DYNAMICALLY SHIFTED SCRUBBING FOR FAST FPGA REPAIR

Leonardo P. Santos, Gabriel L. Nazar and Luigi Carro

Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, RS - Brazil
pereira.santos@ufrgs.br, {glnazar, carro}@inf.ufrgs.br

ABSTRACT

Field Programmable Gate Arrays (FPGAs) are very successful platforms that rely on large configuration memories to store the circuit functions required by users. Faults affecting such memories are a major dependability threat for these devices, and the applicability of FPGAs on critical systems depends on efficient means to mitigate their effects. The usual means to effectively remove such faults, namely configuration scrubbing, consists in rewriting the desired contents of the configuration memory. The scrubbing process suffers from high power consumption and a long mean time to repair (MTTR). In this work we propose a novel approach to enable self-diagnosed circuits that, by being aware of their own disposition on the FPGA fabric are able to greatly reduce the MTTR.

1. INTRODUCTION

SRAM-based FPGA play an important role in self-aware systems by adding several attractive characteristics to logic designers: flexibility, high density and high pin count. However, these devices suffer reliability problems caused by Single Event Upsets (SEUs). SEUs in SRAM-based FPGAs are especially dangerous; because flipped bits in a configuration cell might change the device's programmed functionality, creating a persistent error.

Redundancy techniques as Dual Module Redundancy (DMR) and Triple Module Redundancy (TMR) can be used to hide the effects of SEUs, thus enabling the use of SRAM-based FPGAs in critical applications. The use of redundancy comes at a price; as the respective area overheads for DMR and TMR are 100 % and 200 % at least, redundancy also adds to power consumption. As redundancy works by detecting and/or masking the errors, it is possible to accumulate enough SEUs to overwhelm it and cause a failure.

Self-awareness is explored in this work through a system that detects and repairs faults on itself before they become functional failures. Currently, the standard way to achieve this in a SRAM-based FPGA is to use partial reconfiguration [1], [2], to re-write the configuration memory before the chosen redundancy is overwhelmed. This is called scrubbing and is usually accomplished by periodically re-writing the device's configuration memory from start to end. The periodicity is calculated based on a statistical estimate of the SEU rate per time unit on the device's operating environment. This means that a higher than anticipated SEU rate can leave a circuit with a configuration error. Also, scrubbing is not instantaneous, as the configuration bit streams sizes for modern devices are on the order of several megabits [3]; event the fastest configuration interface can pose unacceptable delays. The time required to fix an error with the scrubbing process is called Mean Time To Repair (MTTR).

Due to FPGAs' required flexibility, most configuration bits do not have an effect on the circuit, even for applications that use most of a device. This is due to most of them being routing configuration bits or bits that control unused resources. So SEUs on these idle configuration bits have no practical effects. The work in [4] exploits this fact to discover areas with high concentration of bits that affect the implemented circuit and then to choose an optimum frame start position for the scrubbing process, minimizing the MTTR. In this work we extend this concept to improve the gain in MTTR obtained with a fine-grained error detection technique, which provides enhanced diagnosis. Thus, the FPGA circuits are aware of their own placement on the reconfigurable fabric and of the relation between configuration bits and error detection signals. By not having a single start position, but instead a dynamic one based on fine-grained diagnosis, significant improvements are attainable.

The remainder of this paper is organized as follows: in section 2 we discuss related works. Section 3 presents the proposed technique. The validation and measurement setup is explained in section 4, while section 5 contains the results and their discussion. We close this paper with the conclusions in section 6.

2. RELATED WORK

The opportunities provided from coupling error detection techniques and partial reconfiguration have been explored in the past a mean to provide high availability in SRAM-

This work is sponsored by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)

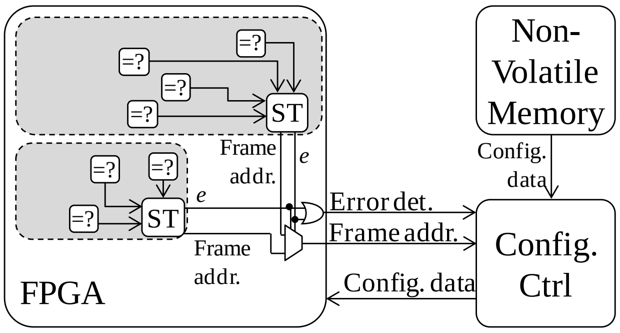

Fig. 1. Embedded detectors and translators in a circuit

based FPGAs. By using configuration readback and a per-frame CRC [5], it is possible to have a high precision on which frame should be corrected; but there's still need for a time-consuming readback and thus a high correction latency. Other works like [6] rely on automatically exploring the design space, using DMR and TMR to meet reliability constrains while minimizing area and repair time. This exploration tests different partitioning schemes and granularities, with different trade-offs between correction latency and area overhead.

