

# Lab2 Report

Name: Haoyang Zhang

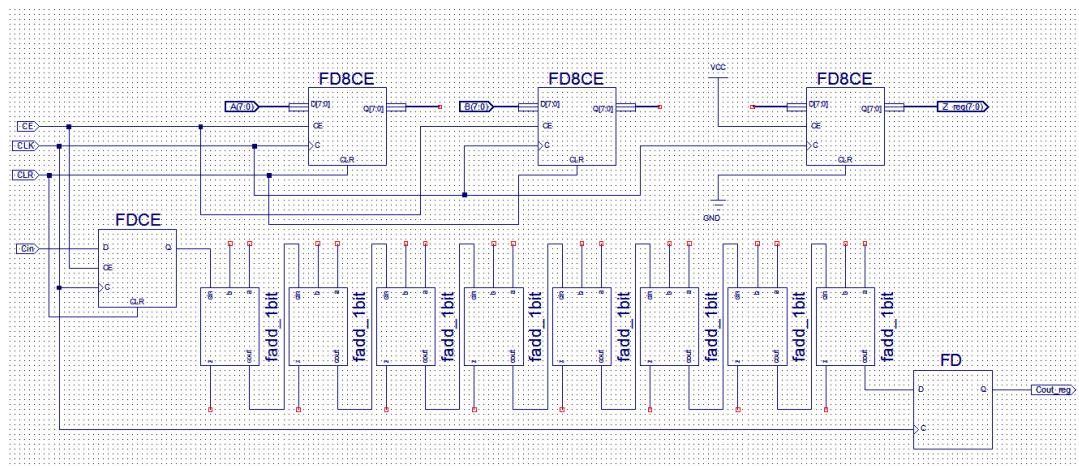
Email: [hzhang11@usc.edu](mailto:hzhang11@usc.edu)

Remote repository address: <https://github.com/hzhang2422/EE-533>

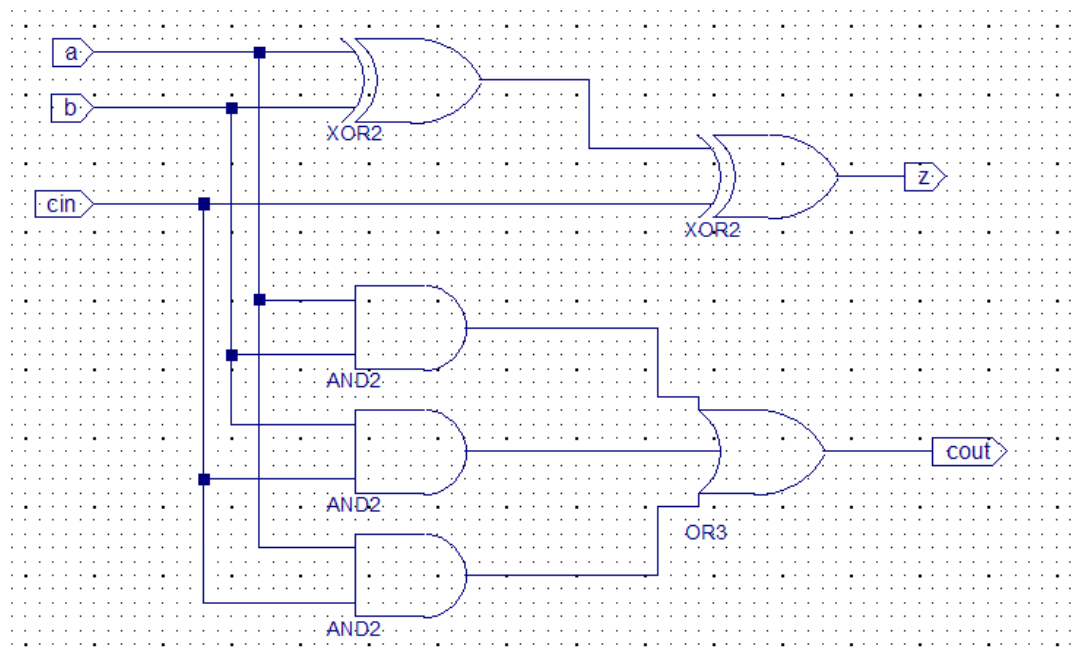
## Part 1: Designing and Simulating Synchronous 8-bit Adder

