Universal Cyclic Sparsely Connected Layers for Compact Convolutional Neural Network Design

Abstract—Model size and computation complexity of deep convolutional neural networks (DCNNs) are two major factors governing their throughput and energy efficiency when deployed to hardware for inference. Recent works on compact DCNNs along with pruning methods are effective, yet with drawbacks. For instance, more than half the size of all MobileNet models lies in their last two layers, mainly because compact separable convolution (CONV) layers are not applicable to their last fully-connected (FC) layers. Also, in pruning methods the compression is gained at the expense of irregularity in the DCNN architecture, which necessitates additional indexing memory to address non-zero weights, thereby increasing memory footprint, decompression delays, and energy consumption. In this paper, we propose cyclic sparsely connected (CSC) architectures, with a memory/computation complexity of $O(N \log N)$ where $N$ is the number of nodes/ channels given a DCNN layer, that, contrary to compact depthwise separable layers, can be used as an overlay for both FC and CONV layers of $O(N^2)$. Also, contrary to pruning methods, CSC architectures are structurally sparse and require no indexing due to their cyclic nature. We show that both standard convolution and depthwise convolution layers are special cases of the CSC layers and whose mathematical function, along with FC layers, can be unified into one single formulation, and whose implementation can be carried out under one arithmetic logic component. We examine the efficacy of the CSC architectures for compression of LeNet and MobileNet models with precision ranging from 2 to 32 bits. Lastly, we design a configurable application-specific hardware that implements all types of DCNN layers including FC, CONV, depthwise, CSC-FC, and CSC-CONV indistinguishably within a unified pipeline and with negligible performance stall. We configure the hardware with 16 processing engines (PEs) and 12 multiply-accumulate (MAC) units per PE for the deployment of the compressed 8-bit CSC-MobileNet-192. Compared to the state of the art, our implementation for ImageNet classification is 1.5× more energy efficient on FPGA.

Keywords—Compressed DCNNs, Cyclic Sparsely Connected Architectures, Neural Network Accelerator, ASIC, FPGA

I. INTRODUCTION

Machine learning today can surpass human-level accuracy in tasks such as keyword-spotting [1], speech/image recognition [2][3], health monitoring [4], and predictions [5], to name a few. In more recent systems, such tasks use artificial neural networks which accepts an input data (e.g. an RGB image) and computes output results (e.g. an evaluated object in an image). Neural networks encompass a wide range of types including deep neural networks (DNNs)$^1$, deep convolutional neural networks (DCNNs), recurrent neural networks, and their combinations and variations. Recently, significant effort has gone towards proposing DCNN models that yield higher classification accuracy and/or less computation for complex datasets.

$^1$Following notation in [1], we refer to a multilayer network as a DNN if all layers are of type FC, otherwise as a DCNN if CONV layers are included.

DCNNs are made of a few layers formed in a directed acyclic graph (DAG) [6]. Typically, the layers are of type convolution (CONV), fully-connected (FC), non-linear activation function, batch-normalization, and max-pool, to name a few. Computation in a DCNN is attributed to an extensive number of multiply-accumulate (MAC) operations, and the model size is attributed to the number of parameters (times their quantization level), both of which are commonly dominated by CONV and FC layers. Modern DCNNs trained on elaborate datasets suffer from large model size and high computation that makes their deployment on resource-bound hardware platforms a challenge.

In this paper, we propose cyclic sparsely connected (CSC) architectures as a factorized overlay for all standard DCNN layers that, compared to standard DCNN layers, can potentially reduce both computation and size of a layer from $O(N^2)$ down to $O(N \log N)$ where $N$ is the number of channels/nodes given a layer. We formulate CSC architectures for both their formation and their operation when being used as a DCNN layer, and evaluate their compression and their impact on accuracy. Furthermore, we show that both standard CONV layers as well as depthwise separable CONV layers [3] can be considered as special cases of CSC layers. Lastly, We design a hardware to implement DCNNs from a CSC perspective and show that the hardware can effectively implement CSC DCNNs as well as standard DCNNs and depthwise separable DCNNs. The main contributions of this paper include:

- Propose two types of cyclic sparsely connected (CSC) architectures with reduced computational complexity and two schemes for CSC-CONV layers as substitutes for FC and CONV layers.
- An algorithm to train CSC DCNNs using a bottom-up approach to trade off between compression and accuracy.
- Train LeNet300-100 & MobileNet-224 and -192 using CSC architectures, with compression of 20×, 6.7×, 7.3×, and 1.5× respectively as compared to their baselines within a margin of 1.5% accuracy loss.
- Propose a method of double compression by adopting CSC architectures in tandem with extreme quantization: a 2-bit LeNet-300-100 and an 8-bit 0.5 MobileNet V1 are compressed by 320× and 6× respectively compared to their 32-bit baselines with negligible accuracy loss.
- Propose a scalable hardware, configured with 192 MAC units and 8-bit dataflow, which implements a compressed 8-bit 0.5 CSC-MobileNet-192 V1 for inference on a tiny Xilinx FPGA.

II. MOTIVATION

The memory size and the required number of operations generally govern the energy consumption of a DCNN on
B. Low bitwidth Neural Networks

When quantization and pruning methods are used in tandem, unlike the DCNN weights and activations, the extra indexing memory imposed by the pruning method is not quantizable, and therefore the extra indexing memory might be intolerable in resource-bound platforms that employ low bitwidth such as binary [8] or ternary [9] weight neural networks. Such neural networks are unyielding to pruning methods, and if pruned, their compressed formats can result in a larger model size as compared to their un-compressed models. Our second goal in this work is to develop and employ structurally compact models that require no indexing for low precision neural networks. Fig. 2 plots the memory vs. the number of operations for the same DNN with 4 of its compressed variants, highlighting the significant compression gained by combining structured sparsity and extreme quantization.

III. CYCLIC SPARSE CONNECTIONS

In this Section, we first introduce and formulate CSC architectures from a graph theory perspective. Throughout this work we use italic lower case letter $x$ to represent generator polynomials (e.g. $p(x) = 1 + x + x^2$), italic capital letters (e.g. $N$) for integer values, bold upper case letters for matrices (e.g. $W$) and vectors (e.g. $X$), and calligraphic uppercase letters for matrix of matrices to represent tensors (e.g. $F$). We use italic lower case letters in brackets for the elements of a matrix or a vector (e.g. $W[i,j]$ or $X[i]$) and parenthesis for matrix elements of a tensor (e.g. $F[h,w](i,j)$). Thus, the parameters of a 3D tensor such as a feature map data or a 4D tensor such as a filter set can be represented with ensemble of brackets and parenthesis (e.g. $X[h,w](i,j)$ or $F[h,w](i,j)$).

A. Problem Statement and Formulation

Concluded from the related works [3][10][11][12][13][14] and more specifically inspired by the structures of Butterfly networks in fast Fourier transform [15], we observe that in their graphs, the degree (number of connected edges) of every node/channel within the same layer are equal and there exists an equal number of paths connecting each Input node to every Output node. The objective of this Section is hence to formulate a directed acyclic graph composed of a few layers where all Input nodes are connected via an equal number of paths to all Output nodes of the graph. The graph is composed of: an Input layer, $L - 1$ layers in between referred to as factorized layers, and an Output layer, each layer sized $N$. We denote the first (Input) and the last (Output) layers by $L_1$ and $L_{L-1}$ respectively, highlighting the advantage of structural sparsity in tandem with extreme quantization.

A. Pruning vs. Structured Sparsity

Despite their effectiveness, fine-grained pruning methods have a drawback that they result in the irregularity of the pattern of non-zero weights in the pruned model that necessitates an additional indexing. Thus, their compressed model has more parameters than solely the amount of non-zero weights, and their implementation is deteriorated by the model decompression. Structured sparsity, on the other hand, eliminates indexing and can be handled as high-performance as dense matrices/layers. Fig. 1 reflects our first motivation by illustrating the computation complexity of a LeNet-300-100 DNN compressed with either pruning or structurally sparse method used in this work.
We also define a connectivity metric $C$ that defines the number of paths that connect each arbitrary Input node to every arbitrary Output node from the graph. Thus, for a heterogeneous graph:

$$\prod_{l=0}^{L-1} F_l = NC. \quad (2)$$

As a result, in homogeneous graphs $E = NFL$ and $F^L = NC$. Mixing Eqs. (1) and (2), we infer that the number of edges $E$ is minimum given $\sum_{i=0}^{N} F_i = e$, where $e$ is the Napier’s constant, meaning that if the average fan-out of individual layers is equal to Napier’s constant, then $E_{\text{min}} = Neln(NC)$.

