SparSynergy: Unlocking Flexible and Efficient DNN Acceleration through Multi-Level Sparsity

Jingkui Yang[1, 2], Mei Wen✉[1, 2], Junzhong Shen[1, 2], Jianchao Yang[1, 2],
Yasong Cao[1, 2], Jun He[1, 2], Minjin Tang[1, 2], Zhaoyun Chen[1, 2], and Yang Shi[1, 2]
[1] *College of Computer Science and Technology, National University of Defense Technology, Changsha China*
[2] *Key Laboratory of Advanced Microprocessor Chips and Systems, Changsha China*
{yangjingkui2001,meiwen,shenjunzhong,yangjianchao16,caoyasong,hejun19,tangminjin,chenzhaoyun,shiyang14}@nudt.edu.cn

***Abstract*—To more effectively address the computational and memory requirements of deep neural networks (DNNs), leveraging multi-level sparsity—including value-level and bit-level sparsity—has emerged as a pivotal strategy. While substantial research has been dedicated to exploring value-level and bit-level sparsity individually, the combination of both has largely been overlooked until now. In this paper, we propose SparSynergy, which—to the best of our knowledge—is the first accelerator that synergistically integrates multi-level sparsity into a unified framework, maximizing computational efficiency and minimizing memory usage. However, jointly considering multi-level sparsity is non-trivial, as it presents several challenges: (1) increased hardware overhead due to the complexity of incorporating multiple sparsity levels, (2) bandwidth-intensive data transmission during multiplexing, and (3) decreased throughput and scalability caused by bottlenecks in bit-serial computation. Our proposed SparSynergy addresses these challenges by introducing a unified sparsity format and a co-optimized hardware design. Experimental results demonstrate that SparSynergy achieves a 5.38× geometric mean improvement in the energy-delay product (EDP) when compared with the tensor core, across workloads with varying degrees of sparsity. Furthermore, SparSynergy significantly improves accuracy retention compared to state-of-the-art accelerators for representative DNNs.**

***Index Terms*—DNN acceleration, sparsity, hardware-software co-design, computer architecture**

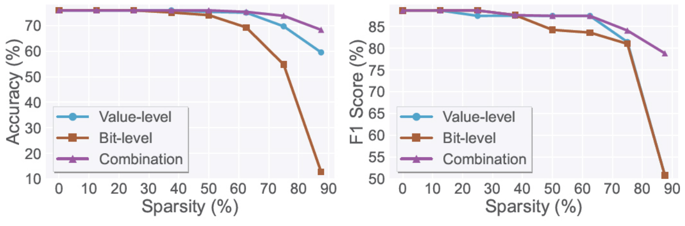
I. Introduction

To address the storage and computational challenges posed by DNN workloads [1], a common approach is to leverage the inherent sparsity within DNNs. This includes value-level sparsity, as discussed in [2]–[5], and bit-level sparsity, as highlighted in [6]–[10]. Techniques such as weight pruning [11], which removes redundant parameters to create value-level sparsity, and quantization [12], which reduces the precision of weights and activations to achieve bit-level sparsity, are often used. Both of these methods serve to effectively reduce the computational demands and memory requirements of DNNs.

Due to the distinct characteristics of DNNs, it is desirable to achieve flexible sparsity support to avoid accuracy degradation as different models and their respective layers may exhibit different sparsity patterns [4]. However, enabling such flexibility involves a trade-off with increased hardware overhead, which includes the additional area cost, and power consumption [13]. Balancing flexibility with hardware overhead is indeed challenging. Fixed sparsity patterns simplify hardware design and reduce overhead but at the cost of limited adaptability, which can lead to suboptimal performance when models exhibit different sparsity degrees. Conversely, more flexible sparsity designs can adapt to varied sparsity patterns and degrees but may incur significantly higher costs in terms of area, power, and latency. As illustrated in Tab. I, different accelerators offer various trade-offs, with some prioritizing flexibility at the expense of increased overhead, while others aim to minimize hardware complexity. In essence, *accelerating DNNs efficiently while accommodating diverse sparsity remains a core challenge in the design of many accelerators.*

TABLE I: Comparison of designs from different categories.

Categories	**Representative Designs**	**Overhead**	**Flexibility**
Value-level	STC [16]	**Very Low**	**Low**
Sparsity	SIGMA [2]	**High**	**Very High**
Bit-level	BitWave [7]	**Low**	**Medium**
Sparsity	Bitlet [8]	**High**	**Very High**
Combination	**SparSynergy**	**Low**	**High**



(a) ResNet-50 (b) BERT-base

Fig. 1: Accuracy comparison of ResNet-50 and BERT-base models under various sparsity levels. The weights are sparsified by setting a unified $N:8$ sparsity pattern.

