Built-in self-repair (BISR) technique widely Used to repair embedded random access memories (RAMs)

V.SRIDHAR [1] M.RAJENDRA PRASAD[2]

[1] Assistant Professor, ECE, Vidya Jyothi Institute of Technology, Hyderabad
[2] Associate Professor, ECE, Vidya Jyothi Institute of Technology, Hyderabad
[1]varadalasri@gmail.com,[2]rajendraresearch@gmail.com

ABSTRACT: With the trend of SOC technology, high density and high capacity embedded memories are required for successful implementation of the system. In modern SOCs, embedded memories occupy the largest part of the chip area and include an even larger amount of active devices. As memories are designed very tightly to the limits of the technology, they are more prone to failures than logic. Thus, memories concentrate the large majority of defects. That is, RAMs have more serious problems of yield and reliability. Keeping the memory cores at a reasonable yield level is thus vital for SOC products. As a matter, Built-In Self-Repair is gaining importance. Built-in self-repair (BISR) technique has been widely used to repair embedded random access memories (RAMs). If each repairable RAM uses one self contained BISR circuit (Dedicated BISR scheme), then the area cost of BISR circuits in an SOC becomes high. This, results in converse effect in the yield of RAMs. This paper presents a reconfigurable BISR (ReBISR) scheme for repairing RAMs with different sizes and redundancy organizations. An efficient redundancy analysis algorithm is proposed to allocate redundancies of defective RAMs. In the ReBISR, a reconfigurable built-in redundancy analysis (ReBIRA) circuit is designed to perform the redundancy algorithm for various RAMs. The ReBISR structure has been synthesized and found that the area cost when compared with the Dedicated BISR structure is very small. This paper is implemented using Verilog HDL. Simulation and Synthesis is done using ModelSim and Xilinx ISE 12.4 Tools.

Keywords: REBISR,HDL,SOC TECHNOLOGY,BISR,RAM

I . INTRODUCTION:

The VLSI manufacturing technology advances has made possible to put millions of transistors on a single die. This advancement in IC technology enabled the integration of all the components of a system into a single chip. A complex IC that integrates the major functional elements of a complete end product into a single chip is known as System on Chip (SOC). It enables the designers to move everything from board to chip. SOC incorporates a portable / reusable IP, Embedded CPU, Embedded Memory, Real World Interfaces, Software, Mixed-signal Blocks and Programmable Hardware. Reduction in size, lower power consumption, higher performance, higher reliability, reuse capability and lower cost are the benefits of using SOC. However, before SOC products can be widely seen on the market, many design and manufacturing issues have to be solved first. One of them is testing plural, heterogeneous cores of the SOC and the chip itself.

With the trend of SOC technology, high density and high capacity embedded memories are required for successful implementation of the system. Memories are the widely used cores on the SOC products. Typically, many RAMs with various sizes are included in an SOC and they occupy a significant portion of the chip area. But, due to the continual advances in the manufacturing process of IC, provide designers the ability to create more complex and dense architectures and higher functionality on a chip. Although it benefits the end user, but these manufacturing process

advances are not without limitations. In particular, embedded high density memories in combination with these process limitations can result in poor over all yields. That is, RAMs have more serious problems of yield and reliability than any other embedded cores in an SOC. Depending upon the application and design, much of the low yield can be attributed to faults in the memory. Keeping the memory cores at a reasonable yield level is thus vital for SOC products. For such purpose, memory designers usually employ redundancy repair logic using spare rows and/or columns to improve the yield.

However, redundancy increases the silicon area and thus has a negative impact on yield. To maximize the yield, redundancy analysis is necessary. Conventionally, redundancy analysis (RA) is performed on the host computer of the ATE if it is an online process or on a separate computer if it is an offline process. Either way it is time consuming since RA algorithms are complicated and the memories that implement the redundancies are usually large. Moreover, embedded memories are harder to deal with Automatic Test Equipment (ATE). The

BISR (Built in Self Repair) technique is a promising and popular solution for enhancing the yield of memories with the redundancy logic.

Furthermore, redundancies of memories are not just for defects, redundancies can also be used to recover yield due to process variation in addition to yield recovery for defects.

2.Built-in self-repair (BISR): Today's deep submicron technologies allow the implementation of multiple memories on a single chip. Due to their high density, memories are more prone to faults. These faults impact the total chip yield. One way to solve this problem is to enhance the memory by redundant memory locations. The address mapping of the fault free working memory is programmable within certain limits. In order to do so, a memory test is needed to identify the faulty regions.

The memory is tested by external test hardware or by on chip dedicated hardware (memory BIST). The second testing strategy is the preferred method for embedded memories. After memory testing the memory address map is programmed by means of volatile or non-volatile storage on or off chip. To provide the test pattern from a memory BIST a multiplexer in front of the memory is widely used. The redundant spare rows and spare columns are often included into the memory. This impacts the performance and area conditions of the memory. The memory is repaired during testing by storing faulty addresses in registers. These addresses can be streamed out after test completion. Furthermore, the application can be started immediately after the memory BIST passes. The redundancy logic calculation will not increase the test time of the memory BIST.

The memory BISR(MBISR) concept contains an interface between memory BIST(MBIST) logic and redundancy wrapper for storing faulty addresses. This allows using already existing MBIST solutions. The MBIST controller output must provide three signals to the wrapper logic during test.

· A fail signal to store data in the fuse register
· The expected data that is compared to the results of RAM data
· The failing address

Figure 2.1 shows the block diagram of a MBISR scheme for a RAM, which consists of four major. components

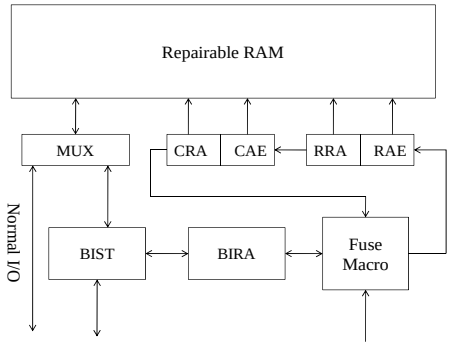


Figure 2.1: MBISR scheme for embedded RAMs.

1) **Repairable RAM:** A RAM with redundancies and reconfiguration circuit is called as a repairable RAM. Figure 2.2 depicts an example of an 8*8 bit-oriented RAM with 1 spare row and 1 spare column. If a spare row is allocated to replace a defective row, then the row address of the defective row is called row repair address (RRA). Then a decoder decodes the RRA into control signals for switching row multiplexers to skip the defective row if the row address enable (RAE) signal is asserted. The reconfiguration of the defective column and the spare column is performed in a similar way, i.e., give a column repair addresses (CRA) and assert the column address enable signal to repair the defective column using the spare column.

2) **BIST Circuit:** It can generate test patterns for RAMs under test. While a fault in a defective RAM is detected by the BIST circuit, the faulty information is sent to the BIRA circuit.

3) **BIRA Circuit:** It collects the faulty information sent from the BIST circuit and allocates redundancies according to the collected faulty information using the implemented redundancy analysis algorithm.

