A 17–95.6 TOPS/W Deep Learning Inference Accelerator with Per-Vector Scaled 4-bit Quantization for Transformers in 5nm

Ben Keller*[1], Rangharajan Venkatesan*[1], Steve Dai[1], Stephen G. Tell[2],
Brian Zimmer[1], William J. Dally[1], C. Thomas Gray[2], Brucek Khailany[3]
*Equally Credited Authors; [1]NVIDIA, Santa Clara, CA, USA; [2]NVIDIA, Durham, NC, USA; [3]NVIDIA, Austin, TX, USA
Email: benk@nvidia.com, rangharajanv@nvidia.com

Abstract

We present a deep neural network (DNN) accelerator designed for efficient execution of transformer-based DNNs, which have become ubiquitous for natural language processing tasks. DNN inference accelerators often employ specialized hardware techniques such as reduced precision to improve energy efficiency, but many of these techniques result in catastrophic accuracy loss on transformers. The proposed accelerator supports per-vector scaled quantization and approximate softmax to enable the use of 4-bit arithmetic with little accuracy loss. The 5nm prototype achieves 95.6 TOPS/W in benchmarking and 1711 inferences/s/W with only 0.7% accuracy loss on BERT, demonstrating a practical accelerator design for energy-efficient inference with transformers.

Keywords: DNN inference accelerator, BERT, transformers.

Introduction

Deep neural networks (DNNs) using attention-based transformer architectures have seen significant interest due to their superior performance in natural language processing [1] and vision tasks [2]. These trends suggest that transformers will become an increasingly common workload across a wide range of applications, making them an important target for hardware specialization to improve performance and energy efficiency. We present a DNN inference accelerator that includes specialized capabilities targeting transformers, enabling high energy efficiency while maintaining high task accuracy.

Accelerating Transformers

Reducing arithmetic precision is a common technique to improve energy efficiency on DNN workloads [3], as it reduces the cost of both multiply-accumulate (MAC) operations and data movement. However, reduced-precision math introduces quantization error, which can cause loss of task accuracy. Traditional quantization techniques that enable 4b math on small networks fail catastrophically when applied to more sophisticated transformer networks, and attention layers in transformers differ significantly from convolution layers in their shape and types of computations (see Fig.1).

The inference accelerator presented in this work uses per-vector scaled quantization (VSQ) to enable energy-efficient 4b math while maintaining task accuracy on transformers [4]. In addition to applying a coarse-grained scale factor for each output column, VSQ applies a scale factor at vector granularity within each input matrix (see Fig.2). When input vectors are multiplied, their 8b scale factors are multiplied to scale the result. VSQ enables low-precision arithmetic with reduced quantization error and minimal hardware overhead (see Fig.3).

Transformers include non-linear operations such as softmax and GELU, which are expensive in hardware. This work implements a hardware-friendly softmax approximation that uses base 2 instead of base *e*, low-precision fixed-point data formats, and online normalization to reduce data movement [5], greatly reducing hardware cost with minimal impact on accuracy (see Fig.4). GELU is replaced with the simpler ReLU operation.

Deep Learning Inference Accelerator

Fig.5 shows a block diagram of the accelerator, which contains 16 vector lanes, each of which implements independent 8b and 4b datapaths. Each lane performs a 64-element (32-element for 8b) multiply followed by an accumulating sum each cycle, as well as a scale factor multiplication to implement VSQ. The accelerator contains 132KB of SRAM storage for input matrices A and B. After accumulation is complete, a post-processing unit (PPU) optionally performs operations such as ReLU, approximate softmax, bias addition, and scaling. Final results are stored in an 8.5KB output memory.

The accelerator minimizes expensive SRAM reads by employing an output-stationary, local-A-stationary dataflow [6] (see Fig.6). A-matrix inputs are read out of the A-buffer once every 16 cycles and stored in a register for temporal reuse. B-matrix inputs are read once each cycle and reused across the 16 lanes. 24b partial sums are temporally accumulated in a 16-entry latch array before sending the completed sum to the PPU. This process is repeated to compute the output matrix by reusing A-matrix and B-matrix inputs. The accelerator can compute matrix multiply, convolution, and fully connected layers of different sizes by configuring the SRAM address generators. Buffer managers enable the pipelining of computation with the streaming of input data. The accelerator could be tiled spatially to compose a larger system.

