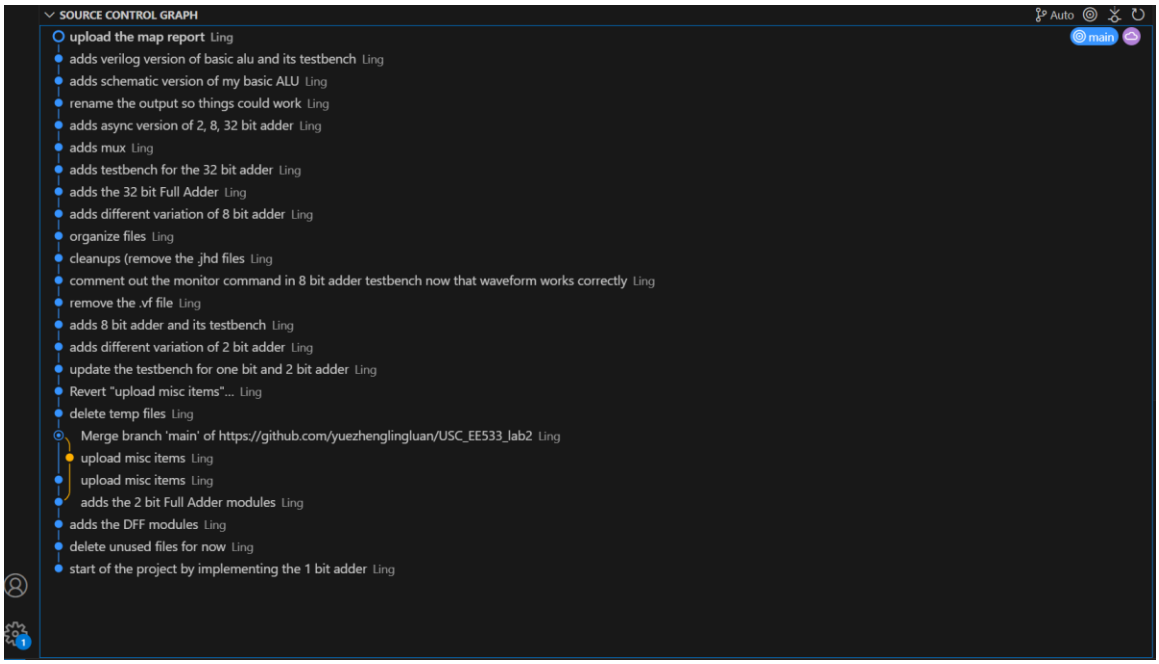
Here are my commit history of the github repo:




And the link to the public repo:

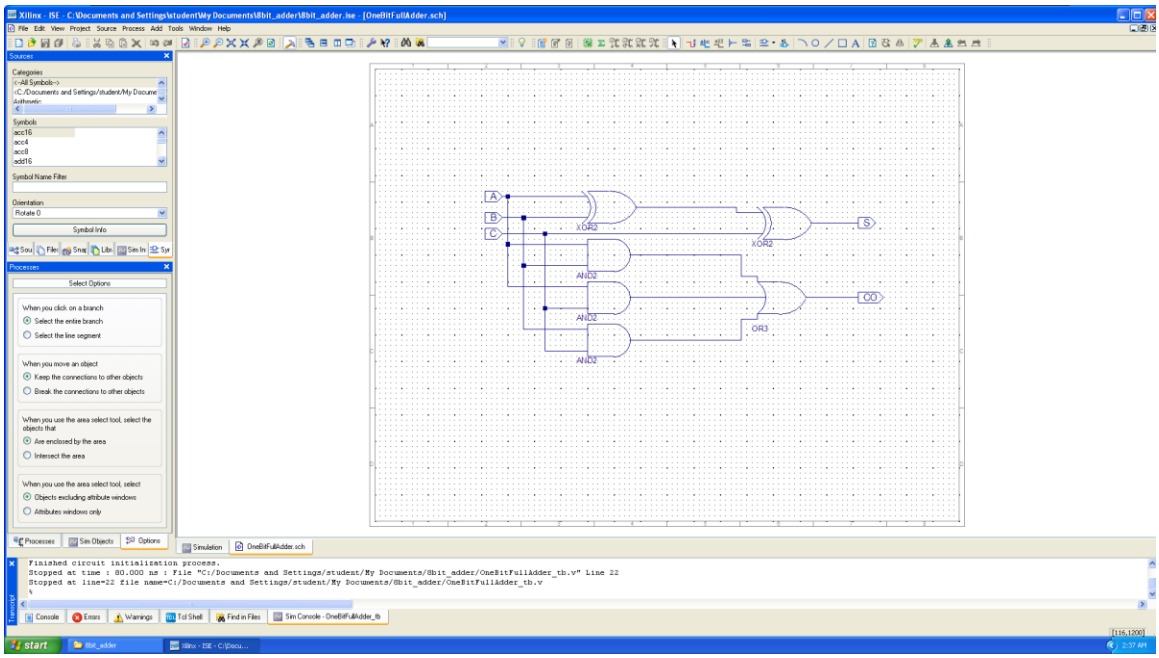
https://github.com/yuezhenglingluan/USC_EE533_lab2/

Part 1: The TA says on Piazza that we do not need to submit anything and the tutorial is a hands on for us to get comfortable with the tool:

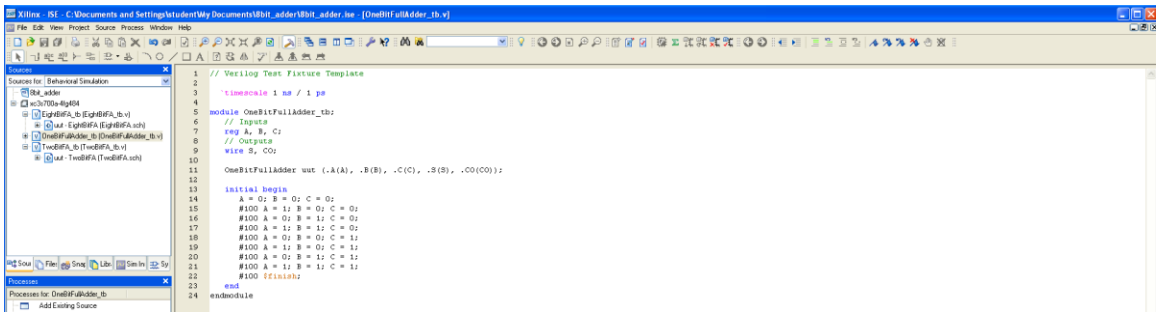
Part 2:

A: 1-bit Adder:

Here is my schematic of my 1-bit adder:

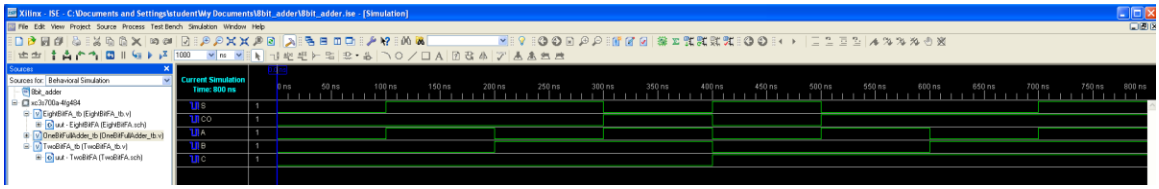



Here is my testbench for the 1-bit adder:



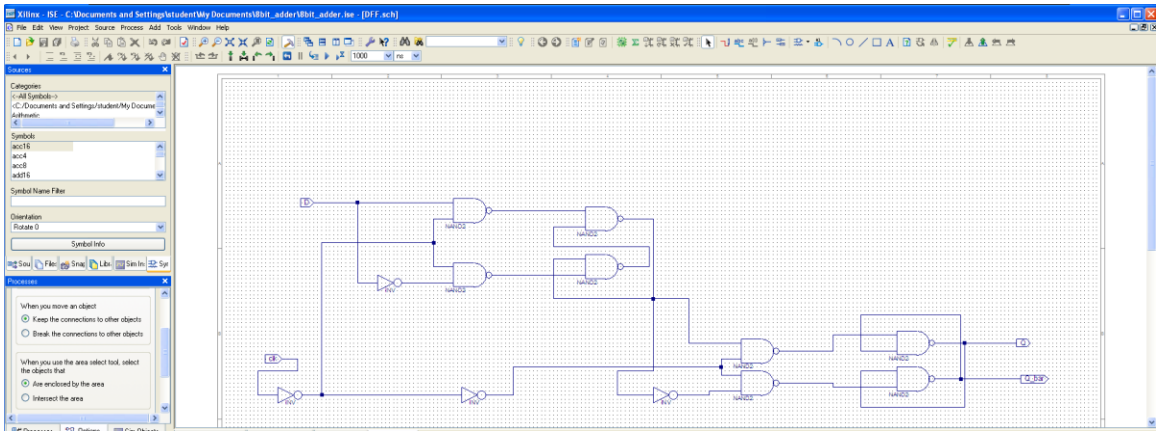
```
// Verilog Test Fixture Template

`timescale 1 ns / 1 ps

module OneBitFullAdder_tb;
  // Inputs
  reg A, B, C;
  // Outputs
  wire S, CO;

  OneBitFullAdder uut (.A(A), .B(B), .C(C), .S(S), .CO(CO));

  initial begin
    A = 0; B = 0; C = 0;
    #100 A = 1; B = 0; C = 0;
    #100 A = 0; B = 1; C = 0;
    #100 A = 1; B = 1; C = 0;
    #100 A = 0; B = 0; C = 1;
    #100 A = 1; B = 0; C = 1;
    #100 A = 0; B = 1; C = 1;
    #100 A = 1; B = 1; C = 1;
    #100 $finish;
  end
endmodule
```

And here is the simulation waveform of the 1-bit adder:

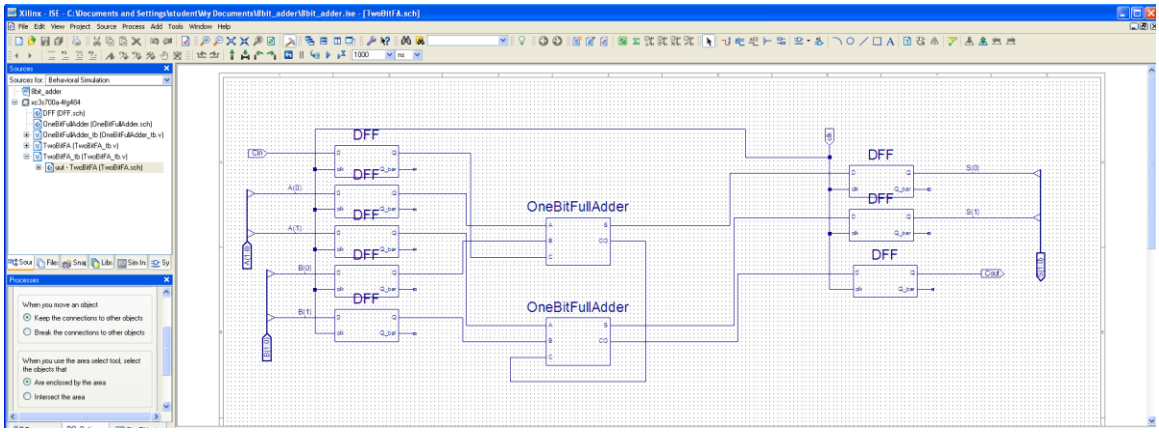



B: 2-bit Adder:

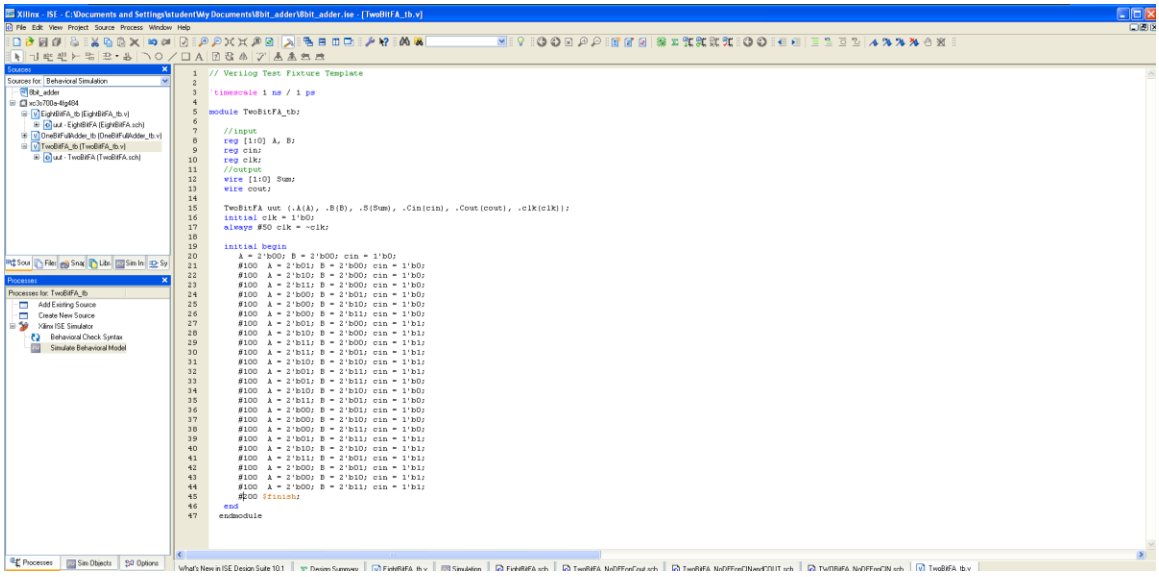
I decided to add the DFF that I designed:



At the phase of designing and testing the 2-bit adder, so that I don't have to deal with the hassle of having to find place on the schematic for it to fit more than 20 DFF:

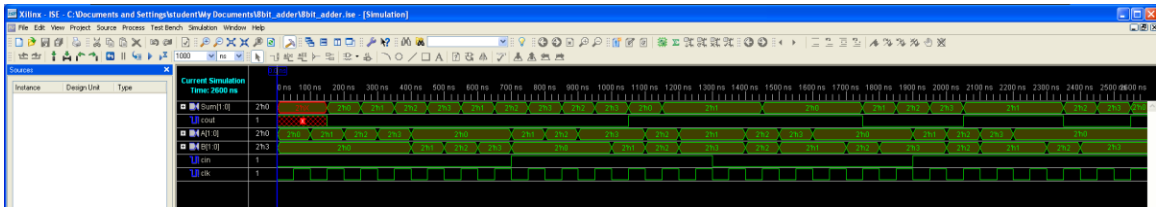



Here is the testbench I wrote to test the design:



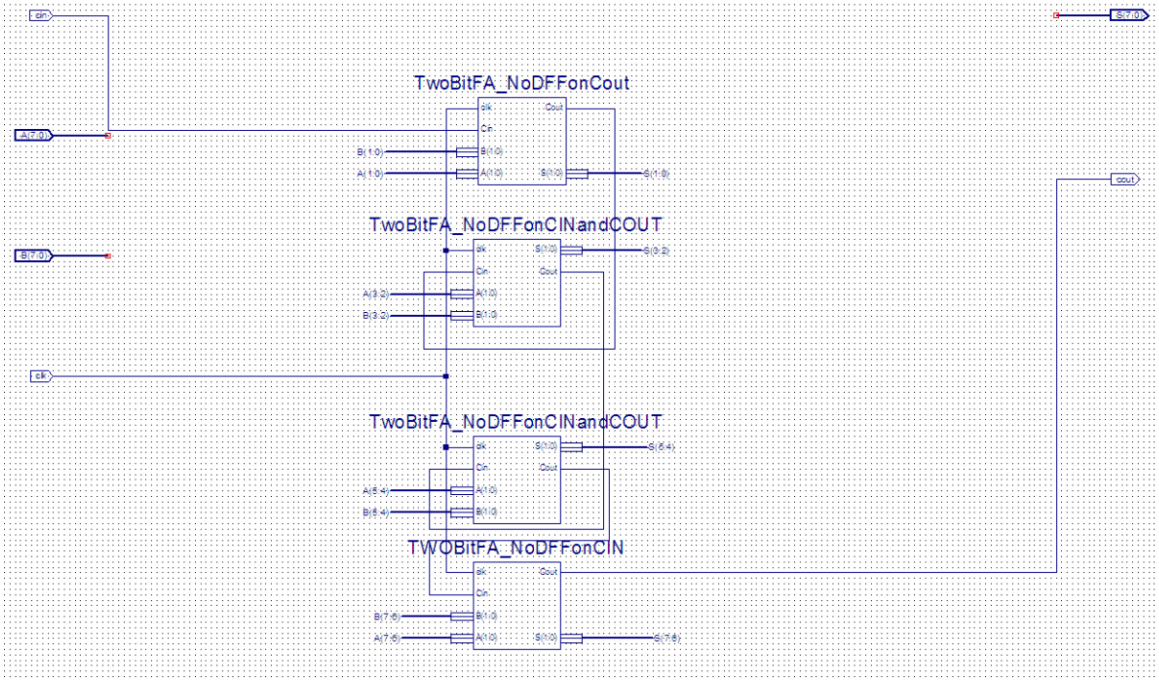
```
// Verilog Test Fixture Template

`timescale 1 ns / 1 ps

module TwoBitFA_tb;

   //input
   reg [1:0] A, B;
   reg cin;
   reg clk;
   //output
   wire [1:0] Sum;
   wire cout;

   TwoBitFA uut (.A(A), .B(B), .S(Sum), .Cin(cin), .Cout(cout), .clk(clk));
   initial clk = 1'b0;
   always #50 clk = ~clk;

   initial begin
      A = 2'b00; B = 2'b00; cin = 1'b0;
      #100  A = 2'b01; B = 2'b00; cin = 1'b0;
      #100  A = 2'b10; B = 2'b00; cin = 1'b0;
      #100  A = 2'b11; B = 2'b00; cin = 1'b0;
      #100  A = 2'b00; B = 2'b01; cin = 1'b0;
      #100  A = 2'b00; B = 2'b10; cin = 1'b0;
      #100  A = 2'b00; B = 2'b11; cin = 1'b0;
      #100  A = 2'b01; B = 2'b00; cin = 1'b1;
      #100  A = 2'b10; B = 2'b00; cin = 1'b1;
      #100  A = 2'b11; B = 2'b00; cin = 1'b1;
      #100  A = 2'b01; B = 2'b01; cin = 1'b1;
      #100  A = 2'b10; B = 2'b10; cin = 1'b1;
      #100  A = 2'b11; B = 2'b11; cin = 1'b1;
      #100  A = 2'b01; B = 2'b11; cin = 1'b0;
      #100  A = 2'b10; B = 2'b10; cin = 1'b0;
      #100  A = 2'b11; B = 2'b01; cin = 1'b0;
      #100  A = 2'b00; B = 2'b01; cin = 1'b0;
      #100  A = 2'b00; B = 2'b10; cin = 1'b0;
      #100  A = 2'b00; B = 2'b11; cin = 1'b0;
      #100  A = 2'b01; B = 2'b11; cin = 1'b1;
      #100  A = 2'b10; B = 2'b10; cin = 1'b1;
      #100  A = 2'b11; B = 2'b01; cin = 1'b1;
      #100  A = 2'b00; B = 2'b01; cin = 1'b1;
      #100  A = 2'b00; B = 2'b10; cin = 1'b1;
      #100  A = 2'b00; B = 2'b11; cin = 1'b1;
      #100 $finish;
   end
endmodule
```

and here is the waveform:

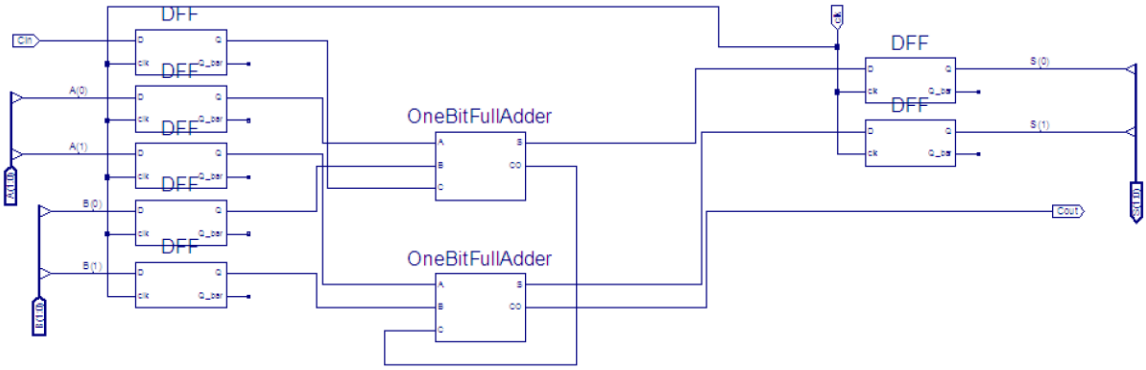


C: 8 bit adder:

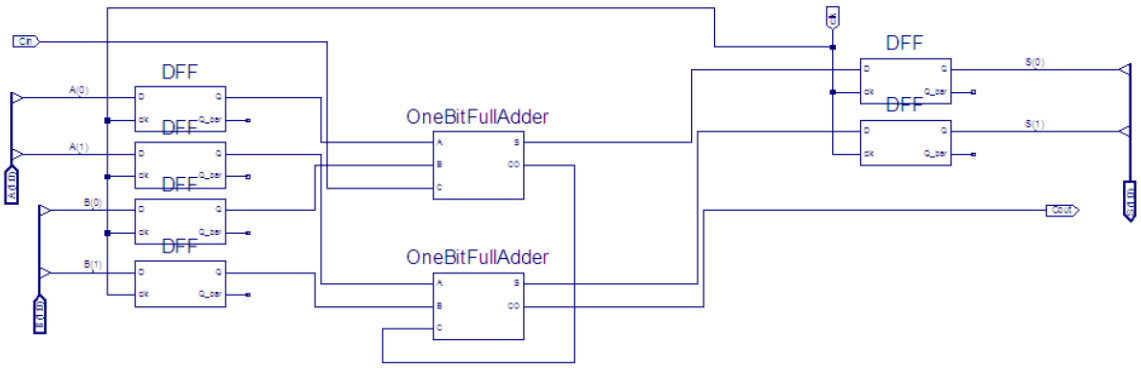
Here is my schematic of the 8 bit adder:



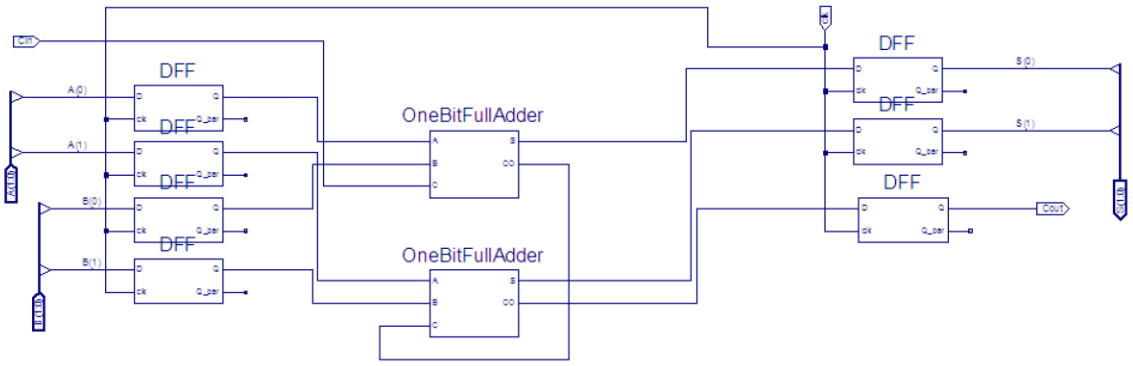

The Top 2bit adder doesn't have DFF connected to the Cout output:



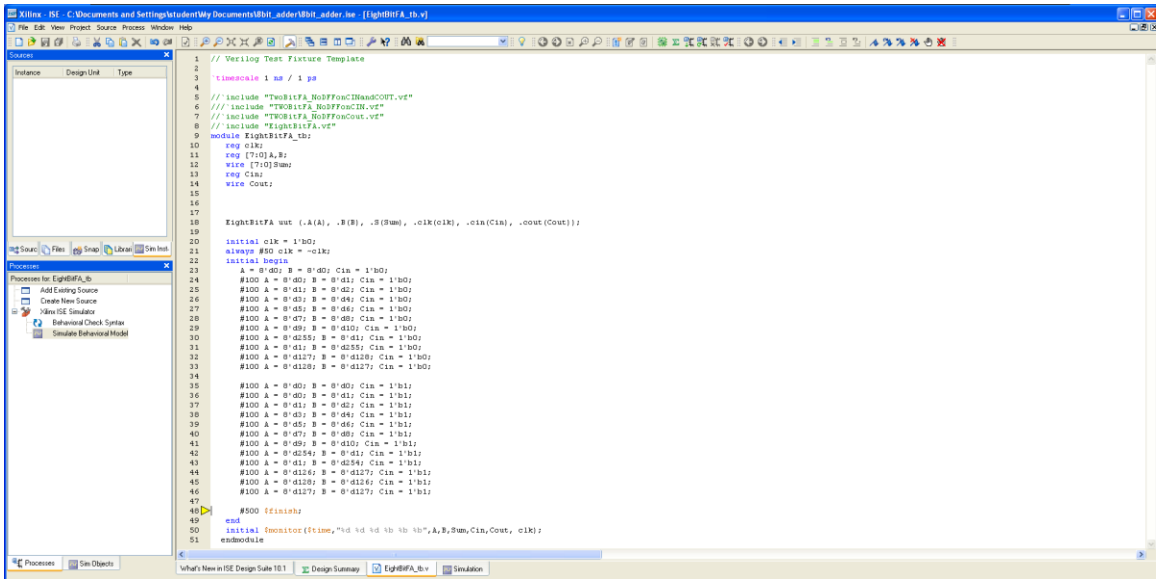

The middle 2 2bit adder doesn't have DFF implemented at all for carries:




The last 2bit adder doesn't have dff implemented at cin:

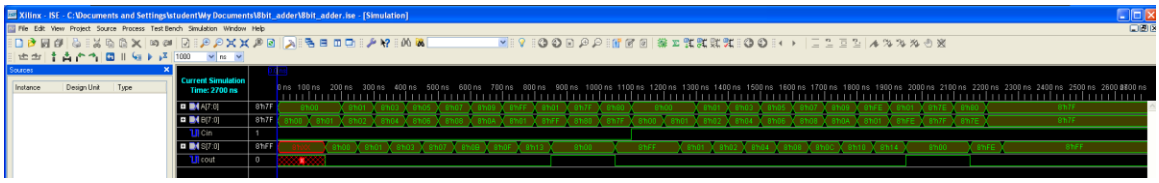



Here is the testbench I wrote for the adder:



```
// Verilog Test Fixture Template

`timescale 1 ns / 1 ps

//`include "TwoBitFA_NoDFFonCINandCOUT.vf"
//`include "TWOBitFA_NoDFFonCIN.vf"
//`include "TWOBitFA_NoDFFonCout.vf"
//`include "EightBitFA.vf"
module EightBitFA_tb;
    reg clk;
    reg [7:0]A,B;
    wire [7:0]Sum;
    reg Cin;
    wire Cout;


    EightBitFA uut (.A(A), .B(B), .S(Sum), .clk(clk), .cin(Cin), .cout(Cout));

    initial clk = 1'b0;
    always #50 clk = ~clk;
    initial begin
        A = 8'd0; B = 8'd0; Cin = 1'b0;
        #100 A = 8'd0; B = 8'd1; Cin = 1'b0;
        #100 A = 8'd1; B = 8'd2; Cin = 1'b0;
        #100 A = 8'd3; B = 8'd4; Cin = 1'b0;
        #100 A = 8'd5; B = 8'd6; Cin = 1'b0;
        #100 A = 8'd7; B = 8'd8; Cin = 1'b0;
        #100 A = 8'd9; B = 8'd10; Cin = 1'b0;
        #100 A = 8'd255; B = 8'd1; Cin = 1'b0;
        #100 A = 8'd1; B = 8'd255; Cin = 1'b0;
        #100 A = 8'd127; B = 8'd128; Cin = 1'b0;
        #100 A = 8'd128; B = 8'd127; Cin = 1'b0;

        #100 A = 8'd0; B = 8'd0; Cin = 1'b1;
        #100 A = 8'd0; B = 8'd1; Cin = 1'b1;
        #100 A = 8'd1; B = 8'd2; Cin = 1'b1;
        #100 A = 8'd3; B = 8'd4; Cin = 1'b1;
        #100 A = 8'd5; B = 8'd6; Cin = 1'b1;
        #100 A = 8'd7; B = 8'd8; Cin = 1'b1;
        #100 A = 8'd9; B = 8'd10; Cin = 1'b1;
        #100 A = 8'd254; B = 8'd1; Cin = 1'b1;
        #100 A = 8'd1; B = 8'd254; Cin = 1'b1;
        #100 A = 8'd126; B = 8'd127; Cin = 1'b1;
        #100 A = 8'd128; B = 8'd126; Cin = 1'b1;
        #100 A = 8'd127; B = 8'd127; Cin = 1'b1;

        #500 $finish;
    end
    initial $monitor($time,"%d %d %d %b %b %b",A,B,Sum,Cin,Cout, clk);
endmodule
```

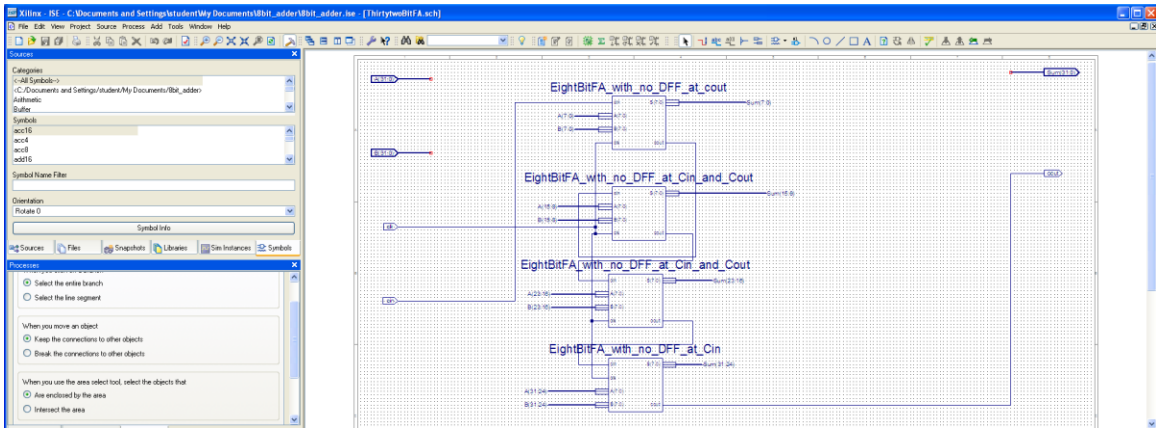
Here is the simulation waveform of the 8 bit adder:



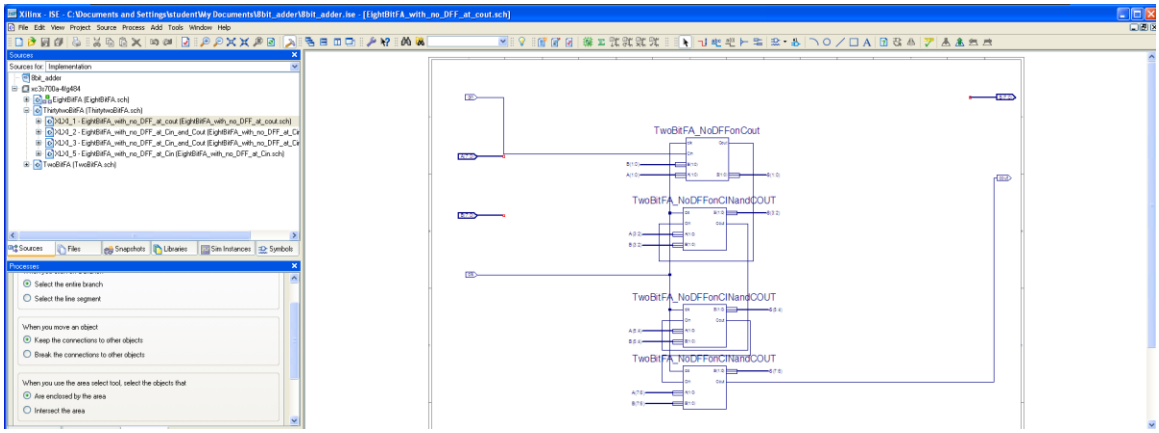
Part 3: ALU

A: 32bit Full Adder

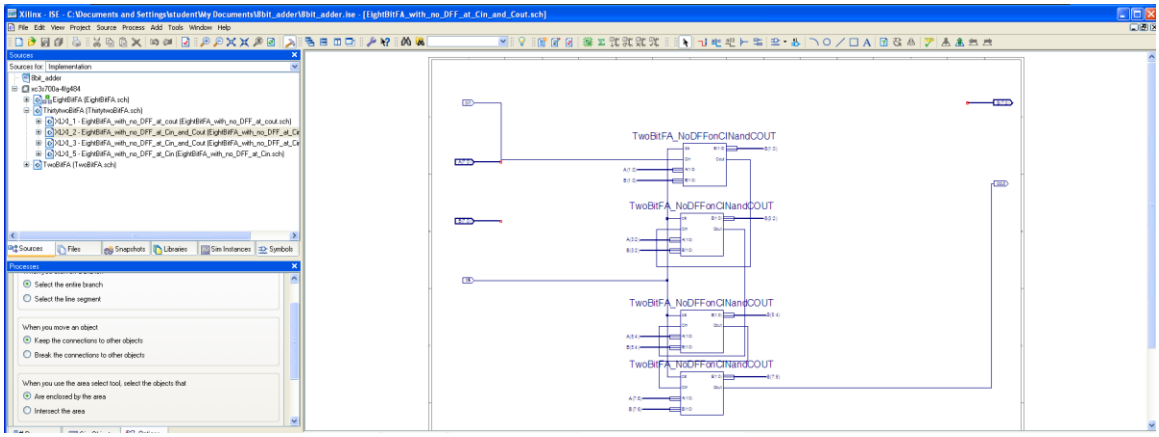
Here is my schematic for the 32bit full adder:



