

Verilog HDL Simulation

LAB 2 - 硬體描述語言模擬及驗證

負責助教：陳憲億、黃琮閔、鄭東昇、趙文駿、陳韋廷

Outline

Verilog HDL Simulation

Review Data type 、 Data assignment

Equivalence checking

Simulating framework

Example

Waveform viewer

Observation of circuit wave

Homework

Demo

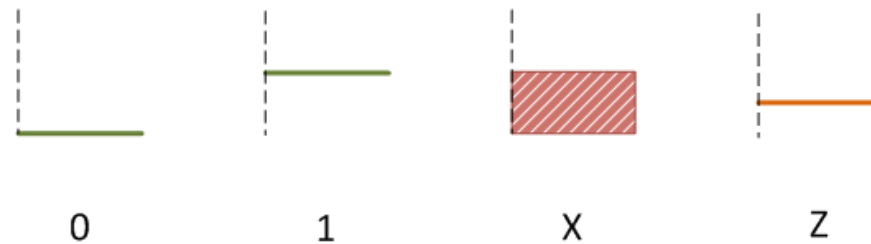
Review - Data type

- variables 的 data types 具有以下四種狀態

0	represents a logic zero , or a false condition
1	represents a logic one , or a true condition
x	represents an unknown logic value // It can be 1 or 0, where it means “ don't care ”
z	represents a high-impedance state

下圖是如何在時序圖和 simulation 的波形中表示這些值

大部分 simulation tool 皆是如此，其中紅色代表任意值，橙色代表高阻抗



Review - Data assignment

□ Continuous assignment

- Imply that whenever any change on the RHS of the assignment occurs, it is evaluated and assigned to the LHS.

Ex. wire [3:0] a, b, c ;
assign a = b + c ;

□ Procedural assignment

- Assignment to “register” data types may occur within always, initial, task and function. These expressions are controlled by **triggers which cause the assignment to evaluate.**

Ex. reg a, clk ;
Always **#5** clk = ~clk;

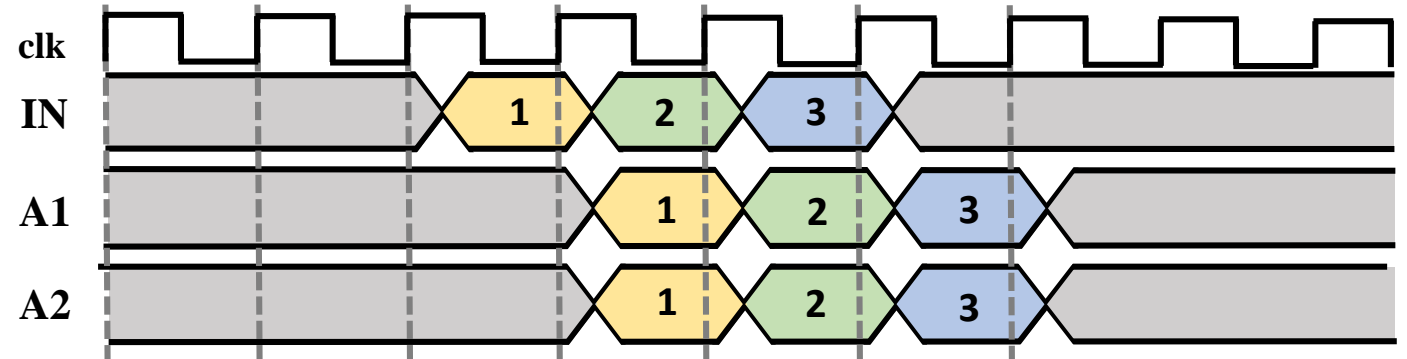
Ex. always **@(b)**
if (clk == 1'b1)
a = ~b;

Data assignment

- Blocking assignment

Ex :

