Self-Repairable Reconfigurable Circuit using Embedded Autonomously Restructuring Cores

Ms.Irala.Suneetha
Research Scholar

Dr.T.Venkateswarlu
Professor of ECE

Abstract
Field Programmable Gate Arrays (FPGA) are widely used to implement complex systems. It can be reprogrammed in a system and allows a system using reconfigurable hardware to adapt to changes in external environment. They are used to extend initial capabilities by implementing new functions on the same hardware platform. A FPGA typically consists of a number of configurable logic blocks (CLBs) arranged in rows and columns. Based on the application, some CLBs are activated and other CLBs are kept as spare resources. When any activated CLB becomes faulty, it is possible to repair permanent internal faults in FPGA by using back up circuits. However, this approach becomes complex when the size of the FPGA increases and optimal algorithms need to coexist to make it suitable for online use. In this work, an autonomous restructuring circuit is designed with an objective to reduce the latency, (i.e.) the speed with which the faulty CLBs can be identified and replaced is to be reduced. Also, when any fault occurs in the active CLB, the fault is rectified using the spare CLB in an autonomous way

Keywords:
Autonomous Recover(AR), Configuration Bits(CFB), Configuration Logic Block(CLB), Field Programmable Gate Array(FPGA), Look Up Table(LUT).

1. Introduction

The conventional repair strategies for repairing a FPGA includes Hierarchical model, Optimal model, Coarse redundancy model and Tile-based model. The major drawbacks of conventional methods are increased latency and the time between fault detection and repair is more [1]. For large circuits the test vectors to be generated by the Linear feedback shift register (LFSR) block is high and this imposes a serious limitation for on-line use. In this work, the above disadvantages are overcome by proposing a novel autonomous restructuring algorithm for an FPGA based filter application. The circuit considered has 217 configuration bits for a FPGA with 25 CLBs, 25 bits from the fault identification module and another 25 bits from decode evolved architecture [2]. These bits identify the next state and produce an architecture corresponding to the modified set of bits. Autonomous restructuring circuit is able to map the faulty CLB onto a spare CLB.

Manuscript received July 5, 2010
Manuscript revised July 20, 2010

2. Conventional Repair Strategies

2.1 Hierarchical Model

The Hierarchical model consists of two levels. At lower level, each tile consists of spare CLBs. At higher level, faulty tiles are replaced by spare tiles [3]. A main disadvantage of this method is that heterogeneous tiles must be joint or disjoint to form homogeneous tiles (increases latency [4]). This is shown in Fig. 1.

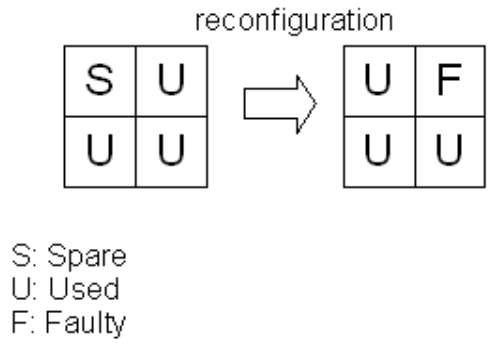

Fig. 1. Hierarchical model

2.2 Optimal Model

The optimal model represents the best possible case. The problems associated with rerouting are not taken into account. However, the drawback is in terms of both time to repair and size of precompiled bit stream. It is a time consuming method and most of the application cannot be done on-line. It must be executed at compile time and a more suitable method must be developed to reduce the size of precompiled bit stream.

2.3 Coarse Redundancy

In coarse redundancy model when a fault is detected in a column [5], the whole column is marked as faulty and it is replaced by a spare. But, in this approach, even the fault free CLB in the columns are marked as faulty. Hence, there is low resource utilization [6].

2.4 Tile based Model

In tile based model each tile has a spare that can repair only a single fault in a tile. Diagnosis must locate the faulty resource with a granularity better than the dimension of a tile such that the faulty resources can be replaced with the spares in the tile [7].

3. Autonomous Restructuring Model

The following steps are involved in autonomous restructuring model:

(i) Decoding the Configuration BITS
This involves both function decoding, input and output decoding (i.e. structural identification).

(ii) Spare CLB Selection
This involves selecting a spare CLB to replace the faulty one.

