TAICHI: A Tiled Architecture for In-Memory Computing and Heterogeneous Integration

Xinxin Wang, *Graduate Student Member, IEEE*, Reid Pinkham, *Graduate Student Member, IEEE*, Mohammed A. Zidan, Fan-Hsuan Meng, *Graduate Student Member, IEEE*, Michael P. Flynn, *Fellow, IEEE*, Zhengya Zhang, *Senior Member, IEEE*, and Wei D. Lu, *Fellow, IEEE*

***Abstract*—We present TAICHI, a general in-memory computing deep neural network accelerator design based on RRAM crossbar arrays heterogeneously integrated with local arithmetic units and global co-processors to allow the system to efficiently map different models while maintaining high energy efficiency and throughput. A hierarchical mesh network-on-chip is implemented to facilitate communication among clusters in TAICHI to balance reconfigurability and efficiency. Detailed deployment of the different circuit components is discussed, and the system performance is estimated at several technology nodes. The heterogeneous design also allows the system to accommodate models larger than the on-chip storage capability.**

***Index Terms*—DNN accelerator, in-memory computing, heterogeneous architecture, tiled architecture, RRAM.**

I. Introduction

MACHINE learning (ML), represented by deep neural networks (DNNs), has become a major branch of artificial intelligence (AI). However, hardware implementations of DNN models have become a bottleneck, as conventional computing architectures are not well optimized for this application [1]. Different types of DNN hardware accelerators have been recently proposed [2]. Among them, in-memory computing (IMC) architectures utilizing emerging non-volatile memories such as resistive random-access memory (RRAM) have shown great promise, since they can minimize external memory access and circumvent the von Neumann bottleneck. In-memory computing (IMC) DNN accelerators based on a tiled architecture of RRAM crossbars [3]–[8] can theoretically offer high throughput and excellent energy efficiency for practical DNN models. Challenges such as finite tile size, device non-idealities and ADC quantization of the partial sums (Psums) can be addressed through tiling optimizations, column-wise quantization, and architecture-aware training techniques.

Manuscript received April 21, 2021; revised June 16, 2021; accepted July 03, 2021. Date of publication July 14, 2021; date of current version January 31, 2022. This work was supported in part by SRC and DARPA through the Applications Driving Architectures (ADA) Research Center, and in part by the National Science Foundation (NSF) under Award CCF-1900675. This brief was recommended by Associate Editor L. A. Camunas-Mesa. *(Xinxin Wang and Reid Pinkham contributed equally to this work.) (Corresponding author: Wei D. Lu.)*

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan at Ann Arbor, Ann Arbor, MI 48109 USA (e-mail: wluee@umich.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2021.3097035.

Digital Object Identifier 10.1109/TCSII.2021.3097035

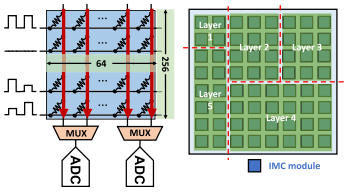

Fig. 1. Schematics of (a) an IMC module with ADC sharing among outputs and (b) mapping of DNN layers in the tiled architecture.

However, several questions remain to be answered. First, a single chip design needs to be able to map different models and maintain high efficiency. Prior studies mostly focus on the IMC module implementation while modern DNN models often contain irregular operations which are not optimal for IMC. Second, system-level implementations such as flexible and efficient data routing among the IMC blocks need to be carefully considered for a broad range of DNN models. Finally, as ML models are quickly evolving, the IMC DNN accelerator designed in present day may not be compatible with newer operators or may no longer be large enough to store all weights on chip as the models get updated/deployed in the future. Running out of space for the larger models in the fixed-size RRAM array is a significant concern for practical IMC accelerators. In this brief, we specifically address these issues in a heterogeneous architecture design. Through techniques such as optimally designed IMC modules, Hierarchical Mesh Network-on-Chip (HM-NoC), local Arithmetic Units (AUs), and an integrated Global Co-Processor (GCP), the proposed TAICHI architecture allows efficient mapping of different models and operators and alleviates memory capacity anxiety.

