EFFORT: Enhancing Energy Efficiency and Error Resilience of a Near-Threshold Tensor Processing Unit

Noel Daniel Gundi, Tahmoures Shabanian, Prabal Basu, Pramesh Pandey, Sanghamitra Roy,
Koushik Chakraborty, Zhen Zhang
BRIDGE Lab, Electrical and Computer Engineering, Utah State University
{noeldaniel, tahmoures, prabalb, pandey.pramesh1}@aggiemail.usu.edu,
{sanghamitra.roy, koushik.chakraborty, zhen.zhang}@usu.edu

Abstract— Modern deep neural network (DNN) applications demand a remarkable processing throughput usually unmet by traditional Von Neumann architectures. Consequently, hardware accelerators, comprising a sea of multiplier and accumulate (MAC) units, have recently gained prominence in accelerating DNN inference engine. For example, Tensor Processing Units (TPU) account for a lion's share of Google's datacenter inference operations. The proliferation of real-time DNN predictions is accompanied with a tremendous energy budget. In quest of trimming the energy footprint of DNN accelerators, we propose EFFORT—an energy optimized, yet high performance TPU architecture, operating at the Near-Threshold Computing (NTC) region. EFFORT promotes a better-than-worst-case design by operating the NTC TPU at a substantially high frequency while keeping the voltage at the NTC nominal value. In order to tackle the timing errors due to such aggressive operation, we employ an opportunistic error mitigation strategy. Additionally, we implement an in-situ clock gating architecture, drastically reducing the MACs' dynamic power consumption. Compared to a cutting-edge error mitigation technique for TPUs, EFFORT enables up to $2.5\times$ better performance at NTC with only 2% average accuracy drop across 3 out of 4 DNN datasets.

I. Introduction

Advancements in artificial intelligence have entered a new realm owing to the development of domain specific architectures dedicated to neural networks (NN) processing. Tensor Processing Unit (TPU), a custom application specific integrated circuit (ASIC) built by Google, is one such accelerator, which is exclusively built to handle most of the deep neural networks (DNN) inference workloads in their servers.

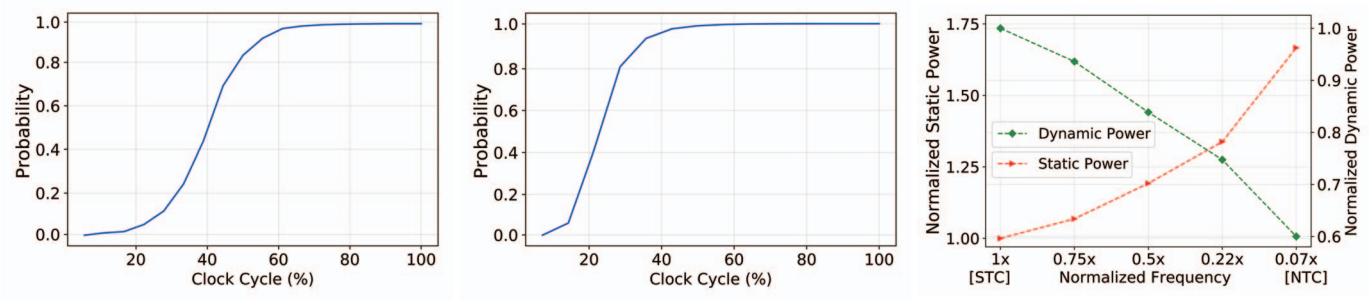
The rapidly increasing workloads calls for an increase in the processing speed and deployment volume [1]. It, however, comes at a cost of a heavy power usage, thus affecting the energy efficiency of the system. In order to preserve the energy efficiency, we operate the TPU at the Near-Threshold Computing (NTC) region, where we scale down the transistor supply voltage to just above its threshold voltage [2].

Accelerators like TPUs are designed to offer a very high throughput for DNN inference workloads. Although NTC operating conditions can ensure a low energy consumption, the throughput is heavily declined due to the slower transistors and longer computational delays. Shabanian et al and Bal et al, demonstrated that both NTC-GPU and NTC-CPU are highly susceptible to process variation which cause timing violation induced performance bottleneck [3], [4]. NTC-TPU is not an exception, this sensitivity can impact the DNN inference accuracy significantly [2], [5]. This paper underlines the significance of the computational delays and order of execution of the arithmetic units, to handle timing violations in NTC TPUs. Additionally, by exploring the predictable data flow pattern in the TPU systolic array, we further enhance its energy efficiency, thereby promoting an error-resilient and energy-efficient TPU design paradigm.

