Power, Performance, and Area Benefit of Monolithic 3D ICs for On-Chip Deep Neural Networks Targeting Speech Recognition

KYUNGWOOK CHANG, Georgia Institute of Technology, USA
DEEPAK KADETOTAD, YU CAO, and JAE-SUN SEO, Arizona State University, USA
SUNG KYU LIM, Georgia Institute of Technology, USA

In recent years, deep learning has become widespread for various real-world recognition tasks. In addition to recognition accuracy, energy efficiency and speed (i.e., performance) are other grand challenges to enable local intelligence in edge devices. In this article, we investigate the adoption of monolithic three-dimensional (3D) IC (M3D) technology for deep learning hardware design, using speech recognition as a test vehicle. M3D has recently proven to be one of the leading contenders to address the power, performance, and area (PPA) scaling challenges in advanced technology nodes. Our study encompasses the influence of key parameters in DNN hardware implementations towards their performance and energy efficiency, including DNN architectural choices, underlying workloads, and tier partitioning choices in M3D designs. Our post-layout M3D designs, together with hardware-efficient sparse algorithms, produce power savings and performance improvement beyond what can be achieved using conventional 2D ICs. Experimental results show that M3D offers 22.3% iso-performance power saving and 6.2% performance improvement, convincingly demonstrating its entitlement as a solution for DNN ASICs. We further present architectural and physical design guidelines for M3D DNNs to maximize the benefits.

CCS Concepts: • **Computing methodologies** → **Speech recognition**; • **Hardware** → **3D integrated circuits**;

Additional Key Words and Phrases: Monolithic 3D IC, on-chip deep neural networks, speech recognition, low power design, high performance design

ACM Reference format:
Kyungwook Chang, Deepak Kadetotad, Yu Cao, Jae-Sun Seo, and Sung Kyu Lim. 2018. Power, Performance, and Area Benefit of Monolithic 3D ICs for On-Chip Deep Neural Networks Targeting Speech Recognition. *J. Emerg. Technol. Comput. Syst.* 14, 4, Article 42 (November 2018), 19 pages.
https://doi.org/10.1145/3273956

1 INTRODUCTION

Deep neural networks (DNNs) have become ubiquitous in many machine-learning applications, from speech recognition [10, 13] and natural language processing [7] to image recognition [17, 21]

Authors' addresses: K. Chang and S. K. Lim, Georgia Institute of Technology, Atlanta, GA; emails: k.chang@gatech.edu, limsk@ece.gatech.edu; D. Kadetotad, Y. Cao, and J.-S. Seo, Arizona State University, Tempe, AZ; emails: {deepak.kadetotad, Yu.Cao, jaesun.seo}@asu.edu.

1550-4832/2018/11-ART42 $15.00
https://doi.org/10.1145/3273956

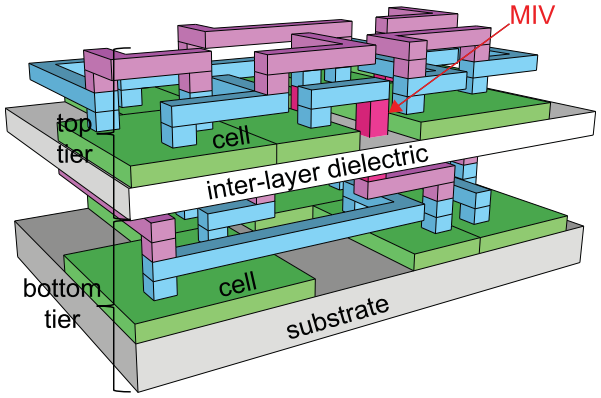

Fig. 1. A schematic showing a gate-level monolithic 3D IC (M3D).

and computer vision [20]. Large neural network models have proven to be very powerful in all the stated cases, but implementing high-speed (i.e., high-performance), energy-efficient DNN ASIC is still challenging, because (1) the required computations consume large amounts of processing time and energy, (2) the memory needed to store the weights are prohibitive, and (3) excessive wire overhead exists due to a large number of connections between neurons, which makes a DNN ASIC a heavily wire-dominated circuit.

Modern DNNs may require >100M parameters [29] for large-scale speech recognition tasks. This is impractical using only on-chip memory due to power density and temperature instability [23], and hence offloading storage to an external DRAM is required. With the introduction of an external DRAM, however, the bottleneck for computation efficiency is now determined by the parameter fetching from DRAM [28]. To mitigate this bottleneck, recent works have compressed the neural network weights *in architectural perspective* and substantially reduced the amount of computation required to obtain the final output [5, 16, 18, 19], which becomes crucial for efficient DNN ASICs. An alternate method of reducing the complexity caused by the vast requirement of memory for DNNs is in-training quantization of the network parameters [8, 9], this method, however, is not explored in the current work.

With the weight-compressed DNN architecture, we adopt monolithic three-dimensional (3D) IC (M3D) technology [1] to further improve the energy efficiency and performance *in physical design perspective*. As device scaling in advanced technology nodes is slowly saturating due to low volume and yields, 3D IC technology has come into the spotlight as an alternative for continuing Moore's low. M3D has shown its strength in reducing power consumption and enhancing performance by effectively minimizing wirelength as well as routing congestion, especially in wire-dominated circuits like DNN ASICs. As shown in Figure 1, in M3D, transistors are fabricated onto multiple tiers, and the connections crossing the tiers are established by nano-scale monolithic inter-tier vias (MIVs) [1]. Owing to the minuscule size of MIVs (<100nm), M3D achieves orders-of-magnitude denser vertical integration with lower RC parasitics compared with through-silicon via (TSV)-based 3D ICs [24]. In so-called gate-level M3D, each standard cell occupies a single tier—as opposed to being split into multiple tiers—and MIVs are utilized for inter-cell connections that cross tiers. Efficient CAD tool flows exist [3, 25], and studies have demonstrated its performance and power improvements across multiple technology generations [4].

In this article, for the first time, we investigate the benefit of M3D on DNN ASIC implementations and explore architectural and physical design decisions that impact its power consumption [2] and performance. We present two DNN architectures for speech recognition with different

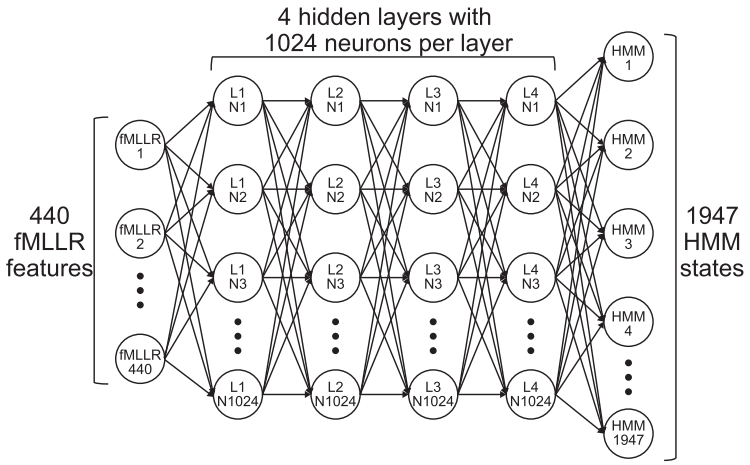

Fig. 2. Diagram of our DNN for speech recognition.

granularity of weight compression and implement them in both 2D and M3D designs. We also examine two schemes for memory floorplan in M3D designs, and comprehensively compare power, performance and area (PPA) benefits. The main contributions of this article are as follows: (1) The impact of M3D on DNN architectures with different granularity in sparsity is investigated, (2) we study the impact of tier partitioning in our M3D designs to better handle memory blocks, and (3) feed-forward classification and pseudo-training workloads are examined thoroughly to investigate their impact on power reduction. (4) We demonstrate an in-depth analysis on the performance benefit of M3D DNN ASICs over their 2D counterparts and (5) present key guidelines on optimal architectural and physical design decisions for M3D DNN ASICs.

