REDRAW: Fast and Efficient Hardware Accelerator with Reduced Reads And Writes for 3D UNet

Tom Glint
IIT Gandhinagar, India
tom.issac@iitgn.ac.in

Manu Awasthi
Ashoka University, India
manu.awasthi@ashoka.edu.in

Joycee Mekie
IIT Gandhinagar, India
joycee@iitgn.ac.in

Abstract—**Hardware Accelerators (HAs) proposed so far have been designed with a focus on 2D Convolutional Neural Networks (CNNs) and 3D CNNs using temporal data. To the best of our knowledge, there is no existing HA for 3D CNNs using spatial data. 3D UNet is a 3D CNN with significant applications in the medical domain. However, the total on-chip buffer size (>20 MB) required for the complete stationery approach of processing 3D UNet is cost prohibitive. In this work, we analyze the 3D UNet workload and propose a HA with an optimized memory hierarchy with a total on-chip buffer of less than 4 MB while conceding near theoretical minimum memory accesses required for processing 3D UNet. We demonstrate the efficiency of the proposed HA by comparing it with SOTA Simba architecture with the same number of MAC Units and show a 1.3× increase in TOPS/watt for an ISO-area design. Further, we revise the proposed architecture to increase the ratio of compute operations to memory operations and to meet the latency requirement of 3D UNet-based embedded applications. The revised architecture, compared against a dual instance of Simba, has similar latency. Against the dual instance of Simba, the proposed architecture achieves a 1.8× increase in TOPS/watt in a similar area.**

Index Terms—**DNN Accelerator, 3D UNet, Modeling**

I. Introduction

Convolutional Neural Networks (CNN) have shown tremendous applications in the field of computer vision. Typically, 2D CNNs are used for vision tasks such as image segmentation and classification [1], [2]. In 3D CNNs, each input channel has a depth dimension apart from width and height. Traditionally, the depth dimension holds temporal data, like successive frames of a video, and consequently, they have data redundancy along this dimension [3]. Acceleration of such 3D CNN workloads using conventional processors, coprocessors, GPUs, FPGAs, and ASIC accelerators have been explored in past works, which have exploited temporal data redundancy for efficient execution [3], [4]. However, in 3D UNet [5], the depth dimension represents spatial data. 3D UNet has multiple applications in the field of medicine, like tumor identification and segmentation. For example, 3D UNet is used for dosage calculation during radiation therapy [6], [7], which makes this workload an edge application with a single stream of input. 3D UNet has also been incorporated into the MLPerf [2], [8] Inference benchmark suite. 3D UNet is computationally intensive and has 7.48 trillion algorithmic multiply-and-accumulate (MAC) operations, as shown in Table I. Hence, inference edge applications based on 3D UNet require acceleration [2], [4], [8]. Since depth channels in 3D UNet are spatial data, previous acceleration techniques based on temporal data redundancy are not applicable.

The challenges to designing a fast and efficient accelerator for 3D UNet are numerous. First, the SOTA DNN accelerator design methodology uses architectural space search, which uses workload specification as one of the inputs. However, due to the large size and extra-dimensional shape of 3D UNet, the search space is larger than 2^{128}, making the search intractable [9], [10]. Second, due to the nature of the application, 32-bit number formats have to be used instead of 8-bit numbers to preserve accuracy [11]. This significantly increases the compute unit's energy and the size of the on-chip buffers. Third, similar to quantization techniques, approximation techniques are not a viable option since they lead to accuracy degradation. Fourth, the input, output, or filters must be kept fully inside the accelerator chip for efficient computation of layers of the workload to implement stationary mapping, which in turn minimizes costly DRAM accesses. However, the on-chip memory of 2D Hardware Accelerators (HA) is limited to a few hundred kilobytes, while the input volume and filter volume are in the order of 100s of megabytes for 3D UNet, as shown in Table I. Due to this, partial products have to be stored off-chip and accumulated on-chip by re-fetching [4]. Finally, due to the large word width, buffer access energy is also quadrupled compared to typical HAs, and therefore, the updates to the on-chip buffer need to be minimized. Table I shows the layer shape and data reuse (number of time a data word participate in computation) of each layer of 3D UNet.

The acceleration of 3D UNet has not been explored for HAs, and previous approaches have been limited to coprocessors or GPU-based approaches [3], [8]. In this work, we perform a layer-by-layer analysis of 3D UNet and find capacity and bandwidth constraints for designing a more efficient HA than the SOTA CNN accelerator for edge applications. We identify the smallest on-chip buffer size (~3.5 MB) for achieving near theoretical external memory access for 3D UNet. Further, we propose a unified architecture for reducing reads and writes to the buffer by exploiting the size of related entities in computation (shape of the workload, input and output channel sizes).

