International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Design of Microprocessor Hardware Self-Test Unit on FPGA

Savitha Kamble

PG student, VTU PG Regional Center, KALABURGI

Abstract: ***The rapid advancement in the microelectronics design is making a significant demanding situation to design validation in general, but dealing with pipelined microprocessor is remarkable more challenging. The main concern for industry nowadays is testing processor cores embedded in system on chip (SOC). Software based self Test SBST and Built-In Self Test BIST are some of techniques as being developed dynamically and used in many VLSI chips including microprocessor. The proposed project mention a new technique which combines the SBST and BIST method which is done by insertion of programmable infrastructure IP named of Microprocessor Hardware Self-Test (MIHST) that is connected to the system bus. The merits of using the MIHST technique is: it has same or higher defect coverage compare to SBST method, test execution time is reduced, Intellectual Property (IP) of the processor core is preserved, for the storing of the test program or the test data does not need the system memory. An experimental result clearly shows the feasibility and effectiveness of the approach which are evaluated on the pipelined processor and implemented on FPGA***

Keywords: Microprocessor testing, SBST, functional testing, BIST

1. Introduction

At the end of the SOC production cycle one of the most challenging task is testing embedded microprocessor significant changes in testing methodologies for memory arrays. Relatively large die is wasted; hence the failure of embedded memories in a SoC is more expensive than that of commodity memories.

Mostly testing technique are largely depends on the introduction of extra design for testability(DfT) hardware meant for structural testing; scan chain and built-in-self test(BIST) were widely used solution. The use of software-based self test (SBST) is a functional method its use is increasing.

Traditional hardware self-test (BIST) starts the testing task from external resources (ATE) to internal hardware, used to generate test patterns and check test responses of the circuit under test. It achieves at-speed testing reducing the overall cost of the test. But Logic BIST has less applicable to high-performance and power-optimized embedded processors.

Various approaches has been grouped together under the term, software-based self-testing (SBST) and several SBST techniques have been proposed as an effective alternative to hardware self-test for embedded processors. Since it utilizes existing processor resource and instructions to perform self-test.

Therefore, without any impact on performance, area or power consumption during normal operation the SBST can potentially provide sufficient testing quality. Recently SBST approaches with respect to the single stuck-at faults have been proposed. Unfortunately, the SBST approach shows some drawbacks;

- Some faults modulate the test program flow and lead to an endless execution (i.e., caused by a wrong jump outside the test code portion), making it difficult to take back the control of the system at the end of the test
- IP protection is not guaranteed, since the test program may reveal details about the processor core implementation.

In this project, to overcome the above issues we discussed a novel solution which combines the **SBST** and BIST **principl**es. The method we propose forces the processor to execute a compact SBST-like test sequence by using a hardware module called **Microprocessor Hardware Self-Test (MIHST)** unit, which is made to be connected to the system bus like a normal memory core, providing no modulation of the processor core internal structure system generates and provides an instruction stream to the processor core, while also observing the processor behavior. When generating the instruction stream, the MIHST unit does not care about the sequence of instruction addresses generated by the processor for instruction fetches purposes

2. Design Methodology

The design of all the functional blocks and its working along with its RTL view is briefly discussed below.

2.1 Main Bock Diagram

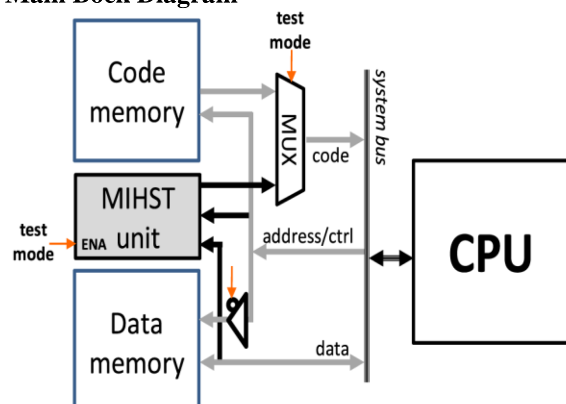

Figure 2.1.1: Architecture of the system with MIHST UNIT

To address the issues of traditional SBST a novel method is proposed, named Microprocessor Hardware Self-Test

Licensed Under Creative Commons Attribution CC BY

(MIHST), which is particularly suitable testing of embedded processors in Systems-on-Chip (SoCs), and overcomes most of the drawbacks of the current SBST approach. The proposed methodology merges features of hardware-based and functional-based techniques. The approach basically consists in the insertion of a programmable Infrastructure-IP (I-IP) named MIHST unit that is connected to the system bus 2.1.1.