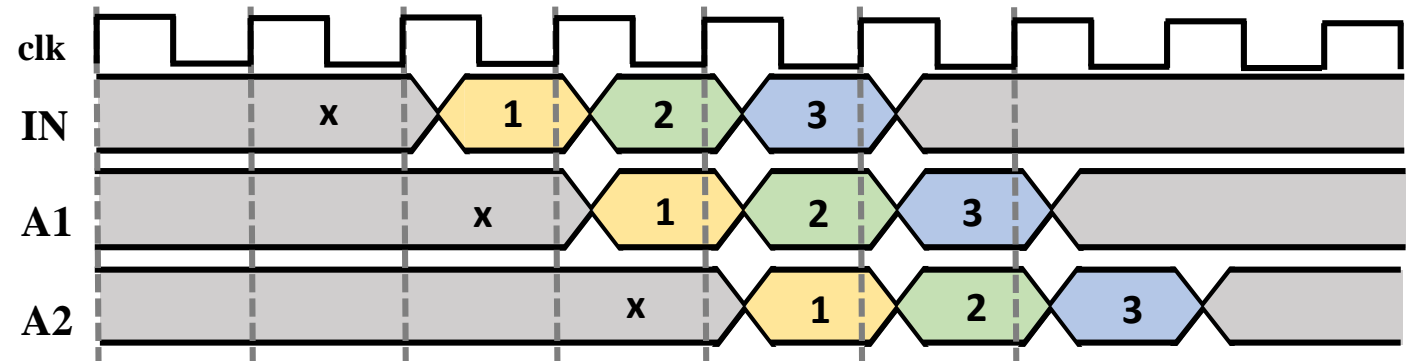
```
always @( posedge clk )  
begin  
  A1 = IN;  
  A2 = A1;  
  A3 = A2;  
end
```



- Non-Blocking assignment

Ex :

```
always @( posedge clk )  
begin  
  A1 <= IN;  
  A2 <= A1;  
  A3 <= A2;  
end
```

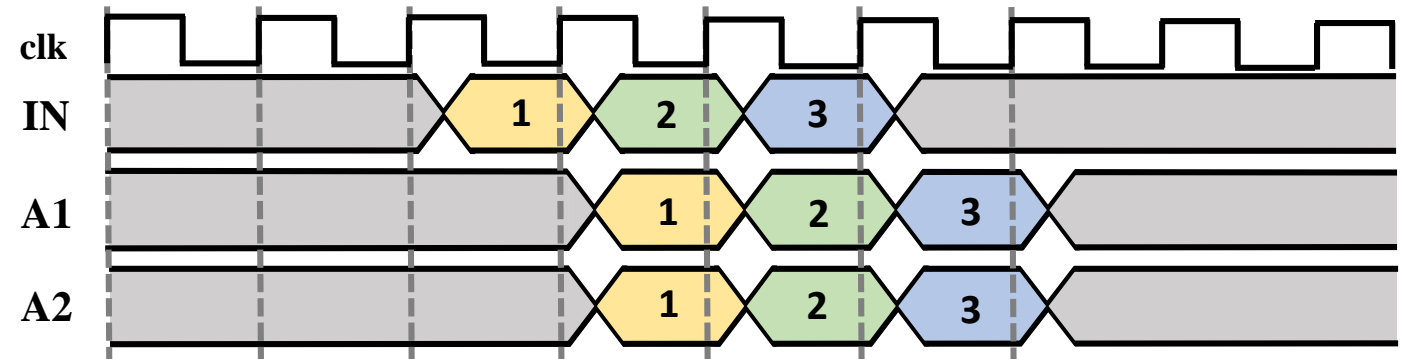
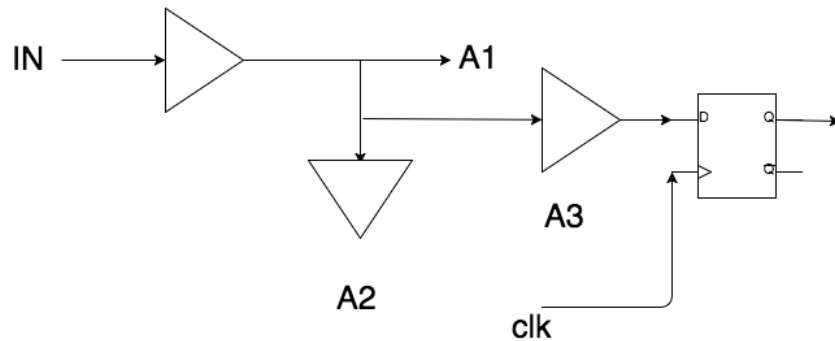


Data assignment

- Blocking assignment

Ex :

```
always @( posedge clk )  
begin  
  A1 = IN;  
  A2 = A1;  
  A3 = A2;  
end
```

