TRIP: An Ultra-Low Latency, TeraOps/s Reconfigurable Inference Processor for Real-Time Multi-Layer Perceptrons

Ahmed Sanaullah Chen Yang *Yuri Alexeev *Kazutomo Yoshii Martin C. Herbordt
Boston University, Boston, MA *Argonne National Laboratory, Lemont, IL

I. Introduction

Multi-Layer Perceptron (MLP) is one of the most commonly deployed Deep Neural Network, representing 61% of the workload in Google data centers [1]. MLPs have low arithmetic intensity which results in memory bottlenecks. To the best of our knowledge, the Google Tensor Processing Unit (TPU) [1], the state-of-the-art implementation of MLP inference, achieves ≈10% of the 92 TeraOps/s peak. TPU addresses the memory bound by processing multiple test vectors simultaneously to increase operations per weight byte loaded from DRAM. However, inference typically has hard response time deadlines and prefers latency over throughput [2]. Waiting for sufficient input vectors to get good performance is not feasible.

In this work, we have designed a quantized TeraOps/s Reconfigurable Inference Processor for MLPs (TRIP) on FPGAs that alleviates the memory bound by storing all weights on-chip and ensured performance is invariant of the input batch size. Through this approach, we not only guarantee stall-free data availability to pipelines during steady-state, but also reduce power consumption significantly by reducing DRAM access. For large databases that cannot directly fit on chip, Deep Compression relaxes memory footprint requirements with no effect on accuracy [3]. TRIP is deployed as a standalone device directly connected to data acquisition devices, as a co-processor where input vectors are supplied through OpenCL wrappers from the host machine, as well as in a cluster configuration where on-chip transceivers communicate between FPGAs [4]. By comparison, TPU can only be used in a co-processor configuration.

II. System Design

MLP based neural networks typically have asymmetric logical configurations, with average neurons per layer being orders of magnitude larger than the number of layers. Consequently, while relatively small delays for inter-layer function evaluations have negligible impact, addressing compute bottlenecks through high compute capability and sustained throughput for intra-layer computations is performance critical. This served as motivation for the TRIP architecture presented in Figure 1.

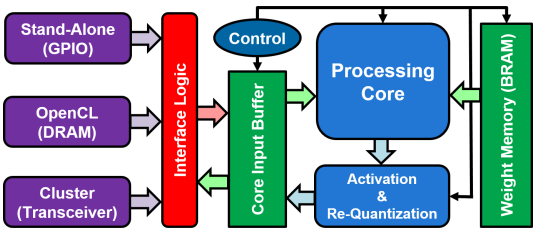

Fig. 1. TRIP Architecture Overview

1) **Processing Core**: The Processing Core implements up to 8192 - 8 bit integer multipliers in a $M \times N$ 2D array. We employ both DSP and ALM multipliers to ensure sufficient resources on medium-high end FPGAs. Each slice of M multipliers has an associated adder tree to enable scalar product evaluation. TRIP provides the flexibility of choosing array configurations that maximize processing core utilization. Resulting (signed) 32-bit N partial sums are accumulated into output buffers which are preloaded with bias vectors.
2) **Activation Pipeline**: The Activation pipeline performs non-linear operations on layer result vectors and re-quantizes them from 32-bit to a lower bit size for the next layer. TRIP currently supports ReLU activation ($x = max(0, x)$), implemented using an array of N MUXs.
3) **Interface Logic**: The Interface Logic structure depends on the TRIP deployment configuration. It operates concurrently with the compute hardware to mask the latency of fetching test vectors from DAQ devices, DRAM or transceivers.
4) **Control Module**: The control module coordinates flow of data based on the MLP model. Parameters for execution and computation patterns are initialized on-chip when the device is programmed, removing the need for streaming instructions from the host machine and enable stand-alone/cluster operation.

III. TPU Comparison

As we did not have access to TPU, we approximately quantify the inference latency bound that allows TPU to have sufficiently large input batch sizes required to outperform our design (Figure 2). The first generation of TPU has 64K MACs, 30GB/s off-chip bandwidth, and 700MHz operating frequency. On the other hand, TRIP is deployed with 8192 multipliers for stand-alone/cluster, and 4096 multipliers for co-processor designs. The latter is due to resource usage by

the OpenCL wrapper. Operating frequency is 200MHz. Input data fetch latency is assumed to be negligible. From the figure, we estimate that TRIP outperforms TPU for input batch sizes of less than 53 test vectors.