Several timing error resilient schemes have been explored for CPUs and custom ASICs [6]–[8]. However, these schemes are inefficient for combating timing violation in TPUs. Razor is one such popular timing violation detection method which uses a double sampling flip-flop to detect the errors [6]. Using instruction replay, the erroneous data is recomputed and the correct value is propagated to the next stage of the pipeline. TPU has a massive systolic array of 256×256 multiplier-and-accumulate (MAC) units. So, using an instruction replay in one MAC unit, results in stalling the operation of the entire systolic array, leading to a massive drop of throughput and increase in the energy consumption. TE-Drop is a recently proposed technique to handle timing violations in TPU-like systolic arrays. In this technique, the MAC unit encountering a timing error, steals an execution cycle from its downstream MAC, and recomputes the correct value [7]. In the process, the downstream MAC's computation is bypassed. However, there will be multiple levels of bypassing, in case of timing errors in consecutive rows of the same column of MACs, in the same clock cycle. Bypassing multiple computations can cause a severe drop in the inference accuracy. Additionally, the timing errors encountered in the last row of MACs will not be tackled by TE-Drop, also resulting in an accuracy drop. A naive approach to tackle timing violations is to allow the erroneous data to flow through the successive stages of operations [8], [9]. This technique undermines the effects of the erroneous data in DNN computations, as a large number of timing errors causes a significant drop in the inference accuracy [10].

In order to overcome the drawbacks of these error handling schemes, we propose a unique timing error correction technique which handles timing errors in the same cycle of the execution while enhancing the energy efficiency of the TPU. We observe that in a MAC, multiplier takes relatively higher execution time than accumulator (Section II-C). Additionally, we observe a predictable data flow pattern in the TPU systolic array (Section III-C). Analyzing these computational delays, data flow patterns, and utilizing the computational order of the arithmetic units, we propose EFFORT—an error resilient, low-power, novel TPU design paradigm. Following are the specific

978-1-7281-4123-7/20/$31.00



(a) CDF of multiplier delay distribution. (b) CDF of accumulator delay distribution. (c) Power consumption of a MAC unit.

Fig. 1: *CDFs of the delay distributions for multiplier (Figure 1(a)) and accumulator (Figure 1(b)) show that the multiplier has higher computational delay compared to the accumulator. Accumulator takes less than a half clock cycle for its part of computation. Figure 1(c) portrays the increase of static power and decrease in dynamic power for decreasing frequencies. Voltages are scaled accordingly to depict the shift from STC to NTC.*

contributions in this work:

- We experimentally demonstrate that 8-bit multiplier takes higher computation time than 24-bit accumulator (Section II-C). We exploit the computational delays and operational order of these arithmetic units to tackle timing errors.
- We also observe a predictable data flow pattern in the TPU and utilize this data flow pattern to reduce the energy consumption in the systolic array (Section III-C).
- We propose a low-overhead dynamic timing error detection/correction and a dynamic power management technique, called EFFORT (Section III-B). Our technique detects the timing errors, obtains the corrected data and propagates it to preserve the output accuracy. Additionally, we employ a low-overhead clock gating technique to improve the energy efficiency of the TPU (Section III-C).
- In comparison to TE-Drop [7] and the Baseline-TPU, EFFORT delivers 2× better performance for 3 out of 4 DNN datasets, while incurring only 2% loss in inference accuracy (Section V-B).
- We show that EFFORT consumes up to 6% and 27% less power and gives up to 1.06× and 1.35× better performance per unit power, than Baseline-TPU and TE-Drop (Section V-C).

II. Motivation

In this section, we illustrate the unseen opportunities that can be availed to tackle timing errors in a TPU systolic array. Section II-A sheds light on the background of the TPU systolic array and inherent opportunities which can be exploited for an improved performance. Using the cross-layer methodology in Section II-B, we investigate the MAC units' delay profiles. Section II-C elaborates the significance of the results and establishes the ground work for our timing error correction and dynamic power management scheme.

A. Background

1) DNN Accelerators: DNN obtains inference using multiple layers of computation. Outputs of neurons from each layers are referred to as activation streams. An activation matrix is multiplied with the weight matrix in each layer. To accelerate the matrix multiplication, a systolic array of MAC units are employed in DNN accelerators [11]. TPU–a DNN accelerator–uses a 256×256 systolic array of MACs. The weight matrices are pre-loaded into the MACs. The activation streams flow from left to right in consecutive clock cycles. The activation and weight matrices maintain an 8-bit integer precision, while the accumulator maintains a 24-bit integer precision.