II. Proposed System

Following [3], DNN layers can be mapped onto multiple IMC modules, with each IMC module consisting of a RRAM crossbar array and multiple ADCs, as shown in Fig. 1. We assume the IMC module is based on 256×64 RRAM arrays. The small array size helps mitigate parasitic effects [9], although other array sizes can be readily adopted. For our analysis, we use 8-bit models however the results apply to other precisions. For ease of application, the 8-bit input activations are applied in a bit-serial manner using two voltage pulses, where each pulse represents a 4-bit value (16 levels). The RRAM array performs the MAC operations in the analog domain [10]. The current outputs at the columns are quantized through the 8-bit ADCs and shift-added from the two voltage inputs to produce an 8-bit output, which represents a Psum

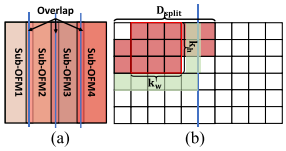

Fig. 2. (a) With multiple copies of weight arrays, the OFM can be split into several smaller sub-OFMs. (b) The data that should be stored in IA registers are marked as red.

TABLE I
SPECS OF IMC MODULES DESIGNED AT 65 NM TECHNOLOGY

Design		ADC sharing	Power (μW)	Area (μm^2)	Output frequency
Single ADC		-	40.0	3,000	Base (40 MHz)
Single RRAM array (256×64)		-	111	2,800	-
IMC module (RRAM +ADCs)	Type 1	8	334	26,800	Base / 16
	Type 2	32	83.5	8,800	Base / 64
	Type 3	128	20.9	4,300	Base / 256
	Type 4	256	5.22	3,550	Base / 1024

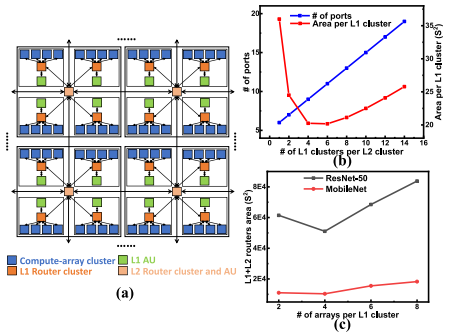

Fig. 3. (a) Hierarchical architecture of TAICHI. Two levels of clusters are connected via the HM-NoC. From analysis in (b-c), we assign 4 L1 clusters in 1 L2 cluster and 4 IMC modules in 1 L1 cluster to minimize router area.

for the output activations. An optimized ADC design at the 65 nm node operating at 40 MHz baseline frequency is used in our study. To balance the latency of different layers and minimize ADC area and power overhead, we consider 4 different IMC module designs (i.e., crossbar array + ADCs) in the system, ranging from 1 ADC shared with 8 columns (via time-multiplexing) to 1 ADC shared with all 256 columns of four arrays at half the baseline frequency. We define one output cycle as the time it takes for all columns shared by the ADC to be processed. Table I shows area and power of the IMC modules obtained from circuit simulations.

To maximize pipelined operation, the latencies of different layers needs to be balanced. Since the latency of each layer depends on the output feature map (OFM) size, layers with the same OFM dimensions are preferably mapped to the same type of IMC module. To speed up slower layers with larger OFMs, weights can be duplicated so that the OFM can be split to several smaller sub-OFMs to be processed in parallel, as shown in Fig. 2 (a).

The OFMs are stored in local L1 registers. When sufficient data in an OFM is produced, the next layer can start processing. Taking advantage of pipelining can significantly reduce the OFM storage size. Assuming the OFMs are computed row by row, the required register size for input activation (IA) is:

$$S_{IA} = D_{split} \cdot (k_h - 1) + k_w \quad (1)$$

where the D_{split} is the effective width of the OFM that is processed by a single copy of the weight array, and k_h and k_w are the convolutional (Conv) kernel height and width, as shown in Fig. 2 (b). To process the OFMs properly, overlaps between the sub-OFMs are also carefully considered.

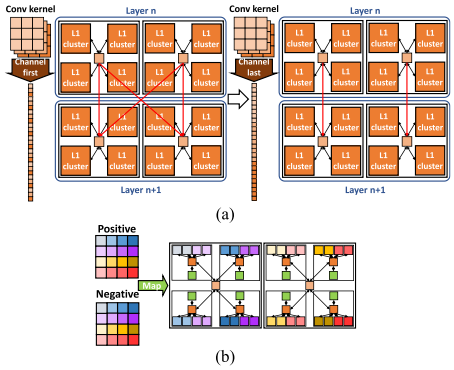

Fig. 4. Mapping methods to reduce data transfer between clusters: (a) channel-last vectorization of Conv kernels can avoid broadcasting outputs of one L2 cluster to all L2 clusters in the next layer. (b) Positive-negative array pairs should be mapped in the same L1 cluster, followed by arrays corresponding to the same output channels but receiving different inputs.

