*Oz*MAC: An Energy-Efficient Sparsity-Exploiting Multiply-Accumulate-Unit Design for DL Inference

Harideep Nair*§, Prabhu Vellaisamy*§, Tsung-Han Lin†, Perry Wang†, Shawn Blanton*, John Paul Shen*
* Carnegie Mellon University
† MediaTek USA Inc.

arXiv:2402.19376v1 [cs.AR] 29 Feb 2024

***Abstract*—General Matrix Multiply (GEMM) hardware, employing large arrays of multiply-accumulate (MAC) units, perform bulk of the computation in deep learning (DL). Recent trends have established 8-bit integer (INT8) as the most widely used precision for DL inference. This paper proposes a novel MAC design capable of dynamically exploiting bit sparsity (i.e., number of '0' bits within a binary value) in input data to achieve significant improvements on area, power and energy. The proposed architecture, called *Oz*MAC (*Omit-zero*-MAC), skips over zeros within a binary input value and performs simple shift-and-add-based compute in place of expensive multipliers. We implement *Oz*MAC in SystemVerilog and present post-synthesis performance-power-area (PPA) results using commercial TSMC N5 (5nm) process node. Using eight pretrained INT8 deep neural networks (DNNs) as benchmarks, we demonstrate the existence of high bit sparsity in real DNN workloads and show that 8-bit *Oz*MAC improves all three metrics of area, power, and energy significantly by 21%, 70%, and 28%, respectively. Similar improvements are achieved when scaling data precisions (4, 8, 16 bits) and clock frequencies (0.5 GHz, 1 GHz, 1.5 GHz). For the 8-bit *Oz*MAC, scaling its frequency to normalize the throughput relative to conventional MAC, it still achieves 30% improvement on both power and energy.**

I. Introduction and Background

A. Deep Learning Accelerators

General matrix multiply (GEMM) hardware, employing large arrays of multiply-accumulate (MAC) units, is the core compute fabric for modern deep learning accelerators (DLAs) [27], [28]. A typical deep neural network (DNN) consists of convolution, activation, pooling, and fully-connected layers [12]. Empirical results show that most DNN computation (80% to 90%) rely heavily on the GEMM hardware in DLAs [9]. Hence, the die area, dynamic power, and energy efficiency of the GEMM hardware, and in turn, its primary building blocks, i.e., MAC units, are of utmost importance, especially for edge devices. This is illustrated in Fig. 1.

Leveraging large number of MAC units, DLAs are dedicated processors to optimize the compute for DNN workloads. DLAs are widely deployed in data centers for accelerating both the training phase as well as the inference phase of diverse deep learning workloads [24]. One prominent example of such DLAs is the Google TPU [14], which can employ a 2D array of upto 256x256 MAC units to perform GEMM on 256x256 matrices.

§Equal contribution

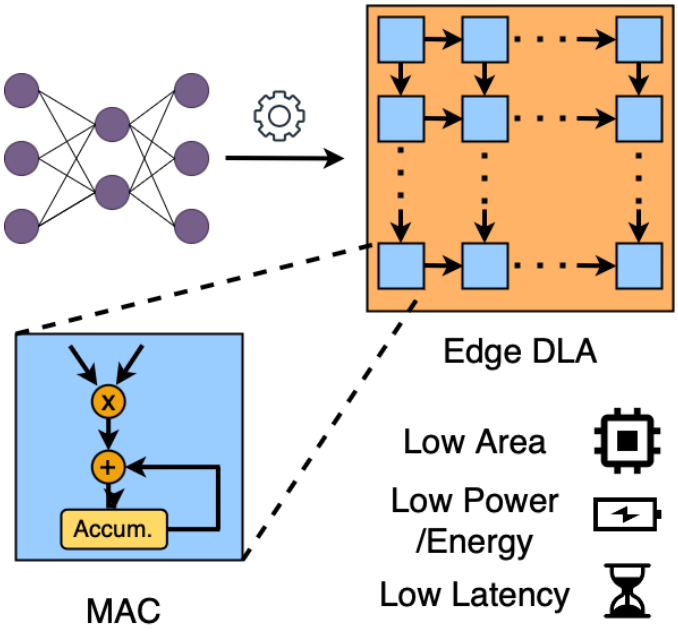

Fig. 1. Illustration of an edge DLA comprising of an array of MAC units performing DNN inference compute targeting low area, power and latency

B. Importance of MAC Units in DLAs

Multiply-accumulate (MAC) units form the fundamental building blocks in a deep learning accelerator (DLA), which are scaled to form large arrays of processing elements (PEs), or PE arrays, that perform large scale GEMM operations. Fig. 2 shows the typical DLA system hierarchy consisting of an array of MAC units. A MAC operation is defined as $c \leftarrow c + (a \times b)$. It computes the product of two values (in this context, the weight and activation inputs of a DNN), and adds the resultant product to an accumulator (c). A conventional bit-parallel-based hardware MAC unit consists of a combinational array multiplier and an adder to accumulate the product, and a register to store the value. Since scaled arrays of these MAC units form the compute fabric of the DLA, optimizing the MAC units can yield a significant reduction in hardware cost and power consumption of DLAs. Any improvement on the MAC unit design is replicated many fold in the large number of MAC units in MAC arrays of typical DLAs. This has led to a plethora of research efforts on hardware-efficient MACs [29]

