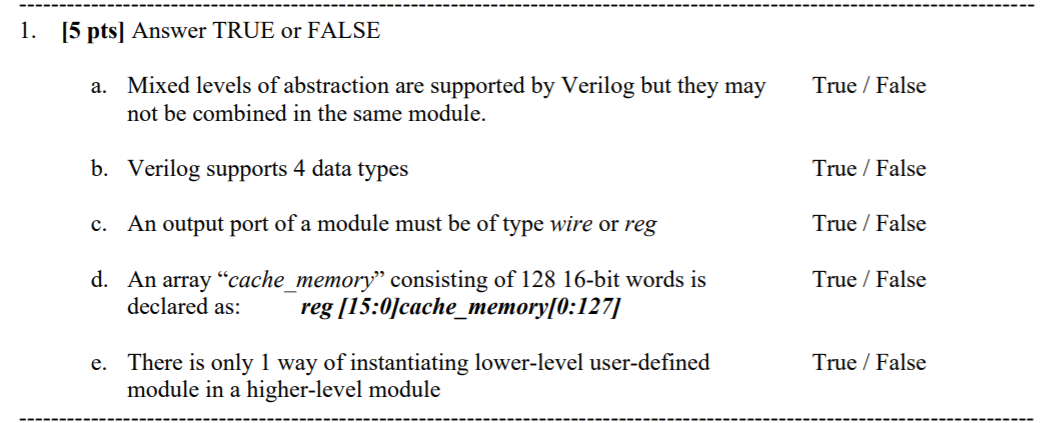
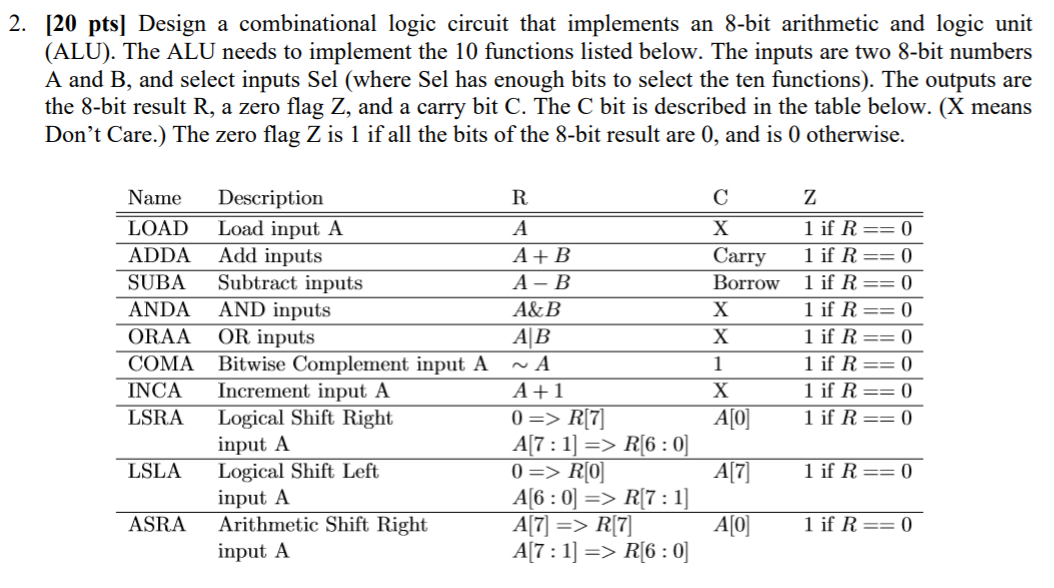
Chase Mulder

EGR 224 Homework #6

4/7/2020



1. True
2. True
3. True
4. False
5. False

****

**8- bit ALU Code**

`timescale 1ns / 1ps

module ALU(A,B,R,C,Z,SEL,FLAG);

input [7:0] A,B;

input [3:0] SEL;

output reg [7:0] R = 7'b0; //result

output reg C = 0; //carry

output reg Z = 0;

output reg [8:0] FLAG = 8'b0; // test for carry flag

parameter LOAD = 4'b0000;

parameter ADDA = 4'b0001;

parameter SUBA = 4'b0010;

parameter ANDA = 4'b0011;

parameter ORAA = 4'b0100;

parameter COMA = 4'b0101;

parameter INCA = 4'b0110;

parameter LSRA = 4'b0111;

parameter LSLA = 4'b1000;

parameter ASRA = 4'b1001;

always @ (A or B or SEL)

begin

case(SEL)