(iii) Updating Configuration bits

To achieve the above, four cores are used in this work. This is shown in Fig. 2. The cores are Decoding Core (DC), Fault Identification Core (FIC), Spare Selection Core (SSC) and Autonomous Restructuring Core (ARC).

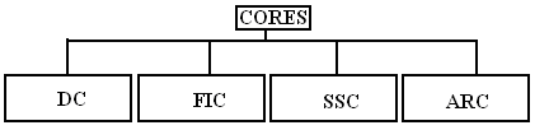

Fig. 2. Cores Redundancy types

The complete Autonomous restructuring model is illustrated in Fig. 3.

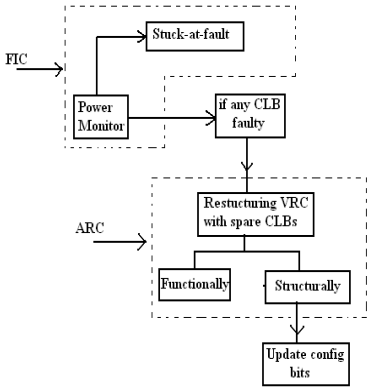

Fig. 3. Fault rectification using ARC

4. Reliability Metrics

4.1 Problem of Scaling

The major problem in the evolutionary design is the problem of scaling. As the search size becomes larger, the number of gates increases and the time to test the fitness of the circuit also grows non-linearly with the truth table.

4.2 Latency

The speed with which faulty CLBs are identified and replaced with spares must be high so as to reduce the latency.

4.3 Fault Coverage

A FPGA must have maximum fault coverage percentage. This refers to the number of active CLBs that can be replaced by spares. This ideal condition (100% fault coverage) is highly difficult to achieve.

5. Proposed work

In this work, when any fault occurs in the active CLB, the fault is replaced by the spare CLB in an autonomous way by using the architecture in Fig. 4. Accordingly the configuration bits are modified. Thus, it is enough for different architectures, just to alter the configuration bits stored in the SRAM. In this work, algorithms to monitor the status of the internal elements of the evolved circuit, and if found faulty, to restructure them with spare CLBs both functionally and structurally are presented.

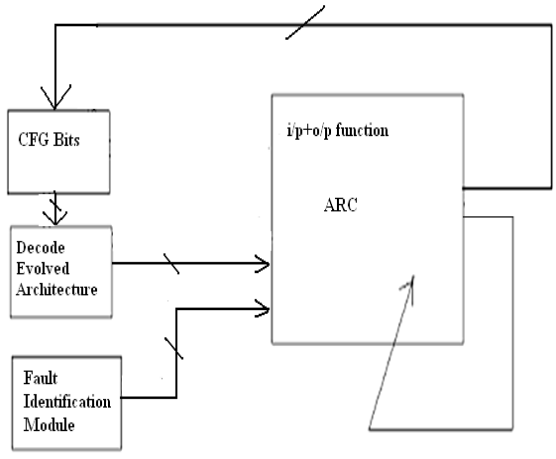

Fig. 4. Faulty CLBs

5.1 Objective

The main objectives of this research work are:

- To provide a repair model for FPGA if any fault is detected. Here, both the permanent and transient faults are detected by using different testing techniques [8] and the faulty CLBs are fixed by reprogramming.
- To develop high speed reliable circuits.
- To identify the evolved FPGA architecture, (i.e.) structural identification [9].
- To make use of spare resources to repair a fault.

6. Methodology

The inputs to the autonomous restructuring circuit are (i) the original configuration bits, (ii) the output of the virtual reconfiguration circuit (VRC) architecture decoder and (iii) the output of the fault identification module. The VRC architecture decoder identifies the spare and the active CLBs. The fault identification module uses a power monitoring circuit to identify the faulty CLB [10]. The fault decision core decides whether the configuration bits must be altered or not. When any fault is identified the configuration bits are altered autonomously. Otherwise, the existing configuration word is retained. The proposed model is given in Fig. 5. A sample application corresponding to a noise filter 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) [11]. The VRC decoder gives a 25 bit word as input to represent the active and spare CLBs among the 25 CLBS in the VRC.

