FAULT TOLERANCE IN FPGA THROUGH KING SHIFTING

R.V.Kshirsagar[1] and S. Sharma[2]
[1]Professor and [2]Lecturer, Priyadarshini College of Engineering, Nagpur, India

ABSTRACT
A wide range of fault tolerance methods for FPGAs have been proposed. Approaches range from simple architectural redundancy to fully on-line adaptive implementations. The homogeneous structure of field programmable gate arrays (FPGAs) suggests that the defect tolerance can be achieved by shifting the configuration data inside the FPGA. All methods and schemes are qualitatively compared and some particularly promising approaches are highlighted. The applications of these methods also differ; some are used only for manufacturing yield enhancement, while others can be used in-system. This survey attempts to provide an overview of the current state of the art for fault tolerance in FPGAs.In this paper we have discussed the king shifting allocation method.

KEYWORDS: *Fault tolerance, Field programmable gate array (FPGA), King allocation.*

I. INTRODUCTION

As process technology scaling continues, manufacturing large defect-free integrated circuits become increasingly difficult. There is also the added problem of degradations over time after a device has been successfully deployed in the field. These reliability issues are particularly acute with FPGAs.
Recently, the defect and fault tolerance in FPGA has been subject in several researches [2]. But the makers are still searching more reliable chips with low cost (area, hardware, complexity, delay, etc.) and high yield improvement. On the other hand for the user side point of view, the requirement of systems with fault tolerance becomes a real necessity, especially for some applications where the time required for the reparation process (downtime) have a critical impact on the system. A recent study suggested that future FPGAs at and beyond the 45nm technology node will have such low yield that defect tolerance scheme will be unavoidable in large FPGAs [1].
By the application of king shifting allocation method, the proposed design makes 3 selections of chip usage possible as follows:

1. Defect-free chips which can be used as a maximum array size of N × N CLBs,
2. Defect-free chips which can be used as fault-tolerant chips of an array size M × M CLBs, where M < N,
3. Defective chips (which are avoidable by the proposed design) can be used as an array size of M × M CLBs.

To reach this goal, the FPGA’s SRAM part is modified so that it will be able to shift the on-chip data vertically and horizontally. It means that the whole user data can be shifted vertically by one row or horizontally by one column. The user data is shifted on-chip, but not modified. Some CLBs are reserved as spare and distributed regularly among the whole FPGA area. When the defect is detected and located by one testing method it can easily be avoided by shifting the whole user data so that the data corresponding to the most nearest spare is shifted in the defective CLB. Consequently the defective CLB will not be used. In this paper the allocation of the spare CLBs within the 2-dimensional arrays, which is named as king-shifting allocation is introduced. In this, when a defect is observed, the data can be shifted in the convenient direction so that the defect is avoided. In the below section we have given the overview of FPGA architecture and discussed the fault, especially the stuck-at-fault associated with it. Thereafter the basic idea of implementation is given and the king allocation method is discussed. The result of the method has been shown along with the conclusion.

II. BACKGROUND

2.1 FPGA Architecture

An FPGA is an integrated circuit composed of programmable routing resources and programmable logic resources. The programmable logic, called configurable logic blocks (CLBs), are composed of k-input lookup tables and flip-flops. Each lookup table can implement any k-input logic function, and connects with a designated flip-flop. Together, the lookup table and flip-flop pair form a basic logic element (BLE). Figure 1 shows a CLB with I inputs and N BLEs. Programmable routing can be further divided into three parts: the wires, the switch blocks (S blocks) and the connection blocks (C blocks). Together, the logic blocks, wires, S blocks and C blocks for modern commercial FPGAs are organized into island-style architecture as shown in Figure 2.2. This architecture has proven to be very successful in modern FPGA architectures and in VLSI. As such, the island-style architecture was also used as the foundation for the new defect-tolerant architecture.

