CSE331
ASSIGNMENT 2
REPORT

Ahmet Ergani
161044011

Summarization:

I have implemented:

I have implemented half, full and 32bit adder and subtractor; 2to1 and 8to1 multiplexer; left and right shifter; and the 32bit versions of the gates: AND OR XOR NOR. Then in the alu32 module, I calculated the results of additions, subtractions, shifters and gates and gave them as inputs to the 8to1 multiplexer and gave the operation specifiers as selection inputs.

Explanation for modules:

ADDERS:

I implemented the basic half adder. Then I implemented the full adder using this half adder. Finally I implemented the 32 bit adder by using full adder 32 times.

```
module half_adder(sum, carry_out, a, b);
input a, b;
output sum, carry_out;

xor sum_of_digits(sum, a, b);
and carry_of_sum(carry_out, a, b);

endmodule
```

```
module full_adder(sum, carry_out, a, b, carry_in);
input a, b, carry_in;
output sum, carry_out;
wire temp_sum, first_carry_out, second_carry_out;

half_adder first_sum(temp_sum, first_carry_out, a, b);
half_adder second_sum(sum, second_carry_out, temp_sum, carry_in);

or final_carry_out(carry_out, second_carry_out, first_carry_out);

endmodule
```

```
module adder_32bit(S,C,A,B,C0);
input [31:0] A,B;
input C0;
output C;
output [31:0] S;
wire C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16,C17,C18,C

full_adder FA0(S[0], C1, A[0], B[0], C0),
           FA1(S[1], C2, A[1], B[1], C1),
           FA2(S[2], C3, A[2], B[2], C2),
           FA3(S[3], C4, A[3], B[3], C3),
```

SUBTRACTOR:

I reversed the input using the not gate for each bit. Then I gave it as input to 32 bit adder and finally I added 1 to the result using -again- the 32 bit adder

```
1   module subtractor_32bit(final_sub,sign,A,B);
2   input [31:0] A,B;
3   output [31:0] final_sub;
4   output sign;
5   wire [31:0] reversedB,first_sub;
6   wire tempsign1,tempsign2;
7
8   not(reversedB[0],B[0]);
9   not(reversedB[1],B[1]);
10  not(reversedB[2],B[2]);
```

```
37  not(reversedB[29],B[29]);
38  not(reversedB[30],B[30]);
39  not(reversedB[31],B[31]);
40
41  adder_32bit subtract(first_sub,tempsign1,A,reversedB,1'b0);
42  adder_32bit finalize(final_sub,tempsign2,first_sub,1,1'b0);
43  or(sign,tempsign1,tempsign2);
44
45  endmodule
```

AND OR NOR XOR:

I implemented the 32 bit version of this gates by using the subject gate for each bits of my inputs. (Going to give just one example because logic is same amongst them)

```
1   module or_32bit(R,A,B);
2   input [31:0] A,B;
3   output [31:0] R;
4
5   or(R[0],A[0],B[0]);
6   or(R[1],A[1],B[1]);
7   or(R[2],A[2],B[2]);
8   or(R[3],A[3],B[3]);
9   or(R[4],A[4],B[4]);
10  or(R[5],A[5],B[5]);
11  or(R[6],A[6],B[6]);
12  or(R[7],A[7],B[7]);
13  or(R[8],A[8],B[8]);
14  or(R[9],A[9],B[9]);
15  or(R[10],A[10],B[10]);
16  or(R[11],A[11],B[11]);
17  or(R[12],A[12],B[12]);
```

MULTIPLEXERS:

I implemented the basic 2to1 mux. Then by using 7 of them, I implemented the 8to1 mux. Also -for my ease- I implemented a 2to1 mux that takes 32 bit inputs too.

```
module mux_2x1(R,A,B,S);
input A,B,S;
output R;
wire t1,t2,t3;

not(t1,S);
and(t2,B,S);
and(t3,t1,A);
or(R,t2,t3);

endmodule
```

```
module mux_2x1_32bit(R,A,B,S);
input [31:0] A,B;
input S;
output [31:0] R;

mux_2x1 mux0(R[0],A[0],B[0],S);
mux_2x1 mux1(R[1],A[1],B[1],S);
mux_2x1 mux2(R[2],A[2],B[2],S);
mux_2x1 mux3(R[3],A[3],B[3],S);
mux_2x1 mux4(R[4],A[4],B[4],S);
```

```
module mux_8x1(R,i0,i1,i2,i3,i4,i5,i6,i7,s0,s1,s2);
input [31:0] i0,i1,i2,i3,i4,i5,i6,i7;
input s0,s1,s2;
output [31:0] R;
wire [31:0] w0,w1,w2,w3,w4,w5;

mux_2x1_32bit  m0(w0,i0,i1,s0);
mux_2x1_32bit  m1(w1,i2,i3,s0);
mux_2x1_32bit  m2(w2,w0,w1,s1);


mux_2x1_32bit  m3(w3,i4,i5,s0);
mux_2x1_32bit  m4(w4,i6,i7,s0);
mux_2x1_32bit  m5(w5,w3,w4,s1);

mux_2x1_32bit  m6(R,w2,w5,s2);

endmodule