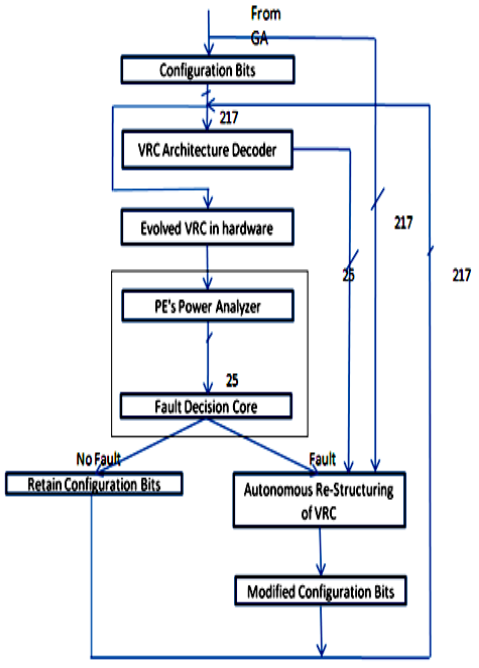

Fig. 5. VRC architecture decoder

7. Implementation of the Autonomous Self Repair Configurable Processor

A reconfigurable processor can be viewed as taking its inspiration from the paging mechanism in operating systems with the concept extended to hardware logic circuits. However, in the paging concept the code being swapped between RAM and disk usually represents a single thread of control, whereas in the hardware context the logic can be concurrent. Similarly, in software paging placing the code into RAM does not necessarily imply immediate execution, whilst hardware will potentially run immediately if it is loaded into the FPGA. This also requires that the new hardware page brought into the FPGA must be sure that its inputs are stable before its computation. To perform these hardware page swaps in and out of the FPGA the reconfigurable logic is extensively used [12]. The proposed implementation of the autonomous restructuring and self repair dynamic programming module of the reconfigurable processor is given in this section. It consists of the following modules: (1) spare CLB identification (2) functional identification (3) input decoding (4) structural identification (5) existing CLB identification and (6) modifying the configuration bits. These are described in the following sections:

7.1 Pseudo Code for Spare CLB identification

```
P=Existing CLB column head
Do
{
If (p not active)
        Allocate p as spare
        break;
Else
        Increment p
} while (p<next column head)
```

7.2 Pseudo Code for Autonomous Restructuring Circuit

```
X= CLB number   // Copy the existing CLB configuration bits and paste in spare CLB configuration bits
Read 'x'        //whose input is the output of CLB TO BE INCLUDED
while   // (till configuration word is formed)
{
If (x)
{
Get the starting position of 'x' //This represents the CLB TO BE INCLUDED output.
Replace it with spare CLB output.
Get the next CLB number x whose input is the output of CLB TO BE INCLUDED
```

}
Retain the other bits in their respective positions.
}

7.3 Steps Involved in Autonomous Restructuring

7.3.1 Functional Identification

This module identifies the function performed by each and every CLB in the FPGA. The different functions are represented by 'm' bits and hence total number of functions performed by each CLB is 2^m.when m=3, each of the CLB can be configured to operate with any of the 8 functions provided in LUT shown in Table I.

Table I

Function Number	Function Code	Function Type
F_0	000	x&y
F_1	001	$x^{!}y$
F_2	010	x^y
F_3	011	x+y
F_4	100	(x+y)+1
F_5	101	(x+y)>>1
F_6	110	X&0x0f
F_7	111	X&0xf0

7.3.2 Input-Output Decoding

This module identifies the input given to each and every CLB. This module also identifies the inputs to other stage CLBs.

7.3.3 Structural Identification

This module identifies the outputs of CLBs that are given as inputs to the succeeding CLBs. This structural identification helps in describing the entire architecture of the FPGA.

7.3.4 Existing CLB Identification

The existing CLB identification module is an 'n' bit register and loads a value of '0' corresponding to those CLBs that are already involved in the circuit creation. For the circuit evolved, the output of n=25 bit register is given in Fig. 6.where a bit '0' represents faultless CLB and a bit '1' represents spare CLB. These 25 bits are given as input to the autonomous restructuring unit for evolving the new circuit.

0000100001100010000100010

Fig. 6. Output of CLB identification register

7.3.5 Spare Selection

In Fig.7 there are certain spare resources (CLBs) which can be included along with the existing ones. Each spare CLB is represented by bit 0 and the active ones are represented by bit 1. For example in Fig. 7 the CLBs 3, 5, 10, 12, 18, 20, 23 whose outputs are not given as inputs to any succeeding columns of CLBs represent spare resources.

