National Cheng Kung University

Department of Electrical Engineering

Introduction to VLSI CAD (Spring 2024)

Lab Session 3

Design of ALU and Multiplication Using Verilog Coding

Name	Student ID	
劉冠妤	E24116152	
Practical Sections:	Points	Marks
Prob A	30	
Prob B	30	
Prob C	20	
Report	15	
File hierarchy, naming…etc.	5	
Notes		

Due Date: 15:00, March 13, 2024 @ moodle

Deliverables

1) All Verilog codes including testbenches for each problem should be uploaded.
   NOTE: Please **DO NOT** include source code in the paper report!
2) All homework requirements should be uploaded in this file hierarchy or you will not get the full credit.
   NOTE: Please **DO NOT** upload waveforms!
3) Important! TA will use the command in Appendix A to check your design under SoC Lab environment, if your code can not be recompiled by TA successfully using the commands, you will not get the full credit.
4) If you upload a dead body which we can't even compile you will get **NO** credit!
5) All Verilog file should get at least 90% superLint Coverage.
6) File hierarchy should not be changed; it may cause your code can not be recompiled by TA successfully using the autograding commands

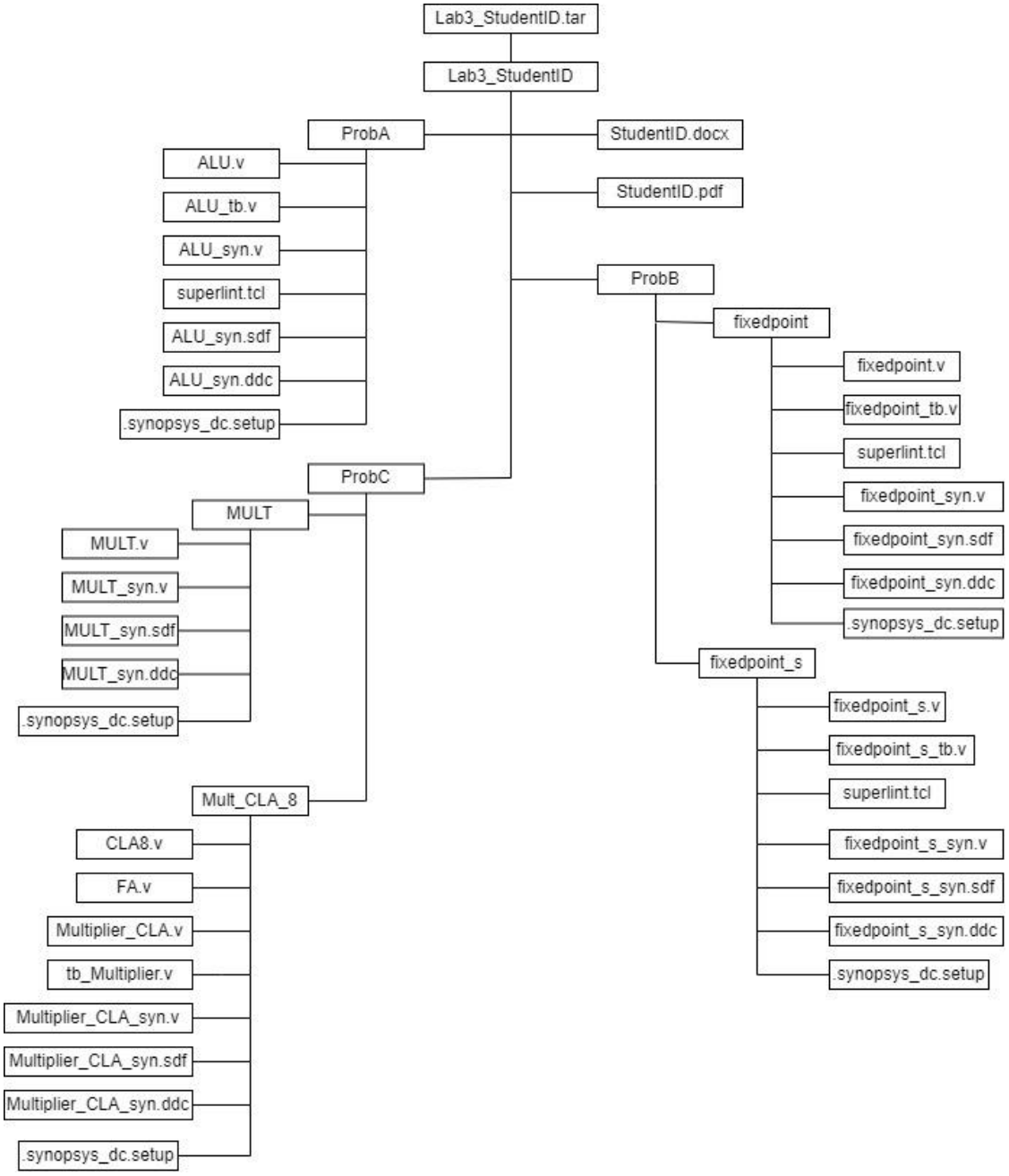
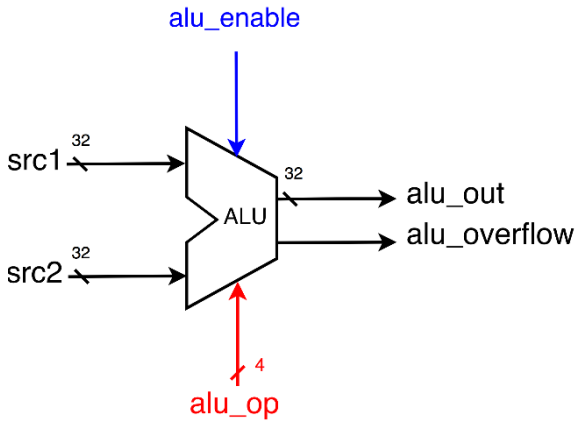


