2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 978-1-6654-7652-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/HPCA56546.2023.10071031

Sibia: Signed Bit-slice Architecture for Dense DNN Acceleration with Slice-level Sparsity Exploitation

Dongseok Im, Gwangtae Park, Zhiyong Li, Junha Ryu, and Hoi-Jun Yoo
School of Electrical Engineering
Korea Advanced Institute of Science and Technology (KAIST)
{*dsim, gwangtaepark, zhiyong_li, junha.ryu, hjyoo*}*@kaist.ac.kr*

***Abstract*—Deep neural networks (DNNs) have achieved high performance in many AI fields such as 1-D language, 2-D image, and 3-D point cloud processing applications. Since recent DNN tasks require dense matrix operations with various bit-precision and non-ReLU activation functions, mobile neural processing units (NPUs) suffer from the acceleration of diverse DNN tasks within their limited hardware resources and power budget. Although bit-slice architectures benefit from slice-level computation and slice-level sparsity exploitation, the conventional bit-slice representation is inefficient in bit-slice architectures resulting in poor dense DNN execution. This paper proposes the efficient signed bit-slice architecture, Sibia, with the signed bit-slice representation (SBR) for efficient dense DNN acceleration. The SBR adds a sign bit to each bit-slice and changes signed 1111_2 bit-slice to 0000_2 by borrowing a value of 1 from its lower order of the bit-slice. This scheme generates large numbers of zero bit-slices in dense DNNs even not relying on accuracy-sensitive pruning methods or retraining processes. Moreover, the SBR balances positive and negative values of 2's complement data, allowing accurate bit-slice-based output speculation that pre-computes high orders of bit-slices. Sibia integrates the signed multiplier-and-accumulate (MAC) units for efficient signed bit-slice computations, and the flexible zero skipping processing element (PE) supports the zero input bit-slice skipping and output skipping for high throughput and energy-efficiency. Additionally, the dynamic sparsity monitoring unit monitors sparsity ratio between input and weight data and determines the more sparse one for zero bit-slice skipping. The heterogeneous network-on-chip (NoC) benefits from data reusability during bit-slice computation, reducing transmission bandwidth. Finally, Sibia outperforms the previous bit-slice architecture, Bit-fusion, over $3.65\times$ higher area-efficiency, $3.88\times$ higher energy-efficiency, and $5.35\times$ higher throughput.**

***Keywords*—Hardware accelerator, deep neural network, binary representation, bit-slice, sparsity, output speculation, non-ReLU**

I. Introduction

Recently, many mobile devices have implemented diverse deep neural network (DNN) tasks for 1-D to 3-D applications such as natural language processing (NLP) [4], [19], photography improvement [15], visual question and answering [40], and AR/VR [43]. Although a DNN model requires a lot of multiplier-and-accumulate (MAC) operations, lowering DNN bit-precision improves DNN execution performance by lessening the memory bandwidth and on-chip memory footprint with minimal accuracy degradation. Therefore, bit-slice architectures [9], [17], [22], [32], [34] have been designed for bit-slice-level computation to support various reduced bit-precision of DNNs. They integrate a massive number of low-bit MAC units and dynamically match DNN bit-precision in a spatial- and time-multiplexing method. Further performance improvements in the bit-slice architecture are possible by exploiting input and output sparsity. The zero bit-slice skipping architecture [9] increases the hardware performance by skipping zero input bit-slice computations caused by not only the ReLU activation function but also positive near-zero values whose high orders of bit-slices are zeros. On the other hand, the bit-slice-based output skipping architectures [35], [38] exploit output sparsity presented by a softmax operation or a max pooling operation. Since most outputs after a softmax layer or max pooling layer are very small or zero values, the output skipping architectures speculate those insensitive output computations by pre-computing high orders of bit-slice and skips their remaining computations to achieve high throughput and energy-efficiency.

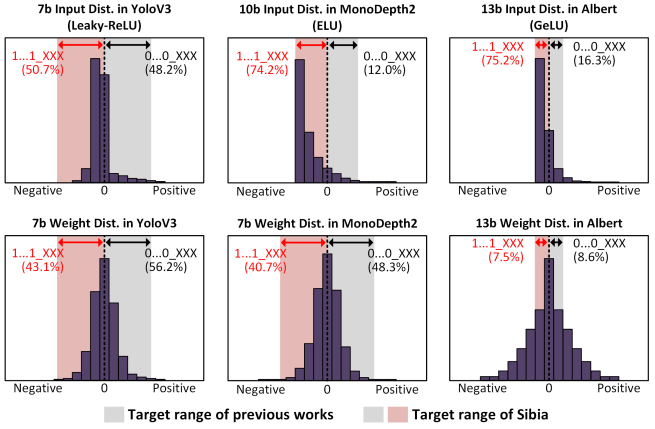

Fig. 1. Input and weight distribution in dense DNNs with a target range of the previous zero bit-slice skipping architecture and the proposed Sibia.

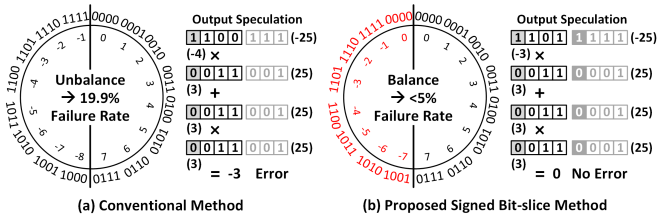

Fig. 2. The bit-slice-based output speculation with (a) unbalanced bit-slices and (b) the proposed balanced signed bit-slices.

978-1-6654-7652-2/23/$31.00 ©2023 IEEE

However, the conventional bit-slice representation which decomposes 2's complement fixed-point data into bit-slices poses problems for bit-slice architectures. First, a generation of zero bit-slices is limited to zero data and positive near-zero data. It is a critical issue when bit-slice architectures execute dense DNNs. For example, recent state-of-the-art DNNs have used non-ReLU activation functions such as GeLU [14], Leaky ReLU, and ELU [2], which saturate negative data with smaller values. Moreover, weight values are also dense data because they follow the Gaussian distribution during DNN training [6], [12], [20]. Therefore, the previous bit-slice generation misses a significant amount of slice-level sparsity at negative near-zero data in both input and weight as shown in Fig. 1. Specifically, bit-slice architectures only exploit 12.0% of zero bit-slice even though 1111_2 data accounts for 74.2% after the ELU activation function. As a result, zero bit-slice skipping architectures cannot exploit a lot of zero data, and their performances degrade as the number of zero bit-slices is insufficient in dense DNNs.

Second, the output speculation, which pre-computes high orders of bit-slices and then avoids remaining computations of insensitive outputs, easily fails because of an unbalance of 2's complement number between positive and negative as shown in Fig. 2 (a). For example, high order of bit-slice of 1100111_2 (-25) and 0011001_2 (25) is 1100_2 (-4) and 0011_2 (3), respectively, by using the conventional bit-slice decomposition. Then, the speculation output of $(-25) \times (25)$ is (-12), but the speculation output of $(25) \times (25)$ is 9. Even though their full bit-width MAC result is 0, their speculation result is (-3). Therefore, this unbalanced bit-slice causes output speculation error. Specifically, the 32-to-1 max pooling output speculation using a pre-computation of both 4-bit high order of input and weight bit-slice causes 19.9% of wrong speculation results in VoteNet [28] while a pre-computation of full bit-width (8-bit) input and 4-bit high order of weight bit-slice reduces the speculation error to $< 5\%$. Therefore, output skipping architectures with the conventional bit-slice representation have to exploit a high bit-width of bit-slices for the output speculation, limiting the performance enhancement.