2) Opportunities in a Systolic Array: The asymmetric delay distributions of the multiplier and accumulate blocks in a MAC unit, open up an unique opportunity to tackle timing errors in a systolic array. The accumulate operation in a MAC, adds the output of the upstream MAC to the output of its own multiplier block. Due to a relatively large computation time of the multiplication operation, the output from the upstream MAC has ample time to reach the current accumulate unit, presuming the synchronization takes place at the primary output of the MAC. *Exploiting this available timing window, correcting an erroneous operation at the upstream MAC can be overlapped with the multiplication operation of the current MAC, without paying any additional performance penalty.*

The wavefront propagation of data in a systolic array leads to a static pattern of busy and idle phases. Such a predictable pattern creates an avenue to conserve power of the idle MAC units. Next, we briefly discuss our experimental methodology, used to demonstrate these opportunities in the systolic array of an NTC TPU.

B. Methodology

We synthesize a multiplier and an accumulator unit at NTC, using 15-nm FinFET library from NanGate [12]. To model the PV at NTC for FinFET, we use VARIUS-NTV models [13]. For a conservative estimate, we consider PV-induced delays in randomly chosen 2% of the gates in the circuit [14]. We use our in-house statistical analysis tool to investigate the delay distribution of the sensitized path for different inputs to the multiplier and accumulator unit.

C. Results and Significance

Figures 1(a) and 1(b) show the delay distributions of the multiplier unit and accumulate unit, respectively. The multiplier unit is tested for a set of all possible 8-bit activation streams created against all possible 8-bit weight streams, which results in a total of 65536, 16-bit output combinations. Each one these 65536 outputs serve as one of the inputs for the accumulator while, the other input of the accumulator is fed with its own output from the previous cycle.

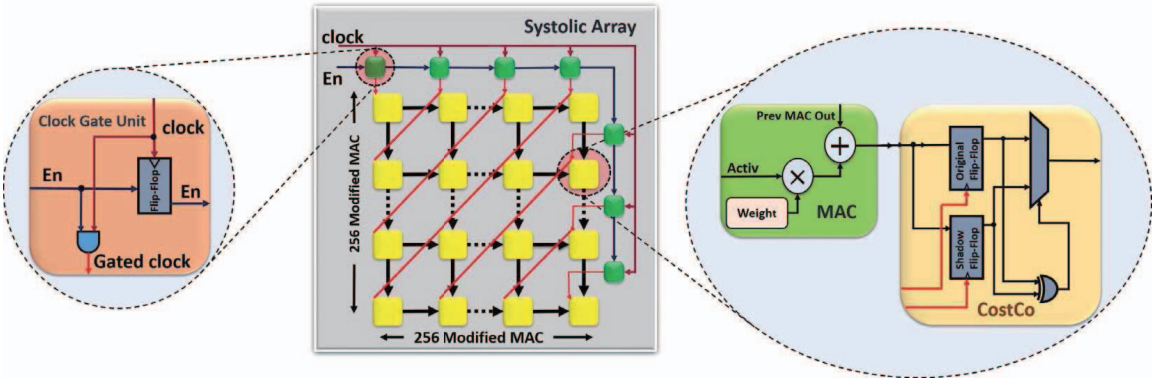


Fig. 2: *CostCo implemented inside the MAC unit to detect and correct and timing violations.*

From Figure 1(a) and 1(b), it can be inferred that, even if the output of multiplier unit is sensitized only to the activation stream due to the preloaded weight, the multiplier unit still induces a higher combinational delay to the MAC operation in comparison to the accumulator, during a clock period of operation. Figure 1(b) shows that the accumulation requires less than half cycle of the clock period. *This disparate timing characteristics of the multiplier and accumulate operations create an opportunistic timing window to correct any timing violation in an upstream MAC, thereby preventing any erroneous value to be propagated down the column of the systolic array.*

