Built in Self-Test for 4 × 4 Signed and Unsigned Multipliers in FPGA

Shrikant Vaishnav
M.Tech Scholar, NIIST
Bhopal (M.P), India

Puran Gaur
HOD, NIIST
Bhopal (M.P), India

Braj Bihari Soni
Asst. Prof, NIIST
Bhopal (M.P), India

ASTRACT

Built in self-test (BIST) is a technique or a method which allow the circuit to test itself. BIST increases the controllability and observability of integrated circuit therefore it is easier to apply inputs and then detect faults from it [11]. BIST also decreases the time of testing integrated circuits & gives very high fault coverage. Therefore in many ways BIST help us in detecting fault in integrated circuits. This paper presents an efficient fault detection algorithm for 4 × 4 signed & unsigned multiplier in field programmable gate array (FPGA). These techniques were successfully applied on booth, braun & unsigned array multipliers.

Keywords

Built In Self-Test, Multiplier, FPGA, Test Pattern Generator, DSP, Output Response Analyzer, Booth, Braun, Unsigned Array, Verilog HDL, Altera.

1. INTRODUCTION

Today's digital systems are so complex therefore it is highly desirable to make a system that has the self-testing capability also the fast-rising cost of test equipment are one of the main reason to incorporate BIST in a circuit. This paper show how BIST helps us in finding faults in 4 × 4 signed & unsigned multipliers. For signed operations booth multiplier and for unsigned operations braun and unsigned array were used in this paper. BIST reduces the need of external test equipment's because the testing circuit is already a part of device under test (DUT) therefore there is no need to use external costly test equipment also it saves our precious time to setup a connection every time therefore in many ways BIST help us in detecting faults present in our device under test [10].

2. BIST ARCHITECTURE

Basically BIST system consists of an BIST controller, test pattern generator (TPG), output response analyzer (ORA) & the device under test (DUT).The basic principle of BIST technique is to generate test vectors automatically, then apply these vectors to the circuit under test (DUT) and at last compare the outputs with the known correct outputs. If an output matches then this means we have the non-faulty circuit and if not then this means we have the faulty circuit In this paper DUT is nothing but an multiplier present in FPGA. There are wide varieties of BIST system are present but it must be noted that different circuit require different type of BIST system that means BIST used for one system is not similar for the other system so we need different type of BIST for different type of circuits. BIST is used in aviation, medical, military, integrated circuit manufacturing and in various fields. Nowadays BIST are not only used to test digital circuit but also used to detect faults in mixed signal ICs. We must not use BIST for one-time testing of a particular circuits because it consumes unnecessary space on chip but for repeatedly testing BIST is the best choice. It must be noted that firstly we had designed multipliers using verilog HDL and then applied our test algorithms on it to verify the effectiveness of our algorithms in all types of multipliers. We have tested our algorithms in altera cyclone II FPGA and in the verilogger extreme simulator.

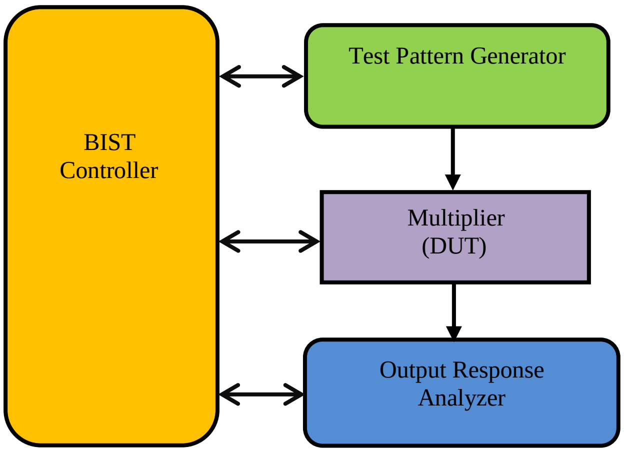


Fig.1 A BIST System

MAC unit calculate the product or output of two inputs and adds that output to an accumulator (register). MAC unit is the very important part of DSP system. If there is an error present in multiplier, then it will degrades the overall performance of DSP therefore we add a BIST system that tests the multiplier and detect fault if present. Performance of system is totally dependent on the performance of MAC unit that's why MAC unit must be work well for optimum results.