7.3.6 Modifying the Configuration Bits

The autonomous restructuring unit recovers the system from its faults by replacing the configuration bits of SRAM in the FPGA and introduces a new circuit. For example when CLB 10 needs to be introduced for a faulty CLB 9 to recover the system performance, the inputs and functions already evolved are mapped into CLB 10. The configuration bits are accordingly reconfigured autonomously.

8. Implementation Results

The result obtained by implementing the above autonomous restructuring algorithm is presented in this section. The results have been presented for both single CLB fault recovery and multiple CLB fault recovery.

Case (i): Single CLB fault recovery
The results obtained by applying the algorithms discussed in section 7 for single CLB fault is presented in this section. The CLB number 13 is assumed to be faulty and accordingly the nearest spare CLB is autonomously identified and the faultless VRC is evolved. Fig. 8 shows the output of the decoding logic in which the VRC architecture is decoded by using the 217 bits as input. This figure also shows the active and spare CLBs along with their connectivity and function performed. This module performs the functional decoding. In Fig.9 the structural decoded architecture output is shown. In Fig.10 the Fault Identification Register (FIR) gives an output of 13 which indicates that CLB13 is faulty. Accordingly the nearest spare CLB (i.e. CLB12) is autonomously restructured and the VRC is made self-repairable. This is illustrated in Fig. 10.

Case (ii): Multiple CLB fault recovery
In this case more than one active CLB is assumed to be at fault simultaneously and the proposed algorithms are tested for its effectiveness in evolving the new faultless VRC autonomously. The Fault Identification Register (FIR) gives multiple outputs 9 and 13 which indicate that both CLB9 and CLB13 are faulty. Accordingly the nearest two spare CLBs(i.e CLB10 and CLB12

respectively) are autonomously restructured and the VRC is made self-repairable. This is illustrated in Fig. 11.

