Analyzing the SEU-induced Error Propagation in Systolic Array on SRAM-based FPGA

Analyzing the SEU-induced Error Propagation in Systolic Array on SRAM-based FPGA

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 this paper, we evaluated the radiation-induced Single Event Upset of an open-source TPU-like platform implemented on SRAM-based FPGA while its high performance parallel datapath is exploited to implement multiple feature extractions task.**

***Keywords*—Systolic Arrays, Tensor Processing Unit, Hardware Accelerator, FPGA, Single Event Upset, Parallel Processing.**

I. Introduction

As the workload required to run sophisticated Neural Networks (NN) increases, requiring multiple feature extraction from raw data to perform object detection and classification, the need for hardware accelerators capable of meeting the computational demand by ensuring high-performance, low-power, and real-time response is growing. Modern detection systems such as Convolution Neural Networks (CNNs) perform feature extraction by applying multiple filters on the input data exploiting the convolution operator and meet the real-time response of AI-based systems, filters need to be applied in parallel. Hence, new domain-specific hardware accelerators have been proposed. Specifically, systolic-array-based accelerators, such as Tensor Processing Units (TPUs), gained popularity thanks to their specialized design that enables to process data in a highly parallel and efficient manner [1]

As the CNNs approach is spreading in numerous safety-critical applications such as autonomous driving, medical imaging, and surveillance systems, several works addressed the reliability of systolic array-based accelerators performing fault injection at different abstraction levels [2][3][4] [6][6], mostly referring to ASIC implementations of such accelerators, focusing on permanent or transient faults.

On the other hand, the new generation of high-performance FPGAs equipped with on-chip DSP offers a new perspective of implementation solutions for systolic array-based accelerators by reducing costs and gaining flexibility. However, when a systolic array is implemented on SRAM-based FPGAs, reliability becomes the major concern since radiation-induced alteration of FPGA configuration memory (CRAM) can cause a structural change to the implemented design and then propagate up to the application level [7].

In this work, we evaluate the SEU effects, occurring in the CRAM, on the parallel processing capability of systolic array-based accelerators when implemented on SRAM-FPGA, conducting an accurate analysis of the circuit typology and correlating it to the functional behavior of the computation cores. Combining the high-level features of the circuit with the FPGA architecture, and analyzing the mapping between logical and physical resources on the device, we propose a fault propagation model by targeting the critical resources for the design. Then, by exploiting the bit-flips fault model in the CRAM, to emulate radiation-induced SEU, we evaluate the effectiveness of the proposed approach.

In our experimental campaigns, we considered single fault scenarios and multiple filters parallel application to (i) analyze the fault propagation on the parallel processing (ii) evaluate the correlation between the error rate and the filters' data values. Finally, we profile the most sensitive resources by correlating application-level errors with resources affected by SEU. The collected experimental results proved the validity of our fault model.

II. Systolic Arrays

Systolic arrays are widely used as AI engines to speed up the computational load required by neural networks (NN). They are equipped with a large number of identical processing elements (PEs) operating in parallel.

Systolic arrays are characterized by a peculiar Datapath where PEs are organized as a two-dimensional array and interconnected through homogeneous, modular fixed paths designed for efficient data flow and easy scalability. Indeed, interconnection layout plays a key role in the systolic array computation mapping.

Specifically, PEs belonging to the same column of the 2D array contributes to the production of one output value, which propagates top-down, from one PE to the following one. Complementary, the path of interconnections between PEs of the same row is designed to transfer inputs. Fig. 1a presents a schematic overview of the systolic array datapath.

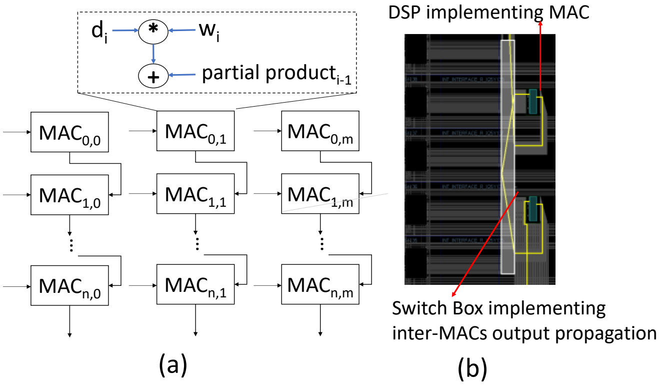


Fig. 1. Overview of systolic array datapath (a) the circuit topology, in (b) the corresponding hardware implementation on FPGA.