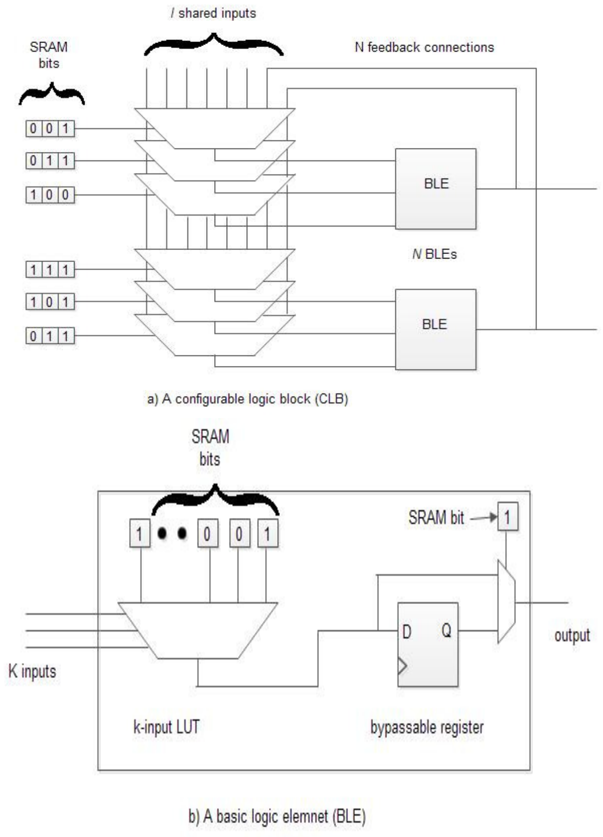


Fig.1 Overview of Fine-Grain Hardware Redundancy

Wires within an FPGA reside in routing channels and are indexed by track numbers. A channel, as shown in Figure 2, spans the width or height of the FPGA and has its boundaries defined by the CLBs. A wire’s track number is based on its position relative to the width of the channel. The convention of this paper is that a wire at the bottom-most/leftmost position in each channel is assigned a track number of 0. Similarly, wires at the top-most/right-most position are assigned a track number of channel width – 1.

2.2 Other types of fault

In addition to degradation, there are two other types of faults which can affect FPGAs. These are highly relevant to this study as some of the techniques which have been developed in response to them can also be applied to faults caused by degradation. The first of these is manufacturing defects. Manufacturing defects can be exhibited as circuit nodes which are stuck-at 0 or 1 or switch too slowly to meet the timing specification. Defects also affect the interconnect network and can cause short or open circuits and stuck open or closed pass transistors. Test and repair of manufacturing defects is well established in VLSI. The second type of fault which is widely discussed in relation to FPGAs comprises of Single Event Upsets (SEUs) and Single Event Transients (SETs) caused by certain types of radiation.

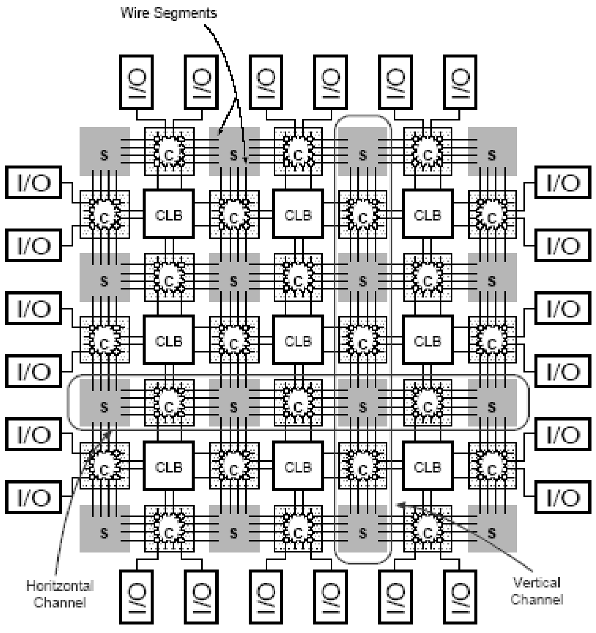


Fig.2 Island-style Architecture

III. THE BASIC IDEA OF IMPLEMENTATION

This design aims at giving the FPGA the possibility to shift the data of the configurations cells corresponding to the unit elements serially, vertically, and horizontally.

