**National Cheng Kung University**

**Department of Electrical Engineering**

***Introduction to VLSI CAD (Spring 2021)***

Lab Session 2

**Design and Simulation of Priority Encoder and Ripple Carry Adder**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student ID | | |
| 陳旭祺 | E24099059 | | |
| Practical Sections: | | Points | Marks |
| Prob A | | 25 |  |
| Prob B | | 25 |  |
| Prob C | | 30 |  |
| Report | | 20 |  |
| Notes | | | |

**Due Date: 15:00, March 10, 2021 @ 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!

1. All homework requirements should follow the naming rule in this file hierarchy or you will not get the full credit. (Hint: **%sh check.sh**)

NOTE: Please **DO NOT** upload waveforms!

1. 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.
2. **If you upload a dead body, which we cannot even compile, you will get NO credit!**
3. **All Verilog file should get at least 90% Superlint Coverage.**
4. The name of uploaded file should be **Lab2\_StudentID.tar**, and the student ID must be **uppercase**. (Ex. Lab2\_E24081234.tar)

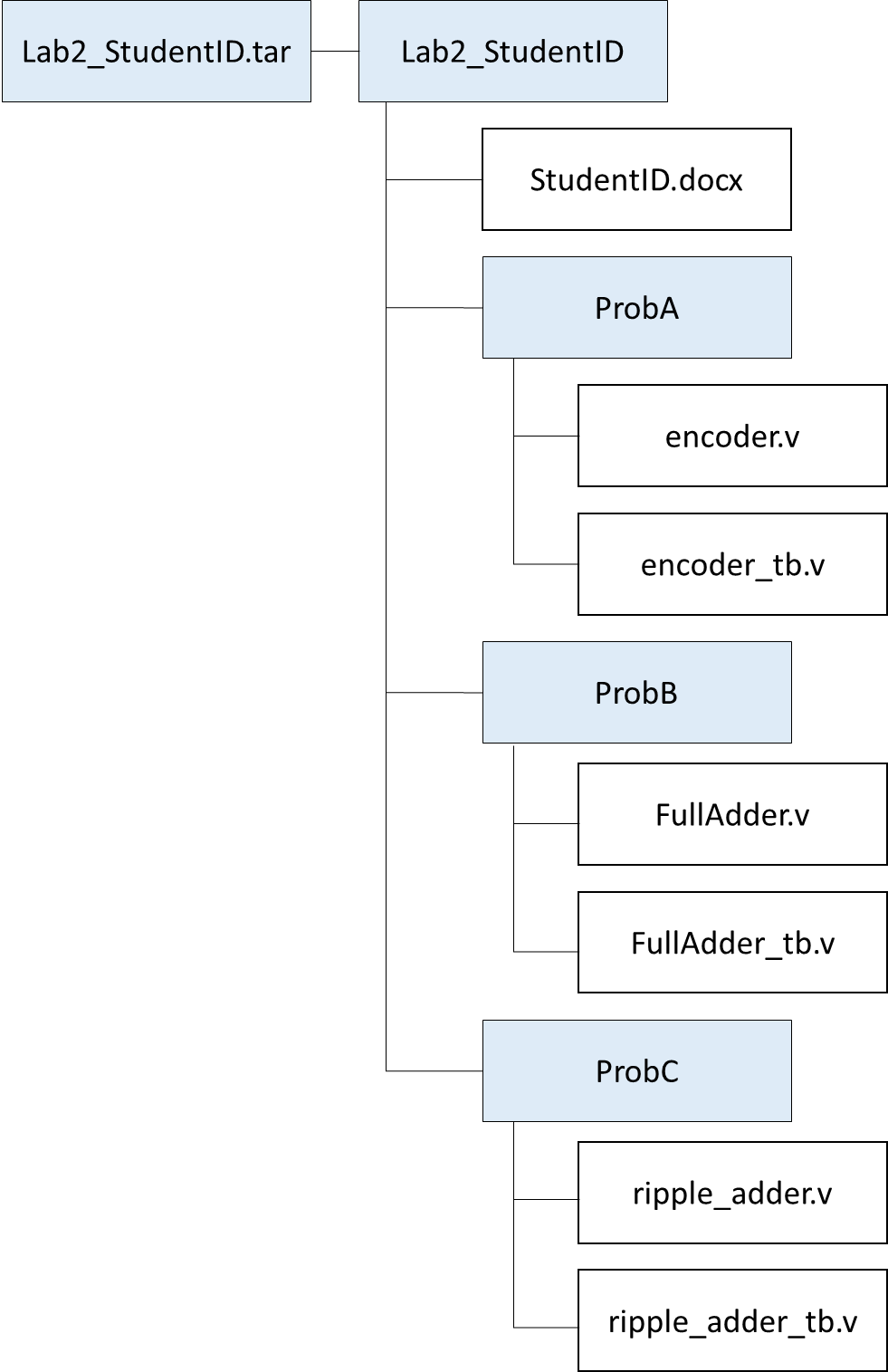
****

Fig.1 File hierarchy for Homework submission

**Objectives:**

**Help students get familiar with the CAD tools (NC-Verilog & Verdi) for digital logic design. Please go through the hands-on exercise step-by-step.**

Prob A: Design Steps

1. ****Understand the function of the target component.****

A priority encoder is a device that compresses several inputs into a smaller number of outputs.

