INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE

ISSN: 2057-5688

Realization of Built-In Self Test (BIST) Enabled Memory(RAM) Using VHDL and Implementation in Spartan6 FPGA board

Y.MOUNIKA GOWRI[1], NAGA MALLIKARJUNA[2]
[1]PG Student, Dept of ECE, SITS, Kadapa, AP, India.
[2]Assistant Professor, Dept of ECE, SITS, Kadapa, AP, India.

Abstract— Nowadays for developing any embedded system the microprocessor is used extensively, but they comes to any developer as an black-box in which signals are provided from one side and corresponding output is extracted out from other side. So if developer knows what is hardware present inside the processor which he/she is currently using, then it would be helpful for making his/her design simple & optimized. In this paper, we have represented the design and simulation result of basic three elementary modules of microprocessor i.e. ALU, RAM and Instruction decoder and finally combined all three modules using Structural modeling .This four bit processor has been designed as a prototype for designing advanced processors.

1. INTRODUCTION

For any kind of system, testing and fault diagnosis play a significant role to identify unwanted defects. There are some complicated applications have severe cost constraint but still require high-level of performance and safety. Hence, the utmost concern is where the diagnosis of the defects has to be done even under inadequate resource and time. An approach to solve this dilemma, namely BIST, has been widely implemented in semiconductor industry. With a built in test system positioned inside the circuit, the analysis of every single part within the circuit could become simple task as the period and cost for testing are cut down. This BIST technology is capable of saving the time and cost of maintenance that also allow on line diagnosis, which can deal with even greater advance of embedded systems in future.

In order to fulfill the market need almost every day novel products are through of and are developed using top-notched technology. The first and the foremost expectation from such products is, they are expected to be super quick. To make any product super responsive, the processing capacity and the storage component play equal and vital role. Apart from this, self-diagnosis has gaining significant place in the development of the smart products. The product capable of self-testing reduces time to market, increases the live time and also prevent the possible data loss in case if failure occurs. Different testing mechanisms are employed to implement self-diagnosing procedures in the products. In this review paper we are about to disclose a detailed analysis on different methodologies reported for implementation of the self-diagnosis procedures.

Fault models produce a direct function which is the fault coverage of the test patterns generated by TGP and apply to Circuit Under Test (CUT). There are several classes of test patterns, e.g. Deterministic test patterns, Algorithmic test patterns, Exhaustive test patterns, Pseudo-exhaustive test patterns, Pseudo-random test patterns, Weighted pseudo-random test patterns and Random test patterns. As a result, TPGs are sometimes classified according to the class of test patterns they produce.

In most ORA techniques, the output responses of the CUT to the test patterns are “compacted” into a “signature” which is compared to the expected signature for the fault-free circuit. As we shall see, the Pass/Fail indication is often composed of a set of data bits that contain the signature of the BIST sequence, rather than a single Pass/ Fail bit. Note that the term compaction is used rather than compression since compression implies no loss of information; since most ORA techniques incur some loss of information; compaction is a more accurate term. Since all ORAs perform data compaction, there will be some loss of information. When the lost information contains fault detection data, it is possible for the BIST results to indicate that a faulty CUT is fault-free. This is often referred to as fault masking or, in the cases of some ORA techniques, signature aliasing. A microprocessor control program (embedded software) can be easily tailored to different needs of a product line, allowing upgrades in performance with minimal redesign of the product. Different features can be implemented in different models of a product line at negligible production cost.

Microprocessor control of a system can provide control strategies that would be impractical to implement using electromechanical controls or purpose-built electronic controls. For example, an engine control system in an automobile can adjust ignition timing based on engine speed, load on the