Third, a slice-level sparsity exploitation architecture has larger hardware overheads than a full bit-width sparsity exploitation architecture. Since a 4-bit bit-slice architecture adopts $4\times$ the numbers of MAC units compared to 8-bit fixed bit-width MAC units, it requires $4\times$ the numbers of zero skipping units as shown in Fig 3 (a). Moreover, since bit-slices are either signed or unsigned data after the decomposition, a sign bit extension is required to multiply each other. Therefore, the width of a multiplier and an accumulation register are extended up to the most significant bit (MSB) of bit-slice data, resulting in area overheads in MAC units. As a result, the bit-slice architecture requires a $2.07\times$ larger logic area than the full bit-width architecture to achieve the same throughput in a 28 nm technology node. Furthermore, after sparse data compression, such as run-length encoding (RLE), bit-width of a non-zero index becomes large relative to non-zero data after the bit-slice decomposition which halves the bit-width

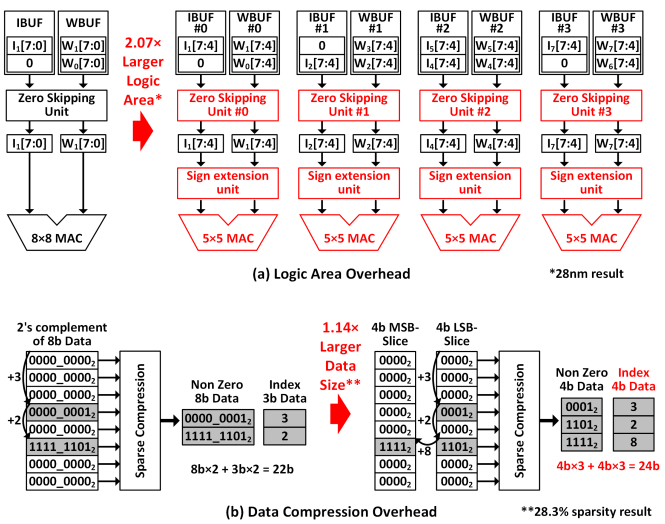

Fig. 3. Hardware challenges in a zero bit-slice skipping architecture: (a) logic area overhead and (b) data compression overhead.

and doubles the numbers as shown in Fig 3 (b). Therefore, 4-bit data compression shows $1.14\times$ larger data size than 8-bit data compression in 28.3% sparsity data. Consequently, an efficient zero bit-slice skipping architecture is necessary to achieve both high throughput and high energy-efficiency.

To solve the challenges of a bit-slice architecture, the paper presents a new method of bit-slice decomposition and an energy-efficient signed bit-slice architecture, Sibia. The summary of Sibia is below:

- The novel signed bit-slice representation (SBR) is proposed to increase bit-slice sparsity at both positive and negative near-zero data. The SBR adds a sign bit to each bit-slice and borrows a value of 1 from their lower order of bit-slice. It allows great performance enhancement and high data compression ratio of a bit-slice architecture and also solves the unbalanced problem of 2's complement data, which enables the accurate low-bit output speculation.
- The signed MAC unit is designed for high MAC efficiency of signed bit-slice computations. It does not need a sign extension unit, a sign-extended multiplier, and an accumulation register which a conventional bit-slice architecture requires, improving MAC efficiency compared to a bit-slice architecture.
- The flexible zero skipping processing element (PE) is designed to exploit signed bit-slice sparsity with the output speculation. It efficiently processes sparse signed bit-slices with minimum overheads of zero slice skipping unit, improving the hardware performance.
- Sibia supports hybrid zero skipping and hybrid zero compression depending on a sparsity ratio. the dynamic sparsity monitoring (DSM) unit monitors a sparsity ratio of input and weight bit-slices during data fetching and exploits more sparse data for zero skipping. Moreover, it determines a zero compression operation based on a

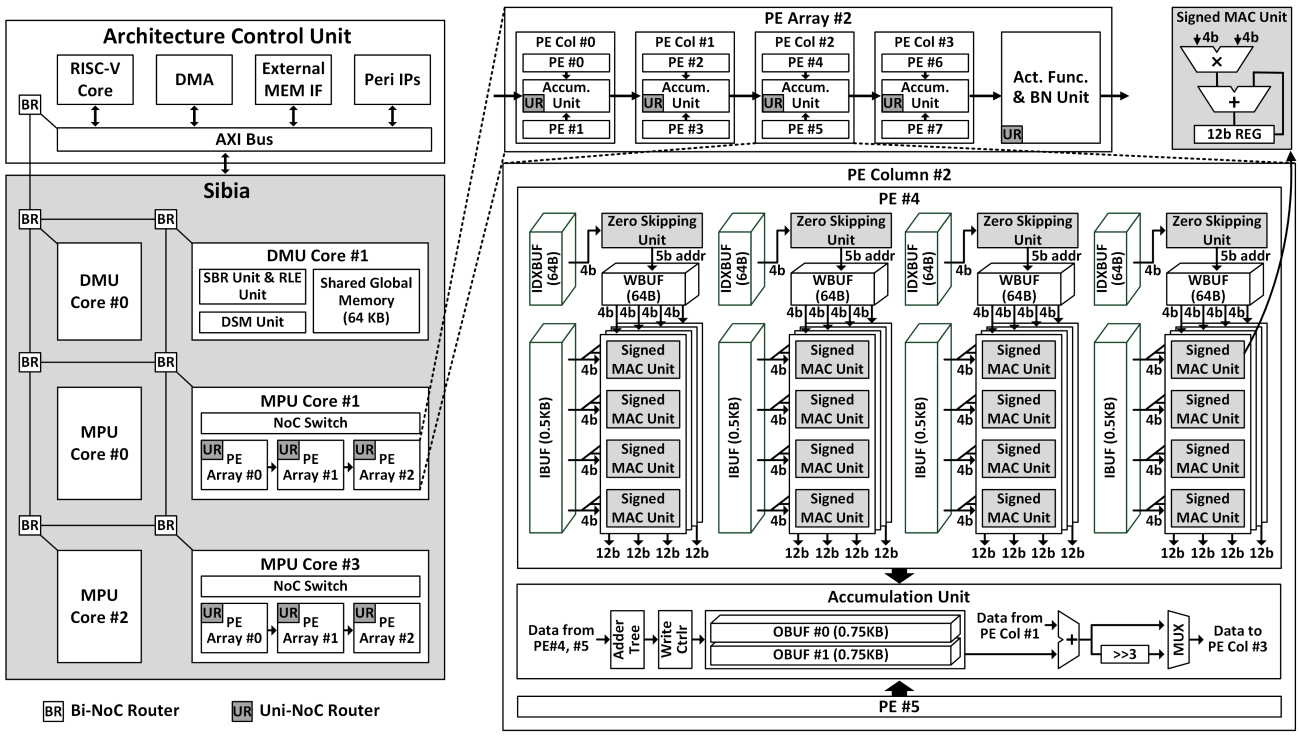

Fig. 4. Overall architecture of Sibia and the architecture control unit.

sparsity ratio to increase the overall compression ratio.

- The heterogeneous NoC is designed for flexible data transactions. A bi-directional 2D mesh-based NoC (Bi-NoC) supports versatile workload allocations by transferring inputs, weights, and convolution outputs among the PEs considering data reusability while a uni-directional NoC (Uni-NoC) accumulates output partial sums across accumulation units with optimized bandwidth.

II. Sibia Architecture

A. Overall Architecture

Fig. 4 illustrates an architecture of Sibia. Sibia consists of a quad-core matrix processing unit (MPU) and a dual-core data management unit (DMU). Each MPU core has three PEs, one of which has four PE columns (PE Cols) along with an activation function unit and a batch normalization unit. Each PE column is comprised of two PEs and one accumulation unit. A total of 64 4b×4b signed MAC units are integrated into the PE. The accumulation unit adds up output partial sums of the PEs and stores the results in the output buffers (OBUFs). It also accumulates output partial sums across the PE Cols by passing them to the subsequent accumulation unit. Each DMU core integrates 64 KB global memory shared across all units along with memory-centric units, such as an SBR unit, an RLE unit, and a DSM unit, which can access the global memory without occupying the on-chip network. The on-chip network is built hierarchically with the custom interconnection network connecting different units, and transfers input data, weight data, and output partial sums.

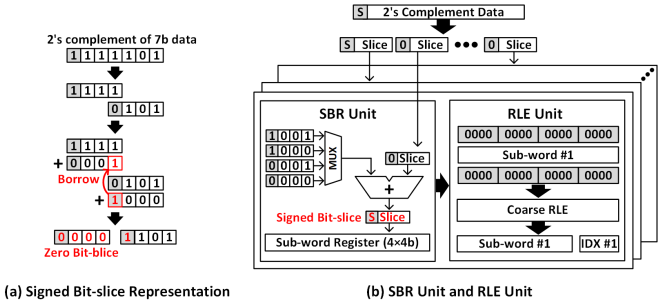

Fig. 5. Concept of the signed bit-slice: (a) the signed bit-slice representation (SBR) and (b) the SBR unit and the run-length encoding (RLE) unit.