### 1. Screen capture of schematics

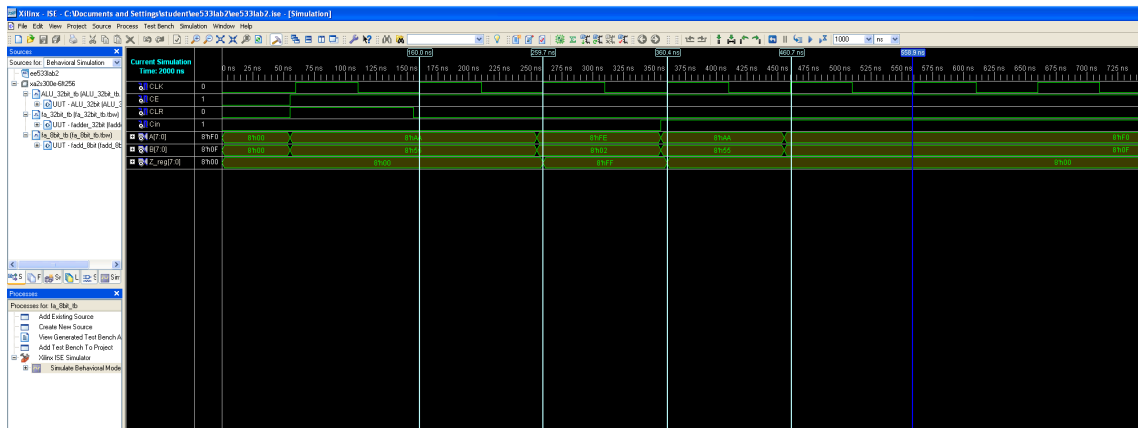
- 8-bit Synchronous Full Adder



- 1-bit Full Adder



### 2. Screen capture of the waveforms generated by the behavioral simulation tools

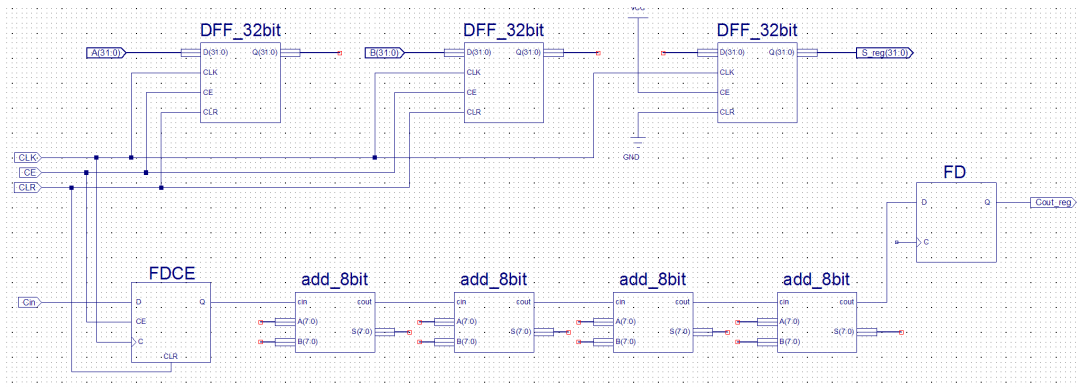


## Part 2: Extending Adder into 32-bit ALU

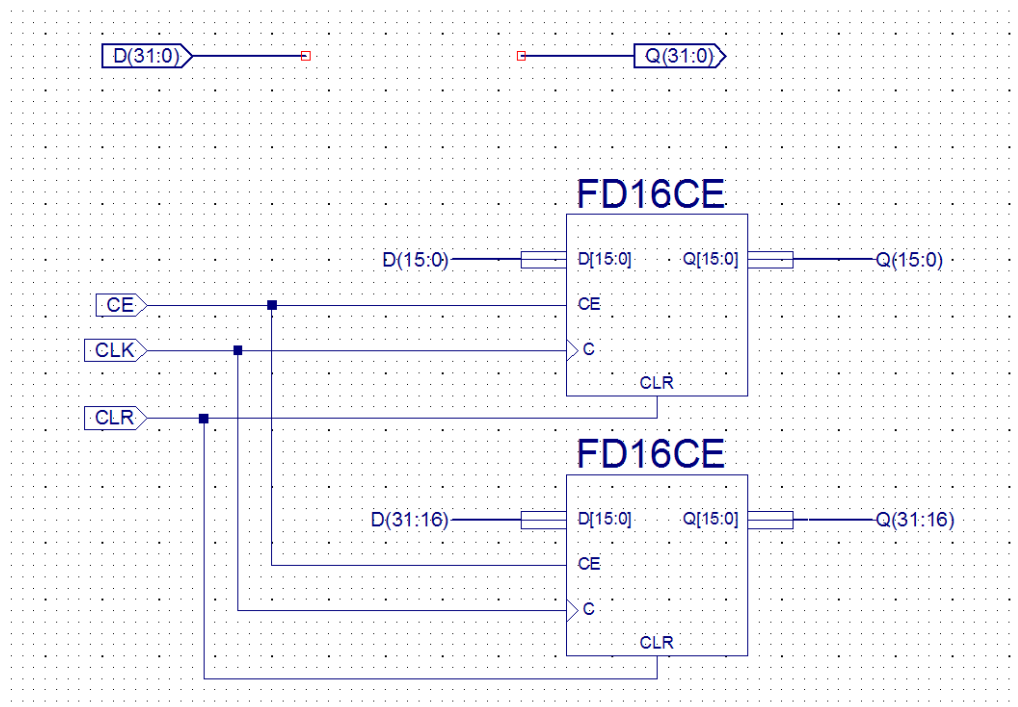
**A : Extend the 8-bit Adder into 32-bit Adder by instantiating and connecting 4 adders**