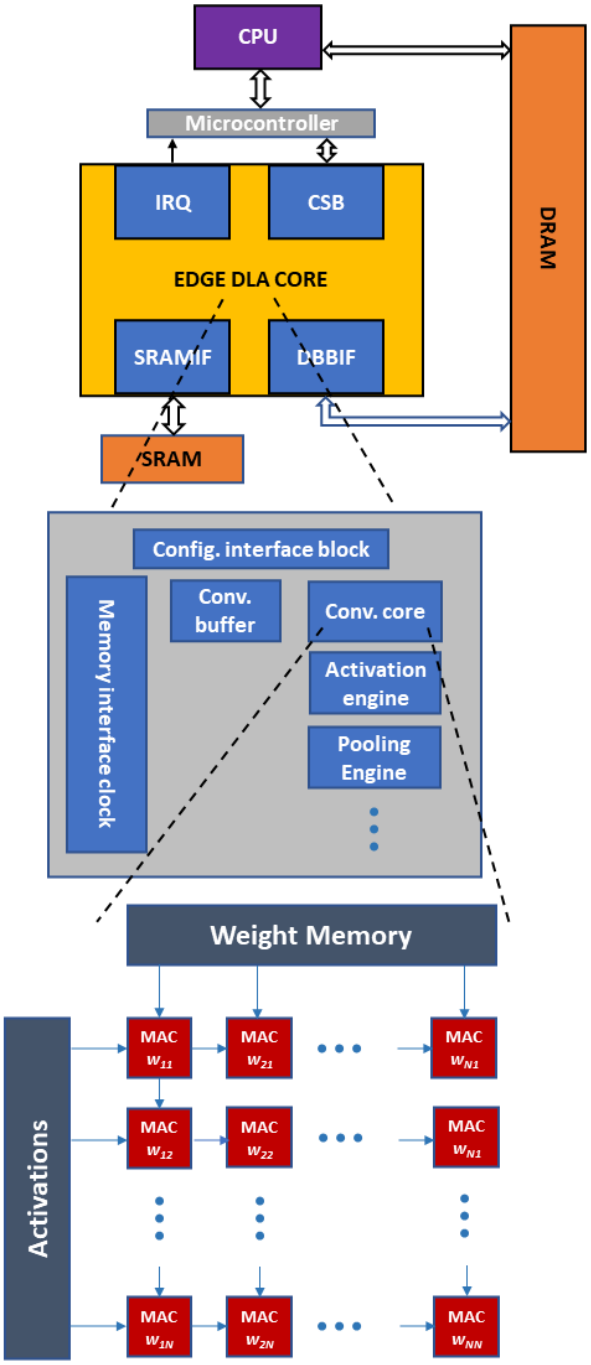

Fig. 2. Typical DLA system level organization with interface to hosting CPU and memories, and an large array of MAC units as its core GEMM hardware

[34], improved dataflow schemes [5] [6] [19], novel memory topologies [13] [16], circuit-level optimizations [20] and low-precision DL compute [4] [7] [11] [25].

C. Industry Trend Towards Low Precision

There are two current industry trends with deep learning: 1) moving to lower precisions for both training and inference [33], and 2) moving inference from data centers to edge devices [17]. A recent study from IBM [31] highlights the trend towards lower precision. They indicate that the precisions needed to achieve the same accuracy are scaling down (from 2012 to 2018) from 32-bits to 8-bits for training and from 16-bits to 4-bits for inference. They show that the primary precision format used for training evolved from FP32 to FP16 to FP8, and for inference from INT16 to INT8 to INT4. FP16 format has been widely adopted in the industry, with 16-bit accelerators showing 4x to 8x speedup in performance compared to 32-bit designs [26]. Both PyTorch [22] and TensorFlow [3] support quantized INT8 models, allowing a 2x to 4x faster compute relative to FP32 [1]. ResNet-101 trained with 4-bit weights and 8-bit activations has been reported to incur only a TOP-1 accuracy loss of 2% [18]. Additionally, training on CIFAR-10 datasets with 5-bit weights and 4-bit activations has been shown with minimal performance degradation [21]. Furthermore, successful 4-bit training on a set of DL workloads has been reported to provide 7x hardware acceleration over state-of-the-art FP16 systems, with minimal accuracy losses [26].

The current industry standard for inference has moved from 32-bit floating-point (FP32) format to 16-bit floating-point (FP16) and 8-bit integer (INT8) formats. For instance, NVIDIA TensorRT [30] allows models to be deployed in both FP16 and INT8 without compromising on accuracy.

D. Industry Trend Towards Edge Inference

Another trend is moving inference from the data center to edge devices [32]. This has a number of benefits. The inference latency for the user can be significantly reduced. Overall network bandwidth required can also be reduced. There is also the benefit of ensuring personal privacy and supporting personalized services. However, to support such edge inference, the on-device DLAs must be extremely energy efficient and incur significantly lower costs [35]. This is the main focus of our work: low-cost and energy-efficient DLAs for low-precision (INT8) edge-based DNN inference.

E. Exploiting Value Sparsity in DL Workloads

Unlike general GEMM computation in other domains, DL workloads exhibit strong value sparsity, both word sparsity (zero values) and bit sparsity (zero bits in non-zero values). Previous works [10], [23] have shown that activations and weights can be transformed to exhibit up to 50% and 90% sparsity with only 50% and 10% non-zero values, respectively. Our goal in this work is to exploit value sparsity to achieve dynamic power and energy efficiency which can also reduce the die area cost.

To depict the inherent sparsity of edge inference benchmarks, we evaluate popular pretrainedv and quantized INT8 models in Section IV on the basis of their *bit-sparsity*, referring to the majority of the bits in the 8-bit word being zeros. The sparsity metric that is generally leveraged for model compression techniques to increase performance and lower compute costs is *word-sparsity*, which refers to the whole word being zero, in this context 8-bit words are equal to 0. From our evaluation, it is found that the weight values are

significantly sparser than the activation values. In this work, word-sparsity can be naturally leveraged to curb computations for zero values, just like conventional techniques. However, bit sparsity is not leveraged by existing MAC hardware and hence is the focus of our work.

In this research work, we present *Oz*MAC, a novel sparsity-exploiting multiply-accumulate MAC unit for integer-based inference, and show improvements in die area, power and energy, when compared to conventional bit-parallel MAC with the same throughput. Key contributions of our work are:

- Present a novel *Oz*MAC design for DL inference, capable of exploiting dynamic bit sparsity to reduce power by skipping over zero bits in the input binary values.
- Evaluate and demonstrate high bit sparsity for eight state-of-the-art pretrained INT8 DNN benchmark models.
- Implement a wide range of *Oz*MAC designs using commercial design tools and the current TSMC N5 PDK (process design kit) for the 5nm process node.
- Generate and evaluate power-performance-area (PPA) results for various configurations of *Oz*MAC with scaling of data precisions (4-bits, 8-bits, 16-bits) and clock frequencies (500 MHz, 1 GHz, 1.5 GHz).
- Show the benefits of all the *Oz*MAC designs in simultaneously achieving significant improvements in area, power and energy, at the same clock frequency.
- Demonstrate the significant power reduction of *Oz*MAC and how this can be leveraged to increase throughput with frequency scaling.

The paper is organized as follows. We present the *Oz*MAC microarchitecture in Section II and describe the hardware evaluation methodology including commercial tools and libraries used, in Section III. Next, we perform sparsity evaluation using eight well-known DNN benchmarks in Section IV, followed by rigorous performance-power-area (PPA) and energy evaluations for 8-bit designs at 500 MHz frequency in Section V. We further provide bit-width scaling analysis for *Oz*MAC against the bit-parallel counterpart for precisions of INT4, INT8, and INT16 in Section VI, followed by frequency scaling evaluation in Section VII. Finally, we discuss key conclusions and future directions in Section VIII.

