International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 10, October 2015, PP 37-43
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

# An Area Efficient RISC Architecture using ALU

S.V.Sridevi[1], Dr.G.Kanaka Durga[2]

[1](PG Scholar), Department of ECE, MVSR Engineering College, Hyderabad, India
[2](HOD), Department of IT, MVSR Engineering College, Hyderabad, India

## ABSTRACT

One of the key components of memory BIST is the Address Generator (AG). In order to detect speed-related faults, the address generator has to generate different address sequences to allow for appropriate address transitions. Its complexity is a major design issue since it requires large area and limits the BIST speed. For generating the address for Radom Accesses Memory (RAM) it requires extra hardware. An Arithmetic and Logic Unit (ALU) is used for generating the address for RAM as it reduces the hardware overhead. The primary issue in the design of any processor is to have an area efficient ALU. In order to reduce the area of the Microprocessor without Interlock Pipeline Stage (MIPS) processor and improve the delay, the existing ALU is replaced by an ALU with multiplexer architecture.

ALU based address generation are used in many applications like Smartcard Chips, Digital Signal Processors and Microcontrollers etc. This technique is executed using Verilog HDL. Simulation is done to verify the functionality and synthesis is done to get the Netlist. The code is simulated and synthesized using Xilinx ISE 13.2.

**Keywords:** Memory BIST, Address Generation, ALU-based implementation

## INTRODUCTION

Built-In Self-Test (BIST) has become a standard industrial practice for testing memories since memory cores constitute a major part of the die area. One of the key components of memory BIST is the address generator (AG). In order to detect speed-related faults, the address generator has to generate different address sequences to allow for appropriate address transitions. Its complexity is a major design issue, since it requires large area and limits the BIST speed. For memory testing, the address generator has to generate different Counting Methods (CMs) since each counting method has its own detection. The main purpose of the address generator is to generate the test patterns. Such address generators are Linear Feedback Shift Registers (LFSRs), Pipeline LFSR, Binary Up down Counter and Gray Code up down counter.

For memory testing, the address generator has to generate different Counting Methods (CMs) as each counting method has its own detection capability. Hence, the detection of different memory fault classes requires different address generators [1, 2, 3, and 4]. Table 1 highlights two of the most well-established counting methods by giving an example for a memory with N=4 words. The Linear (LI) counting method specifies the address sequence: 0, 1, 2, 3... N-1 when going Up; and N-1... 3, 2, 1, 0 when going Down. The Linear counting method is used for detecting single-cell and coupling faults. The Address Complement (AC) counting method specifies an address sequence: 0000, **1111**, 0001, **1110**, 0010, **1101**, etc. Each bold address is the one’s complement of the previous address, as shown in

**Address for correspondence:*

sridevisura123@gmail.com

column denoted 'AC' in Table 1. The AC counting method stresses the address decoders, because all address bits switch upon address transitions. This causes lots of noise, a large power surge, and maximal delay and is used for detecting speed-related faults.

**Table1.** *Linear and Address Complement counting methods*

| Step | LI | AC |
|---|---|---|
| 0 | 00000000 | 00000000 |
| 1 | 00000001 | **11111111** |
| 2 | 00000010 | 00000001 |
| 3 | 00000011 | **11111110** |
| 4 | 00000100 | 00000010 |
| 5 | 00000101 | **11111101** |
| 6 | 00000110 | 00000011 |
| 7 | 00000111 | **11111100** |
| 8 | 00001000 | 00000100 |
| 9 | 00001001 | **11111011** |
| 10 | 00001010 | 00000101 |
| 11 | 00001011 | **11111010** |
| 12 | 00001100 | 00000110 |
| 13 | 00001101 | **11111001** |
| 14 | 00001110 | 00000111 |
| 15 | 00001111 | **11111000** |

An address generator based on an ALU is used for memory BIST in this paper. ALU modules commonly exist in current circuits; moreover, the outputs of ALU modules drive the address input of RAM modules, as shown in the elementary schematic diagram of the MIPS architecture of Figure1 [5]. Therefore, the utilization of the existing ALUs as address generators for the testing of RAMs may drive down the hardware overhead.

