Jomar Pueyo UXT302

**LAB ASSIGNMENT 4: Subtractor & ALU**

**Subtractor**

Procedure: First step I took in creating the subtractor was translating the truth table into a k-map and then creating the logic formula of diff=(a&~b&~bin)|(!a&~b&bin)|(!a&b&~bin) and bout=(!a&bin)|(!a&b)|(b&bin). This creates a one bit subtractor. We abstantiate this four times in a module to create a four bit subtractor.

Observations: Subtractor subtracts A input from the B input with the Difference of the two as the output, and Bin as the borrow in and the Bout as borrow output.

Results:

Code:

module subtractor(a,b,bin,diff,bout);

input a,b,bin;

output diff,bout;

//truthtable to kmap logic

assign diff=(a&~b&~bin)|(!a&~b&bin)|(!a&b&~bin);

assign bout=(!a&bin)|(!a&b)|(b&bin);

endmodule

module fourbitsubtractor(a,b,bin,diff,bout);

input [3:0] a,b;

input bin;

output [3:0] diff;

output bout;

wire [2:0] carry;

//abstaniate 1bit subtractor four times for four bit subtractor

subtractor bit1(a[0],b[0],bin,diff[0],carry[0]);

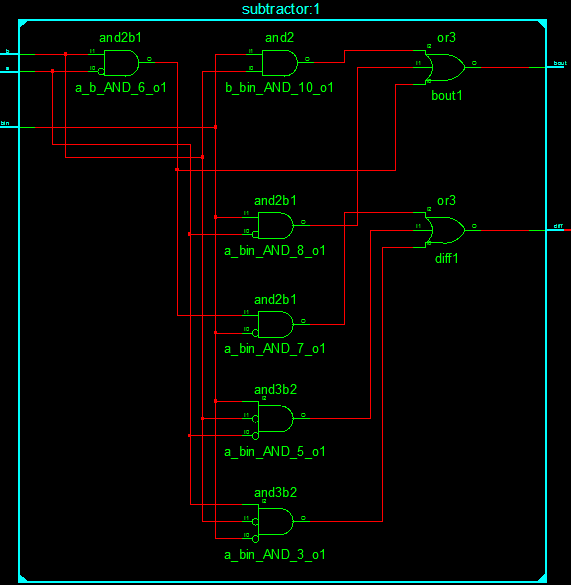
subtractor bit2(a[1],b[1],carry[0],diff[1],carry[1]);

subtractor bit3(a[2],b[2],carry[1],diff[2],carry[2]);

subtractor bit4(a[3],b[3],carry[2],diff[3],bout);

endmodule

Schematic:



Test Bench:

`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 19:29:21 02/15/2017

// Design Name: fourbitsubtractor

// Module Name: C:/temp/ALU/Subtractor\_tb.v

// Project Name: ALU

// Target Device:

// Tool versions:

// Description:

//

// Verilog Test Fixture created by ISE for module: fourbitsubtractor

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

////////////////////////////////////////////////////////////////////////////////

module Subtractor\_tb;

// Inputs

reg [3:0] a;

reg [3:0] b;

reg bin;

// Outputs

wire [3:0] diff;

wire bout;

// Instantiate the Unit Under Test (UUT)

fourbitsubtractor uut (

.a(a),

.b(b),

.bin(bin),

.diff(diff),

.bout(bout)

);

initial begin

// Initialize Inputs

a = 0;

b = 0;

bin = 0;

#100;

a = 1;

b = 1;

bin = 0;

#100;

a = 0;

b = 1;

bin = 0;

#100;

a = 0;

b = 0;

bin = 1;

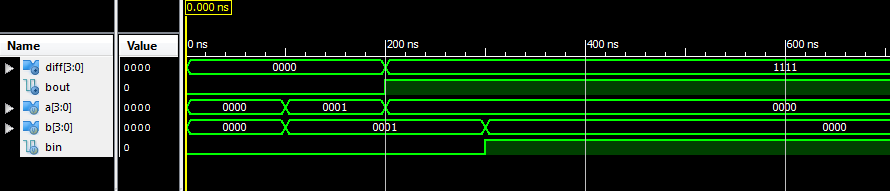
#100;

// Add stimulus here

end

endmodule

Timing Diagram:

 Conclusion: I learned how to create a four bit subtractor from a single subtractor truth table.