II. OzMAC Microarchitecture and Design

On a functional level, a single MAC operation comprises of a multiplication and an addition as follows: A * B + C, where A and B are binary input values and C is the previously accumulated value. As discussed earlier, this function is fundamental to all deep neural networks (DNNs) and hence is implemented as a MAC hardware unit in dedicated deep learning accelerators (DLAs). Conventional MAC unit presently implemented in DLAs performs this compute in one cycle, employing expensive multilplier and is referred to as "bMAC" in this paper. In this section, we present the proposed *Oz*MAC architecture and its key components, and highlight its differences relative to bMAC. It is to be noted that *Oz*MAC architecture displays significant similarities to existing works on Stripes [15] and Bit Fusion [25].

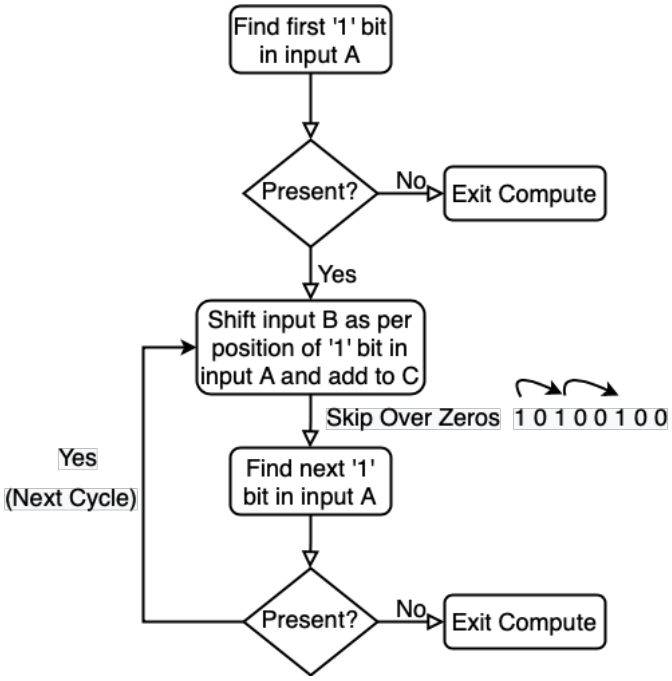

Fig. 3. *Oz*MAC Zero-Skipping Compute Algorithm

*Oz*MAC (Omit-zero-MAC) is a novel shift-and-add-based multiply-accumulate design that is capable of exploiting dynamic bit sparsity by omitting or skipping over '0' bits in the input binary values. Compared to single-cycle bMAC, *Oz*MAC performs compute over multiple cycles, taking one cycle for every '1' bit in the input data. The key compute algorithm is depicted in Fig. 3. Given binary inputs A and B, in the first compute cycle, it finds the first '1' bit in input A beginning from the most significant bit (MSB), deciphers its position and shifts input B accordingly, eventually placing it in the accumulator. In the next cycle, it finds the next '1' bit in A, skipping over any intermediate '0' bits, shifts B and adds it to the accumulator. This continues until it goes through all the '1' bits in A, taking as many cycles as the number of '1's in A. In the best case scenario when A is a power of 2, it consumes only 1 cycle similar to bMAC. On the other hand, when A is of the form $2^n - 1$, it takes n cycles. It is to be noted that *Oz*MAC ingests inputs in bit-parallel format just like conventional bit-parallel MACs (and unlike bit-serial hardware); the zero-skipping serialization occurs within *Oz*MAC.

The microarchitecture of *Oz*MAC consists of three simple functional modules as shown in Fig. 4: 1) *Oz-encoder*, 2) *shifter*, and 3) *accumulator* (adder and register for storage). *Oz-encoder* is a Finite State Machine which keeps track of the current and next positions of '1' in the input bit pattern. Using this information, it outputs a one-hot encoded value capturing the bit positions of '1's every clock cycle for as many cycles as the number of '1's. For example, as illustrated in Fig. 4, the input '0101_2' is encoded as two one-hot values spanning two clock cycles: '0100_2' in the first cycle and '0001_2' in

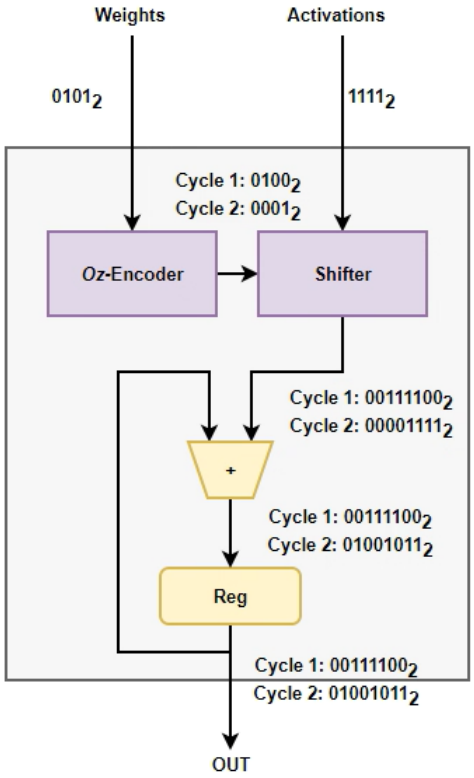

Fig. 4. *Oz*MAC microarchitecture with example compute

the next cycle. By doing this, it skipped over the two '0's and only incurs compute cycles for the '1's. The fundamental microarchitectural difference between *Oz*MAC and bMAC designs is the replacement of combinational multiplier with an encoder finite state machine (FSM) and a shifter. Here, the number of compute cycles incurred for one MAC operation by *Oz*MAC is equal to the number of '1's in the input binary value, different from bit-serial MAC designs where the number of clock cycles is equal to the bit-width n. The *Oz-encoded* input then goes to the shifter and determines the shift magnitude of the second input. The appropriately shifted second input is then added to the accumulator value. This is described in further detail via an example below (also depicted in Fig. 4).

Consider two inputs: A = '0101_2' and B = '1111_2'. Input A is fed into *Oz*-encoder, while input B is provided to the shifter. Assume compute begins after a reset, hence accumulator register will have value 0 stored in it. As there are two '1's in input A, this computation will execute in two clock cycles, regardless of the '0' bits and their positions. This is termed as *zero-skipping* or *zero-omitting*.

In the first cycle, the encoder outputs '0100_2' as the one-hot encoded value after detecting the first '1' from MSB side. Based on this output, the shifter left shifts input B by two positions, owing to '1' positioned at the 'second power of 2' in the one-hot value, and outputs a 16-bit value. Therefore, in cycle 1, the shifter outputs '$1111_2 << 2 = 00111100_2$' which gets accumulated in the register. In the second cycle, encoder outputs 0001_2 to the shifter, but no left shift occurs due to the zeroth position of '1' in the one-hot value. Hence, shifter outputs '00001111_2' to the adder, and the register accumulates '$00111100_2 + 00001111_2 = 01001011_2$' and passes it as its final output at the end of second cycle.

