A Comprehensive Analysis of Transient Errors on Systolic Arrays

A Comprehensive Analysis of Transient Errors on Systolic Arrays

Eleonora Vacca, Sarah Azimi, Luca Sterpone
Politecnico di Torino, Dipartimento di Automatica e Informatica
Turin, Italy
{eleonora.vacca, sarah.azimi, luca.sterpone}@polito.it

***Abstract*— In recent years, the growth of interest in adopting deep neural network techniques across various domains led to new architectures for supporting the required computational effort. Tensor Processing Units (TPUs), which are based on a systolic array matrix multiplication unit (MMU), became widely popular thanks to their specific structure suitable for Artificial Intelligence. This work investigates Single Event Transient (SET) effects on TPU's MMU. The analysis demonstrates the impact of SETs on the functionality of MMU when executing digital image processing filtering. The experimental results identify the static and dynamic SET sensitivity of TPU and depict meaningful information on the data dependency of the filters' kernel values.**

***Keywords*— AI, Hardware Accelerator, Single Event Transient, Soft Errors, Tensor Processing Unit.**

I. INTRODUCTION

Recently, Deep Learning techniques are expanding in various application fields such as image classification, autonomous driving, and medical imaging. They require the execution of Deep Learning Neural Networks (DNNs) which elaborate large datasets to improve their autonomous learning skills and data, through hundreds of computational nodes organized in layers, processed in parallel to ensure real-time response capabilities [1].

The real-time execution of DNNs on traditional computing architectures (e.g., CPUs, GPUs) is limited in terms of performance and power by the load and store of intermediate results.

To overcome these limitations, the interest in application-specific architectures optimized for DNNs execution has risen [3]. While the training of DNN is mostly dominated by GPUs, Tensor Processing Units (TPUs) become one of the most interesting players in the inference hardware platforms. The TPU architecture is designed to limit memory access while handling the tremendous amount of multiplication-based operations that characterize DNN inference [4].

A crucial component of the TPU is the systolic array which is a grid of processing elements (PEs) able to perform basic math operations such as addition, element-wise multiplication, or matrix multiplication. A commercial example of a TPU architecture adopting a systolic array is the Google Tensor Core Processing Unit, which is based on a PE grid of 256 x 256 elements and it can provide a performance improvement of more than 85 times CPU or GPU computations [5]. In this paper, we use the systolic array architecture of the TinyTPU open-source design [6] that implements the Google Tensor Core architectural baseline. However, our approach can be applied to any systolic array for DNN acceleration and can be adapted to the interface modules of any TPU core.

Since TPUs are manufactured with nanometric CMOS technology, they are extremely sensitive to radiation-induced Single Event Transient (SET) [7]. Some research works have investigated transient faults mainly as bit-flips within the DNN computational architectural memory resources, therefore affecting inference weights and data during the AI task execution [8] while the generation of SET pulse in the combinational logics, propagating, broadening, or filtering during the propagation and eventual sampling by the memory resources is not evaluated yet.

The main contribution of this paper consists of techniques for the analysis of plain circuits and non-invasive architectural mitigation of the SET effects in the matrix multiplication unit (MMU) of systolic-array-based architecture. The circuit sensitivity assessment acts in two phases: (i) **static assessment**, using an in-house tool, to identify the sensitive circuit nodes where SET propagation is most pronounced leading to the higher probability of SET sampling by FFs or circuit I/O. This evaluation is based on the circuit topology, type of gates, and routing interconnection, used to build a netlist timing graph, covering all the possible pulse propagation cases(ii) **dynamic assessment**, to evaluate which of the sensitive nodes detected in the previous step, are vulnerable points at runtime, i.e. inducing application soft errors. This is performed by injecting the pulses identified in the previous step within the Datapath during the execution of an application. The injected pulse widths are based on the broadening and filtering effects estimated in the static assessment.
Experiments were conducted on a post-layout netlist of the systolic array synthesized with a 45nm high-performance technology library.

The paper is organized as follows. Section II gives an overview of the related works within the domain of fault-tolerance analysis and methods versus transient errors on TPU architectures. Section III provides the background on TPU architecture and the SET phenomena. Section IV elaborates on the methodology to assess the impact of SETs on the TPU systolic array architecture. Section V is dedicated to the deep experimental analysis conducted and elaborates on the achieved results. Conclusions and future works are finally drawn in Section VI.

II. RELATED WORKS

As the use of DNNs is increasingly extended to safety-critical domains, many previous studies dealt with the reliability assessment of neural networks [11][12][13].

