Monolithic 3D IC Designs for Low-Power Deep Neural Networks Targeting Speech Recognition

Kyungwook Chang[1], Deepak Kadetotad[2], Yu Cao[2], Jae-sun Seo[2], and Sung Kyu Lim[1]
[1]School of ECE, Georgia Institute of Technology, Atlanta, GA
[2]School of ECEE, Arizona State University, Tempe, AZ
k.chang@gatech.edu, limsk@ece.gatech.edu

***Abstract*—In recent years, deep learning has become widespread for various real-world recognition tasks. In addition to recognition accuracy, energy efficiency is another grand challenge to enable local intelligence in edge devices. In this paper, we investigate the adoption of monolithic 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 energy efficiency, including DNN architectural choices, underlying workloads, and tier partitioning choices in M3D. Our post-layout M3D designs, together with hardware-efficient sparse algorithms, produce power savings beyond what can be achieved using conventional 2D ICs. Experimental results show that M3D offers 22.3% iso-performance power saving, convincingly demonstrating its entitlement as a solution for DNN ASICs. We further present architectural guidelines for M3D DNNs to maximize the power saving.**

I. Introduction

Deep neural networks (DNNs) have become ubiquitous in many machine learning applications, from speech recognition and natural language processing, to image recognition, and computer vision. Large neural network models have proven to be very powerful in all the stated cases, but implementing energy-efficient DNN ASIC is still challenging because (1) the required computations consume large amounts of 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 [1] for large-scale speech recognition tasks. This is impractical using only on-chip memory, 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 [2]. To mitigate this bottleneck, recent works have compressed the neural network weights and substantially reduced the amount of computation required to obtain the final output [3], [4], which becomes crucial for efficient DNN hardware implementations.

To further improve the energy-efficiency of compressed DNN designs, we adopt monolithic 3D IC (M3D) technology, which has shown its strength in reducing power consumption by effectively minimizing wirelength as well as congestion, especially in wire-dominated circuits. In M3D, transistors are fabricated onto multiple tiers, and the connections crossing the tiers are established by nano-scale monolithic inter-tier vias (MIVs) [5]. Owing to the minuscule size of MIVs ($<100nm$), M3D achieves orders of magnitude denser vertical integration with lower RC parasitics compared with through-silicon vias (TSVs). 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. An efficient CAD tool flow exists [6], and studies have demonstrated its performance and power improvements across multiple technology generations [7].

In this paper, for the first time, we investigate the benefit of M3D on DNN ASIC implementations and explore architectural and design decisions that impact its power consumption. We present two DNN architectures with different 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 paper 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, (3) feed-forward classification and pseudo-training workloads are examined thoroughly to investigate their impact on power reduction, and (4) we present key guidelines on optimal architecture and logic/memory design decisions for M3D ICs.

II. Deep Neural Network for Speech Recognition

A. Our DNN Topology

Starting from a fully-connected DNN, we adopt a Gaussian Mixture Model (GMM) for acoustic modeling [8]. Since it has been shown that DNNs in conjunction with Hidden Markov Models (HMMs) increase recognition accuracy [9], 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 coarse-grain sparsification (CGS) methodology presented in [4] in our DNN architecture to reduce the memory footprint as well as the computation for DNN classification.

As shown in Fig. 1, our DNN for speech recognition consists of 4 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 [10]. The Kaldi toolkit [11] is utilized for the transcription of the words and sentences for the particular set of phonemes.

B. 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 Eq. (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 i^{th} output node, and t_i is the i^{th} target value or label. The mini-batch stochastic gradient

978-1-5090-6023-8/17/$31.00

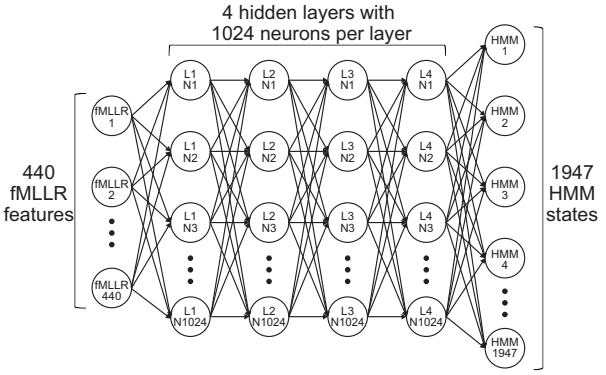

Fig. 1. Diagram of our DNN for speech recognition.

