VLSI Implementation of Dead Pixel Removal using Three Cell Sorting Median Filter

K.J.Lokesh[1], Prof. A.Vivekraj, M.E., (Ph.D)[2]
[1]PG Scholar, [2]Assistant Professor
Department of ECE,
AVS Engineering College, Salem.

Abstract: Median filtering (MF) is one such non-linear manipulation technique which is quite often used in number of applications such as to hide impulse noises. An SRAM-based FPGA implementation of this filter is then susceptible to configuration memory bit flips induced by single event upsets (SEU). The median finding algorithm often requires a basic two cell sorter which finds the higher pixel intensity and lower pixel intensity of two pixels. The proposed algorithm extensively operates on three pixels at a time either in row, column or right diagonal. In this paper, a fault-tolerant implementation of the median filter is presented and studied indepth. Our protection technique checks if the median output is within a dynamic range created with the remaining non-median outputs. So a three pixel sorting is the basic operation for this algorithm Hence we introduce three cell sorter to facilitate sorting. The output of the three cell sorter is maximum, middle and minimum of three pixels which are used for sorting. Our protection technique checks if the median output is within a dynamic range created with the remaining non-median outputs. An output error signal is activated if a corrupted image pixel is detected, then, a partial or complete reconfiguration can be performed to remove the configuration memory error

Keywords: SEU, MF, SRAM, VLSI

I. INTRODUCTION

Noise is any undesired information that contaminates an image. Noise [10] appears in an image from a variety of sources. The Salt and Pepper type noise is typically caused by malfunctioning of the pixel elements in the camera sensors, faulty memory locations, or timing errors in the digitization process. For the images corrupted by Salt and Pepper noise [10], the noisy pixels can take only the maximum and the minimum values in the dynamic range. To recover the image from its noise there exits many filtering techniques which are application oriented. Some filtering techniques have better performance than the others according to noise category.

The working procedure of the existing mean filtering technique is very simple. For the existing mean filtering technique [3, 10] one pixel is taken at a time and a sub window is considered around that pixel. Then mean is calculated using the pixel values of that sub window. Then the considered pixel is replaced with that mean. In this way, all the mean filtering techniques work.

Therefore the median filter is a nonlinear image processing technique used to remove impulsive noise from images. This spatial filtering operation applies a two-dimensional (2D) window mask to an image region and replaces its original center pixel value with the median value of the pixels contained within the window. The window is then moved to the next image region and the cycle is repeated until the entire image is processed.

As a result, the ideal filtered image would have no impulsive noise and perfectly sharp edges. These are intrinsic properties of the median filter that cannot be achieved by traditional linear filtering techniques without resorting to time-consuming data manipulations. However, the median filter is not an ideal filtering operator, and edge preservation worsens as the total percentage of impulsive noise increases.

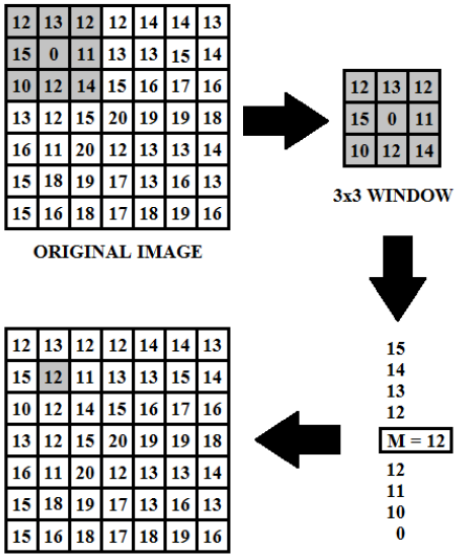

Fig.1 Median substitution process

Fig.1 illustrates the described median substitution process with a 3x3 pixels square window. It should be

noted that the analyzed "dead pixel" is removed from the original image. In order to process every image pixel, the 3x3 square window should be moved through the image. However, in FPGAs, this can also be achieved using a nine-pixel stream that sequentially passes through the median filter. Each group of nine pixels can be sorted using a structure composed of two-input exchange nodes as the one. The exchange node in Fig performs a two-input sorting using an 8-bit comparator and two 2:1 multiplexers. The two inputs are internally compared and the higher (H) and lower (L) values are obtained.