```

SHIFTERS:

This was the hardest part of the homework. I implemented the 32 bit version of the 8 bit right shifter that has been showed to us at the PS. Then by using that, I implemented the left shifter. I managed to do that by inverting the first input, shifting it right, then inverting the result again. The right shift that I used to left shift is a little bit different because of the need to use rightmost digit instead of the leftmost.

```
 1  module right_shift(R,A,B);
 2  input [31:0] A;
 3  input [4:0] B;
 4  output [31:0] R;
 5  wire [31:0] w0,w1,w2,w3;
 6
 7  mux_2x1 m0(w0[0],A[0],A[1],B[0]);
 8  mux_2x1 m1(w0[1],A[1],A[2],B[0]);
 9  mux_2x1 m2(w0[2],A[2],A[3],B[0]);
10  mux_2x1 m3(w0[3],A[3],A[4],B[0]);
11  mux_2x1 m4(w0[4],A[4],A[5],B[0]);
12  mux_2x1 m5(w0[5],A[5],A[6],B[0]);
13  mux_2x1 m6(w0[6],A[6],A[7],B[0]);
14  mux_2x1 m7(w0[7],A[7],A[8],B[0]);
15  mux_2x1 m8(w0[8],A[8],A[9],B[0]);
16  mux_2x1 m9(w0[9],A[9],A[10],B[0]);
17  mux_2x1 m10(w0[10],A[10],A[11],B[0]);
18  mux_2x1 m11(w0[11],A[11],A[12],B[0]);
```

```
102   mux_2x1 x29(w2[29],w1[29],w1[31],B[2]);
103   mux_2x1 x30(w2[30],w1[30],w1[31],B[2]);
104   buf(w2[31],w1[31]);
105
```

```
134   mux_2x1 y28(w3[28],w2[28],w2[31],B[3]);
135   mux_2x1 y29(w3[29],w2[29],w2[31],B[3]);
136   mux_2x1 y30(w3[30],w2[30],w2[31],B[3]);
137   buf(w3[31],w2[31]);
```

```
 1  module left_shift(R,A,B);
 2  input [31:0] A;
 3  input [4:0] B;
 4  output [31:0] R;
 5  wire [31:0] right,inverteded;
 6
 7  buf(inverteded[0],A[31]);
 8  buf(inverteded[1],A[30]);
 9  buf(inverteded[2],A[29]);
10  buf(inverteded[3],A[28]);
11  buf(inverteded[4],A[27]);
12  buf(inverteded[5],A[26]);
```

```
37  buf(inverteded[30],A[1]);
38  buf(inverteded[31],A[0]);
39
40  right_for_left a(right,inverteded,B);
41
42  buf(R[0],right[31]);
43  buf(R[1],right[30]);
44  buf(R[2],right[29]);
```

