A 95.6-TOPS/W Deep Learning Inference Accelerator With Per-Vector Scaled 4-bit Quantization in 5 nm

Ben Keller, *Member, IEEE*, Rangharajan Venkatesan, *Member, IEEE*, Steve Dai, Stephen G. Tell, *Member, IEEE*, Brian Zimmer, *Member, IEEE*, Charbel Sakr, William J. Dally, *Fellow, IEEE*, C. Thomas Gray, *Senior Member, IEEE*, and Brucek Khailany, *Senior Member, IEEE*

***Abstract*— The energy efficiency of deep neural network (DNN) inference can be improved with custom accelerators. DNN inference accelerators often employ specialized hardware techniques to improve energy efficiency, but many of these techniques result in catastrophic accuracy loss on transformer-based DNNs, which have become ubiquitous for natural language processing (NLP) tasks. This article presents a DNN accelerator designed for efficient execution of transformers. The proposed accelerator implements per-vector scaled quantization (VSQ), which employs an independent scale factor for each 64-element vector to enable the use of 4-bit arithmetic with little accuracy loss and low energy overhead. Using a multilevel dataflow to maximize reuse, the 5-nm prototype achieves 95.6 tera-operations per second per Watt (TOPS/W) at 0.46 V on a 4-bit benchmarking layer with VSQ. At a nominal voltage of 0.67 V, the accelerator achieves 1734 inferences/s/W (38.7 TOPS/W) with only 0.7% accuracy loss on BERT-Base and 4714 inferences/s/W (38.6 TOPS/W) with 0.15% accuracy loss on ResNet-50 by using quantization-aware fine-tuning to recover accuracy, demonstrating a practical accelerator design for energy-efficient DNN inference.**

***Index Terms*— Accuracy-efficiency trade-off, BERT, deep neural network (DNN) inference accelerator, quantization, transformers.**

I. Introduction

DEEP neural networks (DNNs) are now a foundational building block in a wide variety of application domains, including computer vision (CV) [1], natural language processing (NLP) [2], computer graphics [3], autonomous driving [4], and many others. While early DNNs generally employed convolutional neural network (CNN) architectures [5], the introduction of attention-based transformer architectures [6] and the subsequent success of very large transformer-based language models on natural language tasks [7] has motivated the development of transformers for CV as well [8] (see Fig. 1). These trends suggest that transformers will become an increasingly common workload across a wide range of applications and deployment targets.

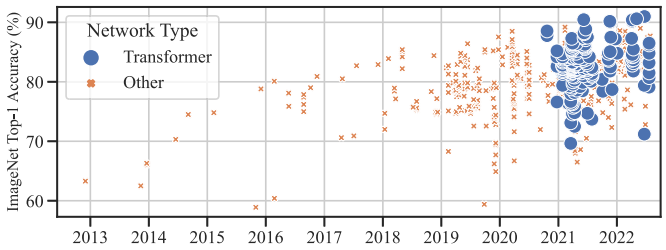


Fig. 1. Transformers achieve high accuracy on image classification [11].

The high arithmetic intensity and deterministic structure of DNN inference computation makes this workload an ideal candidate for dedicated hardware acceleration to improve energy efficiency [measured in tera-operations per second per Watt (TOPS/W)] and area efficiency (measured in TOPS/mm^2) compared with software implementations on CPUs or GPUs. One common technique to improve hardware efficiency is quantization, in which data are represented as integers of eight or fewer bits, reducing both arithmetic and data movement costs. However, excessive quantization can introduce errors that impact the task accuracy of the DNN [9], and transformers are more sensitive than CNNs to quantization error [10].

This work presents a DNN inference accelerator that achieves very high energy efficiency while maintaining task accuracy on transformer workloads with a technique called per-vector scaled quantization (VSQ). After providing additional background in Section II, we describe our accelerator architecture and key design decisions in Section III. Section IV describes our approach to quantization-aware training (QAT) and mapping of transformers, key software workflows developed in tandem with the accelerator to maintain high accuracy. Section V highlights key measurement results, including an energy efficiency of 95.6 TOPS/W, a 5.8× improvement over prior work.

Manuscript received 2 September 2022; revised 10 November 2022 and 5 December 2022; accepted 25 December 2022. Date of publication 18 January 2023; date of current version 28 March 2023. This article was approved by Associate Editor Mototsugu Hamada. *(Ben Keller and Rangharajan Venkatesan contributed equally to this work.) (Corresponding author: Ben Keller.)*

Ben Keller, Rangharajan Venkatesan, Steve Dai, Brian Zimmer, Charbel Sakr, and William J. Dally are with NVIDIA, Inc., Santa Clara, CA 95051 USA (e-mail: benk@nvidia.com).

Stephen G. Tell and C. Thomas Gray are with NVIDIA, Inc., Durham, NC 27713 USA.

Brucek Khailany is with NVIDIA, Inc., Austin, TX 78717 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2023.3234893.

Digital Object Identifier 10.1109/JSSC.2023.3234893

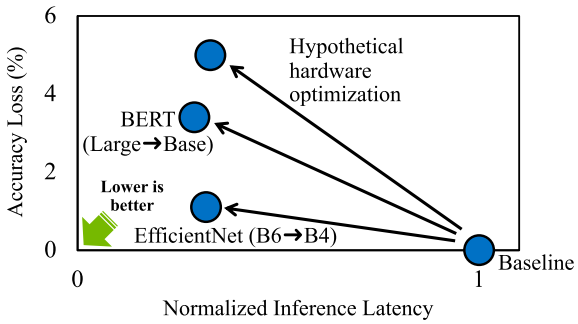


Fig. 2. Simpler network models can often outperform hardware optimizations. Latency data are gathered from [19] and [20].

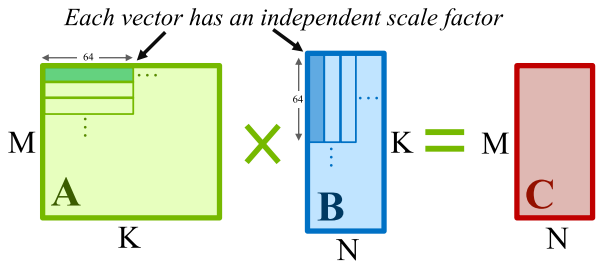


Fig. 3. MM with VSQ.

II. Background

A. Accuracy/Energy Trade-Off

Energy efficiency is a critical metric for DNN inference, both for power-constrained datacenter servers and battery-limited mobile devices. Prior DNN inference accelerators have proposed energy-saving techniques, such as quantization [12], sparsity [13], low-rank approximation [14], analog and in-memory computing [15], and approximate computing [16], all of which improve efficiency by either reducing the energy cost of each operation or reducing the number of operations required to complete the inference task. Each of these techniques introduces error in the computed result, and while DNN inference is resilient to small errors in computation [17], large enough errors can propagate to the network output, resulting in a reduction in accuracy. These approaches, therefore, introduce a trade-off between energy and accuracy, with more aggressive energy-saving measures resulting in larger errors and accuracy drops on a given network and task.

One challenge of realizing hardware techniques for energy efficiency improvement is that the amount of accuracy that can be sacrificed for improved efficiency is very small in practice. Consider a hypothetical hardware optimization that achieves a 3× reduction in inference latency while incurring a 5% accuracy loss. This may seem like a worthwhile trade-off, but as Fig. 2 shows, deep networks, such as BERT [2] and EfficientNet [18], can easily be downsized to shallower counterparts with a more favorable efficiency-accuracy trade-off, with no special hardware required. (We compare latency in this example, because benchmarking data are more readily available, but energy efficiency follows similar trends.) It is critical that the proposed energy-saving techniques have a negligible impact on task accuracy, so that they can outperform the straightforward energy-saving technique of employing a simpler network.

B. Per-Vector Scaled Quantization

Reducing arithmetic precision by quantizing 32-bit floating-point (FP32) number representations to low-bitwidth integers is a powerful technique to save energy. The energy cost of data movement and storage scales linearly with bitwidth, while the cost of the multiply-and-accumulate (MAC) operations that comprise the core arithmetic of deep learning (DL) scales quadratically with input bitwidth. Typical quantization techniques employ a coarse scale factor granularity, using one common scale factor applied to many thousands of values within an input tensor. While per-tensor scaling is computationally inexpensive as the cost of the scale factor multiplication is amortized across many elements, it tends to require mapping a wider range of values to the low-precision representation, increasing quantization error. Quantization to 8-bit integers (INT8) can generally be realized with minimal (<0.1%) accuracy loss on most DNN architectures, but naive quantization to smaller representations can lead to catastrophic accuracy loss. For example, BERT-Large can be quantized to INT8 with only 1.5% accuracy loss using post-training quantization, but further reduction in precision to 4 bits (INT4) results in an unacceptable accuracy loss of 84% [21].

This work instead uses VSQ with fine-grained scale factors to enable high accuracy on transformers with 4-bit integer representations [21]. VSQ exploits the vector granularity of computation in many DNN inference accelerators. In addition to a coarse-grained per-output-column scale factor, VSQ quantizes data at a per-vector granularity within each input matrix in the matrix–matrix multiplication (MM), as shown in Fig. 3. The range that must be captured by each per-vector scale factor is much smaller than per-tensor scaling, greatly reducing quantization error (see Fig. 4). Per-vector scale factors require minimal hardware overhead, with one additional data word stored per input vector, and one additional scalar multiplication of the two vector scale factors for each vector dot product. The decreased quantization error from VSQ with 4-bit arithmetic (INT4-VSQ) enables minimal DNN accuracy loss on transformers compared with unquantized floating-point representations, while imposing only a small hardware overhead compared with INT4 math.

