California State University, Chico
Department of Electrical and Computer Engineering

EECE 343 Advanced Logic Design

ALU

by

Moses P. Mccabe & Anthony Arevalo

March 12, 2019

Simulation

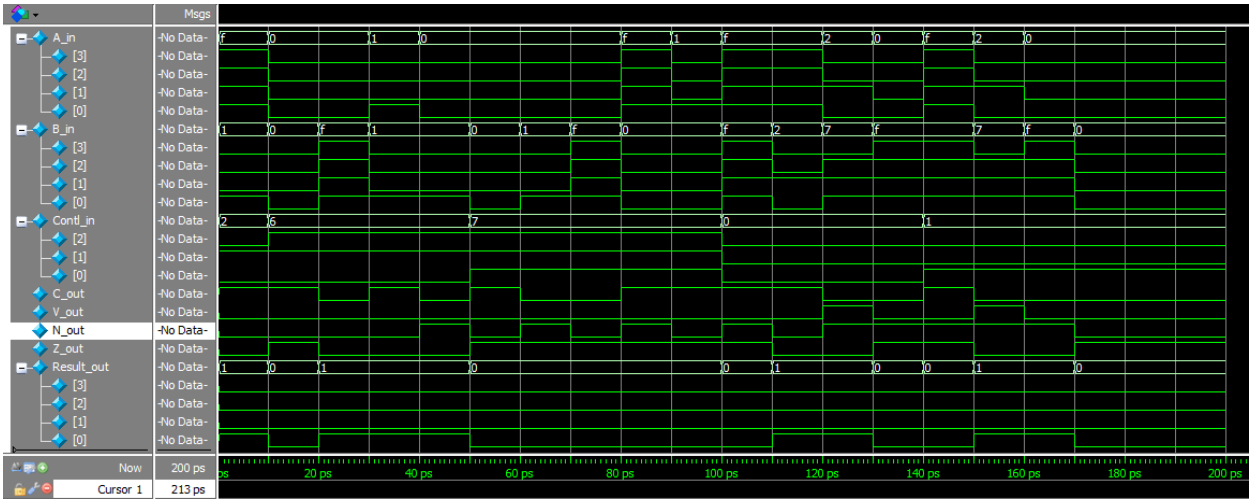
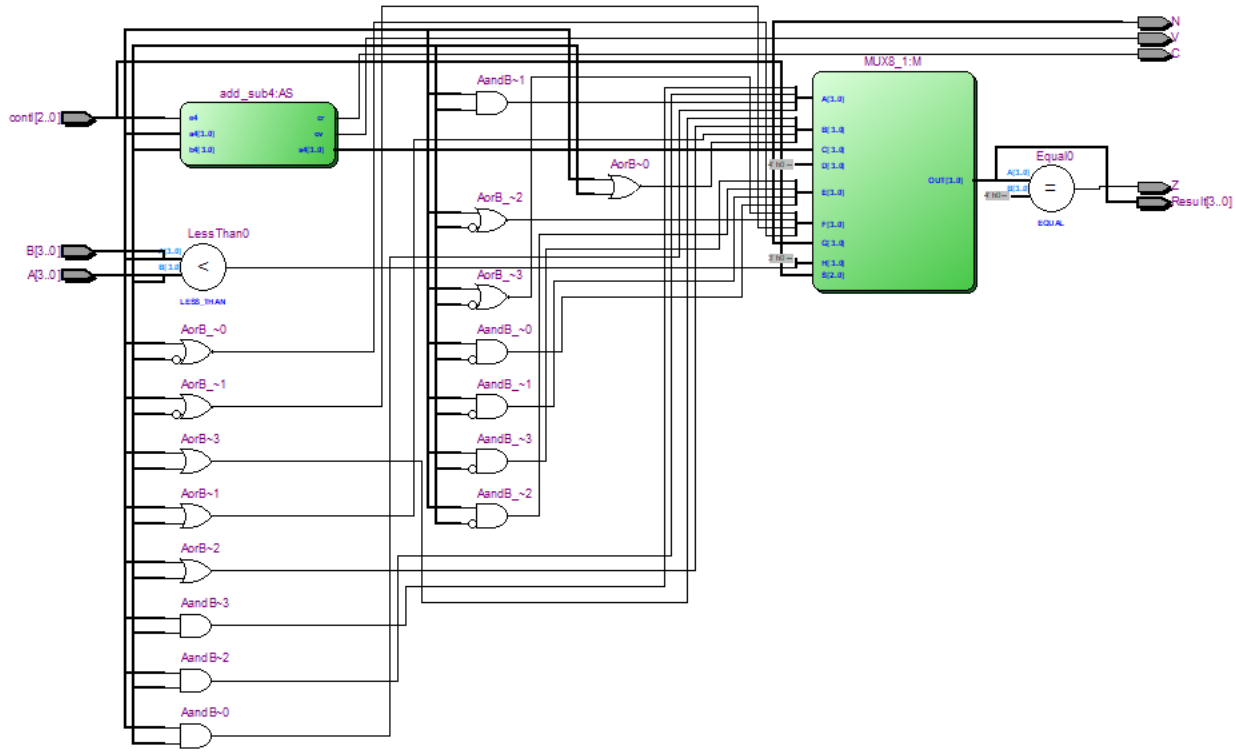


