An Isotropic Shift-Pointwise Network for Crossbar-Efficient Neural Network Design

Ziyi Guan[1*], Boyu Li[1*], Yuan Ren[1], Muqun Niu[1], Hantao Huang[2], Graziano Chesi[1], Hao Yu[2], and Ngai Wong[1]

[1]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
[2]School of Microelectronics, Southern University of Science and Technology, Shenzhen

***Abstract*—Resistive random-access memory (RRAM), with its programmable and nonvolatile conductance, permits compute-in-memory (CIM) at a much higher energy efficiency than the traditional von Neumann architecture, making it a promising candidate for edge AI. Nonetheless, the fixed-size crossbar tiles on RRAM are inherently unfit for conventional pyramid-shape convolutional neural networks (CNNs) that incur low crossbar utilization. To this end, we recognize the mixed-signal (digital-analog) nature in RRAM circuits and customize an isotropic shift-pointwise network that exploits digital shift operations for efficient spatial mixing and analog pointwise operations for channel mixing. To fast ablate various shift-pointwise topologies, a new reconfigurable energy-efficient shift module is designed and packaged into a seamless mixed-domain simulator. The optimized design achieves a near-100% crossbar utilization, providing a state-of-the-art INT8 accuracy of 94.88% (76.55%) on the CIFAR-10 (CIFAR-100) dataset with 1.6M parameters, which sets a new standard for RRAM-based AI accelerators.**

***Index Terms*—Shift operation, crossbar utilization, RRAM, algorithm-hardware co-design, isotropic architecture**

I. Introduction

Convolutional neural networks (CNNs) have exhibited exceptional performance in various computer vision tasks and applications [1], [2]. Nevertheless, their escalating size and complexity pose grand computational challenges, especially for edge devices. To this end, compute-in-memory (CIM) has emerged as a promising solution, with resistive random-access memory (RRAM) [3] being a leading contender due to its high density, reduced power, and compatibility with monolithic 3D integration. However, an often overlooked issue is the mismatch between standard CNNs and the RRAM infrastructure. Classical networks like ResNet [1] and DenseNet [2] feature the de facto pyramidal CNN configurations, leading to sub-optimal crossbar utilization and performance when realized on RRAM with fixed crossbar array sizes. From Fig. 1, when constrained to a parameter count below 2.5M for edge AI realization, most pyramidal CNNs demonstrate a crossbar utilization below 70% at 64 × 64 crossbar sizes. Such sub-optimal utilization considerably impacts the hardware performance of RRAM-based neural networks, leading to significant latency and diminished energy efficiency.

On the other hand, isotropic architectures like ConvMixer [4] and MLP-Mixer [5] have equal sizes and shapes for all elements throughout the network, showing better accuracy compared to pyramid CNN. However, these isotropic architectures suffer from a huge number of parameters and FLOPS, which pose challenges to deploying on resource-limited CIM devices. Meanwhile, a parameter-free shift operation [6] is proposed to replace spatial convolution without introducing parameters. Nonetheless, this operation can not be supported on RRAM crossbar, leading to low crossbar utilization.

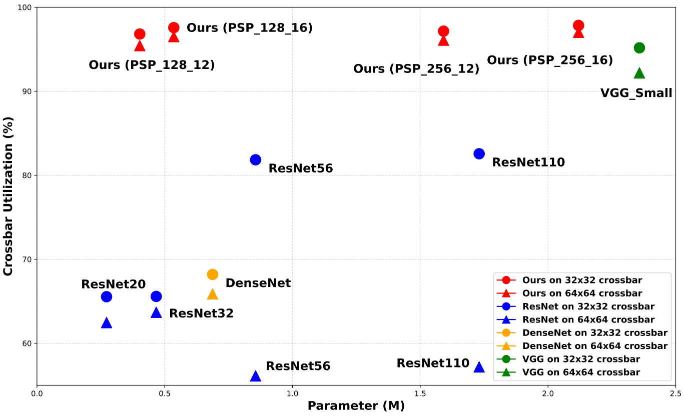


Fig. 1: Comparison of crossbar utilization between the proposed isotropic networks and some mainstream CNNs.

To fully harness the benefits of analog RRAM crossbars embedded in a digital system with peripheral ADCs and DACs, we first introduce the isotropic shift-pointwise network, namely Pointwise-Shift-Pointwise (PSP) and Shift-Pointwise (SP), which contrasts them with mainstream CNNs on typical crossbar sizes of 32×32 and 64×64 (cf. Fig. 1), maintaining a near-100% crossbar utilization at a reduced parameter count.

