2025 Design, Automation & Test in Europe Conference (DATE 2025)

Axon: A novel systolic array architecture for improved run time and energy efficient GeMM and Conv operation with on-chip im2col

Md Mizanur Rahaman Nayan*, Ritik Raj*, Gouse Basha Shaik*, Tushar Krishna*, Azad J Naeemi*
*Department of Electrical and Computer Engineering,
Georgia Institute of Technology, USA

***Abstract*—General matrix multiplication (GeMM) is a core operation in virtually all AI applications. Systolic array (SA) based architectures have shown great promise as GeMM hardware accelerators thanks to their speed and energy efficiency. Unfortunately, SAs incur a linear delay in filling the operands, due to unidirectional propogation via pipeline latches. In this work, we propose a novel in-array data orchestration technique in SAs where we enable data feeding on the principal diagonal followed by bi-directional propagation. This improves the runtime by up to $2\times$ at minimal hardware overhead. In addition, the proposed data orchestration enables convolution lowering (known as im2col) using a simple hardware support to fully exploit input feature map reuse opportunity and significantly lower the off-chip memory traffic resulting in $1.2\times$ throughput improvement and $2.17\times$ inference energy reduction during YOLOv3 and RESNET50 workload on average. In contrast, conventional data orchestration would require more elaborate hardware and control signals to implement im2col in hardware because of the data skew. We have synthesized and conducted place and route for 16×16 systolic arrays based on the novel and conventional orchestrations using ASAP 7nm PDK and found that our proposed approach results in 0.211% area and 1.6% power overheads.**

***Index Terms*—Systolic array, GeMM, im2col, AI accelerators**

I. Introduction

General Matrix Multiplication (GeMM) is fundamental to modern AI applications, with throughput often constrained by execution speed. General-purpose computers struggle with matrix multiplication due to their complex architectures, prompting significant research into domain-specific accelerators. GPUs(Graphical Processing Units), designed with SIMD (Single Instruction Multiple Data Stream) architecture, are a popular choice for accelerating AI applications due to their high parallelization capability. However, GPUs are power-intensive, largely due to their general-purpose design and reliance on memory hierarchy for communication [1], [2].

Systolic array (SA)-based architectures have gained traction for GeMM due to their simple design, lower memory bandwidth requirements, and energy efficiency [3]–[10]. Nearly all commercial deep neural network (DNN) accelerators utilize systolic arrays [1], [2], [11]–[21], and open-source frameworks like Gemmini [22] also generate SA-based designs. Unlike GeMM, convolution operations require the im2col algorithm to reshape 3-D tensors into 2-D matrices, enabling their execution on GeMM accelerators. Conventional SA implementations rely on software-based im2col, which stores the matrices in SRAM buffers. While computationally efficient, this approach demands significant on-chip memory due to element repetition in convolution windows. Optimized SA designs address this for convolution workloads [8], [21], [23]–[25]. Eyeriss [15] addresses the memory challenges of systolic arrays by introducing a novel row stationary dataflow. However, its processing elements (PEs) require larger Multiply-Accumulate (MAC) units and register files (RF) compared to simpler systolic arrays. This is because each PE buffer must store a filter row, and input feature maps (IFMAPs) need to be copied into each set of PEs working on a filter, unlike systolic arrays where the same IFMAP copy is reused across filters. Similar to Eyeriss, FlexFlow [23] was proposed to tackle these issues, but neither adheres to the fundamental systolic array architecture. While computationally efficient, their reliance on data bus communication limits system frequency [21]. Gemmini [22] mitigates CPU dependency for im2col operations using optional im2col units but still suffers from frequent memory accesses. SPOTS [26] reduces off-chip memory access but introduces a large intermediate buffer. Recent efforts on on-the-fly im2col eliminate software im2col requirements, reducing data transfer [27], [28] but incurs significant hardware overhead, including counters, registers, FIFOs, and complex control signals.

In this work, we introduce Axon, a novel systolic array architecture with in-array data orchestration and optimized data transfer from the scratchpad to the array, preserving the simplicity and reuse ability of traditional systolic arrays. Unlike conventional uni-directional propagation, Axon feeds data to the PE on the principal diagonal and propagates it bi-directionally. This design improves runtime for the three dataflows - output stationary (OS), weight stationary (WS), and input stationary (IS). We derive a runtime equation to quantify the speedup and enable convolutional data reuse by replacing software im2col with a simple 2-to-1 multiplexer (MUX), significantly reducing hardware overhead compared to previous works [27], [28]. Axon also incorporates zero gating to exploit sparsity in both input feature maps (IFMAP) and filters. Our experiments validate speedups on SOTA workloads (transformer, conformer, CNN) and show reduced power consumption and hardware overhead through ASIC synthesis and physical design. The key contributions are as follows:

- A novel in-array data orchestration that speeds up GeMM irrespective of dataflow achieving $2\times$ speedup on relatively memory bound operations like GEMV and DW-conv due to reduced feeding latency and elimination of data skew.
- An im2col support adding minimal hardware overhead (2-to-1 MUXes), enabling input feature map reuse and significantly reducing off-chip memory traffic during convolution.
- Unified PE design to flexibly exploit all the three dataflows
- Integrating Zero gating to reduce power consumption by leveraging sparsity in IFMAP and filters.
- Physical design and layout implementation of Axon using a

7-nm FiNFET PDK for scalability and backend analysis.

