

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**

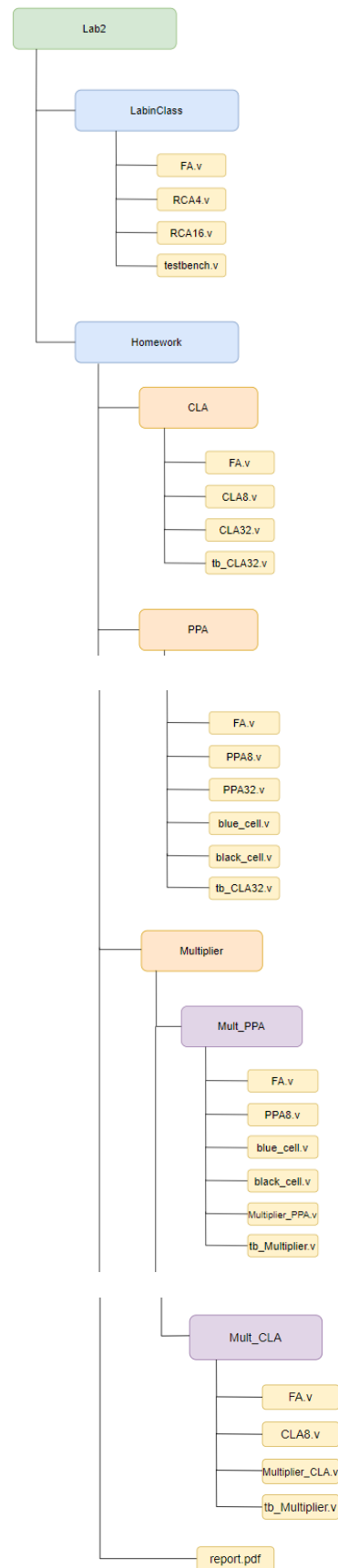


Fig.1 File hierarchy for Homework submission

Objectives:

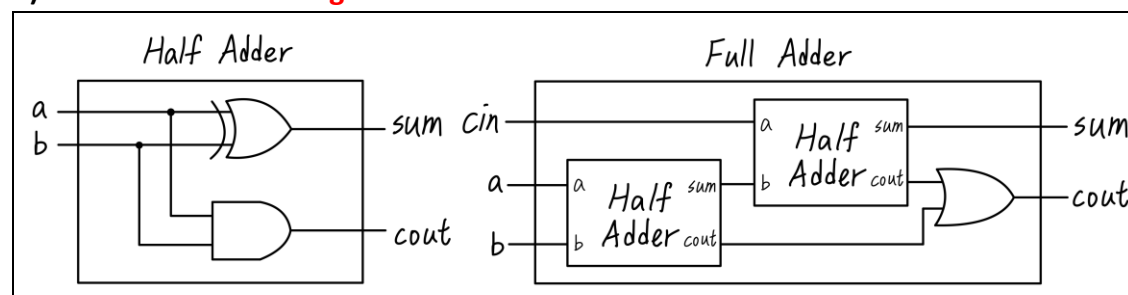
Help students get familiar with the CAD tools (VCS & Verdi) for digital logic design. Introduce different adder architectures and the algorithms behind them. Please go through the hands-on exercise step-by-step.

Prob A: Design Steps & Carry-Lookahead Adder

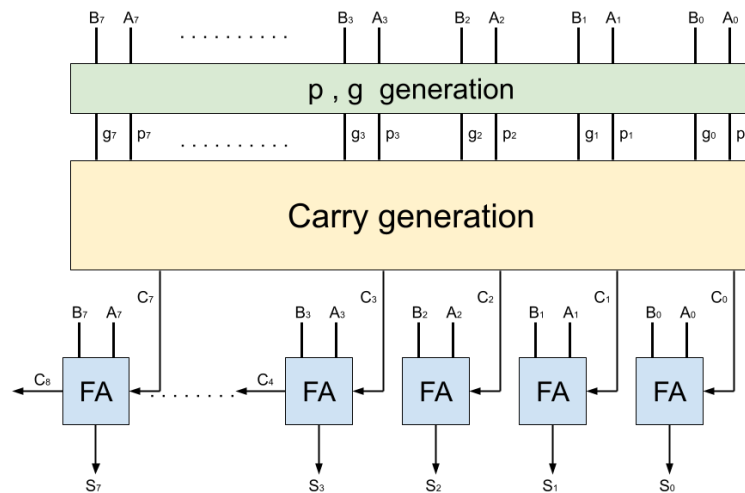
- 1) An adder is a digital circuit that performs addition of number. Please design a full adder in gate level.
- 2) The truth table of full adder

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**.