We first develop a mixed-domain simulator to accurately estimate the hardware and software performances of the proposed networks. This is achieved through the adoption of an algorithm-hardware co-design approach, which integrates the shift operation in the digital domain with dense pointwise convolutions in the analog domain. Depicted in Fig. 2, except for the patch embedding and final classifier, viz. the stem and head, PSP and SP maintain an isotropic architecture in their backbone which is the key to high RRAM crossbar utilization. The main contributions of this paper are:

- We are among the first to design a lightweight isotropic shift-pointwise network with near-100% RRAM crossbar utilization. The proposed PSP and SP networks outperform standard CNNs in model accuracy and hardware metrics.
- A novel reconfigurable and energy-efficient shift module is developed, enabling accurate characterization of the hardware metrics affiliated with the shift operation.

[*]Equal contribution.

- We utilize an algorithm-hardware co-design to exploit shift operation in digital domain for spatial mixing and pointwise operation in analog domain for channel mixing.

II. Related Work

A. Resistive Random-Access Memory

An RRAM macro consists of the RRAM crossbar and its peripheral ADC/DAC circuit blocks, forming a critical component of a CIM accelerator. Previous research has investigated crossbar-based CNN accelerators. Ref. [7] mentioned hardware costs but did not offer a comprehensive analysis of various metrics or an exhaustive hardware-aware evaluation. Ref. [8] explored the quantization of CNNs on RRAM crossbars. Nonetheless, they neither considered crossbar utilization nor ways to improve it. In fact, all these prior studies primarily focused on simulating and implementing standard CNNs on RRAM via pruning and/or quantization, without the RRAM utilization-oriented viewpoints emphasized in our work.

B. Shift Operation and Isotropic Architectures

The shift layer was initially proposed in [6] which employs a "zero-flop" shift operation to facilitate the exchange of spatial information between feature maps. Such shift-based networks have then been expanded in subsequent studies to variants like group shift [6], and active shift [9], etc. In the present study, a novel partial shift operation is employed in an isotropic structure when realized on the RRAM crossbar. Compared to previous work, the proposed shift block achieves higher model accuracy with less energy consumption and latency.

As mentioned in Section I, various frameworks have adopted isotropic architectures, such as MLP-Mixer [5] and ConvMixer [4]. These frameworks have demonstrated exceptional performance on large datasets. However, implementing these networks on RRAM is often prohibitive due to their substantial number of parameters. In this work, we propose novel PSP and SP frameworks as depicted in Fig. 2. Such shift plus pointwise combo amalgamates the simplicity and structure of the ConvMixer and MLP-Mixer, while circumventing the redundancy of the expensive kernel matrix. Specifically designed for RRAM arrays, our framework constitutes the first isotropic architecture that employs a digital spatial mixer and an analog channel mixer to attain high chip utilization and outstanding performance on an RRAM embedded system.

III. Design of the Isotropic Shift-Pointwise network

Inspired by ConvMixer [4] and MLP-Mixer [5], the isotropic architecture with a simple patch embedding stem is itself a powerful template for deep learning. Therefore, we follow the isotropic paradigm and improve the architecture performance with consideration of RRAM platform.

To investigate the impact of skip connection placement on overall hardware performance, we proposed two structures, PSP and SP. In the PSP architecture, a skip connection is placed between the first pointwise convolution layer and shift block, while in SP architecture, only a single skip connection is retained within the shift block, enabling the latter to be done purely digitally without extra hardware. In the SP network, the architecture is to emulate the token-mixer and channel-mixer structures in the MLP-Mixer. In contrast to MLP-Mixer, we employ a shift layer to replace the MLP, which largely reduces the number of parameters.

In the PSP network, we substitute the large kernel depthwise convolutions in ConvMixer with shift blocks and pointwise convolutions in the RRAM crossbar to maintain high crossbar utilization. The motivation for removing depthwise convolutions arises from their incompatibility with crossbars: given a kernel of size 3×3, during depthwise convolution, the `im2col` transform results in a block-diagonal mapping on crossbar cells, while wasting the off-diagonal storage which is enabled during crossbar column-wise read operations. Therefore, depthwise convolutions are practically incompatible for RRAM crossbars. The substitution results in a decrease in parameters while retaining comparable performance and, most importantly, achieving high crossbar utilization.