II. Background and Motivation

A. *Conventional systolic array and dataflows*

Dataflows in a systolic array define how the elements of operand matrices and partial sums propagate inside the array. In the OS dataflow, partial sums remain stationary while operands from both input and weight propagate through the array in a systolic order. At the end of the operation, outputs are extracted from the array. In WS, weights are pre-filled inside the array and remain stationary until the input propagates through the whole systolic array. In each cycle, partial sums are generated in each PE and propagate downward to be added in the next cycle with another PE-generated partial sum. Finally outputs are collected from the bottom row of the array. IS is similar to WS with the only difference being that inputs instead of weights are pre-filled in the array and remain stationary. Fig. 1 demonstrates in-array dataflows in the conventional systolic array, in which the input feature map/one operand matrix is loaded from the input feature map buffer to the systolic array through the leftmost PEs and propagates from left to right. Weights/another operand matrix are loaded from the weight buffer through PEs in the top rows and propagate from top to bottom. Each PE in the array is responsible for one MAC operation and propagates inputs and outputs based on the dataflow type.

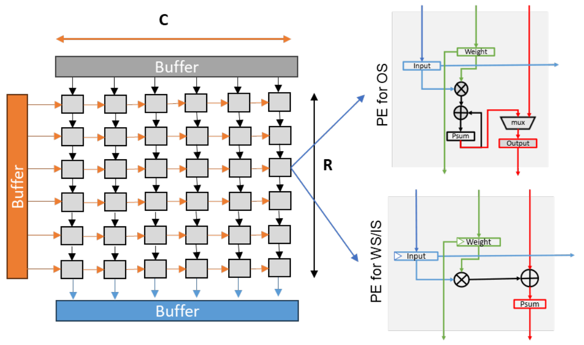

Fig. 1. Data feeder from the buffer and in-array dataflow in conventional systolic array with PE's architecture

B. *Runtime Modeling*

SCALE-SIM [29] developed an analytical model for systolic array runtime, which is applicable for all the three dataflows discussed above. According to the model, for the operand matrices of dimensions $S_R \times T$ and $T \times S_C$ respectively the runtime is as follows:

$$\tau = 2S_R + S_c + T - 2 \tag{1}$$

where S_R and S_C are the spatial dimensions along which computation is mapped, and T represents the corresponding temporal dimension. The original operand matrices are projected into the available spatio-temporal dimensions. For example, during the multiplication of two matrices of shapes $M \times K$ and $K \times N$, the dimension M is mapped to S_R, dimension N is mapped to S_C and the dimension K to T for OS dataflow. Mapping for three dataflows are summarized in Table.I. However, Large GEMM problems are managed on smaller systolic arrays through tiling in scale-up (one large array) or scale-out (multiple smaller arrays) as depicted in

Dataflow	Mapping
OS	$(S_R = M, S_C = N, T = K)$
WS	$(S_R = K, S_C = M, T = N)$
IS	$(S_R = K, S_C = N, T = M)$

TABLE I
Mapping of GEMM dimension for different dataflows

Fig. 2. Runtime equation for the scale up and scale out is modified as Eq. 2 and 3, respectively,

$$\tau_{scaleup} = (2R + C + T - 2) * (S_R/R) * (S_C/C) \tag{2}$$

$$\tau_{scaleout} = (2R + C + T - 2) * (S'_R/R) * (S'_C/C) \tag{3}$$

where R and C are the number of rows and columns of the systolic array, S'_R and S'_C are (S_R/P_R) and (S_C/P_C), respectively, P_R is the partition number across the rows, and P_C is the partition number across the columns. If we break down the runtime equation, we find the following three components:

- Time for both operands to reach the farthest PE with respect to the feeder PEs, $R + C - 2$
- Number of multiplications each PE performs, T
- Readout from the array, R

For any dataflow (OS, WS, and IS) the number of multiplications each PE performs is determined by the temporal dimension, T. The readout time is determined by the number of rows of the systolic array, R. However, the first component is determined by the in-array data propagation direction. For the conventional systolic array, it is the Manhattan distance where the distance is defined by the distance from the vertical and horizontal axes. For the farthest PE (bottom right corner) in the array, the distance is $(R + C - 2)$.

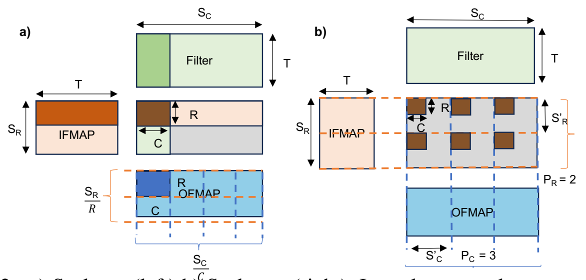

Fig. 2. a) Scale up (left) b) Scale out (right). In scale up one large monolithic array is used whereas in scale out multiple systolic arrays are used.

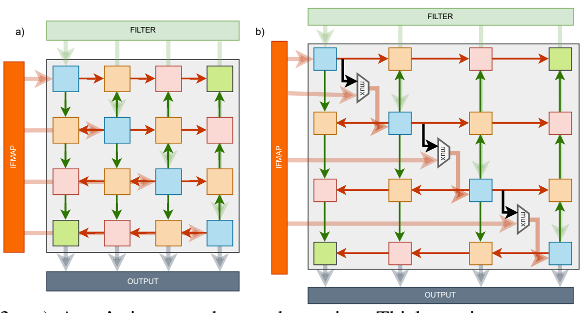

Fig. 3. a) Axon's in-array data orchestration. Thick semitransparent arrows indicate data movement into the systolic array from buffers and thin solid arrows indicate data movement inside the array among the PEs. The same colors on PEs represent operands' arrival at the same cycle whereas PEs on the principal diagonal receive the operands on the first cycle directly from the buffers. b) im2col implementation. Note that, each MUX allows feeder PEs to receive data either from buffer or from immediate PE on the diagonal.