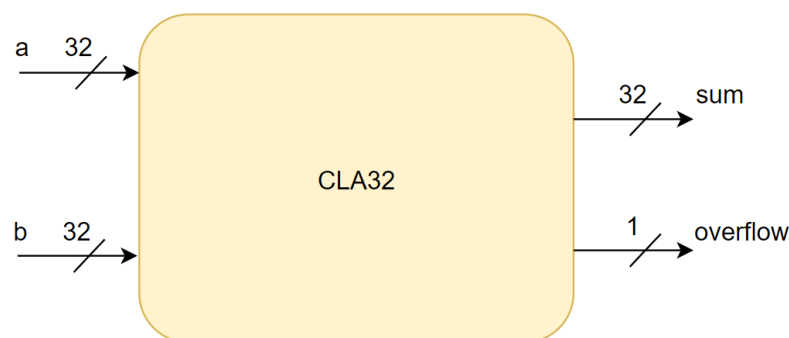
- 4) Design a full adder in **Structural coding** (The module name should be **FA**. And the file you include should be **FA.v**)
- 5) Carry-lookahead-adder uses carry generator to alleviate the lengthy carry propagation procedure in ripple-carry-adder.
- 6) Derive the 8th carry bit C₈(ex. $C_2 = A_1B_1 + (A_1 + B_1)C_1 = g_1 + p_1C_1 = g_1 + p_1g_0 + p_1p_0C_0$)



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.

$$\begin{aligned}
 C_8 &= A_7B_7 + (A_7 + B_7)C_7 = g_7 + p_7C_7 = g_7 + p_7g_6 + p_7p_6C_6 = g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5C_5 \\
 &= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4C_4 \\
 &= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3C_3 \\
 &= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 + p_7p_6p_5p_4p_3p_2C_2 \\
 &= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 + p_7p_6p_5p_4p_3p_2g_1 + \\
 &\quad p_7p_6p_5p_4p_3p_2p_1C_1 \\
 &= g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 + p_7p_6p_5p_4p_3p_2g_1 + \\
 &\quad p_7p_6p_5p_4p_3p_2p_1g_0 + p_7p_6p_5p_4p_3p_2p_1p_0C_0
 \end{aligned}$$

