Volume : 2 | Issue : 9 | Sept 2013 ISSN - 2250-1991

Research Paper Engineering

Autonomous Self Healing of Reconfigurable Circuits

*B.Harikrishna **Dr.S.Ravi

* Research Scholar, Sathyabama University, Chennai, India

** Professor &Head, Department of Electronics Engineering, Dr.M.G.R University, Chennai

ABSTRACT

This paper presents an analysis of the fault tolerance achieved by an autonomous evolvable system. By using this method the system may self recover from both transient and cumulative faults. In this paper we present a new technique NSCLB for reconfiguring FPGA circuits. An example of 24 CLBs is tested and results show that it may properly recover more number of faults. The faulty CLB is replaced both structurally and functionally. By selecting the nearest spare the routing path is decreased. The method is implemented using VHDL language in Xilinx10.1 version.

Keywords : NSCLB, CLB, FPGA, Xilinx

INTRODUCTION

Field programmable Gate Arrays (FPGAs) arrived in 1984 as an alternative to programmable logic devices (PLDs) and ASICs. As their names implies, FPGAs offer the significant benefit of being readily programmable. Depending upon the requirement a portion of the FPGA can be partially reconfigured while the rest of an FPGA is still running. FPGAs can be programmed again and again, giving designers multiple opportunities to tweak their circuits. Any future updates in the final product can be easily upgraded by simple downloading the new application bit stream. There are many different FPGA architectures available from various vendors for example- Altera [10], Xilinx [11]. A static RAM based FPGA is composed of a two-dimensional array of configurable logic blocks (CLBs), programmable interconnects, and programmable input/output blocks (IOBs). To realize a specific user-given application on a FPGA chip, compiler software provided by the FPGA vendor is required to divide the application

into several parts with each of them small enough to be fit in a CLB, implement each part in a CLB, and finally connect all used CLBs through a programmed interconnect network. Only if all the programmable resources of the FPGA chip used by the application configuration function correctly, the application can run well on the chip. Routing terminology includes routing switch, track, routing channel etc. Routing in FPGAs consists of wire segments of varying lengths which can be interconnected using electrically programmable switches. Density of logic block used in an FPGA depends on the length and number of wire segments used for routing. Number of segments used for the interconnection typically is a tradeoff between density of logic blocks used and amount of area used up for routing.

In this paper we present an efficient online method for reconfiguration of FPGA. In this approach first, we determine if the system can continue to work correctly in the presence of the located faults. In most of the situations this is possible and no reconfiguration is needed. If a fault does affect the system function, we determine the best alternate configurations that avoid the faulty resources. To enable automatic recovery of a device after damage, an autonomous NSCLB algorithm is implemented and tested.

RELATED RESEARCH

In this section, a brief description about some techniques for tolerating faults in FPGAs is discussed. in this paper, we have limited the scope to reflect a few key efforts that provided some information for our work. For a more detailed, quantitative analysis of different online and offline FT techniques, see [2]. Important efforts have been done during the last years within the Immunotronics [3] and Embryonics [4] European projects in order to create an“electronic tissue” to be used for building distributed systems with biological-like self-diagnosis, self-replicating and self-repairing capabilities. All the people involved in these projects agree in stating that the ideal architecture for building AFTSs (Autonomous fault tolerant system) would include several independently and dynamically reconfigurable areas. In fact, the desired architecture has been implemented as ASICs (e.g.OEtic [9], CONFETTI [5]), which incorporate a 2-D array of “electronic cells” that are architecturally equivalent but can be individually reconfigured in order to implement a different functionally at runtime For an average FPGA utilization or 80%, 20% of the resources should be available for spares. In [6], Cuddapah and Corba used Xilinx SRAM-based FPGAs to demonstrate the FT capabilities of FPGAs. In their study, they randomly picked PLBs to be faulty. They reconfigured the circuit around these faults using commercially available PAR tools. The main contributions of their work were an algorithm to determine fault coverage (the ability to reconfigure around a given number of faults) of a design and a definition of the fault recovery rate for any given design implemented in an SRAM-based FPGAs. Additionally, they demonstrated that fault recovery was feasible on FPGAs by other than modular redundant methods. Similar to Kelly and Ivey, their method requires some unused or spare resources, and the fault tolerance depends on how many spares are available. Dutt and Hanchek et al. developed a method to increase FPGA yield. Their method used node covering and reserved routing resources to replace the functionality of faulty PLBs. One row (column) of PLBs is reserved for spares. If a PLB in any given column (row) was faulty, the functionality of all PLBs in the column (row) from the faulty PLB to the spare PLB was shifted toward the spare PLB. Spare routing resources were used to eliminate overhead of rerouting the updated circuit placement. The main advantage of this method is that it is very fast relative to reconfiguration time. Since the spare resources have already been allocated to cover a limited number of faults, the reconfiguration time is linear with respect to the number of faults. Like many of the previous methods, the main problem with this offline technique is the limited number of faults that can be tolerated in each column (row). At most, they guarantee toleration of one fault per spare row or column. This paper is organized as follows section III NSCLB approach describing the overall flowchart. In section IV the implementation part is discussed with explanation at each stage. In section V results and discussion