To investigate the design requirements and end-to-end performance of DNN models mapped on TAICHI, we chose ResNet-50 [11] and MobileNet [12] as examples. ResNet-50 is a popular large-scale network, while MobileNet is a smaller highly optimized model.

A. Hierichical Mesh Netowrk-on-Chip

The DNN models are mapped following the approach discussed in [3]. With tiled mapping and pipelining between the layers, the total NoC load for ResNet-50 and MobileNet are 67.6 GB/s and 27 GB/s, respectively. To efficiently handle the data movement, we use a HM-NoC [13] as depicted in Fig. 3 (a). Our HM-NoC uses two levels to partition the bandwidth. The level 1 (L1) router distributes data received from a level 2 (L2) router to the internal IMC modules, and sends Psum outputs from the IMC modules and arithmetic results from L1 AUs to the L2 router. Besides communicating with internal L1 clusters, an L2 router communicates with a local L2 AU and externally between adjacent L2 routers. The tradeoff between number of IMC modules per L1 cluster and router complexity is shown in Fig. 3 (b-c). Based on this tradeoff, we assign 4 IMC modules in one L1 cluster and 4 L1 clusters in one L2 cluster to minimize the router area.

To minimize bandwidth requirements, within one L1 cluster we avoid mapping weight arrays belonging to different layers, different output channels, or different copies of a layer, so that Psums within 1 L1 cluster can always be accumulated internally. As a result, only 64 B data (instead of up to 256 B) need to be transferred from one L1 cluster in every output cycle. This reduces the maximum bandwidth for both L1 and L2 clusters by 75% with minimal increase of the number of required L2 clusters (7.5% increase for ResNet-50 and 1.7% for MobileNet compared with maximum array utilization).

TAICHI uses multiple techniques to reduce HM-NoC bandwidth which are summarized in Fig. 4. When mapping Conv kernels, we use channel-last rather than channel-first vectorization to avoid broadcasting outputs to all L2 clusters in the next layer. Additionally, operations whose outputs are accumulated together are intentionally mapped in the same L1 cluster. These include weight arrays representing positive and negative weights when using dual array mapping [3], as well as arrays receiving different inputs. These approaches serve to minimize data movement between L2 and L1 routers.

Since the peak data movement for DNN models typically happens among the first several layers, we visualized these layers for MobileNet and ResNet-50 (Fig. 5) to check

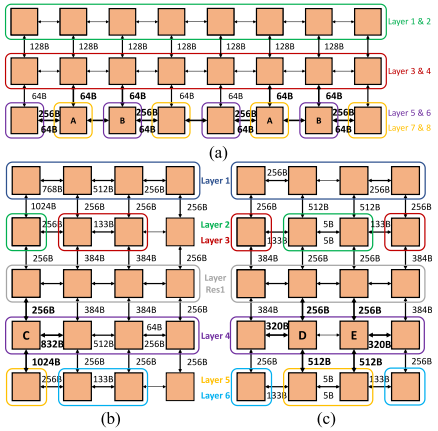

Fig. 5. (a, b) Mapping of the first 8 (6) layers of MobileNet (ResNet-50). Each square represents an L2 cluster. The numbers represent bytes of data to be ransfered per output cycle. (c) By splitting layer 2 and 5 in ResNet-50 into 2 L2 clusters each, the peak HM-NoC bandwidth can be reduced from 6.56 GB/s to 4.00 GB/s with minimal area utilization loss.

TABLE II
ENERGY COST OF 16 B DATA TRANSFER AND THE AREA OF THE HM-NOC

HM-NoC	65 nm	28 nm	16 nm
Energy/transfer (pJ)	6.40	6.40	6.40
Area/L2 router+wires (μm^2)	72,000	13,300	6,500
Area/L1 router+wires (μm^2)	32,000	5,600	2,900

actual bandwidth requirements for the HM-NoC. By splitting high data movement layers (e.g., splitting layers 2 and 5 in ResNet-50), the peak data transfer can be further reduced from 6.56 GB/s to 4.00 GB/s, with a minimal 1.3% loss in array utilization. The performance of the NoC is extrapolated from [14], as shown in Table II.