Our key insight is that value-level and bit-level sparsity are orthogonal, which means they can be leveraged simultaneously to achieve higher accuracy at the same sparsity degree. This combined approach offers greater flexibility, enabling maximal retention of the representational ability of weights in DNN models. Fig. 1 illustrates the accuracy comparison across different sparsity degrees in ResNet-50 [14] and BERT-base [15], which are representative of CNN- and Transformer-based models, respectively. The results clearly demonstrate that the combined approach consistently outperforms the methods that utilize a single level of sparsity across all tested degrees. While both value-level and bit-level sparsity lead to significant accuracy degradation as sparsity increases, *the combined approach allows for significantly better accuracy retention.*

However, most existing accelerators [2], [3], [5], [7], [8], [10], [16] are primarily designed to optimize either value-level or bit-level sparsity, making it difficult to fully harness the

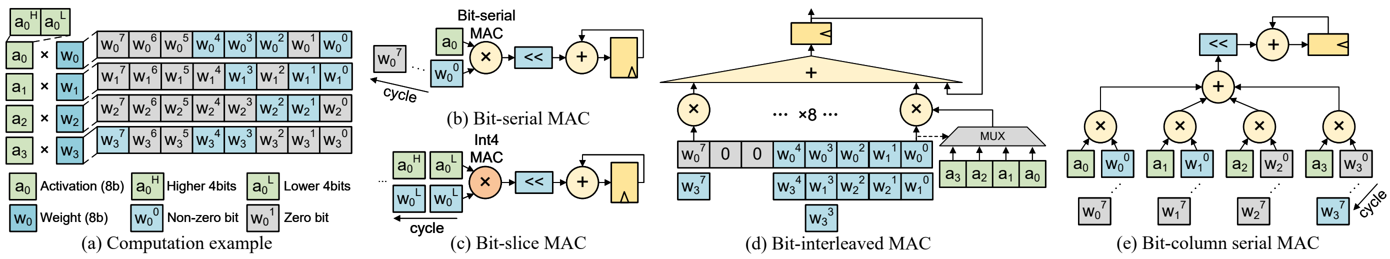


Fig. 2: The comparison among four types of bit-level MAC.

combined benefits of multi-level sparsity integration. Moreover, directly incorporating multi-level sparsity into hardware introduces several challenges: (1) additional hardware complexity and overhead in managing both forms of sparsity concurrently, (2) the need for high bandwidth to accommodate irregular data transmission caused by value-level sparsity, alongside inefficient bandwidth utilization due to bit-level sparsity, and (3) compromised throughput and scalability resulting from the inherent limitations in parallelism of bit-by-bit operations, compounded by synchronization difficulties and bandwidth bottlenecks. These challenges pose substantial obstacles to maximizing performance.

In this paper, we introduce SparSynergy, a novel architecture designed to leverage multi-level sparsity. SparSynergy overcomes these challenges, enabling flexible and effective DNN acceleration and setting a new standard for cutting-edge DNN acceleration. Our key contributions are summarized as follows:

- We represents a novel approach by leveraging both bit-level and value-level sparsity simultaneously to achieve higher accuracy at equivalent sparsity degrees, proving its superiority and feasibility.
- We introduce the *unified sparsity format* that enables a flexible representation of various sparsity degrees and an efficient hardware design by orthogonally combining structured sparsity from multiple levels.
- We present *SparSynergy*, a flexible and efficient design that exploits multi-level sparsity with minimal overhead.

The experimental results indicate that SparSynergy achieves up to a 5.38× geometric mean reduction in energy-delay product (EDP) across various degrees of sparsity and is positioned on the Pareto frontier for EDP-accuracy when considering representative models.

II. Background

A. Value-Level Accelerators

Existing value-level accelerators often face the trade-off between preserving model accuracy and maximizing hardware acceleration. To improve hardware efficiency, such accelerators typically employ straightforward sparsity patterns like $2:4$ pattern in STC [16] and $N:8$ in S2TA [3]. However, these fixed and conservative pruning strategies may not be effective for models capable of higher sparsity degrees. HighLight [4] stands out as the first accelerator harnessing hierarchical structured sparsity, achieving flexible sparsity support at a relatively low cost associated with sparsity.

In contrast, hardware accelerators that support complex sparsity patterns can maintain higher accuracy due to their flexibility. However, this flexibility introduces greater complexity into the hardware design. As a result, such accelerators may be less efficient for models with lower degrees of sparsity. For instance, DSTC [5] requires large accumulation buffers to manage the irregular distribution of output values, whereas SIGMA [2] employs complex distribution networks. These design choices can lead to significant routing and reorganization complexity, ultimately leading to increased power consumption and higher hardware expenses.