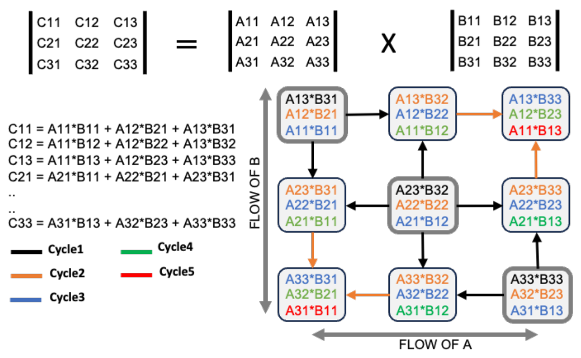

Fig. 4. 3 × 3 GeMM example to validate Axon data orchestration. Partial products labeled with the same colors indicate that they are generated in the same cycle. All operands are fed to SA through PE on the principal diagonal.

III. Axon Data Orchestration Strategy

In the conventional orchestration, it takes $(R + C - 2)$ cycles for the operands to reach the farthest PE. The Axon data orchestration shown in Fig. 3 improves this portion of the runtime by feeding the operand matrices to the systolic array through the PEs on the principal diagonal which we call feeder PEs (semi-transparent thick arrows). Once data is inserted into the systolic array, data movement inside the array is orchestrated according to Fig. 3 (solid-thin arrows) to perform the matrix multiplication. The feeder PEs except the two corner PEs transfer the operands in two directions, unlike other PEs on the array. Filter elements are allowed to propagate towards both adjacent upward and downward PEs whereas IFMAP elements are allowed to propagate towards both adjacent right and left PEs. Other PEs in the array propagate the operand in the same direction as they receive data. Note that in this way, we do not need any extra hardware. We only need to rearrange the in-array dataflow direction and feed the array from the PEs on the principal diagonal. Moreover, Unlike conventional SA Axon does not need to stream the operand matrices in a skewed manner which increases PE utilization. Fig. 4 shows a toy example to demonstrate the Axon data orchestration through a two 3 × 3 matrix multiplication. While systolic arrays are often square, the Axon data orchestration can be extended for rectangular systolic arrays and reduce runtime as shown in Fig. 5. For the columns that do not have any PEs on the principal diagonal, operands are fed from the bottom PE of the column with spatially skewed data similar to a conventional systolic array, ensuring accurate data orchestration for the PEs.

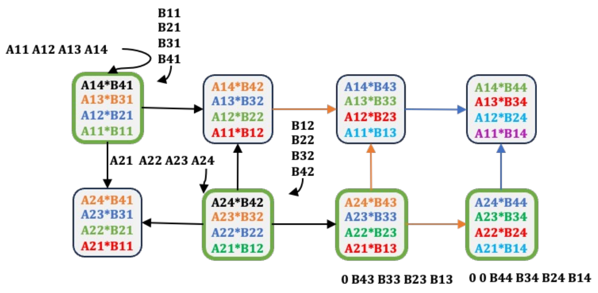
A11 A12 A13 A14
B11
B21
B31
B41
A14*B41 A13*B31 A12*B21 A11*B11
A14*B42 A13*B32 A12*B22 A11*B12
A14*B43 A13*B33 A12*B23 A11*B13
A14*B44 A13*B34 A12*B24 A11*B14
B12
B22
B32
B42
A21 A22 A23 A24
A24*B41 A23*B31 A22*B21 A21*B11
A24*B42 A23*B32 A22*B22 A21*B12
A24*B43 A23*B33 A22*B23 A21*B13
A24*B44 A23*B34 A22*B24 A21*B14
0 B43 B33 B23 B13
0 0 B44 B34 B24 B14

Fig. 5. Axon data orchestration in the rectangular systolic array. The columns that do not have any PEs on the principal diagonal will be fed through PEs at the bottom of the array with zero padding based on the distance. Notice third column is fed with zero-padded by one and the fourth column is fed with zero-padded by two.

Dataflow	Systolic array	Axon
OS	$2M + K + N - 2$	$\max(M, N) + M + K - 1$
WS	$2K + M + N - 2$	$\max(M, K) + K + N - 1$
IS	$2K + M + N - 2$	$\max(N, K) + K + M - 1$

TABLE II
Runtime for SA and Axon

A. Runtime modeling

The 2nd and 3rd terms in the runtime equation presented by Samajdar et al [29] remain unchanged in this orchestration while the 1st term is changed to $max(R, C) - 1$. For a square systolic array$(R = C)$, it is simply $(R - 1)$ which is half of the $(2R - 2)$ for the conventional systolic array. The improvement for non-square arrays is smaller but is always greater than 1. Table. II summarizes the runtime for various dataflows w.r.t their operand matrix shape considering $S_R = R$ and $S_C = C$. Fig. 6 shows that for any shape of the array (R, C) the factors that determine the time it takes for the operands to reach the farthest PE are always lower in Axon data orchestration. For example, in a systolic array of shape $(256, 256)$ the compute time required to reach the farthest PE is reduced from 510 cycles to 255 cycles. However, the overall runtime improvement for a scale-out design with smaller arrays is limited by two other factors (i.e. Temporal dimension length and Readout time) according to Amdahl's law.