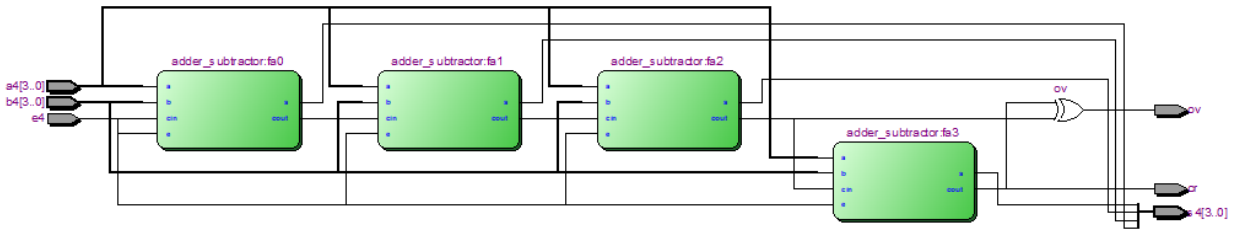
Table 2

Test (hex)	F[2:0]	A (hex)	B (hex)	Y (hex)	N	V	Z	C
ADD 0+0	2	0	0	0	0	0	1	0
ADD 0+(-1)	2	0	F	F	1	0	0	0
ADD 1+(-1)	2	1	F	0	0	0	1	0
ADD F+1	2	F	1	1	0	0	0	1
SUB 0-0	6	0	0	0	0	0	1	1
SUB 0-(-1)	6	0	F	1	0	0	0	0
SUB 1-1	6	1	1	1	0	0	0	1
SUB 0-1	6	0	1	1	1	0	0	0
SLT 0,0	7	0	0	0	0	0	1	1
SLT 0,1	7	0	1	0	1	0	1	0
SLT 0,-1	7	0	F	0	0	0	1	0
SLT 1,0	7	1	0	0	0	0	1	1
SLT -1,0	7	F	0	0	1	0	1	1
AND F,F	0	F	F	0	1	0	1	1
AND F,2	0	F	2	1	0	0	0	1
AND 2,7	0	2	7	1	1	1	0	0
AND 0,F	0	0	F	0	1	0	1	0
OR F,F	1	F	F	0	1	0	1	1
OR 2,7	1	2	7	1	1	1	0	0
OR 0,F	1	0	F	1	1	0	0	0
OR 0,0	1	0	0	0	0	0	1	0