In terms of gate complexities, *Oz*MAC scales differently than bMAC. Accumulator is consistent in both designs and scales $O(n)$. Hence, the scaling trends between both designs are largely determined by the multiplication modules. In *Oz*MAC, the encoder scales $O((logn)^2)$, while the shifter scales $O(nlogn)$ due to the multiplexer-based barrel-shifter design elaborated. The two functional modules scale relatively more efficiently than a complex combinational multiplier, especially for higher bit widths.

The energy consumption of *Oz*MAC depends on two main factors: 1) power consumption per cycle, and 2) the number of compute cycles. *Oz*MAC exploits bit-sparsity present in DL workloads to cut down on the number of compute cycles by zero-skipping the weight binary values. This inherent advantage of this novel MAC design ensures that the sparser the DNN model, the higher the speedup and the lesser the energy consumption.

It is to be noted that *Oz*MAC can directly replace existing conventional bMACs as it takes in conventional bit-parallel inputs just like bMAC (*Oz*-encoding is performed within the *Oz*MAC hardware). The only additional overhead needed is a handshaking protocol to synchronize different MAC operations, as each MAC compute takes variable number of cycles depending on the input bit sparsity. This handshaking protocol is implemented as "ready-valid" signals as part of the I/O interface for each *Oz*MAC, and we consider this overhead for all our hardware analysis.

III. Hardware Framework for Evaluation

We perform rigorous, industry-standard evaluation of the *Oz*MAC design to get accurate PPA results and energy numbers and compare the same to a conventional bit-parallel MAC design. The PDK (process design kit) and technology library used for evaluation is the TSMC N5 (5nm) process node, with Synopsys design tools employed for simulation, synthesis, and power calculations.

The *Oz*MAC RTL design is first created in SystemVerilog, with functional verification performed using Synopsys VCS. A synthetic dataset comprising 1000 sample weights and activation values is developed, with the values reflecting the sparsity levels of the DNN benchmarks under consideration. This allows for the appropriate switching activity to be captured, as well as resultant average *Oz*MAC compute cycles to be reported by the means of a testbench.

Consequently, lint check is performed on the SystemVerilog source file using Synopsys SpyGlass and then synthesis is performed to convert the RTL-level design into a gate-level netlist using Synopsys Design Compiler, sourcing TSMC N5 library files. Gate-level netlist simulation is then performed for verification and collection of the switching activity of the

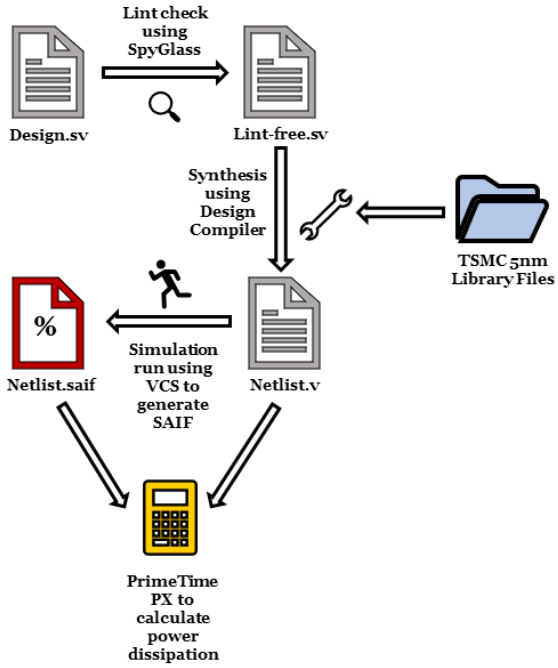

Fig. 5. Hardware design process flow for generating *Oz*MAC PPA reports using Synopsys toolchain and TSMC 5nm library

design in the form of a SAIF dump. The SAIF dump is then sourced along with the netlist to perform accurate power calculations using Synopsys PrimeTime PX. This design process flow is illustrated in Fig. 5.

We present sparsity scaling trends for *Oz*MAC in Section IV to illustrate the correlation between compute latency and bit sparsity in DNN weights. We report the INT8 inference PPA results and resultant energy consumption values for both conventional bit-parallel MAC (bMAC) and *Oz*MAC in Section V. Further, we report the hardware complexity scaling in TSMC N5 process node for *Oz*MAC in Section VI, including PPA and energy values for 4-bits, 8-bits, and 16-bits precisions at the clock frequency of 500 MHz. Finally, to evaluate the frequency scaling for *Oz*MAC, we synthesize the *Oz*MAC designs for differing frequency values (500 MHz, 1 GHz, 1.5 GHz) to show PPA and energy trends as frequency is scaled. We also scale just the *Oz*MAC frequency to match the bMAC throughput and show that *Oz*MAC still exhibitt area, power, and energy benefits at the same throughput.

IV. Sparsity Scaling Analysis

*Oz*MAC performs highly efficient shift-and-add operations and trades off latency for lower area and power. The "omit-zero" capability of *Oz*MAC is key to mitigating this latency overhead by exploiting dynamic bit sparsity in input data. In other words, higher bit sparsity (i.e., more zero bits in the input data) will result in shorter compute latency (in cycle count). In this section, we perform two types of sparsity evaluations: 1) We derive the compute latency per MAC operation across varying input bit sparsity for *Oz*MAC and assess the latency overhead (cycle counts) relative to the conventional (single cycle) bMAC; and 2) We use eight pretrained and quantized INT8 DNN models as benchmarks to assess the typical bit sparsity present in real DNN workloads.

