# COMPAC: Compressed Time-Domain, Pooling-Aware Convolution CNN Engine With Reduced Data Movement for Energy-Efficient AI Computing

Aseem Sayal, *Member, IEEE*, Shirin Fathima, SS Teja Nibhanupudi, *Student Member, IEEE*, and Jaydeep P. Kulkarni, *Senior Member, IEEE*

***Abstract*— In this work, we demonstrate a compressed time-domain, pooling-aware convolution (COMPAC) convolutional neural network (CNN) engine for energy-efficient edge AI computing by performing multi-bit input and multi-bit weight multiply-and-accumulate (MAC) operations in the time domain. The multi-bit inputs are compactly represented as a single pulsewidth encoded input. This translates into reduced switching capacitance ($C_{DYN}$), compared with the baseline digital implementation, and can enable low-power neural network computing in an edge device. COMPAC CNN engine employs a novel and an improved version of the memory delay line (MDL) supporting the time residue scaling to perform the signed accumulation of multi-bit input and multi-bit weight products in the time domain. The compressed time-domain (CTD) approach is proposed to improve the throughput in time encoding of the input activations. The simulation results of the proposed CTD approach on the AlexNet CNN over 1000 ImageNet images show that 14.71 and 7.15 input clock cycles are consumed to time-encode an 8-bit input activation in two different CTD modes, improving the throughput by 88.60% and 94.46%, respectively, compared with the conventional pulsewidth modulation-based time-domain encoding. Furthermore, a pooling-aware convolution (PAC) technique is proposed to reduce the number of redundant MAC computations for the convolution layers that are followed by the max-pooling layer. The simulation results on the AlexNet CNN over 1000 ImageNet images show up to 31.47% (21.79%) reduction in the number of non-zero input activations MACs with a top-five classification accuracy loss of 0.60% (0.90%) with an on-chip access overhead of 60.53% (8.03%) for the PAC modes 1 (2) respectively. Finally, energy-efficient data flow for optimal on-/off-chip memory accesses for the time-domain MAC computation is proposed. COMPAC data flow the results in 86.97% reduced on-chip accesses and 29.74% reduced off-chip accesses compared with the Eyeriss approach, at iso-bit precision. COMPAC CNN engine implemented in 65-nm CMOS test chip demonstrates an energy efficiency of 1.044 TOPS/W and the throughput of 0.1278 GOPS at 720 mV for the AlexNet. The top-five classification accuracy of 76.90% measured over 1000 ImageNet images and 77.15% by simulating over 50 000 ImageNet images is achieved. The simulation results comprehending MDL circuit non-idealities for the AlexNet over 50 000 ImageNet validation set images show a classification accuracy loss within 1% compared with the 8-bit fixed-point software implementation.**

***Index Terms*— AlexNet, compressed time-domain (CTD), convolutional neural network (CNN), energy-efficient edge computing, ImageNet, memory delay line (MDL), multiply-and-accumulate (MAC), pooling-aware convolution (PAC), time residue scaling (TRS), time-domain processing.**

## I. Introduction

MACHINE learning (ML) approaches are being explored for a wide variety of applications requiring complex computations [1]–[3]. 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 with their fully connected network (FCN) counterpart [4], [5]. The CNN inference process requires intensive compute and memory resources to compute an enormous number of multiply-and-accumulate (MAC) operations, which consumes a significant fraction of a CNN accelerator's total power budget [6], [7]. The computation in the cloud suffers from issues, such as high latency, limited bandwidth, security, and privacy when transferring data from the edge to the cloud. There arises a critical need to develop energy-efficient computing capabilities to perform MAC operations locally on an edge device. In this article, a compressed time-domain, pooling-aware convolution (COMPAC) CNN engine is proposed for energy-efficient MAC operation by leveraging the concept of a memory delay line (MDL) (presented in our previous work [8], [9]). The key contributions of this work are as follows.

1) Time residue scaling (TRS) technique is proposed in the MDL to perform a multi-bit input and multi-bit weight MAC operation in the time-domain accurately.
2) The compressed time-domain (CTD) approach is proposed, in which the total time taken to encode an input is proportional to its magnitude only rather than the full-scale magnitude. This is achieved by eliminating the dead time in the pulsewidth modulated (PWM) input

Manuscript received June 23, 2020; revised October 19, 2020; accepted November 24, 2020. This work was supported by the Cockrell School of Engineering and by the NASCENT Center's Nanoengineering Online Education Program at the University of Texas at Austin. This article was approved by Guest Editor Jonathan Chang. *(Corresponding author: Aseem Sayal.)*

The authors are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: aseem.sayal@utexas.edu; shirin15@utexas.edu; subrahmanya_teja@utexas.edu; jaydeep@austin.utexas.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2020.3041502.

Digital Object Identifier 10.1109/JSSC.2020.3041502

signal when it is low. The proposed CTD approach combines the throughput benefits obtained by skipping the zero-magnitude input activations and eliminating the dead time for the input activations, which have a magnitude less than the full-scale value ($2^n - 1$ for $n$-bit input). The simulation results on the AlexNet CNN over 1000 ImageNet images show a throughput improvement of 88.60% and 94.46% to time-encode an 8-bit input activation in two different CTD modes compared with the conventional PWM-based time-domain approach [8], [9].

3) Pooling-aware convolution (PAC) technique is proposed to reduce the number of redundant MACs for the convolution layers that are followed by a max-pooling layer. For example, in the $2 \times 2$ max-pooling operation, the maximum value out of the four adjacent convolution outputs is chosen and applied as an activation to the subsequent convolution layers. In other words, the remaining three non-maximum MAC values are discarded. In the proposed PAC approach, these redundant MACs are identified during the time-domain MAC operation and are not completely computed to save energy. A comparison logic is added to keep a track of all four MAC values while accumulating the products of inputs and weights with reduced precision. In the scenario where a given MAC value lags behind the maximum MAC value (out of four MAC values) by a certain pre-defined threshold, the given MAC value is not further computed; thereby resulting in a fewer number of MACs. The simulation results on the AlexNet CNN over 1000 ImageNet images show up to 31.47% (21.79%) reduction in the number of non-zero input activations MACs with a top-five classification accuracy loss of 0.60% (0.90%) in two different PAC modes.
4) An energy-efficient data flow (EEDF) for optimal on-/off-chip memory accesses is also proposed for time-domain MAC computation. A configurable static random access memory (SRAM) bank architecture to allocate dynamic storage space to the input activations and weights depending upon the convolution layer architecture is proposed. Furthermore, a convolution-stride-aware mapping of input activations and weights to the SRAM array is proposed to optimize for on-/off-chip memory accesses. In addition, input–output activation and weight-reuse techniques are proposed for on-chip access reduction, and the run-length coding (RLC) technique is used for off-chip memory access reduction. COMPAC data flow results in 86.97% reduced on-chip accesses and 29.74% reduced off-chip accesses compared with an Eyeriss [4], compared at an iso-8-bit precision.
5) The COMPAC CNN engine is demonstrated on the 65-nm CMOS node, implementing the AlexNet CNN, achieving energy efficiency of 1.044 TOPS/W, a throughput of 0.1278 GOPS, and a top-five classification accuracy of 76.90% at 720 mV.

This article is organized as follows. Section II describes the background of this work and discusses the basics of CNNs and motivations for the CTD multi-bit input/weight MAC computation, PAC technique, and EEDF. Section III presents the concept, architecture, design, and implementation details of the proposed COMPAC CNN engine. Section IV presents the 65-nm CMOS test-chip measurement results. Section V concludes this article with the key findings.

## II. Background

### A. AlexNet CNN

A typical CNN consists of convolution layers, pooling layers, activation layers, and FCN layers. The convolution layer serves as the core layer, which implements multiple trainable filters that are used to extract features, such as edges, gradients, and colors, in an image. The major operation in CNNs is the MAC operation, which is computed by performing the dot product of a weight matrix and a portion of the input image matrix [8]. Each filter is convolved across the width and height of the input activation, computing multiple MACs and producing a 2-D output activation map for a given filter. Typically, an activation layer is used between consecutive convolutional or fully-connected layers. This layer is used to add non-linearity to a neural network. The rectified linear unit (ReLU) is the most commonly used activation layer that returns 0 if it receives any negative input and returns the same value if it is positive [10]. Pooling (sub-sampling) is performed to reduce the data size feeding into the next layer [11], [12]. Max-pooling is commonly used for the sub-sampling operation that chooses the maximum MAC value across a small output window (e.g., $2 \times 2$) [12]. The FCN layers are usually used in the end, once the data size of input feature maps is reduced after passing through the pooling layers. A typical FCN layer comprises of a finite number of feature maps, each of size $1 \times 1$. Each of these feature maps is connected to all the feature maps of the previous layer. For example, the AlexNet CNN contains eight trainable layers; the first five are the convolution layers, and the remaining three are the fully connected layers [3]. The output of the last fully connected layer is fed to a 1000-way soft-max layer that produces a distribution over the 1000 class image labels. Max-pooling layers follow the first, second, and fifth convolution layers. The ReLU non-linearity is applied to the output of every convolution and FCN layer.

### B. Prior Work

As MAC operations constitute a significant portion of the total CNN power budget [6], various methods have been explored for compact data representation to improve energy efficiency. The data in digital domain are represented as multi-bit digital vector [4], [13]–[18]. This form of data representation is implemented using a higher number of nets, translating into a high dynamic switching capacitance ($C_{\mathrm{DYN}}$) and, consequently, higher power and area. In the analog-domain approach, data are represented as a continuously varying voltage signal. Various design techniques using charge manipulation schemes and analog-to-digital (A/D) converters have been proposed to realize efficient MAC computations in a CNN accelerator. Analog approaches [19]–[21] compute MAC operation in analog voltage domain using

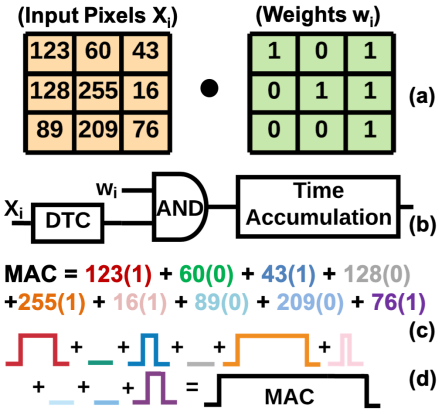



Fig. 1. Concept of the time-domain MAC operation [9]. (a) Dot product operation of input matrix and weight matrix. (b) Time-domain MAC circuit concept. (c) MAC operation in the digital domain. (d) MAC operation in the time domain.

a SRAM array, capacitors, and data converters (ADCs and DACs). However, finite voltage headroom and the sensitivity of circuit parameters to a slight change in analog-domain signals limit the voltage scalability, thereby degrading the MAC accuracy [20]. In the frequency-domain approach, data are represented as signals with varying frequencies using ring oscillators or *RC* loaded circuits. In [23] and [24], MAC operations are computed in the frequency domain using a digital controlled oscillator (DCO) with either resistor or capacitor loading. Various nodes of a ring oscillator are loaded with different capacitor banks to represent multi-bit weights altering the *RC* time constant of the oscillator. Thus, large-range frequency generators/modulators limit the performance scalability of such frequency-domain approaches. Moreover, larger magnitude inputs would result in higher toggle activity incurring higher switching power. In the time-domain computing approach, data can be represented as the PWM inputs or pulse position modulated inputs or a combination of thereof. In [24], the time-domain analog-digital mixed-signal processing approach is proposed to implement a time-domain binary neural network. A time-mode adder circuit is proposed in [25] to realize a trans-linear principle in time.

