Mixed-Signal Charge-Domain Acceleration of Deep Neural Networks through Interleaved Bit-Partitioned Arithmetic

Soroush Ghodrati Hardik Sharma* Sean Kinzer Amir Yazdanbakhsh†
Jongse Park‡ Nam Sung Kim§ Doug Burger♭ Hadi Esmaeilzadeh

Alternative **C**omputing **T**echnologies (**ACT**) Lab
University of California, San Diego

*Bigstream, Inc. †Google Research ‡KAIST §University of Illinois, Urbana-Champaign ♭Microsoft

{soghodra,skinzer}@eng.ucsd.edu hardik@bigstream.co ayazdan@google.com
jspark@casys.kaist.ac.kr nskim@illinois.edu dburger@microsoft.com hadi@eng.ucsd.edu

ABSTRACT

Albeit low-power, mixed-signal circuitry suffers from significant overhead of Analog to Digital (A/D) conversion, limited range for information encoding, and susceptibility to noise. This paper aims to address these challenges by offering and leveraging the following mathematical insight regarding vector dot-product—the basic operator in Deep Neural Networks (DNNs). This operator can be reformulated as a wide regrouping of spatially parallel low-bitwidth calculations that are interleaved across the bit partitions of multiple elements of the vectors. As such, the computational building block of our accelerator becomes a wide bit-interleaved analog vector unit comprising a collection of low-bitwidth multiply-accumulate modules that operate in the analog domain and share a single A/D converter (ADC). This bit-partitioning results in a lower-resolution ADC while the wide regrouping alleviates the need for A/D conversion per operation, amortizing its cost across multiple bit-partitions of the vector elements. Moreover, the low-bitwidth modules require smaller encoding range and also provide larger margins for noise mitigation. We also utilize the switched-capacitor design for our bit-level reformulation of DNN operations. The proposed switched-capacitor circuitry performs the regrouped multiplications in the charge domain and accumulates the results of the group in its capacitors over multiple cycles. The capacitive accumulation combined with wide bit-partitioned regrouping reduces the rate of A/D conversions, further improving the overall efficiency of the design.

With such mathematical reformulation and its switched-capacitor implementation, we define one possible 3D-stacked microarchitecture, dubbed BiHiwe[1], that leverages clustering and hierarchical design to best utilize power-efficiency of the mixed-signal domain and 3D stacking. We also build models for noise, computational non-idealities, and variations. For ten DNN benchmarks, BiHiwe delivers 5.5×speedup over a leading purely-digital 3D-stacked accelerator Tetris, with a mere of less than 0.5% accuracy loss achieved by careful treatment of noise, computation error, and various forms of variation. Compared to RTX 2080 TI with tensor cores and Titan Xp GPUs, all with 8-bit execution, BiHiwe offers 35.4×and 70.1×higher Performance-per-Watt, respectively. Relative to the mixed-signal RedEye, ISAAC, and PipeLayer, BiHiwe offers 5.5×, 3.6×, and 9.6× improvement in Performance-per-Watt respectively. The results suggest that BiHiwe is an effective initial step in a road that combines mathematics, circuits, and architecture.

[1] **BiHiwe**: **Bi**t-Partitioned and **I**nterleaved **Hi**erachy of **W**ide Acceleration through **E**lectrical Charge

PACT '20, October 3–7, 2020, Virtual Event, GA, USA

ACM ISBN 978-1-4503-8075-1/20/10.
https://doi.org/10.1145/3410463.3414634

CCS CONCEPTS

• **Computer systems organization → Analog computers**; **Neural networks**; **Special purpose systems**.

KEYWORDS

Accelerators; Deep Neural Networks; DNN; DNN Acceleration; Analog/Mixed-Signal Computing; Mixed-Signal Acceleration; Bit-Partitioning; Spatial Bit-Level Regrouping; Analog Error Modeling

ACM Reference Format:

Soroush Ghodrati, Hardik Sharma, Sean Kinzer, Amir Yazdanbakhsh, Jongse Park, Nam Sung Kim, Doug Burger, and Hadi Esmaeilzadeh. 2020. Mixed-Signal Charge-Domain Acceleration of Deep Neural Networks through Interleaved Bit-Partitioned Arithmetic. In *Proceedings of the 2020 International Conference on Parallel Architectures and Compilation Techniques (PACT'20), Oct. 3–7, 2020, Virtual Event, GA, USA.* ACM, NY, NY, USA, 13 pages.
https://doi.org/10.1145/3410463.3414634

1 INTRODUCTION

With the diminishing benefits from general-purpose processors [1–4], there is an explosion of digital accelerators for DNNs [5–26]. Mixed-signal acceleration [27–37] is also gaining traction. Albeit low-power, mixed-signal circuitry suffers from limited range of information encoding, is susceptible to noise, lacks fine-grained control mechanism and imposes significant overheads for Analog to Digital (A/D) conversions. As a point of reference, for an 8-bit×8-bit MACC which produces a 16-bit output at 500 Mhz, A/D conversion costs about 1,000× higher energy than the MACC itself at 45 nm. In addition, encoding 256 levels for 8-bit inputs in less than 1 Volt allocates 3.9 mV for each level, significantly restricting both the representation capabilities as well as the noise margins. This paper sets out to

address these challenges by inspecting the mathematical foundation of deep neural networks and makes the following contributions.

(1) This work offers and leverages the insight that the set of MACC operations within one vector dot-product can be partitioned, interleaved, and regrouped at the *bit level* without affecting the mathematical integrity of dot-product. Unlike prior work [28, 37, 38], this work does not rely on changing the mathematics of the computation to enable mixed-signal acceleration. Namely, PRIME [38] leverages memristive technology to enable analog computation, but relies on several truncations during computations of intermediate data to overcome the overheads of A/D conversions. In contrast, this work only rearranges the bit-wise arithmetic calculations across multiple elements of the vectors to utilize a group of lower bitwidth analog units for higher bitwidth operations. The key insight is that a binary value can be expressed as the sum of products similar to dot-product, which is also a sum of multiplications ($a = \vec{X} \bullet \vec{W} = \sum_i x_i \times w_i$). Each x_i or w_i can be expressed as $\sum_j (2^j \times b_j)$ where b_js are the individual bits or as $\sum_j (2^{4j} \times bp_j)$, where bp_js are 4-bit partitions for instance. Our interleaved arithmetic utilizes the *distributive and associative property* of multiplication and addition at the *bit granularity* for partitioning and regrouping.

The proposed model, first, bit-partitions all elements of the two vectors, and then *distributes* the MACC operations of the dot-product over these bit partitions. Then, our mathematical formulation exploits the *associative property* of the multiply and add to group and colocate bit-partitions that are at the same significance position. This significance-based rearrangement enables factoring out the power-of-two multiplicand that signifies the position of the bit-partitions. The factoring enables regrouping the partial results from a set of lower-bitwidth MACCs as one spatially parallel operation in the analog domain, while the group shares a *single* A/D converter (ADC). The power-of-two multiplicand will be applied later digitally to the accumulated result of the group operation. To this end, we reformulate vector dot-product as a wide regrouping of interleaved and bit-partitioned operations across multiple elements of the two vectors (see section 2). This spatial regrouping of operations and parallel execution is in contrast with prior analog-based accelerators such as PRIME [38] and ISAAC [27], which although use bit-partitioning but perform MACC operations serially over multiple cycles on bit (partitions) of the operands or RedEye [30] that does not exploit any sort of bit-partitioning.

The bit-partitioning lowers the resolution of ADCs while the wide regrouping amortizes the cost of each A/D conversion across multiple bit-partitions of the vector elements. Using low-bitwidth operands for analog MACCs also provides a larger headroom between the value encoding levels in the analog domain. The headroom tackles the limited range of encoding and offers more robustness to noise, an inherent non-ideality in the analog mode.

(2) At the circuit level, the accelerator is designed using switched-capacitor circuitry that stores the partial results as electric charge over time without conversion to the digital domain at each cycle. The low-bitwidth MACCs are performed in charge domain with a set of charge-sharing capacitors. This design choice lowers the rate of A/D conversion as it implements accumulation as a gradual storage of charge in a set of parallel capacitors. These capacitors not only aggregate the result of a group of low-bitwidth MACCs, but also enable accumulating results over time. As such, the architecture enables dividing the longer vectors into shorter sub-vectors that are multiply-accumulated over time with a single group of spatially parallel low-bitwidth MACCs. The results are accumulated over multiple cycles in the group's capacitors. Because the capacitors can hold the charge from cycle to cycle, the A/D conversion is not necessary in each cycle. This reduction in rate of A/D conversion is in addition to the amortized cost of ADCs across the analog low-bitwidth MACCs of the group (see section 3).

(3) We take a systematic approach and perform a step-by-step analysis to evaluate the contribution of each technique in tackling the challenges of mixed-signal design and maximizing its benefits. This analysis shows that Interleaved Bit-Partitioning is the most effective technique. On one hand, Spatially Wide Regrouping is the second most effective technique in improving area efficiency of the arithmetic operations that enables integrating more in a given area, improving design parallelism. The benefit stems from sharing a single ADC across the groupings of the low-bitwidth analog MACC units, amortizing its area. On the other hand, Charge-Domain Computation ranks second in improving the power efficiency that is the fruit of reducing the rate of the A/D conversions, through accumulation and storage of the intermediate results in capacitors.

With these insights, we devise a hierarchical 3D-stacked instance of the microarchitecture, named BiHiwe, that leverages the proposed arithmetic and building blocks, yet offers programmability and domain generality. Evaluating this carefully balanced design of BiHiwe with a diverse set of ten DNN benchmarks shows that BiHiwe delivers 5.5×speedup over the purely digital 3D-stacked DNN accelerator, Tetris [7], with only 0.5% loss in accuracy achieved after mitigating noise, computation error, and Process-Voltage-Temperature (PVT) variations. With 8-bit execution, BiHiwe offers 35.4×and 70.1×higher Performance-per-Watt compared to RTX 2080 TI and Titan Xp, respectively. Compared to the mixed-signal CMOS RedEye [30], memristive ISAAC [27] and PipeLayer [39], BiHiwe delivers 5.5×, 3.6×, and 9.6× higher Performance-per-Watt, respectively. With these benefits, this paper marks an initial effort to use mathematical insights for devising mixed-signal DNN accelerators.

2 WIDE, INTERLEAVED, AND BIT-PARTITIONED ARITHMETIC

A key idea of this work is the mathematical insight that enables utilizing low bitwidth mixed-signal units in spatially parallel groups.

Bit-Level partitioning and interleaving of MACCs. To further detail the proposed mathematical reformulation, Figure 1(a) delves into the bit-level operations of dot-product on vectors with 2-elements containing 4-bit values. As illustrated with different colors, each 4-bit element can be written in the form of sum of 2-bit partitions multiplied by powers of 2 (shift). As discussed, vector dot-product is also a sum of multiplications. Therefore, by utilizing the distributive property of addition and multiplication, we can rewrite the vector dot-product in terms of the bit partitions. However, we also leverage the associativity of the addition and multiplication to regroup the bit-partitions that are in the same positions, together. For instance, in Figure 1, the black partitions that represent the Most Significant Bits (MSBs) of the $\vec{W}$ vector are multiplied in parallel to the teal[2] partitions, representing the MSBs of the $\vec{X}$. Because of the distributivity of multiplication, the shift amount of (2+2) can be postponed after

[2] Color teal in Figure 1 is the darkest gray in black and white prints.