1. Screen capture of schematics

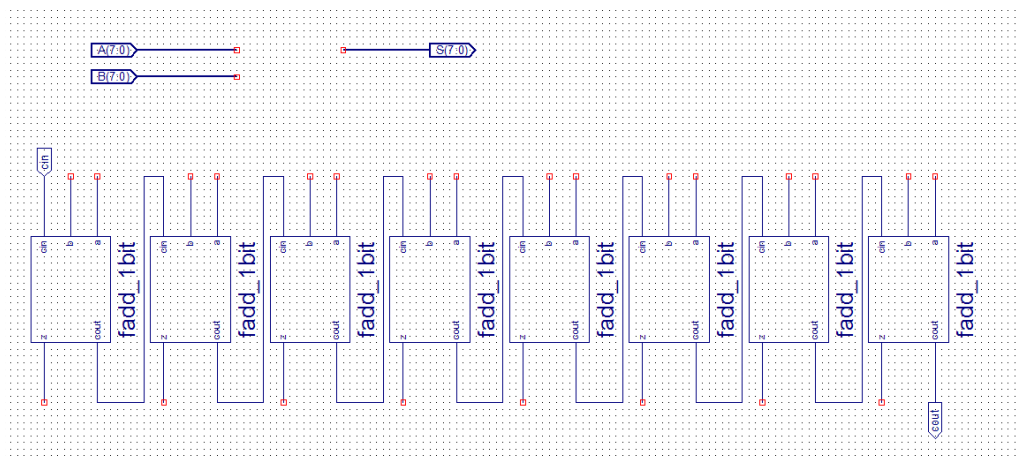
- 32-bit Synchronous Full Adder



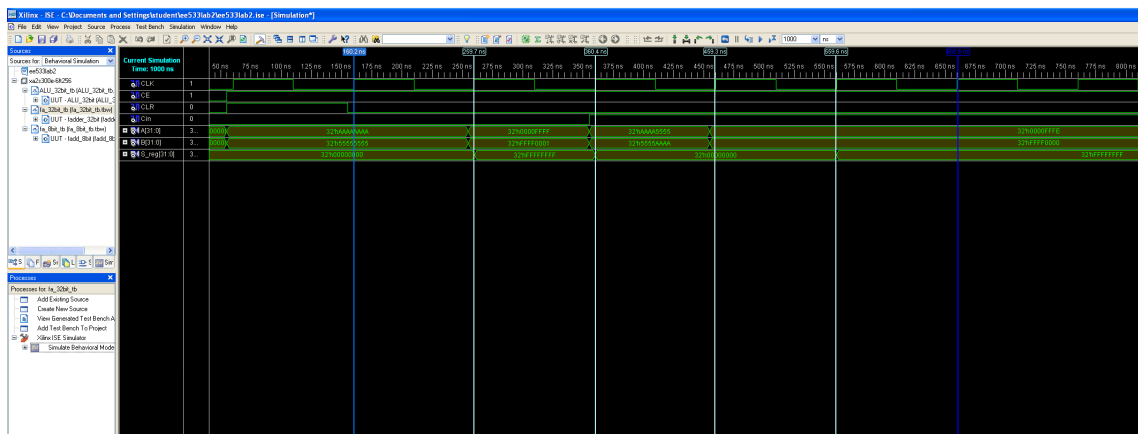
- 32-bit DFF



- 8-bit Full Adder



## 2. Screen capture of the waveforms generated by the behavioral simulation tools



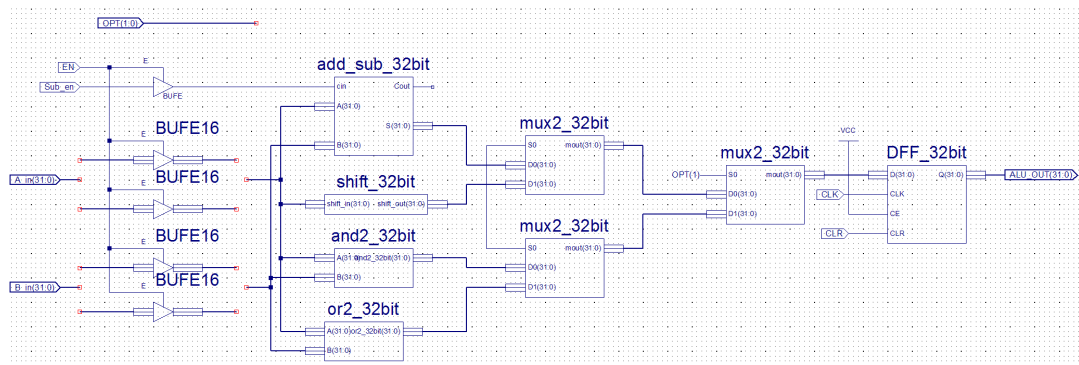
**B : Extend the 32-bit Adder to have other functions including subtractor, shifter, and two other functions(AND & OR) of your choice. Go through the mapping process of the tools to get the gate counts such as number of D-FF and LUTs.**

### 1. Brief description of all of the functions

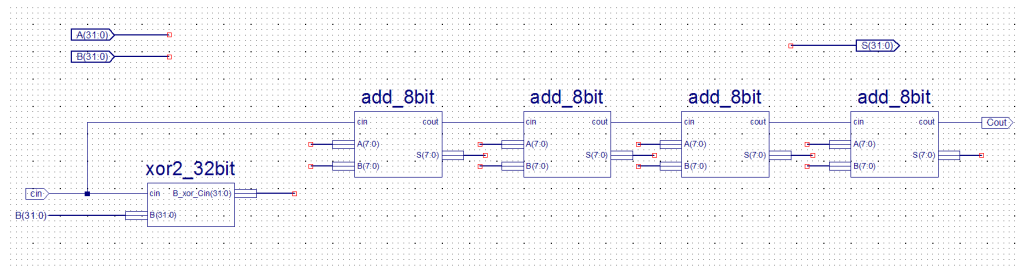
- Addition or Subtraction: When option is '00', the first function is active. A(31:0) plus B(31:0) when Sub\_enable is 0, and 1 is for A(31:0) minus B(31:0).
- Left shift for 1 bit on A.
- Bitwise AND by A(31:0) and B(31:0).
- Bitwise OR by A(31:0) and B(31:0).

