

# 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	
劉冠妤	E24116152	
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**

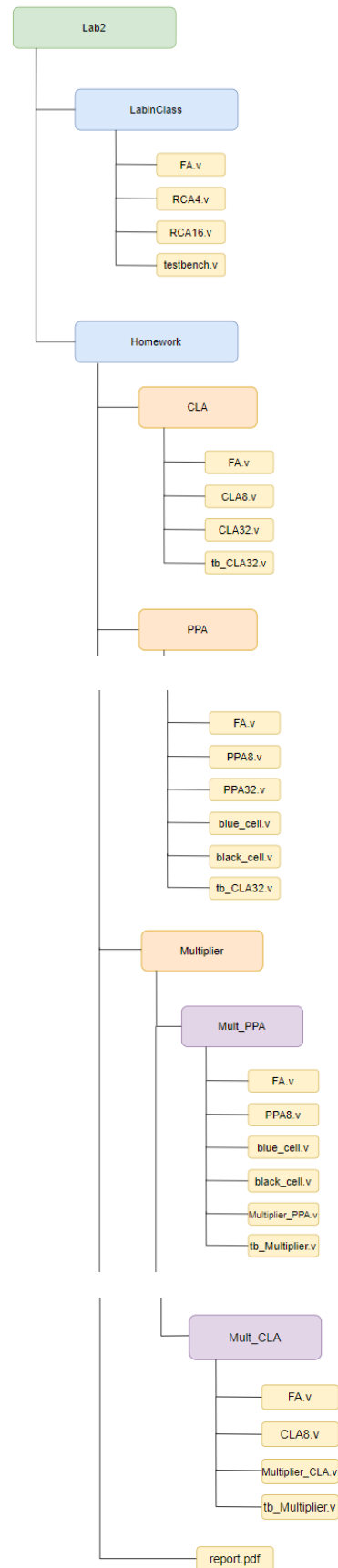


Fig.1 File hierarchy for Homework submission

## 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.

---

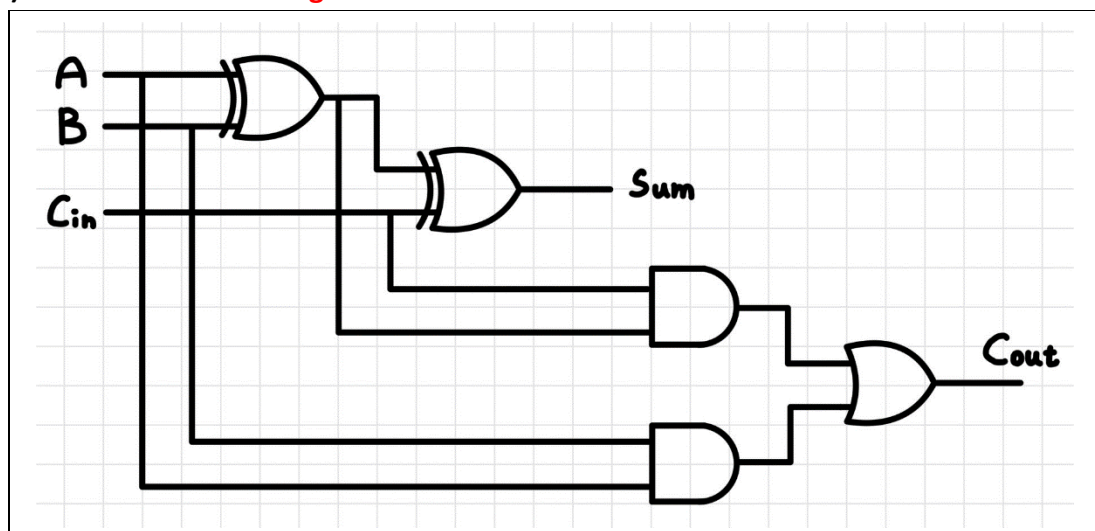
### *Prob A: Design Steps & Carry-Lookahead Adder*

---

- 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

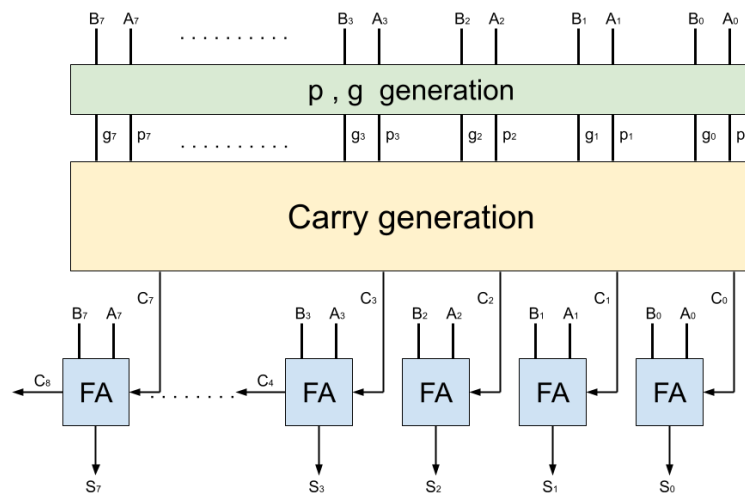
Inputs			Outputs	
A	B	C <sub>in</sub>	C <sub>out</sub>	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- 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 8<sup>th</sup> carry bit  $C_8$  (ex.  $C_2 = A_1B_1 + (A_1 + B_1)C_1 = g_1 + p_1C_1 = g_1 + p_1g_0 + p_1p_0C_0$ )