1. With the serial shifting, the FPGA can load the original configuration data into the SRAM serially.
2. With the horizontal shifting, the FPGA shifts in parallel the original configuration data of each unit column (one CLB column with its adjacent switching column) to the next unit column. Therefore, by shifting the data of one unit column, the data of column 1 will be shifted to the column 2 and the data of column 2 to the column 3, and the data of the last column will be shifted to the first column.
3. With the vertical shifting, the FPGA will be able to shift in parallel the original configuration data of each unit row (one CLB row with its adjacent switching row) to the next unit row. Therefore, by shifting the data of one unit row, the data of row 1 will be shifted to the row 2 and the data of row 2 to the row 3, and so on, and the data of the last row will be shifted to the first row.

For achieving this goal, the original hardware of the SRAM must be slightly modified in a new design. Since the column SF (stuck-at-fault) at the right side and the row SF at the bottom are not constructed into unit elements, we divide the SRAM into 3 parts: the homogeneous part, the non homogeneous part and the IOB part as shown in Fig. 3. The first part is constructed by the configuration cells of the unit elements only, and the second part is constructed by the configuration cells of the most right column and the row at the bottommost. In addition all configuration cells corresponding to the same unit element are juxtaposed in one set serially. For the SRAM part (Fig. 3

(a)) corresponding to each unit element, one multiplexer is added. These multiplexers allow the data corresponding to each unit element to be shifted serially or vertically or horizontally following the multiplexers.

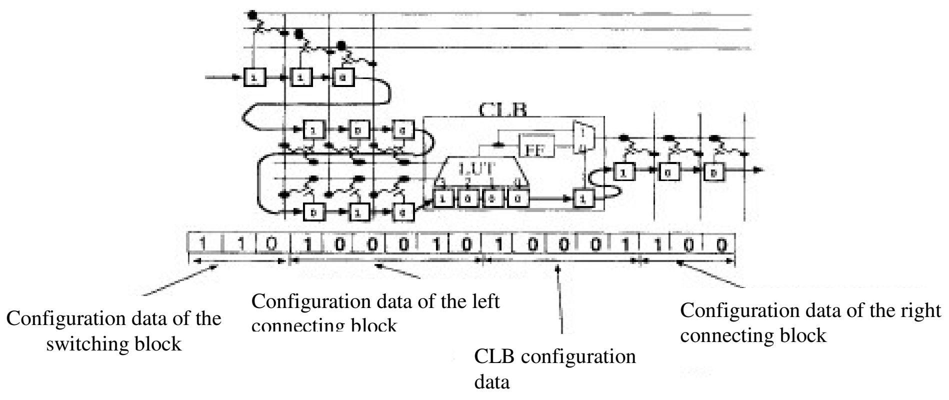
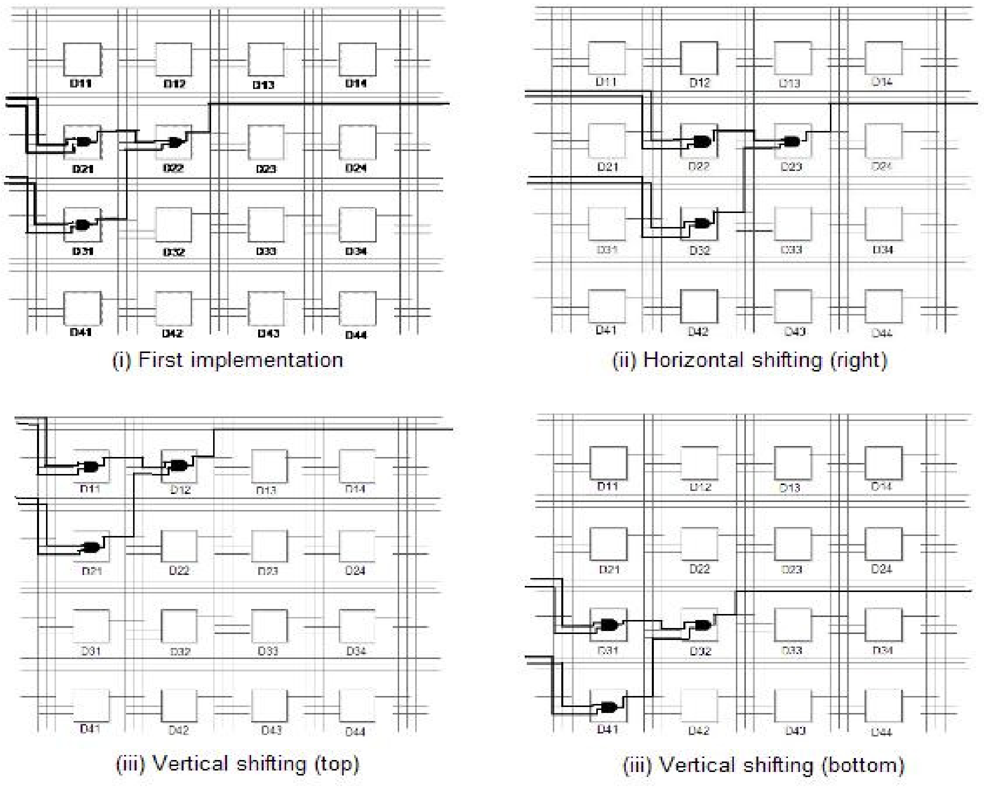