- 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.

```
Applications Places System
Thu Mar 7, 3:38 PM vlsd202

File Edit View Search Terminal Help
Warning : License for product VCS-BASE-COMPIL will expire within 25 days, on: 3
1-Mar-2024.

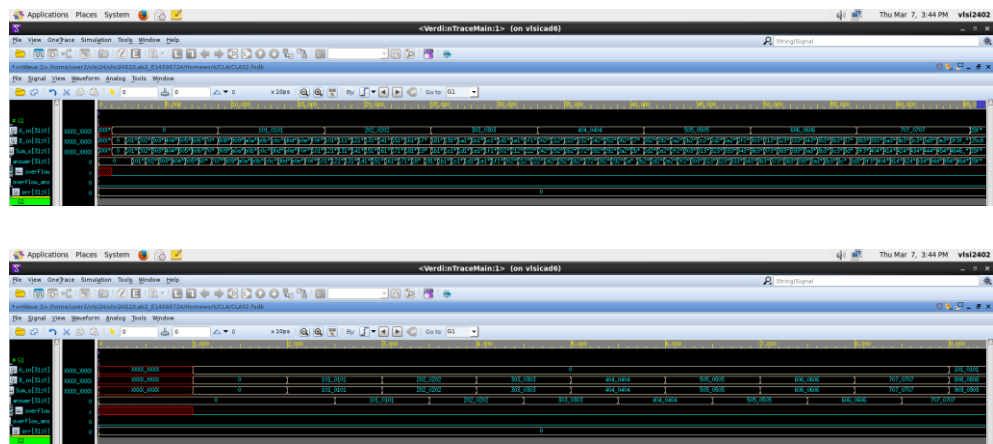
If you would like to temporarily disable this message, set
the VCS_LIC_EXPIRE_WARNING environment variable to the number of days
before expiration that you want this message to start (the minimum is 0).
m -f - source-as cur-as pre vcslib > no share vcslib.v && no
if [ x _./simv ] ; then chmod a-x _./simv ; fi
rm -o ./simv -r dynamic -Wl,-path=/usr/lib64/ld.so.1 -Wl,-path=/usr
lib64/ld -Wl,-path=/usr/cad/synopsys/vcs/cir/Linux4/ld -Wl,-path=/usr/cad/synopsys/vc
scir/Linux4/ld -Wl,-path=Linux4 /usr/lib64/libc.so.1 obj/mchx.d.o
_36236 archive.1.rho SIM 1.0 rmmap.s map.o rmap.o mmar.mn.d.p
mmar.lib 9.1.o mmar.lib 0.0.o -lcvsim -terrfmt -lspallacirc -lcf
s -lcvsim -lslaprefile -liclinateur /usr/cad/synopsys/vcs/cir/Linux4/libvc
s.tls.o -Wl,--whole-archive -lcvsutil -Wl,--no-whole-archive -vcs.pli
libab /usr/cad/synopsys/vcs/cir/Linux4/libvcs save restore new.a /usr/cad
/synopsys/verdi/Curr/share/PLI/VCS/LINUX64/pli.a -ldl -lc -ln -lpthread -ld
./simv up to date
Info: [VCS SAVE RESTORE INFO] ASLR (Address Space Layout Randomization) is dete
cted on the machine. To enable Save functionality, ASLR will be switched off and
simv re-executed.
Please use '-no save' simv switch to avoid re-execution or '-suppress-ASLR_DETEC
TED_INFO' to suppress this message.
Chronologic VCS simulator copyright 1993-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Pull64; Runtime version U-2023.03-SP2_Pull64; Ma
r 7 14:38 2024
Warning : License for product VCS-BASE-RUNTIME will expire within 25 days, on: 3
1-Mar-2024.

If you would like to temporarily disable this message, set
the VCS_LIC_EXPIRE_WARNING environment variable to the number of days
before expiration that you want this message to start (the minimum is 0).
Verdi Loading Libraries vcs202383.so
FSDB Dumper for VCS, Release Verdi U-2023.03-SP2, Linux x86_64/64bit, 08/28/2023
(C) 1998 - 2023 by Synopsys, Inc.
Verdi+ Create FSDB file "CLA32.fdb"
Verdi+ Begin traversing the scopes, layer (0).
Verdi+ End of Traversing.

*****
**                               / | \
**   Congratulations !!         / 0  \
**                               \|___\
**   Simulation PASS!!          | + + + |
**                               \__ _ _ /
*****

$finish called from file "db_CLA32.v", line 381.
$finish at simulation time                                66080
VCS Simulation Report
Time: 66080 ps
CPU Time:    0.680 seconds;      Data structure size:    0.0PB
Thu Mar 7 14:39:13 2024
CPU time: 512 seconds to compile + .573 seconds to elab + .382 seconds to link +
.837 seconds in simulation
vcsidcom:/home/user2/vlsi24/vlsi2402/Lab02/E4096724/homework/CLAV1A
```