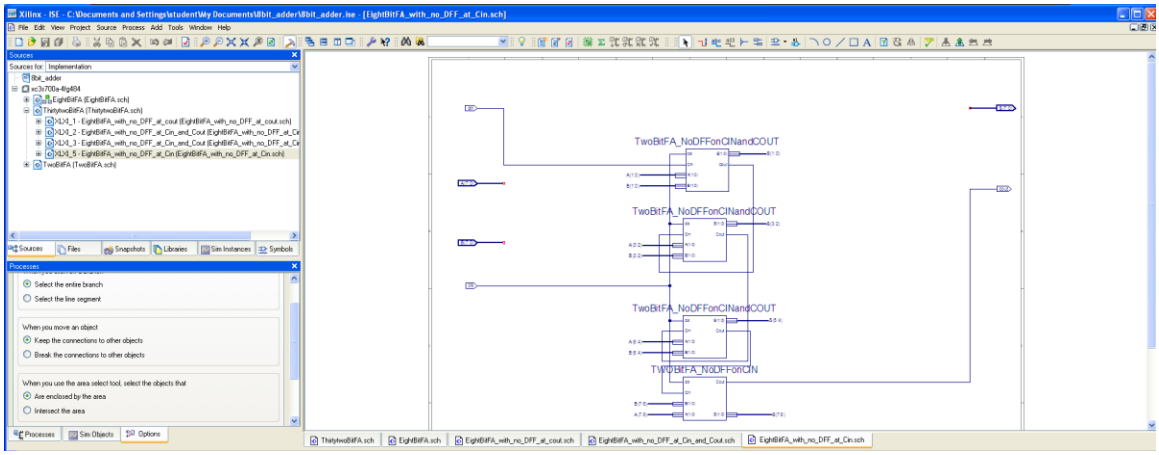

Its created from the modified version of the 8 bit adder, that the first adder doesn't have DFF at Cout:



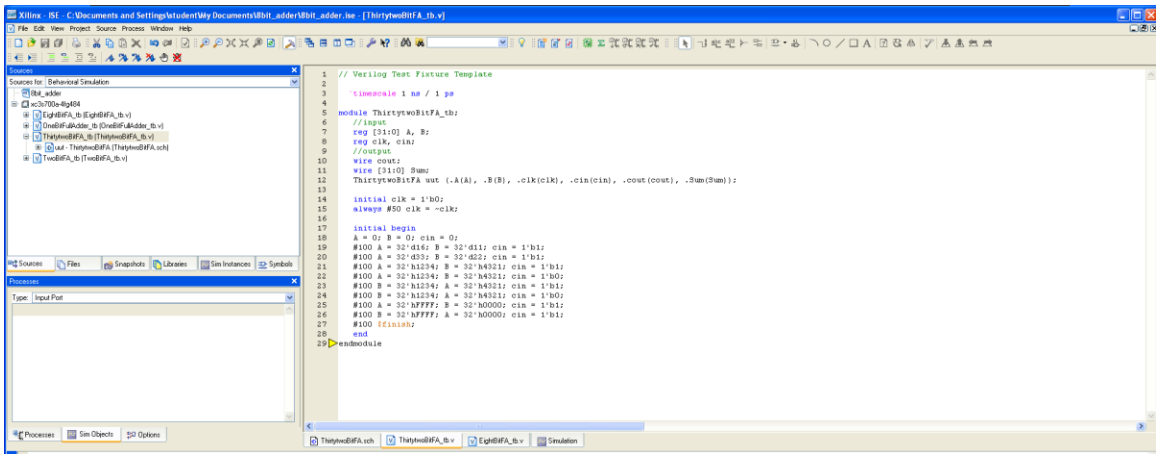
The second and third one doesn’t have DFF at all on both Cin and Cout:



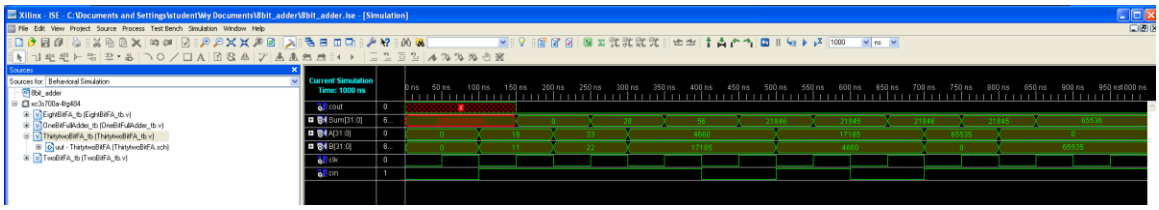

The last one doesn't have DFF on Cin:



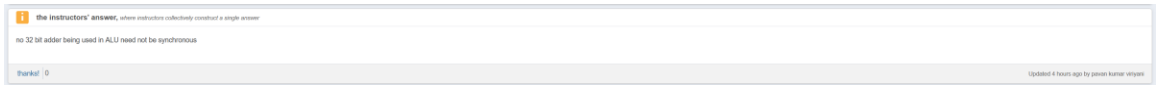

Here is the testbench I designed for the 32 bit adder:



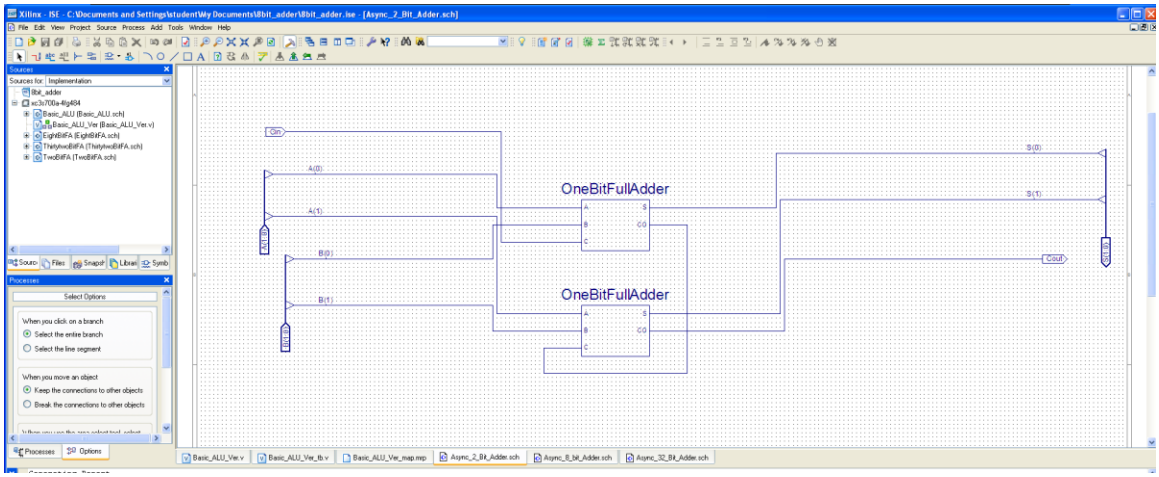
And here is the resulting waveforms:



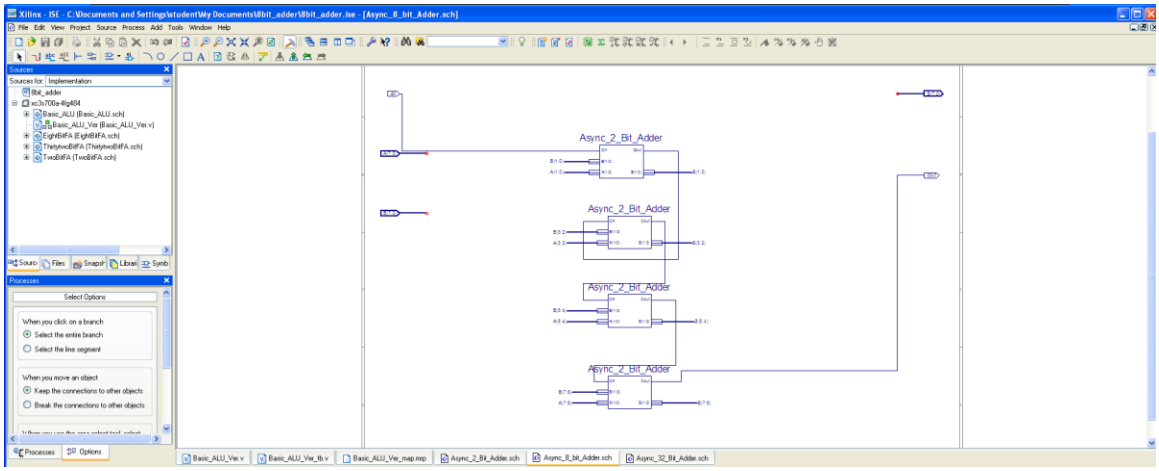
B: I made some modifications by removing all the DFF in the 32bit adder after asking the TA about it and TA confirms that the ALU is asynchronous:



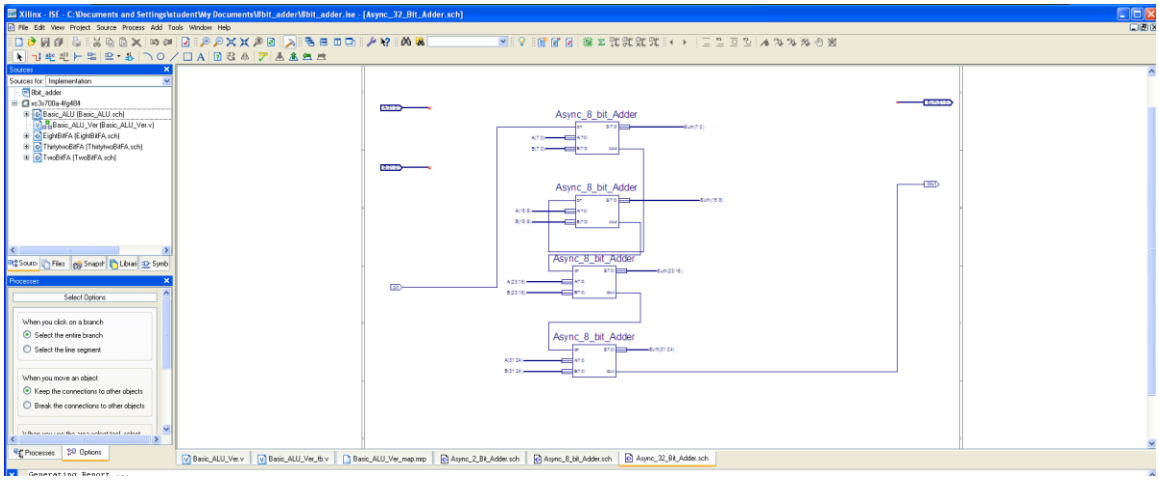

Here is the schematic of the asynchronous 2 bit adder:




Here is the schematic of the asynchronous 8 bit adder:



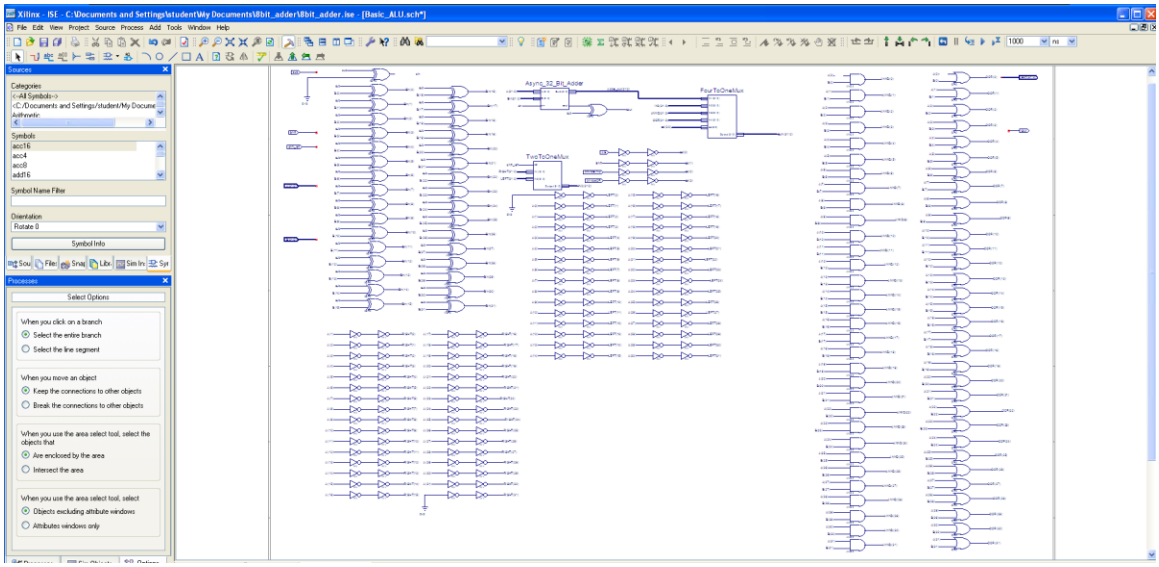

And the asynchronous 32 bit adder:



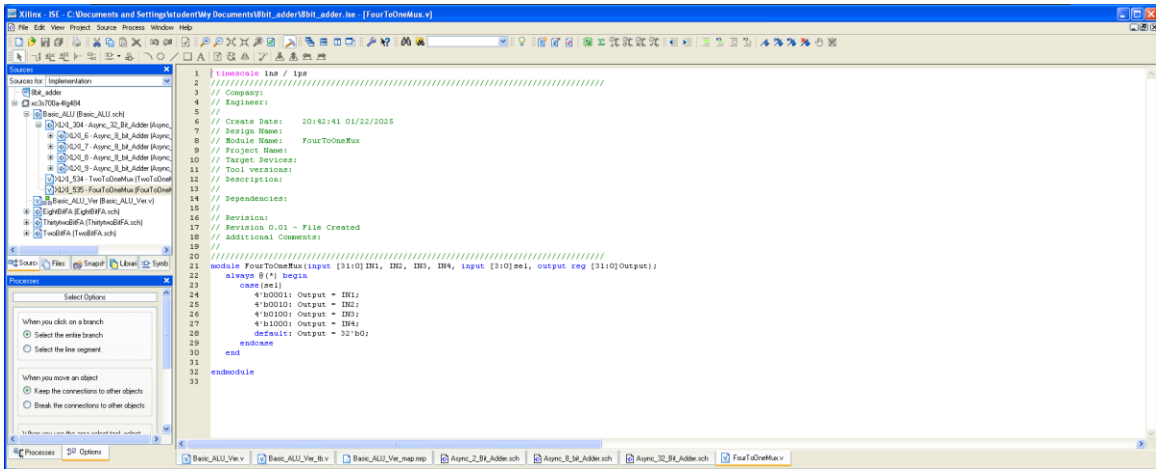

The 2 features that I chose to implement is bitwise AND & bitwise OR feature; so the feature of my ALU are:

1. Add
2. Subtract
3. Do logical shift left/right
4. Do bitwise AND
5. Do bitwise OR

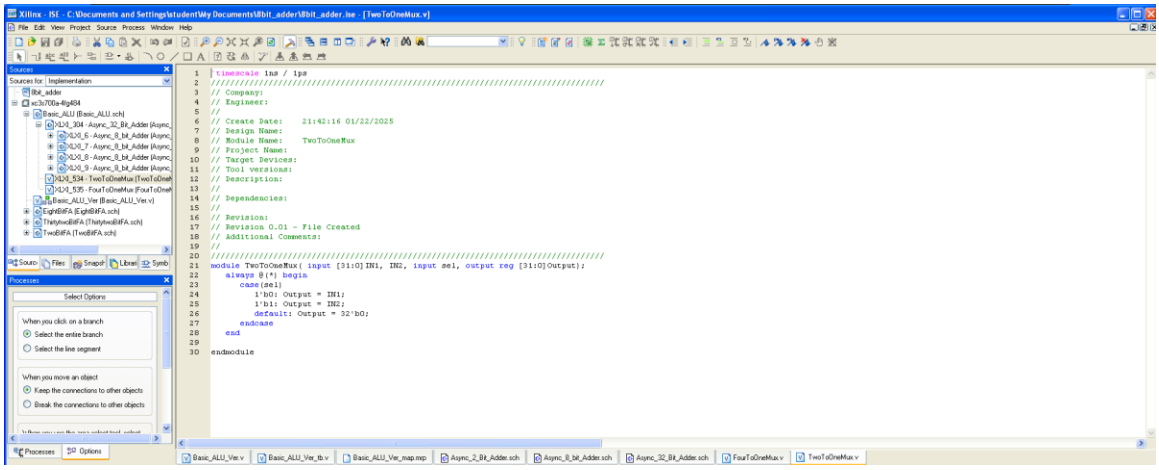
Here is the schematic of my Basic ALU:



I implemented the Mux using Verilog because it would be easier to implement using just one case statement:

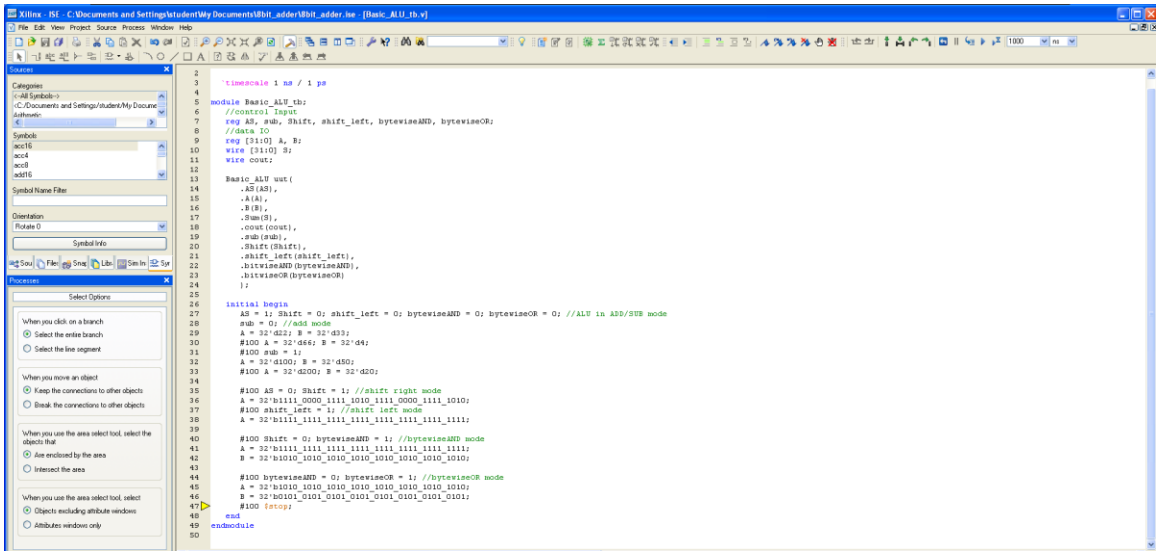


```
timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    20:42:41 01/22/2025
// Design Name:
// Module Name:    FourToOneMux
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module FourToOneMux(input [31:0] IN1, IN2, IN3, IN4, input [3:0]sel, output reg [31:0]Output);
  always @(*) begin
     case(sel)
        4'b0001: Output = IN1;
        4'b0010: Output = IN2;
        4'b0100: Output = IN3;
        4'b1000: Output = IN4;
        default: Output = 32'b0;
     endcase
  end

endmodule
```



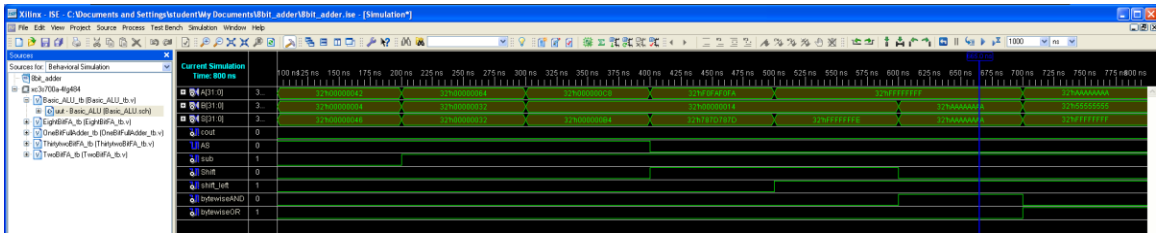
```
timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    21:42:16 01/22/2025
// Design Name:
// Module Name:    TwoToOneMux
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module TwoToOneMux( input [31:0]IN1, IN2, input sel, output reg [31:0]Output);
  always @(*) begin
     case(sel)
        1'b0: Output = IN1;
        1'b1: Output = IN2;
        default: Output = 32'b0;
     endcase
  end

endmodule
```

And here is the testbench that I wrote:

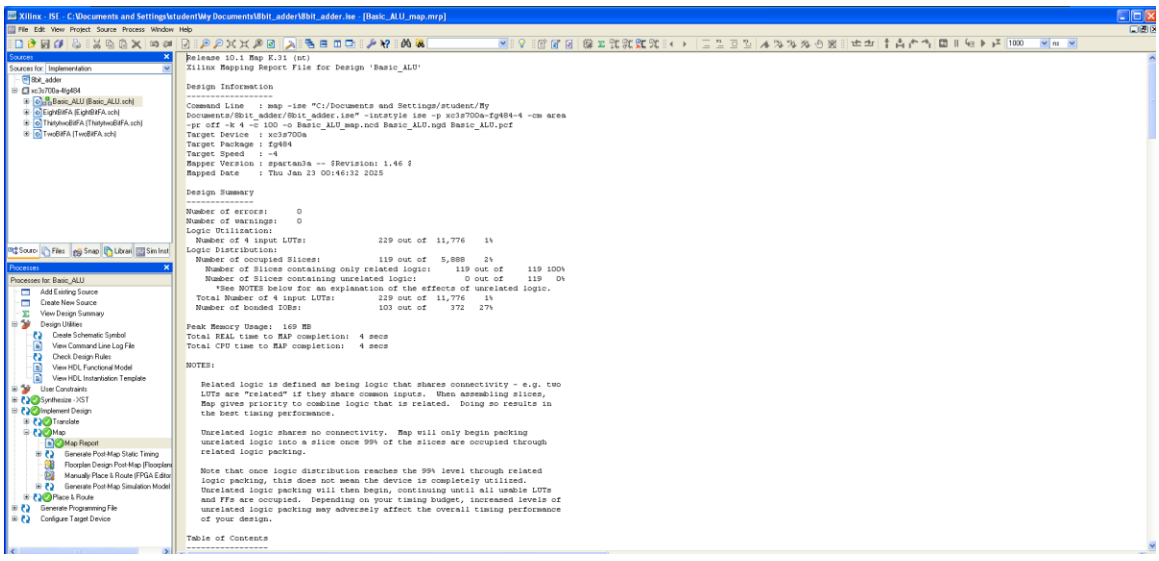


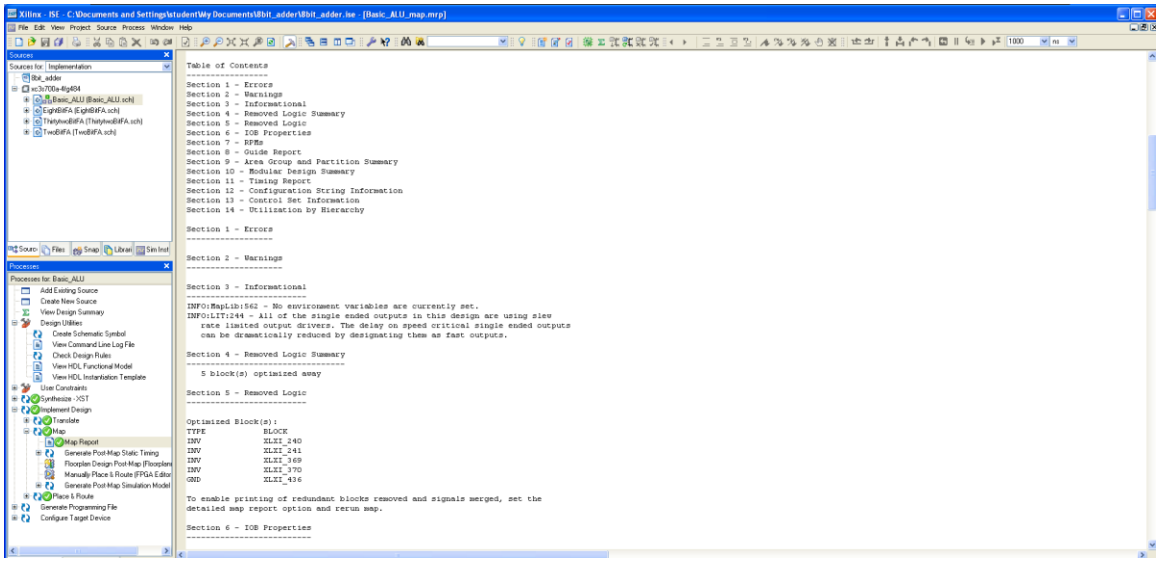
```
`timescale 1 ns / 1 ps

module Basic_ALU_tb;
  //control Input
  reg AS, sub, Shift, shift_left, bytewiseAND, bytewiseOR;
  //data IO
  reg [31:0] A, B;
  wire [31:0] S;
  wire cout;

  Basic_ALU uut(
     .AS(AS),
     .A(A),
     .B(B),
     .Sum(S),
     .cout(cout),
     .sub(sub),
     .Shift(Shift),
     .shift_left(shift_left),
     .bitwiseAND(bytewiseAND),
     .bitwiseOR(bytewiseOR)
     );

  initial begin
     AS = 1; Shift = 0; shift_left = 0; bytewiseAND = 0; bytewiseOR = 0; //ALU in ADD/SUB mode
     sub = 0; //add mode
     A = 32'd22; B = 32'd33;
     #100 A = 32'd66; B = 32'd4;
     #100 sub = 1;
     A = 32'd100; B = 32'd50;
     #100 A = 32'd200; B = 32'd20;

     #100 AS = 0; Shift = 1; //shift right mode
     A = 32'b1111_0000_1111_1010_1111_0000_1111_1010;
     #100 shift_left = 1; //shift left mode
     A = 32'b1111_1111_1111_1111_1111_1111_1111_1111;

     #100 Shift = 0; bytewiseAND = 1; //bytewiseAND mode
     A = 32'b1111_1111_1111_1111_1111_1111_1111_1111;
     B = 32'b1010_1010_1010_1010_1010_1010_1010_1010;

     #100 bytewiseAND = 0; bytewiseOR = 1; //bytewiseOR mode
     A = 32'b1010_1010_1010_1010_1010_1010_1010_1010;
     B = 32'b0101_0101_0101_0101_0101_0101_0101_0101;
     #100 $stop;
  end
endmodule
```

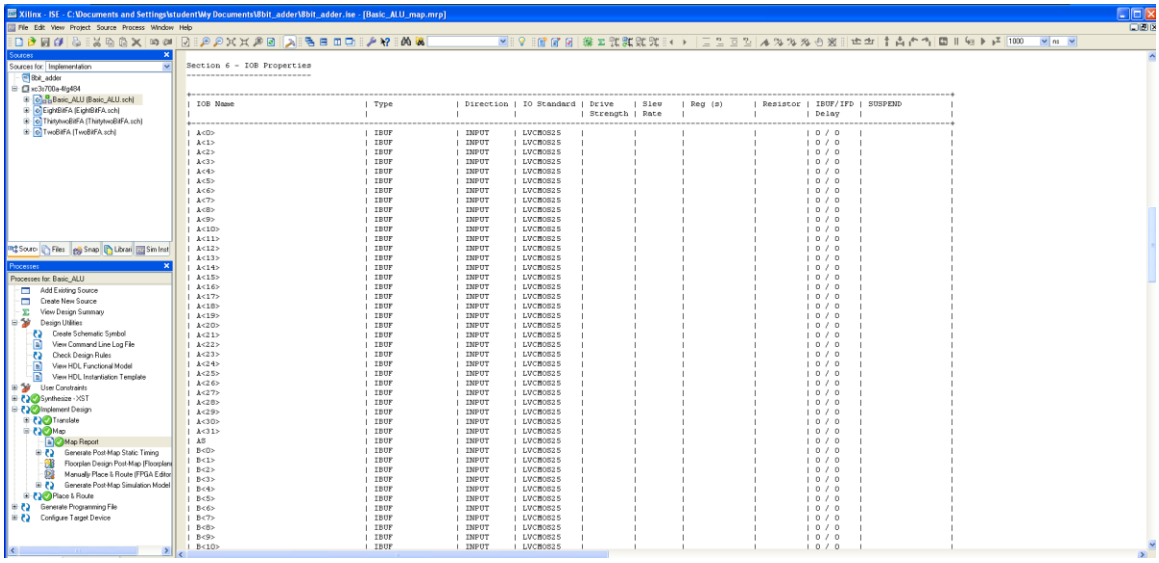
Here is the simulation waveform:

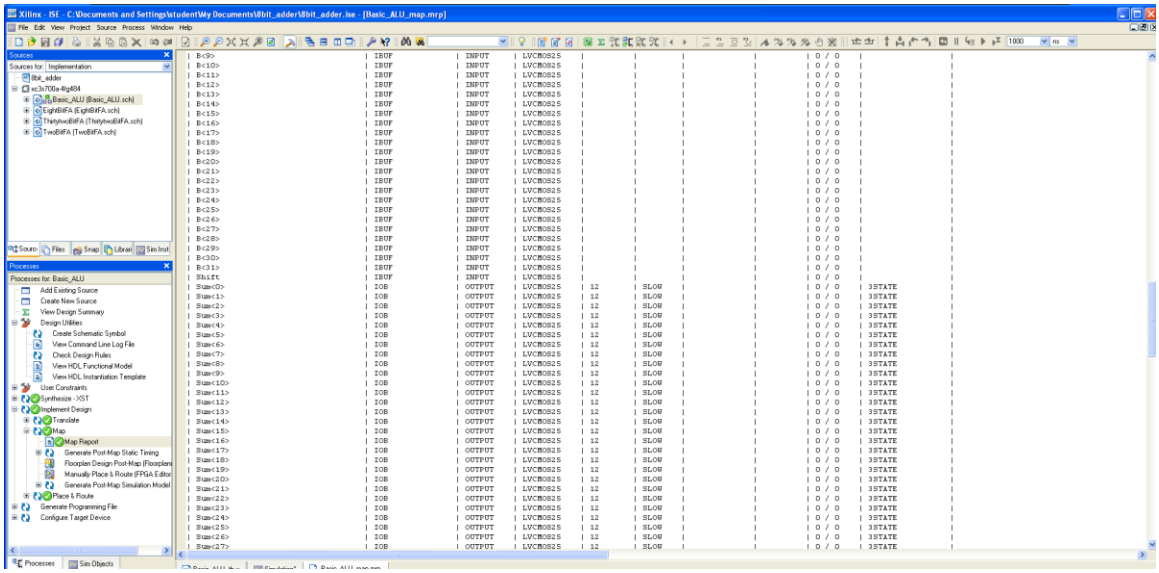


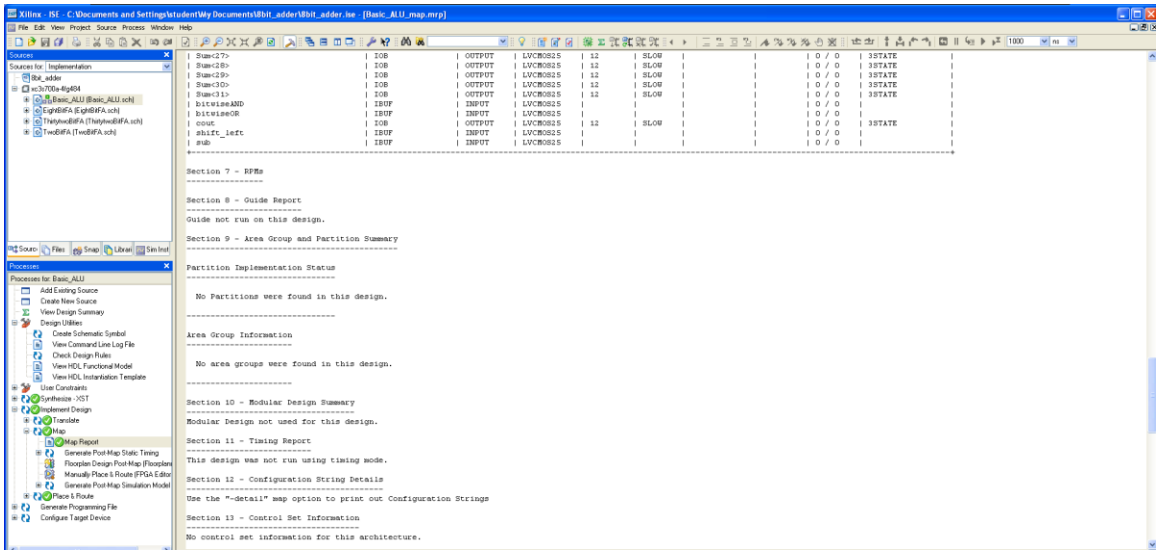
Here is the screenshot of the map report file:

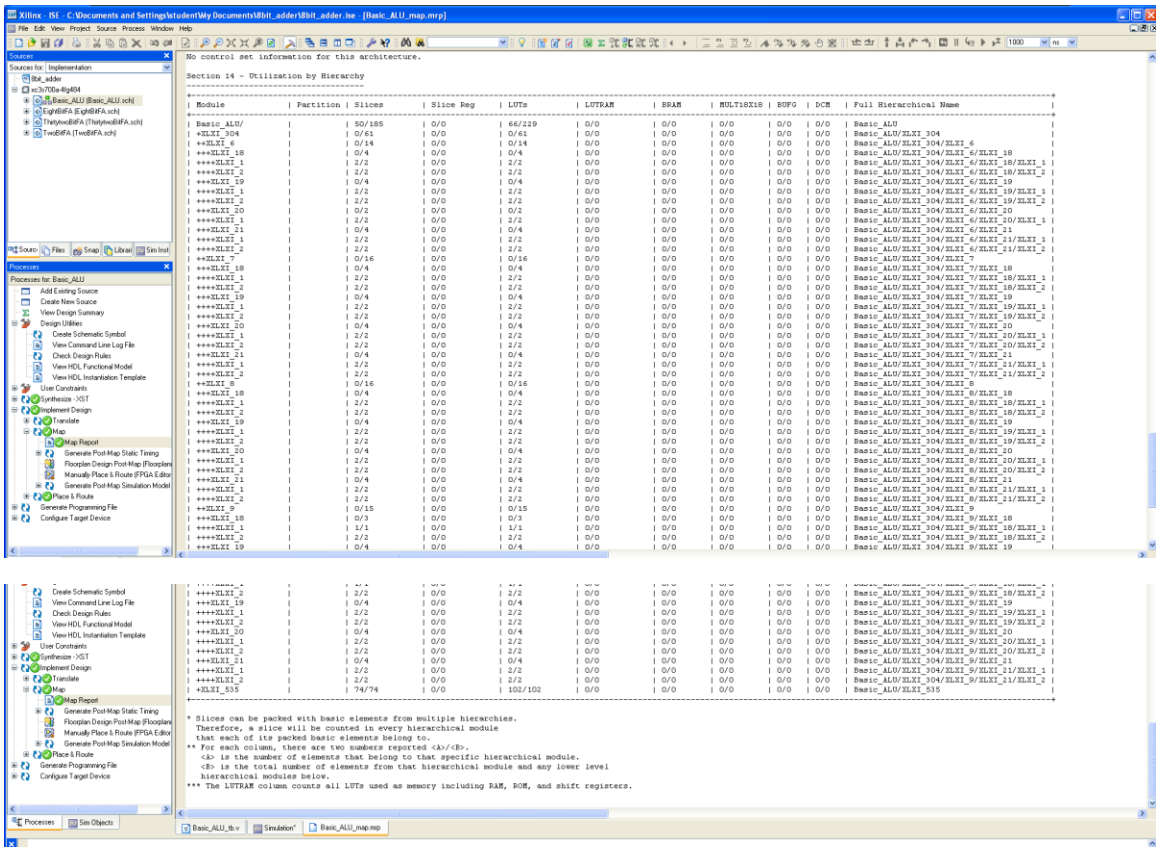




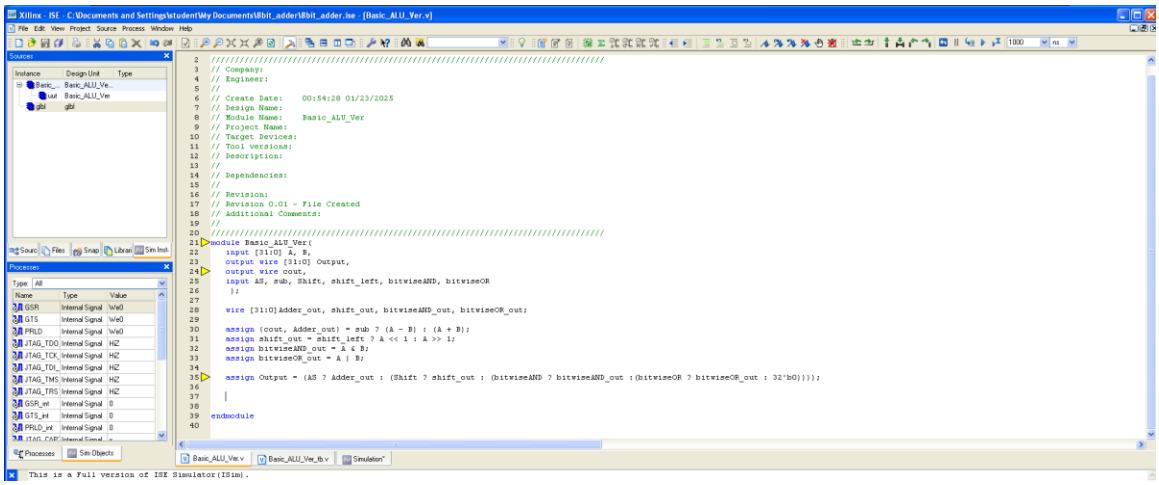

Xilinx - ISE - C:\Documents and Settings\student\My Documents\8bit_adder\8bit_adder.ise - [Basic_ALU_map.mrp]
File Edit View Project Source Process Window Help
Sources
Processes
Section 6 - IOB Properties
| IOB Name | Type | Direction | IO Standard | Drive Strength | Slew Rate | Reg (s) | Resistor | IBUF/IFD Delay | SUSPEND |
| A<0> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<1> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<2> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<3> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<4> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<5> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<6> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<7> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<8> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<9> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<10> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<11> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<12> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<13> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<14> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<15> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<16> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<17> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<18> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<19> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<20> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<21> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<22> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<23> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<24> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<25> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<26> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<27> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<28> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<29> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<30> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| A<31> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| AS | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<0> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<1> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<2> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<3> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<4> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<5> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<6> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<7> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<8> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<9> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<10> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |


Xilinx - ISE - C:\Documents and Settings\student\My Documents\8bit_adder\8bit_adder.ise - [Basic_ALU_map.mrp]
| B<9> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<10> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<11> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<12> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<13> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<14> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<15> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<16> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<17> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<18> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<19> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<20> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<21> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<22> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<23> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<24> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<25> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<26> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<27> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<28> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<29> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<30> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| B<31> | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| Shift | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| Sum<0> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<1> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<2> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<3> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<4> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<5> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<6> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<7> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<8> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<9> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<10> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<11> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<12> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<13> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<14> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<15> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<16> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<17> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<18> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<19> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<20> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<21> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<22> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<23> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<24> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<25> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<26> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<27> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |

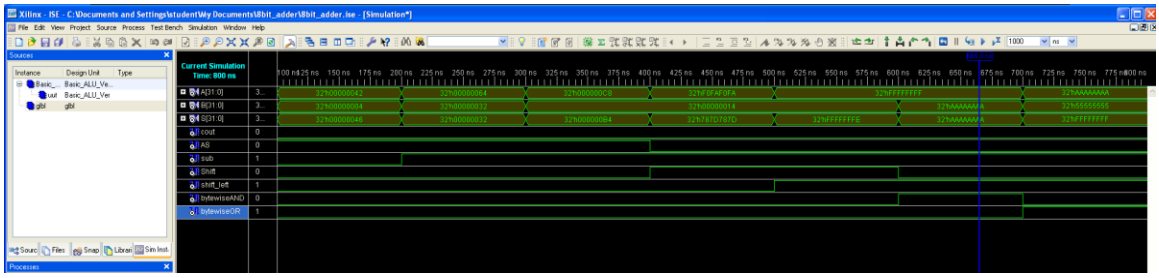

Xilinx - ISE - C:\Documents and Settings\student\My Documents\8bit_adder\8bit_adder.ise - [Basic_ALU_map.mrp]
| Sum<27> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<28> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<29> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<30> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| Sum<31> | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| bitwiseAND | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| bitwiseOR | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| cout | IOB | OUTPUT | LVCMOS25 | 12 | SLOW | | | O / 0 | 3STATE |
| shift_left | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
| sub | IBUF | INPUT | LVCMOS25 | | | | | O / 0 | |
Section 7 - RPMs
Section 8 - Guide Report
Guide not run on this design.
Section 9 - Area Group and Partition Summary
Partition Implementation Status
No Partitions were found in this design.
Area Group Information
No area groups were found in this design.
Section 10 - Modular Design Summary
Modular Design not used for this design.
Section 11 - Timing Report
This design was not run using timing mode.
Section 12 - Configuration String Details
Use the "-detail" map option to print out Configuration Strings
Section 13 - Control Set Information
No control set information for this architecture.



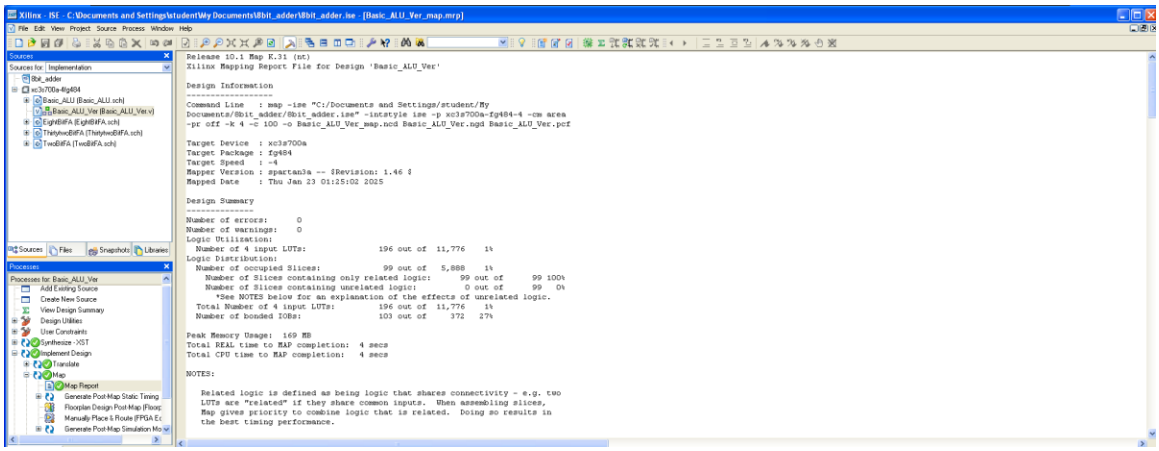
C: I implement the same ALU using Verilog, and here is the screenshot of the code:

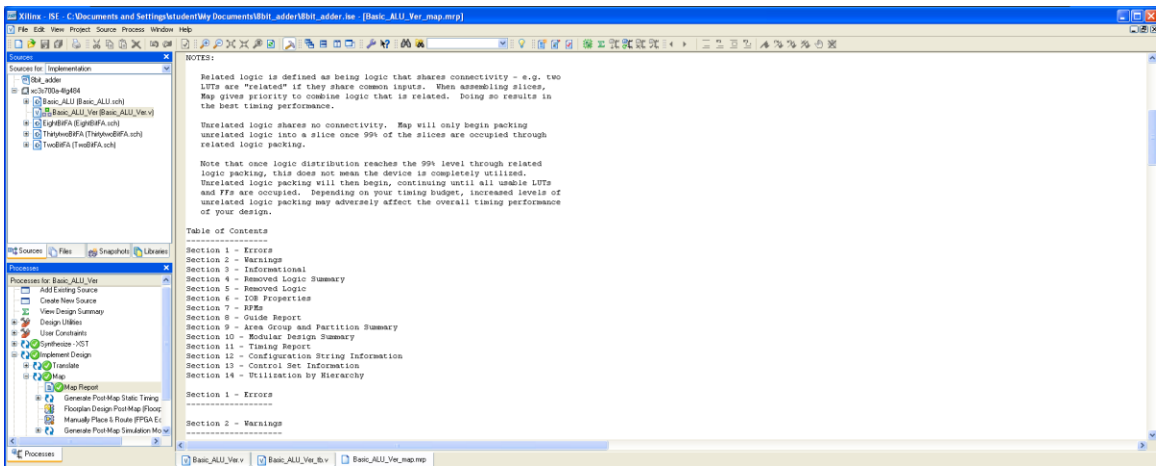


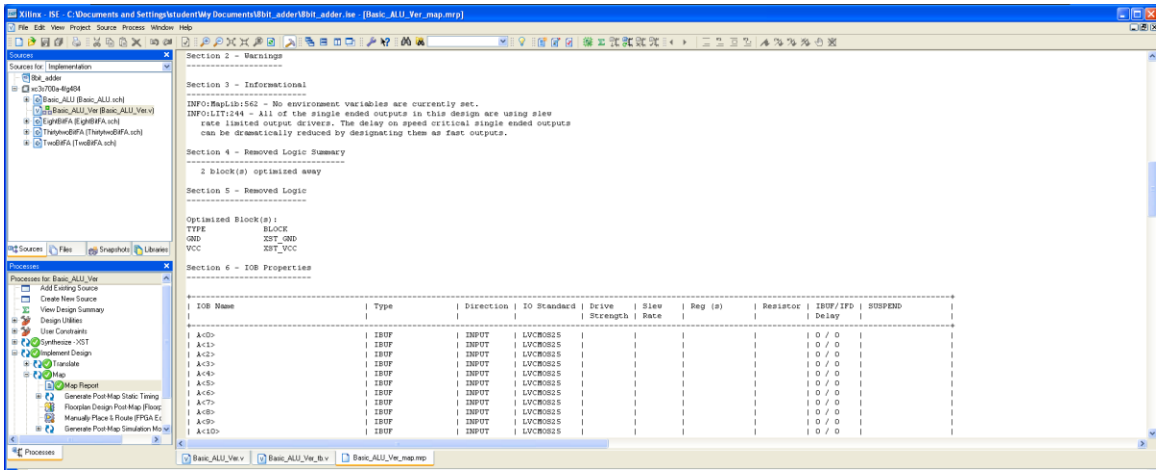
And here is the simulation waveform generated using the same testbench as the testbench in part B:

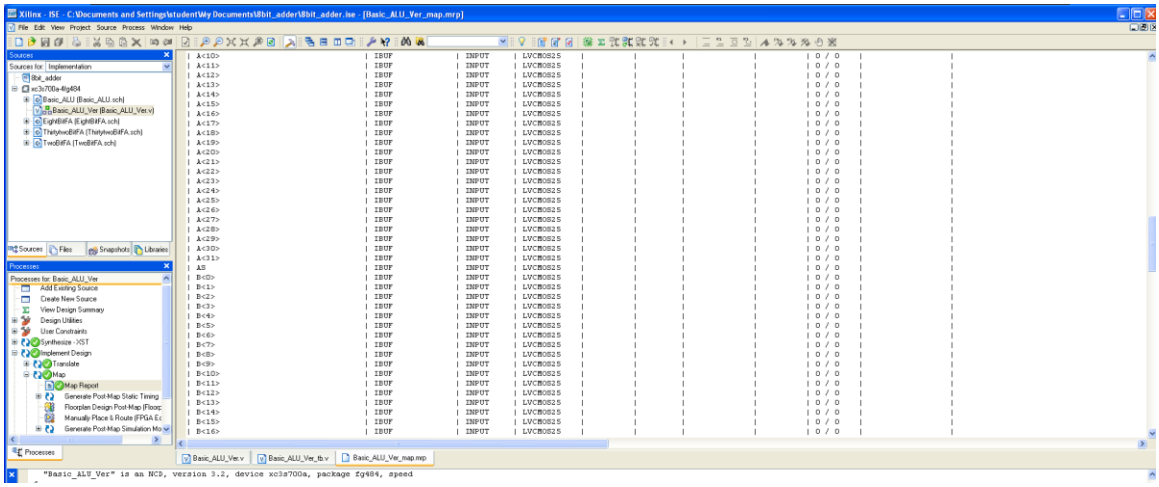


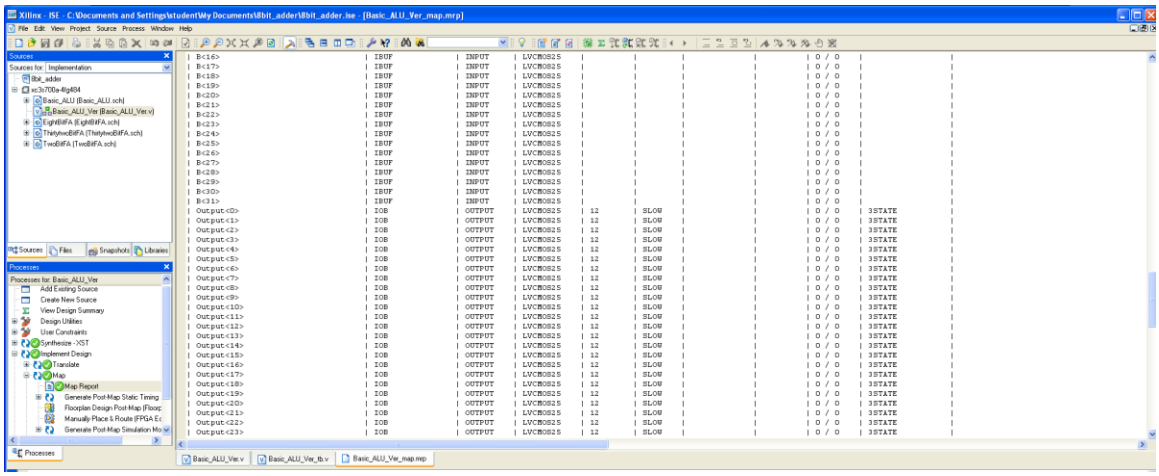
Here is the screenshot of the map report:

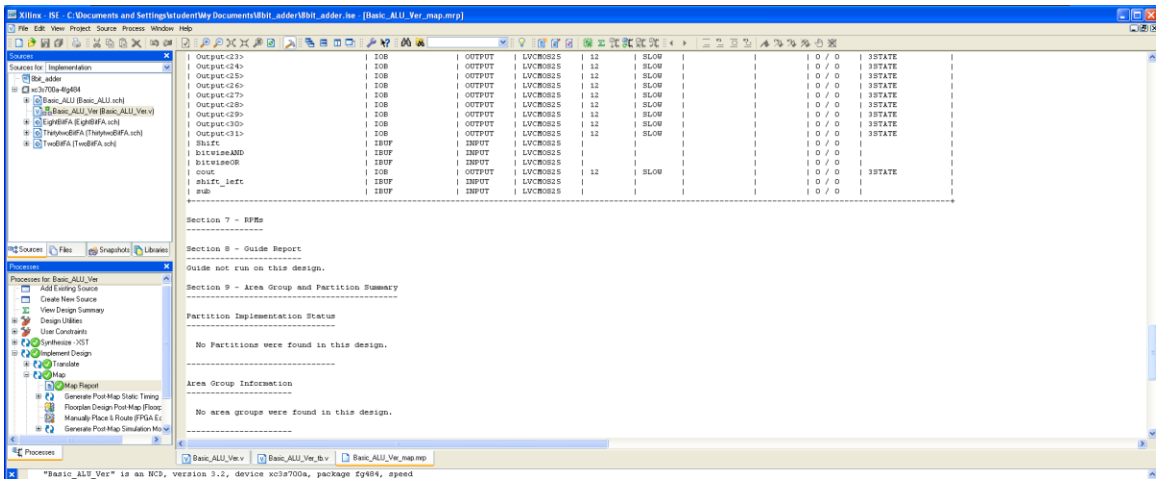


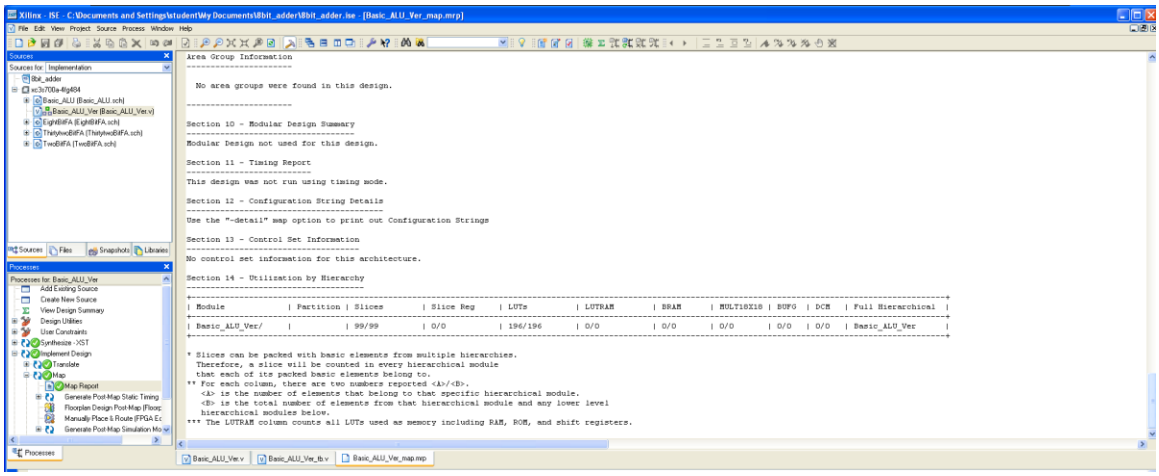




Xilinx - ISE - C:\Documents and Settings\student\My Documents\8bit_adder\8bit_adder.ise - [Basic_ALU_Ver_map.mrp]
File Edit View Project Source Process Window Help
Sources
Processes
Section 2 - Warnings
Section 3 - Informational
INFO:MapLib:562 - No environment variables are currently set.
INFO:LIT:244 - All of the single ended outputs in this design are using slew
rate limited output drivers. The delay on speed critical single ended outputs
can be dramatically reduced by designating them as fast outputs.
Section 4 - Removed Logic Summary
2 block(s) optimized away
Section 5 - Removed Logic
Optimized Block(s):
TYPE BLOCK
GND XST_GND
VCC XST_VCC
Section 6 - IOB Properties
Basic_ALU_Ver.v
Basic_ALU_Ver_tb.v
Basic_ALU_Ver_map.mrp


Xilinx - ISE - C:\Documents and Settings\student\My Documents\8bit_adder\8bit_adder.ise - [Basic_ALU_Ver_map.mrp]
"Basic_ALU_Ver" is an NCD, version 3.2, device xc3s700a, package fg484, speed
Basic_ALU_Ver.v
Basic_ALU_Ver_tb.v
Basic_ALU_Ver_map.mrp


Xilinx - ISE - C:\Documents and Settings\student\My Documents\8bit_adder\8bit_adder.ise - [Basic_ALU_Ver_map.mrp]
Basic_ALU_Ver.v
Basic_ALU_Ver_tb.v
Basic_ALU_Ver_map.mrp





Comparison: We can see that the Verilog version of the ALU would use less slices on FPGA (119 vs. 99); the same number of IOBs (both 103); and less 4 inputs look up tables (229 vs. 196).