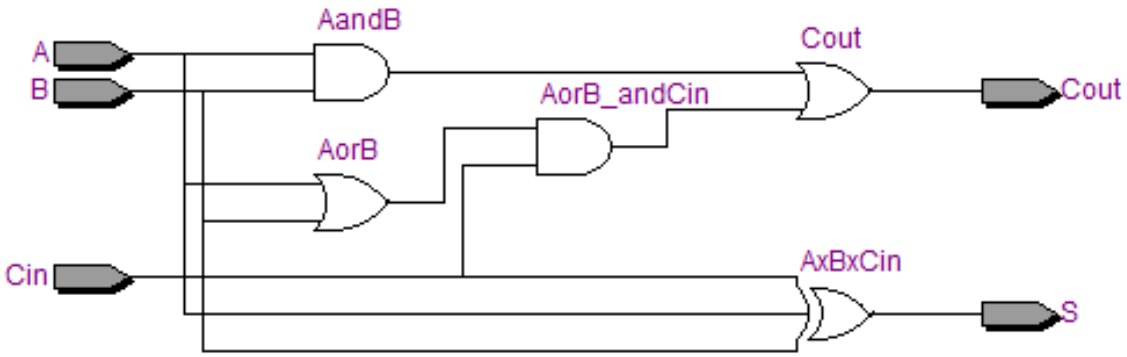
ALU



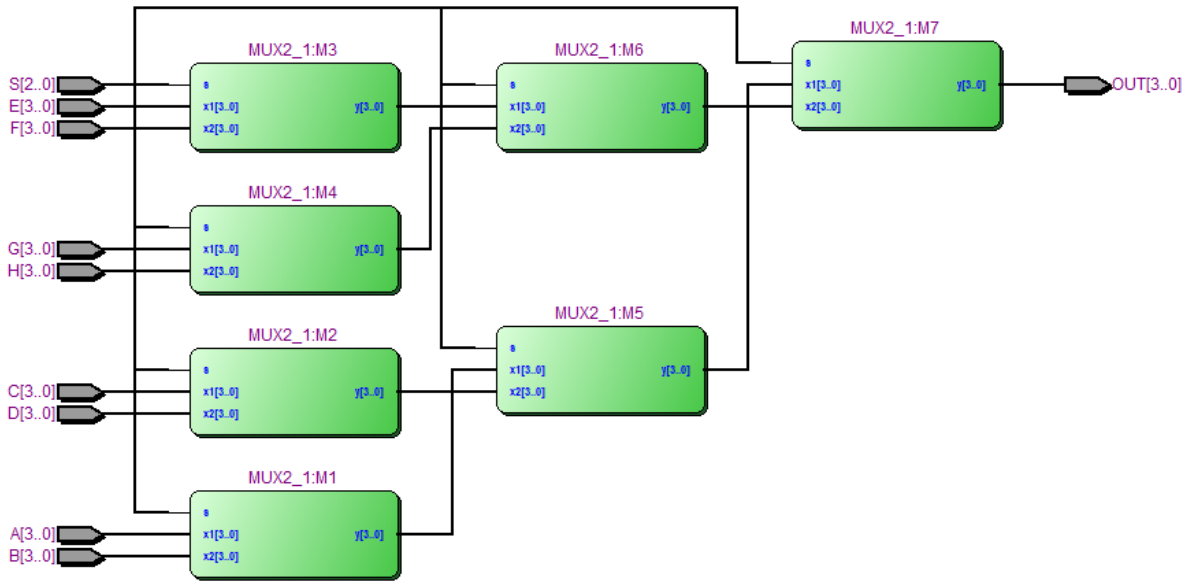
Adder and Subst.



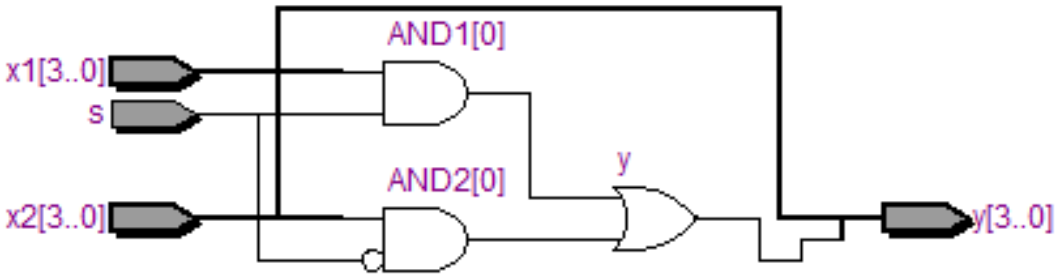
Inside one of the Adder/Subst



8X1 MUX



Inside the one of the MUX



Date: March 12, 2019 ALU.v Project: adder_subtractor

```
1   // ALU
2   module ALU(A,B,contl,C,V,N,Z,Result);
3   //input variables
4   input [3:0] A,B;
5   input [2:0] contl;
6   //output variables
7   output C, V, N, Z;
8   output [3:0] Result;
9   // wire use to connect ALU
10  wire [3:0]notB,AandB,AorB,AorB_,AandB_,S_as,andsubT,notUse,AsubB;
11
12  assign notB = ~B;              // Not gate
13  assign AandB = A &B;           // AND gate
14  assign AandB_ = A & notB;      // NOT AND gate
15  assign AorB = A | B;           // OR gate
16  assign AorB_ = A | notB;       // NOT OR gate
17  assign notUse = 0;             // assign to GND
18  // equivalent to comparing the MSB of A and B
19  assign AsubB = ($signed(A) < $signed(B))?1:0;
20
21     // Passing value to full adder
22     add_sub4 AS(.a4(A),.b4(B),.e4(contl[2]),.s4(S_as),.cr(C),.ov(V));
23
24
25     // assigning input and output to 8-1 MUX
26     MUX8_1 M(.A(AandB),.B(AorB),.C(S_as),.D(notUse),.E(AandB_),.F(AorB_),.G(S_as),.H(
       AsubB),.S(contl),.OUT(Result));
27
28
29
30  assign N = S_as[3];       // assign  N to MSB of full adder solution
31
32  assign Z = (Result == 4'b0000) ? 1 : 0;
33  endmodule
34
```

