2011 14th Euromicro Conference on Digital System Design

Fault Tolerance of Multiple Logic Faults in SRAM-based FPGA Systems

Farid Lahrach Abderrahim Doumar Eric Châtelet
ICD/LM2S, STMR, UMR CNRS 6279, Université de Technologie de Troyes, 12 rue Marie Curie BP 2060
10010 Troyes Cedex, France
E-mail: {farid.lahrach, abderrahim.doumar, eric.chatelet}@utt.fr

***Abstract*—The very hight levels of integration and submicron device sizes used in current and emerging VLSI technologies for SRAM-based FPGAs lead to higher occurrences of defects and operational faults. Thus, there is a critical need for fault tolerance and reconfiguration techniques for SRAM-based FPGAs to increase chip reliability with field reconfiguration. We first propose a technique utilizing the principle of master-slave to tolerate logic or cells in SRAM-based FPGAs. We show that this architectural technique can be used to build redundancy for defect and fault tolerance with limited area and performance overhead. Our algorithm improves reliability of the SRAM-based FPGAs by performing two operations: TMR (*triple modular redundancy*) (in which CLBs are used to triplicate a logic function whose value is obtained at the voter output) and *partitioning* (in which the design is partitioned into a set of MSUs (master-slave unit) to reduce the amount of configuration memory required). In response to a component failure, a functionality equivalent MSU that does not rely on the faulty component replaces the affected MSU. Our technique can handle a large numbers of faults (we show tolerance of 16 logic faults in look-up tables LUTs belonging to the same MSU). Experimental results conducted on a subset of the ITC'99 benchmarks demonstrate a high level of reliability in term of fault tolerance with low hardware overhead compared to TMR which has a $5\times-6\times$ area overhead and high power consumption.**

***Index Terms*—SRAM-based FPGA, configurable logic block, look-up table, fault tolerance, triple modular redundancy.**

I. Introduction

Faults are becoming increasingly pronounced in emerging applications and technologies, from permanent faults arising from circuit processing at nanometer scales to transient soft errors arising from high-energy particle hits. This has spurred much research on techniques to develop fault tolerance without increasing substantial area, power consumption, or performance penalties [1].

Indeed, the expanded use of field programmable gate arrays (FPGA) in remote, long life, and system-critical applications requires the development and implementation of effective, efficient SRAM-based FPGA fault-tolerance techniques. SRAM-based FPGA have inherent redundancy and in-the-field reconfiguration capabilities, thus providing alternatives to standard integrated circuit redundancy-based fault-recovery techniques. Runtime reliability can be enhanced by using such unique features. Recovery from permanent logic and interconnect faults without runtime computer-aided design (CAD) support can be efficiently performed with the use of fine-grained physical design partitioning, or partial dynamic reconfiguration managed by software or hardware processor embedded in recent FPGAs. Faults are localized to small partitioned blocks that have fixed interfaces to the surrounding portions of the design, and the affected blocks are reconfigured with previously generated, functionally equivalent block instances that do not use the faulty resources. This technique minimizes the post-fault-detection system downtime, while requiring little area overhead. Only the finely located faulty portions of the FPGA are removed from use.

In this paper, we propose a technique for increasing SRAM-based FPGA system reliability with acceptable overhead. The proposed fault-tolerance technique deals with multiple faults using selective TMR combined with master-slave technique (MST). The target device for demonstration is the Xilinx Virtex-5 XC5VFX70T, which considered in the first part composed of an array of configurable logic blocks (CLBs). Nonetheless, our proposed technique is applicable to a wide range of SRAM-based FPGA architectures. The Xilinx ISE 11.4 place-and-route tool maps a circuit netlist onto the array of CLBs and routing resources. We propose an SRAM-based FPGA architecture based on partitioning the physical design onto a set of master-slave units (MSUs), each MSU is composed of one CLB master (CLB-M) surrounded by a set of CLBs slaves (CLB-S), an interface specification which denotes the connectivity to neighboring MSUs, and a portion of a netlist. Fault tolerance is achieved by multiple reconfigurations of each MSU. Furthermore, by using locked MSU interfaces, the effects of swapping a MSU configuration do not propagate to other MSUs, thus reducing the storage overhead. In contrast to related fault-tolerance techniques proposed for SRAM-based FPGAs, such as TMR which has a $5\times$ to $6\times$ area/power overhead [2], our approach is lightweight and incurs minor area and performance overhead compared to TMR technique. Overall, this paper provides a new and viable solution for the spectrum of techniques for reliability and overhead trade off.

The rest of this paper is organized as follows. Section II and Section III discuss background information, approach restrictions and work related to fault-tolerance. Section IV describes the details of the approach and implementation, and Section V introduces techniques used to implement the proposed approach and experimental results. In Section VI presents reli-

978-0-7695-4494-6/11 $26.00 © 2011 IEEE
DOI 10.1109/DSD.2011.33

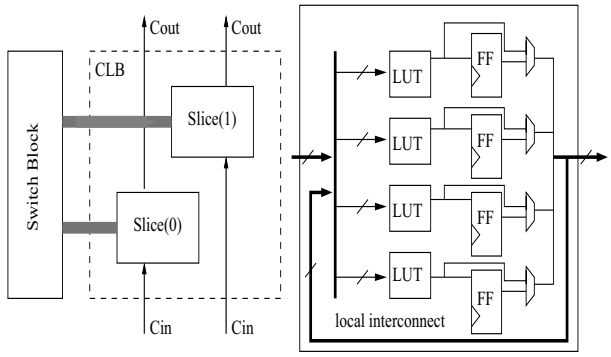
ability calculation, we adopt independent uniformly distributed faults model for reliability calculation. Section VII discusses future work on this topic, and Section VIII closes the paper with some remarks summarizing the benefits of this fault-tolerance approach. To the best of our knowledge, this paper is the first systematic study on fault tolerance dealing with high rate of logic faults.