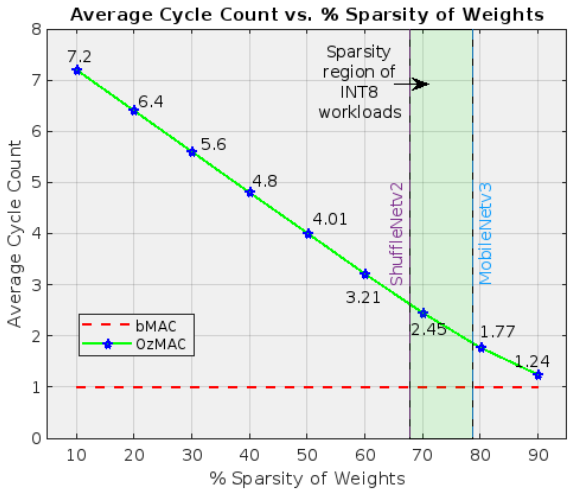

Fig. 6. Average cycle count vs. % bit-sparsity. Green-shaded region depicts the sparsity regions for INT8 workloads from Table I

TABLE I
Bit Sparsity and Cycle-Count Overhead for Pretrained Weights for Eight INT8 Quantized DNN Benchmarks

DNN Benchmark	Average number of "1" bits (Actual cycle-count overhead)	Bit Sparsity Percentage
MobileNetV2	2.334	70.83%
MobileNetV3	1.711	78.61%
InceptionV3	2.430	69.62%
ShuffleNetV2	2.583	67.71%
GoogleNet	2.461	69.24%
ResNet18	2.398	70.02%
ResNet50	2.495	68.81%
ResNeXt101	2.289	71.39%

Fig. 6 plots the cycle count for a single 8-bit MAC operation for both *Oz*MAC and bMAC, as bit sparsity is increased from 10% to 90%. The cycle counts for *Oz*MAC are derived via functional simulation, averaged across 1000 randomly generated test vectors exhibiting a particular bit sparsity. Note that bMAC takes only 1 cycle regardless of sparsity. *Oz*Mac incurs 7.2 cycles on average when bit sparsity is only 10%, which reduces to 4 cycles at 50% and significantly down to 1.2 cycles at 90% bit sparsity. This implies that, in order to achieve lower energy consumption than bMAC at any given bit sparsity, *Oz*MAC's power consumption has to be proportionally lower to offset the relatively higher latency. This is discussed in detail in Section V. Next, we analyse the typical bit sparsity present in real DNN workloads to provide the context for Section V.

TABLE II
TSMC N5 PPA AND ENERGY (AVERAGED ACROSS THE EIGHT DNN BENCHMARKS) FOR 8-BIT OZMAC AND 8-BIT BMAC AT 500 MHZ

MAC Hardware	Area (μm^2)	Power (mW)	Latency (ns)	Energy (pJ)
bMAC	25.361	0.084	2	0.167
OzMAC	19.996	0.025	4.76	0.120
% Improvement	21.2	69.7	-	28.0

We use eight pretrained and quantized INT8 DNN models [2], available as part of PyTorch's Torchvision library, that are widely used in state-of-the-art deep learning literature and MLPerf benchmarking, namely, 1) *MobileNetV2*, 2) *MobileNetV3*, 3) *InceptionV3*, 4) *ShuffleNetV2*, 5) *GoogleNet*, 6) *ResNet18*, 7) *ResNet50* and 8) *ResNeXt101*. Layer-by-layer analysis of the converged weights and activation values resulting from running ImageNet [8] inputs illustrate the sparsities inherent in these benchmarks. For each model, we extract the average number of "0" bits in every 8-bit weight value across all the layers and calculate the bit sparsity as the percentage of "0" bits over the total number of bits.

These values are summarized in Table I. The highlighted region in Fig. 6 indicates the bit sparsity range observed in the eight benchmark models, which results in effective cycle counts in the range of 1.7-2.5 cycles of compute latency on *Oz*MAC for these benchmarks. Comparing this to 1 cycle latency of bMAC, *Oz*MAC's power consumption must be 1.7-2.5x lower than that of bMAC to achieve lower energy consumption. Next section provides actual post-synthesis PPA results using commercial design tools and the commercial TSMC N5 (5nm) process node to illustrate this point.

Key Takeaway: The intrinsic capability of *Oz*MAC to exploit bit sparsity dynamically in real DNN workloads, allows it to significantly reduce the actual latency overhead from the worst possible case of 8 cycles down to only 1.7-2.5 cycles.

V. OZMAC EVALUATION FOR INT8 INFERENCE

In this section, we compare the PPA of 8-bit *Oz*MAC and 8-bit bMAC generated using commercial design tools and TSMC N5 process node for the INT8 DNN benchmarks introduced in the previous section. Note that the power consumption values are obtained through the PTPX tool using benchmark-specific test vectors that capture their bit sparsity characteristics.

Table II provides the die area, power, latency and energy consumption of *Oz*MAC and bMAC across the eight DNN benchmark models. The operating frequency for both designs is 500 MHz. An 8-bit conventional bMAC computes 1 MAC operation in 2 ns (1 cycle) while consuming about 25 μm^2 area, 84 μW power, and 167 fJ energy, whereas *Oz*MAC only consumes about 20 μm^2 area, 25 μW power, and 120 fJ energy while incurring 4.76 ns latency on average. Compared to conventional bMAC, this amounts to 21% less die area, 70% less power, and 28% less energy with 2.38x higher latency. This

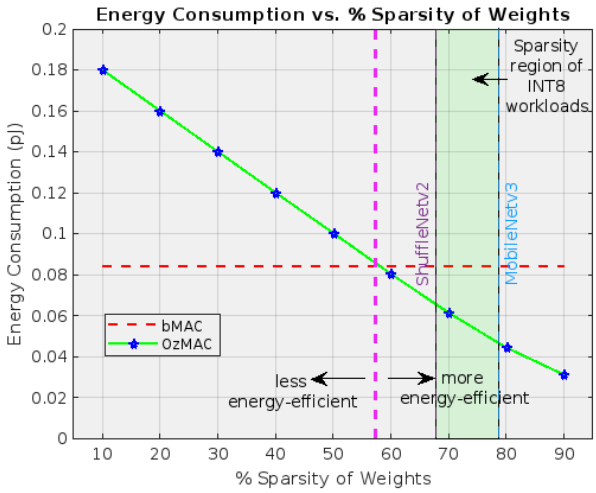

Fig. 7. Energy consumption vs. % bit-sparsity. Green-shaded region depicts the sparsity regions for INT8 workloads from Table I

significant improvement in all three metrics can be attributed to three key factors: 1) simpler shift-and-add hardware with less area and leakage power footprint, 2) serial *Oz*-encoder that enables significant reduction in signal transitions at the input stage, thereby improving dynamic power, and 3) capability to exploit the high dynamic bit sparsity present in the DNN benchmarks (from Table I). Skipping over "0" bits drastically reduces the latency overhead and decreases its energy consumption relative to conventional bMAC. For throughput-sensitive applications, the higher latency of *Oz*MAC can be addressed via frequency scaling, as will be demonstrated later in Section VII-B.

Given the power consumption values from Table II, it can be seen that *Oz*MAC reduces power by 3.36x on average. This implies that for an 8-bit *Oz*MAC design, it can incur up to 3.36 clock cycles on latency overhead per MAC operation, before its energy consumption exceeds that of bMAC. We can calculate the minimum bit sparsity needed for *Oz*MAC to maintain superior energy efficiency as $1 - \frac{3.36}{8} = 58\%$. Fig. 7 plots the energy consumption derived from corresponding latencies in Fig. 6 across varying bit sparsity, to demonstrate this cross-over point at 58% sparsity. Interestingly, all eight DNN benchmarks exhibit bit sparsity in the range of 68-79%, higher than the threshold of 58% as can be seen from Fig. 7. Significant reduction in power consumption, coupled with sparsity-induced latency reduction, allows *Oz*MAC to maintain superior energy efficiency over bMAC even accommodating multi-cycle latency overhead.

