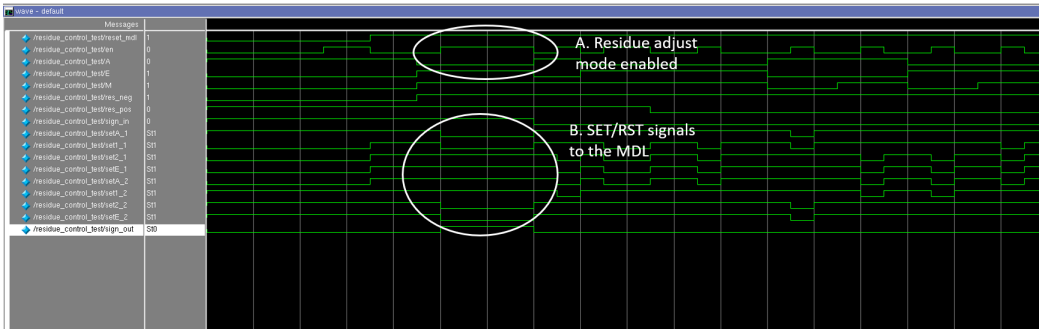


Figure 4.7: Functional Simulation of Residue Control Logic.

SET/RESET signals are fed appropriately to each MDL unit to scale residue time on MDL.

4.3.3 Address Controllers for On-chip and Off-chip Memory Accesses

For deep neural networks such as AlexNet and VGG, millions of MAC operations are computed. Approximately, 90% of the energy is spent in accessing the input pixel values and weights in and out of chip [5]. Hence, it becomes

essential to optimize this data movement to minimize the energy spent in data communication. In this regard, memory is divided into multiple hierarchies, i.e. DRAM $\leftrightarrow$ SRAM $\leftrightarrow$ Shift Register (Fig. 4.8). The filter (processing element) accesses weight values from weight shift registers, and pulse generation/selection module accesses the input pixel value from input shift registers. The values in these shift registers are loaded from on-chip global SRAM buffer. The pooled output values from each filter are also stored in this global SRAM buffer. Since, the on-chip SRAM memory is limited, portion of input activation and weight values and their pooled output values are stored on SRAM at a given time. Once the MAC computations for stored inputs and weights are computed, pooled output values are stored back in off-chip DRAM. Then, the next batch of inputs and weights are accessed from the off-chip DRAM memory. This process is repeated till all the MAC computations are performed. In this work, DRAM memory of the Xilinx Virtex VC707 FPGA platform is used.

The on-chip SRAM memory banks are generated using commercial memory compiler. Fig. 4.9 shows the functional simulation of SRAM bank demonstrating correct operation of SRAM in read, write and retention modes. The data accesses between different memory hierarchies (DRAM $\leftrightarrow$ SRAM $\leftrightarrow$ Shift Register) are controlled by different address generators. These memory address controllers play an important role to maintain the sequence of computations, read from and write to the correct banks of the SRAM and DRAM. Fig. 4.10 shows the functional simulation of the weight memory address con-

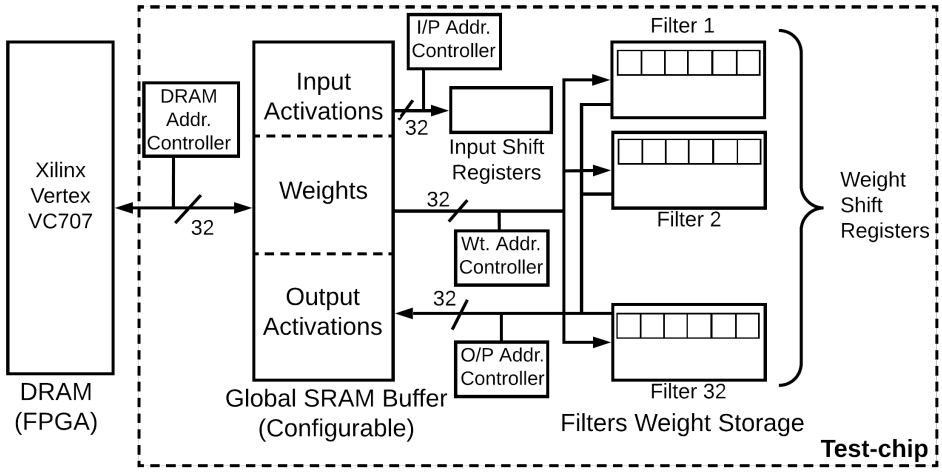


Figure 4.8: Memory Hierarchy and Architecture Overview.

