Journal of Computer Science 10 (3): 544-551, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.544.551 Published Online 10 (3) 2014 (http://www.thescipub.com/jcs.toc)

RECONFIGURING EVOLVED CIRCUITS USING CONTROLLER: A REAL TIME APPROACH

[1]B. Hari Krishna and [2]S. Ravi

[1]Research Scholar, Sathyabama University, Chennai, India
[2]Research Supervisor, Dr.M.G.R University, Chennai, India and
Professor (PG studies), GKM College of Engg., Chennai, India

Received 2013-10-16; Revised 2013-12-02; Accepted 2013-12-02

ABSTRACT

FPGA is a device that contains a matrix of reconfigurable gate arrays that can implement different complex functions and also vary structurally. In this study, a novel method of reconfigurable circuit switching action is done with the help of a controller. BPSK Modulation application is chosen to demonstrate the reconfigurable action. When an event occurs the context/core switching is done at both input and output and also preserving the structural and functional relationship. The reconfiguration ability of FPGA with an embedded core performing the switching in real time is the focus of this study. A controller running on Linux platform is used along with the FPGA core. In this study, we propose a real time approach for reconfiguring evolved circuits using controller. It has an adaptive hardware that can continuously change in response to the input data and there by perform reconfiguration. Here the reconfiguration process is achieved by using a real time controller which gives a command to the FPGA core (ALTERA (Cypress processor)) for reconfiguration.

Keywords: Switching, FPGA, LINUX, BPSK

1. INTRODUCTION

In this study, a novel method of reconfigurable circuit switching action is done with the help of a controller. BPSK Modulator application is chosen and implemented in FPGA with the ability to reconfigure its circuitry for a variety of applications. It has an adaptive hardware that can continuously change in response to the input data and there by perform reconfiguration. Here the reconfiguration process is achieved by using a real time controller which gives a command to the FPGA core (ALTERA (Cypress processor)) for reconfiguration. This study is organized as follows: Section 2 explains the evolvable hardware with autonomous Reconfiguration describing the block diagram and structural and functional description. Section 3 describes some techniques for tolerating faults in FPGA. Section 4 describes the separation of inputs and outputs and structural description of evolved hardware. Section 5 explains the status monitoring of FPGA core handling in FPGA. Section 6 details about internal mux for selecting inputs and functions. Sections 7 describes about the method of mapping our application in to the FPGA. In section 8 describes the Reconfiguration process done in controller followed by results and conclusions are done.

2. EVOLVABLE HARDWARE WITH AUTONOMOUS RECONFIGURATION

The proposed Evolvable hardware is shown in **Fig. 1** the module consists of;

2.1. Real Time Controller

The Real time controllers have to be designed so that they use low power and have to provide high performance. The LPC 178x/177x is used for embedded applications and it is based on microcontroller called ARM Cortex-M3. The ARM microcontroller provides optimal performance and enhancements like a higher level of support block integration and modernized debugging features. Pipeline techniques are use in order to continuously operate the memory systems as well as the parts of the processing.

Corresponding Author: B. Hari Krishna, Research Scholar, Sathyabama University, Chennai, India