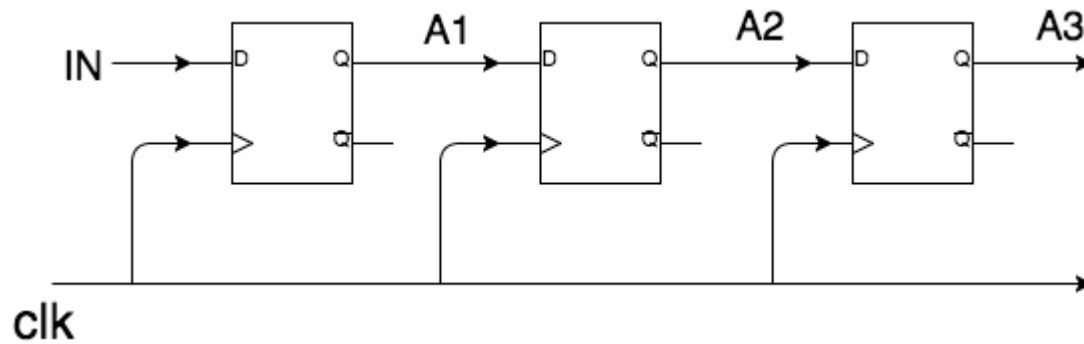
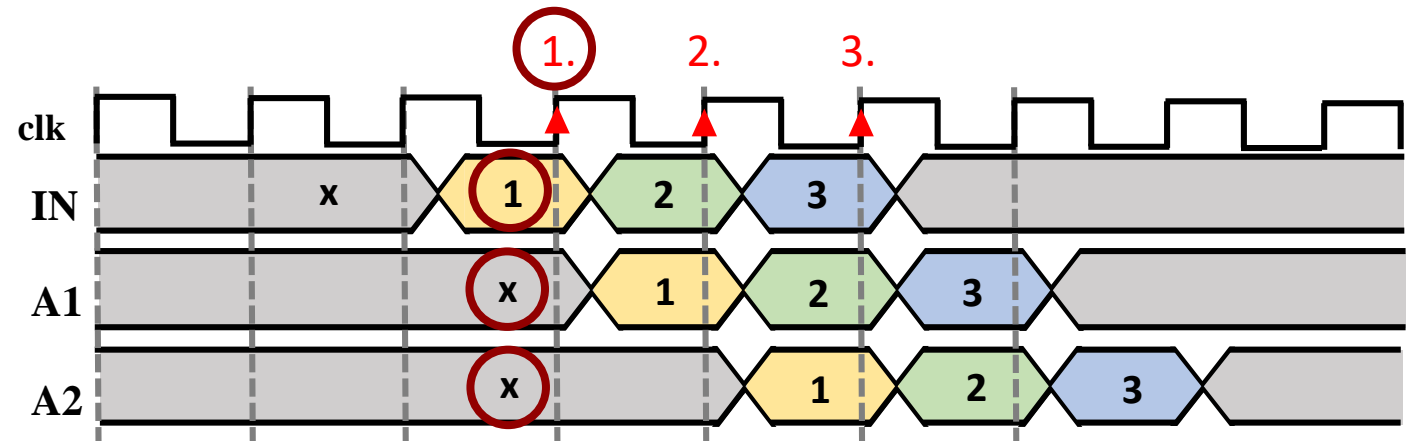


Data assignment

- Non-Blocking assignment

Ex :

```
always @( posedge clk )  
begin  
  A1 <= IN;  
  A2 <= A1;  
  A3 <= A2;  
end
```