The MIHST approach is works on two ideas

1) The system may be either in the normal or in the test state; in the former, the processor executes the instructions read from the code memory; in the latter, the MIHST unit
2) The MIHST unit internally encodes the test program in a custom manner that exploits the test program regularity, minimizing the hardware required to store it.

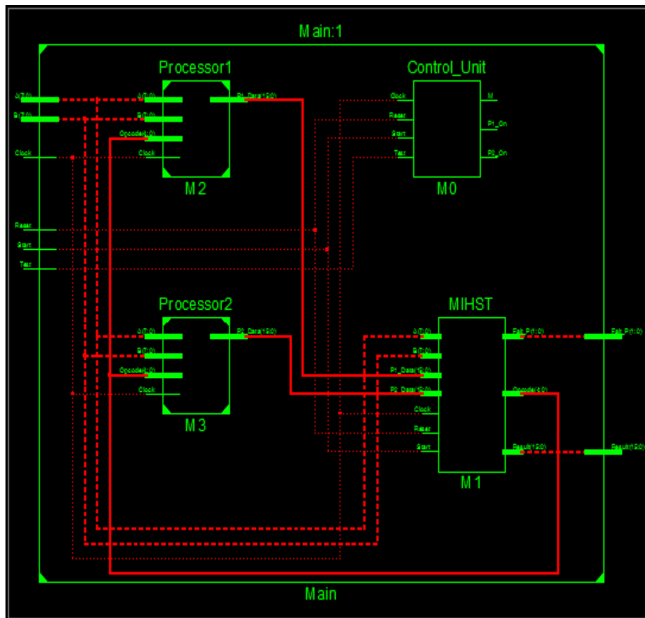

Figure 2.1.2: RTL view of the system

2.2 MIHST Unit

The MIHST unit performs similar to a memory core; provide the processor with appropriate instruction, thus execution of program take place. A multiplexer is provided to impel the data bus with instruction coming either from the code memory (normal mode) or from MIHST unit (test mode) this phenomenon is similar to a usual memory BIST approach. For the purpose of this project we assume that processor core may use any cache or it should be disable during test mode. In a system when MIHST unit included, during test mode, the system memories are o longer used. All the pipelined instructions that is fetch, read and write operations are carried out directly with the MIHST unit and the unit behaves as cache. When the test condition is "ON", the MIHST unit eliminates the use of code memory and communicates with the processor continuously performing the following operation

- An instruction fetch cycle is detected
- Instruction from memory is read
- For the processor instruction code is generated
- Processor behavior is observed

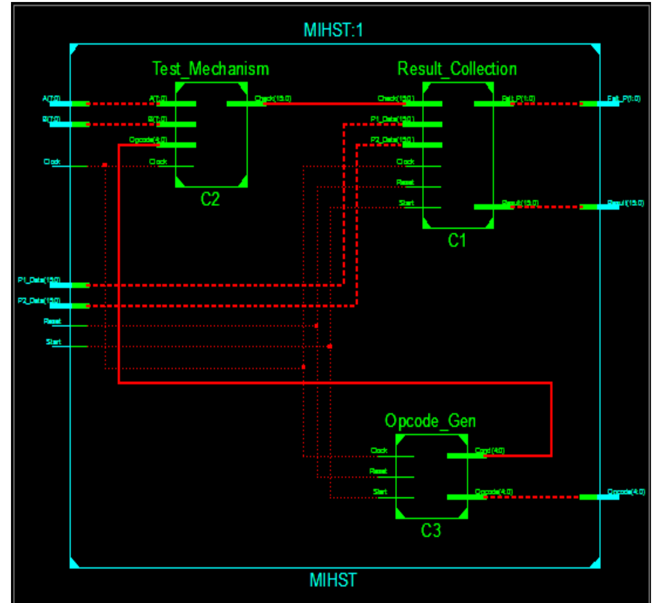

Figure 2.2: RTL schematic of MIHST unit architecture

The MIHST unit includes

- An instruction register, it is used to hold the value which has to put on the bus during the read cycle by CPU.
- A set of internal memories for the storing the values of operand, opcode etc.

