# National Cheng Kung University

## Department of Electrical Engineering

# *Introduction to VLSI CAD (Spring 2024)*

# Lab Session 2

# Design and Simulation of Carry-Lookahead Adder & Parallel-Prefix Adder & Multiplier

| Name | Student ID | |
|---|---|---|
| 鄭喆嚴 | E14096724 | |
| Practical Sections: | Points | Marks |
| Prob A | 15 | |
| Prob B | 30 | |
| Prob C | 40 | |
| Report | 15 | |
| Bonus | 10 | |
| Notes | | |

**Due Date: 14:59, 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

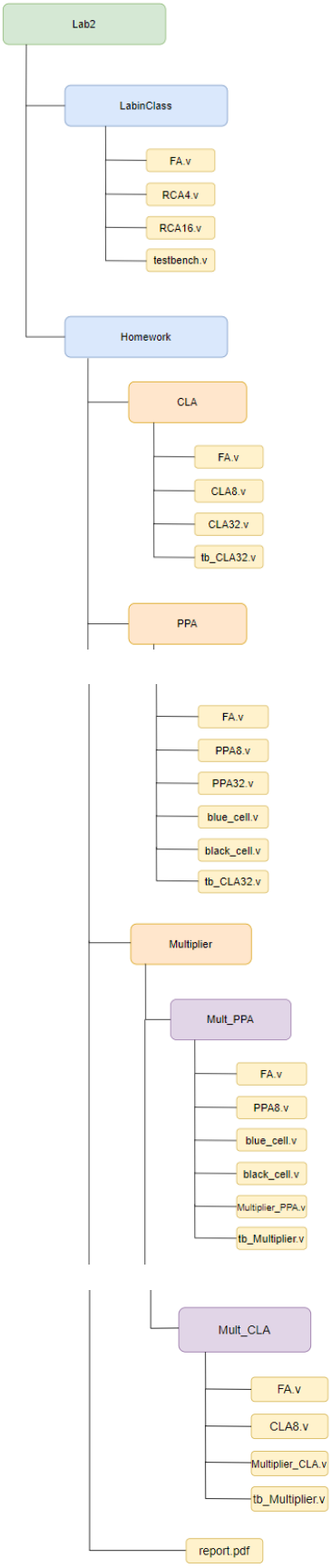



Fig.1 File hierarchy for Homework submission

## Objectives:

Help students get familiar with the CAD tools (VCS & Verdi) for digital logic design. Introduce different adder architectures and the algorithms behind them. Please go through the hands-on exercise step-by-step.

---

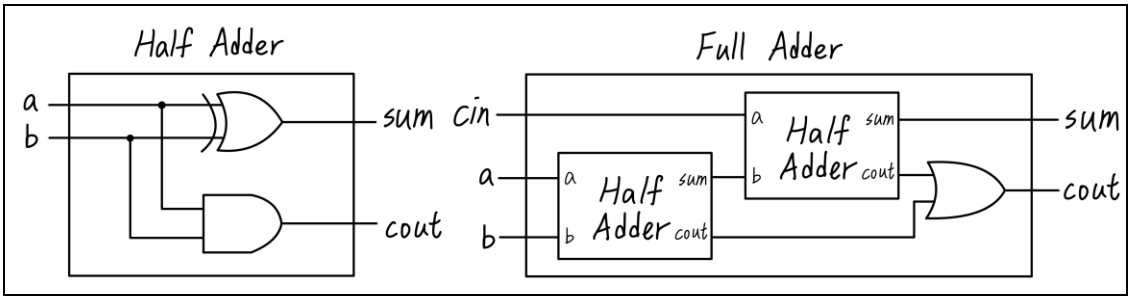
*Prob A: Design Steps & Carry-Lookahead Adder*

---

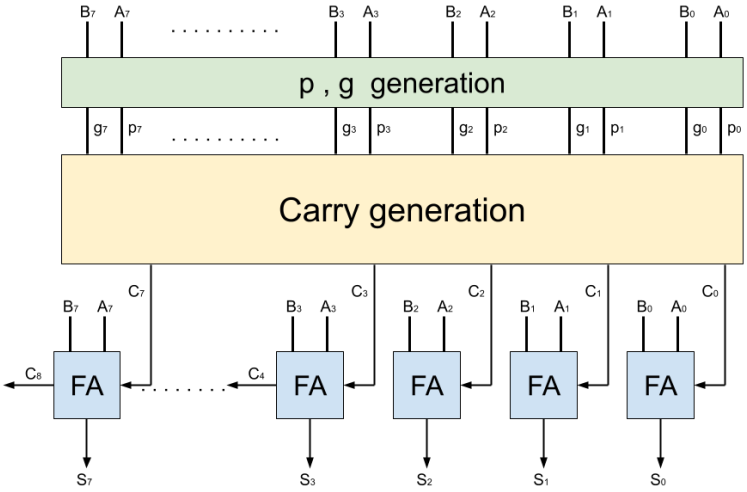
1) **An adder is a digital circuit that performs addition of number. Please design a full adder in gate level.**
2) **The truth table of full adder**

| Inputs | | | Outputs | |
|---|---|---|---|---|
| **A** | **B** | **$C_{in}$** | **$C_{out}$** | **S** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