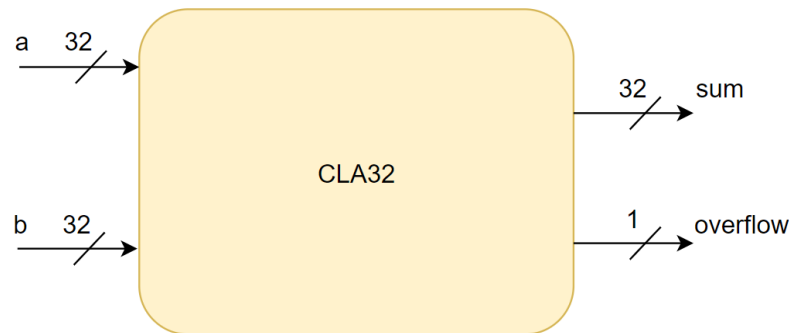


ex.  $C_2 = A_1B_1 + (A_1 + B_1)C_1 = g_1 + p_1C_1 = g_1 + p_1g_0 + p_1p_0C_0$ ,  $g_1 + p_1g_0 + p_1p_0C_0$  is the final result.

$$C_8 = g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 + p_7p_6p_5p_4p_3p_2g_1 + p_7p_6p_5p_4p_3p_2p_1g_0 + p_7p_6p_5p_4p_3p_2p_1p_0C_0$$

$$\begin{aligned} C_1 &= g_0 + p_0C_0 \\ C_2 &= g_1 + p_1g_0 + p_1p_0C_0 \\ C_3 &= g_2 + p_2g_1 + p_2p_1g_0 + p_2p_1p_0C_0 \\ &\vdots \\ C_n &= g_{n-1} + p_{n-1}g_{n-2} + p_{n-1}p_{n-2}g_{n-3} + \dots + (p_{n-1} \dots p_0C_0) \\ C_8 &= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 \\ &\quad + p_7p_6p_5p_4p_3g_2 + p_7p_6p_5p_4p_3p_2g_1 \\ &\quad + p_7p_6p_5p_4p_3p_2p_1g_0 + p_7p_6p_5p_4p_3p_2p_1p_0C_0 \end{aligned}$$

- 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

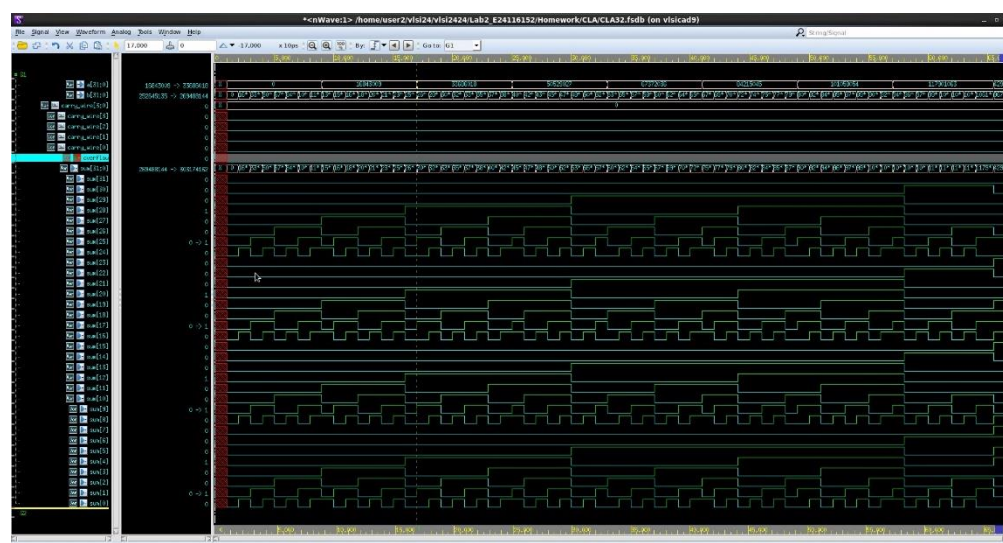
- 9) Simulate your design with the following test pattern in sample testbench.  
 (Hint: The command for compiling is **% vcs -R tb\_CLA.32v -full64**)  
 (Hint: The command for simulation is **%vcs -R tb\_CLA32.v -debug\_access+all -full64 +define+FSDB**)
- 10) Verify your design by comparing the simulation 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.

```
Terminal
File Edit View Search Terminal Help
*Verdi* Loading libsscore_vcs202303.so
FSDB Dumper for VCS, Release Verdi_U-2023.03-SP2, Linux x86_64/64bit, 08/28/2023
(C) 1996 - 2023 by Synopsys, Inc.
*Verdi* : Create FSDB file 'CLA32.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
*****
**                               **
** Congratulations !!           **
**                               **
** Simulation PASS!!           **
**                               **
*****
                               |_||
                               / 0.0 |
                               / ^ ^ \
                               | ^ ^ ^ |w|
                               \ m m _|_

$finish called from file "tb_CLA32.v", line 100.
$finish at simulation time          66000
          V C S   S i m u l a t i o n   R e p o r t
Time: 660000 ps
CPU Time:      0.660 seconds;      Data structure size:  0.0Mb
Fri Mar  8 22:32:33 2024
CPU time: .643 seconds to compile + .625 seconds to elab + .471 seconds to link
+ .705 seconds in simulation
vlsicad9:/home/user2/vlsi24/vlsi2424/Lab2_E24116152/Homework/CLA %
```