On the other hand, the emergence of new systolic-array-based architectures aimed at accelerating artificial intelligence (AI) models introduced the need to correlate the faults affecting the hardware accelerator with the accuracy of the implemented models [14]. In this branch, most of the previous research converges on the analysis of permanent faults, mainly referring to stuck-at induced in the memory elements.

Authors in [15] evaluated the effects of faults occurring in the DRAM when employed by Google's TPU as the primary memory subsystem. The reliability assessment was performed by inducing errors in the form of bit-flips and evaluating the accuracy drop of the network while varying the bit position of the induced fault in the 8-bit weight stored in memory. The same authors evaluated the impact of stuck-at faults in the

979-8-3503-3277-3/23/$31.00 ©2023 IEEE

Datapath [16]. As a mitigation solution, they proposed a fault-aware training methodology where some of the accuracy loss due to faults in multiply and accumulate (MAC) units can be recovered by incorporating the fault effects in the backpropagation-based weight update. Moreover, authors in [17] proposed a fault-tolerant TPU architecture based on the prior knowledge of permanent fault location and the static weight map feature of the systolic-array architecture. They developed a mechanism that prunes all weights related to faulty MACs using a bypass path and retraining the network.

Further evaluation on stuck-at-zero affecting the MMU of a TPU is proposed in [18] where the authors extended the concept of [17] studying the impact of row and column faults on the network and designed a weight pruning method to bypass the faulty element. In [19], authors propose an API-based fault simulation to model both permanent and transient faults in systolic arrays. The method relies on developing a model of computation (MoC) and propagating the fault through sequential mathematical operations. A comparison between the reliability of Xilinx's DPU and custom systolic array implemented on SRAM FPGA is presented in [20] where Single Event Effects (SEE) are induced through device exposure to neutron beam. The same authors in [21] focus on the SEE on the matrix multiplication performed on systolic arrays implemented in SRAM FPGA. Here, transient faults are emulated through the bit-flip fault model in the device Configuration RAM.

The aforementioned approaches have shown a bias toward permanent faults in the datapath, with SEE receiving comparatively less attention. Physical fault injection techniques with particle beams while inducing SEE, lack of controllability and observability, thereby making the identification of failure sources challenging. Our method, on the other hand, identifies the most susceptible points in the design, reducing the time cost required for further analysis by enabling targeted testing of the detected sensitive areas.

III. BACKGROUND

A. Tensor Processing Units

DNN inference requires the computation of a huge amount of multiplications between huge matrices.

Traditionally, DNN tasks were accelerated by GPUs, since they can provide thousands of ALUs to sustain the computation effort. However, they are general-purpose co-processors that must support multiple applications and software. Therefore, the main limitation due to the application flexibility of GPUs is the need to access registers or shared memory to load and store intermediate results of computations. This translates into high power consumption.

Recently, the limitations of GPU-based DNNs were overcome through custom hardware architectures, such as TPUs. TPUs are application-specific computational cores dedicated to DNN inference with a focus on performance and power efficiency. A key optimization is the reduction of memory accesses during the execution of a task, achieved through the hardwired matrix organization of its PEs. Each PE is an independent unit equipped with a MAC. The core is a 2D-array in which PEs belonging to the same column are connected vertically to each other, as shown in Fig. 1. The result produced by each $MAC_{i,j}$ is directly passed as the second operand to the adder of $MAC_{i+1,j}$, without passing through a register file.

Each matrix of weights W defining a layer of the DNN is deterministically mapped onto the matrix of multipliers.

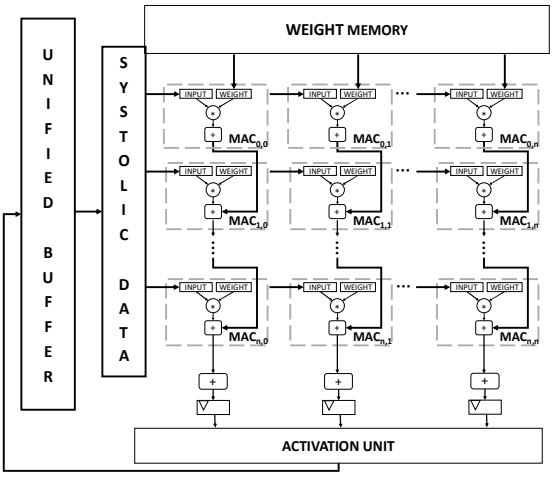

Fig. 1. Overview of the TPU internal structure.