### 2. Screen capture of schematics

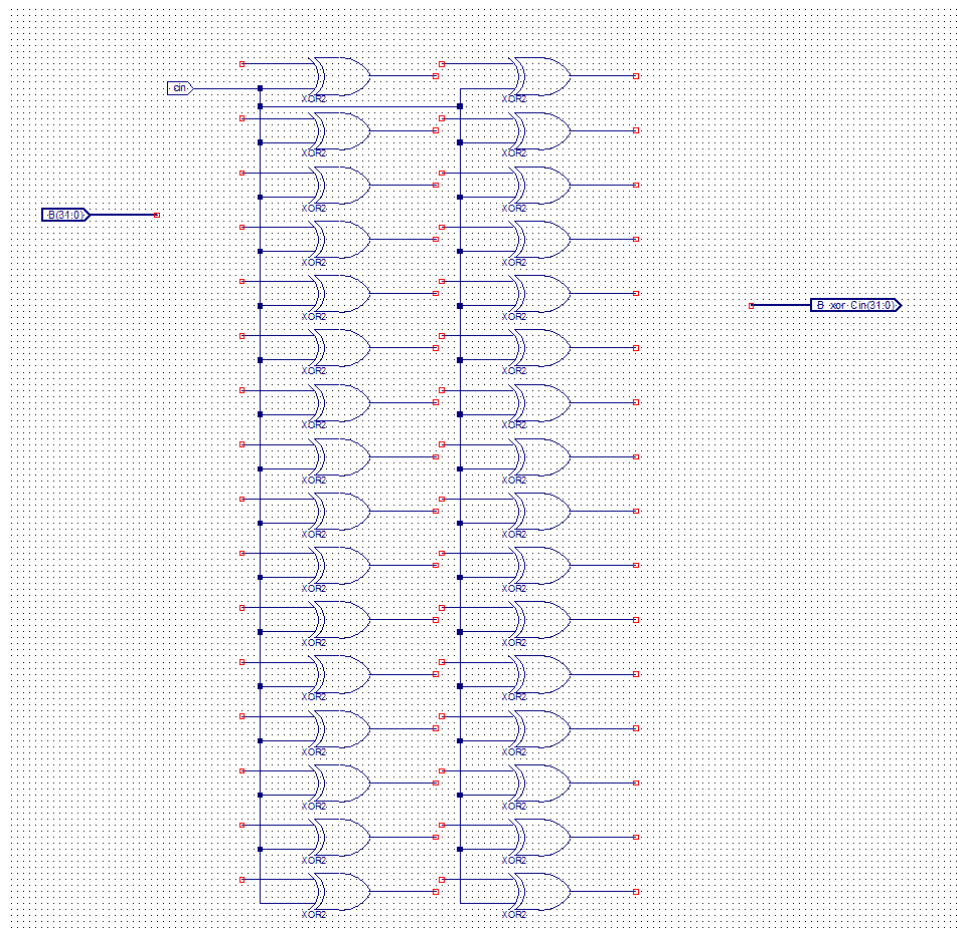
- 32-bit ALU



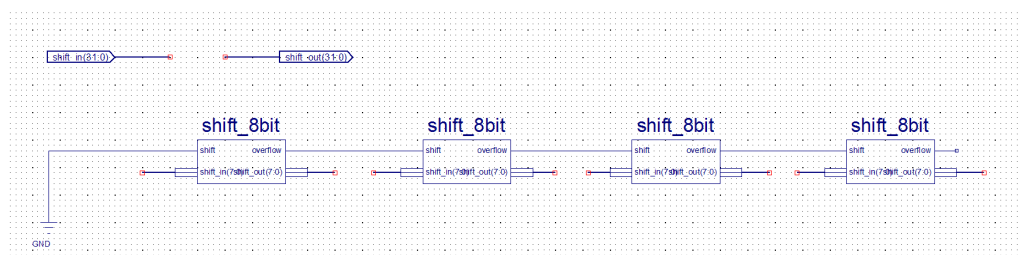
## ■ 32-bit Add\_Sub



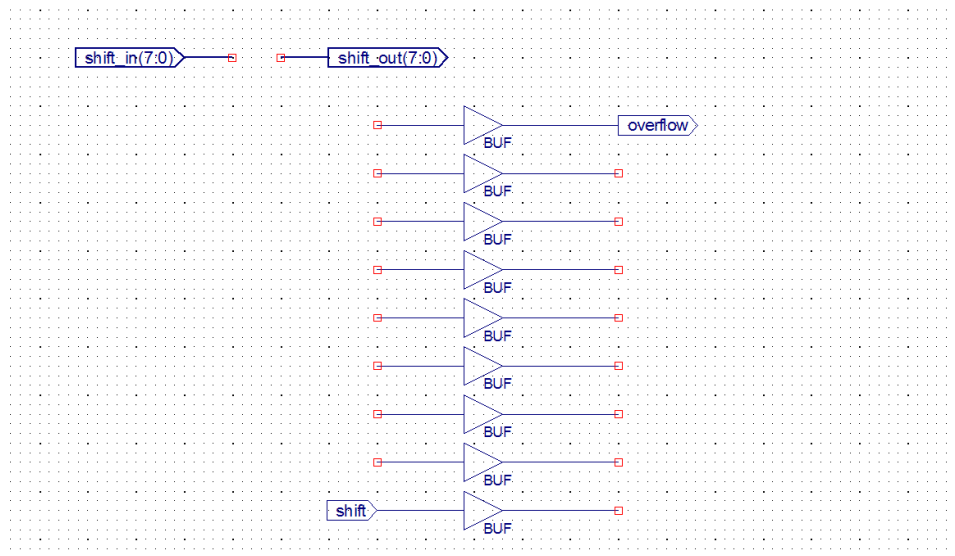
## ■ 32-bit Bitwise XOR



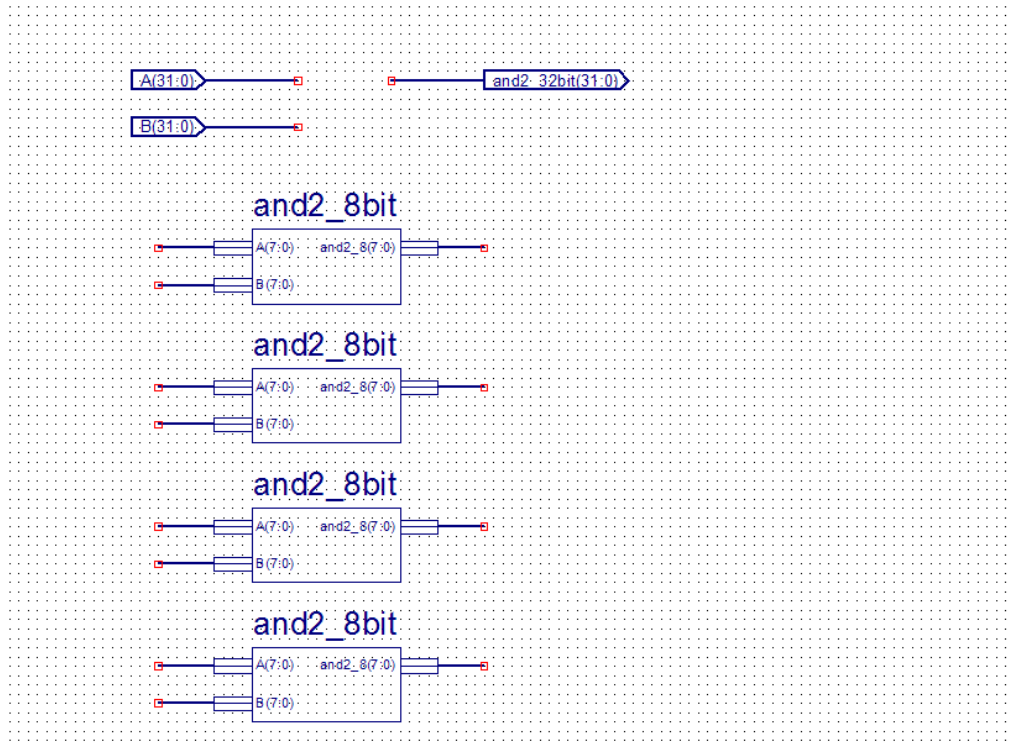
## ■ 32-bit Shift



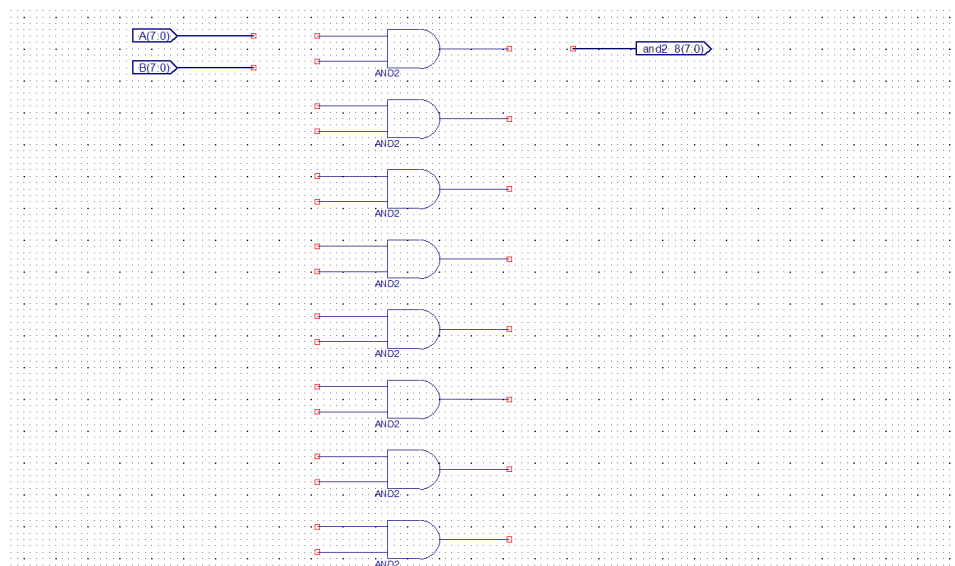
## ■ 8-bit Shift



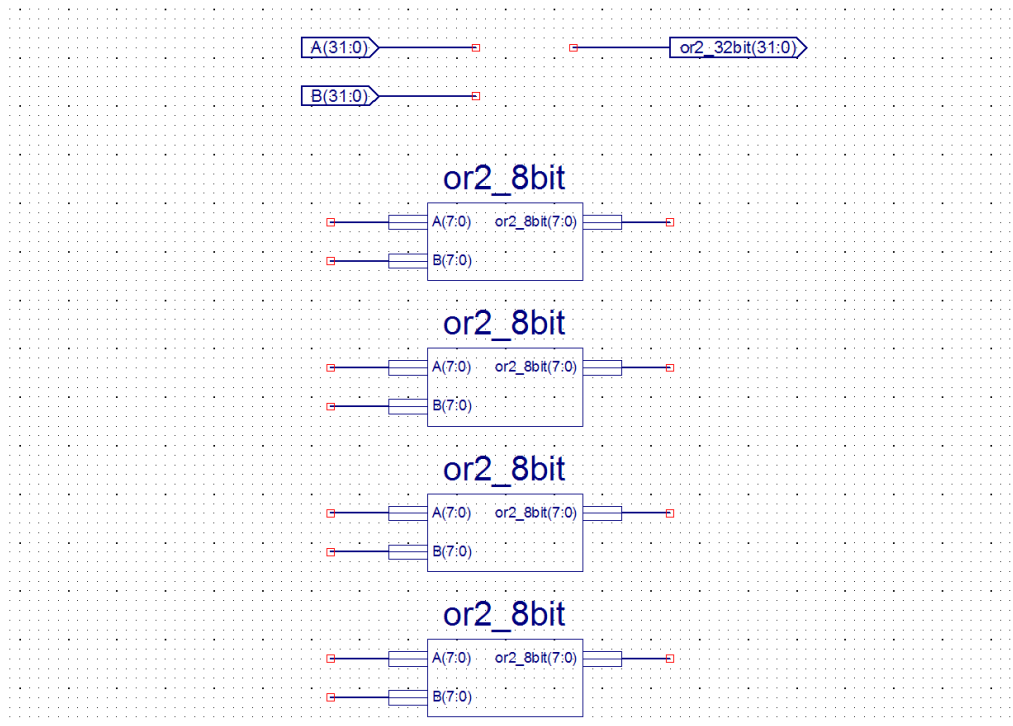
### ■ 32-bit Bitwise AND



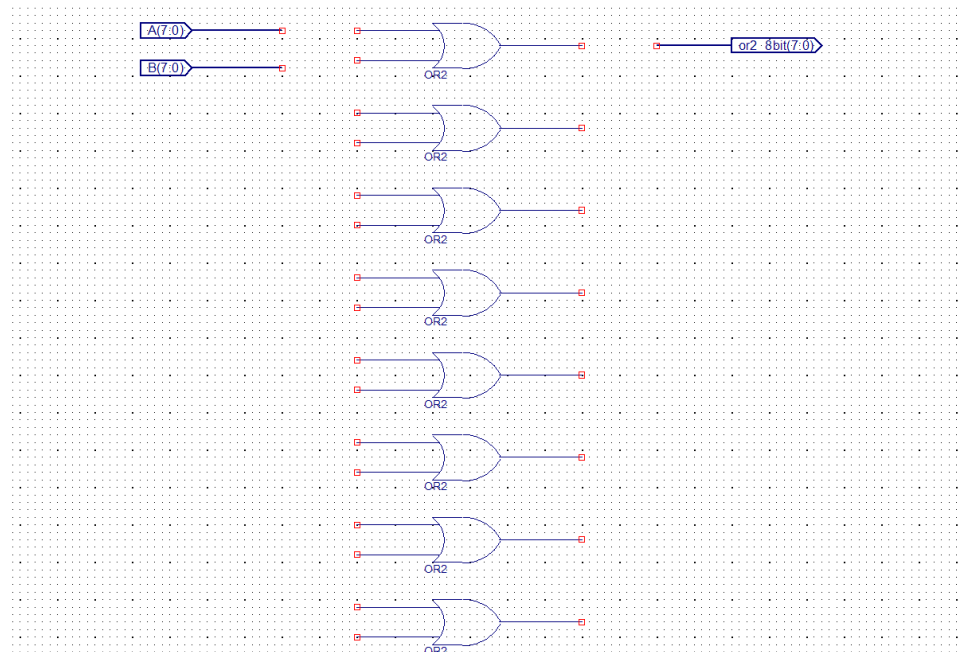
### ■ 8-bit Bitwise AND



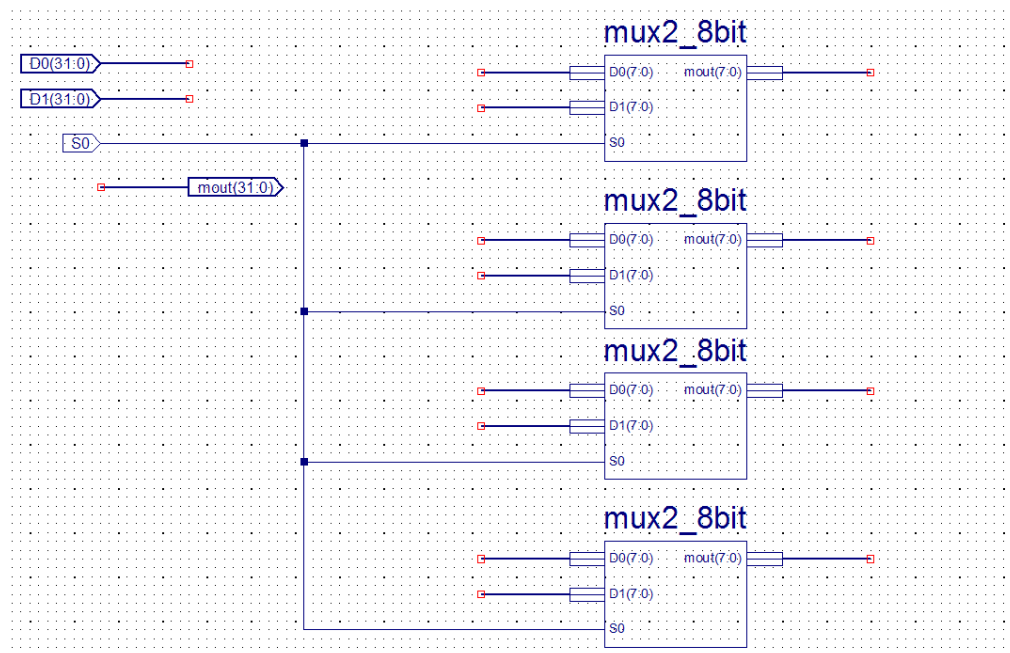
### ■ 32-bit Bitwise OR



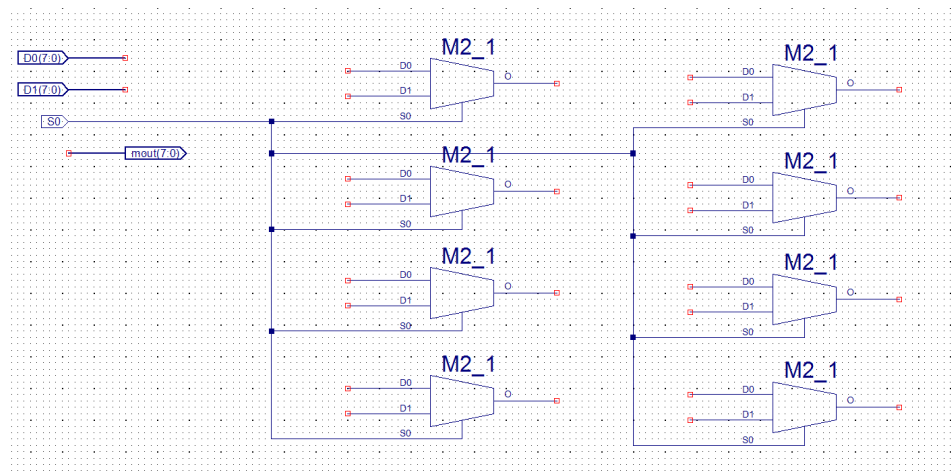
#### ■ 8-bit Bitwise OR



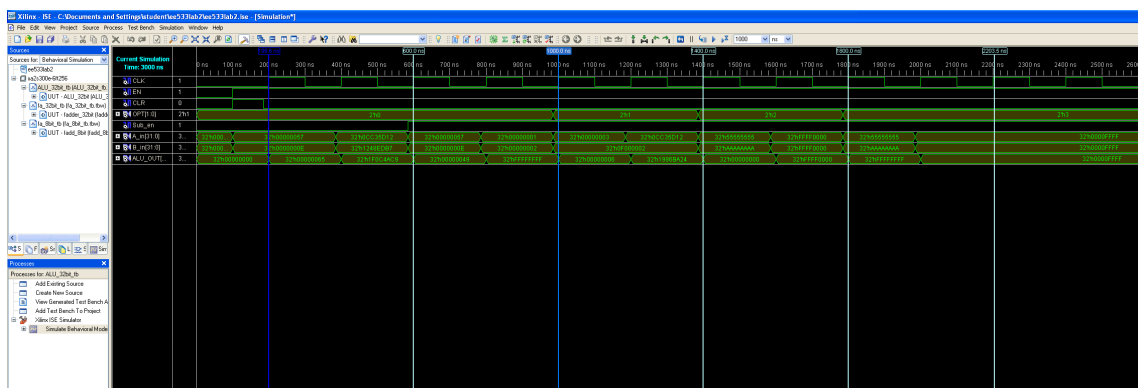