3. MULTIPLIER OVERVIEW

An multiplier is a device that accepts two inputs in digital form (in 1s and 0s) and gives their output (product) in the same digital form that is in 0s and 1s.

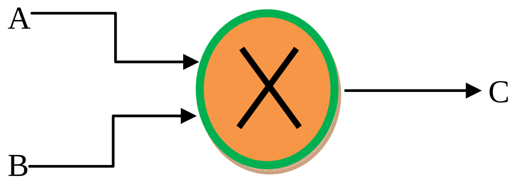


Fig.2 Multiplier

Multiplication is used in signal processing. Most modern DSP chips have dedicated multiplication hardware to maximize the performance. Thus fast multiply-accumulate operations are performed by the help of MAC unit. If MAC (multiply and accumulate) unit are not in DSP then we have only one option that is to combine multiplier, adder & accumulator (memory) to make an MAC unit for the desired DSP applications. But if fault generates in DSP applications then it is difficult to locate for this we have invented an efficient fault detection algorithms that detects fault present in multiplier and tells us if multiplier is faulty. Multiplication is also used in process like fourier transforms, IIR filtering & in Convolution.

4. MULTIPLIERS USED

(1)Unsigned Multipliers-If a multiplier take unsigned inputs and gives unsigned output then these types of multipliers are called unsigned multipliers. There are various types of unsigned multipliers are available for multiplication we select braun and unsigned array multipliers for this paper. Both of them are the types of array multiplier. Array multipliers calculate partial product parallelly and each of them are independent to each other.

(a)Braun Multiplier- Braun Multiplier in one of the array multiplier used for unsigned multiplication only. In this paper we had design 4 × 4 braun multiplier.

(b)Unsigned Array Multiplier- Unsigned array multiplier is also one of the types of array multiplier used for unsigned multiplication. In this paper we had design 4 × 4 unsigned array multiplier.