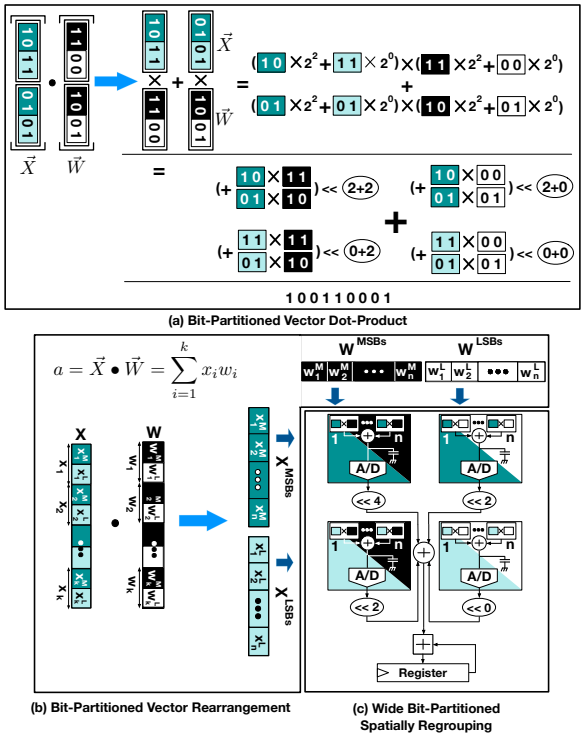


Figure 1: Wide, interleaved, and bit-partitioned mathematical formulation.

the bit-partitions are multiply-accumulated. The different colors of the boxes in Figure 1 illustrates the interleaved regrouping of the bit-partitions. Each group is a set of spatially parallel bit-partitioned MACC operations that are drawn from different elements of the two vectors. The low-bitwidth nature of these operations enables execution in the analog domain without the need for A/D conversion for each individual bit-partitioned operation. As such, our proposed reformulation amortizes the cost of A/D conversion across the bit-partitions of different elements of the vectors as elaborated below.

Wide, interleaved, and bit-partitioned vector dot-product. Figure 1(b) illustrates the proposed vector dot-product operation with 4-bit elements that are bit partitioned to 2-bit sub-elements. For instance, as illustrated, the elements of vector X, denoted as x_i, are first bit partitioned to x_i^L and x_i^M. The former represents the two Least Significant Bits (LSBs) and the latter represents the Most Significant Bits (MSBs). Similarly, the elements of vector W are also bit partitioned to the w_i^L and w_i^M sub-elements. Then, each vector (e.g., W) is rearranged into two bit-partitioned sub-vectors, W^{LSBs} and W^{MSBs}. In the current implementations of BitHiwe architecture, the size of bit-partitioning is fixed across the entire architecture. Therefore, the rearrangement is just rewiring the bits to the compute units that imposes modestly minimal overhead (less than 1%). Figure 1 is merely an illustration and there is no need for extra storage or movement of elements. As depicted with color coding, after the rewiring, W^{LSBs} represents all the least significant bit-partitions from different elements of vector W, while the MSBs are rewired in W^{MSBs}. The same rewiring is repeated for the vector X. This rearrangement, puts all the bit-partitions from all the elements of the vectors with the same significance in one group, denoted as W^{LSBs}, W^{MSBs}, X^{LSBs}, X^{MSBs}. Therefore, when a pair of the groups (e.g., X^{MSBs} and W^{MSBs} in Figure 1(c)) are multiplied to generate the partial products, (1) the shift amount ("$\ll 4$" in this case) is the same for all the bit-partitions and (2) the shift can be done after partial products from different sub-elements are accumulated together.

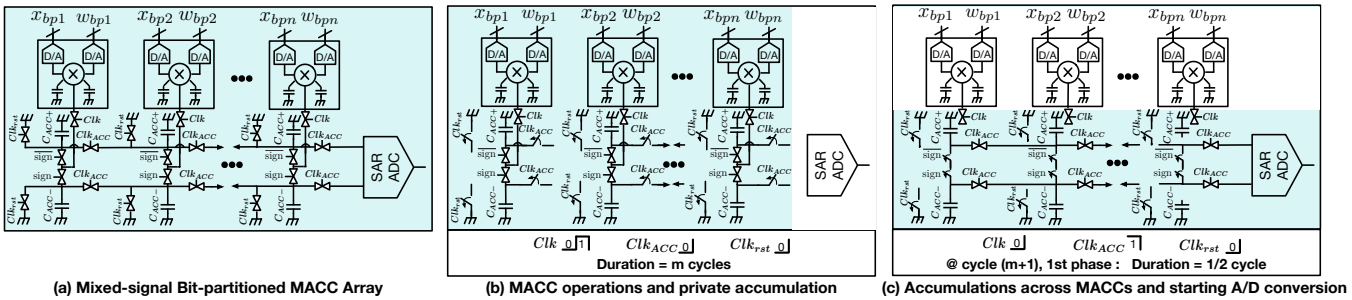
As shown in Figure 1(c), the low-bitwidth elements are multiplied together and accumulated in the analog domain. Accumulation in the digital domain would require an adder tree which is costly compared to the analog accumulation that merely requires connectivity between the multiplier outputs. It is only after several analog multiply-accumulations that the results are converted to digital for shift and aggregation with partial products from the other groups. This is not only because of spatially wide grouping of the low-bitwidth MACC operations, but also, as will be discussed in the next section, due to the accumulation of the partial results in the analog domain by storing electric charge in capacitors before ADCs (see Figure 1(c)). If the size of vectors exceeds the predefined value of (size of spatially low-bitwidth array) × (number of capacitive accumulation cycles), these converted partial results will be added up in the digital domain using a register. For this pattern of computation, we are effectively utilizing the *distributive and associative property* of multiplication and addition for dot-product but at the *bit granularity*. This rearrangement and spatially parallel (i.e., wide) bit-partitioned computation is in contrast with temporally bit-serial digital [8, 12, 26, 40] and analog [27] DNN accelerators.

3 SWITCHED-CAPACITOR DESIGN FOR INTERLEAVED BIT-PARTITIONING

To exploit the aforementioned arithmetic, an analog vector unit needs to be designed. This building block is a collection of low-bitwidth analog MACCs that operate in parallel on sub-elements from the two vectors under dot-product. This wide structure is dubbed Mixed-Signal Bit-Partitioned MACC Array ($\mathcal{MS}$-BPMacc). Within the $\mathcal{MS}$-BPMacc, we design the low-bitwidth MACC units using switched-capacitor circuitry [29, 31, 36, 37, 41], implementing the MACC operations in the charge-domain rather than using resistive-ladders to compute in current domain [27, 35, 38]. Compared to the current-domain approach, switched-capacitors (1) enable result accumulation in the analog domain by storing them as electric charge, eliminating the need for A/D conversion at every cycle, and (2) make the relative ratio of capacitors the determining factor in analog multiplication. Dependence to ratio and not the absolute sizes makes the design more resilient to process variation.

3.1 Mixed-Signal Bit-Partitioned MACC Array

Figure 2(a) depicts an array of n low-bitwidth MACCs, constituting the $\mathcal{MS}$-BPMacc unit, which perform operations for m cycles in the analog domain. Each low-bitwidth MACC unit receives a pair of bit-partitions (x_{bpi}, w_{bpi}) from the sub-vectors. These bit-partitions are fed to Digital to Analog (D/A) converters to enable charge-domain MACC operations. Low-bitwidth MACC units are equipped with their own pair of accumulating capacitors (C_{ACC+}, C_{ACC-}), which perform the accumulation over time across multiple sub-vectors. The pair is used to handle positive and negative values by accumulating them separately on one or the other capacitor. Figure 2(b) illustrates the MACC computation mode of the $\mathcal{MS}$-BPMacc unit. Over m cycles, each low-bitwidth MACC unit works separately and accumulates the partial results privately on its own pair of C_{ACC}s. To enable



(a) Mixed-signal Bit-partitioned MACC Array (b) MACC operations and private accumulation (c) Accumulations across MACCs and starting A/D conversion

Figure 2: $\mathcal{MS}$-BPMacc and its operational modes.

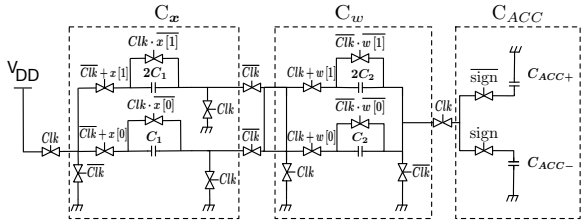
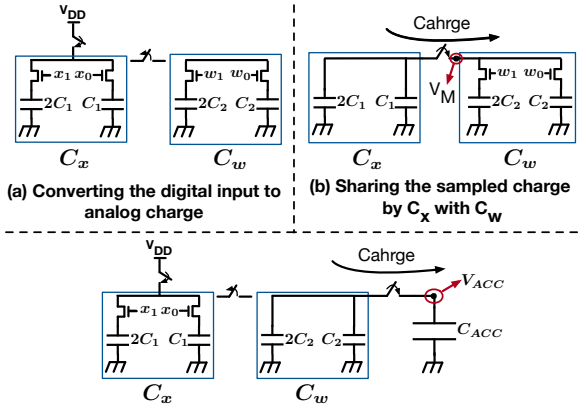


Figure 3: Low-bitwidth switched-capacitor MACC.



(a) Converting the digital input to analog charge (b) Sharing the sampled charge by C_x with C_w

(c) The sampled charge by C_w is transferred to C_{ACC} while a new input is sampled by C_x

Figure 4: Charge-domain MACC; phase by phase.

the private accumulation mode the transmission gates between different MACC units are all disconnected (shown with open switches) and only the capacitors' private transmission gates are connected. Aggregation across the multiple low-btiwidth MACCs happens in the first half of cycle m+1, shown in Figure 2(c) . In this half cycle, the private results get aggregated across all n MACC units within the $\mathcal{MS}$-BPMacc. The transmission gates between the capacitors connect them and a simple charge sharing between the capacitors yields the aggregated result of $m \times n$ number of multiply-adds. Clk_{ACC} is the control signal which connects the C_{ACC}s. This aggregation happens for both positive and negative values (across $C_{\text{ACC+}}$ s and $C_{\text{ACC-}}$ s respectively) at the same time. The *single* ADC in the $\mathcal{MS}$-BPMacc is responsible for converting the aggregated result, which also starts at the first stage of cycle m+1. The accumulating capacitors (C_{ACC}s), are connected to a Successive Approximation Register (SAR) ADC and share their stored charge with the Sample and Hold block (S&H) of the ADC. This (S&H) block has differential inputs which samples the positive and negative results separately and holds them for the process of A/D conversion. In the second phase of cycle m+1 all the C_{ACC}s get disconnected from the ADC and Clk_{rst} connects them to the ground to clear their charge for the next iteration of wide, bit-interleaved calculations. There is a trade-off between resolution and sample rate of ADC, which also defines its topology. For instance, Flash ADCs are suitable for high sample rate but low resolution designs. SAR ADC is a better choice when it comes to medium resolution (8-12 bits) and sample rate (1-500 Mega-Samples/sec). We choose a 10-bit, 15 Mega-Samples/sec SAR ADC [42] as it strikes the best balance between rate and resolution for $\mathcal{MS}$-BPMaccs based on design space exploration shown in Figure 17. The process of A/D conversion takes m+1 cycles, pipelined with vector dot-products.