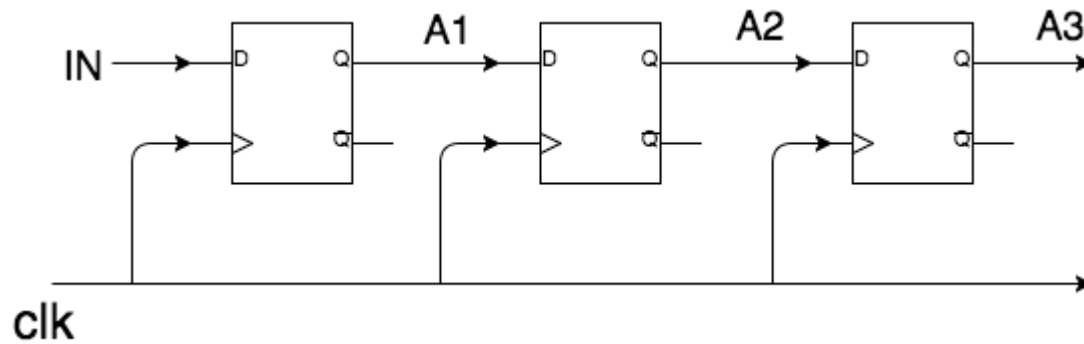
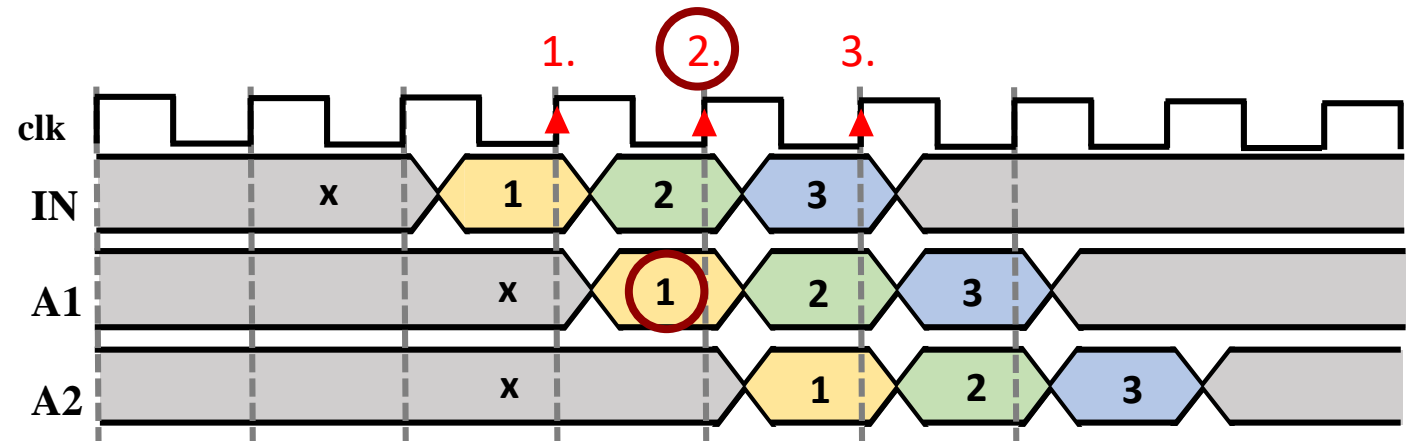


Data assignment

- Non-Blocking assignment

Ex :

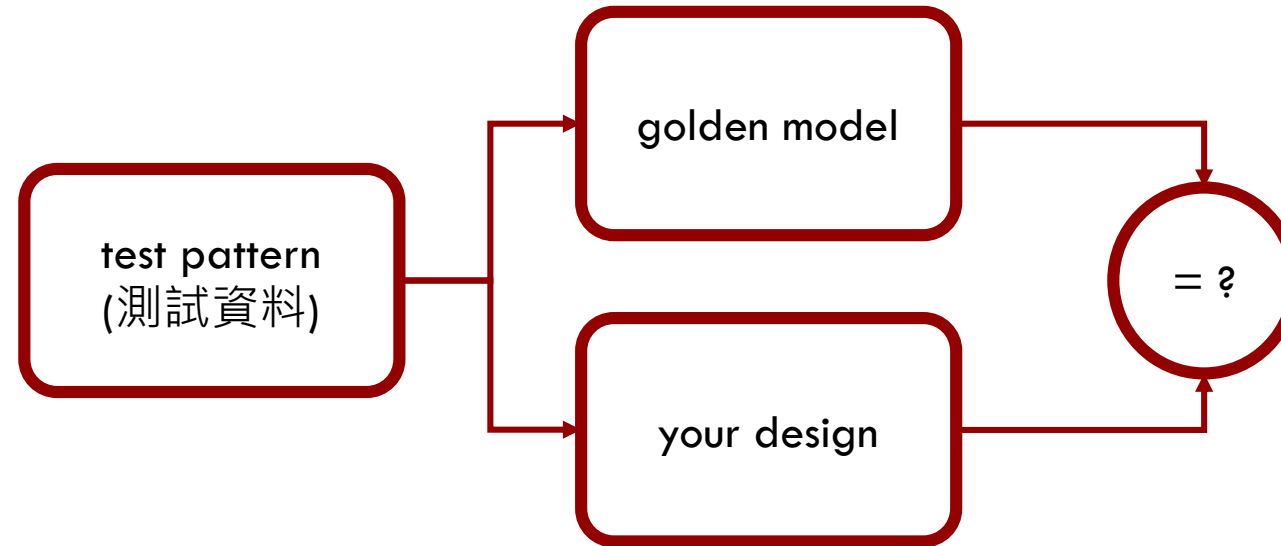
```
always @( posedge clk )  
begin  
  A1 <= IN;  
  A2 <= A1;  
  A3 <= A2;  
end
```