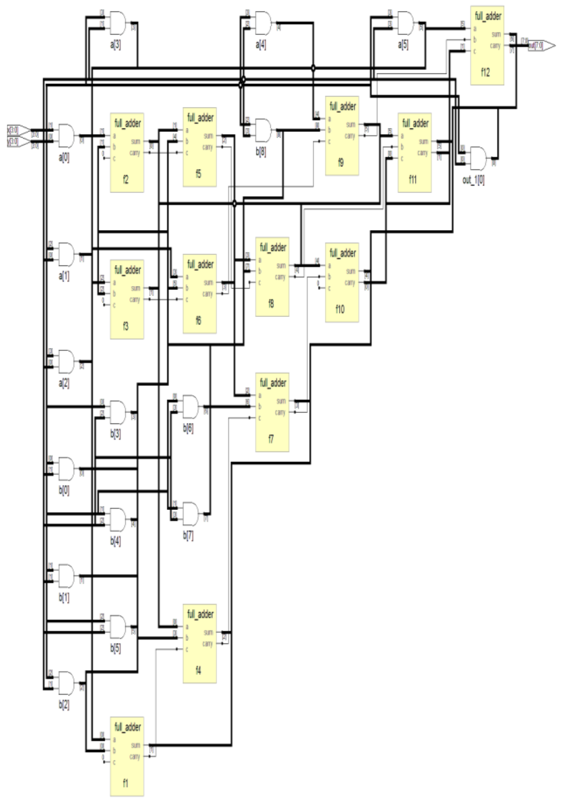


Fig.3 RTL view of Braun Multiplier

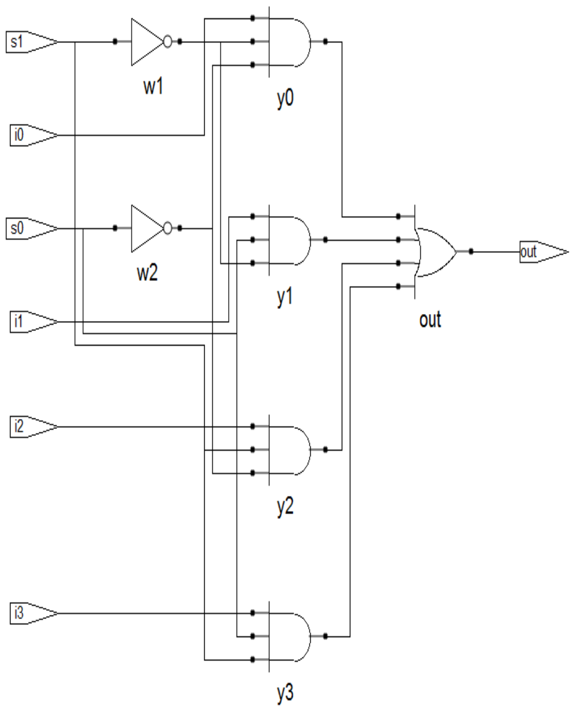


Fig.4 RTL view of Unsigned Array Multiplier

(2)Signed Multipliers- If a multiplier take signed inputs and gives signed outputs then this types of multipliers are called signed multipliers. There are various types of signed multipliers are available for multiplication we select booth multiplier for this paper. It is able to perform both signed as well as unsigned multiplication.

(a)Booth Multiplier Booth Multiplier is able to perform both signed as well as unsigned multiplication. In this paper we had design 4 × 4 booth multiplier.

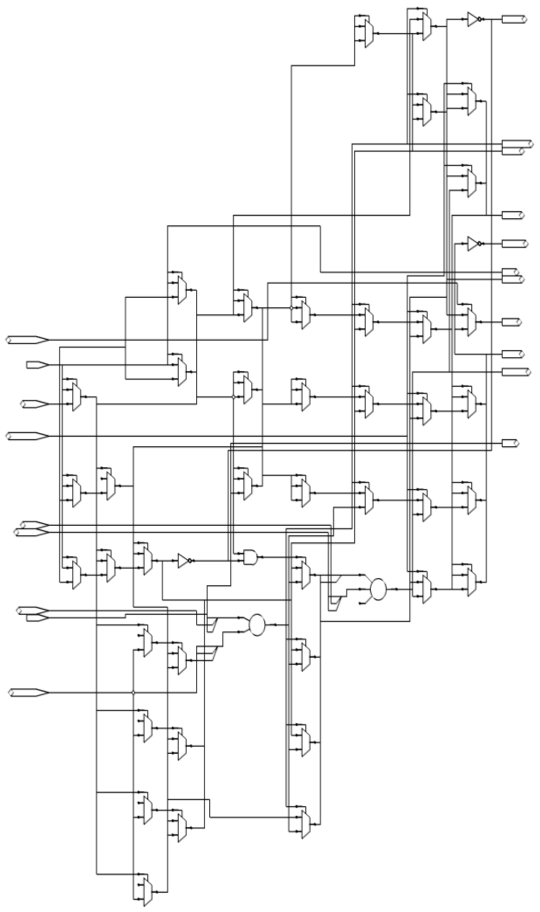


Fig.5 RTL view of Booth Multiplier

The RTL (Register-Transfer Level) view comes after the synthesis process of multipliers used in this paper are given in figure 1.3, 1.4 & 1.5.We had used "Synopsys Synplify" tool for FPGA synthesis.

All these multiplier is widely used for multiplication in digital world. It must be noted that although there are various variants of booth multiplier are available now each of them offers some advantages but we had selected the original booth multiplier for this paper. We had also selected the original braun and the unsigned array multiplier for this paper.

5. PROPOSED ALGORITHM

We have developed the architecture independent multiplier fault detection algorithm that means it's not depends on architecture and therefore whether it is signed or unsigned multiplier we only need this fault detection algorithm to test it. We have tested each algorithm in braun, booth and unsigned array multiplier. We are using the deterministic as well as the exhaustive testing approach. There are various types of testing approaches are present for testing. In deterministic testing approach all the test patterns (Inputs) and their outputs are stored in memory and this outputs is compared with the outputs of output response analyzer (ORA) unit if it is match then this mean our multiplier is non-faulty and if not then this means our multiplier is faulty [11].