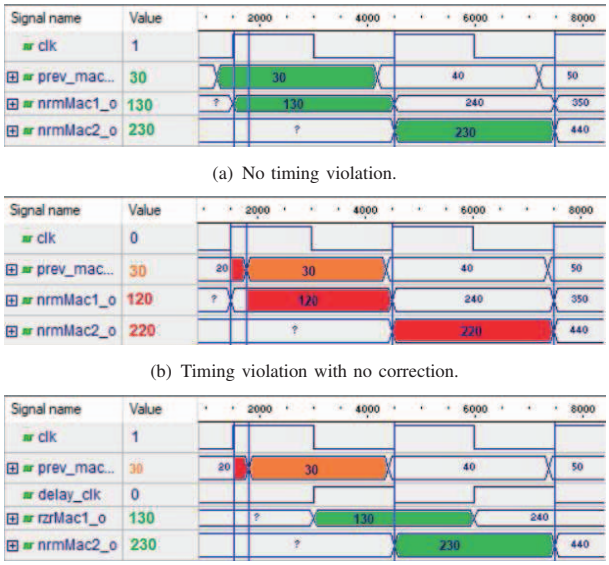
Figure 1(c) depicts the decrease in dynamic power and domination of static power in a MAC unit as the region of operation is changed from super-threshold computing (STC) to NTC. The X-axis is normalized to 1GHz. Operating voltage is set at 0.85V and scaled linearly to depict the shift in operating conditions. Static energy consumption can be reduced by operating the systolic array at frequencies, above the nominal NTC frequency. However, increasing the operating frequency linearly increases the dynamic energy consumption of the MAC units. In order to curb the increase in dynamic energy, an in-situ clock gating technique can be employed in the systolic array. With this opportunistic window in-sight, we explore our performance enhancing TPU systolic array design–EFFORT.

III. EFFORT Design

In this section, we discuss *Energy eFFicient and errOr Resilient TPU (EFFORT)*, a novel design paradigm to improve the performance of an NTC TPU by enhancing the timing error resilience of its MAC units and managing the dynamic energy consumption of the systolic array. We describe the overview of EFFORT in Section III-A, and explain its detailed components in Section III-B and Section III-C.

A. Design Overview

Figure 2 demonstrates the high-level design of EFFORT. We add two key modifications to the baseline NTC TPU. First, we augment each MAC unit with a novel penalty-free error detection and correction logic, thus preserving a high performance. Second, a low-overhead clock gating technique is implemented to conserve the dynamic power of the systolic array. These two components are discussed next.



(a) No timing violation.

(b) Timing violation with no correction.

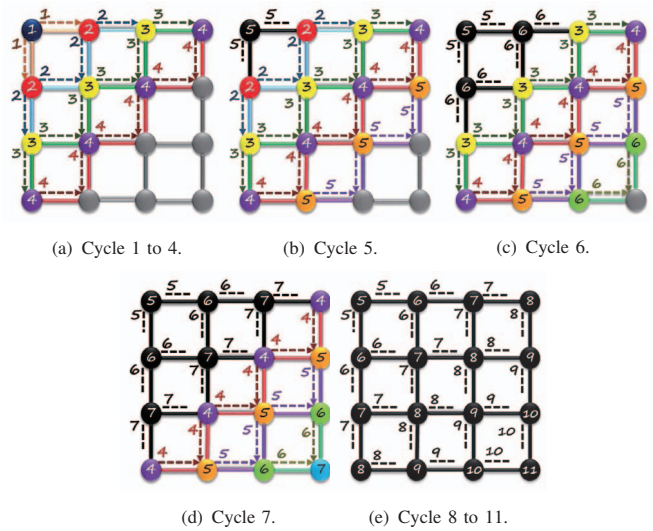
(c) Timing violation with correction.

Fig. 3: *CostCo can diagnose timing violation and propagate the corrected value within one clock cycle.*

B. Costless Correction (CostCo)

In this section, we introduce Costless Correction (CostCo). We augment the conventional Razor [6] with a multiplexer (MUX) and an Exclusive-OR (XOR) gate, as demonstrated in Figure 2. Since CostCo controls its output with the comparison of the shadow latch and the main latch, it is capable of propagating the correct value to the downstream logic within the same clock cycle that timing error detection happens. Figure 3 demonstrates the RTL simulation waveforms for two consecutive MAC units within a column, in a systolic array, both in absence and presence of a timing violation. Note that, we synchronize *only* the primary output of the MAC units with the system clock, to enable our proposed CostCo design.

Figure 3(a) demonstrates the normal output waveforms of two consecutive column MACs in absence of any timing violation. Figure 3(b) shows how a small additional delay in the input of the first MAC, engenders timing violation in its immediate downstream MAC, leading to an erroneous result. Figure 3(c) exhibits how CostCo can detect the timing violation and propagate the correct value to its succeeding downstream MAC, within the same clock cycle. CostCo can be employed as a competent method to tackle timing violation if the downstream combinational logic has sufficient time-



(a) Cycle 1 to 4. (b) Cycle 5. (c) Cycle 6.

