STRIVE: Enabling Choke Point Detection and Timing Error Resilience in a Low-Power Tensor Processing Unit

Noel Daniel Gundi, Zinnia Muntaha Mowri, Andrew Chamberlin, Sanghamitra Roy, Koushik Chakraborty
BRIDGE Lab, Electrical and Computer Engineering, Utah State University
{noeldaniel.gundi, zinnia.mowri, andrew.chamberlin, sanghamitra.roy, koushik.chakraborty}@usu.edu

***Abstract*—Rapid growth in Deep Neural Network (DNN) workloads has increased the energy footprint of the Artificial Intelligence (AI) computing realm. For optimum energy efficiency, we propose operating a DNN hardware in the Low-Power Computing (LPC) region. However, operating at LPC causes increased delay sensitivity to Process Variation (PV). Delay faults are an intriguing consequence of PV. In this paper, we demonstrate the vulnerability of DNNs to delay variations, substantially lowering the prediction accuracy. To overcome delay faults, we present STRIVE—a post-fabrication fault detection and reactive error reduction technique. We also introduce a time-borrow correction technique to ensure error-free DNN computation.**

I. Introduction

The emergence of Deep Neural Networks (DNNs) and their growing usage in diverse applications has led to a rapid advancement in the development of Application Specific Integrated Circuit (ASIC) architectures for the Artificial Intelligence (AI) computing realm. A Tensor Processing Unit (TPU) is one such ASIC, developed by Google and has been deployed in their datacenters for the inference phase of the DNN computation [1]. With the rapid deployment of AI hardware at the edge, there is a tremendous need to investigate these circuit-architectures in the Low-Power Computing (LPC) region [2].

Figure 1 demonstrates a key challenge in realizing AI hardware at the edge: a massive delay variance with voltage at LPC in comparison to Super Threshold Computing (STC). The figure shows a FO4 inverter delay characteristics at these two regions, when the supply voltage is varied by the same percentage variation of the respective voltage domains. For example, a 40% voltage variation can cause a 45% delay variation at STC, and a huge 250% delay variation at LPC. Due to these extreme sensitivities, the omnipresent manufacturing Process Variation (PV) in the devices can realize gate delays up to 20× of the nominal values in the LPC region [3]. Collectively, a small set of PV-affected gates, in combination with a minute variation in operating condition (e.g., voltage), can completely alter the critical path of the circuit and protract the combinational delay, which subsequently leads to frequent timing violations. We term these delay faults as Low-Power Faults (LP-faults) in this paper.

LP-faults may remain benign and exhibit the correct operation at nominal voltages. However, they are exposed and cause frequent timing faults at LPC. The process of frequency guard-banding which works efficiently at STC, becomes highly ineffective at LPC due to the extreme variation in delays. In a tightly pipelined architecture such as TPU with a large number of interconnected Multiplier-and-Accumulate (MAC) units, detection of an LP-fault in an individual MAC unit can be a formidable challenge. Furthermore, as these faults manifest only after fabrication and at certain operating conditions, detecting and correcting these faults become even more challenging. In this paper, we analyze the damage an LP-fault can induce on a DNN inference operation.

Interestingly, many modern datasets used in DNN computations exhibit a plethora of zero weights [4], [5], as well as, zero activation elements. In conjunction with the perceived resilience of DNN software from errors, it is intuitive to expect that DNN inference may be inherently tolerant to LP-faults [6]. However, our rigorous cross-layer analysis reveals otherwise. For example, an otherwise innocuous zero result from a MAC can lead to non-zero outcome under an LP-fault, causing a havoc in the inference accuracy. Not only do we find that inference accuracy can drop dramatically from LP-faults, our results show extreme unpredictability in these inference drops, demonstrating a critical need to rigorously analyze and mitigate LP-faults for successful deployment of these AI hardware accelerators at the edge.

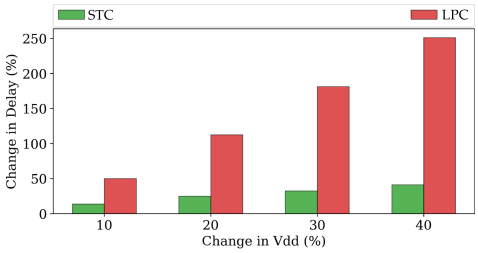


Fig. 1: *Delay sensitivity for power supply at STC vs LPC. HSPICE simulations shown for a 31 fan-out-of-four (FO4) inverter chain at the 14nm multi-gate technology node [9].*

Razor is a popular technique which uses a double sampling flip-flop to detect a timing error in a pipelined circuit and employs an instruction replay to recompute the erroneous data [7]. However, replaying an instruction in a TPU pipeline requires stalling of the entire systolic array operation, incurring a massive loss in throughput for such a data-parallel architecture. *To our knowledge, our paper is the first work to introduce a post-fabrication fault detection technique to identify the faulty MAC units in a TPU.*