B. Bit-Level Accelerators

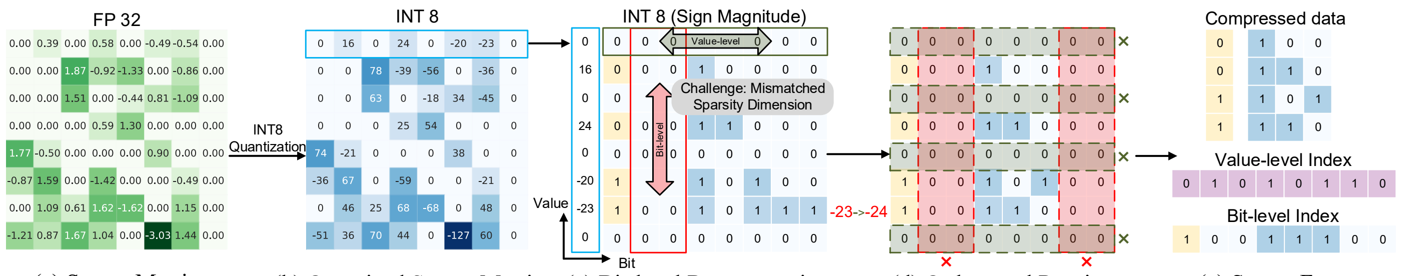
As illustrated in Fig.2, bit-level accelerators utilize specialized multiply-accumulate units (MACs) to skip computations involving zero bits. However, they face inherent limitations. Bit-serial MACs (Fig. 2(b)) operate by decomposing multi-bit operands into individual bits, transforming multiplications into a series of sequential additions. Despite this capability, they encounter synchronization issues and suffer from inefficient encoding schemes, which limit their efficiency, as discussed in [9]. Bit-slice MACs (Fig. 2(c)), exemplified by SPARK [17], store and process data in INT4 format, which eliminates the need for indexing overhead. However, they often lack flexibility and have limited applicability. Bit-interleaving MACs (Fig. 2(d)), such as those seen in Bitlet [8], aim to improve throughput by interleaving non-zero bits. Although this approach successfully skips zero bits, it incurs significant runtime processing to extract the indices of non-zero weight bits, leading to increased memory overhead and decreased efficiency. Bit-column serial MACs (Fig. 2(e)) achieve high efficiency by grouping every eight elements and employing bit-columns as the unit of storage and computation, while simultaneously skipping zero bits.

III. Orthogonal Sparsity Pruning

A. Sparse Dimension Mismatch

Fig.3 delineates the process of exploiting both value-level and bit-level sparsity. Upon applying pruning techniques to dense DNNs followed by retraining, numerous non-zero elements, which minimally impact prediction accuracy, are removed, as exhibited by the sparse matrix in Fig. 3(a). The substantial presence of zero-value elements significantly reduces the computational load during matrix operations.

Further optimization can be achieved by exploring bit-level sparsity, which focuses on the zero bits within the binary representation of non-zero elements. Additionally, quantization strategies can confine values within a narrower range, thus potentially increasing the number of zero bits within their binary representation. For instance, the sparse matrix in Fig.3(b), quantized from FP32 data precision to INT8, displays its first



(a) Sparse Matrix (b) Quantized Spares Matrix (c) Bit-level Representation (d) Orthogonal Pruning (e) Sparse Format

Fig. 3: **Visualization of orthogonal sparsity combination on an** 8×8 **weight matrix.** (a) Sparse matrix after pruning. (b) INT8-quantized sparse matrix. (c) Bit-level sparsity in the first row of the quantized sparse matrix, using sign-magnitude representation, demonstrating value-level sparsity along rows and bit-level sparsity in coordinates. (d) Orthogonal pruning achieved through bit-column pruning, which removes zero bits along columns. This combination aligns the granularity of bit-level and value-level sparsity. (e) Sparse format after orthogonal pruning.

row of elements in a bit-level representation as shown in Fig. 3(c). A significant number of zero bits in the horizontal direction appear in the binary representation of each value. This bit-level compression introduces finer granularity, complicating data encoding and decoding processes.

B. *Orthogonal Pruning and Sparse Format*

Our objective is to simultaneously prune data at both value-level and bit-level sparsity. Fig. 3(d) illustrates the concept of orthogonal pruning, which is capable of identifying and removing unimportant elements within value-level sparsity (i.e., zero elements, marked by greenish gray boxes) as well as unimportant bits within bit-level sparsity (i.e., zero bits in certain binary positions, marked by red boxes). To address the challenge of finer granularity in bit-level sparsity, we employ bit-column compression techniques [7], as indicated by the red boxes in Fig.3(d), to eliminate unimportant bits. This provides a coarser granularity of bit-level sparsity.