Key Takeaway: *Oz*MAC achieves significant reduction in area, power and energy relative to bMAC for typical DNN workloads, by exploiting their inherent bit sparsity.

VI. PRECISION SCALING ANALYSIS

Inference precision for DNN workloads has been trending from 16-bits in the past to the current 8-bits with projection of further trending towards just 4-bits. In this section we broaden

TABLE III
TSMC N5 PPA AT 500 MHZ ACROSS VARYING BIT PRECISION OF WEIGHTS AND ACTIVATIONS: 4 BITS, 8 BITS AND 16 BITS

MAC Hardware (wgt x act)	Area (μm^2)	Power (mW)	Latency (ns)	Energy (pJ)
bMAC (4x4)	5.451	0.015	2	0.031
OzMAC (4x4)	4.712	0.008	2.794	0.022
% Improvement	13.6	49.4	-	29.2
bMAC (4x8)	9.693	0.031	2	0.061
OzMAC (4x8)	8.3752	0.013	2.794	0.035
% Improvement	13.6	58.5	-	42.0
bMAC (8x8)	25.361	0.084	2	0.167
OzMAC (8x8)	19.996	0.025	4.76	0.120
% Improvement	21.2	69.7	-	28.0
bMAC (8x16)	45.282	0.177	2	0.355
OzMAC (8x16)	30.909	0.041	4.76	0.196
% Improvement	31.7	76.8	-	44.9
bMAC (16x16)	74.199	0.297	2	0.594
OzMAC (16x16)	60.608	0.065	9.28	0.601
% Improvement	18.3	78.2	-	-1.2

our experiments to examine the scaling of precision across INT4, INT8, and INT16 for *Oz*MAC.

So far we assume both the weight and activation values employ the same precision. Hence both inputs to the MAC unit employ the same precision format. However one can argue that one of the inputs representing converged weight values may not require the same precision as the activation values. Meaning that the precision for weight values can use a lower precision format. For example we can use 4-bits for weights and 8-bits for activations in a mixed-precision MAC design.

In our precision scaling analysis in this section we broaden our MAC configurations to also include two mixed-precision MAC configurations, i.e. 4x8 and 8x16, in addition to the INT4 (4x4), INT8 (8x8), and INT16 (16x16) configurations. We use all five configurations in our precision scaling experiments to assess the trends in PPA trade-offs for *Oz*MAC and bMAC designs for all five precision configurations.

Table III provides TSMC N5 PPA for the following five integer precision configurations: 1) 4-bit weights and 4-bit activations (4x4), 2) 4-bit weights and 8-bit activations (4x8), 3) 8-bit weights and 8-bit activations (8x8), 4) 8-bit weights and 16-bit activations (8x16), and 5) 16-bit weights and 16-bit activations (16x16). The mixed precision configurations, namely, 4x8 and 8x16, are used to accommodate typical workloads that demand higher activation precision compared to weight precision. The corresponding area, power and energy results are also plotted in Fig. 8, Fig. 9 and Fig. 10 respectively.

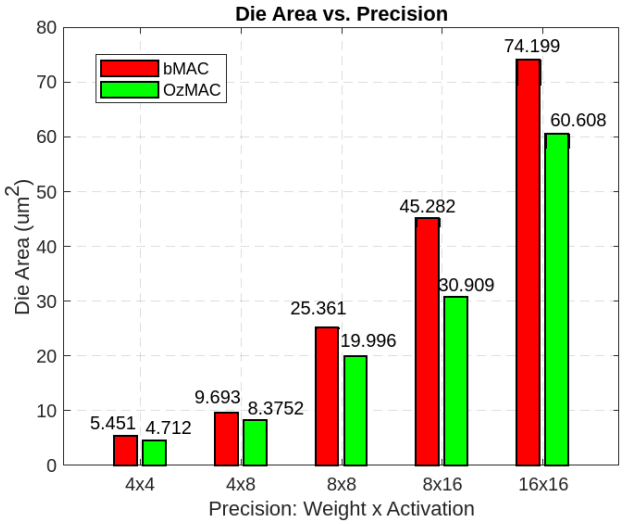

Fig. 8. Die area costs vs precision configurations

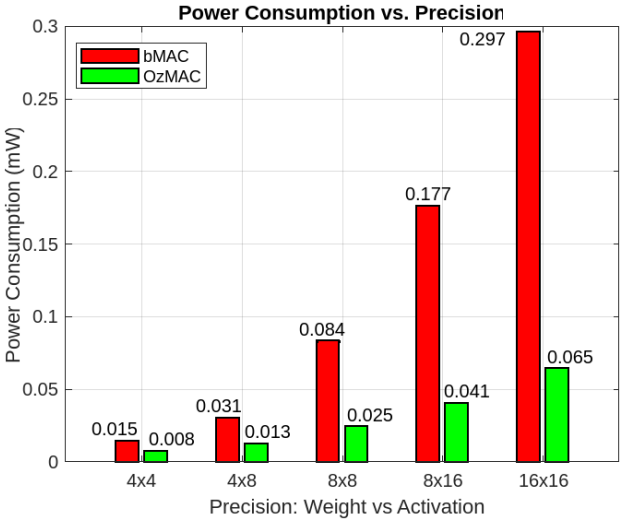

Fig. 9. Power consumption vs precision configurations

Based on Table III, the smallest (4x4) *Oz*MAC and bMAC designs consume 4.7 μm^2 area, 8 μW power, 22 fJ energy, and 5.4 μm^2 area, 15 μW power, 31 fJ energy, respectively. In contrast, the largest (16x16) *Oz*MAC and bMAC designs consume 60.6 μm^2 area, 65 μW power, 601 fJ energy, and 74.2 μm^2 area, 297 μW power, 594 fJ energy, respectively. Compared to 4x4 designs, 16x16 *Oz*MAC incurs about 13x, 8x and 27x increase whereas 16x16 bMAC incurs close to 14x, 20x and 20x increase in area, power and energy, respectively. Both 16x16 designs yield comparable energy even though *Oz*MAC still possess area and power benefits. This indicates going beyond 16-bits precision for *Oz*MAC is not beneficial.