II. Background material for the proposed approach

In this section, we survey the background material for the proposed approach. We present the target SRAM-based FPGA architecture, the fault models assumed, and techniques envisioned for supporting the testing and fault diagnosis steps of the approach.

A. SRAM-based FPGA architecture

The new fault-tolerance approach is demonstrated using the Xilinx Virtex-5 [3] family as the target architecture, specifically the XC5VFX70T. The Virtex-5 logic block, which is referred to us as configurable logic block (CLB), comprising two slices and a switch block (SB). The switch block allows for connections from a slice back to the same slice, between the two slices, as well as into rows and columns of general interconnect. As shown in Figure.1, each slice contains four 6-input LUTs and 4 flip-flops. The LUTs in Virtex-5 are implemented as "true" 6-LUTs, rather than being constructed using smaller LUTs that can be optionally combined together via multiplexers. The true 6-LUTs provide performance for implementing large 6-input logic functions. In Xilinx's Virtex-5, the 6-LUT has two usable output pins, i.e. it can can implement two independent LUTs if the total number of unique pins in each does not exceed five, this property is exploited in [4] to achieve fault tolerance.



(a) Configurable logic block (b) Architecture of a SLICE

Fig. 1. Architecture of a CLB and SLICE

The configurable logic blocks (CLBs) are the main logic resources for implementing sequential as well as combinatorial circuits. Each CLB element is connected to a switch block for access to the general routing network. A CLB element contains a pair of slices. These two slices do not have direct connections to each other, and each slice is organized as a column. Each slice in a column has an independent carry chain. For each CLB, slices in the bottom of the CLB are labeled as SLICE(0), and slices in the top of the CLB are labeled as SLICE(1).

The Xilinx tools designate slices with the following definitions. An "X" followed by a number identifies the position of each slice in a pair as well as the column position of the slice. The "X" number counts slices starting from the bottom in sequence 0, 1 (the first CLB column); 2, 3 (the second CLB column); etc. A "Y" followed by a number identifies a row of slices. The number remains the same within a CLB, but counts up in sequence from one CLB row to the next CLB row, starting from the bottom. Figure.2 shows four adjacent CLBs of the die.

The two main configurable structures in an SRAM-based FPGA are the look-up table (LUT) and the routing multiplexers, both of which are configured using SRAM cells. Configuration bits control the select lines of the routing multiplexers. Therefore, a change in the configuration value could change the routing of a function being implemented. If a change occurs in the configuration bits of the LUT table, the entire function being implemented could be affected. For instance, a LUT implementing an AND-gate could suddenly be implementing an OR-gate.

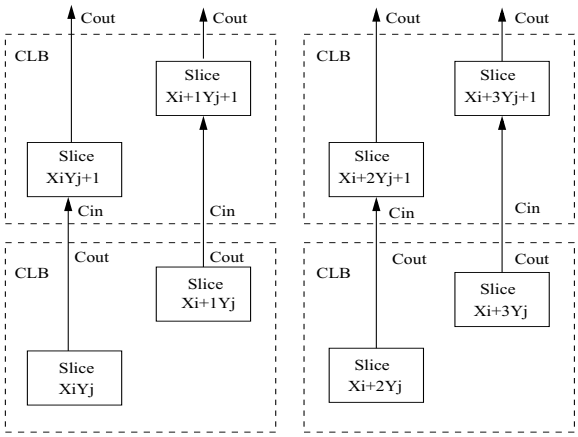


Fig. 2. Rows and column relationship between CLBs and Slices

B. Fault model, Testing, and Diagnosis

The proposed approach requires fault detection and diagnosis method as a preprocessing step. We assume a widely used stuck-at-1 or stuck-at-0 fault model (the SRAM cell value of an LUT in the CLB is always 1 respectively 0), faults in the SRAM-based FPGA interconnect can be categorized into two major groups, namely opens and shorts. An open fault can be a switch stuck-open or an open on a net (line segment). A switch stuck-open fault causes the switch to be permanently open regardless of the value of the SRAM cell controlling the switch. A short fault can be a switch stuck-closed fault or a bridging fault between two routing resources. A switch stuck-closed fault causes the switch to be permanently closed

regardless of the value of the memory cell controlling the switch. Two different bridging fault types are considered in the literature [5]. In the first type, any two arbitrary nets can be shorted together. In the second type, by considering physical information, only adjacent nets can have a bridging fault. In its current form, our approach does not address interconnect faults. Note that for local interconnects, faults will be expressed as fault of the CLB to which it connects.