The $\mathcal{MS}$-BPMacc computes the low-bitwidth MACC operations in charge-domain as the following discusses.

3.2 Low-Bitwidth Switched-Capacitor MACC

Figure 3 depicts the design of a single 3-bit sign-magnitude MACC. The $x_s x_1 x_0$ and $w_s w_1 w_0$ denote the bit-partitions operands. The result of each MACC operation is retained as electric charge in the accumulating capacitor (C_{ACC}). In addition to C_{ACC}, the MACC unit contains two capacitive Digital-to-Analog Converters (DACs), one for inputs (C_{X}) and one for weights (C_{W}). The C_{X} and C_{W} convert the 2-bit magnitude of the input and weight to the analog domain as an electric charge proportional to $|x|$ and $|w|$ respectively. C_{X} and C_{W} are each composed of two capacitors ($(C_1, 2C_1)$ for C_{X} and $(C_2, 2C_2)$ for C_{W}) which operate in parallel and are combined to convert the operands to analog domain. Each of these capacitors are controlled by a pair of transmission gates which determine if a capacitor is active or inactive. Another set of transmission gates connect the two D/A converters and share charge when partitions of x and w are multiplied. The resulting shared charge is stored on either $C_{\text{ACC+}}$ or $C_{\text{ACC-}}$ depending on the sign control signal produced by $x_s \oplus w_s$. During multiplication, the transmission gates are coordinated by a pair of complimentary non-overlapping clock signals, Clk and $\overline{Clk}$.

Charge-domain MACC. Figure 4 shows the phase-by-phase process of a MACC, the phases of which are described below.

$Clk_{\phi(1)}$**:** The first phase (Figure 4(a)) consists C_{X} converting digital input (x) to a charge proportional to its magnitude. Since, the sampled charge (Q_{sx}) by C_{X} in the first phase is equal to:

$$Q_{sx} = v_{DD} \times (|X| C_1) \tag{1}$$

$\overline{Clk}_{\phi(2)}$**:** In the second phase (Figure 4(b)), the multiplication happens via a charge-sharing process between C_{X} and C_{W}. The 2-bit partition of the weight is applied to C_{W} and sets its equivalent capacitance to $|w|C_2$. At the same time, the C_{X} redistributes its sampled charge (Q_{sx}) over all of its capacitors ($3 \times C_1$) as well as the equivalent capacitor of C_{W}. The voltage (V_M) at the junction of C_{X} and C_{W} is as follows:

$$V_M = \frac{Q_{sx}}{C_{tot}} = \frac{v_{DD} \times (|X|C_1)}{3C_1 + |w|C_2} \quad (2)$$

Because the sampled charge is shared with the weight capacitors, the stored charge (Q_{sw}) on C_W is equal to:

$$Q_{sw} = V_M \times |w|C_2 = |x| \times |w| \left(\frac{C_2 C_1 v_{DD}}{3C_1 + |w|C_2} \right) \quad (3)$$

Equation 3 shows that Q_{sw} is proportional to $|x| \times |w|$, but includes a non-linearity term in the denominator ($|w|$). To mitigate that C_1 must be much larger than C_2. Further mitigation is considered as discussed in Section 6. With this choice, Q_{sw} becomes $|x| \times |w| \frac{C_2 v_{DD}}{3}$.

$Clk_{\phi(3)}$: In the last phase, (Figure 4(c)), the charge from multiplication is shared with C_{ACC} for accumulation. The sign bits (x_s and w_s) determine which of C_{ACC+} or C_{ACC-} is selected for accumulation. The sampled charge by $|w|C_2$ is then redistributed over the selected C_{ACC} as well as all the capacitors of C_W ($=3C_2$). Theoretically, C_{ACC} must be infinitely larger than $3C_2$ to completely absorb the charge from multiplication. However, in reality, some charge remains unabsorbed, leading to a pattern of computational error, which is mitigated as discussed in Section 6. Ideally, the V_{ACC} voltage on C_{ACC} is:

$$V_{ACC} = |x||w| \left(\frac{C_2 v_{DD}}{3 \times C_{ACC}} \right) \quad (4)$$

While the charge sharing and accumulation happens on C_{ACC}, a new input is fed into C_x, starting a new MACC process in a pipelined fashion. This process repeats for all low-bitwidth MACC units over multiple cycles before one A/D conversion.

4 MIXED-SIGNAL ARCHITECTURE DESIGN FOR SPATIAL BIT-PARTITIONING

Last section provided the detailed innards of low-bitwidth MACCs and how they can be used to construct a *low-bitwidth* spatially interleaved dot-product unit (MS-BPMACC). This section, focuses on architecting a *higher bitwidth* dot-product engine, called MS-WAGG, from a collection of MS-BPMACCs. This engine is named MS-WAGG as it is a Mixed- Signal Wide Aggregator that operates on bit-partitioned vectors in SIMD fashion. Instead of just describing the design, we take a quantitive journey that step-by-step highlights how much each design decision contributes to improving the power and area efficiency. Finally, we elaborate on how to utilize this engine to construct a full-fledged programmable mixed-signal DNN accelerator.

4.1 Mixed-Signal Wide Aggregator

To better understand the tradeoffs in designing MS-WAGG, we contrast it with a basic mixed-signal dot-product engine, called MS-BASIC, that does not utilize bit-partitioning (see Figure 5). Consequently, the D/A converters in Figure 5 are stained with two different shades of a color to highlight that all of the different bit-partitions of each operand are kept together. Each analog multiplier receives all operands'bits and converts the multiplication result to digital domain to go through an adder tree. Mixed-signal DNN accelerators are essentially an optimized transformation of this basic engine. Here, we discuss how much each of our innovations contributes to the design transformation that yields MS-WAGG. Figure 6 illustrates a possible MS-WAGG design, comprising 16 MS-BPMACCs, necessary to perform 8-bit by 8-bit vector dot-product with 2-bit partitioning[3]. In contrast to the MS-BASIC, each D/A converter in Figure 6 is colored with one shade to show each input is just a bit-partition. In

[3] 2-bit partitioning is the optimal choice (design space exploration in Figure 15).

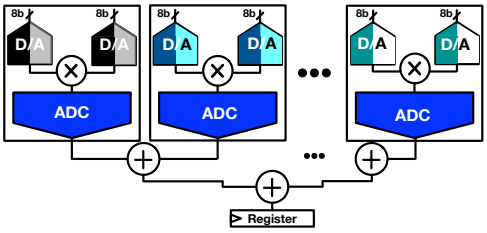


Figure 5: Basic mixed-signal dot-product engine.

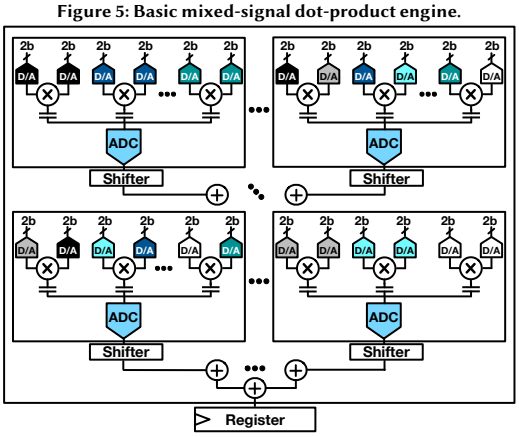


Figure 6: Our mixed-signal dot-product engine (MS-WAGG).

this case, the number of MS-BPMACCs, 16 (=*four*×*four*), comes from the fact that each of the two 8-bit operands can be partitioned to *four* 2-bit values. Each of the *four* 2-bit partitions of the multiplicand need to be multiply-accumulated with all the multiplier's *four* 2-bit partitions. As discussed in Section 2, each MS-WAGG also performs the necessary shift operations to combine the low-bitwidth results from its 16 MS-BPMACCs. By aggregating the partial results of each MS-BPMACC in the digital domain, the MS-WAGG engine generates a scalar which is stored on its output register.

4.2 MS-WAGG Design Decisions and Tradeoffs

The design of MS-WAGG stems from three main techniques: (1) Interleaved Bit-Partitioning, (2) Spatially Wide Regrouping, and (3) Charge-Domain Computation. For all the analyses in this section, 500 Mhz frequency at 45 nm is used to design an 8-bit vector dot-product engine. Figure 7(a) and (b) illustrates the contribution of each technique in power and area improvement, respectively. Improving area efficiency has a direct effect on performance as it enables integrating more compute units in a given area, improving design parallelism. The pie charts show how much of the total power/area is consumed by each of hardware components: analog multiplication, digital shift-and-add logic, register, ADC. D/A conversion is part of the analog multiplier as discussed in Section 3. The size of the pie is pictorially reduced to show that the total power/area is decreasing. The first pie chart belongs to MS-BASIC–merely a point of reference–that performs an 8-bit×8-bit MACC in the analog domain and converts the 16-bit result to digital, while the last chart is of MS-WAGG. The following discusses each technique and its effects on power/area efficiency.

(1) Interleaved Bit-Partitioning is the most effective technique in improving both power and area of the mixed-signal dot-product engines. This technique partitions each operand to lower bitwidth

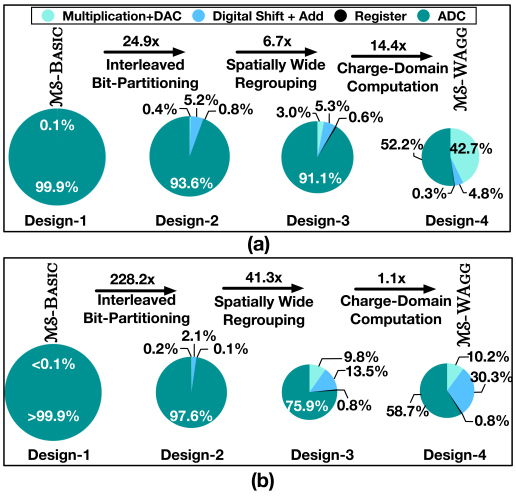


Figure 7: Step-by-step analysis of improvement in (a) power and (b) area.

suboperands, and then interleaves the bit-partitions. Interleaved Bit-Partitioning enables replacing the 8-bit×8-bit MACC and its high resolution (16-bit) ADC in MS-Basic with 16 2-bit×2-bit MACCs and significantly lower resolution (4-bit) ADCs . By applying this technique, the power and area for an 8-bit MACC operation improves by 24.9× and 228.2×, respectively (Comparing Design-2 with Design-1 in Figure 7). This improvement stems from the fact that power and area of ADC increases dramatically with its resolution.

