**Group ID:** 10 **Names:** A.S.Akil Arif Ibrahim & Ashutosh Parija

**Enrollment Numbers:** 15116001 & 15116010

**Description Of The Design**

8-bit ALU must perform 4 different operations based on the Opcode – 00 : Addition , 01: AND , 10 : XOR , 11 : Multiplication.

Addition: 8-bit Ripple Carry Adder is designed in adder8bit.v . In case of overflow, the overflow bit becomes 1 and the result becomes 0.

AND: Designed in and8bit.v . And operation is performed bitwise.

XOR: Designed in xor8bit.v . Xor operation is performed bitwise.

Multiplication: 8-bit Boothe’s Multiplier is designed in mult8bit.v . The result is stored in a register pair.

The top level module is alu8bit.v . The register Y0 contains the higher order 8 bits in the 16 bit product in case of multiplication. For all the other operations, Y0 is zero. The Carry bit C and Overflow bit O are zero for the operations And, Xor and Multiplication.

The Test Bench is written in alu8bit\_tb.v . It is then simulated and the results are obtained.

**Verilog Code For 8-bit Ripple Carry Adder**

// GroupID-10 (15116001\_15116010) - A.S.Akil Arif Ibrahim & Ashutosh Parija

// Date: October 28, 2016

// adder8bit.v - 8-bit Ripple Carry Adder

module FullAdder(

input A,

input B,

input Cin,

output S,

output Cout

);

xor(S,A,B,Cin);

wire w1,w2,w3;

xor(w1,A,B);

and(w2,Cin,w1);

and(w3,A,B);

or(Cout,w2,w3);

endmodule

module adder8bit(

input [7:0] A,

input [7:0] B,

input Cin,

output [7:0] S,

output Cout,

output O

);

wire [7:0] S0;

FullAdder FA0(A[0],B[0],Cin,S0[0],Co0);

FullAdder FA1(A[1],B[1],Co0,S0[1],Co1);

FullAdder FA2(A[2],B[2],Co1,S0[2],Co2);

FullAdder FA3(A[3],B[3],Co2,S0[3],Co3);

FullAdder FA4(A[4],B[4],Co3,S0[4],Co4);

FullAdder FA5(A[5],B[5],Co4,S0[5],Co5);

FullAdder FA6(A[6],B[6],Co5,S0[6],Co6);

FullAdder FA7(A[7],B[7],Co6,S0[7],Cout);

//Overflow Check

xor(O,Cout,Co6);

and(S[0],S0[0],~O);

and(S[1],S0[1],~O);

and(S[2],S0[2],~O);

and(S[3],S0[3],~O);

and(S[4],S0[4],~O);

and(S[5],S0[5],~O);

and(S[6],S0[6],~O);

and(S[7],S0[7],~O);

endmodule

**Verilog Code For 8-bit Boothe’s Multiplier**

// GroupID-10 (15116001\_15116010) - A.S.Akil Arif Ibrahim & Ashutosh Parija

// Date: October 28, 2016

// mult8bit.v - 8-bit Boothe's Multiplier

module mult8bit(

input [7:0] Q0,

input [7:0] M,

output reg [7:0] A, //A is a 8-bit register which stores higher order 8 bits in the 16-bit product

output reg [7:0] Q //Q is a 8-bit register which stores lower order 8 bits in the 16-bit product

);

integer i;

reg Q\_1;

integer count;

reg clk;

wire [7:0] Sum,Difference;

wire t1,t2,t3,t4;

initial begin

Q = Q0;

A = 0;

Q\_1 = 0;

count = 8;

clk = 0;

end

always begin

#5 clk = ~clk;

end

always @(Q0)

begin

Q <= Q0;

count = 8;

end

adder8bit Add(A,M,0,Sum,t1,t2);

adder8bit Subtract(A,~M,1,Difference,t3,t4);

always @(posedge clk)

begin

if(count > 0)

begin

if(Q[0] == 0 && Q\_1 == 1)

A <= Sum;

else if(Q[0] == 1 && Q\_1 == 0)

