IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 4, Issue 5, Ver. II (Sep-Oct. 2014), PP 14-20
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Designing of BIST Architecture of Generic Multipliers

Anju Rajput
(VLSI Design, JECRC University, India)

Abstract: *The aim of this paper is to present the BIST architecture of generic multipliers. The generic architecture of all the four multipliers i.e. array, column bypass, wallace tree and booth multiplier has been designed .The verification of all these four different multipliers has been done by using BIST controller.The LFSR (linear feedback shift register) has been designed, which will generate the automatic inputs.Then the BIST controller has been designed which will check these multipliers by using input patterns obtained from LFSR. BIST controller will verify these four generic architectures whether they are faulty or not By using it, the logics or design of these multipliers need not to be verified by any other means.This is the advantage of this design that it does not require third party verification.The simulation has been carried out on ModelSim (Student Editon) EDA tool 10.0c and synthesis has been carried out on ISE Design Suite 14.4.*

Keywords: *Array Multiplier, BIST, Booth Multiplier, Column Bypass Multiplier, Wallace tree Multiplier*

I. Introduction

The one of the key element of microprocessor and digital signal processing are multipliers. They are one of the unit in ALUs. There are several types of Multipliers. They can be differentiate on the basis of algorithms. Multipliers are also the major source of power dissipation. Using different algorithms, power dissipation can be reduced. Multiplication is a process of adding an integer to itself by a specified number of times. Earlier many researchers have developed different multipliers like 4-bit, 8-bit, and 16-bit. These designs cannot do their verification by itself.

The purpose of this paper is to present the generic architecture of array, booth, column bypass and Wallace tree multiplier. These multipliers can do n-bit multiplication. In multiplication there are two things, multiplicand and multiplier. Multiplicand get added to itself by number of times as specified by Multiplier to give the result which is known as "product". The Multiplicand is multiplied by each digit of Multiplier which starts with right hand side. These intermediate results are known as partial products. Partial products are simply the array of partial AND gates. The nxn multiplier require n^2 AND gates Then these partial products are added using adder to give the result of multiplication. Area and speed are the important design criteria. By using these constraints as design criteria suitable multiplier can be choose, depend upon the applications. This paper also present the BIST (built in self-test) architecture of these generic multipliers, which will do its on-chip verification. It means this design need not to be verified by third party. The test pattern are generated automatically by using LFSR (linear feedback shift register).

II. Proposed Design

The BIST architecture of generic multipliers i.e. array, booth, column bypass and wallace tree multiplier is proposed. In this proposed model, we are creating the generic architecture of these four multipliers, so that they can do n-bit multiplication. Then we will design the BIST (built in self-test) architecture of these generic multipliers. The BIST architecture then do the verification of these generic designs by using automatic test pattern generator. For this purpose LFSR (linear feedback shift register) technique is used, which will generate the sequence of test pattern randomly. The four different modes are used 00, 01, 10, and 11 for selection of multipliers. On the basis of these modes we can choose different multipliers for verification purpose. This design can further be implemented on FPGA. The flow diagram of complete design is shown in Fig.1 below-

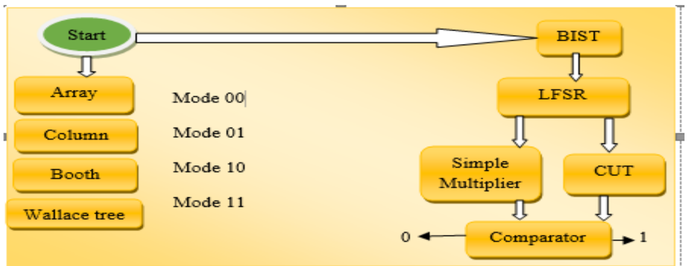


Fig.1: Proposed Design Flow

The above flow diagram represent the steps carried out in design process. Firstly we design the generic multipliers, then their BIST architecture get designed. Then the results from the BIST and generic multipliers will get compared by using comparator. Here CUT represent circuit under test i.e. generic multipliers. By using Modes we can choose the desired multiplier as CUT. Then Comparator will give result either 0 (when the design is faulty) or 1 (when the design is true).