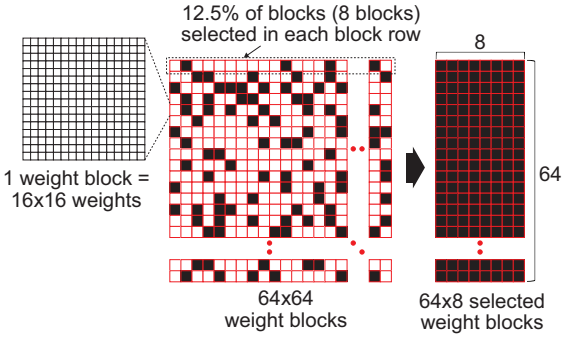

Fig. 2. 1024×1024 weight matrix is divided into 64×64 weight blocks with each weight block having 16×16 weights (i.e. block size of 16×16). 87.5% of weight blocks are dropped using coarse-grain sparsification (CGS). The remaining 12.5% weight blocks are stored in memory.

method [12] is used to update the weights. The weight W_{ij} is updated in the $(k+1)^{th}$ iteration using Eq. (2).

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

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:

$$\Delta W = \frac{\delta E}{\delta W}, \quad (3)$$

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 [13] is used for the non-linear activation function at the end of each hidden layer.

C. Coarse-Grain Sparsification (CGS)

To efficiently map sparse weight matrices to memory arrays, CGS methodology [4] is employed. In CGS, connections between two consecutive layers in a DNN are compressed in a block-wise manner. An example of block-wise weight compression is demonstrated in Fig. 2. 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.

In order 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 I.

TABLE I
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%

III. 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 design flow presented in [6]. 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. In order 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.

In order 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.

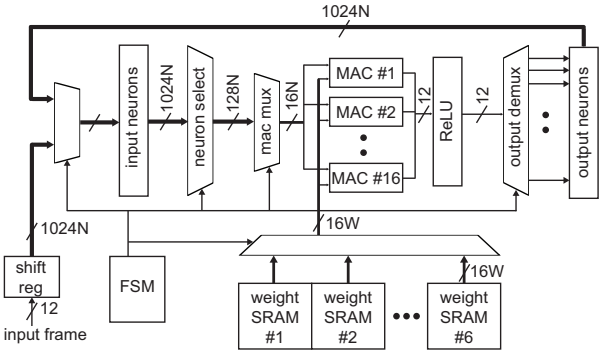

Fig. 3. Block diagram of the proposed CGS-based DNN architecture for speech recognition.

IV. DNN Architecture Description

The block diagram of our CGS-based DNN architecture is shown in Fig. 3. 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 II-B, 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 (Fig. 2). 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 architecture 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).

V. Circuit Design Discussions

To analyze the advantage of M3D on different DNN architectures, two DNN architectures (CGS-16 and CGS-64) are implemented using TSMC 28nm HPM technology with a target clock frequency of 400MHz, which is the highest achievable frequency of the design. 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. On the other hand, in the M3D-one design, all standard cells are placed on one tier, and only memory blocks exist on the other tier. Fig. 4 shows the GDS layouts of 2D and M3D designs.

A. Area, Wirelength, and Capacitance Comparisons

Several key metrics of the 2D and M3D designs are presented in Table II. We summarize our findings as follows:

- Footprint: our M3D-both designs achieve 50.1% footprint reduction compared with 2D designs, whereas 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 design of CGS-16, for example. These large memory blocks, if placed in the same tier, cause the footprint to increase significantly.
- 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 2D counterparts. Our savings in CGS-64 architecture are 8.2% and 14.3% for the cell count and area, respectively.
- 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.
- 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 VI-A) 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).
- 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 Fig. 5. In the M3D-both design shown in Fig. 5(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. On the other hand, we see that logic gates are rather spread out across the top tier in the M3D-one design shown in Fig. 5(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.