This work is supported through grants received from Science and Engineering Research Board (SERB), Government of India, under SERB-CRG grant CRG/2018/005013, SERB-MATRICS grant MTR/2019/001605, and SERB-SUPRA grant SPR/2020/000450, and funds received for YFRF Visvesvaraya PhD fellowship from MEITY and Semiconductor Research Corporation (SRC) through contracts 2020-IR-3005 and 2020-IR-2980, and is partially supported through Ashoka University startup and Huawei Technologies India grants.

TABLE I
INDEX OF UNIQUE LAYERS IN 3D UNET

Layer Index	Input Shape (Width x Height x Depth x Channels)	Filter Shape	Input Reuse	Filter Reuse	Filter Size (words)
1	224x224x160x32	3x3x3x64	1728	8028160	55296
2	112x112x80x64	3x3x3x64	1728	1003520	110592
3	112x112x80x64	3x3x3x128	3456	1003520	221184
4	56x56x40x128	3x3x3x128	3456	125440	442368
5	56x56x40x128	3x3x3x256	6912	125440	884736
6	28x28x20x256	3x3x3x256	6912	15680	1769472
7	28x28x20x256	3x3x3x512	13824	15680	3538944
8	28x28x20x512	2x2x2x512	4096	15680	2097152
9	56x56x40x768	3x3x3x256	6912	125440	5308416
10	56x56x40x256	3x3x3x256	6912	125440	1769472
11	56x56x40x256	2x2x2x256	2048	125440	524288
12	112x112x80x384	3x3x3x128	3456	1003520	1327104
13	112x112x80x128	3x3x3x128	3456	1003520	442368
14	112x112x80x128	2x2x2x128	1024	1003520	131072
15	224x224x160x192	3x3x3x64	1728	8028160	331776
16	224x224x160x64	3x3x3x64	1728	8028160	110592

We further optimize the Multiply and Accumulate (MAC) unit for low latency and low energy for 32-bit computation. The proposed architecture with ISO-area as SOTA achieves 1.35× TOPS per watt. Further, we extend the architecture to achieve the same throughput as two instances of SOTA while increasing the energy efficiency by 1.5× TOPS per watt and 1.8× as compared to SOTA and 2× instance of SOTA, respectively (with only 10% area overhead).

II. Background

A. 3D UNet

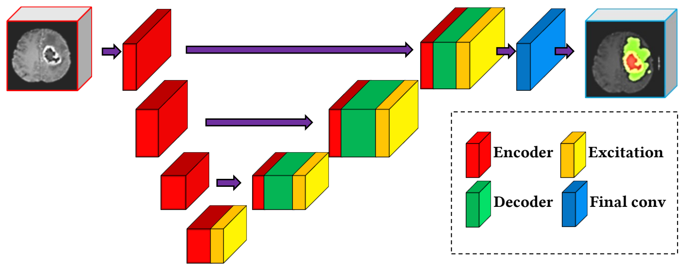

Fig. 1. 3D UNet with concatenation and up-convolution

3D UNet has very high accuracy in identifying and segmenting anomalies in medical data such as MRI data [6], [7], [12]. As shown in Fig. 1, the input to 3D UNet is a collection of voxels that stores spatial data. They are represented as multiple channels of 3D matrices, as detailed in Table I. In 3D convolution operation, the kernel is 3D in nature and is slid along the depth dimension apart from the width and height. Due to this sweep, a large number of MAC operations (~7 trillion) have to be performed for inference. Traditional processors will take multiple minutes to process an input for 3D UNet, while the SOTA GPU with a power draw of 400 W and 826 mm^2 chip area will take ~2 second [8], [13]; making both these approaches infeasible and cost prohibitive for edge applications. Further, the information stored in these voxels is not temporal and is spatial, reducing opportunities for compression.

III. Current SOTA for Spatial Workloads on Edge

Simba [14] is the state-of-the-art and data-agnostic Hardware Accelerator (HA) architecture for CNN-based inferencing on edge. Simba is very efficient in inferencing 2D Convolutional Neural Network (CNN) workloads compared to conventional processors, coprocessors, FPGAs, and GPUs. Table II shows the comparison of Simba with other edge accelerators. We note that there are other accelerators with higher performance per area, but they employ approximation techniques [15], which is not desired for 3D UNet.