In the process of bit-level pruning, we prioritize pruning bit-columns that are entirely zero, followed by pruning trailing bit-columns based on the chosen $N : 8$ sparsity. This approach ensures minimal loss in accuracy, as demonstrated in Fig.3(d) where the penultimate element is pruned from -23 to -24. Additionally, we have validated this methodology using complete DNNs as shown in Fig. 1 (§ I), confirming that orthogonal pruning significantly enhances accuracy retention compared to employing either value-level or bit-level pruning alone.

We further introduce the concept of **orthogonal sparse format** for efficient storage and computation of data following orthogonal pruning, as illustrated in Fig.3(e). This format transforms the original dense data into three independent streams, comprising compressed data, value-level indexes, and bit-level indexes. Compressed data holds the non-zero elements and their bits in the binary representation remaining after pruning, value-level indexes indicate the positions of these non-zero elements in the original matrix, and bit-level indexes reveal which bits of the non-zero values are valid. Through this format, we are able to store and manipulate only the necessary information, thereby significantly reducing storage space and enhancing computational efficiency.

IV. Unified Sparsity Format

A. *Unified Format Specification*

The varying degrees and patterns of sparsity exhibited by different DNN models and workloads [4] highlight the increasing need for flexible sparsity support. However, providing such flexibility in multi-level sparsity presents a significant challenge for hardware design, as each level requires separate indexing and hardware support. In § III, we propose a unified sparsity format that enables the coexistence of multiple sparsity levels in a structured manner. By leveraging the orthogonal relationship between value-level and bit-level sparsity, each type can be pruned independently, avoiding interference or redundancy. This independence allows the hardware accelerator to optimize data representation and processing across various DNN workloads.

To strike a balance between the flexibility of the sparsity format and the efficiency of hardware implementation, we have constrained the sparsity format to the following structure: $(N_1 : M) \rightarrow (N_2 : M, N_3 : 8)$, as depicted in Fig. 4. The $(N_1 : M)$ format represents vector-level sparsity within a subtensor, where each vector of M elements contains N_1 non-zero entries. The patterns $(N_2 : M)$ and $(N_3 : 8)$ correspond to distinct levels of sparsity within the individual vectors: $(N_2 : M)$ denotes value-level sparsity, and $(N_3 : 8)$ targets sparsity at the bit-column level, enabling compression to finer bit-width granularity. This sparsity format provides a flexible representation that can adapt to the varying degrees of sparsity across different DNN models. It supports dynamic adjustments, reducing overhead between different sparsity patterns while preserving computational accuracy.

B. *Hardware Support at Each Sparsity Level*

To efficiently handle multi-level sparsity while maintaining computational performance, we describe how the hardware accommodates it within a unified framework. Each processing unit (PE) is assumed to perform matrix multiplication using subtensors and sustain an output stationary data flow with local partial sum registers.

To achieve vector-level sparsity, which skips entire rows of computation, we adopt memory access logic for partial sum registers combined with vector indexing. To prevent overflow in the partial sum registers, we use an $N : 8$ format with

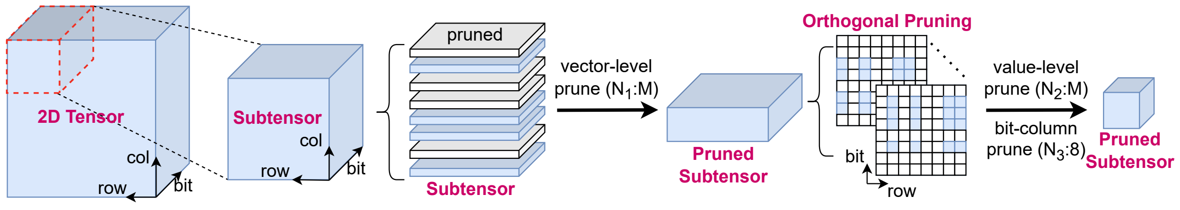
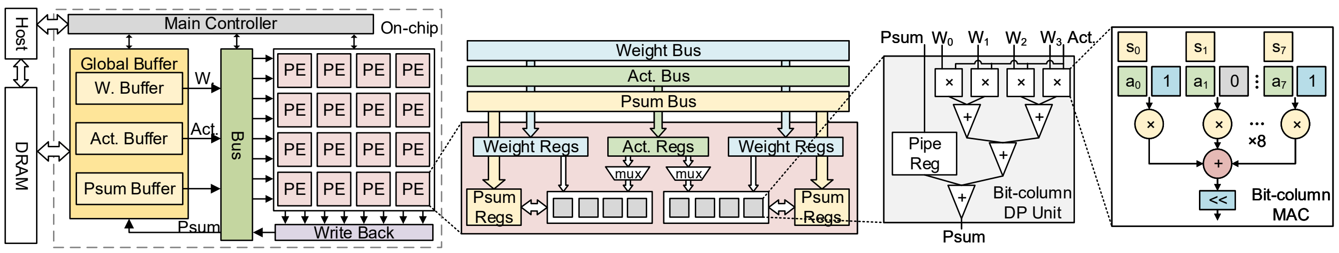


Fig. 4: General concept of unified sparsity format.