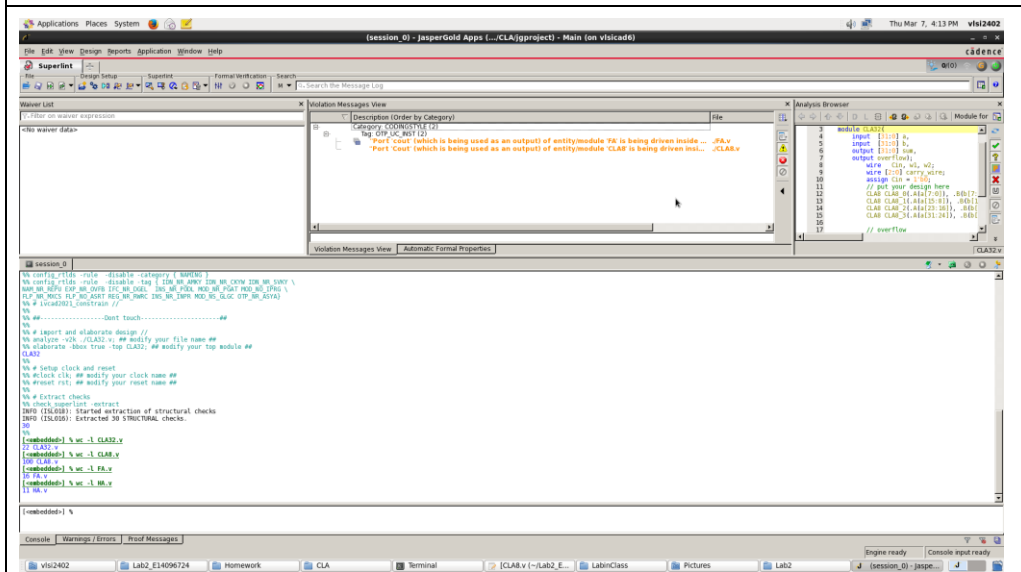
Your waveform :



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:

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

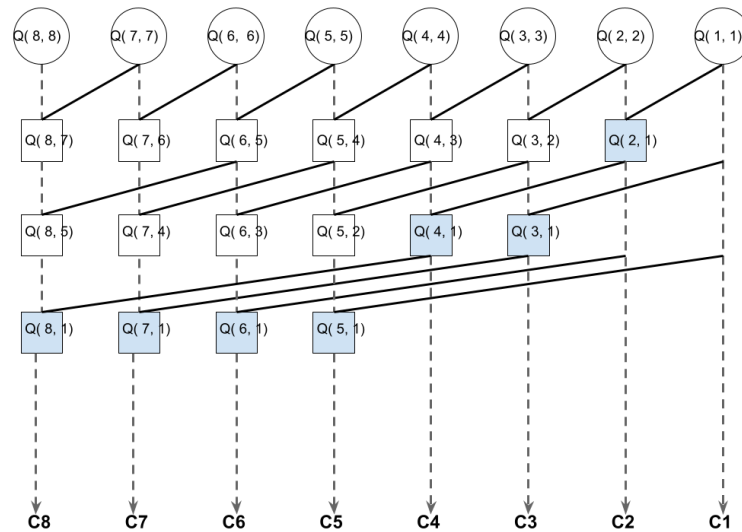
$$C_0 = 0$$

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

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

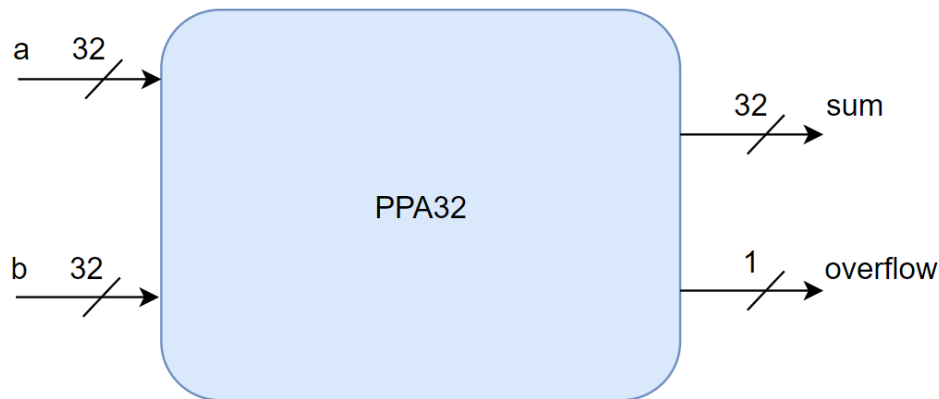
$$C_3 = A_3 B_3 + (A_3 \oplus B_3) C_2 = g_3 + p_3 C_2 = g_3 + p_3 g_2 + p_3 p_2 g_1 = Q(3, 1) = p_3 p_2 Q(1, 1) + Q(3, 2)$$

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



- 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.