3) **Draw a full adder in gate level.**




4) **Design a full adder in Structural coding (The module name should be FA. And the file you include should be FA.v)**
5) **Carry-lookahead-adder uses carry generator to alleviate the lengthy carry propagation procedure in ripple-carry-adder.**
6) **Derive the 8th carry bit $C_8$(ex. $C_2 = A_1B_1+(A_1+B_1)C_1 = g_1 + p_1C_1 = g_1 + p_1g_0 + p_1p_0C_0$)**




**ex.** $C_2 = A_1B_1+(A_1+B_1)C_1 = g_1 + p_1C_1 = g_1 + p_1g_0 + p_1p_0C_0$, $g_1 + p_1g_0 + p_1p_0C_0$ is the final result.

$C_8 = A_7B_7+(A_7+B_7)C_7 = g_7 + p_7C_7 = g_7 + p_7g_6 + p_7p_6C_6 = g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5C_5$

$= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4C_4$

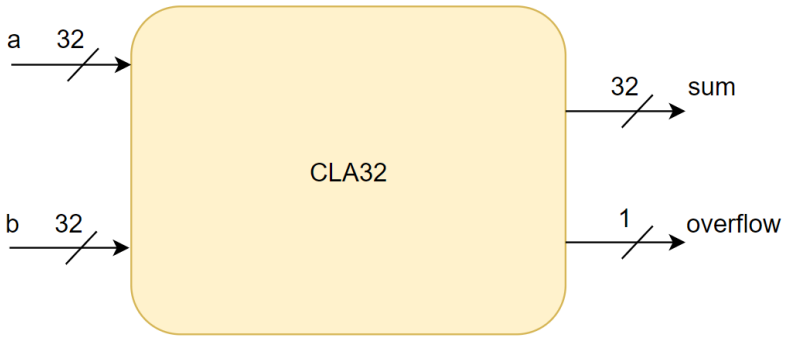
$= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3C_3$

$= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 + p_7p_6p_5p_4p_3p_2C_2$

$= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 + p_7p_6p_5p_4p_3p_2g_1 + p_7p_6p_5p_4p_3p_2p_1C_1$

$= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 + p_7p_6p_5p_4p_3p_2g_1 + p_7p_6p_5p_4p_3p_2p_1g_0 + p_7p_6p_5p_4p_3p_2p_1p_0C_0$

7) **Design a 8-bit carry lookahead adder** in **Structural coding** (The module name should be CLA8. And the file you include should be CLA8.v)
8) **Design a 32-bit carry lookahead adder in hierarchical coding** using previously designed CLA8 module**.** (The module name should be CLA32. And the file should be CLA32.v)



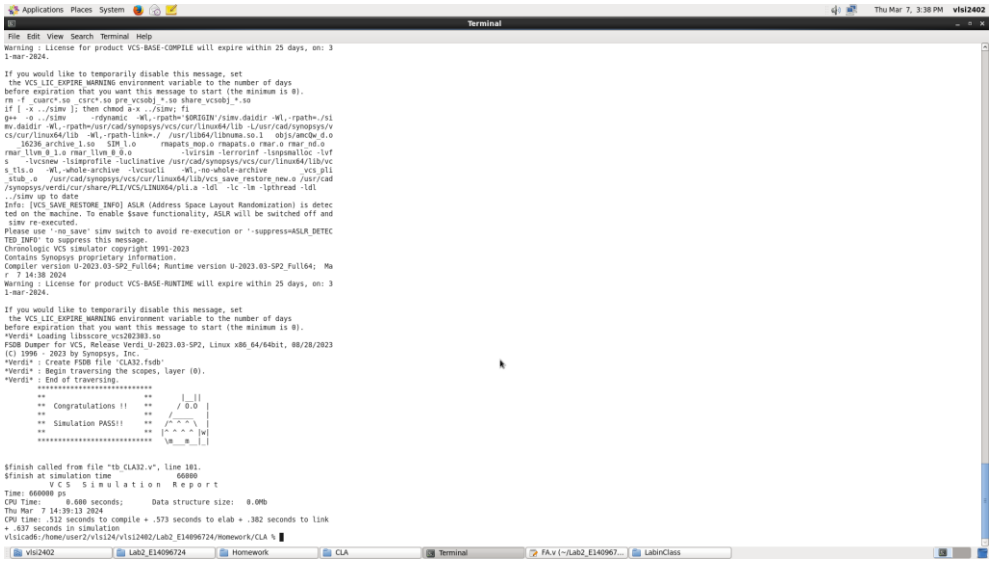
| Signal | IO | Bits | Description |
|---|---|---|---|
| a | Input | 32 | Addend |
| b | Input | 32 | Augend |
| sum | Output | 32 | Result after calculating |
| overflow | Output | 1 | Overflow detection |

9) **Simulate your design with the following test pattern in sample testbench.**
**(Hint: The command for compiling is % vcs -R tb_CLA.32v -full64)**
**(Hint: The command for simulation is %vcs -R tb_CLA32.v -debug_access+all -full64 +define+FSDB)**