C. Transformer Networks

Fig. 5 shows the structure of a typical transformer network, which consists of a sequence of transformer layers, each of which contains multihead attention and feed-forward blocks. The structure of transformers differs significantly from that of CNNs. Some MM operations are performed with pretrained weights that do not change during inference; these operations are similar to 1 × 1 convolution (CONV) kernels. However, the batch matrix multiply (BMM) operations have no pretrained weights, with both inputs computed dynamically during runtime. While CNNs contain one softmax calculation, transformers perform softmax more frequently, once per

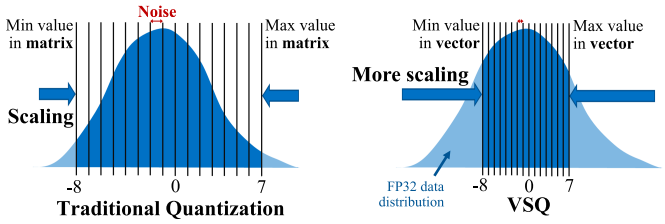


Fig. 4. VSQ reduces quantization noise.

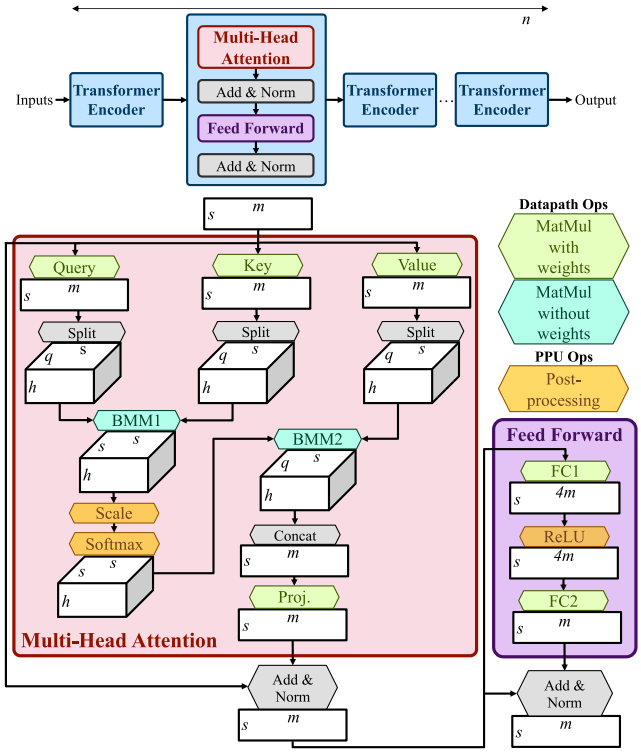


Fig. 5. Structure of transformer networks.

encoder layer. Dimensions of the transformer computations in BERT and DeiT [22] are listed in Table I (except for sequence length s, which varies with workload).

Note that we modify some of the computations within the standard transformer for better hardware efficiency. We replace the expensive nonlinear Gaussian error linear unit (GELU) operation with the simpler rectified linear unit (ReLU) function (the accuracy impact of this change is minimal after retraining). For simplicity, we also ignore some operations with low arithmetic intensity (shown in gray in Fig. 5) in our accelerator design and evaluation. In practice, these operations would be performed at a higher level of system hierarchy (see Section III-F).

Since MMs form a key building block for transformers, we use them as an example to illustrate different optimizations proposed in this work. For generality across MMs with and without weights, we denote the input matrices as $\boldsymbol{A}$ and $\boldsymbol{B}$ with dimensions $M \times K$ and $K \times N$, respectively, with the output matrix $\boldsymbol{C}$ of dimensions $M \times N$ (see Fig. 3).

D. Prior Work

Recent years have seen a proliferation of DNN inference accelerator designs. Digital accelerators exploit

TABLE I
TRANSFORMER NETWORK PARAMETERS

Network	BERT-Large	BERT-Base	DeiT-Base	DeiT-Small
Model (m)	1024	768	768	384
Heads (h)	16	12	12	6
Query Size (q)	64	64	64	64
Layers (n)	24	12	12	12
MACs (x10^9)	123	35	8.5	1.1

quantization [12], [23], [24], [25], [26], [27], [28], sparsity [14], [23], [24], [27], and other energy-saving hardware optimizations [16], [29], while compute-in-memory accelerators make use of analog circuit properties to achieve very high reported energy efficiencies [15], [24], [25], [30], [31], [32]. However, many prior hardware accelerator proposals fail to evaluate task accuracy entirely [15], [27], [31], evaluate their proposals only on very small networks [25], [29], [30], or suffer large (>1%) drops in accuracy on realistic workloads [24], [32]. This work focuses on quantization techniques that can achieve large energy efficiency gains with minimal accuracy loss on transformers.

Prior work has also pursued optimizations specific to transformers. Algorithmic optimizations have been employed to reduce network size using techniques, such as knowledge distillation [33], inductive biases [34], and approximations [35], [36], [37], [38]. Quantization techniques with coarse-grained scaling have been used to enable INT8 for transformers [39], [40]. This work explores more aggressive quantization with INT4-VSQ while maintaining accuracy. Prior works also exploit sparsity to skip unnecessary computations in attention layers [16], [41], [42], [43], [44], and other hardware-software techniques have been explored to improve energy efficiency through early-exit prediction [45], approximate softmax [36], [38], and customized tiling schemes [46]. These techniques are complementary to our work, which also includes a hardware-efficient approximate softmax engine.

III. ACCELERATOR DESIGN

The accelerator design focused on three key objectives: 1) high energy efficiency by exploiting low-precision computation and data reuse across different operands; 2) high area efficiency by maximizing compute throughput while amortizing memory and control overheads; and 3) programmability to support different data formats and types of computations.

A. Inference Accelerator Architecture

Fig. 6 shows the architecture of the accelerator, which computes a CONV, MM, or fully connected (FC) layer followed by optional post-processing operations. It consists of an array of vector MAC units, SRAM buffers to store the input matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and the output matrix $\boldsymbol{C}$, a latch-array-based collector to accumulate partial sum values, an input register for the $\boldsymbol{A}$ operand, and a post-processing unit (PPU). The vector MAC units are configurable, performing 32-wide (INT8) or 64-wide (INT4 and INT4-VSQ) vector MAC operations each cycle, which enables direct energy efficiency comparison between the three formats and support for workloads with coarse-grained

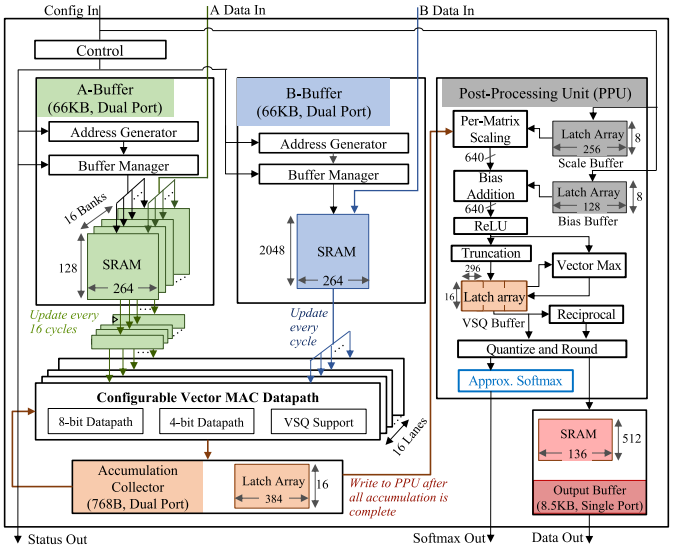


Fig. 6. Accelerator block diagram.

heterogeneous precision. The accelerator has 16 vector lanes; the 16 vector MAC units together perform a matrix-vector product and produce 16 different partial sum values each cycle that are stored in the accumulation collector.

The accelerator achieves multiple levels of reuse across all three matrices: temporal $\boldsymbol{A}$-input reuse by storing vectors in the $\boldsymbol{A}$ register for 16 cycles, temporal output reuse through the accumulation collector, spatial reuse in the $\boldsymbol{B}$ input by broadcasting to each of the 16 vector MAC units, and spatial output reuse in the adder tree of vector MAC units [47]. The accumulation collector is implemented with a dual-ported latch array to enable simultaneous read/write operations every cycle at low energy cost. More details of the dataflow and mapping approach are presented in Section IV-B.