B. Axon's hardware support for im2col

During im2col, IFMAP is converted to conv windows depending on the shape of the FILTER. Each conv window is responsible for generating one element of OFMAP. In the example shown in Fig. 7, Filter and IFMAP are of shape 3×3 and 6×6 respectively. Thus OFMAP shape is 4×4. In other words, there are in total 16 conv windows corresponding to 16 elements of the 4×4 OFMAP. In the example only 4 conv windows correspond to 1st row of OFMAP has been shown. Note that in conv windows, there are 18 unique elements and the remaining 18 elements have been repeated (50% repetition) in a pattern that can be formulated in terms of the FILTER length, n. The number of common elements in consecutive conv windows is $n(n - 1)$, which is 3×2 or 6 for the demonstrated example. Also between each adjacent conv window, there is a cyclic pattern with a period of n where there are $n - 1$ common elements. This repetition becomes more prominent for large FILTER and IFMAP shapes which results in excessive memory traffic and a need for either a large on-chip memory or expensive DRAM access. In Axon, we leverage this repetition pattern to reuse the elements directly from the PEs instead of DRAM access or having a large on-chip memory.

Fig. 3(b) demonstrates the proposed Axon's simple im2col hardware support where each PE on the principal diagonal can be fed with IFMAP elements either directly from IFMAP SRAM buffers or from the adjacent top PE on the principal diagonal using multiplexers (MUX), where the control signal is 0 for 1 cycle and 1 for the other $(n - 1)$ cycles. With this configuration, during the 1st cycle, all the 4 elements (the rightmost element from each row of IFMAP conv window matrix in Fig. 7(d)) will be loaded from the SRAM buffer through feeder PEs of the array. In the 2nd cycle, only the

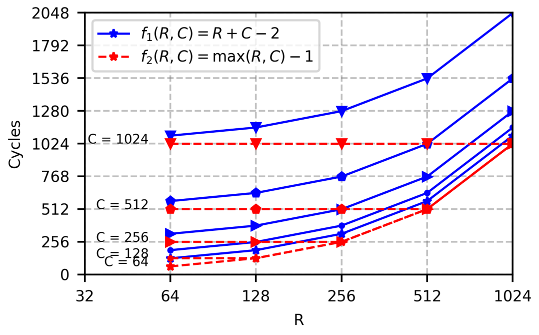

Fig. 6. Runtime (cycles) for feeding operand to the farthest PE. $f_1(.)$ represents conventional systolic array, $f_2(.)$ represents Axon data orchestration.

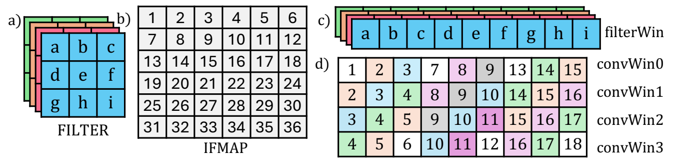

Fig. 7. Im2col on a) FILTER of shape (3x3) and b) IFMAP of shape (6x6). After im2col each filter is flattened like c) and each conv window is flattened as shown in d). (Only 4 conv windows have been shown related to producing 1st row of the output).

row with convWin0 will load the next element from SRAM buffer while all the other three conv windows will get their 2nd elements from the adjacent top feeder PEs via mux i.e. the feeder PE of convWin1 will get element from the feeder PE of convWin0. Similarily the feeder PE of convWin2 will get the element from convWin1's feeder PE and so on. In the 3rd cycle, convWin1, convWin2, convWin3 will get their element from the top adjacent feeder PE just like 2nd cycle. The same dataflow will repeat from the 4th cycle since the FILTER length is 3). Hence, Axon can readily exploit the reuse and just load the PEs with elements from immediate feeder PE instead of loading from memory for $(n-1)$ cycles out of n periodic cycles. Thus, we can avoid the high memory traffic imposed by software-based im2col with a minimal hardware overhead unlike hardware support proposed in prior works. The proposed novel data orchestration makes this simple scheme possible. In our approach data is loaded in an ordered fashion to the feeder PEs from memory in contrast to the conventional orchestration where the data is skewed.

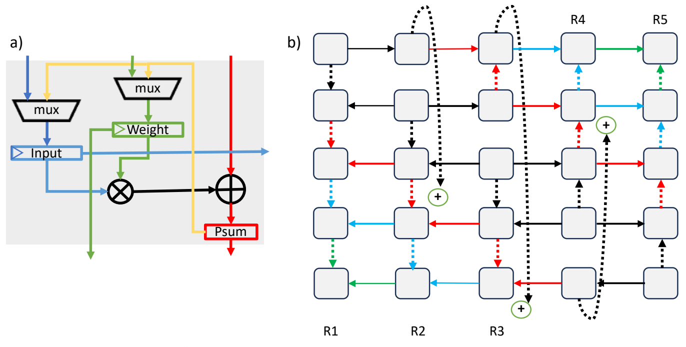

Fig. 8. a) IS/WS PE for Axon, b) Partial sums are coordinated using bypassing at the same stage apart from the feeder PE in the column to avoid partial sum data corruption with different output elements. Identical arrow color represents same cycles during the dataflow.

IV. PE Design for Axon

PEs are commonly designed based on the dataflow [3], and the design of PEs is slightly different for IS and WS than the OS. Fig. 1 depicts the standard design for PEs of OS and IS/WS. For Axon data orchestration, the only change in the design of the PEs is in terms of data propagation. We will discuss the design of PEs for each dataflow especially for WS/IS due to their design challenges for Axon architecture in the following section.

