T-MAC: Temporal Multiplication with Accumulation

Zhewen Pan, Di Wu, Joshua San Miguel
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
zhewen.pan@wisc.edu, di.wu@ece.wisc.edu, jsanmiguel@wisc.edu

***Abstract*—With the advance of Artificial Neural Networks, GEMM has become a dominant arithmetic operation in compute-intensive applications. Multipliers contribute to a major portion of area and power consumed by conventional systems. This work proposes T-MAC to improve the power and area efficiency of machine learning accelerators by exploiting a new dimension of computation reuse empowered by Hybrid Unary-Binary computing paradigm. Leveraging the seamless binary interface and internal unary computing kernel, T-MAC achieves higher hardware efficiency and better scalability compared against conventional binary hardware.**

I. Introduction

Customized machine learning accelerators have gained tremendous attention in both academia and industry. Prior works have well explored the architecture design space of binary design, with representative microarchitectures such as systolic array [1], SIMD [2], [3]. In the field of digital signal processing, prior works [4], [5] have also explored bit-serial architectures for machine learning accelerators. At another end of the spectrum, there are also works purposing fully streaming unary computing architectures (FSU) [6], with the most well-known sub-field being stochastic computing [7]–[12].

In this work, we purpose T-MAC, a high-efficiency Hybrid Unary-Binary (HUB) [13] architecture, which is essentially a look-up table using temporal signals to perform multiplication. Thanks to the unary computing paradigm, the hardware can be trimmed to be extremely simple and compact thus achieving high compute density and hardware efficiency. The bandwidth requirement of unary designs is naturally low because unary computing incurs an exponential latency penalty, which can be amortized by upscaling the compute array dimension. We expect that T-MAC to have better scalability under similar memory pressure. Moreover, since T-MAC operates upon temporal bitstreams, it is capable of early terminating at arbitrary cycles with deterministic accuracy loss. This property calls for careful application co-design to amplify the performance gain. With the specific hardware architecture in mind, we recognize that the way to program the hardware has a nontrivial performance impact. With naive programming, the performance gain could be severely diluted or even hurt performance [14]. Therefore we aim to co-design the dataflow with hardware.

II. Background

A. Fully Streaming Unary Computing Architecture

Prior works have proposed architectures fully based on bitstream computing [13]. The two most common bitstream computing paradigms are temporal [15] and stochastic computing [7]. In both works, the binary number with bit-width N is converted to its bitstream representation with length 2^N and exploits the opportunity of using hardware with low area footprint and power to perform various operations. The systematic view of unified bitstream computing scheme for both stochastic and temporal has also been studied in [16], where the set of computing kernels for both computing paradigms are extended to include addition and multiplication with higher accuracy.

B. Hybrid Unary-Binary Architecture

The closest resemblance of our proposed processing element architecture is the Hybrid Unary-Binary architectures [6], [8], [13], where the computation is done in both binary regime and bit-stream regime. For example, authors in [8] propose to only perform multiplication in stochastic computing, and addition is still performed in binary. Authors in [6] purposes partitioning the binary numbers into sub-blocks and performing unary computation on the shorter bit-streams to amortize the long latency. Authors in [13] propose using binary interface as T-MAC but internally weights are rate coded, whereas in T-MAC, weights are binary.

C. Bit-Serial Architecture

Prior works have also proposed bit-serial architectures [4], [5], [17], where the MSBs proceed through the computation pipeline before the LSBs, achieving flexible bit-width computation with a linear increase in latency but greater throughput. Bit-serial architectures do not support seamless early termination as the unary computing architectures.

III. Solution

A. Architecture

The proposed T-MAC, shown in Fig. 1, adopts a binary interface but converts inputs to temporal coding to look up the accumulator (ACC) value as the product. The accumulator for each weight is shared by all the inputs, denoted by *temporal sharing degree*, therefore achieving a smaller compute unit area for high efficiency. T-MAC processes multiple weights in parallel, denoted by *weight parallel degree*. Fig. 1 shows the T-MAC architecture with temporal sharing degree of i and weight parallel degree of w.