B. Signed Bit-slice Representation and Its Encoding Unit

Conventional bit-slice representation [9], [34] decomposes 2's complement fixed-point data into an MSB bit-slice, which is a signed bit-slice, and lower unsigned bit-slices. In this work, the SBR adds a sign bit to each unsigned bit-slice to produce signed bit-slices, and it adds 1 by borrowing from its lower bit-slice if data is a negative value. An example of 7-bit fixed-point data is shown in Fig. 5(a). A 2's complement data 1111101_2 is decomposed into 1111_2 of the MSB bit-slice and 101_2 of the lower unsigned bit-slice. The SBR adds a sign bit to 101_2 to produce 0101_2 of the signed bit-slice and then adds 1 to 1111_2 of the MSB bit-slice by borrowing from 0101_2. Finally, they become the signed bit-slices, 0000_2 and 1101_2, and the SBR produces the zero signed bit-slice in a negative

value.

The SBR can apply any 2's complement data. The N-bit $(N = 4, 7, 10, 13, ...)$ integer of a 2's complement number A is given by

$$A = -a_{N-1}2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i \quad (1)$$

where $a_i \in \{0, 1\}$ for $i = N-1, N-2, ..., 0$, and the MSB a_{N-1} indicates the sign bit. To produce bit-slices, the three consecutive bits are grouped except for the sign bit,

$$\begin{aligned} A = & -a_{N-1}2^{N-1} \\ & +(a_{N-2}2^{N-2} + a_{N-3}2^{N-3} + a_{N-4}2^{N-4}) \\ & +(a_{N-5}2^{N-5} + a_{N-6}2^{N-6} + a_{N-7}2^{N-7}) \\ & +\cdots \\ & +(a_2 2^2 + a_1 2^1 + a_0 2^0) \end{aligned} \quad (2)$$

Equation (2) is extended by adding and subtracting $a_{N-1}2^i$ for $i = N-1, N-4, N-7..., 3$,

$$\begin{aligned} A = & -a_{N-1}2^{N-1} + a_{N-1}2^{N-1} \\ & +(-a_{N-1}2^{N-1} + a_{N-2}2^{N-2} + \cdots + a_{N-4}2^{N-4} + a_{N-1}2^{N-4}) \\ & +(-a_{N-1}2^{N-4} + a_{N-5}2^{N-5} + \cdots + a_{N-7}2^{N-7} + a_{N-1}2^{N-7}) \\ & +\cdots \\ & +(-a_{N-1}2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0) \end{aligned} \quad (3)$$

Now, each group has a sign bit and becomes a 4-bit signed bit-slice as a 2's complement number A',

$$A = (A'_{N-4} \times 2^{N-4}) + (A'_{N-7} \times 2^{N-7}) + \cdots + A'_0 \quad (4)$$

Therefore, the SBR decomposes a 2's complement number into the multiple signed bit-slices by adding and subtracting the MSB a_{N-1} to all bit-slices.

Fig. 5(b) illustrates the SBR unit and the run-length encoding (RLE) unit. The SBR unit applies the SBR to data before the data is processed by the matrix processing unit (MPU) in Sibia. The SBR unit has four registers that indicate the borrowing and lending bit. Then, full bit-width data is decomposed into multiple bit-slices, and the SBR unit adds each bit-slice to one of the four registers based on the order of the bit-slice and the sign bit. For example, the middle-order of bit-slice can borrow 1_2 from the lower order of bit-slice and lend 1000_2 to the higher order of bit-slice. On the other hand, the MSB bit-slice only borrows 1_2, and the least significant bit (LSB) bit-slice only lends 1000_2. After producing signed bit-slices, four 4-bit signed bit-slices are stored in the sub-word (16b) register and sent to the RLE unit for zero compression if they are all zeros. Only non-zero sub-word data and its index are transferred to the MPU. Even though a sign bit is added to each bit-slice during the SBR, the total data size is significantly reduced by the zero compression.

Fig. 6 describes the sparsity enhancement by the SBR in dense DNNs. By using the SBR, high orders of signed bit-slice becomes extremely sparse data (80∼99%). Then, the SBR increases the total signed bit-slice sparsity of input data

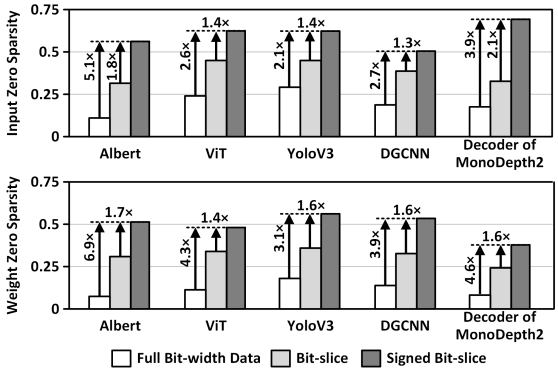

Fig. 6. Sparsity ratio of full bit-width data, bit-slice, and signed bit-slice in dense DNNs.

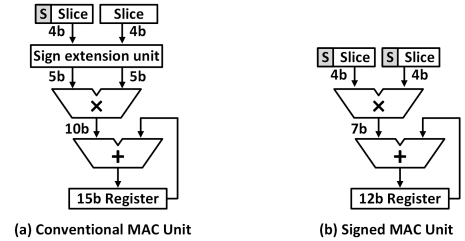

Fig. 7. MAC unit architecture: (a) the conventional MAC unit and (b) the signed MAC unit.

in Albert [19] by $5.1\times$ and $1.8\times$ higher than full bit-width data and bit-slice data produced by the conventional bit-slice representation, respectively. Similarly, sparsity of a weight signed bit-slice in Albert increases by $6.9\times$ and $1.7\times$ higher. ViT [5] has $2.6\times$ and $1.4\times$ higher sparsity in input data and $4.3\times$ and $1.4\times$ higher sparsity in weight data. Sparsity of YoloV3 [30] shows $2.1\times$ and $1.4\times$ higher in input data and $3.1\times$ and $1.6\times$ higher in weight data. DGCNN [39] has $2.7\times$ and $1.3\times$ higher input sparsity and $3.9\times$ and $1.6\times$ higher weight sparsity. In the case of a dense decoder network of MonoDepth2 [7], it has $3.9\times$ and $2.1\times$ higher input sparsity and $4.6\times$ and $1.6\times$ higher weight sparsity. As a result, the SBR makes the majority 1111_2 bit-slices at the small negative values to 0000_2 bit-slices, and sparsity of signed bit-slice significantly increases that a bit-slice architecture has a greater chance of increasing throughput and energy-efficiency in dense DNN executions.

The SBR enables the accurate bit-slice-based output speculation by balancing the positive and negative values of the 2's complement data as shown in Fig. 2 (b). Since the SBR makes both high order of bit-slice of 1100111_2 (-25) and 0011001_2 (25) to 1101_2 (-3) and 0011_2 (3), respectively, the bit-slice-based speculation error becomes smaller than the conventional bit-slice decomposition. For example, the success rate of the output speculation with both 4-bit input and weight signed bit-slices shows roughly 95% in VoteNet [28], but the previous approach requires 4-bit input bit-slices and 8-bit weight bit-slices to obtain a similar success rate. As a

result, the SBR takes advantage of accurate speculation with low bit-width of bit-slices and achieves the high throughput and energy-efficiency enhancement. This work conducts the linear symmetric quantization for the accurate bit-slice-based output speculation.

C. Signed MAC Unit

The conventional bit-slice architecture integrates inefficient MAC units to compute both signed and unsigned bit-slices. It requires a sign extension unit, an enlarged bit-width of a multiplier, and an enlarged bit-width of an output accumulation register in a MAC unit as shown in Fig. 7(a). On the other hand, since the SBR produces only signed bit-slices, Sibia directly computes the signed bit-slices using the signed MAC units without sign extension as illustrated in Fig. 7(b). Furthermore, the output accumulation register has a low bit-width because the SBR eliminates the 1000_2 representation in a 4-bit signed bit-slice as shown in Fig. 2(b), resulting in a 7-bit output after the multiplication of two 4-bit signed bit-slices. As a result, the signed MAC unit can achieve high efficiency by reducing the bit-width of a multiplier and output accumulation register compared to the conventional MAC unit in the bit-slice architecture. For example, the previous work [9] uses a 5b×5b MAC unit with sign extension to compute 4-bit, 8-bit, 12-bit, and 16-bit precision data with its best MAC efficiency. On the other hand, the 5b×5b signed MAC unit can support 5-bit, 9-bit, 13-bit, and 17-bit precision data which shows a higher bit capability than the previous work. The other work [34] uses a 3b×3b MAC unit for 2-bit, 4-bit, 6-bit, and 8-bit precision data while the 3b×3b signed MAC unit supports 3-bit, 5-bit, 7-bit, and 9-bit precision. In this paper, Sibia integrates 4b×4b signed MAC units to support 4-bit, 7-bit, 10-bit, and 13-bit precision data that the previous bit-slice architecture has to adopt the 5b×5b MAC units to support those precision. As a result, Sibia saves 21.9% of energy consumption of the MAC unit at 7-bit precision of DNNs compared to the conventional bit-slice architecture.