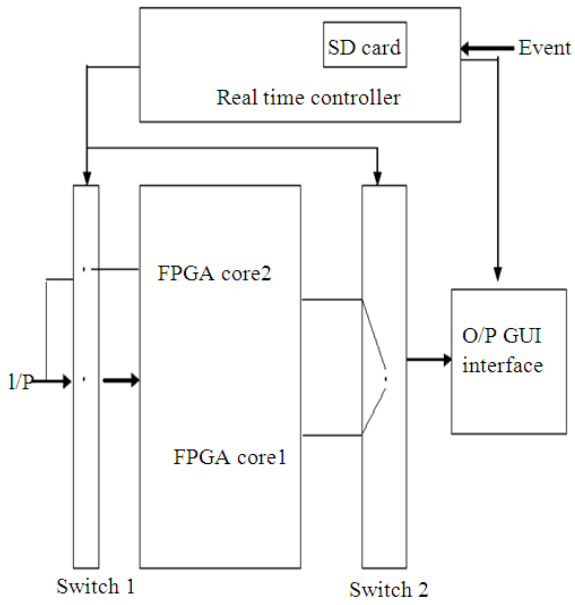


Fig. 1. Block diagram of the proposed evolvable hardware with controller

The task of real time controller is (i) To log the activity inside the core (in the sdcard) (ii) Whenever a reconfiguration initiation event occurs, the context/core switching both at i/p and o/p is performed (iii) Preserve the structure and functionality with Reusability and fault tolerance. Here the event is the fault that occurs on the FPGA core. When reconfiguration is done between the cores a new configuration structure is loaded to start a new evolution.

2.2. FPGA Cores

The configuration bits for each task is preloaded into an SD card. The configuration bits are loaded in the FPGA core by Real time controller. The FPGA consists of CLB's, Input/output blocks, Block RAMS.

2.3. Configurable Logic Blocks

CLB's are the main resource for implementing any circuitry. A CLB is a collection of slices. A slice includes:

- Multiplexers
- Inverters
- Buffers
- Flipflops
- Look-Up Table (LUT)

When a fault has been identified in FPGA, it needs to be reconfigured. To replace faulty CLBs, we use spare CLB's and continue the functionality. To use spare CLB's effectively, the following main problems must be solved:

- Detection and location of failure during the application
- Reconfigure the FPGA structure

In this core if any CLB becomes faulty then reconfiguration process is done and accordingly the switching action is done by the controller and status displayed in a GUI.

3. RELATED RESEARCH

In this section, a brief description about some methods for fault toleration in FPGAs is discussed. In this study, we provide some efforts that provides some key information for our proposed work. Cheatham *et al*. (2006), a detailed quantitative analysis of various FT methods (both offline and online) are given. Several techniques used either row-wise shifting or column-wise shifting (Hatori *et al*., 1993; Caponetto *et al*., 2007). Hatori *et al*. (1993), the authors provided a single spare column technique for fault toleration. The authors used specialized selector circuitry, which is useful to reconfigure faulty FPGAs. Like SRAMs fault tolerance methods, the faulty column in the PLB is eliminated and all operations of the columns between the faulty one and the spare are shifted using the spare column algorithm. Narasimhan *et al*. (1991; 1994), the authors developed a pebble shift method for fault tolerance in FPGAs. Their methods are based on using unused resources, which is used in fault tolerance. Dhia *et al*. (2013), the authors used redundancy method for bypassing faults in FPGAs. Their fault tolerant method depends on shifting, which is useful for reconfiguration. The authors used normal place and route tools for mapping circuits in FPGAs. Hatori *et al*. (1993) and Caponetto *et al*. (2007), the authors developed a switch matrix network. Emmert *et al*. (2007), the authors proposed a technique that uses a spare or an unused resource for fault tolerance. This FT depends on the routability and the available unused resources. Sedcole *et al*. (2006), the authors proposed a reconfiguration technique that can unload and load modules dynamically. Raghuraman *et al*. (2005), the authors proposed a method for size reduction of bits in reconfiguration of FPGA. This can be done by adjusting LUT inputs orders. Thus the relocation of memory areas into common frames is done.

4. REPAIR MODEL

The repair model process is as follows:

- The inputs for the repair model are the configuration bits that are stored in SD-Card. This configuration bits define the architecture of the given module
- Separate these bits in to inputs, outputs and functions
- Identify the interconnections of the separated bits

4.1. Separation of Bits

Once the event from the faulty resource to reconfigure the FPGA is obtained, the bit streams are decoded as inputs, outputs and functions. A sample application is chosen and realized on an evolved circuit is considered. The application uses 217 bits as the configuration word for a FPGA and has 25 Configurable Logic Blocks (CLBs). The VRC decoder gives a 25 bit word as input to represent the active and spare CLBs among the 25 CLBS in the VRC.

4.2. Structural and Functional Decoding

This is the one of the main task. In this we identify the CLB number to which the present CLB is connected and similarly identify for all the CLBs and whose structure is explicitly identified by this way. Once whole structural description is known then spare, active CLBs are known explicitly. Once fault location is known the next step is to perform the reconfiguration.