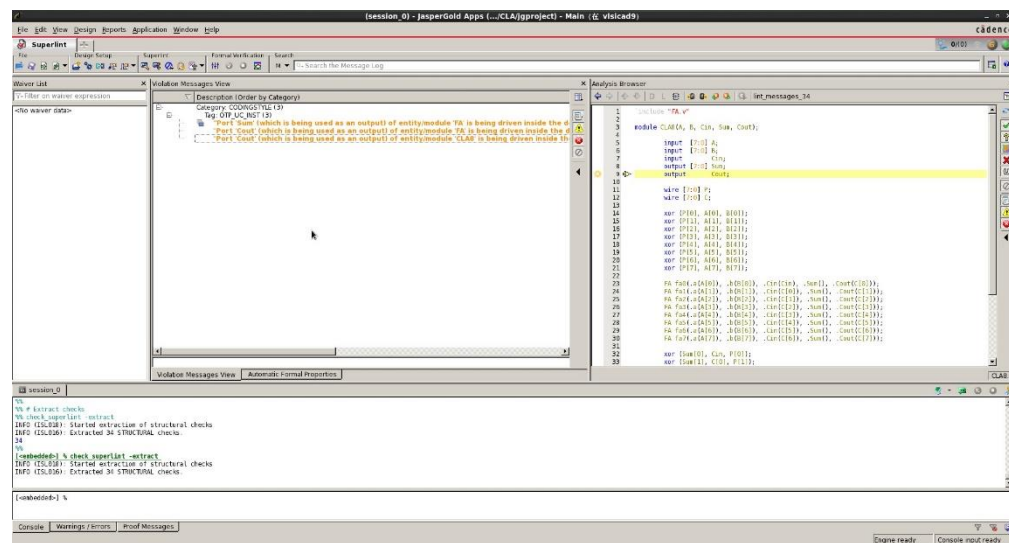
Your waveform :



Explanation of your waveform :

以圖中數據為例， $16843009_{10} = 00000001000000010000000100000001_2$ ，  
和  $252645135_{10} = 00001111000011110000111100001111_2$ ，兩者相加為  
 $269488144_{10} = 00010000000100000001000000010000_2$ ，而到下一個數據，  
 $33686018_{10} = 00000010000000010000000100000001_2$  與  
 $269488144_{10} = 00010000000100000001000000010000_2$  相加時，得到  
 $303174162_{10} = 00010010000100100001001000010010_2$ ，可以觀察到兩個  
結果只有第 2、10、18、26 位不同，可從 Sum 的波型中看出，且兩個加法都沒有出現 overflow。