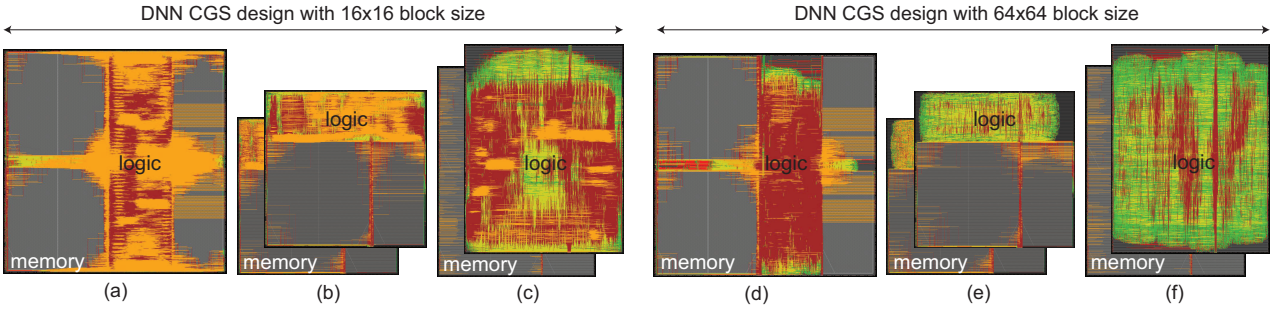

Fig. 4. 28nm full-chip GDSII layouts of DNN CGS-16 and CGS-64 architectures. (a) 2D IC design, (b) M3D design with memory blocks on both tier (M3D-both), (c) M3D design with memory blocks on a single tier (M3D-one), (d) 2D IC, (e) M3D-both, (f) M3D-one.

TABLE II
ISO-PERFORMANCE (400MHZ) COMPARISON OF DESIGN METRICS OF 2D AND M3D DESIGNS OF DNN CGS-16 AND DNN CGS-64 ARCHITECTURES. ALL PERCENTAGE VALUES SHOW THE REDUCTION FROM THEIR 2D COUNTERPARTS.

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

TABLE III
ISO-PERFORMANCE (400MHZ) POWER COMPARISON OF TWO ARCHITECTURES (CGS-16 VS. CGS-64) USING TWO WORKLOADS (CLASSIFICATION VS. PSEUDO-TRAINING). ALL PERCENTAGE VALUES SHOW THE REDUCTION FROM THEIR 2D COUNTERPARTS.

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

B. Power Comparisons

Table III 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 VI-B). From examining the power metrics of 2D designs only, we observe the following:

- 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 complicated weight selection process (to be further discussed in Section VI-A). 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, Eq. (4) is employed, which describes the components comprising dynamic power consumption.

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

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}.

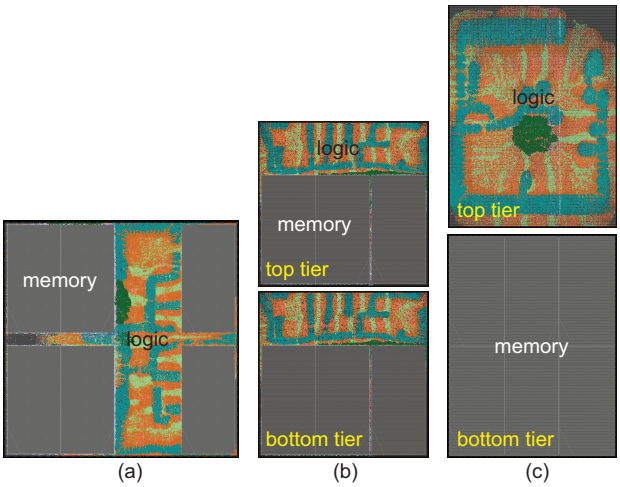

Fig. 5. Cell placement of the modules in CGS-16 architecture. (a) 2D, (b) M3D-both, (c) M3D-one. Each module is highlighted with different colors.