5. STATUS MONITORING OF FPGA CORES

The structural and functional monitoring of PEs in the FPGA core is done in runtime and logged continuously into a secondary device like SD-Card. The monitored details include:

- Function performed by each PE
- Structural relationship (both i/p and o/p) among PEs
- Active and Spare PE
- Context switching time in the event of reconfiguration
- Application specific features

The pseudo code of data logging into the SD-card is shown in **Fig. 2**.

```
n=0;
            do
             {
              ch = _DG;
              Buff[n] = ch;
              n++;
              if (n>19)
              {
                        ret = f_write(&File, Buff,
20, &write_size );
                        Buff[10] = 0x00;
                        _DBG_("Written to file:");
                                  _DBG_(Buff);
                             n=0;
                        }
              }         while (ch! =STOP);

                         if (n>0)
                         {
                              n--;
                         Buff[n] = 0x00;
 ret = f_write (&File, Buff, n, &write_size );
              _DBG_("Written to file:");
                                  _DBG_(Buff);
                        }

            ret = f_utime(fname,&Finfo);
            _DBG_("\n\r");
            _DBD32(size);
            _DBG_(" bytes has been written to
the file");
            f_close (&File);
```

Fig. 2. The pseudo code of data logging

In the data logger module (Ex: SD-card) the following steps are followed:

- Initialization (Upon successful logger initialization a non-zero value is returned)
- To log the contents into a file and then interface to a media (ex: SD Card)
- To buffer and display in GUI or hyper terminal

6. INTERNAL MUX FOR SELECTING INPUTS AND FUNCTIONALITY

During evolution it is necessary to evaluate different circuits and this is most efficiently undertaken in reconfigurable hardware such as the FPGA. A typical FPGA consists of large array of reconfigurable blocks whose inputs and outputs are connected through a set of wires. This is because of that the VRC does not depend on the desired platform; Thus VRC can be attached to evolutionary part of the FPGA itself. **Figure 3** shows internal multiplexer for selecting inputs and functions.

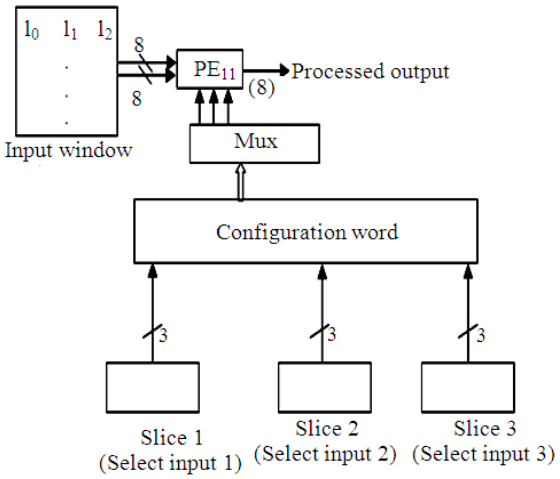


Fig. 3. Internal MUX for selecting inputs and functions

The inputs controlling the functionality of the VRC is selected through the multiplexer. Three bit MUX are used for the development of every PE. The VRC's memory component is developed as a register set. Now the configuration bit stream is linked to MUX that is used for functioning in each PE. This also helps in controlling routing. Feedback is not preferred (to avoid delay) and thus combinational circuits alone are included in the functional design.

7. METHOD OF MAPPING BPSK INTO HARDWARE

- Separate the algorithm into sections to be implemented on hardware and software
- Synthesize the algorithm destined for reconfigurable hardware into gate-level or circuit level description
- Map the circuit onto reconfigurable blocks and connect them using reconfigurable routing
- After compilation, the circuit is ready for configuration onto the hardware at runtime
- Continuously monitor the status of the evolved circuit with the help of fault models
- Whenever a reconfiguration initiation event occurs, the context/core switching both at i/p and o/p is performed

7.1. Fault Models

It is necessary to detect the occurrence of faults in the circuit. The most popular faults are Struck-At-model. In struct at model, a faulty gate input is modeled as Struck-At-zero (S-A-0) and struct-at-one (s-A-1) fault. These faults most frequently occur due to gate oxide shorts or metal to metal shorts.