III. Multipliers

a) Array Multiplier: It is based on shift and add algorithm. It is used for small circuits. It require less hardware but takes more time. Each digit of multiplier is multiplied with the multiplicand. We write the result which is called the partial products. Then we take the second digit of the multiplier and multiply with the same multiplicand, write it down from the first partial product, but we do the shifting. We will continue to do that until we exhaust all the digit of the multiplier. Finally we add the whole thing, to get the fellow product. This algorithm is known as shift /add algorithm, because we have to add it after shifting. . All the partial products are added to get the fellow product, but before adding we have to do shifting of the partial product to one bit position in left as shown in fig.2 below.

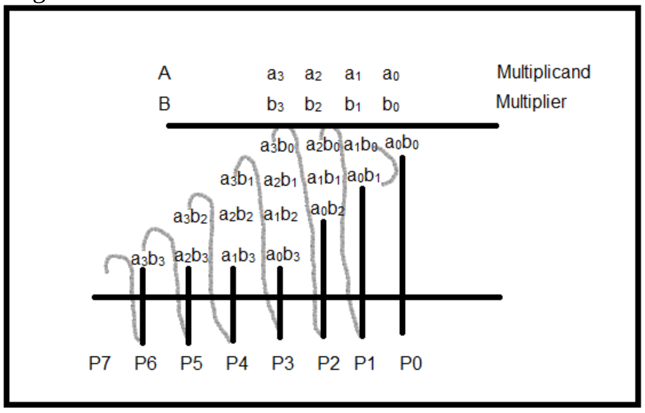

A
a_3 a_2 a_1 a_0
Multiplicand
B
b_3 b_2 b_1 b_0
Multiplier
a_3b_0 a_2b_0 a_1b_0 a_0b_0
a_3b_1 a_2b_1 a_1b_1 a_0b_1
a_3b_2 a_2b_2 a_1b_2 a_0b_2
a_3b_3 a_2b_3 a_1b_3 a_0b_3
P7 P6 P5 P4 P3 P2 P1 P0

Fig. 2 Basic idea of Array Multiplier

b) Column Bypass Multiplier: Column Bypassing in case of multiplier means turning off some columns in the multiplier array in case when certain multiplicand bits are zero. In this technique, during working, the operations in a column can be disabled if the corresponding bit in the multiplicand is 0, to save the power. The modified cell of adder circuit is shown in fig.3 below. In fig.3, there are two tristate buffers, a and b are the inputs, S(pre) is the previous sum and Cin is the previous carry. When a=0, multiplexer's select line is 0, then b will not get propagated and S(pre) will be the output. When a=1, select line is 1 then a, b and S(pre) will get added by the adder and Sout is the output[1].

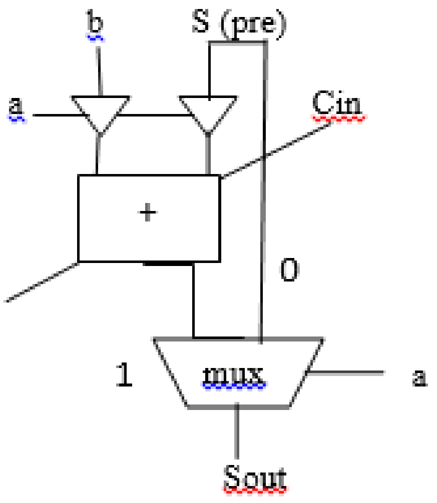


Fig.3 Modified cell of Adder circuit

c) Booth Multiplier: To enhance the addition among the partial products, requirement of fast adder architectures are required. The Modified- Booth algorithm is mostly used for high speed multiplier circuits. By decreasing the number of generated partial products, we can improve the multiplier performance.For the signed numbers multiplication booth is a powerful algorithm. It treats both signed and unsigned numbers. Booth algorithm is a technique which will reduce number of multiplicand multiples. The booth multiplication is shown in fig.4 below [2].