Several studies have been extensively developed the testing of SRAM-based FPGAs through exhaustive testing of the device architecture. The majority of these approaches can be classified as off-line. A number of schemes have been developed for detecting faults in CLBs, assuming that the interconnect resources were fault free [6]-[8], others studied testing only interconnect resources with the assumption that CLBs were fault free [9][10]. To date, however, there has been relatively little research conducted on both interconnect resources and CLBs testing. For example, with built-in self-test (BIST), the SRAM-based FPGA is configured with a small testing circuit that is restricted to a specific physical region of the chip, which is then used to test an other region of the device. The test circuit is moved across the device in a systematic manner until the entire device is thoroughly tested.

III. Related Work

Related work can be traced along the following three lines of research: SRAM-based FPGA synthesis, fault-tolerant design, and SRAM-based FPGA yield enhancement.

Lach et al. [11]-[13] proposed a method of offline logic tolerance that involves partitioning the FPGA into tiles. Each tile contains a part of a design and some spare logic and interconnects resources. In some ways this method is similar to the block structured architecture mentioned [14]. Both methodologies group CLBs into larger structures in an attempt to minimize the effects of fault tolerance on the system function. However, tiling does not use redundant blocks like the block structured architecture. Instead, each tile contains spare resources, with different configurations for a given tile using some different resources within the tile. Initially, multiple configurations of each tile are stored in memory. Each of these configurations places spare logic in a different location while maintaining the same tile boundary connections. When a logic fault occurs within a tile, the tile is replaced by a configuration that does not use the faulty resources. Tile configurations can be made any size and distribution that satisfies timing and area constraints. Faults on inter-tile interconnect may be tolerated if additional routing is reserved. Although this method was developed for offline use, Lach et al. [11]-[13] speculated that an online fault detection method developed by Shnidman et al. [15] might be compatible with their method. Because all configurations are precomputed and are significantly smaller than configurations for an entire FPGA, this method offers very quick reconfiguration time. Experimental results showed between 14% and 45% variations in circuit speed for nine test circuits partitioned into 6×2 tiles with one spare CLB each in a Xilinx 4000. However, pre-computing all the tile configurations allows timing characteristics to be known before a system function is loaded, so Tiling can therefore be said to make timing guarantees. However, this method become inefficient when multiple faults occurs in the tile because of the number of spare CLBs allocated. The proposed approach requires fault detection and diagnosis as a preprocessing step. The Tiling approach is not guaranteed to tolerate all interconnect faults.

Doumar and Ito [16][17] proposed a custom SRAM-based FPGA architecture and a method that could be used with it for yield enhancement as well as user FT. In their architecture, each CLB and the surrounding routing resources of it are defined as a unit element. The configuration memory of each unit element is arranged as a shift register. A multiplexer is added to each unit element so that the configuration memory can be shifted vertically, horizontally, or serially. Wiring channels on the right and bottom of the FPGA are not associated with any unit element, and cannot be used in a fault tolerant circuit configuration. The circuit is initially placed and routed around a set of spare unit elements and downloaded to the FPGA. Testing determines which unit elements are faulty and feeds this information to the shifter circuitry on the FPGA. The shifter then shifts the entire circuit configuration so that the faulty unit cells are in the reserved spare unit elements. Two different spare allocations were proposed: the *king* allocation and the *horse* allocation. The king shifting allocation sets a spare in the middle of each 3×3 square set of unit elements, so that each spare can replace one of its eight neighboring unit elements. The horse allocation sets a spare in the middle of a 3×3 cross, so that each horse spare can replace one of its four neighboring unit elements. The benefits of this method are (1) no external reconfiguration algorithm is required, and (2) the timing of the circuit is almost fixed. This method is not guaranteed to tolerate all interconnect faults, and will not tolerate any faulty SRAM cell.

The routing resources play an important role in SRAM-based FPGA, over than 80% of transistors inside recent SRAM-based FPGAs are dedicated to the programmable routing network as programmable switches and buffers [5]. In [18][5] Huang et al. studied fault tolerance of switch blocks and switch block array in FPGA using a probabilistic analysis, and they modeled the switch block as a bipartite graph. However, regarding the timing limitations and the routing limitation of signals in real FPGAs the approach is far from being consistent.

Our approach is completely transparent to the existing compute-aided design (CAD) tool chain and exists as an intermediate step that is used in conjunction with existing synthesis and place-and-route tools. We are able to dynamically tolerate faults in the field. Finally, we are able to make timing guarantees (which is critical for real-time systems), require less system downtime, and less performance penalties.

IV. The proposed fault-tolerant Approach

The corner stone of our approach to fault-tolerance is partially reconfiguring the SRAM-based FPGA to alternate

configuration in response to faults. If the new configuration implements the same function as the original, while avoiding the faulty hardware block, the system can be restarted. The challenging step is to identify an alternate configuration efficiently. Note that the principle of the approach is published in [19][20] and in this section, we elaborate on the key elements of our approach.

A. Master-Slave Unit Fault-Tolerant Blocks

We reduce the amount of configuration memory required by reducing the size of the component that is configured. This is enabled by logically partitioning a design in a way that components can be independently reconfigured without impacting the rest of the design. In comparison to other alternatives, this approach can also reduces the down time for recent SRAM-based FPGAs that support partial reconfiguration and, more importantly, significantly increases the level of fault-tolerance with only nominal hardware and time overhead. The key concepts for implementing the new approach are master slave units (MSUs) fault-tolerant blocks, CLBs masters (CLB-M), and CLBs slaves (CLB-S).

