Chris Repanich

CECS 341

Lab 3: ALU & ALU\_Control

**ALU:**

**Source Code:**

module ALU(A, B, ALU\_Ctl, Zero\_Flag, Output);

input [31:0] A, B;

input [3:0] ALU\_Ctl;

output reg Zero\_Flag;

output reg [31:0] Output;

always@(A, B, ALU\_Ctl, Zero\_Flag, Output)

begin

Zero\_Flag <= 1'b0;

case(ALU\_Ctl)

4'b0000://A & B

begin

Output <= A&B;

end

4'b0001://A | B

begin

Output <= A|B;

end

4'b0010://A + B

begin

Output <= A + B;

end

4'b0110://A - B

begin

Output <= A - B;

end

4'b0111://set on less than

begin

if (A<B)

Output <= 32'b1;

else

Output <= 32'b0;

end

4'b1100://nor

begin

Output <= ~(A|B);

end

default:

begin

Output <= 32'hXXXXXXXX;

Zero\_Flag <= 1'bX;

end

endcase

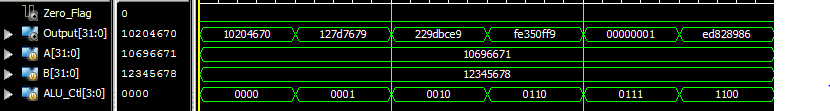
if (Output == 0)

Zero\_Flag <= 1'b1;

end

endmodule

**Screenshot:**

****

**ALU\_Control:**

**Source Code:**

module ALU\_Control(FuncCode, ALU\_op, ALU\_Ctl);

input [5:0] FuncCode;

input [1:0] ALU\_op;

output reg [3:0] ALU\_Ctl;

always@(FuncCode, ALU\_op, ALU\_Ctl)

begin

case(FuncCode)

6'b100000://ADD

begin

ALU\_Ctl <= 4'b0010;

end

6'b100010://SUB

begin

ALU\_Ctl <= 4'b0110;

end

6'b100100://AND

begin

ALU\_Ctl <= 4'b0000;

end

6'b100101://OR

begin

ALU\_Ctl <= 4'b0001;

end

6'b101010://SLT

begin

ALU\_Ctl <= 4'b0111;

end

6'b100111://NOR

begin

ALU\_Ctl <= 4'b1100;

end

6'bXXXXXX://Address Calculation, Equality Comparison

begin

if (ALU\_op == 2'b00)

ALU\_Ctl <= 4'b0010;

else

ALU\_Ctl <= 4'b0110;

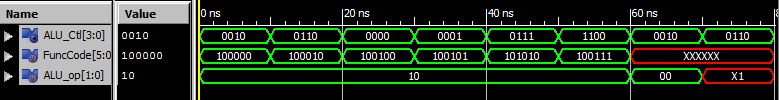
end

endcase

end

endmodule

**Screenshot:**

****

**ALU\_Unit:**

**Source Code:**

module ALU\_Unit(FuncCode, ALU\_op, A, B, Zero\_Flag, Output);

input [31:0] A,B;

input [5:0] FuncCode;

input [1:0] ALU\_op;

wire [3:0] ALU\_Ctl;

output Zero\_Flag;

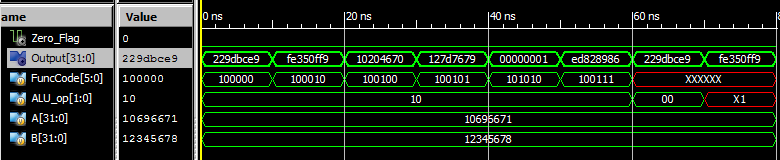
output [31:0] Output;

ALU\_Control aluctl(FuncCode, ALU\_op, ALU\_Ctl);

ALU alu1(A,B,ALU\_Ctl, Zero\_Flag,Output);

endmodule

**Screenshot:**

****