(d) Cycle 7. (e) Cycle 8 to 11.

Fig. 4: *Data flow pattern in a 4×4 systolic array for 11 consecutive cycles. Gray MACs are yet to receive their inputs, black MACs have completed their operations, while the rest of the MACs are presently computing their respective outputs.*

window before the next rising edge of the clock, to replay its logical operations on the corrected data.

We consider 24 CostCo flip-flops at the output of each MAC that provide the correct values to the downstream MACs, in case of a timing error. The output of each MAC is utilized in the accumulation operation in its succeeding MAC. As accumulation in each MAC requires less than 50% of the clock cycle (Section II), we decide a 50% shift in the system clock to provision the CostCo flip-flop clock. This shift of clock provides an opportunity to detect timing errors up to 50% beyond the system clock, while it guarantees the succeeding MACs to have adequate time to accomplish their accumulation operations in the remaining time window. We discuss the hardware overhead and performance gain of this design in Section V.

C. Systolic Clock Gating

In EFFORT, we aim to increase the operating frequency, while keeping the supply voltage at the nominal NTC value, in order to provide a better performance compared to a baseline NTC TPU. To reduce the power consumption due to a high-frequency operation, we exploit the application independent data-flow pattern within the TPU systolic array, and employ a low-overhead clock gating technique.

1) Application Independent Data Flow: Figure 4 demonstrates the pattern of data flow inside a 4×4 systolic array for 11 consecutive clock cycles. In this figure, the gray nodes represent the MACs that have not received their data yet, the nodes in black denote the MACs that completed their operations, and other colored nodes demonstrate the MACs which are doing their operations. Numbers on black nodes show the cycle when they completed their operations, numbers on other colored nodes represent the cycle in which they received their first data, and numbers on each edge display the cycle in which the preceding node attempts to activate or deactivate its subsequent nodes either on the right-hand side or down a row. As Figure 4(a) exhibits, considering the upper left node as the start point from cycle 1 to 4, all the MACs from start point down to the main diagonal, receive their data respectively in a sequential fashion, while the rest of them are yet to receive their data. After cycle 4 (Figure 4(b) through Figure 4(d)), as a new set of MACs receive their data in each cycle, another set of MACs accomplish their tasks. Figure 4(e) displays the systolic array after 11 clock cycles, when the entire systolic array operation is completed. We observe that all the MACs on the same diagonal of the systolic array, are active or idle in the same cycles.

2) Clock Gating Components: Based on the activity pattern of a systolic array, we propose a low overhead clock gating technique to shutdown the clock of idle MACs, improving the dynamic energy consumption of the TPU. Since all of the MACs on the same diagonal of the systolic array are active or idle in the same clock cycle, instead of endowing a separate clock gating unit for each MAC, we provide one for each set of MACs on each diagonal. Since an $n \times n$ matrix has $(2n-1)$ diagonals, we thus reduce the total on number of required clock gating units from n^2 to $(2n-1)$. Figure 2 shows that each clock gating unit consists of one flip-flop to register the *enable signal* for the downstream clock gating unit, and an AND gate to control the clock for its corresponding group of MACs.

3) MAC Activity Analysis: Generalizing from Figure 4, an $n \times n$ systolic array needs $(3n-2)$ cycles to complete its operation. However, not all MACs are active during each clock cycle. For an $n \times n$ systolic array, at each clock cycle in the range [Cycle 1, Cycle n] and [Cycle ($2n$ - 1), Cycle ($3n$ - 2)], the total number of active MACs is $\frac{n \times (n+1)}{2}$. Furthermore, in the interval [Cycle $(n+1)$, Cycle $(2n)$], at each cycle i, the number of active MACs change by $(n - (2 \times i))$, where a positive (negative) value indicates an increase (decrease) in the number of MACs. Applying our proposed clock gating technique by summing the number of active MACs during the $(3n-2)$ clock cycles, we reduce the order of active sequential logic from $(3n^3)$ to (n^3). We discuss the experimental results and hardware overhead of this technique in Section V.

IV. Methodology

In this section, we expound our comprehensive cross-layer methodology, used to implement our proposed design and evaluate its capabilities across DNN applications.