(2) Spatially Wide Regrouping is the second technique that regroups a wide array of interleaved lower bitwidth MACC units to share a single ADC. The outputs of lower-bitiwidth MACC units is aggregated in the analog domain, the result of which is fed to the ADC. This technique ranks second in improving area efficiency (41.3× when comparing Design-3 with Design-2 in Figure 7(b)). Sharing a single ADC across a wide group of lower-bitwidth MACC units, reduces the effective number of ADCs, leading to lower area. This sharing increases 4-bit resolution of the ADCs to 7-bits (sharing an ADC with 8 2-bit×2-bit MACCs) as more number of low-bitwidth MACC operations are aggregated in the analog domain before conversion; however, this increase in the ADC's power/area is sub-exponential. The benefit comes from the fact that Spatially Wide Regrouping enables shifting the ADC design style from Flash to Pipelined or SAR in the same frequency. Exploiting this technique also improves the power efficiency by 6.7×.

(3) Charge-Domain Computation is the second most effective technique in improving power-efficiency. Accumulating the partial results as electric charge in capacitors eliminates the necessity of A/D conversion at each cycle, leading to significant power reduction. This additional accumulation in the analog domain requires higher resolution ADCs (10-bits); however, the reduced rate of the A/D conversion trumps the resolution increase. This technique yields an additional 14.4× improvement in power efficiency (Design-4 vs Design-3 in Figure 7(a)). The number of the ADC remains the same but the reduced rate enables choosing an ADC with lower sample rate. Lower sample rate ADCs require lower-area subcomponents that can reduce its overall area. However, the increase in resolution counteracts this benefit to a large degree. As such, this technique only reduces the area by 1.1× (Design-4 compared to Design-3 in Figure 7(b)).

4.3 Hierarchically Clustered Architecture

As illustrated in Figure 8, a collection of MS-WAggs constitute an accelerator core from which the clustered architecture of BiHiwe is designed. The three aforementioned optimization techniques, results in 5.4× less energy for a single 8-bit MACC in comparison with a digital logic. Hence, it is possible to integrate a larger number of mixed-signal compute units in a given power budget compared to a digital architecture. To efficiently utilize this increase in compute units, a high bandwidth memory substrate is required. To maximize the benefits of the mixed-signal computation, 3D-stacked memory is an attractive option since it reduces the energy cost of data accesses and provides a higher bandwidth for data transfer between the on-chip compute and off-chip memory [7, 20]. Based on these insights, we devise a clustered architecture for BiHiwe with a 3D-stacked memory substrate as shown in Figure 8. As the results in Section 7.2 Figure 16 shows, a flat design would result in significant underutilization of the compute resources and bandwidth from 3D stacking. Therefore, BiHiwe is a hierarchically clustered architecture that allocates multiple accelerator cores as a cluster to each vault (Figure 8(a)). Figure 8(b) depicts a single core. As shown in Figure 8(b), each core is self-sufficient and packs a mixed-signal systolic array of MS-WAggs as well as the digital Pooling Unit, Activation Unit, and Normalization Unit, etc. The mixed-signal array is responsible for the convolutional and fully connected layers. Generally, wide and interleaved bit-partitioned execution within MS-WAggs is orthogonal to the organization of the accelerator architecture. This paper explores how to embed them and the proposed compute model, within a systolic design and enables end-to-end programmable mixed-signal acceleration for a variety of DNNs.

Accelerator core. As Figure 8(b) depicts, the first level of hierarchy is the accelerator core and its 2D systolic array that utilizes the MS-WAggs. As depicted, the Input Buffers and Output Buffers are shared across the columns and rows, respectively. Each MS-WAgg has its own Weight Buffer. This organization is commensurate with other designs and reduces the cost of on-chip data accesses as inputs are reused with multiple filters [21]. However, what makes our design different is the fact that each buffer needs to supply a sub-vector not a scalar in each cycle to MS-WAggs. The rewiring of the inputs and weights is already done inside the MS-WAggs since the size of bit-partitions is fixed. Consequently, there is no need to reformat any of inputs, activations, or weights. To preserve the accuracy of

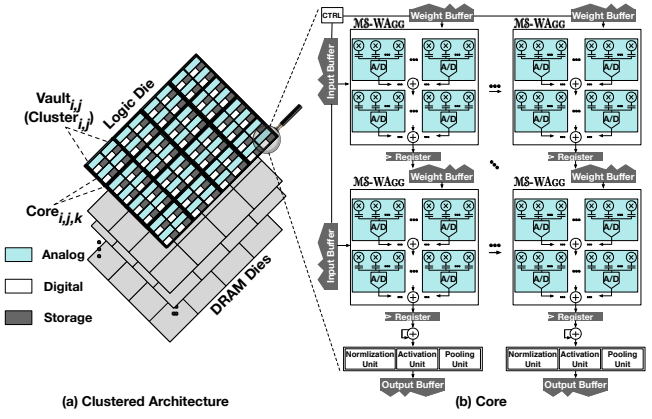


Figure 8: Hierarchical clustered architecture

the DNNs, intermediate results are stored as 32-bit digital values and intra-column aggregations are performed in digital mode.

On-chip data delivery for accelerator cores. To minimize data movement and exploit the abundant data-reuse in DNNs, BiHiwe uses a statically-scheduled interconnect that is capable of multicasting/broadcasting data across accelerator cores. Static scheduling enables the BiHiwe compiler stack to do exhaustive search over variegated possibilities of cutting and tiling DNN layers across cores to maximizing inter- and intra-core data-reuse. The static schedule is encoded in the form of data communication instructions.

Parallelizing computations across accelerator cores. To minimize data movement, the BiHiwe clustered architecture (1) divides the computations into tiles that fit within the on-chip capacity of the scratchpads, and (2) *cuts* the tiles of computations across cores to minimize DRAM accesses by maximally utilizing the multicast/broadcast capabilities of BiHiwe on-chip data delivery network. To simplify the hardware, scratchpad buffers are private to each core and the shared data is replicated across multiple cores. Thus, a single *tile* of data can be read once from the memory and then be broadcasted/multicasted across cores to reduce DRAM accesses. The cores use double-buffering to hide the latency for memory accesses for subsequent tiles. Cores use *output-stationary* dataflow that minimizes the number of A/D conversions by accumulating results in the charge-domain. Section 5 discusses the cutting/tiling optimizations in compiler.

4.4 BiHiwe Instruction Set

The BiHiwe ISA provides a layer of abstraction that exposes the following unique properties of its architecture to the compiler (1) mixed-signal execution within a BiHiwe core; and (2) data-movement for both 3D-stacked memory and on-chip software-managed scratchpads between different BiHiwe cores. As such, BiHiwe uses a block-structured ISA where the blocks have repetition counters due to the tile-based execution and segregates the execution of the DNN into (1) data communication instruction blocks that transfer tiles of data between 3D-stacked memory and on-chip scratchpads (Input Buffer-/Weight Buffer/Output Buffer in Figure 8) using address generation instructions, and (2) compute instruction blocks that consumes the tile of data from a communication instruction block to produce an output tile. The communication block and compute block together specify a static schedule for DNN execution in BiHiwe.

Using the compute instruction block, the compiler has complete control over on-chip scratchpads, A/D conversion rate, and bit-partitioning across $\mathcal{MS}$-WAggs. These pieces of information are encoded in the header of the compute instruction blocks. The granularity of bit-partitioning and charge-based accumulation is determined for each microarchitectural implementation based on technology node and circuit design style. As such, to support different technology nodes and designs and allow extensions to the architecture, the BiHiwe ISA encodes the bit-partitioning and accumulation cycles. Using the communication instruction blocks, the compiler stack exploits the broadcasting/multicasting capabilities to optimize data movement while maximizing data locality for the on-chip scratchpad memories in each core.

5 BIHIWE COMPILER STACK

Figure 9 illustrates the BiHiwe compiler stack that accepts a high-level Caffe2 [43] specification of the DNN to generate an instruction

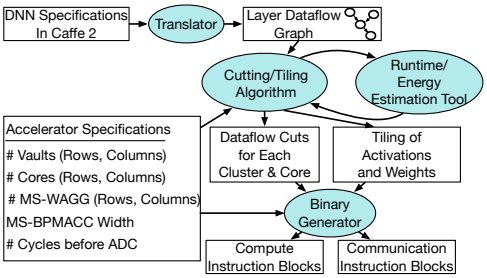


Figure 9: BiHiwe compilation stack.

Initialize $cut_{opt}[N] \leftarrow \emptyset$; Initialize $tiling_{opt}[N] \leftarrow \emptyset$
for $layer_i \in DFG_{DNN}$ **do**
  $s_{opt} \leftarrow \infty$
  for $tiling_{i,j} \in layer_i$ **do**
    for $cut_{i,j,k} \in tiling_{i,j}$ **do**
      $(\text{runtime}_{i,j,k}, \text{energy}_{i,j,k}) \leftarrow \text{EstimTool}(tiling_{i,j}, cut_{i,j,k})$
      $s_{i,j,k} \leftarrow \text{runtime}_{i,j,k} \times \text{energy}_{i,j,k}$
      if $s_{i,j,k} < s_{opt}$ **then**
        $cut_{opt}[i] \leftarrow cut_{i,j,k}$; $tiling_{opt}[i] \leftarrow tiling_{i,j}$
return $cut_{opt}, tiling_{opt}$

Algorithm 1: Cutting/tiling algorithm for clustered acceleration.

binary (BiHiwe ISA). The first step in the compiler stack is a translation of the Caffe2 file into a layer DataFlow Graph (DFG) that preserves the structure of the DNN. The BiHiwe compiler stack also accepts a specification of the accelerator configuration that includes the organizations and configurations (# rows, #columns) of the clusters, vaults, and cores as well as details of the $\mathcal{MS}$-BPMAccs. Using the layer DFG and the accelerator configuration, the compiler then proceeds with an optimization step that determines the optimal cut of the DFG nodes across BiHiwe clusters and cores, and optimal tile sizes for the multidimensional arrays (DFG edges) to fit into the limited on-chip memory. For each node in the layer DFG, the optimization algorithm performs an exhaustive search of different cuts and tile sizes for incoming and outgoing edges. For each candidate cut and tile-size, the compiler stack uses an analytical estimation tool that determines the total energy consumption and runtime. Estimation is viable, as the DFG does not change, there is no hardware managed cache, and the accelerator architecture is fixed during execution. Thus, there are no irregularities that can hinder estimation. Algorithm 1 depicts the cutting/tiling procedure. When cuts and tiles are determined, the compiler generates the binary code that contains the communication and computation instruction blocks in BiHiwe ISA.