II. Soft-core Processor against SEUs by using TMR and Partial Reconfiguration

Yoshihiro Ichinomiya, ShiroTanoue, Motoki Amagasaki, Masahiro Iida presents a technique for ensuring reliable soft-core processor implementation on SRAM based FPGAs. Although an FPGA is susceptible to SEUs, configurability. It proposes techniques for SEU mitigation and recovery of a soft-core processor using triple modular redundancy (TMR) and partial reconfiguration (PR) with state synchronization. By carrying out an experiment, we confirm that a faulty soft-core processor can be recovered and synchronized with other soft-core processors.

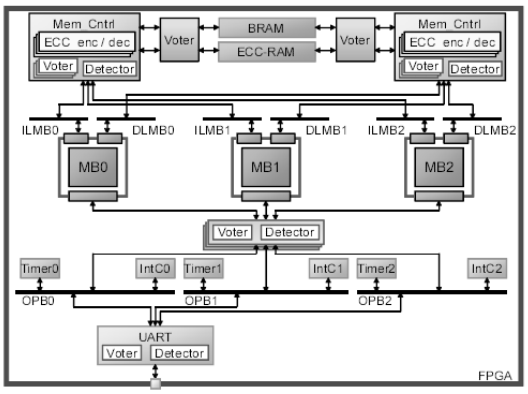

Fig.2 TMR and Partial Re-Configurations

This paper investigated a reliable soft-core processor design using the TMR scheme Figure shows the proposed TMR soft-core processor, which includes three redundant MBs, timers, and Int Cs. The voter selects the correct signal. In this system, the BRAM is not triplicated because the BRAM can be handled by using a technique that helps in improving the reliability of memory, such as the use of an Error Correcting Code (ECC). In the present study, the fault recoverability of the soft-core processor and the fault tolerance capability of the system are discussed. Therefore, the Partially Reconfigurable Region (PRR) is defined for MBs only. We achieve partial reconfiguration using the Early Access Partial Reconfiguration (EA PR) flow.

When an SEU occurs on one of the MBs, the effect of the SEU detected by the detector and mitigated by the voter. Then, the detector outputs an error signal to an external pin, and the faulty MB is reconfigured immediately. While the faulty MB is reconfigured, the other MBs continue to run. Therefore, the reconfiguration process is performed on the fly. After reconfiguration, the interrupt process of the RTOS triggers the synchronization process. This is why the registers of the reconfigured MB are different from those of the other MBs. The synchronization process is performed. After these recovery processes, the SEU effect is removed from the faulty MB, and the registers of all the MBs are synchronized. As a result, the operation of all MBs are synchronized, and the operation returns to the normal state.

III. PROPOSED ALGORITHM

In this paper, Fault-tolerant implementation of the median filter has been implemented using proposed march test. Our protection technique checks if the median output is within a dynamic range created with the remaining non-median outputs.

An output error signal is activated if a corrupted image pixel is detected, then, a partial or complete reconfiguration can be performed to remove the configuration memory error.

Fig.2 proposed block diagram

3.1 Proposed March test algorithm

The Proposed March test algorithm M is a test algorithm with a finite number of March elements M = M1; M2; . . . ; Mk , where each March element Mi consists of an

addressing order Ai and a finite number of Read/Write operations

$$M_i = A_i(O_1D_1, \ldots, O_mD_m).$$

1) $A_i \in \{\Uparrow, \Downarrow, \Updownarrow\}$—addressing order: $\Uparrow$—ascending, $\Downarrow$—descending, $\Updownarrow$—arbitrary.
2) $O_j \in \{R, W\}$—operations: R—Read, W—Write.
3) D_j—background pattern.