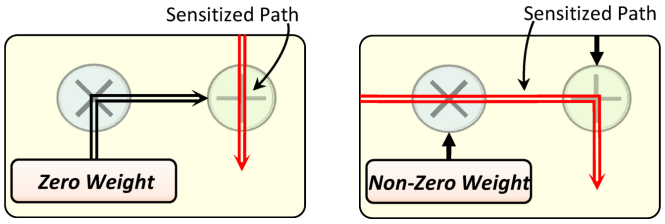
Our specific contributions in this paper are as follows:

- We explore the impact of delay faults in a systolic array of MAC units. We investigate the problem an LP-fault can pose in transforming a zero output from a multiplier to an incorrect non-zero value (Sections II-B and II-C). We also show how a subset of zero computations in a DNN matrix multiplication magnifies this threat (Section II-C).
- We introduce STRIVE—a low-overhead faulty MAC detection technique for a TPU systolic array (Section III-E), to identify the PV-affected MAC units and timing error resilience techniques to mitigate the effect of LP-faults (Sections III-D and III-F).
- We demonstrate that STRIVE incurs less than 1% loss in inference accuracy for 6 DNN benchmarks in a TPU affected by a gate level fault rate of 1% (Section V-B). Additionally, STRIVE gives 1.8× and 1.3× better performance per unit power than Fault-Aware Pruning [8].

II. Motivation

In this section, we will uncover a post fabrication phenomenon which poses a severe threat to the error resilience of DNNs. Moreover, we also demonstrate the threat posed by a multiplier unit, when an expected zero computational output results in an unpredictable non-zero value. Sections II-A and II-B provide a background of the TPU and LP-faults, respectively. Sections II-C and II-D elaborate the results and the significance of our demonstration.

979-8-3503-2348-1/23/$31.00 ©2023 IEEE



(a) Zero Weight—Masks LP-fault. (b) Non-zero Weight—Exposes LP-fault.

Fig. 2: *Sensitized paths in a MAC unit for zero and non-zero weights, respectively.*

A. TPU Systolic Array

DNNs utilize multiple layers of computations during the inference stage. The multiple layers of DNNs are translated into a matrix multiplication of the activation input, with the weight matrix. A TPU employs a systolic array of $n \times n$ MAC units to expedite the matrix multiplication operations. Figure 5 shows a TPU–the yellow array, where each yellow box is a MAC unit comprising a multiplier and an accumulator. Activation inputs are stored in a unified buffer and subsequently streamed across the MAC array, while the weight matrices are pre-loaded into the MAC units. The activation inputs and stationary weights maintain an 8-bit precision.

B. Impact of PV at Low-Power: LP-faults

PV sensitivities experienced at STC are severely exacerbated at LPC. Therefore, a small effect of PV in a post fabrication circuit can produce a substantially large delay variation, when the operating region is switched from STC to LPC. Although the actual variation in physical dimensions (i.e., $gate$ length or t_{ox} length) remains identical, the observed variation will be in the transformed or elongated delays. Hence, any sensitized circuit path that exceeds the guard-banded clock period leads to a timing violation. These faults are termed as ***LP-faults*** or ***Delay Faults***. LP-faults can be completely hidden at STC, as the timing guard-band essentially covers up the relatively smaller delay variations (Figure 1). As we move towards LPC, the voltage and frequency are lowered appropriately, and a suitable timing guard-band is applied. However, as the delay variations are significantly higher at LPC, the prolonged combinational delay due to PV will exceed the timing guard-band and trigger an LP-fault. Furthermore, as factors leading to LP-faults emanate during fabrication, their effects can be perceived only during the real-time working environment.

1) Threats due to LP-faults:

Effect on Inference Accuracy: In a TPU systolic array (at every clock cycle), the output of each MAC unit, is added to the resulting activation and weight product of the consecutive downstream MAC unit. Hence, the accumulated product from the lowermost row forms a single element of the output matrix. However, as the systolic array is operated at LPC, the effects of PV in a MAC unit can instigate a delay fault and produce an erroneous output. As the PV-affected gates are distributed across the systolic array, more and more MAC units will be rendered faulty. Consequently, a large magnitude of LP-faults will increase the propagation of erroneous outputs and eventually incorrect values will be stored in the output matrix (later shown in Figure 8). Therefore, inference accuracy processed using the erroneous values will be significantly lowered.

Significance of Zero Activation: Figures 2(a) and 2(b) compare the sensitized paths in a MAC unit between a zero weight and a non-zero weight. A zero weight in the MAC unit drives the output of the multiplier to zero for the entirety of the matrix multiplication operation, thus sensitizing the second accumulator input path (i.e., output from the upstream MAC) to the output. Eventually, the upstream MAC output is forwarded to the MAC unit in the immediate lower row. Hence, a zero weight masks the LP-faults. However, for a non-zero weight, the circuit path from the activation input will be sensitized. *Consequently, even for a zero activation input, a prolonged delay needed to stabilize the faulty multiplier output can eventually expose an LP-fault, thereby resulting in a non-zero output.* We term such computations as *Fault Prone* zero computations.