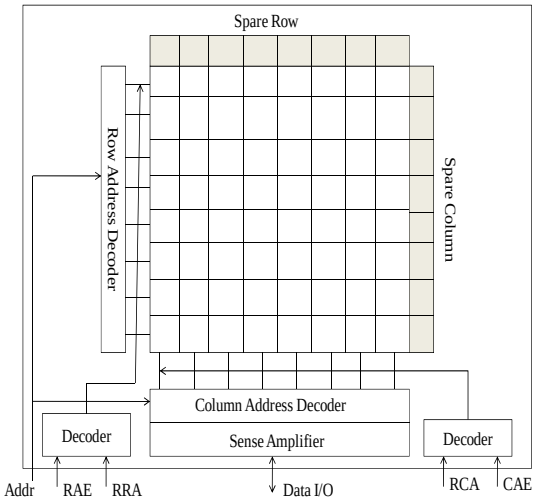
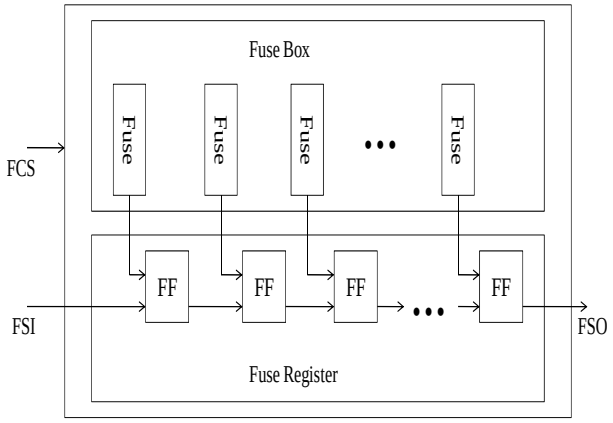


Figure 2.2: Conceptual diagram of an 8*8 bit oriented repairable RAM with one spare row and one spare column

4) **Fuse Macro:** It stores repair signatures of RAMs under test. Figure 2.3 shows the conceptual block diagram of a typical implementation of fuse macro. The fuses of the fuse box can be implemented in different technologies, e.g., laser blown fuses, electronic-programmable fuses, etc. The fuse register is the transportation interface between the fuse box and the repair register in the repairable RAM.



FCS: Fuse Control Signals;

FSI: Fuse Scan In ;

FSO: Fuse Scan Out

Figure 2.3: Conceptual block diagram of fuse macro

As figure 2.1 shows, the overall RAM BISR flow is roughly described as follows. Firstly, the BIST tests the repairable RAM. If a fault is detected, then the fault information is stored in the BIRA circuit. Then, the BIRA circuit allocates redundancies to replace defective elements. As soon as the repair process is completed, the repair signatures are blown in the fuse box. Subsequently, the repair signatures are loaded into the fuse register first and then are shifted to the repair registers (i.e., registers in the wrappers for storing RRA, RAE, CRA, and CAE data) in normal operation mode. Finally, the repairable RAM can be operated correctly.

2.1.1 Reconfiguration techniques: Redundancy analysis and repair procedures are interrelated. During testing, when failures are identified and located, a redundancy analysis procedure determines which failures can be repaired using the given redundant rows and columns. On the basis of this redundancy analysis, either a hard or a soft repair procedure is executed.

- **Hard repair:** In general, hard repair uses fuses, antifuses, or laser programming to disconnect rows and columns with faulty bits and replace them with redundant rows or columns. This method, long used by commodity memory vendors, has been adopted by ASIC and SOC vendors and is currently the most widely used method for stand-alone as well as embedded memories.
- **Soft repair:** Soft repair uses an address-mapping procedure to bypass the faulty address location. This scheme links BIST and power-on, so that every time power is switched on, memory is tested via BIST. During this testing, the addresses of all failing locations are stored separately, and an address mapping procedure maps them onto redundant fault-free addresses. Although a dedicated finite-state machine serves to implement an address-mapping procedure, it can also be implemented efficiently through an on-chip microprocessor or other logic cores.

2.2 Random access memory (RAM)

Random access memory (RAM) is the best known form of computer memory. RAM is considered "random access" because we can access any memory cell directly if we know the row and column that intersects at that cell. Similar to a microprocessor, a memory chip is an integrated circuit (IC) made of millions of transistors and capacitors. The opposite of RAM is serial access memory (SAM). SAM stores data as a series of memory cells that can only be accessed sequentially. If the data is not in the current location, each memory cell is checked until the needed data is found. SAM works very well for memory buffers, where the data is normally stored in the order in which it will be used. RAM data, on the other hand, can be accessed in any order.

2.2.1 Types of RAM:

The two main forms of modern RAM are static RAM (SRAM) and dynamic RAM (DRAM).

SRAM

In static RAM, a bit of data is stored using the state of a flip-flop.

DRAM

Dynamic RAM stores a bit of data using a transistor and capacitor pair, which together comprise a memory cell.

ECC RAM

ECC memory (which can be either SRAM or DRAM) includes special circuitry to detect and/or correct random faults (memory errors) in the stored data, using parity bits or error correction code.

SDRAM

Synchronous Random Access Memory, synchronizes the input and output signals with the system board.

DDR SDRAM

Double data rate synchronous dynamic RAM is just like SDRAM except that is has higher bandwidth, meaning greater speed.

2.2.2 Memory fault models:

Testing of embedded memory isn't new and because memory is used as a test vehicle for technology development, memory testing has been fairly well studied. Nevertheless, such testing remains a major problem in embedded-core-based SOC as well as in complex microprocessors. In many cases, built-in self-tests are used to test embedded memory. In SOC designs, however, several test methods are used. Memory repair is also a key issue for large embedded memories in SOC.

A wide variety of internal faults can occur in semiconductor memories, causing various types of failures in the memory function. Test procedures necessary to detect these faults can be classifed as in any digital circuit test into three classes DC parametric testing, AC parametric testing and functional testing.

1. **DC parametric testing:**

As the name suggests, measures the DC parameters in semiconductor RAMs. It checks for unaccepatble output level, high power consumption, fanout capability, noise margins, rise times and fall times in electrical signals. It contains open/short test, power consumption test, leakage test, threshold test, output drive current test, output short current test. This kind of testing depends on the implementation details of RAM under test and usually the DC parametric test procedure is supplied by the manufacturer.

2. **AC parametric testing:**

AC parametric testing or dynamic testing measures AC parameters of the RAM to detect any malfunction. The parameters measured include memory access time, output signal – the rise and fall times, relationship between input signals – the setup and hold times, relationship between input and output signals – the delay and

access times, successive relationship between input and output signals – the speed test. Examples of malfunctions that can be detected by AC testing include

2.2.3 Basic BIST Architecture: The various components of BIST hardware are the test pattern generator (TPG), the test controller, circuit under test (CUT), input isolation circuitry and the output response analyzer (ORA). This is shown in the figure 2.2.3 below.

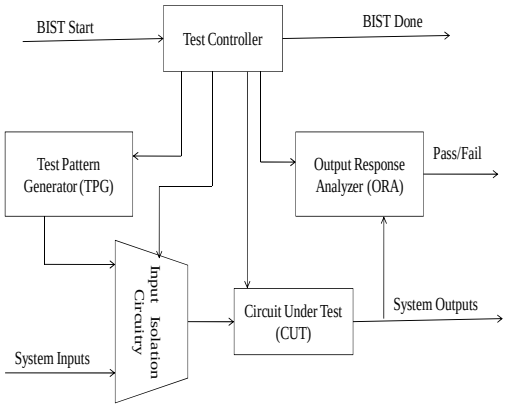


Figure 2.2.3: Basic BIST architecture.

Test Pattern Generator (TPG):

Responsible for generating the test vectors according to the desired technique (i.e. depending upon the desired fault coverage and the specific faults to be tested for) for the CUT. Linear feedback shift register (LFSR) and pseudo random pattern generator (PRPG) are the most widely used TPGs.

Test Controller:

Responsible for controlling the other components to perform the self test. The test controller places the CUT in test mode and allows the TPG to drive the circuit's inputs directly. During the test sequence, the controller interacts with the ORA to ensure that the proper signals are being compared. The test controller asserts its single output signal to indicate that testing has completed, and that the ORA has determined whether the circuit is faulty or fault-free.

Output Response Analyzer (ORA):

Responsible for validating the output responses i.e. the response of the system to the applied test vectors needs to be analyzed.

Also, a decision is made about the system being faulty or fault-free. LFSR and multiple input signature register (MISR) are the most widely used ORAs.