The buffers are designed to provide adequate bandwidth to the vector MAC units to achieve high throughput. The $\boldsymbol{A}$ buffer has 16 banks to load a different $\boldsymbol{A}$-input vector for each vector lane. Each entry of the banks in the $\boldsymbol{A}$ and $\boldsymbol{B}$ buffers is 264 bits wide to store either 1) a 32-wide 8-bit vector for INT8 format; 2) a 64-wide 4-bit vector for INT4 format; or 3) a 64-wide 4-bit vector and an 8-bit scale factor for INT4-VSQ format. This matches the input width of a vector MAC unit, providing the required read bandwidth to achieve high utilization. The 16 banks of the $\boldsymbol{A}$ buffer supply 16 unique $\boldsymbol{A}$-input vectors to each of the lanes, while the $\boldsymbol{B}$-input vector read from the $\boldsymbol{B}$ buffer is shared across all the lanes. The input buffers $\boldsymbol{A}$ and $\boldsymbol{B}$, which are dual-ported to support one read and one write operation per cycle per bank, employ buffer managers that enable fine-grained overlap of load and store operations in the buffers [48], avoiding the area overheads associated with double-buffering. Data are stored sequentially into these buffers, exploiting the streaming nature of DL applications. Read accesses need not be sequential; different strides, reuse patterns, and tile sizes are enabled by address generators that are programed to generate the required sequence of addresses for read operations. The buffers enable temporal data reuse and reduce data movement cost. In particular, the partial sum values have wider widths than input operands, and their load/store from SRAMs is expensive. Therefore, the $\boldsymbol{A}$ buffer and $\boldsymbol{B}$ buffer are sized to achieve temporal output partial sum reuse up to 128 times in the accumulation collector. This enables the accelerator to execute all of the computations in BERT-Large without requiring accesses to external memories for partial sum values.

The SRAM buffers and the latch-array collectors are shared across different data formats to improve the area efficiency of the accelerator. The width of the $\boldsymbol{A}$ and $\boldsymbol{B}$ buffers is sized so that they can optionally store per-vector scale factors alongside each vector in the INT4-VSQ format. With a bitwidth of 264, each entry in the buffers can store a 32-wide 8-bit vector, a 64-wide 4-bit vector, or a 64-wide 4-bit vector and an 8-bit per-vector scale factor. The accumulation width of each vector MAC unit is 24 bits for all the three data formats, with the accumulation collector storing 24-bit partial sum values from 16 vector lanes each cycle. All vector lanes operate in lockstep to amortize control overheads.

Once accumulation is complete, the final sum values are read from the accumulation collector to perform post-processing operations in the PPU. To ensure high utilization, the final sum reads are pipelined with the first set of store operations of the next set of partial sums. The PPU performs scaling and quantization of the partial sum values. It also supports bias addition and non-linear operations, such as ReLU and approximate softmax. The approximate softmax implementation is designed to improve hardware efficiency by using base 2 instead of base e, low-precision fixed-point data formats, and online normalization to reduce data movement [49]. QAT (described in IV-A) is used to mitigate the accuracy impact of the approximate softmax implementation.

The PPU is configurable to quantize the 24-bit outputs to INT8, INT4, or INT4-VSQ, depending on the desired data format for the output activations. Computing per-vector scale factors for 64-wide vectors presents certain challenges, as only 16 values are available across vector lanes and consecutive values in the accumulation collector are not ordered along the vector dimension in order to exploit weight reuse. To address this issue, the PPU includes additional latch-array buffers that store partial sum values after scaling and bias addition. Once the required values are available, the vector-max and reciprocal units compute per-vector scale factors, and the vector is quantized.

B. Reconfigurable Datapath Design

Fig. 7 shows the microarchitecture of the single-cycle vector MAC datapath. It has dedicated datapaths that perform 32-wide INT8 and 64-wide INT4 signed dot-product computations. The output of the dot product is accumulated with signed 24-bit partial sum values for all data formats. To support INT4-VSQ, the datapath includes an unsigned 8-bit multiplier that computes the product of the two input vector scale factors. Multiplication of the 8-bit scale factors produces a 16-bit output, which is rounded to 8 bits before multiplication with the 14-bit output of the dot product. This rounding reduces the logic complexity of the datapath, improving energy efficiency with negligible impact on the accuracy of the computation.

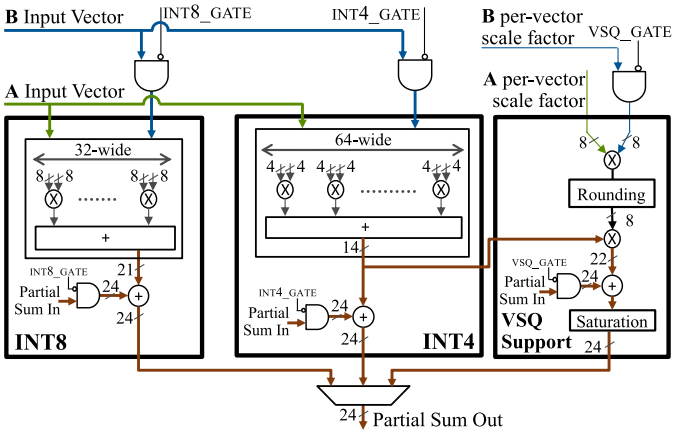


Fig. 7. Reconfigurable vector datapath.

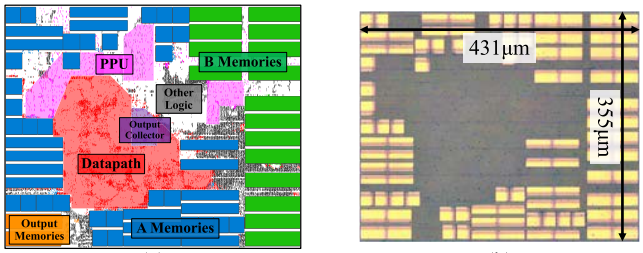


Fig. 8. (a) Annotated accelerator floorplan. (b) Die plot.

To avoid overflow errors during temporal accumulation, the datapath supports saturation of output partial sums.

The datapath can be reconfigured to support three different data formats. As shown in Fig. 7, inactive datapaths are gated by ANDing only the ***B*** input with zeroes, which minimizes area overhead while improving energy efficiency. For example, to configure the datapath for INT4-VSQ mode, INT4_GATE and VSQ_GATE are deasserted to enable the INT4 datapath and VSQ logic, while INT8_GATE is asserted to gate the INT8 datapath.

C. Physical Design

The physical implementation flow used to implement the accelerator favored simplicity, flexibility, and rapid turnaround time over absolute quality of results [50]. The accelerator was designed with a SystemC-based object-oriented high-level synthesis flow [51], which improved productivity by enabling design at a higher level of abstraction and speeding verification. Fig. 8(a) shows an annotated diagram of the accelerator floorplan. The entire accelerator, which is composed of 379k gates, was synthesized, placed, and routed as a single flat partition, enabling push-button SystemC-to-layout spins in <48 h. Macros were placed by the place-and-route (P&R) tool in an initial floorplanning step prior to standard cell placement. The tool was permitted to re-floorplan the macros optimally in each subsequent P&R spin, even just prior to tape out, which proved valuable in accommodating late design changes. Additional guard rings were added around each macro to ensure that arbitrary macro placements would not cause DRC or routability issues. Partition utilization was deliberately kept low to reduce P&R runtime and accommodate the macro-dominated floorplan, with a final standard cell utilization of 60.1%.

TABLE II
SIMULATED ACCELERATOR ENERGY BREAKDOWN

Data Format	Datapath	***A***-Buffer	***B***-Buffer	Accum. Collector	Other
INT8	2.76	0.36	0.44	0.71	1.05
INT4	1.00	0.18	0.22	0.30	0.60
INT4-VSQ	1.01	0.18	0.22	0.30	0.61

D. Clocking

The accelerator clock is generated entirely on die via a simple multiplexor-based 64-tap ring oscillator (RO) located near the accelerator instance that is placed and routed automatically, with false-path timing constraints specified to break the combinational loop. The generated clock adapts its frequency to any changes in supply voltage, approximating the delay of the critical paths in the design. The partition is entirely synchronous except for a small amount of logic for external data transfer that is clocked by a 100-MHz reference and communicates with the accelerator via bisynchronous FIFOs. The accelerator met a clock period target of 1.1 ns at the typical 0.67-V signoff corner, with the critical paths primarily occurring from the ***A*** register to the accumulation collector through the INT8 datapath. The relatively relaxed frequency target improved spin times and energy efficiency while maintaining reasonable area efficiency.

E. Energy Breakdown

Table II shows a simulated energy breakdown of the accelerator running a synthetic benchmark consisting of Gaussian random input data with 70% of the inputs set to zero (30% sparsity) in each of the supported data formats, normalized to INT4 datapath energy. Energy was measured with PrimePower at the typical 0.67-V corner using traces of glitch-aware gate-level simulations of the post-P&R netlist and normalized by the number of MAC operations performed each cycle. For all data formats, the datapath is the single largest consumer of energy, as buffer energy is reduced by amortizing accesses via the efficient dataflow described in Section IV-B and employing energy-efficient latch arrays for the shallow accumulation collectors. Control overheads are amortized by the wide vector MAC operations, and a sufficiently large tile size minimizes the relative cost of PPU computations. While some prior works claim that memory accesses dominate DL inference accelerator energy costs [52], [53], [54], [55], we demonstrate that an efficient implementation can greatly reduce non-datapath overheads.

F. Scalable System

The described accelerator lacks a number of capabilities that would be required to accelerate neural network inference end-to-end, including a flexible high-capacity memory system, an integrated controller, and hardware support for operations such as layerwise addition, normalization, and transpose.