#### ■ 32-bit Bitwise 2 to 1 MUX



### ■ 8-bit Bitwise 2 to 1 MUX



### 3. Screen capture of the waveforms generated by the behavioral simulation tools



### 4. Log file of the mapper

**Design Information**

Command Line : map -ise "C:/Documents and Settings/student/ee533lab2/ee533lab2.ise" -intstyle ise -p xa2s300e-ft256-6 -cm area -pr off -k 4 -c 100 -tx off -o ALU 32bit map.ncd ALU 32bit.ncd

ALU 32bit.pcf

Target Device : xa2s300e  
 Target Package : ft256  
 Target Speed : -6  
 Mapper Version : aspartan2e -- \$Revision: 1.46 \$  
 Mapped Date : Sat Jan 25 22:32:01 2025

**Design Summary**

Number of errors: 0  
 Number of warnings: 0

**Logic Utilization:**

Number of Slice Flip Flops:	32 out of 6,144	1%
Number of 4 input LUTs:	160 out of 6,144	2%

**Logic Distribution:**

Number of occupied Slices:	80 out of 3,072	2%
Number of Slices containing only related logic:	80 out of 80	100%
Number of Slices containing unrelated logic:	0 out of 80	0%

\*See NOTES below for an explanation of the effects of unrelated logic

Total Number of 4 input LUTs: 160 out of 6,144 2%  
 Number of bonded IOBs: 101 out of 178 56%  
 Number of Tbufs: 65 out of 3,200 2%  
 Number of GCLKs: 1 out of 4 25%  
 Number of GCLKIOBs: 1 out of 4 25%

Peak Memory Usage: 125 MB  
 Total REAL time to MAP completion: 0 secs  
 Total CPU time to MAP completion: 0 secs

**NOTES:**

Related logic is defined as being logic that shares connectivity - e.g. two LUTs are "related" if they share common inputs. When assembling slices, Map gives priority to combine logic that is related. Doing so results in the best timing performance.

Unrelated logic shares no connectivity. Map will only begin packing unrelated logic into a slice once 99% of the slices are occupied through related logic packing.

Note that once logic distribution reaches the 99% level through related logic packing, this does not mean the device is completely utilized. Unrelated logic packing will then begin, continuing until all usable LUTs and FFs are occupied. Depending on your timing budget, increased levels of unrelated logic packing may adversely affect the overall timing performance of your design.

**Table of Contents**

Section 1 - Errors  
 Section 2 - Warnings  
 Section 3 - Informational  
 Section 4 - Removed Logic Summary