2.3 Classification of Test Patterns:

The fault coverage that we obtain for various fault models is a direct function of the test patterns produced by the TPG. There are several classes of test patterns. TPGs are sometimes classified according to the class of test patterns that they produce. The different classes of test patterns are briefly described below:

Deterministic Test Patterns:

These test patterns are developed to detect specific faults and/or structural defects for a given CUT. The deterministic test vectors are stored in a ROM.

- **Algorithmic Test Patterns:** Algorithmic test patterns are specific to a given CUT to test for specific fault models. Because of the repetition and/or sequence associated with algorithmic test patterns, they are implemented in hardware using finite state machines (FSMs).
- **Exhaustive Test Patterns:** Every possible input combination for an N-input combinational logic is generated and the exhaustive test pattern set will consist of 2^N test vectors. This number could be really

huge for large designs, causing the testing time to become significant. TPG could be implemented using an N-bit counter.

- **Pseudo-Exhaustive Test Patterns:** The large N-input combinational logic block is partitioned into smaller combinational logic sub-circuits.
  Each of the M-input sub-circuits (M<N) is then exhaustively tested by the applying all the possible 2^K input vectors. TPG could be implemented using counters, Linear Feedback Shift Registers (LFSRs).
- **Random Test Patterns:** A truly random test vector sequence is used for the functional verification of the large designs(microprocessors). However, the generation of truly random test vectors for a BIST application is not very useful since the fault coverage would be different every time the test is performed as the generated test vector sequence would be different and unique every time.
- **Pseudo-Random Test Patterns:** These are the most frequently used test patterns in BIST applications. Pseudo-random test patterns have properties similar to random test patterns, but in this case the vector sequences are repeatable. The repeatability of a test vector sequence ensures that the same set of faults is being tested every time a test run is performed.

2.4 **MBIST implementations:**

A memory BIST unit consists of a controller to control the flow of test sequences and other components to generate the necessary test control and data. The various types of memory BIST are categorized according to the schemes of their controllers. Designs of a memory BIST controller could be roughly classified into three different types: (1) a hardwired based, (2) microcode-based, and (3) processor-based. The following three subsections give the specifics of each scheme.

2.4.1 Hardwired-based MBIST:

A hardwired-based controller is a hardware realization of a selected memory test algorithm, usually in the form of a FSM. This type of memory BIST architecture has optimum logic overhead, however, lacks the flexibility to accommodate any changes in the selected memory test algorithm. This results in re-design and re-implementation of the hardwired-based memory BIST for any minor changes in the selected memory test algorithm. Although it is the oldest memory BIST scheme amongst the three, hardwired-based BIST is still much in use and techniques have been kept developing.

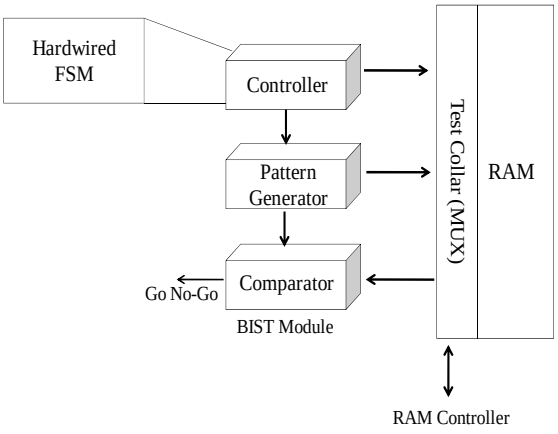


Figure 2.4.1: Hardwired based MBIST

2.4.2 Microcode-based MBIST:

A microcode-based memory BIST features a set of predefined instructions, or microcode, which is used to write the selected test algorithms. The written tests are loaded in the memory BIST controller. This microcode-based type of memory BIST allows changes in the selected test algorithm with no impact on the hardware of the controller. The approach of this microcode-based design is to focus on capturing Address Decoder Open Faults (ADOF) and detecting some NPSFs in addition to the conventional SF, TF, CF, DRF faults. Figure 2.8

illustrates a microcode-based memory BIST which consists of storage unit, instruction counter, instruction decoder, LHCA address generator, and comparator. The storage unit is a 6x10 bits ROM that stores the conventional march test algorithms. The instruction counter is a log2(X)+1 bit binary up-down counter which selects the instruction address of the ROM. The instruction decoder generates the up-down address and hold/enable signals by taking the instruction condition bits and LHCA terminal signals. The up down LHCA is a randomly inversed LHCA that is generated to generate 2000 independent random address pattern in hope to provide better NPSF coverage. Finally, the comparator produces an error signal by comparing the test data and RAM output data.

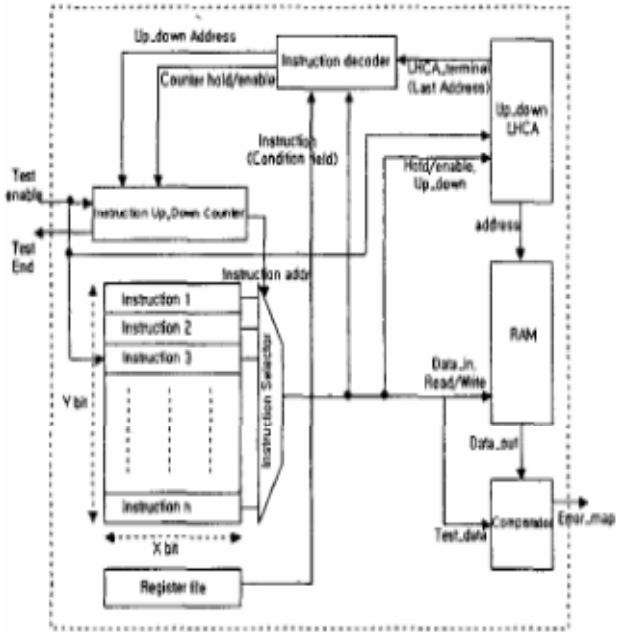


Figure 2.4.3: Microcode-based MBIST

2.4.4 Processor-based MBIST:

To provide a low cost test solution for the on-chip memory cores is a challenging task. Conventional hardwired-based MBIST approach is one possible solution and its advantages are short test time and small area overhead. However, sometimes it is not feasible to have one BIST circuit for each memory core. If each memory core on chip requires a BIST circuit, then the area and test pin overhead will be unacceptably high. Therefore, a new type of MBIST scheme which utilizes an on-chip microprocessor to test the memory cores was proposed. The processor-based MBIST is done by executing an assembly-language program in the on-chip microprocessor to generate test patterns including the address sequence, data patterns, and control signals. The memory outputs are then compared with the expected correct data. As shown in Figure 2.9, the BIST core is inserted between the CPU core and the on chip bus, which also connects the memory cores. In normal operation mode, the CPU transparently accesses the system bus with slight time overhead introduced by the multiplexers. In memory BIST mode, the BIST circuitry takes over the control of the on chip bus.

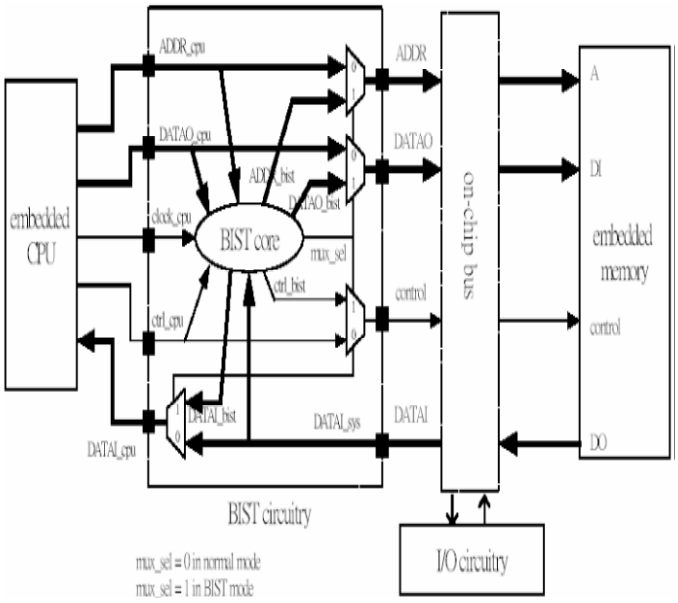


Figure 2.9: Modified processor-based MBIST architecture

It executes certain test algorithm programmed by the CPU and generates the addresses, input data, and control signals of the memory core. It also compares the memory output response with the expected correct data. In order to allow these two different modes, several multiplexers are used to multiplex the address bus, data input bus, data output bus, and control bus between the CPU core and the BIST circuitry.