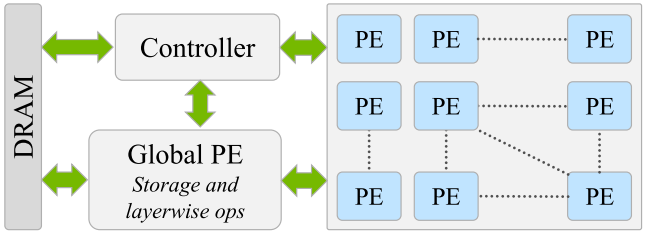


Fig. 9. Complete DL inference system composed of many instances of the proposed accelerator as processing elements (PEs).

We made the intentional decision to design an optimized accelerator core that could be scaled to a full system via replication. The core was designed to include all hardware components that would be needed for replicated cores comprising a full system. For example, the accelerator includes hardware support for streaming $\boldsymbol{A}$ and $\boldsymbol{B}$ inputs into the local buffers during execution, even though this operating mode is not supported in the silicon implementation due to the lack of a memory system. Fig. 9 shows an example of how the proposed design could be tiled into a full system.

IV. Software Workloads

This section describes how pretrained DNN models are fine-tuned and mapped onto the accelerator. The accelerator design was tightly coupled with the development of the software workflow.

A. Quantization-Aware Fine-Tuning

To achieve the best accuracy, each target model is fine-tuned from a pre-trained full-precision checkpoint targeting a specific task. Each iteration of fine-tuning consists of a hardware-aware forward pass to faithfully simulate the effects of hardware optimizations on the accuracy of the model deployed on the accelerator, followed by a backward pass based on the straight-through estimator (STE) to update the model with hardware optimizations in the loop.

1) Forward Pass: Tensor computations, such as CONV and MM layers, can be expressed as a set of partial sums $S(k)$ as shown in (1), where I_A and I_B are the integer input vectors, V is the vector size, J is the number of vectors in the tensor, and s_A and s_B are the per-vector scale factors

$$S(k) = \sum_{j=0}^{J-1}\left(\sum_{i=0}^{V-1} I_A(j,i)I_B(j,i)\right)s_A(j)s_B(j). \tag{1}$$

For the INT8 and INT4 data formats, $s_A(j) = s_B(j) = 1$. For INT4-VSQ, the scale factors are computed using a two-level quantization scheme [21]. The first quantization level computes per-vector floating-point scale factors, and the second level quantizes the per-vector scale factors into integer per-vector components and a floating-point per-tensor component.

While a hardware-agnostic tensor-level forward pass could serve as a good baseline for abstracting the computations in an accelerator, it is insufficient for deriving the actual accuracy realized by the accelerator. As shown in Fig. 7, the per-vector scale factor product is rounded from 16 to 8 bits prior to scaling the dot product. In addition, the partial sum saturates at 24 bits, even though its theoretical maximum exceeds this bitwidth. Moreover, fixed-point math with finite precision is used in the reciprocal operation of the PPU shown in Fig. 6 to dynamically compute the per-vector scale factors. Considering these optimizations, the actual partial sums on the accelerator $S'(k) = S'(k, J)$ are expressed recursively as in (2), where $\mathcal{R}$ denotes rounding and $\mathcal{C}$ denotes saturation

$$S'(k, j_p) = \mathcal{C}\Bigg[S(k, j_{p-1}) + \left(\sum_{i=0}^{V-1} I_A(j_p, i)I_B(j_p, i)\right) \times \mathcal{R}\big[\mathcal{R}\big[s_A(j_p)\big]\mathcal{R}\big[s_B(j_p)\big]\big]\Bigg]. \tag{2}$$

Accurate modeling of the accelerator requires a forward pass that represents all elements of (2). For this purpose, our hardware-aware forward pass splits the tensor-level computations into vector-level computations with attached intermediate rounding and saturation operations. Input tensors are divided into many 64-element chunks, transforming a single tensor multiplication into multiple parallel multiplications of corresponding chunks, as shown in Fig. 3. With this transformation, the result of each vector dot product is presented independently in the resulting tensors, enabling rounding and saturation to be applied on a per-vector granularity. As described in Section III-A, the accelerator uses an approximate softmax implementation and replaces the GELU operation with ReLU, so the forward pass in the QAT framework also modifies the non-linear functions to accurately capture these optimizations.

While it is necessary to divide the operations based on vector size to exactly represent the accelerator, the software model remains abstract and incurs only around 2× runtime overhead, because it does not include details such as dataflow and buffer sizes.

2) Backward Pass: During the backward pass, the training algorithm learns the effects of quantization, saturation, and rounding on the model using backpropagation and adapts the weights of the neural network accordingly to minimize accuracy loss. To backpropagate through these rounding and saturation operations, we apply STE [56] in the backward pass by approximating these operations as the identity function. With these approximations, it follows from (2) and (1) that $S'(k) \approx S(k)$, which implies that $\nabla S'(k) \approx \nabla S(k)$. An interpretation of this result is that since the accelerator approximates the tensor computations represented in (1), the gradients from those tensor computations can be used to train the model to target the accelerator. With this approach, QAT properly models the accelerator in the forward pass as $S'(k)$ to correctly evaluate the actual accuracy, while reasonably approximating the gradients as $\nabla S(k)$ in the backward pass to enable effective weight updates. $\nabla S(k)$ can be implemented by calling well-optimized backpropagation routines of common DL packages, such as PyTorch [57], which is especially helpful in realizing performant QAT.

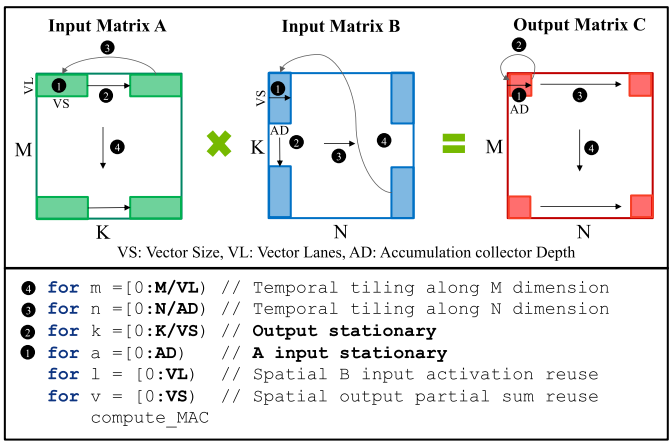


Fig. 10. Mapping MM to the accelerator.

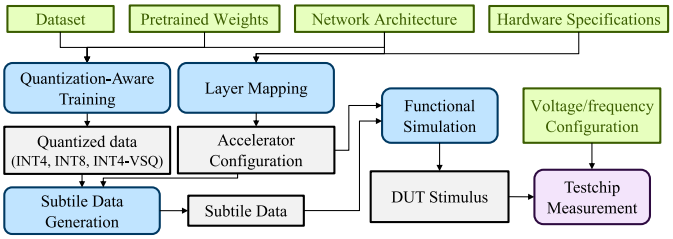


Fig. 11. Workload generation flow.

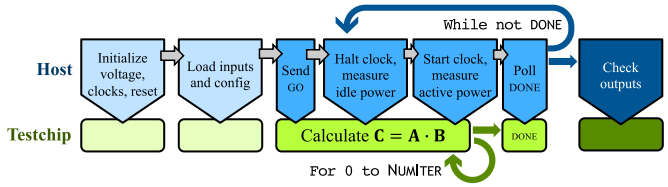


Fig. 12. Flow diagram of the chip measurement procedure.

B. Mapping to the Accelerator

Fig. 10 shows how the different computations in deep neural networks are tiled and executed on the accelerator. In the figure, *VS* denotes the vector size, *VL* the vector lanes, and *AD* the accumulation collector depth. The input matrices ***A*** and ***B*** are allocated to ***A*** and ***B*** buffers, respectively.

First, matrix-vector product computation is performed by reading a sub-matrix of ***A*** of size $VS \times VL$ and a vector of ***B*** of size *VS*. For INT4-VSQ format, *VL* per-vector scale factors of the ***A*** inputs and one per-vector scale factor of the ***B*** input are read along with the ***A*** sub-matrix and ***B*** vector. The product of the ***A*** and ***B*** per-vector scale factors is then multiplied by the matrix-vector product output within the datapaths. This produces a partial sum vector of size *VL*, which is stored in the accumulation collector. The ***A*** sub-matrix along with per-vector scale factors is stored in the ***A*** register and temporally reused *AD* times. Different ***B*** vectors are read each cycle to compute partial sum output vectors that are stored in different entries of the accumulation collector. Since the ***A*** input remains constant, it is referred to as ***A*** input stationary in Fig. 10.

Next, partial sum values in the accumulation collector are reused K/VS times to perform temporal reduction. This stage implements an output stationary dataflow by reusing partial sum outputs in the accumulation collector. This multilevel dataflow is referred to as output stationary and local ***A*** stationary, because outputs are reused in the outer loop, and ***A*** inputs are reused in the inner loop [47]. In this way, an output sub-matrix of size $VL \times AD$ is computed in the accelerator. After the matrix-vector product, the partial sum values are quantized to the desired output format in the PPU. If the output format is INT4-VSQ, VS/VL partial sum vectors (each *VL* wide) are used to compute per-vector scale factors. The entire output matrix is computed via temporal tiling along the *M* and *N* dimensions. The accelerator can also efficiently execute CONV and FC layers by dividing the computation into a sequence of matrix-vector operations.