TABLE II
COMPARISON OF OTHER EDGE INFERENCE ACCELERATOR FOR PEAK PERFORMANCE

	ENVISION [16]	STICKER [17]	UNPU [18]	Exynos [19]	Simba
Technology	28nm	65nm	65nm	8nm	16nm
Core Area	1.87 mm^2	7.8 mm^2	13 mm^2	5.5 mm^2	3.1 mm^2
Precision	4-16b	8b	1-16b	8b,16b	8b
MACs per cycle	512@8b	256	1728@8b	1024	1024
Performance (TOPS)	~0.15@8b	0.1	0.69@8b	1.91	2.05

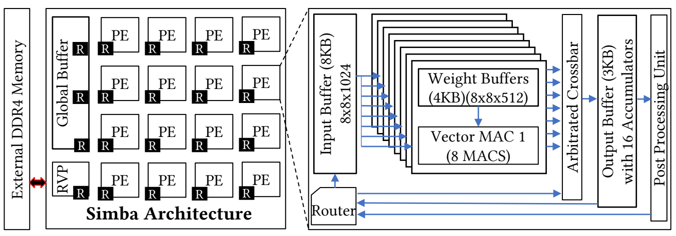

Fig. 2. Left Top: Simba Architecture [14] connected to an external memory. The architecture consists of 16 Processing Elements (PE) connected with each other and the global buffer over a Network-on-Chip (NoC). Right Top: shows the internal components of a PE

As shown in Fig. 2, each Simba chip has a global buffer, 16 Processing Elements (PE) arranged in mesh Topology and a RISC-V processor, and is connected to an external memory, which stores the inputs, outputs and weights used in CLs and FCLs. The RISC-V processor coordinates the flow and execution of data to each PE. Each PE has separate buffers for input, weight, and partial sums. As shown in Fig. 2, eight input words are commonly shared across eight vector MAC units. Each vector MAC unit has eight multipliers whose products are summed up into a single word using an adder tree and stored at the accumulation buffer after adding to the previous partial sum. Each vector MAC unit has a dedicated weight buffer. Inputs and weights of each layer are fetched from the external memory and buffered at various levels for optimum data reuse. The final outputs of each layer are written back to memory.

IV. Proposal

Embedded systems such as field scanners augmented with Machine Learning algorithms can significantly increase patient outcomes [6], [7]. For adopting and employing 3D UNet in such scenarios, we require a DNN accelerator for 3D UNet which can process an input under 5 seconds [12] and draw less than 50 W of power [20]. Three significant challenges need to be overcome to meet these operational constraints in the context of processing 3D UNet with accelerators. First, external DRAM access is energy-intensive, so access to external memory must be minimized. Second, due to the 32-bit number used in computation, internal buffers are large, and access energy is high; therefore, reads and writes to the internal buffer must be minimized. Finally, 32-bit multiplication is energy intensive at low latencies, and therefore, techniques to increase compute throughput without increasing energy have to be developed at the Processing Element. In this work, we propose a DNN accelerator architecture that tackles these three challenges in the following manner. First, we analyze the 3D UNet workload and identify the mapping of workload to the hardware such that access to the DRAM is near the theoretical minimum while only allocating $1/6^{th}$ the buffer size required to perform stationary mapping approaches. This results in a total chip area of only 59 mm^2 in 45 nm technology. Second, we identify compute patterns that can be combined together to reduce reads and writes to the internal buffers. Finally, we propose a dual-MAC architecture that uses multicycle 32-bit multipliers for low-energy computation inside PEs while maintaining high throughput. A detailed analysis is presented in Section VI. Further, we compare the proposed architecture with SOTA DNN accelerator Simba and show component-wise improvement.

V. Experimental Setup and Modeling

The experimental framework used for a fair and detailed comparison has two aspects: First, the baseline hardware (Simba) and the proposed hardware are modeled, and its hardware-bound aspects are captured using a framework made up of Timeloop [21], Accelergy [22], and Cacti [23]. The parameters for multipliers and adders are obtained from Cadence Genus Synthesis at the 45 nm technology node. Second, the optimal workload mapping (for least latency and energy) is found using Timeloop-mapper for each hardware, rather than a static mapping policy, for a fair comparison of accelerators. The scaled (8-bit to 32-bit conversion) configuration of the Simba baseline used in this work is given in Table III.