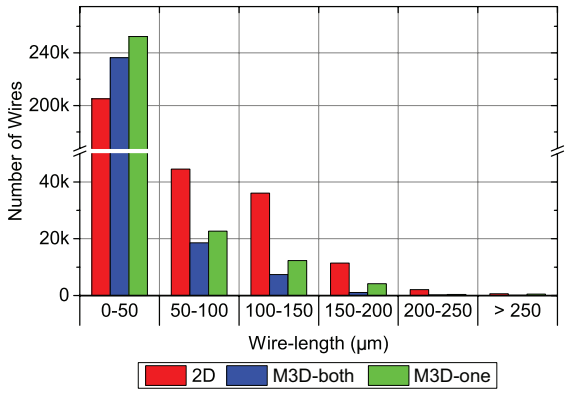

Fig. 6. Wirelength distribution of CGS-16 architecture.

The resulting footprint of M3D-both designs is reduced by half, thereby reducing the wirelength between the cells. Fig. 6 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 reduction of wire capacitance C_{wire} in Eq. (4), therefore the saving of P_{SW}. Fig. 7 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. $\times$0-$\times$0.8) and fewer high drive-strength cells (i.e. $\times$1-$\times$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 Eq. (4).

VI. Architectural Impact Discussions

A. CGS-16 vs. CGS-64 Architecture Comparisons

Table III shows that the total power reduction of M3D designs is higher in DNN CGS-16 architecture than CGS-64. This difference is caused by the granularity of weight selection methodology, i.e., coarse-grain sparsification (CGS) algorithm. The 1024$\times$1024 weight matrix is divided into 256 (= 16$\times$16) weight blocks in CGS-64 architecture. This count becomes 4,096 (= 64$\times$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. Fig. 8 shows the comparison of standard cell area of each module in CGS-16

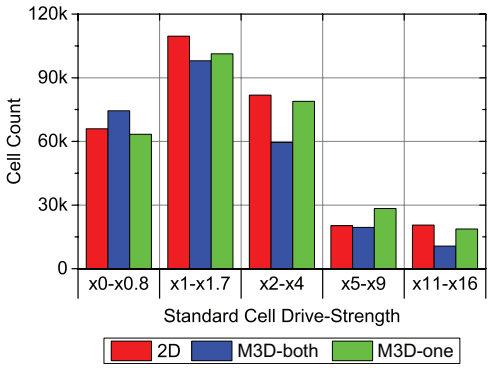

Fig. 7. Cell drive-strength distribution of CGS-16 architecture.

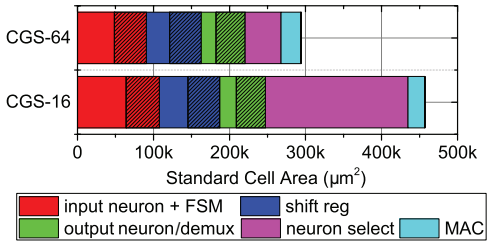

Fig. 8. Standard cell area breakdown of 2D CGS-16 and 2D CGS-64 architectures. Non-dashed and dashed boxes respectively indicates combinational and sequential elements. Only five largest modules are shown.

and CGS-64 architectures. We show both sequential (dashed box) and combinational logic (non-dashed box) portion in each module. We observe that the neuron selection unit in CGS-16 architecture (shown in purple) occupies more area than that in CGS-64 architecture.

As discussed in Section V-A, 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). On the other hand, 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.

Fig. 9 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, 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.