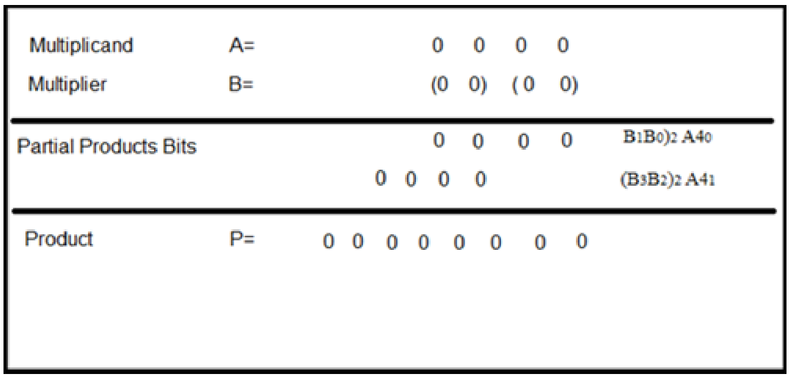
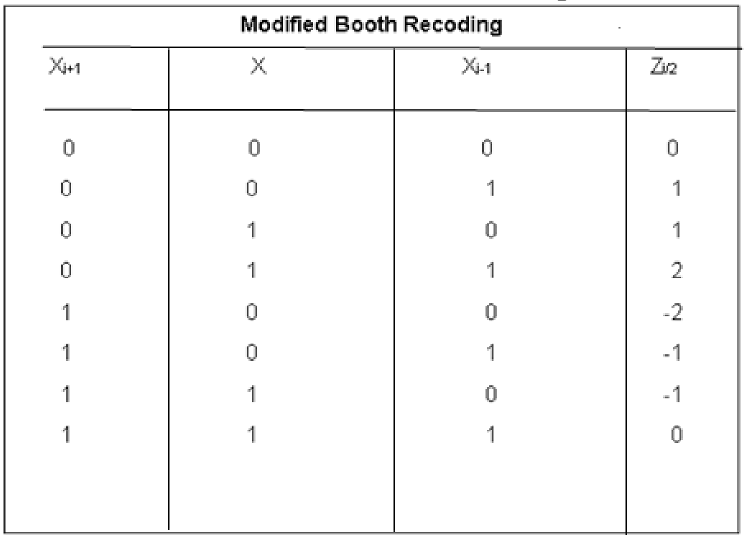

Multiplicand A= 0 0 0 0
Multiplier B= (0 0) (0 0)
Partial Products Bits 0 0 0 0 $B_1B_0)_2 A_{40}$
0 0 0 0 $(B_3B_2)_2 A_{41}$
Product P= 0 0 0 0 0 0 0 0

Fig.4 Booth Multiplication

Booth algorithm scans the three bits at a time to reduce the number of partial products.

Table 1: Modified booth Recoding



Modified Booth Recoding			
X_{i+1}	X	X_{i-1}	$Z_{i/2}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

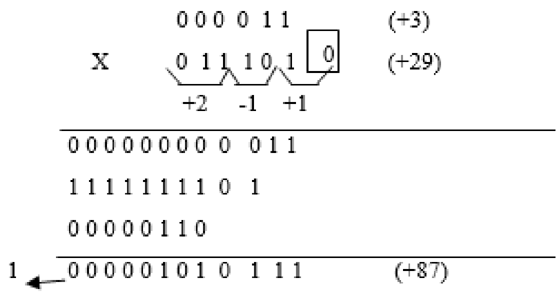


Fig.5 Example of Booth Multiplication

Modified Booth algorithm table 1 is shown above [3], on the basis of which binary number can be converted to radix-4 number [4]. Here is the example in fig.5, in which both multiplicand and multiplier is positive.

d) Wallace Tree Multiplier: For fast multiplication of two numbers, Wallace tree multiplier is used. It is faster than simple array multiplier. It has logarithmic height. But Wallace tree has irregular wiring. Due to this reason, it is often avoided by designer .Wallace tree use log-depth tree network for reduction [5]. This multiplier is not used for low power applications, because excess wiring consume extra power. But it is faster as it uses carry save adder. It is high speed multiplier. Basic idea of Wallace tree is shown below in fig.6 [6].