NSCLB ALGORITHM

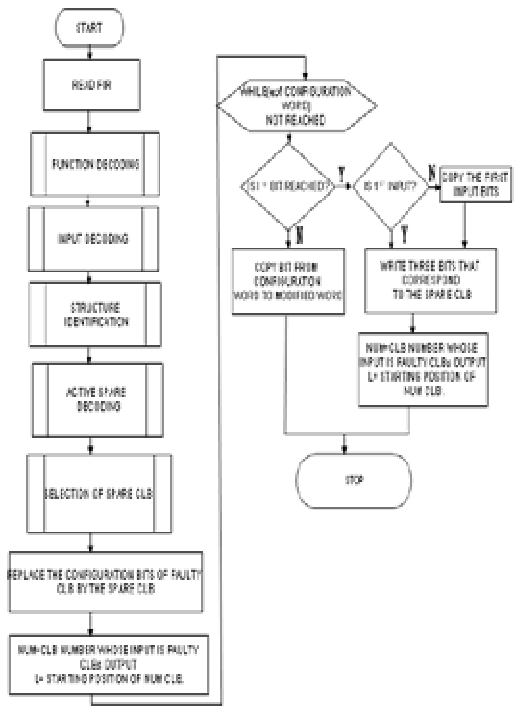
START
READ FIR
FUNCTION DECODING
INPUT DECODING
STRUCTURE IDENTIFICATION
ACTIVE SPARE DECODING
SELECTION OF SPARE CLB
REPLACE THE CONFIGURATION BITS OF FAULTY CLB BY THE SPARE CLB
NUM=CLB NUMBER WHOSE INPUT IS FAULTY CLBs OUTPUT L= STARTING POSITION OF NUM CLB
WHILE(end of CONFIGURATION WORD) NOT REACHED
S1st BIT REACHED?
Y
N
IS 1st INPUT?
N
Y
COPY THE FIRST INPUT BITS
COPY BIT FROM CONFIGURATION WORD TO MODIFIED WORD
WRITE THREE BITS THAT CORRESPOND TO THE SPARE CLB
NUM=CLB NUMBER WHOSE INPUT IS FAULTY CLBs OUTPUT L= STARTING POSITION OF NUM CLB
STOP

Figure 1: showing the overall implementation of approach

IMPLEMENTATION
In this NSCLB (Best left right block) approach when ever any fault occurs the fault is replaced by the best near spare that is available. The algorithm is explained as follows

IV.1.STRUCTURAL IDENTIFICTION
In this the CLB number to which the present CLB is connected is known and in similar way identify for all the CLBs and whole structure is known by this way. Once whole structural description is identified the spare, active CLBs are known explicitly.