The objective of homogeneity is to provide a basis where every arbitrary node in the graph is equally exploited, every arbitrary pair of nodes from Input and Output layers are equally connected. The information flux through the factorized layers is fairly equal, and the rank of the transforming weighted matrix can remain full. This foundation will be proposed as an overlay for FC and CONV layers, and is defined such that both standard FC and CONV layers are special cases of which, given $F = N$, $L = 1$, and $C = 1$. Combining Eqs. (2) and (1) for a homogeneous graph, a logarithmic relationship between number of edges and size of the layers is inferred:

$$E_{\text{homo}} = NFL\log_e(NC), \quad (3)$$

that reflects the compression it provides as compared to a fully-connected graph with $N^2$ synapses. $E$ is the number of edges that counts the non-zero elements of the $L$ layers, and governs the number of MAC operations provided the total weighted graph operates on an input vector. As an example, given $F = 2$ and $C = 1$, Eqs. (2) and (3) infer that $L = log_2 N$ and $E = 2Nlog_2 N$ which is the case in radix-2 butterfly networks for fast Fourier transform [15].

B. Solution to the Problem Statement

The butterfly graphs used in the FFT give one set of solutions. We show that circulant matrices generated from generator polynomials as in cyclic codes give another set of solutions for the bi-adjacency matrices of the layers which satisfy all the requisites in our problem statement. From here on, we call these graphs cyclic sparsely connected (CSC) architectures. Suppose the bi-adjacency matrix $A_l$ ($0 \leq l \leq L-1$) of every factorized layer $l$ in our CSC graph has a generator polynomial $p_l(x)$ that has $F$ terms which generates a cyclic bi-adjacency matrix of block length $N$. It can be shown that the product of the $L$ bi-adjacency matrices attributed to the $L$ layers is a matrix $A_T$ where $A_T(i, j)$ represents the number of paths between the Input node $i$ and Output node $j$ of the CSC graph. In the problem statement, the connectivity should be equal to $C$ for all arbitrary pairs of Input and Output nodes, and thus, $A_T = C I$, where $I$ is an $N$-by-$N$ all-one matrix. The all-$C$ matrix $A_T$ can be attributed to another generator polynomial $p_T(x) = C \sum_{i=0}^{N-1} x^i$. The generator polynomial constructs a cyclic matrix as follows: the first row of the matrix corresponds to the coefficients of the polynomial—represented as big-endian in this paper—and then, every next row is a cyclic right shift of its previous row. We provide two different factorization sets of $p_T(x)$:

1) CSCI: Connectivity Equal to One:

If $C = 1$ and $F$, $L$ and $N$ are such that $N = F^L$, then $p_T(x) = \sum_{i=0}^{N-1} x^i$ can be factorized as follows:

$$\begin{cases} \sum_{i=0}^{N-1} x^i = \prod_{l=0}^{L-1} p_l(x) \\ p_l(x) = \sum_{i=0}^{F-1} x^{D_{l,i}}, D_0 = 1 \\ p_l(x) = \sum_{i=0}^{F-1} x^{D_{l,i}}, D_1 = \frac{F}{e} \end{cases} \quad (4)$$

By assigning $p_l(x)$ as the generator polynomial of factorized layer $l$, the CSC layers of the graph are completely defined. $D_l$ is the dilation parameter that indicates the distance between elements of value 1 in the first row of the bi-adjacency matrix of layer $l$. It also represents the dilation between the fanned-out edges in its corresponding layer. For small values of $F$ (e.g. 2 or 3), and consequently more number of layers (e.g. $\log_2 N$ or $\log_3 N$), the least number of synapses is resulted, but at the expense of elongating the graph that, by itself, will increase the memory communication during hardware implementation.

2) CSCII: Layers Equal to Two:

If $L = 2$ and $F$, $C$ and $N$ are integers to satisfy $NC = F^2$, then $p_T(x) = C \sum_{i=0}^{N-1} x^i$ can be factorized as follows:

$$\begin{cases} C \sum_{i=0}^{N-1} x^i = p_1(x)p_0(x) \mod(x^N-1) \\ p_0(x) = \sum_{i=0}^{F-1} x^{D_{0,i}}, D_0 = 1 \\ p_1(x) = \sum_{i=0}^{F-1} x^{D_{1,i}}, D_1 = \frac{F}{e}. \end{cases} \quad (5)$$

By assigning $p_0(x)$ and $p_1(x)$ to the first and second layers respectively, the CSCII graph is defined. In CSCII graphs, $\gamma = \frac{N}{NC}$ that declares compression given $C < \frac{N}{2}$ and $E = 2N\sqrt{NC}$ that declares a computation of $O(N \sqrt{N})$.