Fig.1 File hierarchy for Homework submission

Prob A: Arithmetic Logic Unit

Design your Verilog code with the following specifications:



1. Based on the reference code, please implement the following operations.

alu_op	Operation	Description
00000	ADD	$src1_{signed}$ + $src2_{signed}$
00001	SUB	$src1_{signed}$ - $src2_{signed}$
00010	OR	src1 or src2
00011	AND	src1 and src2
00100	XOR	src1 xor src2
00101	NOT	Invertion of src1
00110	NAND	src1 nand src2
00111	NOR	src1 nor src2

alu_op	Operation	Description
01011	SLT	alu_out = (src1 signed < src2 signed) ? 32'd1 : 32'd0
01100	SLTU	alu_out = (src1 unsigned < src2 unsigned) ? 32'd1 : 32'd0
01101	SRA	alu_out = src1 signed >>> src2 unsigned
01110	SLA	alu_out = src1 signed <<< src2 unsigned
01111	SRL	alu_out = src1 unsigned >> src2 unsigned
10000	SLL	alu_out = src1 unsigned << src2 unsigned
10001	ROTR	alu_out = src1 rotate right by "src2 bits"
10010	ROTL	alu_out = src1 rotate left by "src2 bits"
10011	MUL	alu_out = lower 32 bits of (src1 * src2)
10100	MULH	alu_out = upper 32 bits of (src1 signed * src2 signed)
10101	MULHSU	alu_out = upper 32 bits of (src1 signed * src2 unsigned)
10110	MULHU	alu_out = upper 32 bits of (src1 unsigned * src2 unsigned)

a. The frame code and testbench are given. Follow the frame code to finish this homework. The decimal part should be rounded.
b. Follow the PPT file to synthesize your code.

After you synthesize your design, you may have some information about the circuit. Fill in the following form.

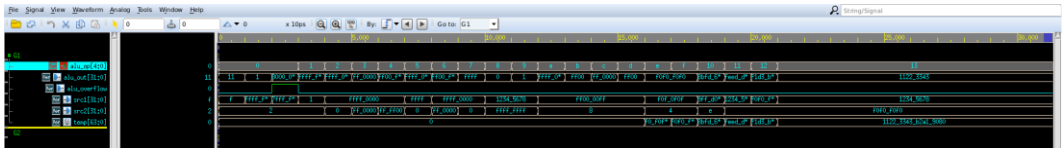
Timing (slack)	Area (total cell area)	Power (total)
0.0	**3961.820251**	**3.8089mW**

Please attach your design waveforms.