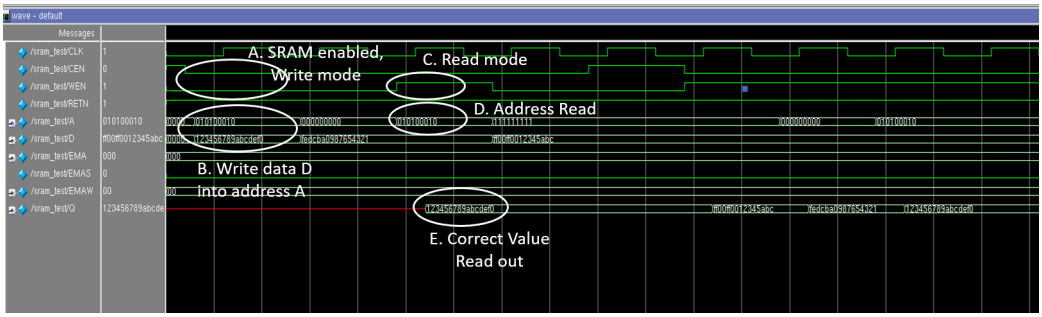


Figure 4.9: Functional Simulation of SRAM Bank generated by commercial memory compiler.

troller, which is generating the addresses to store the weights from on-chip SRAM to weight shift registers. Fig 4.10.a shows the convolutional layer and bit-position of the weight value which is required to be read from SRAM. Fig 4.10.c-d describes address generation process to fetch consecutive 32-bit weight values sequentially. This controller also generates the filter enable signals (Fig.

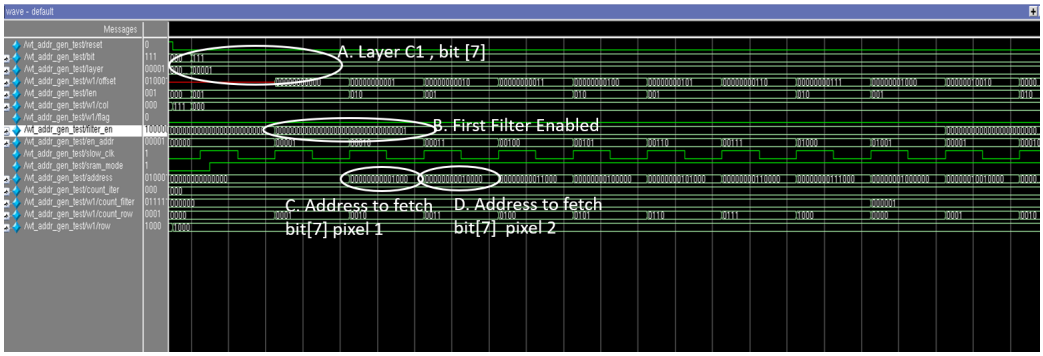


Figure 4.10: Functional Simulation of the Weight Memory Address Controller (SRAM → Weight Shift Register).

4.10.b), which allows storing of the weight values in only those filter modules for which filter enable signal is held high.

4.3.4 ReLU and Pooling Operations using Comparator

The Rectified Linear Unit (ReLU) is the most commonly used activation layer which returns 0 if it receives any negative input, and returns back the value, if it is positive. The ReLU module is designed which receives input from barrel shifter. If the SIGN bit is '1' (negative), the output values of ReLU module are reset (all 0's). Otherwise, the output value is assigned same as the input value. Fig. 4.5.b-d show the ReLU operation. When the ReLU mode is switched on, the output goes to zero if the output shifter value is negative.

The sub-sampling operation using max-pooling across 2x2 window is then implemented to reduce the intermediate layer output memory footprint by 75%. This is achieved by 4 concurrent MDL operations and feeding the MDL shifter ReLU outputs to three 24-bit comparators. The pooled output from

each filter is stored off-chip and reused as the input to the next convolution layer.

Chapter 5

Conclusion and Future Work

This chapter concludes the research work done in this project. The conclusion of this work is presented in section 5.1. The scope of future work which includes the next steps are discussed in section 5.2.

5.1 Conclusion

In this work, an energy efficient time-domain CNN engine suitable for edge computing is presented. The proposed time-domain CNN engine deploys a bi-directional memory delay line to perform signed accumulation of dot-products. The fully digital and technology scaling friendly design is compact and does not use any capacitors, and data converters (DACs and ADCs). It supports near-threshold voltage operation; useful for low power edge computing applications. Four speed-up modes are supported to address the throughput *vs.* accuracy trade-off in Phase I. The proposed design is tolerant to process variations and the delay mismatch among MDLs are offset by calibration unit. In phase I of this work, 40nm CMOS test-chip is taped-out implementing the LeNet-5 CNN. The energy efficiency of 12.08 TOPS/W, throughput of 0.365 GOPS and classification accuracy of 97% is achieved over

100 MNIST images at 537mV in 16x speed-up mode. In phase II of this work, concept of time-domain MAC computation is scaled up for bigger CNN network, AlexNet to classify ImageNet dataset. The proposed bi-directional MDL supports signed accumulation of multi-bit weights and time-encoded multi-bit pixel value. Furthermore, throughput is improved by skipping zero-value input activation values, and supporting asynchronous dot-product computation by eliminating dead-time in the memory storage phase of MDL.