A. *Output Stationary dataflow*

PE design of Axon OS dataflow is identical to the conventional design as depicted in Fig.1. In Axon, PE on principle diagonal except the ones at the two corners propagate data in both directions, unlike the PE in conventional design. So the only change in hardware is the addition of an interconnect to allow bi-directional propagation of data. The PE also integrates the zero-gating approach presented in [30] where MAC operation is simply skipped if zero exists in either IFMAP or FILTER operand. This helps to reduce power during sparse GEMM.

B. *Input/Weight Stationary dataflow*

Axon data orchestration is straightforward to implement for OS dataflow. IS and WS dataflows; however, raise the two following challenges:

1) Preloading: From Fig.3 we observe that data is fed from the buffer to the systolic array through the diagonal elements and the flow is in both directions (top-bottom, left-right). Unlike OS, we require preloading weight/input for WS/IS dataflow. The loading takes S_R cycles. Loading from the buffer to the array according to the proposed architecture becomes faulty because of dataflow in both directions. To solve the problem, we have utilized an output interconnect which is used to propagate the output vertically (Fig.8a) yellow route). Two additional 2-to-1 MUXes are required to determine the target buffer based on the dataflow. Note that the in-array dataflow direction during computation remains unchanged like the OS dataflow.

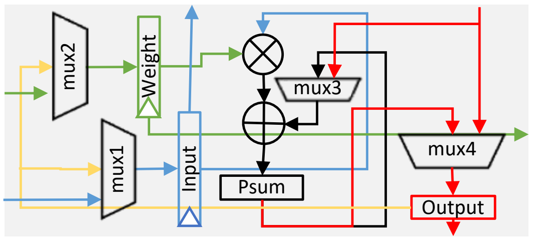

Fig. 9. Axon Unified PE for OS, IS, and WS.

2) Partial sum synchronization: Another challenge that comes from implementing the Axon data orchestration in WS/IS is partial sum synchronization. In a conventional systolic array, the operand elements propagate from top to bottom and left to right and the partial sum is also calculated and propagated downward. In each stage, a partial sum gets added to the previous stage's total sum along a column and finally leaves the array from the bottom row PEs as the output. Thus, it ensures that the correct partial sums are added in the right order. However, in the proposed architecture the operand elements propagate in both directions; hence, the partial sums corresponding to the same output element are generated in parallel. They must be collected for accurate output generation before leaving the array. To address the issue we use the bypass and add approach which adds two portions of the final output elements separated by the PEs on the principal diagonal on that column. The PEs on the principal diagonal propagate operands in both directions but propagate their output in only one direction (either to the bottom or to the top). Fig.8(b) depicts the bypassing for IS/WS dataflow where the dashed arrows represent the direction of partial sums propagation and the solid arrows indicate the direction of the operand propagation. Remember that in IS/WS one operand (either IFMAP or FILTER operands) is preloaded and kept stationary where the other operand is allowed to propagate. Bypassing allows to avoid output being corrupted and ensures output is collected without pushing stalls.

C. *Axon Unified PE design for OS, IS/WS dataflow*

In this section, we introduce a unified PE design as depicted in Fig. 9 that is programmable to switch the data to any of the three dataflows for the Axon data orchestration. Two Multiplexers (MUX1 and MUX2) are used to select the target buffer based on WS/IS dataflow during preloading. They forward data from the buffer to the input/weight buffer through the output path (yellow route). For the WS/IS dataflows, MUX3 will forward the previous partial sum to the adder and then forward the final sum to the output buffer to propagate to another PE's MUX to be summed in the next cycle. For the OS, the black route will be active. Through MUX3 and Psum, the previous partial sum buffered in Psum will be summed with the newly generated partial sum, and finally, through MUX4 the result is written to the output buffer.

Workload	M	K	N	Workload	M	K	N
TFO	31999	84	1024	NCF1	256	2048	256
TF1	84	4096	1024	DB0	1024	50000	16
GNMTO	128	4096	2048	DB1	35	2560	4096
GNMT1	2048	32	4096	Resnet50_0_conv2d	64	147	62500
GPT3_0 (matmul0)	1024	1024	80	Resnet50_1_conv2d	512	4608	676
GPT3_1 (matmul1)	1024	2560	7680	YOLO_v3_0_conv2d	64	288	42436
GPT3_2 (addmm)	1024	2560	10240	YOLO_v3_1_conv2d	128	576	10404
GPT3_3 (lmhead)	1024	2560	50257	GEMM_0	128	10	128
NCF0	2048	128	1	GEMM_1	2048	10	2048
				GEMM_2	1024	1024	128
				GEMM_3	64	2560	2560

TABLE III
VALUES OF M, K, AND N FOR DIFFERENT WORKLOADS

V. EVALUATION

In this section, we evaluate Axon architecture against conventional systolic array and configurable multi-directional systolic array(CMSA) [31]. CMSA adds an additional datapath to the systolic array to improve computing efficiency. We used the analytical model adopted from SCALEsim [29] for SA runtime calculation and the analytical model from the CMSA paper. For im2col hardware support we compared with Sauria [27]. Sauria has a data feeder to support on-the-fly im2col. TSMC 45nm PDK and ASAP 7nm PDK [32] have been used for the RTL synthesis and PnR. We used Synopsys VCS and DCSHELL for RTL design and functional verification. We evaluated the speedup of Axon architecture over the baselines for GEMV, DW-Convolution, GEMM, and Convolution workloads. Workloads from Transformers (e.g. GPT3) [33], [34], CNN(e.g. RESNET, Efficientnet, Mobilenet and YOLO V3), Conformer and Sparse GEMM have been used. Table. III summarizes the M, K, and N values corresponding to GEMM and Conv (mapped to GEMM) workloads. For comparison, we used scale-up to calculate the run time for the experimental demonstration. The run-time improvement in scale-up due to the proposed orchestration will be reflected linearly in the scale-out as well.