[illegible]

The top screenshot shows the initial Verilog code for a 4-bit ripple-carry adder. The carry-in is set to 0. The code defines the carry chain logic for each bit position from 0 to 3.

```

`timescale 1ns/1ps
module adder4bit (
    input [3:0] a,
    input [3:0] b,
    output [3:0] sum,
    output carryout
);
    carry0 = 0;
    for (i = 0; i < 4; i = i + 1)
        carry[i] = a[i] + b[i] + carry[i-1];
    carryout = carry[3];
    sum = carry;
endmodule

```

The bottom screenshot shows the modified code with a carry-in input 'cin'. The carry chain logic is updated to include the carry-in for the first bit position.

```

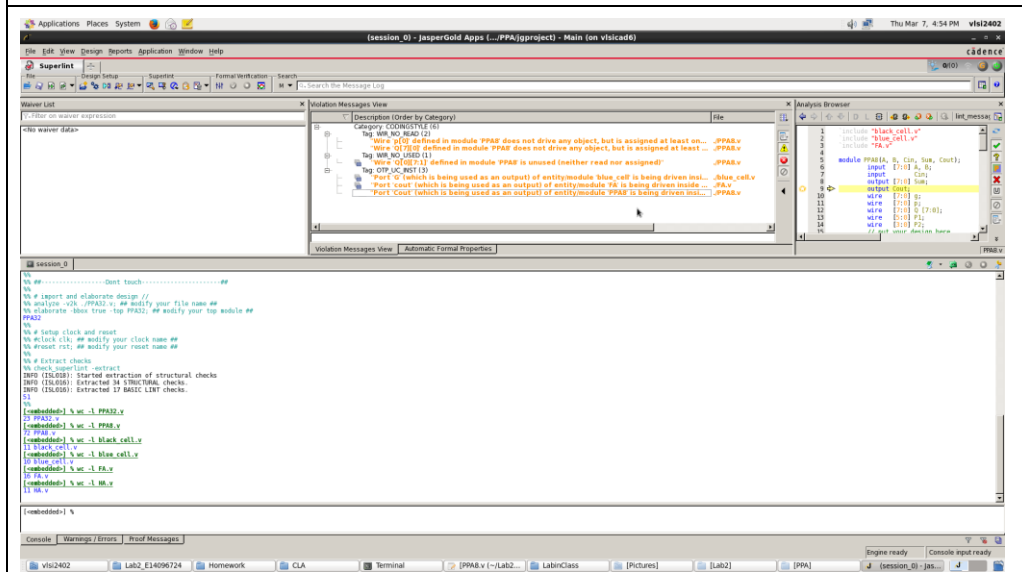
`timescale 1ns/1ps
module adder4bit (
    input [3:0] a,
    input [3:0] b,
    input cin,
    output [3:0] sum,
    output carryout
);
    carry0 = cin;
    for (i = 0; i < 4; i = i + 1)
        carry[i] = a[i] + b[i] + carry[i-1];
    carryout = carry[3];
    sum = carry;
endmodule

```

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 : $(1 - 1/16) * 100\% = 93.75\%$