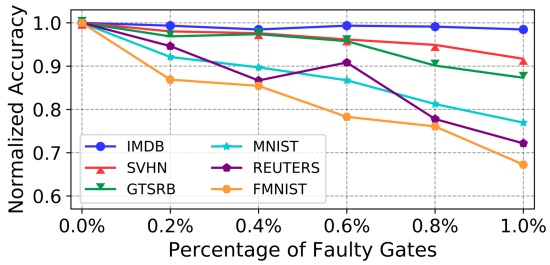


Fig. 3: *Increasing number of faulty gates increases the magnitude of LP-faults, thereby deteriorating the inference accuracy.*

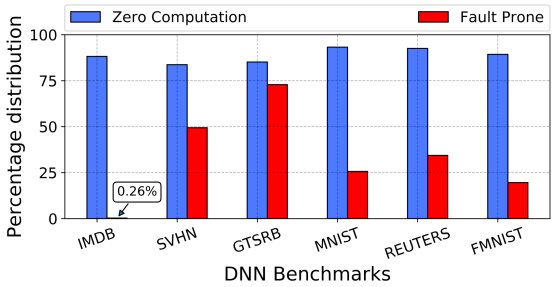


Fig. 4: *Zero computations and Fault Prone zero computations elaborated as a percentage of total computations for 6 different DNN benchmarks.*

We present the motivational data to demonstrate the serious effects of LP-faults, by employing a rigorous cross-layer methodology as described in Section IV.

C. Results

Figure 3 depicts a drop in the inference accuracy when the percentage of faulty gates increases in a TPU systolic array, for 5 out of 6 DNN benchmarks. The increase in the LP-fault level challenges the inherent timing error tolerance of DNNs [6], thereby dwindling the DNN inference accuracy. An outlier benchmark is IMDB, where the significant number of zero weights sensitizes the accumulator paths to the output (case of Figure 2(a)), consequently reducing the damaging effects of LP-faults.

Figure 4 shows the percentage of zero computations and *Fault Prone* zero computations for 6 different benchmarks. From Figure 4, it is evident that even though more than 80% of the computations involve zero computations, more than 20% of the zero computations are *Fault Prone* in 5 out of the 6 DNN benchmarks. Therefore, even a zero computation in a PV-affected multiplier can pose a considerable threat and contribute in lowering the inference accuracy.

D. Significance

Our findings demonstrate that *the effects of PV at LPC can lead to significant deterioration of the DNN inference accuracy*. Additionally, our study shows that even a predictable zero computation is not safe from the threat of an LP-fault. Since the phenomenon leading to a delay fault is born during fabrication, identical chips can exhibit different variations in the sensitized paths. As AI Edge computing is migrating more towards LPC, the emergence of delay faults can

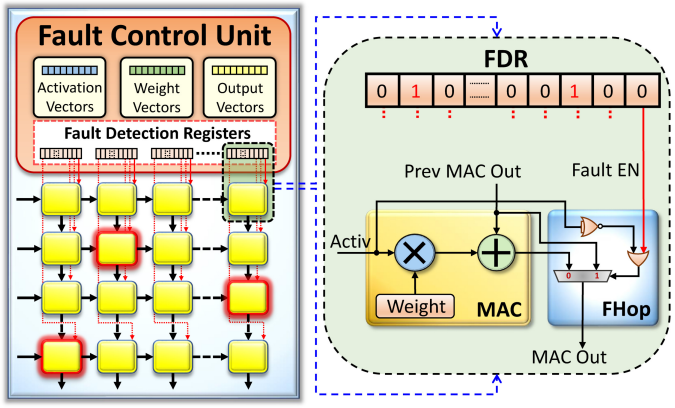


Fig. 5: *Design block and dataflow of STRIVE.*

severely hamper the DNN predictions. *Hence, we need a runtime mechanism to identify the faulty MAC units when a systolic array is operated at LPC and an efficient design paradigm to reduce the effect of LP-faults*. Since a zero activation input can be identified due to the redundant bit pattern and the output of a MAC unit for a zero activation is predictable, we can doctor such characteristics to our benefit. With this premise, in the next section, we explore our proposed scheme—STRIVE, to detect and handle LP-faults affected MAC units.

III. STRIVE Design

In this section, we discuss STRIVE, our novel design paradigm to detect and mitigate the effect of delay faults in an LPC TPU. STRIVE uses a low-overhead *post-fabrication faulty MAC detection technique* to identify the error-prone MACs. Furthermore, by inferring the location of faulty MACs, we devise and employ a *position-aware timing error mitigation* scheme to tackle an LP-fault. We describe the challenges and the overview of STRIVE in Sections III-A and III-B. The threat posed by an LP-fault is explained in Section III-C. Sections III-D through III-F elaborate the design components in detail.