IV.2.SPARE SELECTION

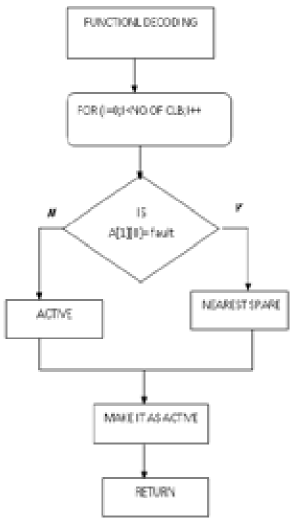

Figure 2: Selection of spare

Finding the nearest spare from the spares available. Here we are finding out the best spare that is suitable for the fault .by selecting the nearest spare the interconnection path is reduced when compared to selecting any of the available spares. And latency is also reduced and reconfiguration is done in time and path delay is less by selecting the nearest spare .by reducing the path delay the time required to get the output in time may achieve. If the spare selected is not near then the path delay is more and signal from input to output is more by this there is considerable delay at the final output.

Lefts [S1] = I; Rights [S1]=I ;

Where S1 indicates the first spare found near the fault. From these two Lefts and Rights the best CLB is selected and given for reconfiguration.

IV.3.RECONFIGURTION OF CLB
Once fault location is known using well known detection and diagnosing fault the next step is reconfiguration. The reconfiguration is done to the selected spare structurally as well as functionally. The fault CLB configuration bits i.e. inputs a function bits will be copied to found spare. Now left thing is structural connection of the fault CLB connection to the spare CLB connection. The following are the steps for connection

1. Decode the CLB number to which the present CLB inputs are connected by using Equ 1.
2. Generate new input bit stream for spare using Equation 1
3. Generate reconfigured configuration bits by replacing new input bit stream and functional bits at spare location.
4. Now update the active spare bit stream indicating the faulty CLB location. By this from the input stream given by resource there will be an extra fault added and one spare less from available 64 CLB. The autonomous restructuring unit recovers the FPGA from its faults by replacing the configuration bits of faulty CLBs with the configuration bits of spare ones. For example when CLB 9 is faulty then the CLB 10 is used as a spare to repair the fault. The inputs and functions performed by CLB 9 are mapped into CLB 10. Thus the configuration bits are reconfigured autonomously.

V.RESULTS AND DISCUSSION
In this example randomly some spares are identified and accordingly some faults are identified and by using the a NSCLB approach the reconfiguration is done .After identifying

the fault next process is finding the spare that is suitable for the fault identified .After finding the best spare that is suitable reconfiguration is done to that faulty spare. Here reconfiguration is done structurally and functionally to the fault with the help of spare that is found by our technique.

IMPLEMENTATION RESULTS

The results obtained by implementing the algorithm presented in section IV is presented here. In this the method finds for nearest spare that is suitable for reconfiguration .The algorithm dynamically selects the nearest spare and fault is reconfigured accordingly. Fig 3 shows the RTL schematic of the 24 CLB structure. The faulty CLB is identified by making it as '1'. Here randomly some faults are identified. The identified faulty status is shown in fig 4.After the faults has been identified the next step is reconfiguration. Fig 5 shows the status of nearest spare selected for the faults that are identified. Here randomly 0, 1, 2 CLBs are identified as faults. According to the method NSCLB the spares selected identified are 3, 5, and 7. The status can be seen in from fig 4 and 5.figure 6 shows the power analysis of the method. The total power is 0.056W. The work is implemented VHDL language in XILINX ISE 10.1 version.

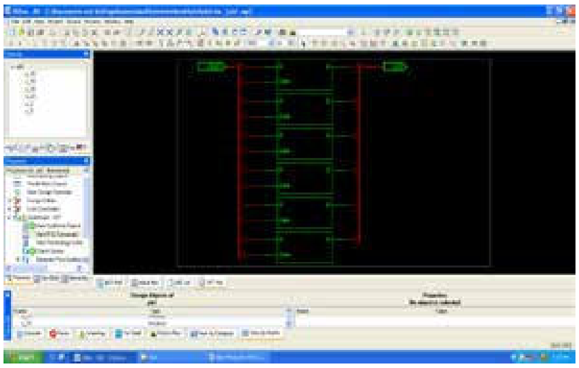

Figure 3:The RTL schematic of the FPGA strucure

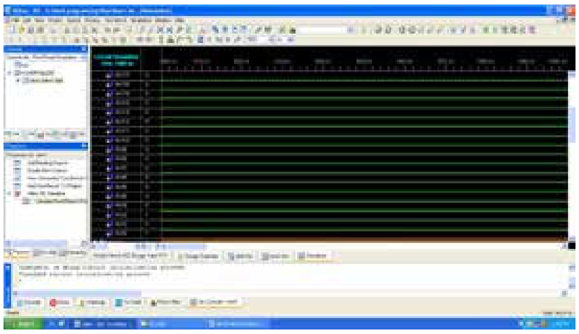

Figure 4: Here '1' indicates the fault.