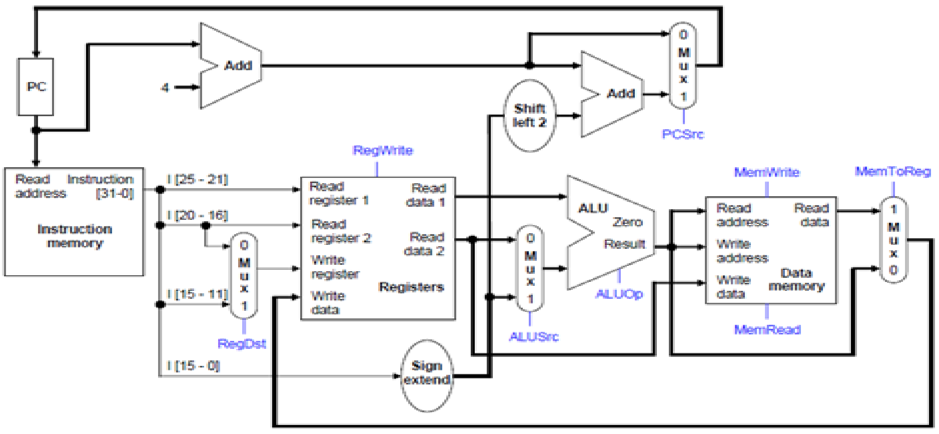


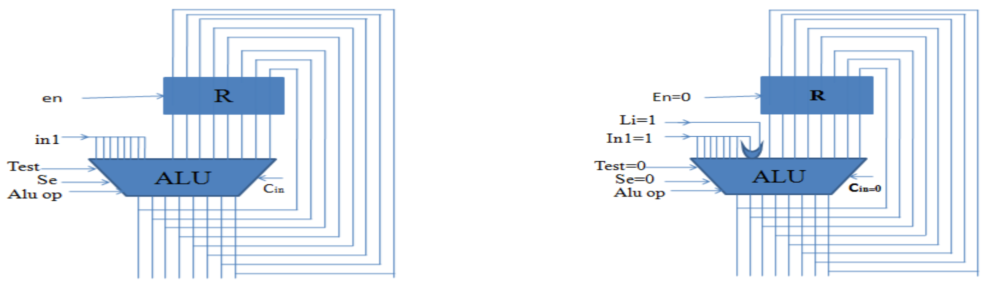
**Figure1.** *Basic diagram of the MIPS architecture*

## PREVIOUS WORK

An ALU based address generator is proposed; which assumes an accumulator-like structure, like the one shown in Figure 2(a).

In computing, an Arithmetic and Logic Unit (ALU) is a digital circuit that performs arithmetic and logical operations. The ALU is divided into two different units that perform two different operations. The Arithmetic unit performs the arithmetic operations such as addition, subtraction, multiplication and division whereas the logic unit performs the bit-wise AND, OR, XOR, NOT, NAND, NOR.

There will be three data inputs for the ALU block that is given by Data in, Cin and from the Register (R) block. The select signal to ALU block is provided by the Add/Sub that performs the required operation. The block diagram for the ALU based address complement and combined linear and address complement are shown in Figure 2.



(a) (b)

**Figure2.** *Generation of the counting methods (a) Address Complement (b) Combined Linear and AC*

For the address complement counting method, the signal denoted Li is constantly 0. The operation of this scheme utilize the following signals: the 'in-signal' is one input of the ALU, the selection decides on the addition or subtraction to be performed by the ALU, the en signal enables the Register (R) to capture the ALU output. The Cin is the carry input to the ALU. The Address Complement counting sequence based on input signals generates a sequence and is shown in the below Table 2.