Equivalence checking

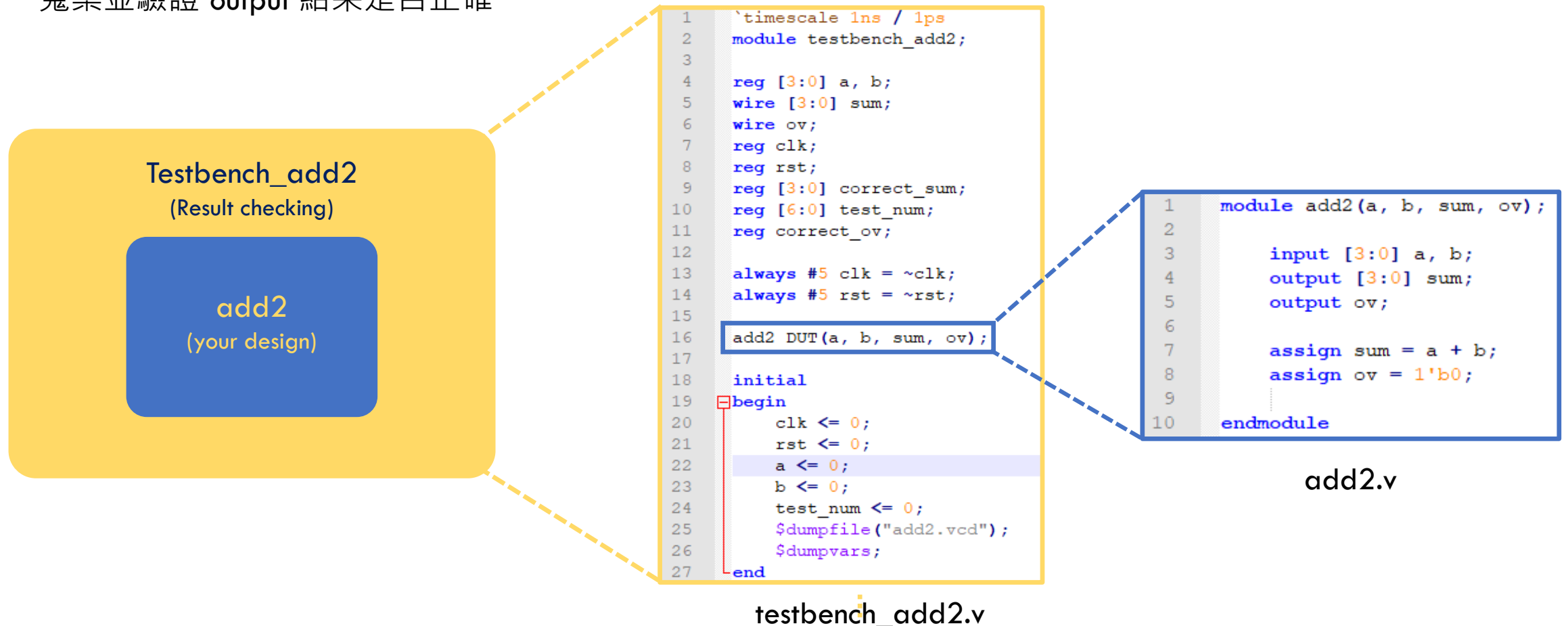
- 等效性檢查 (Equivalence Checking, EC) :

簡單來說，EC 即是確認你的設計是否與標準樣本 (golden model) 有相同的結果；其中 golden model 可以是設計者本身或是可執行之規格，或以不易出錯的方式完成的設計



Simulating framework

- Verilog 模擬是為了驗證待測設計的正確性，透過撰寫 testbench，產生 input 訊號的值給待測設計，蒐集並驗證 output 結果是否正確



Example(1 / 4)