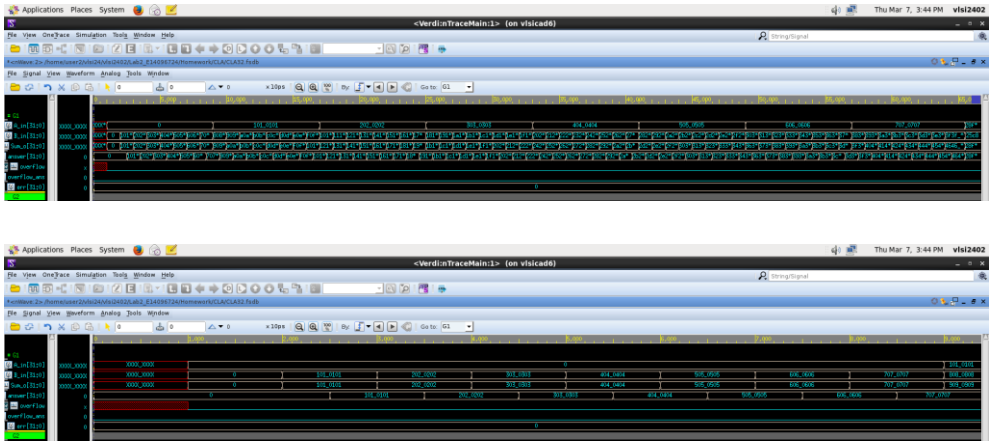
10) **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 2) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform.**
**(Hint : The command to open nWave is %nWave &)**
**In addition, you should check your coding style, and make sure that there are no error messages and coverage with Superlint must > 90 %. Snapshot the result and *calculate Superlint coverage*. (Hint: The command to open Superlint is %jg -superlint superlint.tcl)**

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

Your simulation result on the terminal.




Your waveform :

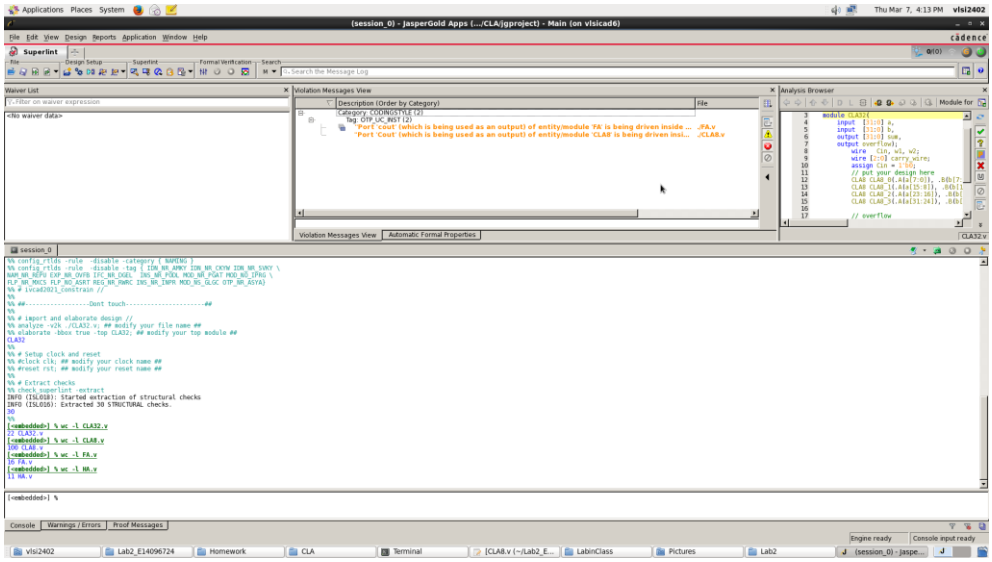


Explanation of your waveform :

上面波形圖 error 一直都為 0，代表全部的測資皆正確。

下面波形圖中可以看到所有測資皆有執行加法，並且加法的結果和 overflow 皆與測資的答案相同。

Superlint screenshot and coverage



Coverage :

CLA32.v : (1 - 0/22) * 100% = 100%

CLA8.v : (1 - 1/100) * 100% = 99%

FA.v : (1 - 1/16) * 100% = 93.75%

HA.v : (1 - 0/11) * 100% = 100%

## *Prob B: Parallel-Prefix Adder*

1) **Parallel-prefix-adder reduces the complexity of the carry generation circuitry in carry-lookahead-adder.**
2) **Recursive formulation of carry bits:**