6 MITIGATING ANALOG NON-IDEALITIES

Although analog circuitry offers significant reduction in energy, they might lead to accuracy degradation. Thus, their error needs to be properly modeled and accounted for. Specifically, $\mathcal{MS}$-BPMAccs, the main analog component, can be susceptible to (1) thermal noise, (2) computational error caused by incomplete charge transfer, and (3) PVT variations. Traditionally, analog circuit designers mitigate sources of error by just configuring hardware parameters to values which are robust to non-idealities. Such hardware parameter adjustments require rather significant energy/area overheads that scale linearly with number of modules. However, due to the scaled-up nature of our design, we need to mitigate these non-idealities in a higher and algorithmic level. We leverage the training algorithm's inherent mechanism to reduce error (loss) and use mathematical

models to represent these non-idealities. We, then, apply these models during forward pass to *adjust and fine-tune pre-trained neural models with just a few more epochs* across the chips within a technology node. Our approach is commensurate with recent work [44] that uses fine-tuning passes to incorporate analog non-idealities. The rest of this section details non-idealities and their modeling.

Thermal noise. Thermal noise is an inherent perturbation in analog circuits caused by the thermal agitation of electrons. This noise can be modeled according to a normal distribution, where the ideal voltage deviates relative to a value comprised of the working temperature (T), Boltzmann constant (k), and capacitor size (C) which produce the deviation $\sigma = \sqrt{kT/C}$. Within BiHiwe, switched-capacitor MACC units are mainly effected by the combined thermal noise resulting from weights and accumulator capacitors (C_W and C_{ACC} respectively). The noise from these capacitors gets accumulated during the m cycles of computation for each individual MACC unit and then gets aggregated across the n MACC units in $\mathcal{MS}$-BPMacc. By applying the thermal noise equation used for similar MACC units [37] to a $\mathcal{MS}$-BPMacc unit, the standard deviation at the output is described by Equation 5:

$$\sigma_{ACC} = \sqrt{\frac{kT(\alpha|W_{m-1}|+3\alpha+3)}{9\alpha(\alpha+1)^2 C_w}\left(\sum_{i=0}^{m-1}\left(\frac{\alpha}{1+\alpha}\right)^{2i}\right)\times n} \quad (5)$$

In the above equation, α is equal to $\frac{C_{ACC}}{3C_W}$. We add error tensors to outputs of convolutional/fully connected layers in DNN forward propagation, to incorporate thermal noise effect. Elements of error tensors are sampled from a normal distribution as $\mathcal{N}(\mu = 0, \sigma^2 = (\sigma_{ACC}\times r\times 85)^2)$. σ_{ACC} is scaled by r, the amount of $\mathcal{MS}$-BPMacc operations required to generate an element in the output feature map, as well as the amount of total bit-shifts applied to each result by $\mathcal{MS}$-WAgg engine, 85.

Computational error. Another source of error in BiHiwe's computations arises when charge is shared between capacitors during the multiplication and accumulation. Within each MACC unit, the input capacitors (C_X) transfer a sampled charge to the weight capacitors (C_W) to produce charge proportional to the multiplication result. But the resulting charge is subject to error dependent on the ratio of weight and input capacitor sizes ($\beta = C_1/C_2$) as shown in Equation 3. This shared charge in the weight capacitors introduces more error when it is redistributed to the accumulating capacitor (C_{ACC}) which cannot absorb all of the charge, leaving a small portion remaining on the weight capacitors in subsequent cycles. The ideal voltage ($V_{ACC,Ideal}$) produced after m cycles of multiplication can be derived from Equation 4 as follows:

$$V_{ACC,Ideal}[m] = \sum_{i=1}^{m} \frac{V_{DD}}{9\alpha} W_i X_i \quad (6)$$

By considering the computational error from incomplete charge sharing, the actual voltage at the accumulating capacitor after m cycles of MACC operations ($V_{ACC,R}[m]$) becomes:

$$\frac{3\alpha}{3\alpha+|W_m|} V_{ACC,R}[m-1] + \frac{W_m X_m \beta}{(3\alpha+|W_m|)(3\beta+|W_m|)} V_{DD} \quad (7)$$

We consider computational error in the fine-tuning pass by including the multiplicative factors shown in Equation 7 in weights. During the forward pass, the fine-tuning algorithm decomposes weight tensors in convolutional/fully-connected layers into groups corresponding to $\mathcal{MS}$-WAgg configuration and updates the individual weight values (W_i) to new values (W_i') with the computational error:

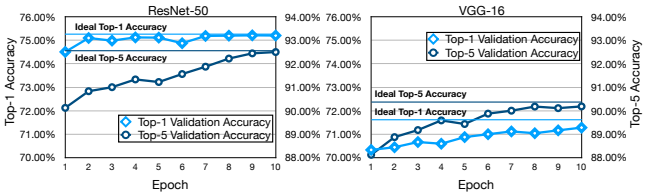


Figure 10: ResNet-50 and VGG-16 accuracy after fine-tuning.

Table 1: Evaluated benchmarked DNNs

DNN	Type	Domain	Dataset	Multiply-Adds	Model Weights
AlexNet [48]	CNN	Image Classification	Imagenet [49]	2,678 MOps	56.1 MBytes
CIFAR-10 [50, 51]	CNN	Image Classification	CIFAR-10 [52]	617 MOps	13.4 MBytes
GoogLeNet [53]	CNN	Image Classification	Imagenet	1,502 MOps	13.5 MBytes
ResNet-18 [54]	CNN	Image Classification	Imagenet	4,269 MOps	11.1 MBytes
ResNet-50 [54]	CNN	Image Classification	Imagenet	8,030 MOps	24.4 MBytes
VGG-16 [50]	CNN	Image Classification	Imagenet	31 GOps	131.6 MBytes
VGG-19 [50]	CNN	Image Classification	Imagenet	39 GOps	137.3 MBytes
YOLOv3 [55]	CNN	Object Recognition	Imagenet	19 GOps	39.8 MBytes
PTB-RNN [51]	RNN	Language Modeling	Penn TreeBank [56]	17 MOps	16 MBytes
PTB-LSTM [57]	RNN	Language Modeling	Penn TreeBank	13 MOps	12.3 MBytes

$$W_i' = \frac{W_i}{3\alpha+|W_i|}\frac{\beta V_{DD}}{3\beta+|W_i|}\prod_{j=i+1}^{m-1}\frac{3\alpha}{3\alpha+|W_j|} \quad (8)$$
$$\forall 0 \le i \le m-1$$

Process variations. We use the sizing of the capacitors to provision and mitigate for the process variations to which the switched-capacitor circuits are generally robust. This is effective because the capacitors are implemented using a number of smaller unit capacitors with common-centroid layout technique [45]. We, specifically, use the metal-fringe capacitors for MACCs with mismatch of just 1% standard deviation [46] with the max variation of 6% (6σ) which is well below the error margins considered for the computational error.

Temperature variations. This is modeled by adding a perturbation term to T in Equation 5 as a gaussian distribution $\mathcal{N}_T(\mu,\sigma^2)$. We consider the maximum value of the temperature as 358°K, commensurate with existing practices [47], and the minimum value as 300°K (This is the peak-to-peak range for the gaussian distribution (6σ)).

Voltage variations. We also model the voltage variation by adding a gaussian distribution to V_{DD} term in Equation 8. Our experiments show that, variations in voltage can be mitigated up to 20%. The extensive amount of vector dot-product operations in DNNs, allows for the minimum and maximum values of the distributions being sampled sufficient amount of times, leading to coverage of the corner cases.

Atop all these considerations, we use differential signaling for ADCs which attenuates the common-mode fluctuations such as PVT variations. To show the effectiveness of our techniques, Figure 10 plots the result of fine-tuning process of two benchmarks, ResNet-50 and VGG-16 for ten epochs. Table 3 reports the summary of accuracy trends for all the benchmarks, which achieve less than 0.5% loss. As Figure 10 shows, the fine-tuning pass compensates the initial loss (0.73% for top-1 and 2.41% for top-5) to only 0.04% for top-1 and 0.02% for top-5. VGG-16 is slightly different and reduces the initial loss (1.16% for top-1 and 2.24% for top-5) to less than 0.18% for top-1 and 0.13% for top-5 validation accuracy. The trends are similar for other benchmarks and omitted due to space constraints.

7 EVALUATION

7.1 Methodology

Benchmarks. We use ten diverse CNN/RNN models including real-time object recognition and word-level language modeling, described

Table 2: BiHiwe and baselines platforms

Parameters	ASIC		Parameters	GPU	
Chip	BiHiwe	Tetris	Chip	RTX 2080 TI	Titan Xp
MACCs	16,384	3,136	Tensor Cores	544	–
On-chip Memory	9216 KB	3698 KB	Memory	11 GB (GDDR6)	12 GB (GDDR5X)
Chip Area (mm^2)	122.3	56	Chip Area (mm^2)	754	471
			Total Dissipation Power	250 W	250 W
Frequency	500 Mhz	500 Mhz	Frequency	1545 Mhz	1531 Mhz
Technology	45 nm	45 nm	Technology	12 nm	16 nm

in Table 1. These benchmarks includes medium to large scale models and variety of multiply-add operations.

Simulation infrastructure. We develop a cycle-accurate simulator and a compiler for BiHiwe. The simulator dumps the statistics of runtime and accesses to all components and calculates the power. Since, all the instructions are statically scheduled, the simulator can calculate the exact number of accesses to components.

Iso-power and iso-area comparison with Tetris. We match the on-chip power of BiHiwe and Tetris and compare the total runtime and energy, including DRAM accesses. Tetris supports 16-bit execution while BiHiwe supports 8-bit. For fairness, we modify the open-source Tetris simulator [58] and proportionally scale its runtime/energy. BiHiwe supports 8-bit since this representation has virtually no impact by itself on the accuracy of the DNNs [51, 59–62].

Comparison with analog/digital accelerators. We also compare BiHiwe to mixed-signal RedEye [30], two analog memristive accelerators [27, 63], and Google TPU [21], all in 8-bits. The original designs [27, 63] use 16-bits. We optimistically increase the efficiency of the competitor designs by 4× to model 8-bit execution.

GPU comparison. We also compare BiHiwe to two Nvidia GPUs, RTX 2080 TI with tensor cores and Titan Xp (Table 2). For a fair comparison, we use 8-bit on GPUs using Nvidia's TensorRT 5.1 [64] library compiled with the optimized cuDNN 7.5 and CUDA 10.1.

Energy and area measurement. All hardware modelings are performed using FreePDK 45-nm standard cell library [65]. We implement the switched-capacitor MACCs in Cadence Analog Design Environment V6.1.3 and use Spectre SPICE V6.1.3 to model the system. We then, use Layout XL of Cadence to extract the energy/area. The energy/area for ADCs are obtained from [66]. We implement digital blocks of BiHiwe, including adders, shifters, and interconnection in Verilog RTL and use Synopsys Design Compiler (L-2016.03-SP5) for synthesis and measuring energy/area. We use CACTI-P [67] to model on-chip buffers. 3D-stacked DRAM is based on HMC [68, 69], same as Tetris, and the bandwidth and access energy are adopted form that work.