**Arithmetic and Logic Unit (ALU)**

Procedure: An ALU implements eight different functions. In our case, we introduced add, subtraction, or, and, shift left, shift right, rotate left, and rotate right. In three operations we have to design in gate level, data flow level, and behavioral level. Using the previous labs knowledge, we reuse the 4 bit adder and 4 bit subtractor. To select which function will be used, a control is used in a mux.

Observation: Wires had to be used specifically to bring the values from the abstantiated adder, subtractor, and or logic. In addition to the main mux to be used to select the function outputted, a secondary mux is made to output the cout, carry bit, of the adder and subtractor. I learned that big endian and little endian make a huge difference in my code which caused my shifting, rol, and ror to have improper functions.

Results:

Code:

//////////////////////////////////////////////////adder//////////////////////

module fourbitadder(sum,cout,cin,a,b);

//4 full adders together

output [3:0] sum;

output cout;

input [3:0] a,b;

input cin;

//connects the adders together

wire c1,c2,c3;

//sum,carryout,a in, b in, carryin

full adder1(sum[0],c1,a[0],b[0],cin);

full adder2(sum[1],c2,a[1],b[1],c1);

full adder3(sum[2],c3,a[2],b[2],c2);

full adder4(sum[3],cout,a[3],b[3],c3);

endmodule

module full(s,co,ci,a,b);

//full block of adding

output s,co;

input ci,a,b;

//connects the two half adders to create one full adder

wire abxor, aband, abciand;

// sum,carry,a,b

half add1(abxor,aband,a,b);

half add2(s,abciand,ci,abxor);

// carryout of fulladder

or (co,abciand,aband);

endmodule

module half(s,c,a,b);

//half block of adding

output s,c;

input a,b;

xor (s,a,b);

and (c,a,b);

endmodule

////////////////////////////////////////////adder////////////////////////////

////////////////////////////////////////////subtractor//////////////////////

module subtractor(a,b,bin,diff,bout);

input a,b,bin;

output diff,bout;

//truthtable to kmap logic

assign diff=(a&~b&~bin)|(!a&~b&bin)|(!a&b&~bin);

assign bout=(!a&bin)|(!a&b)|(b&bin);

endmodule

module fourbitsubtractor(a,b,bin,diff,bout);

input [3:0] a,b;

input bin;

output [3:0] diff;

output bout;

wire [2:0] carry;

//abstaniate 1bit subtractor four times for four bit subtractor

subtractor bit1(a[0],b[0],bin,diff[0],carry[0]);

subtractor bit2(a[1],b[1],carry[0],diff[1],carry[1]);

subtractor bit3(a[2],b[2],carry[1],diff[2],carry[2]);

subtractor bit4(a[3],b[3],carry[2],diff[3],bout);

endmodule

/////////////////////////////////////////subtractor/////////////////////////

///////////////////////////////////////or////

module operationor(a,b,out);

input [3:0] a,b;

output reg [3:0] out;

always@\*

out = a|b;

endmodule

///////////////////////////////////////or////

module ALU(a,b,cin,cout,out,control);

input [3:0] a,b;

input [2:0] control;

input cin;

output reg [3:0] out;

output reg cout;

wire [3:0] w1,w2,w3;

wire c1,c2;

// wires to carry the abstantiation output/cout to the mux

fourbitadder add(w1,c1,cin,a,b);

fourbitsubtractor sub(a,b,cin,w2,c2);

operationor orr(a,b,w3);

always@\*

begin

case(control)

3'b000: out = w1;

3'b001: out = w2;

3'b010: out = w3;

3'b011: out = a&b;

3'b100: out = {a[1],a[2],a[3],1'b0};

3'b101: out = {1'b0,a[0],a[1],a[2]};

3'b110: out = {a[3:1],a[0]};

3'b111: out = {a[3],a[2:0]};

endcase

end

//separate mux for the cout

always@\*

begin

case(control)

3'b000: cout<= c1;

3'b001: cout<= c2;

default: cout<= 1'b0;

endcase

end

endmodule

Testbench:

`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 17:59:27 02/15/2017

// Design Name: ALU

// Module Name: C:/temp/ALU/ALU\_TB.v

// Project Name: ALU

// Target Device:

// Tool versions:

// Description:

//

// Verilog Test Fixture created by ISE for module: ALU

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

////////////////////////////////////////////////////////////////////////////////

module ALU\_TB;

// Inputs

reg [3:0] a;

reg [3:0] b;

reg cin;

reg [2:0] control;

// Outputs

wire cout;

wire [3:0] out;

// Instantiate the Unit Under Test (UUT)

ALU uut (

.a(a),

.b(b),

.cin(cin),

.cout(cout),

.out(out),

.control(control)

);

initial begin

// Initialize Inputs

a = 0;

b = 0;

cin = 0;

control = 0;

#100;

a = 4'b1010;

b = 4'b0101;

cin = 0;

control = 3'b000;//add

#100;

a = 4'b1010;

b = 4'b1010;

cin = 0;

control = 3'b001;//sub

#100;

a = 4'b1010;

b = 4'b0101;

cin = 0;

control = 3'b010;//or

#100;

a = 4'b1010;

b = 4'b0101;

cin = 1'b1;

control = 3'b000;//test cout

#100;

a = 4'b1010;

b = 4'b0101;

cin = 1'b1;

control = 3'b011;

#100;

a = 4'b1010;

b = 4'b0101;

cin = 1'b1;

control = 3'b100;

#100;

a = 4'b1010;

b = 4'b0101;

cin = 1'b1;

control = 3'b101;

#100;

a = 4'b1010;

b = 4'b0101;

cin = 1'b1;

control = 3'b110;

#100;

a = 4'b1010;

b = 4'b0101;

cin = 1'b1;

control = 3'b111;

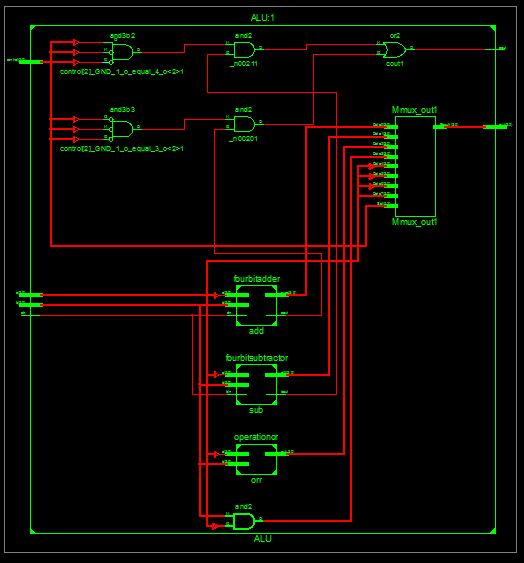
#100;

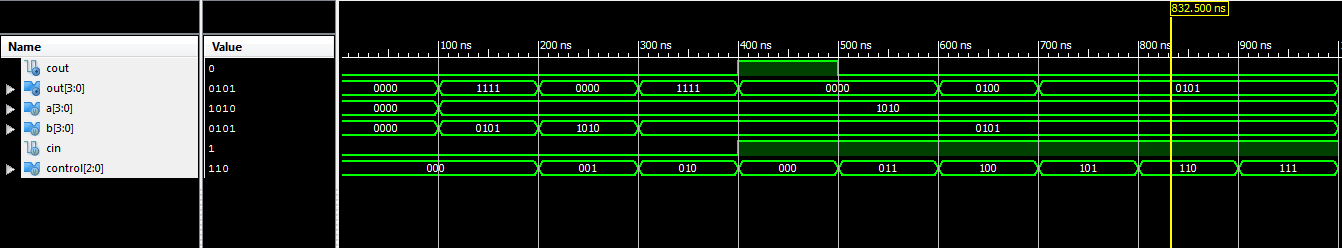
// Add stimulus here

end

endmodule

Schematic:



Timing Diagram: 

Conclusion: I successfully created an ALU with 8 functions consisting of adding, subtracting, bitwise or, bitwise and, shift left, shift right, rotate right, and rotate left.