Improve defect tolerance in a cluster of a SRAM-based Mesh of Cluster FPGA using hardware redundancy

Adrien Blanchardon, Roselyne Chotin-Avot, Habib Mehrez, Emna Amouri

Improve defect tolerance in a cluster of a SRAM-based Mesh of Cluster FPGA using hardware redundancy

Adrien Blanchardon[1,2], Roselyne Chotin-Avot[1,2], Habib Mehrez[1,2] , Emna Amouri[1,2,3]
1- Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
2- CNRS, UMR 7606, LIP6, F-75005, Paris, France
3- Institut TELECOM, TELECOM ParisTech, Paris, France
Email: adrien.blanchardon@lip6.fr

***Abstract*—The technological evolution involves a higher number of physical defects in circuits after manufacturing. One of the future challenge is to find a way to use a maximum of defected manufactured circuits. In this paper, multiple techniques are proposed to avoid defects in the cluster local interconnect of a SRAM-based Mesh of Clusters FPGA. Using defect tolerance, area and timing metrics, two previous hardware redundancy strategies are evaluated on the Mesh of Clusters architecture : Fine Grain Redundancy (FGR) and Improved Fine Grain Redundancy (IFGR). We show that using these techniques on a cluster of a Mesh of Clusters architecture permits to tolerate 8 times more defects than on an industrial Mesh FPGA with a low area overhead (-6% for FGR and 22% for IFGR) and a low increase of Critical Path Delay (CPD)(6% for FGR and 2% for IFGR). We also proposed three new redundancy strategies using spare resources : Distributed Feedbacks (DF) for crossbar down, Adapted Fine Grain Redundancy (AFGR) to avoid defective multiplexers and Upward Redundant Multiplexer (URM) for the crossbar up. Compared to the Mesh of Clusters architecture without defect tolerance techniques, the best trade off between defect tolerance (36.4%), area overhead (11.56%) and CPD (+7.46%) is obtained using AFGR. Using the other methods permits to considerably limit the area overhead (10.4% with URM) with a lesser number of defective elements bypassed (18% max).**

I. Introduction

According to the Moore's law, the transistor density integration doubles every 18 months. However, this technological evolution involves a higher number of physical defects after manufacturing circuit. Thereby, circuits based on nanotechnology may have a defect rates of 20% [1] [2]. As manufacturing yield decreases, one of the future challenges is to find solutions to use defect manufactured circuits [3] [4]. Compared to ASICs, FPGAs become the perfect target architecture due to their ability to integrate more complex applications, their flexibility and good performance. Propose techniques to bypass these defects in FPGAs is necessary. These techniques can be classified in 2 categories :

- Software-based techniques : these techniques use configuration tools to avoid defective resources [5] [6] [7] [8]. However, their efficiency rely on their ability to bypass the defects and on the number of unused resources.
- Hardware-based techniques : extra hardware resources are added in the architecture. Correcting a defect consists in bypassing the defective element by the added one [9] [10] [11]. These techniques can easily be implemented in FPGA but increase the area.

A Fine Grain Redundancy technique (FGR) was introduced in [12] and suggests to add two levels of multiplexers in a Mesh FPGA interconnect. Output multiplexers (OMUX) are used for defect avoidance and input (IMUX) multiplexer for signal restoration. IMUX and OMUX allow signals to route around defective resources (Fig.1). This approach can tolerate defects in the switch boxes and in the wiring.