ISSN: 2057-5688

engine, ambient temperature, and any observed tendency for knocking—allowing an automobile to operate on a range of fuel grades. In the mid-1980s to early 1990s, a crop of new high-performance reduced instruction set computer (RISC) microprocessors appeared, influenced by discrete RISC-like CPU designs such as the IBM 801 and others. RISC microprocessors were initially used in special-purpose machines and Unix workstations, but then gained wide acceptance in other roles. The first commercial RISC microprocessor design was released in 1984, by MIPS Computer Systems, the 32-bit R2000 (the R1000 was not released). In 1986, HP released its first system with a PA-RISC CPU. In 1987, in the non-Unix Acorn computers' 32-bit, then cache-less, ARM2-based Acorn Archimedes became the first commercial success using the ARM architecture, then known as Acorn RISC Machine (ARM); first silicon ARM1 in 1985. The R3000 made the design truly practical, and the R4000 introduced the world's first commercially available 64-bit RISC microprocessor. Competing projects would result in the IBM POWER and Sun SPARC architectures. Soon every major vendor was releasing a RISC design, including the AT&T CRISP, AMD 29000, Intel i860 and Intel i960, Motorola 88000, DEC Alpha. In the late 1990s, only two 64-bit RISC architectures were still produced in volume for non-embedded applications: SPARC and Power ISA, but as ARM has become increasingly powerful, in the early 2010s, it became the third RISC architecture in the general computing segment.

2. DIGITAL SYSTEM TESTING

Digital Systems Testing

Reliable electronic systems are not only needed in the areas where failures can lead to catastrophic events but also increasingly required in all application domains. A key requirement for obtaining reliable electronic systems is the ability to determine that the systems are error-free [7]. Although electronic systems contain usually both hardware and software, the main interest of this thesis is hardware testing and especially digital hardware testing. Hardware testing is a process to detect failures primarily due to manufacturing defects as well as aging, environment effects and others. It can be performed only after the design is implemented on silicon by applying appropriate stimuli and checking the responses. Generation of such stimuli together with calculation of the expected response is called test pattern generation. Test patterns are in practice generated by an automatic test pattern generation tool (ATPG) and typically applied to the circuit using automatic test equipment (ATE). Due to the increasing speed of systems and external tester bandwidth limitations, there exist approaches where the main functions of the external tester have been moved onto the chip. Such practice is generally known as built- in self-test (BIST).

Test pattern generation belongs to a class of computationally difficult problems, referred to as NP-complete. Several approaches have been developed to handle test generation for relatively large combinational circuits in a reasonable time. Test generation for large sequential circuits remains, however, an unsolved problem, despite rapid increase of computational power. According to available test techniques can be classified into the following categories:

1. Functional testing, which relies on exercising the device under test (DUT) in its normal operational mode, and consequently, at its rated operational speed;
2. Testing for permanent structural faults (like stuck-at, stuck- open, bridging faults) that do not require the circuit to operate at rated speed during test;
3. Testing based on interactive fault analysis in which faults are derived from a simulation of the defect generation mechanisms in an integrated circuit (IC) (such faults tend to be permanent and do not require the circuit to be tested at rated speed);
4. Testing for delay faults that require the circuit to operate at rated speed during test;
5. Current measurement based testing techniques, which typically detect faulty circuits by measuring the current drawn by the circuit under different input conditions while the circuit is in the quiescent state.

Failures and Fault models

A failure is defined as an incorrect response in the behavior of the circuit.

According to there are two views of failures:

1. Physical/Design domain: defects (they produce a deviation from specification)

- On the device level: gate oxide shorts, metal-to-polysilicon shorts, cracks, seal leaks, dielectric breakdown, impurities, bent-broken leads, solder shorts and bonding.
- On the board level: missing component, wrong component, miss-oriented component, broken track, shorted tracks and open circuit.
- Incorrect design (functional defect).
- Wearout/environmental failures: temperature related, high humidity, vibration, electrical stress, crosstalk and radiation (alpha particles, neutron bombardment).

ISSN: 2057-5688

2. Logical domain: faults (structural faults). A fault is a model that represents the effect of a failure by means of the change that is produced in the system signal.

- Stuck-at faults: single, multiple.
- Bridging faults: AND, OR, non-feedback and feedback.
- Delay faults: gate and interconnect.