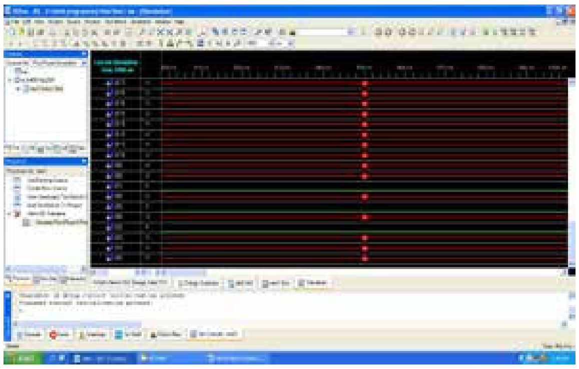

Figure 5: figure showing the status of spares selected for the faulty CLBs.

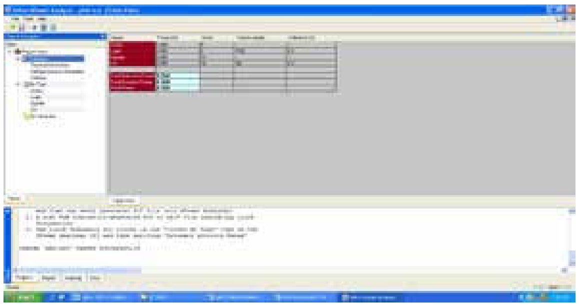

Figure 6: Power analysis of the method

Table 1: shows the total power in Watts.

Total Quiescent power (w)	Total dynamic power (w)	Total power (w)
0.056	0.000	0.056

CONCLUSION

In this approach a new approach to reconfigure the faulty CLB by the best Spare CLB is presented. This approach can work for multiple faults and reconfiguration is done in online. By selecting the nearest spare for reconfiguration the path between the CLB's is short even after occurrence of fault. The autonomous restructuring circuit is designed to modify the configuration word with reduced latency.

REFERENCES

[1]. John M. Emmert, Charles E. Stroud, and Miron Abramovici, "Online Fault Tolerance for FPGA Logic Blocks", IEEE Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 15, No. 2, February 2007, Pp.No. 216-226. | [2] J. Cheatham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for FPGAs," ACM Trans. Des. Autom. Electron.Syst., vol. 11, no. 2, pp. 501–533, Apr. 2006. | [3] D. Bradley, C. Ortega-Sanchez, and A. Tyrrell, "Embry-onics immunotronics:a bio-inspired approach to fault tolerance," NASA/DoD Workshop on Evolvable Hard-ware, 2000. | [4] G. Tempesti, D. Mange, P.A. Mudry, J. Rossier, and A. Stauffer, "Self-replicating hardware for reliability:The embryonics project," ACM Journal on Emerging Technologies in Computing Systems, vol. 3, no. 2, 2007. | [5].P. A. Mudry, F. Vannel, G. Tempesti, and D. Mange,"CONFETTI : A reconfigurable hardware platform for prototyping cellular architectures,"International Paral-lel and Distributed Processing Symposium, 2007. | [6] R. Cuddapah and M. Corba, Reconfigurable Logic for Fault Tolerance. New York: Springer-Verlag, 1995 | [7] Altera Inc., Data Book, 1999. | [8] Xilinx Inc., Data Book, 1999. | [9] J. M. Moreno, Y. Thoma, and E. Sanchez, "POEtic: A prototyping platform for bio-inspired hardware," Evolvable Systems: From Biology to Hardware (LNCS), 2005. | [10] Altera Inc., Data Book, 1999. | [11] Xilinx Inc., Data Book, 1999 |