(a) High level architecture of accelerator. (b) Design details of PE. (c) Bit-column DP Unit. (d) Bit-column MAC.

Fig. 5: Design details of SparSynergy architecture.

a fixed constraint on M (the maximum number of rows the partial sum registers can store), thus avoiding frequent memory replacement.

For value-level sparsity, we employ the $2 : M\{M \leq 4\}$ format to ensure full utilization of the computational units, with multiplexers controlled by value-level indexing. Bit-level sparsity is exploited through bit-column-serial MAC units that detect and skip zero bit-columns. To mitigate the increased activation bandwidth and load imbalance introduced by the bit-serial mechanism, we unfold the bit-serial computation into a fixed $4 : 8$ format or use dense representations.

The hardware architecture manages multi-level sparsity synergistically within a unified framework, enabling dynamic resource allocation. Overall, the framework supports 48 distinct sparsity configurations, optimizing hardware complexity and minimizing overhead when handling both value-level and bit-level sparsity simultaneously.

V. Hardware Architecture

A. High Level Architecture

The top-level architecture of SparSynergy consists of several key components, as shown in Fig.5:

- The *On-Chip Global Buffer* serves as the primary memory storage unit, holding weights, activations, and partial sums before being dispatched to the PEs. It ensures that data is readily accessible for computation, minimizing data retrieval latency.
- The *PE Array* is the core computational unit in SparSynergy. Each PE is capable of performing vector-level, value-level, and bit-level sparse operations simultaneously. This capability enables more efficient utilization of sparsity, reducing unnecessary computations and boosting performance.
- The *Main Controller and Data Paths* manage the flow of data and optimize data reuse. By orchestrating the movement of data between the global buffer and PEs, the controller ensures maximum resource utilization, reduces latency, and enhances overall performance.

B. Hierarchical Memory with Adaptive Dataflow

To alleviate bandwidth-intensive data transmission, SparSynergy employs a hierarchical memory system coupled with an adaptive dataflow mechanism. This approach maximizes data reuse and reduces memory access overhead. The on-chip global buffer temporarily stores weights, activations, and partial sums, ensuring that necessary data is readily available to the PEs, thereby minimizing latency in data retrieval. The PEs are organized similarly to Tensor Core's *Octet* architecture, with dual weight registers, partial sum registers, and a shared activation register, as shown in Fig.5(b). This hierarchical memory structure minimizes data movement, reducing bandwidth pressure.

The main controller orchestrates the computation process, using subtensors as the basic unit. Weights are shared horizontally, and activations are shared vertically, creating a systolic-based dataflow in an output-stationary configuration. To further enhance performance, an adaptive dataflow mechanism is introduced to improve the scalability of the accelerator and meet the high throughput demands of DNNs. This adaptability is achieved by leveraging the flexibility of subtensors. When a particular tensor dimension is too small, resulting in under-utilization of the PE array, the main controller dynamically resizes the subtensors, grouping taller subtensors together to better fit the PE array. This resizing allows for better PE utilization by eliminating idle cycles and preventing the waste of computational resources.

C. Processing Element Design

To counteract the reduction in throughput and scalability caused by limited parallelism, we have redesigned the Dot Product (DP) unit based on the bit-column MAC architecture [7], as shown in Fig.5(c) and Fig.5(d).

The bit-serial MAC operates on one bit at a time, whereas the bit-column MAC processes one bit-column, which alleviates some of the parallelism limitations. To further enhance throughput and parallelism, we enable sharing of activations across four bit-column MACs. Each MAC is designed to be pipelined with its corresponding bit-column, ensuring continuous data processing and improving computational efficiency when handling sparse data at the bit level.

VI. Evaluation

A. Experimental Setup

Baselines. We implement SparSynergy alongside 3 SOTA representative architectures in RTL[1].: (1) We utilize dense Tensor Core (TC) [19] to be a baseline for evaluating the performance gains achieved through sparsity exploitation. (2) We also compared SparSynergy with several value-level sparse accelerators, including STC [16] (GPU A100), SIGMA [2], S2TA [3], and HighLight [4]. (3) Additionally, we compared SparSynergy with bit-level sparse accelerators equipped with distinct MAC units, such as Bit-Balance [9], Bitlet [8], and BitWave [7].

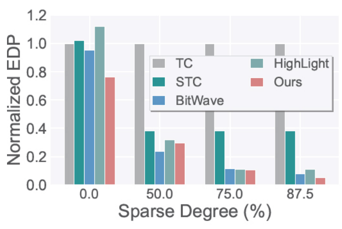
Benchmarks. To evaluate SparSynergy, we employ a comprehensive set of workloads encompassing both synthetic and real-world scenarios. (1) **Synthetic workloads** (§ VI-C) consist of matrix multiplication tasks, specifically designed with operand matrices sized 1024×1024, to simulate varying levels of sparsity at both the value and bit levels. (2) Alongside, we also test SparSynergy using **real-world DNN models** (§ VI-D), notably ResNet-50 and BERT-base. These networks have been carefully pruned and quantized to introduce controlled levels of sparsity.