A. Challenges

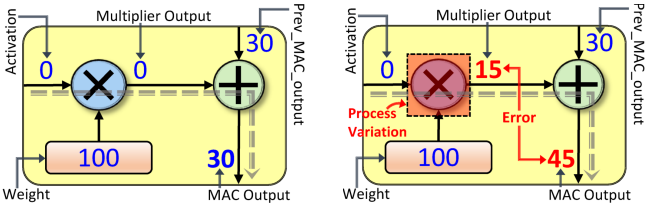
In this section, we highlight the problems which need to be addressed by STRIVE to effectively counter the threat posed by LP-faults.

1) A faulty MAC unit is susceptible to *Fault Prone* zero computation. Hence, a MAC unit needs to be equipped to handle this scenario (addressed in Section III-D) .
2) Diagnosing faulty MAC units spread across the TPU systolic array (addressed in Section III-E1).
3) Reclaiming the DNN inference accuracy from an LPC TPU housing PV-affected MAC units (addressed in Sections III-E2 and III-F).

B. Design Overview

Figure 5 depicts the top-level overview of STRIVE. We enhance the LPC TPU with a Fault Control Unit (FCU) and each MAC unit with Fault Hop (FHop). Initially, the FCU generates the activation and weight matrices, and the output map. Later, the FCU compares the TPU-generated output matrix and the output map to deduce the location of PV-affected MACs (Section III-E1). The faulty MAC locality is later exploited by the FCU, to tackle the LP-faults, using FHop (Section III-E2).

Additionally, we discuss an alternate technique Fault Hop Time-Borrow (FHop-TB), to mitigate the effects of an LP-fault, by enhancing the design of FHop (Section III-F).



(a) Fault free MAC. (b) PV-affected MAC.

Fig. 6: *Working of a PV-free and PV-affected MAC.*

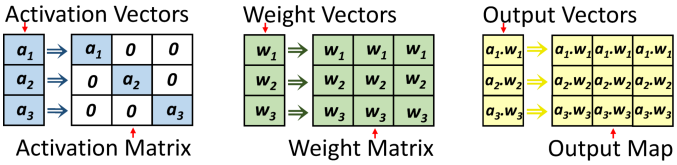


Fig. 7: *Development of input matrices and output map—correct outputs—by FCU.*

C. Illustrative example for an LP-fault

Figures 6(a) and 6(b) describe the operations of a fault-free and faulty MAC, respectively. For a fault-free MAC (Figure 6(a)), the multiplier concludes its computation within the clock period and delivers an error-free output, thereby leading to a correct output from the MAC unit (i.e., ***30***). However, the significant delay stemming from the PV-affected multiplier in a faulty MAC (Figure 6(b)), instigates an LP-fault. Hence, an incorrect value will be processed at the multiplier output (i.e., ***15*** in Figure 6(b)) and an erroneous value (i.e., ***45***) will be delivered to the next logic stage. So, a faulty MAC unit needs to be protected from a zero activation input, which is discussed next.

D. Fault Hop (FHop)

In this section, we introduce FHop. As a zero activation input produces a zero computation, we intend to entirely skip the MAC operation for a MAC unit. We augment the MAC unit with a NOR gate, an OR gate and a multiplexer (MUX), as demonstrated in Figure 5. The MUX output is controlled either by the NOR gate (viz., the zero activation input) or the *Fault EN* signal from FCU (discussed in Section III-E). Thus, for a *zero activation input or when a Faulty EN signal is set*, the erroneous MAC operation is bypassed and the accumulator input is directly presented to the MAC output through the MUX. Hence, *FHop prevents a possible Fault Prone zero computation and aids in the identification of faulty MACs* (discussed in Section III-E1b).

E. Fault Control Unit (FCU)

FCU houses the Fault Detection Registers (FDRs) along with the fault detection vectors. We dedicate one FDR to each column of the systolic array (Figure 5). Each bit of the FDR is labeled as a *Fault EN* signal and maps to a corresponding MAC unit in that column. The FCU operates in two modes: (a) *Fault Detection Mode*, to determine the faulty MACs, and (b) *Fault Resilient Mode*, which is the nominal operating mode of the TPU.

In *Fault Detection Mode*, all the bits of the FDR are reset to zero, to skip a MAC operation only for a zero activation input. The FCU later sets the FDR bits for all the faulty MACs. Hence, during the *Fault Resilient Mode*, the errant MAC operations are also skipped.

Next, we discuss the two operating modes of the FCU.

1) Fault Detection Mode: We will demonstrate the detection technique for a 3×3 systolic array for illustration purposes, as the same technique will be scaled up to any dimension of the systolic array.