2.2.1 Opcode generation

The fig 2.2.1 a shows the RTL schematics of opcode generation block.

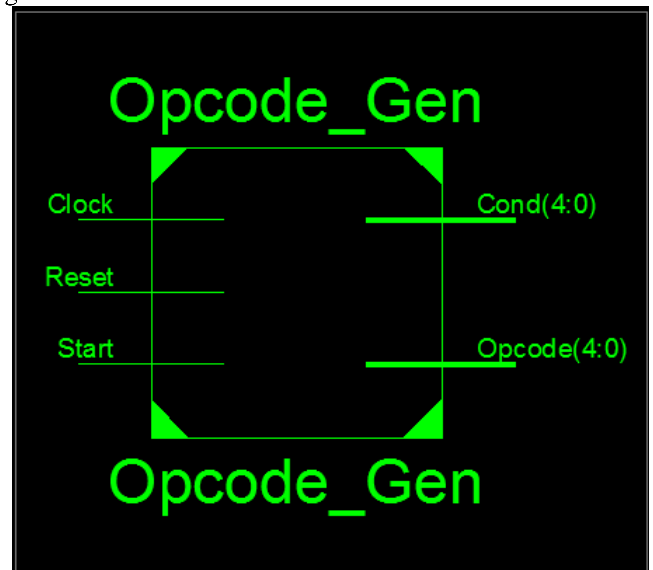

Figure 2.2.1: RTL view of opcode generation

The opcode generation module,it is used to generate a instruction opcodes for microprocessor. The opcode generated is of 5-bits and the two operand is of 8-bits.Opcode generation is actually a test pattern generation method it uses multiple input signature analysis algorithm to generate a test pattern.MISR is a multiple input signature register which is used to test logic with a multiple output.It consists of linear feedback shift register(LFSR) ,where exclusive-OR gates and shift register use to produce a pseudorandom binary sequence(PRBS) at each of the flip-flop outputs.

Licensed Under Creative Commons Attribution CC BY

2.2.2 Test mechanism block

The fig 2.1.2 shows the RTL schematics of test mechanism block

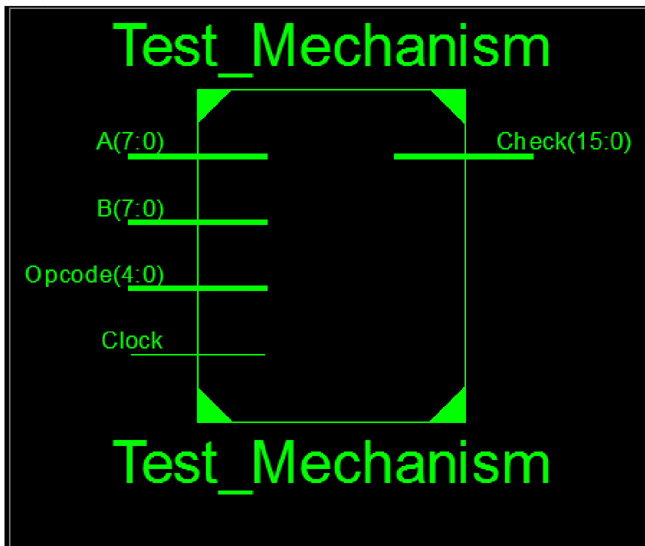

Figure 2.2.2: RTL view of test mechanism block

The block is used to carry out the spefied operation according to opcode generated by the MIHST unit which is automatic test pattern.this unit even provide the interfacing the MIHST unit with the automatic test equipment.

2.2.3 Result collection block

The result collection block is used to compare the result of processor and result generated by the test mechanism unit and indicates which processor has fault.