Fine-grained DMR is also used in [7], with a focus in softcore processors. The authors propose using precompiled bit streams to bypass faulty components, while halting the processor to avoid corrupting its current state and memory. As is discussed in [6], the extra precision afforded by finer-grain techniques create a greater area and power overhead. One way to mitigate these overheads is offered in [8]. The use of hardwired resources, in this case the carry chains of each slice, hides some of the costs, as this chain is part of the device itself and underused in many situations.

3. DYNAMICALLY SHIFTED SCRUBBING

Because in [8] there is approximately one error detection bit for each of the device's slices, we can create the concept of an "error signature" that is formed by the concatenation of all error bits. These error signatures provide a more precise diagnosis information that thus can be used to guide a local repair procedure, provided the system is aware of its own signature-to-frame relations. The concept explored in this work is that a scrubbing procedure does not necessarily have to start at the first frame of the partition, as proposed in [4]. That work makes use of a previous error analysis to choose a single starting position for the scrubbing process, thus requiring only a very simple error detection scheme (primary output voters, watchdog). In this work, we make use of error signatures generated by a fine-grain error detection technique to dynamically guide the choice of the optimum starting frame, instead of relying on a statically chosen address. We aim for the ideal situation of having a fine-grained technique embedded on the final circuit and of using the

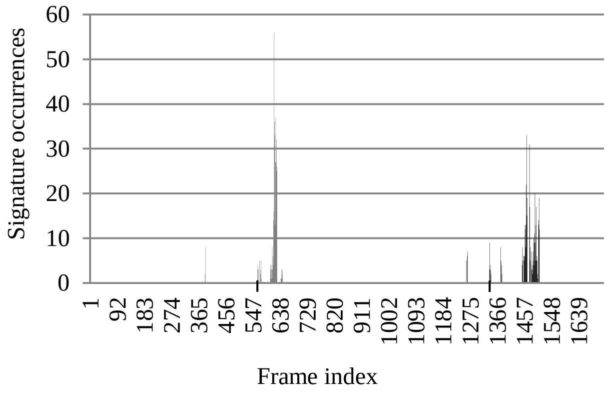

Fig. 2. Example of a histogram for *misex3*

error signature to jump to the best frame possible, MTTR-wise. Figure 1 shows the fine-grained error detectors (represented by the "*=?*" boxes) and Signature Translators (*ST*) embedded on a circuit.

To collect the error signatures, an error injection block is used, as described in the next section. The injector allows us to collect not only the signatures, but the frame address associated with them and when a different bit is tested within a frame. With this information, we can construct a histogram for each signature, with the frame number of the horizontal axis and the number of occurrences of that particular signature on the vertical axis. Figure 2 show the histograms for two different signatures for the *misex3* circuit.

It is possible to see in the histogram that one signature happens over 50 times for the same frame, frame 617. So it is fair to say that if that signature is detected, we could achieve a good precision if we simply corrected this frame. But it can also be seen that other frames generate the same signature as well and that they are near each other. So we can speculate that by starting the scrubbing by frame 617 we might achieve a low MTTR, but it might not be lowest possible. Because the scrubbing would not start at the first frame, we call this technique *shifted* scrubbing. To find the best starting position, we calculate the MTTR for each possible starting frame *f*:

$$MTTR_s(f) = \frac{FS}{BR} \sum_{i=PB}^{PE} \frac{h_s[i]}{O_s} \cdot \left(dist(i, f) + 1\right). \qquad (1)$$

Where $MTTR_s(f)$ is the MTTR for a given signature *s* and starting frame *f*, *FS* is the frame's configuration size in bits, *BR* is the scrubbing bit rate, *PB* is the partition beginning and *PE* is the partition end. $h_s[i]$ is histogram value for *s* for the *i*-th frame and O_s is the total amount of occurrences of *s*. Therefore, $h_s[i]/O_s$ is the probability that the error is located in the *i*-th frame, whenever *s* is received. *dist*(*i*, *f*) is the distance between *f* and the *i*-th frame, i.e., the amount of frames that have to be written before reaching the *i*-th. It is defined as:

$$dist(i,f)=\begin{cases} i-f, \text{ if } i \geq f \\ PE-f+1+i-PB, \text{otherwise.}\end{cases} \quad (2)$$