B. Arithmetic Units

Several studies have proposed to also process activations functions with memristor-based approach to maintain the whole process within the analog domain. However, these approaches are largely limited to simple datasets or spiking neural networks, and their applicability to large DNN applications is not proven yet [15]. The CMOS-based local AUs in L1 and L2 clusters are used to accumulate Psums, and process activation functions such as ReLU, Softmax, Tanh, etc. Additionally, we find it beneficial to also include augmented AUs in the design to perform functions which are not optimal for the IMC modules. Although IMC modules can process Conv and fully connected (FC) layers with very high efficiency, it has low efficiency for layers not based on vector-matrix multiplication (VMM). One example is the depthwise Conv (DW) layers in MobileNet. Since one kernel in a DW layer only operates on a single input channel, in an IMC system different kernels need to be mapped diagonally in crossbar to allow parallel processing. When mapped onto the IMC, the DW layers result in low area utilization, and thus ADC power. Fundamentally, this is because, without batching, the DW layer uses independent vector-vector multiplications (VVMs) instead of VMM.

For example, in [3], the DW layers in MobileNet consume 88.2% of the array area and 86.8% of the ADCs, while only performing 3.06% of MACs and using 1.06% of parameters. Denser mapping of the DW layer can be achieved by considering ADC sharing since the outputs are computed sequentially already in the shared ADC case. However, the area utilization is still low compared with other layers, with the DW layers of MobileNet occupying 26.2% of the crossbar arrays and 42.9% of the ADCs.

TABLE III
AU DESIGNS FOR THE 4 IMC MODULES

IMC module type	Normal AU	Special AU
1	16 Adders	64 VMs, 16 Adders
2	4 Adders	16 VMs, 4 Adders
3	1 Adder	4 VMs, 1 Adder
4	1 Adder (1/4 duty cycle)	1 VM, 1Adder

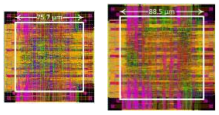

Fig. 6. Layout views of (a) a normal and (b) an augmented AU with 1 VM at 28 nm.

TABLE IV
COMPARISON AMONG DIFFERENT MAPPING OF DW LAYERS

DW layers	# of IMCs	# of ADCs	Power (W)
Diagonal mapping	9072	17856	2.478
Dense mapping	432	2040	0.328
AU mapping	2	0	0.280

Since DW layers are not as compute-intensive as Conv and FC layers, we instead propose to map them onto AUs augmented with vector multiplication units (VMs) which are only included in a small portion of L2 clusters. The augmented AUs are only used by selected layers such as the DW layers which represent a small fraction of the compute in DNNs. Our synthesized designs thus include both the baseline AUs for all L1 clusters and most L2 clusters, and augmented AUs with VMs for some L2 clusters. The AU configurations are summarized in Table III. Layout views of a normal AU and an augmented AU with 1 VM are shown in Fig. 6. AUs with different number of adders and VMs are designed to balance the load of the four IMC modules used in TAICHI.

Comparison of the simple diagonal mapping, dense mapping, and AU mapping for DW layers is shown in Table IV. The AU mapping does not need IMC modules, and consumes 17.2% less power than the dense mapping. Moreover, the model accuracy is found to strongly depend on the precision of the Psums of the DW layers. When the Psums are computed with the IMC modules, 8 different dynamic ranges are required for the 8-bit ADCs to prevent the accuracy of MobileNet from dropping below 60% for ImageNet classification [3]. Additionally, the lower utilization of the crossbar arrays means the output currents associated with the DW layers will be much lower in value than those of Conv and FC layers, requiring ADCs with lower input ranges. Utilizing AUs to perform DW eliminates these outlier ADC designs and improves accuracy with only 4 different ADC dynamic ranges for the system. Using AUs to perform these special operations can significantly improve both throughput and accuracy of the system.

III. GENERAL CHIP DESIGN AND MAPPING RULES

As a general DNN accelerator, the design needs to support a wide range of models. Here, we analyze Inception-v4 [16] and Transformer [17] as additional examples. Again, during model mapping, we make copies of the weight arrays for the slow layers and share ADC with more columns for the fast layers to balance latency in the layers to ensure a smooth pipeline. The 4 different IMC modules offered in the system allow the mapping to achieve area and latency balance.