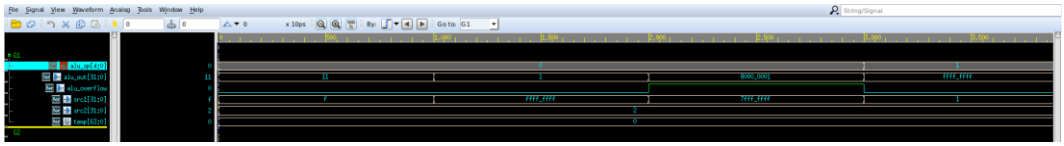
Your simulation result on the terminal.

Your waveform (RTL & Synthesis) :

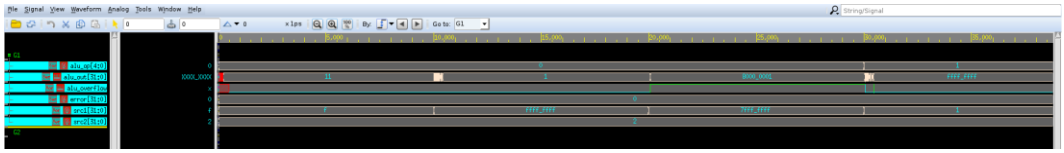
complete RTL waveform:



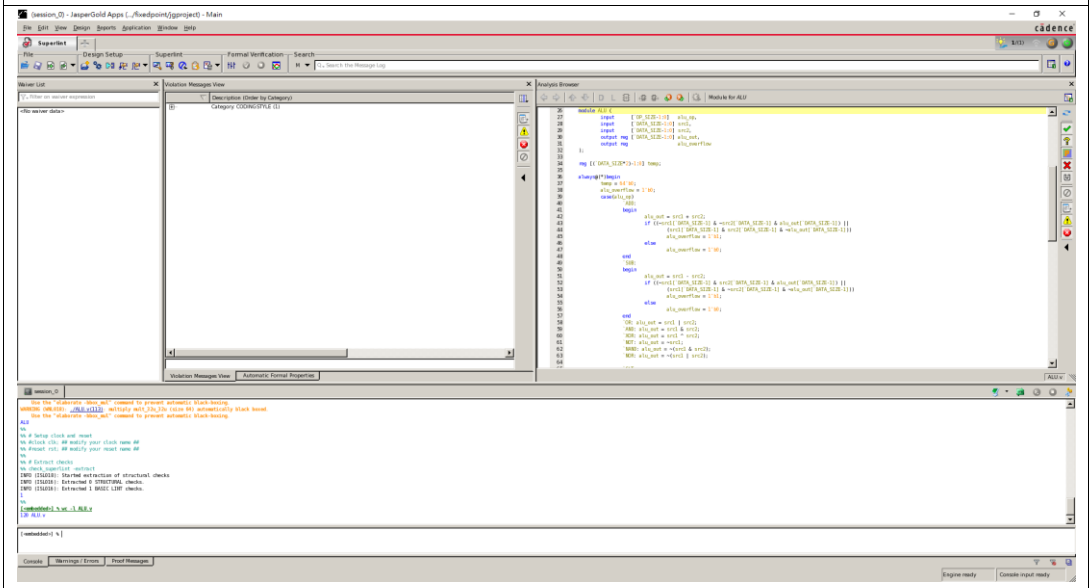
local RTL waveform:



local Synthesis waveform:



SuperLint Coverage



Warning lines = 1

Total lines = 120

Coverage percentage = 99.167%

Prob B-1: Practice fixed point

Design your Verilog code with the following specifications: Number format: unsigned numbers.

c. The frame code and testbench are given. Follow the frame code to finish this homework. The decimal part should be rounded.

d. Follow the PPT file to synthesize your code.

After you synthesize your design, you may have some information about the circuit. Fill in the following form.

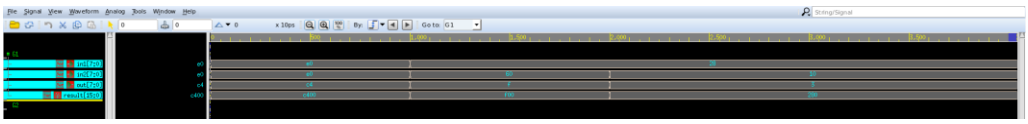
Timing (slack)	Area (total cell area)	Power (total)
0.01	**79.418881**	**5.7229×10^{-5}W**

Please attach your design waveforms.