In our earlier work [8], [9], a multi-bit digital bit-stream is encoded as a single PWM signal with its pulsewidth representing the magnitude of the input data. The multi-bit data are compacted to a single data signal, resulting in reduced $C_{\mathrm{DYN}}$ compared with the conventional digital data representation. We presented a detailed comparison between the time-domain computing approach and the sequential digital implementation, performing 1-/8-bit weight and 8-bit input MAC operations, and observed 2.09×-2.32× higher energy efficiency and 2.22×–3.45× smaller area for the time-domain implementations [9]. Furthermore, this approach leverages standard digital gates with full rail-to-rail voltage swing outputs. This mitigates the reduced voltage headroom challenges encountered in an analog-voltage-domain approach enabling ultralow voltage operation. In addition, this approach does not require multiple clock sources (unlike frequency-domain approaches), which results in reduced $C_{\mathrm{DYN}}$. The concept of the time-domain MAC computation using PWM inputs is illustrated in Fig. 1. In this example, the MAC operation is performed by adding the products of a 3 × 3 weights matrix and input pixel matrix [see Fig. 1(a) and (c)]. An input pixel value is encoded into a PWM signal using a digital-to-time converter (DTC). Then, the input PWM signal is multiplied by the corresponding weight bit using an AND gate. This time-encoded product signal is then passed onto the time accumulator circuit [see Fig. 1(b)] to compute MAC operation in the time domain. Sequentially, all input pixel values are encoded as PWM signals, and its product with respective weight bit is loaded on the time accumulator circuit. Thus, all time-encoded products are applied onto the time accumulator circuit, and the width of these timing pulses gets accumulated to realize the MAC operation in the time domain [see Fig. 1(d)].

### *C. Motivation for CTD Multi-Bit Input and Weight MAC Operation*

Complex CNNs, such as AlexNet [3] and VGG [26], compute multi-bit input and multi-bit weight MAC operations to achieve the state-of-art classification accuracy on complex data sets, such as 1000-class ImageNet [3]. In our previous work [8], [9], MAC operation with binary weights is demonstrated. Earlier proposed techniques to perform multi-bit weight time-domain MAC operation using either multiple MDLs or a single configurable-length MDL incur high power/area along with classification accuracy loss; thereby making the time-domain MAC approach less feasible for complex CNNs [8], [9]. There arises a critical need to devise better MDL designs to perform energy-efficient multi-bit input/weight MAC operation in the time domain to make it feasible for complex CNNs. In this article, the TRS technique is proposed to perform the multi-bit input and multi-bit weight MAC operation with minimal MDL time residue, minimal area, and power overhead while maximizing energy efficiency.

Moreover, one of the tradeoffs of the time-domain computation is its sequential nature, resulting in low MAC throughput. In the conventional time-domain approach, as discussed in our previous work [8], [9], the PWM signal is generated synchronously consuming $2^{(n-1)}$ clock cycles to time-encode an $n$-bit input. PWM signal is held high for time duration proportional to its magnitude and held low for the remaining time duration (dead time), thereby resulting in a significant throughput degradation. In order to overcome this low throughput limitation, a CTD approach is proposed in this work for compact input encoding, in which the total time taken to time-encode an input is proportional to its magnitude (realized by eliminating the dead time using a negative-edge and zero-value detector circuit).

### *D. Motivation for PAC Operation*

The CNN inference process requires the computation of an enormous number of MAC operations, thereby making it challenging to implement in the energy-constrained edge devices. For example, widely known CNNs, such as AlexNet [3] and VGG [26], comprise of 60 million and 138 million filter weights, respectively, which are used to compute 724 million and ≈15.5 billion MAC operations [4]. Thus, reducing the

number of MACs without compromising the classification accuracy is highly desirable. In a typical CNN, a max-pooling layer is used for the sub-sampling operation that chooses a maximum MAC value across a small output window. For example, in a 2 × 2 max-pooling operation, the maximum value out of the four convolution outputs is chosen and fed to the subsequent convolution layers. In other words, the remaining three values are not used. In this article, a PAC approach is proposed, in which these redundant MACs are identified during the MAC operation and are not completely computed to save energy. A comparison logic is added to keep a track of all four MACs while accumulating the products of inputs and weights with reduced precision. A given MAC value is not further computed if it lags behind the maximum MAC value by a certain threshold, thereby reducing # MACs.

### *E. Motivation for EEDF*

A significant portion of the total accelerator energy is spent in accessing the parameters and activations from either on-chip or off-chip memory. Since deep CNNs comprise a large number of parameters, it is not feasible to store all the activations and weights on-chip. This results in significant energy consumption in off-chip memory accesses, which is one of the critical factors affecting energy efficiency. The energy spent to access the data from the off-chip memory is around 200× compared with the MAC computation energy [4]. Despite pruning the input data and filtering out zero weights and/or inputs, a large number of MAC operations are still performed; which leads to a large number of off-chip accesses. The other critical component of power consumption in these networks is the on-chip memory buffer (SRAM) accesses. The energy spent to access the data from an on-chip memory buffer is around 6× compared with the MAC computation energy [4]. Moreover, the number of on-chip data accesses is significantly more than the number of off-chip dynamic random access memory (DRAM) accesses, making the on-chip memory access an important concern for energy-efficient edge computation.

## III. Proposed COMPAC CNN Engine Design

### *A. Architecture Overview*

The proposed COMPAC CNN engine supports 32 filters, a CTD controller, multiple address generators, registers, and a 67-kB on-chip SRAM to perform a pooling-aware 8-bit input and 8-bit signed weight MAC operation for the AlexNet CNN in the time domain (see Fig. 2). The overall operation is briefly discussed as follows. First, the input activations and the weights of 32 filters are accessed from the off-chip DRAM memory (Xilinx VC707 FPGA resources) using a 32-bit input data bus and stored in the global on-chip buffer (SRAM) with a storage capacity of 67 kB. The input activations are then accessed from the global buffer and stored locally in 144 input activation cyclic shift registers (each 32-bit wide). The address generator and controller are used to access the input activations from the on-chip SRAM and efficiently broadcast them to the activation registers such that each activation is accessed only once from on-chip SRAM and stored in multiple registers,

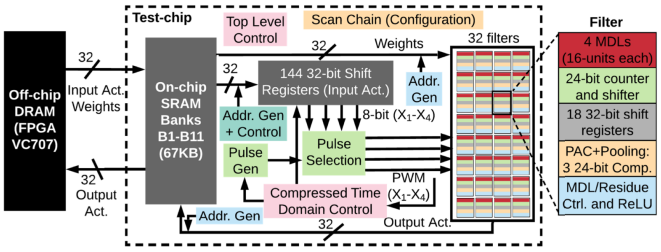



Fig. 2. Proposed COMPAC CNN engine system-level architecture.

thereby optimizing the total number of on-chip accesses. The pulse generator generates 16 free-running PWM signals, one of which gets selected in the pulse-selection module depending upon the input activation value. To improve the throughput, a CTD approach is proposed in this work, where the pulse generation occurs asynchronously, to eliminate the dead time using a CTD controller. The four time-encoded PWM inputs for $X_1$–$X_4$ are sent to all 32 filters to perform the MAC operations in the time domain. Each filter comprises of four 16-unit long MDLs, a 24-bit counter, a 24-bit shifter, nine 32-bit shift registers for weight bit storage, nine 32-bit cyclic shift registers to store the sign bit of the weight, three 24-bit comparators to perform comparison during PAC and max-pooling operations, an ReLU activation controller, and an MDL/residue controller for the reliable and correct MDL operation. The 8-bit input/weight MAC operation is computed sequentially by accumulating the products of time-encoded input with the most significant weight bit and repeating it for the next less-significant weight bits. The weight bit and the sign bit are optimally accessed from the on-chip global buffer using an address generator and stored locally in 18 32-bit registers of each filter. The pooled output value of each filter is stored in the global-buffer, and once the MAC computation for all the input activations stored on-chip for 32 filters is completed, the output activations are sent through a 32-bit output data bus and stored off-chip (FPGA VC707). This process is repeated until all the MAC operations for the entire input activation map and for all the filters are computed, for each convolution layer.

### *B. Multi-Bit Input and Weight MAC Using MDL With TRS*

*1) Background: Time-Domain MAC Computation Using MDL:* The concept of an MDL to perform 1-bit weight time-domain MAC operation with reduced switching capacitance for an energy-efficient edge computing was discussed in our previous work [8], [9]. MDL is comprised of a string of gated delay units (termed MDL units). The MDL design is derived from the concept of a time register used in high-precision time-to-digital converters, which can perform time addition and time storage using a string of delay cells [27]. Each MDL unit is controlled by an EN signal. The product of time-encoded input activation and weight bit acts as an EN signal. MDL acts as a forward (backward) delay line for positive (negative) weight when EN is held high and retains the MDL state when EN is held low. The 0→1 transition is detected at either end of the MDL (node A or node E) when a

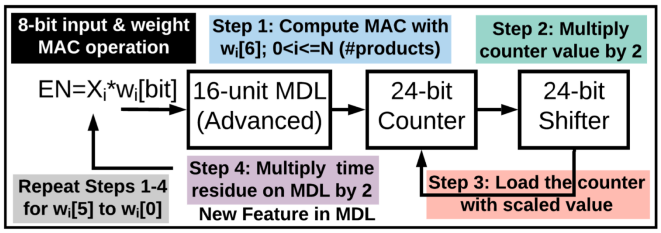



Fig. 3. 8-bit input/weight MAC operation in time-domain using a single 16-unit MDL (with TRS), counter, and shifter.

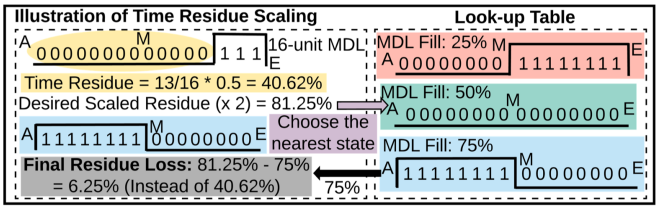



Fig. 4. Proposed concept of TRS on MDL for multi-bit weight/input MAC operation.

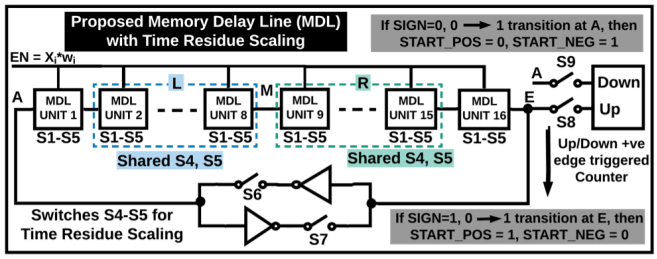
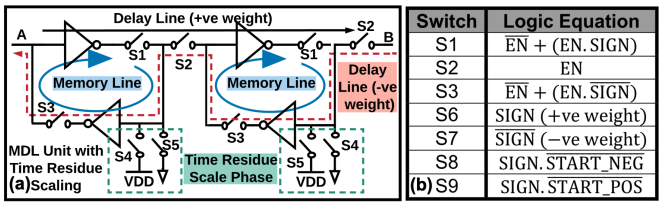



Fig. 5. Proposed MDL with TRS features for multi-bit weight/input MAC operation.




| Switch | Logic Equation |
|---|---|
| S1 | $\overline{EN} + (EN.SIGN)$ |
| S2 | EN |
| S3 | $\overline{EN} + (EN.\overline{SIGN})$ |
| S6 | SIGN (+ve weight) |
| S7 | $\overline{SIGN}$ (−ve weight) |
| S8 | $SIGN.\overline{START\_NEG}$ |
| (b) S9 | $SIGN.\overline{START\_POS}$ |

Fig. 6. (a) Proposed MDL unit with TRS features for multi-bit weight/input MAC operation. (b) Switch configurations for the proposed MDL supporting multi-bit weight/input MAC operation.

string of "0's" followed by a string of "1's" is propagated. The delay variations in MDL due to process variations (random errors) or temperature changes (systematic error) are mitigated by the addition of a calibration unit in MDL. Thus, a finite MDL length along with an up–down counter can be used to perform signed time accumulation of a long duration to perform MAC operation.

*2) Proposed Multi-Bit Input/Weight MAC Operation in Time Domain:* The multi-bit weight time-domain MAC operation is performed by sequentially accumulating the products of time-encoded input with each weight bit, using a single fixed-length MDL (see Fig. 3), thereby overcoming the limitations of multi-bit weight time-domain MAC, as discussed in our previous work [9]. The counter value and the time residue on MDL are scaled by a factor of 2 to match its bit precision, before accumulating the product of the input with the next less-significant weight bit (see Fig. 3). The counter value is scaled by a factor of 2 using a shifter (left shift by 1), whereas time residue on MDL is scaled by a factor of 2 using the TRS technique, as discussed in Section III-B3. This process is repeated until the products of time-encoded input with all the weight bits are accumulated (see Fig. 3).