Illustrated in Fig. 2, the whole architecture is isotropic architecture and consists of four sequentially stacked components: pointwise convolution, GELU, batch normalization, and shift block. The blue dashed box represents the depth layers that are repeatedly stacked. Algorithm 1 depicts the shift block. This process involves taking an input tensor and shifting a subset of its channels by one pixel in nine spatial directions (left, right, up, down, left_up, right_up, left_down, and right_down and zero shift). During the shifting process, pixels outside the input tensors' bounds are discarded, and empty pixels are padded with zeros. As illustrated in Fig. 3(c), our approach is distinct from both the grouped shift in Fig. 3(a) and the active shift in Fig. 3(b). In particular, our shift block is tailored such that the initial 1/3 of channels undergo vertical and horizontal shifts (left, right, up, and down), the central 1/3 of channels exercise diagonal shifts (left_up, right_up,left_down, and right_down), and the final 1/3 remain unchanged. To improve the ability to mix spatial features and increase accuracy, the proposed shift block combines vertical and diagonal movements for a more comprehensive extraction of information in the spatial dimension. To reduce energy consumption associated with address and data change flow in digital circuits, a partial shift scheme is leveraged, affecting only 2/3 of the channels.

The remaining components are patch embedding, average pooling, and fully connected layers. Given an input image shape of $W \times H \times 3$, after the first patch embedding layers with 2×2 patch size, the image is split into halved width and height, viz. $\frac{W}{2} \times \frac{H}{2} \times 3$. The subsequent components follow the isotropic paradigm to remain equal in size and shape throughout the network, achieving the key of a near-100% RRAM crossbar utilization. The pointwise convolution which executes the basic matrix multiplication is regarded as a channel mixer and can be implemented on the analog RRAM crossbar. The shift block which exchanges spatial information across various channels, is regarded as a spatial mixer and can be implemented on the digital circuits. The specific design of the shift module and mapping will be discussed in Section IV.

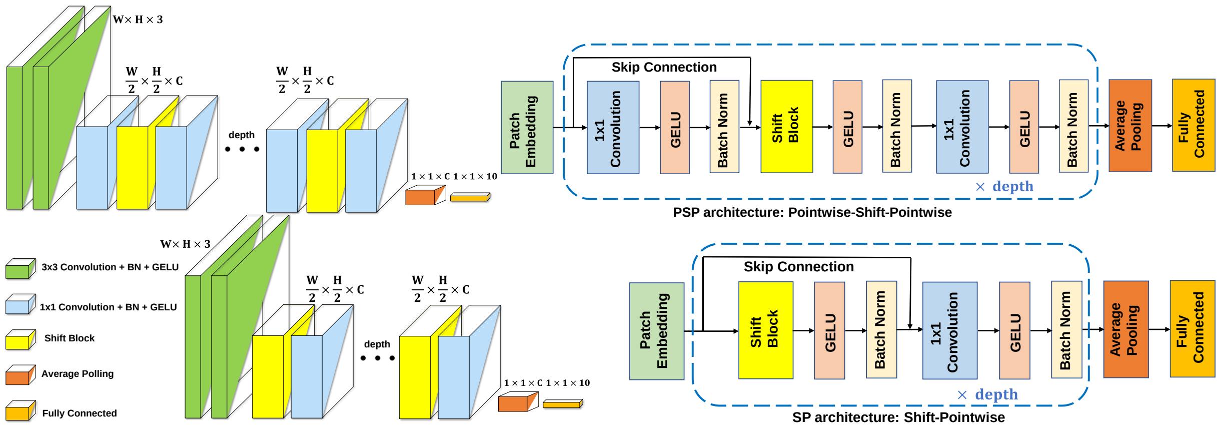


Fig. 2: Isotropic Shift-Pointwise Network Architecture: **(Upper)** Pointwise-shift-pointwise (PSP) and **(Lower)** Shift-pointwise (SP). Except for the stem and head, the number of channels in all layers remains unchanged to form an isotropic architecture.