$$Q(m, n) = \sum_{i=n}^{m}\left(\prod_{r=i+1}^{m} p_r\right) g_i$$

$C_0 = 0$

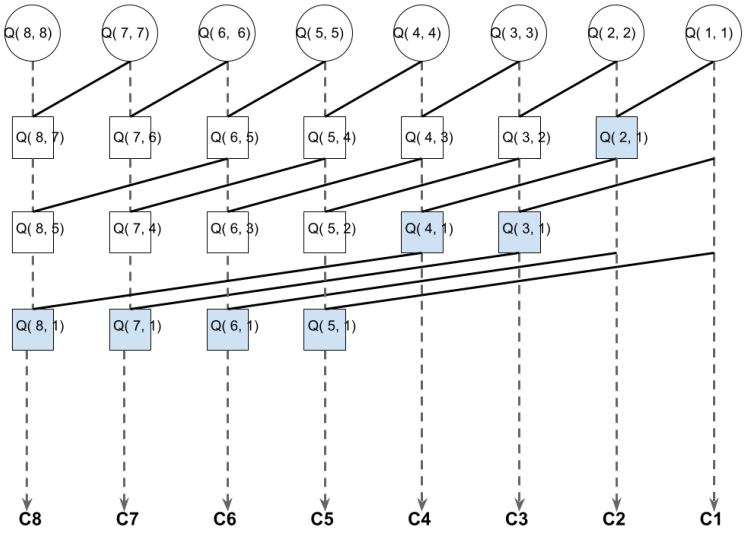
$C_1 = A_1B_1+(A_1 \oplus B_1)C_0 = g_1 + p_1C_0 = g_1$ $= Q(1, 1)$

$C_2 = A_2B_2+(A_2 \oplus B_2)C_1 = g_2 + p_2C_1 = g_2 + p_2g_1$ $= Q(2, 1) = p_2Q(1, 1) + Q(2, 2)$

$C_3 = A_3B_3+(A_3 \oplus B_3)C_2 = g_3 + p_3C_2 = g_3 + p_3g_2 + p_3p_2g_1$ $= Q(3, 1) = p_3p_2Q(1, 1) + Q(3, 2)$

$= p_3p_2Q(1, 1) + p_3Q(2, 2) + Q(3, 3)$

$C_4 = A_4B_4+(A_4 \oplus B_4)C_3 = g_4 + p_4C_3 = g_4 + p_4g_3 + p_4p_3g_2 + p_4p_3p_2g_1$ $= Q(4, 1) = p_4p_3Q(2, 1) + Q(4, 3)$




3) **Design a full adder in Structural coding (The module name should be FA. And the file you include should be FA.v)**
4) **Design black cell & blue cell in Structural coding** (The module name should be black_cell & blue_cell. And the file you include should be black_cell.v & blue_cell.v)
5) **Design a 8-bit parallel-prefix-adder** in **hierarchical coding using black cell & blue cell as basic unit** (The module name should be PPA8. And the file you include should be PPA8.v)
6) **Design a 32-bit parallel-prefix-adder in hierarchical coding** using previously designed PPA8 module**.** (The module name should be PPA32. And the file should

be PPA32.v)

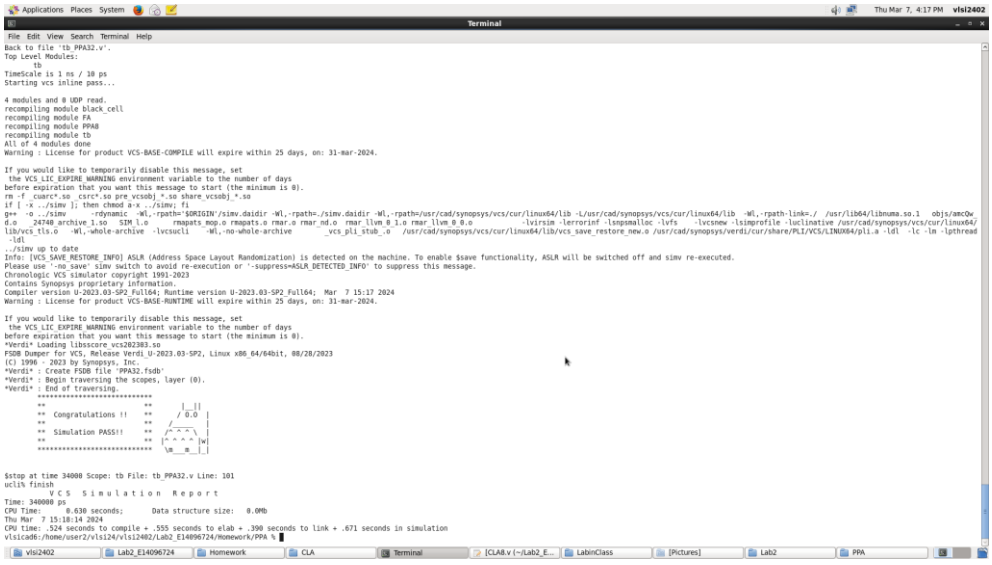

| Signal | IO | Bits | Description |
|---|---|---|---|
| a | Input | 32 | Addend |
| b | Input | 32 | Augend |
| sum | Output | 32 | Result after calculating |
| overflow | Output | 1 | Overflow detection |

7) **Simulate your design with the following test pattern in sample testbench.**
**(Hint: The command for compiling is % vcs -R tb_PPA32.v -full64)**
**(Hint: The command for simulation is %vcs -R tb_PPA32.v -debug_access+all -full64 +define+FSDB)**