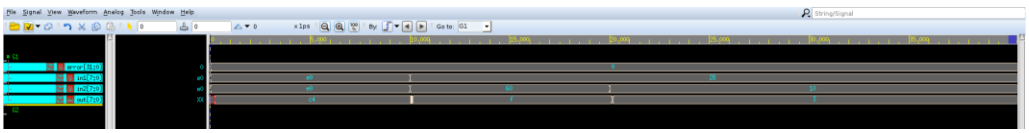
Your simulation result on the terminal.

Your waveform (RTL & Synthesis) :

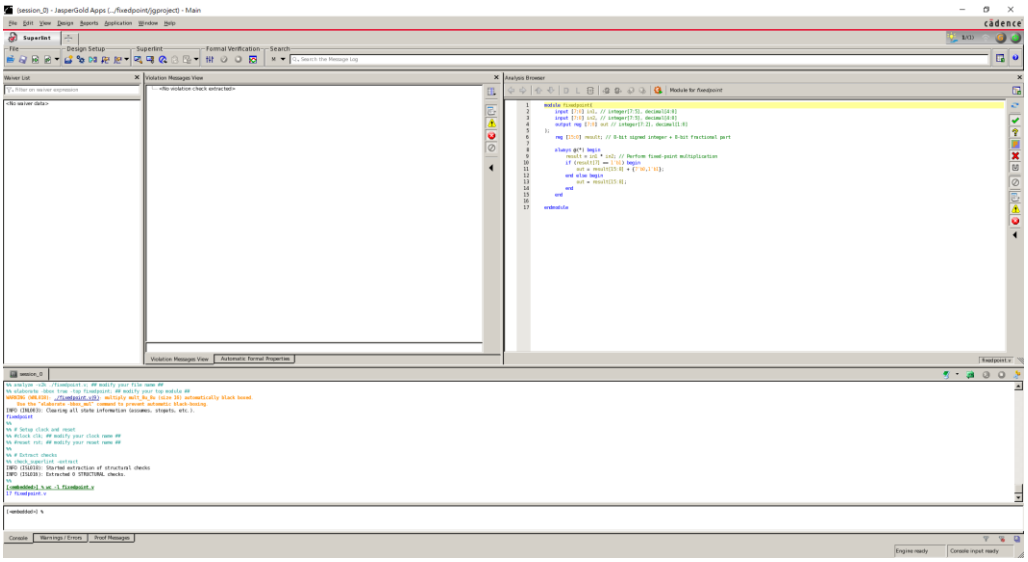
RTL waveform:



Synthesis waveform:



SuperLint Coverage



Warning lines = 0

Total lines = 17

Coverage percentage = 100%

Prob B-2: Practice fixed point (signed)

Design your Verilog code with the following specifications: Number format: signed numbers.

a. The frame code and testbench are given. Follow the frame code to finish this homework. The decimal part should be rounded.

b. Follow the PPT file to synthesize your code.

After you synthesize your design, you may have some information about the circuit. Fill in the following form

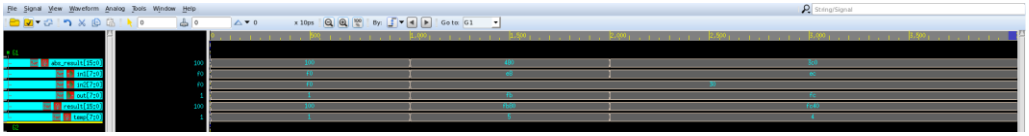
Timing (slack)	Area (total cell area)	Power (total)
0.0	**104.250243**	**8.0239×10^{-2}W**

Please attach your design waveforms.

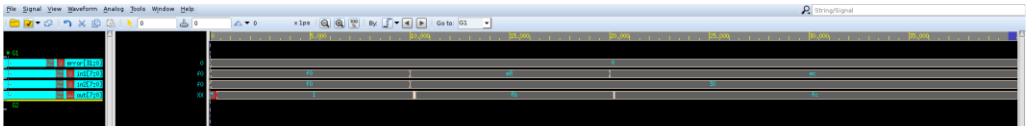
Your simulation result on the terminal.