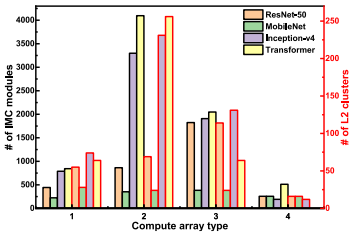

Fig. 7. Number of the IMC modules and L2 clusters used by the four example models when mapped on TAICHI.

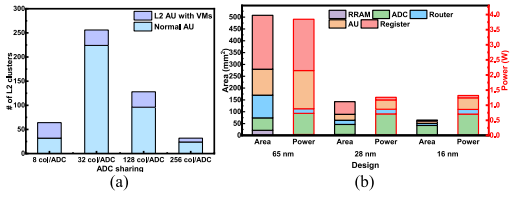

Fig. 8. (a) Number of L2 clusters assigned with normal and augmented L2 AUs in the proposed TAICHI chip design. (b) Area and power breakdown of TAICHI at 65 nm, 28 nm, and 16 nm.

TABLE V
ESTIMATED PERFORMANCE OF TAICHI AT 65 NM, 28 NM AND 16 NM

Design	Weight Memory (MB)	Register (MB)	Total area (mm^2)	Power (W)	TOPS/W
65 nm	62.91	2.81	496.58	3.85	23.32
28 nm	62.91	2.81	120.53	1.14	78.60
16 nm	62.91	2.81	64.04	1.04	87.91
PUMA [7]	-	-	90.6	62.5	0.84
ISAAC [8]	-	-	85.4	65.8	0.64

Fig. 7 shows the number of IMC modules and L2 clusters required by different models. We find that a general design consisting of 64, 256, 128 and 32 L2 clusters for the four IMC module types, respectively, work well for these models. The estimated area, memory capacity, power, and throughput of the chip, as well as the comparison between TAICHI and the state-of-art designs [7], [8] are summarized in Fig. 8 and Table V. Here, OFMs, Psums, ADC scales and L1 AU results are stored in L1 registers, while L2 AU results, and DW weights if the L2 AU contains VMs, are stored in the L2 registers.

At 65 nm, the power and area are dominated by the AUs and registers, where the optimized IMC module designs with ADC sharing to match the load reduces the power and area overhead of ADCs. At 28 nm and 16 nm, power and area become increasingly dominated by ADCs as the performance of the digital components improves with technology scaling.

When mapping a DNN model on TAICHI, the compute/weight ratio largely dictates the architecture efficiency. For each layer in a DNN model, we have:

$$N_{MAC} = N_W \cdot OFM_{size} \tag{2}$$

where OFM_{size} is the size of the OFM, N_{MAC} and N_W denote the number of MAC operations and weights in the layer, respectively. The compute/weight ratio is N_{MAC}/N_W. In general, layers with the same N_{MAC}/N_W, i.e., the same OFM size from (2), are preferred to be mapped in the same type of IMC module. Similarly, shallower layers, which usually have a higher compute/weight ratio, will have higher OFM size and should be mapped onto IMC modules with less ADC sharing (i.e., faster modules). If a particular module type runs out, additional weight arrays are allocated based on availability and latency requirements. For example, the FC layers in Transformer should in principle all be mapped in the same module type to balance pipelining. However, since none of the module types offers enough storage to map all the FC layers, the layers need to be split across different module types. For example, if the split is made to slower modules, lower area utilization is necessary to balance the latency requirements.

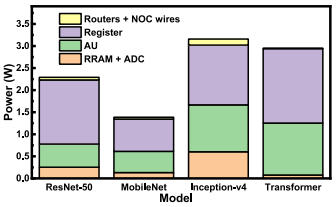

Fig. 9. Power breakdown of four models on 65 nm TAICHI architecture.

TABLE VI
PERFORMANCE OF FOUR MODELS ON 65 NM TAICHI ARCHITECTURE

Performance	ResNet-50	MobileNet	Inception-v4	Transformer
FPS	3189	6378	2041	39063
FPS/W	1391	4602	646	12911

By analyzing the compute/weight ratio of different layers and following the mapping methods discussed in Section II and [3], the 4 example models can be directly mapped on TAICHI. The partition among different compute modules affects the throughput and energy efficiency, but will not affect the accuracy since all 4 types of compute modules have the same crossbar array size, with the only difference being how many cycles it takes to get the access to the ADC to generate the Psum. The power breakdown and the estimated performance metrics are shown in Fig. 9 and Table VI. At 65 nm, TAICHI can attain many thousands of frames per second (FPS) across the tested networks, while maintaining flexibility to fit and run diverse network models.