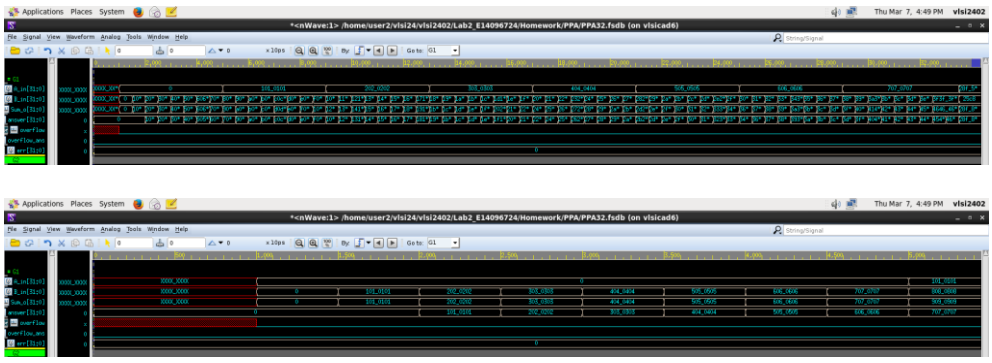
8) **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 2) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform.**
**(Hint : The command to open nWave is %nWave &)**
**In addition, you should check your coding style, and make sure that there are no error messages and coverage with Superlint must > 90 %. Snapshot the result and *calculate Superlint coverage*. (Hint: The command to open Superlint is %jg -superlint superlint.tcl)**

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

Your simulation result on the terminal.




Your waveform :

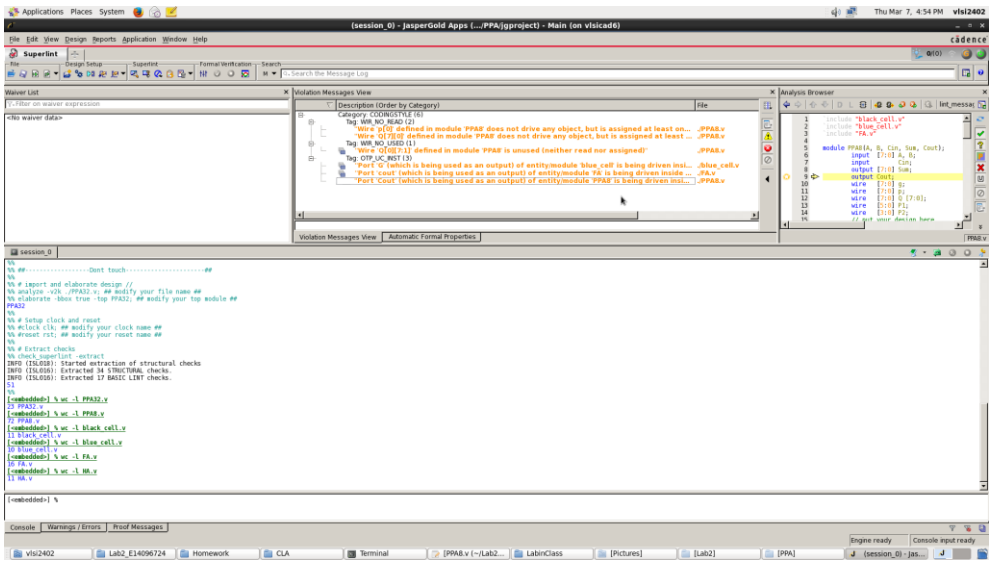


Explanation of your waveform :

上面波形圖 error 一直都為 0，代表全部的測資皆正確。

下面波形圖中可以看到所有測資皆有執行加法，並且加法的結果和 overflow 皆與測資的答案相同。

Superlint screenshot and coverage



Coverage :