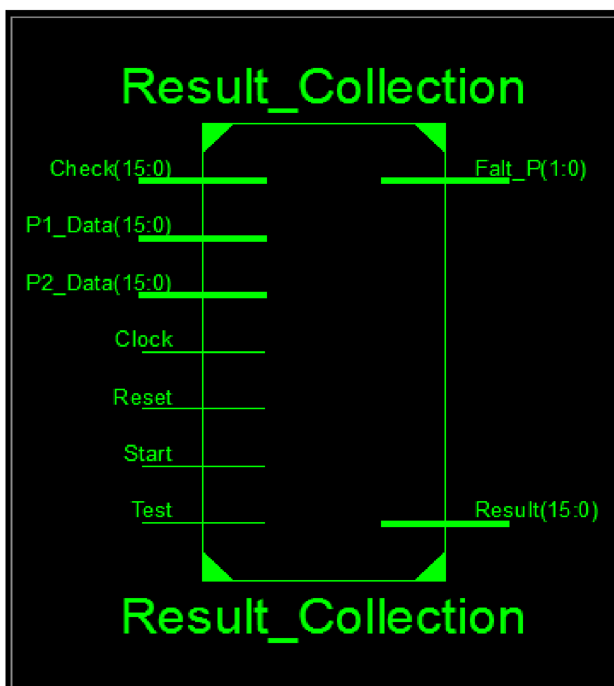

Figure 2.2.3: RTL view of result collection block

2.3 Processor Block

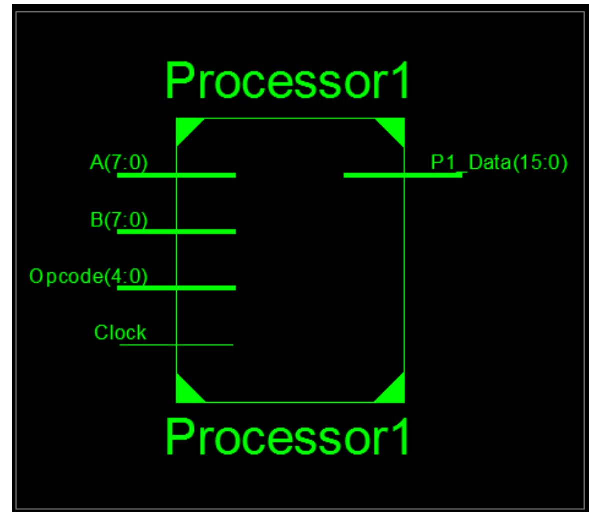

Figure 2.3: RTL view of processor

In our project we are using two 8-bit processor which has input as two 8-bit operand which is user define,4-bit opcode which is generated by MIHST unit and clock is provided.As a result we get 16-bit data after the processing has been done on two operand depending on the opcode.Using two processor so that MIHST unit checks which processor has faultless and which processor has to be modified.The output of processor is fed to the MIHST testing unit so as to test the processor.

3. Simulation Results

The overall simulation results of all blocks are shown in fig 3. It has two conditions

Case1- When TEST signal is OFF that is when TEST signal is LOW “0”

Case2-When TEST signal is ON that is TEST=”1”

The fig 3.1shows the simulation results when testing unit is off, it is indicated by the TEST signal. When the entire input signals are provided with TEST signal as ‘0’ then MIHST unit is off but both the processor runs but no testing mechanism is carried out. for example in above case the inputs are clock, reset and start signal which are provided by the appropriate values and two operands values A and B are provided by user in above case we took A=00001111 and B=01010101.As a result both the processor run but no testing mechanism is carried out.