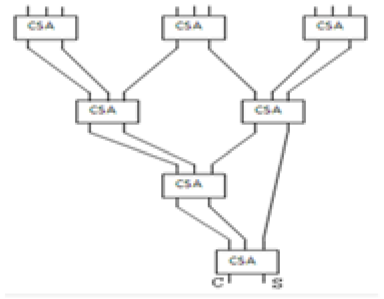


Fig.6 Wallace Tree

IV. BIST

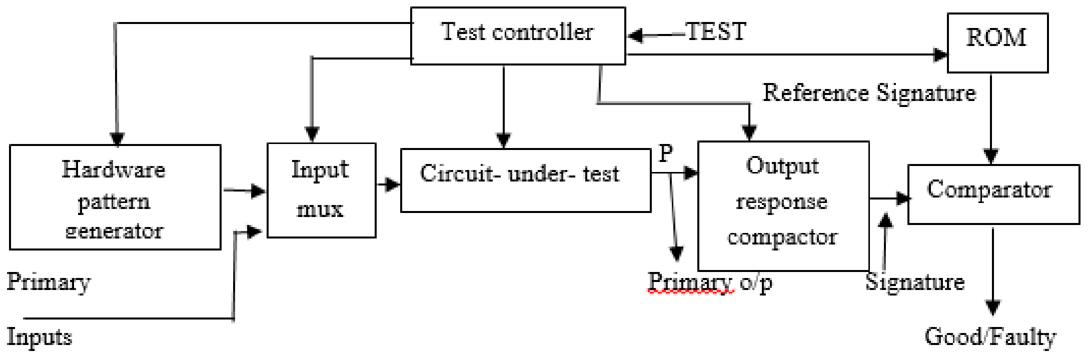


Fig.7 Block Diagram of BIST Architecture

Fig 7 shows the architecture of BIST. In built- in self-test (BIST) design, parts of the circuit are used to test the circuit itself. On-line BIST is used to perform the test under normal operation, whereas off-line BIST is used to perform the test off-line. In this paper, BIST methodology will be used for verification of these multipliers.

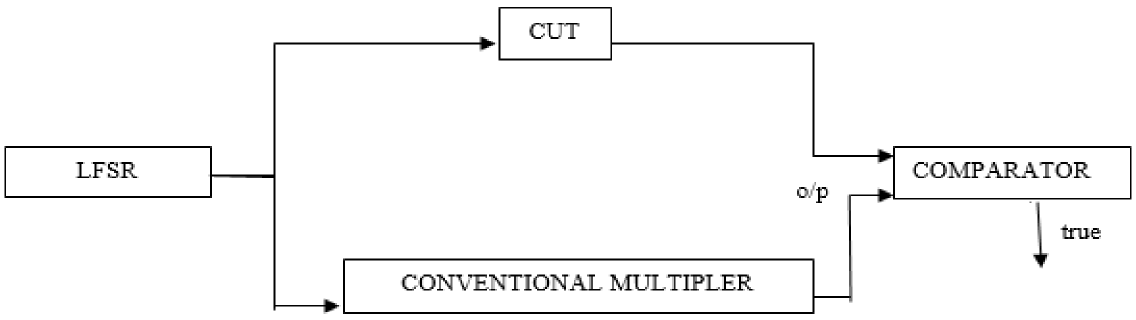


Fig.8 BIST Methodology

As shown in fig.8, the LFSR will generate automatic test pattern, which will goes to the CUT (circuit under test). CUT will be the four multipliers that we are used. Then output from conventional multiplier and CUT will be get compared by using comparator. Comparator result shows whether the CUT is faulty or not. The signature (the output from the conventional multiplier), is then compared with the expected signature (output from the CUTs), to determine whether the device under test is faulty. An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit .Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. A linear feedback shift register can be formed by performing exclusive-OR on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops.