Evaluation Frameworks. We implement the data path components, including TC, STC, HighLight, BitWave, and SparSynergy, in RTL. Additionally, on-chip SRAMs are generated utilizing an in-house SRAM compiler. These designs are synthesized with a clock frequency of 500MHz, targeting the standard corner of 7nm CMOS technology. To facilitate a fair comparison in performance analysis, we ensure a consistent hardware configuration featuring a 256KB global buffer and 512 MAC units. We use PyTorch and NVIDIA's Apex frameworks for model pruning and quantization to achieve both value-level and bit-level sparsity.

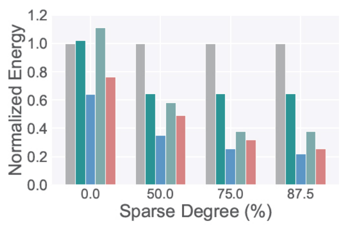
B. Throughput and Efficiency Analysis

We evaluate SparSynergy's throughput metrics, alongside its energy and area efficiency, in comparison to SOTA accelerators, as detailed in Tab. II. For consistency, the area is normalized to a 28nm technology node using the scaling approach proposed in [18]. Value-level sparse accelerators achieve the highest performance through their parallel computing logic, but at the cost of higher energy consumption. Regarding energy efficiency, SparSynergy achieves an exceptional efficiency of 2.44 TOPS/W, which is comparable to that of the A100 GPU and significantly exceeds that of other value-level sparsity accelerators. In contrast, bit-level accelerators such as BitWave and Bitlet, which focus on bit-serial operations, find their throughput gains inherently limited. Specifically, SparSynergy showcases a superior throughput of 862.4 GOPS, significantly outstripping BitWave's 215.6 GOPS and Bitlet's 744.7 GOPS. Furthermore, although Bit-Balance achieves a higher throughput of 2,048 GOPS, this comes at the cost of substantially higher power consumption and a larger chip area.

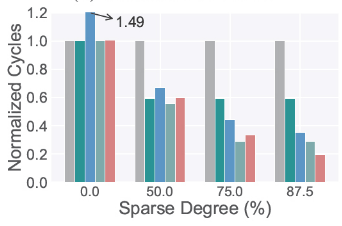
[1]open source at https://github.com/kelvin0207/SparSynergy



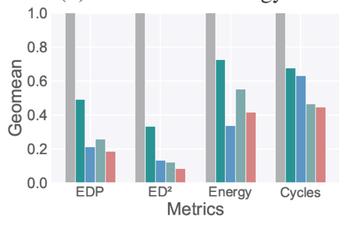
(a) Normalized EDP.



(b) Normalized Energy.



(c) Normalized Cycles.



(d) Geomean.

Fig. 6: Comparative hardware efficiency for sparse matrix multiplication: an evaluation of energy-delay product (EDP), Energy-Delay Squared (ED^2), Energy Consumption, and Processing Speed with varied left matrix sparsity.

SparSynergy uniquely combines the energy efficiency of bit-serial MACs with the high throughput of bit-parallel MACs, effectively addressing trade-offs between throughput, power consumption, and area efficiency.

C. Performance Across Sparsity Levels

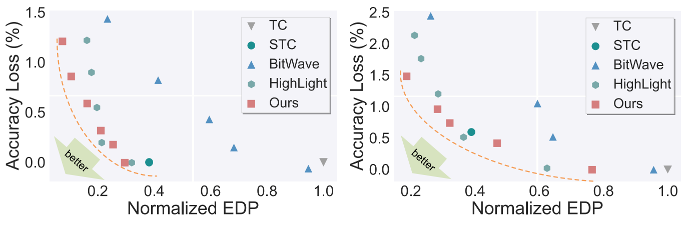
Fig. 6(a) to Fig. 6(c) depict the results of a comparative study on hardware efficiency for sparse matrix multiplication, specifically evaluating various metrics across our approach and four other designs at varying degrees of matrix sparsity, namely at 0%, 50%, 75%, and 87.5%. STC suffers from limited sparsity support, which restricts its ability to efficiently process sparse matrices with sparsity degrees above 50%. BitWave, while achieving excellent energy efficiency across multiple sparsity degrees due to its bit-serial column MACs, faces higher latency owing to inherent design limitations. HighLight achieves lower latency but at the cost of increased energy consumption.