*3) Concept of TRS:* The time residue on each MDL needs to be scaled by a factor of 2, before accumulating the products of input activation with the next less-significant weight bit on MDL (to match the precision). This scaling has been realized efficiently using a lookup table approach (see Fig. 4). The TRS lookup table comprises of three MDL states when MDL is 25% fill, 50% fill, and 75% fill. The state of the MDL is first determined by observing the start (node A), middle (node M), and end (node E) nodes of the MDL, and then, the scaled residue state is chosen from one of the TRS lookup table MDL states (see Fig. 4). For instance, the time residue on MDL is 40.62% of the MDL full-length delay, which needs to be scaled by a factor of 2. The desired scaled residue is $40.62\% \times 2 = 81.25\%$. The nearest residual value of this desired scaled value is chosen, which is 75% of the full-length MDL delay, thereby minimizing the time residue loss (see Fig. 4). The simulation results of the AlexNet CNN on 1000-class ImageNet data set have shown ≈3% improvement in the top-five classification accuracy with the proposed TRS technique compared with the baseline MDL approach (wo/TRS).

*4) Implementation of TRS in MDL:* The bi-directional MDL is designed to support the TRS for multi-bit weight MAC operation (see Fig. 5). The state of each MDL unit [see Fig. 6(a)] is controlled by observing the start (node A), middle (node M), and end (node E) nodes of the MDL such that the time residue loss is minimized to achieve high classification accuracy. MDL unit acts as a forward (backward) delay unit for positive (negative) weight when EN is held high and retains MDL state when EN is held low during normal MDL operation (controlled using switches S1—S3) and sets its logic node value to 0 or 1 to scale the residue on MDL during the TRS phase (controlled using switches S4 and S5). Fig. 6(b) lists down the switch configurations (S1–S3 and S6–S9) for the memory and delay phases of the MDL operation. The basic operation of the memory and delay phases is briefly discussed in Section III-B1. A comprehensive account of the working principle of MDL has been presented in our previous work [9].

In order to realize energy and area-efficient implementation of the TRS, only three nodes (nodes A, M, and E) are observed to determine the desired MDL state with scaled time residue. There are 128 TRS controllers (one per each MDL), each occupying a core area of 96 $um^2$. Thus, the TRS modules occupy 0.0123 $mm^2$, contributing to a 0.71% increase in the total core area. The power and latency overheads of the TRS are insignificant compared with the MAC accumulation phase, which comprises of many clock cycles [equals filter dimensions × channels × 7.15 cycles (to time-encode 8-bit input)]. This efficient lookup table-based TRS approach has

TABLE I
LOOKUP TABLE FOR THE PROPOSED TRS APPROACH

| MDL State Vector and Phase | | | | Orig. Residue Loss (%MDL full length delay) | Switches S4/S5 (Output MDL State Vector) and Counter Incr./Decr. | | | | | Final Residue Loss (% MDL) |
|---|---|---|---|---|---|---|---|---|---|---|
| A | M | E | Phase | | Unit 1 | Units 2-8 | Units 9-15 | Unit 16 | Counter | |
| 0 | 1 | 1 | +ve | 0-25% | 1/0 | 1/0 | 0/1 | 0/1 | No | <±25% |
| 0 | 0 | 1 | +ve | 25-50% | 0/1 | 0/1 | 1/0 | 1/0 | No | <±25% |
| 1 | 0 | 0 | +ve | 50-75% | 1/0 | 1/0 | 0/1 | 0/1 | Incr. | <±25% |
| 1 | 1 | 0 | +ve | 75-100% | 1/0 | 1/0 | 1/0 | 0/1 | Incr. | <50% |
| 1 | 1 | 0 | -ve | 0-25% | 0/1 | 0/1 | 1/0 | 1/0 | No | <±25% |
| 1 | 0 | 0 | -ve | 25-50% | 1/0 | 1/0 | 0/1 | 0/1 | No | <±25% |
| 0 | 0 | 1 | -ve | 50-75% | 0/1 | 0/1 | 1/0 | 1/0 | Decr. | <±25% |
| 0 | 1 | 1 | -ve | 75-100% | 0/1 | 1/0 | 1/0 | 1/0 | Decr. | <50% |

resulted in $\leq\pm 25\%$ of MDL full-scale delay, thereby results in high classification accuracy. The logic values on nodes A, M, and E are observed to determine the initial state (time residue) of the MDL. The phase of the MDL, i.e., whether it is holding a positive or negative residue, is also determined from the MDL controller. The switch S4 and S5 values for the MDL unit 1, units 2–8, and units 9–16 along with the counter increment/decrement signal are generated from the TRS controller to scale the time residue on MDL such that the final residue loss is minimized (see Table I). Besides ensuring minimal accuracy loss, it is also ensured that weights can be reused to reduce the on-chip data accesses. This requires a co-design of the TRS controller and weight address generator to scale the time-domain approach for multi-bit weight MAC operation. It should be noted that the residue loss can be further minimized by designing a relatively complex TRS controller with a higher area/power overhead, which observes more number of MDL internal nodes and outputs more sets of S4/S5 switch values to precisely control MDL state. This would lead to lower time residue loss and better accuracy.

### C. Proposed CTD Approach

*1) Concept of CTD Approach:* In the conventional time-domain approach as discussed in our previous work [8], [9], the PWM signal is generated synchronously consuming $2^{(n-1)}$ clock cycles to time-encode an $n$-bit input. PWM is held high for the time duration proportional to its magnitude and low for the remaining time duration (dead time), thereby resulting in a significant throughput degradation [see Fig. 7(a)]. A CTD approach is proposed for the compact input encoding in which the total time taken to encode an input is proportional to its magnitude only [see Fig. 7(c)]. This is achieved by eliminating the dead time (time duration when a PWM signal is held low). The proposed CTD approach combines the throughput benefits obtained by skipping the zero-magnitude input activations [see Fig. 7(b)] and eliminating the dead time for the input activations that have a magnitude less than the maximum value ($2^n - 1$ for an $n$-bit input).

*2) Implementation of CTD Approach:* The proposed CTD approach is implemented using a negative-edge and zero-value detector circuit (see Fig. 8). The proposed COMPAC CNN engine performs an on-chip $2 \times 2$ max-pooling by computing four MAC values simultaneously. Each MAC value is obtained by computing a dot product of input activation ($X_1$–$X_4$) with

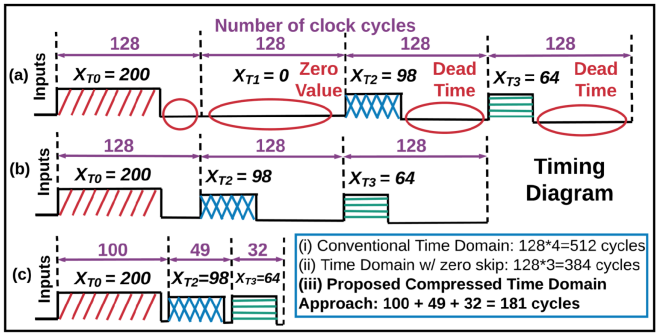



Fig. 7. (a) Conventional time domain approach. (b) Conventional time domain approach with zero-skipping. (c) Proposed concept of the CTD approach in the COMPAC CNN engine.

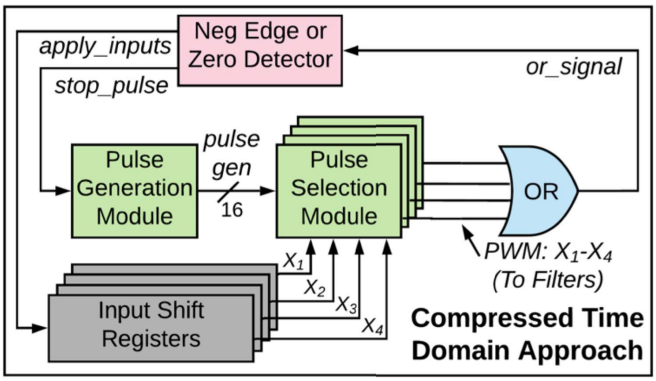



Fig. 8. Circuit implementation of the CTD approach in the COMPAC CNN engine.

the weight matrix. These input activations are stored in the input activation shift registers: 16 free-running PWM signals are generated in the pulse generation module such that the pulsewidth of each consecutive signal increases by *to*, where *to* equals the half period of the input clock. Thus, the pulsewidth of the PWM signal varies from $0 \times to$ to $15 \times to$.The pulse generation scheme of the proposed work generates different PWMs based on synchronous input clock operation. The pulse generation approach is not based on a logic path delay difference, which is susceptible to pulse shrinking and expansion due to process variations. Thus, no such shrinking and expansion effects are expected in pulse generation. The pulse-selection module comprises one 16:1 MUX. The 4-bit input activation value serves as the four select lines of this 16:1 MUX to select one of the 16 PWM signals, based on input activation magnitude. Four pulse-selection modules are used to time-encode four 4-bit input activation values ($X_1$–$X_4$). These time-encoded input activation signals are passed through a four-input OR gate to find the time-encoded input with a maximum pulsewidth (*or_signal*). This signal is passed to the negative edge and the zero value detector to determine the negative edge or zero-magnitude value. Once any of these two events occur, a trigger signal (*stop_pulse*) is generated on the next edge of the input clock (adding a delay of half input clock period), and the pulse generator is stopped on the next subsequent clock edge. Then, the next input activations are applied to the next subsequent clock edge by shifting

the input activations from the shift registers, by asserting an *apply_inputs* signal. Finally, the pulse generator starts on the next subsequent clock edge.

Thus, the proposed CTD approach time encodes the input activations in an asynchronous manner and incurs a time overhead of two input clock cycles (half-cycle each for trigger generation, stopping the pulse generator, applying the next inputs, and starting the pulse generator) to ensure a reliable time encoding (without any glitches) of input activations. Thus, the time taken to encode input activation equals ($\max(X_1$ to $X_4) \times$to$) + 2 \times$ input clock cycle period (time overhead) in the proposed CTD approach, whereas it takes $(2^{(n-1)} \times to) + 1 \times$ input clock cycle period (time overhead to ensure no race around condition and/or glitch formation) in the baseline MDL approach [8], [9].

The conventional time-domain approach, as discussed in our previous work [8], [9], employs a pulse generator module (generating 16 pulses in MSB phase and 16 pulses in LSB phase) and a pulse selector module (four 2:1 Muxes and a 16:1 MUX), whereas the proposed CTD approach employs a smaller pulse generator module (16 pulses only), a smaller pulse selector module (16:1 MUX only), and a CTD controller. The CTD circuit occupies a core area of 920 um$^2$, whereas the conventional time-domain pulse approach occupies 642 um$^2$. Thus, CTD results in a core area overhead of 278 um$^2$, which is 0.016% of the total core area. Furthermore, CTD modules consume a total power of 18.72 $\mu$W, whereas the conventional time-domain approach consumes 14.78 $\mu$W, thereby resulting in a power overhead of 3.94 $\mu$W and amounting to $\approx$3% of the total core power.

*3) Simulation Results of CTD on AlexNet CNN:* The proposed concept of the CTD approach is validated for all the five convolution layers of the AlexNet CNN on 1000 images of the ImageNet validation data set (1000 class). One image is randomly chosen from each class of the validation data set. The time encoding of an 8-bit input ($X$[7:0]) consumes 14.71 input clock cycles on an average (over five convolution layers) using CTD, whereas the conventional time-domain approach consumes 129 cycles when simulated on AlexNet over 1000 images, thereby improving the throughput by 88.60% compared with the conventional time-domain approach [see Fig. 9(a)]. To further improve the throughput, an 8-bit input is time encoded using CTD in two phases: four MSBs ($X$[7:4]) and four LSBs ($X$[3:0]) of input activation, as shown in Fig. 9(b). Thus, the time encoding of 8-bit input ($X$[7:0]) in two phases (4 bits each) on the AlexNet over 1000 images consumes 7.15 input clock cycles on an average, further improving the throughput by 51.40%, compared with the CTD encoding of an 8-bit input activation in a single phase.