**Table2.** *Address Complement Counting Method.*

| Cycle# | In | Se | R | En | Cin | ALU |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 00000000 | 1 | 0 | 00000000 |
| 1 | 1 | 0 | 00000000 | 0 | 0 | 11111111 |
| 2 | 0 | 1 | 00000000 | 1 | 1 | 00000001 |
| 3 | 1 | 0 | 00000001 | 0 | 0 | 11111110 |
| 4 | 0 | 1 | 00000001 | 1 | 1 | 00000010 |
| 5 | 1 | 0 | 00000010 | 0 | 0 | 11111101 |
| 6 | 0 | 1 | 00000010 | 1 | 1 | 00000011 |
| 7 | 1 | 0 | 00000011 | 0 | 0 | 11111100 |
| 8 | 0 | 1 | 00000011 | 1 | 1 | 00000100 |
| 9 | 1 | 0 | 00000100 | 0 | 0 | 11111011 |
| 10 | 0 | 1 | 00000100 | 1 | 1 | 00000101 |
| 11 | 1 | 0 | 00000101 | 0 | 0 | 11111010 |
| 12 | 0 | 1 | 00000101 | 1 | 1 | 00000110 |
| 13 | 1 | 0 | 00000110 | 0 | 0 | 11111001 |
| 14 | 0 | 1 | 00000110 | 1 | 1 | 00000111 |
| 15 | 1 | 0 | 00000111 | 0 | 0 | 11111000 |

For the case of the combined Linear or Address Complement address generator shown in Figure 2(b), when linear count is required the Li or Ac signal enables the low order of the ALU to be constantly to 1, the in signal is constantly 0, the en signal is enabled in every cycle and the $C_{in}$ signal is constantly 0.The selection is 0 or 1 depending on whether up or down linear count is required.

## Existing ALU design

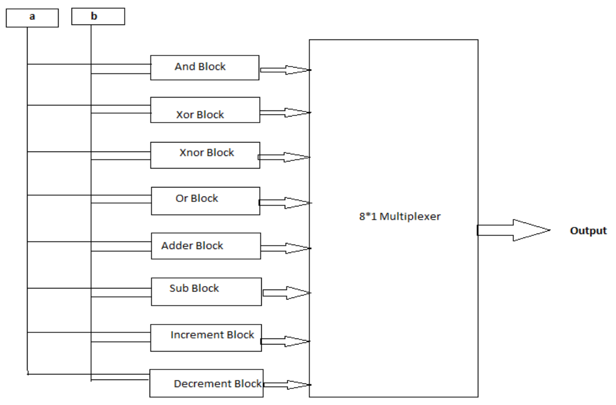


**Figure 3.** *Existing ALU Design*

The ALU has two inputs a and b. Every logic is written separately with a and b as inputs. The output of each block is given to the 8*1 multiplexers shown in Figure 3. Depending upon the select signal, one of the logic is selected and is passed to the output. In this ALU architecture, all the operations are performed individually and only one is selected depending upon the use. The drawback of this ALU is as each block operation is performed separately it takes more area and also it increases the delay.

## IMPLEMENTATION OF ADDRESS GENERATOR USING ALU WITH MUX DESIGN

The ALU using Multiplexers architecture is implemented in MIPS processor to reduce the area and delay. This ALU is also implemented in BIST circuit to test the memory.