C. End-to-End Evaluation Framework

Fig. 11 shows the end-to-end workload generation flow used to fine-tune the model and evaluate the chip. Hardware-aware QAT is used to recover accuracy loss from quantization and obtain quantized inputs for each layer in the desired data format. In many cases, a full network layer is too large to be executed on the accelerator, so a representative subtile of the output matrix is selected for execution. The mapping flow uses heuristics to tile the computation to exploit parallelism and data reuse in the hardware, generating the settings that configure the accelerator to perform the required computations in the vector MAC units and PPU. The quantized inputs and configuration settings are used to determine the correct data for input tiles, input scale factors, bias values, and reference outputs that represent each portion of the end-to-end DNN inference workload. This collection of data and configuration settings is used in simulation, which generates the collateral for bench measurement.

V. Measurement Results

A prototype of the accelerator was fabricated in a TSMC 5-nm process [58]. Fig. 8(b) shows a die micrograph; the testsite measures 431×355 μm.

A. Test Setup

Fig. 12 shows the chip measurement workflow. In the initialization phase, voltages and clocks are set, and the accelerator is brought out of reset, so that its configuration registers and input memories can be programed via the external JTAG interface for execution of a single-layer subtile. After initialization is complete, a GO command is sent to the accelerator, beginning the execution phase. Subtile execution completes in much less than a millisecond, which is an insufficient duration for accurate power measurement, and executing subsequent subtiles requires a new initialization sequence. Therefore, the accelerator state machine can be configured to execute the same subtile many millions of times in succession via a configurable NUMITER register, enabling measurement of average power. Execution termination is detected by polling a DONE register over JTAG, which is written by the accelerator once the requested number of workload iterations has been completed. In the verification phase, the outputs calculated

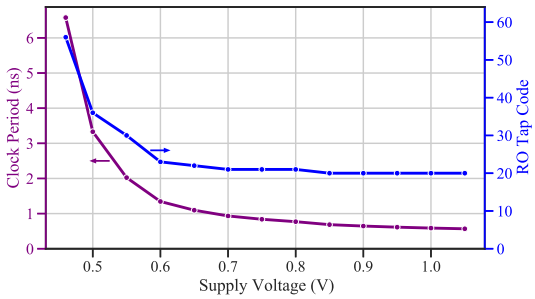


Fig. 13. Clock generator calibrated tap code and period.

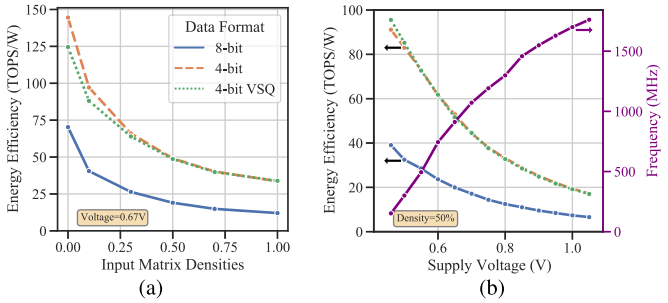


Fig. 14. Measured energy efficiency of the accelerator executing a benchmarking workload with different (a) workloads and (b) operating voltage.

by the accelerator are read out via JTAG and compared with reference outputs to confirm correct execution. Each measurement under a particular voltage and frequency condition is performed once for each subtile, with measured runtime and power extrapolated to determine the execution cost of the full tile and, hence, the full network.

The testsite supply is shared by other circuits on the die, so direct measurement of accelerator power was not possible. Instead, for a given workload, we infer the testsite dynamic power by subtracting idle power from active power and add a leakage power estimate to determine total power. To measure dynamic power, overall rail power is measured via bench measurement of board sense resistors using Agilent 34401A digital multimeters. Then, active and idle power are measured by running and then halting the accelerator clock during workload execution, with power measurements recorded during each phase (see Fig. 12). By repeating this process, five or more measurements of active and idle power are collected for each workload; the average difference between active and idle rail power is the dynamic power of the testsite for that workload. Leakage power is estimated from PrimePower reports and scaled for voltage and temperature as measured by on-die sensors. This estimate was corroborated against measured rail leakage power scaled by the area ratio of the accelerator to that of the entire voltage domain.

All measurements were performed at nominal voltage (0.67 V) unless otherwise noted. The number of operations ("ops") used for TOPS/W and TOPS/mm^2 calculations is based only on the count of MAC operations, with one MAC consisting of two ops, ignoring scale factor multiplications and PPU operations.

B. Clocking

The RO frequency is set by offline calibration of the on-die clock generator by sweeping the RO tap code and using the fastest frequency that maintains functional correctness across a representative workload subset. Fig. 13 shows the best achievable RO tap code at each voltage. Frequency is measured using integrated counters that compare against the 100-MHz reference. At voltages above 600 mV, the RO delay closely matches that of the critical paths in the design, so little adjustment in tap code is necessary to operate at peak frequency. At lower voltages, critical paths likely become dominated by wires or SRAM macros. These structures are not represented in the RO clock generator, so the number of multiplexors in the delay chain must be dramatically increased to track the low-voltage behavior. Each RO tap adjusts the system frequency by ≈5%, so a finer adjustable delay could enable further marginal performance improvement.

C. System Benchmarking

Fig. 14 shows the measured frequency and energy efficiency of the accelerator under various operating conditions when executing a synthetic benchmark. The workload consists of an MM with $\boldsymbol{A}$ and $\boldsymbol{B}$ input values selected from a Gaussian distribution and quantized to each of the three data format modes, with values randomly zeroed out to achieve the target data density. The chip operates correctly down to a minimum voltage V_{min} of 0.46 V and achieves a maximum frequency of 1.76 GHz at the maximum voltage V_{max} of 1.05 V.

As Fig. 14(a) shows, the achieved energy efficiency depends not only on the choice of data format, but also on both the operating voltage and input data density. Even though the accelerator does not explicitly support sparsity through techniques, such as compressed representations or zero skipping, datapath activity is very dependent on workload sparsity, so the power consumption of the accelerator executing identical layer shapes and data format at a fixed voltage can vary by as much as 5.9×. At V_{max} with fully dense data, the INT4-VSQ mode achieves only 11.8 TOPS/W, while at V_{min} using 10% dense data, the accelerator achieves 156.7 TOPS/W. Using the even more unrealistic operating condition of 0% dense data (i.e., all zeroes) increases energy efficiency further to 204.5 TOPS/W. We feel that a fair comparison for benchmarking is to report energy efficiency on 50% dense data at nominal and minimum voltage. Using these criteria, our accelerator achieves 48.7 TOPS/W at 0.67 V using INT4-VSQ and 95.6 TOPS/W at 0.46 V. The use of VSQ imposes a small (0.8%) energy overhead at nominal voltage compared with baseline INT4 math.

Table III shows a comparison to prior DNN inference accelerators that support INT8 or INT4 data formats. Direct comparisons are difficult, because many prior works do not report the data density of the workloads used to measure energy efficiency. Using the reported data, our accelerator achieves a 2.2–4.1× improvement in energy efficiency and a 3.4–15.6× improvement in area efficiency for INT8 inference, and a 5.8× improvement in energy efficiency and a 3.4–4.5× improvement in area efficiency for INT4 inference.

TABLE III
COMPARISON WITH PRIOR WORK

	This Work			Agrawal et al., ISSCC'21 [12]	Mo et al., ISSCC'21 [26]	Park et al., ISSCC'22 [23]	Zimmer et al., JSSC'21 [59]	Wang et al., ISSCC'22 [16]
Process Technology	5 nm			7 nm	28 nm	4 nm	16 nm	28 nm
Area (mm^2)	0.153			19.6	1.9	4.74	6.0	6.82
Supply Voltage (V)	0.46–1.05			0.55–0.75	0.6–0.9	0.55–1.0	0.41–1.2	0.56–1.1
Frequency (MHz)	152–1760			1000–1600	100–470	332–1196	161–2001	50–510
On-Chip SRAM (kB)	141			8192	206	2048	625	336
Target Workloads	CONV, FC, Attention[5]			CONV, FC, RNN	CONV, FC	CNN, FC, DW-CONV	CONV, FC	Attention
Supported Data Formats	INT4	INT4-VSQ	INT8	INT2/4, FP8/16/32	INT8	INT4/8/16, FP16	INT8	INT12
Performance at V_{max} (TOPS)	3.6	3.6	1.8	102.4 (4b)	1.43	39.3 (4b), 19.7 (8b)	4.01	4.07
Energy Eff. at V_{min} (TOPS/W)	91.1[1]	95.6[1]	39.1[1]	16.5[2] (4b)	17.5[2]	11.59[2] (8b)	9.5[3]	27.56[4]
Area Eff. at V_{max} (TOPS/mm^2)	23.3	23.3	11.7	5.22 (4b)	0.75	6.9 (4b), 3.45 (8b)	1.29	0.597

[1]50% input density. [2]Input density not reported. [3]40% input density. [4]90% weakly related tokens.
[5] Operations with low arithmetic intensity such as element-wise addition and normalization are not supported.