A. Device Layer

We measure the delay distributions of basic logic gates (e.g., NOR, NAND and Inverter) by performing HSPICE simulations with 16-nm Predictive Technology Model [15]. We consider the impact of with-in die process variation at NTC by using VARIUS-NTV model [16]. In addition, we employ VARIUS-TC model to integrate the FinFET attributes [17]. The delay values are used in the circuit layer (Section IV-B) to analyze the delays in a MAC unit.

B. Circuit Layer

We implement a TPU systolic array, as well as, the components of our proposed design in Verilog RTL. We synthesize the developed RTLs using Synopsys Design Compiler. We use the synthesized netlists in our in-house statistical timing

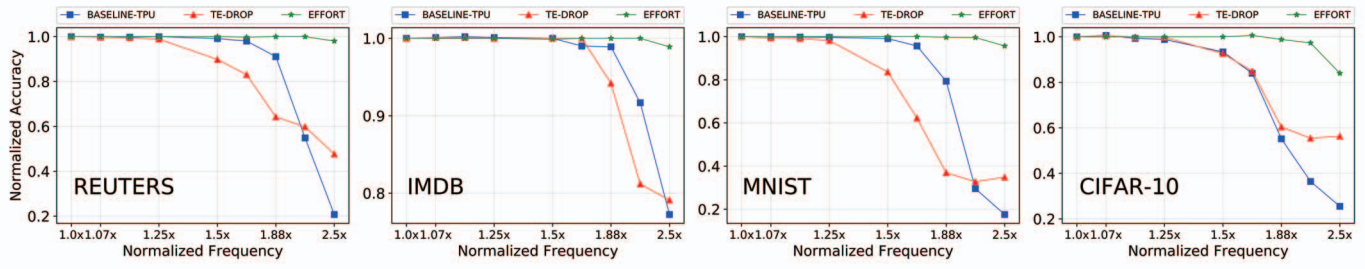


Fig. 5: *Normalized inference accuracies of the 4 DNN datasets for different comparative schemes.*

analysis (STA) tool. The STA tool employs libraries of delay distributions for basic logic gates from HSPICE simulations (Section IV-A), to provide the delays of the sensitized paths in the MAC circuit. We utilize the sensitized delays for further evaluations of our proposed technique.

C. Architecture Layer

We use our in-house TPU Systolic array simulator developed using C++, based on the detailed architecture of a TPU [11]. The delays from the STA tool (Section IV-B) are incorporated into the TPU Simulator to simulate timing errors in the MAC units. We interface Keras [18] with our TPU simulator to replicate a real-life inference engine. The DNN applications (viz., Reuters [19], IMDB [20], MNIST [21], CIFAR-10 [22]) are trained using Keras. Activation inputs and trained weights from each layers are extracted and separated into 256×256 matrices. Inference accuracy is obtained by combining the output matrices from the simulator.

V. Experimental Results

In this section, we compare the efficacy of different schemes for NTC TPU operation. The baseline frequency for our scheme is (0.45V, 67.5MHz), which offers an error-free execution of the systolic array. Section V-A introduces the comparative schemes. Section V-B presents the inference accuracies. Section V-C discusses the power savings and Section V-C presents the overheads of EFFORT.

A. Comparative Schemes

- **Baseline-TPU :** This scheme operates an NTC TPU without any error detection and correction. It allows the erroneous data to propagate throgh all the computation stages in the systolic array [8].
- **TE-Drop :** This technique handles the timing errors by dropping the subsequent downstream MAC operation [7]. The erroneous MAC recomputes the output by stealing the clock cycle from its downstream MAC.
- **EFFORT :** This is our proposed technique which uses the opportunistic timing window in the MAC operation to detect and correct timing errors (Section III). However, if a computational delay falls beyond that opportunistic timing window, an erroneous value will be propagated.

B. Inference Accuracy

Figure 5 shows the normalized accuracies for different comparative schemes at various operating frequencies. The operating voltage is set to 0.45V for all frequencies. Y-axis is normalized to the error free accuracy for the baseline operation and X-axis is normalized to the baseline frequency. Error free accuracy for datasets are REUTERS: 0.80, IMDB: 0.89, MNIST: 0.98 and CIFAR-10: 0.77, respectively.