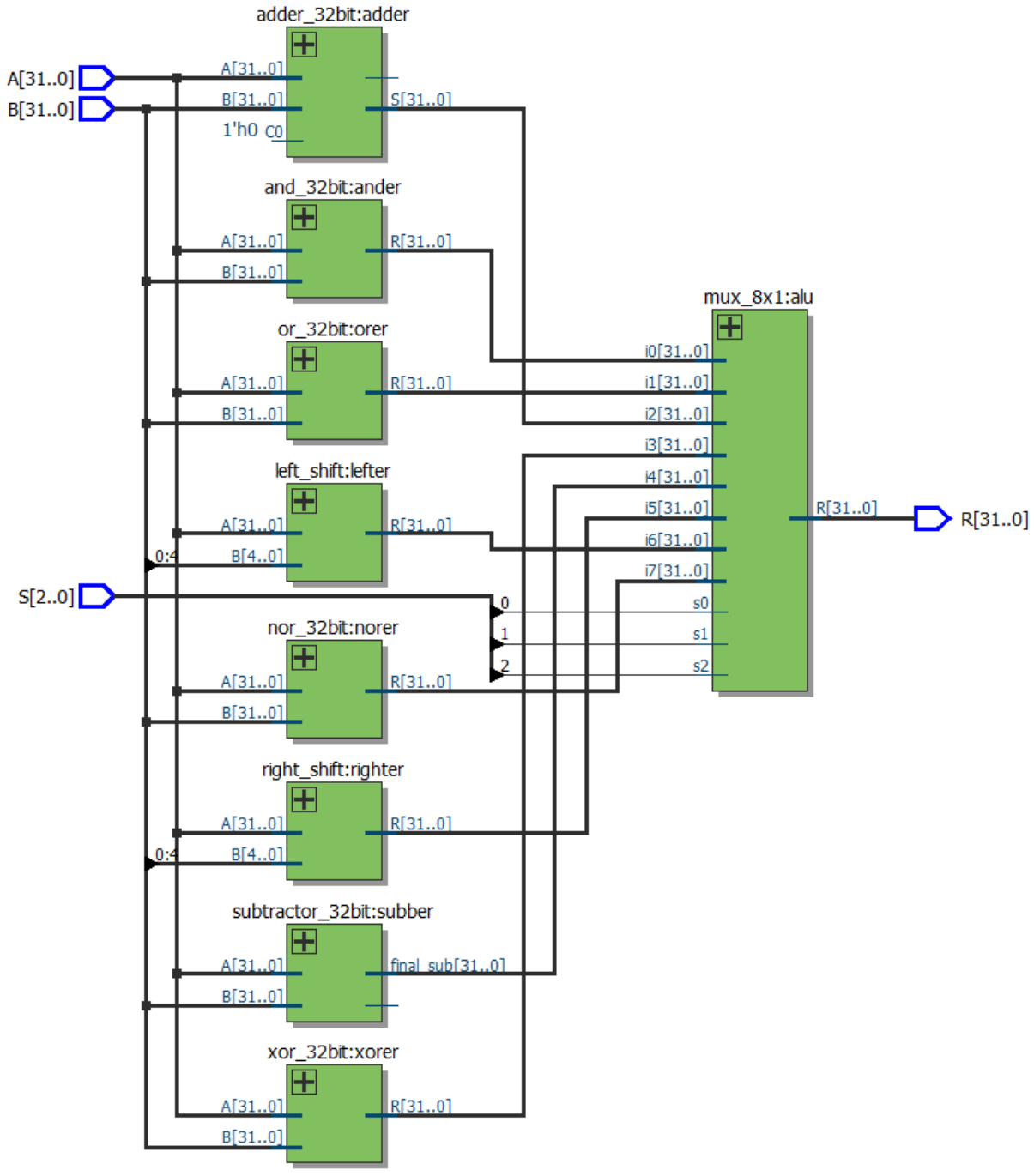
```
module right_for_left(R,A,B);
input [31:0] A;
input [4:0] B;
output [31:0] R;
wire [31:0] w0,w1,w2,w3;

mux_2x1 m0(w0[0],A[0],A[1],B[0]);
mux_2x1 m1(w0[1],A[1],A[2],B[0]);
mux_2x1 m2(w0[2],A[2],A[3],B[0]);
mux_2x1 m3(w0[3],A[3],A[4],B[0]);
mux_2x1 m4(w0[4],A[4],A[5],B[0]);
mux_2x1 m5(w0[5],A[5],A[6],B[0]);
mux_2x1 m6(w0[6],A[6],A[7],B[0]);
mux_2x1 m7(w0[7],A[7],A[8],B[0]);
mux_2x1 m8(w0[8],A[8],A[9],B[0]);
mux_2x1 m9(w0[9],A[9],A[10],B[0]);
mux_2x1 m10(w0[10],A[10],A[11],B[0]);
```

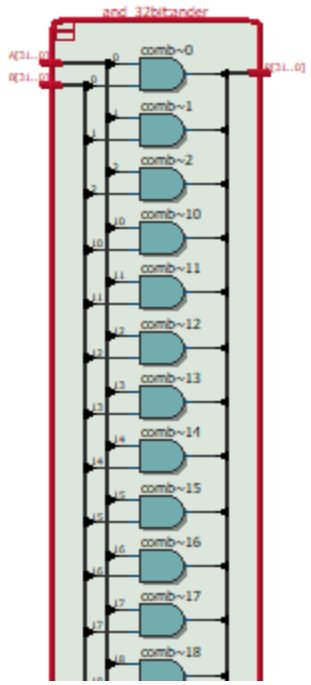
```
mux_2x1 m30(w0[30],A[30],A[31],B[0]);
mux_2x1 m31(w0[31],A[31],1'b0,B[0]);

mux_2x1 n0(w1[0],w0[0],w0[2],B[1]);
mux_2x1 n1(w1[1],w0[1],w0[3],B[1]);
mux_2x1 n2(w1[2],w0[2],w0[4],B[1]);
mux_2x1 n3(w1[3],w0[3],w0[5],B[1]);
```