➢ LFSR Technique: An n-bit LFSR ia a n- bit length shift register with feedback to its input. The feedback is formed by XORing the outputs of selected stages of the shift register- referred to as ‘taps’ and then inputting this to the least significant bit (stage0). Each stage has a common clock. The ‘linear’ part of the term ‘LFSR’ derives from the fact that XOR and XNOR are linear function. An example of a 5-bit LFSR is shown below in fig.9

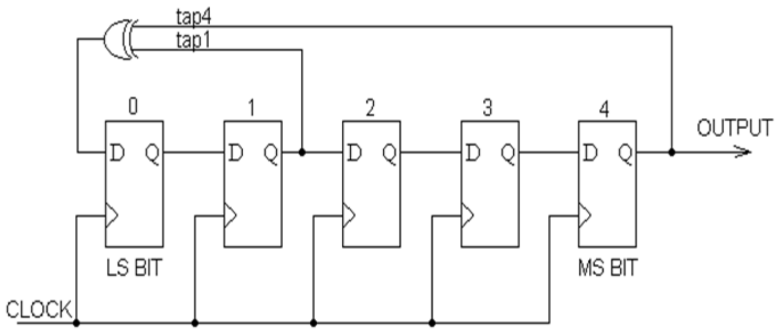
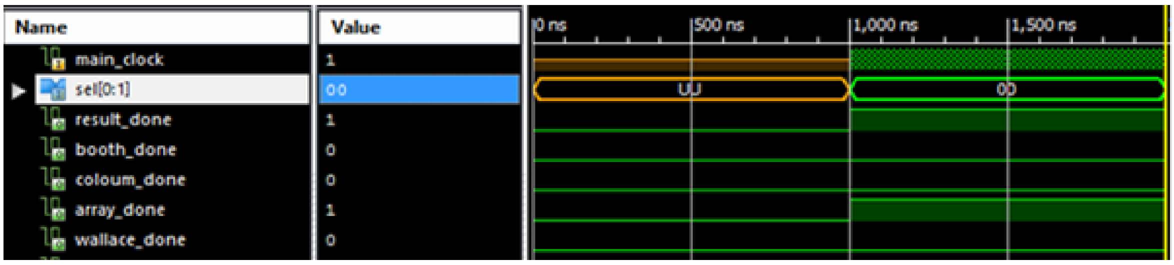


Fig.9 LFSR

This has taps at stages 1 and 4 with XOR feedback. Least significant bit of shift register is shown at the left hand side, with the output being taken from the MSB at right hand side. LFSR produce a pseudorandom sequence of length 2^{n-1} states (where n is the number of stages) if the LFSR is of maximal length. The sequence will then repeat from the initial state for as long as the LFSR is clocked.

V. Simulation Results of BIST Multiplier

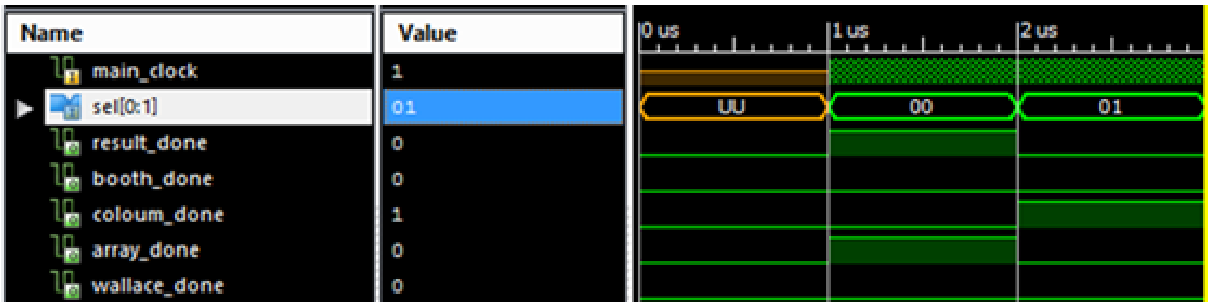
1. BIST Array Multiplier: This is the simulation Waveform of verification of array multiplier. From the waveform in fig.10, it can be easily depicted that when SEL= 00, then verification of array multiplier occur. Clock is given as input, and array_done is the output which is '1'.