D. Flexible Zero Skipping Processing Element for Input and Output Skipping

Fig. 8 describes the architecture and data path of the flexible zero skipping PE unit. The SBR generates a large number of 4-bit zero bit-slices even in a non-ReLU activation function, and a large number of computations can be removed using zero skipping. To minimize the overheads of a fine-grained zero bit-slice skipping unit, Sibia processes the four spatially adjacent 4-bit input bit-slices as sub-word data and skips the zero sub-word data. The PE fetches the non-zero sub-word data and its RLE index from the input buffer (IBUF) and the index buffer (IDXBUF), respectively. The sub-word data is split into four 4-bit bit-slices, and each bit-slice is allocated to each row of the signed MAC arrays and shared with the four signed MAC units, generating four spatially adjacent output partial sums. At the same time, the corresponding weight bit-slice is loaded by calculating the next address of the weight buffer (WBUF) using the RLE index at the zero skipping unit. The weight bit-slice is shared by the four rows of the signed MAC units, generating four output channels of the output partial sums. Therefore, the PE can skip the 16 MAC operations with zero input sub-word data. In the next cycle, the PE computes the next input channels of the data and accumulates those output partial sums to a 12-bit accumulation register in the signed MAC unit.

By exploiting the output sparsity in a softmax and large-scale max-pooling layer, a large number of redundant MAC operations can be removed. To find the insensitive outputs generated by the redundant MAC operations, Sibia executes MAC operations of high order of input bit-slice (I_H) and weight bit-slice (W_H) in advance and finds the candidates of the maximal values among their outputs. For the output speculation of a max pooling layer, non-maximal outputs are speculated as the insensitive outputs, and the remaining low order of bit-slice computations, $I_H \times W_L$, $I_L \times W_H$, and $I_L \times W_L$, are skipped while the maximal candidates are completed. In an Albert [19] task, on the other hand, maximal outputs are insensitive if they are smaller than a pre-defined threshold [38], and only non-maximal candidates are completed. Then, the flexible zero skipping PE skips the insensitive outputs by setting the corresponding input bit-slices to zeros. The speculated insensitive outputs are masked with a binary map, and the RLE unit loads the binary map and regenerates the non-zero sub-word data. Finally, the non-zero sub-word data and its index are transferred to the PEs, where zero skipping operation is performed as shown in Fig. 8. To minimize the complexity of the zero skipping unit, successive four output channels of insensitive outputs are skipped. As a result, the PE supports the output skipping with the data path of input skipping.

As shown in Fig. 4, four columns of the signed MAC arrays in the PE calculate the different input channels, and their outputs are added up at the accumulation unit when each column's channel accumulation is completed. However, since the finish times of the four columns are different during zero skipping, the early finished columns become idle until all of them have completed the computation of allocated input channels. For the high utilization of the PE, the accumulation unit temporally latches the column's outputs to registers. Therefore, the finished column directly passes its outputs to the accumulation unit and proceeds with the convolution of the next spatial input data. After finishing all of the columns, the accumulation unit adds up the column's outputs and stores the results in the OBUF. As a result, the PE maintains high PE utilization.

E. Hybrid Zero Skipping and Compression with Dynamic Sparsity Monitoring

Although the SBR increases sparsity of the high orders of bit-slices, sparsity of the low order is typically low, especially in a non-ReLU activation function. Among the convolution of bit-slices, $I_H \times W_H$ and $I_H \times W_L$ show a high performance due to high sparsity of I_H. However, dense I_L degrades the performance of $I_L \times W_H$ and $I_L \times W_L$, lowering the overall

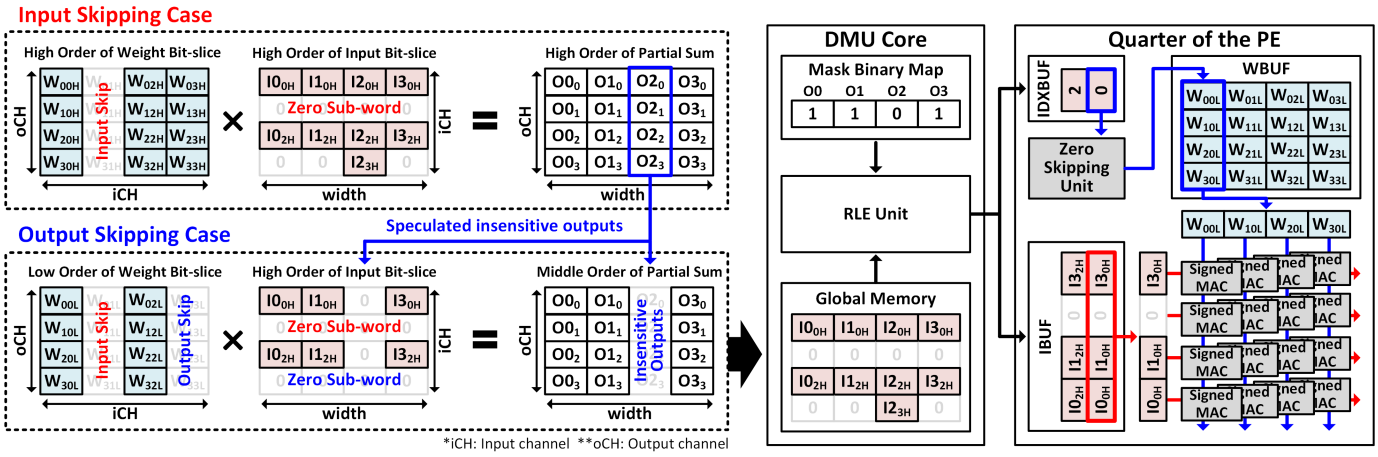

Fig. 8. Computation methods of input and output skipping and a datapath of the flexible zero skipping PE.

performance. Therefore, weight bit-slices are exploited for skipping instead of input bit-slices if weight sparsity is higher than input sparsity. Moreover, to increase the efficiency of $I_L \times W_L$ when both I_L and W_L are dense, Sibia disables the zero skipping units and IDXBUFs during computation of dense bit-slices to reduce the dynamic power. Similarly, the compression of the dense bit-slices deteriorates the compression ratio due to a relatively high bit-width of the non-zero index. Therefore, the compression is not applied to them but still maintains a high compression ratio by the high orders of sparse signed bit-slices.

The flexible zero skipping PE fully supports the weight skipping. Four adjacent output channels of weight bit-slices become the sub-word data, and the PE skips them if they are all zero. The IBUF is then assigned this sub-word data, and the WBUF fetches four adjacent spatial input bit-slices. Finally, the accumulation unit rearranges the computation results before storing them in the OBUF to match the same pattern of the input sub-word. Similar to the input and ouput skipping, the four adjacent output channels of weights corresponding to insensitive outputs are encoded to zeros for the weight and output skipping. Consequently, Sibia supports input, weight, and output skipping without changing the data path.

The DSM unit monitors data sparsity on run-time and decides the types of sparsity exploitation. It compares sparsity of input and weight bit-slices and exploits the more sparse one. The DSM unit counts the number of zero input and weight bit-slices while loading the data from the external memory and storing the data in the global memory. It estimates the total sparsity of a layer by measuring sparsity of a tiled layer that is first fetched from the external memory. It can reduce the overheads of the run-time decision while improving performance significantly. After obtaining sparsity of input and weight bit-slices, the DSM unit writes sparsity information to the registers and interrupts the RISC-V core to exploit higher sparsity. It also disables the sparsity exploitation if both of them are lower than the threshold. Finally, it shows a high performance of zero skipping by selecting the more sparse data and a high compression ratio by deciding the compression operation only on more sparse data.