Fig. 6(d) shows the geometric mean of each metric across evaluated workloads. For the EDP, our method shows reductions of 81%, 62%, 12%, and 28%, respectively, when compared to the TC, STC, BitWave, and HighLight accelerators. Regarding ED^2, there's a 92% decrease in performance against TC, 75% against STC, 38% against BitWave, and 31% against HighLight. In terms of energy consumption, there is a 58% reduction compared to TC, 42% compared to STC, an increase of 25% relative to BitWave, and a 24% reduction against HighLight. Finally, for the cycles metric, our method achieves decreases of 55%, 34%, 29%, and 4% respectively, compared to the same accelerators. Compared to existing designs, **SparSynergy achieves the best geomean for EDP, ED^2, and latency**, although it slightly trails BitWave in energy consumption due to BitWave's inherent advantage in power efficiency.

TABLE II: Performance comparison of SparSynergy versus State-of-the-Art accelerators.

	GPU A100 [16]	**SIGMA** [2]	**S2TA** [3]	**Bit-Balance** [9]	**Bitlet** [8]	**BitWave** [7]	**SparSynergy**[*]
Technology	7nm	28nm	16nm	65nm	28nm	16nm	7nm
Frequency (MHz)	1410	500	1000	1000	1000	250	500
Power	400W	22.33W	559.44mW	1130mW	366mW	17.56mW	353.26mW
Area (mm^2)	826	65.10	3.80	7.34	1.54	1.14	0.17
Sparsity support[†]	W. value	W./A. value	W./A. value	W. bit	W. bit	W. bit	W. bit/value
Performance (GOPS)	1248	10800	8000	2048	744.7	215.6	862.4
Energy Efficiency (TOPS/W)	1.5 - 3.1	0.48	0.01	1.81	2.03	12.28	2.44
Normalized Area (mm^2)[‡]	13216	65.10	11.64	1.36	1.54	3.49	2.74
Normalized Area[‡] **Efficiency (GOPS/mm^2)**	31.6 - 65.2	165.90	0.69	1503.64	483.57	61.86	314.29

[*] Evaluate based on 50% sparsity of bit-level and value-level.
[†] W./A. stands for sparsity in weight/activation, and bit/value stands for bit-level/value-level sparsity respectively.
[‡] Normalized to 28 nm technology node based on [18].



(a) ResNet-50. (b) BERT-base.

Fig. 7: EDP-Accuracy loss trade-off on various accelerators and the Pareto Frontier of the optimal design (represented by the orange dashed line) for ResNet-50 and BERT-base.

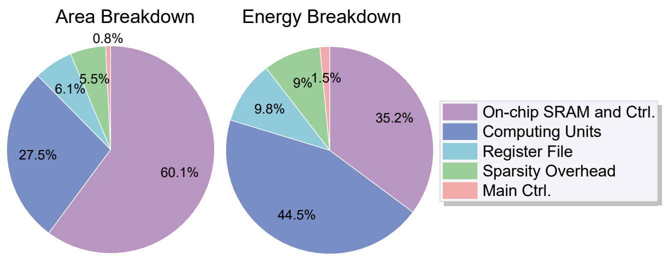


Fig. 8: Area and Energy breakdown of SparSynergy.

D. Accuracy vs. Efficiency Trade-off Analysis

We conduct an in-depth analysis of the trade-off between EDP and accuracy loss across different designs in comparison with SparSynergy. Fig. 7 illustrates the trade-off in representative DNNs at various sparsity levels. Specifically, we compare SparSynergy with several popular designs: (1) dense accelerators (e.g., represented by TC data points), (2) value-level sparse accelerators (e.g., STC and HighLight data points), and (3) bit-level sparse accelerators (e.g., BitWave data points). The optimal design should balance different workloads, ideally positioning itself on the Pareto frontier of the EDP-accuracy loss curve for both ResNet-50 and BERT-base. Moreover, SparSynergy consistently resides on the Pareto frontier, making it an ideal candidate for supporting diverse DNNs with high hardware efficiency while maintaining accuracy.

As shown in Fig. 7, STC demonstrates good performance at only a specific sparsity degree. BitWave struggles to achieve optimal EDP, and HighLight exhibits a slightly higher accuracy loss. In contrast, SparSynergy offering high hardware efficiency while maintaining acceptable accuracy loss across a range of DNNs. **This positions SparSynergy as an excellent accelerator candidate, offering high hardware efficiency while maintaining acceptable accuracy loss across a range of DNNs.**

E. Energy & Area Breakdown

Fig.8 presents the area and energy consumption breakdown for SparSynergy while processing of a workload with 75% sparsity in the left input matrix during matrix multiplication. The on-chip SRAM (Global Buffer) occupies the largest fraction of the total area at 60.1%. Conversely, the computing units is the most power-intensive component and consumes the largest fraction of the total power consumption at 44.5%, while taking up 27.5% of the total area. Notably, the cost associated with sparsity amounts to just 5.5% of the design's area and and 9.0% of the total energy consumption, underscoring **SparSynergy's efficiency as a sparsity-accommodating accelerator with minimal area and energy penalties for sparsity management.**