A <= Difference;

Q\_1 <= Q[0];

for (i = 0; i < 7; i = i+1)

begin

Q[i] <= Q[i+1];

end

#2 Q[7] <= A[0];

#2 for (i = 0; i < 7; i = i+1)

begin

A[i] <= A[i+1];

end

count <= count-1;

end

end

endmodule

**Verilog Code For 8-bit AND**

// GroupID-10 (15116001\_15116010) - A.S.Akil Arif Ibrahim & Ashutosh Parija

// Date: October 28, 2016

// and8bit.v - 8-bit AND

module and8bit(

input [7:0] A,

input [7:0] B,

output [7:0] Y

);

and(Y[0],A[0],B[0]);

and(Y[1],A[1],B[1]);

and(Y[2],A[2],B[2]);

and(Y[3],A[3],B[3]);

and(Y[4],A[4],B[4]);

and(Y[5],A[5],B[5]);

and(Y[6],A[6],B[6]);

and(Y[7],A[7],B[7]);

endmodule

**Verilog Code For 8-bit XOR**

// GroupID-10 (15116001\_15116010) - A.S.Akil Arif Ibrahim & Ashutosh Parija

// Date: October 28, 2016

// xor8bit.v - 8-bit XOR

module xor8bit(

input [7:0] A,

input [7:0] B,

output [7:0] Y

);

xor(Y[0],A[0],B[0]);

xor(Y[1],A[1],B[1]);

xor(Y[2],A[2],B[2]);

xor(Y[3],A[3],B[3]);

xor(Y[4],A[4],B[4]);

xor(Y[5],A[5],B[5]);

xor(Y[6],A[6],B[6]);

xor(Y[7],A[7],B[7]);

endmodule

**Verilog Code For The Top Module- 8-bit ALU**

// GroupID-10 (15116001\_15116010) - A.S.Akil Arif Ibrahim & Ashutosh Parija

// Date: October 28, 2016

// alu8bit.v - 8-bit ALU. This is the Top Module.

module alu8bit(

input [7:0] A,

input [7:0] B,

input [1:0] Opcode,

output reg [7:0] Y,

output reg [7:0] Y1, //Y1 stores higher order 8 bits in 16-bit product. For other operations, it is 0.

output reg C,

output reg O

);

wire [7:0] S,A1,X1,P1,P0;

wire C1,O1;

adder8bit Sum(A,B,0,S,C1,O1);

and8bit AND(A,B,A1);

xor8bit XOR(A,B,X1);

mult8bit Product(A,B,P1,P0);

always@(\*)

begin

case(Opcode)

2'b00: begin

Y = S;

C = C1;

O = O1;

Y1 = 0;

end

2'b01: begin

Y = A1;

O = 0;

C = 0;

Y1 = 0;

end

2'b10: begin

Y = X1;

O = 0;

C = 0;

Y1 = 0;

end

2'b11: begin

Y = P0;

Y1 = P1;

O = 0;

C = 0;

end

endcase

end

endmodule

**Verilog Code For The Test Bench**

// GroupID-10 (15116001\_15116010) - A.S.Akil Arif Ibrahim & Ashutosh Parija

// Date: October 28, 2016

// alu8bit\_tb.v - Test Bench For Top Module 8-bit ALU

module alu8bit\_tb;

// Inputs

reg [7:0] A;

reg [7:0] B;

reg [1:0] Opcode;

reg [17:0] Instruction;

// Outputs

wire [7:0] Y;

wire [7:0] Y1;

wire C;

wire O;

// Instantiate the Unit Under Test (UUT)

alu8bit uut (

.A(A),

.B(B),

.Opcode(Opcode),

.Y(Y),

.Y1(Y1),

.C(C),

.O(O)

);

always@(Instruction)

begin

B = Instruction[7:0];

A = Instruction[15:8];

Opcode = Instruction[17:16];

end

initial begin

Instruction = 18'b000000011000000100; //A = 6, B = 4

#100 Instruction = 18'b010000011000000100;