For both CSCI and CSCII, the bi-adjacency matrix of a layer with size $N$, fan-out $F$, and a dilation $D_l$ is inferred as:

$$A_l(i, j) = \begin{cases} 1 & \text{if } (i-j+kD_l) \mod N = 0, \forall k = 0, 1, \ldots, (F-1) \\ 0 & \text{otherwise}. \end{cases} \quad (6)$$

IV. CSC Architectures in DCNNs

In this Section, we expand on adopting CSC architectures to compress the size and computation of FC and CONV layers in DCNNs. We then explain a bottom-up algorithm to seek appropriate CSC architectures. For simplicity, we ignore the term bias in all of the equations, assume that convolution operations are of same size, and the number of nodes/channels in I/O of the FC/CONV are all equal to $N$. As a result, vectors/matrices/tensors in this Section are $Y \in \mathbb{R}^N$, $W \in \mathbb{R}^{N \times N}$, $X \in \mathbb{R}^{N \times N}$, $\gamma \in \mathbb{R}^{W_y \times W_y \times N}$, $\Gamma \in \mathbb{R}^{W_f \times W_f \times N}$, and $X \in \mathbb{R}^{W_x \times W_x \times N}$, where $W_y$, $W_f$ and $W_x$ as well as $H_y$, $H_f$, and $H_x$ represent width and height in channels of tensors $\gamma$, $\Gamma$, and $X$ respectively. If layers are not equally sized, we adjust their I/O according to the note in Subsection III-A-2.

A. Fully-Connected Layers

A standard FC layer is representable with a dense weight matrix $W$ whose operation on an input vector $X$ is equivalent to a matrix-vector multiplication that generates a vector $Y$, s.t.: 

$$Y[j] = (WX)[j] = \sum_{i=0}^{N-1} W[i,j]X[j], \quad (7)$$

where $j \in \{0, 1, \ldots, N-1\}$.
which has a computation of $O(N^2)$. Now, if the $W$ is factorizable into $L$ matrices, or correspondingly, if a cascade of $L$ CSC layers with linear activations and with parameters $N$ and $F$ are weighted to equate a factorizable fully-connected layer, then the equivalent $W_T = \prod_{l=0}^{L-1} W_l$. Because of associative property of matrix-matrix multiplication, the computation of $W_TX$ can start from the rightmost matrix-vector multiplication and propagate to the leftmost matrix. As such, every individual operation between layer $l$ with $W_l$ and input $X_l$ results in:

$$Y_l[i] = \sum_{j=0}^{F-1} W_l[i, (i+jD_l) \mod N]X_l[j]$$  \hspace{1cm} (8)

that inherently skips the zero values in $W_l$ by taking only the non-zero values that are diluted $D_l$ elements apart. Thus, the computation is of $O(NF)$ for one single $WX$, and of $O(NF\log(NC))$ for $W_TX = (\prod_{l=0}^{L-1} W_l)X$. Now, if $W_l$ is rearranged in its compressed format $\hat{W}_l$ that contains only $NF$ non-zero entries of $W_l$, where:

$$\hat{W}_l[i, j] = W_l[i, (i+jD_l) \mod N]$$  \hspace{1cm} (9)

then Eqn. (8) can be altered as:

$$Y_l[i] = \sum_{j=0}^{F-1} \hat{W}_l[i, j]X_l[(i+jD_l) \mod N]$$  \hspace{1cm} (10)

that we refer to as a cyclic dilated matrix-vector multiplication for a CSC-FC layer. For $F = N$ and $D = 1$, and given a new re-arrangement of the dense $W_l$ into a compressed format $\hat{W}_l$, Eqn. (10) corresponds to Eqn. (7), revealing a novel approach to computation in standard matrix-vector multiplications using cyclic dilated matrix-vector multiplication.