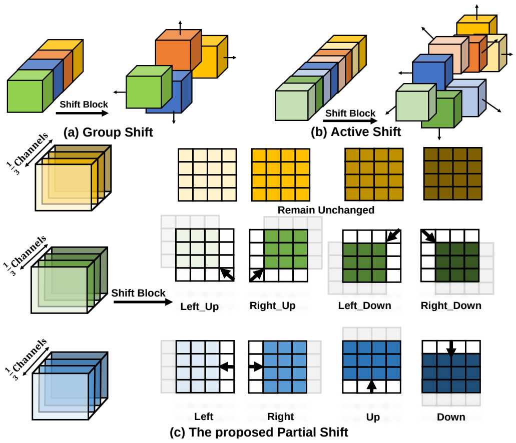


Fig. 3: Comparison of existing shift operations **(a)&(b)** and the proposed **(c)** which comprises 8 directions. 2/3 of all channels are shifted in 8 directions, limiting data movement energy consumption while up-keeping output accuracy.

Algorithm 1 Pytorch-based Pseudo code for Shift Block

Input: Input feature tensor x, with a shape of [Batch, Channel, Height, Width], γ a divider to divide the input feature map into nine parts.
Output: The feature map information after the shift block.

```
def shift($x, g = 1/12$):
    out = torch.zeros_like(x);
    # initial 1/3 of channels with vertical and horizontal shifts
    out$[:, g*0 : g*1, :, : -1] = x[:, g*0 : g*1, :, 1 :]$
    out$[:, g*1 : g*2, :, 1 :] = x[:, g*1 : g*2, :, : -1]$
    out$[:, g*2 : g*3, : -1, :] = x[:, g*2 : g*3, 1 :, :]$
    out$[:, g*3 : g*4, 1 :, :] = x[:, g*3 : g*4, : -1, :]$
    # central 1/3 of channels with diagonal shifts
    out$[:, g*4 : g*5, : -1, : -1] = x[:, g*4 : g*5, 1 :, 1 :]$
    out$[:, g*5 : g*6, :, 1 :] = x[:, g*5 : g*6, :, : -1]$
    out$[:, g*6 : g*7, 1 :, : -1] = x[:, g*6 : g*7, : -1, 1 :]$
    out$[:, g*7 : g*8, 1 :, 1 :] = x[:, g*7 : g*8, : -1, : -1]$
    # final 1/3 of channels with zero shifts
    out$[:, g*8 :, :, :] = x[:, g*8 :, :, :]$
    return out
```

IV. Design of the Hardware Shift Module

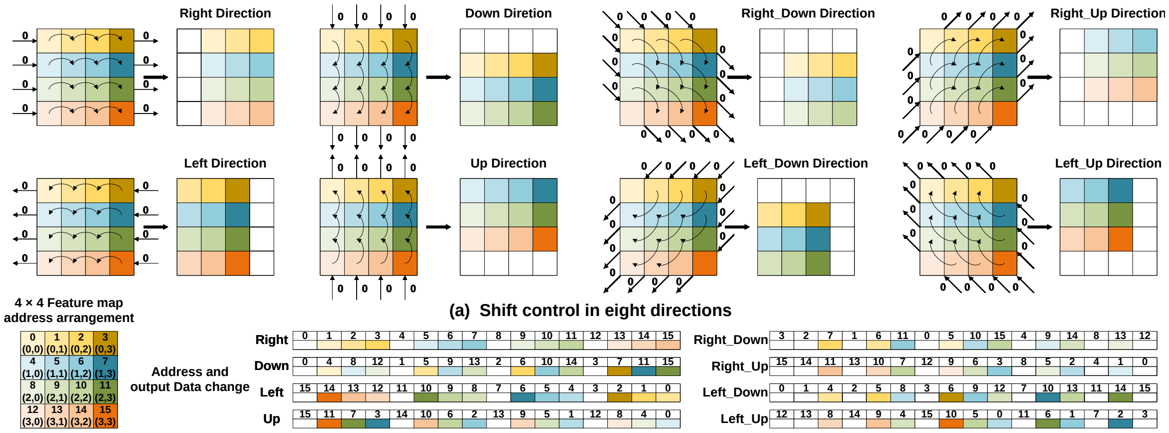
A. Architecture of Proposed Shift Module

Previous research [10] has implemented shift operations for feature maps by utilizing *linebuffers* which, however, is not specifically tailored for such purpose. Consequently, using linebuffer leads to substantial consumption of hardware resources and induces considerable delays. This in turn significantly undermines the energy efficiency of the whole-system design. To address this shortcoming, we present a novel shift module based on address pointer alterations. Different from the linebuffer structure, our proposed shift module achieves 8-way shifting using only an address control module plus a few registers. This module employs a parametric design, allowing its circuits to be custom-instantiated to support shift operations according to the required feature map sizes. Such design can readily be encapsulated as an Intellectual Property (IP) and embedded into other neural network accelerators requiring shift operations.