- 我們使用 2-operand adder (二輸入加法器) 作為我們要驗證的 design
- 另外設計了 overflow 永遠等於 0，這樣當有 overflow 發生時，testbench 就會報錯

```
1 module add2(a, b, sum, ov);  
2  
3     input [3:0] a, b;  
4     output [3:0] sum;  
5     output ov;  
6  
7     assign sum = a + b;  
8     assign ov = 1'b0;  
9  
10 endmodule
```

add2.v

Example(2/4)

- 在 testbench 利用 high-level 描述產生正確的結果，並與 add2 的結果做比對

```
41 ~ else begin
42     {correct_ov, correct_sum} = a + b;
43 ~     if({ov, sum} == {correct_ov, correct_sum}) begin
44         $display ("Test %d ", test_num);
45         $display ("CORRECT! No overflow.");
46         $display ("%d + %d = ?", a, b);
47         $display ("your answer: ov = %d, sum = %d", ov, sum);
48         $display ("correct answer: ov = %d, sum = %d", ov, sum);
49         $display ("\n");
50     end
51 ~ else begin
52     $display ("Test %d ", test_num);
53     $display ("////////////////");
54     $display ("//////// Fail! //////////");
55     $display ("/// Occur Overflow! ///");
56     $display ("////////////////");
57     $display ("%d + %d = ?", a, b);
58     $display ("your answer: ov = %d, sum = %d", ov, sum);
59     $display ("correct answer: ov = %d, sum = %d", correct_ov, correct_sum);
60     $display ("\n");
61 end
62 end
```

testbench_add2.v

Example(3/4)

- 1 ns 為仿真時間，1 ps 為時間精度
- # 是延遲的意思，井號後面數字是延遲的數量，延遲的單位由`timescale控制

```
1  `timescale 1ns / 1ps

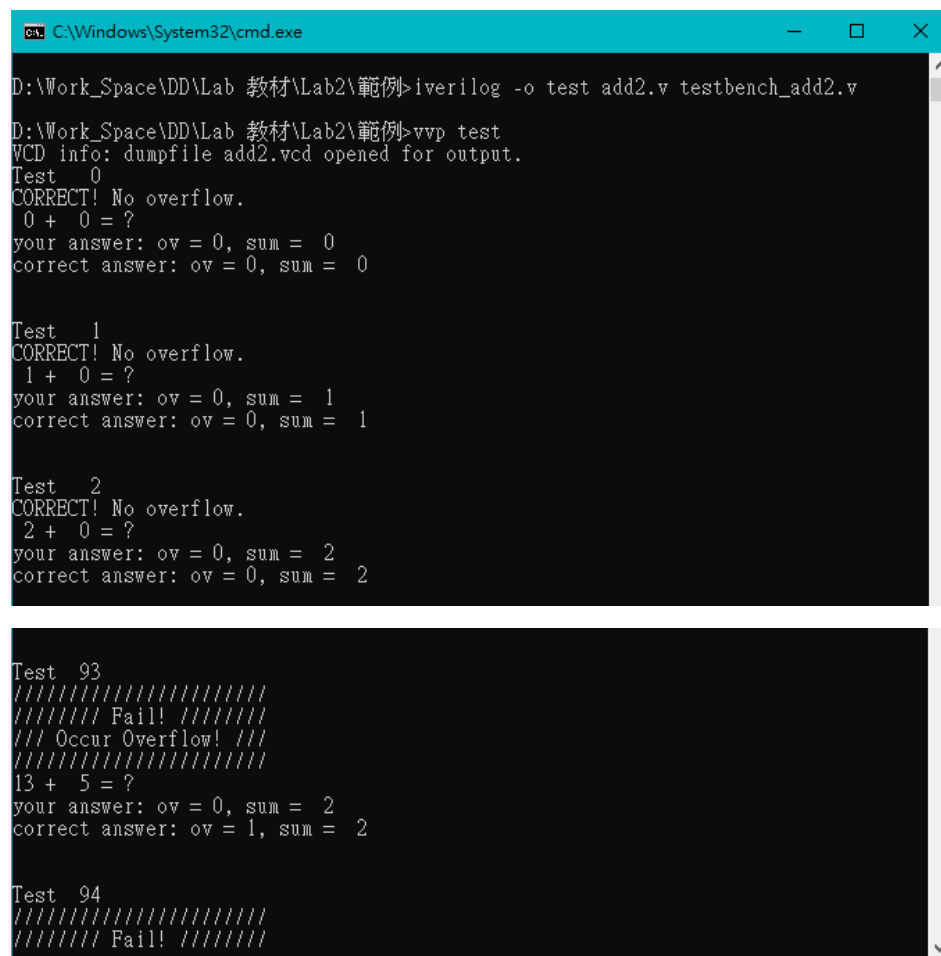
13  always #5 clk = ~clk;
14  always #5 rst = ~rst;
15
16  add2 DUT(a, b, sum, ov);
17
18  initial
19  begin
20      clk <= 0;
21      rst <= 0;
22      a <= 0;
23      b <= 0;
24      test_num <= 0;
25      $dumpfile("add2.vcd");
26      $dumpvars;
27  end
28
```