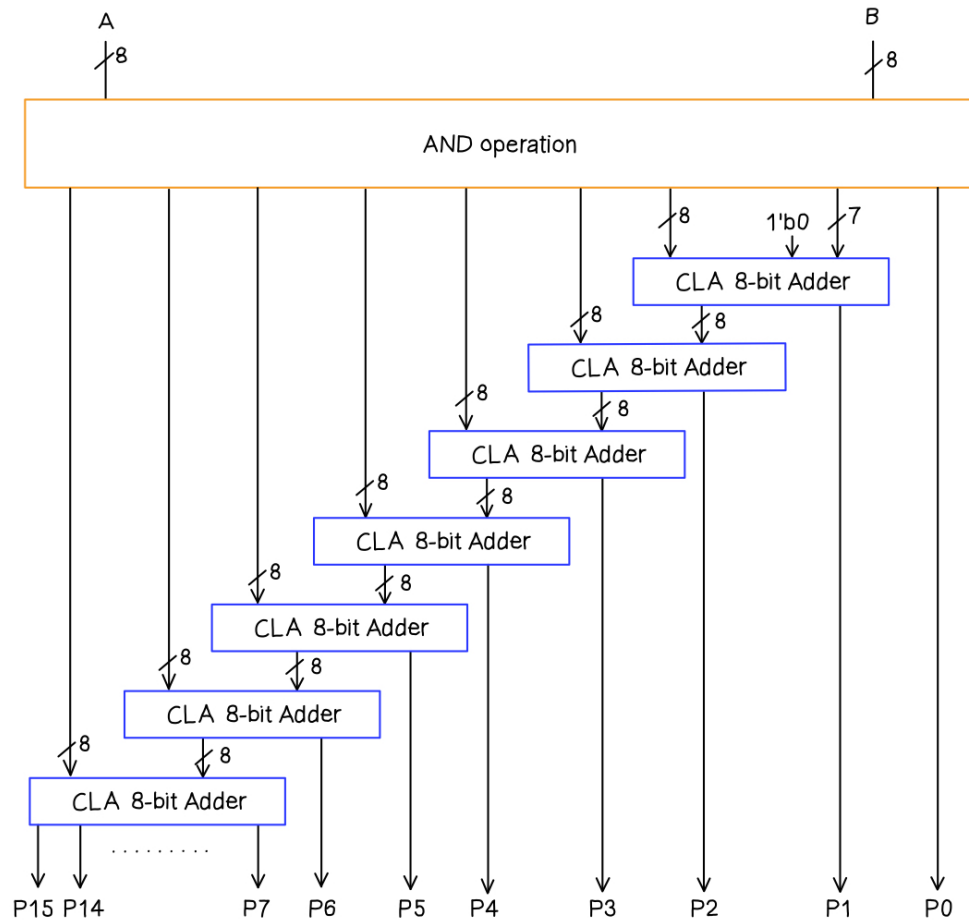
HA.v : $(1 - 0/11) * 100\% = 100\%$

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.

```
File Edit View Search Terminal Help
The VCS LIC_EXPIRE WARNING environment variable to the number of days
before expiration that you want this message to start (the minimum is 0).
rm -f ./csrcrc.so csrcrc.so pre vcslib) *.so share vcslib) *.so
if [ x . /sim &#x26; fi
prev o ./sim -dynamic WL -rpath=$ORIGIN/$sim.dir $dir WL -rpath=/si
my.dir $WL -rpath=/usr/cad/synopsys/vcs/curlinux64/lib -L/usr/cad/synopsys/v
cscurlinux64/lib WL -rpath=Linux:/usr/lib64/libnss.so.1 obj/mwcb.o a
BRO2 archive 1:so_prev archive 1:so SIM 1:lo rnapatt sop.o rnapatt.o
rmap a rmap.mt.o rmap Linux 1:o rmap Linux 8:0 lvirsize -lerrorinf
-lsynapslic -lvfs -lcvsim -lsimprofile -luclnative /usr/cad/synopsys/vcs
cur/linux64/libvcs lib_o WL-whole-archive -lcvccli WL-no-whole-archiv
le vcs pli stub.o ./usr/cad/synopsys/vcs/curlinux64/lib/vcs save rest
are new.o /usr/cad/synopsys/verdi/curl/share/PLI/VCS/Linux64/pli.a -ldl -lc -lm
-lpthread -ldl
./sim up to date
Note: [VCS SAVE RESTORE INFO] ASLR (Address Space Layout Randomization) is detec
ted on the machine. To enable Ssave functionality, ASLR will be switched off and
ASLR re-executed.
Please use '-no_save' sim switch to avoid re-execution or '--suppress=ASLR_DETEC
TED.INFO' to suppress this message.
Chromicle VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2 Pull64; Runtime version U-2023.03-SP2 Pull64; Ma
r 7 16:09 2024
Warning: license for product VCS-BASE-HUNTME will expire within 25 days, om: 3
1-mar-2024.

If you would like to temporarily disable this message, set
the VCS LIC_EXPIRE WARNING environment variable to the number of days
before expiration that you want this message to start (the minimum is 0).
Verlet Loading Libscore vcs202383.so
FSDM Dumper for VCS Release VerDI-U-2023.03-SP2, Linux x86_64/64bit, 08/28/2023
(C) 1996 - 2023 by Synopsys, Inc.
Verlet FSDM WARNING: The FSDM file already exists. Overwriting the FSDM file ma
y crash the programs that are using this file.
Verlet Create FSDM File Multiplier.CLA.Fsdm
Verlet Begin Traversing the scopes Layer (0).
Verlet End of traversing.
*****
** Congratulations !! ** |   |
**                    ** | O |
** simulation PASS!! **  /---\
**                    ** | ^--^|
**                    ** | ^--^|
**                    ** | ^--^|
**                    ** | ^--^|
***** | _ _ |

$finish called from user "tb.Multiplier.v", line 90.
32769800
Time: 327698000 ps
CPU Time: 1.818 seconds; Data structure size: 0.0MB
Thu Mar 7 16:03:54 2024
CPU time : 542 seconds to compile + 615 seconds to elab + .390 seconds to link
+ 1.051 seconds in simulation
vlscmdcad/home/user/vslz12/v140/F1400672/Lab2_E1409672A/Homework/Multiplier/TML_CUA
```