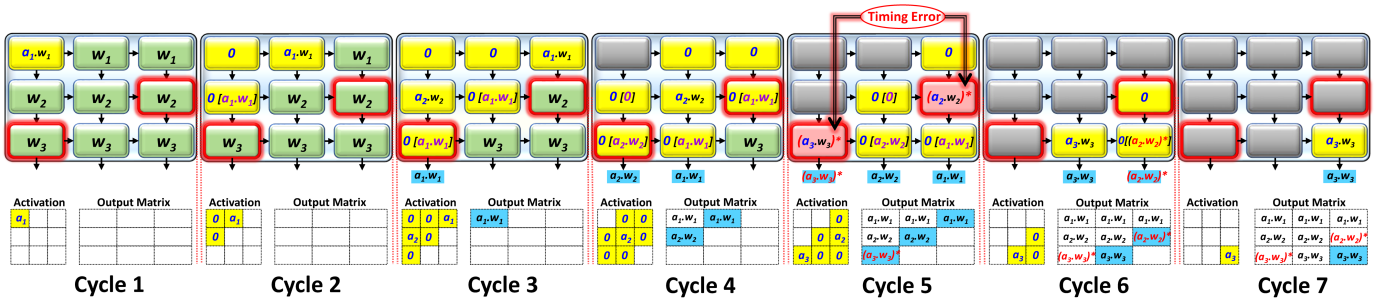


Fig. 8: *Green MACs are yet to receive the activation, yellow MACs are in operation and faulty MACs are highlighted with a dark red outline. Red MACs denote the occurrence of a timing error. Grey MACs have completed their execution. Only the final operation from each MAC is shown on the yellow MACs for space constraints. "0[$a_n.w_n$]" operation on a yellow MAC indicates that the activation input is "0" and accumulator input "$a_n.w_n$" will be forwarded to the next stage.*

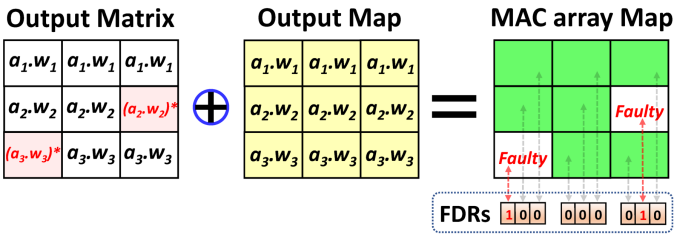


Fig. 9: *Comparing the entries of the Output Matrix and the Output Map yields the locality of the Faulty MACs.*

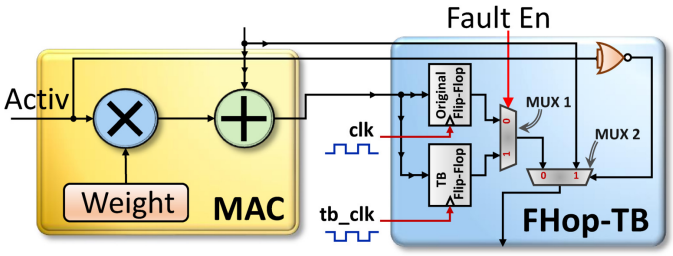


Fig. 10: *Detection and correction of timing errors by FHop-TB, using the Time-Borrow technique.*

a) Matrix and map—correct outputs—generation

Figure 7 depicts the low-overhead matrix/map generation methodology. The FCU generates the activation matrix by allocating the individual activation vectors along the diagonal and zero value for the non-diagonal entries, thereby creating a diagonal matrix. For the weight and output matrix, all the elements in a row are assigned the same weight and output vector. However, each row is allocated a different vector.

b) Faulty MAC detection

Systolic Array Operation with Faulty MACs : Figure 8 demonstrates the cycle-wise accurate matrix multiplication between the activation and weight matrices for the fault detection. The systolic array is pre-loaded with the weight matrix, while the activation matrix is transposed and streamed to the individual rows (i.e., from left to right). Activation table and the Output Matrix table shown below the systolic array in Figure 8, coherently maps the activation input being streamed to the corresponding MAC units and the output accumulated from the last row of the systolic array in the respective clock cycles.

- In Cycle 1, activation a_1 is multiplied with weight w_1 in the only active MAC unit.
- In Cycle 2, zero activation values are streamed to the first and second row. The activation a_1, from Cycle 1, is forwarded to the successive column (i.e., second column) of the first row. As the MAC units are enhanced with FHop, the active MAC unit in the second row encountering the zero activation will skip the MAC operation and forward the upstream MAC output (i.e., $a_1.w_1$) to the downstream MAC unit.
- Nominal operation continues in Cycles 3 and 4, as non-zero activation has not reached the faulty MACs.
- However, in Cycle 5, activation inputs a_2 and a_3 reach the faulty MAC units in rows two and three, respectively. As the non-zero activation inputs are multiplied by the respective weights, the extended combinational delay will trigger a timing error and erroneous values (i.e., $(a_2.w_2)^*$ and $(a_3.w_3)^*$) are generated. The erroneous value from the faulty MAC unit in the third row (i.e., $(a_3.w_3)^*$) is stored in the output matrix.
- In Cycles 6 and 7, the erroneous value $(a_2.w_2)^*$ along with other output entries are accordingly forwarded and stored in the output matrix.

Detection of Faulty MACs : Figure 9 presents the process of detecting the Faulty MACs, using the output matrix. FCU compares the respective entries of the output matrix with the output map (Figure 7) and determines the location of faulty MACs. FCU thus sets the bits of the FDR targeting the faulty MACs in each column (Figure 5). An entire systolic array operation is utilized by STRIVE for the errant MACs detection process.