TABLE III
Specification of Simba (baseline)

Property	Value (baseline)
External memory used	DDR4 DRAM
Baseline Architecture	Simba
Bandwidth to DRAM from chip	25 GBps
Access energy to DRAM from logic die	46 pJ/bit
Global buffer size	128 KB
No. of PEs	16
MAC Units per PE	64
PE weight buffer size	128 KB
PE input buffer size	32 KB
PE output buffer size	12 KB
MAC latency	979 ps
Word size	32 Bit float

A. Hardware model and optimal mapping

The model in this work captures the spatial organization of a DNN accelerator, including the hierarchy of different memory like DRAM, SRAM buffers, registers, and the placement of MAC units in relation to the memory hierarchy. Further, it captures the bandwidth available for communication between each unit. The model accumulates every event during the inference and calculates energy values at the granularity of an event.

B. Workloads

This work primarily focuses on the layer-by-layer data-agnostic processing of 3D UNet during its inference. The compute precision is 32-bit float. This work does not consider HW/SW codesign techniques which usually alter the data, and since 3D UNet is used for medical applications, changes in accuracy introduced by these methods are not acceptable [11].

C. Timeloop Mapper

It identifies all possible permutations of mapping a workload to a given architecture and calculates the latency and energy associated with each mapping based on Timeloop and Accelergy models. Further, our mappings are optimized for least latency followed by least energy.

VI. Methodology

Architectural search using workload definition is not tractable for 3D UNet due to its size and complexity [9], [10]. Hence, approaches to finding bottlenecks and opportunities are utilized to obtain efficient architecture.

A. Reads and Writes to buffers

A MAC operation translates to 3 read operations from and 1 write operation to the buffer. Identifying and combining computations that share data can reduce these reads and writes. From Table I, it can be observed that a single input channel participates in the production of N$\times$32 output channels for some value of N. Hence, a single SRAM read can be shared and fed to 32 multipliers resulting in the generation of 32 output channels. Similarly, M$\times$32 input channels result in the generation of one output channel, for some value of M. So, results of multiplication that align in the depth dimension of the input can be accumulated together using an adder tree. This reduces the number of updates to the partial sum in the output buffer to 1:32 compared to no alignment. Further, this combination reduces the area for peripheral circuitry of buffer structures.

B. Low energy multiplication

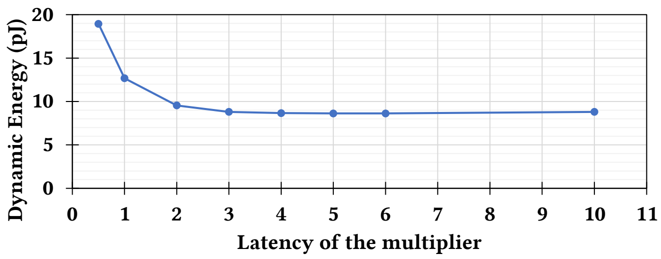

Fig. 3. Energy vs. latency of 32-bit floating point multiplier

Implementations of SOTA DNN accelerators operate between 1 GHz to 2 GHz. However, they operate on 8-bit numbers and thus have low energy consumption. However, for maintaining high precision, 3D UNet requires 32-bit float point operations [11]. Increasing the bit-width of the number representation increases energy in a quadratic manner. Nevertheless, while processing DNNs, sequential multiplication does not use the product from previous multiplication as an input; therefore, using multiple, multi-cycle multipliers could have energy benefits while maintaining high frequency and throughput.

Fig. 3 shows the energy associated with 32-bit multipliers operating at different latencies. It can be observed that a multiplier operating with a latency of 2 ns (500 MHz) uses much lower energy than a multiplier operating at 1 ns (1 GHz).

Any further increase in latency does not provide energy benefits but would significantly increase the design area required for high throughput similar to 1 GHz frequency. The 2 GHz multiplier is not considered in this work as it is a multi-stage design and has very high dynamic energy, which is unsuitable for edge applications considered in this work.

C. *Identifying internal buffer capacity*

From Table I, the smallest volume among inputs, weights and outputs are weights. For layer-by-layer processing using stationary approaches, which result in theoretical minimum external memory accesses, the on-chip buffer should have 5.3 million entries, and with 32-bit numbers, this translates to a buffer size of 21.2MB, based on Table I. We explore the possibility of having an alternative processing scheme that requires smaller buffer sizes while achieving near theoretical minimum external memory accesses. To do so, we construct a simple accelerator model with buffer and MAC connected to the external memory, as shown in Fig. 4. We vary the buffer capacity from 32-kilo entries to 32 million entries and try all possible mappings to find the optimal mapping for the least memory accesses.