3. IMPLEMENTATION OF THE PROPOSED REBISR SCHEME FOR MULTIPLE RAMS:

3.1 Architecture of the ReBISR Scheme: Figure 3.1 shown below is the simplified block diagram of the proposed ReBISR scheme for repairing multiple RAMs in an SOC. Four repairable RAMs with various sizes having different number of redundancies are considered in our proposed ReBISR scheme as shown in the below figure. All these RAMs are word oriented memories and there configurations are as listed in the below table 3.1.

RAM No	Data Width * Memory Depth	No. of Spare Rows	No. of Spare Columns
RAM 0	16 * 32	2	2
RAM 1	32 * 64	2	3
RAM 2	128 * 64	3	2
RAM 3	256 * 64	0	0

Table 3.1: Configurations of RAMs

The table 3.2 shows the stuck-at-faults in the four repairable RAMs at their respective fault row and fault column locations as shown below.

RAM Number	Fault Row	Fault Column	Stuck-at-fault
RAM 0	6,7 14	18 28, 29	0 1,1
RAM 1	15, 16 28	43 61	1,1 0
RAM 2	-	-	-
RAM 3	243	31	0

Table 3.2: Location of stuck-at-faults

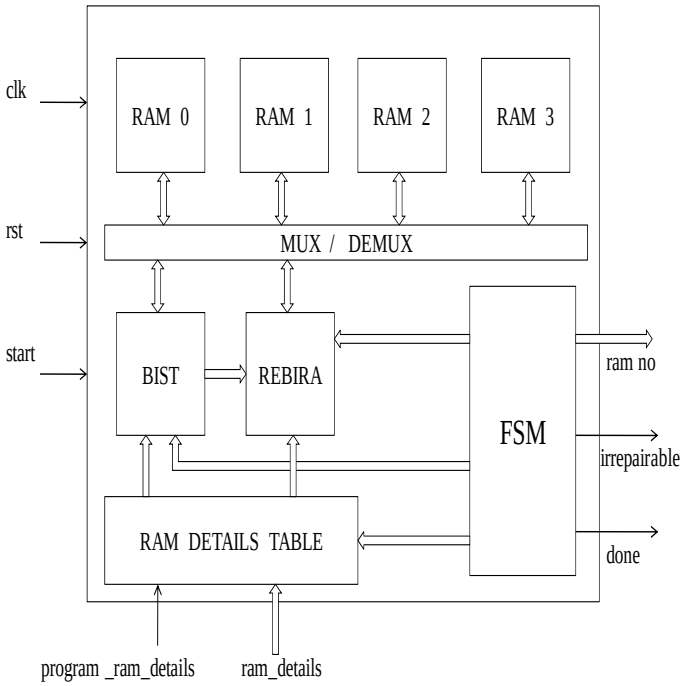


Figure 3.1: Block diagram of the proposed REBISR scheme for repairing multiple RAMs

The block called RAM details table is used for storing the configurations of RAMs which includes the memory data width, memory depth, number of spare rows and number of spare columns. RAM details table is of size 4*16. FSM is the main block that acts as a controller for generating the control signals during testing and repairing processes.

The over all RAM ReBISR flow is described as follows. Before the BIST circuit and the ReBIRA circuit start testing and repairing the RAMs, the RAM configurations (details) should be known by these two circuits. This is done by the FSM, where it generates the necessary control signals that are required for sending the RAM configurations from RAM details table to BIST and ReBIRA circuits. Because, the memory depth and memory data width are required by the BIST circuit for testing the RAM and the number of spare rows and number of spare columns are required by the ReBIRA circuit to perform the analysis before repairing. The process of entering the RAM configurations into the RAM details table is described as follows.

When reset is high, all the locations in the RAM details table are filled with zeroes. Else, if the signal program_ram_details is high, the RAM details are entered into RAM details table through the pin ram_details using the write pointer. As the details are entered one by one, the write pointer is incremented by 1.Once the RAM details table is full, the write pointer stops incrementing and holds the value.

Once the RAM details table is full, the BIST and ReBIRA circuits start the testing and repairing processes of RAMs one by one. If the BIST circuit detects a fault, then the fault information is exported to the ReBIRA circuitry, and then the ReBIRA performs redundancy allocation on the fly using the rules of the implemented redundancy algorithm. The redundancy algorithm implemented in our scheme is Range Checking First Algorithm (RCFA). The ReBIRA allocating redundancy on the fly means that the redundancy allocation process and the BIST process are performed concurrently. The proposed ReBIRA scheme uses a local bitmap (i.e., a small bitmap) to store fault information of the faults detected by the BIST circuit. The bitmap or the fault table present in the ReBIRA circuitry is of size 4*64 in our proposed REBISR scheme. Once the local bitmap is full, the BIST is paused and the ReBIRA allocates redundancies according to the faulty information. After the ReBIRA allocates a redundancy to repair a corresponding faulty row or column, the local bitmap is updated and the BIST is resumed. This process is iterated until the test and repair process is completed. The repair signatures from the ReBIRA circuit are then sent to the repair registers that are present in the repairable RAMs. Repair signatures include repair register data (defective row/column address), repair register address (the address location in the row/column repair registers for storing the defective row/column address) and repair register write signal. The repairing procedure involves the entering of the repair register data in the repair registers. When the repair register write signal is high, then the repair register data is written in the repair registers at the address location specified by the repair register address. The BIST tests the RAMs once again (after the testing and repairing processes) to ensure that there are no faults present.

4. SIMULATION AND SYNTHESIS RESULTS:

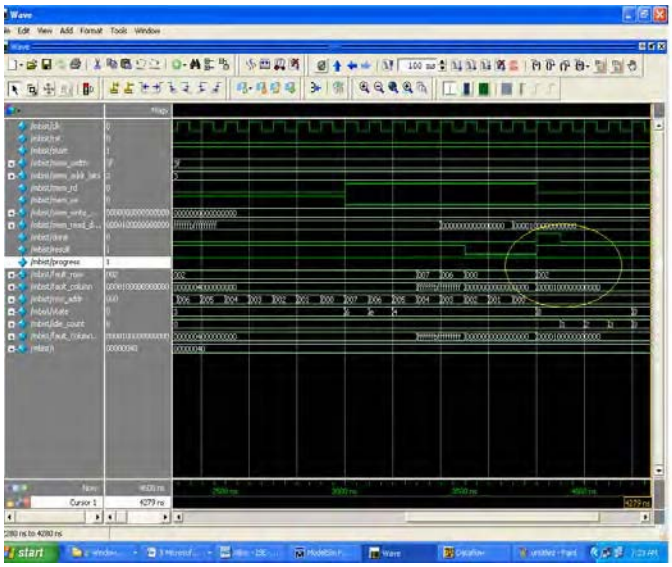


Figure 4.1: Memory BIST waveform

The input pins are clk (clock), rst (reset), start, mem_width (memory width), mem_addr_bits (memory address bits), mem_rd_data(memory read data). The output pins are mem_rd(memory read), mem_wr(memory write), mem_write_data(memory write data), done, result, progress, fault row and fault column. When memory write data is not equal to memory read data, then it is called as fault. It is indicated by the output pin result. The defective row/ column addresses are given by the fault row and fault column addresses respectively. The mem_write_data is 64'h 0000_0000_0000_0000 and the mem_rd_data is 64'h 0000_1000_0000_0000. The Fault row and fault column address are given as 10'h 002 and 64'h 0000_1000_0000_0000 as shown in the above figure