It is observed that the variation in power level is sufficient enough to detect the occurrence of (S-A-0) and (S-A-1) fault. "Struck-at" refers to a condition where a defect cause a circuit node to become "struck" at a logical one or logical zero.

From **Fig. 4** we can see the power dissipation of the evolved PE under fault and fault less condition. This is given as input for the reconfiguration module. By monitoring the fault status reconfiguration is done by selecting spare or reusing faulty CLB (Krishna and Ravi, 2012).

8. RECONFIGURATION

When an event occurs the context/core switching is done at both input and output and also preserving the structural and functional relationship. The reconfiguration process algorithm in the controller is shown in **Fig. 5**.

From **Fig. 5** the correcting factor will be dependent on the FPGA that is taken. The algorithm presented in the **Fig. 5** is written in the controller. Whenever any fault occurs reconfiguration will be done automatically by selecting the spare or by reusing the faulty CLB for reconfiguring the fault. After reconfiguration the changed configuration bits are again downloaded in to the FPGA core.

9. SIMULATION RESULTS

Figure 6 shows the VRC evolved for BPSK modulation scheme. This design is based on sysgen Xilinx. The platform used is simulink. The sysgen file generates the .ise Project file along with the .ucf file for the hardware and this bit file is downloaded into the FPGA target. With the system generator wave scope, the waveforms generated in the design are verified.

The selection details for the evolved BPSK modulation architecture is listed in **Table 1**. The reconfigured and reusable architecture details are listed in **Table 2**.

A fault has been introduced into the FPGA as a result decoded bits are erroneous a change in the decoded bits. The sequence under consideration is:

1000001101

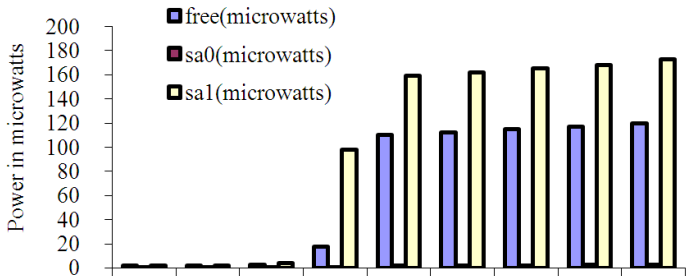
Because of the result of faulty CLB in hardware the decoded sequence is

1001001101

Error.

Hence hardware need to be reconfigured up on doing reconfiguration with the method as explained above the corrected decoded sequence is:

1000001101



	1	2	3	4	5	6	7	8	9
free(microwatts)	2	2	3	18	110	112	115	117	120
sa0(microwatts)	1	1	1	1	2	2	2	3	3
sa1(microwatts)	2	2	4	98	159	162	165	168	173

Vdd in volts

Fig. 4. Power graph for evolved PE under fault and fault less outputs

1)Find the faults in the FPGA
2) Find the spare to reconfigure for the faulty CLB
3) Next step is reconfiguration:
i) Find the inputs and outputs of the faulty CLB and the function.
Let us take CLB X as fault and for this identified spare is Y.

To this CLB X output is connected to X0, X1, and X2.

Take T=X0 i.e first output
Now a = X0 consider the configuration numbers for this are 001 and b= 101
So a = 001=1 & b= 101=5
So X CLB number correcting factor is Z. the correcting factor will be dependent on the FPGA chosen.
a=Z+001=Z0
b=Z+101 = Z1
then we check 'a' and 'b' with X then subtract the correcting factor from spare
a= Spare - correcting factor
a = spare – correcting factor of X0
a= Y-Z=0
So update this value in the i.e a=000 in X0.
By this the reconfiguration is done. To check this take 'a' value if we add correcting value we get spare number .by this we understand that it is connected from spare (Y) not from X that is a faulty CLB.
4) The same process is repeated for other outputs also to map to the spare CLB.