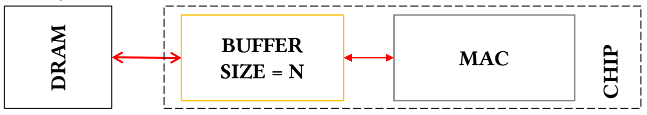

Fig. 4. Model used to identify ideal buffer size and mapping scheme

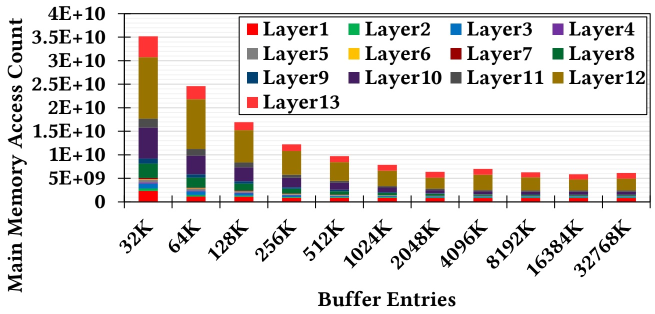

Fig. 5. External memory access vs buffer capacity for optimal mapping

Fig. 5 shows the total and layer-wise accesses to external memory when processing 3D UNet on the one-MAC-one-buffer model with various buffer capacities. From the graph, $\sim$1 million entries provide near minimum access - any further increase in buffer capacity has diminishing returns.

VII. Proposed Architecture

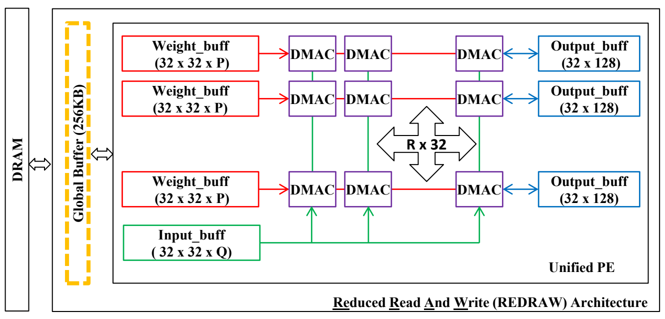

Fig. 6. REDRAW Architecture.

We propose the REDRAW architecture based on the observations made in Section VI in Fig. 6. The architecture consists of an optional global buffer (GLB) and a single unified PE. As shown in later sections, the optional GLB buffer can be omitted from implementation if the area is a priority constraint and will not significantly impact overall performance. The layer-wise input and weight data are fetched from the external DRAM for processing, and the final layer output is written back to the external DRAM. Based on the optimal mapping, the chip controller can selectively bypass GLB for input, weights, and outputs, and data can be directly written to the buffers inside the PE. There are separate buffers for inputs, weights and outputs within the PE. As seen in Fig. 6, the input buffer is 32 words wide, and Q deep, and each word is shared across R Dual-MACs (DMAC). Each DMAC contains two multipliers that operate for every other cycle in a tick-tock fashion. There are R weight and output buffers inside the PE, each corresponding to an output channel that is processed in parallel. Here, the local weight buffer is 32 words wide and P deep, and provides weight data to all DMACs in a row, while the output buffer is only one word wide as the sum of products from the DMAC units are accumulated using an adder tree. Since all the local weight buffers together can result in 1024 reads in a cycle, to reduce overall energy, partial sums corresponding to different output words that use the same weight word are performed sequentially till the output buffer is filled before moving to the next weight word. This processing strategy ensures 100% utilization of DMACs during the processing of 3D UNet. The rest of the order of computation (which determines the optimal mapping) along parameter N and M (discussed in Section VI-A), and height, width and depth of the output channel differs from layer to layer based on size and shape and is found using Timeloop to get the lowest latency and energy.

TABLE IV
Variations of REDRAW architecture

Name	Parameters	Description	Area

R_AM	P=217, Q=1024
	R=32
	NO GLB
		Same peek
		performance (TOPS)
		as Simba.
			10% less
			area than
			1x Simba

R_MMA	P=432, Q=1024
	R=64
	NO GLB
		Same peek
		performance
		as 2x instance of Simba
			10% more
			area than
			2x Simba

R_GMMA	P=432, Q=1024
	R=64, with GLB
		For sensitivity analysis of
		including
		the optional global buffer
			13% more
			area than
			2x Simba

R_xMMA	P, Q=variable R=64	For sensitivity analysis of buffer distribution	

Table IV shows the implementations of the REDRAW architecture considered for evaluation. In the table, R_AM is used to compare against a single instance of SOTA (Simba) to show the energy efficiency that can be achieved under ISO-area, whereas R_MMA is the optimal design with near minimum theoretical memory accesses. Simba2x represents a system with two instances of Simba chiplet and has the same peek performance as R_MMA.