2 DEEP NEURAL NETWORK FOR SPEECH RECOGNITION

In this article, we focus on DNN for speech recognition, but the proposed methodologies can be adopted to DNNs for other applications as well. In this section, we present the topology, the training and classification strategy of our DNN architectures and the coarse-grain sparsification (CGS) that effectively reduces area and computation overhead of DNNs.

2.1 Our DNN Topology

Starting from a fully connected DNN, we adopt a Gaussian Mixture Model (GMM) for acoustic modeling [27]. Since it has been shown that DNNs in conjunction with Hidden Markov Models (HMMs) increase recognition accuracy [10], a HMM is also employed to model the sequence of phonemes. The most likely sequence is determined by the HMM utilizing the Viterbi algorithm for decoding. Then, we adopt the CGS methodology presented in Reference [19] in our DNN architecture to reduce the memory footprint as well as the computation for DNN classification.

As shown in Figure 2, our DNN for speech recognition consists of four hidden layers with 1,024 neurons per layer. There are 440 input nodes corresponding to 11 frames (5 previous, 5 future, and 1 current) with 40 feature-space Maximum Likelihood Linear Regression (fMLLR) features per frame. The output layer consists of 1,947 probability estimates, and they are sent to the HMM unit to determine the best sequence of phoneme using the TIMIT database [12]. The Kaldi toolkit [26] is utilized for the transcription of the words and sentences for the particular set of phonemes.

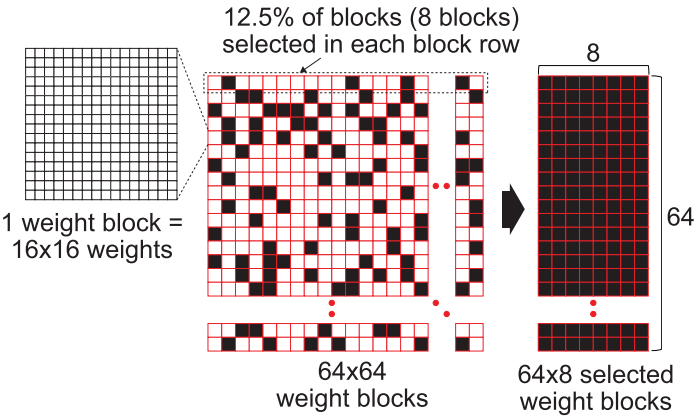

Fig. 3. A 1024×1024 weight matrix is divided into 64×64 weight blocks with each weight block having 16×16 weights (i.e., block size of 16×16). Of the weight blocks, 87.5% are dropped using CGS. The remaining 12.5% weight blocks are stored in memory.

2.2 DNN Training and Classification

Our DNN is trained with the objective function that minimizes the cross-entropy error of the outputs of the network, as described in Equation (1),

$$E = -\sum_{i=1}^{N} t_i \cdot ln(y_i), \tag{1}$$

where N is the size of the output layer, y_i is the ith output node, and t_i is the ith target value or label. The mini-batch stochastic gradient method [11] is used to update the weights. The weight W_{ij} is updated in the $(k+1)$th iteration using Equation (2),

$$(W_{ij})_{k+1} = (W_{ij})_k + C_{ij}(-lr(\Delta W_{ij})_k + m(\Delta W_{ij})_{k-1}), \tag{2}$$

where m is the momentum, lr is the learning rate, and C_{ij} is the binary connection coefficient between two subsequent neural network layers for CGS. In CGS, only the weights that correspond to the location where $C_{ij} = 1$ are updated. The change in weight for each iteration is the differential of the cost function with respect to the weight value:

$$\Delta W = \frac{\delta E}{\delta W}, \tag{3}$$

such that the loss reduces in each iteration. The training procedure is performed on a GPU with 32-bit floating point values.

After training, feed-forward computation is performed for classification, through matrix-vector multiplication of weight matrices and neuron vectors in each layer to obtain the output of the final layer. The Rectified Linear Unit (ReLU) function [21] is used for the non-linear activation function at the end of each hidden layer.

2.3 Coarse-Grain Sparsification

To efficiently map sparse weight matrices to memory arrays, CGS methodology [19] is employed. In CGS, connections between two consecutive layers in a DNN are compressed in a blockwise manner. An example of blockwise weight compression is demonstrated in Figure 3. For a given block size of 16×16, it reduces a 1024×1024 weight matrix to 64×64 weight blocks. With a compression ratio of 87.5%, only eight weight blocks (12.5%) remain non-zero for each block row, thus allowing for efficient compression of the entire weight matrix with minimal index.

Table 1. Key Parameters of the Two CGS-based DNN Architectures Used in Our Study: Block Size of 16×16 (DNN CGS-16) and Block Size of 64×64 (DNN CGS-64)

Parameter	DNN CGS-16	DNN CGS-64
Block size	16×16	64×64
Compression rate	87.5%	87.5%
Phoneme error rate	19.8%	19.9%

CGS, when compared to recent neural network compression algorithms such as in References [6, 15], offers simpler hardware implementation through CGS multiplexers and MACs. In Reference [15], a complex sparse matrix vector multiplication module is required. However, the methodology in Reference [6] offers to reduce the order of computations needed for a matrix of size n to $O(n \log n)$ and reduce the space required to store the matrix to $O(n)$. However, there is considerable loss in accuracy when the size of the matrix increases, and hardware for computing Fast Fourier transform (FFT) and Inverse Fast Fourier transform (IFFT) is required. The issue of matrix size is resolved in Reference [22] using block-circulant matrices, but the advantage of using FFT and IFFT to compute matrix vector multiplications is lost if the size of the blocks reduce significantly. This restriction is not present if CGS is used.

GPU-accelerated DNN computations can also benefit from CGS. With CGS, along with the testing inference, training complexity can also be reduced due to the sparse nature of the weight matrices. The structured sparseness allows for writing customized GPU kernels that only need to operate on the non-zero elements, significantly speeding up training and reducing GPU power consumption as shown in Reference [14].

To study the impact of M3D on PPA in different DNN architectures, the block sizes are swept for the compression ratio of 87.5%, and the two DNN architectures that have the two lowest phoneme error rates (PER) for the TIMIT dataset are selected for hardware implementation. The two architectures chosen are the DNN with 16×16 block size (DNN CGS-16) and the DNN with 64×64 block size (DNN CGS-64), as shown in Table 1.

3 FULL-CHIP MONOLITHIC 3D IC (M3D) DESIGN FLOW

To implement two-tier full-chip M3D designs of the chosen DNN architectures, we use the state-of-the-art M3D design flow presented in Reference [25]. The flow starts with scaling width and height of all standard cells and metal layers by $1/\sqrt{2}$, so that an overlap-free design can be implemented in half the footprint of the corresponding 2D design. The shrunk cells and metal layers are then used to implement a shrunk 2D design by performing all design stages including placement, pre-CTS (clock tree synthesis) optimization, CTS, post-CTS optimization, routing, and post-route optimization in Cadence Innovus. From this shrunk 2D design, only the cell placement information (x-y location of cells) is retained, and all other information is discarded.

Next, the cells in the shrunk 2D design are scaled back to their original size, resulting in overlap between the cells. To remove the overlap, the cells in the shrunk 2D design are partitioned into two tiers. This is accomplished using an area-balanced min-cut partitioning algorithm, which enables half of the cells to be placed on the top tier, and the other half on the bottom tier while minimizing the number of connections between them. The connections between the top and bottom tiers utilize MIVs in the final M3D design. After partitioning, the remaining overlapped cells on both tiers are removed through legalizing.

