IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 4, April 2015.
www.ijiset.com
ISSN 2348 – 7968

A 4-bit Arithmetic and Logical Unit with fault detection capability using an informal testing process and tested using CPLD EPM7128SLC84-15

[1]Abhishek Singh, [2]Mohd. Arif, [3]Kalpita Agrawal, [4]Anshita Gupta

[1]abhisheksingh@ggits.org, [2]arif@ggits.org, [3]agrawalkalpita@gmail.com, [4]anshi4619@gmail.com

Abstract: An arithmetic and logical unit is the most important unit in any processing system. ALU finds its utility in digital system, microprocessor, microcontroller, DSP, and, communication system. In this paper we have implemented an ALU and have made it fault detection capable using a very simple process known as Ad-hoc testing process. Altera Quartus II.0 simulator has been used and implementation has been done using CPLD EPM7128SLC84-15.

Keywords: Ad-hoc, ALU, CPLD EPM7128SLC84-15, Fault detection, Verilog**.**

1.1 Introduction

Even the simplest microprocessors or central processing unit (CPU) of a computer has an arithmetic logic unit (ALU). ALU is a digital circuit that performs many arithmetic and logic operations. The purpose of using ALU for handling short but yet frequent computational operations is to speed up computer processor and increase efficiency. Thus, ALU must be designed to perform micro operations. A micro operation is an elementary operation performed with the data stored in registers. Complex operations can be implemented but will result in an expensive ALU that will dissipate power as well as space of the processor.

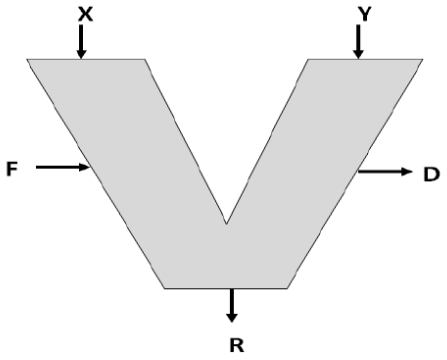

Fig.1.1A typical schematic symbol for an ALU: X&Y are the data (registers): R is the output; F is the operand (instructions) from the central unit; D is an output status.

The number of micro operations implemented in an ALU affects the number of selections signals required. If we have 2m micro operations we will need m select inputs from control unit. The word size of the ALU must confirm to the word size of the digital circuit. Block diagram of an ALU is given in figure 1.2.

1.1.1 Ad-hoc Process

Ad hoc testing is a commonly used term for software testing performed without planning and documentation (but can be applied to early scientific experimental studies).The tests are intended to be run only once, unless a defect is discovered. Ad hoc testing is the least formal test method. As such, it has been criticized because it is not structured and hence defects found using this method may be harder to reproduce (since there are no written test cases) [6]. However, the strength of ad hoc testing is that important defects can be found quickly.It is performed by improvisation: the tester seeks to find bugs by any means that seem appropriate

ISSN 2348 – 7968

[6]. Ad hoc testing can be seen as a light version of error guessing, which itself is a light version of exploratory testing.