## Superlint screenshot and coverage



Superlint coverage = 90.3 %



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

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

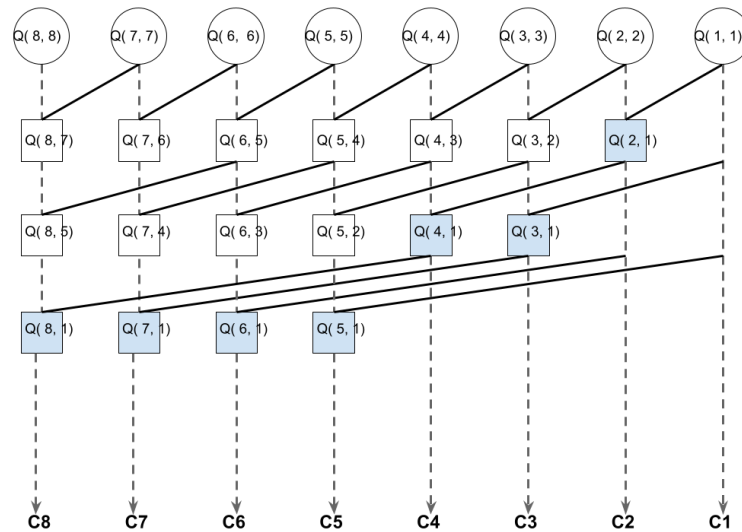
$$C_0 = 0$$

$$C_1 = A_1 B_1 + (A_1 \oplus B_1) C_0 = g_1 + p_1 C_0 = g_1 = Q(1, 1)$$

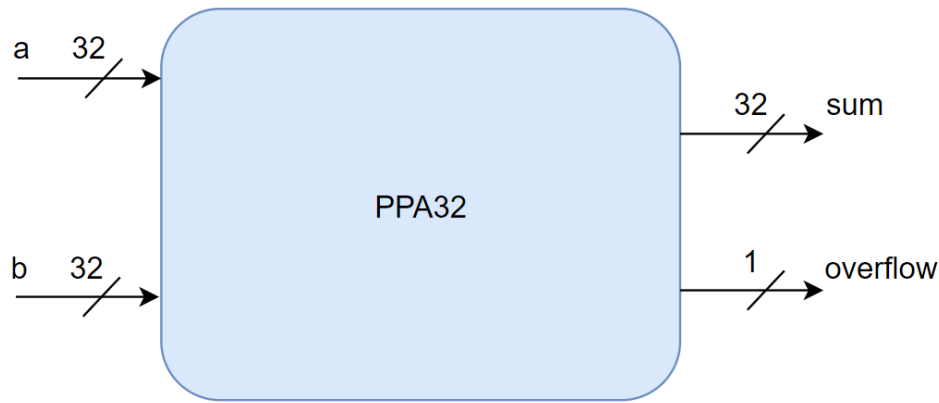
$$C_2 = A_2 B_2 + (A_2 \oplus B_2) C_1 = g_2 + p_2 C_1 = g_2 + p_2 g_1 = Q(2, 1) = p_2 Q(1, 1) + Q(2, 2)$$

$$C_3 = A_3 B_3 + (A_3 \oplus B_3) C_2 = g_3 + p_3 C_2 = g_3 + p_3 g_2 + p_3 p_2 g_1 = Q(3, 1) = p_3 p_2 Q(1, 1) + Q(3, 2)$$

$$C_4 = A_4 B_4 + (A_4 \oplus B_4) C_3 = g_4 + p_4 C_3 = g_4 + p_4 g_3 + p_4 p_3 g_2 + p_4 p_3 p_2 g_1 = Q(4, 1) = p_4 p_3 Q(2, 1) + Q(4, 3)$$



- 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.

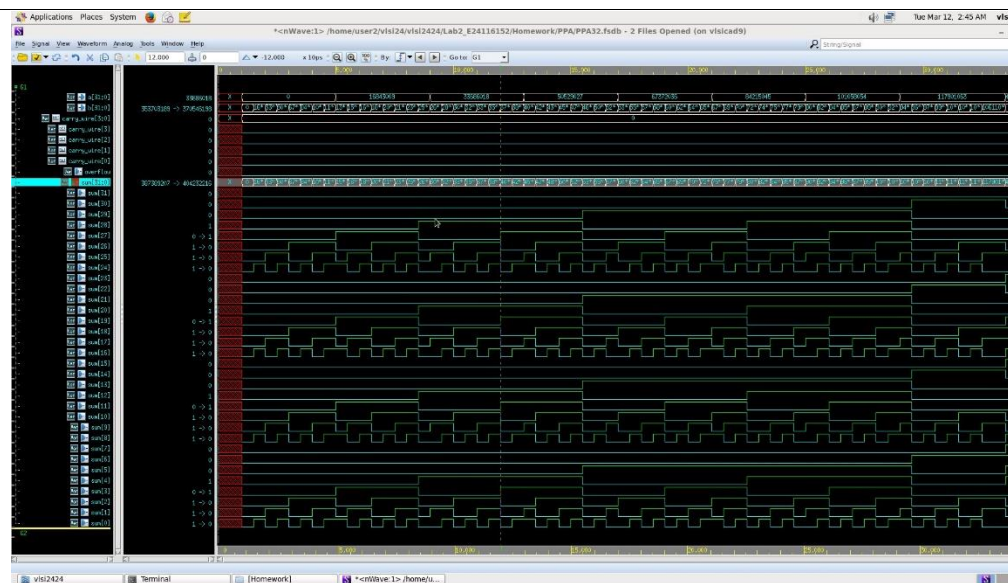
Your simulation result on the terminal.

```

Terminal
File Edit View Search Terminal Help
FSDB Dumper for VCS, Release Verdi_U-2023.03-SP2, Linux x86_64/64bit, 08/28/2023
(C) 1996 - 2023 by Synopsys, Inc.
*Verdi* FSDB WARNING: The FSDB file already exists. Overwriting the FSDB file may
crash the programs that are using this file.
*Verdi* : Create FSDB file 'PPA32.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
*****
**                                     **
**      Congratulations !!           **
**                                     **
**      Simulation PASS!!           **
**                                     **
**      \^ ^ ^ ^ \w|                 **
**      \m _ m _ |                  **
*****
$stop at time 34000 Scope: tb File: tb_PPA32.v Line: 101
uccli% finish
          V C S   S i m u l a t i o n   R e p o r t
Time: 340000 ps
CPU Time:          0.660 seconds;      Data structure size:   0.0Mb
Fri Mar 8 22:40:40 2024
CPU time: .708 seconds in simulation
vlsicad9:/home/user2/vlsi24/vlsi2424/Lab2_E24116152/Homework/PPA %

```

Your waveform :



Explanation of your waveform :