Error modeling. We use Spectre SPICE V6.1.3 to extract noise behavior of MACCs. Thermal noise, computational error, and PVT variations are considered based on details in Section 6. We implement extracted hardware error models and corresponding mathematical modelings using PyTorch v1.0.1 [70] and integrate them into Neural Network Distiller v0.3 framework [71] for a fine-tuning pass over evaluated benchmarks.

7.2 Experimental Results

7.2.1 *Comparison with Tetris.*

Iso-power and iso-area comparisons. Figure 11 shows the performance and energy reduction of BiHiwe over Tetris. On average, BiHiwe delivers a 5.5×speedup over Tetris in iso-power setting. The low power and wide bit-partitioned mixed-signal design of $\mathcal{MS}$-WAggs in BiHiwe enables us to integrate 5.2× more compute units than Tetris in the same power budget. The highest speedup is observed in YOLOv3 and CIFAR-10, where the network topology favors the wide vectorized execution in BiHiwe. The lowest speedup is

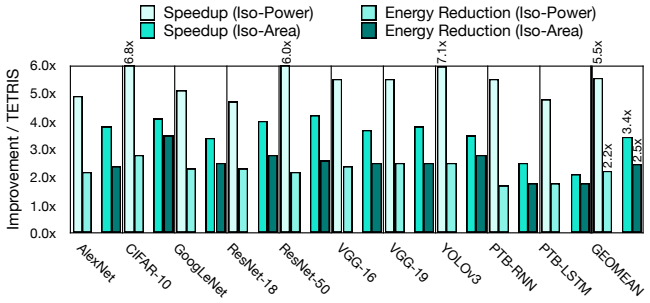


Figure 11: Iso-Power/Iso-Area speedup and energy reduction over Tetris.

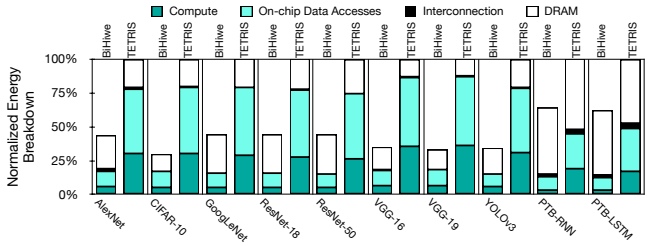


Figure 12: Energy breakdown of BiHiwe and Tetris.

observed in ResNet-18, since its relatively small size leads to underutilization of compute resources in BiHiwe. Figure 11 also demonstrates total energy reduction for BiHiwe as compared to Tetris in iso-power setting. On average, BiHiwe yields 2.2×energy reduction. The lowest energy reduction is observed in RNN benchmarks, PTB-RNN and PTB-LSTM, since matrix-vector operations in RNNs require significant number of DRAM accesses for weights, limiting benefits.

Figure 11 also shows iso-area comparisons. Scaling-up computes in Tetris by 2.25× to match the area of BiHiwe results in ≈ 60% increase in Tetris performance. This improvement in performance comes at a cost of reduced energy-efficiency due to an increase in memory accesses to feed the additional compute units. Trends in speedup and energy-reduction remain the same with the exception of ResNet-18, which now sees resource underutilization in Tetris. Overall, BiHiwe shows 3.4×speedup and 2.5×energy reduction.

Energy breakdown. Figure 12 shows the energy breakdown normalized to Tetris across: (1) on-chip compute units, (2) on-chip memory, (3) interconnect, and (4) 3D-stacked DRAM. DRAM accesses account for the highest portion of the energy in BiHiwe, since BiHiwe significantly reduces the on-chip compute energy. While BiHiwe has a larger number of compute resources compared to Tetris, the number of DRAM accesses remain almost the same. This is because the statically-scheduled interconnect allows data to be multicasted/broadcasted across multiple cores in BiHiwe without significantly increasing the number of DRAM accesses. Unlike the fully-digital PEs in Tetris, BiHiwe uses $\mathcal{MS}$-WAggs which perform wide vectorized operations. Each MACC operation in BiHiwe consumes 5.4× less energy compared to Tetris. The output-stationary dataflow enabled by capacitive accumulation in addition to the systolic organization of $\mathcal{MS}$-WAggs in each core of BiHiwe eliminates the need for register file, leads to 4.4× reduction in on-chip data movement.

7.2.2 *Comparison with Other Baselines.*

Figure 13 depicts power efficiency (GOPS/s/Watt) and area efficiency (GOPS/s/mm^2) of BiHiwe with other recent accelerators. Due to the lack of available raw performance/energy numbers for specific

DNNs and the fact that the simulation/compilation infrastructures for prior accelerators are not open sourced, we use these metrics that is commensurate with comparisons for recent designs [16, 63, 72, 73] to provide a best effort analysis. On average for the evaluated benchmarks, BiHiwe achieves 81% of its peak efficiency.

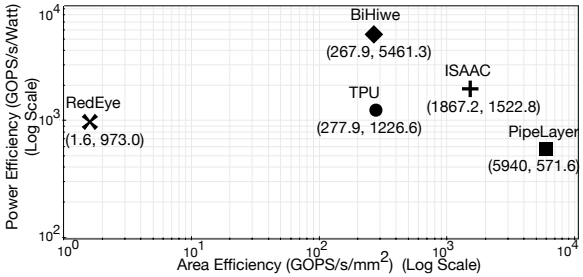


Figure 13: Comparison with other accelerators.

Mixed-signal CMOS: RedEye [30]. RedEye also uses switched-capacitor circuitry. Compared to RedEye, BiHiwe offers 5.5× power efficiency and 167× area efficiency. In contrast to RedEye [30] which does not exploit any sort of bit-partitioning, the proposed wide, interleaved, and bit-partitioned arithmetic amortizes the cost of ADCs in BiHiwe and yields these benefits.

Analog Memristive designs [27, 63]. Prior work in ISAAC and PipeLayer have explored memristive technology for DNN acceleration, which integrates both compute and storage in the same die, offering higher compute density compared to traditional CMOS. Generally, memristive designs perform computations in current domain, requiring costly ADCs to sample currents at high rates, curtailing the power-efficiency. Overall, compared to ISAAC and PipeLayer, BiHiwe improves the power efficiency by 3.6× and 9.6×, respectively.

Google TPU [21]. Compared to TPU, which also uses systolic design, BiHiwe delivers 4.5× more peak power efficiency and almost the same area efficiency. Leveraging the wide, interleaved, and bit-partitioned arithmetic with its switched-capacitor design in BiHiwe, reduces the cost of MACC operations significantly.

Comparison with GPUs. Figure 14 compares performance of BiHiwe with Titan Xp and RTX 2080 TI, normalized to Titan Xp. BiHiwe, on average, yields 1.9× speedup over Titan Xp and is just 5% slower than RTX 2080 TI. CNNs require abundant matrix-matrix multiplications, well-suited for tensor cores, leading to RTX 2080 TI's outperformance on both BiHiwe and Titan Xp. However, BiHiwe outperforms RTX 2080 TI in PTB-RNN and PTB-LSTM with 11.2× and 11.4×, respectively. RNNs require matrix-vector multiplications–particularly suitable for the wide vectorized operations supported in $\mathcal{MS}$-WAggs. However, BiHiwe outperforms both Titan Xp and RTX 2080 TI GPUs in Performance-per-Watt by large margins of 70.1×and 35.4×, respectively.

7.2.3 *Design Space Explorations.*

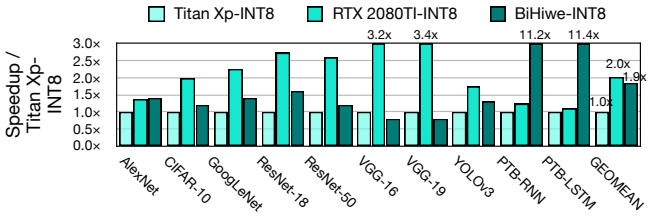


Figure 14: Performance comparison to GPUs.

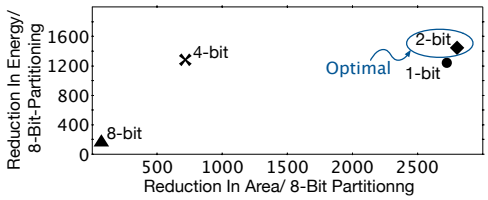


Figure 15: Design space exploration for bit-partitioning.

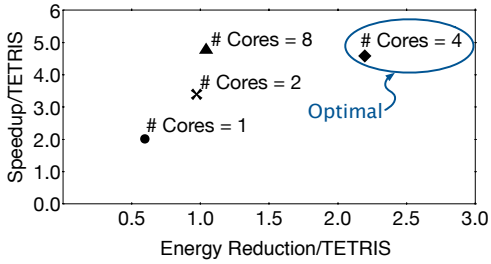


Figure 16: Design space exploration for # core per cluster.

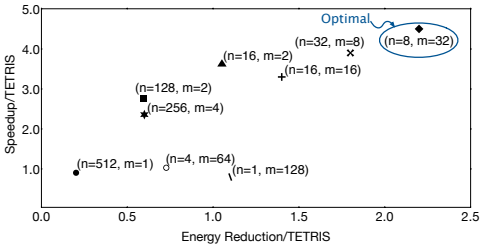


Figure 17: Design space exploration for $\mathcal{MS}$-BPMacc.

Design space exploration for bit-partitioning. Figure 15 shows the reduction in energy and area for different bit-partitioning design points that are algorithmically identical and perform the same 8-bit×8-bit vector dot-product with 32 elements. However, the baseline design uses 8-bit×8-bit MACC units while the rest use our wide and interleaved bit-partitioned arithmetic. As depicted, 2-bit partitioning strikes the best balance in energy/area with switched-capacitor design of MACC units at 45 nm. Compared to 2-bit, single-bit partitioning quadratically increases the number of low bitwidth MACCs from 16 (2-bit partitioning) to 64 (1-bit partitioning) to support 8-bit operations. This imposes disproportionate overhead that outweighs benefit of decreasing MACC units energy/area.

Design space exploration for clustered architecture. BiHiwe uses a hierarchical architecture with multiple cores in each vault. Having a larger number of small cores for each vault yields increased utilization of compute resources, but requires data transfer across cores and replication. We explore the design space with 1, 2, 4, and 8 cores per cluster. As Figure16 shows, BiHiwe with four cores per each vault (default configuration) is optimal by striking a better balance between data accesses and compute resource utilization. Configuration with 8-cores results in higher data accesses, hence higher energy.

Design space exploration for $\mathcal{MS}$-BPMacc configuration. The number of accumulation cycles (m) before A/D conversion and the number of MACC units (n) are two main parameters of $\mathcal{MS}$-BPMacc which define ADC resolution and sample rate, determining its power. Figure 17 shows the design space exploration for different configurations of $\mathcal{MS}$-BPMacc. In a fixed power budget for compute units, we measure total runtime/energy of BiHiwe across benchmarks and

Table 3: Accuracy before and after fine-tuning.