CLB
FF
LUT
1 1 0 1 0 0 0 1 0 1 0 0 0 1 1 0 0
Configuration data of the switching block
Configuration data of the left connecting block
CLB configuration data
Configuration data of the right connecting block

Fig.3(a) Example of unit element configuration data

The data in the homogeneous part can be shifted, while the non-homogeneous part does not allow the shifting. The separation between the homogeneous part and non homogeneous part is assured by one control signal commanding to one pass transistor.


D11 D12 D13 D14
D21 D22 D23 D24
D31 D32 D33 D34
D41 D42 D43 D44
(i) First implementation
(ii) Horizontal shifting (right)
(iii) Vertical shifting (top)
(iii) Vertical shifting (bottom)

Shifting the configuration data vertically and horizontally

Fig.3(b) Basic example of shifting the configuration data.

Based on the same idea, the configuration cells corresponding to the IOBs at every side (north, south, west and east) of the chip are gathered serially as shown in Fig. 3(b). One multiplexer permits to the data to be looped serially in each side separately. The multiplexer in each side are controlled by the shift controller. The memory cells corresponding to the IOBs in all sides are connected serially with the memory cells corresponding to the unit elements. The shift controller controls the additional hardware as the additional multiplexers, additional switches, etc.

IV. KING SHIFTING ALGORITHM FOR ALLOCATION OF SPARE RESOURCES

Fault tolerance is achieved only by shifting the original configuration inside the FPGA. King-shifting algorithm provides an effective way of distribution of spare allocation. In the chess game, the king has the possibility to protect the 8 cells around the cell where it is standing. Inspiring from this fact, the king-shifting distribution shown in Figure 5(a) reserves the CLB in the middle of each 3 x 3 CLBs within the chip as spare and gives the possibility to each spare cell to cover the 8 cells around it position. The original configuration data, which is mapped in the usable cells, can be shifted on-chip by some columns. When a defect/fault is detected the whole user data is shifted by one of the 8 directions. The defect/fault wherever it is situated within the 2-dimensional arrays, is assured to be covered by this shifting.