A. *Hardware implementation*

We designed and verified Axon architecture with OS dataflow of shape 16×16 that includes the proposed im2col hardware support and zero gating module. The OS dataflow is used because it offers high IFMAP and filter reuse opportunities [35]. We used a simplified version of the FPnew floating-point unit from the open-source parallel ultra-low power (PULP) platform [36] to implement FP16 MAC unit. Fig. 11 summarizes the specifications. We found $0.9992m^2$ of Si area for conventional SA and $0.9931mm^2$ for Axon. The slight reduction in the area is due to buffer sharing between two adjacent PEs of the feeder PE on the principal diagonal. The Input and weight buffers of PEs can be shared between two PEs horizontally and vertically separated by PE on principal diagonal respectively as they receive the same data in the same cycle and are at the same distance from the feeder PE. After adding Im2col hardware support, the Si area becomes increases by only 0.2% (total area of $0.9951mm^2$). The total power increases by 1.6% (59.98mW) with im2col support compared to SA (59.88mW) which indicates that power change is insignificant compared to the conventional SA.

B. *Results*

1) Comparison with SA: Fig. 10a) depicts the performance comparison of the systolic array with Axon for GEMM and Convolution workloads as summarized in Table. III for various array shapes. We have normalized the runtime (cycles) w.r.t corresponding systolic array runtime. On average, we observe Axon offers $1.47\times$ speedup over the SA for an array shape of 64×64. For larger arrays Axon performs even better for most workloads. However, for some workloads (e.g. NCF0 and DB0) for which the runtime is limited by the temporal dimension (e.g. NCF0 and DB0), scaling up doesn't help to improve performance. We achieve an average of $1.76\times$ speedup over conventional systolic array for the array shape of 256×256. Fig. 10c) depicts the performance comparison on depthwise convolution(DW-Conv) and general matrix-vector multiplication (GEMV) workloads between SA and Axon. It shows that Axon is suited for low AI (Arithmetic Intensity) operations like GEMV and DW-Conv because of its bidirectional data propagation. We achieve an average speedup of $1.8\times$. We have used the simple integrated zero gating technique presented in [27] to lower the power consumption by leveraging the sparsity and were able to achieve a 5.3% total power reduction for the case of 10% sparsity. Fig.12 demonstrates the capacity of the proposed on-chip im2col to reduce memory access for various convolution workload shapes. We observe that the memory access can be reduced by more than 60% by the on-chip im2col hardware for workloads generally used in SOTA neural networks. The hardware overhead for the im2col is $< 1\%$ found by performing PnR of a 16×16 array. We calculate the energy consumption reduction due to lowering DRAM access

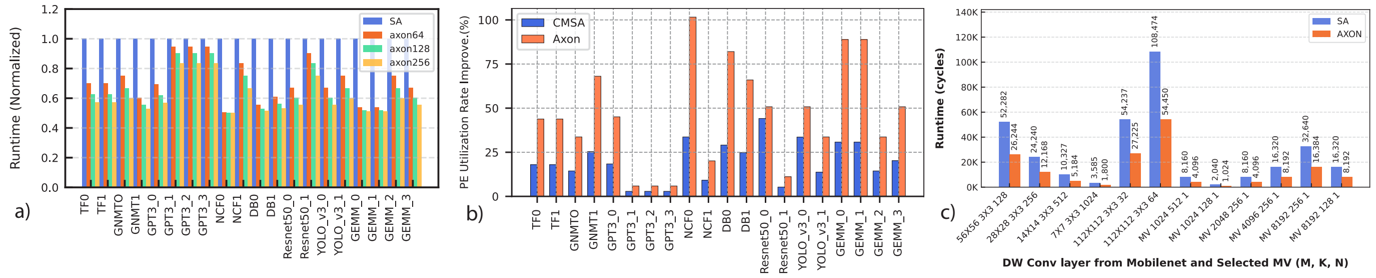

Fig. 10. a) Runtime improvement evaluation on GEMM and Conv workloads b) PE utilization rate improvement over conventional systolic array for CMSA and Axon architecutre c) Runtime improvement evaluation on DW Conv and MV workloads.

Technology	ASAP7 (7 nm)
Chip Area	1 mm x 0.995 mm
PEs	256 (16 x 16)
Arithmetic Precision	FP 16
Frequency	550 MHz
Supply Voltage	0.7 V
Average Power	59.98 mW
Peak Throughput	284 GFLOP/s
Peak Energy Efficiency	4.73 TFLOP/sW

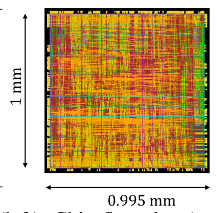

Fig. 11. Implemented Axon Specifications (left). Chip floorplan (post PnR)

for Resnet50 and YOLOv3 models. Memory access (for conv layer only) reduced from $261.2MB$ to $153.5MB$ for Resnet50 and from $2540MB$ to $1117MB$ for YOLOv3. Considering LPDDR3 memory where energy cost is 120pJ/byte as found by [37] we find that the inference energy is reduced by $12mJ$ for Resnet50 and $170mJ$ for YOLOv3. We considered 32-bit-wide LPDDR3 DRAM memory at $800MHz$ with a $6.4GB/s$ maximum bandwidth and found about $1.25\times$ speedup due to lower memory traffic enabled by im2col support which is comparable as reported by [27]. But the proposed im2col area overhead is only 0.2% where the feeder network of [27] is 4%.