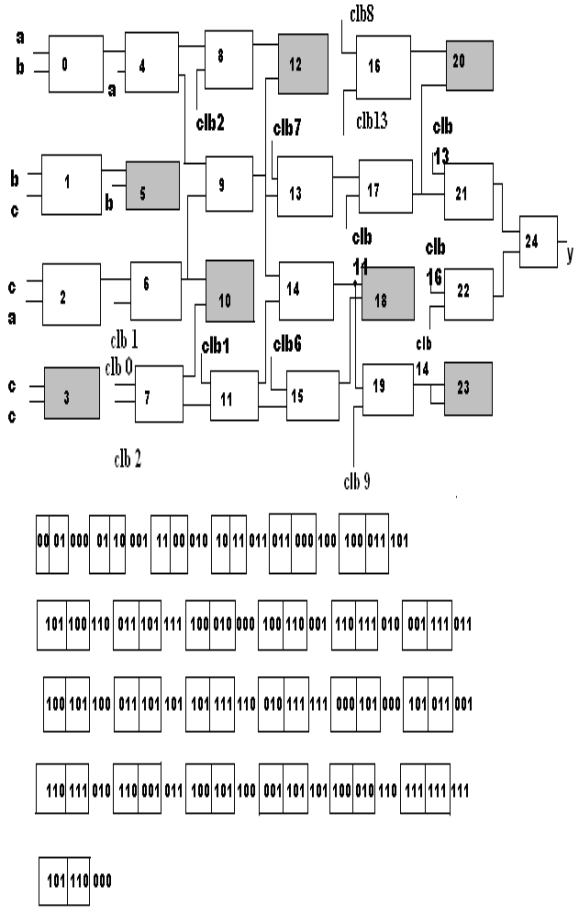

Fig. 7. Evolved FPGA architecture

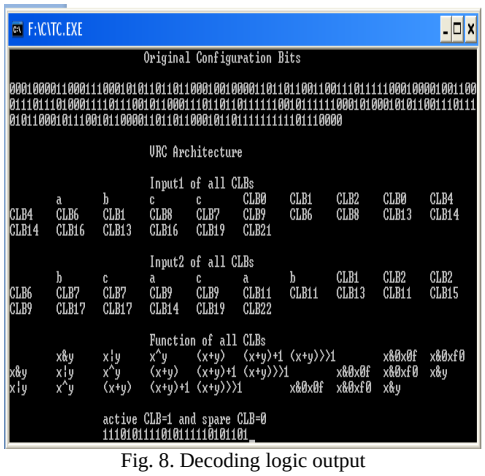

Fig. 8. Decoding logic output

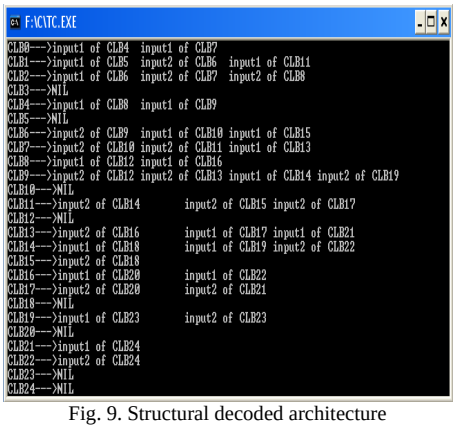

Fig. 9. Structural decoded architecture

Fig. 10. SSC and ARC output

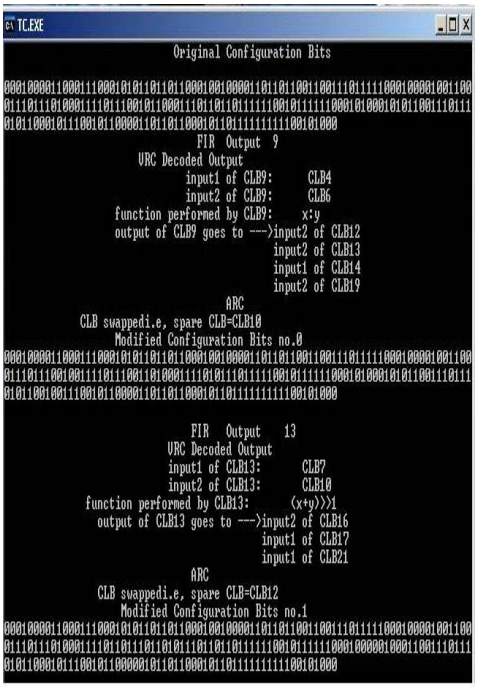

Fig. 11 Autonomously reconfigured VRC for multiple fault CLB scenario

9. Conclusion

In this work, the drawbacks of the conventional repair strategies in a FPGA are studied and an alternate autonomous restructuring model is proposed. This model reconfigures the FPGA when affected with any permanent or transient faults. In this work, decoding of configuration word and an evolved architecture, fault identification module and autonomous restructuring module are discussed. The proposed autonomous restructuring enables functional recovery for the devices after the occurrence of unavoidable damages and makes the circuit suitable for online. The work has been presented to take care of both single CLB fault and multiple CLB fault.

References

[1] Nathan R. Shnidman, William H. Mangione-Smith, and Miodrag Potkonjak, "On-Line Fault Detection for Bus-Based Field Programmable Gate Arrays", in IEEE Transactions on very large scale integration (VLSI) systems, vol. 6, no. 4, Dec. 1998.

[2] Stephen Brown, and Jonathan Rose, "FPGA and CPLD Architecture: A tutorial", in IEEE Design and Test of Computers, pp. 42-57, in 1996.

[3] S. J. Wang and T.-M. Tsai, "Test and diagnosis of faulty logic blocks in FPGAs", in IEE Proc.-Comput. Digit. Tech. Vol. 146, No. 2, March 1999.

[4] Jim Torresen, "An Evolvable Hard-ware Tutorial".

[5] Wei-Je Huang and Edward J.Mc Cluskey, "Column - Based Precompiled Config--uration Techniques for FPGA Fault Tolerance", Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, in 2001.

[6] S.Pontarelli, M.Ottavi, V.Vankamamidi, A.Salsano, F. Lombardi, "Reliability Evaluation of Repairable / Reconfigur-able FPGAs", Proceedings of the 21st IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, in 2006.

[7] Anna Antola, Vincenzo Piuri, Mariagiov--anna Sami, "On-line Diagnosis and [8] Wei configuration of FPGA Systems" Proceedings of the First IEEE International Workshop on Electronic Design, in –July, 02.

[8] Kang Huang, Fred J. Meyer, Xiao-Tao Chen and Fabrizio Lombardi, "Testing Configurable LUT-Based FPGA's", in IEEE Transactions on very large scale integration (VLSI) systems, vol. 6, no.2, June 1998.

[9] Jonathan Rose, Abbas El Gamal and Alberto, "Architecture of Field-Programmable Gate Arrays" Proceedings of the IEEE vol. 81. No 7. July 1993.