Measurement Results

The 0.15mm^2 accelerator is fabricated in a TSMC 5nm process. Fig.7 shows the measured energy efficiency of the accelerator under different operating conditions and input densities. The system achieves 95.6 TOPS/W with 50%-dense 4b input matrices and VSQ enabled. VSQ imposes a 0.8% energy efficiency overhead compared to 4b math with per-matrix scaling with 50% non-zero inputs at 0.67V. Table 1 shows achieved performance, energy efficiency, and task accuracy on a variety of workloads. Quantization-aware fine-tuning is applied to pretrained FP32 weights, enabling ≤1.1% accuracy loss on BERT when VSQ is enabled. Without VSQ, 4b inference for transformers results in unacceptable accuracy loss even when retraining techniques are applied. The system achieves 1711 inferences/s/W running BERT-Base with a sequence length of 128 on SQuAD. Table 2 presents a comparison with prior work. Fig.8 shows an annotated die micrograph.

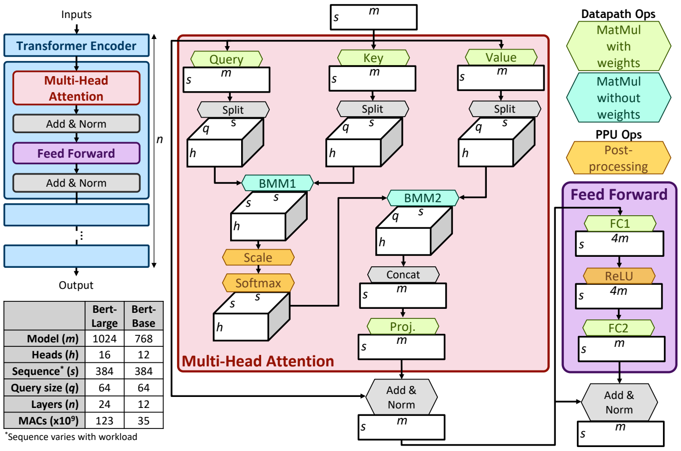
Conclusion

The presented system achieves 2.2-5.8X better energy efficiency and 4-16X better area efficiency than prior work. Unlike many prior proposals to improve energy efficiency, VSQ and approximate softmax allow the accelerator to maintain task accuracy on cutting-edge DNN workloads.

References

[1] A. Vaswani *et al.*, *NeurIPS*, 2017. [2] A. Dosovitskiy *et al.*, *ICLR*, 2021. [3] A. Agrawal *et al.*, *ISSCC*, 2021. [4] S. Dai *et al.*, *MLSys*, 2021. [5] J. Stevens *et al.*, *DAC*, 2021. [6] R. Venkatesan *et al.*, *ICCAD*, 2019. [7] H. Mo *et al.*, *ISSCC*, 2021. [8] J-S. Park *et al.*, *ISSCC*, 2021. [9] C. Lin *et al.*, *ISSCC*, 2020.

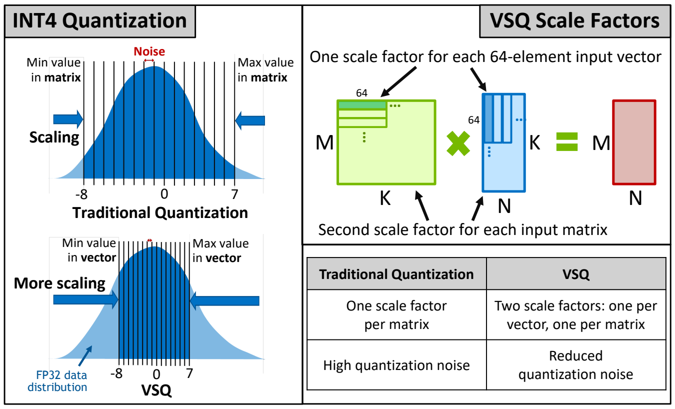
978-1-6654-9772-5/22/$31.00 ©2022 IEEE



	Bert-Large	Bert-Base
Model (*m*)	1024	768
Heads (*h*)	16	12
Sequence[*] (*s*)	384	384
Query size (*q*)	64	64
Layers (*n*)	24	12
MACs (x10^9)	123	35

[*] Sequence varies with workload

Fig.1: Workload components of transformers.