Name	Value
main_clock	1
sel[0:1]	00
result_done	1
booth_done	0
coloum_done	0
array_done	1
wallace_done	0

Fig.10 Simulation Waveform of BIST Array Multiplier

2. BIST Column Bypass Multiplier: This is the simulation Waveform of verification of column bypass multiplier. From the waveform in fig. 11, it can be easily depicted that when SEL= 01, then verification of column bypass multiplier occur. Clock is given as input, and column_done is the output which is '1'**.**



Name	Value
main_clock	1
sel[0:1]	01
result_done	0
booth_done	0
coloum_done	1
array_done	0
wallace_done	0

Fig.11 Simulation Waveform of BIST Column Bypass multiplier

3. BIST Booth Multiplier: This is the simulation Waveform of verification of booth multiplier. From the waveform in fig.12, it can be easily depicted that when SEL= 10, then verification of booth multiplier occur. Clock is given as input, and booth_done is the output which is '1'.

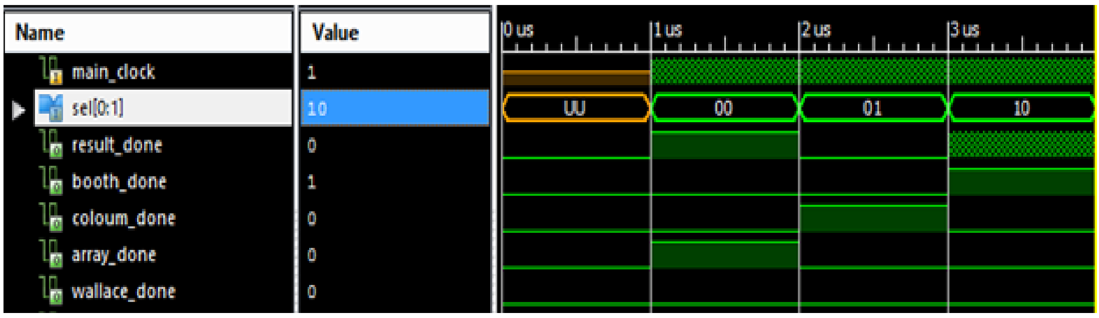


Fig.12 Simulation Waveform of BIST Booth Multiplier

4. BIST Wallace Tree Multiplier: This is the simulation Waveform of verification of Wallace tree multiplier. From the waveform in fig.13, it can be easily depicted that when SEL= 11, then verification of Wallace tree multiplier occur. Clock is given as input, and wallace_done is the output which is '1.