Referring to Fig. 4(a), the 8 direction shifts can be categorized into two groups: one comprising right, down, left, and up shifts, which require movement in a single direction; and another including right_down, right_up, left_down, and left_up shifts, which necessitates movement in two directions. Fig. 4(b) depicts the address change and output data rewritten to memory during different modes in the shift module. We consider a 4×4 feature map as an example. The 3D feature map is transformed into a 1D data array, following the order of width, height, and channel, and subsequently stored in the DRAM. The row and column addresses are used to control the DRAM address. During the shift operation, we leverage the unique capability of the Read First DRAM, which allows for simultaneous read and write operations within the same cycle, effectively halving the time required for the shift module. When the boundary address is reached, a value of 0 is written to the DRAM, while at other times, the data from the previous-cycle address will be written to the current address.

B. Circuit Implementation

The logic of the shift hardware is shown as follows: The Initialize function resets the entire system. By default, the shift module is in the idle state, and it enters the config state once the shift start signal is enabled. After configuring the first read/write address in the config state, the module transitions into the data shift state in the next cycle which processes a channel's feature map, sends a shift end signal, and reenters the idle state before shifting the next channel. This process continues until all shift blocks are shifted as shown in Fig. 3(c).


Right Direction
Down Diretion
Right_Down Direction
Right_Up Direction
Left Direction
Up Direction
Left_Down Direction
Left_Up Direction
0
4 × 4 Feature map address arrangement
0 (0,0) 1 (0,1) 2 (0,2) 3 (0,3)
4 (1,0) 5 (1,1) 6 (1,2) 7 (1,3)
8 (2,0) 9 (2,1) 10 (2,2) 11 (2,3)
12 (3,0) 13 (3,1) 14 (3,2) 15 (3,3)
Address and output Data change
Right 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Down 0 4 8 12 1 5 9 13 2 6 10 14 3 7 11 15
Left 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Up 15 11 7 3 14 10 6 2 13 9 5 1 12 8 4 0
Right_Down 3 2 7 1 6 11 0 5 10 15 4 9 14 8 13 12
Right_Up 15 14 11 13 10 7 12 9 6 3 8 5 2 4 1 0
Left_Down 0 1 4 2 5 8 3 6 9 12 7 10 13 11 14 15
Left_Up 12 13 8 14 9 4 15 10 5 0 11 6 1 7 2 3

(a) Shift control in eight directions

(b) Address change and output data rewritten to memory in shift module

Fig. 4: Addressing mechanism and data flow in the proposed shift module: (a) Changes of the address pointer on the feature map in 8 different shift modes; (b) Address change and output data rewritten to memory in shift module in 1-dimension.

The starting address configured in the config state and the address change in each cycle during the data shift state vary for different shift modes. For unidirectional shift operations, the row and column addresses configured in the config state for right and down shifts are both 0. In the right mode, the column address increases by one with each clock cycle change until it reaches the feature map column number. Then the column address will reset to 0, and the row address increases by 1, with the data written to the DRAM also set to 0 in the next cycle. This operation continues until both the row and column addresses reach the feature map's maximum value. In down mode, the row address increases by 1 in each cycle first, followed by the column address change. For left and up shifts, the row and column addresses configured in the config state are the row and column numbers, with the row or column address decreased by 1 in each cycle.

For compound shift operations, a straightforward approach is to combine moving up or down and moving left or right in directional shifts, but this would double the delay. To address this issue, we designed the address change logic as shown in Fig. 4(b). In the config state, the read and write addresses of the four compound shift configurations are located at the four corners of the feature map. After entering the data shift state, both the row and column addresses change simultaneously, either increasing or decreasing by 1, or one increasing while the other decreases which enables diagonal data shifting. When boundary conditions are reached, the data written to DRAM in the next cycle will be set to 0.