This specific interconnection layout, combined with high-performance PEs mapping Multiply and Accumulate (MAC) operation makes the systolic array suitable to speed up matrix multiplication. Indeed, AI applications require performing multiplications between large arrays for feature extraction, object detection, and classification. Even convolution operations can be accelerated through systolic arrays. By exploiting per-column parallelism and computational isolation, it is possible to map 2D convolutional kernels separately per PEs' column, after expanding them into 1D vectors and transposing them. Therefore, each column will be in charge of convolving the input data samples for the corresponding kernel, performing a row-by-column multiplication.

III. SEU-INDUCED FAULTS PROPAGATION MODEL ON SYSTOLIC ARRAY IN FPGA

Taking advantage of the heterogeneous architecture of the latest generation FPGAs, which are equipped with numerous on-chip DSPs (configurable as MAC units), the implementation of systolic array-based accelerators in programmable logic is gaining popularity.

When systolic arrays are implemented on FPGAs, the interconnection path is synthesized and implemented through Switch Box. A Switch Box is a tile in the FPGA architecture that groups programmable routing muxes, called Programmable Interconnection Points (PIPs), used to route signals. PIPs connect two wires in the same tile. Switch boxes are designed to connect to all fabric resources, such as CLB, DSP, and BRAM, each assigned to a different FPGA tile.

Fig.1b reports a portion of the post-implementation design of a systolic array on the Xilinx design tool. It shows the mapping of resources used to implement a column of cascaded MAC units. Highlighted in yellow is the output propagation from one PE to the following one, while the blue boxes are DSPs.

Each PIP's state is encoded as a sequence of bits in the CRAM, properly set to avoid signal conflicts or routing congestion. Taking as reference the Xilinx Artix7 FPGAs, each Switch Box contains roughly 3,000 PIPs. As such, a large portion of the configuration memory is dedicated to encoding PIPs. Moreover, DSPs are implemented as hardwired computational units and configured only by routing proper signals to their interface. As a consequence, SEU can affect DSP behavior only if a routing resource is hit. Logic resources (LUTs) become marginal in this design topology and are mainly used to implement very basic control logic functionalities.

Additionally, the programmability and flexibility of FPGAs come at the price of resource sharing since a single Switch Box is typically used to route multiple signals belonging to different modules.

These observations suggest that abnormal functioning of the design can be primarily attributed to SEUs occurring in the memory bits employed for interconnection configuration. This is particularly noteworthy in systolic arrays, where the layout of interconnection paths plays a critical role in shaping the processing core's behavior. A single fault in the Switch Box can be catastrophic for the entire application.

From the electrical point of view, the effects of SEUs in PIP configuration bits result in open faults and bridge faults. Open faults are interruptions in the routing path of a signal, typically associated with the radiation-induced flip of the value of the PIPs configuration memory cells from 1 to 0. Similarly, a bridge fault is due to the creation of a short between two or more signals, routed in the same Switch Box due to a bitflip from 0 to 1.

By combining the aforementioned fault models with circuit topology and physical resource mapping we are able to draw the fault propagation path and estimate which fault locations are the most critical from the application point of view. Fig 2 shows the two cases of open faults happening to different data signals. To elaborate more, in Fig. 2a an open fault (red dashed line) happening to the output propagation of one MAC unit is illustrated while in Fig. 2b, the open fault occurring on the input of a MAC is shown.

Although faults share the same nature, the severity of the error induced is quite different. Following the red path in Fig 2, it can be drawn that an open fault occurring on the signal driving the output propagation in a column of resources is less critical when compared to one open fault in one input data.