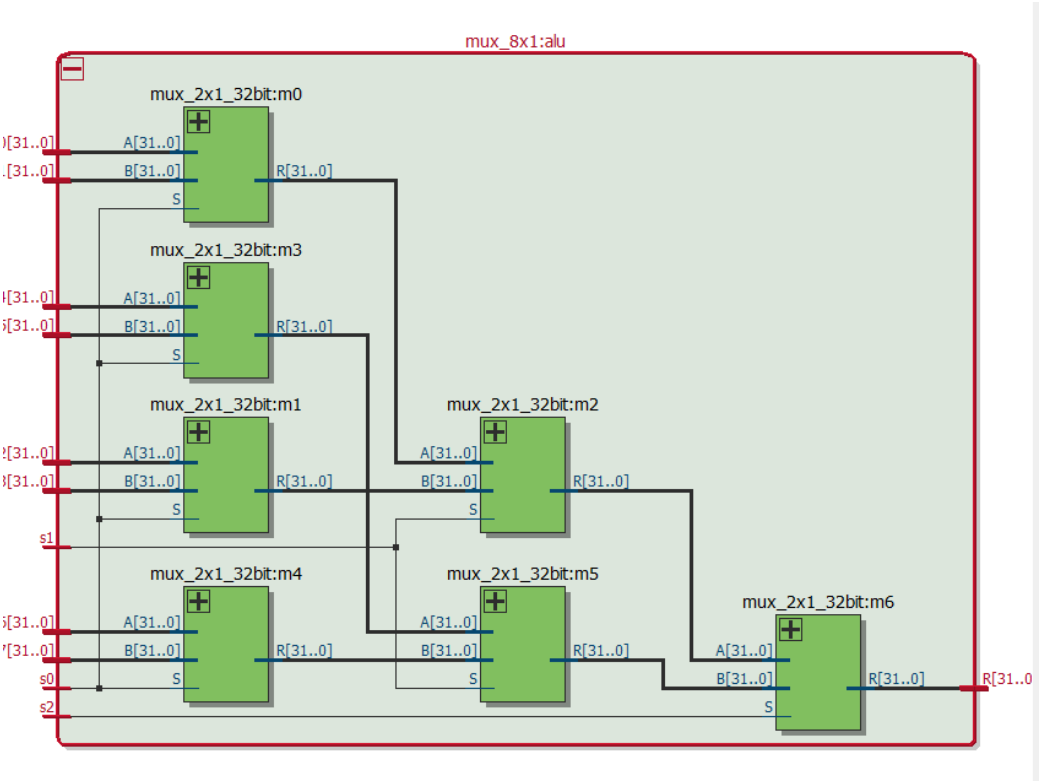
IMAGES OF THE STRUCTURE



My alu32

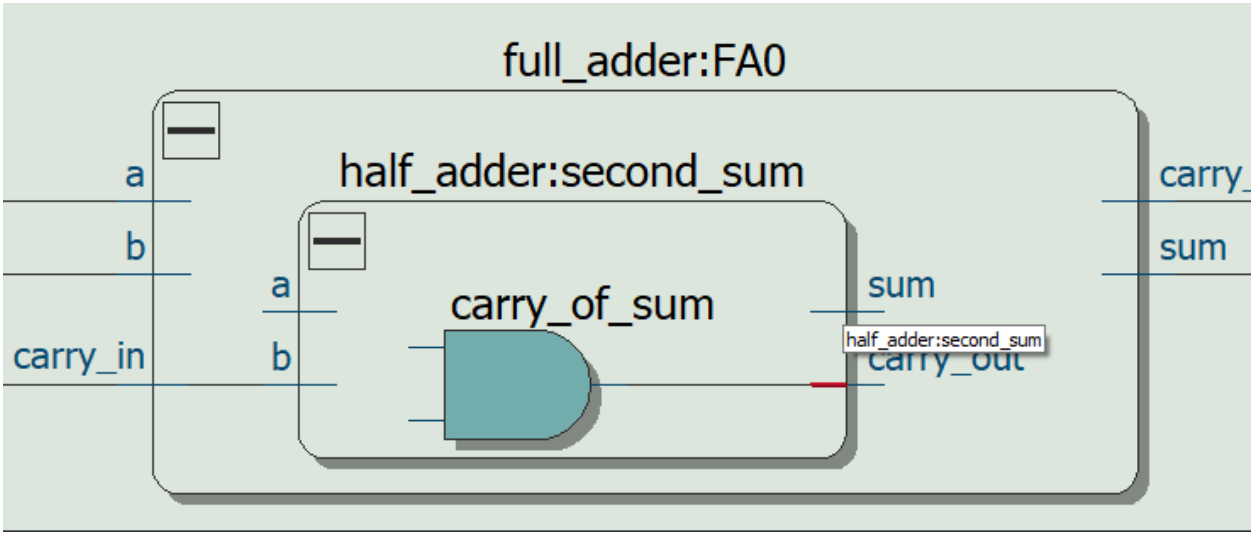


32 bit module using 32 AND gates

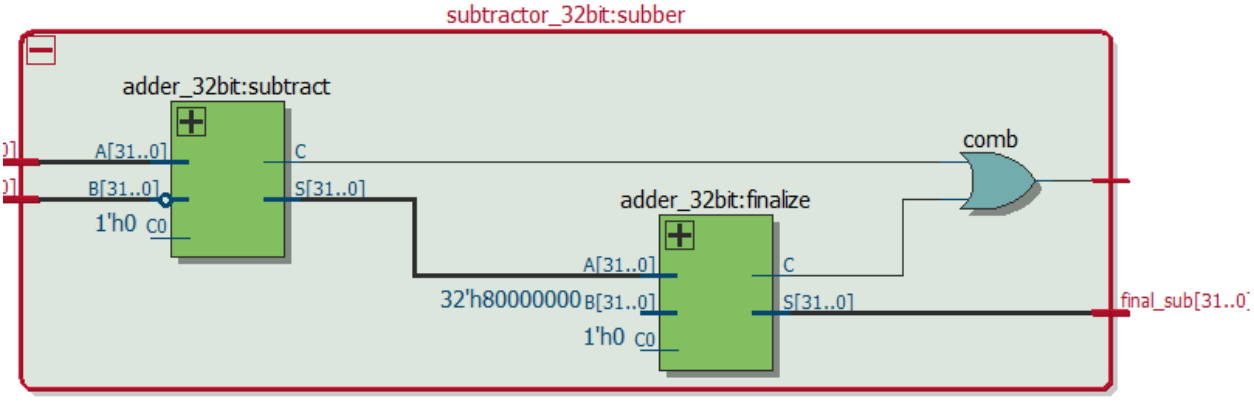


my 8to1 multiplexer which consists of 7 2to1 multiplexer

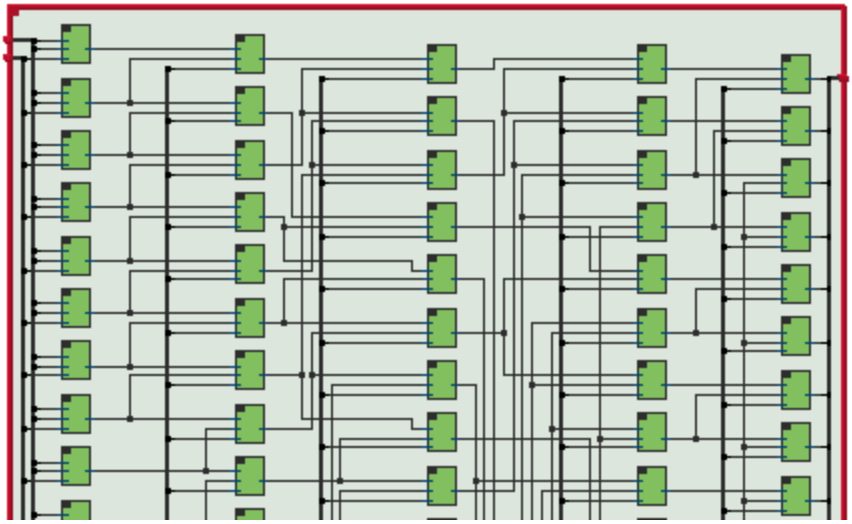
My full adders in 32 bit adder



A full adder



My 32 bit subtractor (uses 32bit adder as mentioned)



Right shifter the uses lots of 2to1 multiplexers

NOTES:

-I ignored the carry outs in addition and did not use the sign output

-I am late to upload this assignment because I was really ill (not a excuse just a context)