TABLE IV
MEASURED PERFORMANCE ON DNN WORKLOADS

Dataset, Task	SQuAD v1.1, Reading Comprehension												ImageNet, Image Classification							
Network	BERT-Base						BERT-Large						DeiT-Small			DeiT-Base			ResNet-50	
Sequence Length	128			384			128			384			197			197			-	
Baseline FP32 Accuracy (%)	87.46			87.50			90.29			90.93			79.82			81.80			76.16	
Data Bitwidth (4V = 4b VSQ)	4	4V	8	4	4V	8	4	4V	8	4	4V	8	4	4V	8	4	4V	8	4	4V
Accuracy Loss (%)	80	0.7	0.7	81	0.5	0	88	1.1	1.1	89	0.8	0.1	29	3.6	0.7	25	1.3	0.4	0.98	0.15
MAC Utilization (%)	-	98	99	-	98	99	-	98	99	-	98	99	-	94	96	-	97	98	89	89
Throughput (inferences/s)	-	87.3	44.2	-	27.7	14.0	-	24.9	12.6	-	7.8	4.0	-	209	108	-	56.2	28.5	212	212
Energy Efficiency (inferences/s/W)	-	1734	768	-	596	281	-	513	219	-	170	80	-	3519	1451	-	1010	406	6355	4714
Energy Efficiency (TOPS/W)	-	38.7	17.2	-	42.1	19.9	-	40.4	17.3	-	42.0	19.8	-	32.0	13.2	-	35.2	14.2	51.2	38.6

D. DNN Workload Performance

Table IV shows achieved performance, energy efficiency, and task accuracy on a variety of DNN workloads. The accelerator achieves high energy efficiency on transformers for NLP (BERT), vision transformers (DeiT), and classical CNNs (ResNet-50 [1]). For the transformer-based workloads, the reported accuracy includes the impact of hardware approximations of non-linear functions, such as softmax and GELU.

With the exception of ResNet-50, the use of INT4 results in catastrophic accuracy loss, even with QAT. INT4-VSQ achieves similar accuracy to INT8 (as well as the FP32 baseline) while achieving $>2\times$ energy efficiency over INT8. Energy efficiencies on these workloads range from 32.0 to 42.0 TOPS/W for INT4-VSQ to 13.2 to 19.9 TOPS/W for INT8. The evaluated workloads exhibit little weight or activation sparsity, so the measured energy-efficiency results are consistent with the benchmarking measurements on dense inputs at nominal voltage.

Performance on NLP tasks is evaluated on BERT-Base and BERT-Large performing a reading comprehension task using the Stanford Question Answering Dataset (SQuAD) [60] with the sequence lengths of 128 and 384. Each task uses weights pretrained on the full English Wikipedia dataset. QAT was applied during five epochs of fine-tuning using the Adam optimizer [61] and a learning rate of $3 \cdot 10^{-5}$. For all transformer networks, elementwise addition and layer normalization (performed only in software) use full FP32 precision. For INT4-VSQ evaluations, FC2 layers used INT8 outputs with per-vector scale factors, which improved accuracy with negligible energy impact. All BERT workloads achieve an accuracy within 1.1% of the FP32 baseline using INT4-VSQ, and BERT-Base achieves 1734 inferences/s/W with a sequence length of 128, a $2.26\times$ energy-efficiency improvement over the 8-bit result.

TABLE V
DATA FORMATS FOR RESNET-50 LAYERS

Data Format	Layer Name
INT8	conv1, fc
INT4	res2a_branch2a, res2a_branch1, res4{a-f}_branch2c, res5{a-c}_branch2c
INT4-VSQ	res2{a-c}_branch2{b-c}, res2{b-c}_branch2a, res3{a-d}_branch2{a-c}, res{3-4}a_branch1, res4{a-f}_branch2{a-b}, res5{a-c}_branch2{a-b}

Performance on CV tasks is evaluated via image classification on ImageNet [62] using both DeiT and ResNet-50. DeiT used QAT for 20 epochs, with a learning rate of $4 \cdot 10^{-5}$ for DeiT-Small and 10^{-5} for DeiT-Base. ResNet-50 was retrained using stochastic gradient descent for 30 epochs with an initial learning rate of 10^{-4}, which was decayed by a factor of 10 every ten epochs, with methods in [63] used for static coarse-grained scale factor calibration and gradient backpropagation. For ResNet-50, different layers use different precision formats, as shown in Table V. The first and last layers are quantized to INT8 in both the INT4 and INT4-VSQ evaluations, as is typical in CNN quantization studies [64]. For the INT4-VSQ evaluation, layers that require large reduction length ($K > 2048$) are too large to fit in the on-chip buffers while still enabling full temporal reduction of partial sums in the accumulation collector, so these layers are configured to product INT4 outputs, which is used as the data format

TABLE VI
SPARSE BERT MEASUREMENT RESULTS

Network	BERT-Base						BERT-Large					
Sequence Length	128			384			128			384		
Weight Density (%)	67	50	33	67	50	33	67	50	33	67	50	33
Accuracy Loss (%)	-.2	.4	2.3	-.3	.4	2.6	0	.2	1.0	.2	.6	1.8
Energy Eff. Gain (%)	11	21	30	8	8	20	8	17	23	7	15	22

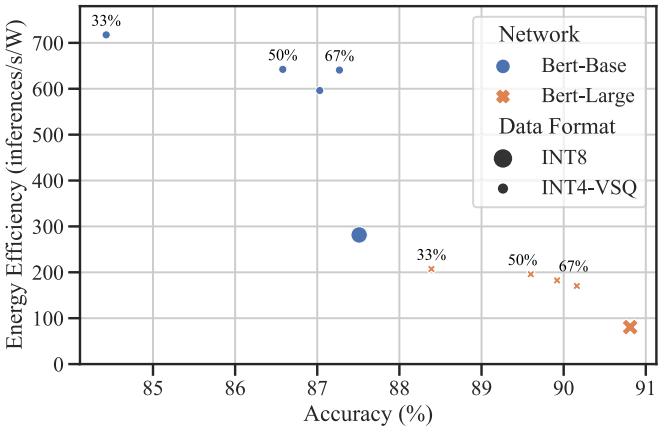


Fig. 15. Accuracy-energy trade-off for BERT with a sequence length of 384. Labeled percentages indicate weight densities after pruning.

in subsequent layers. For ResNet-50, INT4-VSQ achieves a 0.15% accuracy loss compared with the baseline, while INT4 shows a 0.98% accuracy drop.

E. Energy Optimizations

Software changes and hardware reconfigurability can be exploited to further improve accelerator energy efficiency. For example, while most transformer weight matrices exhibit little "natural" sparsity, weight sparsity can be increased by forcing a percentage of weights to zero during the fine-tuning process. Table VI shows the results of applying uniform per-tensor sparsity to BERT weights using magnitude-based pruning followed by retraining, similar to the approach in [65]. All accuracy and energy values are given as percent change compared with the INT4-VSQ baseline. The most aggressive sparsity (33% dense weights) results in a 20%–30% improvement in energy efficiency, though accuracy is reduced by 1.0%–2.3% due to the loss of weight information. Higher density (67% dense weights) minimizes accuracy loss, but energy efficiency gains are limited. The use of sparse training, network size, and data format can be used to generate an energy-accuracy Pareto curve (see Fig. 15).

Energy efficiency can also be improved via sparsity-aware operand ordering. Results presented thus far have mapped the $\boldsymbol{A}$ matrix into the $\boldsymbol{A}$ buffer and the $\boldsymbol{B}$ matrix into the $\boldsymbol{B}$ buffer. However, the MM transpose identity allows these operands to be reversed

$$\boldsymbol{A}\boldsymbol{B} = \boldsymbol{C} \implies \boldsymbol{B}^T\boldsymbol{A}^T = \boldsymbol{C}^T. \tag{3}$$

Thus, $\boldsymbol{B}^T$ can be mapped to the $\boldsymbol{A}$ buffer and $\boldsymbol{A}^T$ to the $\boldsymbol{B}$ buffer to compute a transposed version of the same result. This can save energy, because the toggle rate of the datapath is more sensitive to the sparsity of the $\boldsymbol{B}$ input vectors, since these update every cycle, while the $\boldsymbol{A}$ input is held constant for 16 cycles in the local $\boldsymbol{A}$ stationary dataflow. Accordingly, when $\boldsymbol{A}$ is sparser than $\boldsymbol{B}$, the transposed multiplication often costs less energy than the original. (Transpose operations are performed offline.) For INT8 BERT-Base with a sequence length of 128, layer 0 BMM2 has an $\boldsymbol{A}$ density of 98% and a $\boldsymbol{B}$ density of 64%, and energy is reduced 21% by performing the transposed multiplication with swapped operands. Additional hardware support would be required for full flexibility in operand ordering, but this preliminary result shows the promise of this simple technique when $\boldsymbol{A}$ and $\boldsymbol{B}$ have different densities.