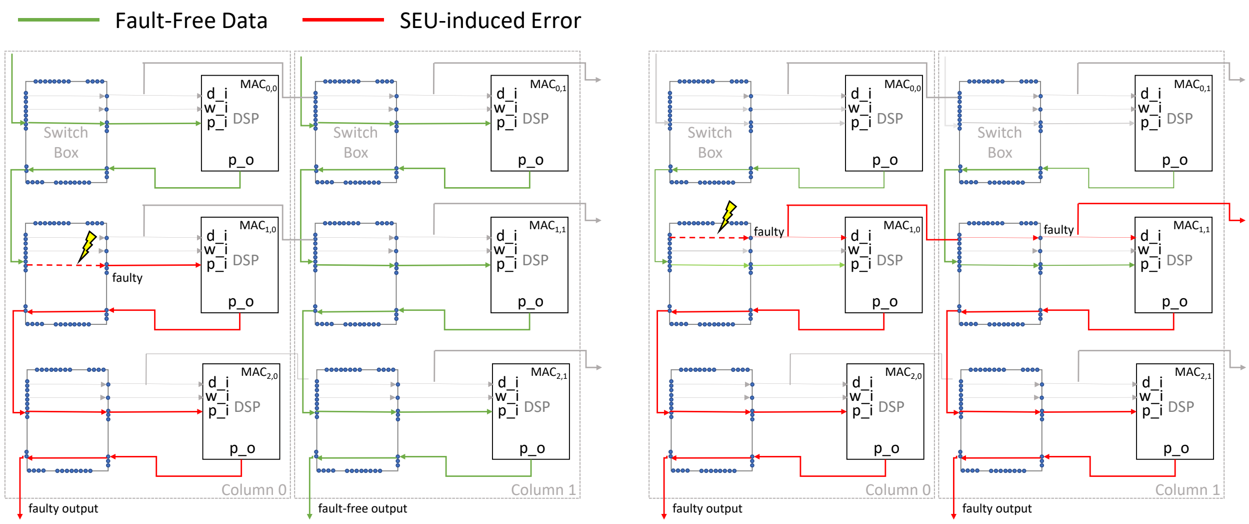
Considering the first case, each DSP has the task of producing an output value p_O as the sum of its current computation with the computation produced by the previous DSP in the same column p_I. Both p_O and p_I are partial products, received as input and produced as output, respectively.

Specifically, the partial product production of each $DSP_{i,j}$ can be formalized by the following equation:

$$p_{O_{i,j}} = p_{I_{i,j}} + (d_i * w_{i,j}) = p_{O_{i-1,j}} + (d_i * w_{i,j}) \quad (1)$$

where p_O is the output partial product generated p_I is the input partial product, d_i is the input sample and $w_{i,j}$ is the weight data.

A fault that happens in the output propagation routing path can occur either on the signal that drives the p_I input or on



(a) Open fault in partial product propagation

(b) Open fault in the input sample propagation

Fig. 2. SEU-induced error propagation in systolic arrays datapath. In (a) the propagation of an SEU-induced open fault on the routing path of the output propagation signal, while in (b) the case of an open fault affecting the routing path of the input sample.

the signal that carries the newly produced output p_O to the next DSP.

If we consider a fault in p_i as shown in Fig. 2b and propagate it to the next computations, represented in Equation 2,

$$\begin{aligned}
p_{o_{i,j}} &= p_{i_{i,j}} + (d_i * w_{i,j}) \qquad (2)\\
p_{o_{i+1,j}} &= p_{o_{i,j}} + (d_{i+1} * w_{i+1,j}) = p_{i_{i,j}} + (d_i * w_{i,j}) + (d_{i+1} * w_{i+1,j})\\
p_{o_{i+2,j}} &= p_{o_{i+1,j}} + (d_{i+2} * w_{i+2,j}) =\\
&= p_{i_{i,j}} + (d_i * w_{i,j}) + (d_{i+1} * w_{i+1,j}) + (d_{i+2} * w_{i+2,j})
\end{aligned}$$

we can observe that starting from $DSP_{i,j}$ receiving the faulty $p_{i_{i,j}}$ partial product, the error propagates throughout the column computations, involving all the units that follow the corrupted one.