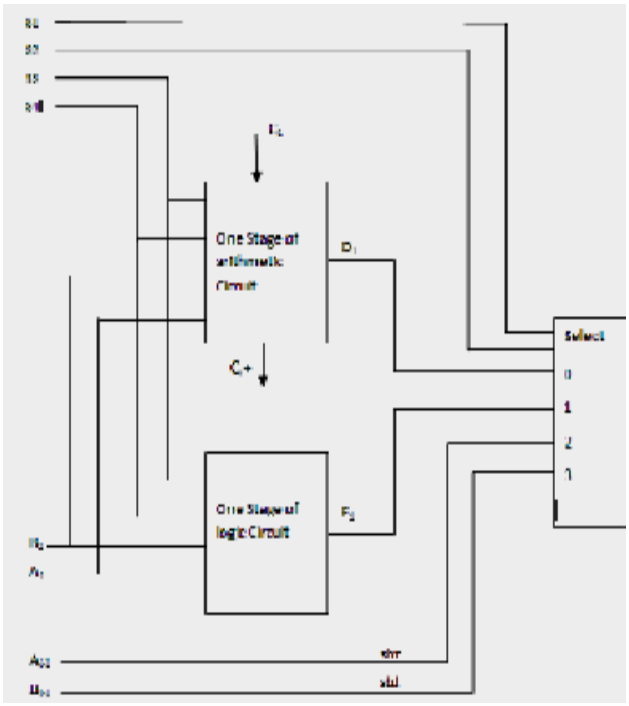

Figure 1.2 One stage of 4-bit ALU

1.2 Working and Description

The first aim is to design a 4-bit ALU which is executing 5 different operations whose sequence is based on a control signal. A Verilog code is shown below for module ALU.

```
modulealu(a,b,cin,alu,carry,ctl);
input [3:0] a,b;        // port A,B
input  cin ;           // carry input from carry flag
register
output [6:0] alu;       // the result
output carry;          // carry output
input [2:0] ctl ;       // functionality control for ALU
wire [7:0] result;      // ALU result

assign result = alu_out(a,b,cin,ctl);
assignalu    = result[6:0];
assign carry  = result[7] ;

function [7:0] alu_out;
input  [3:0] a,b ;
inputcin ;
input  [2:0] ctl ;
case ( ctl )

3'b000: alu_out=incr(a);// Increment data on port a
 3'b001: alu_out=decr(a);// Decrement data on port a
3'b010: alu_out=addrc(a,b,cin);// ADD with CARRY
3'b011: alu_out= grtr(a,b); //Greater of a and b
3'b100: alu_out= smlr(a,b); //Smaller of a and b
default : begin
                alu_out=8'bxxxxxxxx;
  end
          endcase
endfunction
```

```
// Increment function //
function [7:0] incr;
    input [3:0] a;
    begin
    if(a==4'b1111)
    begin
    incr={4'b0111,a}+1'b1;
    end
    else
    begin
    incr=a+1'b1;
    end
    end
endfunction
```

```
// Decrement function //
function [7:0] decr;
    input [3:0] a;
    decr=a - 1'b1;
endfunction
```

```
// Add with carry function //
function [7:0] addrc;
    input [3:0] a,b;
    input cin;
    begin
    if((a+b+cin)>=8'b00010000)
    begin
    addrc={4'b0111,a}+{4'b0000,b}+cin;
    end
    else
    begin
    addrc=a+b+cin;
    end
    end
endfunction
```

```
// Greater of two no. function //
function [7:0] grtr;
    input [3:0]a,b;
    if (a<b)
    begin
    grtr = b;
    end
    else
    begin
    grtr =a;
    end
endfunction
```

```
// Smaller of two no. function //
function [7:0] smlr;
    input [3:0]a,b;
    if (a<b)
    begin
    smlr = a;
    end
    else
    begin
    smlr = b;
    end
endfunction
endmodule
```

Figure 1.3 Verilog codes for Increment, Decrement, Addition with carry, Greater than and less than operations

3inputs a, b, cin are 4 bits, control input ctl is 3 bit input and output alu is a 7 bit output. The input and output waveforms for these operations have been shown below using Altera Quartus II.0 version 5.0. Version 5.0 has been selected because it supports the CPLD EPM7128SLC84-15.