During the computation, each $MAC_{i,j}$ always holds the same weight $w_{i,j}$ while the input data changes at each clock cycle spanning all the columns. Hence, the MMU is characterized by a vertical data stream associated with the MACs result, flowing row by row, and by a horizontal stream of input data that is transferred from column to column.

Since weight matrices of DNN layers are typically huge, despite the significant number of multipliers available in TPUs, it is unfeasible to process all the layer weights in parallel. Therefore, the weight matrix is split into submatrices of the same size as the MMU, and the layer operations are computed in multiple iterations. The result of each iteration is added to the previous one using accumulators placed at the end of the chains of MACs. Only at the end of the execution of all the multiplications that compose a layer, the final result of the accumulators is stored in memory.

IV. SINGLE EVENT TRANSIENT ANALYSIS ON TPU

To perform the transient error sensitivity analysis of the systolic array of the TPU under study, we implemented the systolic array circuit with the 45 nm Free PDK ASIC design library. The information regarding the physical implementation of the systolic array was used to perform two steps of sensitivity analysis. First, static sensitivity analysis is performed by exploiting an in-house Single Event Transient Analysis (SETA) tool [22] which identified the vulnerable nodes of the circuit under study. Secondly, the identified sensitive nodes are used for performing the dynamic sensitivity evaluation. To evaluate the dynamic sensitivity, the SET pulses were injected during the simulation of the post-synthesis netlist. The process reports the sensitive circuits node as well as the dynamic error rate. The developed SET analysis workflow is reported in Fig. 2.

A. Static Analysis of Single Event Transient

The SET sensitivity of the target circuit is performed by an in-house SETA tool elaborating on the Physical Design Description (PDD) file. This file contains a graph representation of the circuit, in which the combinational logic nodes are represented as intermediate nodes connected through routing segments, while I/O pins and sequential components are considered terminal nodes. SETA performs SET analysis by inserting SET pulse in each input and output of the intermediate nodes, propagating the pulse until it reaches terminal nodes while considering the propagation-induced pulse broadening effects. The static analysis is a pre-

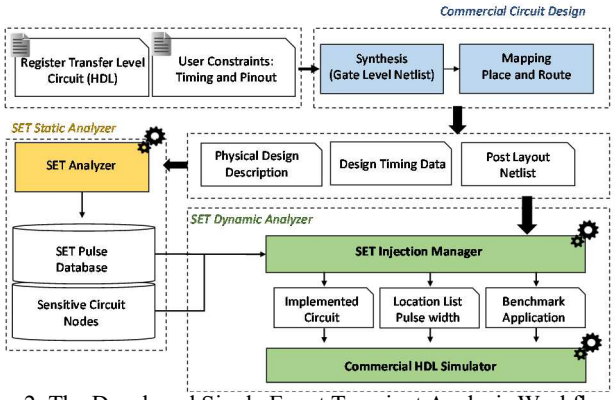

Fig. 2. The Developed Single Event Transient Analysis Workflow.

test approach, it does not require the actual device to be tested and it allows the identification of the most sensitive areas of the device. The reported information is then used to apply mitigation techniques targeting those sensitive areas.

B. *Dynamic Analysis of Single Event Transient*

The dynamic analysis is performed by instrumenting a simulation environment that inserts SET pulses in sensitive circuit nodes during the execution of the application while monitoring the output to estimate the dynamic error rate. This is achieved by stimulating the systolic array through the execution of computationally demanding tasks, such as 2D convolution. During task execution, the DUT is subject to the SET effect through the injection of a voltage glitch, where the position and characteristics of the pulse, such as amplitude and duration, are derived from the results of the static SET analysis. The output of the application is compared with the one produced by the golden circuit. Using the pulses and nodes identified by static analysis, not only the logical behavior is considered, but also the timing of the gates in the design. The broadening or the filtering of the SET pulse is mainly related to the logic and physical masking capability of the circuit.

V. SET Static And Dynamic Assessments

This section elaborates on the experimental campaigns conducted on a systolic array in terms of both static and dynamic analysis considering both the plain circuit and mitigated circuits, i.e., with filtering techniques applied. Benchmark applications are image-filtering tasks that use popular kernels in the field of digital applications and are particularly exploited in AI for feature extraction.

A. *Implementation of Systolic Array*

The systolic array taken as a case is based on the open-source TPU architecture presented in [6]. For the sake of this paper, we only focus on the multiplier array, similar to related works, due to the amount of sensitive area of MMU resources. In compliance with the commercial implementation, the architecture works with 8-bit integer input data. On the other hand, the MMU core was sized to 3x3 MAC units. The reduced size aims to comprehensively investigate the circuit topology and its sensitivity to SET by covering a variety of case studies.