F. Heterogeneous Network-on-chip

A heterogeneous NoC is adopted for flexible and efficient data transaction among different top-level units as shown in Fig. 4. For the input, weight, and output data transmission, the bi-directional Bi-NoC is used. The data management unit (DMU) provides the input and weight data to the matrix processing unit (MPU), and the MPU transfers the convolution outputs to the DMU through the Bi-NoC. The Bi-NoC also flexibly transfers weight data to IBUF and input data to WBUF if weight sparsity is higher than input sparsity for hybrid skipping. After receiving the data through the router, the NoC switch unicasts, multicasts, and broadcasts the data to the PE arrays with the data reusability. For example, since input bit-slice can be multiplied with both different bit-orders, different output channels, and different spatial weight bit-slice, input data is broadcasted to four PE Cols or three PE arrays Cols while different bit-orders, different output channels, or different spatial weight data is unicasted to the PEs. As a result, the various combinations of workload allocation are exploited in Sibia for data reusability through the Bi-NoC.

Output partial sum requires a high bit-width compared to the input and weight data, and its bit-width is also enlarged after applying an arithmetic shift operation to accumulate the other order of partial sums. To minimize the transmission bandwidth of the partial sums, each accumulation unit applies the right arithmetic shift by 3 to the partial sums before passing them to the subsequent accumulation unit. Then, the Uni-NoC transfers the reduced bit-width of partial sums, and the subsequent accumulation unit accumulates them to obtain the final convolution outputs. As a result, the right arithmetic shift unit reduces the bandwidth of the Uni-NoC by 40% compared to the previous bit-slice architecture [9]. The Uni-NoC connects two adjacent accumulation units and routes the data from right to left. Then, Sibia assigns the low order of bit-

slices to the left side of the PE and the high order of bit-slices to the right side of the PE to accumulate partial sums across the PEs using the Uni-NoC. If the outputs of two adjacent accumulation units are the same bit order, they are passed to the next accumulation unit without a bit-shift operation. As a result, the Uni-NoC transfers the partial sums with minimum bandwidth.

III. Evaluation

A. Benchmarks

Sibia evaluates the various dense 1-D, 2-D, and 3-D DNNs: Albert [19], Vision Transformer (ViT) [5], YoloV3 [30], MonoDepth2 [7], and DGCNN [39]. Albert is a transformer-based DNN and a light-weight Bert [4] variant with shared weights across layers and shows a competitive accuracy on the 1D General Language Understanding Evaluation (GLUE) tasks [37] (MNLI, QQP, SST-2). A base model shows an average 11.9% of data sparsity and requires 7-bit and 10-bit precision in the attention modules with softmax layers and 10-bit and 13-bit precision of input and weight data in the linear layers including feed-forward networks with the GeLU activation function. ViT is a transformer-based 2-D image classification DNN. A base model requires 7-bit and 10-bit precision in both the attention modules and the linear layers. It shows a 24.0% of data sparsity for 384×384 sized ImageNet dataset [31]. YoloV3 is a popular 2-D object detection application that is a dense DNN with the leaky-ReLU activation function. It shows a 29.2% of input sparsity in 7-bit precision for COCO dataset [23]. MonoDepth2 is a monocular depth estimation application that consists of an encoder network and a decoder network. Although an encoder network requires a low bit-precision (7-bit) with high input sparsity (57.3%) due to the ReLU activation function, a decoder network requires 10-bit precision of inputs and 7-bit precision of weights with low input sparsity (17.5%) for NYU-Depth v2 dataset [26] because of the ELU activation function. DGCNN is a graph neural network for 3-D vision tasks and is composed of the leaky-ReLU activation functions and four 40-to-1 max pooling layers. It shows a 17.3% of input sparsity with 7-bit precision for ModelNet40 [41].

Sibia also evaluates sparse DNNs: MobileNetV2 [33], ResNet-18 [13], and VoteNet [28]. MobileNetV2 and ResNet-18 are well-known 2D image classification DNNs, and they require 10-bit and 7-bit precision with a 34.4% and 53.1% of input data sparsity for ImageNet dataset, respectively. VoteNet is a 3D object detection application which is based on PointNet++ [29]. It has 7-bit precision of inputs and weights for SUN RGB-D dataset [36], and it shows a 46.2% of input sparsity with a 64-to-1, a 32-to-1, and three 16-to-1 max pooling layers.

B. Evaluation Methodology

Sibia is designed in RTL Verilog and evaluated through RTL simulations to count exact cycles during the DNNs execution. It is synthesized in Samsung 28 nm CMOS technology using Synopsys Design Compiler and designed with Synopsys IC Compiler II for the place and route. Fig. 9 shows the post-layout of the MPU core in Sibia. The three PE arrays are aligned horizontally in the $1.024mm \times 1.043mm$ sized MPU core. In the layout, the top of the power metal layer in the MPU core is the IA layer, and the top of the signal metal layer is the M7 layer. The power consumption is measured with the Synopsys PrimeTime after the place and route. Then, the evaluation results report the performance of the MAC-based DNN operations excluding the point processing algorithms such as a neighbor search and 3D point sampling operation in VoteNet and DGCNN.

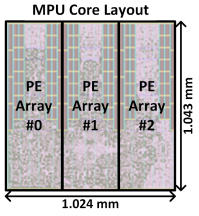

Fig. 9. Layout image of the Sibia's MPU core.

TABLE I
SPEC COMPARISON AMONG BIT-SLICE ACCELERATORS.

	Revised Bit-Fusion Core	Revised HNPU Core	**1 MPU Core**
Technology	28 nm Samsung	28 nm Samsung	**28 nm Samsung**
Frequency (MHz)	250	250	**250**
Area (mm^2)	0.746	1.125	**1.069**
Type of MAC Unit	5b×5b MAC Unit	5b×5b MAC Unit	**4b×4b Signed MAC Unit**
# of MAC Units	1536	1536	**1536**
DNN (Input 7-bit, Weight 7-bit) Performance			
Peak Throughput (GOPS)	144.0	309.6	**770.4**
Power (mW)	73.3	131.3	**100.7**
Energy-efficiency (TOPS/W)	1.97	2.36	**7.65**
Area-efficiency (GOPS/mm2)	192.9	275.2	**703.4**

The architecture control unit controls Sibia for DNN executions. It consists of a RISC-V (RV64IMAFDC) core, direct memory access (DMA), an external memory interface, and peripheral IPs, all of which share an AXI bus within the architecture control unit as shown in fig 4. The RISC-V core runs the system and transfers the instruction stream of DNN workloads to Sibia through the AXI bus for DNN accelerations. Sibia executes DNN models by retrieving the instructions and raises an interrupt to the RISC-V core after finishing the allocated DNN workloads. Off-chip memory is modeled using Cypress Semiconductor's HyperRAM [3] for an external DRAM. The architecture control unit integrates the HyperRAM interface to access the HyperRAM.

C. Overall Performance Comparison with Bit-slice Accelerators

Sibia is compared to the previous bit-slice accelerator, Bit-fusion [34], and zero bit-slice skipping accelerator, HNPU [9].

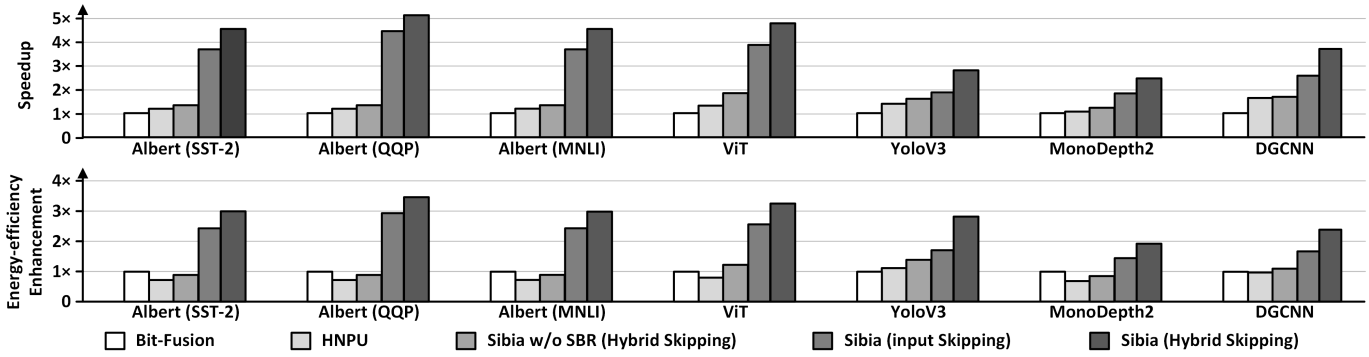

Fig. 10. Speedup and energy-efficiency comparison among bit-slice accelerators on dense DNN benchmarks.