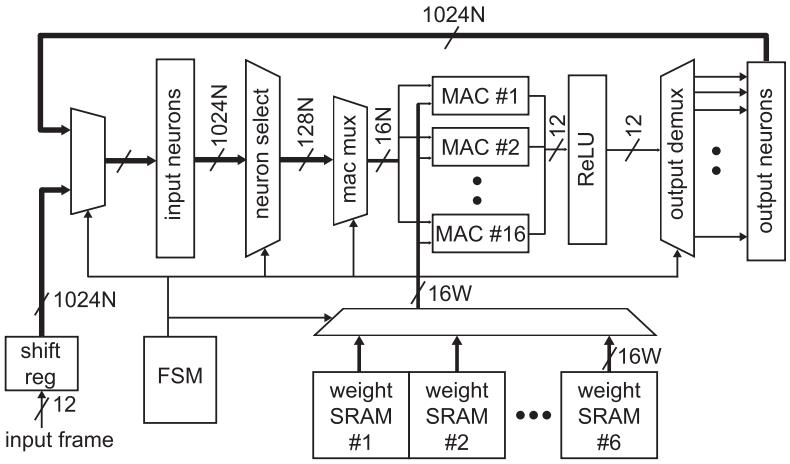

Fig. 4. Block diagram of the proposed CGS-based DNN architecture for speech recognition.

To determine the location of MIVs, we first duplicate all metal layers used in the design, so that the original metal layers represent the metal layers on the bottom tier, and the duplicated layers represent those on the top tier. Then, we define two flavors for all standard cells and memory blocks: the bottom tier cells and the top tier cells. Pins on the bottom tier cells are assigned to the original metal layers, and those on the top tier cells to the duplicated metal layers. After mapping all cells and memory blocks onto their corresponding flavor, the structure is routed in Cadence Innovus. The locations of vias between the top metal layer of the original stack and the bottom metal layer of the duplicated stack become MIVs in the final M3D design.

Once the cell and MIV locations are determined, two designs, the top and bottom tier designs, are generated, and trial routing is performed for each tier. Using Synopsys PrimeTime and trial-routed designs for each tier, timing constraints for both tiers are derived. The timing constrains are used to perform timing-driven detailed routing for each tier, which results in the final M3D design.

4 DNN ARCHITECTURE DESCRIPTION

The block diagram of our CGS-based DNN architecture is shown in Figure 4. The DNN operates on one layer at a time and consists of 16 multiply and accumulate (MAC) units that operate in parallel. The weights of the network are stored in the SRAM banks, while the input and output neurons are stored in registers. The finite-state machine (FSM) coordinates the data flow such as layer control and computational resource allocation (i.e., MAC units).

Since the target compression ratio of our architectures is 87.5%, the neuron select unit chooses 128 neurons (12.5%) among 1,024 input neurons that proceed to the MAC units. This selection-based computation eliminates unnecessary MAC operations (i.e., MAC operation of neurons corresponding to zero weights in CGS-based weight matrix). The neuron select unit is controlled by the binary connection coefficients discussed in Section 2.2, and the coefficients are stored in the dedicated register file in the FSM unit.

The size of the register file is determined by the block size used in the DNN architecture. For example, for each hidden layer, eight weight blocks per each row of 64×64 weight blocks are selected for MAC operation in the DNN CGS-16 architecture (Figure 3). Thus, eight multiplexers are required in the neuron select unit, and each multiplexer selects one weight block among 64 in a block row, so that each multiplexer requires six selection bits ($=\log_2 64$). Since there are 64 total block rows in the architecture, the total number of bits to obtain 64×8 selected weight block

for a hidden layer is 3,072 bits (= eight multiplexers × 6 selection bits × 64 block rows). Although the DNN has four hidden layers, the number of coefficients for the last hidden layer should be doubled, because the number of neurons in the output layer (1,947 HMM states) is almost 2× of other layers. Therefore, the size of the coefficient register file in the DNN CGS-16 is 15,360 bits (= 3,072 bits × 5 effective layers). This value is calculated in the same way for the DNN CGS-64 architecture, resulting in 640 bits in total.

On-chip SRAM arrays store the compressed weight parameters in six banks for the four hidden layers and the output layer (~ 2× parameters). The size of the SRAM bank is determined by the number of MAC units in the architecture. Since our DNN architectures operate 16 units in parallel, the row size of each SRAM bank is 128 bits (= 16 MAC units × 8-bit weight precision). Since we assume 8,192 rows for each SRAM bank, the total size of the six SRAM banks in the DNN is 6Mb (=6 banks × 128 bits × 8,192 rows). This compact memory size with the CGS methodology enables the DNN to store the compressed weight parameters on chip.

5 M3D IMPACT ON ENERGY-EFFICIENCY

To analyze the advantage of M3D on energy efficiency of different DNN architectures, two DNN architectures (DNN CGS-16 and CGS-64) are implemented using TSMC 28nm HPM technology with a target clock frequency of 400MHz. The footprint of 2D designs are set by targeting the initial standard cell density (excluding memory block area) before place-and-route to 65%. The impact of tier partitioning scheme is examined by comparing two memory floorplan schemes for M3D designs, one with memory blocks on both tiers (M3D-both), and the other with memory blocks on a single tier only (M3D-one). In the M3D-both design, memory blocks are evenly split on the top and bottom tiers using similar floorplan for both tiers. However, in the M3D-one design, all standard cells are placed on one tier, and only memory blocks exist on the other tier. Figure 5 shows the full-chip layouts of the implemented 2D and M3D designs.

5.1 Area, Wirelength, and Capacitance Comparisons

Iso-performance comparison of several key metrics of the 2D and M3D designs is presented in Table 2. We summarize our findings as follows:

- Footprint: Our M3D-both designs achieve 50.1% footprint reduction compared with the 2D designs, whereas the M3D-one designs obtain only 33.9% reduction. This difference is attributed to the large memory area compared with logic: 1.287mm^2 vs. 0.505mm^2 in the 2D CGS-16 design, for example. These large memory blocks, if placed in the same tier, cause the footprint to increase significantly.
- Wirelength: Our wirelength saving reaches 29.9% and 33.7% in CGS-16 and CGS-64, respectively, with our M3D-both designs. This significant wirelength saving comes from 50% smaller footprint and shorter distance among cells in M3D designs.
- Cell area: We achieve 12.1% cell count reduction, which leads to 14.6% total cell area saving in our M3D-both design for CGS-16 architecture. This saving mainly comes from fewer buffers and smaller gates needed to close timing in M3D designs compared with the 2D counterparts. Our savings in CGS-64 architecture are 8.2% and 14.3% for the cell count and area, respectively.
- MIV usage: We use 77K MIVs in our CGS-16 architecture, while 48K MIVs are used in CGS-64. This is mainly because CGS-16 design is more complex than CGS-64 (to be further discussed in Section 7.1) so that our tier partitioning cutline cuts through more inter-tier connections in CGS-16. In the M3D-one design, logic and memory are separated into different tiers. This logic-memory connectivity is not high in our DNN architecture (= 1.7K).

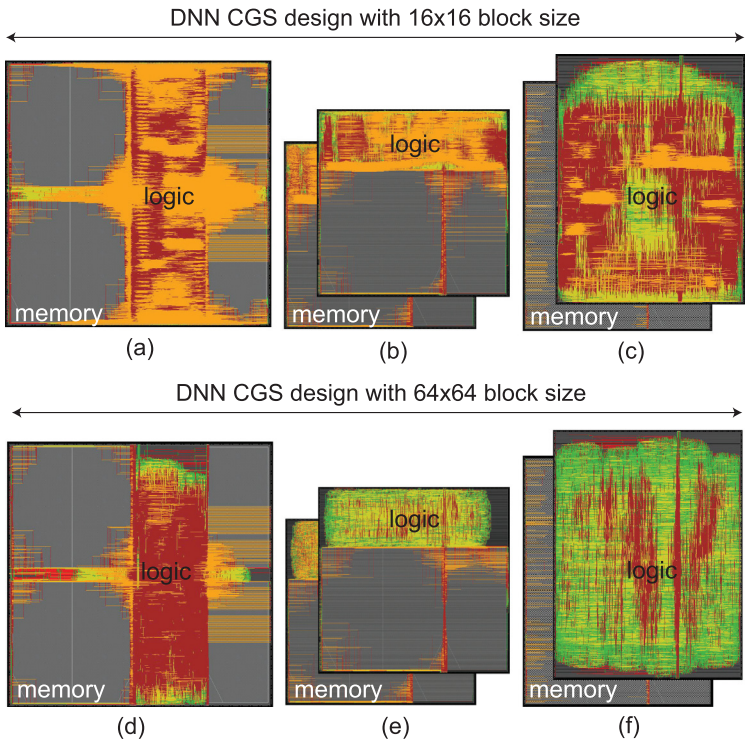

Fig. 5. The 28nm full-chip layouts of DNN CGS-16 and CGS-64 architectures at 400MHz target clock frequency. (a) 2D IC design, (b) M3D design with memory blocks on both tier (M3D-both), (c) M3D design with memory blocks on a single tier (M3D-one), (d) 2D IC, (e) M3D-both, and (f) M3D-one.