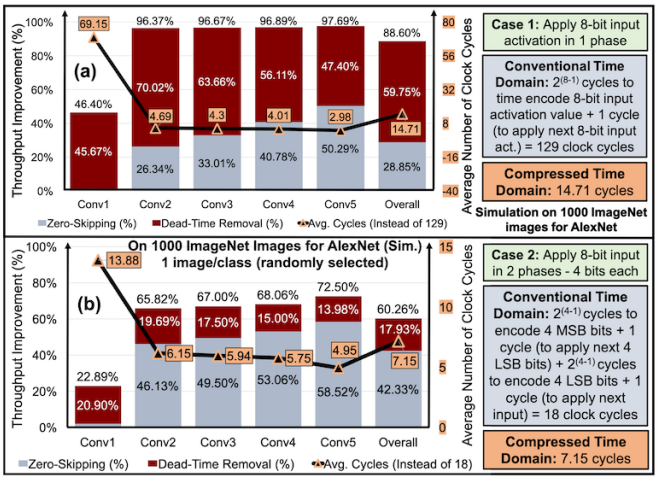



Fig. 9. Throughput Improvement in the proposed CTD approach to time-encode 8-bit input activation in (a) one phase of 8 bit and (b) two phases of 4 bit.

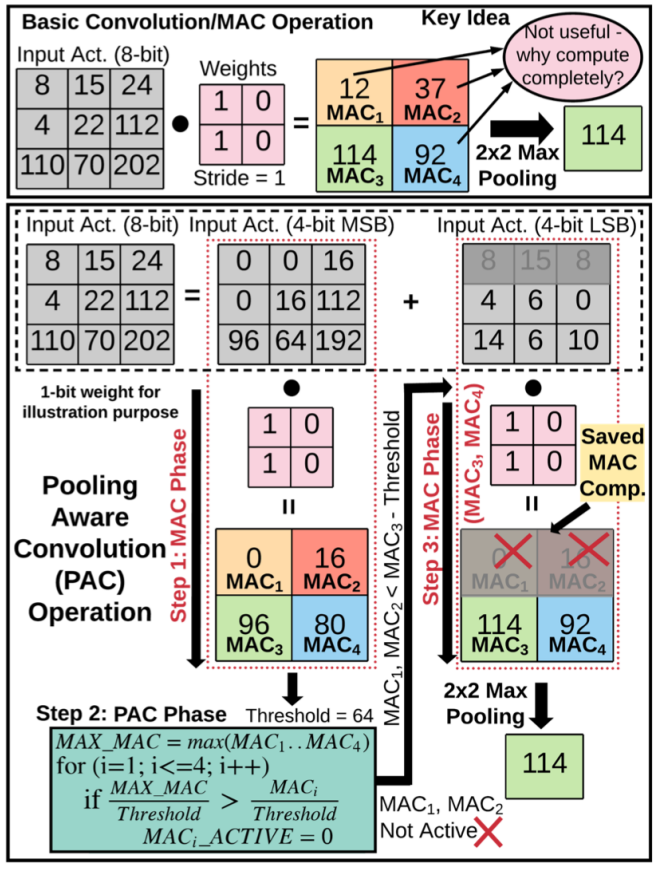



Fig. 10. Proposed concept of the PAC operation.

## D. PAC to Reduce # of MACs

*1) Concept of PAC:* A generic CNN comprises of a pooling layer to perform a sub-sampling operation. For instance, in $2 \times 2$ max-pooling, only the maximum value among the four ($2 \times 2$) MACs is fed to the next layer, and the remaining MAC outputs are not used (see Fig. 10). In the proposed PAC approach, these redundant MACs are identified during the MAC operation and are not completely computed to save energy. MAC operation is computed in multiple phases (accumulating products of input/weight with a smaller bit width in each phase), separated by a comparison phase. In the comparison phase, the MAC values that are smaller than the maximum MAC value by a certain predefined threshold are

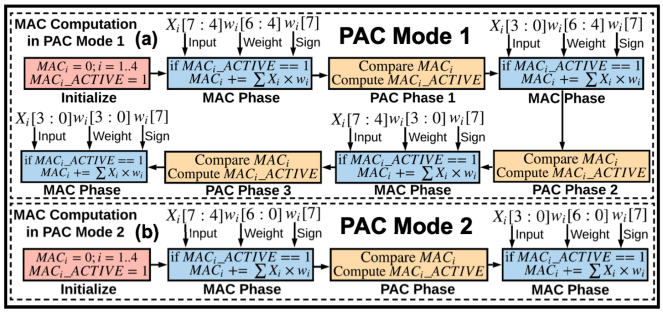



Fig. 11. MAC computation process in (a) PAC Mode 1 and (b) PAC Mode 2.

TABLE II

THRESHOLD VALUES FOR DIFFERENT RUNS AND ALEXNET CONV. LAYERS IN (A) PAC MODE 1 AND (B) PAC MODE 2

| (a) Run | Threshold Values for PAC Mode 1 - AlexNet CNN | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Layer – Conv1 | | | Layer – Conv2 | | | Layer – Conv5 | | |
| | Pha-se1 | Pha-se2 | Pha-se3 | Pha-se1 | Pha-se2 | Pha-se3 | Pha-se1 | Pha-se2 | Pha-se3 |
| Run1 | $2^{16}$ | $2^{15}$ | $2^{10}$ | $2^{12}$ | $2^{12}$ | $2^{9}$ | $2^{10}$ | $2^{11}$ | $2^{7}$ |
| Run2 | $2^{15}$ | $2^{14}$ | $2^{10}$ | $2^{11}$ | $2^{12}$ | $2^{8}$ | $2^{10}$ | $2^{11}$ | $2^{7}$ |
| Run3 | $2^{15}$ | $2^{14}$ | $2^{10}$ | $2^{11}$ | $2^{11}$ | $2^{7}$ | $2^{10}$ | $2^{11}$ | $2^{7}$ |
| Run4 | $2^{15}$ | $2^{14}$ | $2^{10}$ | $2^{10}$ | $2^{10}$ | $2^{6}$ | $2^{10}$ | $2^{11}$ | $2^{7}$ |
| Run5 | $2^{14}$ | $2^{13}$ | $2^{9}$ | $2^{9}$ | $2^{10}$ | $2^{6}$ | $2^{10}$ | $2^{10}$ | $2^{6}$ |
| Run6 | $2^{13}$ | $2^{12}$ | $2^{8}$ | $2^{8}$ | $2^{9}$ | $2^{5}$ | $2^{9}$ | $2^{9}$ | $2^{5}$ |
| Run7 | $2^{12}$ | $2^{12}$ | $2^{8}$ | $2^{8}$ | $2^{8}$ | $2^{4}$ | $2^{8}$ | $2^{8}$ | $2^{4}$ |
| Run8 | $2^{12}$ | $2^{12}$ | $2^{8}$ | $2^{7}$ | $2^{8}$ | $2^{4}$ | $2^{8}$ | $2^{8}$ | $2^{4}$ |

| (b) Run | Threshold Values for PAC Mode 2 - AlexNet | | |
|---|---|---|---|
| | Layer Conv1 | Layer Conv2 | Layer Conv5 |
| RunA | $2^{12}$ | $2^{10}$ | $2^{8}$ |
| RunB | $2^{12}$ | $2^{10}$ | $2^{7}$ |
| RunC | $2^{12}$ | $2^{9}$ | $2^{6}$ |
| RunD | $2^{11}$ | $2^{8}$ | $2^{5}$ |
| RunE | $2^{10}$ | $2^{7}$ | $2^{5}$ |
| RunF | $2^{10}$ | $2^{6}$ | $2^{6}$ |
| RunG | $2^{10}$ | $2^{6}$ | $2^{5}$ |

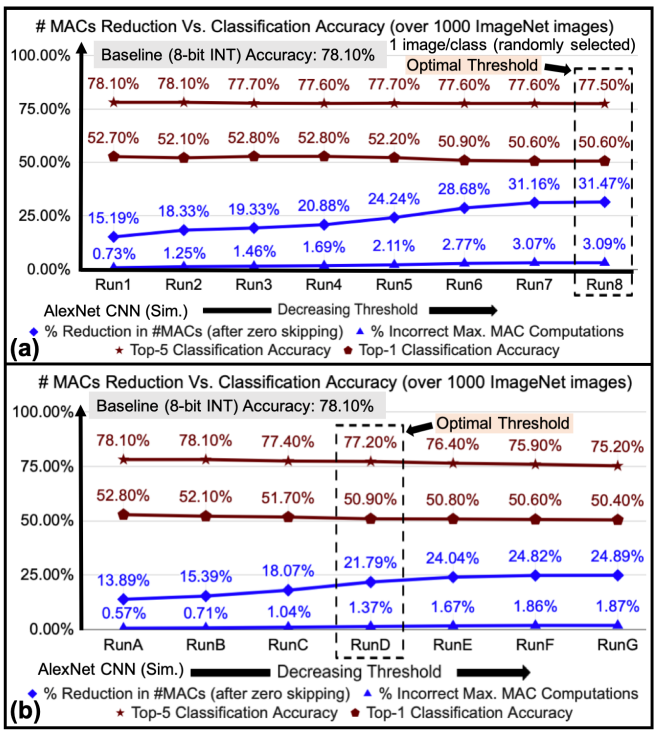



Fig. 12. Reduction in # of MACs and classification accuracy on AlexNet for different threshold values in (a) PAC Mode 1 and (b) PAC Mode 2.

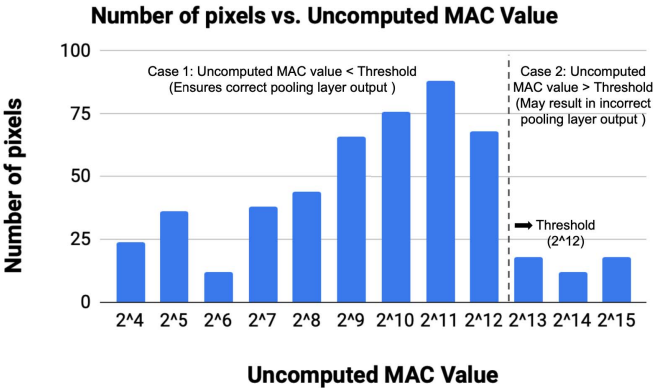



Fig. 13. Illustration of selecting optimal threshold value.

eliminated, and the remaining bits of these MACs are not computed in subsequent phases, thereby reducing the total number of bit-wise MACs. For example, the MAC operation with four MSBs ($X$[7:4]) of the input activation and weight is first computed (see Fig. 10). This is followed by a comparison operation that identifies and eliminates those MAC values (MAC1 and MAC2), which are smaller than the maximum value by a certain predefined threshold. The non-eliminated MAC values (MAC3 and MAC4) are then fully computed by accumulating the products of four LSBs ($X$[3:0]) of the inputs and weight.

*2) PAC Modes in COMPAC CNN Engine:* In this work, two modes of PAC are proposed for the 8-bit input activation and 8-bit weight MAC operation. In PAC mode 1, input activation and weight values are applied in four MAC phases (4 bits of input and weight in each phase), with a comparison PAC phase in between each MAC phase [see Fig. 11(a)], whereas PAC mode 2 comprises of two MAC phases (computing MAC for 4-bit input and 8-bit weight in each phase) separated by a comparison PAC phase [see Fig. 11(b)]. It should be noted that the chosen threshold values are multiples of $2^n$ such that the shifters and comparators can be reused during the PAC phase operation, thereby eliminating the need for additional circuitry.