Licensed Under Creative Commons Attribution CC BY

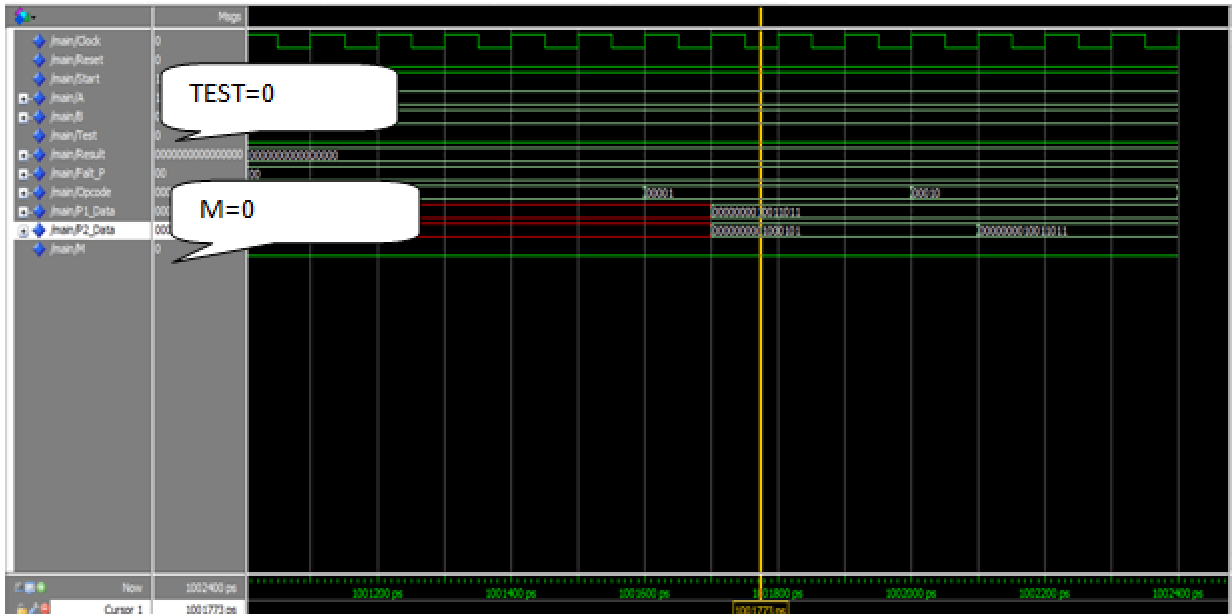

Figure 3.1: simulation result when MIHST is off

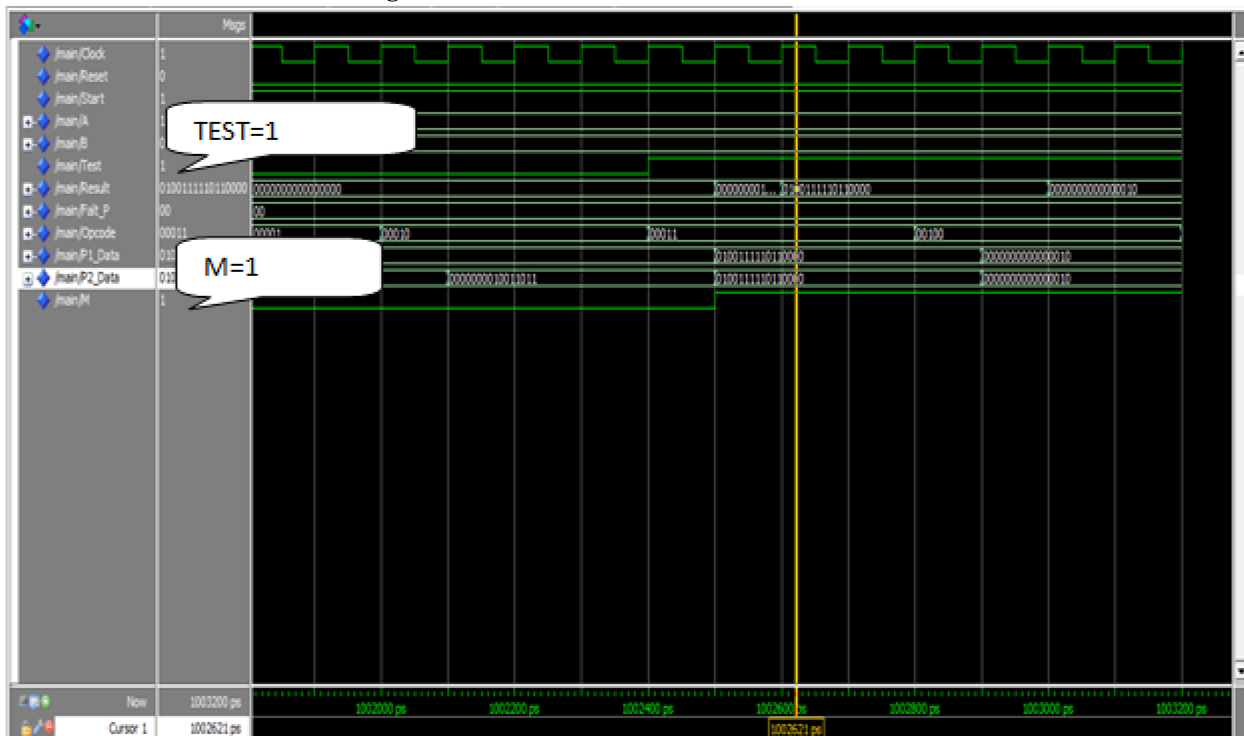

Figure 3.2: simulation result when MIHST unit is ON

The above fig3.2 shows the simulation results when test unit is ON, it is indicated by the TEST signal. When all the input signal are provided with TEST signal as '1' then MIHST unit is ON both the processor runs and testing mechanism is carried out. for example show in fig 5.2.1 the inputs are clock, reset and start signal which are provided by the appropriate values and two operands values A and B are provided by user in above case we took A=11100011 and B=01101100.As a result both the processor and testing mechanism is carried out. The output of both processor is compared with the testing unit results, by which we come to know which processor has fault which is indicated by the signal fault_p.If this signal is '0' means no is both the processor, if its '1' then there is fault in the processor. The operation of processor depends on the opcode generated by testing unit which is 5bit so we can test 32 instructions.

When opcode is 00001 addition operation is carried out in both of the processors named P1 and P2.The results of both the processor is compared with the testing unit, which in turn indicate which of the processor has fault which is shown by signal FALT_P signal.
00-indicates both the processors are faultless
01-indicates processor P1 has fault
10-indicates processor P2 has fault
11-indicates both the processors are fault

As our proposed approach has same fault coverage as SBST technique by it has less application time and test program size. The experimental result is shown below in the table format.

Licensed Under Creative Commons Attribution CC BY

Table 1: Comparison of MIHST unit and SBST approach

	MIHST approach	SBST approach
Execution Time (secs)	7.36	9.43
Test Program Size (kilobytes)	193996	200140

4. Conclusion