testbench_add2.v

- 利用 \$dumpfile 指令產出 vcd 檔，也就是等等會使用到的波形檔

Example(4/4)

- 此圖為編譯結果，並可看到資料夾內產生的 .vcd 電路波形檔



```
C:\Windows\System32\cmd.exe
D:\Work_Space\DD\Lab 教材\Lab2\範例>iverilog -o test add2.v testbench_add2.v
D:\Work_Space\DD\Lab 教材\Lab2\範例>vvp test
VCD info: dumpfile add2.vcd opened for output.
Test 0
CORRECT! No overflow.
0 + 0 = ?
your answer: ov = 0, sum = 0
correct answer: ov = 0, sum = 0

Test 1
CORRECT! No overflow.
1 + 0 = ?
your answer: ov = 0, sum = 1
correct answer: ov = 0, sum = 1

Test 2
CORRECT! No overflow.
2 + 0 = ?
your answer: ov = 0, sum = 2
correct answer: ov = 0, sum = 2

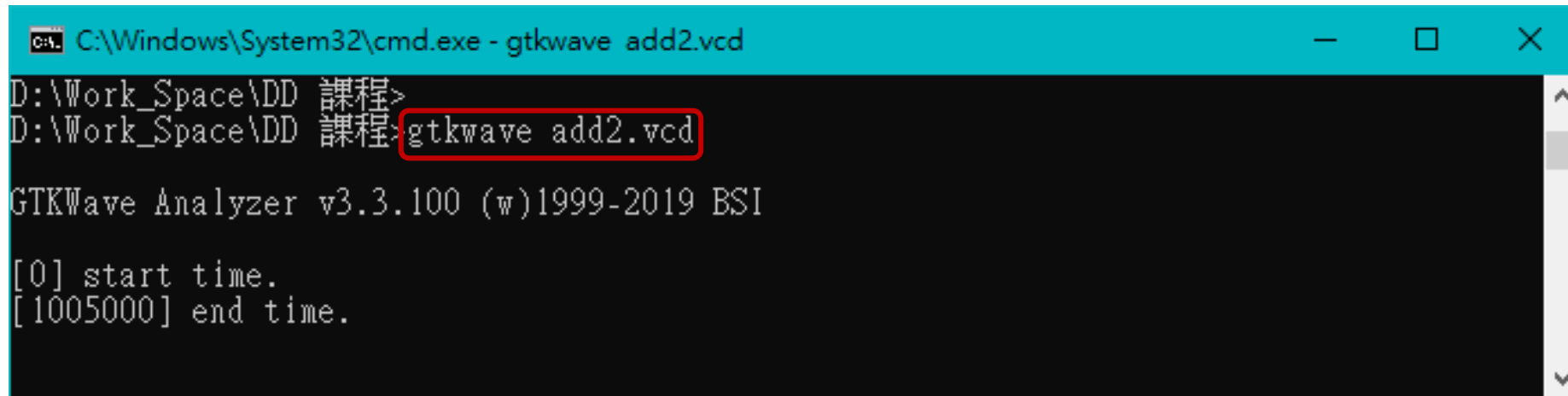
Test 93
////////////////////
//////// Fail! //////////
/// Occur Overflow! ///
////////////////////
13 + 5 = ?
your answer: ov = 0, sum = 2
correct answer: ov = 1, sum = 2

Test 94
////////////////////
//////// Fail! //////////
```

Waveform viewer (1 / 2)