The top screenshot shows a logic analyzer interface with the title bar "Applications Places System" and the file path "c:\nWave1> \home\user2\visi24\visi2402\Lab2_E14096724\Homework\Multiplier\Mult_CLA_Multiplier_CLA.fdbd (on visicad6)". The signal list on the left includes "a_in[7:0]", "b_in[7:0]", "c_in[7:0]", and "sum[10:0]". The waveform shows "a_in[7:0]" as a constant high signal, "b_in[7:0]" as a constant low signal, "c_in[7:0]" as a constant high signal, and "sum[10:0]" as a constant low signal. A red highlight is visible on the "sum[10:0]" signal.

The bottom screenshot shows the same logic analyzer interface with the title bar "Applications Places System" and the file path "c:\nWave1> \home\user2\visi24\visi2402\Lab2_E14096724\Homework\Multiplier\Mult_CLA_Multiplier_CLA.fdbd (on visicad6)". The signal list on the left includes "a_in[7:0]", "b_in[7:0]", "c_in[7:0]", and "sum[10:0]". The waveform shows "a_in[7:0]" as a constant high signal, "b_in[7:0]" as a constant low signal, "c_in[7:0]" as a constant high signal, and "sum[10:0]" as a constant low signal. A red highlight is visible on the "sum[10:0]" signal, and a green highlight is visible on the "a_in[7:0]" signal.

