Name & ID: Laiba Fatima Khan (Lk04067)

CA Lab05 Report

TASK 1

1-bit ALU:

```
module ALU_1_bit
(
	input a, b, CarryIn,
	input[3:0] ALUop,
	output Result, CarryOut
);
	wire abar, bbar, mux1out, mux2out;
	assign abar = ~a;
	assign bbar = ~b;
	assign mux1out = ALUop[3] ? abar : a;   //Ainvert is ALUop bit 3
	assign mux2out = ALUop[2] ? bbar : b;   //Binvert is ALUop bit 2
	reg regRes;
	assign Result = regRes;
	reg car;
	assign CarryOut = car;

//* so always block runs in all conditions
	always@(*)
	begin
		case(ALUop[1:0]) //Operation is last two bits of ALUop
			2'd0: regRes = (mux1out & mux2out);   //AND when Operation is 0
			2'd1: regRes = (mux1out | mux2out);   //OR  when Operation is 1
			2'd2: regRes = (mux1out + mux2out + CarryIn);   //add when Operation is 2
		endcase
```

```
                car = mux1out & CarryIn | mux2out & CarryIn | mux1out & mux2out
        end
endmodule
```

TestBench:

```
module tb
(
);
reg a, b, CarryIn;
reg[3:0] ALUop;
wire Result;
wire CarryOut;

ALU_1_bit alu
(
        .a(a),
        .b(b),
        .CarryIn(CarryIn),
        .ALUop(ALUop),
        .Result(Result),
        .CarryOut(CarryOut)
);

        initial
        begin
        a = 1'b1;
        b = 1'b1;
        CarryIn = 1'b0;
```

```
        ALUop = 4'b0000; // 0 = And
        #100 ALUop = 4'b0001; //1 = OR
        #100 ALUop = 4'b0010; //2 = Add
        #100 ALUop = 4'b1100; //12 = NOR
        #100 ALUop = 4'b0110; //6 = Subtract
        CarryIn = 1'b1;
        end

        initial
        $monitor ("Time = ", $time, "  Operation = ", ALUop, " --------> Result = %d", Result, "  Carry Out = %d", CarryOut);
endmodule
```

TASK 1

Using 1-bit ALU from Task1

6-bit ALU:

```
module ALU_6_bit
(
        input [5:0] a,
        input [5:0] b,
        input CarryIn,
        input[3:0] ALUop,
        output [5:0] Result,
        output CarryOut
);
wire [4:0] carr;
```

```
ALU_1_bit alu0
(
        .a(a[0]),
        .b(b[0]),
        .CarryIn(CarryIn),
        .ALUop(ALUop),
        .Result(Result[0]),
        .CarryOut(carr[0])
);

ALU_1_bit alu1
(
        .a(a[1]),
        .b(b[1]),
        .CarryIn(carr[0]),
        .ALUop(ALUop),
        .Result(Result[1]),
        .CarryOut(carr[1])
);

ALU_1_bit alu2
(
        .a(a[2]),
        .b(b[2]),
        .CarryIn(carr[1]),
        .ALUop(ALUop),
        .Result(Result[2]),
        .CarryOut(carr[2])
);
```

```
ALU_1_bit alu3
(
        .a(a[3]),
        .b(b[3]),
        .CarryIn(carr[2]),
        .ALUop(ALUop),
        .Result(Result[3]),
        .CarryOut(carr[3])
);
ALU_1_bit alu4
(
        .a(a[4]),
        .b(b[4]),
        .CarryIn(carr[3]),
        .ALUop(ALUop),
        .Result(Result[4]),
        .CarryOut(carr[4])
);
ALU_1_bit alu5
(
        .a(a[5]),
        .b(b[5]),
        .CarryIn(carr[4]),
        .ALUop(ALUop),
        .Result(Result[5]),
        .CarryOut(CarryOut)
);

endmodule
```

TestBench:

```
module tb
(
);
reg [5:0] a;
reg [5:0] b;
reg CarryIn;
reg[3:0] ALUop;
wire [5:0] Result;
wire CarryOut;

ALU_6_bit alu
(
        .a(a),
        .b(b),
        .CarryIn(CarryIn),
        .ALUop(ALUop),
        .Result(Result),
        .CarryOut(CarryOut)
);

        initial
        begin
        a = 6'b111111;
        b = 6'b000111;
        CarryIn = 1'b0;

        ALUop = 4'b0000; //And
        #100 ALUop = 4'b0001; //OR
```

```
        #100 ALUop = 4'b0010; //Add
        #100 ALUop = 4'b1100; //NOR
        #100 ALUop = 4'b0110; //Subtract
        CarryIn = 1'b1;
        end

        initial
        $monitor ("Time = ", $time, "  Operation = ", ALUop, " --------> Result = %d", Result, "  Carry Out = %d", CarryOut);
endmodule
```

EXERCISE

64-bit ALU:

```
module ALU_64_bit
(
	input [63:0]a,
	input [63:0]b,
	input[3:0] ALUop,
	output [63:0]Result,
	output ZERO
);
	reg [63:0]regRes;
	assign Result = regRes;
	reg zer;
	assign ZERO = zer;

//* so always block runs in all conditions
	always@(*)
	begin
		case(ALUop[3:0]) //Operation is last two bits of ALUop
			4'd0: regRes = (a & b);   //AND when Operation is 0
			4'd1: regRes = (a | b);   //OR  when Operation is 1
			4'd2: regRes = (a + b);    //add when Operation is 2
			4'd6: regRes = (a - b);
			4'd12: regRes = ~(a | b);
		endcase
		case(regRes)
		64'd0: zer = 1;
```

```
                default: zer = 0;
                endcase
        end
endmodule
```

TestBench:

```
module tb
(
);

reg [63:0] a;
reg [63:0] b;
reg CarryIn;
reg[3:0] ALUop;
wire [63:0] Result;
wire ZERO;

ALU_64_bit alu
(
        .a(a),
        .b(b),
        .ALUop(ALUop),
        .Result(Result),
        .ZERO(ZERO)
);
        initial
        begin
        a = 64'd6000;
        b = 64'd4;
```

```
        ALUop = 4'b0000; //And
        #100 ALUop = 4'b0001; //OR
        #100 ALUop = 4'b0010; //Add
        #100 ALUop = 4'b0110; //Subtract
        #100 ALUop = 4'b1100; //NOR
        end

        initial
        $monitor ("Time = ", $time, "  Operation = ", ALUop, " --------> Result = %d", Result, "  ZERO =
%d", ZERO);
endmodule
```