VIII. Results and Evaluation

A. *Area breakdown*

Fig. 8 shows the breakdown of the area of the proposed accelerator. R_AM has a similar area as that of Simba and is used to evaluate the efficacy of observation made in Section VI-A and VI-B. R_MMA exploits the observation made in Section VI-C and is designed to meet the constraints of

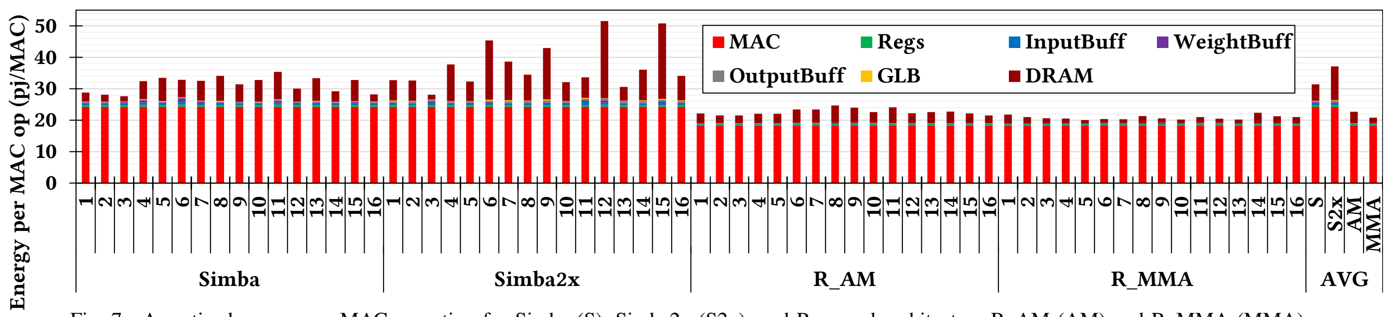

Fig. 7. Amortized energy per MAC operation for Simba (S), Simba2x (S2x), and Proposed architecture R_AM (AM) and R_MMA (MMA)

processing 3D UNet (as detailed in Section II-A). From Fig. 8, compared to Simba, REDRAW dedicates more chip area for multipliers and adders.

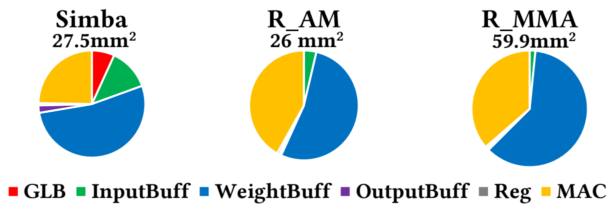

Fig. 8. Area Breakdown

B. Processing Time

Fig. 9 shows the latency for inferencing an input of 3D UNet and the contribution of each layer to processing time. Two instances of Simba and R_MMA (each with 4.4 second latency) meet the latency requirements for the edge use case.

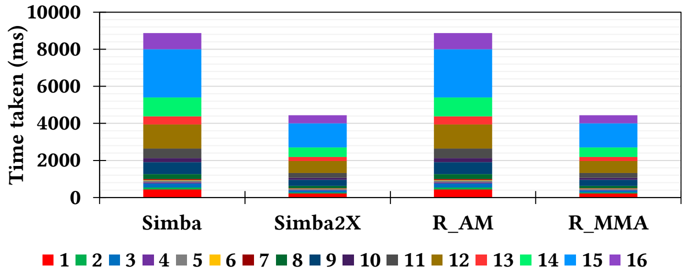

Fig. 9. Time to inference one input of 3D UNet

C. Power and Energy

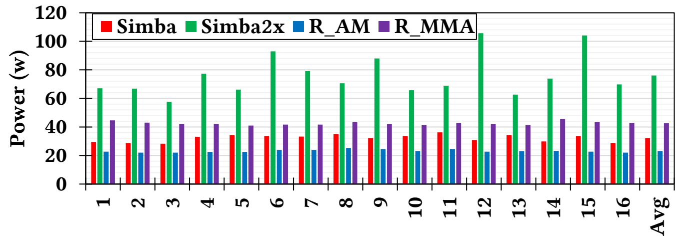

Fig. 10. Power consumed while processing each layer of 3D UNet

Fig. 10 shows the power required to process each layer of 3D UNet. Simba, R_AM, and R_MMA each meet the application's power requirements. However, only R_MMA meets the timing requirements. Scaling the workload across two instances of Simba creates additional overheads in communications, causing increased power, thus making Simba2x unattractive for the presented application.