**The number of D-FF: 32**

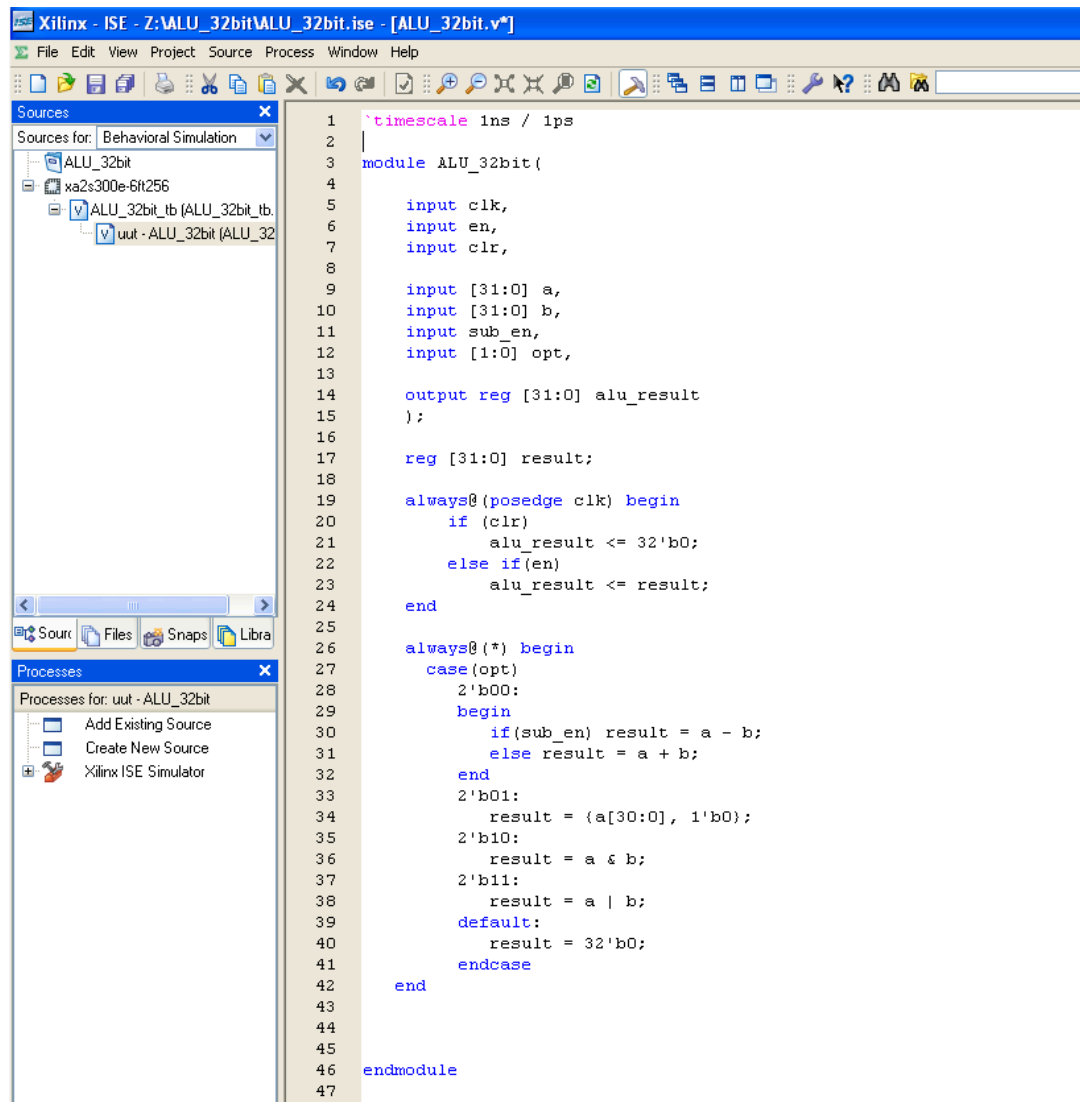
**The number of LUTs: 160**

## C : Write a Verilog equivalent of 32-bit ALU

### 1. Code for logic and testbench

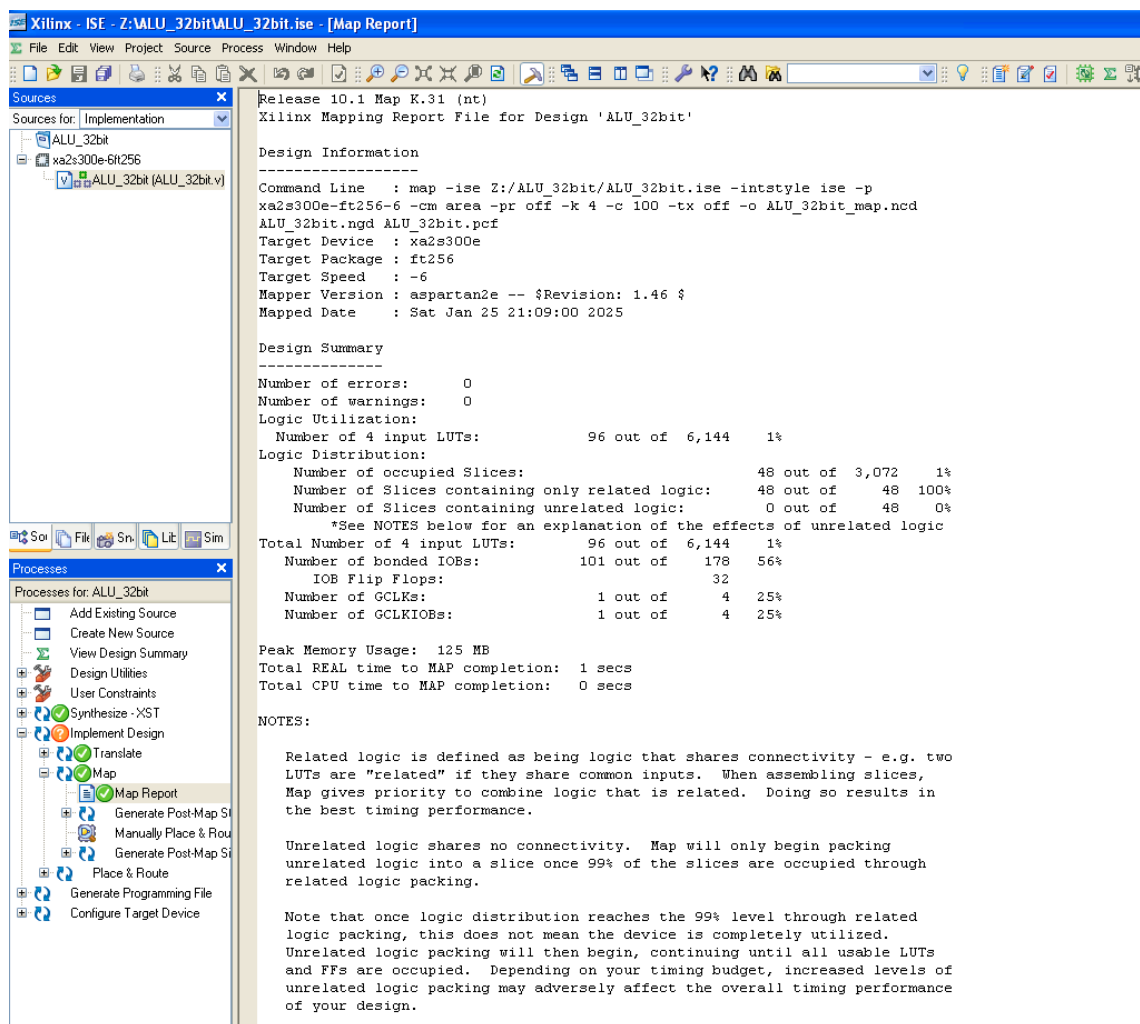
- Logic





- Testbench





**The number of D-FF: 32**

**The number of LUTs: 96**

#### 4. Brief comment on the number of gates as compared to the schematic version

The number of gates generated by Verilog code is higher than that generated by schematics because redundant gates are produced when designing multi-layer nested wide-bit components using schematics.