上面波形圖 error 一直都為 0，代表全部的測資皆正確。

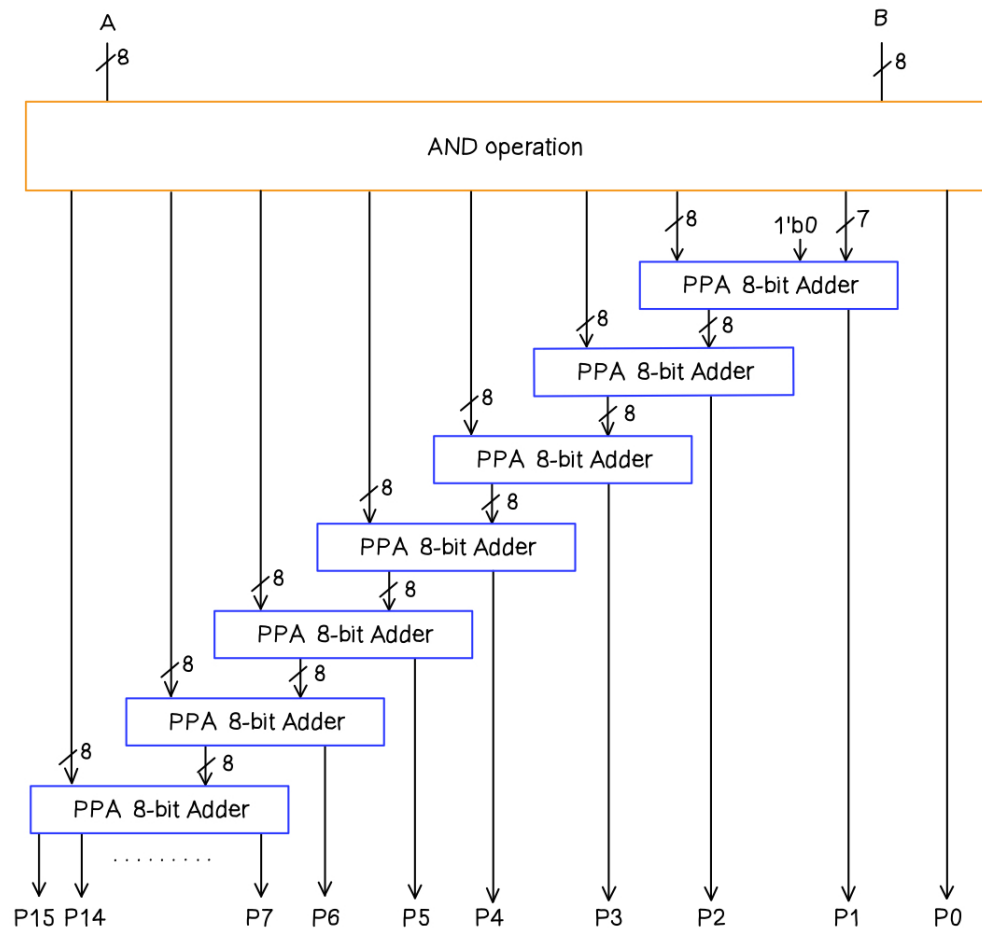
下面波形圖中可以看到所有測資皆有執行乘法，並且乘法的結果皆與測資的答案相同。

[illegible]

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

$$\text{FA.v} : (1 - 1/16) * 100\% = 93.75\%$$
$$\text{HA.v} : (1 - 0/11) * 100\% = 100\%$$

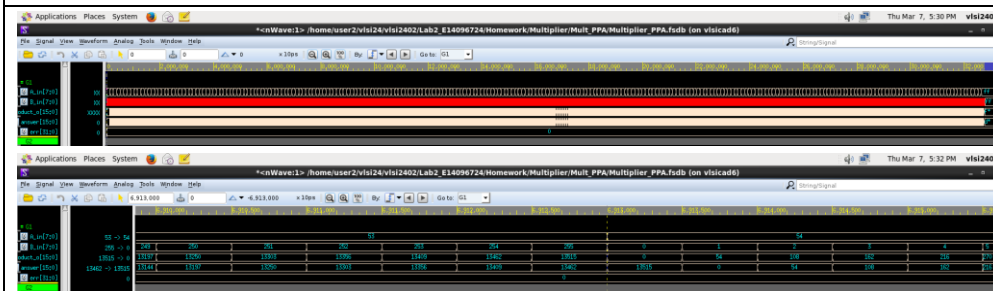
Draw your hieratical structure.



Your simulation result on the terminal.

[illegible]

Your waveform :

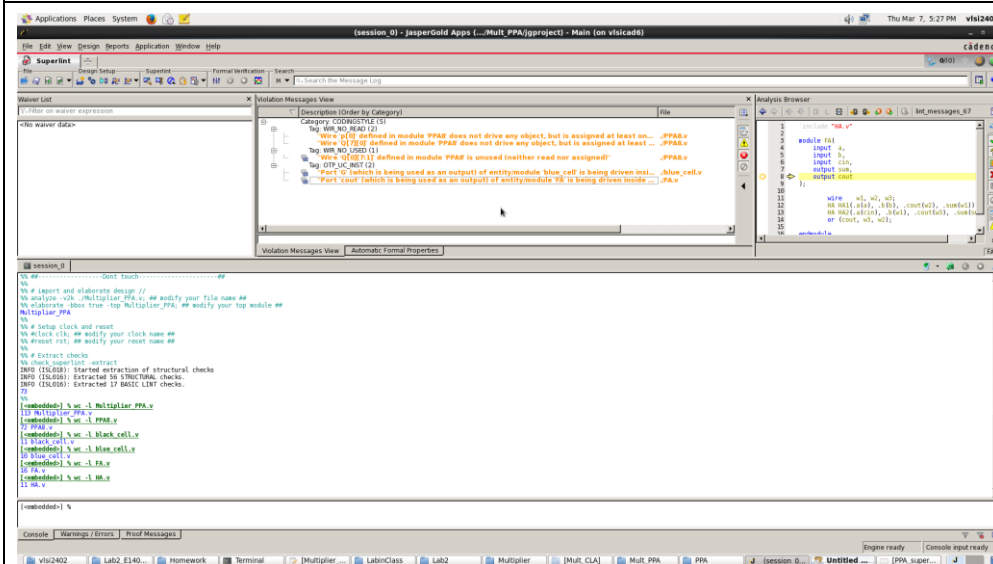


Explanation of your waveform :

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