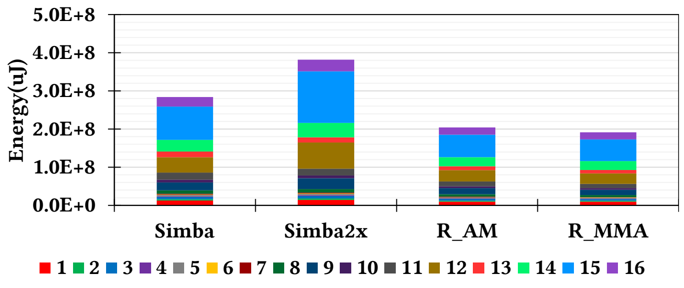

Fig. 11. Comparison of energy required for processing each layer of 3D UNet on Simba and proposed architecture

Fig. 11 shows Simba and REDRAW's total and layer-wise energy consumed for inferencing 3D UNet. SOTA Simba consumes 30% more energy than R_AM with the same area. Simba2x consumes 2× energy than R_MMA with 9% area overhead while Simba consumes 1.5× than R_MMA.

Fig. 7 shows the amortized breakdown of energy spent per MAC operation at each accelerator component. The proposed architecture (R_MMA) reduces the multiplication and accumulation energy by 24% compared to Simba due to the multi-cycle multipliers. Due to the shared usage of input words across 32 MAC units, the input buffer energy is reduced by 3×. Compared to Simba, there is 4× reduced peripheral circuitry for the weight buffers. Further, the depth of the output buffer for each output channel is twice that of Simba, resulting in less number of weight reads while partial products are generated - this results in 80% lesser energy at the weight buffer level for the proposed architecture. Since the number of updates to the output buffer is reduced by 4× because of the adder tree and due to the smaller overall size of the output buffer, the output buffer energy is reduced by 75% compared to Simba. Further, due to the near-optimal count of memory accesses, the external DRAM access energy is only 45% of Simba's external DRAM access energy.

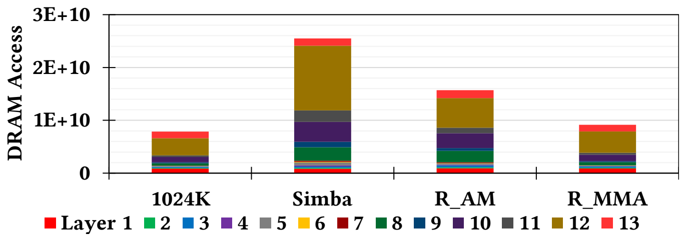

Fig. 12. Accesses to external DRAM compared to 1024K entry optimal system from Section. VI-C

D. Memory Accesses

Fig. 12 shows the access counts to the external DRAM. Simba has 3.2× more memory accesses while R_MMA has an additional 15% accesses compared to a 1024K entry near optimal system.

IX. Sensitivity Analysis

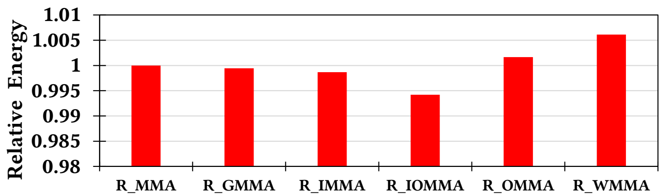

Fig. 13. Relative energy of variations of REDRAW.

We also perform sensitivity analysis on different variants of the REDRAW architecture based on the allocation of buffers. These choices do not lead to higher compute throughput but affect energy consumption. Further, these changes should not bring large changes in energy consumption as that might suggest a technology-dependent architectural benefit. The sensitivity analysis results are provided in Fig. 13, and the variants are explained in the following subsections.

A. Global Buffer (GLB)

We observe that including a 256KB GLB to REDRAW (R_GMMA) at the cost of $2mm^2$ area does not reduce the overall energy. We observed that optimal mapping bypassed the GLB except for the middle layers (5, 6, and 7).

B. Buffer allocation

Here we consider R_MMA as the baseline design. We reallocate 256 KB of buffer volume from the baseline to study the effect on inference energy. In R_IMMA and R_OMMA, the reallocation is from the weight buffer to the input buffer and output buffer, respectively. In R_IOMMA, 256KB is reallocated from the weight buffer to both input and output buffers equally. Lastly, in R_WMMA, 256KB is reallocated from the output buffer to the weight buffer. Fig. 13 shows that these variants have less than 1% change in energy consumption. Further, we note that Simba2x with 60% more buffer capacity than R_MMA consumes 78% more energy.