The 3x3 MMU was synthesized and implemented with a 45nm high-performance FreePDK library. Accordingly, the graph of the circuit is generated. The graph description of the circuit, post-layout netlist, and timing information are provided to the SETA to identify the sensitive circuit nodes.

TABLE I
3x3 MMU post-Layout Characteristics

Circuit	Logic Gate [#]	Sequential Gate [#]	Performance [MHz]	Power [µW]	Sensitive FFs [#]
Original	20,039	2,377	74.54	840	128
300ps	25,248	2,377	68.70	848	119
600ps	27,540	2,377	62.45	865	94

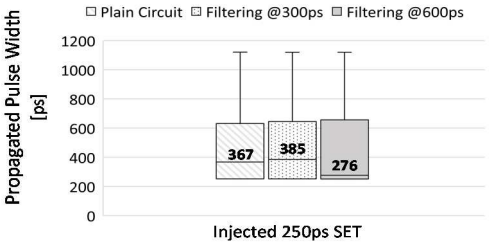

Fig 3. SET broadening effect on the Plain Circuit and the mitigated designs.

B. *Static Analysis of Single Event Transient*

SET analysis is used to determine the likelihood of an SEU occurring in memory elements of the circuit. It is performed by the SETA tool which simulates the effects of the impact of high-energy particles on different regions of the circuit and then computes the probability of an SEU occurrence.

The analyzer performs SET injections on the post-layout implementation netlist into each combinational path and evaluates their propagation considering the electrical broadening and filtering characteristics up to each terminal node. Terminal nodes are labeled as sensitive when they are reached by a SET, injected at any position in the path. Static evaluation of sensitivity to SETs was performed through two campaigns. An exhaustive one for a fixed pulse width of 250ps and an investigation with pulse widths varying from 150ps to 450ps on a random subset of paths.

1) *Exhaustive SET Evaluation*

In the exhaustive SET evaluation of the MMU, a SET with the amplitude 1V and width 250ps was inserted into each of the 5,819 combinational logic nodes of the circuit. According to the analysis, only 128 of the terminal nodes (FFs) are sensitive, i.e., the pulse propagated up to their input. In such nodes, the broadening effect due to gate-to-gate propagation increased the pulse widths from 250ps up to 367ps on average.

Based on this information, the design was selectively mitigated by inserting a delay-based filtering circuit at the input of the concerned sensitive nodes [22].

The impact on the number of sensitive nodes and the broadening effect was evaluated with delay factors of the SET filtering circuit of 300ps and 600ps.

As it is possible to notice, the 300ps and 600ps filtering factors reduce the number of sensitive FFs with respect to the plain design, by 7% and 26% respectively. However, the 300ps also induces a worsening in the pulse broadening effect, as shown in Fig. 3, thus increasing the probability of SET sampling. On the other hand, the 600ps solution improves also this aspect, achieving a reduction of 25% in the average propagated pulse width. Full details including power, performance, and area of the three implemented designs are shown in Table I. To characterize the architecture more comprehensively and avoid the correlation to the particular pulse width the behavior and the sensitivity to SETs, a second

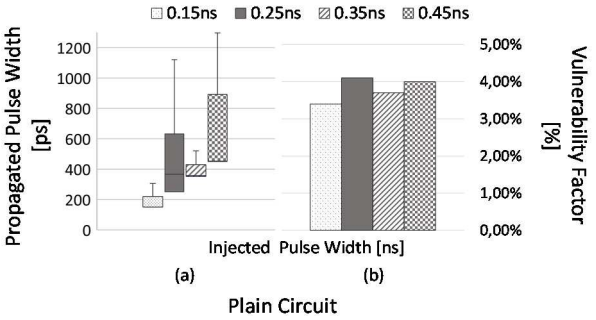

Fig. 4. SET sensitivity analysis on the Plain Circuit for variable pulse width (a) broadening effect (b) vulnerability factor.

static analysis campaign was performed in which different pulse widths were tested.

2) Exploration with variable SET pulse width

The experiment conducted considered pulses of width 150ps, 250ps, 350ps, and 451ps. For each pulse width, 1000 SETs were injected into random combinational nodes of the plain circuit and propagated to the terminal nodes.

The experimental results show two noteworthy aspects. From the propagated pulse width distribution presented in Fig. 4.a, it appears that the circuit is more susceptible to 250ps and 450ps SET and that the circuit topology is prone to broadening these pulses.