From Fig. 8, area for *Oz*MAC and bMAC scale up in a similar fashion almost linearly with respect to product of weight and activation bits. However, Fig. 9 depicts how *Oz*MAC's power consumption scales much better than that of bMAC which incurs a much sharper increase as precision

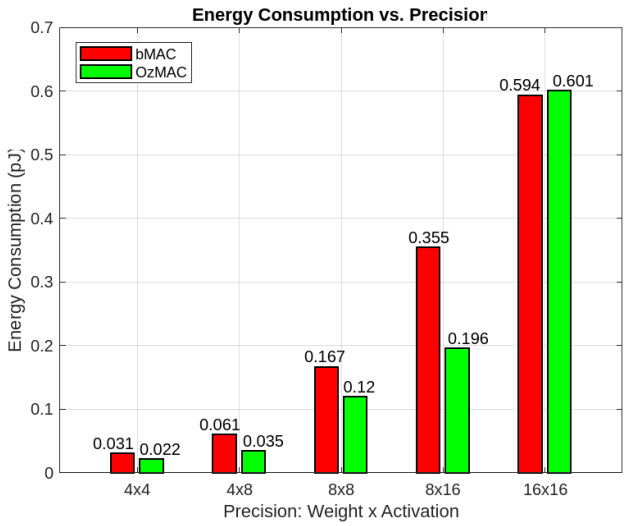

Fig. 10. Energy consumption vs precision configurations

is scaled up. In terms of PPA improvements relative to bMAC, 8x16 *Oz*MAC delivers the most area benefit (32% improvement), whereas the mixed precision 4x8 and 8x16 *Oz*MAC designs offer the highest energy benefit up to 45%. Mixed precision designs deliver the highest relative energy improvements, as they can leverage the lower of the two precisions for *Oz*-encoding, incurring minimum latency (and thereby energy) overhead while taking advantage of the lower hardware complexity of the simpler 16-bit shifter as compared to conventional bMAC multiplier. Power benefits increase monotonically with precision due to the serial nature of *Oz* computation with signal transitions that get relatively sparser with higher precision.

Key Takeaway: *Oz*MAC is more area and power-efficient than bMAC across all precision configurations, and more energy efficient across all but one (16x16) configuration. Energy consumption for both designs evens out at 16-bit weight precision, and would become worse for *Oz*MAC as weight precision is increased beyond 16 bits due to high latency overhead.

VII. Frequency Scaling Analysis

In this section, we perform two types of frequency scaling evaluation to assess the effect of scaling frequency on PPA trends between *Oz*MAC and bMAC: 1) iso-frequency comparison where both designs are compared at the same clock frequency, for three different clock frequencies (0.5 GHz, 1 GHz and 1.5 GHz), and 2) iso-latency comparison where bMAC at 0.5 GHz is compared against *Oz*MAC at a higher frequency where the compute latency (throughput) of both designs are the same.

A. Iso-Frequency Evaluation

As can be seen from Table IV, *Oz*MAC consumes 50 μW power and 118 fJ energy at 1 GHz, and only 75 μW and 119 fJ even at 1.5 GHz. On the other hand, the corresponding values for bMAC are 166 μW and 166 fJ at 1 GHz, and 251 μW and 167 fJ at 1.5 GHz. As expected, power consumption scales linearly with frequency and energy stays almost constant since power increases and latency reduces (due to clock period) by similar amounts. At all three frequencies, *Oz*MAC improves power and energy by almost 70% and 29% respectively.

TABLE IV
TSMC N5 PPA for INT8 (8-bits) OzMAC across varying frequencies: 500 MHz, 1 GHz and 1.5 GHz

MAC Hardware	Power (mW)	Latency (ns)	Energy (pJ)
bMAC (0.5 GHz)	0.084	2	0.167
OzMAC (0.5 GHz)	0.025	4.76	0.120
% Improvement	69.7	-	28.0
bMAC (1 GHz)	0.166	1	0.166
OzMAC (1 GHz)	0.050	2.38	0.118
% Improvement	70.1	-	28.7
bMAC (1.5 GHz)	0.251	0.667	0.167
OzMAC (1.5 GHz)	0.075	1.587	0.119
% Improvement	70.2	-	29.0

TABLE V
TSMC N5 PPA for OzMAC and bMAC across varying bit precisions at throughput-matching frequencies

MAC Hardware (wgt x act)	Freq GHz	Power (mW)	Latency (ns)	Energy (pJ)
bMAC (4x4)	0.5	0.015	2	0.031
OzMAC (4x4)	0.7	0.011	2	0.022
% Improvement	-	29.2	Equal	29.3
bMAC (4x8)	0.5	0.031	2	0.061
OzMAC (4x8)	0.7	0.018	2	0.036
% Improvement	-	41.5	Equal	41.6
bMAC (8x8)	0.5	0.084	2	0.167
OzMAC (8x8)	1.2	0.059	2	0.118
% Improvement	-	29.5	Equal	29.6
bMAC (8x16)	0.5	0.177	2	0.355
OzMAC (8x16)	1.2	0.096	2	0.192
% Improvement	-	46.0	Equal	46.0

B. Iso-Latency Evaluation

So far this work has demonstrated *Oz*MAC's potential for significant area, power and energy improvements over conven-

tional bMAC at varying bitwidth precision configurations and varying clock frequencies. However, it is to be noted that these improvements are achieved at the cost of increased latency (1.4x for 4 bits and 2.4x for 8 bits). These *Oz*MAC designs are ideal for edge inference applications that can tolerate the slight increase in latency (and reduction in throughput) but with stringent constraints on area, power, and energy budgets. In this section, we demonstrate that *Oz*MAC can be used effectively even for higher throughput by scaling up its clocking frequency.

To bridge the latency gap between *Oz*MAC and bMAC, we can scale *Oz*MAC's frequency by the corresponding ratio to match bMAC's compute latency and throughput. Table V provides post-synthesis PPA results on TSMC N5 process node for bMAC and *Oz*MAC at throughput-matching frequencies for 4x4, 4x8, 8x8 and 8x16 configurations. The latency gap for these four designs are reasonably low enough to be closed via frequency scaling. The 16x16 configuration incurs 4.6x higher latency and hence is not considered here. The 4x4 and 4x8 configurations can achieve same latency if *Oz*MAC's frequency is increased by 1.4x from 0.5 GHz to 0.7 GHz. Similarly, the frequency for 8x8 and 8x16 *Oz*MAC configurations are increased by 2.4x from 0.5 GHz to 1.2 GHz. Note that in all cases, bMAC is kept at 0.5 GHz. As can be seen from Table V, *Oz*MAC can deliver substantial improvements in power consumption in all cases, even while running at a higher frequency, with no effect on energy efficiency.