2) Fault Resilient Mode: In the Fault Resilient Mode, as the faulty MACs are identified, the set *Fault EN* signal effectively skips the faulty MAC computation. Even though the skipping operation generates a loss in precision for the output matrix entries, the overall inference accuracy is not relatively affected due to the algorithmic level error tolerance of DNNs [6]. However, increase in the number of faulty MACs will eventually lead to a drop in DNN inference accuracy. To address this caveat, we enhance the design of FHop with a *time-borrow* feature to achieve a superior performance, discussed next.

F. Fault Hop Time-Borrow (FHop-TB)

Figure 10 presents FHop-TB—an enhanced variant of FHop that utilizes the combinational delay disparity between the multiplier unit and the accumulate unit to prevent the propagation of corrupted values. FHop-TB uses a Time-Borrow (TB) flop to capture the delayed output of the faulty MAC and direct it to the next logic stage within the same clock cycle.

The accumulation process utilizes less than 50% of the clock cycle. For a faulty MAC, we intend to borrow 50% of the clock cycle from its downstream MAC (i.e., the downstream MAC will be performing its own multiplier operation during this period) to procure the correct MAC output and ensure that the correct output is provided to the

Benchmarks		Error-free Accuracy
Name	Architecture	
IMDB [10]	CONV: 400x(400x50)x(398, 256), FC: 256x1	0.89
SVHN [11]	CONV: (32, 32, 3)x(32, 32, 32)x(32, 32, 32)x (14, 14, 64)x(14, 14, 64)x(5, 5, 128)x (5, 5, 128), FC: 512x512x10	0.94
GTSRB [12]	CONV: (3, 48, 48)x(32, 48, 48)x(32, 46, 46)x (64, 23, 23)x(64, 21, 21)x(128, 10, 10)x (128, 8, 8), FC: 2048x512x43	0.97
MNIST [13]	FC: 784x256x256x10	0.98
REUTERS [14]	FC: 2048x256x256x46	0.80
FMNIST [15]	FC: 784x256x512x10	0.89

TABLE I: *List of DNN benchmarks and their error-free accuracy.*

downstream MAC for its accumulation process. Thus, the TB latch is driven by a delayed clock (i.e., 50% shift from the system clock) to perform the *Time Borrowing* operation. In a faulty MAC, the output of the shadow latch is sensitized to the MUX 1 output; else, the output of the original latch is delivered to the MUX 1 output. For a zero activation, MUX 2 bypasses the upstream MAC output to the downstream MAC unit, irrespective of a faulty/non-faulty MAC unit. Therefore, the output of MUX 2 can switch between MUX 1 output and upstream MAC output.

IV. Methodology

Our extensive cross-layer methodology allows us to combine functional simulation of a DNN inference task (thus, allowing precise estimation of inference accuracy) with a holitic power-timing characteristics spanning three layers: device, circuit and architecture.

We perform HSPICE simulations on basic logic gates (e.g., NOR, NAND and Inverter) using the 16-nm predictive technology model to measure their delay distributions [9]. We use VARIUS-NTV to implement the impacts of with-in die PV at LPC [16]. We incorporate the FinFET attributes using the VARIUS-TC model [17]. We synthesize the Verilog RTL of the TPU systolic array augmented with our design components using Synopsys Design Compiler. The synthesized netlists are utilized by our in-house Static Timing Analysis (STA) tool along with the libraries of delay distributions to generate the sensitized path delays of the MAC unit for all the benchmark driven inputs. We use Cadence SoC Encounter to place and route the design and measure the area, power, and wirelength overheads.

We use our in-house cycle-accurate TPU systolic array simulator, modeled on the detailed TPU architecture [1]. To accurately simulate a timing violation, the combinational delays for a PV-affected MAC unit and a nominal MAC unit, developed from the STA tool are integrated into the TPU simulator. Initially, we train the DNN benchmarks by interfacing the TPU simulator with Keras (running tensorflow in the background) [18]. Table I lists the DNN benchmarks along with their error-free accuracies. Activation inputs from each layers are streamed across the weight matrices from the trained model for matrix multiplication. The output matrices are appropriately combined to obtain the inference accuracy.

V. Experimental Results

In this section, we examine the efficacy of STRIVE at LPC operating conditions–a typical use case for an edge system deployed for an inference task. The baseline operation is set to (0.45V, 67.5MHz) and guarantees an error-free execution for an LPC TPU (i.e., TPU without any faulty MACs). Section V-A introduces the comparative schemes. Sections V-B and V-C elaborate the inference accuracies and energy efficiency of the schemes. Section V-D discusses the overheads of STRIVE.