The sum in (1) is, therefore, the "mean frames to repair" when signature *s* is received and *f* is used as starting frame. It is converted to a time unit with the time required to write a frame (*FS*/*BR*). The scrubbing controller would start on the best frame and reconfigure the whole device. If during the scrubbing it reaches the end of the partition, it would continue the scrubbing on the partition's beginning, until it reaches the last frame before the starting frame. This can be seen in equation (2), the first condition is the distance between *f* and *i* if *f*, the starting frame, is before *i*. In this case, the error is corrected before reaching the end of the partition. The second condition occurs when the error is only corrected after reaching the end of the partition and returning to its beginning. In this case, $PE - f + 1$ is the amount of frames written until the partition end and $i - PB$ is the distance between the partition beginning and *i*. One improvement would be stopping the scrubbing process after the error detection signals turn off, saving power and readying the controller for a new scrubbing round faster.

It is possible to leverage on the FPGAs high density if the error detectors and the blocks that translate the error signature to the optimum frame address, indicated as *ST* in Figure 1, are embedded on the device itself. This arrangement gives designers a high density device with self-error identification.

4. EXPERIMENTAL SETUP

In order to extract the error signatures, and thus identify which bits are critical in a design, it was used an error injection platform run on a Xilinx XUPV5-LX110T board, containing a Xilinx Virtex 5 XC5VLX110T FPGA device. This error injection platform relies on an error detection scheme, in our case, the one presented in [8]. It uses LUT-level DMR and the device's embedded carry chain to create an error detection bit for each of the device's slices. Bundling these all the error detection bits together, we form the error signature for that bit.

With the error detection in place, the injector platform exercises the Circuit Under Test (CUT) buy reading the configuration memory of a single frame through the Internal Configuration Access Port (ICAP); it then flips one bit in the read configuration and writes back this "errored" configuration on the device. The platform then excites the CUT by creating several pseudo-random input vectors by means of LSFR. While exciting the circuit, if one or more bits on the error signature turn on, the platform sends to a host PC the frame address being tested, the error signature itself and a flag bit if that signature is the first one for the bit being tested using a serial interface.

Table 1. Benchmark circuits

Circuit	LUTs	FFs	PIs	POs	S_{Size}
alu4	402	0	14	8	192
apex2	798	0	39	3	395
apex4	655	0	9	18	332
bigkey	575	224	264	197	354
clma	1269	34	384	82	609
des	550	0	256	245	355
diffeq	470	244	29	3	234
dsip	635	224	230	197	370
elliptic	143	71	20	2	73
ex1010	487	0	10	10	215
ex5p	128	0	8	63	81
frisc	1718	853	21	116	894
misex3	699	0	14	14	349
pdc	1253	0	16	40	603
s298	17	14	5	6	11
s38417	1709	1447	30	106	884
s38584.1	2001	1233	40	304	1080
seq	846	0	41	35	430
spla	221	0	16	46	114
tseng	598	260	53	122	337
Avg.	758.7	230.2	74.95	80.85	395.60

After all the bits in a frame's configuration frame are tested, that frame's original configuration is written back and the test of a new frame is begins. Because the signatures are sensible both to the flipped bit and to the input vector, it was chosen to limit the number of different signatures for the same bit to 20.

To determine the signatures' behavior for different types of circuits, we selected a set of 20 benchmark circuits from the MCNC suite; obtained at [9]. As the CUT and the injection platform are placed on the same device, it was necessary to limit the action of the injection platform on just the CUT and not on itself by the use of placement constrains to create an Area Under Test (AUT) in which the CUT is placed completely and exclusively.

To analyze the data collected, we wrote a C++ application to map the different signatures and then calculate for each signature the optimal beginning frame for scrubbing process. The application also calculates the MTTR for the standard scrubbing approach. An example of the optimum starting position for two signatures is shown in Figure 2 as the two black marks on the horizontal axis. As the errors are sensitive to the routing and placement choices of the synthesis tools, it is essential that this information is kept in the final design. It is possible to achieve this by the use of incremental design flow, among other means such as placement and routing constraints.