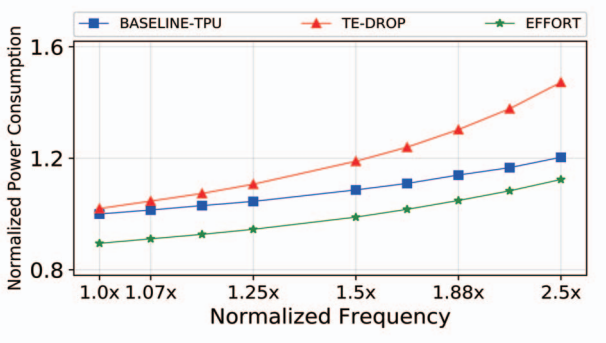


Fig. 6: *Power Consumption (Lower is better).*

A modest timing error resilience can be observed in all the schemes up to $1.25\times$ the baseline frequency of operation. Accuracy begins to decline as the number of errors drastically increases at higher frequencies. EFFORT outperforms other schemes by detecting and correcting most of the timing errors. However, for CIFAR-10, the computational delay at the highest frequency is relatively higher than other datasets, which increases the number of undetected errors in EFFORT and consequently, causes more reduction in inference accuracy. Baseline-TPU has a relatively sudden fall in inference accuracy as propagating errors in successive stages massively deteriorates the quality of the output matrices [10]. Inference accuracy for TE-Drop, however, falls at a slower pace, compared to the baseline-TPU. At higher frequencies, due to a large number of timing errors, TE-Drop bypasses a higher number of MAC computations, resulting in inferior accuracies compared to EFFORT. Hence, an NTC TPU, enhanced with EFFORT, results in only 2% average accuracy loss, when operated up to $2.5\times$ the baseline frequency, for 3 out of 4 DNN datasets.

C. Energy Efficiency

Figure 6 shows the average power consumption for the 4 DNN datasets for different comparative schemes. Power consumption for the comparative schemes are normalized to the power consumption of the Baseline-TPU at the baseline frequency. With the increasing operational frequency, power consumption steadily increases for all the schemes. However, EFFORT has lower power consumption compared to other schemes. The clock gating scheme implemented in EFFORT yields lower dynamic power in MAC units which are idle. Hence, the overall power consumption for the systolic operation is reduced. Thus, EFFORT consumes up to 6% and 27% less power when compared to Baseline TPU and TE-Drop.

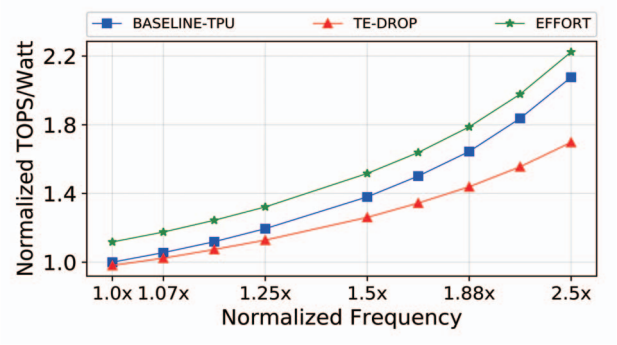


Fig. 7: *TOPS/Watt (Higher is better).*

TE-Drop, due to its Razor flip-flops, has the highest power consumption.

Figure 7 depicts the average of the energy-efficiency, measured in Tera Operations Per Second (TOPS)/Watt, for 4 DNN datasets with the normalized frequencies. TOPS/Watt for all the scheme are normalized to that of the Baseline-TPU at the baseline frequency. All the schemes have the same TOPS measure. However, TE-Drop has the lowest energy-efficiency due to its relatively high power footprint compared to both EFFORT and the Baseline-TPU. Owing to the clocking gating, EFFORT boasts the highest energy-efficiency. EFFORT delivers up to 1.06× and 1.35× better performance per unit power consumption, relative to other schemes. *Hence, EFFORT is a superior NTC TPU design paradigm, offering a high energy-efficiency while providing a high timing error resilience.*

D. Implementation Overhead

EFFORT incurs hardware overheads due to the clock gating circuit, and the CostCo logic added to each MAC. As the systolic array takes almost 24% of the TPU die area [11], EFFORT incurs an area overhead of only 5%.

VI. Related Work

Recent studies explore timing error resilience as well as energy efficiency improvement in DNN accelerators. Reagen et al. presented a co-design technique across the algorithm, architecture and circuit level to improve the energy efficiency of DNN by applying selective pruning through lowering SRAM voltages without compromising the accuracy [23]. Chen et al. proposed a run-time pruning technique, called row stationary, that enhances the efficiency of a convolutional neural network by re-configuring the spatial architecture, in order to map its computations [24]. Zhang et al. introduced an aggressive voltage underscaling method to improve the energy efficiency of DNN accelerators while keeping accuracy drop less than 1% [7]. Yu et al. presented a hardware pruning technique which applies SIMD-aware weight and node pruning synergistically at the design time to improve the energy efficiency of the DNN by reducing the size of the underlying hardware [25].