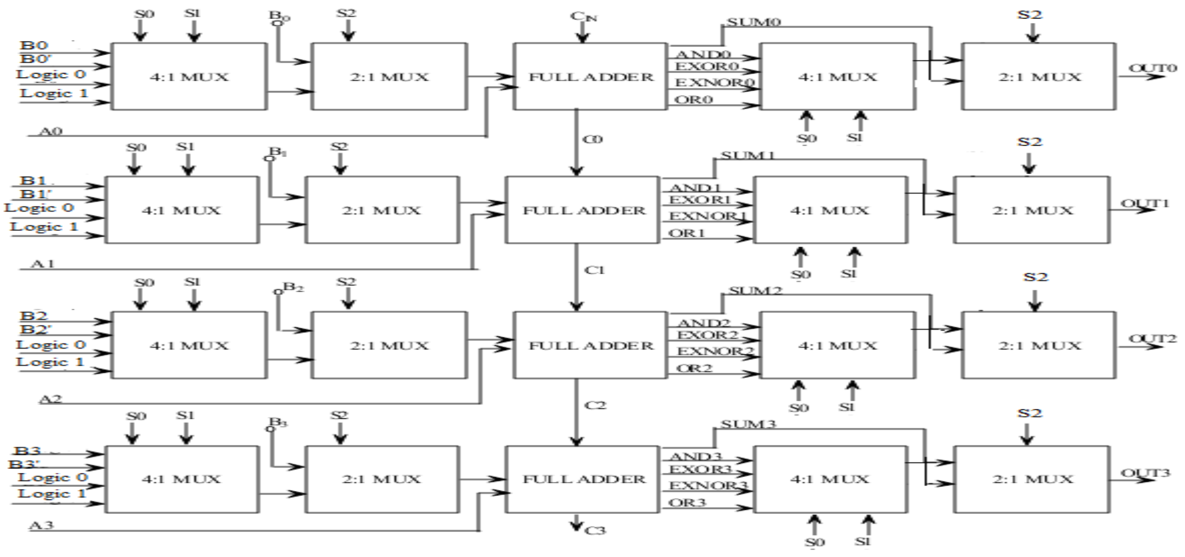


**Figure 4.** *4-bit Arithmetic and Logic Unit*

ALU is designed by using 4x1 multiplexer, 2x1 multiplexer and Full adder. The input and output sections consist of 4x1 and 2x1 multiplexers and the main logic is implemented by using full adder [6]. A set of three select signals have been used in the design to determine the operation being performed and the inputs and outputs being selected. Figure 4 shows the block diagram of 4-bit ALU where first stage to fourth stage is cascaded with the carry bit.

Table 3 shows the truth table for the operations performed by the ALU based on the status of the select signal.

**Table 3.** *Logic Functions of ALU*

| Selection Lines | | | Operations |
|---|---|---|---|
| S2 | S1 | S0 | |
| 0 | 0 | 0 | AND |
| 0 | 0 | 1 | EXOR |
| 0 | 1 | 0 | EXNOR |
| 0 | 1 | 1 | OR |
| 1 | 0 | 0 | ADDITION |
| 1 | 0 | 1 | SUBSTRACTION |
| 1 | 1 | 0 | INCREMENT |
| 1 | 1 | 1 | DECREMENT |

In the ALU shown in Figure 5, the addition and subtraction operation are performed by the circuit itself. It does not require additional blocks to perform operation so it takes less area and delay compared to the previous model shown in the Figure 4. So the ALU in the Figure 5 is implemented in the Microprocessor without Interlock Pipeline Stage (MIPS) processor. By modifying the ALU in MIPS the overall area and delay is reduced to 86.11% and 15.841% respectively.

## ARCHITECTURE DESCRIPTION OF MEMORY BIST

The Test Pattern Generator block in the BIST architecture is shown in the below Figure 5. This provides the address for the memory device that has to be tested. The data has to be stored in the memory location that is generated by the address generator of the memory [7]. The ALU with multiplexer architecture is used in memory BIST.

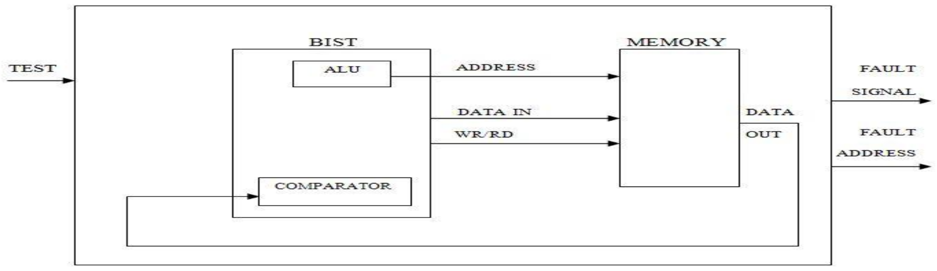


**Figure 5.** *ALU Based Memory Test block diagram*

The above Figure 6 explains the complete operation. The BIST contain Test pattern generator as well as the output response analyzer. As seen earlier the ALU from the processor is required to generate the linear addresses for the data to be stored in the memory hence the TPG is shown as the ALU module. The ALU with multiplexer architecture is used in MBIST.