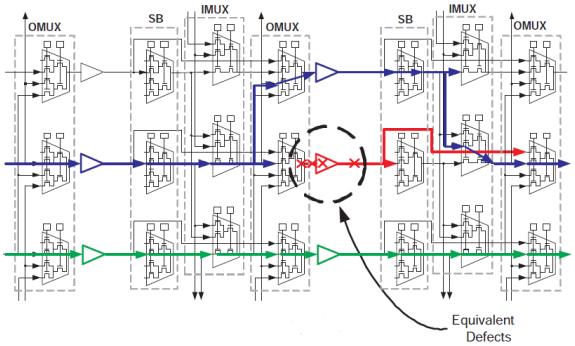

Fig. 1: FGR in mesh FPGA interconnect [12]

In this paper, we present different hardware-based techniques to prevent defects in FPGAs and we discuss their performance. The remainder of the paper is organized as follows. Section II presents the cluster architecture of a Mesh of Clusters FPGA. Section III describes all hardware-based techniques adapted and proposed on the cluster and their implications on the architecture. Simulation results for routability, defect tolerance, area and timing are presented in section IV. Finally, section V concludes this paper.

II. Cluster architecture

The interconnection architecture is one of the major research topics in FPGA design since it occupies up to 90% of the total area and is responsible of large part of the circuit delay [13] [14]. To reduce these costs, the authors proposed in

 DOI: 10.1109/FPL.2014.6927389

[13] a Mesh of Clusters architecture to reduce the total area by 42% compared to the VPR Style Clustered Mesh. The cluster architecture considered in this paper is presented by Fig.2.

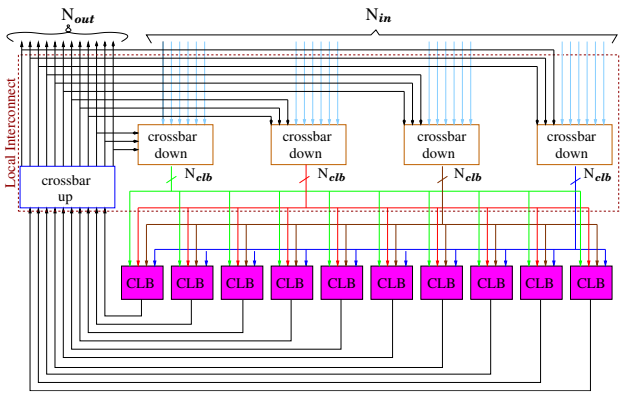

Fig. 2: Initial cluster

III. Hardening the Cluster

The next paragraphs review different hardware-based techniques and their implications on the cluster's architecture to increase defect tolerance. Each technique is based on hardware redundancy by adding multiplexers in the local interconnect. To bypass a defect, its location inside the cluster is needed to configure the application. This is done with a map of all defected multiplexers after test and diagnostic [11].

A. Fine Grain Redundancy in a Mesh of Clusters architecture

In the initial cluster architecture, if a signal is routed by a defective multiplexer in a crossbar, the connection between this signal and its corresponding target is not possible. The presented techniques permit to solve this problem.

1) Classic Fine Grain Redundancy (FGR): this technique introduced in [12] was proposed to avoid defects in connection box of an industrial mesh architecture. Applying this technique to the cluster's local interconnects in a Mesh of Clusters architecture consists in adding four levels of mux2 in each crossbar: two levels of IMUX are added to avoid the defect by shifting the signal to the nearest multiplexer, and two levels of OMUX are used to restore the signal. Fig.3 presents a crossbar architecture hardened with the FGR technique.

2) Improved Fine Grain Redundancy (IFGR): FGR technique is useless if the nearest multiplexer is already use by another inputs. The IFGR was proposed in [15] to avoid this kind of conflicts. This solution doubles the number of outputs per Mux (Fig.3) that allows to use the second output in case of defect on the other one.

B. Proposed techniques

1) Adapted Fine Grain Redundancy (AFGR): Adding 2 levels of IMUX and OMUX on each crossbar has a significant impact on the area of the cluster. To reduce this impact, we propose a new technique called AFGR. With FGR technique, hardware redundancy is applied inside each crossbar(Fig.3). A defective multiplexer can be replaced by another inside the same crossbar down using the IMUX level. With AFGR, the

