Electronic Engineering Practice Lab Problem 1

- Omkar Bhilare
191060901

AIM: Design 8 Bit ALU using Verilog/VHDL.

Software: Iverilog, GTKWAVE, Quartus Prime, Modelsim.

Pin Description:

Pin Names	Description
A0 - A7	Operand Input **(ACTIVE HIGH)**
B0 - B7	Operand Input **(ACTIVE HIGH)**
S0 - S2	Function Select Inputs **(ACTIVE HIGH)**
M	Mode Control Input
C_in/Cn	Carry Input
FO - F7	Function Outputs **(ACTIVE HIGH)**
Cout	Carry Output

Function Table :

Mode Select Inputs			**ACTIVE HIGH OUTPUTS**	
S2	**S1**	**S0**	**LOGIC (M = 0)**	**ARITHMETIC (M = 1)**
0	0	0	A & B	A plus B plus cn
0	0	1	A | B	A minus B minus cn
0	1	0	A ^ B	A + 1
0	1	1	~(A | B)	B + 1
1	0	0	A << 2	A - 1
1	0	1	B << 2	B - 1
1	1	0	A >> 2	A
1	1	1	B >> 2	B

Procedure:

1. In following ALU, I had used the three bit Control Line to select the operation of ALU and to select the mode there is separate pin M which is active High.
2. For Example: for M : 1 (Arithmetic Mode) Control Word: 010, Then ALU will perform Increment A.
3. In Verilog Code I have used simple case statements for different functions of ALU.
4. And Implemented same verilog Code in **Hardware(on CYCLONE2 FPGA)**

Verilog Code:

```
//Verilog Code for 8 bit ALU.
//Omkar Bhilare, 191060901

module ALU( A, B, c_in, control_line, mode_select, out, c_out);

input [7:0] A, B;         //8 Bit input A and B
input c_in;               //C in input
input [2:0] control_line;  //3 Bit control line for selecting ALU
operation
input mode_select;        //Mode = 1 = ARITHMETIC :: Mode = 0 =
LOGICAL

output reg c_out;         //C out output for arithmetic
output reg [7:0] out;       //8 Bit output reg

always@(*)
begin
    if(mode_select)
    begin
       case(control_line)
            3'd0 :  {c_out, out} = A + B + c_in;
            3'd1 :  {c_out, out} = A - B - c_in;
            3'd2 :  {c_out, out} = A + 1'b1;
            3'd3 :  {c_out, out} = B + 1'b1;
            3'd4 :  {c_out, out} = A - 1'b1;
            3'd5 :  {c_out, out} = B - 1'b1;
            3'd6 :  begin out = A; c_out = 0; end
            3'd7 :  begin out = B; c_out = 0; end
            default: out  = 8'd0;
```

```
        endcase
    end
    else
    begin
        c_out = 0;        //Keeping C out LOW in Logical Mode
        case(control_line)
             3'd0 :  out = A & B;
             3'd1 :  out = A | B;
             3'd2 :  out = A ^ B;
          3'd3 :  out = ~ (A | B);
             3'd4 :  out = A << 2;
             3'd5 :  out = B << 2;
             3'd6 :  out = A >> 2;
             3'd7 :  out = B >> 2;
             default: out  = 8'd0;
        endcase
    end
end

endmodule
```

TestBench Code:

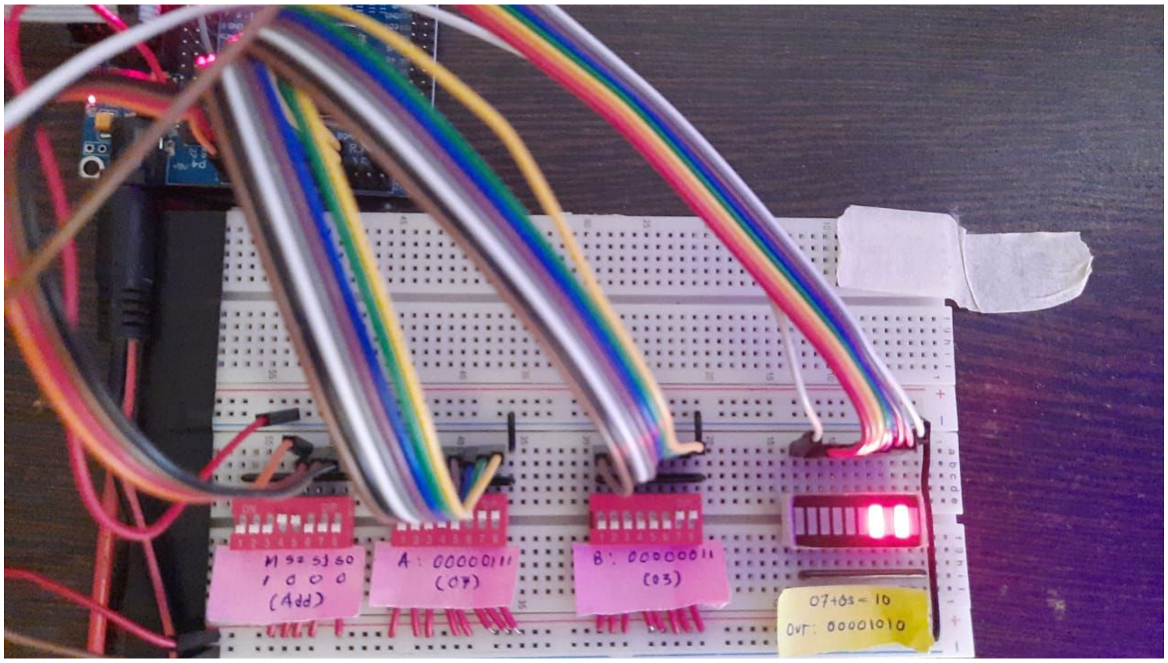
```
//Test Bench for ALU
`include "ALU.v"

module ALU_TB();

reg [7:0] a, b;
reg C_IN;
reg [2:0] CONTROL_LINE;
reg MODE_SELECT;

wire [7:0]OUT;
wire C_OUT;
integer i;
integer j;
//Creating ALU instant
```

```
ALU i1
(
    .A(a),
    .B(b),
    .c_in(C_IN),
    .control_line(CONTROL_LINE),
    .mode_select(MODE_SELECT),
    .out(OUT),
    .c_out(C_OUT)
);

initial
begin
    $monitor($time, "  a = %b b = %b cin = %b ::  out = %b  cout = %b
:: mode = %b control_word = %b", a, b, C_IN, OUT, C_OUT, MODE_SELECT,
CONTROL_LINE);
    //$dumpfile("a.vcd");
    //$dumpvars(0, ALU_TB);
end

initial
begin
    a = 8'd2; b = 8'd3; C_IN = 0;
    MODE_SELECT = 0;
    CONTROL_LINE = 3'd0;
    for(i = 0; i < 2; i++)
    begin
    #5  MODE_SELECT = i; CONTROL_LINE = 3'd0;
        for(j = 1; j < 8; j++)
            #5 CONTROL_LINE = j;
    end
    #5  $finish;
end
endmodule
```

Output:

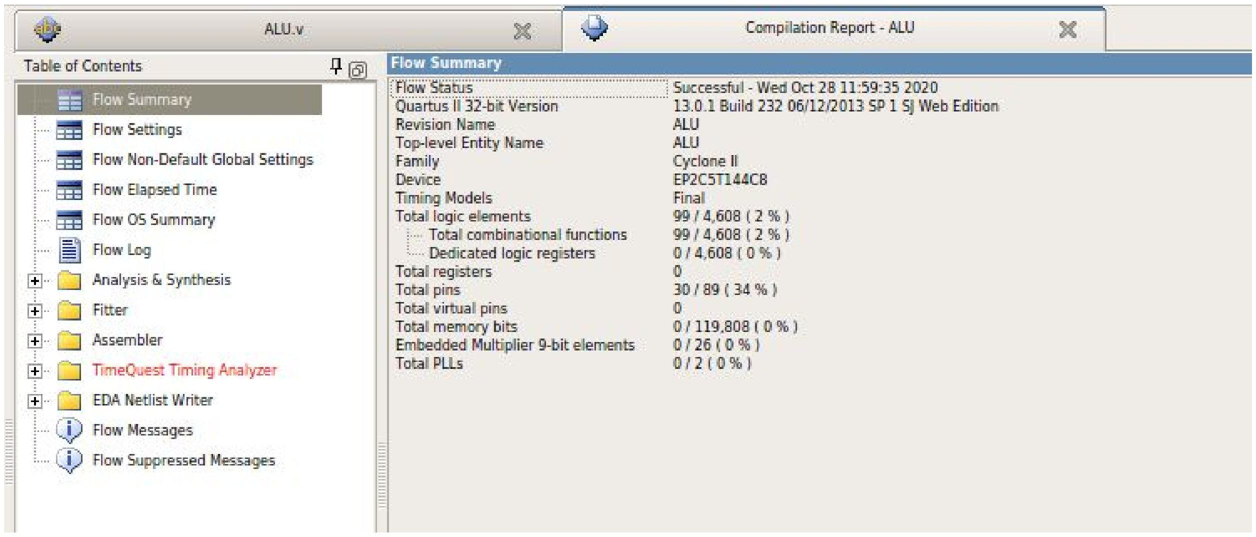
- **Hardware Implementation on Cyclone 2 FPGA:**

A = 07h
B = 03h
ALU mode = 1 (Arithmetic)
ALU control signal = 000 (ADD)
Output = 0Ah