C. Mapping Isotropic Networks on RRAM Crossbar

NeuroSim [11], a state-of-the-art simulator for RRAM-based CNNs, currently lacks the ability to support shift operations and estimate the associated hardware overhead. Hence, incorporating the shift module and integrating it into a mixed-domain simulator is crucial for hardware profiling and optimization. To optimize the isotropic shift-pointwise network for the RRAM platform, we employ an algorithm-hardware co-design that successfully enables the deployment of neural networks leveraging both analog and digital domain operations. Initially, the input feature from the Dynamic random access memory (DRAM) is transferred to the DACs for digital signal to analog voltage conversion. After conversion, the product of the voltage and the conductance of each RRAM cell is added through Kirchhoff's law to obtain the current. Afterwards, the current is sampled by the ADCs to obtain the digital signals and then input to adder and shift register to get the output feature. Finally, the output feature is restored to DRAM after being shifted by our proposed shift module. In PSP inference process, the first pointwise convolution extracts features in the channel dimension. Meantime, the feature map undergoes shift operations through the shift module to perform spatial mixing. After the input feature has been operated in both digital and analog domain, the output feature then returns to the DRAM and restarts the next cycle.

V. Experiments

A. Experiment Setup

Our experiments are conducted on the CIFAR-10/100 [13], and ImageNet [14] datasets, utilizing the proposed PSP and SP network architectures for training and validation on the PyTorch 1.7.1 platform. Experiments are conducted using one NVIDIA RTX 3090 card equipped with a 32GB frame buffer. To accurately assess the system performance, we meticulously recreated the entire inference process employing a mixed-domain simulator, which combines both the Neurosim and shift module. The RRAM crossbar size in Neurosim is configured as 64×64, whose experimental findings and observations generalize to other popular crossbar sizes. Meantime, we instantiate a shift module of 16×16 to match the feature size and implement it using the SMIC 28nm process library with typical parameters (1.00V and 25°C) through the Synopsys Design Compiler. We set our clock frequency to be inline with that of the Neurosim. After obtaining the netlist file, we extract the feature maps for all channels requiring shifting across all blocks of a network inference image as a testbench. This testbench is input into Synopsys VCS to generate a waveform diagram, which is then fed into Primetime to accurately characterize the shift module's dynamic power consumption. Furthermore, the shift module has undergone functional verification and resource consumption analysis on the Artix-7 AC701 FPGA.

TABLE I: Comparison of PSP and SP networks vs. mainstream CNNs (ResNet, VGG, and DenseNet40) trained on CIFAR-10 and deployed on 64×64 RRAM crossbars.

Model	Parameters(M)	Top-1 Accuracy(%)	Crossbar Utilization(%)	Latency(ms)	Energy Efficiency(Tops/W)	Chip Area(mm^2)
ResNet110 [1]	1.73	94.52	57.18	9.82	1.89	23.18
DenseNet (40,12) [2]	0.17	91.04	60.53	11.00	3.32	35.50
VGG8 [12]	12.97	90.58	99.39	5.00	4.83	284.18
ShiftResNet110 [6]	0.20	90.67	57.18	10.04	1.88	23.18
PSP_128_8	0.26	92.97	93.40	2.40	6.90	6.84
PSP_256_8	1.06	94.24	94.29	3.57	**7.42**	12.52
PSP_256_12	1.59	94.98	96.07	5.33	7.34	17.91
SP_256_24	1.60	95.21	96.07	6.43	4.68	26.80
PSP_256_16	2.12	**95.64**	97.01	8.13	6.86	23.62

TABLE II: Comparison of various PSP and SP architectures and ResNet for w.r.t. QAT Top-1 Accuracy for FP32, INT8 and INT4 on CIFAR-10/100 datasets.

Model	Parameters(M)	FP32(%)	INT8(%)	INT4(%)
ResNet32	0.47	92.61/71.05	92.76/70.11	92.39/69.78
ResNet110	1.73	94.52/73.44	92.76/71.44	92.39/71.20
PSP_128_12	0.40	93.73/74.22	93.62/72.05	92.76/70.94
PSP_256_12	1.59	94.98/**77.94**	**94.88/76.55**	94.50/**76.10**
PSP_256_16	2.12	**95.64**/77.86	94.72/76.02	94.59/75.64

B. Algorithm Performance Comparison of Model Accuracy

We train multiple networks based on the PSP and SP architectures with two varying factors: (1) the width or hidden dimension h, representing the dimensionality of patch embeddings, and (2) the depth d, indicating the number of PSP and SP layer repetitions. The naming conventions for PSP and SP bodies are designated as PSP_h_d and SP_h_d, respectively, based on their hidden dimension and depth. For example, in Table I, PSP_256_16 denotes PSP with a width of 256 and a depth of 16, achieving the highest accuracy (95.64%) on CIFAR-10 and the maximum crossbar utilization (97.01%) within 2.1M parameters.