*3) Simulation Results of PAC on AlexNet CNN:* The simulation results on the Alex Net CNN over 1000 ImageNet validation data set images (one image randomly chosen from each class) for different thresholds (see Table II) show up to 31.47% (21.79%) reduction in the number of non-zero input activations MACs with a top-five classification accuracy loss of 0.60% (0.90%) and a top-one classification accuracy loss of 2.20% (1.90%) for the PAC modes 1 (2), respectively (see Fig. 12). The reduction in the number of MACs translates to power savings since the MDLs remain idle for those MACs that are not completely computed. Thus, MDL power is saved by the same amount by which the number of MAC operations is reduced. The power overhead due to shifters and comparators is insignificant compared with MAC computation since the PAC phase (duration: two cycles) is followed by a MAC accumulation phase, comprising of many cycles [equals the filter dimensions $*$ channels $*$ 50.05 (cycles for 1 MAC)]. Thus, the power overhead of PAC circuit elements can be ignored since they remain inactive most of the time, and effective power savings equal reduction in MDL power.

Table II lists down the threshold values used in PAC modes 1 and 2. The optimal values of these threshold values are determined using an empirical model discussed as follows. First, for each PAC phase in PAC modes 1 or 2,

TABLE III

COMPARISON OF NUMBER OF MACS, ACCURACY, AND ON-CHIP ACCESSES OF THE PROPOSED PAC MODES 1 AND 2 WITH BASELINE, NON-ZERO INPUT, AND PC [28] APPROACHES

| Operation Mode – AlexNet CNN | #MACs (Million) | Top-5 Acc. (%) | Top-1 Acc. (%) | Incorrect Max MAC (%) | On-chip Access (MB) |
|---|---|---|---|---|---|
| Baseline | 666.0 | 78.10% | 52.80% | 0.00% | 13.58 (1.00 x) |
| Non-zero input | 325.3 | 78.10% | 52.80% | 0.00% | 13.58 (1.00 x) |
| Prec. Cascading | 202.5 | 70.50% | 43.60% | 22.13% | 19.60 (1.44 x) |
| PAC Mode 1 | 222.9 | 77.50% | 50.60% | 3.09% | 21.80 (1.60 x) |
| PAC Mode 2 | 270.1 | 77.20% | 50.90% | 1.37% | 14.67 (1.08 x) |

the uncomputed MAC value (MAC value—partially computed MAC value till a given PAC phase) is determined for each pixel for a given convolution layer output over ImageNet validation data set images. Here, the uncomputed MAC value signifies the remaining MAC value, which would be added to the partially computed MAC value after a given PAC phase. Next, the frequency bin distribution of these uncomputed MAC values for all the pixel values for a given layer is plotted (see Fig. 13), where the $X$-axis represents the range of uncomputed MAC values, and the $Y$-axis represents the frequency (number of uncomputed MAC values). The lowest value on the $X$-axis, which covers most of the area under the graph (most of the bins), is chosen as the optimal threshold value for a given PAC phase and convolution. The uncomputed MAC values that lie to the left of this chosen threshold value on the graph (case 1 in Fig. 13) would not result in incorrect output values after max-pooling layer operation, thereby ensuring minimal accuracy loss while maximizing the reduction in the number of MACs.

There is a tradeoff between classification accuracy and a reduction in the number of MACs. A higher threshold value results in a lesser reduction in the number of MACs, thereby having a low chance of predicting the incorrect maximum MAC value, leading to lesser accuracy loss, as shown in Fig. 12. Table III summarizes the number of MACs, top-one/top-five classification accuracy, and the on-chip SRAM access overhead for the AlexNet CNN without/with PAC modes. The proposed PAC approach results in an on-chip SRAM access overhead of 60.53% (8.03%) for the PAC modes 1 (2) (see Table III). It should be noted that the proposed PAC approach does not affect the off-chip DRAM accesses. Considering the on-chip access overhead, PAC still results in overall energy reduction. PAC mode 2 results in a 21.79% reduction in the number of MAC operations, which translates to a reduction of 145.12 million MAC operations (total MACs $=$ 666M for AlexNet). This is achieved at an on-chip access overhead of 14.67–13.58 MB $=$ 1.09 MB (refer Section IV-C). Assuming that the on-chip access energy is 6$\times$ times the MAC compute operation [4], it translates to the MAC compute overhead of $1.09 * 1024 * 1024 * 6 = 6.86$ million MACs. Therefore, the net MAC savings using the proposed PAC Mode 2 approach are $145.12 - 6.86 = 138.26$ million MACs (equals 20.76% of total MACs). Thus, the tradeoffs between the reduced number of MACs, classification accuracy loss, and on-chip access overhead need to be carefully analyzed. Note that the proposed PAC technique is independent of underlying MDL-based time-domain implementation and can be applied for other digital/analog MAC implementations.

*4) Comparison of PAC With Precision Cascading (PC) [28]:* Kim and Seo [28] proposed a PC approach to reduce the bit-wise MACs for the convolution layers that are followed by a max-pooling layer. The goal of the PC approach is similar to the proposed PAC approach, i.e., not to fully compute the redundant MACs. In the PC approach, the MAC value is computed in eight rounds, computing MAC for 8-bit weight and 1-bit input value in each round. The maximum MAC value is predicted if it exceeds the remaining MAC values by at least 1. Thus, the PC approach may lead to higher MAC savings at reduced classification accuracy compared with the proposed PAC approach since it has higher chances of predicting incorrect maximum MAC value having a discard threshold of only 1. We compared the PC approach with the proposed PAC approach for the # of MACs and classification accuracy on the AlexNet CNN. The simulation results on the AlexNet CNN over 1000 ImageNet data set images show that the PC approach results in 9.15% fewer # of MACs (202.5 million instead of 222.9 million) at a cost of 7% drop (70.5% instead of 77.5%) in the top-five classification accuracy compared with the PAC-1 mode proposed in this work (see Table III).

### *E. EEDF for Optimal Data Movement*

In deep neural networks, such as AlexNet and VGG, millions of MAC operations are computed using a large number of weight parameters and input activations. Significant energy is spent on accessing these parameters and activations. All the activations and parameters cannot be stored on-chip due to limited on-chip SRAM capacity, resulting in significant off-chip access energy. COMPAC proposes an EEDF to reduce the on-/off-chip accesses. The input activations and weights for any given convolution stride are stored on an on-chip SRAM, as given by (1), such that all the column values for the accessed row are utilized in the MAC operation, thereby minimizing the total number of on-chip accesses (see Fig. 14). This is achieved by saving the slice (part) of input activation/weight in each row such that the width ($w$) and height ($h$) of the slice do not exceed the convolution stride. The depth ($d$) of the slice is chosen such that it maximizes the storage utilization of the SRAM array. In the proposed COMPAC CNN engine implementing the AlexNet CNN, 256 column-wide SRAM banks are used. The $w$, $h$, and $d$ values (width, height, and depth of the slice storing input activations and weight parameters in a single SRAM row) are set as 4, 2, 3 and 1, 1, 32 for the convolution layers 1 and 2–5, respectively

$$w \times h \times d \times \text{data-width} \leq \#\text{ of columns in SRAM Array} \quad \text{where, w and h} \leq \text{convolution stride.} \tag{1}$$

Besides optimally mapping the parameters and activations to the SRAM array for the time-domain MAC computation, the proposed COMPAC CNN engine comprising of 11 SRAM banks (B1–B11) with an on-chip capacity of 67 kB provides configurable bank architecture to maximize the input activation and weight reuse for different convolution layers of the

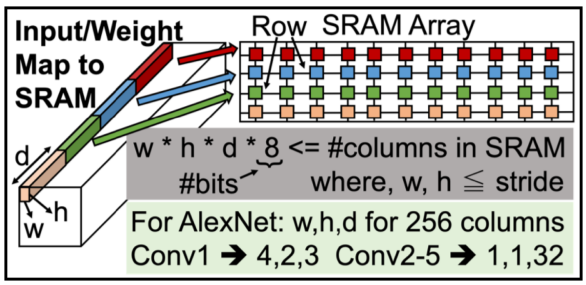



Fig. 14. Convolution-stride-aware data storage technique for optimal on-/off-chip accesses.

TABLE IV
CONFIGURABLE SRAM BANK STRUCTURE TO STORE INPUT ACTIVATIONS, WEIGHTS, AND OUTPUT ACTIVATIONS FOR ALEXNET CNN

| SRAM Banks: B1-B7 (8KB), B8-B9 (4KB), B10 (2KB), B11 (1KB) | | | | | | |
|---|---|---|---|---|---|---|
| Layer | Input Activation | | Weights | | Output Activation | |
| | Banks | Capacity | Banks | Capacity | Banks | Capacity |
| Conv1 | B1-B5, B8 | 44 KB | B6-B7, B10 | 18 KB | B9, B11 | 5 KB |
| Conv2 | B1, B8 | 12 KB | B2-B7, B10 | 50 KB | B9, B11 | 5 KB |
| Conv3 | B1, B8, B9 | 16 KB | B2-B7, B10 | 50 KB | B11 | 1 KB |
| Conv4 | B1, B8 | 12 KB | B2-B7, B9, B10 | 54 KB | B11 | 1 KB |
| Conv5 | B1, B8 | 12 KB | B2-B7, B9, B10 | 54 KB | B11 | 1 KB |

AlexNet CNN. The configurable bank architecture helps in achieving the optimum number of accesses by allocating more space to the input activations for convolution layer 1 and to the weight parameters for the layers 2–5 (see Table IV).

*1) On-Chip Access Reduction Techniques:* The on-chip SRAM accesses are optimized using a configurable SRAM bank architecture that supports the convolution-stride-aware mapping of the input activations and weights. Furthermore, input activations, weights, and partial sum data movement are further optimized, as discussed in the following.

The input activation values ($X_1$–$X_4$) stored on the SRAM array are accessed and temporarily stored in the input activation shift registers. The inputs are accessed from these registers while performing time encoding and the MAC operations. Typically, a convolution stride is less than a filter window size. Therefore, a lot of input activation values ($X_1$–$X_4$) are replicated when stored in these shift registers. To save the on-chip input activation accesses, the inputs from the SRAM array are broadcasted to these shift registers such that each input activation is accessed only once and stored in multiple input shift registers (if required), as shown in Fig. 15. Furthermore, inputs ($X_1$–$X_4$) are shared across 32 filters (F1–F32 implemented in COMPAC) to perform four MACs per each filter (see Fig. 15).

Each filter comprises four MAC engines, each supporting an MDL, a counter, and a shifter to perform the MAC operation. Four MAC operations are performed using the same weight (stored in weight register of a filter) and different input activation ($X_1$–$X_4$). Thus, the weight value is reused across the four MACs for each filter, as shown in Fig. 16. As discussed earlier, multi-bit input/weight MAC operation is performed sequentially by adding the product of time-encoded input pulse with each weight bit (wt[6]–wt[0]). The MAC operation is

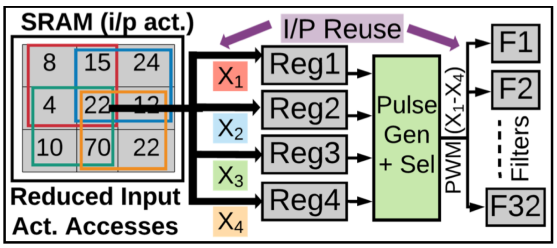



Fig. 15. Proposed input activation reuse techniques in COMPAC CNN engine.

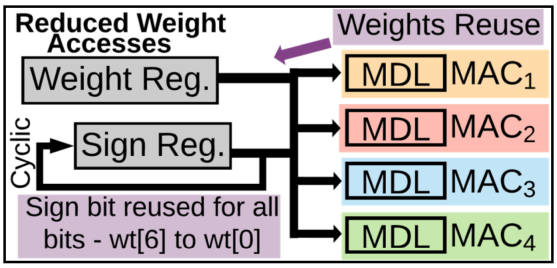



Fig. 16. Proposed weight-reuse techniques in the COMPAC CNN engine.