Type	Multiplier	Algorithm
Unsigned	Braun Multiplier	2 × 4 Algorithm 4 × 2 Algorithm 4 × 4 Algorithm
Unsigned	Unsigned Array Multiplier	2 × 4 Algorithm 4 × 2 Algorithm 4 × 4 Algorithm
Signed/Unsigned	Booth Multiplier	2 × 4 Algorithm 4 × 2 Algorithm 4 × 4 Algorithm

Table.1 Multipliers & Algorithms

While in exhaustive testing approach all of the inputs are applied to the multipliers (DUT) and then compared its outputs (responses) with the outputs of ORA if it matches then this means our multiplier is non-faulty and if not then this means multiplier is faulty. We are using both of the testing approaches in this paper.

(i) 2 × 4 Algorithm- In 2 × 4 algorithm the 2 LSBs of the 4 bit input "a" is multiplied with the all 4 bits of input "b" that means it gives total 64 outputs(Test vectors).It must be noted that the remaining two MSBs of a is remain zero. In this way we minimize the number of test vectors. We had successfully applied this algorithm to braun, booth & unsigned array multipliers. In this algorithm we are using only the deterministic testing approach.

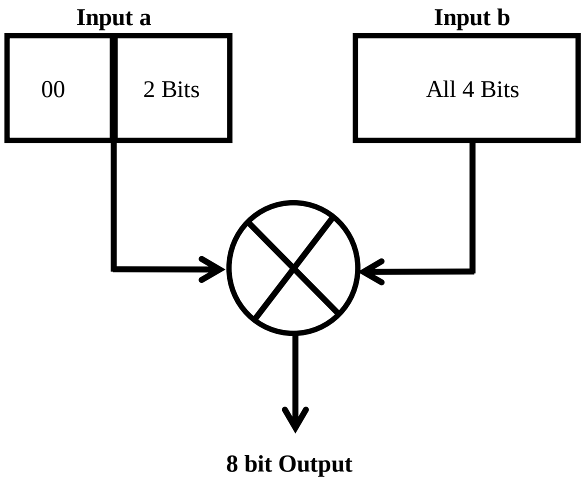


Fig.6 2 × 4 Algorithm

(ii) 4 × 2 Algorithm- In 4 × 2 algorithm all the 4 bits of input "a" is multiplied with the 2 LSBs of 4 bit input "b" that means it gives total 64 outputs(Test vectors). It must be noted that the remaining two MSBs of b is remain zero. In this way we minimize the number of test vectors. We had successfully applied this algorithm to braun, booth & unsigned array multipliers. In this algorithm we are using only the deterministic testing approach.

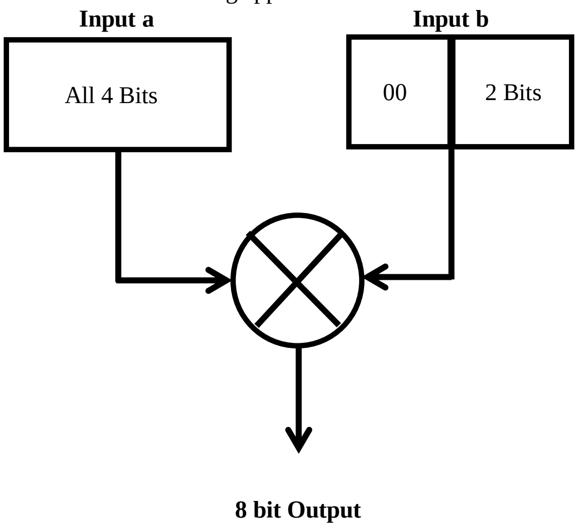


Fig.7 4× 2 Algorithm

(iii) 4 × 4 Algorithm-In 4× 4 Algorithm all the 4 bits of input “a” is multiplied with the all 4 bits of input “b” that means it gives total 256 outputs (Test vectors). It must be noted that in this algorithm all for bits of both the inputs a and b are used that means we were not keeping any bits to zeros like in 2 × 4 and 4 × 2 algorithm. Although the number of test vectors are increases but it is more reliable than other because it gives large numbers of test vectors to analyze. We had successfully applied this algorithm to braun, booth & unsigned array multipliers. In this algorithm we are using both the deterministic as well as exhaustive testing approach.