B. Impact of Workloads

In order to investigate the impact of different DNN workloads on M3D power reduction, we analyzed two main types of speech DNN workloads: feed-forward classification and training. Real-world test vectors are used for feed-forward classification. However, since our current architecture only supports offline training to avoid computational overhead of finding gradients, we create customized test vectors

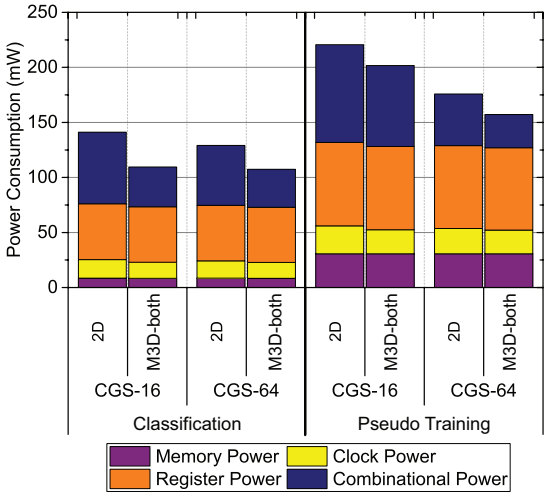

Fig. 9. Power breakdown under two architectures (CGS-16 vs. CGS-64), two workloads (classification vs. pseudo-training), and two designs (2D vs. M3D).

for "pseudo-training". There are two phases in our pseudo-training test vectors. 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.

Table III 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. On the other hand, 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 III.

As shown in Fig. 9, 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.

VII. 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.
- If memory blocks occupy a large area in DNN architectures, wisely tier partitioning memory blocks results in better footprint saving, which in turn maximize the total power saving.
- 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.
- In our DNN, it was combinational logic power, not the commonly believed memory power, that dominated the overall power saving. Moreover, compute-intensive classification workload gave us more power saving than memory-intensive training workload. 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 paper pave a road for practical and convincing studies with other DNN architectures and their ASIC implementations.

VIII. Conclusions

In this paper, 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 study convincingly demonstrates the low power benefits of M3D on DNN hardware implementations and offers architectural guidelines to maximize power saving.

References

[1] W. Xiong *et al.*, "The Microsoft 2016 Conversational Speech Recognition System," *arXiv preprint arXiv:1609.03528*, 2016.
[2] V. Sze *et al.*, "Hardware for Machine Learning: Challenges and Opportunities," *arXiv preprint arXiv:1612.07625*, 2016.
[3] S. Han *et al.*, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in *International Conference on Learning Representations (ICLR)*, 2016.
[4] D. Kadetotad *et al.*, "Efficient Memory Compression in Deep Neural Networks using Coarse-Grain Sparsification for Speech Applications," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2016.
[5] P. Batude *et al.*, "Advances in 3D CMOS Sequential Integration," in *Proc. IEEE Int. Electron Devices Meeting*, 2009, pp. 1–4.
[6] S. A. Panth *et al.*, "Design and CAD Methodologies for Low Power Gate-level Monolithic 3D ICs," in *Proc. Int. Symp. on Low Power Electronics and Design*, 2014.
[7] K. Chang *et al.*, "Match-making for Monolithic 3D IC: Finding the Right Technology Node," in *Proc. ACM Design Automation Conf.*, 2016.
[8] D. Su, X. Wu, and L. Xu, "GMM-HMM Acoustic Model Training by a Two Level Procedure with Gaussian Components Determined by Automatic Model Selection," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2010.
[9] L. Deng, G. Hinton, and B. Kingsbury, "New Types of Deep Neural Network Learning for Speech Recognition and Related Applications: An Overview," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2013.
[10] J. S. Garofolo *et al.*, "DARPA TIMIT Acoustic-Phonetic Continous Speech Corpus," *NASA STI/Recon Technical Report N*, 1993.
[11] D. Povey *et al.*, "The Kaldi Speech Recognition Toolkit," in *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.
[12] W. A. Gardner, "Learning Characteristics of Stochastic-Gradient-Descent Algorithms: A General Study, Analysis, and Critique," *Signal processing*, vol. 6, 1984.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012.