從波型圖可以觀察到 PPA 和 CLA 的 Sum 部分結果波型一致。但比較兩者的執行速度可以看出，PPA 的計算速度較快。

The screenshot displays the JasperGold IDE interface. The top menu bar includes File, Edit, View, Design, Reports, Application, Window, and Help. The main window is titled '(session\_0) - JasperGold Apps (...\\CLA\\jproject) - Main (z. Vaiscad)'. Below the menu bar, there are tabs for 'Superlint', 'Design Setup', 'Scripting', 'Format Indentation', and 'Search'. A search bar is present with the text 'Find the Message Log'. The 'Violation Messages View' is active, showing a list of violations under the category 'Category CORDINATE (1)'. The first violation is 'First Count (which is being used as an output of node's node, there is being drawn inside to'. The 'Analysis Browser' is also visible, showing a list of analysis results. The bottom status bar indicates 'Engine ready' and 'Console input'.

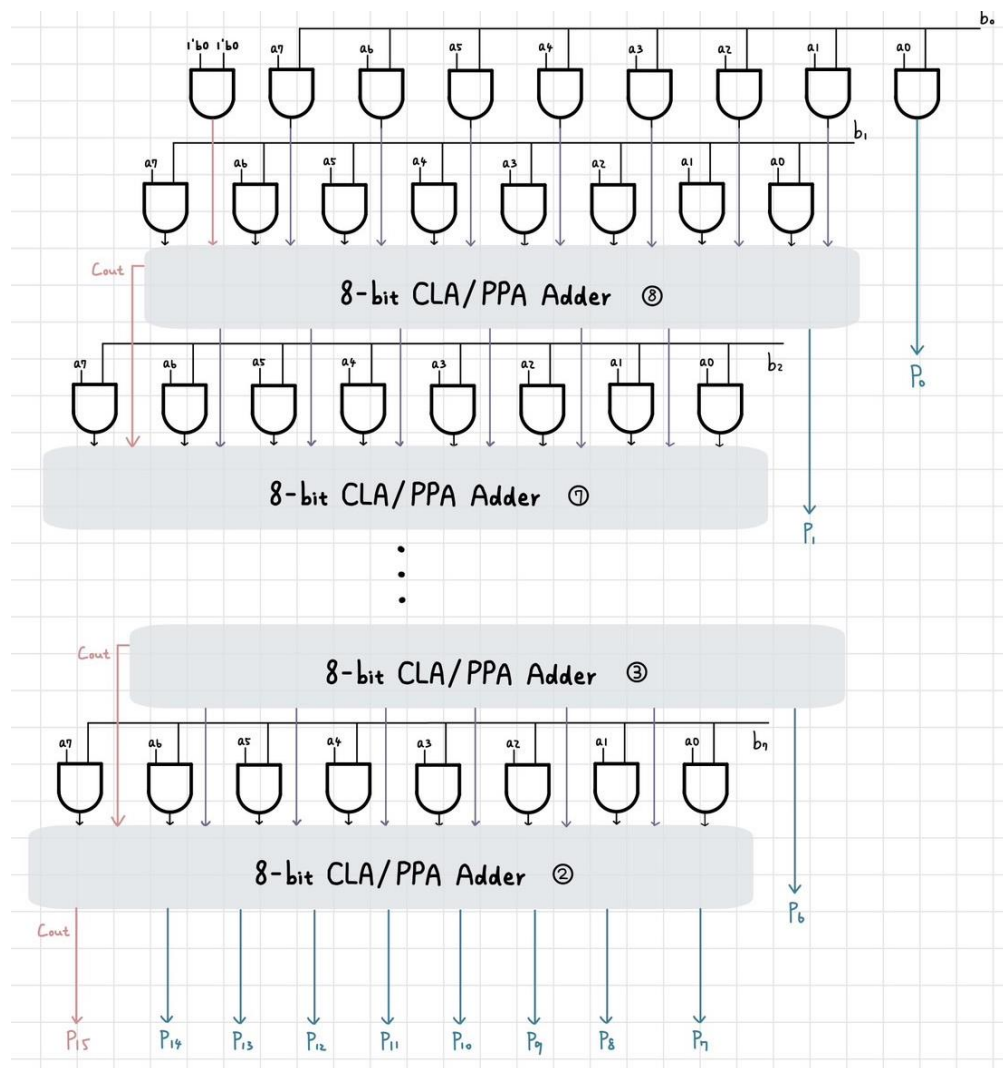
---

*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**)

Draw your hieratical structure.