The oldest form of testing relies on a functional approach, where the main idea is to exercise the DUT in its normal operational mode. The main task of functional testing is to verify that the circuit operates according to its specifications. For functional testing, the same set of test vectors that was used by the designer for verification during the design phase can be used. Functional testing can cover a relatively large percentage of faults in an IC but the disadvantage of this technique is the large size of the test sequences needed to achieve high test quality. Using this approach alone for testing complex digital circuits is therefore notpractical. Structural fault model based techniques are the most investigated testing techniques. The earliest and the most well-known structural fault model is the single stuck-at (SSA) fault model (also called single stuck line (SSL) fault model), which assumes that the defect will cause a line in the circuit to behave as if it is permanently stuck at a logic value 0 (stuck-at-0) or 1 (stuck-at-1). The SSA model assumes that the design contains only one fault. However, with decreased device geometry and increased gate density on the chip, the likelihood is greater that more than one SSA fault can occur simultaneously and they may mask each other in such a way that the SSA test vectors cannot detect them. Therefore it may be necessary to assume explicitly multiple stuck-at faults as well.

The single stuck-at fault model became an industrial standard in 1959. Experiments have shown that this fault model can be very useful (providing relatively high defect coverage) and can be used even for identifying the presence of multiple faults which can mask each other's impact on the circuit behavior. The possibility to analyze the behavior of the circuit using Boolean algebra has contributed to research in this domain very much. There are several approaches to identify test vectors using purely Boolean-algebraic techniques, search algorithm based techniques or techniques based on the combination of the two. But there are also several problems related to the SSA fault model, which become more obvious with the growth of the size of an IC. The main problem lies on the fact that the computation process to identify tests can be extremely resource and time intensive and, additionally, the stuck-at fault model is not good at modeling certain failure modes of CMOS, the dominant IC manufacturing technology at the present time.

During recent years several other fault models (e.g. stuck-OPEN and bridging) have gained popularity but these fault models still cannot solve the problems with CMOS circuits. As a solution to these problems, two technologies have been proposed: Inductive fault analysis (IFA) and, more recently, inductive contamination analysis (ICA). These techniques present a closer relationship between physical defects and fault models. The analysis of a fault is based on analyzing the given manufacturing process and layout of a particular circuit. A completely different aspect of fault model based testing is testing for delay faults. An IC with delay faults operates correctly at sufficiently low speed, but fails at rated speed. Delay faults can be classified into gate delay faults (the delay fault is assumed to be lumped at some gate output) and path delay faults (the delay fault is the result of accumulation of small delays as a signal propagates along one or more paths in a circuit).

All methods mentioned above rely on voltage measurement during testing; but there are also techniques which are based on current measurement. These techniques are commonly referred as IDDQ test techniques. The technique is based on measuring the quiescent current and can detect some of the faults which are not detectable with other testing techniques (except exhaustive functional testing). IDDQ testing can be also used for reliability estimation. The disadvantage of this technique is the very slow testing process, which makes testing very expensive.

Test Pattern Generation

Test pattern generation is the process of determining the stimuli necessary to test a digital system. The simplest approach for combinational circuits is exhaustive testing where all possible input patterns will be applied, which means applying 2^n test patterns (where n is the number of inputs). Such large number of test patterns means that exhaustive testing is possible only with small combinational circuits. As an example, a circuit with 100 inputs needs already 2^{100}-10^{30} test patterns and is therefore practically infeasible. An alternative for exhaustive testing is pseudorandom testing, where test patterns are generated in pseudorandom manner. The cost of this type of test is considerably reduced but pseudorandom patterns cannot detect all possible faults and for so called random pattern-resistant faults we still need some type of deterministic tests. To overcome those

ISSN: 2057-5688

problems, several structural test generation techniques have been developed. In this case we assume that the elementary components are fault-free and only their interconnects are affected. This will reduce the number of test patterns to 2nin the case of the single stuck-at fault model.

There has been a lot of research in the area of test pattern generation and the current status is that test pattern generation for combinational circuits as well as for sequential circuits without global feedback is a solved problem and there exist commercial tools for it. Test pattern generation for complex sequential circuits remains still an unsolved problem (due to high complexity involving multiple time frames and other factors) and there is some skepticism about the possibility to have efficient commercial solutions available in the nearest future.