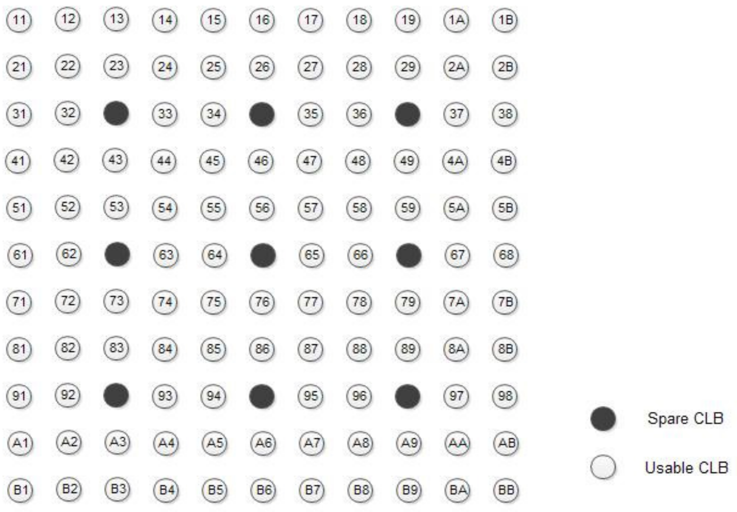
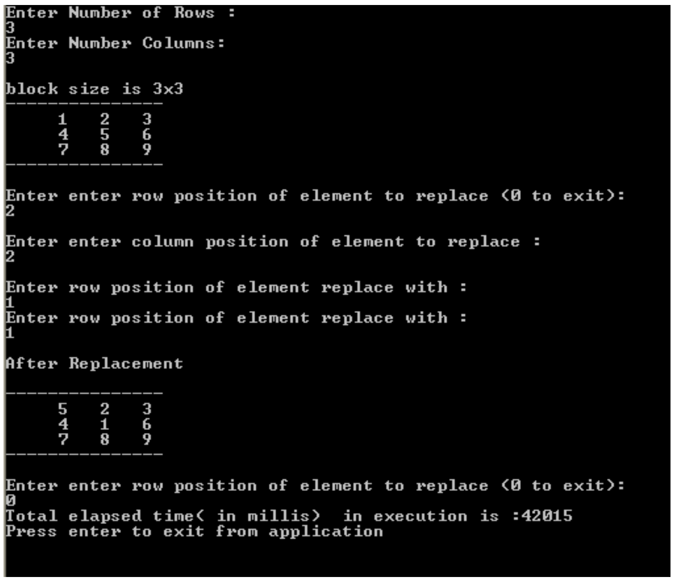


Figure 4: King Shifting Allocation

The basic algorithm for King-shifting method is shown below:
1. Create the CLBs block size of 3 X 3.
2. Each 3 X 3 are having 9 CLBs.
3. Allot one spare CLB in each 3 X 3 block.
4. If one of the 9 CLB is faulty then it should get swap with the spare CLB present in 3 X 3.

V. EXPERIMENTAL RESULTS

In all the previous work that has been done in this research area, parameters like yield, memory overhead and CLB overhead has been taken into account. In our paper we have highlighted on the execution time of this algorithm. We have implemented the proposed algorithm by writing a code using C language. The result that we are getting is according to the above algorithm, the swapping of the CLBs has been achieved. The spare blocks in this case are dependent on the number of CLBs and we didn’t take any spare CLBs across the rows and columns as it was taken in the previous algorithm.



```
Enter Number of Rows :
3
Enter Number Columns:
3

block size is 3x3
---------------
     1   2   3
     4   5   6
     7   8   9
---------------

Enter enter row position of element to replace (0 to exit):
2

Enter enter column position of element to replace :
2

Enter row position of element replace with :
1
Enter row position of element replace with :
1

After Replacement

---------------
     5   2   3
     4   1   6
     7   8   9
---------------

Enter enter row position of element to replace (0 to exit):
0
Total elapsed time( in millis)  in execution is :42015
Press enter to exit from application
```

Figure 5: Result of King Shifting Allocation

VI. CONCLUSIONS

In this paper, a novel fault tolerant technique for FPGA-based systems is presented. Here fault tolerance is achieved by only shifting the design data automatically, without changing the physical design of the running application, without loading other configurations data from the on-chip FPGA. This paper highlights on the King-Shifting algorithm for the effective distribution of spare resources. In our paper we have computed the execution time of our algorithm and thus has taken into account the timing overhead associated with it.

REFERENCES

[1] N. Campregher et al, “Analysis of yield loss due to random photolithographic defects in the interconnect structure of FPGAs”, ACM Int. Workshop on FPGAs, p.138-148, 2005.

[2] J N. Hastie and R. Cli, “The implementation of hardware subroutines on field programmable gate arrays,” Proc. IEEE Custom Integ. Circuits Conf., pp.31.4.1–31.4.4, 1990.

[3] XILINX data book, Field programmable gate arrays, 1998.

[4] ALTERA data book, Field programmable gate arrays, 1993.