SETA reports the overall circuit sensitivity to SET effects expressed as the vulnerability factor. The vulnerability factor is computed as the total number of circuit paths that undergo transient pulse propagation over the total number of injected transient pulses. The circuit vulnerability, plotted in Fig. 4.b. tends to be stable around an average value of 3.8%. Thus, neither a larger pulse width nor a larger broadening effect implies an increase in the number of sensitive nodes. To confirm this saturated trend of such an architecture, an additional campaign was run with a pulse width of 1000ps. The results show that with a pulse width 7 times the smallest pulse width tested of 150ps, there is only a 2% increase in sensitive FFs.

A final key aspect of static campaigns is the location of sensitive FFs. A cross-analysis of the various experimental results obtained showed that the most sensitive nodes are those predominantly associated with critical FFs within individual MAC units that receive the value of weight or input data to be processed. Thus, potentially compromising the functionality during task execution.

C. Dynamic Analysis of Single Event Transient

Since the memory elements related to weights storage were found to be among the most sensitive within the architecture and because weights' values and their organization in the systolic array define the type of task to be implemented, the impact of SETs on the systolic architecture during the execution of computation has been evaluated dynamically. As a case study, it was decided to analyze the architecture SET sensitivity when performing popular digital image processing filtering operations. A secondary goal of dynamic analysis is also to assess the influence of the data patterns on susceptibility to SETs, with the idea that certain patterns can be more robust than others. To fulfill this dual purpose, four different kernels were chosen to investigate whether weights' values have a SET mitigation impact. The used kernels and the effects on an image are presented in Fig. 5. The dynamic SET injection was performed by

TABLE II
SOFT-ERROR RATE FOR SET INJECTIONS AT EACH SENSITIVE NODE IDENTIFIED DURING THE STATIC EXHAUSTIVE CAMPAIGN.

	Edge Detection	Sharpen	Emboss	Bottom Sobel
Soft Errors [%]	44.89	25.28	44.53	44.28
Dyn.Sens.Nodes[%]	48.43	30.34	48.43	48.43

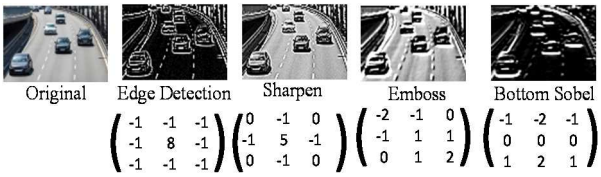

Fig. 5. Image filters used during the SET dynamic evaluation.

instrumenting a commercial HDL simulation environment. In particular, the SET was implemented by forcing a signal to logic state 1 for a time interval corresponding to the pulse width.

The injection time of the SET is random but reproducible since it is established before the start of the simulation. The signal forcing is superimposed on the normal signal behavior but with higher priority. Therefore, the signal restores its current correct logic state when the injection duration (i.e., pulse width) expires. The detected sensitive nodes coming from the static approach described in Section V-B have been meticulously investigated by performing different dynamic injection campaigns.

1) SET injection experiment on the unmitigated design

The dynamic injection considers SETs having fixed pulse width as the estimated broadening effect reported by the SET analyzer. However, in avoidance of losing the stochasticity typical of SETs, the choice of the target injection instant is kept random. It follows that even if the SET is sampled, it may be the case that the current computation step is not using, or will not be used in the next clock cycles, the data stored in the affected memory element. Thus, even if the SET turns into an SEU, this does not strictly imply a soft error in the running task. For each sensitive FF detected during the exhaustive static analysis presented in Section V-B.1, 100 SETs have been injected, each evaluated for different values of the image filter. Table II shows the results of the experiment as a percentage of soft errors detected during the task execution. A soft error is identified when there is a computational mismatch concerning the application's golden result. Due to propagation, a single SET may induce more than one computational discrepancy in the same run. However, the corruption event is considered a single soft error regardless of the number of different values. Therefore, the percentage of soft error reported in Table II indicates how many of the total injected SETs (related to different nodes) resulted in at least one computational mismatch compared to the golden output. The severity of the errors concerns the number of corrupted output pixels, which depends on the SEU location. Specifically, bit position and meaning of the data stored (i.e., input, weights, partial result, etc..) lead to different soft errors. Specifically, transients captured by FFs used to store a weight value inside a MAC unit are more likely to affect all the computations, since we adopted a Weight Stationary (WS) data mapping policy. As an example, we propose the effect of transient pulse captured by weight register bit 2 of $MAC_{4,0}$ in Fig. 6. The percentage of sensitive nodes is calculated starting from the total number of nodes tested. A node is targeted

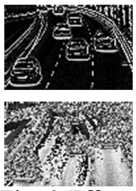

Fig. 6. Effects of a sampled SET.