This behavior implies that (i) fault remains localized in only one column of computations, while the computations performed in parallel by the other columns are preserved, (ii) all processing performed by the affected column will produce *potentially erroneous* output at each iteration.

Because of the input data sharing mechanism between DSPs of different columns, used to facilitate parallel processing and reduce memory accesses, the second fault model has a greater propagation impact. Taking Equation (1) as a reference and applying it to the output produced by resources belonging to different columns, Equation 3,

$$\begin{aligned}
p_{o_{i,j}} &= p_{i_{i,j}} + (d_i * w_{i,j}) \qquad (3)\\
p_{o_{i,j+1}} &= p_{i_{i,j+1}} + (d_i * w_{i,j+1})
\end{aligned}$$

we can observe that DSP units having the same row index process the same input data d_i. Hence, if we consider a fault on d_i and let it propagate,

$$\begin{aligned}
p_{o_{i,j}} &= p_{i_{i,j}} + (d_i * w_{i,j}) \qquad (4)\\
p_{o_{i,j+1}} &= p_{i_{i,j+1}} + (d_i * w_{i,j+1})\\
p_{o_{i+1,j}} &= p_{o_{i,j}} + (d_{i+1} * w_{i+1,j}) = p_{i_{i,j}} + (d_i * w_{i,j}) + (d_{i+1} * w_{i+1,j})\\
p_{o_{i+1,j+1}} &= p_{o_{i,j+1}} + (d_{i+1} * w_{i+1,j+1}) =\\
&= p_{i_{i,j+1}} + (d_i * w_{i,j+1}) + (d_{i+1} * w_{i+1,j+1})
\end{aligned}$$

as represented in Equation 4, we can observe that starting from the resources at row index i receiving corrupted input, the error propagates both horizontally and vertically, involving all columns. This behavior implies that all the computations executed by the whole systolic array may be characterized by a *potentially erroneous* output.

It is good to emphasize the *potentially erroneous* terms since, in addition to the propagation mechanism, other factors come into play that may mask the faults. These include the rounding mechanisms adopted during output propagation, the value of the data involved in the calculations, and, considering that a PIP route a single-bit data signal, the bit position in the data (i.e., MSBs, LSBs, etc.).

As an explanatory example, let's consider the potentially most critical scenario of a fault affecting an input data d_i. Although the fault reaches multiple resources at the same time, the corrupted data is multiplied by different values $w_{i,j}$ in each involved DSP. Hence, $w_{i,j}$ value may mask (or accentuate) the fault-induced error, which means that despite all columns having a DSP receiving corrupted data, fault effects and its propagation depends also on the data values of the other operands involved in the computations. Therefore, despite all the columns being faulty, they may not show the same faulty behavior. Also in the case of a fault in the output propagation mechanism, the fault may be masked.

When systolic arrays are employed to accelerate NN applications also weight input $w_{i,j}$ in DSP plays a key role. However, there's no propagation mechanism that directly involves weight data induced by the systolic array interconnection path topology. Moreover, SEU-induced weight data corruption, either due to a bitflip in memory elements or to interconnection fault, directly involves just one DSP computation at a time. Hence the impact of the fault strongly depends on the executed application.

The fault propagation model obtained by combining the circuit topology and its functional behavior with the FPGA architecture and resource mapping enable us to predict which resources are most critical in the design. Specifically, the interconnections related to input data turn out to be the most critical in relation to workloads in which all the array resources are used in parallel to process the same data. Thus, from a purely theoretical point of view, we expect that even a single SEU can have critical consequences on the parallel processing capabilities of the architecture.

IV. Experimental Analysis

To evaluate the propagation of the SEU from the hardware level to the application we implemented the open-source systolic array based TinyTPU [8] architecture on the Xilinx Zynq XC7Z020 SoC which incorporates an SRAM FPGA. The systolic array inside the TPU main core is equipped with 14 x 14 PEs, exploiting all the available on-chip DSPs. Table I reports the post-implementation details of the circuit. Please note that additional DSP are used outside the systolic array main core to implement additional accumulator's module required by the TPU architecture.