ISSN 2348 – 7968

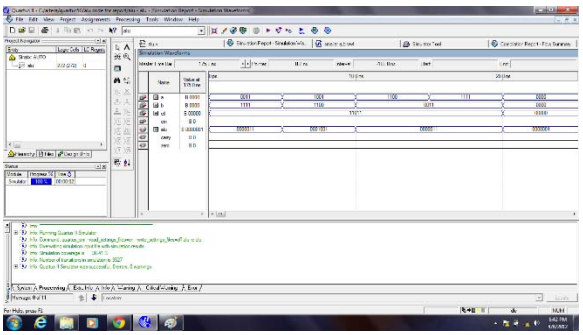

Figure 1.4 I/O waveform of 'less than' operation

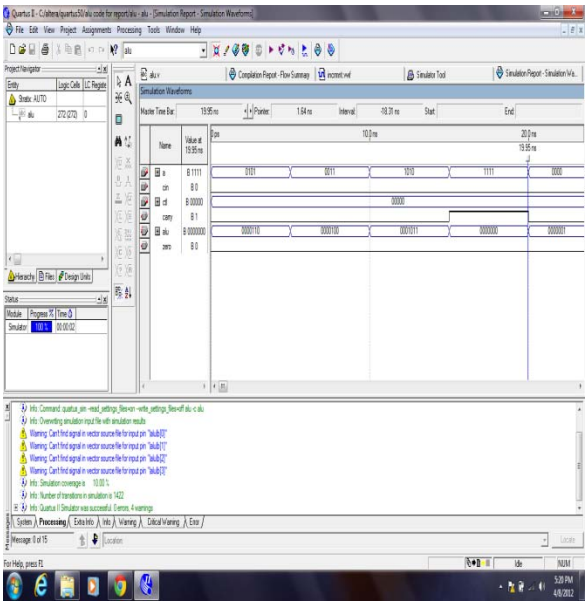

Figure 1.5 Increment 'a' simulation result

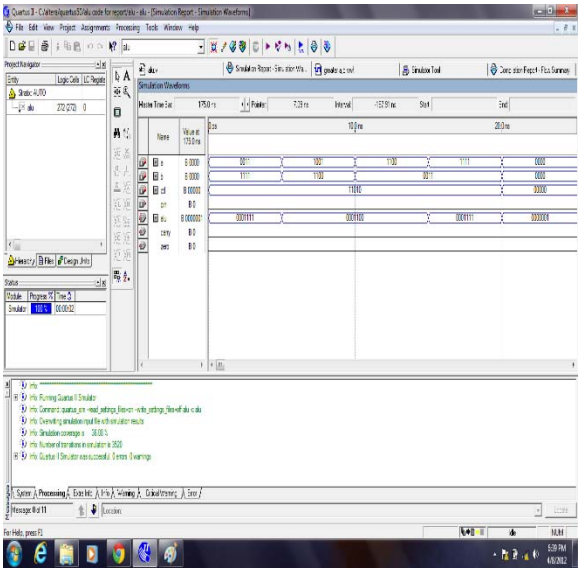

Figure 1.6 Greater than operation simulation result

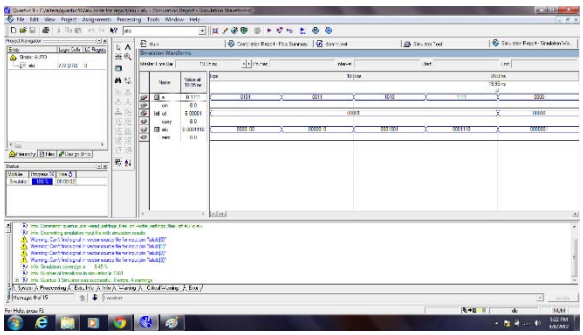

Figure 1.7Decrement 'a' simulation result

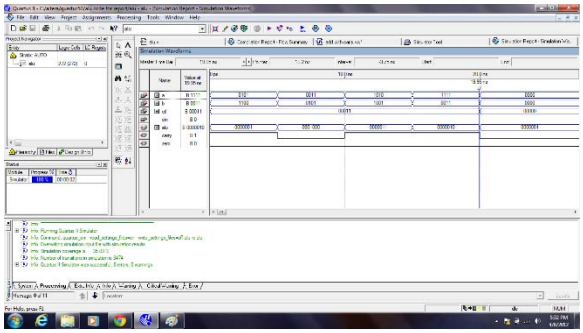

Figure 1.8 I/O waveform of Add with carry operation

The second step is now to implement a faulty ALU. We are introducing a fault in one function. We are introducing Stuck at Zero fault in Smaller than function. This means that when Smaller Than function is executed, whatever input is less, it will be generated at the output, but, since, we have stuck the output pin to zero, the Smaller Than function will not be executed. The input and output waveforms have been shown along with the faulty function.

```
// Smaller of two no. function //
function [7:0] smlr;
    input [3:0]a,b;
    if (a<b)
    begin
    smlr = a;
    end
    else
    begin
    smlr = 7'b0000000;
    end
endfunction
endfunction
```

Figure 1.9 Code showing that smaller function has been set to stuck-at-0 fault