The Memory block has three inputs i.e. Address, Data in and control signals. The Address is given by the ALU. The control signal is active high, when the control signal is high it performs write operation and when it goes low it performs the read operation. During write operation the data to be stored is sent to the memory from the Data in and the location where it has to be stored is specified by the Address line. During read operation the location from which the data has to be read is specified in the Address line. The Data is read out from the Data out.

The Test signal is applied to the main block when Test is low the ALU performs normal operation and when Test is high then the ALU provides the Address to the Memory. The Data out line is connected to the Comparator in the BIST controller. The Comparator compares the data that is read. If the data doesn't match then the fault signal goes high and fault address line gives out the Faulty address location in the memory. The outputs of ALU modules drive the address input of RAM modules, as shown in the elementary schematic diagram of the MIPS [8] (Microprocessor without Interlock Pipeline Stage) architecture. Therefore, the utilization of the ALU with multiplexer architecture as address generators for the testing of RAMs will reduce the area.

## SIMULATION RESULTS AND ANALYSIS

This section describes performance of the ALU with multiplexers architecture using Xilinx tool. The simulated output of ALU, Linear address generator, Complement address generator and Fault detection are shown in Figure 6, Figure 7, Figure 8 and Figure 9. The area and delay of the ALU is listed in Table 4.