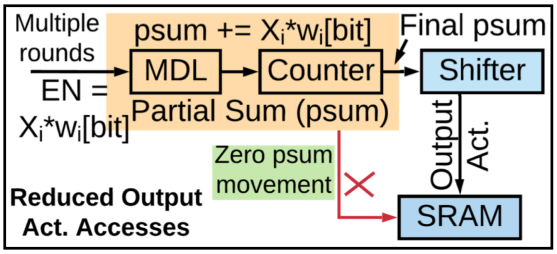



Fig. 17. Proposed output activation reuse techniques in the COMPAC CNN engine with zero partial sum movement.

computed by accumulating products of input pulsewidth with the highest significant weight bit (stored locally in weight shift register), and the products of these inputs with less-significant weight bits are computed in subsequent cycles. However, the weight sign bit (wt[7]) is accessed only once from the SRAM and stored locally in a weight sign cyclic shift register. The sign bits are rotated to reuse for the different weight bits (wt[6]–wt[0]), thereby reducing the on-chip access energy in accessing the sign bit of the weight (see Fig. 16). The weight/input activation reuse and configurable bank architecture schemes are implemented using the address generators, controllers, and bank select logic that occupy 0.025 $mm^2$ (1.44% of the total core area). The power overhead of these elements can be ignored since only 0.00431 cycles (discussed in Section IV-C) are consumed on an average for each MAC operation (compute time equals 50.05 cycles) to access the on-chip data.

In the proposed COMPAC CNN engine, the MAC operation is computed in the time domain by sequentially accumulating the products of input activation with each weight bit. The partial sum value is stored locally in the counter and is not sent (received) to (from) the global buffer. Once all the products of input activation and weight are accumulated, the fully

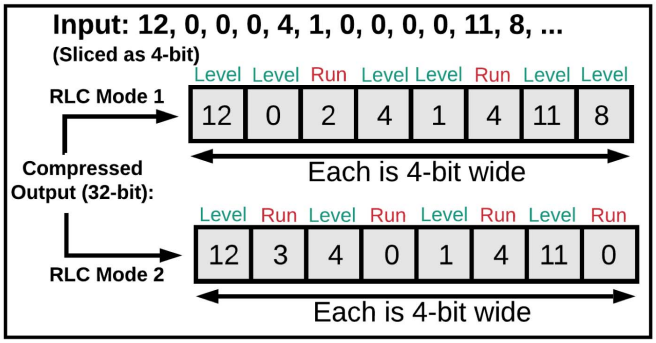
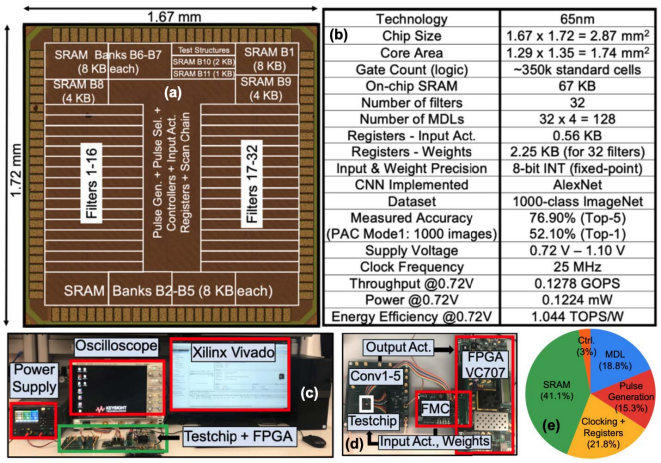



Fig. 18. Run length compression codes for the proposed COMPAC CNN engine in RLC modes 1 and 2.




| | Technology | 65nm |
|---|---|---|
| (b) | Chip Size | 1.67 x 1.72 = 2.87 mm² |
| | Core Area | 1.29 x 1.35 = 1.74 mm² |
| | Gate Count (logic) | ~350k standard cells |
| | On-chip SRAM | 67 KB |
| | Number of filters | 32 |
| | Number of MDLs | 32 x 4 = 128 |
| | Registers - Input Act. | 0.56 KB |
| | Registers - Weights | 2.25 KB (for 32 filters) |
| | Input & Weight Precision | 8-bit INT (fixed-point) |
| | CNN Implemented | AlexNet |
| | Dataset | 1000-class ImageNet |
| | Measured Accuracy (PAC Mode1: 1000 images) | 76.90% (Top-5) 52.10% (Top-1) |
| | Supply Voltage | 0.72 V – 1.10 V |
| | Clock Frequency | 25 MHz |
| | Throughput @0.72V | 0.1278 GOPS |
| | Power @0.72V | 0.1224 mW |
| | Energy Efficiency @0.72V | 1.044 TOPS/W |

Fig. 19. (a) 65-nm test-chip die micrograph. (b) Summary table. (c) and (d) FPGA-based lab measurement setup. (e) Test-chip power breakdown.

computed MAC output is written to the on-chip global buffer (see Fig. 17). This results in a significant reduction in the on-chip memory accesses of the output activations since there is no partial sum movement.

*2) Off-Chip Access Reduction Techniques:* The off-chip DRAM accesses are optimized using a configurable SRAM bank architecture supporting convolution-stride-aware mapping of input activations and weights. In addition, the RLC technique [4] can be used in the proposed COMPAC approach to further reduce the off-chip DRAM accesses, by exploiting the zeros in the input activations (due to preceding ReLU activation layer) and weights. It should be noted that RLC is not implemented in our test chip. However, simulations supporting two different RLC codes are performed to evaluate the off-chip DRAM access savings. Fig. 18 shows both the modes of RLC encoding proposed for the COMPAC CNN engine. There are two types of fields in an RLC compressed output—"Run" and "Level." The consecutive number of zeros to a maximum run length of 15 is represented using a 4-bit number in the Run field, whereas a 4-bit value is inserted directly in the level field. Two (four) runs and Six (four) levels are packed into a 32-bit word, which is sent through a 32-bit input data bus in RLC modes 1 (2), respectively (see Fig. 18). These two RLC modes are supported to maximize the overall compression (minimize the total off-chip accesses) for the weights/input activations for all the convolution layers of the AlexNet. RLC Mode 1 is used to compress the weights for all the convolution layers 1–5 and input activations of convolution layer 1 (since the input activations have a lesser number of zeros), whereas RLC mode 2 is used to compress input activation for the remaining (2–5) convolution layers of the AlexNet CNN (since the input activations for these layers have a large number of zeros due to the preceding ReLU activation layer). The simulation results of the RLC access savings on the AlexNet CNN over the ImageNet data set are discussed later in Section IV-D2.

## IV. Measured Results

### A. Measurement Setup and Test-Chip Summary

Fig. 19 shows the die micrograph, test-chip summary, and FPGA-based test measurement setup of the proposed COMPAC CNN engine implemented using commercial 65-nm CMOS technology. The test chip implements the AlexNet CNN to classify 1000-class ImageNet data set and occupies a total core area of 1.74 mm$^2$. The overall CNN inference and training methodology consists of: 1) training input data using TensorFlow/Keras [29] software framework; 2) feeding trained filter weights and input activation values for the convolution layers 1–5 to the test chip using Xilinx Virtex-7 FPGA VC707 Evaluation board [30] and Xilinx FMC XM105 Debug Card [31]; 3) performing on-chip MAC, averaging, and pooling operations for convolution layers 1–5 of the AlexNet; and 4) FCN and soft-max layers computation in software (TensorFlow). Due to limited on-chip global buffer capacity and the number of filters, the input activations, the weights, and the output activations (pooled MAV value) are not entirely stored on-chip to compute all the MACs for any given convolution layer. The test chip supports MAC computation for 32 filters and a portion of input activation for any given convolution layer. Thus, multiple iterations are performed, each time feeding in weights (for 32 filters) and a portion of input activations from the Xilinx FPGA/FMC board to the test chip using a 32-bit input data bus and storing the output activations from the test chip on the Xilinx FPGA using a 32-bit output data bus. It should be noted that the input activation and weight address generators/controllers are designed in such a way that it optimizes for data accesses for AlexNet CNN, and the off-chip data access address controller supports only AlexNet CNN. These controllers can be made flexible for supporting other CNNS for optimal data accesses.

### B. Test-Chip Characterization

The functionality of scaling the time residue on the MDL is validated on the 65-nm test chip. Fig. 20 shows the oscilloscope captured waveforms of the MDL nodes—A, M, and E for the time accumulation and TRS modes. MDL operation is performed while operating at a 400-mV core voltage and a 1-V pad voltage. It can be observed that the state vector on MDL is advanced resulting in transitions during the time accumulation phase, whereas the state of MDL is changed

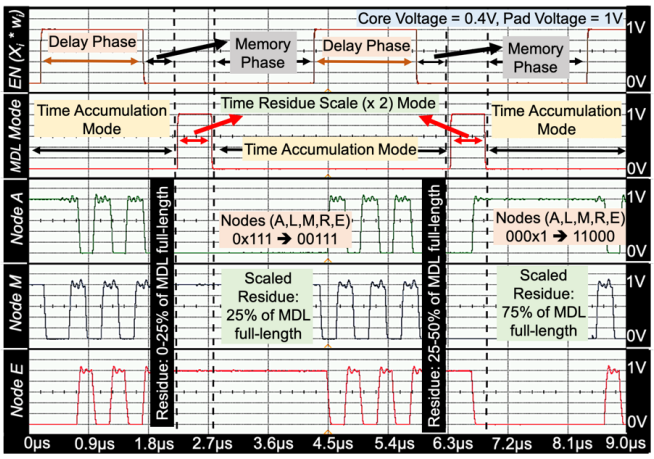



Fig. 20. Experimental demonstration of TRS on bidirectional MDL on 65-nm test chip.

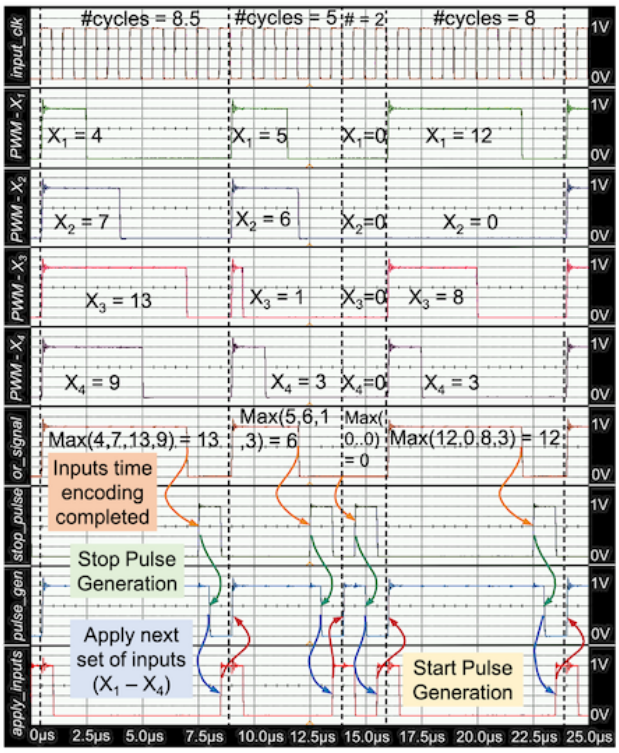



Fig. 21. Experimental demonstration of the CTD approach while operating at 0.5-V core Vcc and 1-V pad Vcc on the 65-nm test chip.

to one of the predefined MDL states (using the TRS lookup table), which has the time residue closest to the actual scaled ($\times 2$) residue. For instance, the actual time residue during the two TRS phases lies in the range of 0%–25% and 25%–50% of the MDL full-length delay, which is scaled to 25% and 75% of the MDL full-length delay values, respectively, to scale the time residue, as shown in Fig. 20. These waveforms confirm the successful operation of the TRS approach.