Traditional Quantization	VSQ
One scale factor per matrix	Two scale factors: one per vector, one per matrix
High quantization noise	Reduced quantization noise

Fig.2: Per-vector scaled quantization (VSQ).

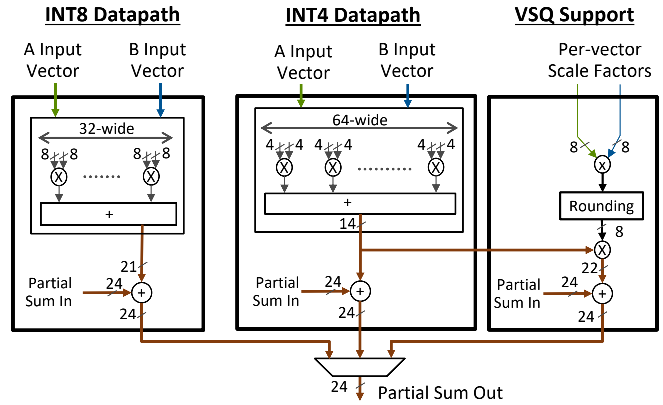
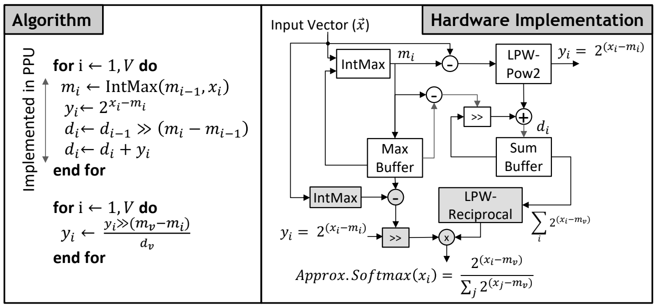


Fig.3: 8b/4b datapath with VSQ support.



```
for i ← 1, $V$ do
    $m_i \leftarrow \mathrm{IntMax}(m_{i-1}, x_i)$
    $y_i \leftarrow 2^{x_i - m_i}$
    $d_i \leftarrow d_{i-1} \gg (m_i - m_{i-1})$
    $d_i \leftarrow d_i + y_i$
end for

for i ← 1, $V$ do
    $y_i \leftarrow \frac{y_i \gg (m_v - m_i)}{d_v}$
end for
```

Fig.4: Approximate softmax implementation.

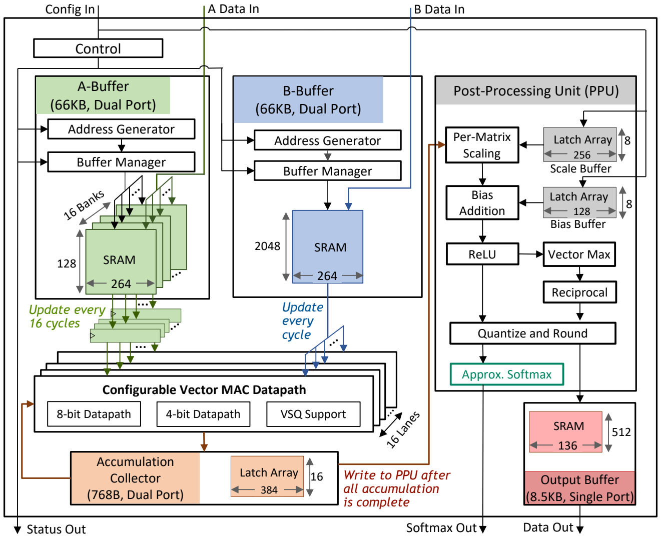
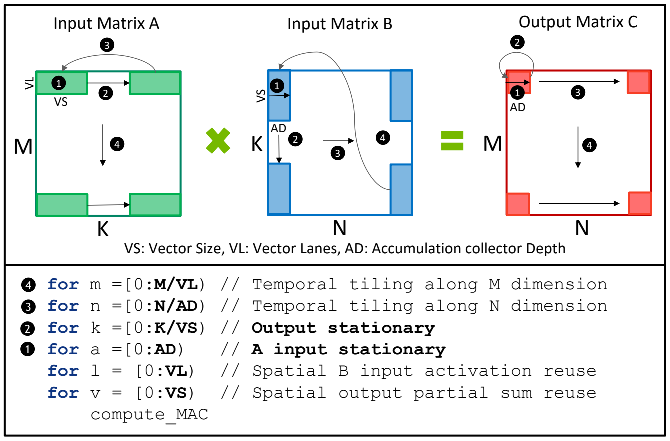


Fig.5: Accelerator block diagram.