5.2 Scope of Future Work

The phase II of this work (supporting multi-bit weights and higher throughput in comparison to phase I) describes the scaled-up time domain CNN engine design in 65nm CMOS technology. The functionality of MDL, residue controller, pulse generation and selection logic supporting zero-skipping and dead-time removal needs to be verified experimentally once this second 65nm test-chip is taped-out. Finally, the throughput, energy efficiency and classification accuracy results need to be measured for AlexNet network over ImageNet validation dataset images.

Bibliography

[1] G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *in IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 8297, Nov. 2012.

[2] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," *in Proc. of IEEE Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014, pp. 17011708.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *in Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 10971105.

[4] Karen Simonyan, Andrew Zisserman, "Very Deep Convolution Networks for Large-Scale Image Recognition," arXiv:1409.1556v6.

[5] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *in IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127138, Jan. 2017.

[6] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based ma-

chine learning applications," *in Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 488-490, San Francisco, CA, 2018.

[7] S. K. Gonugondla, M. Kang and N. Shanbhag, "A 42pJ/decision 3.12TOPS/W robust in-memory machine learning classifier with on-chip training," *in Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, 2018, pp. 490-492.

[8] J. Zhang, Z. Wang and N. Verma, "In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array," *in IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915-924, April 2017.

[9] J. Sim, et al., "A 1.42 TOPS/W Deep Convolutional Neural Network Recognition Processor for Intelligent IoE Systems," *in Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 264-265, 2016.

[10] B. Moons, et al., "A 0.32.6 TOPS/W Precision-Scalable Processor for Real- Time Large-Scale ConvNets," *in Proc. of IEEE Symposium of VLSI Circuits*, 2016.

[11] M. Liu, L. R. Everson and C. H. Kim, "A scalable time-based integrate-and-fire neuromorphic core with brain-inspired leak and local lateral inhibition capabilities," *in Proc. of IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1-4, Austin, TX, 2017.

[12] D. Miyashita, S. Kousai, T. Suzuki and J. Deguchi, "Time-domain neural network: A 48.5 TSOp/s/W neuromorphic chip optimized for deep learn-

ing and CMOS technology," *in Proc. of IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pp. 25-28, Toyama, 2016.

[13] A. Amravati, S. B. Nasir, S. Thangadurai, I. Yoon and A. Raychowdhury, "A 55nm time-domain mixed-signal neuromorphic accelerator with stochastic synapses and embedded reinforcement learning for autonomous micro-robots," *in Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 124-126, San Francisco, CA, 2018.

[14] A. Sayal, S. Fathima, S. S. T. Nibhanupudi and J. P. Kulkarni, "All-Digital Time-Domain CNN Engine Using Bidirectional Memory Delay Lines for Energy-Efficient Edge Computing," *in Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pp. pp. 228-230, San Francisco, CA, USA, 2019.

[15] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *in IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 903914, Apr. 2017.

[16] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *in Proc. of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[17] S. Pan and A. Kak, "A computational study of reconstruction algorithms for diffraction tomography: Interpolation versus filtered-backpropagation," *in IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 31, no. 5, pp. 1262-1275, October 1983.

[18] P. Koistinen and L. Holmstrom, "Kernel regression and backpropagation training with noise," *in Proc. of IEEE International Joint Conference on Neural Networks*, Singapore, pp. 367-372, vol.1, 1991.

[19] J. Zhou, W. Xu and R. Chellali, "Analysing the effects of pooling combinations on in variance to position and deformation in convolutional neural networks," *in Proc. of IEEE International Conference on Cyborg and Bionic Systems (CBS)*, pp. 226-230, Beijing, 2017.

[20] C. Lee, P. Gallagher and Z. Tu, "Generalizing Pooling Functions in CNNs: Mixed, Gated and Tree," *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 863-875, 1 April 2018.

[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," *in Proc. of European Conference on Computer Vision*, pp. 525-542, March. 2016.

[22] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," *in Proc. of Advances in Neural Information Processing Systems (NIPS)*, pp. 31233131, 2015.

[23] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *n Proc. of Advances in Neural Information Processing Systems (NIPS)*, pp. 41074115, 2016.

[24] K. Kim, W. Yu and S. Cho, "A 9 bit, 1.12 ps Resolution 2.5 b/Stage Pipelined Time-to-Digital Converter in 65 nm CMOS Using Time-Register," *in IEEE Journal of Solid-State Circuits*, vol. 49, no. 4, pp. 1007-1016, April 2014.

[25] Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," arXiv:1603.04467.

[26] National Instruments LabVIEW and PXIe. Available: http://www.ni.com /en-us/shop/labview.html.