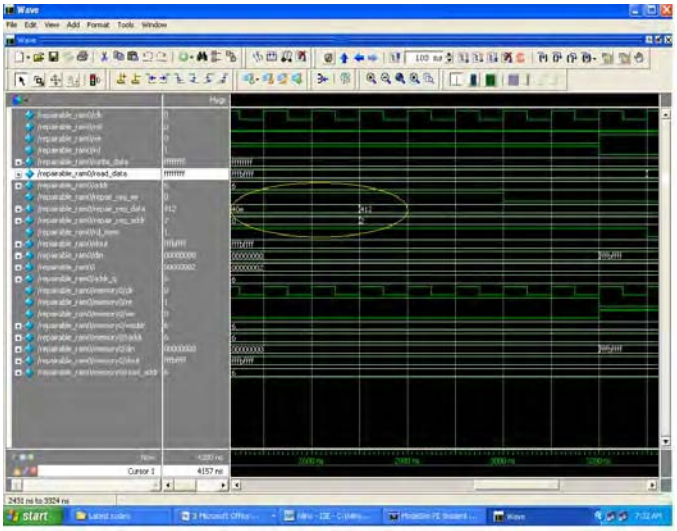


Figure 4.2: Repairable RAM0 waveform

The input pins are clk (clock), rst (reset), rd(read), mem_write_data (memory write data), wr (write), addr (address), rep_reg_wr, rep_regdata, rep_reg_addr (repair register write, data, address). The output pin is mem_rd_data(memory read data). In the repairable RAM0, the fault rows are 6,7,14 and fault columns are 18,28,29. The fault row 14 is replaced with spare row so rep_reg_addr is 0, rep_reg_data is 40e. The fault column 18 is replaced with spare column so rep_reg_addr is 2, rep_reg_data is 412 as shown in the above figure.

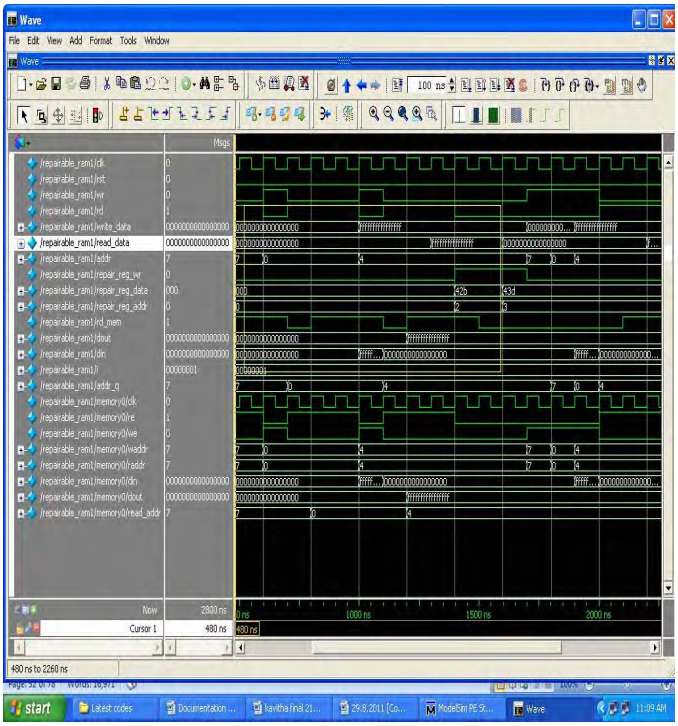


Figure 4.3: Repairable RAM1 waveform

The input pins are clk (clock), rst (reset), rd(read), mem_write_data (memory write data), wr (write), addr (address), repair_reg_wr, repair_reg_data, repair_reg_addr (repair register write, data, address). The output pin is mem_rd_data(memory read data). In the repairable RAM1, the fault rows are 15,16,28 and fault columns are 43,61. The fault column 43 is replaced with spare column so repair_reg_addr is 2, repair_reg_data is 42b. The fault column 61 is replaced with spare column so repair_reg_addr is 3, repair_reg_data is 43d as shown in the above figure.

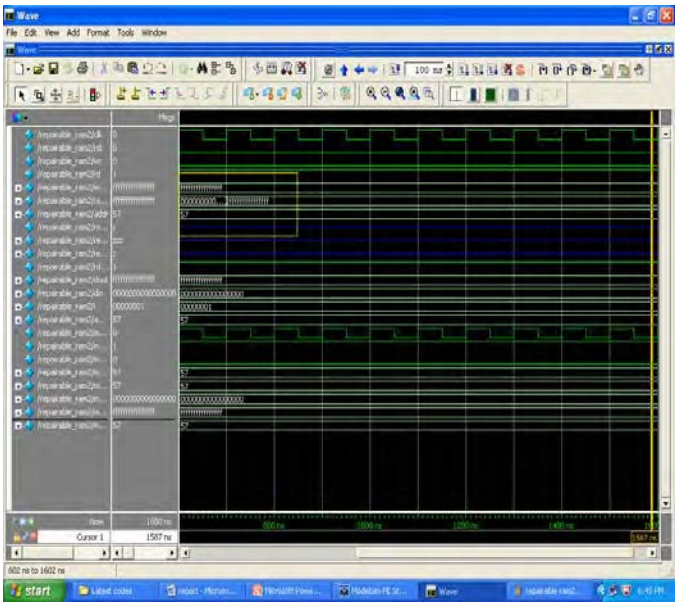


Figure 4.4: Repairable RAM2 wave form

The input pins are clk (clock), rst (reset), rd(read), mem_write_data (memory write data), wr (write), addr (address), repair_reg_wr, repair_reg_data, repair_reg_addr (repair register write, data, address). The output pin is mem_rd_data(memory read data). In the repairable RAM2 there are no faults, so the memory write data and memory read data are equal i.e. mem_write_data is 64'hffff_ffff_ffff_ffff (in the address 64'h57) and mem_rd_data is 64'h ffff_ffff_ffff_ffff as shown in the above figure. The inputs repair_reg_wr, repair_reg_data and repair_reg_addr are not used since there are no faults as shown in the above figure.

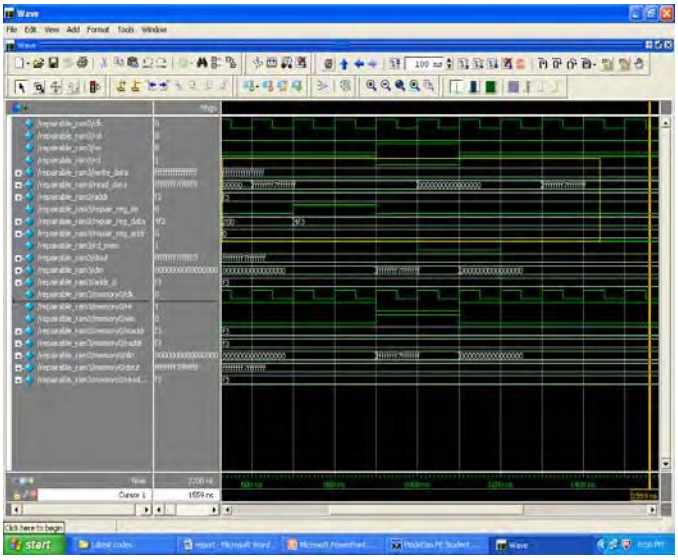


Figure 4.5: Repairable RAM3 waveform

The input pins are clk (clock), rst (reset), rd(read), mem_write_data (memory write data), wr (write), addr (address), rep_reg_wr, rep_regdata, rep_reg_addr (repair register write, data, address). The output pin is mem_rd_data(memory read data). In the repairable RAM3, the fault is at 243 rd row and 31 st column. Hence, the when the mem_write_data is 64'h ffff_ffff_ffff_ffff , the mem_rd_data is 64'hffff_ffff_7fff_ffff. Even though after writing the repair_reg_data (11'h 4f3) in the repair_reg_addr (4'h0), the mem_rd_data is 64'h ffff_ffff_7fff_ffff which means that the fault is not repairable (since there are no spare elements)

