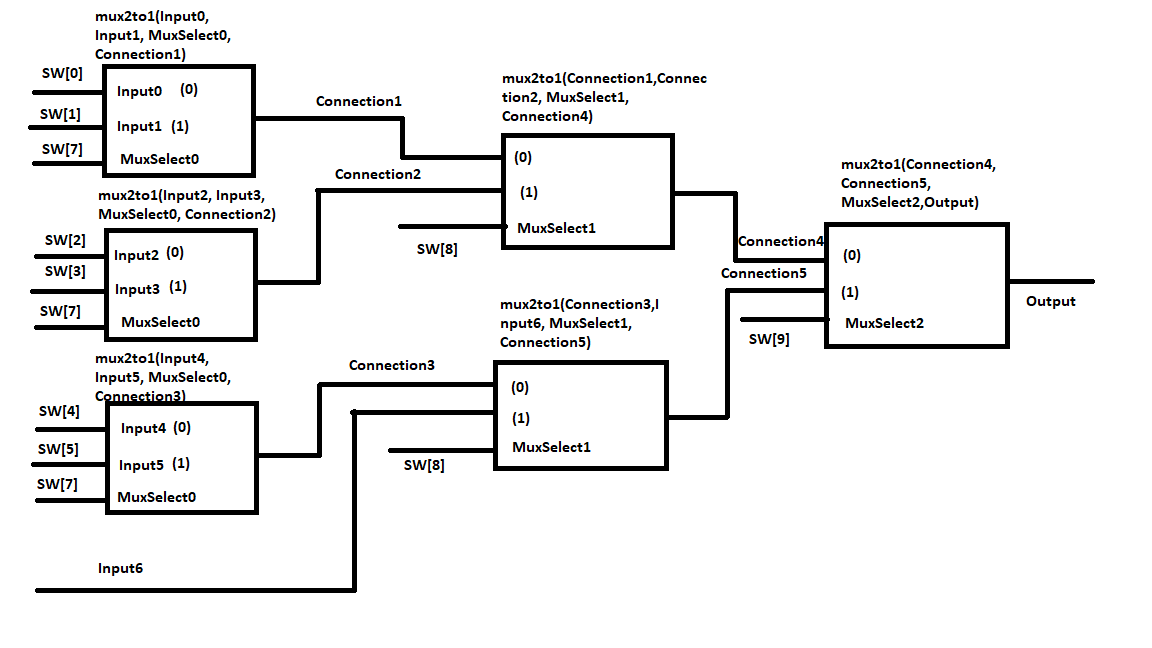
**Lab 3 Pre-lab Report**

Ziyao Chen

1002453409

**Part 1**







module part1(SW, LEDR);

input [9:0] SW;

output reg [9:0] LEDR;

seventoonemux u0(

.Input(SW[6:0]),

.MuxSelect(SW[9:7]),

.Out(LEDR[0]),

);

endmodule

module seventoonemux(Input, MuxSelect, Out);

input [6:0] Input;

input [2:0] MuxSelect;

output reg Out;

always @(\*)

begin

case (MuxSelect[2:0])

3'b000: Out = Input[0];

3'b001: Out = Input[1];

3'b010: Out = Input[2];

3'b011: Out = Input[3];

3'b100: Out = Input[4];

3'b101: Out = Input[5];

3'b110: Out = Input[6];

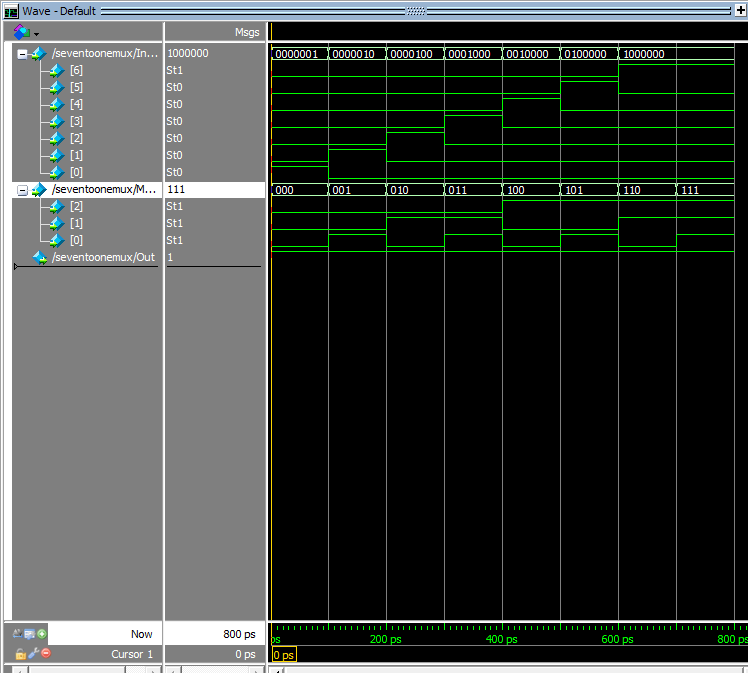
default: Out = Input[6];