Test Application

As previously mentioned, hardware testing involves test pattern generation, discussed above, and test application. Test application can be performed either on-line or off-line. The former denotes a situation where testing is performed during normal operational mode and the latter when the circuit is not in normal operation. The primary interest of this thesis is off-line testing although some of the results can be applied also for on-line testing as well. Off-line tests can be generated either by the system itself or outside the chip and applied by using Automatic Test Equipment (ATE). With the emerging of sub-micron and deep sub-micron technologies, the ATE approach is becoming increasingly expensive, the quality of the tests and therefore also the quality of the device deteriorates, and time to market becomes unacceptably long. Therefore several methods have been developed to reduce the significance of external testers and to reduce the cost of the testing process, without compromising on quality. Those methods are known as design for testability (DFT) techniques. In the following, different DFT techniques are described.

Design for Testability

Test generation and application can be more efficient when testability is already considered and enhanced during the design phase. The aim of such an enhancement is to improve controllability and observability with minimal area and performance overhead. Controllability and observability together with predictability are the most important factors that determine the complexity of deriving a test set for a circuit. Controllability is the ability to establish a specific signal value at each node in a circuit by setting values on the circuit's inputs. Observability, on the other hand, is the ability to determine the signal value at any node in a circuit by controlling the circuit's inputs and observing its outputs. DFT techniques, used to improve a circuit's controllability and observability, can be divided into two major categories:

DFT techniques which are specific to one particular design (adhoc techniques) and cannot be generalized to cover different types of designs. Typical examples are test point insertion and design partitioning techniques. Systematic DFT techniques are techniques that are reusable and well defined (can be even standardized).

In the following sections some systematic DFT techniques are discussed.

Scan-Design

To cope with the problems caused by global feedback and complex sequential circuits, several different DFT techniques have been proposed. One of them is internal scan. The general idea behind internal scan is to break the feedback paths and to improve the controllability and observability of the memory elements by introducing an over-laid shift register called scan path. Despite the increase in fault coverage, there are some disadvantages with using scan techniques:

- Increase in silicon area,
- Larger number of pins needed,
- Increased power consumption,
- Increase in test application time,
- Decreased clock frequency.

There are two different types of scan-based techniques:

1. Full scan
2. Partial scan

In case of partial scan only a subset of the memory elements will be included in the scan path. The main reason for using partial scan is to decrease the cost and increase the speed of testing. In the case of complex chips or printed circuit boards (PCB) it is often useful for the purposes of testing and fault isolation to isolate one module from the others. This can be achieved by using boundary scan. Boundary scan is well defined and standardized (IEEE 1149.1 standard). Boundary scan targets manufacturing defects around the boundary of a device and the interconnects between devices. These are the regions most likely to be damaged during board assembly.

Built-InSelf-Test

As discussed earlier, the traditional form of off-line testing requires the use of ATEs. One of the problems, while using ATEs, is the growing disparity between the external bandwidth (ATE speed) and the internal one (internal frequency of the circuit under test). And as the importance of delay faults is increasing with newer technologies,

ISSN: 2057-5688

and the cost of test pattern generation as well as the volume of test data keep increasing with circuit size, alternative solutions are needed. One such solution is built-in self-test(BIST). The main idea behind a BIST approach is to eliminate the need for the external tester by integrating active test infrastructure onto the chip. A typical BIST architecture consists of a test pattern generator (TPG), usually implemented as a linear feedback shift register (LFSR), a test response analyzer (TRA), implemented as a multiple input shift register (MISR), and a BIST control unit (BCU), all implemented on the chip. This approach allows applying at-speed tests and eliminates the need for an external tester. Furthermore, the BIST approach is also one of the most appropriate techniques for testing complex SoCs, as every core in the system can be tested independently from the rest of the system. Equipping the cores with BIST features is especially preferable if the modules are not easily accessible externally, and it helps to protect intellectual property (IP) as less information about the core has to be disclosed.

There are two widely used BIST schemes: test-per-clock and test- per-scan. The test-per-scan scheme assumes that the design already has existing scan architecture. During the testing phase the TPG fills the scan chains which will apply their contents to the circuit under test (CUT). All scan outputs are connected to the multiple input signature register (MISR), which will perform signature, compaction. There are possibilities to speed up the test process by using multiple scan chains or by using a partial scan solution. An example of such an architecture is Self-Test Using MISR and Parallel Shift Register Sequence Generator (STUMPS).