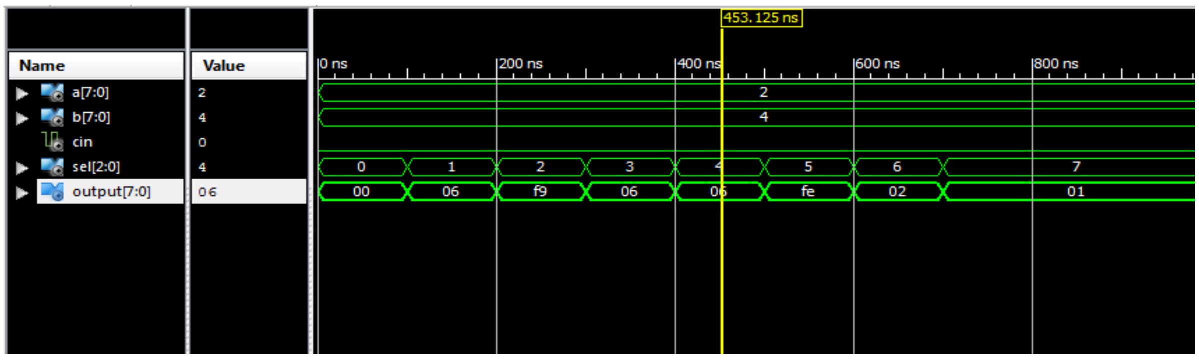


**Figure 6.** *Simulated Output of ALU with Multiplexers*

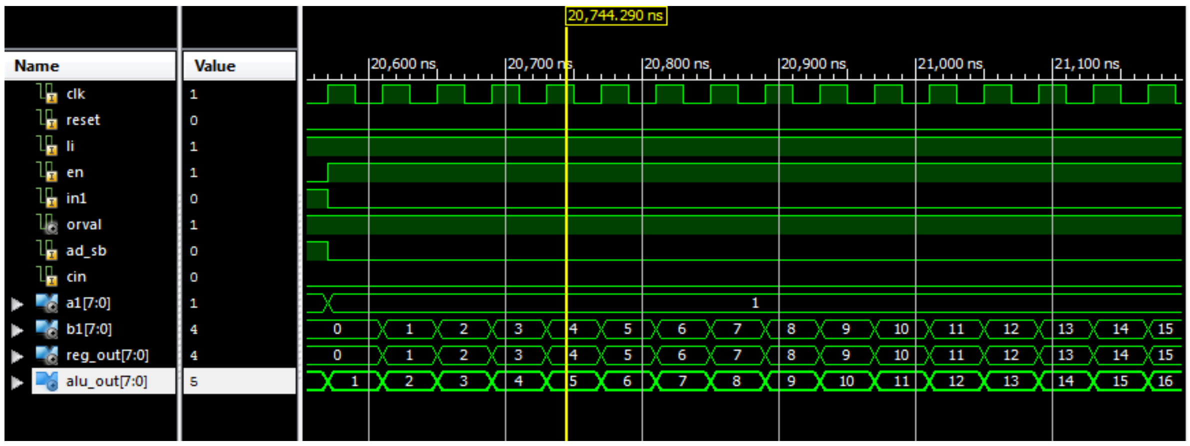


**Figure 7.** *Simulated Output of Linear Address Generator*

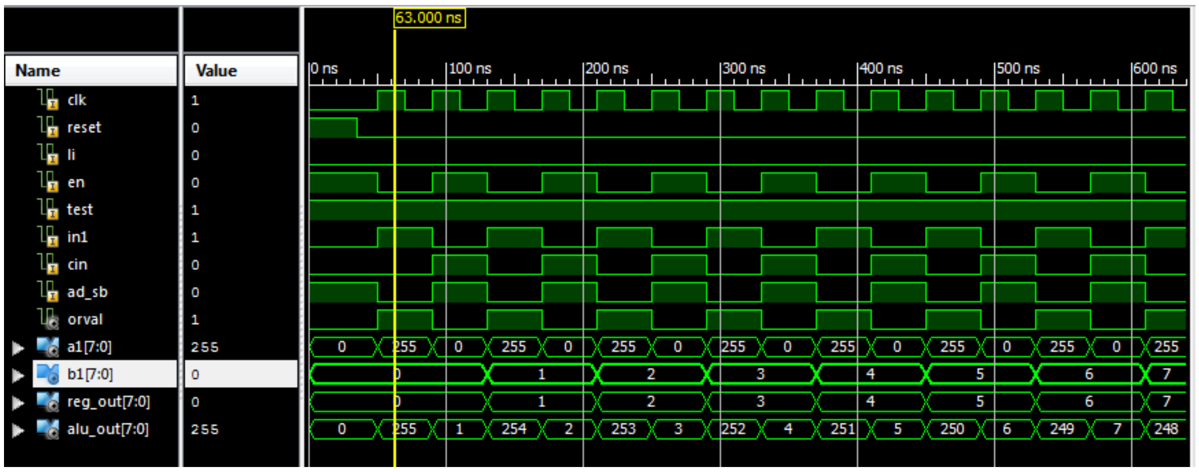


**Figure 8.** *Simulated Output of Address Complement Generator*

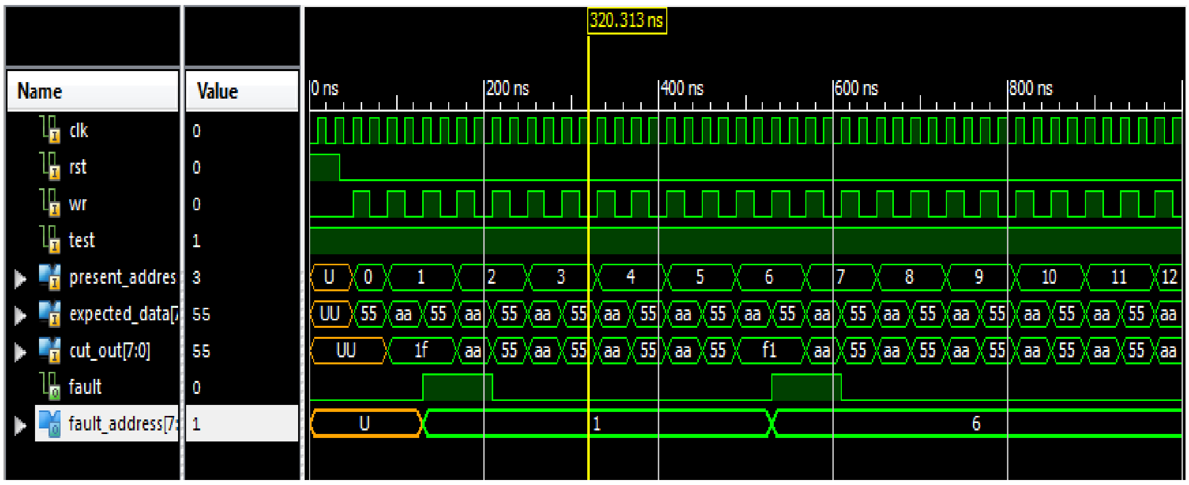


**Figure 9.** *Simulated Output of Fault Detection*

Figure 6 is the output of ALU, when inputs a=2 and b=4 depending upon the select signals it performs AND, XOR, XNOR, OR, Addition, Subtraction and Decrement. Figure 7 is the Linear Address Generator, when inputs a=1, Li=1, in=0, en=1, cin=0, output is generated in the sequence 0, 1,2, 3,4,…and so on. Figure 8 is the Address Complement Generator, when in=0, adsub=1, cin=0,output is 0 and when in=1,adsub=0, cin=0,output is 255. Figure 9 is the Fault Detection, when cut_out=UU, expected data=UU, fault is 0 and when cut_out=aa,expected data=1f,fault is 1.

Table 4. *Area and delay of ALU and ALU with Multiplexers in MIPS processors*

| Types | Number of slices | | Number of 4 input LUTs | | Number of bonded IOBs | | Delay |
|---|---|---|---|---|---|---|---|
| | Used | Available | Used | Available | Used | Available | |
| ALU | 64 | 4656 | 121 | 9312 | 28 | 232 | 16.6 |
| ALU with Mux Architecture | 23 | 4656 | 42 | 9312 | 28 | 232 | 14.7 |
| MIPS with ALU | 1945 | 4656 | 1713 | 9312 | 21 | 232 | 22.3 |
| MIPS using ALU with Mux Architecture | 270 | 4656 | 534 | 9312 | 21 | 232 | 18.8 |