- Capacitance: In our CGS-16 architecture, the 16.5% pin capacitance saving is from cell area reduction, while the 35.0% wire capacitance saving is from wirelength reduction. By comparing the raw data (943.3pF vs. 2,216.8pF in the 2D design), we note that our DNN architecture is wire-dominated. Our pin/wire capacitance saving reaches 25.0% and 37.7% in CGS-64.

To better understand why M3D-one gives significantly worse results than M3D-both, we show a placement comparison among 2D, M3D-both, and M3D-one designs in Figure 6. In the M3D-both design shown in Figure 6(b), the logic cells related to memory blocks in the top tier are placed in the same tier as the memory and densely packed to reduce wirelength effectively. This is the same for the bottom tier in the M3D-both design. However, we see that logic gates are rather spread out across the top tier in the M3D-one design shown in Figure 6(c). This results in 1.1% *increase* in wirelength for CGS-16 and 26.7% *increase* in wirelength for CGS-64 compared with the 2D counterparts. This highlights the importance of footprint management and tier partitioning in the presence of large memory modules in DNN architectures.

5.2 Power Comparisons

Table 3 presents the iso-performance power comparison between 2D and M3D designs of CGS-based DNNs. We report internal, switching, and leakage breakdown for each design. Our sign-off power calculations are conducted using two speech recognition workloads: classification and pseudo-training (more details provided in Section 7.2). From examining the power metrics of the 2D designs only, we observe the following:

Table 2. Iso-performance (400MHz) Comparison of Design Metrics of 2D and M3D Designs of DNN CGS-16 and DNN CGS-64 Architectures

Parameter	2D	M3D-both		M3D-one	
DNN CGS-16					
Footprint (um)	1411×1411	1010×984	**(−50.1%)**	996×1322	(−33.9%)
Wirelength (m)	12.089	8.469	**(−29.9%)**	12.225	(1.1%)
Cell count	298,309	262,084	**(−12.1%)**	290,692	(−2.6%)
Cell area (mm^2)	0.505	0.431	**(−14.6%)**	0.511	(1.1%)
Mem area (mm^2)	1.287	1.287	(0.0%)	1.287	(0.0%)
MIV count	—	77,536		1,776	
Pin cap (pF)	943.3	788.0	**(−16.5%)**	1,004.1	(6.4%)
Wire cap (pF)	2,216.8	1,440.8	**(−35.0%)**	2,087.4	(−5.8%)
Total cap (pF)	3,160.1	2,228.7	**(−29.5%)**	3,091.6	(−2.2%)
DNN CGS-64					
Footprint (um)	1411×1411	1010×984	**(−50.1%)**	996×1322	(−33.9%)
Wirelength (m)	5.631	3.734	**(−33.7%)**	7.134	(26.7%)
Cell count	163,361	149,921	**(−8.2%)**	174,292	(6.7%)
Cell area (mm^2)	0.314	0.269	**(−14.3%)**	0.328	(4.7%)
Mem area (mm^2)	1.287	1.287	(0.0%)	1.287	(0.0%)
MIV count	—	48,636		1,776	
Pin cap (pF)	520.8	390.8	**(−25.0%)**	553.5	(6.3%)
Wire cap (pF)	920.1	573.7	**(−37.7%)**	1,110.5	(20.7%)
Total cap (pF)	1,440.9	964.4	**(−33.1%)**	1,664.0	(15.5%)

All percentage values show the reduction from their 2D counterparts.

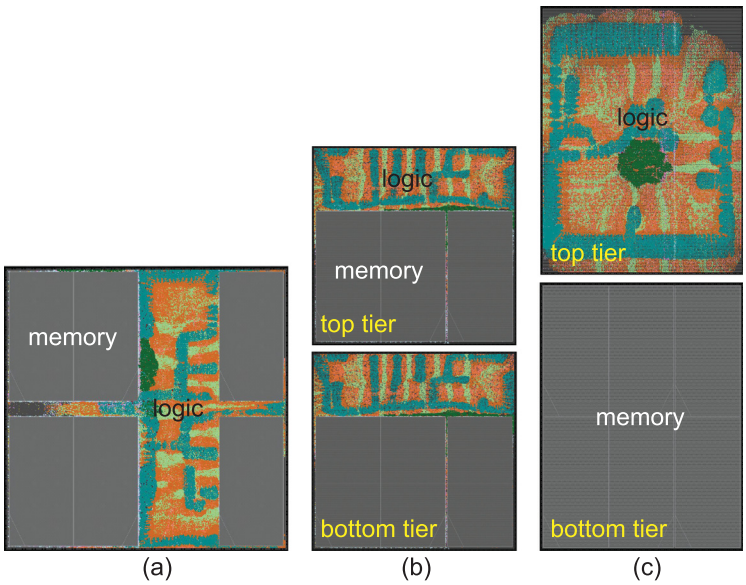

Fig. 6. Cell placement of the modules in CGS-16 architecture. (a) 2D, (b) M3D-both, and (c) M3D-one. Each module is highlighted with different colors.

Table 3. Iso-performance (400MHz) Power Comparison of Two Architectures (CGS-16 vs. CGS-64) Using Two Workloads (Classification vs. Pseudo-training)

Workload	Power breakdown	2D	M3D-both		M3D-one	
DNN CGS-16						
Classification	Internal power (mW)	91.3	76.7	(−16.0%)	90.3	(−1.1%)
	Switching power (mW)	48.6	31.6	(−35.0%)	46.5	(−4.3%)
	Leakage power (mW)	1.3	1.2	(−6.6%)	1.3	(0.5%)
	Total power (mW)	**141.1**	109.6	**(−22.3%)**	138.0	(−2.2%)
Pseudo-training	Internal power (mW)	150.4	142.8	(−5.1%)	148.3	(−1.4%)
	Switching power (mW)	68.4	57.1	(−16.6%)	65.6	(−4.2%)
	Leakage power (mW)	1.3	1.2	(−6.8%)	1.3	(0.7%)
	Total power (mW)	**220.0**	201.0	**(−8.6%)**	215.0	(−2.3%)
DNN CGS-64						
Classification	Internal power (mW)	86.8	76.1	(−12.3%)	84.9	(−2.2%)
	Switching power (mW)	41.2	30.2	(−26.7%)	42.8	(3.9%)
	Leakage power (mW)	1.1	1.1	(−4.7%)	1.1	(1.5%)
	Total power (mW)	**129.1**	107.3	**(−16.9%)**	128.8	(−0.2%)
Pseudo-training	Internal power (mW)	129.2	120.0	(−7.2%)	128.5	(−0.5%)
	Switching power (mW)	46.0	36.3	(−21.2%)	50.3	(9.3%)
	Leakage power (mW)	1.1	1.1	(−4.6%)	1.1	(1.4%)
	Total power (mW)	**176.3**	157.4	**(−10.7%)**	179.9	(2.0%)

All percentage values show the reduction from their 2D counterparts.

- CGS-16 vs. CGS-64: During classification, CGS-16 consumes 141.1mW, while CGS-64 consumes 129.1mW. This confirms that CGS-16 consumes more power to handle more complex weight selection process (to be further discussed in Section 7.1). A similar trend is observed during pseudo-training: 220.0mW vs. 176.3mW.
- Classification vs. pseudo-training: Pseudo-training, as expected, causes more switching in the circuits, and thus more power consumption compared with classification: 220.0mW vs. 141.1mW for CGS-16. A similar trend is observed for CGS-64: 176.3mW vs. 129.1mW.