ISSN 2348 – 7968

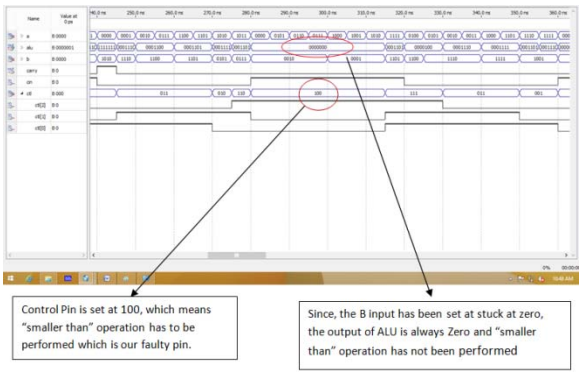

Figure 1.10 I/O waveform of a faulty ALU system

Now the Ad-hoc process has to be implemented for making this ALU fault detection enabled, which means that the fault that we have introduced must be detected otherwise a false output will be executed. For this purpose we are *XORing*the ALU and faulty ALU. The concept being that, the output of XOR is 0 if the inputs are same and 1 if the outputs are different. If ALU and Faulty ALU's outputs will be same the output will be zero. But since, we have introduced a fault the output has to be 1. This is shown in the figure below. The RTL view for this circuit has been shown below.

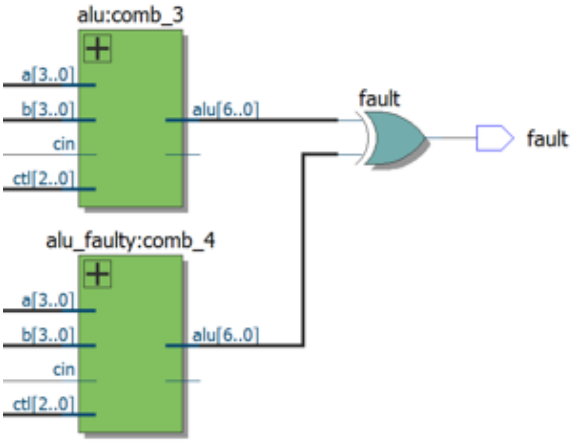

Figure 1.11 ALU and Faulty ALU has been XORed to check for faulty output

The input and output waveforms for this circuit has been shown below. The simulation has been performed using Altera Quartus II.

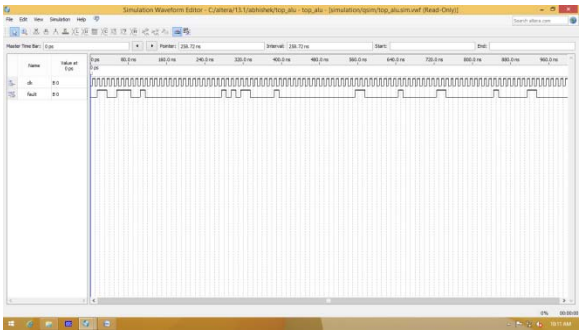

Figure 1.12 I/O waveforms showing a 1 at the output, which shows that there is a fault in the output.

This shows that our ALU can detect the fault present. The final process is to check the operation using CPLD.