Inputs (I_i) and weights (W_w) are both stored as binary data. Each N-bit input is fed into a temporal converter (TC)

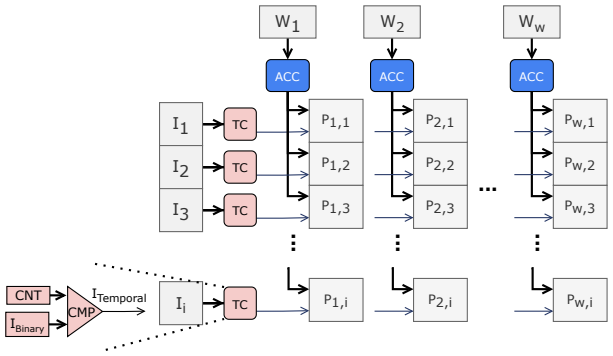

Fig. 1: T-MAC hardware. Adder tree for flexible dataflow is not shown for clarity. Thick arrows denote binary data and thin arrows denote unary bitstreams. Temporal bitstreams are shared across a row horizontally.

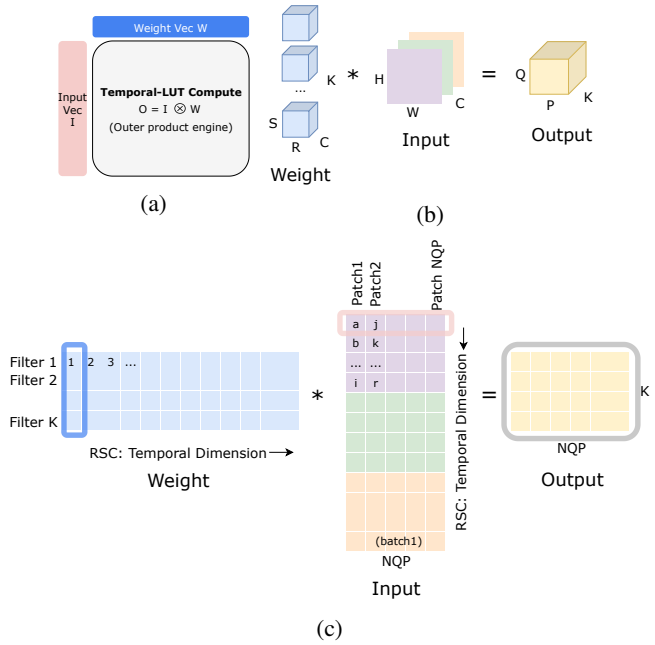

Fig. 2: Dataflow. a) Abstract hardware interface. b) Conv2d representation. c) Output stationary matmul representation. R and S represent filter width and height; W, H represent input width and height; P, Q represent output width and height; C and K represent input and output channel. Dataflow for fully connected is a special case of Conv2d where R, S, H, W, P, Q are set to 1.

which converts input to a temporal bitstream of length 2^N by comparing the input value against a counter. The output of TC acts as the write enable signal of a product register (P). The input data to the product register is the output of a weight accumulator, which accumulates the binary weight value in every cycle. At the end of cycle 2^N, the product register $P_{i,w}$ contains the value of $I_i \times W_w$. Therefore, T-MAC calculates the outer product on input and weight vectors.

B. Dataflow

To support clean compute tiling, we purpose output-stationary dataflow for T-MAC. As is shown in Fig. 2a, for N-bit inputs, *in every* 2^N *cycles*, T-MAC computes the outer product of an input vector I and a weight vector W and generate an output matrix O. The computation of 2d convolution, shown in 2b, can be represented as a single matrix multiplication shown in Fig 2c. The product of kernel size and channel, i.e. $R \times S \times C$, is mapped to temporal dimension. The product accumulation is done temporally.

The purposed output stationary dataflow reads inputs and weights every 2^N cycles and writes final output to memory every $R \times S \times C \times 2^N$ cycles, significantly reducing memory bandwidth pressure, compared against conventional binary designs. Further optimizations can be applied to reduce the latency overhead by splitting the workload between several T-MACs with spatial accumulation among them.

IV. Evaluation Methodology

We aim to develop an architecture simulation framework for performance evaluation. We sample the design space and feed configuration of workload, memory, schedule, and compute to the simulator. We leverage the existing UnarySim bitstream simulator [16] to perform accuracy profiling on the targeted workload. We generate ideal SRAM traces for the target dataflow and profile the trace to retrieve the bandwidth and latency in a memory contention-aware manner. Lastly, we synthesize the hardware RTL and model the memory using CACTI [18] to evaluate hardware energy and power efficiency.