DNN Model	Dataset	Top-1 Accuracy (With Non-Idealities)	Top-1 Accuracy (After Fine-Tuning)	Top-1 Accuracy (Ideal)	Final Accuracy Loss
AlexNet	Imagenet	53.12%	56.64%	57.11%	0.47 %
CIFAR-10	CIFAR-10	90.82%	91.01%	91.03%	0.02 %
GoogLeNet	Imagenet	67.15%	68.39%	68.72%	0.33 %
ResNet-18	Imagenet	66.91%	68.96%	68.98%	0.02 %
ResNet-50	Imagenet	74.5%	75.21%	75.25%	0.04 %
VGG-16	Imagenet	70.31%	71.28%	71.46%	0.18 %
VGG-19	Imagenet	73.24%	74.20%	74.52%	0.32 %
YOLOv3	Imagenet	75.92%	77.1%	77.22%	0.21 %
PTB-RNN	Penn TreeBank	1.1 BPC	1.6 BPC	1.1 BPC	0.0 BPC
PTB-LSTM	Penn TreeBank	97 PPW	170 PPW	97 PPW	0.0 PPW

normalize it to those of Tetris. As shown in Figure 17, increasing number of MACCs, limits the number of accumulation cycles and results in high sample rate ADCs. Using high sample-rate ADCs significantly increases power. On the other hand, increasing number of accumulation cycles, limits the number of MACCs, which restricts the number of $\mathcal{MS}$-WAggs that can be integrated under given power budget. Overall, the optimal design point that delivers the best performance and energy constitutes eight MACC units and 32 accumulation cycles.

7.2.4 Evaluation of Circuitry Non-Idealities.

Table 3 shows the Top-1 accuracy With Non-Idealities, After Fine-Tuning, Ideal, and the Final Accuracy Loss. As shown in Table 3 AlexNet and ResNet-18 are more sensitive to the non-idealities, leading to a higher initial accuracy degradation. To recover the accuracy loss, we perform a fine-tuning step for a few epochs. By performing this fine-tuning step, the accuracy loss of the CIFAR-10, ResNet-18, and ResNet-50 networks is fully recovered (loss is less than 0.04%) which within these networks, CIFAR-10 and ResNet-50 are more robust to non-idealities. Accuracy loss for other networks is below 0.5% which within those AlexNet has maximum loss. Both PTB-RNN and PTB-LSTM recover all the loss after fine-tuning. The final results after fine-tuning step show the effectiveness of this approach in recovering the accuracy loss due to the non-idealities pertinent to analog computation.

8 RELATED WORK

There is a large body of work on digital DNN accelerators [5–26, 74–79]. Mixed-signal acceleration has also been explored previously for neural networks [29, 35] and is gaining traction for deep models [27, 28, 30–34, 36, 37]. This paper fundamentally differs from these inspiring efforts as it delves into mathematics of DNN operations, reformulates and defines the interleaved and bit-partitioned arithmetic combined with charge-domain computation to overcome challenges in mixed-signal acceleration. Below, we discuss the most related works.

Switched-capacitor design. Switched-capacitor circuits [41] have a long history, having been mainly used for designing amplifiers [80], ADC/DAC [81] and filters [82]. They have been used even for the previous generation of neural networks [29]. More recently, they have also been used for matrix multiplication [37, 83], which can benefit DNNs. This work takes inspiration from these efforts but differs from them in that it defines and leverages wide, interleaved, and bit-partitioned reformulation of DNN operations. Additionally, it offers a comprehensive architecture to accelerate a wide variety of DNNs.

Programmable mixed-signal accelerators. PROMISE [28] offers a mixed-signal architecture that integrates analog units within the SRAM memory blocks. RedEye[30] is a low-power near-sensor mixed-signal accelerator that uses charge-domain computations. These works do not offer wide interleavings of bit-partitioned basic operations as described in this paper.

Fixed-functional mixed-signal accelerators. They are designed for a specific DNN. Some focus on handwritten digit classification [83, 84] or binarized mixed-signal acceleration of CIFAR-10 images [33]. Another work focuses on spiking neural networks' acceleration [34]. In contrast, our design is programmable and supports interleaved bit-partitioning.

Resistive memory accelerators. There is a large body of work using resistive memory [27, 38, 63, 72, 73, 85–90]. We provided direct comparison to ISAAC [27] and PipeLayer [63]. ISAAC most notably introduces the concept of temporally bit-serial operations, also explored in PRIME [38], and is augmented with spike-based data scheme in PipeLayer. BiHiwe, in contrast, formulates a partitioning that spatially regroups lower-bitwidth MACCs across different vector elements and performs them in-parallel. PRIME does not provide absolute measurements and its simulated baseline is not available for head-to-head comparisons. PRIME also uses multiple truncations that change the mathematics. Conversely, our formulation does not induce truncation or mathematical changes.

Bit-level composable designs. Bit Fusion [23] proposes bit-level dynamic composability to support quantized DNNs. BitBlade [78] and BPVeC [79] extends bit-level reconfigurability to vector-level composability to amortize the energy and area cost of bit-flexibility. In contrast, this work delves into the details of mixed-signal computing, proposes wide, interleaved, and bit-partitioned arithmetic, and combines it with switched-capacitor circuits to enable mixed-signal acceleration.

9 CONCLUSION

This work proposed wide, interleaved, and spatially bit-partitioned arithmetic to overcome key challenges in mixed-signal acceleration of DNNs. This arithmetic enabled rearranging the highly parallel MACC operations in DNNs into wide low-bitwidth efficient mixed-signal computations. Further, we use switched-capacitor circuitry that reduces the rate of ADC by accumulating partial results in the charge domain. The incarnate design, BiHiwe, offers significant benefits over its state-of-the-art analog and digital counterparts.

10 ACKNOWLEDGEMENT

We thank Mojan Javaheripi for insightful discussions and feedbacks. This work was in part supported by generous gifts from Google, Qualcomm, Microsoft, Xilinx as well as the National Science Foundation (NSF) awards CNS#1703812, ECCS#1609823, CCF#1553192, Air Force Office of Scientific Research (AFOSR) Young Investigator Program (YIP) award #FA9550-17-1-0274, National Institute of Health (NIH) award #R01EB028350, and AirForce Research Laboratory (AFRL) and Defense Advanced Research Project Agency (DARPA) under agreement number #FA8650-20-2-7009 and #HR0011-18-C-0020. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of Google, Qualcomm, Microsoft, Xilinx, Samsung, Bigstream, NSF, AFSOR, NIH, AFRL, DARPA or the U.S. Government.

REFERENCES

[1] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6–15, July–Aug. 2011.

[2] Ganesh Venkatesh, Jack Sampson, Nathan Goulding, Saturnino Garcia, Vladyslav Bryksin, Jose Lugo-Martinez, Steven Swanson, and Michael Bedford Taylor. Conservation cores: Reducing the energy of mature computations. In *ASPLOS*, 2010.

[3] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *ISCA*, 2011.

[4] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *FPGA*, 2015.

[5] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Neural acceleration for general-purpose approximate programs. *in Commun. ACM*, 2013.

[6] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. Dadiannao: A machine-learning supercomputer. In *MICRO*, 2014.

[7] Mingyu Gao, Jing Pu, Xuan Yang, Mark Horowitz, and Christos Kozyrakis. Tetris: Scalable and efficient neural network acceleration with 3d memory. In *ASPLOS*, 2017.

[8] Alberto Delmas, Sayeh Sharify, Patrick Judd, and Andreas Moshovos. Tartan: Accelerating fully-connected and convolutional layers in deep learning networks by exploiting numerical precision variability. *arXiv*, 2017.

[9] Divya Mahajan, Jongse Park, Emmanuel Amaro, Hardik Sharma, Amir Yazdanbakhsh, Joon Kim, and Hadi Esmaeilzadeh. TABLA: A unified template-based framework for accelerating statistical machine learning. In *HPCA*, 2016.

[10] Shijin Zhang, Zidong Du, Lei Zhang, Huiying Lan, Shaoli Liu, Ling Li, Qi Guo, Tianshi Chen, and Yunji Chen. Cambricon-x: An accelerator for sparse neural networks. In *MICRO*, 2016.

[11] Jorge Albericio, Patrick Judd, Tayler Hetherington, Tor Aamodt, Natalie Enright Jerger, and Andreas Moshovos. Cnvlutin: ineffectual-neuron-free deep neural network computing. In *ISCA*, 2016.

[12] Patrick Judd, Jorge Albericio, Tayler Hetherington, Tor M Aamodt, and Andreas Moshovos. Stripes: Bit-serial deep neural network computing. In *MICRO*, 2016.

[13] Hardik Sharma, Jongse Park, Divya Mahajan, Emmanuel Amaro, Joon Kim, Chenkai Shao, Asit Misra, and Hadi Esmaeilzadeh. From high-level deep neural models to fpgas. In *MICRO*, 2016.

[14] Eric Chung, Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Adrian Caulfield, Todd Massengil, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Christian Boehn, Oren Firestein, Alessandro Forin, Kang Su Gatlin, Mahdi Ghandi, Stephen Heil, Kyle Holohan, Tamas Juhasz, Ratna Kumar Kovvuri, Sitaram Lanka, Friedel van Megen, Dima Mukhortov, Prerak Patel, Steve Reinhardt, Adam Sapek, Raja Seera, Balaji Sridharan, Lisa Woods, Phillip Yi-Xiao, Ritchie Zhao, and Doug Burger. Accelerating persistent neural networks at datacenter scale. In *HotChips*, 2017.

[15] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W Keckler, and William J Dally. SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks. In *ISCA*, 2017.

[16] Renzo Andri, Lukas Cavigelli, Davide Rossi, and Luca Benini. Yodann: An ultra-low power convolutional neural network accelerator based on binary weights. *arXiv*, 2016.

[17] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *ISCA*, 2016.

[18] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ISCA*, 2016.

[19] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *JSSC*, 2017.

[20] Duckhwan Kim, Jaeha Kung, Sek Chai, Sudhakar Yalamanchili, and Saibal Mukhopadhyay. Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 380–392. IEEE, 2016.

[21] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *ISCA*, 2017.

[22] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ASPLOS*, 2014.

[23] Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Vikas Chandra, and Hadi Esmaeilzadeh. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks.

[24] Vahide Aklaghi, Amir Yazdanbakhsh, Kambiz Samadi, Hadi Esmaeilzadeh, and Rajesh K. Gupta. Snapea: Predictive early activation for reducing computation in deep convolutional neural networks. In *ISCA*, 2018.

[25] Kartik Hegde, Jiyong Yu, Rohit Agrawal, Mengjia Yan, Michael Pellauer, and Christopher W Fletcher. Ucnn: Exploiting computational reuse in deep neural networks via weight repetition. *ISCA*, 2018.

[26] Jinmook Lee, Changhyeon Kim, Sanghoon Kang, Dongjoo Shin, Sangyeob Kim, and Hoi-Jun Yoo. Unpu: A 50.6 tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision. In *ISSCC*, 2018.

[27] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *ISCA*, 2016.