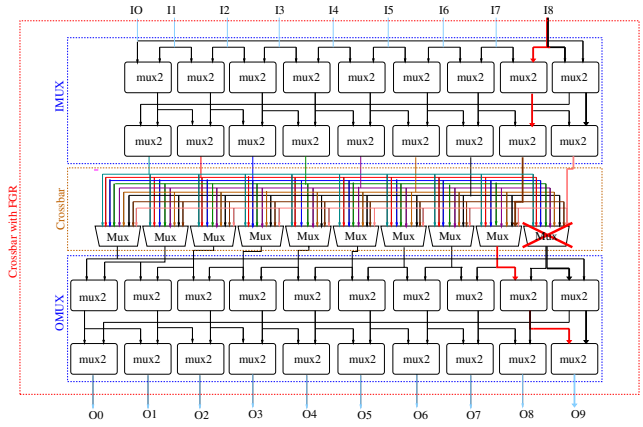

Fig. 3: crossbar with FGR

IMUX level is common to all crossbars down (illustrated in Fig.4), thus a defective multiplexer can be bypassed inside the same crossbar or inside its neighbor. Therefore extra connections between each crossbar are available, that also increases the possibilities to route the same signal. To reduce the area overhead, the OMUX level is removed. In this case, as with FGR technique, a defective output cannot be restored if the nearest multiplexer is already use by another input. But with Mesh of Clusters architecture, as all crossbar down are connected to all CLBs, then a crossbar's output can be lost without performance degradation.

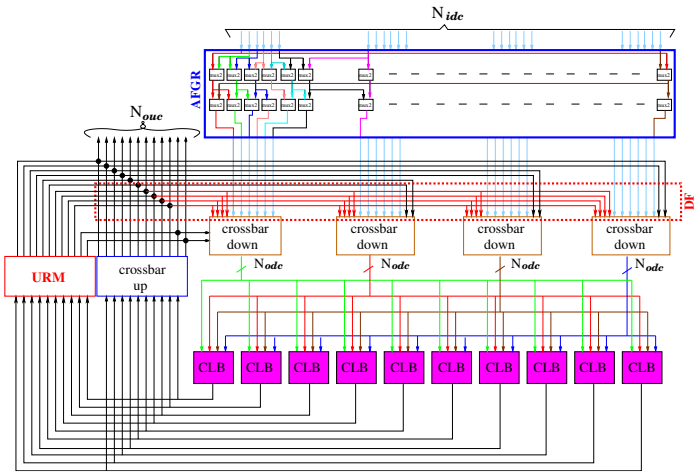

Fig. 4: Cluster with AFGR, DF and URM

2) Distributed Feedbacks (DF) [11]: In the initial cluster architecture, all crossbars up outputs are connected as feedback to crossbars down. Each feedback is connected to only one crossbar down. If a feedback signal is routed by a defective multiplexer in a crossbar down, the connection is not possible and the cluster is useless. To avoid defective multiplexers, we propose a new technique (DF). Each feedback is now connected to all crossbar down (as shown in Fig.4 with bold red lines). The defective connection can be bypassed by another one in another crossbar down.

3) Upward Redundant Multiplexer (URM): In the initial cluster architecture, crossbar up connects CLBs outputs to

Hardware redundancy technique	number of defective multiplexer bypassed in the cluster		Number of mux2 increase in FPGA (% of mux2 added Max)			Critical Path Delay		
	Min	Max	UMSB	DMSB	UMSB+DMSB	Min	Avg	Max
FGR	41 (7%)	214 (36.4%)	+4%	+14%	+18%	+14.9%	+29.7%	+44.6%
IFGR	/	428 (72%)	+12%	+48%	+60%	+14.9%	+29.7%	+44.6%
AFGR	41 (7%)	214 (36.4%)	+2%	+4.5%	+6.5%	-11%	+7.46%	+40%
DF	24 (4%)	312 (50%)	/	+35%	+35%	-13%	-10%	+12%
URM	9 (1.5%)	108 (18%)	+10.4%	/	+10.4%	+0.28%	+1%	+3.6%

TABLE I: FGR, IFGR, Adapted FGR, DF and URM performance

cluster outputs and to crossbars down with the feedbacks. If the crossbar up is defective, all cluster outputs are lost. So we propose a new technique (URM) to increase the defect tolerance of the crossbar up. A multiplexer is added in parallel to the crossbar up (illustrated in Fig.4). Each black dots on cluster's outputs in Fig.4 represents an additional mux2 and allows to choose between the crossbar up or the URM output. This technique require as many extra URM as Mux we want to bypass in the crossbar up. Its main advantage is that we can add as many URM as number of tolerated defect desired. That permits a good trade-off between tolerated defect and area overhead.