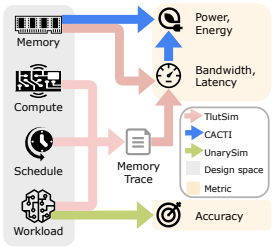

Fig. 3: Evaluation framework

V. Related Work

One of the bit-serial architectures that closely relate to our idea is the Pragmatic architecture [4]. To replace the costly binary multiplications, Pragmatic applies multi-cycle (linear) shifting operations, whereas T-MAC is unary (exponential). One upside of Pragmatic is that the accumulators for accumulating the shifted outputs, related to a single multiplication, can be also used as accumulators for partial sum, related to data scheduling. However, these accumulators cannot be shared, leading to a higher hardware cost. Authors in [6] purpose using HUB computing by converting binary data into parallel unary bitstreams, instead of serial bitstreams in T-MAC, posing exponential area and power overheads. However, it is limited to performing computations with a single input. T-MAC outperforms it by supporting 2-input multiplication.

REFERENCES

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-Datacenter Performance Analysis of a Tensor Processing Unit," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, p. 1–12, jun 2017.

[2] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *International Symposium on Computer Architecture*, 2016.

[3] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, "DaDianNao: A Machine-Learning Supercomputer," in *International Symposium on Microarchitecture*, 2014.

[4] J. Albericio, A. Delmás, P. Judd, S. Sharify, G. O'Leary, R. Genov, and A. Moshovos, "Bit-Pragmatic Deep Neural Network Computing," in *International Symposium on Microarchitecture*, 2017.

[5] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Network," in *International Symposium on Computer Architecture*, 2018.

[6] S. R. Faraji and K. Bazargan, "Hybrid Binary-Unary Hardware Accelerator," *IEEE Transactions on Computers*, vol. 69, no. 9, pp. 1308–1319, 2020.

[7] A. Ren, J. Li, Z. Li, C. Ding, X. Qian, Q. Qiu, B. Yuan, and Y. Wang, "SC-DCNN: Highly-Scalable Deep Convolutional Neural Network using Stochastic Computing," 2017.

[8] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "SCOPE: A Stochastic Computing Engine for DRAM-Based In-Situ Accelerator," in *International Symposium on Microarchitecture*, 2018.

[9] S. Gupta, M. Imani, J. Sim, A. Huang, F. Wu, M. H. Najafi, and T. Rosing, "SCRIMP: A General Stochastic Computing Architecture using ReRAM in-Memory Processing," in *Design, Automation Test in Europe Conference Exhibition*, 2020.

[10] T. Li, W. Romaszkan, S. Pamarti, and P. Gupta, "GEO: Generation and Execution Optimized Stochastic Computing Accelerator for Neural Networks," in *Design, Automation Test in Europe Conference Exhibition*, 2021, pp. 689–694.

[11] Z. Li, J. Li, A. Ren, R. Cai, C. Ding, X. Qian, J. Draper, B. Yuan, J. Tang, Q. Qiu, and Y. Wang, "HEIF: Highly Efficient Stochastic Computing-Based Inference Framework for Deep Neural Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1543–1556, 2019.

[12] W. Romaszkan, T. Li, T. Melton, S. Pamarti, and P. Gupta, "Acoustic: Accelerating convolutional neural networks through or-unipolar skipped stochastic computing," in *Design, Automation Test in Europe Conference Exhibition*, 2020, pp. 768–773.

[13] D. Wu and J. S. Miguel, "uSystolic: Byte-Crawling Unary Systolic Array," in *International Symposium on High-Performance Computer Architecture (HPCA)*, 2022.

[14] Q. Huang, M. Kang, G. Dinh, T. Norell, A. Kalaiah, J. Demmel, J. Wawrzynek, and Y. S. Shao, "CoSA: Scheduling by Constrained Optimization for Spatial Accelerators," *International Symposium on Computer Architecture*, 2021.

[15] A. Madhavan, T. Sherwood, and D. Strukov, "Race Logic: A Hardware Acceleration for Dynamic Programming Algorithms," in *International Symposium on Computer Architecuture*, 2014.

[16] D. Wu, J. Li, R. Yin, H. Hsiao, Y. Kim, and J. S. Miguel, "uGEMM: Unary Computing for GEMM Applications," *IEEE Micro*, 2021.

[17] K. E. Batcher, "Bit-Serial Parallel Processing Systems," *IEEE Trans. Comput.*, vol. 31, no. 5, p. 377–384, May 1982. [Online]. Available: https://doi.org/10.1109/TC.1982.1676015

[18] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A Tool to Model Large Caches," 2009.