Definition 1: A CLB-M is the main concept of our approach, it consists of an architecture that encompasses three CLBs as shown in Fig.3, and routing resources (witch blocks) and a voter. The CLB-M implements three copies of the same function and performing a bit-wise "majority vote" on the output of the triplicate function.

Definition 2: A CLB-S is an ordinary CLB, that in the case of any faulty CLB-S form, the MSU is detected faulty and identified, the faulty CLB-S is tolerated using one CLB of those of CLB-M. The fault tolerance is performed reconfiguring the concerned MSU.

Definition 3: A MSU is composed of four elements: one CLB-M and interconnect resources, four CLB-Ss and interconnect resources, a netlist which must be placed on those CLBs (CLB-M and CLB-Ss) and routed across the interconnect, and a specification of how to interface the MSU to adjacent MSUs, Figure.3 shows the proposed architecture of the MSU implementing five functions (Function 1,2,3,4,5).

In our approach each MSU is associated with both physical resources and portions of the complete netlist, the design can only be partitioned into MSUs after the complete netlist has gone through place-and-route once. By fixing the interfaces between the MSUs, we create the opportunity to produce multiple partial configurations that satisfy the functional specification for given MSU, independently from the remainder of the design. Fault-tolerance is achieved by using two CLBs of the CLB-M as spare resources in the MSU so that, once a fault in particular CLB-S is detected, a configuration of the MSU's functionality that does not utilize the faculty CLB-S can be activated.

Each MSU has a fixed number of of CLB-Ss (4,3 or 2 CLB-Ss) and one CLB-M and it is independent from all MSUs

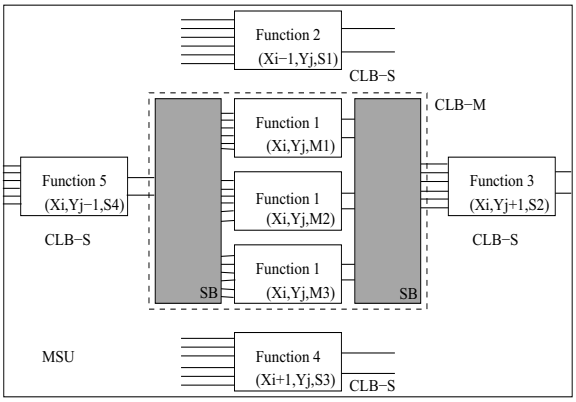


Fig. 3. Proposed Architecture of the MSU, where each CLB-M is surrounded by four CLB-Ss

by virtue of the fixed MSU interface. Thus, selecting one MSU for participating on a design can assemble a complete configuration, under the condition that none of the MSUs rely on a faulty component.

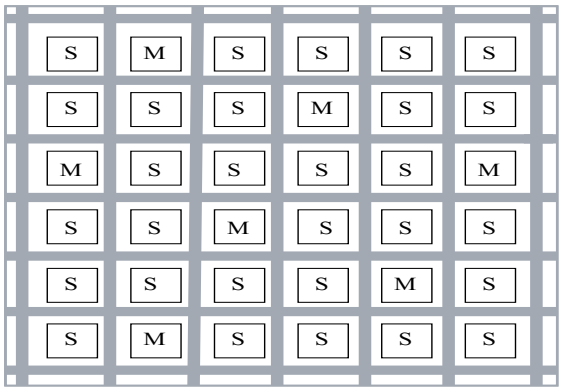


Fig. 4. A 6×6 CLB design partitioned into 7 MSUs, where each CLB-M labeled (M) is surrounded by four or three CLB-S labeled (S)

The master-slave TMR inspired approach (MST) provides many advantages in the implementation of fault-tolerant SRAM-based FPGA systems. First, the approach can tolerates until two CLBs in the same MSU, which means that the MST approach can tolerate until sixteen LUTs in the same MSU (Fig.1). The amount of memory needed to store the set of MSUs is smaller than the amount required to store a set of complete configurations. For example, consider a design that must be able to tolerate any single CLB fault and which maps into 6×6 CLB array. It may be possible to divide the design into 7 MSUs (equation 1), where each CLB-M is surrounded by four or three CLB-S (Fig.4). Assuming that one configuration of the complete 6×6 design requires N bytes of memory, the non partitioned approach would require 36*N of memory for fault-tolerance: one configuration for each CLB that is at risk. With our technique, the 6×6 design is partitioned into seven MSUs, since each MSU ($N/7$ storage

bytes) is independent, the entire storage is only 7*N, a 85% reduction from no-partitioned approach.

The MST approach also increase system reliability. For this example, the non partitioned approach could tolerate only one faulty CLB in the entire design. The MST approach, however, is capable of tolerating any single or double fault in the MSU but up to fourteen faulty CLBs in the entire design.

The cost of increased fault-tolerance and reduced configuration memory is the introduction of two CLBs per MSU to perform fault-tolerance, these two CLBs are used as spare resources. For this example, the no-partitioned approach reserves 2.7% of the CLBs to protect a single fault, while the MST approach requires 38.8%. However, the MST technique increases the reliability by $\times 14$ times and open up the opportunity to deal with high rate of faults in several CLBs belonging two the same MSU.