The project proposes a new method which is based on the introduction of a special hardware module used for generating the instruction required for the test, which are stored in internal memory. Several advantages we come across when we compare proposed model with the SBST approach, while covering same fault coverage. The cost of insertion of the MIHST unit is limited. Experimental result is carried out on the two processors and the fault is identified that the new method achieving the same fault coverage as SBST with reduced test execution time and less usage of system memory. Currently, work is going on regarding developing MIHST unit which can be able to support the adoption of this method to a wider number of processors.

References

[1] **Base Paper:** Bernardi, P. ; Dipt. di Autom. e Inf., Politec. di Torino, Turin, Italy ; Ciganda, L.M. ; Sanchez, E. ; Reorda, M.S. ***"MIHST: A Hardware Technique for Embedded Microprocessor Functional On-Line Self-Test"*** Computers, IEEE Transactions on (Volume:63 , Issue: 11) 13 October 2014

[2] Goncalves Martins, V.M. ***"Low cost fault detector guided by permanent faults at the end of FPGAs life cycle"*** Test Workshop - LATW, 2014 15th Latin American 12-15 March 2014

[3] Dogaru, E. ; ***"A flexible BIST strategy for SDR transmitters"*** Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014 24-28 March 2014

[4] Zhenqian Zhang ***"Face-to-face bus design with built-in self-test in 3D ICs"*** 3D Systems Integration Conference (3DIC), 2013 IEEE International 2-4 Oct. 2013

[5] Valfre, J. ; Syst. Test Directorate, Raytheon Missile Syst., Tucson, AZ, USA ***"Testability modeling usage in design-for-test and product lifecycle cost reduction"*** AUTOTESTCON, 2012 IEEE 10-13 Sept. 2012

[6] Bernardi, P. ; Dipt. di Autom. e Inf., Politec. di Torino, Torino, Italy ; Ciganda, L. ; De Carvalho, M. ; Grosso, M. ***"On-line software-based self-test of the Address Calculation Unit in RISC processors"*** Test Symposium (ETS), 2012 17th IEEE European 28-31 May 2012

Licensed Under Creative Commons Attribution CC BY