X. Conclusion

In this work, we propose a DNN accelerator for inference of 3D UNet in medical embedded applications. To the best of our knowledge, this is the first ASIC-based DNN accelerator for 3D UNet, which processes spatial data - in contrast to the temporal data in other 3D CNNs. We characterize the computational aspects of 3D UNet and reduce the external DRAM access energy by 55%, and internal buffer access energy by 81% compared to the SOTA accelerator for spatial data, Simba. Further, we reduce the energy for MAC operation in the PE by 24%. Further, for lower overall energy and higher power, compared to Simba, the speedup is increased by 2×. The proposed architecture increases the TOPS per watt by 1.5×.

References

[1] Y. Chen *et al.*, “A survey of accelerator architectures for deep neural networks,” *Engineering*, vol. 6, no. 3, pp. 264–274, 2020.

[2] P. Mattson *et al.*, “Mlperf training benchmark,” *arXiv preprint arXiv:1910.01500*, 2019.

[3] S. Mittal *et al.*, “A survey of accelerator architectures for 3d convolution neural networks,” *Journal of Systems Architecture*, vol. 115, p. 102041, 2021.

[4] K. Hegde *et al.*, “Morph: Flexible acceleration for 3d cnn-based video understanding,” in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018, pp. 933–946.

[5] Ö. Çiçek *et al.*, “3d u-net: learning dense volumetric segmentation from sparse annotation,” in *International conference on medical image computing and computer-assisted intervention*. Springer, 2016, pp. 424–432.

[6] F. Mentzel *et al.*, “Fast and accurate dose predictions for novel radiotherapy treatments in heterogeneous phantoms using conditional 3d-unet generative adversarial networks,” *Medical Physics*, vol. 49, no. 5, pp. 3389–3404, 2022.

[7] D. Nguyen *et al.*, “3d radiotherapy dose prediction on head and neck cancer patients with a hierarchically densely connected u-net deep learning architecture,” *Physics in medicine & Biology*, vol. 64, no. 6, p. 065020, 2019.

[8] Reddi *et al.*, “Mlperf inference benchmark,” in *2020 ISCA*, 2020.

[9] H. Kwon *et al.*, “Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects,” *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 461–475, 2018.

[10] P. Darbani *et al.*, “Rasht: A partially reconfigurable architecture for efficient implementation of cnns,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 7, pp. 860–868, 2022.

[11] NVIDIA, “3d-unet medical image segmentation for tensorflow: Nvidia ngc.” [Online]. Available: catalog.ngc.nvidia.com/orgs/nvidia/resources/unet3d_medical_for_tensorflow

[12] F. Xiao *et al.*, “Transdose: a transformer-based unet model for fast and accurate dose calculation for mr-linacs,” *Physics in Medicine & Biology*, 2022.

[13] MLCommons, “Mlperf edge inference v2.1 results.” [Online]. Available: mlcommons.org/en/inference-edge-21/

[14] B. Zimmer *et al.*, “A 0.32–128 tops, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm,” *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, 2020.

[15] G. Armeniakos *et al.*, “Hardware approximate techniques for deep neural network accelerators: A survey,” *ACM Computing Surveys (CSUR)*, 2022.

[16] B. Moons *et al.*, “14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi,” in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2017, pp. 246–247.

[17] Z. Yuan *et al.*, “Sticker: A 0.41-62.1 tops/w 8bit neural network processor with multi-sparsity compatible convolution arrays and online tuning acceleration for fully connected layers,” in *2018 IEEE symposium on VLSI circuits*. IEEE, 2018, pp. 33–34.

[18] J. Lee *et al.*, “Unpu: A 50.6 tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision,” in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 218–220.

[19] J. Song *et al.*, “7.1 an 11.5 tops/w 1024-mac butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile soc,” in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2019, pp. 130–132.

[20] P. Chakraborty *et al.*, “Arts: A framework for ai-rooted iot system design automation,” *IEEE Embedded Systems Letters*, vol. 14, no. 3, pp. 151–154, 2022.

[21] A. Parashar *et al.*, “Timeloop: A systematic approach to dnn accelerator evaluation,” in *2019 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE, 2019, pp. 304–315.

[22] Y. N. Wu *et al.*, “Accelergy: An architecture-level energy estimation methodology for accelerator designs,” in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.

[23] N. Muralimanohar *et al.*, “Cacti 6.0: A tool to model large caches,” *HP laboratories*, vol. 27, p. 28, 2009.