Fig. 5. Reconfiguration process in the controller

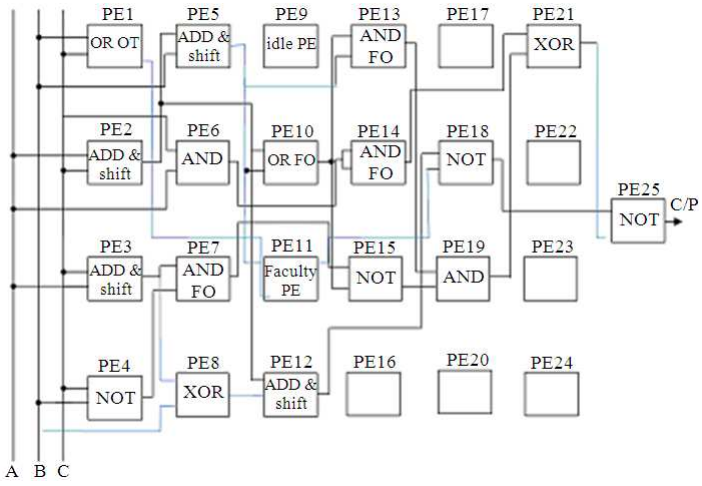


Fig. 6. VRC evolved for BPSK modulation 1

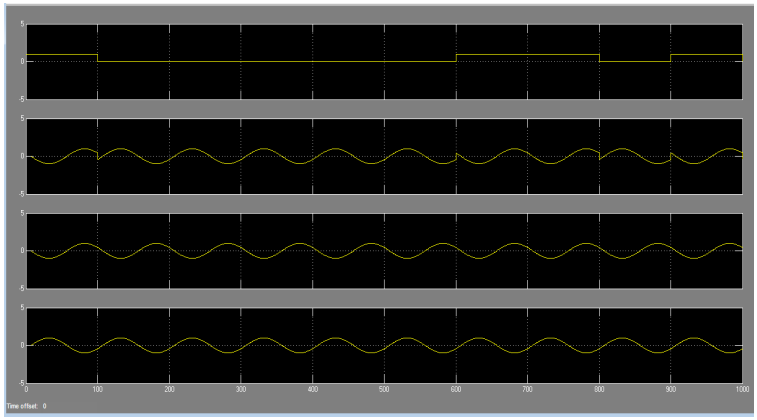


Fig. 7. Data and modulated signal

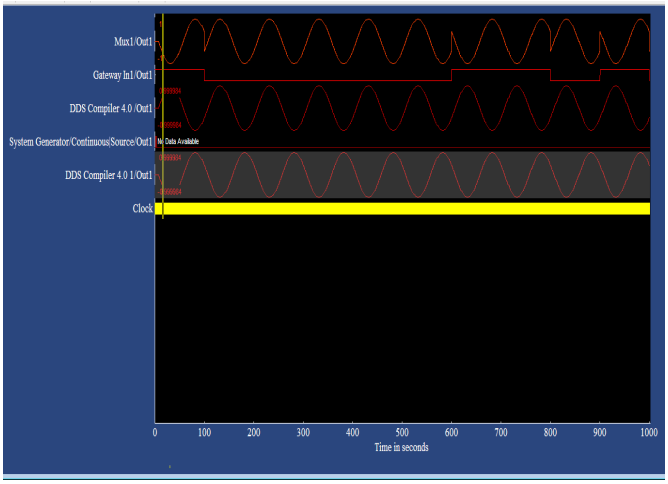


Fig. 8. VHDL simulated waveform

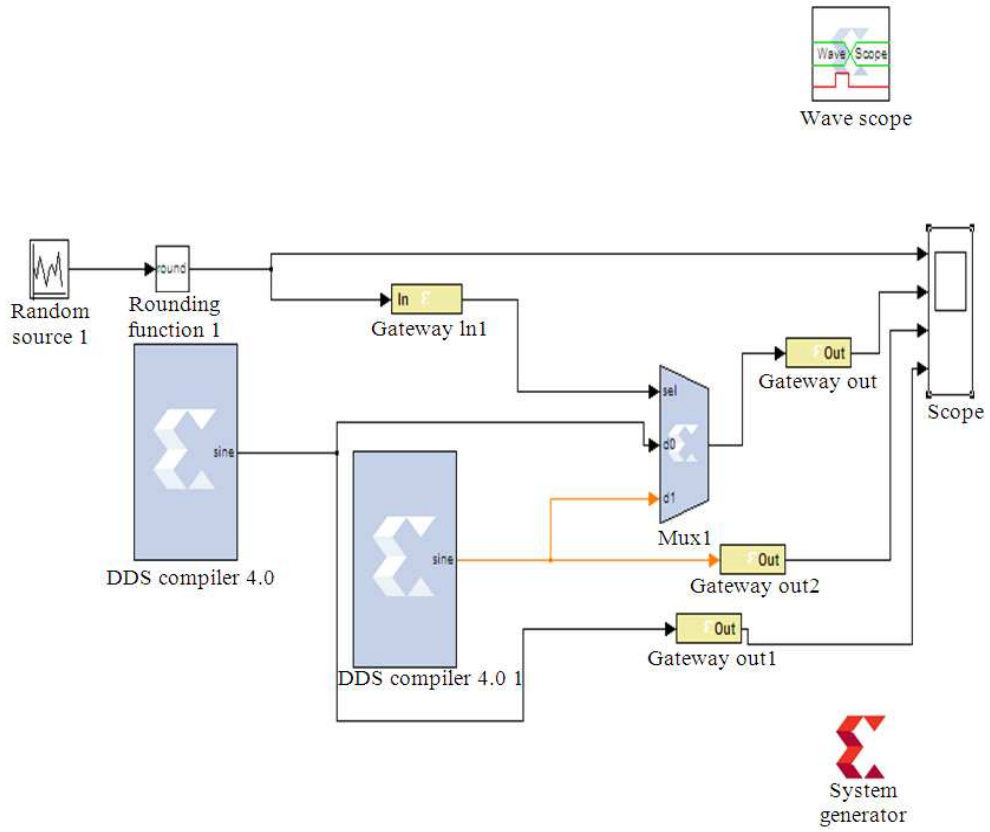


Fig. 9. BPSK modulator implementation

Table 1. BPSK modulation signal

BPSK modulation signal	Function
For mux1: Inputs given as d0, d1	To specify signal modulation.
Multiplexer 1 selection input	Choose between d0, d1.
For mux-2: Depends on the o/p of mux1	Choose either+sinus/ -sinus
Context/core switching (i/p and o/p) perform	Preserving the structure, functionality with reusability and fault tolerance

Table 2. PE Configuration bit

PE Configuration	Function
If corresponding bits to the PE considered	PE is not ideal
Input for faulty PE and Output from the faulty PE	Mapped to the nearest idle PE (present in the 1st row 3rd column)