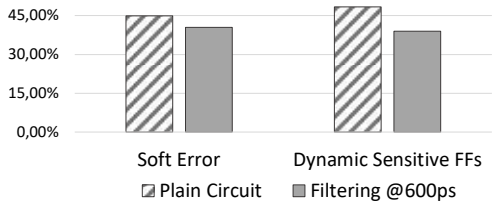

Fig. 7. Result of the dynamic SET injection performed on the mitigated version of the systolic array.

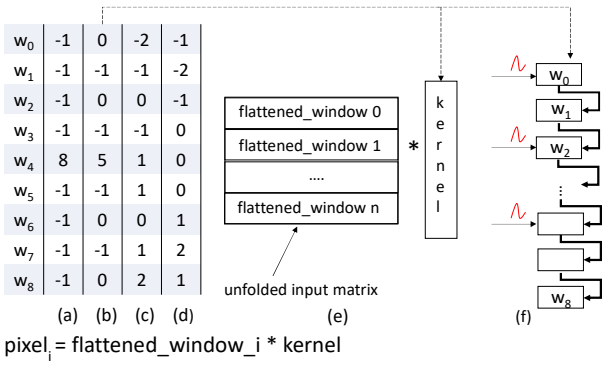

Fig. 8. The Filters column representation and TPU computational structure concerning the kernel sliding window.

dynamically sensitive when at least one of 100 injected SETs causes a soft error.

The preliminary conclusions that can be drawn from these experimental results are that (i) not all FFs found to be sensitive during static analysis are also sensitive during the actual execution of an application (ii) as the 2D convolution kernel value changes, the behavior of SETs and their effect remains almost stable except for the Sharpen filter.

We also performed a dynamic evaluation considering the case of variable pulse width, using the timing information and sensitive nodes identified during the static analysis. We did not experience any relevant aspect confirming that there is no particular dependence on the SET pulse width even in soft error induction. It is worth noting that the experimental results of Table II show a one-to-one correspondence between soft error and sensitive nodes. The percentages of the two metrics only differ slightly. This aspect is translated in the fact that a SET injected in those sensitive FFs, at any instant during task execution, generates a soft error 90% of the time.

2) SET injection experiment on the mitigated design

Section V-B discussed the implementation of a mitigated version of the circuit, based on a method of filtering transients by delaying the input data to the FFs. The experimental results of the static analysis on the mitigated designs were further investigated using the dynamic approach.

The results illustrated in Fig. 7 are consistent with those obtained statically since the filtering approach, by reducing the number of sensitive nodes, also results in a reduction of application soft errors.

D. Remarks on the Experimental Results

Our analysis of the systolic array has revealed that the design is relatively resistant to SETs, thanks to its highly pipelined architecture. However, we have identified the weight and input data register FFs as the most vulnerable nodes in the system, as they play a crucial role in determining the functionality of the executed application. Therefore, although the number of sensitive nodes is rather low, the criticality and impact on the software application must be considered. A canonical approach such as the hardware filtering methodology proposed during static analysis returns a significant improvement. On the other hand, results from the dynamic analysis suggest that better filtering action can be achieved by choosing weight data patterns strategically, e.g., the sharpen filter. Interestingly, we observed a significant reduction in the impact of SETs, up to 50%, in this particular filter even though the SETs used were identical to those used in the other filters.

Considering the TPU instruction set architecture, the 2D convolution used to perform the image filtering task has been implemented by arranging the input data and kernels as shown in Fig. 8, where it is possible to depict the filter weight values for the edge detection (a), sharpen (b), emboss (c) and bottom Sobel (d). Furthermore, it is possible to observe how the weights (f) are used in the unrolled version of the 2D convolution concerning the kernel (e) sliding windows.

Each flattened-column type kernel is mapped to a single MAC column, which, as shown in Fig. 8.f, are concatenated. It is possible that the alternating zero-weight disposition in the sharpen filter and the flattened-column configuration, Fig. 8.b , may embody a blocking nature in the transmission of SET-induced errors associated with the input tensor.

In proof of that, a cross-checking between the list of detected sensitive nodes for each kernel filter, reveals that edge detection, bottom Sobel, and emboss filters share the same sensitive FFs, while for the sharpen case, missing nodes are associated exclusively with the input tensor FFs.