#100 Instruction = 18'b100000011000000100;

#100 Instruction = 18'b110000011000000100;

#100 Instruction = 18'b001111101000000100; //A = -6, B = 4

#100 Instruction = 18'b011111101000000100;

#100 Instruction = 18'b101111101000000100;

#100 Instruction = 18'b111111101000000100;

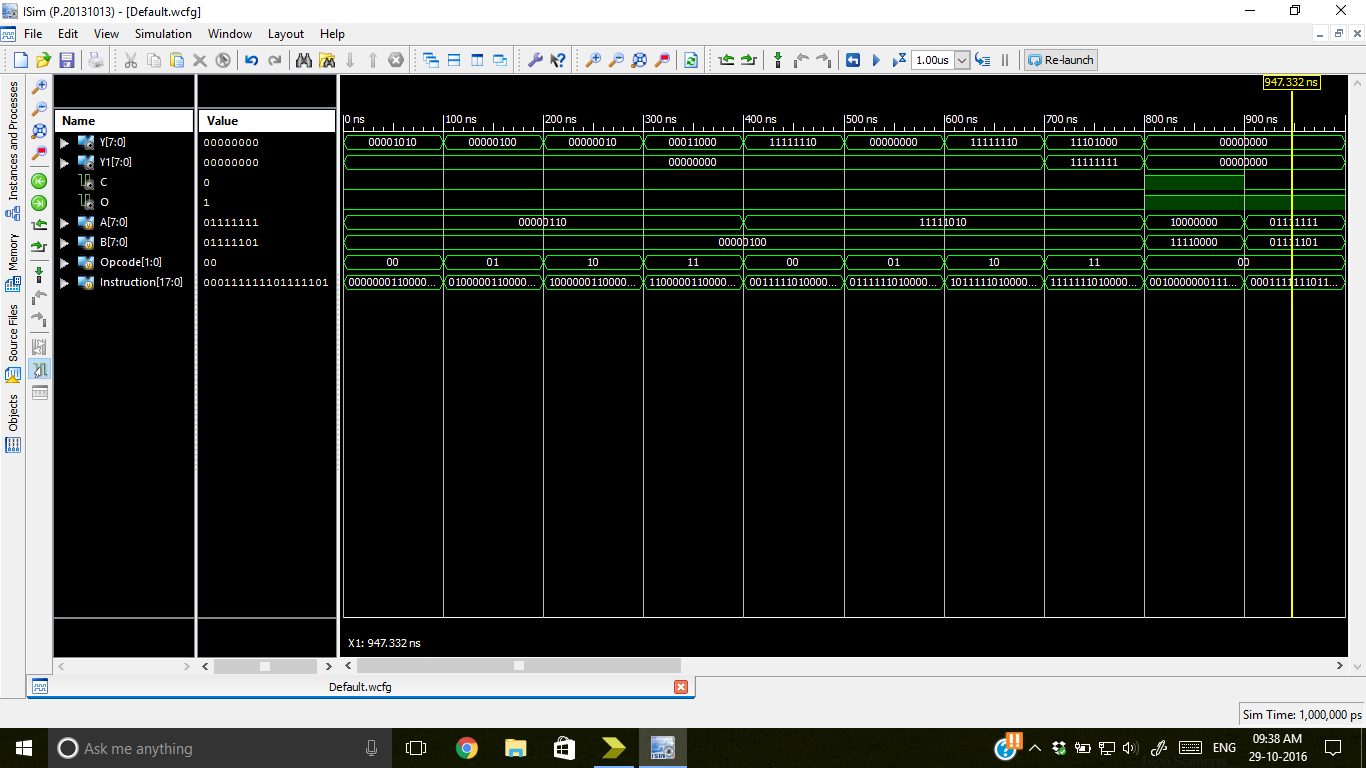
#100 Instruction = 18'b001000000011110000; //A = -128,B = -16 --> Overflow

#100 Instruction = 18'b000111111101111101; //A = 63,B = 16 --> Overflow

end

endmodule

**Simulation Results- Waveform**

****