PPA32.v : (1 - 0/23) * 100% = 100%

PPA8.v : (1 - 4/72) * 100% = 94.44%

black_cell : (1 - 0/11) * 100% = 100%

blue_cell : (1 - 1/10) * 100% = 90%

FA.v : (1 - 1/16) * 100% = 93.75%

HA.v : (1 - 0/11) * 100% = 100%

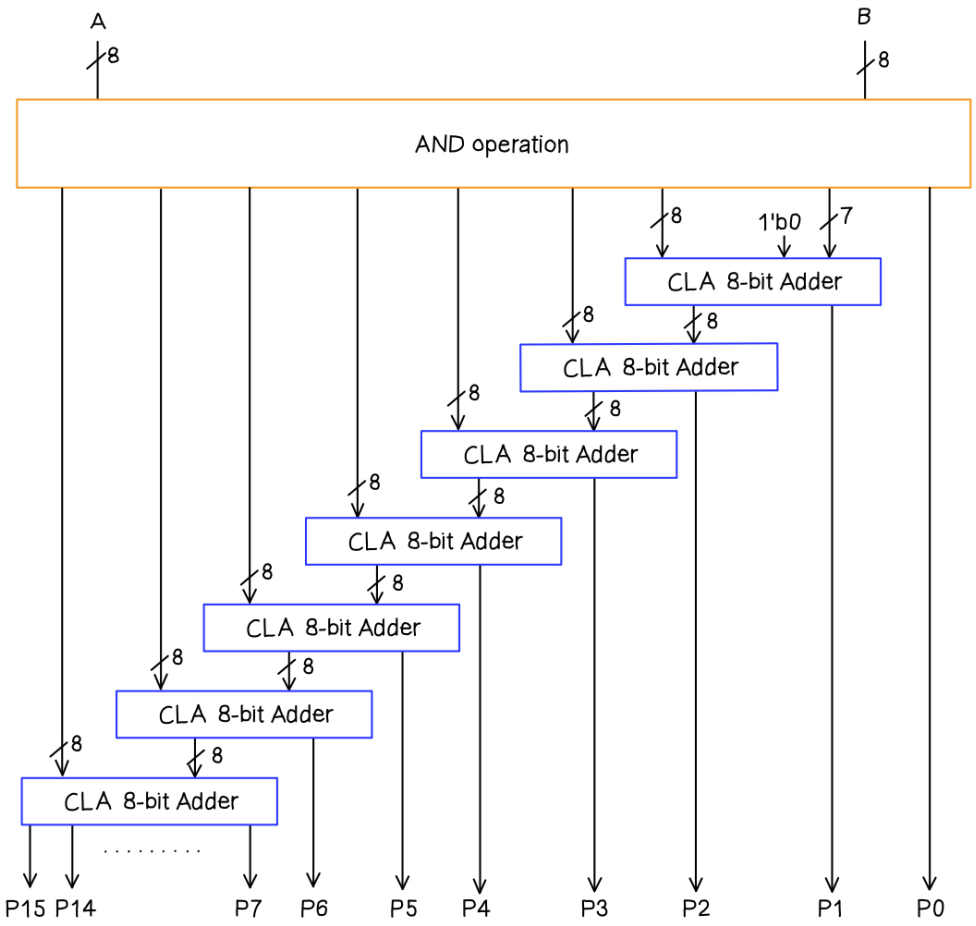
## *Prob C: Application*

1) **Design a 8*8-bit multiplier**
   a. **Using CLAs of Prob A**
   b. **Using PPAs of Prob B**
   c. **Design in gate level rather than behavioural modelling. (The module name should be Multiplier_CLA & Multiplier_PPA. And the file you include should be Multiplier_CLA.v & Multiplier_PPA.v)**
2) **Draw your hieratical structure.**
3) **You only need to upload your v-code to moodle, do not paste your code here.**
4) **Simulate your design with the testbench which includes all case of input.**
   **(Hint: The command for compiling is % vcs -R tb_Multiplier.v -full64)**
   **(Hint: The command for simulation is % vcs -R tb_Multiplier.v -debug_access+all -full64 +define+FSDB)**

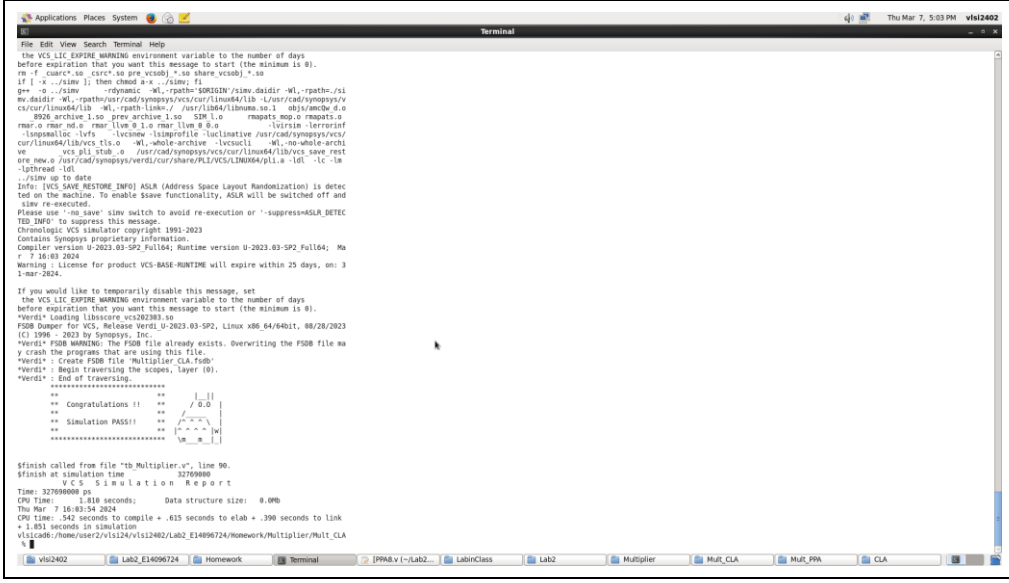
10) **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 2) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain 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)**

## Using CLAs Design a 8*8-bit multiplier

Draw your hieratical structure.




Your simulation result on the terminal.