LOAD: // LOAD

begin

R = A; // set result to A

C = 1'bx; // SET C to don't care

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

ADDA: // ADD

begin

R = A+B; // compute A + B in 8 bit

FLAG = A+B; // compute A + B for 9 bit

C = FLAG[8]; // take 9th bit as carry flag

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

SUBA: // SUBTRACT

begin

R = A-B; // compute A-B

if (B>A) C = 1; //carry if the second number is greater

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

ANDA: // AND

begin

R = A&B; //and gate

C = 1'bx;

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

ORAA: // OR

begin

R = A|B; //or gate

C = 1'bx;

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

COMA: // COMPLEMENT

begin

R = ~A; //inverter gate

C = 1;

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

INCA: // INCREMENT

begin

R = A+1; //A + 1

C = 1'bx;

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

LSRA: // SHIFTRIGHT

begin

R[7] = 0;

R[6:0] = A[7:1];

C = A[0];

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

LSLA: // SHIFTLEFT

begin

R[0] = 0;

R[7:1] = A[6:0];

C = A[7];

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

ASRA: // ARITHSHIFT RIGHT

begin

R[7] = A[7];

R[6:0] = A[7:1];

C = A[0];

if (R ==0) Z = 1; // set z to 1 if result is 0

else Z = 0;

end

endcase

end

endmodule

**8-bit ALU Testbench**

`timescale 1ns / 1ps

module ALU\_tb();

reg [7:0] A,B;

reg [3:0] SEL;

wire [7:0] R;

wire C,Z;

wire [8:0] FLAG;

ALU A1 (A,B,R,C,Z,SEL,FLAG);

initial

begin

A = 7'b0000010; //input a

B = 7'b0000101;// input b

SEL = 4'b0000; // load

# 10 SEL = 4'b0001;// add

# 10 SEL = 4'b0010; // sub

# 10 SEL = 4'b0011; // and

# 10 SEL = 4'b0100; // or

# 10 SEL = 4'b0101; // com

# 10 SEL = 4'b0110; // inc

# 10 SEL = 4'b0111; // lsr

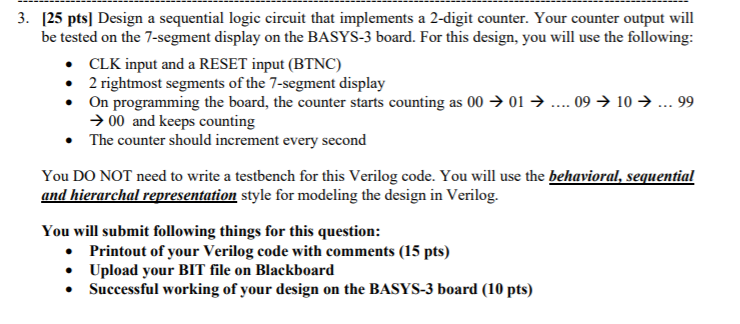
# 10 SEL = 4'b1000; // lsl

# 10 SEL = 4'b1001; // asr

A = 8'b10000001;

end

endmodule



**2-digit counter code**

module Counter\_two\_top (clk,anode,segment);

input clk;

output [3:0] anode;

output [6:0] segment;

///mux inputs/outputs

wire [6:0]seg0;

wire [6:0]seg1;

wire [6:0]seg2;

wire [6:0]seg3;

wire [1:0]two\_bit\_out;

//zero

reg [3:0] zeros1 = 4'b0000;

reg [3:0] zeros2 = 4'b0000;

// clock divide outputs

wire one\_khz\_out;

wire ten\_hz\_out;

wire one\_hz\_out;

//counter outputs

wire [3:0] count\_to\_nine\_out\_1;

wire [3:0] count\_to\_nine\_out\_2;

mux M1(seg0,seg1,seg2,seg3,two\_bit\_out,anode,segment);

seven\_segment S0(zeros1, seg0);

seven\_segment S1(zeros2,seg1);

seven\_segment S2(count\_to\_nine\_out\_1, seg2);

seven\_segment S3(count\_to\_nine\_out\_2,seg3);

counter\_nine N1(ten\_hz\_out,count\_to\_nine\_out\_1);

counter\_nine N2(one\_hz\_out,count\_to\_nine\_out\_2);

counter\_two\_bit C1(one\_khz\_out,two\_bit\_out);

onek\_hz D1(clk,one\_khz\_out);

ten\_hz D2 (clk,ten\_hz\_out);

one\_hz D3 (clk,one\_hz\_out);

endmodule

***Bit file attached on blackboard***