B. Preallocating CLB-Ms Resources

Equation 1 is used to determine if a CLB with $N \times M$ CLB array coordinates (r,c) should be reserved as a CLB-M, where $(r, c) \in \{1, ..., N\} \times \{1, ..., M\}$. If we let $\delta = r(mod\ 5)$.

$$c(mod\ 5) = (2\delta + 1)(mod\ 5) \quad (1)$$

If the equation 1 is verified, then the CLB array location (r,c) is a CLB-M else the CLB of coordinates (r,c) become a CLB-S. Fig.4 shows a 6×6 CLB design partitioned into 7 MSUs, where each CLB-M labeled (M) is surrounded by four or three CLB-S labeled (S). Table.I shows the results obtained by applying the MST technique on several devices of Xilinx Virtex-5 family [21]. The results show that more than 93.50% of MSUs encompass four CLB-Ss, and less than 6.50% of MSUs encompass three CLB-Ss, all MSUs with three CLB-Ss are located on the edges of the CLB array. This strategy guarantees that for working area CLB utilization of less than 98.33% of CLB array. (It should be noted that most SRAM-based FPGA designs use only 80% [22] of the available logic in order to enhance routability [5]. As we aim to reconfigure just affected MSUs, routing complexity does not significantly increase).

To compensate this weakness an auxiliary strategy is envisaged to ensure that fringe CLBs are sufficiently close to a matching spare CLB in order to allocate adjacent spares. This strategy increases the amount of overhead, however it increases the CLB area coverage to 100%.

C. Multiple Logic Faults

In the case of one faulty CLB-S e.g. Fig.5 shows that the CLB-S of coordinates (Xi,Yj-1,1) is faulty, in this case if the new reconfiguration that implements the Function 2 while avoiding the faulty CLB-S (Xi,Yj-1,1) by implementing the Function 2 in the CLB of coordinates (Xi,Yj,M1), the system can be restarted.

When two faults occur in different CLBs within the same MSU, the concerned MSU is marked as a faulty MSU so the two faulty CLBs need to be reconfigured to a two CLBs

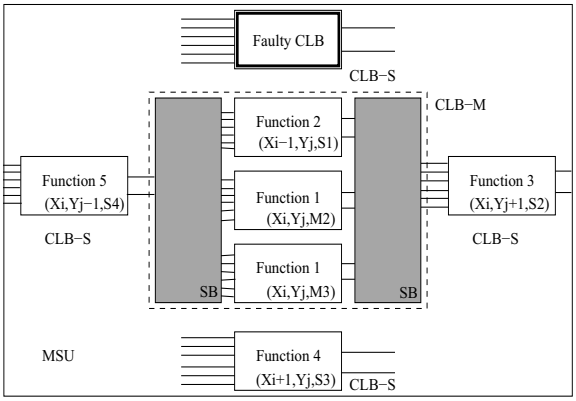


Fig. 5. Example of overcoming single fault, where the CLB-M1 of coordinates (Xi,Yj,M1) implements the Function 2 of the faulty CLB-S1 of coordinate (Xi-1,Yj,S1) rather than the Function F1

of the CLB-M. When this occurs, we use the master-slave technique to match the functions implemented on faulty CLBs to the two CLBs of the CLB-M. The fault-tolerance in this case is achieved by partially reconfiguring the SRAM-based FPGA to an alternate configuration in response to faults. Fig.6 shows an example of two faulty CLB-Ss of coordinates (Xi-1,Yj,S1) and (Xi+1,Yj,S3) consecutively. In this scenario the two CLBs: CLB-M1 and CLB-M3 of coordinates (Xi,Yj,M1) and (Xi,Yj,M3) consecutively implement the functions: Function 2 and Function 4 which were implemented in CLB-S1 and CLB-S2 of coordinates (Xi-1,Yj,S1) and (Xi+1,Yj,S3) consecutively.

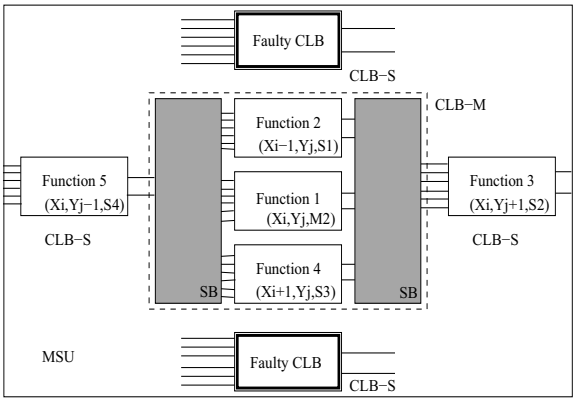


Fig. 6. Example of overcoming double faults, where CLB-M1 and CLB-M3 of coordinates (Xi,Yj,M1) and (Xi,Yj,M3) consecutively implement the functions: Function 2 and Function 4 which were implemented in CLB-S1 and CLB-S2 of coordinates (Xi-1,Yj,S1) and (Xi+1,Yj,S3) consecutively

V. EXPERIMENTAL RESULTS

We conducted an evaluation of the proposed approach in two phases. In the first, we applied the approach to fifteen ITC'99 designs. In the second phase we studied expected reliability improvement trends as a function of the number of used CLBs in a design. We used Xilinx ISE 11.4 for