Next, we compare 2D vs. M3D power consumption. To explain the power reduction of M3D designs, Equation (4) is employed, which describes the components comprising dynamic power consumption.

$$\begin{aligned} P_{dyn} &= P_{INT} + P_{SW} \\ &= \alpha_{IN} \cdot I_{SC} \cdot V_{DD} \cdot f_{clk} \\ &\quad + \alpha_{OUT} \cdot (C_{pin} + C_{wire}) \cdot {V_{DD}}^2 \cdot f_{clk}. \end{aligned} \tag{4}$$

The first term P_{INT} indicates the internal power consumption of standard cells and memory blocks. P_{INT} is the product of short-circuit current (I_{SC}) during input switching, input activity factor α_{IN}, clock frequency f_{clk} and V_{DD}. The second term P_{SW} represents the switching power dissipated during the charging or discharging of output load capacitance of cells ($C_{pin} + C_{wire}$). It is represented by the product of the output load capacitance, output activity factor α_{OUT}, f_{clk}, and V_{DD}.

The resulting footprint of M3D-both designs is reduced by half, thereby reducing the wirelength between the cells. Figure 7 shows the wirelength distribution of the 2D and M3D designs of CGS-16 architecture. The histogram clearly shows that M3D designs contain more number of short wires and fewer long wires compared with 2D. The effect of wirelength saving translates to the

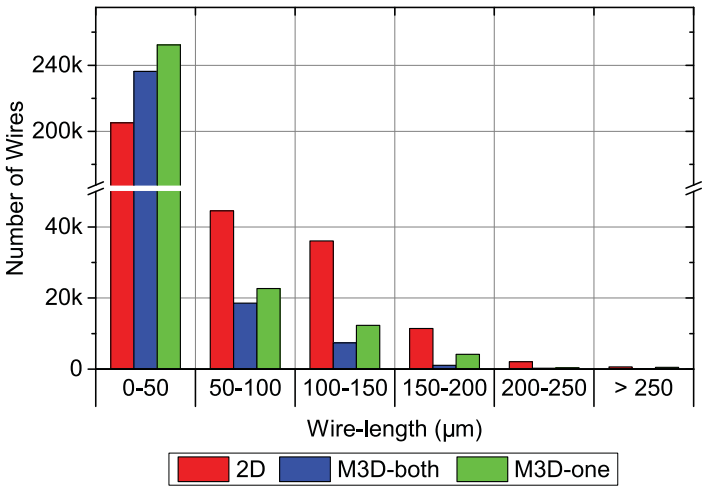

Fig. 7. Wirelength distribution of CGS-16 architecture.

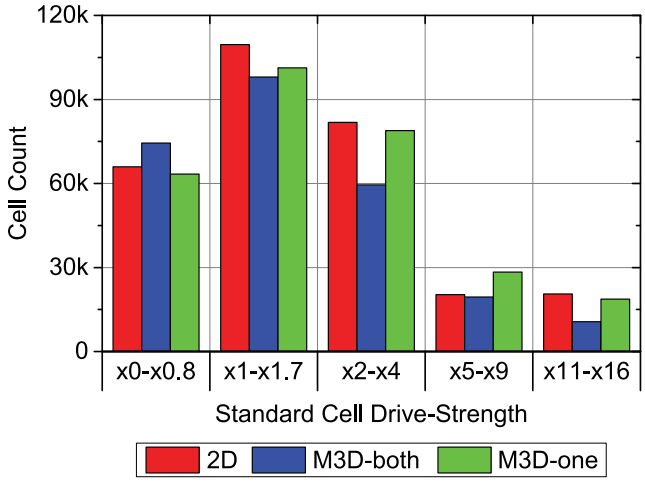

Fig. 8. Cell drive-strength distribution of CGS-16 architecture.

reduction of wire capacitance C_{wire} in Equation (4), therefore the saving of P_{SW}. Figure 8 presents the distribution of standard cells with different ranges of cell drive-strength. We observe that M3D-both design uses more number of low drive-strength cells (i.e., ×0–×0.8) and fewer high drive-strength cells (i.e., ×1–×16). Since low drive-strength cells utilize smaller transistors, their I_{SC} and C_{pin} are lower, which reduces both P_{INT} and P_{SW} in Equation (4).

6 M3D IMPACT ON PERFORMANCE

In this section, we investigate the impact of M3D on performance of CGS-16 and CGS-64 architectures by pushing the target clock frequency of 2D and M3D designs to their maximum clock frequency. Two-dimensional and M3D designs are implemented with TSMC 28nm HPM technology sweeping the target frequency from 400MHz in 25MHz increments. The floorplans of the 2D and M3D designs are same as the ones used in Section 5. As M3D-both designs show better design quality compared to M3D-one designs as discussed in the section, we place memory blocks on both tiers in the M3D designs for this experiment as shown in Figure 9.

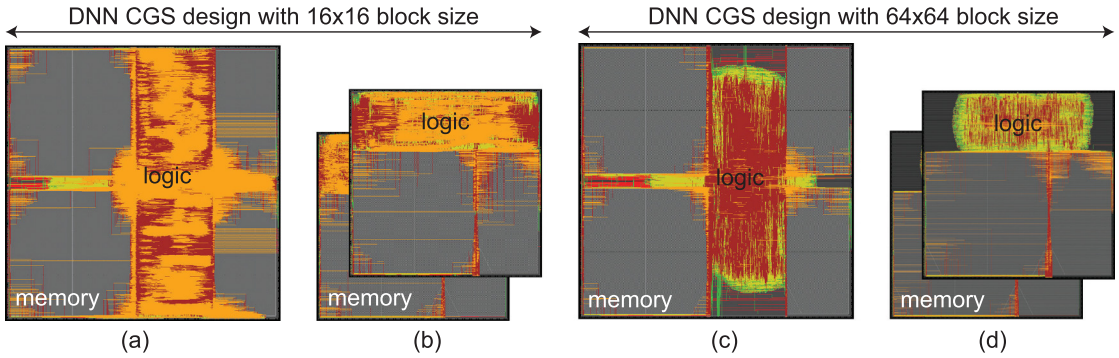

Fig. 9. Full-chip die images of 2D and M3D designs of DNN CGS-16 and CGS-64 architectures at their maximum target clock frequency. (a) Two-dimensional design at 550MHz, (b) M3D design at 575MHz of DNN CGS-16 architecture, (c) 2D design at 600MHz, (d) M3D design at 625MHz of DNN CGS-64 architecture.

Table 4. Maximum Performance Comparison of 2D and M3D Designs of CGS-16 and CGS-64 Architectures

Parameter		DNN CGS-16	DNN CGS-64
2D	Target clk freq (MHz)	550	600
	WNS (ns)	−0.056	0.002
	Effective clk freq (MHz)	**534**	**601**
M3D	Target clk freq (MHz)	575	625
	WNS (ns)	−0.024	−0.046
	Effective clk freq (MHz)	**567**	**608**
Δ% effective clk freq		**6.2%**	**1.2%**

The maximum performance comparison between the 2D and M3D designs of CGS-16 and CGS-64 architectures is presented in Table 4. The table shows the target clock frequency used to place-and-route the designs, the resulting worst negative slack (WNS) from static timing analysis, and the effective clock frequency, which is the maximum achievable clock frequency that the designs are able to operate at without timing violation.

Comparing only the 2D designs of CGS-16 and CGS-64 architectures, we observe the following:

- CGS-16 vs. CGS-64: the effective clock frequency of the 2D CGS-16 design is 11.1% less than the 2D CGS-64 design. As the critical path of the 2D CGS-16 design starts from weight SRAM to MAC unit through weight selection logic, the lower effective clock frequency of the 2D CGS-16 design is attributed to its more complex weight selection logic as shown in a higher design density in Figure 9(a) compared to Figure 9(c).