endcase

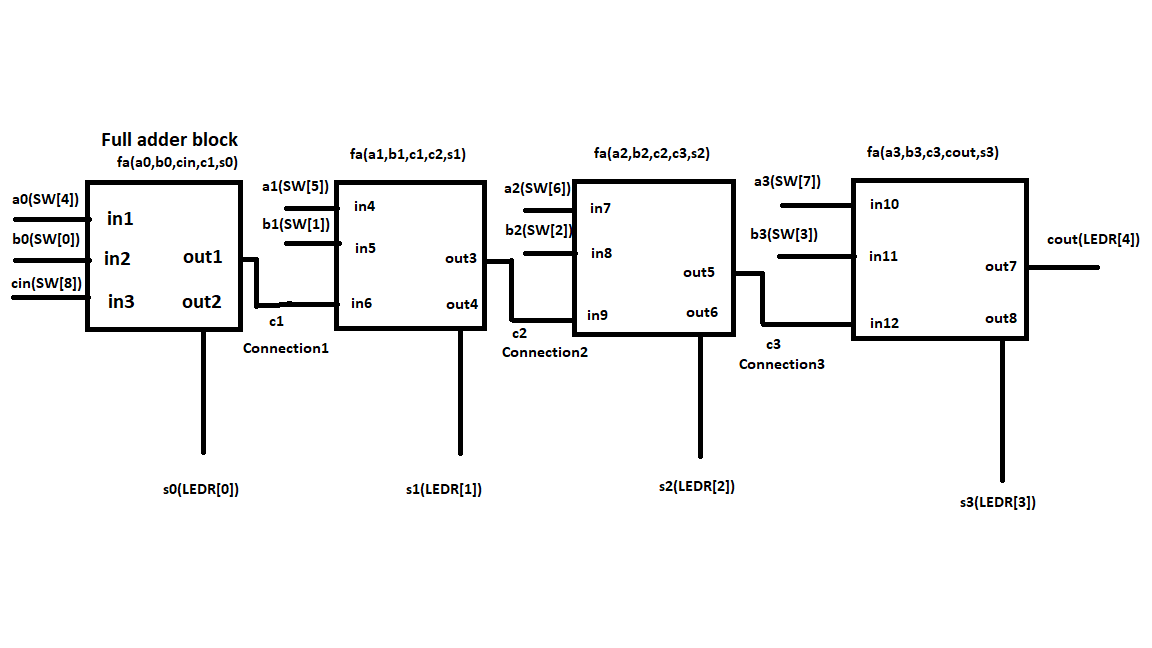
end

endmodule





**Part 2**

1. 

module ripplecarryadder(A, B, cin, S, cout);

input [3:0] A;

input [3:0] B;

input cin;

output [3:0] S;

output cout;

wire Connection1, Connection2, Connection3;

fulladder f1(A[0], B[0], cin, Connection1, S[0]);

fulladder f2(A[1], B[1], Connection1, Connection2, S[1]);

fulladder f3(A[2], B[2], Connection2, Connection3, S[2]);

fulladder f4(A[3], B[3], Connection3, cout, S[3]);

endmodule

module fulladder(a, b, carryin, carryout, sum);

input a, b, carryin; // adding a and b with carryin from the previous bit's carryout

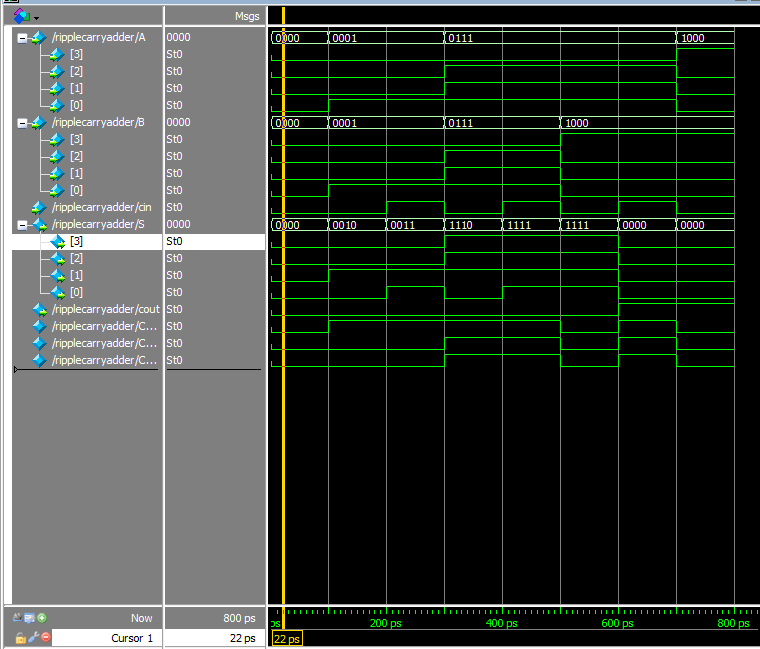
output carryout, sum; // carryout for the next bit's carryin, and sum for XOR of a and b

assign carryout = (a && b) | (b && carryin) | (a && carryin);

assign sum = a ^ b ^ carryin;