[10] S. Pontarelli, G.C. Cardarilli, A. Malvoni, M. Ottavi, M. Re, A. Salsano, "System-on-Chip Oriented Fault-Tolerant Sequential Systems Implementation Methodology", Proceedings of the 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, in 2001.

[11] Miron Abramovici, Charles E. Stroud, and John M. Emmert, "Online BIST and BIST-Based Diagnosis of FPGA Logic Blocks", in IEEE Transactions on very large scale integration (VLSI) systems, vol. 12, no. 12, Dec. 2004.

[12] John Lach, William H. Mangione-Smith, and Miodrag Potkonjak, "Enhanced FPGA Reliability Through Efficient Run-Time Fault Reconfiguration", in IEEE Transactions on reliability, vol. 49, no. 3, Sept. 2000.

Ms. I Suneetha received the B.Tech and M.Tech Degrees in E.C.E from S.V. University College of Engineering (SVUCE) Tirupati, India in 2000 and 2003 respectively. She is currently is pursuing Ph.D., Degree at SVUCE,Tirupati and working with E.C.E department,Annamacharya Institute of Technology and Sciences (AITS), Tirupati. Her research area includes 1D & 2D signal processing and FPGA.

Dr. T.Venkateswarlu received the B.Tech and M.Tech Degrees in E.C.E from S.V. University College of Engineering (SVUCE) Tirupati, India in 1979 and 1981 respectively.. He received the Ph.D degree in Electrical Engineering from Indian Institute of Technology (IIT), Madras in 1990. After working a short period at KSRM College of Engineering, KADAPA, he joined and currently working with the department of E.C.E, SVUCE, Tirupati. During 1986-89 he was a QIP research Scholar at the department of Electrical Engineering, Indian Institute of Technology (IIT), Madras. His teaching and research interest are in the areas of digital systems, communications and multidimensional digital filters.

Appendix-I
Input Output Decoding and Structure Identification

Inputs to CLBs	Binary code
a	00
b	01
c	1X

CLB No.	Input-1	Input-2	Function	Original CFB
CLB0	a	b	F_0	00 01 000
CLB1	b	c	F_1	01 10 001
CLB2	c	a	F_2	11 00 010
CLB3	c	c	F_3	10 11 011

Inputs to CLBs	Binary Code
a	000
b	001
c	010
CLB0 o/p	011
CLB1 o/p	100
CLB2 o/p	101
CLB3 o/p	11X

CLB No.	Input-1	Input-2	Function	Original CFB
CLB4	CLB0 o/p	a	F_4	011 000 100
CLB5	CLB1 o/p	b	F_5	100 001 101
CLB6	CLB2 o/p	CLB1 o/p	F_6	101 100 110
CLB7	CLB0 o/p	CLB2 o/p	F_7	011 101 111

Inputs to CLBs	Binary code
CLB0 o/p	000
CLB1 o/p	001
CLB2 o/p	010
CLB3 o/p	011
CLB4 o/p	100
CLB5 o/p	101
CLB6 o/p	110
CLB7 o/p	111

CLB No.	Input-1	Input-2	Function	Original CFB
CLB8	CLB4 o/p	CLB2 o/p	F_0	100 010 000
CLB9	CLB4 o/p	CLB6 o/p	F_1	100 110 001
CLB10	CLB6 o/p	CLB7 o/p	F_2	110 111 010
CLB11	CLB1 o/p	CLB7 o/p	F_3	0 01 111 011

Inputs to CLBs	Binary code
CLB4 o/p	000
CLB5 o/p	001
CLB6 o/p	010
CLB7 o/p	011
CLB8 o/p	100
CLB9 o/p	101
CLB10 o/p	110
CLB11 o/p	111

CLB No.	Input-1	Input-2	Function	Original CFB
CLB12	CLB8 o/p	CLB9 o/p	F_4	100 101 100
CLB13	CLB7 o/p	CLB9 o/p	F_5	011 101 101
CLB14	CLB9 o/p	CLB11o/p	F_6	101 111 110
CLB15	CLB6 o/p	CLB711o/p	F_7	01 0 111 111

Inputs toCLBs	Binary code
CLB8 o/p	000
CLB9 o/p	001
CLB10 o/p	010
CLB11 o/p	011
CLB12 o/p	100
CLB13 o/p	101
CLB14 o/p	110
CLB15 o/p	111