VII. Conclusion

Increase in the processing workloads in real-time DNN applications calls for a DNN accelerator capable of delivering high classification accuracy while efficiently meeting the energy requirements of the system. This paper demonstrates EFFORT—a high-performance energy-efficient novel design paradigm for a TPU, operating at NTC. EFFORT efficiently detects and tackles timing errors while reducing the power consupmtion of the TPU. EFFORT delivers upto 2.5× increase in performance with a minimum drop in accuracy and consumes in between 6% - 27% less power in comparision to recently proposed schemes. Additionally, EFFORT gives between 1.06× and 1.35× superior performance per unit power against representative timing error resilient schemes.

Acknowledgements

This work was supported in part by National Science Foundation grants (CAREER-1253024, and CNS-1421022). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

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," in *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*. IEEE, 2017, pp. 1–12.

[2] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. N. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proc. of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[3] T. Shabanian, A. Bal, P. Basu, K. Chakraborty, and S. Roy, "Ace-gpu: Tackling choke point induced performance bottlenecks in a near-threshold computing gpu," in *ISLPED*, 2018.

[4] A. Bal, S. Saha, S. Roy, and K. Chakraborty, "Revamping timing error resilience to tackle choke points at ntc systems," in *Proc. of DATE*, 2017, pp. 1020–1025.

[5] U. Karpuzcu, N. S. Kim, and J. Torrellas, "Coping with parametric variation at near-threshold voltages," *IEEE Micro*, vol. 33, no. 4, pp. 6–14, July 2013.

[6] D. Ernst, N. S. Kim, S. Das, S. Pant, R. R. Rao, T. Pham, C. H. Ziesler, D. Blaauw, T. M. Austin, K. Flautner, and T. N. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. of MICRO*, 2003, pp. 7–18.

[7] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "Thundervolt: Enabling aggressive voltage underscaling and timing error resilience for energy efficient deep neural network accelerators," *arXiv preprint arXiv:1802.03806*, 2018.

[8] P. N. Whatmough, S. Das, D. M. Bull, and I. Darwazeh, "Circuit-level timing error tolerance for low-power dsp filters and transforms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 6, pp. 989–999, 2012.

[9] F. Nakhaee, M. Kamal, A. Afzali-Kusha, M. Pedram, S. M. Fakhraie, and H. Dorosti, "Lifetime improvement by exploiting aggressive voltage scaling during runtime of error-resilient applications," *Integration*, vol. 61, pp. 29–38, 2018.

[10] X. Jiao, M. Luo, J.-H. Lin, and R. K. Gupta, "An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations," in *Proceedings of the 36th International Conference on Computer-Aided Design*. IEEE Press, 2017, pp. 945–950.

[11] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, 2018.

[12] NanGate, http://www.nangate.com/?page_id=2328.

[13] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "Varius:a model of process variation and resulting timing errors for microarchitects," *IEEE Tran. on Semicond. Manufac.*, vol. 21, pp. 3 –13, 2008.

[14] P. Pandey, P. Basu, K. Chakraborty, and S. Roy, "Greentpu: Improving timing error resilience of a near-threshold tensor processing unit," in *Proc. of DAC*, 2019, pp. 173:1–173:6.

[15] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm early design exploration," *T. Electron Devices*, vol. 53, no. 11, pp. 2816 –2823, 2006.

[16] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, and J. Torrellas, "Varius-ntv: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages," in *DSN*, 2012, pp. 1–11.

[17] S. K. Khatamifard, M. Resch, N. S. Kim, and U. R. Karpuzcu, "Varius-tc: A modular architecture-level model of parametric variation for thin-channel switches," in *ICCD*, 2016, pp. 654–661.

[18] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[19] "Reuters-21578 dataset." [Online]. Available: http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

[20] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis." Association for Computational Linguistics, 2011, pp. 142–150.

[21] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[22] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[23] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3. IEEE Press, 2016, pp. 267–278.

[24] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[25] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das, and S. Mahlke, "Scalpel: Customizing dnn pruning to the underlying hardware parallelism," in *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2. ACM, 2017, pp. 548–560.