IV. GCP AND HYBRID MAPPING

Although TAICHI can support many popular DNN models, it is possible that the size of future models can outgrow the storage capacity of the on-chip memory after the initial hardware deployment. To address this, we propose to process these too-large-to-fit DNN models in a hybrid fashion using an integrated GCP. To maintain high throughput, function partitioning needs to be carefully analyzed.

To quantitatively decide which layers should be mapped onto the GCP when hybrid mapping is necessary, we compare power and area in the hybrid system with those in a hypothetical large-enough IMC system. We estimate the upper limit of power and area of each layer for the two approaches. The upper limit of power for one L2 cluster is:

$$P_{IMC} = P_{L2,i} \cdot \frac{N_W \cdot 2}{cb_{size} \cdot 16} \cdot \frac{OFM_{size}}{OFM_{size,base,i}} = M_{p1,i} \cdot N_{MAC} \tag{3}$$

where $P_{L2,i}$ is the power of one L2 cluster for a given module type (i), including all the components within it, cb_{size} is the size of a crossbar array, $OFM_{size,base,i}$ is the size of the OFM that one module in type (i) can support. The parameters can be lumped into an module-dependent parameter $M_{p1,i}$ as shown in (3). The power of a single GCP is denoted as P_G, and the number of MACs it can perform per cycle as $N_{MCA,G}$. Since the GCP needs external DRAM access, the power of DRAM is denoted as P_D, the bandwidth of the DDR3 controller as BW_D, and the latency of one layer as T_L. Power to perform one layer in the GCP is then obtained as following:

$$P_{GCP} = \frac{P_G \cdot N_{MAC}}{N_{MAC,G}} + \frac{P_D \cdot N_W}{BW_G \cdot T_L} = M_{p2} \cdot N_{MAC} + M_{p3} \cdot N_W \tag{4}$$

TABLE VII
AREA AND POWER OF THE GCP DESIGN WITH 32 VMS

Design (65nm)	After APR		
	Area (μm^2)	Power (mW)	Idle Power (mW)
GCP with 32 VMs	609,804	219	16.9

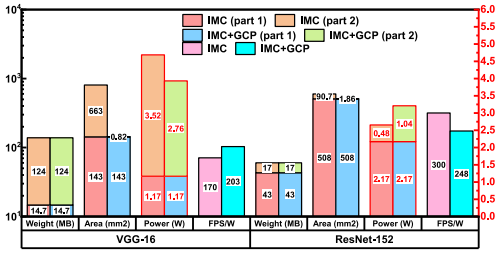

Fig. 10. Performance comparison of VGG-16 and ResNet-152 on the hybrid system consisting of a standard TAICHI and an integrated GCP, against a hypothetical IMC system required for much larger memory capacity.

Thus, the power ratio of one layer in GCP and in IMC is:

$$R_{power} = \frac{P_{GCP}}{P_{IMC}} = \frac{M_{p2}}{M_{p1,i}} + \frac{M_{p3}}{M_{p1,i} \cdot OFM_{size}} \quad (5)$$

Similarly, the area ratio is:

$$R_{area} = \frac{A_{GCP}}{A_{IMC}} = \frac{M_{a2}}{M_{a1,i}} + \frac{M_{a3}}{M_{a1,i} \cdot OFM_{size}} \quad (6)$$

For the GCP, the power parameters M_{p2}, M_{p3} and area parameters M_{a2}, M_{a3} remain mostly constant for all the layers. For IMC, $OFM_{size,base,i}$ decreases quadratically, while $P_{L2,i}$ and $A_{L2,i}$ only decrease sub-linearly as we move to modules with higher ADC sharing. Thus, $M_{p1,i}$ and $M_{a1,i}$ increase following (3) and (4), leading to a decrease in R_{power} and R_{area} for layers implemented in slower IMC (more ADC sharing).

Since the slower IMC modules are normally used to map layers with low compute/weight ratio (typically for deeper layers in a model), these layers may be mapped onto the GCP if hybrid mapping is necessary. In other words, the IMC system is most efficient when accelerating compute bound DNN layers. This conclusion is consistent with results in [18]. For memory-bound layers the benefits of IMC need to be balanced with the area demand of the high storage capacity requirement. The large weight storage requirement and the relatively low compute/weight ratio mean these layers may be delegated to the GCP when the on-chip memory runs out of space for model mapping.