CLB No.	Input-1	Input-2	Function	Original CFB
CLB16	CLB8 o/p	CLB13 o/p	F_0	000 101 000
CLB17	CLB13 o/p	CLB11 o/p	F_1	101 011 001
CLB18	CLB14 o/p	CLB15 o/p	F_2	110 111 010
CLB19	CLB14 o/p	CLB9 o/p	F_3	110 001 011

Inputs to a set of(CLBs)	Binary code
CLB12 o/p	000
CLB13 o/p	001
CLB14 o/p	010
CLB15 o/p	011
CLB16 o/p	100
CLB17 o/p	101
CLB18 o/p	110
CLB19 o/p	111

CLB No.	Input-1	Input-2	Function	Original CFB
CLB20	CLB16 o/p	CLB17 o/p	F_4	100 101 100
CLB21	CLB13 o/p	CLB17 o/p	F_5	001 101 101
CLB22	CLB16 o/p	CLB14 o/p	F_6	100 010 110
CLB23	CLB19 o/p	CLB19 o/p	F_7	111 111 111

Inputs to a set of(CLBs)	Binary code
CLB16 o/p	000
CLB17 o/p	001
CLB18 o/p	010
CLB19 o/p	011
CLB20 o/p	100
CLB21 o/p	101
CLB22 o/p	110
CLB23 o/p	111

CLB No.	Input-1	Input-2	Function	Original CFB
CLB24	CLB21 o/p	CLB22 o/p	F_0	101 110 000

Apendix-2
Single CLB fault recovery

Inputs to CLBs	Binary code
a	00
b	01
c	1X

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB0	a	b	F_0	00 01 000
CLB1	b	c	F_1	01 10 001
CLB2	c	a	F_2	11 00 010
CLB3	c	c	F_3	10 11 011

Inputs to CLBs	Binary code
a	000
b	001
c	010
CLB0 o/p	011
CLB1 o/p	100
CLB2 o/p	101
CLB3 o/p	11X

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB4	CLB0 o/p	a	F_4	011 000 100
CLB5	CLB1 o/p	b	F_5	100 001 101
CLB6	CLB2 o/p	CLB1 o/p	F_6	101 100 110
CLB7	CLB0 o/p	CLB2 o/p	F_7	011 101 111

Inputs to CLBs	Binary code
CLB0 o/p	000
CLB1 o/p	001
CLB2 o/p	010
CLB3 o/p	011
CLB4 o/p	100
CLB5 o/p	101
CLB6 o/p	110
CLB7 o/p	111

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB8	CLB4 o/p	CLB2 o/p	F_0	100 010 000
CLB9	CLB4 o/p	CLB6 o/p	F_1	100 110 001
CLB10	CLB6 o/p	CLB7 o/p	F_2	110 111 010
CLB11	CLB1 o/p	CLB7 o/p	F_3	0 01 111 011

Inputs to a set of(CLBs)	Binary code
CLB4 o/p	000
CLB5 o/p	001
CLB6 o/p	010
CLB7 o/p	011
CLB8 o/p	100
CLB9 o/p	101
CLB10 o/p	110
CLB11 o/p	111

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB12	**CLB7 o/p**	**CLB9 o/p**	F_5	**011 101 101**
CLB13	**Faulty Configuration Logic block**			
CLB14	CLB9 o/p	CLB11o/p	F_6	101 111 110
CLB15	CLB6 o/p	CLB711o/p	F_7	010 111 111

Inputs to a set of(CLBs)	Binary code
CLB8 o/p	000
CLB9 o/p	001
CLB10 o/p	010
CLB11 o/p	011
CLB12 o/p	100
CLB13 o/p	101
CLB14 o/p	110
CLB15 o/p	111

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB16	CLB8 o/p	**CLB12 o/p**	F_0	000 **100** 000
CLB17	**CLB12 o/p**	CLB11 o/p	F_1	**100** 011 001
CLB18	CLB14 o/p	CLB15 o/p	F_2	110 111 010
CLB19	CLB14 o/p	CLB9 o/p	F_3	110 001 011

Inputs to a set of(CLBs)	Binary code
CLB12 o/p	000
CLB13 o/p	001
CLB14 o/p	010
CLB15 o/p	011
CLB16 o/p	100
CLB17 o/p	101
CLB18 o/p	110
CLB19 o/p	111