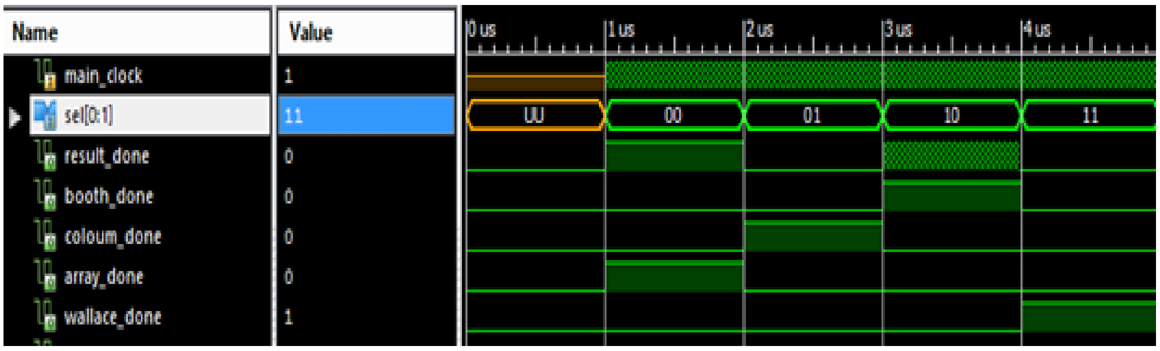


Fig.13 Simulation Waveform BIST Wallace Tree Multiplier

Top level Schematic -

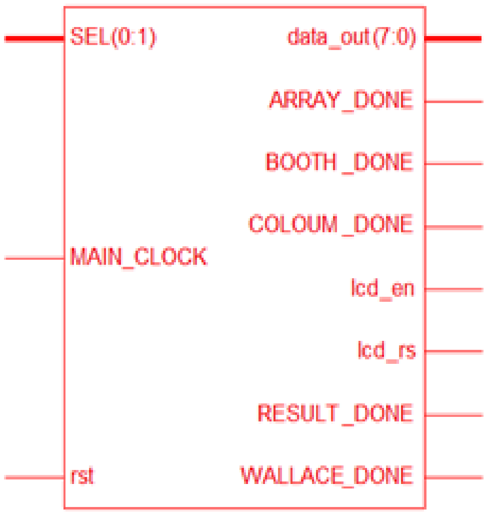


Fig.14 Top level view of BIST Multiplier

RTL View-

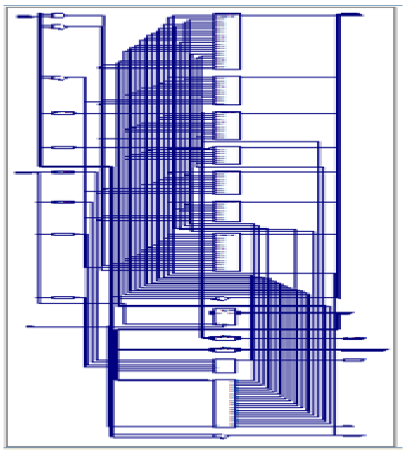


Fig.15 RTL View of BIST Multiplier

VI. Conclusion

By implementing BIST architecture of these multiplier, their on-chip verification can be done. The advantage of using this design is that, it need not to be checked by any other means. It will do the verification by itself. So this design will save the verification time. When the BIST architecture of these four multiplier is implemented, it get observed that implementation of BIST increases the area. This BIST, will not give the optimize result. This is the drawback of this design, that it occupy more area, so cost increases.

Acknowledgment

I would like to acknowledge my mentor Gaurav Jindal Sir who supported me during the period in calculating my results and verifying codes.

References

[1]. N.Ravi, Dr.T.S.Rao, Dr.T.J.Prasad.”Performance Evaluation of Bypassing Array Multiplier with Optimized Design”, International Journal Of Computer Applications (0975-8887), Vol28 No.5, 3 (2011).
[2]. Mahzad Azarmehr., “Multipliers, Algorithm And Hardware Designs”, Research Center for Integrated Microsystem, 10 (2008).
[3]. Nishat Bano, “VLSI Design of low power booth multiplier”, International Journal of Scientific and Engineering Research Vol 3 issue 2, 1-2(feb-2012).
[4]. C.Krishnamacharya ,CH. Sravanthi, K. Avinash, K.V. Uma Maheswaar Rao.,”Design of low power 2-D Multiplier using Bypassing Technique”,International Journal of Innovative Research and Studies ISSN-2319-9725, 198 (May 2013).
[5]. Partha Sarathi Mohanty, “Design and Implementation of low power multipliers” WILLOW project, 16, 26 (2009).
[6]. Sumit Vaidya1 and Deepak Dandekar.,”Delay-Power Performance Comparison of Multiplier in VLSI Circuit Design”, International Journal of Computer Networks and Communication Vol2 No.4, 49 (July2010).