IV. Experimental evaluations

In this section, our objective consists in evaluating the impact of the aforementioned techniques on the defect tolerance, area and timing of the cluster.

A. Methodology

For each 20 MCNC benchmark [16], we determine the smallest architecture in terms of logic blocks number, to place and route all applications on Mesh of Clusters architecture. Then, defects are randomly injected on multiplexers in cluster local interconnects to make the application not routable. A defect is modeled by undefined value (stuck-open) at the multiplexer output that makes the multiplexer unusable. This number of defective multiplexers corresponds to the defect tolerance of each technique. The number of mux2 added with each technique is compared to initial architecture to measure the area increase. Finally, the Critical Path Delay (CPD) of the application is determined for each technique and compared to the initial architecture. Simulations are done for an FPGA with a channel width of 36, a cluster with 10 CLBs, 4 inputs per CLB, 4 crossbars down, 24 inputs (6 per crossbar down), 12 outputs and three feedbacks per crossbar down (12 feedbacks in all). This cluster architecture correspond to the minimal architecture to place and route each benchmark with 10 CLBs per cluster. Each crossbar down is composed by ten 9:1 multiplexers and the crossbar up is composed by twelve 10:1 multiplexers. For all simulations, we considered that all extra multiplexers and the cluster inputs are fault-free. Configuration memory is protected against the defect by using error detection/correction codes [17] [18]. The design under study are the cluster architecture enriched with all the aforementioned techniques : FGR, IFGR, AFGR, DF and URM. Each technique is applied first only on the crossbars down, then only on the crossbar up and finally on all crossbars.

B. Defect tolerance, area and timing results

The table I presents the performance of the different techniques in term of tolerance, area and timing. The best solutions to bypass a high number of mux2 is for IFGR and DF techniques (respectively 72% and 50%). But using these methods, a lot of the multiplexers are added and that increases the FPGA area (respectively +60% and +35%). On the contrary with FGR, AFGR and URM, the impact on the area is reduced (respectively +18%, +6.5% and +10.4%) but the number of defective elements bypassed is also reduced (respectively 36.4%, 36.4% and 18%). In conclusion, each technique has an advantage in term of area or defect tolerance. With Mesh of Clusters architecture, it's possible to fix a trade-off in term of area and defect tolerance and apply one technique only on a crossbar (down or up) to limit the area increase or combine several techniques to avoid maximum defects. For the Critical Path Delay, each proposed methods (AFGR, URM, DF) reduce the CPD compared to FGR and IFGR because the hardware redundancy is not applied on the total cluster architecture. DF is the best solution (-13%) because new paths are added to the cluster that replace defective elements to limit the impact on the Critical Path Delay.

C. Mesh vs Mesh of Clusters architecture

In this paragraph, we compared the FPGA yield to the number of defect (Fig.5) and timing (Fig.6) for industrial mesh FPGA [12] [19] and all presented techniques on a 36x36 Mesh of Clusters architecture. This correspond to the smallest architecture to place and route all MCNC benchmarks. The yield represents the FPGA functionality after manufacturing. We can note that for all presented techniques applied on a Mesh of Clusters architecture the FPGA functionality can be maintained ($yield = 1$) with more than 100 defect compared to FGR, IFGR and CGR applied on a mesh-based industrial FGPA interconnect [9] [12] [19]. We can also compare all techniques applied on Mesh of Clusters architecture. Best results are obtained using IFGR and DF because the yield is perfect until 428 defects. Finally, the yield decreases with respectively a number of defect tolerated of 214 for FGR and AFGR and 108 for URM. Fig.6, shows the impact of each method on the Critical Path Delay when hardware redundancy is applied. Each method applied on Mesh of Clusters architecture has a lesser impact on the CPD compared to industrial mesh. DF is the best solution (-10%) because new paths are added to the cluster that replace defective elements to limit the impact on the Critical Path Delay.