This paper revised Bit-fusion and HNPU to match the same number of MAC units, the same technology node, and the same clock frequency for a fair comparison. Table I summarizes the comparison of the three accelerators. The 1.069 mm^2 MPU core in Sibia achieves 770.4 GOPS of peak throughput at 7-bit DNN performance while consuming 100.7 mW of power. Although area and power consumption of the MPU core is $1.43\times$ larger and $1.37\times$ higher than Bit-fusion, respectively, the peak throughput of the MPU core achieves $5.35\times$ higher. Then, area-efficiency and energy-efficiency of the MPU core are $3.65\times$ and $3.88\times$ higher than Bit-fusion, respectively. In comparison to HNPU, the peak throughput of the MPU core is $2.49\times$ higher while the area and power consumption of the MPU core are 5.0% smaller and 23.3% lower than HNPU, respectively. Therefore, area-efficiency and energy-efficiency of the MPU core is $2.56\times$ and $3.24\times$ higher than HNPU, respectively.

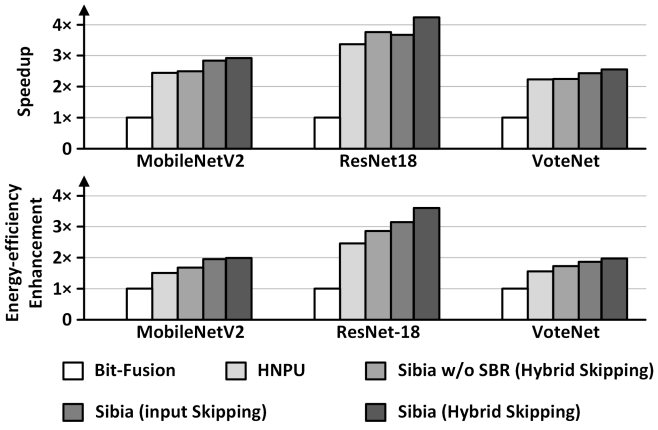

Fig. 11. Speedup and energy-efficiency comparison among bit-slice accelerators on sparse DNN benchmarks.

D. Comparison with Bit-slice Accelerators on Dense DNNs

Fig. 10 shows the performance comparison on the dense DNN benchmarks. Since the conventional bit-slice decomposition cannot generate enough zero bit-slices in dense DNNs, the zero bit-slice skipping of HNPU shows a small speedup on dense DNNs, $1.18\times$, $1.18\times$, $1.19\times$, and $1.31\times$ speedup in dense transformer-based Albert (SST-2, QQP, MNLI) and ViT, $1.35\times$, $1.08\times$ in dense 2-D image-based YoloV3 and MonoDepth2, and $1.63\times$ in dense 3-D point cloud-based DGCNN, respectively. Sibia skips more sparse data between input and weight as the hybrid skipping, resulting in higher speedup than HNPU even without the SBR. Then, the proposed SBR produces many zero bit-slices even in dense DNNs, and Sibia enhances the inference speed by $3.65\times$, $4.41\times$, $3.65\times$, and $3.83\times$ higher in Albert (SST-2, QQP, MNLI) and ViT, $1.88\times$, $1.86\times$ in YoloV3 and MonoDepth2, and $2.56\times$ in DGCNN by using input skipping, respectively. High bit-precision transformer-based DNNs achieve high throughput enhancement because their input data tend to be distributed in near-zero and generate many zero bit-slices after the SBR. Furthermore, Sibia can skip the more sparse data between input and weight by monitoring their sparsity. As a result, hybrid skipping increases the throughput by $4.50\times$, $5.07\times$, $4.50\times$, and $4.73\times$ higher in Albert (SST-2, QQP, MNLI) and ViT, $2.79\times$, $2.48\times$ in YoloV3 and MonoDepth2, and $3.67\times$ in DGCNN, respectively. Hybrid skipping shows higher speedup than input skipping on all of the benchmarks.

Energy-efficiency of HNPU is lower than Bit-fusion on Albert, ViT, MonoDepth2, and DGCNN because HNPU consumes high power to compute low bit-slice sparsity. On the other hand, the energy-efficiency of Sibia outperforms Bit-fusion and HNPU on all of the dense DNN benchmarks thanks to the SBR algorithm and efficient signed MAC unit and flexible zero skipping unit. Similar to the speedup, hybrid skipping on Albert (QQP) shows the highest energy-efficiency enhancement ($3.40\times$) among the dense DNN benchmarks.

E. Comparison with Bit-slice Accelerators on Sparse DNNs

Fig. 11 describes the performance comparison on the sparse DNN benchmarks. Sparse DNNs present sparse input data due to the ReLU activation function, and HNPU can increase the throughput more than twice after the conventional bit-slice decomposition. Sibia with the SBR additionally accelerates non-sparse convolution layers which are not applied ReLU activation function in sparse DNNs, and it can increase the throughput by $2.83\times$, $3.65\times$, and $2.42\times$ higher

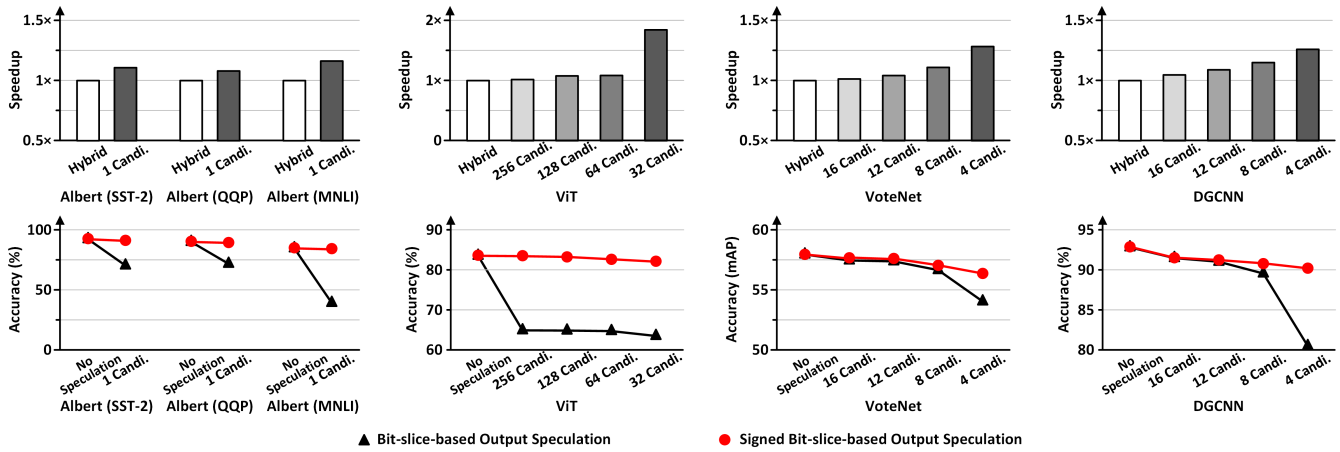

Fig. 12. Performance enhancement of output skipping over hybrid skipping with accuracy loss.

on MobileNetV2, ResNet-18, and VoteNet, respectively. The throughput enhancement by hybrid skipping shows a small speedup in MobileNetV2 and VoteNet because their sparsity of high order of weight signed bit-slice are lower than sparsity of low order of input bit-slice at the majority of convolution layers.

Similar to the speedup, HNPU has higher energy-efficiency than Bit-fusion on the sparse DNN benchmarks. Sibia processes bit-slice data efficiently thanks to the signed MAC unit and the flexible zero skipping unit even without the SBR, Sibia shows higher energy-efficiency than HNPU on all of the benchmarks. Then, the SBR enhances the Sibia performance much higher by generating many zero bit-slices. Finally, hybrid skipping with the SBR achieves $3.59\times$ energy-efficiency enhancement on ResNet-18 which is the highest among the all of the benchmarks.