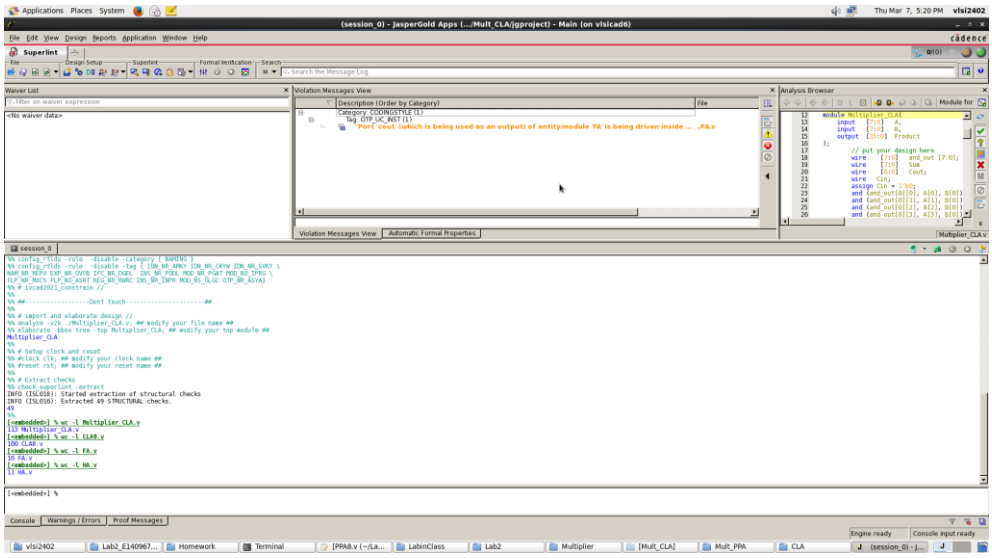
Your waveform :

Explanation of your waveform :

上面波形圖 error 一直都為 0，代表全部的測資皆正確。

下面波形圖中可以看到所有測資皆有執行乘法，並且乘法的結果皆與測資的答案相同。

Superlint screenshot and coverage



Coverage :