INT4 (4x4) and INT8 (8x8) configurations deliver close to 30% improvement in power/energy, while the mixed precision configurations (4x8 and 8x16) provide even higher improvements in power/energy by up to 46%. Note that *Oz*MAC can potentially deliver even higher throughput than bMAC by leveraging the remaining headroom in power reduction (29% to 46%) to further increase the frequency.

Key Takeaway: *Oz*MAC maintains superiority in area, power and energy efficiency at frequencies ranging from 500 MHz to 1.5 GHz, and can leverage relative frequency scaling to achieve equal or lower latency (hence higher throughput) compared to bMAC without adversely affecting its power or energy efficiency.

VIII. Conclusions and Future Works

This paper presents a novel MAC unit design targeting deep learning workloads with low precision inference computation. The proposed design, termed *Oz*MAC, computes MAC operation via a series of simple shift-and-add operations, which only account for the '1's in input binary value. This skipping of zeros (or omitting zeros) is a key trait of *Oz*MAC that enables it to leverage bit sparsity present in DNN workloads. Smaller the number of ones in a binary value, larger its bit sparsity. Existing works typically leverage word sparsity arising from zero values (all bits '0'), rather than bit sparsity. *Oz*MAC can exploit both forms of value sparsity.

Using eight state-of-the-art DNN benchmarks, we demonstrate the presence of high bit sparsity in real DNN workloads, underscoring *Oz*MAC's inherent capability to exploit dynamic bit sparsity. We implement a wide range of *Oz*MAC designs using commercial design tools and the current TSMC N5 PDK (process design kit) for the 5nm process node, and obtain the power-performance-area (PPA) results for the *Oz*MAC design with scaling of data precisions (4-bits, 8-bits, 16-bits) and clock frequencies (500 MHz, 1 GHz, 1.5 GHz). The *Oz*MAC designs show substantial improvements in all three metrics: area (up to 30%), power (up to 80%) and energy (up to 46%) relative to conventional binary bMAC. Finally, we demonstrate the significant power reduction of *Oz*MAC units and how this can be leveraged to increase throughput by increasing frequency without compromising the die area and energy efficiency benefits.

We believe all deep learning accelerators (DLAs) targeting low precision (8-bits) inference on edge devices should adopt the *Oz*MAC design for implementing their large arrays of MAC units for inclusion in their core GEMM hardware.

There are several promising follow-on tasks for future works. First is to evaluate a large array of *Oz*MAC units in an actual DLA to assess the effectiveness of *Oz*MAC at the system level. This will require dealing with synchronization, data buffering, and (potentially asynchronous) clocking of the *Oz*MAC units in a large array of such processing elements (PEs). Next is to evaluate more real-world DNN workloads at the system level, accounting for data access from/to the memory subsystem. Finally, after extensive system-level and application evaluations, the overarching goal is to do a proof-of-concept prototype design of an *Oz*MAC based DLA for inclusion in a mobile System-on-Chip (SoC) targeting DNN workloads for on-device edge inference.

References

[1] "Quantization," https://pytorch.org/docs/stable/quantization.html

[2] "Quantized models," https://pytorch.org/vision/stable/models.html#quantized-models

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[4] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, 2019.

[5] Y.-H. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, 2017.

[6] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[7] C. De Sa, M. Leszczynski, J. Zhang, A. Marzoev, C. R. Aberger, K. Olukotun, and C. Ré, "High-accuracy low-precision training," *arXiv preprint arXiv:1803.03383*, 2018.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[9] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 257–260.

[10] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," *arXiv preprint arXiv:1902.09574*, 2019.

[11] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International conference on machine learning*. PMLR, 2015, pp. 1737–1746.

[12] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, 2015.

[13] A. Inci, M. M. Isgenc, and D. Marculescu, "Deepnvm++: Cross-layer modeling and optimization framework of non-volatile memories for deep learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

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

[15] 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.

[16] B. Li, L. Song, F. Chen, X. Qian, Y. Chen, and H. H. Li, "Reram-based accelerator for deep learning," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 815–820.

[17] Z. Liu, A. Ali, P. Kenesei, A. Miceli, H. Sharma, N. Schwarz, D. Trujillo, H. Yoo, R. Coffee, N. Layad *et al.*, "Bridge data center ai systems with edge computing for actionable information retrieval," *arXiv preprint arXiv:2105.13967*, 2021.

[18] N. Mellempudi, A. Kundu, D. Das, D. Mudigere, and B. Kaul, "Mixed low-precision deep learning inference using dynamic fixed point," *arXiv preprint arXiv:1701.08978*, 2017.

[19] S. Y. H. Mirmahaleh, M. Reshadi, and N. Bagherzadeh, "Flow mapping on mesh-based deep learning accelerator," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 80–97, 2020.

[20] D. Miyashita, S. Kousai, T. Suzuki, and J. Deguchi, "A neuromorphic chip optimized for deep learning and cmos technology with time-domain analog and digital mixed-signal processing," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 10, pp. 2679–2689, 2017.

[21] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *arXiv preprint arXiv:1603.01025*, 2016.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[23] M. A. Raihan and T. Aamodt, "Sparse weight activation training," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[24] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Survey of machine learning accelerators," in *2020 IEEE high performance extreme computing conference (HPEC)*. IEEE, 2020, pp. 1–12.

[25] 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)*, 2018.

[26] X. Sun, N. Wang, C.-Y. Chen, J. Ni, A. Agrawal, X. Cui, S. Venkataramani, K. El Maghraoui, V. V. Srinivasan, and K. Gopalakrishnan, "Ultra-low precision 4-bit training of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1796–1807, 2020.

[27] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[28] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks," *Synthesis Lectures on Computer Architecture*, vol. 15, no. 2, pp. 1–341, 2020.

[29] S.-N. Tang and Y.-S. Han, "A high-accuracy hardware-efficient multiply–accumulate (mac) unit based on dual-mode truncation error compensation for cnns," *IEEE Access*, vol. 8, pp. 214 716–214 731, 2020.

[30] H. Vanholder, "Efficient inference with tensorrt," in *GPU Technology Conference*, vol. 1, 2016, p. 2.

[31] N. Wang, J. Choi, and K. Gopalakrishnan, "8-bit precision for training deep learning systems," 2018.

[32] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[33] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," *arXiv preprint arXiv:1802.04680*, 2018.

[34] H. Zhang, J. He, and S.-B. Ko, "Efficient posit multiply-accumulate unit generator for deep learning applications," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[35] Z. Zou, Y. Jin, P. Nevalainen, Y. Huan, J. Heikkonen, and T. Westerlund, "Edge and fog computing enabled ai for iot-an overview," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2019, pp. 51–56.