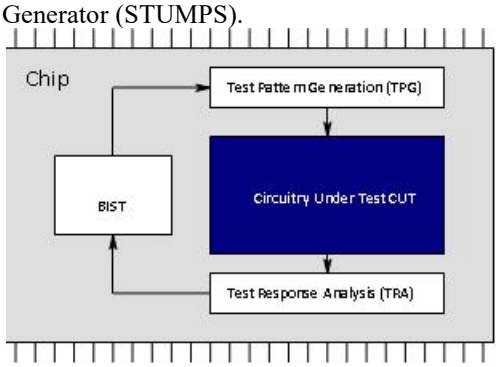

Figure: A typical BIST architecture

The test-per-clock scheme uses special registers that perform pattern generation and response evaluation. This approach allows to generate and to apply a new test pattern in each clock cycle. One of the first proposed test-per-clock architectures was the Built-In Logic Block Observer (BILBO), proposed, which is a register that can operate both as a test pattern generator and a signature analyzer. As the BIST approach does not require any external test equipment it can be used not only for production test, but also for field and maintenance test, to diagnose faults in field-replaceable units. Since the BIST technique is always implemented on the chip, using the same technology as the CUT, it scales very well with emerging technologies and can become one of the most important test technologies of the future.

Constraint Logic Programming

Most digital systems can be conceptually interpreted as a set of constraints, which is a mathematical formalization of relationships that hold in the system. In the context of test generation, there are two types of constraints: the system constraints and the test constraints. The system constraints describe the relationships between the system variables, which capture the system functionality and requirements. The test constraints describe the relationships between the system variables in order to generate tests for the system. Constraint solving can be viewed as a procedure to find a solution to satisfy the desired test constraints for a system, if such a solution exists. The easiest way for constraint solving is to enumerate all the possible values for the constraints and test if there exists a solution. Unfortunately enumeration methods are impractical in most cases. The problem of enumeration methods is that they only use the constraints in a passive manner, to test the result of applying values, rather than using them to construct values that will lead to a solution. There are lots of constraints solving strategies that make use of the types and number of constraints in order to speed up the solving process.

The backtracking strategy is a basic and important approach in constraint solving. Most constraint solvers such as CHIP, SICStus, etc., use the backtracking strategy as a basic method for constraint satisfaction. The search for a solution always involves a decision process. Whenever there are alternatives to solve a problem, one of them is chosen. If the selected decision leads to an inconsistency, backtracking is used in order to allow a systematic exploration of the complete space of possible solutions and recovery from the incorrect decision. Recovery involves restoring the state of the computation to the state existing before the incorrect decision. For example, there are two possible solutions for the problem in Figure. We first choose one of them, D1, as a decision and try it. In this case, D11 and D12 are alternative decisions for finding a solution with decision D1. We can select either D11 or D12 to try to find a solution. If decision D11 leads to an inconsistency,

INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE
ISSN: 2057-5688

it means that the decision {D1,D11} cannot find a solution for the given problem. So the system will recover from D11 and try another alternative decision, D12. If the decision D12 also fails, it will cause the first decision D1 to fail. The system cannot find a solution with the selected decision D1. So we will go back to try another decision, D2, by backtracking. As shown in Figure, the selected decision D21 succeeds. It will lead to a solution for the given problem and the decision {D2,D21} is a correct decision for the problem. It is obvious that the ordering of the variables has an impact on the searching time.

Problem

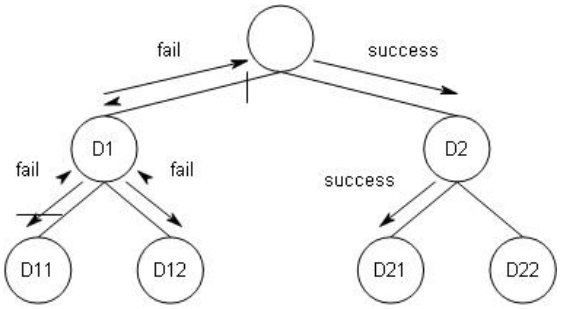

Figure: The backtracking strategy

Depending on the complexity of the problem, the search space, while using the previous search strategy, can become huge, and finding a solution is practically infeasible. Therefore, several heuristics have been developed that explore only part of the search space. Such is, for example, a search strategy which only spends a certain number of search cycles (credits) in each branch. If this credit is exhausted it goes back in the tree and chooses an alternative sub- tree high-up in the (unexplored) tree to further explore. By controlling the amount of credit which is provided, we can control the search quite well. However, this approach may not be able to find the (best) solution, as it explores the search space only partially.