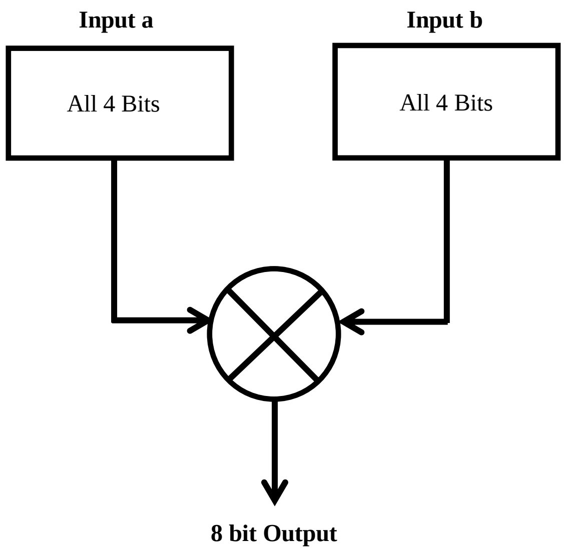


Fig.8 4× 4 Algorithm

It must be noted that we had not used any embedded multipliers of any FPGAs like previous authors did in their paper. We had selected the different multiplier then that of embedded multiplier. We have designed all the three multipliers using “Verilog HDL” and then check its behavior and at last we have applied our test algorithms in all of the three multipliers. It must be noted that before synthesis we have to compare the simulation results with our desired results if it is matches then we go for synthesis process otherwise not because simulator tell us initial design error that may generate later after synthesis therefore it is always better to do synthesis after simulation process.

Fault detection algorithms are capable of detecting faults when faults occur and after that it identify and then isolate it from the main system. In this way it protects system from any undesired conditions. Also in this way we are now able to make a more reliable system that was not practically possible before and also it decreases the testing cost which is so high. That’s why even BIST requires additional circuitry it is now one of the widely used method for testing.

6. SIMULATION RESULTS

To verify our algorithms we had firstly designed braun, booth & unsigned array multipliers in Verilog HDL and

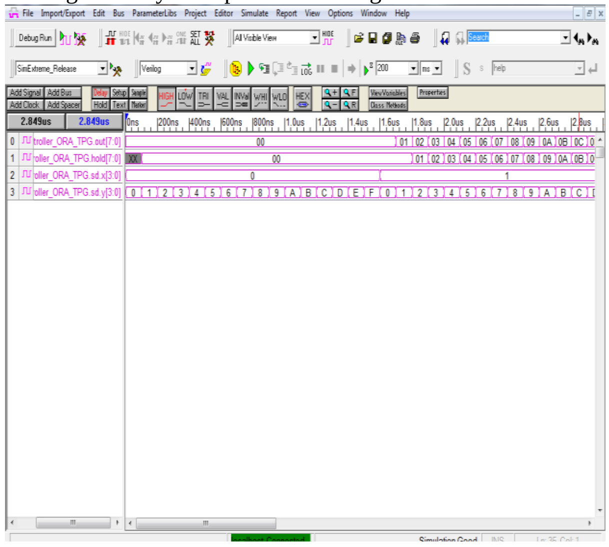


Fig.9 2× 4 Algorithm applied on braun multipliers

then applied 2 × 4, 4 × 2 & 4 × 4 one by one in each of the multiplier that means for each multiplier we had used total 3 algorithms that means total 9 simulator outputs must come but due to limitations of number of pages we only shows 3 outputs of each of the multiplier using 2 × 4, 4 × 2 & 4 × 4 respectively & 2 outputs showing faulty and non-faulty before and after injecting fault in braun multiplier. We had used synapticad’s verilogger extreme simulator for simulation its outputs are given below