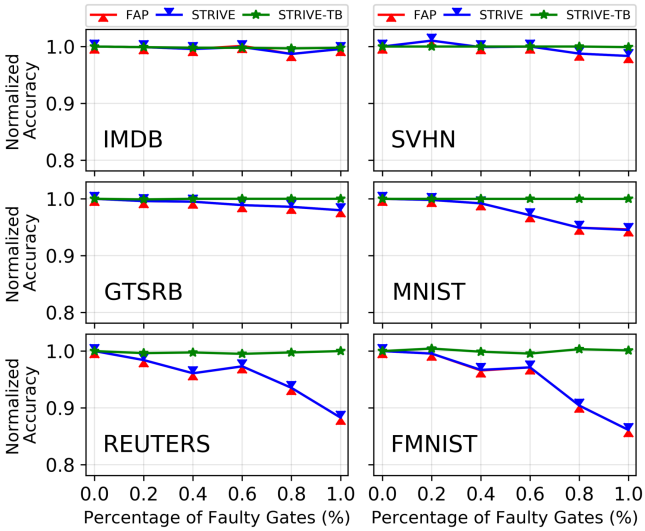


Fig. 11: *Normalized Inference accuracies of FAP, STRIVE and STRIVE-TB across 6 DNN benchmarks for different percentages of faulty gates (***under low-power***) affected in a TPU systolic array. The Y-axis values are normalized to the corresponding error-free accuracy at the baseline LPC TPU operation.*

A. Comparative Schemes

- **Fault-Aware Pruning (FAP) :** This scheme prunes all the weights of faulty MAC units [8]. The weights are implemented for multiple precision values to bypass the faulty multiplier in the MAC unit. FAP does not have an error detection scheme as we have proposed, and therefore *presumes to have an oracular knowledge of the faulty MACs*. As expected for an edge system deployed for inference in the field, we do not expect weights to be retrained, and thus consistently model no retraining for both FAP and our schemes, STRIVE and STRIVE-TB.
- **STRIVE :** This is our proposed technique, which utilizes the locality of the PV-affected MAC or a zero activation input to skip the erroneous MAC operation. FCU enables the *Fault EN* signal for the respective error-prone MAC to restrict the erroneous data from latching on to the accumulator output. FHop is used to perform the bypassing operation (Figure 5).
- **STRIVE-TB :** This scheme is an upgraded variant of STRIVE, which uses FHop-TB (Figure 10). The error-free output from a faulty MAC unit is captured by the Time-Borrow Flop using the delayed clock, and the *Fault EN* signal enables the forwarding of this correct data to the next stage.

B. Inference Accuracy

Figure 11 depicts the normalized inference accuracy for the three comparative schemes, as the percentage of faulty gates are increased in the TPU. All the schemes are operated at the baseline voltage and frequency. The X-axis represents the percentage of faulty gates in the TPU. STRIVE and FAP are able to offer modest error resilience for 4 out of 6 DNN benchmarks up to 0.6% fault rate. For REUTERS and FMNIST, the extreme data-delay variance between the activation sequences causes significant timing errors, thereby dropping the inference accuracy for STRIVE and FAP. The significant number of zero activation computations aids STRIVE in retaining the inference accuracy as the percentage of faulty gates are increased. However, STRIVE-TB incurs less than 1% loss in inference accuracy for up to 1% gate fault rate in a TPU, as it is able to capture the delayed data using the *Time-Borrow* approach. *Overall, STRIVE-TB vastly*

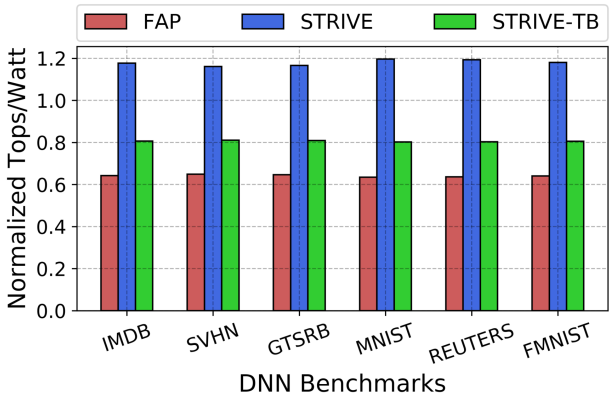


Fig. 12: *Energy Efficiency comparison of FAP, STRIVE and STRIVE-TB (higher is better).*

outperforms FAP, demonstrating remarkable resilience in retaining inference accuracy under LP-faults.

C. Energy Efficiency

Figure 12 presents the energy efficiencies of the comparative schemes measured using the Tera Operations Per Second (TOPS)/Watt metric. TOPS measure will be the same for all the schemes at the corresponding frequency of operation. The TOPS/Watt is normalized to that of the LPC TPU, operated at the baseline operating conditions. FAP has a lower energy efficiency due to its larger power consumption in comparison to STRIVE and STRIVE-TB. STRIVE boasts a higher energy efficiency due to its low overhead operation compared to STRIVE-TB. STRIVE and STRIVE-TB offers an average 1.8× and 1.3× better performance per unit power, compared to FAP. Hence, *STRIVE is an energy efficient design paradigm, able to extract superior performance even from an error-prone TPU.*

D. Implementation Overheads

STRIVE incurs overheads due to the inclusion of FCU and the combinational logic for FHop or FHop-TB. Area overhead added by STRIVE and STRIVE-TB is $\sim 1.8\%$ and $\sim 6\%$. STRIVE and STRIVE-TB incurs a power overhead of $\sim 2\%$ and $\sim 5\%$, and a wire-length overhead of $\sim 1.1\%$ and $\sim 3.5\%$, respectively.