Single-cell FPs
are described by <S/F/R>and two-cell (coupling) FPs are described by <Sa; Sv/F/R>. In notations of FPs, S, Sa, and Sv are the sequences of operations required for fault sensitization(S is applied to the faulty cell, Sa—to the aggressor cell, and
Sv—to the victim cell), F∈*{*0, 1*}* is the observed memory behavior that deviates from the expected one. R ∈*{*0, 1, *-}*is the result of a Read operation applied to the faulty cell, in case if the last operation of S is a Read operation.“- ”is used when the last operation of S is not a Read operation. Note that if Sa is a sequence of operations then Sv should be a state; if Sa is a state then Sv can be a state or a sequence of operations. For example, if a cell has the fault <0W0/1/->,it means that if it contains value 0, then applying operationW0 on it will flip the cell value from 0 to 1. Or if two cellscontain the fault <1; 0W1R1/0/1>, it means the following:
if the aggressor cell has value 1, the victim cell has value 0then applying two sequential operations *{*W1, R1*}* will fail. Though the read operation will return the correct value 1, the victim cell value will remain 0A FFM is defined as a nonempty set of FPs. The difference between static and dynamic faults is determined by the number of operations required in S, Sa, or Sv. Static faults are the faults sensitized by performing at most one operation. Dynamic faults are the faults that are sensitized by performing more than one operation.

We consider all unlinked static faults and all dynamic two operation single-cell faults, as well as the subclass of two operation two-cell unlinked dynamic faults where both of the sensitizing operations are applied either to the aggressor cell or to the victim cell

LF definition: Let FP1 = <S1/F1/R1>and FP2 =<S2/F2/R2>be any static or dynamic fault primitives. If FP1and FP2 share the same victim cell, then it is said that there is a linked fault (denoted as FP1→FP2) if the following three conditions are satisfied.

1) C1: read operations of FP1 and FP2 do not detect a fault.
2) C2: FP2 masks FP1, i.e., F2 = ∼F1.
3) C3: FP2 is compatible with FP1, which means that ifS2 is applied immediately after S1, then the final state of the aggressor or the victim cell after performing S1
should be the same as the initial state required by S2.

In the LFs are divided into five groups as
1) LF1: combination of two single-cell unlinked faults. Both faults have the same faulty (victim) cell.
2) LF2av: combination of one two-cell FP1 and one singlecellFP2 unlinked faults, where FP1 is sensitized first. The victim cell of FP1 coincides with the faulty cell ofFP2.
3) LF2va: combination of one single-cell FP1 and one twocellFP2 unlinked faults, where FP1 is sensitized first. The faulty cell of FP1 coincides with the victim cell of
FP2.
4) LF2aa: combination of two two-cell unlinked faults FP1and FP2. FP1 and FP2 have the same aggressor, as well as the same victim cells.
5) LF3: combination of two two-cell unlinked faults FP1and FP2. FP1 and FP2 have the same victim cells, but different aggressor cells.

The structure-oriented method based on searching of symmetric March test algorithms, where the same March element is usually repeated with opposite addressing orders or with opposite background patterns. For example, in March C-: {M0: ⇑(W0); M1: ⇑(R0, W1); M2:⇑(R1, W0); M3: ⇓(R0, W1); M4: ⇓(R1, W0); M5: ⇑(R0)*}*,March elements M1 and M3 have the same sequence of operations (with the same background patterns) but opposite addressing orders, while March elements M1 and M2 have the same addressing order, the same sequence of operations but opposite background patterns.

The proposed schemes achieves a low-power, high-throughput, and modular hardware design of partial sorting network. By applying a pointer-like design, the comparing modules move the indexes of samples instead of moving the input data directly. Power dissipation is reduced by minimizing switching activities and signal transitions. To prevent unnecessary comparing of the large data set, an iterative architecture is proposed which uses both the low-power sorting module and a novel clipping mechanism.

3.2 Median Sorting Techniches

Configuration memory errors in SRAM-based FPGAs modify the design functionalities permanently until the original
Bit stream is reloaded. A redundant scheme can identify and mitigate user register errors. However, in order to remove these configuration errors, it is more practical to detect and then reconfigure partially or completely the faulty design. For this reason, the proposed fault-tolerant technique activates an output error signal if a corrupted image pixel is detected. Then, a partial or complete reconfiguration can be performed
to remove the permanent error.