2) Comparison with CMSA: We also compared PE utilization rate (UR) improvement over conventional SA for Axon with Configurable multidirectional systolic array (CMSA) architecture proposed by Xu et al [31]. Fig. 10b) illustrates the comparison where UR has been calculated for array shape of 128×128. We observe that the utilization rate improvement varies for Axon and CMSA based on the workload shape and size. Axon outperforms CMSA by an average of 27% in terms of the utilization rate improvement. It should be noted that in some workloads the improvement remains small for both cases (e.g. GPT3 matmul1, GPT3 addmm, GPT3 mhead). This is because for those particular workloads the utilization rate is already high (average 91%) for conventional SA.

3) Energy and Area Comparison: To observe the silicon footprint and power overhead we used 45nm and 7nm advanced technology node for synthesis over different array shapes as

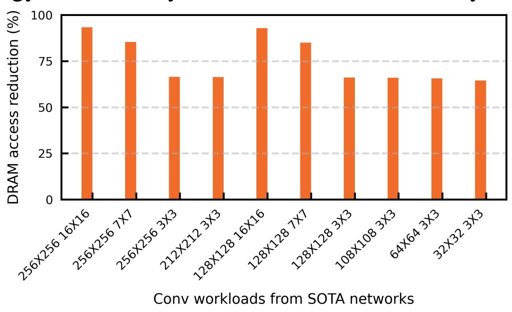

Fig. 12. Memory access reduction using proposed on chip HW support for im2col for different input,weight shapes adopted from SOTA neural networks.

demonstrated in Fig.13a) and b) respectively. We have compared with sauria [27] as it includes hardware im2col support like ours. We find that Axon has an average 3.93% less area and 4.5% less power consumption over Sauria because Axon uses 2to1 mux instead of Sauria's data feeder registers and counters.

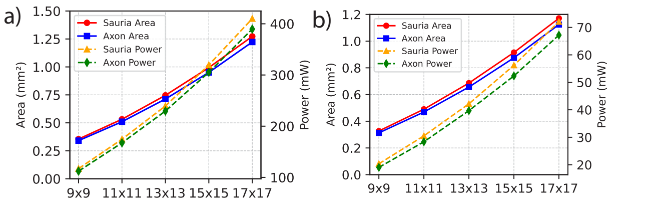

Fig. 13. Power and area comparsion with Sauria a) 45nm node b) 7nm node

VI. Related Work

Samajdar et al [38] proposes a reconfigurable systolic array called SARA to improve mapping flexibility as well as increase data reusability through bypass wires. In MAERI [39] a fabric to support arbitrary dataflows has been proposed. DRACO [40] has been proposed to optimize memory bound DNN workload from algorithm front. AI-MT [20] is designed to maximize the accelerator's computational resources and memory bandwidth by pairing compute-intensive and memory-intensive tasks from different networks, and thus allowing for their parallel execution. COSA [41] is developed to support hybrid data reuse by analyzing the computational characteristics of attention mechanism. However, all these works focus on supporting diverse aspect ratios that arise in DNNs. Our work is orthogonal and can be applied over them. For hardware support of im2col, SPOTS [26] support sparsity with on-the-fly im2col but require a large intermediate buffer. USCA [28] and SAURIA [27] support convolution lowering by using a dedicated data feeder. In contrast, Our work offer im2col through 2to1 mux with exploiting data reusability from adjacent PE buffer.

VII. Conclusion

We present a novel data orchestration in a systolic array that outperforms conventional systolic array and state-of-the-art in terms of run time in all three dataflows (OS, IS, and WS). The Axon architecture offers faster computation (upto $2\times$ speedup) with very small power overhead thus making the design more energy efficient. The proposed architecture of Axon unified PE can leverage workloads' diverse shapes to further improve the runtime. We also propose im2col hardware support enabled by the Axon data orchestration. Axon im2col hardware support can relax the high memory bandwidth requirement by 60% during convolution workloads posed by software im2col with significantly lower hardware (0.2%) and power (1.6%) overhead.

References

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[2] C. Guo, Y. Zhou, J. Leng, Y. Zhu, Z. Du, Q. Chen, C. Li, B. Yao, and M. Guo, "Balancing efficiency and flexibility for dnn acceleration via temporal gpu-systolic array integration," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[3] R. Xu, S. Ma, Y. Guo, and D. Li, "A survey of design and optimization for systolic array-based dnn accelerators," *ACM Computing Surveys*, vol. 56, no. 1, pp. 1–37, 2023.

[4] H.-T. Kung, *Why systolic architecture?* Design Research Center, Carnegie-Mellon University, 1982.

[5] D. Wu and J. S. Miguel, "usystolic: Byte-crawling unary systolic array," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 12–24.

[6] L. Jia, L. Lu, X. Wei, and Y. Liang, "Generating systolic array accelerators with reusable blocks," *IEEE Micro*, vol. 40, no. 4, pp. 85–92, 2020.

[7] X. He, S. Pal, A. Amarnath, S. Feng, D.-H. Park, A. Rovinski, H. Ye, Y. Chen, R. Dreslinski, and T. Mudge, "Sparse-tpu: Adapting systolic arrays for sparse matrices," in *Proceedings of the 34th ACM international conference on supercomputing*, 2020, pp. 1–12.