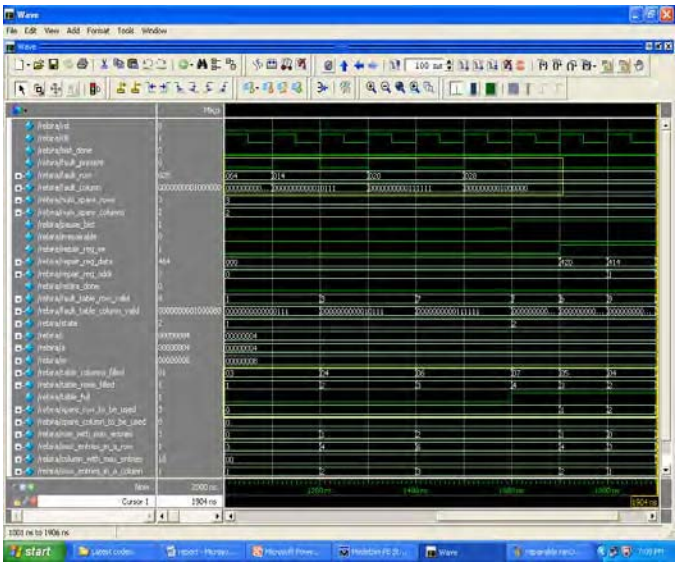


Figure 4.6: Reconfigurable BIRA waveform

The input pins are bist_done(MBIST is done), fault_present, fault row, fault column, no. of spare rows, no. of spare columns, clk (clock), rst (reset). The outputs are pause_bist (Mbist process is paused), irrerepairable, rep_reg_wr, rep_reg_data, rep_reg_addr, rebira_done (rebira process is completed). The fault information like fault rows and fault columns are given as shown in the figure. When fault row and fault column are 10'h028 and 64'h0000000001000000, the table rows filled are 04 and table columns are filled are 07. Spare row to be used is 2 for the fault row 10'h14 and fault column 64'h0000000000000111 as shown in the figure. The table rows filled and columns filled are incremented one by one (1,2,3,4) and (3,4,6,7) as faults are loaded into the bitmap. As the redundancies are allocated one by one, the table rows and columns filled are decremented one by one as (3,2,1) and (5,4,1) as shown in the figure.

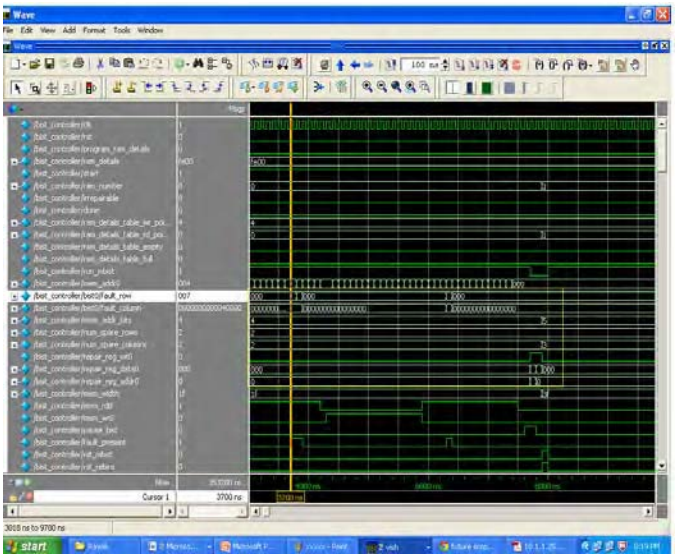


Figure 4.7: Reconfigurable BISR waveform

The RAM details are entered one by one and the RAM details for the RAM3 is 16'hfe00. The RAM details table write pointer is 4 as shown in the figure. When ram number is 0 and when run_mbist is 1, the faults are detected and given as the fault row in RAM0 is 10'h007 and fault column is 64'h0000000000040000 as shown in the figure. The repair signatures are repair_reg_wr is 1'b1, repair_reg_addr is 4'h2 and repair_reg_data is 11'h412 for repairing fault at fault column 18.

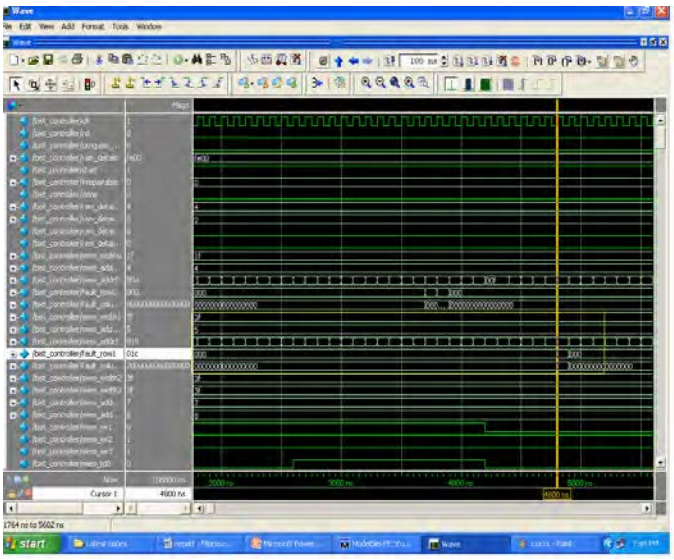


Figure 4.8: Dedicated BISR waveform

The RAM details are entered one by one and the RAM details for the RAM3 is 16'hfe00 and RAM details table write pointer is 4 as shown in the figure. When ram number is 1 and when run_mbist1 is 1, the faults are detected and given as the fault row in RAM1 is 10'h01C and fault column is 64'h2000000000000000 as shown in the figure. The repair signatures are repair_reg_wr is 1'b1, repair_reg_addr is 4'h3 and repair_reg_data is 11'h43d for repairing fault at fault column 61.