[28] Prakalp Srivastava, Mingu Kang, Sujan K Gonugondla, Sungmin Lim, Jungwook Choi, Vikram Adve, Nam Sung Kim, and Naresh Shanbhag. Promise: An end-to-end design of a programmable mixed-signal accelerator for machine-learning algorithms. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018.

[29] YP Tsividis and D Anastassiou. Switched-capacitor neural networks. *Electronics Letters*, 23(18):958–959, 1987.

[30] Robert LiKamWa, Yunhui Hou, Julian Gao, Mia Polansky, and Lin Zhong. Redeye: analog convnet image sensor architecture for continuous mobile vision. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 255–266. IEEE Press, 2016.

[31] Daniel Bankman and Boris Murmann. Passive charge redistribution digital-to-analogue multiplier. *Electronics Letters*, 51(5):386–388, 2015.

[32] E. H. Lee and S. S. Wong. Analysis and design of a passive switched-capacitor matrix multiplier for approximate computing. *IEEE Journal of Solid-State Circuits*, 52(1):261–271, Jan 2017. ISSN 0018-9200. doi: 10.1109/JSSC.2016.2599536.

[33] Daniel Bankman, Lita Yang, Bert Moons, Marian Verhelst, and Boris Murmann. An always-on 3.8 μj/86% cifar-10 mixed-signal binary cnn processor with all memory on chip in 28nm cmos. In *Solid-State Circuits Conference-(ISSCC), 2018 IEEE International*, pages 222–224. IEEE, 2018.

[34] Fred N Buhler, Peter Brown, Jiabo Li, Thomas Chen, Zhengya Zhang, and Michael P Flynn. A 3.43 tops/w 48.9 pj/pixel 50.1 nj/classification 512 analog neuron sparse coding neural network with on-chip learning and classification in 40nm cmos. In *VLSI Circuits, 2017 Symposium on*, pages C30–C31. IEEE, 2017.

[35] Renée St. Amant, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, Hadi Esmaeilzadeh, Arjang Hassibi, Luis Ceze, and Doug Burger. General-purpose code acceleration with limited-precision analog computation. In *ISCA*, 2014.

[36] Jintao Zhang, Zhuo Wang, and Naveen Verma. 18.4 a matrix-multiplying adc implementing a machine-learning classifier directly with data conversion. In *Solid-State Circuits Conference-(ISSCC), 2015 IEEE International*, pages 1–3. IEEE, 2015.

[37] Edward H Lee and S Simon Wong. Analysis and Design of a Passive Switched-Capacitor Matrix Multiplier for Approximate Computing. *IEEE Journal of Solid-State Circuits*, 52(1):261–271, 2017.

[38] Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In *ISCA*, 2016.

[39] Linghao Song, Xuehai Qian, Hai Li, and Yiran Chen. Pipelayer: A pipelined reram-based accelerator for deep learning. In *HPCA*, 2017.

[40] Sayeh Sharify, Alberto Delmas Lascorz, Patrick Judd, and Andreas Moshovos. Loom: Exploiting weight and activation precisions to accelerate convolutional neural networks. *arXiv*, 2017.

[41] Paul R Gray, Paul Hurst, Robert G Meyer, and Stephen Lewis. *Analysis and design of analog integrated circuits*. Wiley, 2001.

[42] Pieter Harpe. A 0.0013 mm2 10b 10ms/s sar adc with a 0.0048 mm2 42db-rejection passive fir filter. In *2018 IEEE Custom Integrated Circuits Conference, CICC 2018*. Institute of Electrical and Electronics Engineers Inc., 2018.

[43] Facebook AI Research. Caffe2. https://caffe2.ai/.

[44] Angad S Rekhi, Brian Zimmer, Nikola Nedovic, Ningxi Liu, Rangharajan Venkatesan, Miaorong Wang, Brucek Khailany, William J Dally, and C Thomas Gray. Analog/mixed-signal hardware error modeling for deep learning inference. In *Proceedings of the 56th Annual Design Automation Conference 2019*, page 81. ACM, 2019.

[45] Mohammed Ismail and Terri Fiez. *Analog VLSI: signal and information processing*, volume 166. McGraw-Hill New York, 1994.

[46] Vaibhav Tripathi and Boris Murmann. Mismatch characterization of small metal fringe capacitors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61 (8):2236–2242, 2014.

[47] Yasuko Eckert, Nuwan Jayasena, and Gabriel H Loh. Thermal feasibility of die-stacked processing in memory. 2014.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. URL http://image-net.org/.

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[51] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv*, 2016.

[52] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 2009.

[53] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[55] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[56] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 1993.

[57] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[58] Mingyu Gao, Jing Pu, Xuan Yang, Mark Horowitz, and Christos Kozyrakis. Tetris: Scalable and efficient neural network acceleration with 3d memory. https://github.com/stanford-mast/nn_dataflow, 2017.

[59] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv*, 2016.

[60] Asit K. Mishra, Eriko Nurvitadhi, Jeffrey J. Cook, and Debbie Marr. WRPN: wide reduced-precision networks. *arXiv*, 2017.

[61] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv*, 2016.

[62] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. *arXiv preprint arXiv:1807.10029*, 2018.

[63] Linghao Song, Xuehai Qian, Hai Li, and Yiran Chen. Pipelayer: A pipelined reram-based accelerator for deep learning. In *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*, pages 541–552. IEEE, 2017.

[64] Nvidia tensor rt 5.1. https://developer.nvidia.com/tensorrt.

[65] NCSU. Freepdk45, 2018. URL https://www.eda.ncsu.edu/wiki/FreePDK45.

[66] B. Murmann. *ADC Performance Survey 1997-2016.* murmann/adcsurvey.html, [Online]. Available. URL http://web.stanford.edu/.

[67] S. Li, K. Chen, J. H. Ahn, J. B. Brockman, and N. P. Jouppi. CACTI-P: Architecture-level Modeling for SRAM-based Structures with Advanced Leakage Reduction Techniques. In *ICCAD*, 2011.

[68] Hybrid Memory Cube Consortium et al. Hybrid memory cube specification 1.0. *Last Revision Jan*, 2013.

[69] Joe Jeddeloh and Brent Keeth. Hybrid memory cube new dram architecture increases density and performance. In *VLSI Technology (VLSIT), 2012 Symposium on*, pages 87–88. IEEE, 2012.

[70] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[71] Neta Zmora, Guy Jacob, and Gal Novik. Neural network distiller, June 2018. URL https://doi.org/10.5281/zenodo.1297430.

[72] Yun Long, Taesik Na, and Saibal Mukhopadhyay. Reram-based processing-in-memory architecture for recurrent neural network acceleration. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, (99):1–14, 2018.

[73] Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R Stanley Williams, Paolo Faraboschi, John Paul Strachan, Kaushik Roy, and Dejan S Milojicic. Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference. *arXiv preprint arXiv:1901.10351*, 2019.

[74] Amir Yazdanbakhsh, Michael Brzozowski, Behnam Khaleghi, Soroush Ghodrati, Kambiz Samadi, Nam Sung Kim, and Hadi Esmaeilzadeh. Flexigan: An end-to-end solution for fpga acceleration of generative adversarial networks. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 65–72. IEEE, 2018.

[75] Mohammad Samragh, mojan javaheripi, and Farinaz Koushanfar. Encodeep: Realizing bit-flexible encoding for deep neural networks. *ACM Transactions on Embedded Computing Systems (TECS)*.

[76] Bita Darvish Rouhani, Mohammad Samragh, Mojan Javaheripi, Tara Javidi, and Farinaz Koushanfar. Deepfense: Online accelerated defense against adversarial deep learning. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[77] Shehzeen Hussain, Mojan Javaheripi, Paarth Neekhara, Ryan Kastner, and Farinaz Koushanfar. Fastwave: Accelerating autoregressive convolutional neural networks on fpga. *arXiv preprint arXiv:2002.04971*, 2020.

[78] Sungju Ryu, Hyungjun Kim, Wooseok Yi, and Jae-Joon Kim. Bitblade: Area and energy-efficient precision-scalable neural network accelerator with bitwise summation. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[79] Soroush Ghodrati, Hardik Sharma, Cliff Young, Nam Sung Kim, and Hadi Esmaeilzadeh. Bit-parallel vector composability for neural acceleration. In *Proceedings of the 57th Annual IEEE/ACM Design Automation Conference (DAC)*, July 2020.

[80] Jan Crols and Michel Steyaert. Switched-opamp: An approach to realize full cmos switched-capacitor circuits at very low power supply voltages. *IEEE Journal of Solid-State Circuits*, 29(8):936–942, 1994.

[81] John K Fiorenza, Todd Sepke, Peter Holloway, Charles G Sodini, and Hae-Seung Lee. Comparator-based switched-capacitor circuits for scaled cmos technologies. *IEEE Journal of Solid-State Circuits*, 41(12):2658–2668, 2006.

[82] Robert W Brodersen, Paul R Gray, and David A Hodges. Mos switched-capacitor filters. *Proceedings of the IEEE*, 67(1):61–75, 1979.

[83] Daniel Bankman and Boris Murmann. An 8-bit, 16 input, 3.2 pj/op switched-capacitor dot product circuit in 28-nm fdsoi cmos. In *Solid-State Circuits Conference (A-SSCC), 2016 IEEE Asian*, pages 21–24. IEEE, 2016.

[84] Daisuke Miyashita, Shouhei Kousai, Tomoya Suzuki, and Jun 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*, 52(10):2679–2689, 2017.

[85] Ximing Qiao, Xiong Cao, Huanrui Yang, Linghao Song, and Hai Li. Atomlayer: a universal reram-based cnn accelerator with atomic layer computation. In *Proceedings of the 55th Annual Design Automation Conference*, page 103. ACM, 2018.

[86] Houxiang Ji, Linghao Song, Li Jiang, Hai Halen Li, and Yiran Chen. Recom: An efficient resistive accelerator for compressed deep neural networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*, pages 237–240. IEEE, 2018.

[87] Bing Li, Linghao Song, Fan Chen, Xuehai Qian, Yiran Chen, and Hai Helen Li. Reram-based accelerator for deep learning. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*, pages 815–820. IEEE, 2018.

[88] Lerong Chen, Jiawen Li, Yiran Chen, Qiuping Deng, Jiyuan Shen, Xiaoyao Liang, and Li Jiang. Accelerator-friendly neural-network training: learning variations and defects in rram crossbar. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 19–24. European Design and Automation Association, 2017.

[89] Yu Ji, Youyang Zhang, Xinfeng Xie, Shuangchen Li, Peiqi Wang, Xing Hu, Youhui Zhang, and Yuan Xie. Fpsa: A full system stack solution for reconfigurable reram-based nn accelerator architecture. *arXiv preprint arXiv:1901.09904*, 2019.

[90] Tzu-Hsien Yang, Hsiang-Yun Cheng, Chia-Lin Yang, I Tseng, Han-Wen Hu, Hung-Sheng Chang, Hsiang-Pang Li, et al. Sparse reram engine: joint exploration of activation and weight sparsity in compressed neural networks. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 236–249. ACM, 2019.