Next, we compare the maximum performance of the 2D and M3D designs. Our M3D designs shows 6.2% and 1.2% performance improvement over 2D counterparts in CGS-16 and CGS-64 architectures, respectively. To analyze this trend, we first conduct the worst timing path comparison of the 2D and M3D designs. Figure 10 compares the same timing path (i.e., the worst timing path of the 2D design) in the 2D and M3D CGS-16 designs at the maximum target clock frequency of the 2D design, and Table 5 presents key metrics of the timing path. The followings summarize our observations:

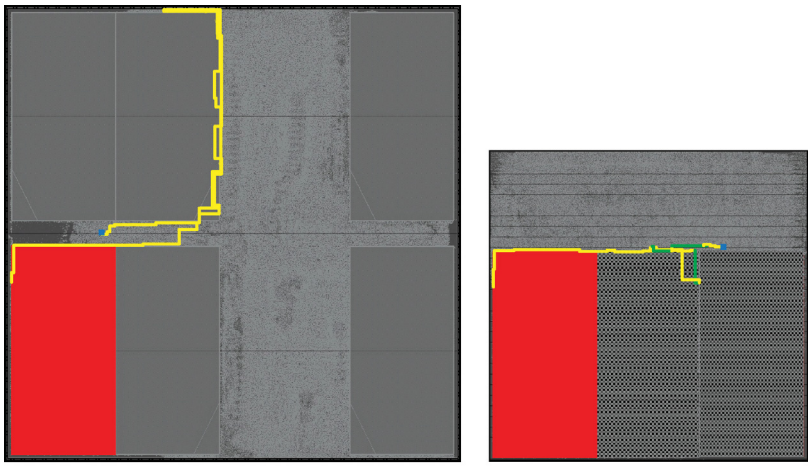

Fig. 10. Worst timing path comparison of 2D and M3D designs of CGS-16 architecture. (a) The worst timing path of 2D design at its maximum target clock frequency, 550MHz. (b) The same timing path in M3D design. Cells in the top tier are projected into the bottom tier for M3D design, and red boxes (i.e., weight SRAM) indicate the start point, whereas blue boxes (i.e., flip-flops in MAC unit) represent the end point of the timing path. Yellow lines show the wires in 2D and the bottom tier of M3D design, whereas green lines are the top tier wires in M3D design.

Table 5. Key Parameter Comparison of the Worst Timing Path in Figure 10 of the 2D and M3D Designs of DNN CGS-16 Architecture

Parameter	2D	M3D	
Wirelength (μm)	3,208	1,488	(−53.6%)
Cell count	65	49	(−24.6%)
Avg. cell drv-str	8.9	7.0	(−21.3%)
Cell area	180.1	66.4	(−63.1%)
MIV count	—	6	
Wire cap (fF)	500	242	(−51.6%)
Pin cap (fF)	486	312	(−35.8%)
Resistance ($k\Omega$)	14.5	9.3	(−35.9%)
Delay (ns)	**2.344**	**2.088**	**(−10.9%)**

- Wirelength: The wirelength of the worst timing path of the 2D design is 53.6% longer than the same timing path in the M3D design. This is attributed to the reduced footprint and the inter-tier connections of the M3D design, which results in shorter distance among cells along the timing path.
- Cell: Our M3D design offers 24.6% cell count saving as well as 21.3% average cell drive-strength reduction, thereby reducing cell area by 63.1% of the timing path. This is because fewer and smaller buffers are needed to drive the reduced wire load, which is a result of the wirelength reduction.
- Capacitance: Compared to the 2D design, the wire and pin capacitance of the timing path in the M3D design are reduced by 51.6% and 35.8%, respectively. The wire capacitance

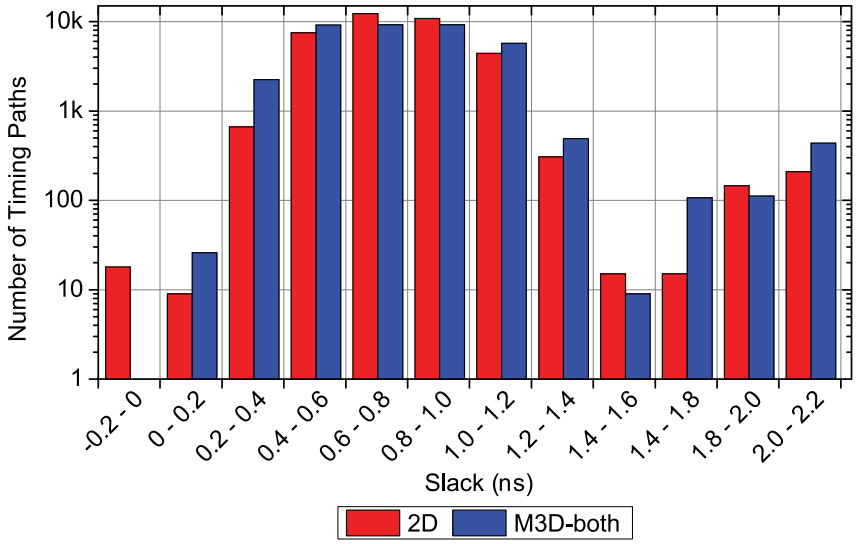

Fig. 11. Slack distribution comparison between 2D and M3D designs of DNN CGS-16 architecture at the maximum clock frequency of the M3D design.

reduction mainly comes from the wirelength reduction of the timing path, whereas the pin capacitance saving results from the cell count and cell drive-strength reduction.

- Resistance: Our M3D design achieves 35.9% resistance reduction in the timing path. The resistance saving is also attributed to the wirelength saving along the timing path.
- Delay: Due to the capacitance and resistance saving of the worst timing path, the delay of the timing path is reduced by 10.9% in the M3D design, thereby offering rooms to improve the performance.

To understand the impact of the above observations to the overall timing paths of the 2D and M3D designs, we report the slack distribution of all timing paths of the 2D and M3D CGS-16 designs in Figure 11. While 18 timing paths of the 2D design violate the timing constraints, the M3D design successfully closes timing without any violation. In addition, we observe that there are more timing paths with high positive slack in the M3D design, which indicates that timing is easily closed in the M3D design due to the reduced delay of the timing paths.

The difference in the performance improvement of the M3D designs of CGS-16 and CGS-64 architecture is also attributed to the complex weight selection logic in CGS-16 and will be discussed in detail in Section 7.1.

7 ARCHITECTURAL IMPACT DISCUSSIONS

7.1 CGS-16 vs. CGS-64 Architecture Comparisons

Table 3 shows that the total power reduction of M3D designs is higher in DNN CGS-16 architecture than CGS-64. Furthermore, we achieve more performance improvement with M3D in DNN CGS-16 architecture as shown in Table 4. These differences are caused by the granularity of weight selection methodology, i.e., coarse-grain sparsification (CGS) algorithm. The 1024×1024 weight matrix is divided into 256 (= 16×16) weight blocks in CGS-64 architecture. This count becomes 4,096 (= 64×64) weight blocks in CGS-16. The implication in DNN architecture is that CGS-16 requires a more complex neuron selection unit than CGS-64. Figure 12 shows the comparison of standard cell area of each module in CGS-16 and CGS-64 architectures. We show both sequential (dashed box) and combinational logic (non-dashed box) portion in each module. We observe that

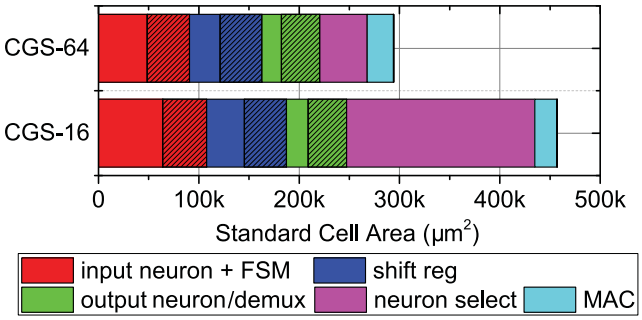

Fig. 12. Standard cell area breakdown of 2D CGS-16 and 2D CGS-64 architectures. Non-dashed and dashed boxes respectively indicates combinational and sequential elements. Only five largest modules are shown.