F. Impact on Output Skipping

Fig. 12 shows additional performance enhancement by the output speculation with DNN accuracy. Sibia pre-computes high orders of bit-slices and skips insensitive outputs of a softmax and max pooling layer by setting corresponding inputs to zeros. In an Albert benchmark, Sibia finds one maximal candidate in each token and skips the remaining low order computations if the maximal candidate is higher than a predefined threshold, resulting in $1.15\times$ higher throughput than hybrid skipping for the MNLI task. For the output speculation of ViT, VoteNet, and DGCNNN, on the other hand, Sibia finds non-maximal candidates and skips their remaining low order computations. Sibia pre-computes $I_H \times W_H$ to speculate the maximal candidates in ViT and achieves $1.84\times$ higher throughput with 32 numbers of the maximal candidate. Similarly, Sibia executes pre-computations of $I_H \times W_H$ at 64-to-1 and 32-to-1 max-pooling layers and $I_H \times W_H + I_L \times W_H$ at three 16-to-1 max-pooling layers in VoteNet, and $I_H \times W_H + I_L \times W_H$ for all of max-pooling layers in DGCNN. The output skipping increases throughput exponentially as the number of candidates decreases. Then, the throughput enhancement of VoteNet and DGCNN shows $1.27\times$ and $1.25\times$ higher than hybrid skipping at 4 numbers of the maximal candidate, respectively.

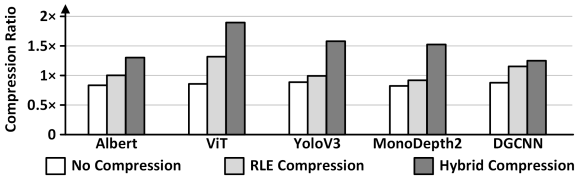

Fig. 13. Compression ratio of Sibia with the various compression modes on dense DNNs.

Although the bit-slice-based output speculation increases throughput, it hurts DNN accuracy. Especially, the output speculation using conventional unbalanced bit-slices rapidly degrades the DNN accuracy. For example, the output speculation using pre-computations of unbalanced bit-slices, $I_H \times W_H$, shows 45.0%p accuracy loss in an Albert (MNLI) benchmark. Moreover, DNN accuracy rapidly declines as the number of maximal candidates is reduced. Therefore, it limits the performance enhancement by requiring the pre-computation of high bit-precision of bit-slices, $I_H \times W_H + I_L \times W_H$ [35], [38], and the exploitation of many candidates for the output speculation. On the other hand, the SBR generates balanced signed bit-slices, increasing the success rate of the output speculation with a few candidates and achieving the minimum accuracy loss ($< 2\%p$) in all of the benchmarks.

G. Performance of Data Compression

Fig. 13 shows the compression ratio of input signed bit-slices on the dense DNN benchmarks. Since the SBR adds the 1-bit sign bit to each bit-slice, the size of raw signed bit-slices (No compression in Fig. 13) becomes bigger than the baseline. Then, the SBR generates a lot of zero signed bit-slices and shows $1.32\times$ and $1.15\times$ compression ratio at ViT and DGCNN, respectively, by using the RLE compression. However, low orders of signed bit-slices easily become

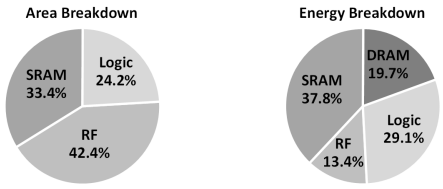

Fig. 14. Area and energy breakdown of Sibia.

TABLE II
COMPARISON AMONG NON-BIT-SLICE ACCELERATORS.

	SparTen [8]	S2TA-AW [24]	**Sibia-65nm**
Technology	45 nm	65 nm	**65 nm**
Type of Sparsity Exploitation	Unstructured Sparsity	Structured Sparsity	**Signed Bit-slice Sparsity**
Frequency	800 MHz	500 MHz	**500 MHz**
Area	0.766 mm^2	24 mm^2	**17.7 mm^2**
# of MAC Units	32 (INT8)	2048 (INT8)	**6144 (INT4)**
Peak Throughput* (TOPS)	-/0.2	2/4	**3.3/4.6**
Energy-efficiency* (TOPS/W)	-/-	-/1.1	**1.6/2.0**

*(<10% of input & weight sparsity / 50% of input & weight sparsity)

dense data which deteriorates the overall compression ratio, resulting in a low compression ratio at Albert, YoloV3, and MonoDepth2. Therefore, the dense low orders of signed bit-slices are not compressed (hybrid compression in Fig. 13), and the compression ratio of Albert, YoloV3, and MonoDepth2 achieves 1.31×, 1.57× and 1.54×, respectively. Consequently, encoding to signed bit-slices overcomes the 1-bit sign bit overhead and achieves a high compression ratio even in dense DNNs.

H. Area and Power Breakdown

Fig. 14 illustrates the area and energy breakdown of Sibia. The area breakdown is measured by Synopsys Design Compiler after logic synthesis. The register file (RF) and on-chip SRAM take up 42.4% and 33.4% of the total area, respectively. Then, control and compute logic accounts for 24.2% of the total area. In the energy breakdown, the on-chip SRAM, RF, and logic take up 37.8%, 13.4%, and 29.1% of the overall energy consumption, respectively. The energy consumption of the external DRAM is estimated by counting the read and write time of the HyperRAM, and it accounts for 19.7% of the overall energy consumption.

I. Comparison with Non-bit-slice Accelerators

Sibia is compared with a state-of-the-art structured sparse DNN accelerator S2TA-AW [24] and a state-of-the-art unstructured sparse DNN accelerator SparTen [8]. Sibia is re-synthesized in 65 nm technology (Sibia-65nm) for a fair comparison. As shown in Table II, Sibia outperforms S2TA-AW [24] over 1.65× higher peak throughput at $< 10\%$ sparsity of input and weight by exploiting zero signed bit-slices. Similarly, Sibia has 1.15× and 1.82× higher peak

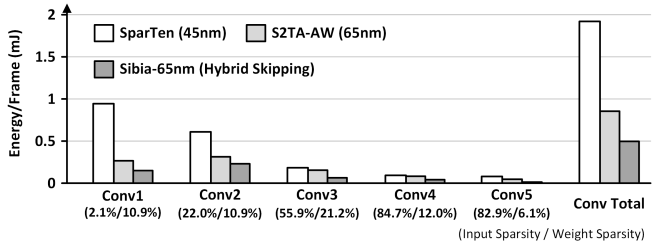

Fig. 15. Energy comparison among non-bit-slice accelerators on AlexNet.

throughput and energy-efficiency than S2TA-AW, respectively, at 50% sparsity of input and weight. On a per-layer energy comparison on AlexNet in Fig 15, Sibia consumes lower energy consumption than S2TA-AW at all of the dense and sparse convolution layers, achieving 1.7× lower total energy consumption. Despite the disadvantage of the older technology node, Sibia has 2.9× lower total energy consumption than SparTen on AlexNet. The previous sparse DNN accelerators including SparTen and S2TA-AW exploit some sparsity at dense input and weight data by using the pruning method with a sacrifice of DNN accuracy and an additional retraining procedure. On the other hand, Sibia secures sufficient signed bit-slice sparsity from dense input and weight data even without using the pruning method or retraining. Finally, Sibia achieves much higher throughput and lower energy consumption than the state-of-the-art sparse DNN accelerators.

J. Comparison with GPUs

Sibia is compared with the (1) high performance GPU - NVIDIA RTX 2080 Ti with 32-bit floating-point calculations compiled with the CUDA and (2) low power mobile GPU in mobile AP - Adreno 650 in Snapdragon 865 with 16-bit floating-point calculations of the TensorFlow-lite models on MonoDepth2. Sibia is $7.8\times$ faster inference speed than Adreno 650. Although the inference speed of RTX 2080 Ti is $4.3\times$ faster than Sibia, the energy-efficiency is the lowest because of its high power consumption (a TDP of 250 W). Finally, Sibia has $144.9\times$ higher energy-efficiency than RTX 2080 Ti, $97.7\times$ higher than Adreno 650.