**Exercise 1-1: Draw the block diagram of a 4-to-2 priority encoder.**

|  |
| --- |
|  |

1. ****Describe in digital logic manner for a simple component. Please fill the blank cells in the truth table and complete the Boolean equation of a**** *****4-to-2 priority encoder.*****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Truth Table of a priority encoder | | | | | |
| Inputs | | | | Outputs | |
| I3 | I2 | I1 | I0 | O1 | O0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | x | 0 | 1 |
| 0 | 1 | x | x | 1 | 0 |
| 1 | x | x | x | 1 | 1 |

Boolean Equation of a priority encoder (the logic of )

1. **Derive a gate-level implementation**

**Draw a Schematic View of a 4-to-2 priority encoder (gate-level diagram)**

|  |
| --- |
|  |

1. Implement a 4-to-2 priority encoder **based on the gate-level diagram you drawn in 3) by verilog code. (The module name should be encoder)**

* You only need to upload your v-code file to moodle, **do not** paste your code here.

1. **Read the testbench for** a 4-to-2 priority encoder **you implemented in 4). (The module is named encoder\_tb)**

* You only need to upload your v-code to moodle, **do not** paste your code here.

1. **Compile the code you implemented in 4) using NC-Verilog. **If there are errors, please go back to 4) to debug your code and re-compile again.****

**(Hint : The command for compiling is %ncverilog encoder.v)**

1. **Simulate your design in 4) with the testbench in 5).**

**(Hint: The command for simulation is**

**%ncverilog encoder\_tb.v encoder.v +access+r +define+FSDB)**

1. **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 4) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform. **(Hint : The command to open nWave is %nWave &)****

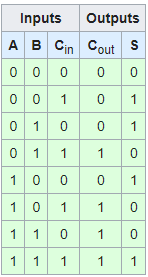
**In addition, you should check your coding style, there are no error messages and over 90% coverage with Superlint. Snapshot the result and *calculate Superlint coverage*. (Hint: The command to open Superlint is %jg -superlint superlint.tcl)**

|  |
| --- |
| Your simulation result on the terminal |
| 一張含有 文字 的圖片  自動產生的描述 |
| Your waveform |
|  |
| Explanation of your waveform |
| Priority : |
| **Superlint** screenshot and coverage |
| 一張含有 文字 的圖片  自動產生的描述 |

Prob B: Structral Coding

1. An adder is a digital circuit that performs addition of number. Please **design a full adder in gate level.**
2. **Draw a full adder in gate level.**

|  |
| --- |
| 經Truth Table與K map化簡得 |



1. ****Design a full adder in Structral coding (The module name should be**** FullAdder ****And the file you include should be**** FullAdder.v****)****

* **You only need to upload your v-code to moodle, do not paste your code here.**

1. Read the testbenche for **full adder** you implemented in 3). (The module is named FullAdder\_tb)

* You only need to upload your v-code to moodle, **do not** paste your code here.

1. Compile the code you implemented in 3) using NC-Verilog. If there are errors, please go back to 3) to debug your code and re-compile again. (Hint: The command for compiling is %ncverilog FullAdder.v)
2. Simulate your design in 3) with the testbench in 4).

**(Hint: The command for simulation is % ncverilog FullAdder\_tb.v FullAdder.v +access+r +define+FSDB)**

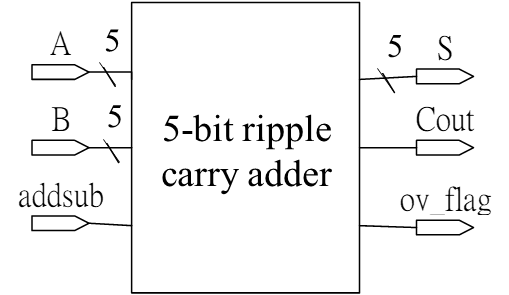
1. **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 3) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform. **(Hint : The command to open nWave is %nWave**** ****&)****