Date: March 12, 2019 MUX8_1.v Project: adder_subtractor

```
1   // 8x1 MUX
2   module MUX8_1(A,B,C,D,E,F,G,H,S,OUT);
3   input [3:0]A,B,C,D,E,F,G,H;                          // inputs
4   input [2:0] S;                                       // control
5   output [3:0]OUT;                                     // output
6   wire [3:0]MUX1,MUX2,MUX3,MUX4,MUX5,MUX6;             // wire use to connect MUX
7
8   MUX2_1 M1(.y(MUX1),.x1(A),.x2(B),.s(S[0]));          // MUX 1
9   MUX2_1 M2(.y(MUX2),.x1(C),.x2(D),.s(S[0]));          // MUX 2
10  MUX2_1 M3(.y(MUX3),.x1(E),.x2(F),.s(S[0]));          // MUX 3
11  MUX2_1 M4(.y(MUX4),.x1(G),.x2(H),.s(S[0]));          // MUX 4
12  MUX2_1 M5(.y(MUX5),.x1(MUX1),.x2(MUX2),.s(S[1]));    //MUX 5
13  MUX2_1 M6(.y(MUX6),.x1(MUX3),.x2(MUX4),.s(S[1]));    // MUX 6
14  MUX2_1 M7(.y(OUT),.x1(MUX5),.x2(MUX6),.s(S[2]));     // MUX 7
15  endmodule
16
```

Date: March 12, 2019 add_sub4.v Project: adder_subtractor

```
1   module add_sub4(a4,b4,e4,s4,cr,ov);
2
3   input e4;
4   input [3:0]a4,b4;
5   output cr,ov;
6   output [3:0]s4;
7
8   wire co0,co1,co2;
9
10  assign ov = co2^cr;
11
12  adder_subtractor fa0(a4[0],b4[0],e4,e4,s4[0],co0);
13  adder_subtractor fa1(a4[1],b4[1],co0,e4,s4[1],co1);
14  adder_subtractor fa2(a4[2],b4[2],co1,e4,s4[2],co2);
15  adder_subtractor fa3(a4[3],b4[3],co2,e4,s4[3],cr);
16
17  endmodule
```

Date: March 12, 2019 adder_subtractor.v Project: adder_subtractor

```
1   module adder_subtractor(a,b,cin,e,s,cout);
2
3   input a,b,cin,e;
4   output s,cout;
5
6   wire xr;
7
8   assign xr = b^e;
9
10  assignment1 a1(a,xr,cin,cout,s);
11
12  endmodule
13
14
```

```
1   // ALU Testbench
2   module ALU_test();
3   reg [3:0] A_in,B_in;
4   reg [2:0] Contl_in;
5   //output variables
6   wire C_out, V_out, N_out, Z_out;
7   wire [3:0] Result_out;
8
9   //module ALU(A,B,contl,C,V,N,Z,Result)
10  ALU A(A_in,B_in,Contl_in,C_out,V_out,N_out,Z_out,Result_out);
11
12  initial
13  begin
14  // when A_in and B_in = 0000
15  A_in = 4'hF; B_in = 4'h1; Contl_in = 3'd2;
16  #10
17  A_in = 4'h0; B_in = 4'h0; Contl_in = 3'd6;
18  #10
19  A_in = 4'h0; B_in = 4'hF; Contl_in = 3'd6;
20  #10
21  A_in = 4'h1; B_in = 4'h1; Contl_in = 3'd6;
22  #10
23  A_in = 4'h0; B_in = 4'h1; Contl_in = 3'd6;
24  #10
25  A_in = 4'h0; B_in = 4'h0; Contl_in = 3'd7;
26  #10
27  A_in = 4'h0; B_in = 4'h1; Contl_in = 3'd7;
28  #10
29  A_in = 4'h0; B_in = 4'hF; Contl_in = 3'd7;
30  #10
31  A_in = 4'hF; B_in = 4'h0; Contl_in = 3'd7;
32  #10
33  A_in = 4'h1; B_in = 4'h0; Contl_in = 3'd7;
34  #10
35  A_in = 4'hF; B_in = 4'hF; Contl_in = 3'd0;
36  #10
37  A_in = 4'hF; B_in = 4'h2; Contl_in = 3'd0;
38  #10
39  A_in = 4'h2; B_in = 4'h7; Contl_in = 3'd0;
40  #10
41  A_in = 4'h0; B_in = 4'hF; Contl_in = 3'd0;
42  #10
43  A_in = 4'hF; B_in = 4'hF; Contl_in = 3'd1;
44  #10
45  A_in = 4'h2; B_in = 4'h7; Contl_in = 3'd1;
46  #10
47  A_in = 0; B_in = 4'hF; Contl_in = 3'd1;
48  #10
49  A_in = 4'h0; B_in = 4'h0; Contl_in = 3'd1;
50  end
51  endmodule
52
```

```
1   // 2x1 MUX
2   module MUX2_1(y, x1,x2,s);
3      input [3:0]x1, x2;
4      input s;
5      output [3:0]y;
6      wire [3:0]S_not, AND1, AND2;
7      assign S_not=~s;
8      //not (S_not,s);
9      assign AND1 = x1 & s;
10     assign AND2 = S_not & x2;
11     assign y = AND1 | AND2;
12  endmodule
```