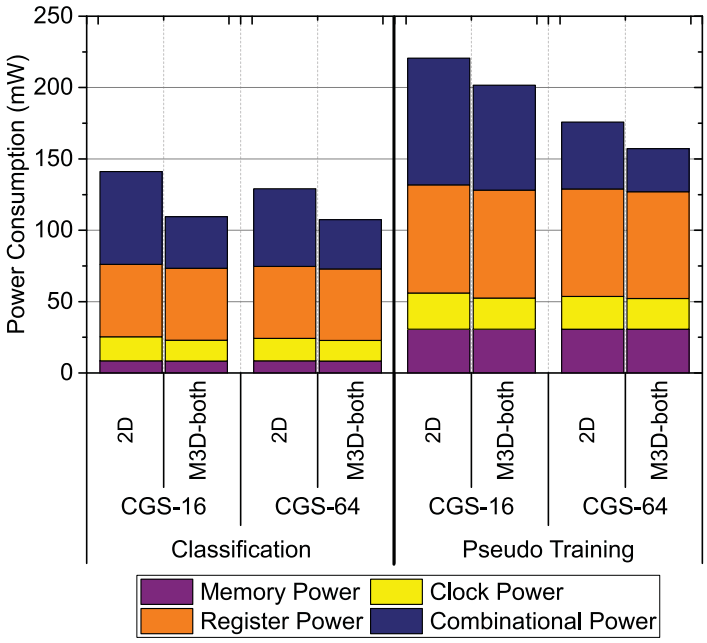

Fig. 13. Power breakdown under two architectures (CGS-16 vs. CGS-64), two workloads (classification vs. pseudo-training), and two designs (2D vs. M3D).

the neuron selection unit in CGS-16 architecture (shown in purple) occupies more area than that in CGS-64 architecture.

As discussed in Section 5.1, M3D designs benefit not only from wirelength reduction but also from standard cell area saving. The number of storage elements (i.e. sequential logic and memory blocks) used in 2D and M3D designs remain the same. Thus, the only possible power reduction coming from storage elements is their drive strength reduction. This does not show a huge impact considering the small portion of sequential elements in our DNN architectures (16.1% on average). However, combinational logic can be optimized in various ways, such as logic reconstructing and buffer reduction. Therefore, our DNN M3D designs benefit more from combinational logic gates than sequential elements.

Figure 13 shows the breakdown of total power consumption into combinational, register, clock, and memory portions. We see that combinational power reduction is the dominant factor in total power saving of M3D designs in both CGS-16 and CGS-64 architectures and in both classification and pseudo-training workloads. We also observe that the saving in other parts including register,

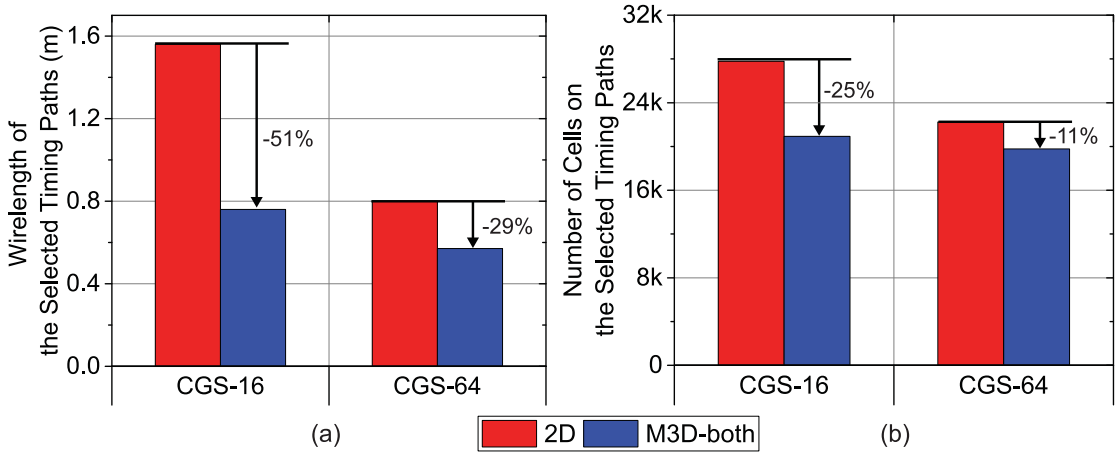

Fig. 14. Comparison of (a) the total wirelength and (b) the total cell count of the timing paths from weight SRAMs to registers in MAC units through neuron selection logic of 2D and M3D-both designs of CGS-16 and CGS-64 architecture.

clock, and memory power largely remain small. In addition, the neuron selection unit in CGS-16 architecture consists of a larger number of combinational logic gates than CGS-64. Thus, its M3D designs have more room for power optimization, resulting in a larger combinational power saving.

The larger neuron selection logic in CGS-16 architecture also offers more opportunity to improve the performance of M3D designs. While 2D designs suffer long timing path due to the complex neuron selection logic, M3D designs effectively reduce the wirelength, providing buffer count/size reduction along the worst timing path. This reduces the capacitance and resistance of timing paths, thereby offering shorter delay and larger performance improvement.

Figure 14 compares the total wirelength and standard cell count along the selected 486 timing paths, which are from weight SRAMs to registers of MAC units through neuron selection logic, in the 2D/M3D CGS-16/CGS-64 designs at the maximum frequency of the 2D designs. Comparing only the 2D designs, the 2D CGS-16 design clearly utilizes longer wirelength as well as more standard cells as the neuron selection logic is more complex. As the M3D CGS-16 design has more combinational logics to optimize with the reduced footprint, it offers more cell count and wirelength reduction compared to the M3D CGS-64 design, providing more rooms for performance improvement in higher clock frequency.

7.2 Impact of Workloads

To investigate the impact of different DNN workloads on M3D power reduction, we analyzed two main types of speech recognition DNN workloads: feed-forward classification and training. Real-world test vectors are used for feed-forward classification. However, since our current architecture does not supports online training to avoid computational overhead of finding gradients in DNN training, we create customized test vectors for "pseudo-training." Online training on DNN consists of feed-forward computation and backward computation. To mimic the online training on the current architecture, there are two phases in our pseudo-training test vectors as shown in Figure 15. In the first phase, the DNN performs feed-forward classification, which represents feed-forward computation during training. In the second phase, the DNN conducts feed-forward classification and writes the weights to memory blocks, which represents backward computation and weight update. These two phases mimic the behavior of logic computation and weight update during training.

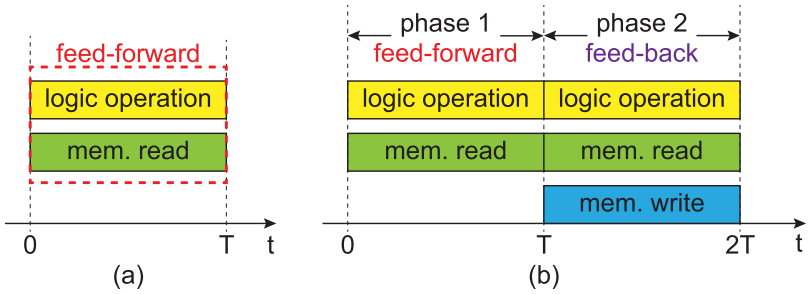

Fig. 15. Comparison between (a) the feed-forward classification and (b) pseudo-training.