Following this principle, we design a hybrid TAICHI system with a GCP consisting of 32 VMs capable of 512 MAC operations per cycle at 500 MHz, shown in Table VII. To verify the hybrid mapping concept, we test VGG-16 [19] and ResNet-152 [11], both of which exceed the storage capacity of the standard TAICHI design in Table V. For VGG-16, the three FC layers contain only 0.8% of the total MAC operations but would occupy 82.3% of the total area if fully mapped on a hypothetical IMC system. It is thus more economical to compute these FC layers with the GCP. For ResNet-152 the standard TAICHI can accommodate the first 142 layers and will run out of space for the last 10 layers. Thus, we map FC layer of VGG-16 and the last 10 layers of ResNet-152 onto the GCP and the other layers onto IMC modules in the hybrid TAICHI. The performance of VGG-16 and ResNet-152 using this hybrid system are summarized in Fig. 10. For comparison, the full models are also mapped onto a hypothetical large enough IMC accelerator. For VGG-16, the hybrid system shows both throughput increase and area saving. For ResNet-152, the hybrid system shows a moderate drop in throughput and energy efficiency, but offers reasonable area savings while eliminating memory capacity concerns for the IMC-only system.

V. CONCLUSION

By taking advantage of the tradeoffs between compute/weight ratio and throughput in typical neural networks, our general DNN accelerator architecture, TAICHI, offers high throughput and energy efficiency for a broad range of DNN models using the same chip design. AUs with simple VMs are added to process functions that are not optimal for parallel in-memory computing and to increase flexibility of the architecture. TAICHI's performance is verified using four popular models, showing the generality of the design. A hybrid IMC+GCP system is further analyzed to allow models too large to fit on an IMC chip to still be processed efficiently to mitigate the "memory capacity anxiety" problem.

REFERENCES

[1] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.
[2] Y. R. Chen, Y. Xie, L. H. Song, F. Chen, and T. Q. Tang, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, no. 3, pp. 264–274, Mar. 2020.
[3] X. X. Wang, Q. W. Wang, S. H. Lee, F.-H. Meng, and W. D. Lu, "A deep neural network accelerator based on tiled RRAM architecture," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, 2019, pp. 1–4.
[4] X. C. Peng, S. S. Huang, Y. D. Luo, X. Y. Sun, and S. M. Yu, "DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, 2019, pp. 1–4.
[5] A. Nag *et al.*, "Newton: Gravitating towards the physical limits of crossbar acceleration," *IEEE MICRO*, vol. 38, no. 5, pp. 41–49, Sep./Oct. 2018.
[6] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, 2016, pp. 27–39.
[7] A. Ankit *et al.*, "PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, 2019, pp. 715–731.
[8] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, 2016, pp. 14–26.
[9] T. Zanotti, F. M. Puglisi, and P. Pavan, "Circuit reliability analysis of RRAM-based logic-in-memory crossbar architectures including line parasitic effects, variability, and random telegraph noise," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Dallas, TX, USA, 2020, pp. 1–5.
[10] M. A. Zidan *et al.*, "A general memristor-based partial differential equation solver," *Nat. Electron.*, vol. 1, no. 7, pp. 411–420, Jul. 2018.
[11] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.
[12] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: arXiv:1704.04861.
[13] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.
[14] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," *ACM Trans. Archit. Code Optim.*, vol. 7, no. 1, pp. 1–28, Apr. 2010.
[15] Z. R. Wang *et al.*, "Fully memristive neural networks for pattern classification with unsupervised learning," *Nat. Electron.*, vol. 1, no. 2, pp. 137–145, Feb. 2018.
[16] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 4278–4284.
[17] A. Vaswani *et al.*, "Attention is all you need," *Advances in Neural Information Processing Systems 30 (NIPS)*, vol. 30. Red Hook, NY, USA: Curran Assoc., Inc., 2017.
[18] S. Q. Wang, A. Pathania, and T. Mitra, "Neural network inference on mobile SoCs," *IEEE Design Test*, vol. 37, no. 5, pp. 50–57, Oct. 2020.
[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arXiv:1409.1556.