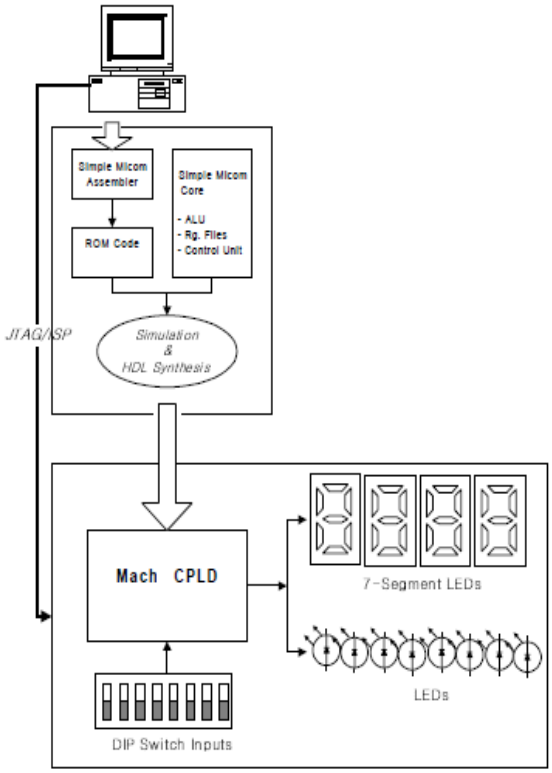

Figure 1.13 Overview of interfacing the Software with Hardware, i.e., downloading the module on CPLD machine

We are using a CPLD of Max 7000S series, EPM7128SLC84-15.

ISSN 2348 – 7968

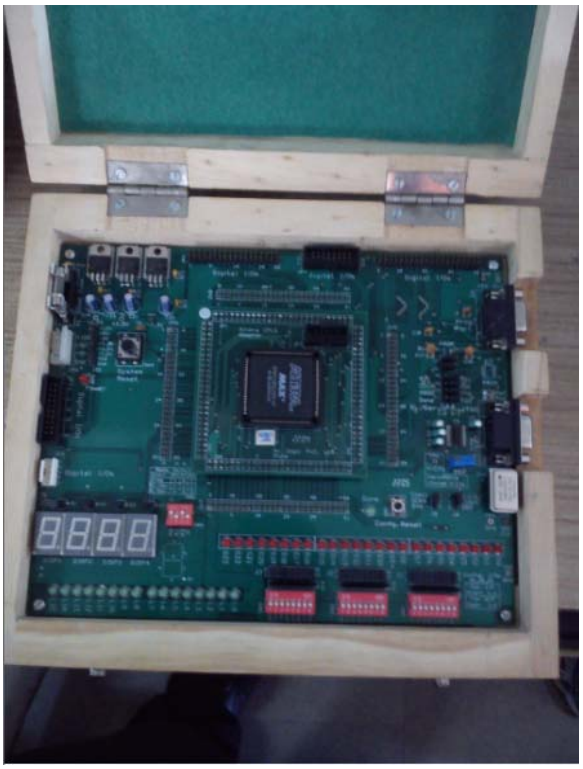

Figure 1.13 Prototype board of CPLD EPM7128SLC84-15

The operation has been verified using CPLD also.