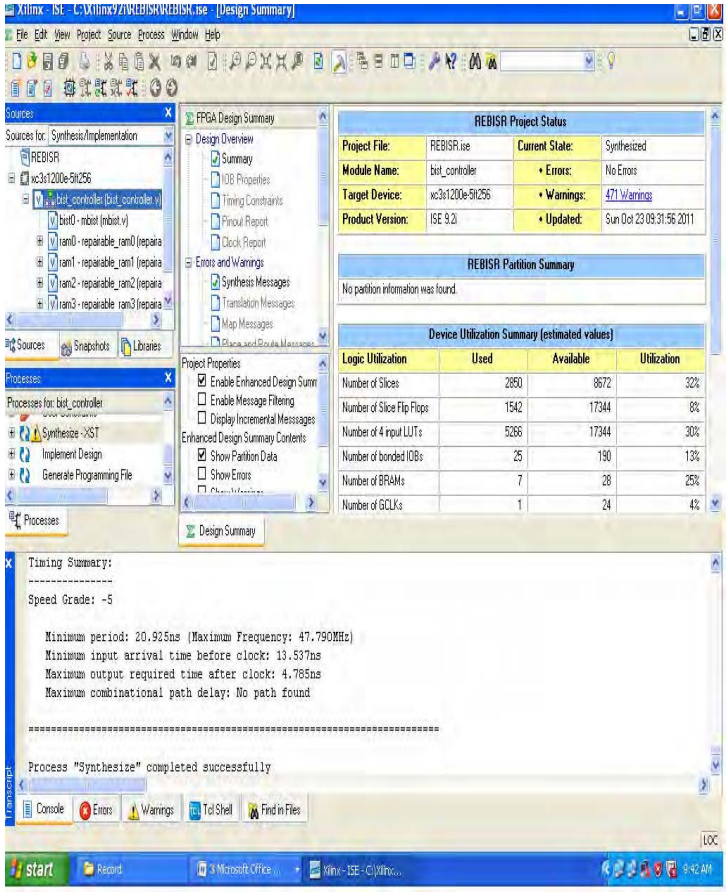


Figure 4.9: Synthesis report of Reconfigurable BISR

Device Utilisation Summary

Selected Device : xc3s1200e-5ft256
Number of Slices : 2850 out of 8672 32%
Number of Slice FFS: 1542 out of 17344 8%
Number of 4 input LUTs 5266 out of 17344 30%
Number of bonded IOBs: 25 out of 19013% Number of BRAMs: 7 out of 28 25% Number of GCLKs 1 out of 24 4%

Timing Summary:

Speed Grade: -5
Minimum period: 20.925ns (Maximum Frequency: 47.790MHz)
Minimum input arrival time before clock: 13.537ns
Maximum output required time after clock: 4.785ns
Maximum combinational path delay:: No path found
Number of errors : 0 (0 filtered)

4.12 RTL Schematic of Reconfigurable BISR

RTL View is a Register Transfer Level graphical representation of your design. This representation (.ngr file produced by Xilinx Synthesis Technology (XST)) is generated by the synthesis tool at earlier stages of a synthesis process when technology mapping is not yet completed. The goal of this view is to be as close as possible to the original HDL code. In the RTL view, the design is represented in terms of macro blocks, such as adders, multipliers, and registers. Standard combinatorial logic is mapped onto logic gates, such as AND, NAND, and OR.

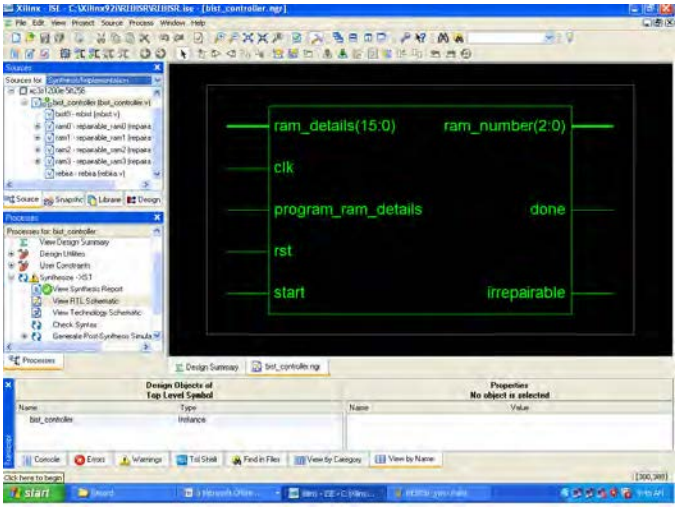


Figure 4.10: RTL Schematic of Reconfigurable BISR

4.13 Technology Schematic of Reconfigurable BISR

Technology view is graphical representation of complete design with main inputs and main outputs, in other words technology view is nothing but graphical representation of top module of the design. In this view only top module will be viewed and none of sub module will be seen.

Figure shows the technology view of the output stage design of Reconfigurable BISR scheme.

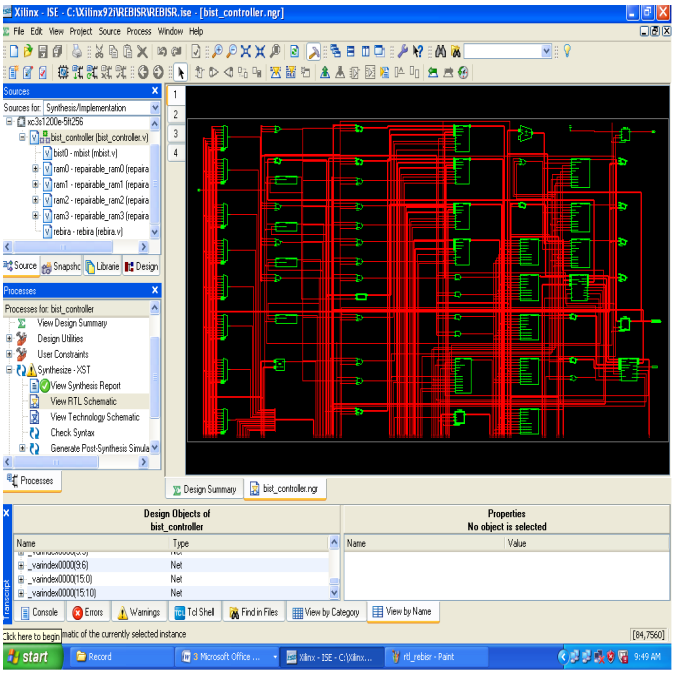


Figure 4.11: Technology Schematic of Reconfigurable BISR

4.14 Synthesis of Dedicated BISR:

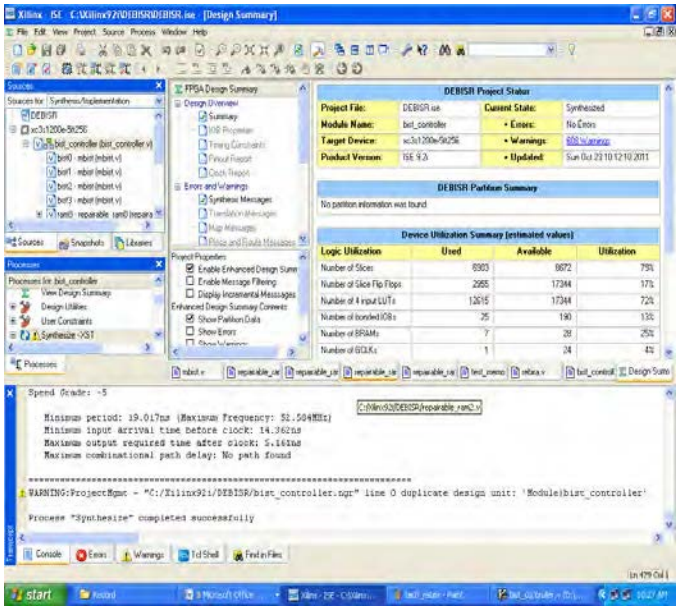


Figure 4.12: Synthesis Report of Dedicated BISR

Device Utilisation Summary

Selected Device : xc3s1200e-5ft256
Number of Slices : 6903 out of 8672 79%
Number of Slice Flip Flops:2955 out of 17344 17%
Number of 4 input LUTs: 12615 out of 17344 72%
Number of bonded IOBs 25 out of 190 13%
Number of BRAMs; 7 out of 28 25%
Number of GCLKs: 1 out of 24 4%

Timing Summary:

Speed Grade: -5
Minimum period: 19.017ns (Maximum Frequency: 52.584MHz)
Minimum input arrival time before clock: 14.362ns
Maximum output required time after clock: 5.161ns
Maximum combinational path delay : No path found
Number of errors : 0 (0 filtered)

4.15 RTL Schematic of Dedicated BISR

RTL View is a Register Transfer Level graphical representation of your design. This representation (.ngr file produced by Xilinx Synthesis Technology (XST)) is generated by the synthesis tool at earlier stages of a synthesis process when technology mapping is not yet completed. The goal of this view is to be as close as possible to the original HDL code. In the RTL view, the design is represented in terms of macro blocks, such as adders, multipliers, and registers. Standard combinatorial logic is mapped onto logic gates, such as AND, NAND, and OR.