Input Matrix A
Input Matrix B
Output Matrix C
VL
VS
M
K
N
AD
1
2
3
4
VS: Vector Size, VL: Vector Lanes, AD: Accumulation collector Depth

```
❹ for m =[0:M/VL) // Temporal tiling along M dimension
❸ for n =[0:N/AD) // Temporal tiling along N dimension
❷ for k =[0:K/VS) // Output stationary
❶ for a =[0:AD)   // A input stationary
  for l = [0:VL)  // Spatial B input activation reuse
  for v = [0:VS)  // Spatial output partial sum reuse
      compute_MAC
```

Fig.6: Workload mapping and data reuse.

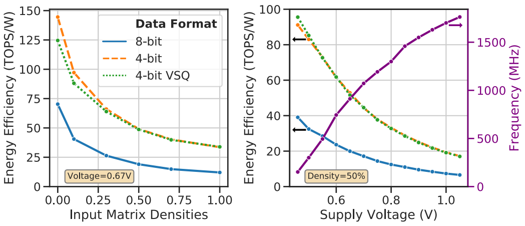


Fig.7: Chip measurements.

Table 1: Measured application performance at 0.67V.

Dataset, Task	SQuAD v1.1, Reading Comprehension												ImageNet, Image Classification					
Network	BERT-Base						BERT-Large						DeIT-Small			DeIT-Base		
Sequence Length	128			384			128			384			197			197		
Baseline FP32 Accuracy (%)	87.5			87.5			90.3			90.9			79.8			81.8		
Data Bitwidth (4V = 4b VSQ)	4b	4V	8b	4b	4V	8b	4b	4V	8b	4b	4V	8b	4b	4V	8b	4b	4V	8b
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
MAC Utilization (%)	-	98	99	-	98	99	-	98	99	-	98	99	-	94	96	-	97	98
Throughput (inferences/s)	-	88	45	-	28	14	-	25	13	-	8.1	4.1	-	210	108	-	56	28
Energy Eff. (inferences/s/W)	-	1.7k	745	-	539	235	-	502	216	-	160	69	-	3.5k	1.5k	-	1.0k	406

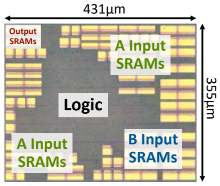


Fig.8: Die micrograph.

Table 2: Comparison to prior work.

	[3]	[7]	[8]	[9]	This work		
Process Technology	7nm	28nm	5nm	7nm	5nm		
Area (mm^2)	19.6	1.9	5.46	3.04	0.153		
Supply Voltage (V)	0.55 – 0.75	0.6 – 0.9	0.55 – 0.9	0.58 – 0.83	0.46 – 1.05		
Frequency (MHz)	1000 – 1600	100 – 470	332 – 1196	290 – 880	152 – 1760		
On-Chip SRAM (KB)	8192	206	3072	2176	141		
Data Formats	INT2/4, FP8/16/32	INT8	INT8, INT16	INT8/16, FP16	INT4	INT4 VSQ	INT8
Performance (TOPS)	102.4 (4b, 0.75V)	1.43 (8b, 0.9V)	14.7 (8b, 0.9V)	3.6 (8b, 0.83V)	3.6 (1.05V)	3.6 (1.05V)	1.8 (1.05V)
Energy Efficiency (TOPS/W)	16.5[*] (4b, 0.55V)	17.5[*] (8b, 0.6V)	13.6[*] (8b, 0.6V)	6.8[*] (8b, 0.58V)	**91.1**[†] (0.46V)	**95.6**[†] (0.46V)	**39.1**[†] (0.46V)
Area Efficiency (TOPS/mm^2)	5.22 (4b, 0.75V)	0.75 (8b, 0.9V)	2.69 (8b, 0.9V)	1.2 (8b, 0.83V)	**23.3** (1.05V)	**23.3** (1.05V)	**11.7** (1.05V)

[*] Input densities not reported. [†] Measured with 50% non-zero input densities. Includes estimated leakage power.