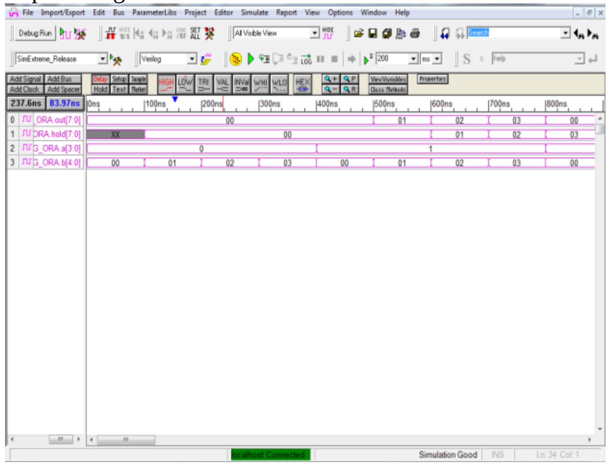


Fig.10 4× 2 Algorithm applied on unsigned array multipliers

It must be noted that the first three figures from figure 1.9 to 2.1 also tell us if fault present but here we have to check it manually by carefully comparing the outputs of ORA and the outputs of the multipliers.

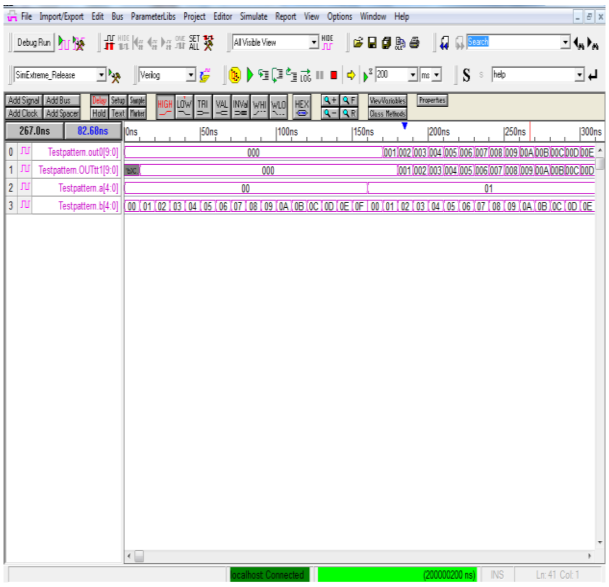
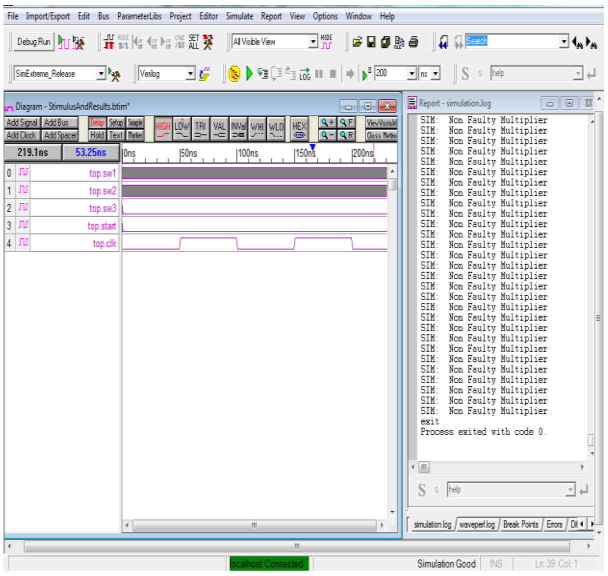


Fig.11 4× 4 Algorithm applied on booth multipliers

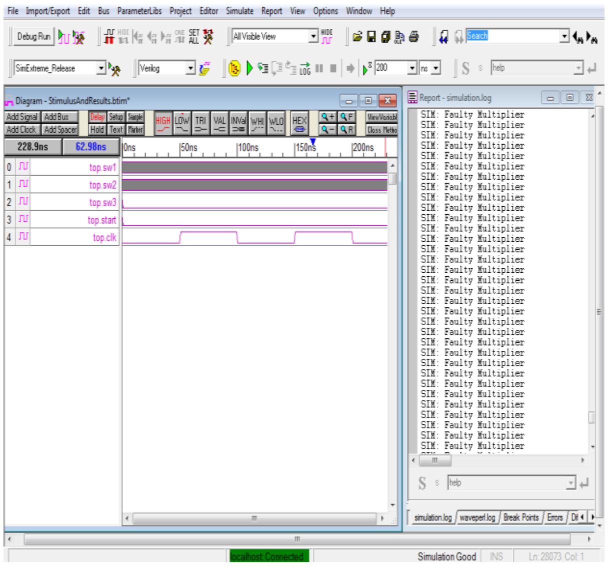
The last two figure 2.2 and 2.3 are differ then the previous 3 figure in the sense that it detect faults immediately and display it and there is no need to check or examine it manually like in previous figures. Thus it automatically performs all calculation and then generates results from it.



```
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
SIM:  Non Faulty Multiplier
exit
Process exited with code 0.
```