TABLE I
APPLICATION OF MST TECHNIQUE TO A SET OF VIRTEX-5 SRAM-BASED FPGA FAMILY

Virtex-5 device	CLB Array Row × Column	# of MSUs with 4 CLB-S	# of MSUs with 3 CLB-S	# of MSUs with 2 CLB-S	# of CLBs not covered	Area coverage (%)
XC5VFX70T	160 × 38	1,137	79	0	79	98.70
XC5VLX85	120 × 54	1,227	69	0	69	98.93
XC5VLX155T	160 × 76	2,338	94	0	94	99.23
XC5VFX200T	240 × 68	3,141	123	0	123	99.25
XC5VLX220	160 × 108	3,349	107	0	107	99.38
XC5VSX240T	240 × 78	3,617	127	0	127	99.32
XC5VLX330T	240 × 108	5,045	139	0	139	99.46
XC5VFX30T	80 × 38	561	47	0	47	98.45
XC5VLX50T	120 × 30	661	58	1	60	98.33
XC5VSX95T	160 × 46	1,390	82	0	82	98.89

synthesis and mapping of the VHDL codes to a Virtex-5 XC5VFX70T target device. We synthesized the circuits first using the original source, and then using our approach. The International Test Conference ITC'99 benchmarks were developed in the CAD Group at Politecnico di Torino as a set of circuits with characteristics typical of synthesized circuits. We used the VHDL descriptions and made the appropriate transformations in order to derive the equivalent MST circuit.

However, in real SRAM-based FPGA such Xilinx Virtex-5 each CLB is composed of eight LUTs (Fig.1), we have adapted our technique to tolerate faulty LUTs. We conducted an evaluation of the proposed approach in two phases: In the first, we implemented the VHDL code of the design (Original design) and recover from the number of used LUTs. In the second phase we partition the design onto MSUs, and as the available number of Slice LUTs in the target FPGA is 44,800 all MSU are of type MSU_4 (see Section VI) then we use the following library declaration "`library UNISIM`" and the package "`use UNISIM.VComponents.all`" to instantiate Xilinx primitives. To instantiate the Xilinx primitives we use the data provided by "`View Technology Schematic`" in "`Synthesize - XST`" process to extract the value of each LUT in the original design.

Table II shows the area metric before and after the application of the new approach for reliability enhancement. The first column in the table indicates the name of the design. The next two columns in Table II show the number of LUTs used for implementing designs before and after the application of the approach. The rightmost column indicates the overhead as a result of enhanced reliability. For all fifteen designs, the area overhead is around 60.0%. This rate is relatively high compared to tile repair where the area cost is 25% [23], because we have used a LUT for implementing a voter function in each MSU.

TABLE II
VARIATION OF RESOURCES USED BEFORE AND AFTER THE APPLICATION OF THE MST APPROACH

ITC'99 Design	Original # of LUTs	Final # of LUTs	Final-Original / Original
b01	10	16	0.60
b02	4	7	0.75
b03	39	63	0.61
b04	115	184	0.60
b05	184	295	0.60
b06	8	14	0.75
b07	99	159	0.60
b08	23	38	0.65
b09	49	79	0.61
b10	38	62	0.63
b11	100	160	0.60
b12	261	420	0.60
b13	80	128	0.60
b14	1218	1950	0.60
b14_1	1291	2065	0.59

VI. RELIABILITY CALCULATION

The first step in calculating reliability is the selection of fault models. As mentioned in the introduction of this paper, there are two major sources of logic faults in SRAM-based FPGA systems: transient soft errors arising from high-energy particle hits and permanent faults arising from circuit processing at nanometer scales.

The second class of faults is related to manufacturing imperfections. These defects are not large enough to impact initial testing, but after a longer period of operation they become exposed. Design errors can also cause a device to stop functioning in response to rate sequences of inputs (e.g., due to a power density surge in a small part of design). For this type of model, we follow the gamma-distribution Stapper fault model [24]. The model is applicable on any integrated circuit with regular repetitive structure, including memories and FPGA devices. In the remainder of this section, we elaborate on technical details related to the independent uniformly distributed faults.

Given an SRAM-based FPGA architecture composed of $N \times M$ CLB array, using the formula indicated in Section IV, we obtain three types of MSUs as shown in Table.I. Suppose that an SRAM-based FPGA architecture is partitioned into α MSUs with four CLB-Ss labeled as MSU_4, β MSUs with three CLB-Ss labeled as MSU_3, and γ MSUs with two CLB-Ss labeled as MSU_2. We assume that the reliability of of a CLB is $R_{CLB}(t)$,i.e, the probability that a CLB being faulty is $(1 - R_{CLB}(t))$. It is easy to see that the reliability $R_{init}(t)$ that the original design is faulty-free is expressed as

$$R_{init}(t) = (R_{CLB}(t))^n \quad (2)$$

The reliability provided by the MST technique can be expressed analytically as follows: define $R_{MST}(t)$ as the overall reliability of a design and define $R_{MSU}(t)$ as the reliability of one MSU. At the high level, the overall reliability is a series expression, i.e.