B. Convolution Layers

With a little abuse of notation, we use the star symbol ‘$*$’ to denote the convolution operation, which is practically a correlation operation in neural networks. A 2D convolution between a 4D macro-matrix $F$ and a 3D macro-vector $X$ results in a 3D macro-vector $Y$. Similar to Eqn. (8), $Y = F*X$ can be broken into summation over partial convolutions between aligned 2D channels from $F$ and $X$, i.e.:

$$Y_l(i, j) = \sum_{v=0}^{N-1} F_l(i, j) * X_l(i, j)$$  \hspace{1cm} (11)

such that:

$$(F_l(i, j) * X_l(i, j))[u, v] = \sum_{w=0}^{W_l-1} \sum_{h=0}^{H_l-1} F_l(i, j)[w, h]X_l[(i+w, j+h)]$$  \hspace{1cm} (12)

The number of parameters of $F_l$ is $W_lH_lN^2$ and the computation of $F_l*X_l$ is of $O(W_lH_lW_gH_gN^2)$ and of $O(W_lH_lN^2)$ for a standard and a pointwise ($W_l = H_l = 1$) CONV2D respectively. If a CONV layer is trained on a CSC layer, then:

$$Y_l(i, j) = \sum_{v=0}^{N-1} \hat{F}_l(i, j) * \hat{X}_l(i, j)$$  \hspace{1cm} (13)

that performs zero-skipping by computing only over non-zero channels in $\hat{F}_l$ that are $D_l$ channels apart. Thus, the number of parameters are $W_lH_lN^2$ and the computation is reduced to $O(W_lH_lW_gH_gN^2)$ for one single $F_l * X$. Given $L$ homogeneous CSC layers with size $N$, fan-out $F$, and connectivity $C$ that on every synapse of which $1$-by-$1$ kernels are laid, the computation of $\{F_l * X\}_{l=0}^{L-1}$ is of $O(W_gH_gN\log(NC))$. Now, if $Y_l$ is rearranged in its compressed format $\hat{Y}_l$, where:

$$\hat{F}_l(i, j) = F_l(i, (i+jD_l) \mod N)$$  \hspace{1cm} (14)

then computation of $Y_l$ in Eqn. (13) can be reformulated:

$$Y_l(i, j) = \sum_{v=0}^{N-1} \hat{F}_l(i, j) * \hat{X}_l[(i+jD_l) \mod N]$$  \hspace{1cm} (15)

Plugging Eqn. (15) in Eqn. (12) results in a CSC CONV layer output, with argument $D_l$ as dilation, between a compressed 4D filter tensor $\hat{F}$ of shape $W_gH_g-by-N-by-F$ and a 3D input $X$ of shape $W_g-by-H_g-by-N$ as follows:

$$Y_l[i, j][u, v] = \sum_{v=0}^{N-1} \hat{F}_l(i, j)[u, v]X_l[(i+jD_l) \mod N][u+w, v+h]$$  \hspace{1cm} (16)
to which, similar to Eqn. (10), we refer as a cyclic channel dilated 2D convolution for a CSC-CONV layer. A pseudo code to implement the Eqn. (16) is illustrated in Fig. 4. For \( F = N \) and \( D = 1 \), both Eqns. (13) through (16) convert to a standard 2D convolution as in Eqns. (11) and (12). For \( F = 1 \) and \( D = 0 \) they convert into a depthwise convolution, highlighting the application span of convolution layers from the perspective of CSC-CONV.

We define two schemes for CONV layers factorized on CSC architecture as depicted in Fig. 3.