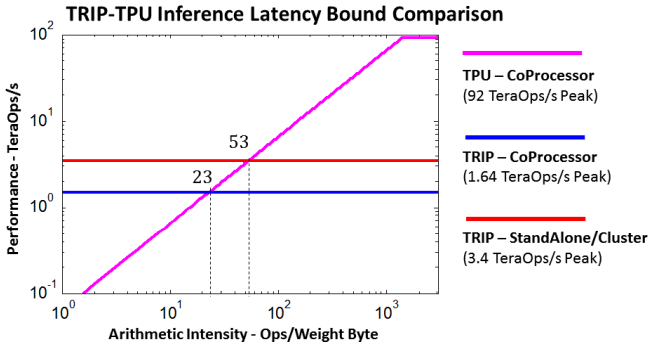

Fig. 2. TRIP-TPU Inference Latency Bound Comparison

IV. Results

A. Benchmark

We use the ECP-Candle-P3B1 Tumor Laterality (TL) and Clinical Grade of Tumors (CGT) benchmarks to test our system. The MLP model is composed of four layers with 3.3M neurons. Weights and biases are trained offline and quantized to 8 bits. For this dataset, TRIP Processing Core is deployed in a 256×32 configuration for stand-alone/cluster and 256×16 configuration for co-processors.

B. Device Specifications

We implement our design on a Nallatech 385A development board which hosts an Altera Arria10 10AX115N3F40E2SG FPGA. The device has 427,200 ALMs, 1,518 DSP blocks (2 18x18 integer multipliers per block) and 54,260Kb on-chip SRAM. The GPU design is implemented on NVIDIA Tesla K80m which has 49992 CUDA cores and 480 GB/s global memory bandwidth. CUDA matrix operations are performed using the cuBLAS library and compiled with CUDA 8.0.

C. FPGA Parameters

Table 1 lists the resource usage and operating frequency for our TRIP deployment configurations. Resource usage for transceivers is estimated based on the Novo-G# [4] router design.

TABLE I
TRIP Deployment based FPGA Parameters

Configuration	ALM	DSP	BRAM	Freq.(MHz)
Stand-Alone	313K	1,280	4MB	207.49
Co-Processor	251K	1,280	4.3MB	201.2
Cluster	315K	1,280	4MB	206

D. Quantization

We evaluated the impact of quantization through truncation on inference accuracy. The average classification error for TL and CGT datasets was 0.98% and 1.8% respectively with respect to floating point implementations.

E. Performance

Table 2 gives the performance comparison for GPU, TPU and TRIP for batch-less evaluation. Values are the best cases from TL and CGT results. Projections of TRIP on Stratix 10 are based on 4x more ALM multipliers, 4 extra DSP scalar product slices and 2x operating frequency. From the results, we see that TRIP is orders of magnitude faster than TPU and has better resource utilization.

TABLE II
ECP-Candle Performance Comparison for Single Input Vector

Architecture	M,N	Useful Op	Performance	Speedup
NVIDIA K80	-	-	0.02 TOps/s	1x
TPU	256,256	79%	0.05 TOps/s	2.5x
TRIP Arria 10 CoProc	256,16	91%	1.5 TOps/s	75x
TRIP Arria 10 Cluster	256,32	89%	3.0 TOps/s	150x
TRIP Stratix 10 CoProc	256,86	88%	15.5 TOps/s	775x
TRIP Stratix 10 Cluster	256,102	86%	18.0 TOps/s	900x

F. Power

TRIP consumes 32W on Arria 10 for the co-processor configuration, of which 30W is the static power. By comparison, TPU consumes 38-43W total depending on workload.

V. Impact

To the best of our knowledge, TRIP is the only TeraOps/s MLP inference engine for single inputs. Its deployment versatility and low power consumption makes it an ideal candidate for a numerous applications and configurations. Use of OpenCL reduces co-processor integration effort in legacy codes (TPU can only run with Tensorflow). Cluster configuration enables larger models to be evaluated by distributing layers to multiple devices. TRIP's reconfigurability allows hardware to adapt to the applications, maximizing utilizing of available compute resources and minimizing high latency memory accesses. TRIP is not constrained to a certain number of quantized bits. Based on the application, the size of weights can be reduced to further increase the size of Processing Core without significant impact on accuracy. Since TRIP is implemented using Off-The-Shelf FPGAs, utilizing newer technologies is simply changing design parameters and compiling for the new device(as opposed to spinning new silicon for ASICs).

References

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," *arXiv preprint arXiv:1704.04760*, 2017.

[2] D. A. Patterson, "Latency lags bandwith," *Communications of the ACM*, vol. 47, no. 10, pp. 71–75, 2004.

[3] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*. IEEE Press, 2016, pp. 243–254.

[4] A. D. George, M. C. Herbordt, H. Lam, A. G. Lawande, J. Sheng, and C. Yang, "Novo-g#: Large-scale reconfigurable computing with direct and programmable interconnects," in *High Performance Extreme Computing Conference (HPEC), 2016 IEEE*. IEEE, 2016, pp. 1–7.