- **Compilation Output:**

Total Logic Cells Used: 99 (of ALTERA CYCLONE2)



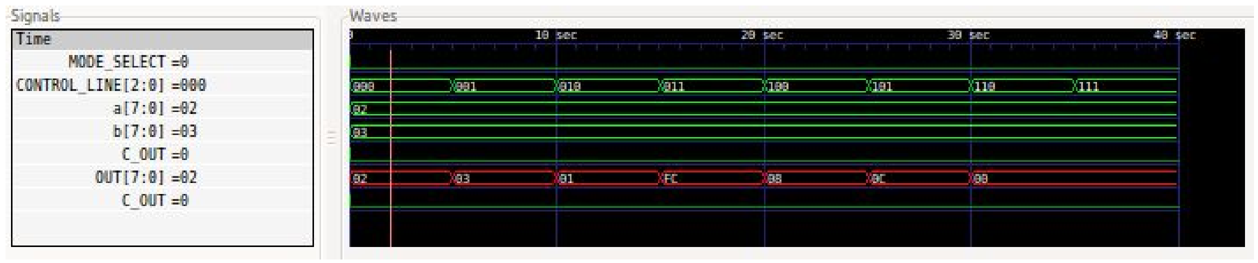
Flow Summary

Flow Status	Successful - Wed Oct 28 11:59:35 2020
Quartus II 32-bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	ALU
Top-level Entity Name	ALU
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Total logic elements	99 / 4,608 (2 %)
Total combinational functions	99 / 4,608 (2 %)
Dedicated logic registers	0 / 4,608 (0 %)
Total registers	0
Total pins	30 / 89 (34 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

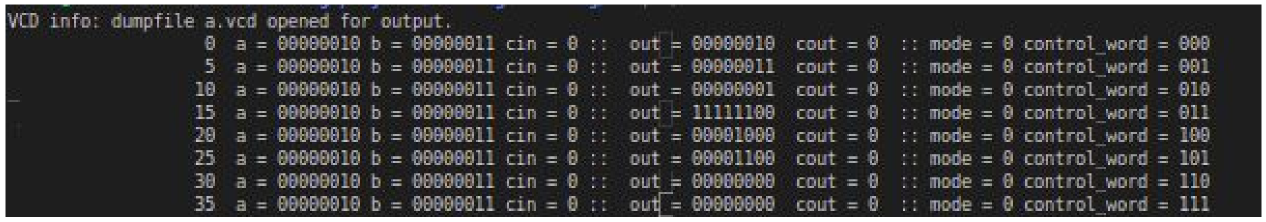
- **GTKWAVE and Iverilog output:**
  In following output, control word is varying from 000 to 111 for mode select 0(**Logical Operation**), then once again changing from 000 to 111 for mode select 1 (**Arithmetic Operation**)

  Operands: **A: 02** :: **B: 03** :: **c_in = 0**

  GTKWAVE output of all Logical Operations:



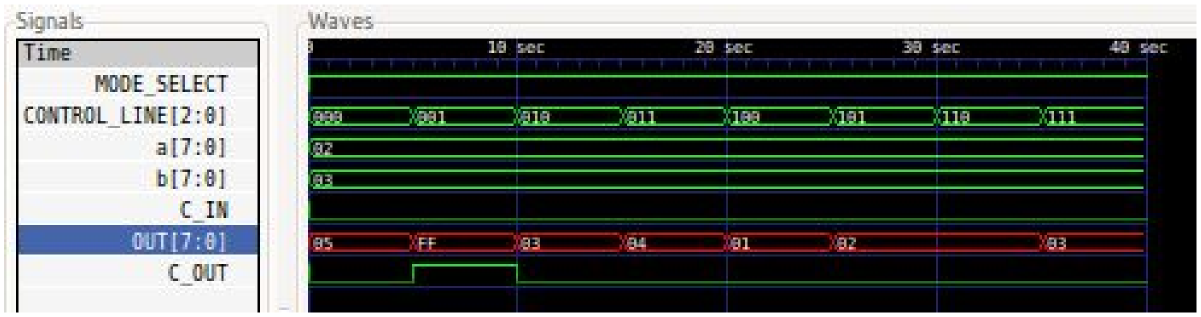
  iVerilog output of all Logical Operations:



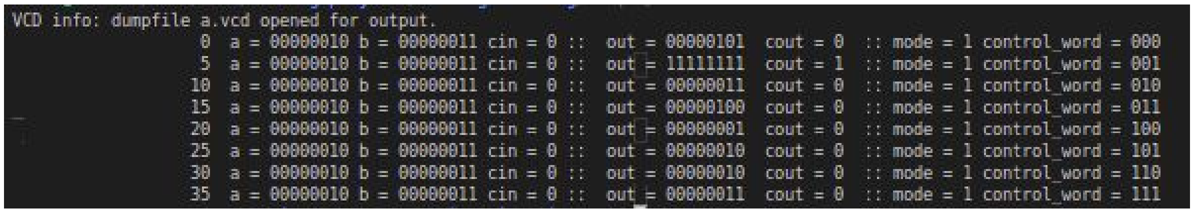
```
VCD info: dumpfile a.vcd opened for output.
                   0  a = 00000010 b = 00000011 cin = 0 ::  out = 00000010  cout = 0  :: mode = 0 control_word = 000
                   5  a = 00000010 b = 00000011 cin = 0 ::  out = 00000011  cout = 0  :: mode = 0 control_word = 001
                  10  a = 00000010 b = 00000011 cin = 0 ::  out = 00000001  cout = 0  :: mode = 0 control_word = 010
                  15  a = 00000010 b = 00000011 cin = 0 ::  out = 11111100  cout = 0  :: mode = 0 control_word = 011
                  20  a = 00000010 b = 00000011 cin = 0 ::  out = 00001000  cout = 0  :: mode = 0 control_word = 100
                  25  a = 00000010 b = 00000011 cin = 0 ::  out = 00001100  cout = 0  :: mode = 0 control_word = 101
                  30  a = 00000010 b = 00000011 cin = 0 ::  out = 00000000  cout = 0  :: mode = 0 control_word = 110
                  35  a = 00000010 b = 00000011 cin = 0 ::  out = 00000000  cout = 0  :: mode = 0 control_word = 111
```

  For Mode Select 0: Logical Operation
  1. Control Word (000h) : A & B: 00000010 & 00000011 = 0000 0010 (**02h**)
  2. Control Word (001h) : A | B: 00000010 | 00000011 = 0000 0011 (**03h**)
  3. Control Word (010h) : A ^ B: 00000010 ^ 00000011 = 0000 0001 (**01h**)
  4. Control Word (011h) : ~(A | B): ~(00000010 | 00000011) = 1111 1100 (**FCh**)
  5. Control Word (100h) : A << 2: 00000010 left shift by 2 = 0000 1000 (**08h**)
  6. Control Word (101h) : B << 2: 00000011 left shift by 2 = 0000 1100 (**0Ch**)
  7. Control Word (110h) : A >> 2: 00000010 right shift by 2 = 0000 0000 (**00h**)
  8. Control Word (111h) : B >> 2: 00000011 right shift by 2 = 0000 0000 (**00h**)

GTKWAVE output of all Arithmetic Operations:



iVerilog output of all Arithmetic Operations:



```
VCD info: dumpfile a.vcd opened for output.
                    0  a = 00000010 b = 00000011 cin = 0 ::  out = 00000101  cout = 0  :: mode = 1 control_word = 000
                    5  a = 00000010 b = 00000011 cin = 0 ::  out = 11111111  cout = 1  :: mode = 1 control_word = 001
                   10  a = 00000010 b = 00000011 cin = 0 ::  out = 00000011  cout = 0  :: mode = 1 control_word = 010
                   15  a = 00000010 b = 00000011 cin = 0 ::  out = 00000100  cout = 0  :: mode = 1 control_word = 011
                   20  a = 00000010 b = 00000011 cin = 0 ::  out = 00000001  cout = 0  :: mode = 1 control_word = 100
                   25  a = 00000010 b = 00000011 cin = 0 ::  out = 00000010  cout = 0  :: mode = 1 control_word = 101
                   30  a = 00000010 b = 00000011 cin = 0 ::  out = 00000010  cout = 0  :: mode = 1 control_word = 110
                   35  a = 00000010 b = 00000011 cin = 0 ::  out = 00000011  cout = 0  :: mode = 1 control_word = 111
```

For Mode Select 1: Arithmetic Operation

1. Control Word (000h) : A + B: 00000010 + 00000011 = 0000 0101 (**05h**)
2. Control Word (001h) : A - B: 00000010 - 00000011 = 1111 1111 (**FFh**) {-1}
3. Control Word (010h) : A + 1: 00000010 + 1 = 0000 0011 (**03h**)
4. Control Word (011h) : B + 1: 00000010 + 1 = 0000 0100 (**04h**)
5. Control Word (100h) : A - 1: 00000010 - 1 = 0000 0001 (**01h**)
6. Control Word (101h) : B - 1: 00000011 - 1 = 0000 0010 (**02h**)
7. Control Word (110h) : A : 00000010 = 0000 0010 (**02h**)
8. Control Word (111h) : B: 00000011 = 0000 0011 (**03h**)

CONCLUSION:

Thus, I have implemented 8 Bit ALU code using verilog and tested the code in software with different operations in GTKWAVE and as well in quartus prime and finally I have implemented 8 BIT ALU verilog code in Altera Cyclone 2 FPGA