The experimental demonstration of the CTD approach in COMPAC CNN engine on a 65-nm test chip, while operating at a 0.5-V core Vcc, a 1-V pad Vcc, and a 1-MHz input clock

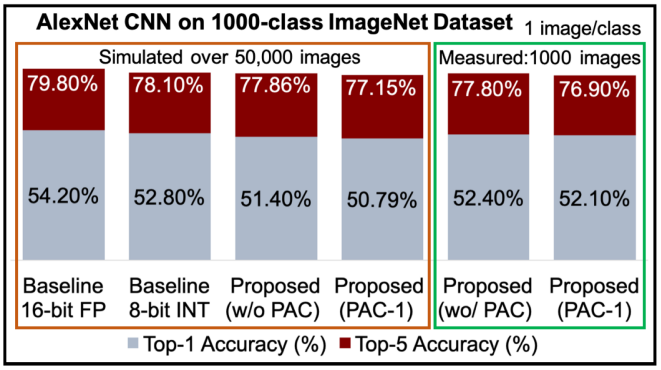



Fig. 22. Measured and simulated classification accuracy of the proposed COMPAC CNN engine for AlexNet CNN on the 1000-class ImageNet data set.

frequency, is shown in Fig. 21. Four 4-bit input activations ($X_1$–$X_4$) are encoded in the time domain by selecting the appropriate PWM signal from the pulse generation module, as shown by the oscilloscope captures PWM $X_1$–$X_4$. The *stop_pulse* control signal is generated on the next clock edge whenever a negative edge or zero value is observed in the *or_signal* (one of the time-encoded input signals with the maximum pulsewidth). The pulse generation stops on the next subsequent clock edge, and a new set of inputs is applied to the pulse-selection module on the next to next subsequent clock edge by shifting the values from the input activation shift registers using an *apply_inputs* control signal. Finally, the pulse generator starts its operation on the next clock edge, thereby adding a total time overhead of two clock cycles in the CTD approach. These waveforms (see Fig. 21) confirm the successful operation of the proposed concept of the CTD approach to improve the throughput in time encoding the input activations.

### C. Accuracy, Throughput, and Energy-Efficiency Results

The test-chip measurements on the AlexNet CNN over ImageNet data set images (randomly chosen 1 image from each class) achieve a top-five classification accuracy of 76.90% (77.80%) and a top-one classification accuracy of 52.10% (52.40%) with (without) the PAC approach (see Fig. 22). In order to quantify COMPAC's CNN engine classification performance on the entire 50 000 ImageNet validation data set images, the non-ideal circuit effects of MDL are incorporated into the COMPAC CNN/Tensorflow simulation framework. The simulation results achieve a top-five classification accuracy of 77.15% (77.86%) and a top-one classification accuracy of 50.79% (51.40%) with (without) the PAC approach (see Fig. 22), showing a classification accuracy loss within 1% compared with the 8-bit fixed-point software implementation.

The proposed COMPAC CNN engine achieves a peak throughput of 0.1278 GOPS on the AlexNet while operating at a 0.72-V Vcc and a 25-MHz frequency and consuming 0.1224 mW of power (measured), thereby making it suitable for an energy-efficient edge computation [see Fig. 19(b)]. The throughput is calculated as follows. There are 32 filters on the test chip, each comprising of four MDLs. Thus,

TABLE V
ON-/OFF-CHIP ACCESSES FOR THE PROPOSED COMPAC CNN ENGINE WITHOUT AND WITH PAC MODES FOR ALEXNET CNN

| Layer | On-chip SRAM Accesses (MB) | | | | | | | | | | | | Off-chip DRAM Accesses (MB) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Without PAC | | | | PAC Mode 1 | | | | PAC Mode 2 | | | | Without RLC | | | | With RLC | | | |
| | Input Act. | Wei-ghts | Output Act. | Total | Input Act. | Wei-ghts | Output Act. | Total | Input Act. | Wei-ghts | Output Act. | Total | Input Act. | Wei-ghts | Output Act. | Total | Input Act. | Wei-ghts | Output Act. | Total |
| Conv1 | 2.67 | 0.05 | 0.07 | 2.79 | 5.34 | 0.12 | 0.07 | 5.53 | 2.67 | 0.12 | 0.07 | 2.86 | 0.66 | 0.05 | 0.07 | 0.78 | 0.62 | 0.04 | 0.07 | 0.73 |
| Conv2 | 3.62 | 0.39 | 0.05 | 4.06 | 7.24 | 0.88 | 0.05 | 8.17 | 3.62 | 0.88 | 0.05 | 4.55 | 0.98 | 0.39 | 0.05 | 1.42 | 0.63 | 0.33 | 0.05 | 1.00 |
| Conv3 | 2.53 | 0.84 | 0.11 | 3.48 | 2.53 | 0.84 | 0.11 | 3.48 | 2.53 | 0.84 | 0.11 | 3.48 | 1.48 | 0.84 | 0.11 | 2.43 | 0.97 | 0.76 | 0.11 | 1.84 |
| Conv4 | 1.27 | 0.63 | 0.07 | 1.97 | 1.27 | 0.63 | 0.07 | 1.97 | 1.27 | 0.63 | 0.07 | 1.97 | 0.74 | 0.63 | 0.07 | 1.44 | 0.47 | 0.60 | 0.07 | 1.14 |
| Conv5 | 0.85 | 0.42 | 0.01 | 1.28 | 1.69 | 0.95 | 0.01 | 2.65 | 0.85 | 0.95 | 0.01 | 1.81 | 0.49 | 0.42 | 0.01 | 0.92 | 0.28 | 0.41 | 0.01 | 0.70 |
| **Total** | **10.94** | **2.33** | **0.31** | **13.58** | **18.07** | **3.42** | **0.31** | **21.80** | **10.94** | **3.42** | **0.31** | **14.67** | **4.35** | **2.33** | **0.31** | **6.99** | **2.97** | **2.14** | **0.31** | **5.41** |

128 MDLs are operational at a time, each computing MAC operation. Since each MAC operation is considered as two operations (MAC), the total number of operations computed in parallel is $128 \times 2 = 256$ operations. In the CTD approach (case 2), as discussed in Section III-C, the total time taken to compute a 8-bit input and a weight MAC operation is 7.15 cycles (to time-encode an 8-bit input) $\times 7$ (sequential accumulation with each weight bit, excluding the sign bit) $= 50.05$ cycles. Thus, 256 operations are computed in 50.05 clock cycles. The total latency overhead due to on-/off-chip accesses is 0.007 cycles/operation (calculated by dividing the total on-/off-chip accesses by a product of data bus width and the total number of MACs). This is calculated as follows. Without RLC, the off-chip DRAM access is 6.99 MB to compute 666M MAC operations for AlexNet (see Table V). The data bus is 32-bit wide. This translates to $(6.99 \times 1024 \times 1024 \times 8)/(32 \times 666 \times 10^6) = 0.00275$ cycles per MAC. The weights and input activations are accessed on-chip for each layer in parallel; hence, the latency overhead is the max (cycles took to store inputs in registers and cycles took to store weights in registers) = cycles took to store inputs in registers (since input access > weight access). Thus, the total clock cycles consumed to access the input activations for 666 million MAC operations in AlexNet CNN are 10.94-MB/32-bit wide bus (see Table V), translating to $(10.94 \times 1024 \times 1024 \times 8)/(32 \times 666 \times 10^6) = 0.00431$ cycles/MAC. Adding the on-chip and off-chip access overhead, the latency overhead in accesses is $(0.00275 + 0.00431) = 0.007$ cycles per MAC operation.

Thus, the measured throughput of the proposed COMPAC CNN engine is 256 operations/(50.05 + 0.007 cycles $\times$ 40-ns clock period) = 0.1278 GOPS. The measured peak energy efficiency is then calculated by dividing the throughput by the measured on-chip (excluding the leakage and off-chip DRAM access power), as 0.1278 GOPS/0.1224 mW = 1.044 TOPS/W. It should be noted that the operating frequency of 25 MHz is limited by the measurement test-infrastructure and not fundamentally by the MDL-based time-domain approach. Based on the post place and route simulations, the proposed COMPAC design is functional up to 800 MHz (potentially 32$\times$ higher throughput).

### D. On-/Off-Chip Data Accesses for AlexNet CNN

*1) On-Chip Data Accesses:* Table V summarizes the on-chip data accesses of the proposed COMPAC CNN engine for input activations, weights, and output activations, over all the five convolution layers of the AlexNet CNN with and without PAC. The COMPAC data flow optimizes for input activation/weight reuse and eliminates the partial sum data movement. COMPAC data flow results in 13.58 MB of on-chip data accesses to compute 666 million MAC operations for the convolution layers 1–5 of the AlexNet CNN, leading to on-chip access of 0.0214 bytes per 8-bit input/weight MAC operation. As discussed earlier, PAC modes 1 (2) lead to an on-chip access overhead of 60.53% (8.03%), thereby leading to 21.80 MB (14.67 MB) of on-chip data accesses. PAC mode 1 incurs higher overhead since both the weights and inputs are accessed again, whereas only weights are accessed again in PAC mode 2, leading to a lower on-chip access overhead, compared with the baseline (wo/PAC) approach.

*2) Off-Chip Data Accesses:* Table V summarizes the off-chip data accesses of the proposed COMPAC CNN engine for the input–output activations and weights over all the five convolution layers of the AlexNet CNN, with and without the RLC technique. The proposed COMPAC data flow supports configurable SRAM banks with 67-KB capacity, optimal mapping methodology to store the input activations and weights for any given convolution stride on the SRAM bank, and RLC techniques, to optimize for the off-chip data accesses. The proposed COMPAC data flow results in 6.99 MB (5.41 MB) of off-chip data accesses to compute 666 million MAC operations for the convolution layers 1–5 of the AlexNet CNN, leading to off-chip access of 0.011 (0.0085) bytes per 8-bit input/weight MAC operation without (with) RLC technique, respectively.

### E. Comparison With Prior Energy-Efficient CNN Accelerators

Table VI compares the proposed COMPAC CNN engine with prior energy-efficient test chips [4], [14]–[18], implementing the AlexNet CNN. The proposed COMPAC CNN engine occupies the least area despite implementing it at a relatively old CMOS technology node (65 nm) compared with other test chips. The proposed COMPAC CNN supports an on-chip global buffer of the least capacity and achieves an energy efficiency of 1.044 TOPS/W (comparable and better) compared with the other test chips at the iso-technology node.

The proposed COMPAC CNN engine results in 86.97% reduced on-chip accesses (13.58 MB instead of 104.25 MB)

TABLE VI

COMPARISON OF THE PROPOSED COMPAC CNN ENGINE WITH PRIOR TEST CHIPS IMPLEMENTING ALEXNET CNN

| Reference | Tech. (nm) | Core Area ($mm^2$) | On-chip SRAM (KB) | Data Width (#bits) | Top-5 Acc. (%) | Energy Efficiency (TOPS/W) |
|---|---|---|---|---|---|---|
| EYERISS [4] | 65 | 12.25 | 181.5 | 16 | N/A | 0.246 |
| ENVISION [14] | 28 | 1.87 | 144 | 1-16 | N/A | 0.80 - 3.80 |
| STICKER [15] | 65 | 7.80 | 170 | 8 | N/A | 1.038 |
| SNAP [16] | 16 | 2.40 | 280.6 | 16 | N/A | 3.86 |
| UNPU [17] | 65 | 16.00 | 256 | 8 | N/A | 4.30 |
| QUEST [18] | 40 | 121.5 | 7864.3 | 4 | 76.70 | 0.877 |
| **This work** | **65** | **1.74** | **67** | **8** | **76.90** | **1.044** |

TABLE VII

COMPARISON OF ON-/OFF-CHIP ACCESSES FOR THE PROPOSED COMPAC CNN ENGINE WITH EYERISS [4] AND UNPU [17] APPROACHES

| Layer | On-chip Access Comparison (MB) | | | | Off-chip Access Comparison (MB) | | | |
|---|---|---|---|---|---|---|---|---|
| | Eyeriss (16-bit) | Proposed CNN (8-bit) | | | Eyeriss (w/ RLC) | UNPU | Proposed (wo/ RLC) | Proposed (w/ RLC) |
| | | wo/ PAC | PAC Mode 1 | PAC Mode 2 | 181.5 KB 16-bit | 256 KB 8-bit | 67 KB 8-bit | 67 KB 8-bit |
| Conv1 | 18.50 | 2.79 | 5.53 | 2.86 | 5.00 | 0.77 | 0.78 | 0.73 |
| Conv2 | 77.60 | 4.06 | 8.17 | 4.55 | 4.00 | 0.89 | 1.42 | 1.00 |
| Conv3 | 50.20 | 3.48 (No pooling layer) | | | 3.00 | 1.59 | 2.43 | 1.84 |
| Conv4 | 37.40 | 1.97 (No pooling layer) | | | 2.10 | 0.97 | 1.44 | 1.14 |
| Conv5 | 24.90 | 1.28 | 2.65 | 1.81 | 1.30 | 0.64 | 0.92 | 0.70 |
| **Total** | **208.50** | **13.58** | **21.80** | **14.67** | **15.40** | **4.86** | **6.99** | **5.41** |

TABLE VIII

IMPACT OF THE KEY IDEAS OF THE PROPOSED COMPAC CNN ENGINE ON PERFORMANCE, ENERGY EFFICIENCY, AND ACCURACY METRICS

| Idea | Through -put | Power | Energy Efficiency | Top-5 Accuracy | Area Overhead | Access Energy |
|---|---|---|---|---|---|---|
| **TRS** | No effect (<0.1%) | No effect (<0.1%) | No effect (<0.1%) | Increase (3%) | Increase 0.71% | No effect |
| **CTD** | Increase (94.46%) | Increase ~3% | Increase (>90%) | No effect | Increase 0.016% | No effect |
| **PAC** | No effect | Decrease 22% MDL | Increase 22% comp. | Decrease <1% | No effect | Increase 8%on-chip |
| **EEDF** | No effect | Increase ~3% | Increase Access Energy Eff. | No effect | Increase 1.44% | Decrease 87(30)% on(off)chip |

compared with an Eyeriss approach [4], considering iso-8-bit precision (see Table VII). The proposed COMPAC CNN engine with PAC also results in significant on-chip access savings: 79.10% and 85.93% in PAC modes 1 and 2, respectively (see Table VII); thereby making the proposed COMPAC CNN engine much more energy efficient in terms of on-chip data access energy compared with Eyeriss [4].