Fig.12 Result when no fault presents in 4 × 4 algorithm

Figure (2.2) tell us that multiplier is non-faulty. It is clearly shown in Verilogger Extreame's simulator window.



```
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
SIM: Faulty Multiplier
```

Fig.13 Result when fault present in 4 × 4 Algorithm

Figure (2.3) tell us that multiplier is faulty. It is clearly shown in Verilogger Extreame's simulator window. It must be noted that we not only simulate our design but also synthesize and run it to FPGA (Altera's Cyclone II) to show its effectiveness.

7. RESULTS ON FPGA KIT

We had already test & implement these algorithms in Altera Cyclone II FPGA (EP2C8Q208C8N).

Welcome screen tell here that we have to select 1, 2 or 3.It must be noted that here 1 means braun multiplier, 2 means unsigned array and 3 means booth multiplier. After selecting braun multiplier and then selecting start input switch it start checking multiplier.

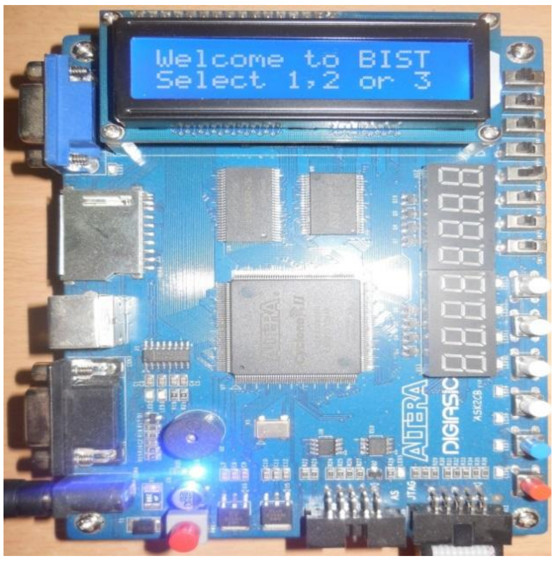


Fig.14 Welcome Screen of FPGA Kit

Whether multiplier is faulty / non faulty our fault detection algorithms easily tell this

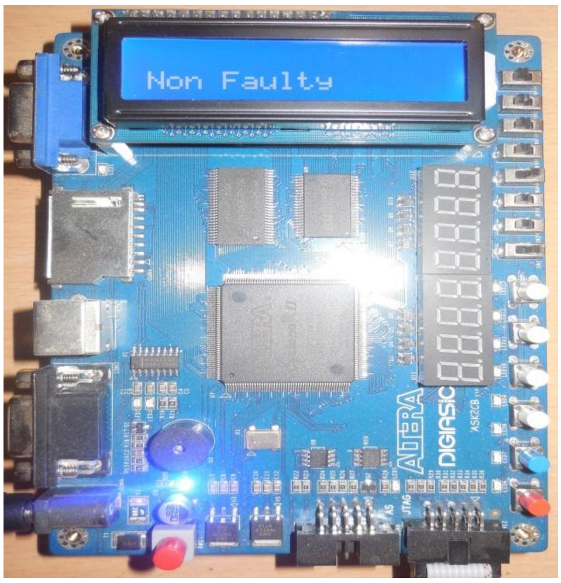


Fig.15 Result when no faults present

The picture shown in figure 2.5 tells us that multiplier is non-faulty.