C. Training Algorithm

The DCNN model with FC/CONV layers is trained first and a reference accuracy \( \lambda_{FC/CONV} \) is obtained. Then, the FC/CONV is replaced with an adjusted CSC-FC/CONV, starting from the most compressed CSC and directed toward the compression reduction. In each experiment, if the accuracy \( \lambda_{CSC} \) is less than \( \lambda_{FC/CONV} \), the procedure is terminated, and the CSC model is accepted. The criteria \( \epsilon \) is chosen to be 2% in all experiments of this paper. To circumvent implementing CSC layers in software for DCNNs, we seek within their open-source libraries where the forward path snippets \( \mathbf{Y} = \mathbf{W} \mathbf{X} + \mathbf{B} \) for FC layers and \( \mathbf{Y} = \mathbf{F} \ast \mathbf{X} + \mathbf{B} \) for CONV layers are implemented, and replace them with \( \mathbf{Y} = (\mathbf{A} \circ \mathbf{W}) \mathbf{X} + \mathbf{B} \) and \( \mathbf{Y} = (\mathbf{A} \circ \mathbf{F}) \mathbf{X} + \mathbf{B} \) respectively, where \( \circ \) is an element-wise multiply operator, \( \mathbf{Y} \) (and \( \mathbf{Y} \)) is the output feature map, \( \mathbf{A} \) (and \( \mathbf{A} \)) with expanded dimensions, inferred from Eqn. (6), is the bi-adjacency matrix of the corresponding CSC, \( \mathbf{W} \) is the weight matrix, \( \mathbf{F} \) is the filter tensor, \( \mathbf{X} \) (and \( \mathbf{X} \)) is the input feature map, and \( \mathbf{B} \) is the bias vector. By doing so, in each forward path of the training, the elements of the \( \mathbf{W} \) (and \( \mathbf{F} \)) are enforced to conform to those of the \( \mathbf{A} \) (and \( \mathbf{A} \)) matrix. Finally, the deformed \( \mathbf{W} \) and \( \mathbf{F} \) will be rearranged in their compressed formats \( \mathbf{W} \) and \( \mathbf{F} \) respectively according to Eqns. (9) and (14) to be stored in compact and deployed onto the hardware.

V. EXPERIMENTS AND RESULTS

We used CSC architectures to evaluate their compression impact for popular DCNNs including LeNet-300-100 [6] and MobileNet [3] on benchmarking datasets MNIST [16] and ImageNet [17]. To follow the bottom-up training procedure, for each experiment, one of the two CSC architectures and one of the two proposed CSC-CONV Schemes from Section IV is selected and only one CSC hyper-param is tweaked per experiment.

1) LeNet-300-100: for LeNet-300-100, which is a 3 layer DNN, we replaced the first two FC layers with CSCI architectures and, taken \( F = 2 \), increased the \( L \) (and consequently \( N \)) in every incremental experiment until the accuracy loss of less than 2% was met. In another set of experiments, we further compressed the LeNet-300-100 by ternarizing its weights. Fig. 5 shows the bottom-up training procedure by tweaking the parameter \( L \) for each CSCI. Table I summarizes the selected compressed 32-bit and 2-bit models and compares them with their baseline and the related work.

2) MobileNet: MobileNet is a compact model already, and the depthwise/pointwise layers are special cases of heterogeneous CSC architectures. However, in all of MobileNet's variations, more than 50% of the model size fall under the last two layers, i.e. a pointwise CONV, and an FC layer. In this experiment, we replaced the last two layers of MobileNets-224 and -192 with CSCII architectures and determined \( C = 16 \) for each CSCII architectures. To facilitate our experiments, we used pre-trained 8-bit TFlite models and used the method of transfer learning to fine-tune the last two compressed layers. Fig. 6 plots the number of parameters vs. number of
operations in the 8-bit quantized models of 0.5, 0.75 and 1.0 MobileNet-224 V1 (blue circle markers), and -192 (red circle markers), and their CSCI compressed counterparts (triangle markers) highlighted with their top-5 accuracy for ImageNet classification. Table II summarizes the experiment for 0.5 MobileNet: the compressed quantized model of 0.5 MobileNet is 6× and 1.5× smaller than its float-32 and 8-bit baselines.

VI. ACCELERATOR ARCHITECTURE

In Section IV, we showed that DCNN’s major layers including standard FC, standard CONV, depthwise separable CONV, as well as CSC-FC and CSC-CONV layers are all special cases of CSC layers, and can be recognized solely by their weights in compressed formats (\(W\) or \(\hat{W}\)) and hyper-parameters \(N\) (and \(N_f\) if adjusted), \(F, D, W_F\) and \(H_F\), and their computation are all governed by Eqn. (15). The computation in Eqn. (15) with a pseudo code illustrated in Fig. 4 for a CSC-CONV can be highly optimized using various tiling schemes for any parallel computing machine. Motivated by works of Han et al. [18][19] and Zhang et al. [1] that emphasize the importance of compressing DCNNs to implement near SRAM processing, we pursue designing of a hardware for ASIC and FPGA that is independent of a DRAM, and that integrates all the processing components along with sufficient on-chip SRAM memory that stores the whole DCNN model and its intermediate feature map during every inference.