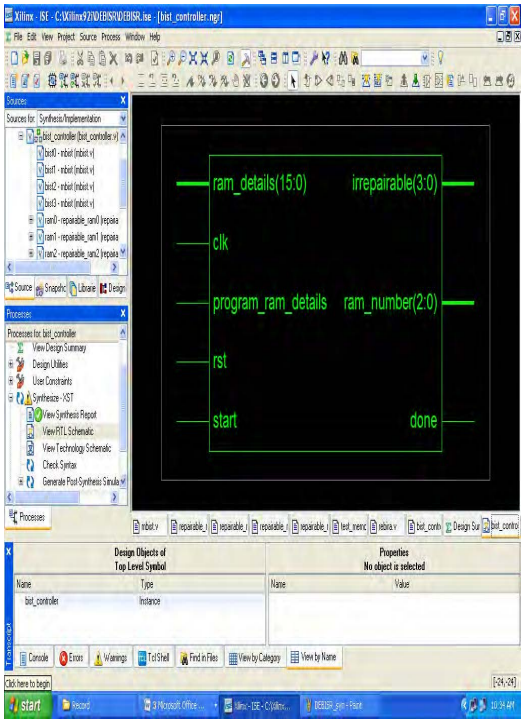


Figure 4.13: RTL Schematic of Dedicated BISR

4.16 Technology Schematic of Dedicated BISR

Technology view is graphical representation of complete design with main inputs and main outputs, in other words technology view is nothing but graphical representation of top module of the design. In this view only top module will be viewed and none of sub module will be seen.

Figure shows the technology view of the output stage design of Dedicated BISR scheme.

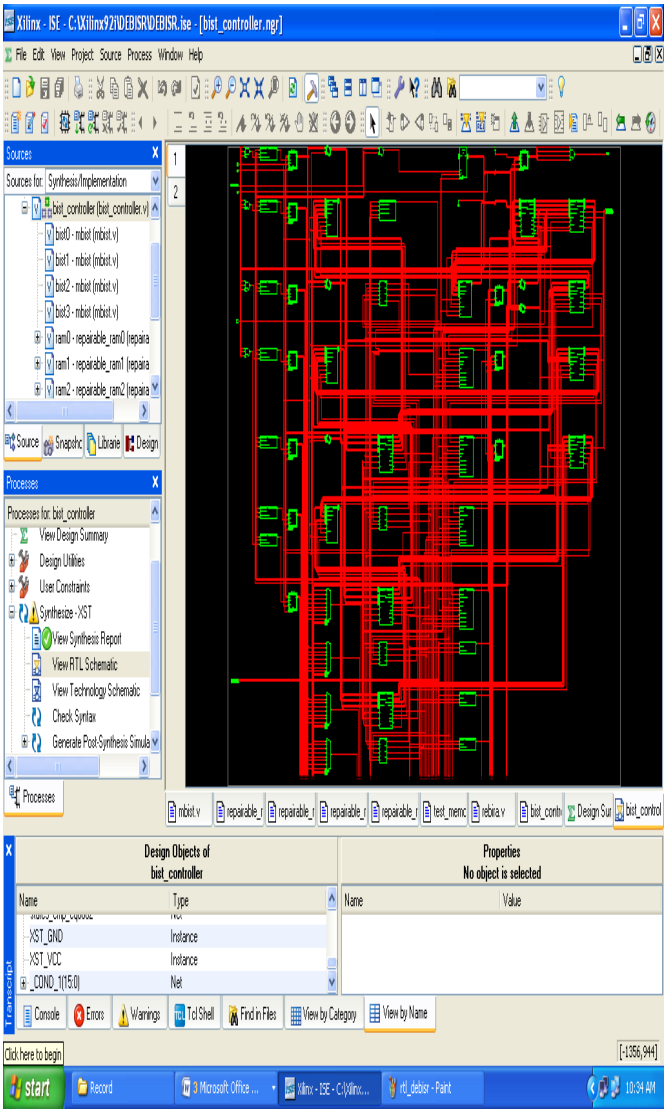


Figure 4.14: Technology Schematic of Dedicated BISR

8. CONCLUSIONS:

If each RAM in an SOC has individual BISR circuit, then the total area of BISR circuits for the RAMs in the SOC is large, since an SOC usually has many RAMs. To reduce the area cost of BISR circuits, we proposed a ReBISR scheme for RAMs in an SOC. A ReBISR circuit can be shared by multiple RAMs such that the total area cost of BISR circuits in an SOC can be drastically reduced.

A reconfigurable BISR scheme for repairing multiple repairable RAMs with different sizes and different numbers of redundancies has been presented. An efficient BIRA algorithm for 2-D redundancy allocation has also been introduced. The BIRA algorithm has been realized in a reconfigurable BIRA hardware, such that it can support different RAMs. Experimental results show that the ReBISR scheme incurs low area cost when compared with the dedicated BISR. Also, the reconfigurable BISR scheme has greater flexibility than the dedicated BISR scheme as the former one supports the repair of multiple memories. Therefore, our ReBISR scheme has low area cost compared with the other BISR scheme for general applications. Future issues include in reducing the time overhead so that, our proposed scheme can be used in specific applications where time is the main criteria.

9. REFERENCES:

[1] John F.Wakerly, Digital Design Principles and practices, Fourth Edition.
[2] Samir Palnitkar, Verilog HDL A guide to Digital Design and Synthesis, SunSoft Press 1996.
[3] Kim, Y. Zorian, G. Komoriya, H. Pham, F. P. Higgins, and J. L. Lweandowski, “Built in self repair for embedded high density SRAM,” in Proc. Int. Test Conf. (ITC), Oct. 1998, pp. 1112–1119.

[4] M. Nicolaidis, N. Achouri, and S. Boutobza, "Optimal reconfiguration functions for column or data-bit built-in self-repair," in Proc. Conf. Des., Autom., Test Eur. (DATE), Munich, Germany, Mar. 2003, pp. 590–595.
[5] M. Nicolaidis, N. Achouri, and S. Boutobza, "Dynamic data-bit memory built-in self-repair," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD), San Jose, CA, Nov. 2003, pp. 588–594.
[6] P. Ohler, S. Hellebrand, and H.-J. Wunderlich, "An integrated built-in self-test and repair approach for memories with 2D redundancy," in Proc. IEEE Eur. Test Symp. (ETS), Freiburg, May 2007, pp. 91–99.

Authors Biography:

[1]VARADALA SRIDHAR is from HYDERABAD,ANDHRAPRADESH,Completed M.TECH in ECE with specialization (WIRELESS AND MOBILE COMMUNICATION SYSTEMS) from vardhaman college of engineering affiliated by JNTUH in 2011.he has completed M.Sc (IT)from Nagarjuna University, guntur, AndhraPradesh.and B.TECH in ECE from vidya jyothi institute of technology affiliated by JNTUH in 2007. Currently he is working as an Assistant professor in ECE department at Vidya Jyothi Institute of Technology, Hyderabad from 2010. His areas of research interests include Wireless and Mobile communications systems,Digitalsignalprocessing,Imageprocessing,
Telecommunications,communication systems, Signal processing,Embedded systems. He has published many international journals papers. He is Lifetime Membership of ISTE, IETE.

[2]M.REJENDRA PRASAD obtained his B.E and M.E Electronics and communication engineering and digital systems from OSMANIA UNIVERSITY, hyderabad.HE has 11 years of experience in embedded and telecom research and development.He is currently working as an
associateprofessor,ECE,DEPARTMENT,VJIT,HYDERABAD.
his main interests are embedded systems,wireless protocol, and RTOS.