Considering the data streams inside the MMU, as defined in Section III, adopting the WS policy implies that each wij element of the flatten-column kernel is loaded into its respective MACij before the image windows start flowing into the array. Hence, they do not change during the computation. while the input data is changing at each clock cycle.Therefore, having a pattern of alternating zero weights in the kernel creates multiple SET-blocking points dislocated in time, since if input data are affected by SET, their contribution is nullified by the data application itself, hence preventing error propagation.

On the other hand, a different assignment between weights matrix-MACs matrix shows even different dynamic behavior.

The same filters arranged as a 2D array as originally, show an increase of soft errors up to 70%, and the sensitive nodes, even in the sharpen case, increase up to 83%. Again, the difference in behavior, given the same data, is to be found in the systolic array structure itself and in the way the processing elements are connected, sharing inputs and transferring partial sums. Therefore, not only does the susceptibility to SETs depend on data patterns, but also on the type of algorithm that is implemented in the architecture, given the same sensitive nodes, resources, and numerical values.

These aspects need to be studied in depth but they certainly pave the way toward new non-invasive mitigation techniques. In modern NN frameworks it is possible to define custom functions to initialize weights and to use the *weight freezing* technique [26] to prevent them from being learned during the training phase. Thi feature can be exploited to assign strategic data patterns. Hence, improving application reliability via pre-training weight initialization, not requiring additional hardware resources or architectural modification.

1) Improving Application Reliability through Weights Initialization in Fixed Pattern

To evaluate the effectiveness of the proposed methodology, a modified edge filter test case, incorporating

the sharpen filter structure, has been realized. The filter and its results on images are shown in Fig. 9 compared with the original and the sharpen. Before conducting further SET evaluations on the sensitivity of the proposed filter, its validity was tested within a more complex application.

Typically edge detection filters are used in autonomous driving applications for lane detection. Therefore, two different lane detection applications were implemented, one using the canonical edge kernel and the other using the modified one. The comparable effect of the two filters is shown in Fig. 10. The modified edge produces a less pronounced detection that results in minimal deviation in the final processing result (green lane). This confident result prompted the evaluation of the effects of SETs on the new filter when used under the same computation conditions (values, resource organization) as the original edge case.

Experiments were conducted by injecting the same SETs used for the original edge filter campaign presented in Section V-C. The results confirm a reduction of SET-induced soft error by 50% as well as a reduction in sensitive nodes, which are reported in Fig. 11.

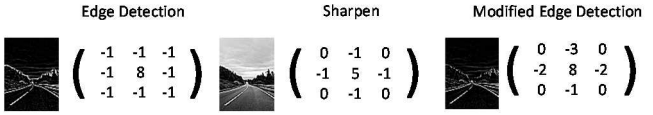

Fig. 9. The SET-optimized edge detection filter and its effect compared with the original and the sharpen one.

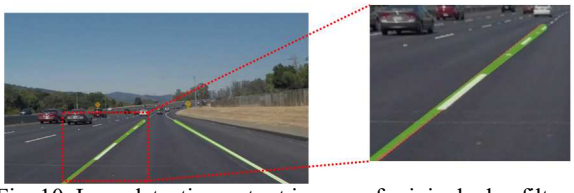

Fig. 10. Lane detection output in case of original edge filter (red lane) and the proposed modified version (green line).

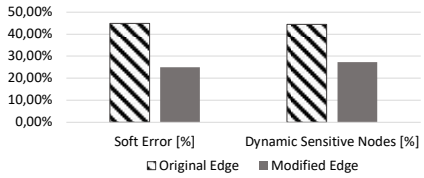

Fig.11. Dynamic SET injections comparative results obtained for the edge filter and its modified version.

VI. Conclusions and Future Works

This paper presents an in-depth analysis of SET effects on systolic array architecture embedded in TPU cores. The analysis conducted at the post-implementation netlist level revealed a slight criticality of these architectures concerning SETs. It was found that the most sensitive nodes are internal to the individual processing elements of the grid, which are used for the storage of network weights and inputs to be processed. Therefore, the adoption of a mitigation technique based on the addition of glitch-filtering circuitry at the input of the sensitive nodes was evaluated. The experimental results obtained on the mitigated design indicate that hardening of the structure can be achieved without performance loss. TPU sensitivity was additionally assessed during the execution of computational tasks, which revealed a dependence on both the values of the data involved and the algorithm employed. The latter aspects pave the way for future work where these discoveries are intended to be further explored and exploited to achieve non-intrusive mitigation techniques that do not involve hardware modifications.

References

[1] Y.-H. Chen, et al., "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292-308, June 2019.