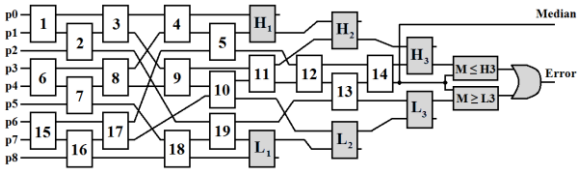

Fig.3 median sorting technique

Fig.3 illustrates the median sorting technique. Grey-shaded blocks are added to the original Fig. 6 scheme in order to create a range with the non-median outputs. The range is dynamically determined for each nine pixel group using identical exchange nodes to the one shown in Fig. The upper range (H3 low output) is calculated as the lower value from the four higher input values, and the lower range (L3 high output) is obtained using the higher value from the four lower input pixel values. Once the range is calculated, two 8-bit comparators check if the median output value is within the range, and set their output signals to 0 if the condition is satisfied or 1 otherwise.

The median filter is a pure combinational circuit, so SEUs in the SRAM-based FPGA configuration memory can modify the internal routing, changing the behaviour of the median sorting network. This may cause a bad median calculation or a non-median input value modification (i.e. the median value is correctly calculated but other inputs are corrupted during the sortening process). Both errors can be detected by the proposed technique if the median value is out of range, or the calculation of the range itself creates a never-satisfied range condition.

IV. EXPERIMENTAL RESULTS

The proposed circuit are simulated and synthesized by using modelsim and xilinx12.1 which occurs low area than the existing. The experimental results are given in Table 1 and the simulation results of layout and the waveforms are shown in the fig.4 and fig.5. Then the RTL schematic of the proposed are shown in fig.6.

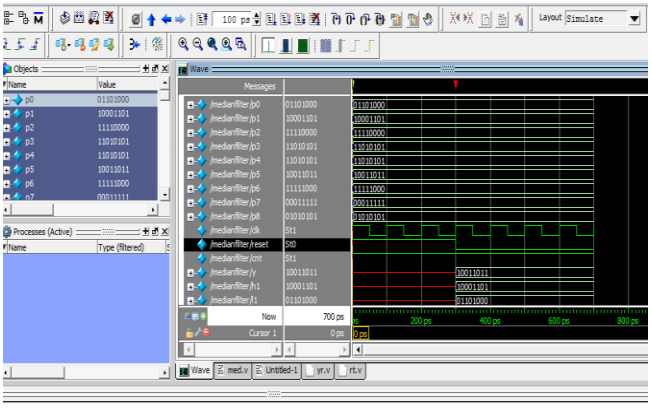

Fig.4 simulation result-I

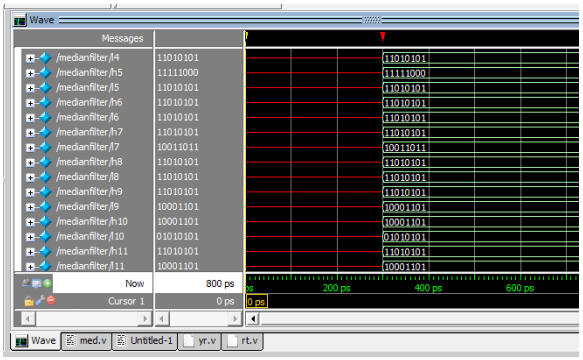

Fig.5 simulation results-II

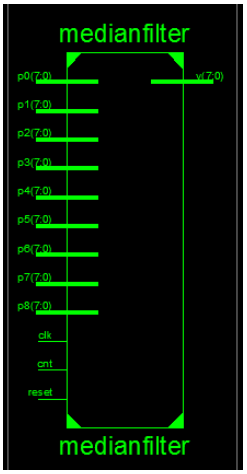

Fig.6 RTL schematic

s.no	Parameter	Existing	Proposed
1	Slice	213	189
2	LUT	392	348
3	IOB	104	80

Table1: comparison table

V. PERFORMANCE ANALYSIS

The Figure given below is shown that there is a considerable reduction based on no of transistors and the performance chart has been shown below in fig.7