VI. CONCLUSION

Energy efficiency is critical for DNN inference on transformer networks, but many prior energy-efficient DNN inference accelerators sacrifice task accuracy. This work presents an accelerator that uses VSQ to realize INT4 arithmetic for DNN inference with minimal accuracy loss on transformers. Wide vector operations and efficient dataflows that maximize reuse and minimize SRAM reads result in a prototype system that can achieve 95.6 TOPS/W in benchmarking and up to 1734 inferences/s/W running BERT on NLP workloads with $<$1.1% accuracy loss. This energy efficiency is 5.8$\times$ greater than the previous state of the art, setting a new benchmark for efficient DNN inference.

ACKNOWLEDGMENT

The authors would like to thank Zhuang Wu and Ashish Dash for bring-up support.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[3] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Eur. Conf. Comput. Vis.*, Aug. 2020, pp. 405–421.
[4] M. Bojarski et al., "Explaining how a deep neural network trained with end-to-end learning steers a car," Apr. 2017, *arxiv:1704.07911*.
[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017.
[6] A. Vaswani et al., "Attention is all you need," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2017, pp. 1–11.
[7] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training multi-billion parameter language models using model parallelism," 2019, *arXiv:1909.08053*.
[8] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2021, pp. 1–22.
[9] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.
[10] C. Sakr, Y. Kim, and N. Shanbhag, "Analytical guarantees on numerical precision of deep neural networks," in *Proc. 34th Int. Conf. Mach. Learn.*, Jul. 2017, pp. 3007–3016.
[11] *Papers With Code—ImageNet Benchmark (Image Classification)*. Accessed: Aug. 25, 2022. [Online]. Available: https://paperswithcode.com/sota/image-classification-on-imagenet

[12] A. Agrawal et al., "A 7 nm 4-core AI chip with 25.6 TFLOPS hybrid FP8 training, 102.4 TOPS INT4 inference and workload-aware throttling," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 144–146.

[13] A. Parashar et al., "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 27–40.

[14] Y. Zhao et al., "I-FlatCam: A 253 FPS, 91.49 μJ/frame ultra-compact intelligent lensless camera for real-time and efficient eye tracking in VR/AR," in *Proc. Int. Symp. VLSI Technol. Circuits (VLSI)*, Jun. 2022, pp. 108–109.

[15] K. Ueyoshi et al., "DIANA: An end-to-end energy-efficient digital and analog hybrid neural network SoC," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 256–258.

[16] Y. Wang et al., "A 28nm 27.5 TOPS/W approximate-computing-based transformer processor with asymptotic sparsity speculating and out-of-order computing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 464–466.

[17] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," 2020, *arXiv:2004.09602*.

[18] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 6105–6114.

[19] M. Tan and Q. Le, "EfficientNetv2: Smaller models and faster training," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jul. 2021, pp. 10096–10106.

[20] *NVIDIA Data Center Deep Learning Product Performance*. Accessed: Oct. 27, 2022. [Online]. Available: https://developer.nvidia.com/deep-learning-performance-training-inference

[21] S. Dai et al., "VS-Quant: Per-vector scaled quantization for accurate low-precision neural network inference," in *Proc. Conf. Mach. Learn. Syst. (MLSys)*, Apr. 2021, pp. 873–884.

[22] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, Jul. 2021, pp. 10347–10357.

[23] J.-S. Park et al., "A multi-mode 8K-MAC HW-utilization-aware neural processing unit with a unified multi-precision datapath in 4 nm flagship mobile SoC," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 246–248.

[24] K. Hirose et al., "Hiddenite: 4K-PE hidden network inference 4D-tensor engine exploiting on-chip model construction achieving 34.8-to-16.0 TOPS/W for CIFAR-100 and ImageNet," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 252–254.

[25] P. Deaville, B. Zhang, and N. Verma, "A 22 nm 128-kb MRAM row/column-parallel in-memory computing macro with memory-resistance boosting and multi-column ADC readout," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Jun. 2022, pp. 268–269.

[26] H. Mo et al., "A 28 nm 12.1TOPS/W dual-mode CNN processor using effective-weight-based convolution and error-compensation-based prediction," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 146–148.

[27] C.-H. Lin et al., "A 3.4-to-13.3 TOPS/W 3.6TOPS dual-core deep-learning accelerator for versatile AI applications in 7 nm 5G smartphone SoC," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 134–136.

[28] Y. S. Shao et al., "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proc. Int. Symp. Microarchitecture (MICRO)*, Oct. 2019, pp. 14–27.

[29] D. Wang, C.-T. Lin, G. K. Chen, P. Knag, R. K. Krishnamurthy, and M. Seok, "DIMC: 2219 TOPS/W 2569F2/b digital in-memory computing macro in 28 nm based on approximate arithmetic hardware," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 266–268.

[30] J.-O. Seo, M. Seok, and S. Cho, "ARCHON: A 332.7 TOPS/W 5b variation-tolerant analog CNN processor featuring analog neuronal computation unit and analog memory," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 258–260.

[31] J. Lee et al., "A 13.7 TFLOPS/W floating-point DNN processor using heterogeneous computing architecture with exponent-computing-in-memory," in *Proc. Int. Symp. VLSI Circuits (VLSI)*, Jun. 2021, pp. 1–2.

[32] S. Yin et al., "PIMCA: A 3.4-Mb programmable in-memory computing accelerator in 28 nm for on-chip DNN inference," in *Proc. Int. Symp. VLSI Circuits (VLSI)*, Jun. 2021, pp. 1–2.

[33] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," 2019, *arXiv:1910.01108*.

[34] Q. Guo, X. Qiu, P. Liu, Y. Shao, X. Xue, and Z. Zhang, "Star-transformer," 2019, *arXiv:1902.09113*.

[35] A. Nagarajan, S. Sen, J. R. Stevens, and A. Raghunathan, "AxFormer: Accuracy-driven approximation of transformers for faster, smaller and more accurate NLP models," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–8.

[36] T. Ham et al., "A^3: Accelerating attention mechanisms in neural networks with approximation," in *Proc. Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2020, pp. 328–341.

[37] T. J. Ham et al., "ELSA: Hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2021, pp. 692–705.

[38] J. Yu et al., "NN-LUT: Neural approximation of non-linear operations for efficient transformer inference," in *Proc. Design Autom. Conf. (DAC)*, Jul. 2022, pp. 577–582.

[39] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8BERT: Quantized 8Bit BERT," in *Proc. 5th Workshop Energy Efficient Mach. Learn. Cognit. Comput. (EMC2-NIPS)*, Dec. 2019, pp. 36–39.

[40] Y. Lin, Y. Li, T. Liu, T. Xiao, T. Liu, and J. Zhu, "Towards fully 8-bit integer inference for the transformer model," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3759–3765.

[41] H. Wang, Z. Zhang, and S. Han, "SpAtten: Efficient sparse attention architecture with cascade token and head pruning," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2021, pp. 97–110.

[42] S. Kim et al., "Learned token pruning for transformers," in *Proc. ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2022, pp. 784–794.

[43] Z. Li, S. Ghodrati, A. Yazdanbakhsh, H. Esmaeilzadeh, and M. Kang, "Accelerating attention through gradient-based learned runtime pruning," in *Proc. 49th Annu. Int. Symp. Comput. Archit.*, Jun. 2022, pp. 902–915.

[44] F. Tu et al., "A 28 nm 15.59 μJ/token full-digital bitline-transpose CIM-based sparse transformer accelerator with pipeline/parallel reconfigurable modes," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 466–468.

[45] T. Tambe et al., "EdgeBERT: Sentence-level energy optimizations for latency-aware multi-task NLP inference," in *Proc. Int. Symp. Microarchitecture (MICRO)*, Oct. 2021, pp. 830–844.

[46] S. Lu, M. Wang, S. Liang, J. Lin, and Z. Wang, "Hardware accelerator for multi-head attention and position-wise feed-forward in the transformer," in *Proc. IEEE 33rd Int. System-on-Chip Conf. (SOCC)*, Sep. 2020, pp. 84–89.

[47] R. Venkatesan et al., "MAGNet: A modular accelerator generator for neural networks," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.

[48] M. Pellauer et al., "Buffets: An efficient and composable storage idiom for explicit decoupled data orchestration," in *Proc. Int. Conf. Architectural Support Program. Lang. Operation Syst. (ASPLOS)*, Apr. 2019, pp. 137–151.

[49] J. R. Stevens, R. Venkatesan, S. Dai, B. Khailany, and A. Raghunathan, "Softermax: Hardware/software co-design of an efficient softmax for transformers," in *Proc. 58th ACM/IEEE Design Automat. Conf. (DAC)*, Dec. 2021, pp. 469–474.

[50] Y. Lee et al., "An agile approach to building RISC-V microprocessors," *IEEE Micro*, vol. 36, no. 2, pp. 8–20, Mar. 2016.

[51] B. Khailany et al., "A modular digital VLSI flow for high-productivity SoC design," in *Proc. Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[52] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan, "Computing in memory with spin-transfer torque magnetic RAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 470–483, Mar. 2018.

[53] F. Gao, G. Tziantzioulis, and D. Wentzlaff, "ComputeDRAM: In-memory compute using off-the-shelf DRAMs," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 100–113.

[54] N. Verma et al., "In-memory computing: Advances and prospects," *IEEE Solid StateCircuits Mag.*, vol. 11, no. 3, pp. 43–55, Aug. 2019.

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

[56] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.