Your simulation result on the terminal.

```
Terminal
File Edit View Search Terminal Help
FSDB Dumper for VCS, Release Verdi_U-2023.03-SP2, Linux x86_64/64bit, 08/28/2023
(C) 1996 - 2023 by Synopsys, Inc.
*Verdi* : Create FSDB file 'Multiplier_CLA.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
*****
**                                     **
** Congratulations !!                **
**                                     **
** Simulation PASS!!                 **
**                                     **
*****
                                     |__|
                                     / 0.0 |
                                     /-----|
                                     / ^ ^ ^ \
                                     | ^ ^ ^ ^ |w|
                                     \m__m__|_

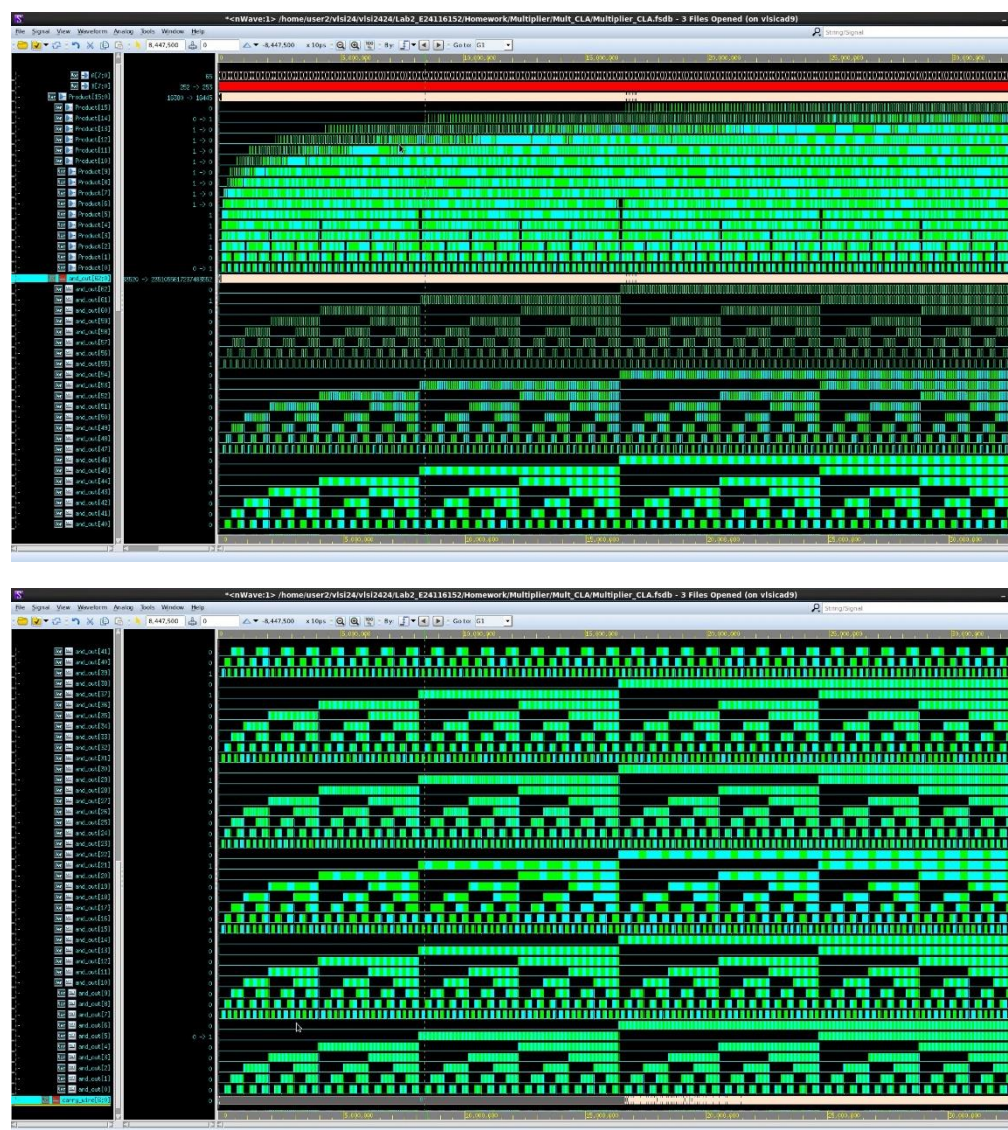
$finish called from file "tb_Multiplier.v", line 90.
$finish at simulation time          32769000
      V C S   S i m u l a t i o n   R e p o r t
Time: 327690000 ps
CPU Time:      2.750 seconds;      Data structure size:  0.0Mb
Fri Mar  8 22:42:13 2024
CPU time: .656 seconds to compile + .681 seconds to elab + .440 seconds to link
+ 2.801 seconds in simulation
vlsicad9:/home/user2/vlsi24/vlsi2424/Lab2_E24116152/Homework/Multiplier/Mult_CLA
% 
```

```
Terminal
File Edit View Search Terminal Help
FSDB Dumper for VCS, Release Verdi_U-2023.03-SP2, Linux x86_64/64bit, 08/28/2023
(C) 1996 - 2023 by Synopsys, Inc.
*Verdi* : Create FSDB file 'Multiplier_PPA.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
*****
**                                     **
** Congratulations !!                **
**                                     **
** Simulation PASS!!                 **
**                                     **
*****
                                     |__|
                                     / 0.0 |
                                     /-----|
                                     / ^ ^ ^ \
                                     | ^ ^ ^ ^ |w|
                                     \m__m__|_

$finish called from file "tb_Multiplier.v", line 91.
$finish at simulation time          32769000
      V C S   S i m u l a t i o n   R e p o r t
Time: 327690000 ps
CPU Time:      4.600 seconds;      Data structure size:  0.0Mb
Fri Mar  8 22:45:51 2024
CPU time: .668 seconds to compile + .660 seconds to elab + .442 seconds to link
+ 4.667 seconds in simulation
vlsicad9:/home/user2/vlsi24/vlsi2424/Lab2_E24116152/Homework/Multiplier/Mult_PPA
% 
```