[8] S. Das, A. Roy, K. K. Chandrasekharan, A. Deshwal, and S. Lee, "A systolic dataflow based accelerator for cnns," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[9] J. Cong and J. Wang, "Polysa: Polyhedral-based systolic array auto-compilation," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[10] Z.-G. Liu, P. N. Whatmough, and M. Mattina, "Systolic tensor array: An efficient structured-sparse gemm accelerator for mobile cnn inference," *IEEE Computer Architecture Letters*, vol. 19, no. 1, pp. 34–37, 2020.

[11] N. P. Jouppi, D. Hyun Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. Patterson, "Ten lessons from three generations shaped google's tpuv4i : Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 1–14.

[12] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "Nvidia a100 tensor core gpu: Performance and innovation," *IEEE Micro*, vol. 41, no. 2, pp. 29–35, 2021.

[13] N. Challapalle, S. Rampalli, M. Chandran, G. Kalsi, S. Subramoney, J. Sampson, and V. Narayanan, "Psb-rnn: A processing-in-memory systolic array architecture using block circulant matrices for recurrent neural networks," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 180–185.

[14] A. NVIDIA, "Nvidia deep learning accelerator (nvdla)," 2017.

[15] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 44, no. 3, pp. 367–379, 2016.

[16] 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 Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.

[17] H. Cho, "Risa: A reinforced systolic array for depthwise convolutions and embedded tensor reshaping," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–20, 2021.

[18] B. Asgari, R. Hadidi, H. Kim, and S. Yalamanchili, "Eridanus: Efficiently running inference of dnns using systolic arrays," *IEEE Micro*, vol. 39, no. 5, pp. 46–54, 2019.

[19] B. Wang, S. Ma, Z. Liu, L. Huang, Y. Yuan, and Y. Dai, "Sadd: A novel systolic array accelerator with dynamic dataflow for sparse gemm in deep learning," in *IFIP International Conference on Network and Parallel Computing*. Springer, 2022, pp. 42–53.

[20] E. Baek, D. Kwon, and J. Kim, "A multi-neural network acceleration architecture," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 940–953.

[21] R. Xu, S. Ma, Y. Wang, Y. Guo, D. Li, and Y. Qiao, "Heterogeneous systolic array architecture for compact cnns hardware accelerators," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2860–2871, 2021.

[22] H. Genc, S. Kim, A. Amid, A. Haj-Ali, V. Iyer, P. Prakash, J. Zhao, D. Grubb, H. Liew, H. Mao *et al.*, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 769–774.

[23] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li, "Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 553–564.

[24] Z.-G. Liu, P. N. Whatmough, and M. Mattina, "Systolic tensor array: An efficient structured-sparse gemm accelerator for mobile cnn inference," *IEEE Computer Architecture Letters*, vol. 19, no. 1, pp. 34–37, 2020.

[25] S. Selvam, V. Ganesan, and P. Kumar, "Fuseconv: Fully separable convolutions for fast inference on systolic arrays," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 651–656.

[26] M. Soltaniyeh, R. P. Martin, and S. Nagarakatte, "Spots: An accelerator for sparse cnns leveraging general matrix-matrix multiplication," *arXiv preprint arXiv:2107.13386*, 2021.

[27] J. Fornt, P. Fontova-Musté, M. Caro, J. Abella, F. Moll, J. Altet, and C. Studer, "An energy-efficient gemm-based convolution accelerator with on-the-fly im2col," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[28] W. Liu, J. Lin, and Z. Wang, "Usca: A unified systolic convolution array architecture for accelerating sparse neural network," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[29] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A systematic methodology for characterizing scalability of dnn accelerators using scale-sim," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2020, pp. 58–68.

[30] H. Shao, J. Lu, M. Wang, and Z. Wang, "An efficient training accelerator for transformers with hardware-algorithm co-optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[31] R. Xu, S. Ma, Y. Wang, X. Chen, and Y. Guo, "Configurable multi-directional systolic array architecture for convolutional neural networks," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 18, no. 4, pp. 1–24, 2021.

[32] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.

[33] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[34] V. Ashish, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, p. I, 2017.

[35] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[36] S. Mach, F. Schuiki, F. Zaruba, and L. Benini, "Fpnew: An open-source multiformat floating-point unit architecture for energy-proportional transprecision computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 774–787, 2020.

[37] K. Chandrasekar, C. Weis, Y. Li, S. Goossens, M. Jung, O. Naji, B. Akesson, N. Wehn, and K. Goossens, "DRAMPower: Open-source DRAM Power & Energy Estimation Tool," http://www.drampower.info, accessed: 2024-08-17.

[38] A. Samajdar, E. Qin, M. Pellauer, and T. Krishna, "Self adaptive reconfigurable arrays (sara) learning flexible gemm accelerator configuration and mapping-space using ml," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 583–588.

[39] H. Kwon, A. Samajdar, and T. Krishna, "Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 461–475, 2018.

[40] N. K. Jha, S. Ravishankar, S. Mittal, A. Kaushik, D. Mandal, and M. Chandra, "Draco: Co-optimizing hardware utilization, and performance of dnns on systolic accelerator," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2020, pp. 574–579.

[41] Z. Wang, G. Wang, H. Jiang, N. Xu, and G. He, "Cosa: Co-operative systolic arrays for multi-head attention mechanism in neural network using hybrid data reuse and fusion methodologies," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.