[57] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2019, pp. 8026–8037.
[58] B. Keller et al., "A 17–95.6 TOPS/W deep learning inference accelerator with per-vector scaled 4-bit quantization for transformers in 5 nm," in *Proc. Int. Symp. VLSI Technol. Circuits (VLSI)*, Jun. 2022, pp. 16–17.
[59] B. Zimmer et al., "A 0.32–128TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE J. Solid-State Circuits (JSSC)*, vol. 55, no. 4, pp. 920–932, Apr. 2020.
[60] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," 2016, *arXiv:1606.05250*.
[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2015, pp. 1–22.
[62] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.
[63] C. Sakr et al., "Optimal clipping and magnitude-aware differentiation for improved quantization-aware training," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jul. 2022, pp. 19123–19138.
[64] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," 2018, *arXiv:1805.06085*.
[65] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Dec. 2015, pp. 1135–1143.

Ben Keller (Member, IEEE) received the B.S. degree in engineering from the Harvey Mudd College, Claremont, CA, USA, in 2010, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 2015 and 2017, respectively.

In 2017, he joined the ASIC&VLSI Research Group, NVIDIA, Inc., Santa Clara, CA, USA, where he works as a Senior Research Scientist. His research interests include digital clocking and synchronization techniques, hardware design productivity, and energy-efficient neural network inference.

Dr. Keller has served on the Design, Automation and Test in Europe (DATE) Technical Program Committee.

Rangharajan Venkatesan (Member, IEEE) received the B.Tech. degree in electronics and communication engineering from the Indian Institute of Technology Roorkee, Roorkee, India, in 2009, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2014.

He is currently a Senior Research Scientist with NVIDIA, Inc., Santa Clara, CA, USA. His research interests include machine learning accelerators, high-level synthesis, variation-tolerant design methodologies, spintronics, and approximate computing.

Dr. Venkatesan has been a member of the Technical Program Committees of several leading IEEE conferences, including the International Solid-State Circuits Conference (ISSCC), the Design Automation Conference (DAC), and the International Symposium on Low Power Electronics and Design (ISLPED). He received the Best Paper Award for the paper on scalable deep learning (DL) accelerator design at the International Symposium on Microarchitecture (MICRO) in 2019. His work on spintronic memory design was recognized with the Best Paper Award at ISLPED in 2012 and the Best Paper Nomination at the Design, Automation and Test in Europe Conference (DATE) in 2017. His work on FinFET-based SRAM also received the Best Paper Nomination at DATE in 2015.

Steve Dai received the B.S. degree in electrical engineering from the University of California at Los Angeles, Los Angeles, CA, USA, in 2011, the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2013, and the Ph.D. degree in electrical and computer engineering from Cornell University, Ithaca, NY, USA, in 2019.

He is currently a Senior Research Scientist with the ASIC&VLSI Research Group, NVIDIA, Inc., Santa Clara, CA, USA. His research interests include energy-efficient deep learning (DL) acceleration and GPU-accelerated and machine-learning-assisted electronic design automation.

Stephen G. Tell (Member, IEEE) received the B.S.E. degree in electrical engineering from Duke University, Durham, NC, USA, in 1989, and the M.S. degree in computer science from the University of North Carolina, Chapel Hill, NC, USA, in 1991.

He worked on parallel graphics systems and high-speed signaling as a Senior Research Associate with the University of North Carolina from 1991 to 1999. In 1999, he joined Chip2Chip Inc., Milpitas, CA, USA (later renamed Velio, Inc.) to develop circuits and control systems for high-speed SerDes products. This work continued at Rambus, Sunnyvale, CA, USA, where he designed the logic for a SerDes with the lowest energy per bit demonstrated up to that time. In 2009, he joined NVIDIA, Inc., Durham, CA, USA, as a Founding Member of the Circuits Research Group, where he works as a Senior Research Scientist. He holds more than 20 patents. His current research interests include custom circuit design and the surrounding test and control logic for intra- and inter-chip communication.

Brian Zimmer (Member, IEEE) received the B.S. degree in electrical engineering from the University of California at Davis, Davis, CA, USA, in 2010, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 2012 and 2015, respectively.

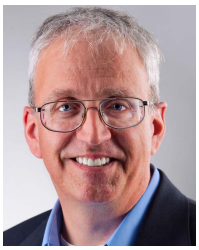
He is currently a Senior Research Scientist with the Circuits Research Group, NVIDIA, Inc., Santa Clara, CA, USA. His research interests include soft error resilience, chip-to-chip interconnects, energy-efficient digital design, low-voltage SRAM design, machine learning accelerators, productive design methodologies, and variation tolerance.

Dr. Zimmer has served on the Technical Program Committees of the Custom Integrated Circuits Conference (CICC) and the Symposium on VLSI Technology and Circuits (VLSI).

Charbel Sakr received the B.E. degree from the American University of Beirut, Beirut, Lebanon, in 2015, and the M.S. and Ph.D. degrees from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2017 and 2021, respectively.

He is currently a Research Scientist with the ASIC&VLSI Research Group, NVIDIA, Inc., Santa Clara, CA, USA. His research interests include resource-constrained machine learning, with a particular focus on analysis and implementation of reduced precision models and algorithms and their co-design with machine learning accelerator hardware.

Dr. Sakr received the Rambus Fellowship from the University of Illinois at Urbana-Champaign from 2018 to 2020 and the Best in Session Award at Techcon 2017.



William J. Dally (Fellow, IEEE) is currently Chief Scientist and Senior Vice President of Research at NVIDIA, Inc., Santa Clara, CA, USA, and an Adjunct Professor and Former Chair of Computer Science at Stanford University, Stanford. He is currently working on developing hardware and software to accelerate demanding applications, including machine learning, bioinformatics, and logical inference. He has a history of designing innovative and efficient experimental computing systems. At Bell Labs, Murray Hill, NJ, USA, he contributed to the BELLMAC32 microprocessor and designed the microprogrammable accelerator for rapid simulation (MARS) hardware accelerator. At the California Institute of Technology, Pasadena, CA, USA, he designed the MOSSIM Simulation Engine and the Torus Routing Chip that pioneered wormhole routing and virtual-channel flow control. At the Massachusetts Institute of Technology, Cambridge, MA, USA, his group built the J-Machine and the M-Machine, experimental parallel computer systems that pioneered the separation of mechanisms from programming models and demonstrated very low overhead synchronization and communication mechanisms. At Stanford University, his group has developed the Imagine processor, which introduced the concepts of stream processing and partitioned register organizations, the Merrimac supercomputer, which led to GPU computing, and the efficient low-power microprocessor (ELM). He also leads projects on computer architecture, network architecture, circuit design, and programming systems. He has authored or coauthored over 250 articles in these areas and authored the textbooks *Digital Design: A Systems Approach* (Cambridge University Press, 2012), *Digital Systems Engineering* (Cambridge University Press, 1998), and *Principles and Practices of Interconnection Networks* (Morgan Kaufmann Publishers, 2004). He holds over 160 issued patents.

Dr. Dally is a member of the National Academy of Engineering and a fellow of the Association for Computing Machinery (ACM) and the American Academy of Arts and Sciences. He received the ACM Eckert-Mauchly Award, the IEEE Seymour Cray Award, the ACM Maurice Wilkes Award, the IEEE-Computer Society (IEEE-CS) Charles Babbage Award, and the Information Processing Society of Japan (IPSJ) Funai Achievement Award.

C. Thomas Gray (Senior Member, IEEE) received the B.S. degree in computer science and mathematics from the Mississippi College, Clinton, MS, USA, in 1988, and the M.S. and Ph.D. degrees in computer engineering from North Carolina State University, Raleigh, NC, USA, in 1990 and 1993, respectively.

From 1993 to 1998, he was an Advisory Engineer with IBM, Research Triangle Park, NC, USA, working on transceiver design for communication systems. From 1998 to 2004, he was a Senior Staff Design Engineer with the Analog/Mixed Signal Design Group, Cadence Design Systems, San Jose, CA, USA, working on SerDes system architecture. From 2004 to 2010, he was a Consultant Design Engineer with Artisan/ARM, Cary, NC, USA and a Technical Lead of SerDes architecture and design. In 2010, he joined Nethra Imaging, Cary, NC, USA as a System Architect. His work experience includes digital signal processing design and CMOS implementation of DSP blocks, and high-speed serial link communication systems, architectures, and implementation. In 2011, he joined NVIDIA, Inc., Durham, NC, USA, where he is currently the Senior Director of circuit research, leading activities related to high-speed signaling, photonics, security circuits, low-energy and resilient memories, circuits for machine learning, and variation-tolerant clocking and power delivery.

Brucek Khailany (Senior Member, IEEE) received the B.S.E. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 1997, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2003.

From 2004 to 2009, he was a Co-Founder and a Principal Architect at Stream Processors, Inc. (SPI), Sunnyvale, CA, USA, where he led research and development activities related to parallel processor architectures. In 2009, he joined NVIDIA, Inc., Austin, TX, USA, where is currently the Senior Director of the ASIC&VLSI Research Group, leading research into innovative design methodologies for integrated circuit (IC) development, machine learning (ML) and GPU-assisted electronic design automation (EDA) algorithms, and energy-efficient ML accelerators. Over 13 years at NVIDIA, he has contributed to many projects in research and product groups spanning computer architecture and VLSI design.