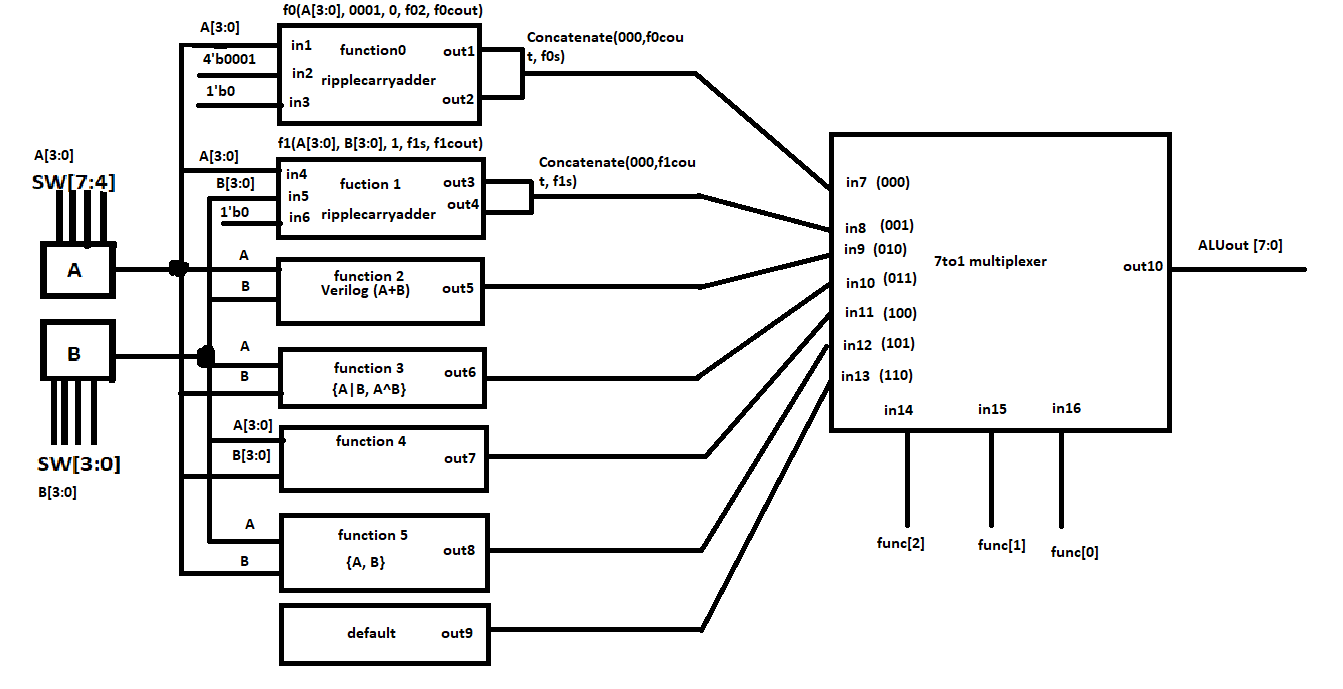
endmodule





**Part3**







module alu(A,B,func,ALUout);

input [3:0] A;

input [3:0] B;

input [2:0] func;

output [7:0] ALUout;

reg [7:0] ALUout;

wire f0s, f0cout; // output storage for f0

wire f1s, f1cout; // output storage for f1

ripplecarryadder f0(A[3:0], 0001, 0, f0s, f0cout); //setup for function0

ripplecarryadder f1(A[3:0], B[3:0], 0, f1s, f1cout); //setup for function1

always @(\*)

begin

case (func[2:0])

3'b000: ALUout[7:0] = {3'b000,f0cout,f0s};

3'b001: ALUout[7:0] = {3'b000, f1cout, f1s};

3'b010: ALUout[7:0] = A+B;

3'b011: ALUout[7:0] = {A|B, A^B};

3'b100: ALUout[7:0] = {7'b0000\_000, A[3]|A[2]|A[1]|A[0]|B[3]|B[2]|B[1]|B[0]};

3'b101: ALUout[7:0] = {A,B};

default: ALUout[7:0] = 8'b0000\_0000;

endcase

end

endmodule

module ripplecarryadder(A, B, cin, S, cout);

input [3:0] A;

input [3:0] B;

input cin;

output [3:0] S;

output cout;

wire Connection1, Connection2, Connection3;

fulladder f1(A[0], B[0], cin, Connection1, S[0]);

fulladder f2(A[1], B[1], Connection1, Connection2, S[1]);

fulladder f3(A[2], B[2], Connection2, Connection3, S[2]);

fulladder f4(A[3], B[3], Connection3, cout, S[3]);

endmodule

module fulladder(a, b, carryin, carryout, sum);

input a, b, carryin; // adding a and b with carryin from the previous bit's carryout

output carryout, sum; // carryout for the next bit's carryin, and sum for XOR of a and b

assign carryout = (a && b) | (b && carryin) | (a && carryin);

assign sum = a ^ b ^ carryin;

endmodule