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB20	CLB16 o/p	CLB17 o/p	F_4	100 101 100
CLB21	**CLB12 o/p**	CLB17 o/p	F_5	**000** 101 101
CLB22	CLB16 o/p	CLB14 o/p	F_6	100 010 110
CLB23	CLB19 o/p	CLB19 o/p	F_7	111 111 111

Inputs to a set of(CLBs)	Binary code
CLB16 o/p	000
CLB17 o/p	001
CLB18 o/p	010
CLB19 o/p	011
CLB20 o/p	100
CLB21 o/p	101
CLB22 o/p	110
CLB23 o/p	111

CLB No.	nput-1	Input-2	Function	Modified CFB
CLB24	CLB21 o/p	CLB22 o/p	F_0	101 110 000

Apendix-3
Multiple CLB fault recovery

Inputs to CLBs	Binary code
a	00
b	01
c	1X

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB0	a	b	F_0	00 01 000
CLB1	b	c	F_1	01 10 001
CLB2	c	a	F_2	11 00 010
CLB3	c	c	F_3	10 11 011

Inputs to CLBs	Binary code
a	000
b	001
c	010
CLB0 o/p	011
CLB1 o/p	100
CLB2 o/p	101
CLB3 o/p	11X

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB4	CLB0 o/p	a	F_4	011 000 100
CLB5	CLB1 o/p	b	F_5	100 001 101
CLB6	CLB2 o/p	CLB1 o/p	F_6	101 100 110
CLB7	CLB0 o/p	CLB2 o/p	F_7	011 101 111

Inputs to CLBs	Binary code
CLB0 o/p	000
CLB1 o/p	001
CLB2 o/p	010
CLB3 o/p	011
CLB4 o/p	100
CLB5 o/p	101
CLB6 o/p	110
CLB7 o/p	111

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB8	CLB4 o/p	CLB2 o/p	F_0	100 010 000
CLB9	**Faulty Configuration Logic block**			
CLB10	**CLB4 o/p**	**CLB6 o/p**	F_1	**100 110 001**
CLB11	CLB1 o/p	CLB7 o/p	F_3	0 01 111 011

Inputs to a set of(CLBs)	Binary code
CLB4 o/p	000
CLB5 o/p	001
CLB6 o/p	010
CLB7 o/p	011
CLB8 o/p	100
CLB9 o/p	101
CLB10 o/p	110
CLB11 o/p	111

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB12	**CLB7 o/p**	**CLB10 o/p**	F_5	**011 101 101**
CLB13	**Faulty Configuration Logic block**			
CLB14	**CLB10 o/p**	CLB11o/p	F_6	**110** 111 110
CLB15	CLB6 o/p	CLB711o/p	F_7	01 0 111 111

Inputs to a set of(CLBs)	Binary code
CLB8 o/p	000
CLB9 o/p	001
CLB10 o/p	010
CLB11 o/p	011
CLB12 o/p	100
CLB13 o/p	101
CLB14 o/p	110
CLB15 o/p	111

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB16	CLB8 o/p	**CLB12 o/p**	F_0	000 **100** 000
CLB17	**CLB12 o/p**	CLB11 o/p	F_1	**100** 011 001
CLB18	CLB14 o/p	CLB15 o/p	F_2	110 111 010
CLB19	CLB14 o/p	**CLB10 o/p**	F_3	110 **010** 011

Inputs to a set of(CLBs)	Binary code
CLB12 o/p	000
CLB13 o/p	001
CLB14 o/p	010
CLB15 o/p	011
CLB16 o/p	100
CLB17 o/p	101
CLB18 o/p	110
CLB19 o/p	111

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB20	CLB16 o/p	CLB17 o/p	F_4	100 101 100
CLB21	**CLB12 o/p**	CLB17 o/p	F_5	**000** 101 101
CLB22	CLB16 o/p	CLB14 o/p	F_6	100 010 110
CLB23	CLB19 o/p	CLB19 o/p	F_7	111 111 111

Inputs to a set of(CLBs)	Binary code
CLB16 o/p	000
CLB17 o/p	001
CLB18 o/p	010
CLB19 o/p	011
CLB20 o/p	100
CLB21 o/p	101
CLB22 o/p	110
CLB23 o/p	111

CLB No.	Input-1	Input-2	Function	Modified CFB
CLB24	CLB21 o/p	CLB22 o/p	F_0	101 110 000