IV. DISCUSSION

A signed-magnitude format represents a signed number with a sign bit and magnitude bits, and the bit-slice decomposition with the signed-magnitude representation also benefits symmetric bit-slices. However, the signed-magnitude MAC unit requires a 2's complementer for product accumulation, which is more inefficient than the 2's complement-based signed MAC unit [25]. In specific, the 4-bit signed-magnitude MAC unit which consists of an unsigned multiplier, an XOR gate for sign bits, a 2's complementer, and an accumulator has a 16.3% larger logic area than the 4-bit signed MAC unit in a 28 nm technology node. The area overhead rises to 45.4% when the bit-width of the MAC unit is 8-bit. Consequently, this paper focuses on software-hardware designs based on a 2's

complement number and would be extended to the previous 2's complement number-based architectures and future proposals.

V. Related Works

A. Bit-precision Reconfigurable Architecture

Quantization is the popular optimization method by reducing the bit-precision of input and weight from full precision data. Since lowering the bit-precision alleviates data transactions and computational complexity, many NPUs take advantage of the quantization. Then, Stripes [18] and UNPU [21] accelerators use bit-serial computing units to accelerate low-bit quantized DNNs by temporally bit-wise computing. Similarly, a bit-slice architecture composes of bit-slice PEs which use 2-bit, 4-bit, or 8-bit MAC units and dynamically matches various bit-width of low-bit quantized DNNs by spatially and temporally bit-slice-level computing. A bit-slice architecture shows much higher area-efficiency and energy-efficiency than a bit-serial architecture while taking advantage of bit-flexibility [34]. With this advantage, a bit-slice architecture is widely used in commercial mobile NPUs [17], [22]. However, a bit-serial architecture and a bit-slice architecture take many cycles to implement high bit-precision applications because of time-multiplexed computations. As a result, Sibia exploits signed bit-slice sparsity to speed up various bit-precision of quantized DNN executions.

B. Sparsity Exploiting Architecture

Sparsity exploitation can enhance throughput and energy-efficiency of architectures. It skips redundant computations caused by zero input or weight data. To increase data sparsity, a pruning method [11] is widely used by removing insensitive weight data. Then, zero weight skipping accelerators [10], [42] skip zero pruned weight data, boosting up hardware performance. Although the pruning method generates considerable zero values, it requires additional DNN retraining to maintain DNN accuracy. Therefore, the commercial mobile NPU [17] does not exploit the pruning method because of generality.

Zero input skipping accelerators [1], [17], [22] exploit input sparsity caused by the ReLU activation function. Similarly, the zero input bit-slice skipping architecture [9] skips zero bit-slices after the bit-slice representation. However, they cannot increase throughput and energy-efficiency at dense DNN models where the non-ReLU activation functions are used instead of the ReLU activation function. On the other hand, Sibia generates many zero bit-slices at dense DNNs, and the flexible zero skipping PE skips more sparse data between input and weight to enhance hardware performance.

C. Output Skipping Architecture

Instead of the exploitation of sparse input and weight data, output skipping architectures exploit the output sparsity presented by a softmax operation [38] or a max pooling operation [16], [35]. For example, transformer-based DNNs [4], [5], [19] use softmax layers for attention probabilities, and most of them show small quantization errors after a softmax operation. Similarly, 3-D point cloud-based DNNs [27]–[29], [39] have large-scale (e.g. 64-to-1) max pooling layers, so 98% of convolution outputs are zeros after a 64-to-1 max pooling layer.

The previous output skipping architecture [35] speculates maximal outputs of a small-scale (4-to-1) max pooling layer by pre-computing high orders of bit-slices and skipping the remaining low orders of bit-slice computations to achieve high throughput and energy-efficiency. Another output skipping architecture [38] speculates insensitive outputs by a softmax layer. Both architectures support the bit-precision reconfigurability to execute both the output speculation and input skipping in a homogeneous hardware unit. Other architecture [16] skips insensitive outputs of a large-scale (64-to-1) max pooling layer. The prediction core executes the binary convolution of 1-bit weights to speculate the insensitive outputs while the convolution core only computes maximal candidates. However, all of the output skipping architectures easily fail the output speculations when using low-bit of both input and weight due to their unbalanced 2's complement numbers, degrading the hardware performance. In contrast, Sibia enables the low-bit output speculation thanks to balanced signed bit-slices, and the flexible zero skipping PE fully supports the output sparsity exploitation.

VI. Conclusion

The paper presents the efficient signed bit-slice architecture, Sibia, for dense DNN acceleration. Unlike a conventional bit-slice decomposition method, the signed bit-slice representation produces signed bit-slices from a 2's complement data, generating a lot of zero bit-slices and balancing positive and negative values. With these advantages, Sibia increases the hardware performance by integrating the efficient signed MAC units and the flexible zero skipping PE for zero input and output skipping. Moreover, Sibia supports zero weight skipping by encoding weight data instead of input data and feeding non-zero weight bit-slices and indices to the IBUF and the IDXBUF. The DSM unit monitors sparsity of input and weight bit-slice and determines the sparsity exploitation modes, resulting in high throughput and energy-efficiency enhancement with a high compression ratio. The heterogeneous NoC flexibly transfers the input and weight data by considering the data reusability and reduces the transmission bandwidth of output partial sum data. As a result, Sibia outperforms previous bit-slice accelerators in both dense and sparse DNN benchmarks.

References

[1] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 1–13, 2016.

[2] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[3] Cypress Semiconductor, "64Mbit HyperRAM Self-Refresh DRAM Data-Sheet," *Cypress Int.*, Inc.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[6] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[7] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[8] A. Gondimalla, N. Chesnut, M. Thottethodi, and T. Vijaykumar, "Sparten: A sparse tensor accelerator for convolutional neural networks," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 151–165.

[9] D. Han, D. Im, G. Park, Y. Kim, S. Song, J. Lee, and H.-J. Yoo, "Hnpu: An adaptive dnn training processor utilizing stochastic dynamic fixed-point and active bit-precision searching," *IEEE Journal of Solid-State Circuits*, 2021.

[10] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[11] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[15] A. Ignatov, J. Patel, and R. Timofte, "Rendering natural camera bokeh effect with deep learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 418–419.

[16] D. Im, S. Kang, D. Han, S. Choi, and H.-J. Yoo, "A 4.45 ms low-latency 3d point-cloud-based neural network processor for hand pose estimation in immersive wearable devices," in *2020 IEEE Symposium on VLSI Circuits*. IEEE, 2020, pp. 1–2.

[17] J.-W. Jang, S. Lee, D. Kim, H. Park, A. S. Ardestani, Y. Choi, C. Kim, Y. Kim, H. Yu, H. Abdel-Aziz, J.-S. Park, H. Lee, D. Lee, M. W. Kim, H. Jung, H. Nam, D. Lim, S. Lee, J.-H. Song, S. Kwon, J. Hassoun, S. Lim, and C. Choi, "Sparsity-aware and re-configurable npu architecture for samsung flagship mobile soc," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 15–28.

[18] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–12.

[19] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[20] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[21] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "Unpu: A 50.6 tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 218–220.

[22] C.-H. Lin, C.-C. Cheng, Y.-M. Tsai, S.-J. Hung, Y.-T. Kuo, P. H. Wang, P.-K. Tsung, J.-Y. Hsu, W.-C. Lai, C.-H. Liu *et al.*, "A 3.4-to-13.3 tops/w 3.6 tops dual-core deep-learning accelerator for versatile ai applications in 7nm 5g smartphone soc," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 134–136.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[24] 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)*. IEEE, 2022, pp. 573–586.

[25] R. Lyon, "Two's complement pipeline multipliers," *IEEE Transactions on Communications*, vol. 24, no. 4, pp. 418–425, 1976.

[26] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[27] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, "Imvotenet: Boosting 3d object detection in point clouds with image votes," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4404–4413.

[28] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9277–9286.

[29] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[30] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014. [Online]. Available: http://arxiv.org/abs/1409.0575

[32] S. Ryu, H. Kim, W. Yi, and J.-J. Kim, "Bitblade: Area and energy-efficient precision-scalable neural network accelerator with bitwise summation," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[34] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 764–775.

[35] M. Song, J. Zhao, Y. Hu, J. Zhang, and T. Li, "Prediction based execution on deep neural networks," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 752–763.

[36] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 567–576.

[37] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[38] H. Wang, Z. Zhang, and S. Han, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 97–110.

[39] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[40] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra, "Embodied question answering in photorealistic environments with point cloud perception," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6659–6668.

[41] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[42] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-x: An accelerator for sparse neural networks," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–12.

[43] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.