[5] S.D.Brown, R.J.Francis, J.Rose, and Z.G.Vranesic, "Field-Programmable Gate Arrays", Kluwer Academic Publisers, 1992.

[6] N. J. Howard, A. M. Tyrrell and N. M. Allinson, “The Yield Enhancement of Field- Programmable Gate Arrays, "IEEE Tran. on VLSI.,Vol. 2, pp. 115-123, March-1994.

[7] J. Narasimhan, K. Nakajima, C. Rim, and A. Dahbura, The Yield Enhancement of Programmable ASIC Arrays by reconfiguration of circuit Placements," IEEE trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.13, no, pp 976-986, Aug. 1994.

[8] A.Mathur and C. L. Liu, “Timing driven placement recon_guration for fault-tolerance and yield enhancement in FPGA's," Proc. Ed and TC96,pp 165-169, 1996.

[9] A.Jain and J. Rajski,\ Probabililistic Analysis of Yield and Area Utilization of Reconfigurable Rectangular Processor Arrays," Defect and Fault Tolerance in VLSI Systems, I. Koren,, New York:Plenum,1989, pp. 269-280.

[10] F. Hatori et al., "Introducing Redundancy in Field Programmable Gate Arrays," Proc. IEEE Custom Integrated Circuits Conf.,pp.7.1.1-7.1.4,1993.

[11] J.Lach, W. H. Mangione-smith, and M. Potkonjak, “Low overhead Fault Tolerance FPGA Systems, "IEEE trans. on VLSI, Vol. 6, No. 2, June 1998.

[12] F. Hanchek and S. Dutt,”Methodologies for Tolerating Cell and Interconnect Faults in FPGAs, " IEEE trans. on computer, pp 15-33, January 1998.

[13] A. Doumar and H. Ito, “ Self-Testing and Diagnosis for SRAM-FPGAs , accepted to appear in Pacific Rim International Sym. on Dependable Computing (PRDC), Hong Kong, Dec. 1999.

[14] A. Doumar and H. Ito , “Testing the Logic Cells and Interconnect Resources for FPGAs," accepted to appear in IEEE Asian Test Symposium, China, Nov. 1999.

[15] A. Doumar, T. Ohmameuda, and H. Ito, "Design of an Automatic Testing for FPGAs," IEEE European Test Workshop, Germany, 1999.
[16] I. Koren and C. H. Stapper, "Yield models for defect tolerant VLSI circuit: A review,"Defect and Fault Tolerance in VLSI Systems, I. Koren, Ed. New York: Plenum,1989, pp. 1-21.
[17] F. Meyer and D. K. Paradham, "Modeling defect spacial distribution," IEEE Trans. on Computer,vol.38, pp. 538-546, Apr.1989.
[18] C. Thibeault, Y. Savaria, and J. L. Houle, "A New Yield Formula for Fault-Tolerant Large-Area Devices," Defect and Fault Tolerance in VLSI Systems, I. Koren, Ed. New York: Plenum,1989, pp.53-64.

Authors

Ravindra V. Kshirsagar (FIETE, LMISTE) is presently working as Professor and Vice Principal, Priyadarshini College of Engineering. He is also the Dean, Faculty of Engineering and Technology of R.T.M.,Nagpur University, Nagpur. He has done his B.E.(E&TC) in 1984 from Govt. Engg. College, Jabalpur. He completed his M.Tech. (Electronics Engg.) and Ph.D. from VNIT,Nagpur and has a vast teaching experience of 20 years and 2 years of industry experience. He has published many research papers in national and international conferences. He is a fellow member of IETE and LMISTE .Also he was Ex - IEEE member. His special field of interest includes Reconfigurable Computing, VLSI Design, Fault tolerance and DFT.

Sanjeev Sharma received his B.Tech degree from IET, Barielly and received his M.Tech degree from RKNEC, RTMNU, India, he received his PG Diploma in VLSI from CDAC, now he is a Ph.D candidate of RTMNU, and he has over 6 years of industrial experience, working at CDAC, now he is a lecturer at PCE. His overall 10 Papers published in international and national conference on selected areas in communication and embedded systems. His research interests include artificial intelligence, wireless communication, embedded systems and soft-core processor.