Table 3 shows that while M3D-both shows 22.3% (CGS-16) and 16.9% (CGS-64) total power reduction in feed-forward classification workload, the power saving of pseudo-training workload is only 8.6% (CGS-16) and 10.7% (CGS-64). This difference stems from different switching patterns of combinational logic and storage elements in our DNN architecture. Our DNN mainly uses combinational logic gates to compute the values of neuron outputs and access memory for read operations only during feed-forward classification. Thus, this workload is classified as a compute-intensive kernel. However, memory operations are heavily used during pseudo-training, since our DNN architecture needs to read and write weights. This becomes a memory-intensive kernel. Therefore, switching activity in memory blocks is much higher during pseudo-training while that of combinational logic remains largely similar. This explains larger power consumption during pseudo-training workload: 220.0mW vs. 141.1mW for CGS-16, and 176.3mW vs. 129.1mW for CGS-64 as shown in Table 3.

As shown in Figure 13, memory power and register power occupy a large portion of the total power during pseudo-training. This means that the combinational logic power saving becomes a smaller portion of the total power saving during training. The opposite is true for classification, where memory and register power are less dominant. In this case, the reduction in combinational power saving becomes more prominent in the total power saving.

8 OBSERVATIONS AND GUIDELINES

We summarize the lessons learned from this study and provide design guidelines to maximize the power benefits of M3D designs targeting DNN architectures as follows.

- M3D effectively reduces the total power consumption of DNN architectures by reducing wirelength as well as standard cell area, showing its efficacy on saving power consumption of wire-dominated DNN circuits.
- M3D enhances the performance of DNN designs. This is mainly attributed to the reduced capacitance and resistance of timing paths in the designs, which comes from both wirelength and buffer count/size reduction.
- If memory blocks occupy more than half area of a DNN design, then partitioning the memory blocks onto two tiers (i.e., M3D-both designs), instead of placing them on one tier (i.e., M3D-one designs), helps maximize the total power saving of the M3D design. It is because M3D-both designs achieve smaller footprint in that case, which makes cell placement denser, and, hence, reduces more wirelength.
- M3D shows larger power savings with smaller CGS block sizes, which consists of more combinational logics, in speech recognition DNNs. This enables the choice of selecting smaller block sizes for CGS in hardware implementations, which was earlier overlooked due to larger power overhead in 2D designs.

- DNNs with smaller CGS block sizes also benefit more on their performance from M3D, effectively reducing the overhead of more complex weight selection logic.
- In our DNN, it was combinational logic power reduction, not the commonly believed memory-related power reduction, that dominates the overall power saving of M3D. Moreover, compute-intensive classification workload gave us more power saving than memory-intensive training workload with M3D. Such a claim cannot become a general statement, and other DNN architectures may prove to be the opposite. However, we believe that the design and analysis methodologies presented in this article pave a road for practical and convincing studies with other DNN architectures and their M3D implementations.

9 CONCLUSIONS

In this article, we investigate the impact of M3D technology on power, performance, and area with speech recognition DNN architectures that exhibit coarse-grain sparsity. Our study shows that M3D reduces the total power consumption more effectively with compute-intensive workloads, compared to memory-intensive workloads. By placing memory blocks evenly on both tiers, M3D designs reduce the total power consumption up to 22.3%. This trend can be further extended to offer greater power reduction by using in-training quantization in conjunction with structured compression as demonstrated in Reference [30], and will be explored in future works. In addition, owing to the reduced footprint and vertical integration, M3D designs offer performance improvement over 2D designs, especially in architecture with complex combinational logics. This study convincingly demonstrates the low power and high performance benefits of M3D on DNN hardware implementations and offers architectural guidelines to maximize the benefits.

ACKNOWLEDGMENTS

This work was in part supported by the National Science Foundation grants 1652866 and 1715443, and C-BRIC, one of six centers in JUMP, a SRC program sponsored by DARPA.

REFERENCES

[1] P. Batude, M. Vinet, A. Pouydebasque, C. Le Royer, B. Previtali, C. Tabone, J. M. Hartmann, L. Sanchez, L. Baud, V. Carron, A. Toffoli, F. Allain, V. Mazzocchi, D. Lafond, O. Thomas, O. Cueto, N. Bouzaida, D. Fleury, A. Amara, S. Deleonibus, and O. Faynot. 2009. Advances in 3D CMOS sequential integration. In *Proceedings of the IEEE International Electron Devices Meeting*.

[2] K. Chang, D. Kadetotad, Y. Cao, J. Seo, and S. K. Lim. 2017. Monolithic 3D IC designs for low-power deep neural networks targeting speech recognition. In *Proceedings of the International Symposium on Low Power Electronics and Design*.

[3] K. Chang, S. Sinha, B. Cline, R. Southerland, M. Doherty, G. Yeric, and S. K. Lim. 2016. Cascade2D: A design-aware partitioning approach to monolithic 3D IC with 2D commercial tools. In *Proceedings of the IEEE International Conference on Computer-Aided Design*.

[4] K. Chang, S. Sinha, B. Cline, G. Yeric, and S. K. Lim. 2016. Match-making for monolithic 3D IC: Finding the right technology node. In *Proceedings of the ACM Design Automation Conference*.

[5] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. 2017. A survey of model compression and acceleration for deep neural networks. (2017). arXiv:1710.09282.

[6] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang. 2015. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*.

[7] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. (2017). arXiv:1705.02364.

[8] M. Courbariaux, Y. Bengio, and J.-P. David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*.

[9] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1. (2016). arXiv:1602.02830.

[10] L. Deng, G. Hinton, and B. Kingsbury. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*.
[11] W. A. Gardner. 1984. Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique. *Signal Processing* 6, 2 (1984), 113–133.
[12] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett. 1993. *DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus*. NASA STI/Recon Technical Report N.
[13] A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*.
[14] S. Gray, A. Radford, and D. Kingma. 2017. *GPU Kernels for Block-Sparse Weights*. Technical Report. OpenAI.
[15] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, et al. 2017. ESE: Efficient speech recognition engine with sparse LSTM on FPGA. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*.
[16] S. Han, H. Mao, and W. J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proceedings of the International Conference on Learning Representations*.
[17] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
[18] T. He, Y. Fan, Y. Qian, T. Tan, and K. Yu. 2014. Reshaping deep neural network for fast decoding by node-pruning. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.
[19] D. Kadetotad, S. Arunachalam, C. Chakrabarti, and J. Seo. 2016. Efficient memory compression in deep neural networks using coarse-grain sparsification for speech applications. In *Proceedings of the IEEE International Conference on Computer-Aided Design*.
[20] A. Karpathy and L. Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.
[22] S. Liao, Z. Li, X. Lin, Q. Qiu, Y. Wang, and B. Yuan. 2017. Energy-efficient, high-performance, highly-compressed deep neural network design using block-circulant matrices. In *Proceedings of the IEEE International Conference on Computer-Aided Design*.
[23] W. Liao, L. He, and K. M. Lepak. 2005. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Trans. Comput.-Aid. Des. Int. Circ. Syst.* 24, 7 (2005), 1042–1053.
[24] D. K. Nayak, S. Banna, S. K. Samal, and S. K. Lim. 2015. Power, performance, and cost comparisons of monolithic 3D ICs and TSV-based 3D ICs. In *Proceedings of the IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference*.
[25] S. Panth, K. Samadi, Y. Du, and S. K. Lim. 2014. Design and CAD methodologies for low power gate-level monolithic 3D ICs. In *Proceedings of the International Symposium on Low Power Electronics and Design*.
[26] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, and P. Schwarz. 2011. The kaldi speech recognition toolkit. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*.
[27] D. Su, X. Wu, and L. Xu. 2010. GMM-HMM acoustic model training by a two level procedure with gaussian components determined by automatic model selection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*.
[28] V. Sze, Y. Chen, J. Emer, A. Suleiman, and Z. Zhang. 2016. Hardware for machine learning: Challenges and opportunities. (2016). arXiv:1612.07625.
[29] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. 2016. The microsoft 2016 conversational speech recognition system. (2016). arXiv:1609.03528.
[30] S. Yin, G. Srivastava, S. K. Venkataramanaiah, C. Chakrabarti, V. Berisha, and J. Seo. 2018. Minimizing area and energy of deep learning hardware design using collective low precision and structured compression. (2018). arXiv:1804.07370.

Received December 2017; revised June 2018; accepted August 2018