Take input from the faulty PE	Re-direct to spare PE
Reconfiguration results of spare PE	Select for the identified fault

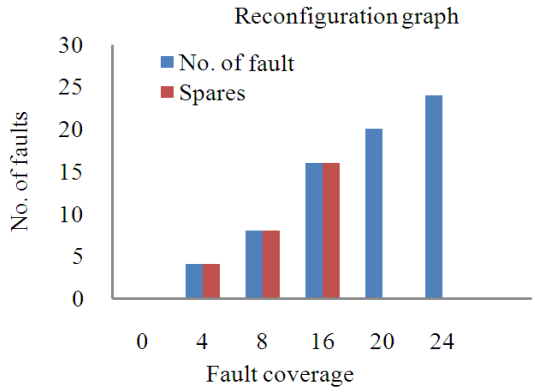


Fig. 10. Reconfiguration of faulty CLB by spare and using faulty CLB

Table 3. CLBs used for reconfiguration

Status	Spare available	Fault available	CLB used for reconfiguration.
CLB 4 faulty	CLB 16	CLB 3	CLB 3
CLB 9 faulty	CLB 17	0	CLB 17

The reconfigured results consist of binary data sequence, in-phase and quadrature components BPSK modulated signal is shown in sysgen scope viewer and the Xilinx ISE output in **Fig. 7-9**. The reconfiguration process uses either spare CLB or Faulty CLB for reconfiguration. The faulty coverage done by our method by reusing the faulty CLB is shown in **Fig. 10**. By reusing the faulty CLB we are able to reconfigure more number of faults than the number of spares available. The selection of spares or faulty CLB is shown in **Table 3**.