3. PROPOSED SYSTEM ARCHITECURE

A 4 bit ALU has been designed to perform various arithmetic and logical operations on 4 bit data. The ALU has three selection bits (S0S1S0), one carry input bit (Cin), two 4-bit data(A and B) and an active low reset as input pins and it has four bit storage (F) and one bit storage (Cout) as output pins. The table given below describes the various operations performed by processor and its pseudo.

Table 1: Operations performed by ALU

S_2	S_1	S_0	C_{in}	OPERATION	DESCRIPTION
0	0	0	0	F=A	Transfer A(Maintain B=0000)
0	0	0	1	F=A+1	Increment A
0	0	1	0	F=A+B	Add b to a
0	0	1	1	F=A+B+1	Add B TO A With Carry
0	1	0	0	F=A+(NOT	Subtract with borrow
0	1	0	1	F=A+(NOT	Subtract
0	1	1	0	F=A-1	Decrement A
0	1	1	1	F=A	Transfer A(Maintain B=1111)
1	0	0	X	F=A | B	A OR B
1	0	1	X	F=A ^ B	A XOR B
1	1	0	X	F =A&B	A AND B
1	1	1	X	F= ~ A	Complement A

The processor instructions can be broadly divided into two classes. Arithmetic operation and Logical operation. At the end of execution of each instruction memory write operation is performed to store the result back to the address specified bythe instruction decoder.

As shown in adjoining figure, the LHS of RTL (Register Transfer Level) Schematic of Arithmetic and Logical unit has various inputs like two 4 bits ‘a’ & ‘b’ signals, 3 bits operation selection signal, carry input from previous execution if multiple fetch and execute cycles are required, etc. On the other hand the 4 bit result is available along with OVERFLOW flag.

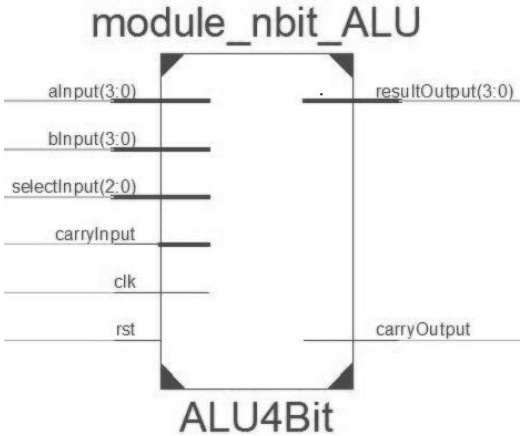

Figure: Pin diagram of 4 bit ALU

DESIGN OF 16X4 RAM

INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE
ISSN: 2057-5688

The RAM is required to store various data helpful for instruction execution purpose. It also contains the intermediate result for multiple fetch and executes operations. Term 16x4 corresponds that the memory module is 4-bit wide and 16-bitdeep. Therefore the 16x4 RAM stores maximum of 8 byte data.

We had used two dimensional array declaration to define memory blocks. The declaration by this fashion also helps in synthesis as Technology schematic will represents the four, 16x1 memory blocks. Whenever Data out is not used i.e., when memory is in write mode or if chip select is high, the Data out is set to "ZZZZ" (meaning nothing is driven onto bus).