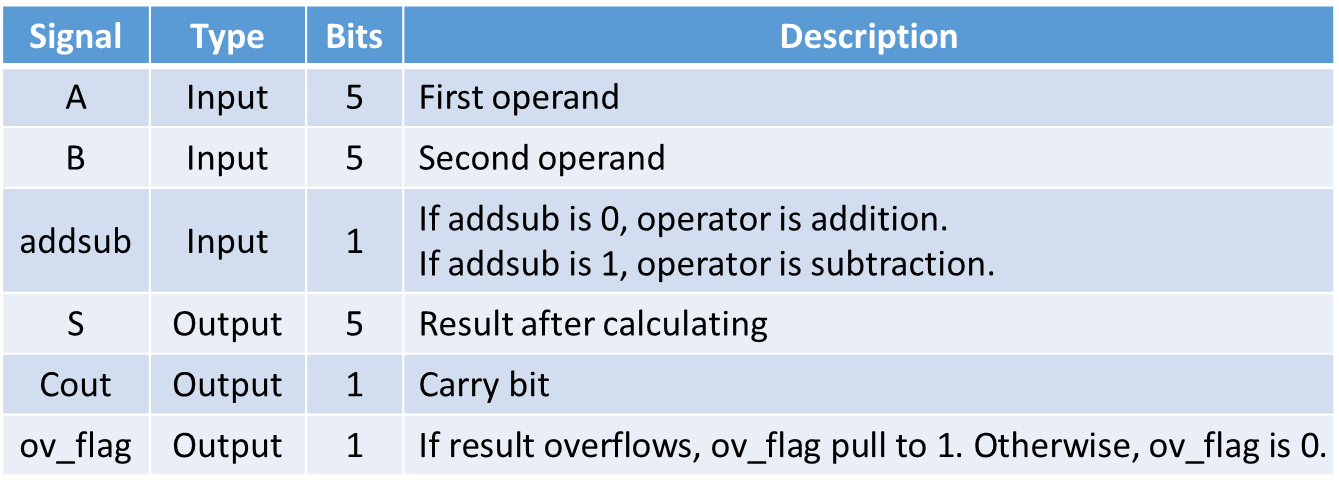
**In addition, you should check your coding style, there are no error messages and over 90% coverage with Superlint. Snapshot the result and *calculate Superlint coverage*. (Hint: The command to open Superlint is %jg -superlint superlint.tcl)**

|  |
| --- |
| Your simulation result on the terminal |
| 一張含有 文字 的圖片  自動產生的描述 |
| Your waveform |
|  |
| Explanation of your waveform |
| I deem it as in binary. |
| **Superlint** screenshot and coverage |
| 一張含有 文字 的圖片  自動產生的描述 |

Prob C: hieratical coding

1. ****Design a 5-bit add/sub ripple carry adder in hierarchical coding.****

****

****

1. **Your design should be named as** ripple\_adder.v and module names in the file should be the same as file name (ripple\_adder).
2. ****Simulate your design in 2) with the following test pattern in sample testbench.****

**(Hint: The command for compiling is %ncverilog ripple\_adder.v)**

**(Hint: The command for simulation is %ncverilog ripple\_adder\_tb.v ripple\_adder.v +define+FSDB +access+r)**

1. **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 2) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform. **(Hint : The command to open nWave is %nWave &)****

**In addition, you should check your coding style, there are no error messages and over 90% coverage with Superlint. Snapshot the result and *calculate Superlint coverage*. (Hint: The command to open Superlint is %jg -superlint superlint.tcl)**

|  |
| --- |
| Your simulation result on the terminal |
| 一張含有 文字 的圖片  自動產生的描述 |
| Your waveform |
|  |
| Explanation of your waveform |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | A | A in Bin | B | D in Bin | Add  sub | S | Cout | ov\_  flag | Ans | | 2 | 00010 | 1 | 00001 | 0 | 00011 | 0 | 0 | 3 | | 7 | 00111 | -5 | 11011 | 0 | 00010 | 0 | 0 | 2 | | -10 | 10110 | 13 | 01101 | 0 | 00011 | 0 | 0 | 3 | | 13 | 01101 | 12 | 01100 | 0 | 11001 | 0 | 1 | 25 | | -12 | 10100 | -5 | 11011 | 0 | 01111 | 1 | 1 | -17 | | -5 | 11011 | 2 | 11110 | 1 | 11001 | 0 | 0 | -7 | | -6 | 11010 | 11 | 10101 | 1 | 01111 | 1 | 1 | -17 | | 8 | 01000 | 11 | 10101 | 1 | 11101 | 0 | 0 | -3 | | 13 | 01101 | 7 | 11001 | 1 | 00110 | 0 | 0 | 6 | | 12 | 01100 | -6 | 00110 | 1 | 10010 | 1 | 1 | 18 | |
| **Superlint** screenshot and coverage |
| 一張含有 文字 的圖片  自動產生的描述 |

* At last, please write the lesson you learned from Lab 2.

I’ve learn how to design a basic logic unit in gate level or data flow level in Verilog. Furthermore, in Lab3, we use command `include "File\_Path/Filename" to call the FullAdder we design in Lab2, which is called hierarchical coding. IMO, it is similar to subfunction in C# and some syntax in Verilog is so closed to C’s that I forgot I’m design hardware instead of software sometimes.

Appendix A : Commands we will use to check your homework

|  |  |  |
| --- | --- | --- |
| **Problem** |  | **Command** |
| **ProbA** | Compile | %ncverilog encoder.v |
| Simulate | %ncverilog encoder\_tb.v encoder.v  +define+FSDB +access+r |
| **ProbB** | Compile | %ncverilog FullAdder.v |
| Simulate | %ncverilog FullAdder\_tb.v FullAdder.v  +define+FSDB +access+r |
| **ProbC** | Compile | % ncverilog ripple\_adder.v |
| Simulate | %ncverilog ripple\_adder\_tb.v ripple\_adder.v  +define+FSDB +access+r |