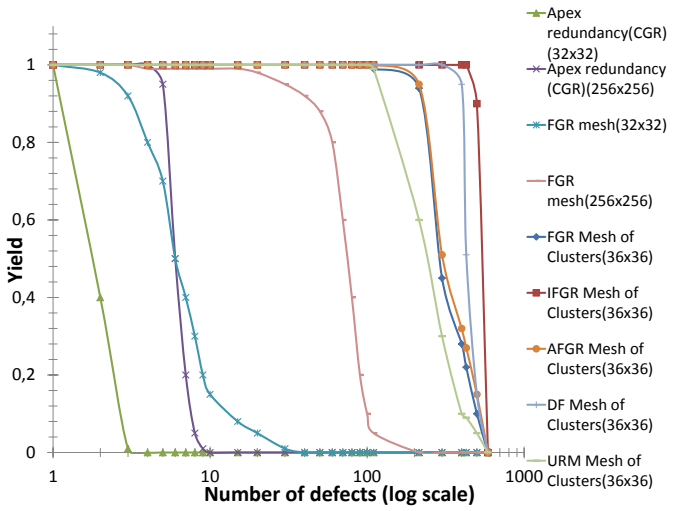

Fig. 5: Impact on FPGA yield

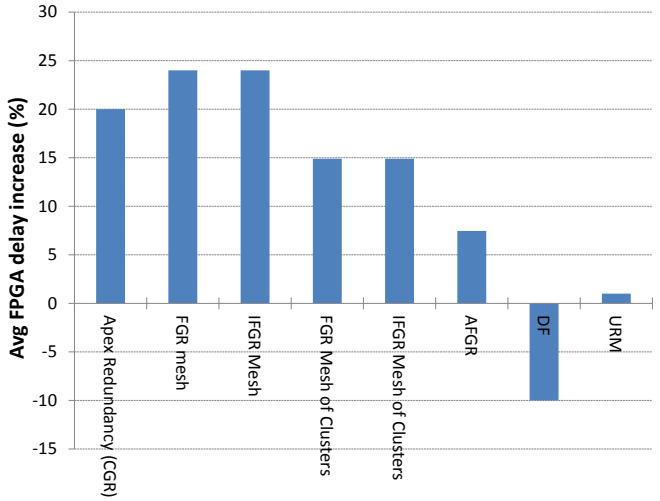

Fig. 6: Impact on Critical Path Delay

V. Conclusion and Future Work

In this paper, we adapted two hardware redundancy techniques used for industrial mesh-based FPGA (FGR and IFGR) to the Mesh of Clusters topology. These techniques permit to tolerate more defects than in a mesh architecture. After new defect tolerant techniques are applied in the cluster of a Mesh of Clusters FPGA. Each method uses hardware redundancy in the cluster local interconnect. However our investigation indicates that the choice of defect tolerant technique has a significant impact on area and Critical Path Delay. With AFGR, we obtain the best trade off between the number of defects bypassed (36.4%), the FPGA area overhead (6.5%) and Critical Path Delay (7.46%). For future works, it will be interesting to mix all these techniques to obtain the best trade off between defect tolerance, area and timing. Then these techniques will be applied to the other element of the Mesh of Cluster FPGA, the Switch-Box to confirm their good performance to the complete FPGA.

Acknowledgment

This work is a part of the project Robust FPGA ANR 11 INS-02, funded by The French National Research Agency, The Pole Systematic and The Pole Minalogic.