## CONCLUSIONS

In this work ALU module is utilized for the address generation for memory BIST. The ALU based Memory Test in MIPS Architecture is coded and area and delay are verified through simulation. The ALU in MIPS architecture is replaced with ALU with multiplexers architecture and are compared. The results show that in the 8-bit ALU with multiplexers architecture the area and delay are significantly reduced by 64.06% and 11.65% respectively. There is a maximum area reduction in this design. The modified ALU with multiplexer architectures is therefore area efficient and high speed design.

## REFERENCES

[1] L. Dilillo, et.al, "Dynamic read destructive fault in embedded-SRAMs: analysis and march test solution", Proceedings Ninth IEEE European Test Symposium, pp. 140-145, 2004.

[2] S. Hamdioui, A. van de Goor and M. Rodgers, "Memory Fault Modeling Trends: A Case Study', Journal of Electronic Testing: Theory and Applications (JETTA)", Vol. 20, Nr. 3, pp. 245-255, 2004.

[3] M. Klaus and A. J. van de Goor, "Tests for Resistive and Capacitive Defects in Address Decoders", Proc. of the 10th Asian Test Symposium, pp. 31-36, 2001.

[4] T. Powell, et.al, "Chasing subtle embedded RAM defects for nanometer technologies", Proc. of Int. Test Conference, pp. 860, 2005.

[5] D. Paterson and J. Hennessy, "Computer Organization and Design, Fourth Edition: The Hardware/Software Interface", the Morgan Kaufmann Series in Computer Architecture and Design.

[6] Vivechan Dubey and Ravimohan "An Arithmetic and Logic Unit optimized for area and power" IEEE, pp. 330-334, 2014.

[7] Sonal Sharma, Vishal Moyal "AREA OPTIMIZED FSM BASED BIST" International Journal of Engineering Sciences & Management, Vol. 2, Issue 2: April-June: 2012.

[8] A.J. van de Goor, H. Kukner, S. Hamdioui, "Optimizing Memory BIST Address Generator Implementations" 6th International conference on Design & Technology of Integrated Systems in Nanoscale, April 2011.