Then, to implement the lightweight models on RRAM, we employ quantization-aware training (QAT) [15] to quantize the model weights and activations. This leads to a significant reduction in the bit-width of the model weights and activations without sacrificing much accuracy. The QAT results (including FP32, INT8, and INT4) compared with ResNet32 and ResNet110 on CIFAR10/100 datasets are shown in Table II. Under similar parameter settings, our model outperforms both ResNet32 and ResNet110 in FP32, INT8, and INT4 QAT Top-1 accuracy. Fig. 5 depicts the different QAT configurations of PSP and SP architectures applied to the CIFAR-10 dataset. The graph illustrates that the PSP_256_8 network exhibits the least accuracy degradation after quantization, with a maximum drop of only 0.3% in accuracy at either INT8 or INT4.

Subsequently, to substantiate the efficacy of the proposed models on large-scale datasets, we evaluate the performance of the PSP architectures on ImageNet. Noted that all training schedules are from scratch without any data augmentation. As enumerated in Table III, with a relatively low number of parameters, PSP_512_16 can achieve a top-1 accuracy of 74.87% and PSP_256_12 reaches 72.20% within 1.8M parameters, further validating the versatility of the PSP architecture.

TABLE III: Comparison of various PSP architectures against mainstream CNNs such as ResNet, MobileNet w.r.t. Top-1 Accuracy on ImageNet dataset.

Model	Parameters(M)	Top-1 Accuracy(%)
ResNet18 [1]	11.17	69.15
MobileNetV1 [16]	4.2	70.60
PSP_256_12	1.8	72.20
PSP_512_12	6.32	73.18
PSP_512_16	8.43	**74.87**

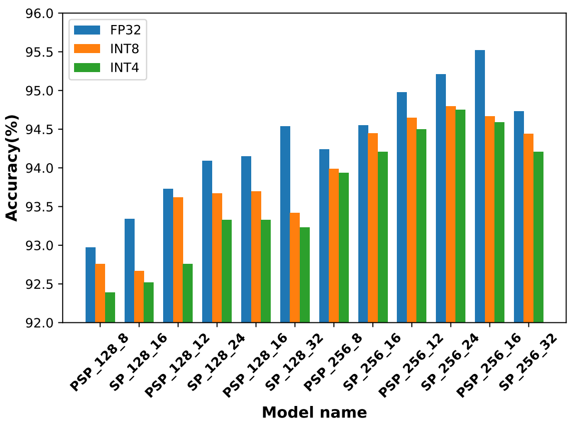


Fig. 5: Comparison between FP32, INT8 and INT4 among various PSP and SP architectures on CIFAR-10 dataset.

C. System Performance Comparison of Hardware Metrics

The shift module boasts an area of 100.058um^2, a critical path delay of 0.65ns, and a low static power consumption of only 0.138W. The energy and latency of five different models in the shift module are comprehensively presented in Table IV. For the first time, we have systematically determined the hardware consumption of the shift module within the overall network flow. Neurosim employs a 22nm technology in its RRAM-based chip, whereas the shift digital domain utilizes a 28 nm technology. To ensure a fair comparison of hardware tests and energy consumption, we use the DeepScaleTool for accurate estimation of deep-submicron technology scaling [17]. Additionally, the shift module incorporates 102 Look-Up Tables (LUTs), 23 Flip-Flops (FFs), and 33 IO pins in the FPGA implementation.

To corroborate the efficacy and robustness of the proposed PSP and SP networks, we benchmark their performance against mainstream CNNs such as ResNet, VGG, and DenseNet in

TABLE IV: Comparison of energy consumption and latency of different models in the shift module.

Model	Energy(nJ)	Normalized Energy(nJ)	Latency(ms)
PSP_128_16	66.800	54.177	0.04301
PSP_256_16	215.000	174.371	0.12902
SP_256_32	284.000	230.333	0.17203

terms of model parameters, Top-1 accuracy, crossbar utilization, latency, energy efficiency, and chip area for the CIFAR-10 dataset, cf. Table I. Compared with classical ResNet110, DenseNet (40,12), and VGG8, our models with similar parameter counts exhibit superior performance across all the software and hardware metrics. For instance, PSP_256_12 exhibits a more advanced performance in contrast to ResNet110. In particular, the model's accuracy experiences a 0.46% improvement, the crossbar utilization witnesses a 38.89% enhancement, latency is diminished by 4.48ms, energy efficiency sees a 5.45TOPS/W increase, and the chip area is reduced by 5.27mm^2.