Table 1: Resource Utilization of the Hardware Accelerator

Resource	Used [#]	Available [#]	Utilization [%]
LUT-Logic	3723	53200	7.0
LUT-Memory	235	17400	1.35
FF	6108	106400	5.74
DSP	218	220	99.09

In order to evaluate the consequences of SEU on the parallel processing capabilities of the systolic array, we chose a multiple-feature extraction task as a benchmark. The task consists of applying convolutional filters in parallel on the same input image.

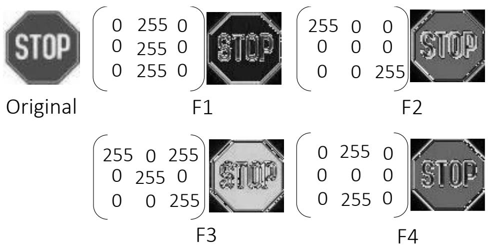


Fig. 3. The evaluated filters and their effects on the original image.

To efficiently handle feature extraction on the systolic array, the convolution operation is converted to a General Matrix Multiply operation. Each filter is expanded into a 1D vector, transposed, and assigned to a column in a matrix matching the size of the systolic array. The input image required the Img2col matrix transformation [9]. The creation of the two matrices, one for the image data and one for the filters allows for efficient computation, since each 2D image window, encoded as a matrix row, convolves with multiple filters in parallel.

To evaluate the real-case scenario, we targeted convolutional filters used to enhance edges and texture, adopted in object detection applications. As a test image, we select a *stop* signal. The filters' effects are shown in Fig.3. Please note that in accordance with commercial TPU devices, the target architecture operates on 8-bit integers.

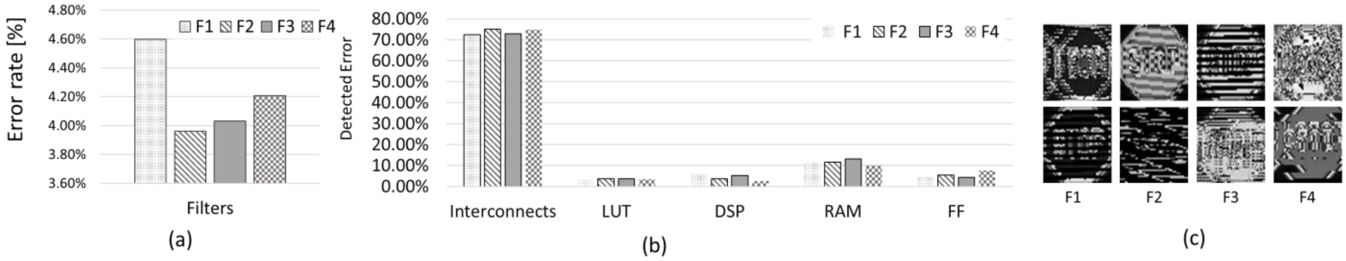


Fig.4. Collection of experimental results. In (a) the error rate detected over the parallel application of the different filters. In (b) the error distribution over the faulty resources. In (c) some of the faulty patterns provoked by interconnection faults.

We emulated the CRAM content alteration by adopting the bitflip fault model in the bitstream file. Starting from the golden bitstream, we utilized the PyXEL [10] framework to generate 3,000 corrupted bitstreams with a single bit-flip alteration, hence emulating a single SEU affecting a memory cell. For each tested bitstream, the filters' matrix, the stop signal, and the instructions defining the task are sent and written to the TPU memory buffers. Thus, the data is not subject to SEU and the detected errors only refer to SEU-induced hardware faults in the architecture. The outcome of each run is labeled as an error if it causes a visible distortion on the output image, which is translated into having at least 30% of pixels' computational mismatch with respect to the golden output.

Injection results reported that a single bitflip in the CRAM causes an average error rate of 4%. This result is in agreement with the fact that the cores in charge of performing the computations are mapped to DSPs, which we recall, are hardwired and therefore exempt from direct configuration via bitstream. This means that the arithmetic units (multipliers and adders) used in the design cannot be affected by a fault and thus cannot cause errors in the application. It follows that the sources of errors are to be found in the control logic implemented with LUTs and in the Interconnects.