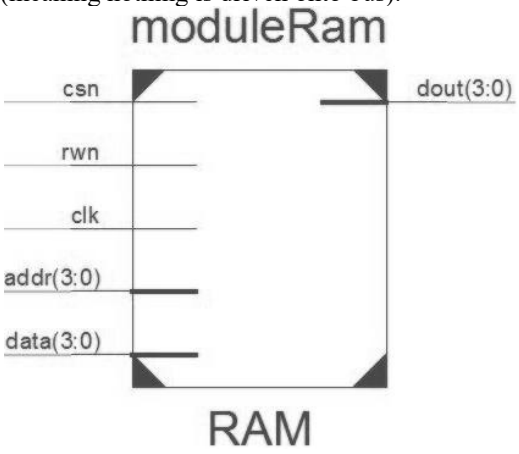

Figure: Pin diagram of 4 bit RAM

DESIGN OF INSTRUCTION DECODER

The instruction decoder can be thought of as a finite state machine which changes its state from the present state to next state on the rising edge of the clock signal. The instruction decoder module takes input compromising of a three bit opcode and two 4-bit operands. When asynchronous reset is high the decoder unit goes to init state. With the positive edge of every subsequent clock the unit cycles between Fetch, Execute and Load state. At each of these states the operations to be performed are as given below**:**

Fetch – Fetches data from the location specified by the first operand. The first operand specifies the reference by memory allocation. This data is store in operand a Input.

Execute – Execute the Instruction using ALU after setting all control lines depending on instruction. Select Input lines are set to the value of opcode, a Input is same as obtained in the Fetch state and b Input is an immediate data supplied to the instruction.

Load – After completion of ALU operation, result is contained in 4-bit storage (F) which is stored back into the location specified by first operand.

International Conference for Convergence of Technology – 2014.

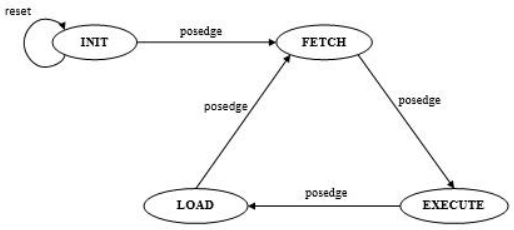

Figure: FSM design for instruction decoder

The figure depicts the RTL (Register Transfer Level) Schematic Pin-Diagram of Instruction Decoder, the LHS pins are Input ports to the instruction decoder. The operand 1 is reference by address value of memory, whereas the operand 2is reference by immediate value. The opcode is 3 bit long so that all the ALU operations can be enclosed. On the other hand on output side, a Input, bInput& select are provided to ALU, rest all are given to 16x4 RAM.

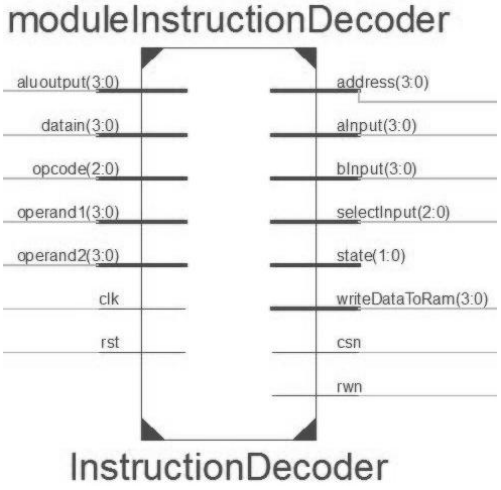

Figure: Pin diagram of Instruction Decoder

RESULTS

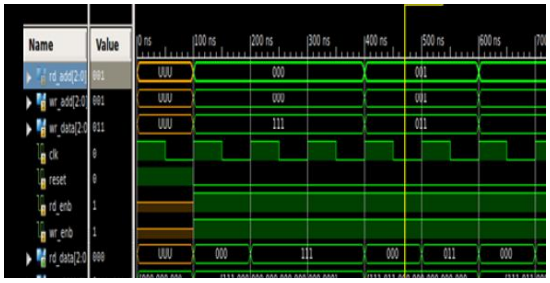

Figure: Simulation Result of RAM

INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE
ISSN: 2057-5688

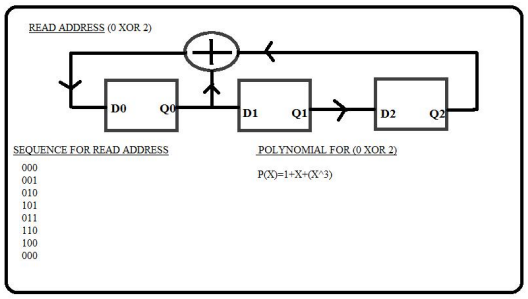

Figure: Architecture of LFSR1 (generate Read Address)