Multiplier_CLA.v : (1 - 0/113) * 100% = 100%

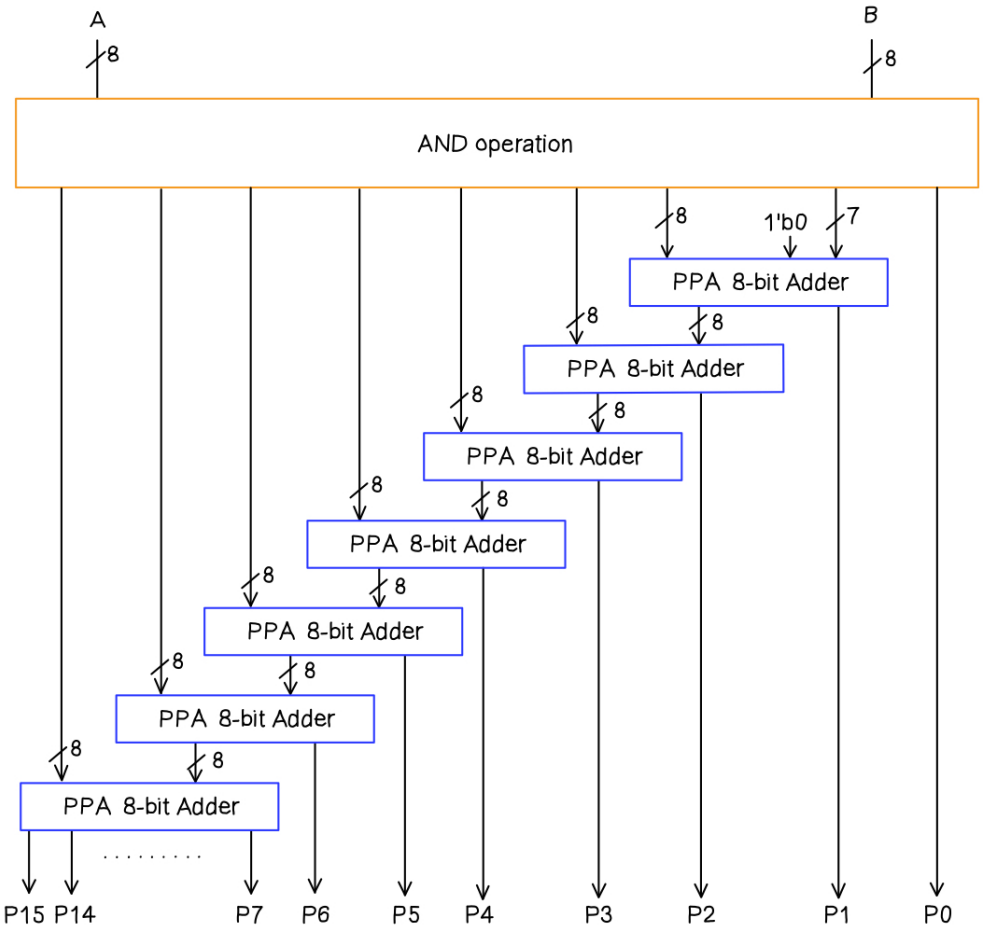
CLA8.v : (1 - 0/100) * 100% = 100%

FA.v : (1 - 1/16) * 100% = 93.75%

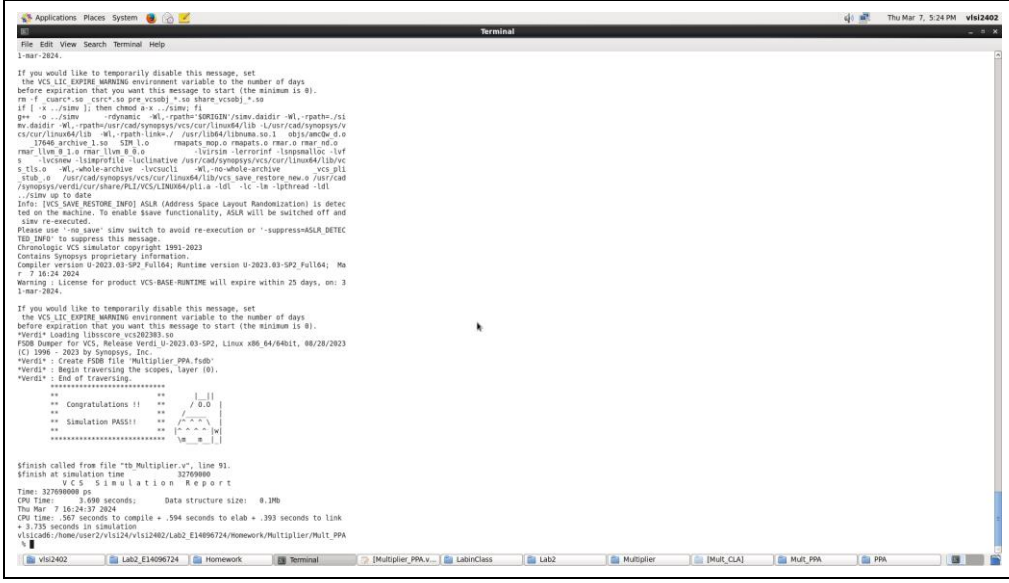
HA.v : (1 - 0/11) * 100% = 100%

## Using PPAs Design a 8*8-bit multiplier

Draw your hieratical structure.




Your simulation result on the terminal.



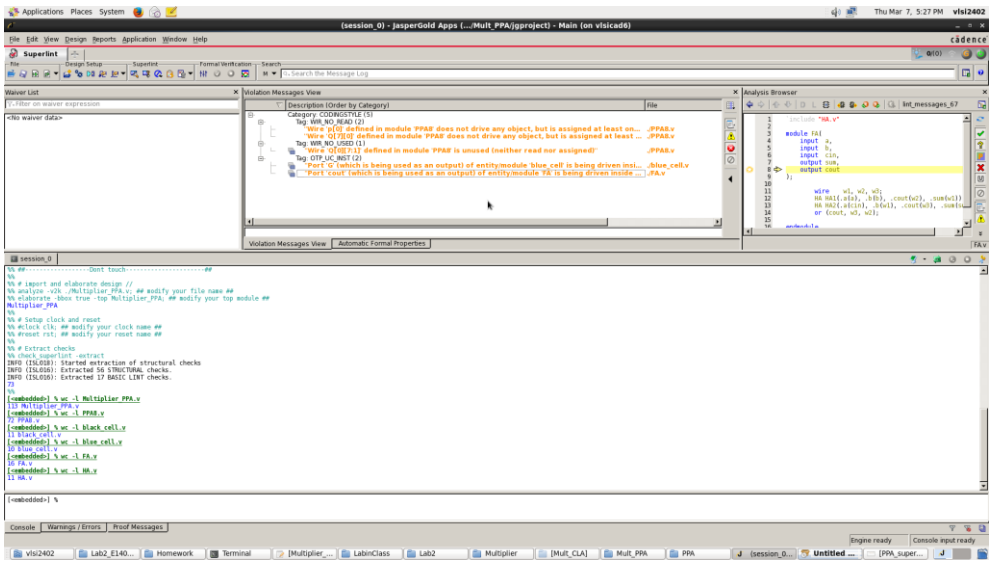
Your waveform :

Explanation of your waveform :

上面波形圖 error 一直都為 0，代表全部的測資皆正確。

下面波形圖中可以看到所有測資皆有執行乘法，並且乘法的結果皆與測資的答案相同。

Superlint screenshot and coverage



Coverage :

Multiplier_PPA.v : (1 - 0/113) * 100% = 100%

PPA8.v : (1 - 3/72) * 100% = 95.83%

black_cell : (1 - 0/11) * 100% = 100%

blue_cell : (1 - 1/10) * 100% = 90%

FA.v : (1 - 1/16) * 100% = 93.75%

HA.v : (1 - 0/11) * 100% = 100%

*Appendix A : Commands we will use to check your homework*

| Problem | | Commands |
|---|---|---|
| **ProbA** | Compile | % vcs -R tb_CLA32.v -full64 |
| | Simulate | % vcs -R tb_CLA32.v -debug_access+all -full64 +define+FSDB |
| **ProbB** | Compile | % vcs -R tb_PPA32.v -full64 |
| | Simulate | % vcs -R tb_PPA32.v -debug_access+all -full64 +define+FSDB |
| **ProbC** | Compile | % vcs -R tb_Multiplier.v -full64 |
| | Simulate | % vcs -R tb_Multiplier.v -debug_access+all -full64 +define+FSDB |