VII. Conclusion

This work presents SparSynergy, an innovative hardware accelerator that leverages both value-level and bit-level sparsity to improve computational efficiency and energy savings. By employing a unified sparse format and novel architectural optimizations, SparSynergy achieves a superior energy-delay product (EDP) while maintaining accuracy across varying sparsity levels. Our evaluations show that SparSynergy outperforms state-of-the-art (SOTA) accelerators, providing an effective solution for diverse deep learning workloads and addressing the increasing demands of modern AI applications.

Acknowledgment

We would like to express our gratitude to the anonymous reviewers for their constructive feedback. We also thank Man Shi for the helpful discussion about BitWave.

This work was supported by National Natural Science Foundation of China (Grant No.U23A20301 and 62102433), Natural Science Foundation of Hunan Province, China (No.2024JJ6470 and 2023JJ40679), and NUDT Foundation (No.ZK2023-16 and 2022-KJWPDL-08).

References

[1] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, " AI and Memory Wall ," *IEEE Micro*, vol. 44, pp. 33–39, May 2024.

[2] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for dnn training," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 58–70, IEEE, 2020.

[3] Z.-G. Liu, P. N. Whatmough, Y. Zhu, and M. Mattina, "S2ta: Exploiting structured sparsity for energy-efficient mobile cnn acceleration," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 573–586, IEEE, 2022.

[4] Y. N. Wu, P.-A. Tsai, S. Muralidharan, A. Parashar, V. Sze, and J. Emer, "Highlight: Efficient and flexible dnn acceleration with hierarchical structured sparsity," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 1106–1120, 2023.

[5] Y. Wang, C. Zhang, Z. Xie, C. Guo, Y. Liu, and J. Leng, "Dual-side sparse tensor core," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1083–1095, IEEE, 2021.

[6] Y. Umuroglu, D. Conficconi, L. Rasnayake, T. B. Preusser, and M. Själander, "Optimizing bit-serial matrix multiplication for reconfigurable computing," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, Aug. 2019.

[7] M. Shi, V. Jain, A. Joseph, M. Meijer, and M. Verhelst, "Bitwave: Exploiting column-based bit-level sparsity for deep learning acceleration," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 732–746, IEEE, 2024.

[8] H. Lu, L. Chang, C. Li, Z. Zhu, S. Lu, Y. Liu, and M. Zhang, "Distilling bit-level sparsity parallelism for general purpose deep learning acceleration," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 963–976, 2021.

[9] W. Sun, Z. Zou, D. Liu, W. Sun, S. Chen, and Y. Kang, "Bit-balance: Model-hardware codesign for accelerating nns by exploiting bit-level sparsity," *IEEE Transactions on Computers*, vol. 73, no. 1, pp. 152–163, 2024.

[10] Y. Chen, J. Meng, J. sun Seo, and M. S. Abdelfattah, "Bbs: Bi-directional bit-level sparsity for deep learning acceleration," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2024.

[11] V. Joseph, G. L. Gopalakrishnan, S. Muralidharan, M. Garland, and A. Garg, "A programmable approach to neural network compression," *IEEE Micro*, vol. 40, no. 5, pp. 17–25, 2020.

[12] C. Tang, K. Ouyang, Z. Wang, Y. Zhu, W. Ji, Y. Wang, and W. Zhu, "Mixed-precision neural network quantization via learned layer-wise importance," in *European Conference on Computer Vision*, pp. 259–275, Springer, 2022.

[13] K. Zhong, Z. Zhu, G. Dai, H. Wang, X. Yang, H. Zhang, J. Si, Q. Mao, S. Zeng, K. Hong, G. Zhang, H. Yang, and Y. Wang, "Feasta: A flexible and efficient accelerator for sparse tensor algebra in machine learning," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, (New York, NY, USA), p. 349–366, Association for Computing Machinery, 2024.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, IEEE, 2016.

[15] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of naacL-HLT*, vol. 1, p. 2, Minneapolis, Minnesota, 2019.

[16] 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.

[17] F. Liu, N. Yang, H. Li, Z. Wang, Z. Song, S. Pei, and L. Jiang, "Spark: Scalable and precision-aware acceleration of neural networks via efficient encoding," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 1029–1042, IEEE, 2024.

[18] Y. Cao, M. Wen, J. Shen, and Z. Li, "Bitshare: An efficient precision-scalable accelerator with combining-like-terms gemm," in *2024 IEEE 35th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 36–44, 2024.

[19] NVIDIA, "Nvidia tesla v100 gpu architecture," tech. rep., 2017.