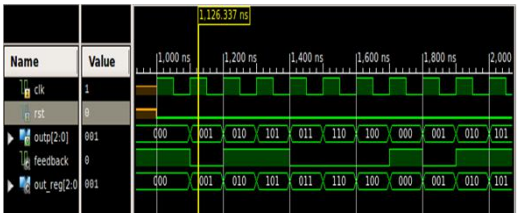

Figure: Simulation Result of LFSR1

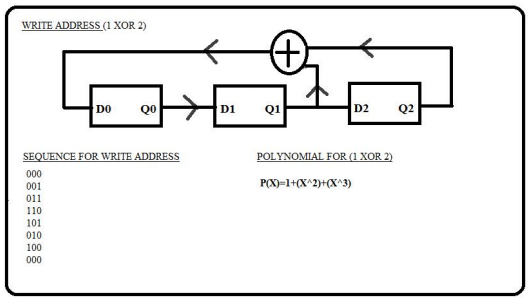

Figure: Architecture of LFSR2 (generate Write Address)

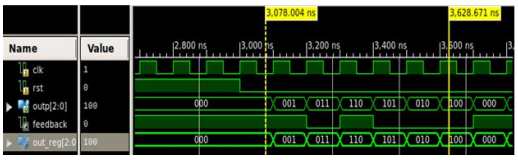

Figure: Simulation Result of LFSR2

CONCLUSION

We have designed and verified the functionality of basic four bit microprocessor using structural modeling which combines three elementary modules i.e. four bit ALU,16x4 RAM and Instruction Decoder. The processor design that we carried out is very basic but for understanding and developing design of new age complex processor architectures like multi-core design, the knowledge of basic design is inevitable. As a part of future enhancement to the architecture, various addition are possible like dual-core processor design, to include more variety of operations to be carried out by ALU, different assortment of DSP (Digital Signal Processing) functions like convolution, time shifting, Fast Fourier transform operations can be added. We can also include serial interface block so that RS232 port can be attached to processor.

REFERENCES

[1] N. Kavvadias, P. Neofotistos, S. Nikolaidis, C. A. Kosmatopoulos, and T. Laopoulos, "Measurements analysis of the software-related powerconsumption in microprocessors," IEEE Trans. Instrum. Meas., vol. 53,no. 4, pp. 1106–1112, Aug. 2004.

[2] J. Teifel and R. Manohar, "An Asynchronous Dataflow FPGA Architecture," IEEE Transactions on Computers, vol. 53, no. 11, pp.1376–1392, 2004.

[3] Balpande, R.S. and Keote, R.S., "Design of FPGA based Instruction Fetch & Decode Module of 32-bit RISC (MIPS) Processor," in Proc.of International Conference on Communication Systems and Network Technologies, pp. 409-413, 2011.

[4] Joaquín Olivares, et all, "Teaching Microprocessor Design Using FPGAs", IEEE EDUCON Education Engineering 2010, April 2009,Spain.

[5] Xilinx® Inc., PicoBlaze 8-bit Embedded Microcontroller User Guide, Xilinx® 2005

[6] Wang Min, The Principle of Computer Organization, Electronics Industry Publishing House, Beijing, 2003

[7] Kilts, Steve, "Advanced FPGA Design.", John Wiley & Sons, 2007.

[8] M. Morris Mano, "Digital Logic & Computer Design," Pearson Edition, 1979

[9] Samir Palnitkar, "Verilog HDL A guide to Digital Design and Synthesis," SunSoft Press, 2003

[10] Rajeev Madhavan, "Quick Reference for Verilog HDL," Ambit DesignSystem, 2001

[11] Eli Sternheim, Rajvir Singh, Rajeev Madhavan, YatinTrivedi, "Digital Design and Synthesis with Verilog HDL". Automata PubCo; 1993.