The proposed COMPAC CNN engine results in 29.74% reduced off-chip accesses (5.41 MB instead of 7.70 MB) compared with an Eyeriss approach [4], with lesser on-chip SRAM buffer capacity (67 kB instead of 90.75 kB) and considering iso-8-bit precision (see Table VII). The proposed COMPAC CNN engine compared with UNPU design [17] has 11.32% increased off-chip accesses (5.41 MB instead of 4.86 MB) at iso-8-bit precision. However, the on-chip SRAM capacity of the UNPU approach is approximately four times that of the proposed CNN engine (256 instead of 67 kB), which resulted in reduced off-chip accesses. Thus, the proposed COMPAC CNN engine is much more energy-efficient in terms of off-chip data access energy compared with the Eyeriss [4] and UNPU [17] design approaches under iso-bit width and iso/smaller on-chip SRAM capacity conditions. However, it should be noted that the proposed COMPAC CNN engine design has been optimized for AlexNet, whereas Eyeriss [4] and UNPU [17] designs are more flexible in terms of supporting various CNNs (and even RNNs for UNPU).

## V. CONCLUSION

In this article, a COMPAC CNN engine for energy-efficient edge AI computing is demonstrated in 65-nm CMOS technology. Four major ideas are proposed in this work, which results in better performance, accuracy, and overall energy-efficiency (see Table VIII). The proposed COMPAC CNN engine supports TRS in the MDL to perform an energy-efficient multi-bit input and weight MAC operation in the time domain while still achieving high classification accuracy. A CTD approach is proposed and deployed in COMPAC to improve throughput in time encoding of the input activation. The simulation results on the AlexNet CNN over 1000 ImageNet images show a significant throughput improvement, consuming on, an average, 14.71 and 7.15 input clock cycles to time-encode an 8-bit input activation in two different CTD modes (see Fig. 9). The PAC technique is proposed to reduce the number of MACs. The simulation results on the AlexNet CNN over 1000 ImageNet images show up to 31.47% (21.79%) reduction in the number of non-zero input activations MACs with a top-five classification accuracy loss of 0.60% (0.90%), top-one classification accuracy loss of 2.20% (1.90%), and an on-chip access overhead of 60.53% (8.03%) for the PAC modes 1 (2), respectively (see Fig. 12 and Table III). The tradeoffs between classification accuracy, reduction in the number of MACs, and on-chip access need to be carefully comprehended to achieve optimal overall energy savings. An EEDF for optimal on-/off-chip memory accesses is also discussed in this article. COMPAC data flow results in 86.97% reduced on-chip SRAM accesses and 29.74% reduced off-chip DRAM accesses compared with an Eyeriss [4], considering iso-8-bit precision. The 65-nm CMOS test chip implementing the AlexNet CNN achieved an energy efficiency of 1.044 TOPS/W and a throughput of 0.1278 GOPS at 720 mV while operating at 25 MHz. The top-five classification accuracy of 76.90% measured over 1000 ImageNet images and 77.15% by simulating over 50 000 ImageNet images is achieved with the proposed COMPAC approach. The simulation results taken into account MDL circuit non-idealities over 50 000 ImageNet validation set images show a classification accuracy loss within 1% compared with the 8-bit fixed-point software implementation.

## REFERENCES

[1] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[2] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1701–1708.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[4] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[5] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, Apr. 2017.

[6] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5687–5695.

[7] M. Horowitz, "Computing's energy problem (and what we can do about it)," *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2014, pp. 10–14.

[8] 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 *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2019, pp. 228–230.

[9] A. Sayal, S. S. T. Nibhanupudi, S. Fathima, and J. P. Kulkarni, "A 12.08-TOPS/W all-digital time-domain CNN engine using bi-directional memory delay lines for energy efficient edge computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 60–75, Jan. 2020.

[10] D. Stursa and P. Dolezel, "Comparison of ReLU and linear saturated activation functions in neural network for universal approximation," in *Proc. 22nd Int. Conf. Process Control (PC19)*, Štrbské Pleso, Slovakia, 2019, pp. 146–151.

[11] J. Zhou, W. Xu, and R. Chellali, "Analysing the effects of pooling combinations on invariance to position and deformation in convolutional neural networks," in *Proc. IEEE Int. Conf. Cyborg Bionic Syst. (CBS)*, Beijing, China, Oct. 2017, pp. 226–230.

[12] C. Lee, P. Gallagher, and Z. Tu, "Generalizing pooling functions in CNNs: Mixed, gated, and tree," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 863–875, Apr. 2018.

[13] J. Sim *et al.*, "A 1.42 TOPS/W deep convolutional neural network recognition processor for intelligent IOE systems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan./Feb. 2016, pp. 264–265.

[14] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 246–247.

[15] Z. Yuan *et al.*, "Sticker: A 0.41-62.1 TOPS/W 8Bit neural network processor with multi-sparsity compatible convolution arrays and online tuning acceleration for fully connected layers," in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, Jun. 2018, pp. 33–34.

[16] J. Zhang, C. Lee, C. Liu, Y. S. Shao, S. W. Keckler, and Z. Zhang, "SNAP: A 1.67–21.55 TOPS/W sparse neural acceleration processor for unstructured sparse deep neural network inference in 16nm CMOS," in *Proc. Symp. VLSI Circuits*, Kyoto, Japan, Jun. 2019, pp. C306–C307.

[17] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.

[18] K. Ueyoshi *et al.*, "QUEST: multi-purpose log-quantized DNN inference engine stacked on 96-MB 3-D SRAM using inductive coupling technology in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 186–196, Jan. 2019.

[19] 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 *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 490–492.

[20] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, Apr. 2017.

[21] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 488–490.

[22] 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. IEEE Custom Integr. Circuits Conf. (CICC)*, Austin, TX, USA, Apr. 2017, pp. 1–4.

[23] 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 *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 124–126.

[24] D. Miyashita, S. Kousai, T. Suzuki, and J. Deguchi, "Time-domain neural network: A 48.5 TSOp/s/W neuromorphic chip optimized for deep learning and CMOS technology," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2016, pp. 25–28.

[25] R. J. D'Angelo and S. R. Sonkusale, "A time-mode translinear principle for nonlinear analog computation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 9, pp. 2187–2195, Sep. 2015.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[27] 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," *IEEE J. Solid-State Circuits*, vol. 49, no. 4, pp. 1007–1016, Apr. 2014.

[28] M. Kim and J.-S. Seo, "Deep convolutional neural network accelerator featuring conditional computing and low external memory access," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Boston, MA, USA, Mar. 2020, pp. 1–4.

[29] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*. [Online]. Available: http://arxiv.org/abs/1603.04467

[30] Xilinx. *Xilinx Virtex-7 FPGA VC707 Evaluation Kit*. Accessed: Dec. 2, 2020. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html

[31] *Xilinx FMC XM105 Debug Card*. Accessed: Dec. 2, 2020. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/hw-fmc-xm105-g.html


**Aseem Sayal** (Member, IEEE) received the bachelor's degree in electrical and electronics engineering from Delhi Technological University (formerly Delhi College of Engineering), Delhi, India, in 2013, and the M.S. and Ph.D. degrees in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA, in 2017 and 2020, respectively.

From 2013 to 2015, he worked as a Physical Design CAD Engineer at Qualcomm India Pvt. Ltd., India. He was also an intern at Apple Inc., Cupertino, CA, USA, and Google LLC, Sunnyvale, CA, USA, where he was involved in developing clock tree simulation methodology and average power estimation solutions for long-running arbitrary workloads, respectively. He is currently working as an ML Design Engineer with the Platforms Datacenter Chip Team, Google LLC. He has filed five patents and published 15 articles in referred journals and conferences. His graduate research was focused on designing energy-efficient hardware accelerators for machine learning applications and developing EDA design methodologies for heterogeneously integrated secured ASICs.

Dr. Sayal was a recipient of the Chancellor Gold Medal and Meritorious Student Award from Delhi Technological University in 2013.

**Shirin Fathima** received the bachelor's degree in electronics and communication engineering from the Birla Institute of Technology and Science, Pilani (BITS, Pilani), Pilani, India, in 2017, and the M.S. degree in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA, in 2019.

She worked on designing energy-efficient hardware accelerators for image classification in her master's thesis project. Currently, she is working at Apple Inc., Cupertino, CA, USA, as an SoC Physical Design Timing Engineer.

**SS Teja Nibhanupudi** (Student Member, IEEE) received the bachelor's degree in electrical engineering from IIT Gandhinagar, Gandhinagar, India, in 2016. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA.

From 2016 to 2017, he worked as a Project Associate at IIT Gandhinagar, where he was involved in the development of high-voltage process flow for Semi-Conductor Laboratory, Ajitgarh, India. His research is focused on circuit-device co-optimization using novel materials, such as RRAM, IMT, and STTRAM.

**Jaydeep P. Kulkarni** (Senior Member, IEEE) received the B.E. degree from the University of Pune, Pune, India, in 2002, the M.Tech. degree from the Indian Institute of Science (IISc), Bengaluru, India, in 2004, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 2009, all in electronics/electrical engineering.

From 2009 to 2017, he was with Intel Circuit Research Lab, Hillsboro, OR, USA, where he worked on energy-efficient integrated circuit technologies. He is currently an Assistant Professor in electrical and computer engineering at The University of Texas at Austin, Austin, TX, USA, and a fellow of the AMD Endowed Chair in Computer Engineering and a fellow of the Silicon Labs Chair in Electrical Engineering. He has filed 35 patents and published 85 articles in referred journals and conferences. His research is focused on machine learning hardware accelerators, in-memory computing, emerging nano-devices, hardware security, heterogeneous/3-D integration, and cryogenic computing.

Dr. Kulkarni is a member of the ACM. He received the Best M.Tech. Student Award from IISc, the Intel Foundation Ph.D. Fellowship Award, the SRC Best Paper and Inventor Recognition Awards, the Purdue Outstanding Doctoral Dissertation Award, seven Intel Divisional Recognition Awards, the 2015 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS Best Paper Award, the SRC Outstanding Industrial Liaison Award, Micron foundation faculty awards, and the Intel Rising Star Faculty Award. He has served as the Conference General Co-Chair for 2018 ISLPED, and is currently participating in the technical program committees of CICC, ICCAD, DAC, and AICAS conferences. He is currently serving as the Chair for the IEEE Central Texas SSCS/CAS Joint Chapter. He is also serving as an Associate Editor for the IEEE SOLID STATE CIRCUIT LETTERS and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.