Your waveform :

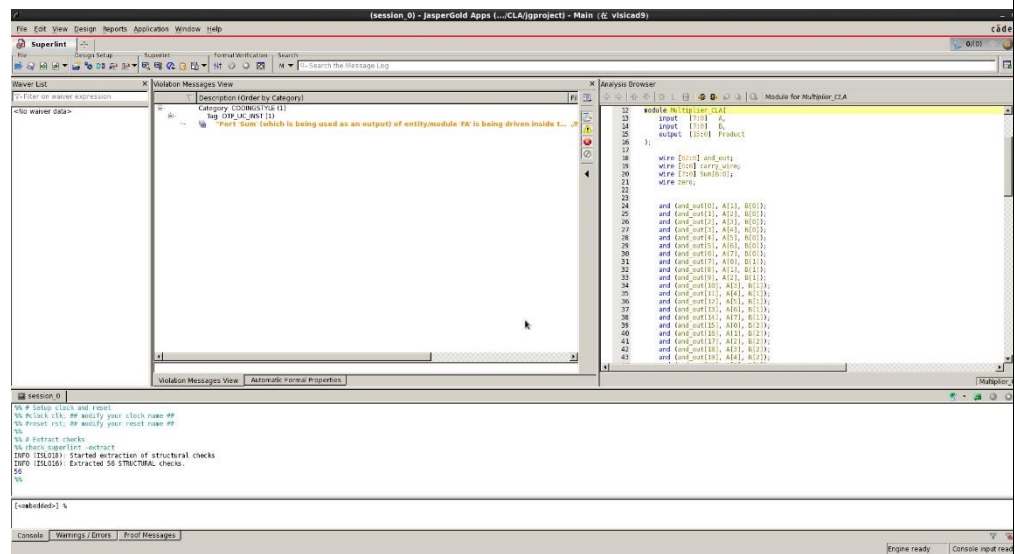


Explanation of your waveform :

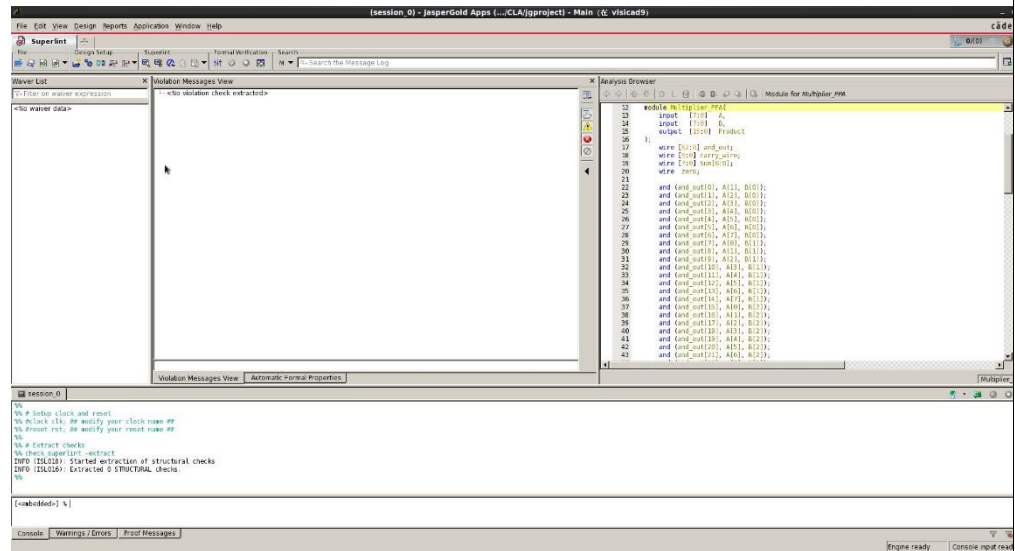
從波型圖可以看出由 4 個 8-bit 加法器組成乘法器的架構，而且是 unsigned，以圖中數據為例， $65_{10} = 01000001_2$  和  $252_{10} = 11111100_2$  相乘得到  $16380_{10} = 0011111111111100_2$ ，可以驗證。