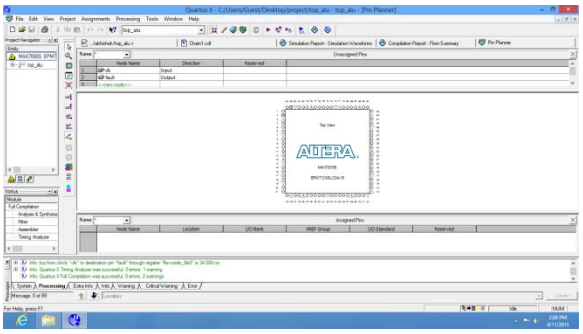

Figure 1.14Pin planner of CPLD EPM7128SLC84-15

Figure 1.15Daughter board of CPLD EPM7128SLC84-15

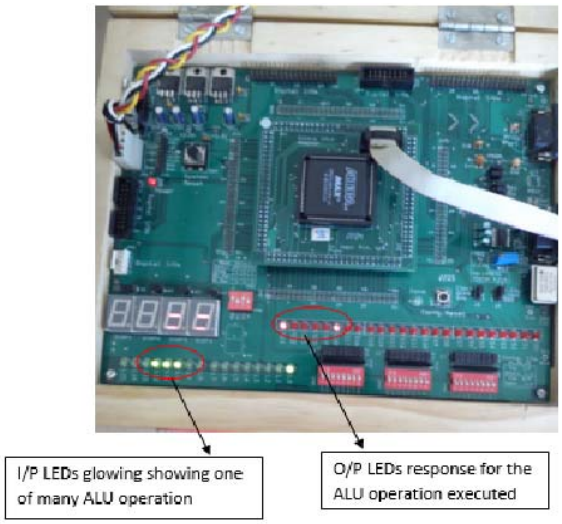

Figure 1.16I/P and O/P LEDs response of CPLD to a given ALU operation execution

1.3 Conclusion

The 4-bit ALU has been implemented using CPLD and simulation result has been verified using Altera Quartus. XOR logic has been implemented to check the working of ALU comparing with a 4 bit faulty ALU. Ad-hoc process has been used to test the Built-In Self-Test capability of ALU. A total of 5 functions has been performed.

1.4 Future Scope

- The ALU can be further enhanced with the design of Control Unit to function as microprocessor.
- More functionality can be implemented in future by increasing the number of bits of CTL input.
- Other than Ad-hoc process for testing, Scan methods can also be implemented.

1.5 References

[1] Ateeq A. Khan, "*An Improved Design of ALU Targeted to State-of-the-Art FPGAs*", European Journal of Scientific Research ISSN 1450-216X Vol.63 No.3 (2011), pp. 456-463 © Euro Journals Publishing, Inc. 2011

[2] Samuel Winchenbach, Department of Electrical and Computer Engineering Mohammed Driss, Department of Electrical and Computer Engineering University of Maine, Orono, "*8-Bit Arithmetic Logic Unit*", European Journal of Scientific Research ISSN 1450-216X Vol.47 No.3 (2009), pp. 469-475 © Euro Journals Publishing, Inc. 2009

[3] *Synthesis and Simulation Design Guide*: A copyright production of Xilinx, Inc.

[4] Jin-Young Kim, Sehoon Kim, Joonhee Kang, *"Construction of an RSFQ 4-bit ALU with half adder cells"*, Applied Superconductivity,IEEE Transactions on June 2005,Volume: 15, Issue 2, Pages: 308 – 311, IEEE Spectrum

[5] Samir Palnitkar, "Verilog Hdl: A Guide to Digital Design and Synthesis"

[6]http://en.wikipedia.org/wiki/Ad_hoc_testing

Abhishek Singh is currently working as an asst. professor in GGITS, Jabalpur. He did his B.E in 2009 in Electronics and Communication Engineering and M. Tech in 2012 in Embedded System and VLSI Design from the same institute. He has over 5 years of teaching experience. His area of interest include Electronics, VLSI, Analog and Digital Communication.

Mohd. Arif is currently working as an asst. professor in GGITS, Jabalpur. He did his B.E in 2008 in Electronics and Communication Engineering and M. Tech in 2010 in Embedded System and VLSI Design from the same institute. He has over 5 years of teaching experience. His area of interest include Electronics, VLSI, Analog and Digital Communication.

Kalpita Agrawal is currently a final year student pursuing her Bachelor of Engineering in Electronics and Communication Engineering. Her area of interest include Electronics, VLSI, Optical Communication and Digital Electronics.

Anshita Gupta is currently a final year student pursuing her Bachelor of Engineering in Electronics and Communication Engineering. Her area of interest include Electronics, VLSI, Optical Communication and Digital Electronics.