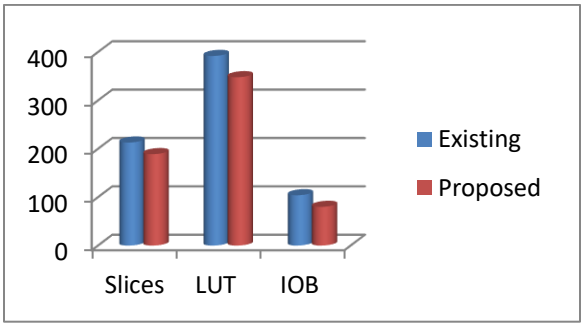

Fig.7 performance chart

VI. CONCLUSION

Digital image sensors are widely used in space applications, so space radiation can affect the sensor or the image processing system itself. The proposed technique checks if the median output value is within a dynamic range created each time with the remaining non-median outputs. An efficient test algorithm March is proposed for detection of the considered faults. A new structure-oriented method for generation of efficient symmetric March test algorithms detecting different combinations (links) of static and dynamic faults is described in dead pixels. A Pixel-level results show that images with homogeneous pixel-value regions are less susceptible to be corrupted when processed by a configuration memory damaged median filter. More homogeneous pixel-value regions imply more median filter inputs with equal values, so the probability that a median filter malfunction alters the median value is decreased.

REFERENCES

[1] H.-L. Eng, and K.-K. Ma, "Noise adaptive soft-switching median filter," IEEE Trans. Image Process., vol. 10, no. 2, pp. 242-251, Feb 2001.

[2] B. Jahne, "Applications and Tools," in Digital Image Processing, 6th ed. Berlin, Germany: Springer, 1991, ch. 1, sec. 3, pp. 14-16.

[3] R. Szeliski, "Image processing," in Computer Vision: Algorithms and Applications, Springer. Science & Business Media, 2010, ch. 3, sec. 2-3, pp. 111-127.

[4] L. Sterpone, M. S. Reorda, M. Violante, F. L. Kastensmidt, and L. Carro, "Evaluating different solutions to design fault tolerant systems with SRAM-based FPGAs," Journal of Electronic Testing, vol. 23, no. 1,pp. 47-54, 2007.

[5] G. R. Hopkinson, "Radiation effects in a CMOS active pixel sensor," IEEE Trans. Nucl. Sci., vol. 47, no. 6, pp. 2480-2484, 2000.

[6] M. Bagatin, "Introduction to the Effects of Radiation on Electronics Devices," in Ionizing Radiation Effects in Electronics. 1st ed. Boca Raton, Florida, 2016, ch.1, sec. 1.4, pp. 15-16.

[7] A. Bovik, Handbook of Image and Video Processing. New York: Academic, 2000.

[8] B. Shim, and N. R. Shanbhag, "Reduced precision redundancy for low-power digital filtering," in Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, pp. 148-152 vol. 1, 2001.

[9] P. Reviriego, J. A. Maestro, I. L´opez- alle, and J. A. de Agapito, "Soft Error Tolerant Infinite Impulse Response Filters Using Reduced Precision Replicas", in Proc. of the Radiation Effects on Components and Systems (RADECS) conference, Sevilla, Spain, pp. 493-496, 2011.

[10] L. A. Aranda, P. Reviriego, and J. A. Maestro, "A Fault-Tolerant Implementation of the Median Filter", presented at the Radiation Effects on Components and Systems (RADECS) Conference, Bremen, Germany, 2016.

[11] L. Yin, R. Yang, M. Gabbouj and Y. Neuvo, "Weighted median filters: a tutorial," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 43, no. 3, pp. 157-192, Mar 1996.

[12] M. A. Vega-Rodr´ıguez, J. M. S´anchez-P´erez, and J. A. G´omez-Pulido, "An FPGA-based implementation for median filter meeting the realtime requirements of automated visual inspection systems," Proc. 10th Mediterranean Conf. Control and Automation, Lisbon, Portugal, 2002.

[13] K. Benkrid, D. Crookes, and A. Benkrid, "Design and implementation of a novel algorithm for general purpose median filtering on FPGAs," IEEE

International Symposium on Circuits and Systems (ISCAS), pp. IV-425-IV-428 vol.4, 2002.

[14] G. L. Bates, and S. Nooshabadi, "FPGA implementation of a median filter," Proceedings of IEEE Speech and Image Technologies for Computing and Telecommunications Conference (TENCON), Brisbane, Qld., pp. 437-440 vol.2, 1997.

[15] S. A. Fahmy, P. Y. K. Cheung, and W. Luk, "Novel FPGA-based implementation of median and weighted median filters for image processing," International Conference on Field Programmable Logic and Applications, Tampere, Finland, pp. 142-147, 2005.