## Superlint screenshot and coverage



Superlint coverage = 99.2 %



Superlint coverage = 100%

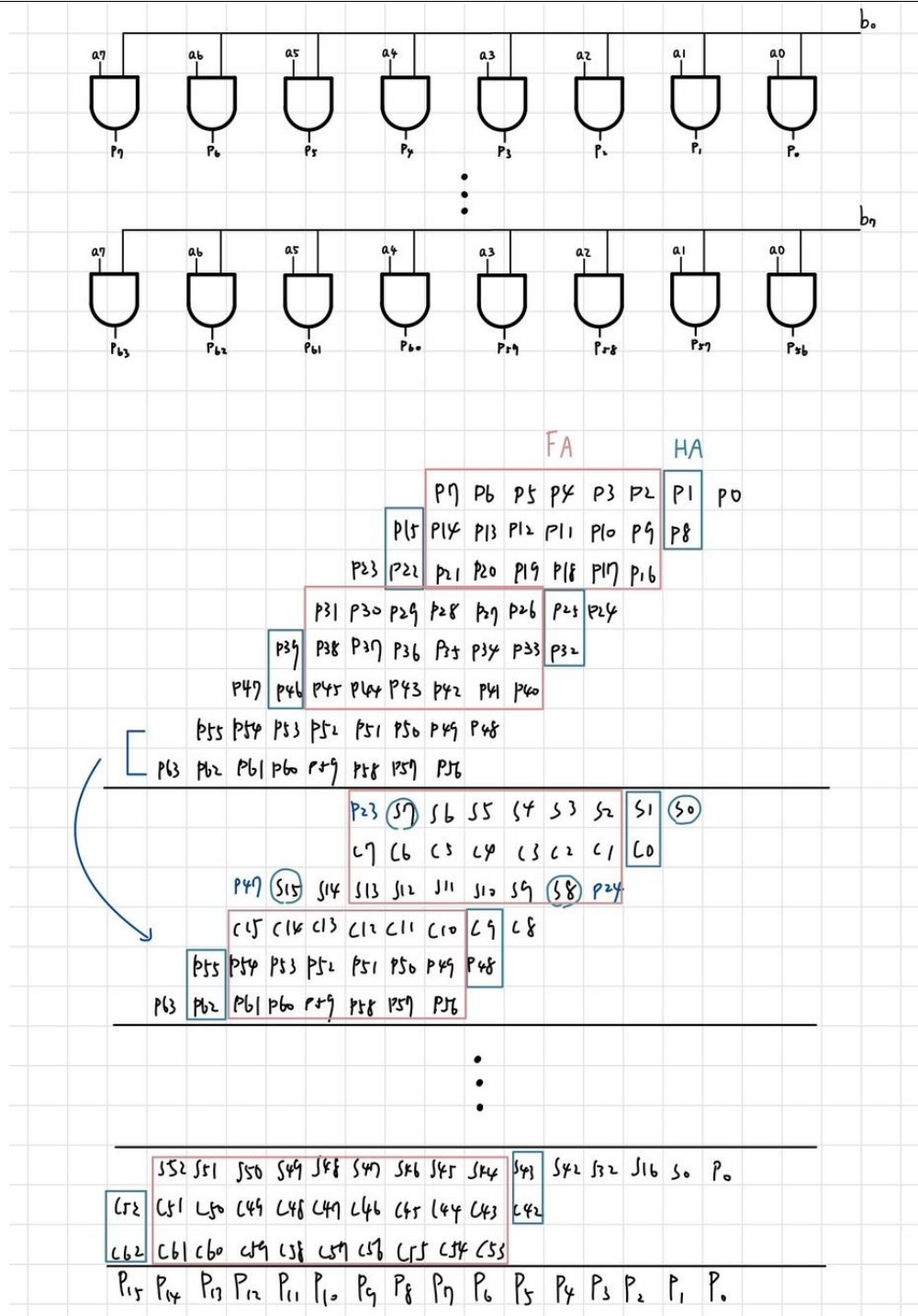
## Different multiplier architectures - Wallace Tree Multiplier

使用一系列全加器和半加器來分階段對部分乘積求和，大致分為二步驟：

1. 將兩個輸入的 8 位元數字的每一個位元兩兩相乘。
2. 通過全加器和半加器的層數將部分乘積的數量減少到兩個。

圖中最後一區的 C53~C62 為 S43 和 C42 到 S52 和 C51 的 Cout。

hierarchical structure



Your simulation result on the terminal.

```

140.116.156.6 - PuTTY
FSDB Dumper for VCS, Release Verdi_U-2023.03-SP2, Linux x86_64/64bit, 08/28/2023
(C) 1996 - 2023 by Synopsys, Inc.
*Verdi* : Create FSDB file 'Multiplier_CLA.fsd'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
*****
**                                     **
**  Congratulations !!               **
**                                     **
**  Simulation PASS!!               **
**                                     **
*****
                                     |_|
                                     / O.O |
                                     /-----|
                                     / ^ ^ ^ \ |
** | ^ ^ ^ ^ |w|
** \m _m _l_|
*****

$finish called from file "tb_Multiplier.v", line 90.
$finish at simulation time      32769000
      V C S   S i m u l a t i o n   R e p o r t
Time: 327690000 ps
CPU Time:      1.490 seconds;      Data structure size:  0.0Mb
Tue Mar 12 17:46:32 2024
CPU time: .641 seconds to compile + .667 seconds to elab + .474 seconds to link
+ 1.536 seconds in simulation
vlsicad9:/home/user2/vlsi24/vlsi2424/Lab2_E24116152/Homework/Multiplier/Mult_WTM
%

```

### 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