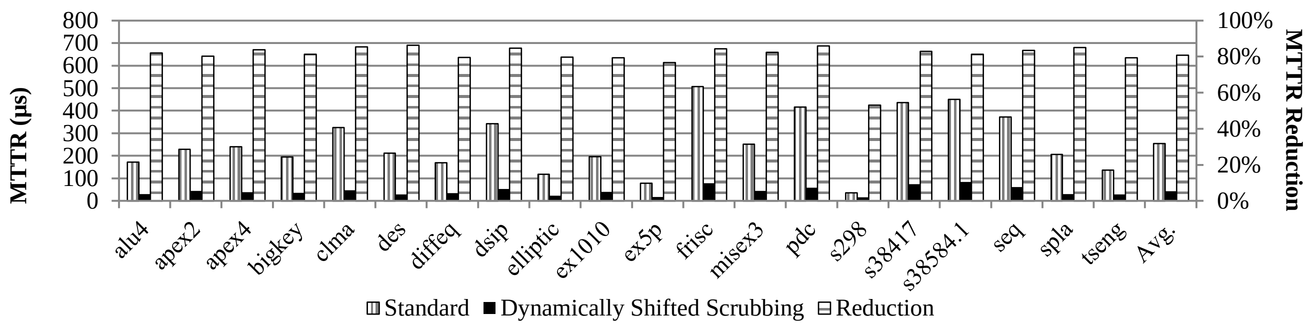

Fig. 3. MTTR for the standard and shifted scrubbing approaches and relative reduction

5. EXPERIMENTAL RESULTS

The list of the tested circuits from the MCNC suite is shown in Table 1, along with the resources used (pre-DMR), number of Primary Inputs (PI), number of POs and the signature size (post DMR) in bits. The circuits were tested according with the procedure described in section 4 and the error signatures were recorded and processed in a host PC.

All results assume a scrubbing interface operating at the maximum speed of the Virtex 5 SelectMAP interface, which is a 32-bit wide port at 100 MHz. It is also taken into account the time required to issue a write command to the interface (25 cycles in our implementation) and to write a dummy frame, which is required by SelectMAP. Such costs represent only 0.39 % and 1.9% of the total MTTR for standard and shifted scrubbing respectively. The MTTR was measured, in µs, for a standard scrubbing approach and for the shifted scrubbing. The obtained results are presented in Figure 3, together with the measured reduction in the MTTR. It can be seen that the gains in MTTR reduction are significant, with a minimum reduction of 77 % for the *ex5p* circuit and a maximum reduction of 86 % for the *des* and *pdc* circuits. The mean reduction for the 20 benchmark circuits was 80.85 %.

6. CONCLUSION

In this paper we have examined the possibility of reducing time needed to repair the configuration of a SRAM-based FPGA with a novel approach, using a shifted scrubbing process. By using a fine-grain error detection scheme allied with partial reconfiguration, it is possible analyze the circuit and discover information that allows us to precisely identify the frame with a configuration error and restore its correct state. The technique was evaluated through exhaustive testing with an error injection platform. The obtained results show that is possible to expect MTTR reductions of over 85 % for many of the benchmarked circuits. These results are very encouraging to further pursue optimizations of this technique.

Such a future work could see the detection and the signature translator circuits allied with a TMR scrubbing controller offering logic designers the advantages of SRAM-based FPGAs with self-repair capabilities.

7. REFERENCES

[1] Altera, "Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs ", ed.

[2] Xilinx. *Partial Reconfiguration User Guide* Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/ug702.pdf

[3] Xilinx. *7 Series FPGAs Configuration User Guide* Available: http://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf

[4] G. Nazar and L. Carro, "Accelerated fpga repair through shifted scrubbing," in *Field Programmable Logic and Applications, 2013. FPL 2013. International Conference on*, 2013.

[5] M. Gokhale, P. Graham, E. Johnson, N. Rollins, and M. Wirthlin, "Dynamic reconfiguration for management of radiation-induced faults in FPGAs," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004, p. 145.

[6] C. Bolchini, A. Miele, and C. Sandionigi, "A Novel Design Methodology for Implementing Reliability-Aware Systems on SRAM-Based FPGAs," *Computers, IEEE Transactions on*, vol. 60, pp. 1744-1758, 2011.

[7] M. Psarakis and A. Apostolakis, "Fault tolerant FPGA processor based on runtime reconfigurable modules," in *Test Symposium (ETS), 2012 17th IEEE European*, 2012, pp. 1-6.

[8] G. L. Nazar and L. Carro, "Exploiting Modified Placement and Hardwired Resources to Provide High Reliability in FPGAs," in *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, 2012, pp. 149-152.

[9] K. Minkovich. *Kirill Minkovich's Home Page*. Available: http://cadlab.cs.ucla.edu/~kirill/