To validate the effectiveness of the isotropic structure and shift blocks, we also compare our models with ShiftNet [6], For a fair comparison, we employ a consistent mixed-domain simulator to facilitate the examination under the same constraints of the ShiftNet110 [6], which features a PSP-like structure. Compared to ShiftNet110 with nearly identical parameters, our PSP_128_8 outperforms it in all software and hardware metrics. In particular, PSP_256_8 is regarded as the optimal RRAM-based model as it achieves the best energy efficiency and comparable accuracy with just 1M parameters.

To further verify the crossbar utilization of PSP networks w.r.t crossbar utilization, we compare them with other advanced CIM accelerator works. At 256 × 256 fixed crossbar sizes, Chen et al. [11] measures the MLP, ResNet50, ResNet152, and DenseNet (100,24) attaining the utilization of 65%, 56%, 46%, 42%, respectively. Krishnan et al. [18] optimized the utilization of these models, reporting 81%, 80%, 76%, and 66%. Our PSP networks, PSP_256_16 and PSP_256_12, outperformed these models with crossbar utilization of 95% and 93%, respectively, under the same constraints.

VI. Conclusion

This work devises a novel mixed-domain digital-analog architecture that leverages an isotropic shift-pointwise network (viz. PSP or SP) to achieve near-100% crossbar utilization, leading to a lower latency and higher energy efficiency versus conventional pyramidal CNNs. The proposed PSP and SP networks further capitalize on the mixed-domain attributes of RRAM circuits, namely, by exploiting digital shifts for efficient spatial mixing and analog RRAM pointwise operations for channel mixing. A reconfigurable and energy-efficient shift module is newly designed to enable streamlined shift operations. Through an algorithm-hardware co-design, the proposed RRAM-optimized lightweight models demonstrate clear advantages over existing RRAM-based CNNs, demonstrating a state-of-the-art INT8 accuracy of 94.88% (76.55%) on the CIFAR-10 (CIFAR-100) dataset with only 1.6M parameters, thus setting a new yardstick for RRAM-based AI accelerators.

VII. Acknowledgement

This work was supported by the Mainland-Hong Kong Joint Funding Scheme Project MHP/066/20, the Theme-based Research Scheme (TRS) project T45-701/22-R, and in part by the General Research Fund (GRF) project 17209721 of the Research Grants Council (RGC), Hong Kong SAR.

References

[1] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] G. Huang *et al.*, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[3] W. Wu *et al.*, "A methodology to improve linearity of analog RRAM for neuromorphic computing," in *2018 IEEE symposium on VLSI technology*. IEEE, 2018, pp. 103–104.

[4] A. Trockman and J. Z. Kolter, "Patches are all you need?" *Transactions on Machine Learning Research*, 2023.

[5] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24 261–24 272, 2021.

[6] B. Wu *et al.*, "Shift: A zero flop, zero parameter alternative to spatial convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9127–9135.

[7] C. Chu, Y. Wang, Y. Zhao, X. Ma, S. Ye, Y. Hong, X. Liang, Y. Han, and L. Jiang, "PIM-Prune: Fine-Grain DCNN Pruning for Crossbar-Based Process-In-Memory Architecture," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[8] X. Ma, G. Yuan, S. Lin, C. Ding, F. Yu, T. Liu, W. Wen, X. Chen, and Y. Wang, "Tiny but Accurate: A Pruned, Quantized and Optimized Memristor Crossbar Framework for Ultra Efficient DNN Implementation," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 301–306.

[9] Y. Jeon and J. Kim, "Constructing fast network through deconstruction of convolution," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[10] Y. Yang *et al.*, "Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded fpgas," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 23–32. [Online]. Available: https://doi.org/10.1145/3289602.3293902

[11] P.-Y. Chen, X. Peng, and S. Yu, "Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067–3080, 2018.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, 2015.

[13] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[14] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[15] C. Tao *et al.*, "FAT: Frequency-aware transformation for bridging full-precision and low-precision deep representations," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[16] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 4510–4520.

[17] S. Sarangi *et al.*, "Deepscaletool: A tool for the accurate estimation of technology scaling in the deep-submicron era," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[18] G. Krishnan, S. K. Mandal, C. Chakrabarti, J.-s. Seo, U. Y. Ogras, and Y. Cao, "Interconnect-aware area and energy optimization for in-memory acceleration of dnns," *IEEE Design & Test*, vol. 37, no. 6, pp. 79–87, 2020.