$$R_{MST}(t) = \left(\prod_{i=1}^{\alpha} R_{MSU_4}(t)\right) \times \left(\prod_{j=1}^{\beta} R_{MSU_3}(t)\right) \times \left(\prod_{k=1}^{\gamma} R_{MSU_2}(t)\right) \quad (3)$$

where:
$R_{MSU_4}(t)$, $R_{MSU_3}(t)$, $R_{MSU_2}(t)$ express the reliability of MSU_4, MSU_3, and MSU_2 respectively.
As each MSU unit has the same reliability, the overall reliability of the system is

$$R_{MST}(t) = (R_{MSU_4}(t))^{\alpha} \times (R_{MSU_3}(t))^{\beta} \times (R_{MSU_2}(t))^{\gamma} \quad (4)$$

On the other hand the fact of the presence of routing resources in the MSU structure gives to that flexibility in the fault tolerance process of such kind. With a reconfiguration of these two switch blocks the structure allows in the case of fault the tolerance of any CLB whether a CLB-S or a CLB constituting the CLB-M by two possibilities. Thus the reliability of the MSU unit is:

$$R_{MSU}(t) = \sum_{i=0}^{n} \binom{l}{i} R_{CLB}(t)^{l-i}(1 - R_{CLB}(t))^{i} \quad (5)$$

where n is the number of the CLBs which can replace the defective CLBs, in our approach $n = 2$. l is the total number of CLBs per MSU, $l = 5, 6, 7 for\ MSU_2$, MSU_3, and MSU_4 respectively.

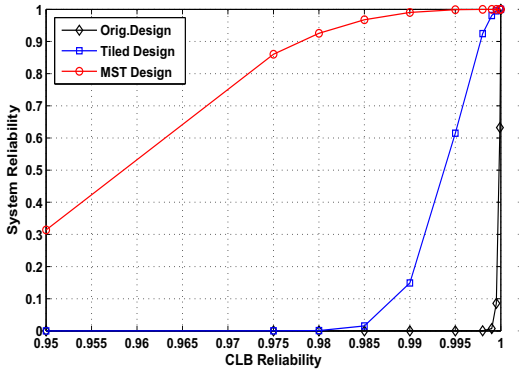


Fig. 7. Reliability of traditional methods and tiled method versus MST method for hypothecal 5000 CLB SRAM-based FPGA

Table III indicates the potential of the proposed approach for reliability enhancement. For example, in the case of a 5000 CLB design, with $R_{CLB}(t) = 0.999$ the probability of the initial design being functional is less than 1% and the probability of the tiled design being functional is 98% , while the probability of the MST design being functional is 99.9%. Fig.7 graphs the reliability results for the 5000 CLB design. Notice that the data of Tiled column in Table III are obtained from [11].

VII. Future Work

Recent SRAM-based FPGAs tend to be dominated by interconnect resources; more than 80% of transistors are dedicated to programmable routing network (Section III). The majority of configuration bits are used to configure the state of interconnects rather than CLBs, and it is likely that these interconnect resources are more susceptible to faults. The fault-tolerance methodology presented above addresses in interconnect inside each MSU faults directly dedicated to specific CLBs because they appera as CLB faults. However, the vast of interconnect resources pass through higher level hierarchical switch structures that are not covered by unique CLB faults. We currently cannot make any specific claims on interconnect fault-tolerance.

The main problems combined with fault tolerance systems include fault detection during system operation, fast fault location, quick recovery and bridging the system back to the state in which it operates correctly. The authors believe that any fault tolerance technique in the field of reconfiguration systems without fault detection and diagnosing, remains unpractical technique. A comprehensive strategy that gathers between fault detection, diagnosis and fault tolerance will form the basis of future work on the SRAM-based FPGA fault-tolerance.

VIII. Conclusion

The rapid progress in SRAM-based FPGA integration and the growing market for these devices make the emergence of fault-tolerant techniques as an important design metric for FPGA-based systems. In this paper, we have developed a fault-tolerance approach to work at the level of physical design. Our hierarchical fault-tolerance technique partitions designs into master-slave units. The key element of our approach to fault-tolerance is partially reconfiguring the SRAM-based FPGA in response to a fault, if the new configuration of the faulty MSU implements the same function as the original, the system can be restarted and can deals with multiple faults. Experimental results conducted on a subset of ITC'99 benchmarks indicate that the technique is effective with acceptable overhead compared to TMR.

Acknowledgment

This work is performed in the framework of the MBSIE (Microcapteurs Biologiques Spectroscopiques Intelligents et Embarqués) project Managed by: Mr. Abderrahim Doumar and Professor Eric Châtelet, (Contrat de Plan Etat Région) CPER with the support of the "Conseil Général de l'Aube" France.

TABLE III
RELIABILITY OF TRADITIONAL DESIGN METHODS AND TILED APPROACH VERSUS MST APPROACH AGAINST CLB RELIABILITY FOR LARGE SRAM-BASED FPGAS