VI. Related Work

Several techniques have been explored to mitigate the adverse effect of faulty Processing Elements (PEs) in deep learning accelerators. D.Xu et al. proposed a hybrid computing architecture to recompute the operations of the processing elements mapped to the faulty PEs in arbitrary locations [19]. Spyrou et al. demostrated a fault-tolerant Spiking Neural Network architecture with simplified error detection and recovery scheme [20]. Salami et al. evaluated an undervolting technique for Neural Network acceleration in Field Programmable Gate Arrays (FPGAs) to improve the power-efficiency [21]. Givaki et al. experimentally evaluated the effect of aggressive voltage underscaling of block RAMs in an FPGA by emulating the real fault maps of SRAM memories [22]. Tang et al. investigated the impact of GPU dynamic voltage and frequency scaling on the energy consumption and performance of DNNs [23]. Lee et al. explored optimization methods for hardware architectures for energy-efficient DNN processing on edge devices [24]. However, to the best of our knowledge, *this paper is the first work that addresses the threat of a delay fault in altering the output of a zero multiplier computation to a non-zero value, and introduce a novel post-fabrication timing error detection scheme for a TPU affected by faulty MAC units.*

VII. Conclusion

In this paper, we highlight the impact of PV in a TPU systolic array under low-power operation. We demonstrate STRIVE—an energy efficient paradigm to identify and nullify the effect of LP-faults, and reclaim the performance from a TPU systolic array affected by faulty MAC units. STRIVE incurs minimum loss in inference accuracy for a TPU infested by a gate level fault rate of 1%. Additionally, STRIVE delivers 1.8× and 1.3× better TOPS/watt compared to FAP.

Acknowledgements

This work was supported in part by National Science Foundation grant CNS-2106237. 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. Jouppi, C. Young and others,"Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, 2018.
[2] M. Capra, R. Peloso and others,"Edge computing: A survey on the hardware requirements in the internet of things world," *Future Internet*, vol. 11, no. 4, p. 100, 2019.
[3] M. Seok, G. Chen and others,"Cas-fest 2010: Mitigating variability in near-threshold computing," in *J. Emerg Selec. Topics Cir. Sys*, vol. 1, no. 1, 2011, pp. 42–49.
[4] B. Reagen, P. Whatmough and others,"Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. of ISCA*, vol. 44, no. 3. IEEE Press, 2016, pp. 267–278.
[5] Y.-H. Chen, T. Krishna and others,"Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *JSSC*, vol. 52, no. 1, pp. 127–138, 2016.
[6] X. Jiao, M. Luo and others,"An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations," in *Proc. of ICCAD*. IEEE Press, 2017, pp. 945–950.
[7] D. Ernst, N. S. Kim and others,"Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. of MICRO*, 2003, pp. 7–18.
[8] J. J. Zhang, K. Basu and others,"Fault-tolerant systolic array based accelerators for deep neural network execution," *IEEE Design & Test*, vol. 36, no. 5, pp. 44–53, 2019.
[9] W. Zhao and Y. Cao, New generation of predictive technology model for sub-45 nm early design exploration," *T. Electron Devices*, pp. 2816 –2823, 2006.
[10] A. L. Maas, R. E. Daly and others,"Learning word vectors for sentiment analysis." Association for Computational Linguistics, 2011, pp. 142–150.
[11] Y. Netzer, T. Wang and others,"Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
[12] J. Stallkamp, M. Schlipsing and others,"Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. –, 2012.
[13] Y. LeCun and C. Cortes, MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist/, 2010.
[14] Reuters-21578 dataset," http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html, 2021.
[15] H. Xiao, K. Rasul and others,"Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747
[16] U. R. Karpuzcu, K. B. Kolluru and others,"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 and others,"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] D. Xu, C. Chu and others,"A hybrid computing architecture for fault-tolerant deep learning accelerators," in *ICCD*. IEEE, 2020, pp. 478–485.
[20] T. Spyrou, S. A. El-Sayed and others,"Neuron fault tolerance in spiking neural networks," in *Proc. of DATE*, 2021, pp. 743–748.
[21] B. Salami, E. B. Onural and others,"An experimental study of reduced-voltage operation in modern fpgas for neural network acceleration," in *DSN*, 2020, pp. 138–149.
[22] K. Givaki, B. Salami and others,"On the resilience of deep learning for reduced-voltage fpgas," in *Proc. of PDNP*. IEEE, 2020, pp. 110–117.
[23] Z. Tang, Y. Wang and others,"The impact of gpu dvfs on the energy and performance of deep learning: An empirical study," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, 2019, pp. 315–325.
[24] J. Lee, S. Kang and others,"The hardware and algorithm co-design for energy-efficient dnn processor on edge/mobile devices," *TCSI*, vol. 67, no. 10, pp. 3458–3470, 2020.