[2] N. P. Jouppi et al., "In-data center performance analysis of a tensor processing unit," *ACM/IEEE International Symposium on Computer Architecture* (ISCA), 2017, pp. 1-12

[3] Z. Wu, et al., "Comprehensive Survey on Graph Neural Networks," in *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[4] D. Shin, et al., "DNPU: An Energy-Efficient Deep-Learning Processor with Heterogeneous Multi-Core Architecture," in *IEEE Micro*, 2018.

[5] N. P. Jouppi et al., "Ten Lessons From Three Generations Shaped Google's TPUv4i: Industrial Product,"*ACM/IEEE Annual International Symposium on Computer Architecture* (ISCA), 2021.

[6] J. Fuhrmann, " Implementierung einer Tensor Processing Unit mit dem Fokus auf Embedded Systems und das Internet of Things", 2018.

[7] R. L. Rech et al., "Reliability of Google's Tensor Processing Units for Embedded Applications," *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022*, pp. 376-381.

[8] G. Yuan et al., "Improving DNN Fault Tolerance using Weight Pruning and Differential Crossbar Mapping for ReRAM-based Edge AI," *International Symposium on Quality Electronic Design (ISQED)*, 2021.

[9] M. Hasan et al.,"Tolerance of Deep Neural Network Against the Bit Error Rate of NAND Flash Memory," *IEEE International Reliability Physics Symposium* (IRPS), 2019, pp. 1-4.

[10] K. T. Chitty-Venkata et al., "Impact of Structural Faults on Neural Network Performance," *IEEE 30th International Conference on Application-specific Systems, Architectures and Processors* (ASAP), 2019, pp. 35-35.

[11] C. De Sio, et al, "An Emulation Platform for Evaluating the Reliability of Deep Neural Networks," *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT),* 2020, pp. 1-4.

[12] N. I. Deligiannis, et al., "Improving the Fault Resilience of Neural Network Applications Through Security Mechanisms," *IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, 2022, pp. 23-24.

[13] S. Azimi, B. Du, L. Sterpone "Evaluation of transient errors in GPGPUs for safety critical applications: An effective simulation-based fault injection environment", in *Journal of Systems Architecture*, 2017.

[14] A. Chaudhuri, et al.,"Special Session: Fault Criticality Assessment in AI Accelerators," IEEE 40th VLSI Test Symposium (VTS), 2022.

[15] S. Kundu et al., "Special Session: Reliability Analysis for AI/ML Hardware," 2021 IEEE 39th VLSI Test Symposium (VTS), 202.

[16] S. Kundu, et al., "Toward Functional Safety of Systolic Array-Based Deep Learning Hardware Accelerators," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 3, pp. 485-498.

[17] J. J. Zhang et al.,"Fault-Tolerant Systolic Array Based Accelerators for Deep Neural Network Execution," in IEEE Design & Test, vol. 36, no. 5, pp. 44-53, Oct. 2019.

[18] K. T. Chitty-Venkata et al. "Model Compression on Faulty Array-based Neural Network Accelerator," IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC), 2020, pp. 90-99.

[19] P. Omland et al., "API-Based Hardware Fault Simulation for DNN Accelerators," in IEEE Design & Test, vol. 40, no. 2, pp. 75-81, April 2023.

[20] F. Libano et al.,"On the Reliability of Xilinx's Deep Processing Unit and Systolic Arrays for Matrix Multiplication," 2020 20th European Conference on Radiation and Its Effects on Components and Systems (RADECS), Toulouse, France, 2020, pp. 1-5.

[21] F. Libano et al., "Efficient Error Detection for Matrix Multiplication with Systolic Arrays on FPGAs," in IEEE Transactions on Computers.

[22] S. Azimi, et al., "A new CAD tool for Single Event Transient Analysis and mitigation on Flash-based FPGAs," Integration, the VLSI Journal, 2019.

[23] Kumar Chellapilla, et al.,"High-Performance Convolutional Neural Networks for Document Processing". International Workshop on Frontiers in Handwriting Recognition, 2006.

[24] S. Azimi, et al., "On the Prediction of Radiation-induced SETs in Flash-based FPGAs", in Elsevier Microelectronics Journal, 2016.

[25] C. De Sio, S. Azimi, A. Portaluri, L. Sterpone," SEU Evaluation of Hardened-by-Replication Software in RISC-V Soft Processor", in EEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, 2021.

[26] Y. Han et al., "Improved convolutional neural network algorithm based on weight freezing method," 2018 24th Asia-Pacific Conference on Communications (APCC), Ningbo, China, 2018, pp. 341-346,