CLB	100 CLB design			1000 CLB design			5000 CLB design		
Reliability	**Orig.**	**Tiled**	**MST**	**Orig.**	**Tiled**	**MST**	**Orig.**	**Tiled**	**MST**
0.9500	0.005921	0.444669	0.977090	0.000000	0.000302	0.793137	0.000000	0.000000	0.313863
0.9750	0.079551	0.800119	0.996995	0.000000	0.107534	0.970356	0.000000	0.000014	0.860311
0.9800	0.132687	0.864375	0.998448	0.000000	0.232820	0.984595	0.000000	0.000684	0.925313
0.9850	0.220739	0.919633	0.999340	0.000000	0.432660	0.993422	0.000000	0.015161	0.967542
0.9900	0.366277	0.962643	0.999803	0.000043	0.683364	0.998031	0.000000	0.149026	0.990197
0.9950	0.606224	0.990317	0.999975	0.006704	0.907280	0.999751	0.000000	0.614762	0.998760
0.9980	0.819220	0.998429	0.999998	0.136145	0.984404	0.999984	0.000047	0.924414	0.999920
0.9990	0.905528	0.999608	0.999999	0.370696	0.996091	0.999998	0.007000	0.980610	0.999990
0.9995	0.951999	0.999903	0.999999	0.611453	0.999028	0.999999	0.085470	0.995153	0.999998
0.9999	0.990868	0.999995	0.999999	0.912346	0.999952	0.999999	0.632119	0.999762	0.999999

REFERENCES

[1] A. Djupdal and P. Haddow, "Yield enhancing defect tolerance techniques for FPGAs," in *MAPLD International Conference*. Citeseer, 2006.

[2] R. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.

[3] Xilinx, "Virtex-5 FPGA Configuration User Guide," *Xilinx User Guide (v3.8) UG191*, vol. 1, 2009.

[4] J. Lee, Y. Hu, R. Majumdar, L. He, and M. Li, "Fault-tolerant resynthesis with dual-output luts," in *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*. IEEE Press, 2010, pp. 325–330.

[5] J. Huang, M. Tahoori, and F. Lombardi, "Fault tolerance of switch blocks and switch block arrays in FPGA," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 13, no. 7, pp. 794–807, 2005.

[6] C. Stroud, E. Lee, and M. Abramovici, "BIST-based diagnostics of FPGA logic blocks," in *To appear in Proc. International Test Conf*, 1997.

[7] M. Abramovici, C. Stroud, C. Hamilton, S. Wijesuriya, and V. Verma, "Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications," 1999.

[8] M. Abramovici and C. Stroud, "BIST-based test and diagnosis of FPGA logic blocks," *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, vol. 9, no. 1, pp. 159–172, 2001.

[9] I. Harris and R. Tessier, "Testing and diagnosis of interconnect faults in cluster-based FPGA architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 11, p. 1337, 2002.

[10] J. Smith, T. Xia, and C. Stroud, "An automated BIST architecture for testing and diagnosing FPGA interconnect faults," *Journal of Electronic Testing*, vol. 22, no. 3, pp. 239–253, 2006.

[11] J. Lach, W. Mangione-Smith, and M. Potkonjak, "Low overhead fault-tolerant FPGA systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 2, pp. 212–221, 1998.

[12] J. Lach and W. Mangione-Smith, "Enhanced FPGA reliability through efficient run-time fault reconfiguration," *IEEE Transactions on Reliability*, vol. 49, no. 3, pp. 296–304, 2000.

[13] J. Lach, W. Mangione-Smith, and M. Potkonjak, "Algorithms for efficient runtime fault recovery on diverse FPGA architectures," *DFT'99*, pp. 386–394, 1999.

[14] N. Howard, A. Tyrrell, and N. Allinson, "The yield enhancement of field-programmable gate arrays," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 1, pp. 115–123, 1994.

[15] N. Shnidman, W. Mangione-Smith, and M. Potkonjak, "On-line fault detection for bus-based field programmable gate arrays," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 6, no. 4, pp. 656–666, 2002.

[16] A. Doumar, S. Kaneko, and H. Ito, "Defect and fault tolerance FPGAs by shifting the configuration data," *DFT'99 VLSI Systems*, pp. 377–385, 1999.

[17] A. Doumar and H. Ito, "Detecting, diagnosing, and tolerating faults in SRAM-based field programmable gate arrays: a survey," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 3, pp. 386–405, 2003.

[18] J. Huang, M. Tahoori, and F. Lombardi, "Fault tolerance of programmable switch blocks," *InternalReport, ECEdept. Northeastern University*, 2003.

[19] F. Lahrach, A. Abdaoui, A. Doumar, and E. Chatelet, "A novel SRAM-based FPGA architecture for defect and fault tolerance of configurable logic blocks," in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*. IEEE, 2010, pp. 305–308.

[20] F. Lahrach, A. Doumar, E. Châtelet, and A. Abdaoui, "Master-Slave TMR Inspired Technique for Fault Tolerance of SRAM-Based FPGA," in *Proceedings of the 2010 IEEE Annual Symposium on VLSI*. IEEE Computer Society, 2010, pp. 58–62.

[21] Xilinx, "Virtex-5 FPGA Configuration User Guide," *Xilinx User Guide (v5.3) UG190*, vol. 1, 2009.

[22] J. Emmert, C. Stroud, and M. Abramovici, "Online fault tolerance for FPGA logic blocks," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 2, pp. 216–226, 2007.

[23] S. Pontarelli, M. Ottavi, V. Vankamamidi, A. Salsano, and F. Lombardi, "Reliability Evaluation of Repairable/Reconfigurable FPGAs," 2006.

[24] C. Stapper, "A new statistical approach for fault-tolerant VLSI systems," in *Fault-Tolerant Computing, 1992. FTCS-22. Digest of Papers., Twenty-Second International Symposium on*. IEEE, 2002, pp. 356–365.