References

[1] H. Naeimi and A. DeHon. "a greedy algorithm for tolerating defective crosspoints in nanopla design". In *FPT, 2004.*

[2] N.Campregher, P.Y.K Cheung, and M.Vasilko. "analysis of yield loss due to random photolithographic defects in the interconnect structure of fpgas". In *FPGA, pages 138-148, February 2005.*

[3] A. Mukherjee R. Jain and K.Paul. "defect-aware design paradigm for reconfigurable architecture". In *Emergecing VLSI Technologies and architecures, 2006, IEEE, Computer Society Annual Symposium on vol. 00, 2006, pp. 6pp.-.*

[4] H. Goel and D. Dance. "yield enhancement challenges for 90 nm and beyond". In *Advanced Semiconductor Manufacturing Conference and Workshop, 2003 IEEEI/SEMI, 2003, pp. 262-265.*

[5] Vijay Lakamraju and Russell Tessier. "tolerating operational faults in cluster-based fpgas". In *Int l Symp. on FPGAs, 2000, pp. 187-194.*

[6] Satoshi Kaneko, Abderrahim Doumar, and Hideo Ito. "defect and fault tolerance fpgas by shifting the configuration data". In *Int l Symp. on Defect and Fault-Tolerance, 1999. IEEE Computer Society, 1999, pp.377-385.*

[7] San Jose CA Xilinx. "easypath solutions". In *http://www.xilinx.com/products/easypath/, 2005.*

[8] R. Rubin and A.Dehon. "choose-your-own-adventure routing: Lightweight load-time defect avoidance". In *presented at TRETS, 2011, pp.33-33.*

[9] Altera Corporation. "apex redundancy". In *http://www.altera.com/products /devices/apex/features/apx-redundancy, 2005.*

[10] John D. Corbett Xilinx. "the xilinx isolation design flow for fault-tolerant systems". In *http://www.xilinx.com/support/documentation/white_papers /wp412_IDF_for_Fault_Tolerant, 2012.*

[11] A. Ben Dhia, S. Ur Rehman, A. Blanchardon, L. Naviner, M. Benabdenbi, R. Chotin-Avot, H. Mehrez, E. Amouri, and Z. Marrakchi. "a defect-tolerant cluster in a mesh sram-based fpga". In *in FPT, Kyoto, Japan, pp. 434-437 (2013).*

[12] Anthony J. Yu and Guy G.F. Lemieux. "fpga defect tolerance: Impact of granularity". In *the International Conference on Field Programmable Technology, 2005.*

[13] E. Amouri, A. Blanchardon, R. Chotin-Avot, H. Mehrez, and Z. Marrakchi. "efficient multilevel interconnect topology for cluster-based mesh fpga architecture". In *International Conference on ReConFigurable Computing and FPGAs, 2013.*

[14] Z. Marrakchi, H. Mrabet, and H. Mehrez. "optimized local interconnect for cluster-based mesh fpga architecture". In *in Microelectronics, 2008. ICM 2008. International Conference on, dec. 2008, pp. 15 18.*

[15] Anthony J. Yu and Guy G.F. Lemieux. "defect tolerant fpga switch block and connection block with fine grain redundancy for yield enhancement". In *the International Conference on Field Programmable Logic and Application, 2005.*

[16] Lgsynth93 benchmark set: Version 4.0. In *Technical report, Collaborative Benchmarking Laboratory, 1993.*

[17] C. Slayman. "cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations". In *Device and Materials Reliability, IEEE Transactions on, vol. 5, no. 3, sept. 2005.*

[18] F. Monteiro, S. Piestrak, H. Jaber, and A. Dandache. "fault-secure interface between fault-tolerant ram and transmission channel using systematic cyclic codes". In *On-Line Testing Symposium, 2007. IOLTS07. 13th IEEE International, July 2007, pp. 199-200.*

[19] Anthony J. Yu and Guy G.F. Lemieux. "defect tolerant fpga switch block and connection block with fine-grain redundancy for yield enhancement". In *the International Conference on Field Programmable Logic, 2005.*