Your waveform (RTL & Synthesis) :

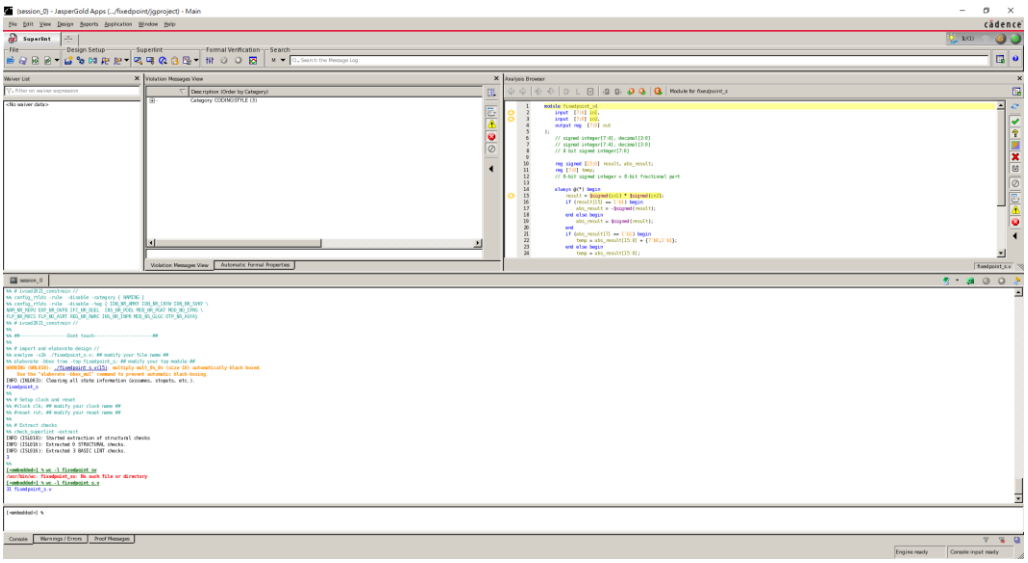
RTL waveform:



Synthesis waveform:



SuperLint Coverage



Warning lines = 3

Total lines = 31

Coverage percentage = 90.32%

Prob C: Performance comparison

Synthesize the 8*8-bit CLA multiplier implemented in Lab2 and the given 8*8-bit multiplier separately.

You should answer the following questions:

1. Determine the lowest achievable clock period for both, along with the corresponding area and power consumption.

	Clock period	Timing (slack)	Area (total cell area)	Power (total)
CLA multiplier	**0.50**	**0.0**	**221.667845**	**0.1572mW**
“*”operator	**0.40**	**0.0**	**74.908801**	**5.0555×10^{-2}W**

2. Considering clock period and area, which structure has the better performance.

The "*" operator structure has better performance because its clock period is smaller than that of the CLA multiplier, and its total cell area is almost one-third of the total cell area of the CLA multiplier, which means that it has better performance.

At last, please write the lessons learned from this lab session, or some suggestions for this lab session. Thank you.

From this class I learned implementation of combinational logic and design of an Arithmetic Logic Unit (ALU), Multiplication of Fixed Point Number and the most impressive thing is Synthesis of Combinational Logic. After several implementations in the computer classroom A, I am now more proficient in operating these programs. However, I still often encounter problems with instructions that need to be repeated many times because the process is not planned well or details are missed. I also encountered a puzzling problem. The contents of the newly purchased USB suddenly became garbled after a few uses. My other USB could not be read at all on the SOC lab computers, and I still cannot enter the soc lab with my student ID card. Although the process is a bit cumbersome, it feels great to discuss it with friends and go to the computer classroom together.

Problem		Command
ProbA	Compile	% vcs -R ALU.v -full64
	Simulate	% vcs -R ALU_tb.v -debug_access+all -full64 +define+FSDB
	Synthesis	% vcs -R ALU_tb.v -debug_access+all -full64 +define+FSDB+syn
ProbB-1	Compile	% vcs -R fixedpoint.v -full64
	Simulate	% vcs -R fixedpoint_tb.v -debug_access+all -full64 +define+FSDB
	Synthesis	% vcs -R fixedpoint_tb.v -debug_access+all -full64 +define+FSDB+syn
ProbB-2	Compile	% vcs -R fixedpoint_s.v -full64
	Simulate	% vcs -R fixedpoint_s_tb.v -debug_access+all -full64 +define+FSDB
	Synthesis	% vcs -R fixedpoint_s_tb.v -debug_access+all -full64 +define+FSDB+syn

Appendix A : Commands we will use to check your homework