Since our purpose is not to perform reliability testing but to validate the theory that considers interconnections as the most critical design resources, we adopted a random fault injection methodology. It follows that the low error rate is not indicative of design robustness, but simply that the affected resources are not used to implement the design.

Exploiting PyXEL's bitstream decoding capabilities, we can trace the type of resource affected by the fault injection. By combining this feature with the parsing of experimental results in which anomalous output occurred, we can realize a distribution of application errors on faulty resources.

Fig. 4a shows three important aspects. First, each filter processing has a similar error trend. This behavior is due to faults related to the input data, which as explained before, propagates through all the processing columns. On the other hand, the slight error variations may be due to the data values in the filters which combined with the approximation mechanism of the architecture, may mask fault effects.

Secondly, the distribution of errors related to each filter computation over the faulty resources confirms the expected behavior. LUTs are the lowest source of error. In fact, few functionalities are mapped to LUTs and only 7% of the available LUTs are used for design under evaluation. The error rate associated with DSPs and BRAMs again relates to the interconnects and/or ports of the physical modules. In fact, as application data are written to memory buffers after fault injection is performed, hence any SEU-induced bitflip is overwritten. The most relevant finding is definitely the one associated with interconnects, which are found to cause 70% of the errors for all parallel processing, confirming the predicted criticality. In addition, the common trend among the filters confirms the single fault propagation pattern among different computational resources. Finally, from Fig. 4c, we can appreciate how severe can be the output distortion induced by an SEU in CRAM provoking an interconnection fault.

V. CONCLUSION

In this paper, we analyzed the SEU-induced error propagation in systolic arrays implemented on Xilinx Zynq XC7Z020 SRAM-based FPGA. We combined information on circuit topology and functional behavior with FPGA architecture, physical resource and FPGA's fault models. This information is used to draw the error paths and predict the most sensitive modules in the design. The developed fault propagation model and targeted sensitive elements have been validated through fault injection campaign conducted by adopting bit-flip fault model in CRAM to emulate SEU. The experimental analysis confirmed the validity of the adopted fault model and the predicted criticality showing that SEU affecting interconnection resources result for the 70% as source of erroneous behavior.

REFERENCES

[1] 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.

[2] J. J. Zhang et al., "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," 2018 IEEE 36th VLSI Test Symposium (VTS), San Francisco, CA, USA, 2018, pp. 1-6.

[3] R. L. Rech Junior et al., "High Energy and Thermal Neutron Sensitivity of Google Tensor Processing Units," in IEEE Transactions on Nuclear Science, vol. 69, no. 3, pp. 567-575, March 2022.

[4] D. P. Ramaswami et al "Single Event Upset Characterization of the Intel Movidius Myriad X VPU and Google Edge TPU Accelerators Using Proton Irradiation," 2022 IEEE Radiation Effects Data Workshop (REDW) (in conjunction with 2022 NSREC), Provo, UT, USA, 2022, pp. 1-3,

[5] 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, March 2021

[6] R. L. R. Junior et al., "Reliability of Google's Tensor Processing Units for Convolutional Neural Networks," 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S), Baltimore, MD, USA, 2022, pp. 25-27.

[7] B.Du et al., "Ultrahigh Energy Heavy Ion Test Beam on Xilinx Kintex7 SRAM-Based FPGA," in IEEE Transactions on Nuclear Science, vol. 66, no. 7, pp. 1813-1819, 2019.

[8] Jonas Fuhrmann, " Implementierung einer Tensor Processing Unit mit dem Fokus auf Embedded Systems und das Internet of Things", 2018, http://hdl.handle.net/20.500.12738/8527

[9] A. V. Trusov et al., "p-im2col: Simple Yet Efficient Convolution Algorithm With Flexibly Controlled Memory Overhead," in IEEE Access, vol. 9, pp. 168162-168184, 2021.

[10] L. Bozzoli, et al., "PyXEL: An Integrated Environment for the Analysis of Fault Effects in SRAM-Based FPGA Routing," International Symposium on Rapid System Prototyping (RSP), pp. 70-75, 2018, Turin, Italy, 2018, pp. 70-75.