The peak performance in a well-engineered accelerator with \(PE\) processing engines and \(MAC_{perPE}\) multiply-accumulate units per PE with a clock rate \(freq\) can potentially approach \(2 \times PE \times MAC_{perPE} \times freq\). According to the roofline model, this peak performance is achievable if and only if the memory bandwidth is high enough to minimize stalls and continuously feed the computing components in PEs with required data/operands. Our proposed hardware, depicted in Fig. 7 and described in Verilog HDL, to compute DCNNs with or without CSC layers is configurable with number of PEs, number of MAC units per PE, and the data-flow bitwidth to execute inference of a DCNN model trained with the same quantization level as the bitwidth. It comprises three main blocks, including an array of PEs that perform MAC operations and accommodate a DCNN weight model partitioned in weight memory units, a partitioned input memory that stores intermediate input feature map (ifmap) data, and a CSC-base router that links and maximizes the bandwidth between array of PEs and array of sub-banks in the ifmap memory. Each PE also incorporates another SRAM memory that stores output feature map (ofmap) data in partitions, later to be swapped with the ifmap memory to drive a next layer in the DCNN. For simplicity, the state machine and pipelines that perform batch-normalization, max-pool, quantization, and non-linear activations are not shown.

We configure this hardware to implement the 0.5 CSC-MobileNet-192. In Section VII, we implement the configuration on FPGA and evaluate the energy and power advantages gained by implementing the compressed model onto a hardware that utilizes less on-chip SRAM resources.
VII. HARDWARE IMPLEMENTATION

The hardware has been configured as per Fig. 7. The DCNN take in an input image of size 111 KB (=192×192×3), and have a largest feature map of size 590 KB (=96×96×64) that generates an output feature map of 295 KB (=48×48×128). The size of the DCNN model according to Fig. 6 and Table II is 873 KB. Therefore, having partitioned the model into 16 Weight SRAM sub-banks, the hardware requires 55 KB of weight memory per PE respectively. We take 57 KB for the weight sub-banks to avail them with enough space for storing quantization offset and scaling parameters along with DCNN weights and hyper-parameters. Table III lists the configuration of the hardware that requires approximately 2 MB on-chip memory to implement ImageNet with top-5 classification accuracy of 81.1%. Fig. 8 plots the number of computation and communication cycles per layer implementation of the 0.5 CSC-MobileNet-192 as workload for latter hardware config.

A. Field-Programmable Gate Array (FPGA)

The hardware configuration for the compressed model was implemented on the Artix XC7A200T FPGA from the AC701 evaluation board using Xilinx Vivado. The Artix-7 FPGA has 1.6 MB of BRAMs and 0.4 MB of distributed LUT RAMs that suffice the total on-chip SRAM requested for implementing the hardware configuration that accommodates the compressed MobileNet, whereas the other uncompressed MobileNet requiring hardware with 2.4 MB memory can not fully fit in the on-chip RAMs of the FPGA. Fig. 9 shows the power dissipation breakdown and the resource utilization for the hardware with 16 PEs and 12 MAC units per PE on the evaluation board, highlighting the full utilization of the BRAMs, and dynamic power dissipation of 90%. The clock frequency of this implementation is set to 100 MHz. More information for this implementation is reported in the Table IV (last column).

VIII. COMPARISON

In this Section, we compare our hardware implementation with those of the state of the art that seek energy efficiency for the same task: AlexNet-level accuracy for ImageNet classification. In FPGA, our implementation is 1.5× more energy efficient as compared to the implementation of Zhu et. al. [22]. Similarly, the main reason of such outperformance is the usage of a small DCNN that fits in a tiny FPGA (Artix-7) which is significantly smaller and an order of magnitude as less power dissipating as the FPGAs used in the related work, since The on-chip BRAMs and resources of the Artix-7 are sufficient for full implementation of the 0.5 CSC-MobileNet-192 according to configuration in Table III.

IX. CONCLUSION

To address the general issues with pruning methods and compact model architectures, we introduced cyclic sparsely connected (CSC) architectures with a memory/computation complexity of \(O(N \log N)\) that can be used as an overlay for both FC and CONV layers of \(O(N^2)\), where \(N\) is the number of nodes/channels given a layer. Furthermore, our proposed solution is structurally sparse and requires no indexing for its cyclic nature. The compressed 0.5 CSC-MobileNet-192 framework can be easily deployed to low power Artix-7 FPGA for full on-chip processing due to the small memory requirement. Compared to state of the art, our implementations are 2.1× and 1.5× superior in terms of energy efficiency for the AlexNet-level accuracy of ImageNet classification when directed for FPGA.
References