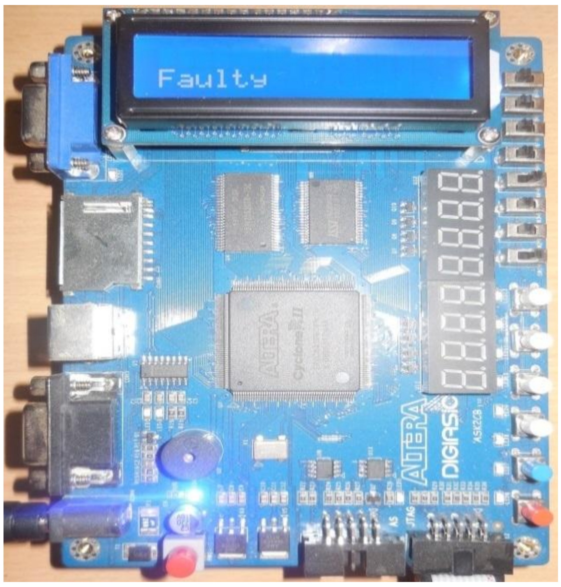


Fig.16 Result when fault present

The picture shown in figure 2.6 tell us that multiplier is faulty. We had designed an interactive graphical user interface to select multiplier from booth, braun & unsigned array multipliers. Firstly it will give you the "Welcome Screen" from which we have to select one multiplier by selecting appropriate switch. After selecting one multiplier and then select start switch to start checking / testing process it again now ready to test another multiplier so after this again if we want to test another multiplier then we have to select desired multiplier for testing. At last it clearly gives message "Non Faulty" if multiplier is non-faulty and gives message "Faulty" if multiplier is faulty. If user want then they also able to test the desired multiplier manually by selecting 4 bits of one input and then 4 bits of second input. It must be noted that here the output is shown in 8 leds means whenever you select both two input it gives you result / output at the same time in 8 leds.

8. CONCLUSION & FUTURE SCOPE

We can conclude from this paper that the 2×4, 4×2 & 4×4 test algorithms are very effective to test 4×4 signed & unsigned multipliers it can easily tell us that the multiplier is faulty or not. In future we will like to test these algorithms in wallace tree and baugh wooley multipliers.

9. REFERENCES

[1] Voyiatzis, C. Efstathiou, H. Antonopoulou ,A. Milidonis,"Arithmetic module-based built in self-test architecture for two-pattern testing "ISSN 1751-8601,IET,2012

[2] Micheal A . Lusco , Justin l.Dailey & Charles E.Stroud," Built in Self-Test for Multiplier I Altera Cyclone II FPGA" ,ISSN:0094-2898,IEEE,2011

[3] Jamuna.s and VK Agrawal, "Vhdl Implementation of BIST Controller",ISSN 1751-8601, IET,2011

[4] Mohsen Amiri Farahani,Sasid Mirzaei,Hossein Amiri Farahani, "Implementation of a Reconfigurable Architecture of Discrete Wavelet Transform With Three Types of Multipliers on FPGA", ISSN:0840-7789, IEEE,2011

[5] Mary D. Pulukuri, George J. Starr, and Charles E. Stroud, "On Built-In Self-Test For Multipliers" ISBN: 978-1-4244-5854-7,IEEE,2010

[6] Mary D. Pulukuri, George J. Starr, and Charles E. Stroud, "Built-In Self-Test of Digital Signal Processors in Virtex-4 FPGAs" ISBN: 978-1-4244-3325-4,IEEE,2009

[7] Xu, Q., Nicolici, N "DFT infrastructure for broadside two-pattern test of core-based SOC"', IEEE Trans., 2006, 55, (4),pp. 470–485

[8] Bhunia, S., Mahmoodi, H., Raychowdhury, A., Roy, K "Arbitrary two pattern delay testing using a low-overhead supply gating technique",J. Electron. Test., Theory Appl. Arch., 2008, 24, (6), pp. 577–590

[9] Samir Palnitkar "Verilog HDL: A Guide to Digital Design & Synthesis", ISBN: 978-81-775-8918-4

[10] Charles E. Stroud "A Designer's Guide to Built-in Self -Test", ISBN: 1-4020-7050-0

[11] Parag K. Lala "An Introduction to Logic Circuit Testing", ISBN:978-15-982-9350-0