10. CONCLUSION

In this study, a novel method of reconfigurable circuit switching action is done with the help of a controller. A BPSK modulation application is chosen to demonstrate the reconfigurable action. Whenever any fault or channel change event occurs the reconfiguration command comes from the controller to the FPGA core to perform reconfiguration. Results prove that this method is effective and can adapt quickly to changing conditions and is switching sensitive. Whenever a reconfiguration initiation event occurs, the context/core switching both at i/p and o/p is performed for preserving the structure and functionality along with Reusability and improved fault tolerance. Choosing the optimal level back is the limitation of the paper. Enhancing the Reusability is the future scope.

11. REFERENCES

Caponetto, R., G. Dongola and L. Fortuna, 2007. A new class of fault-tolerant systems: FPGA implementation of bio-inspired self-repairing system. Proceedings of the Mediterranean Conference on Control and Automation, Jun. 27-29, IEEE Xplore Press, Athens, pp: 1-4. DOI: 10.1109/MED.2007.4433752

Cheatham, J.A., J.M. Emmert and S. Baumgart, 2006. A survey of fault tolerant methodologies for FPGAs. ACM Trans. Des. Autom. Electron. Syst., 11: 501-533. DOI: 10.1145/1142155.1142167

Dhia, A.B., S.N. Pagliarinia, L.A.D.B. Naviner, H. Mehrez and P. Matherat, 2013. A defect-tolerant area-efficient multiplexer for basic blocks in SRAM-based FPGAs. Microelectron. Reliab., 53: 1189-1193. DOI: 10.1016/j.microrel.2013.06.014

Emmert, J.M., C.E. Stroud and M. Abramovici, 2007. Online fault tolerance for FPGA logic blocks. IEEE Trans. Very Large Scale Integr., 15: 216-226. DOI: 10.1109/TVLSI.2007.891102

Hatori, F., T. Sakurai, K. Nogami, K. Sawada andM. Takahashi *et al*., 1993. Introducing redundancy in field programmable gate arrays. Proceedings of the IEEE Custom Integrated Circuits Conference, May 9-12, IEEE Xplore Press, San Diego, CA., pp: 7.1.1-7.1.4. DOI: 10.1109/CICC.1993.590575

Krishna, B.H. and S. Ravi, 2012. A novel approach of reusing of faulty CLBS for reconfiguration. Proceedings of the International Conference on Emerging Trends in Electrical, Communications and Information Technologies, (CIT' 12), pp: 73-81.

Narasimhan, J., K. Nakajima, C.S. Rim and A.T. Dahbura, 1994. Yield enhancement of programmable ASIC arrays by reconfiguration of circuit placements. IEEE Trans. Comput. Aided Design Integr. Circ. Syst., 13: 976-986. DOI: 10.1109/43.298034

Narasimhan, J., K. Nakajima, C.S. Rim and A.T. Daubura, 1991. Yield enhancement of wafer scale integrated arrays. Proceedings of the 3rd International Conference on Wafer Scale Integration, Jan. 29-31, IEEE Xplore Press, San Francisco, CA., pp: 178-184. DOI: 10.1109/ICWSI.1991.151713

Raghuraman, K.P., H. Wang and S. Tragoudas, 2005. A novel approach to minimizing reconfiguration cost for LUT-based FPGAs. Proceedings of the 18th International Conference on VLSI Design, Jan. 3-7, IEEE Xplore Press, pp: 673-676. DOI: 10.1109/ICVD.2005.25

Sedcole, P., B. Blodget, T. Becker, J. Anderson and P. Lysaght *et al*., 2006. Modular dynamic reconfiguration in Virtex FPGAs. IEEE Proc. Comput. Digital Tech., 153: 157-164. DOI: 10.1049/ip-cdt:20050176