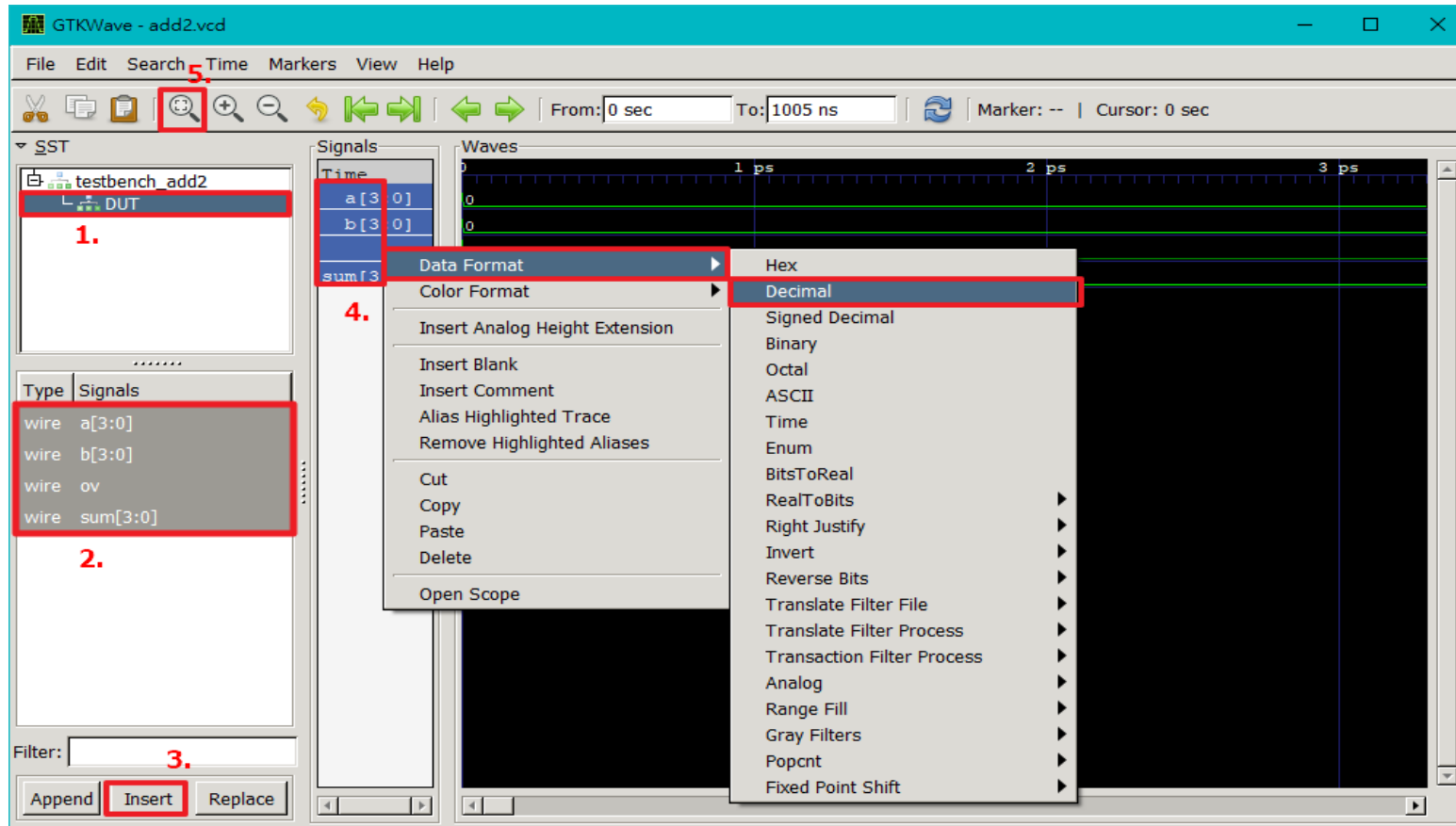
將範例程式編譯，並使用以下指令觀察波形：

> gtkwave add2.vcd



```
C:\Windows\System32\cmd.exe - gtkwave add2.vcd
D:\Work_Space\DD 課程>
D:\Work_Space\DD 課程>gtkwave add2.vcd
GTKWave Analyzer v3.3.100 (w)1999-2019 BSI
[0] start time.
[1005000] end time.
```

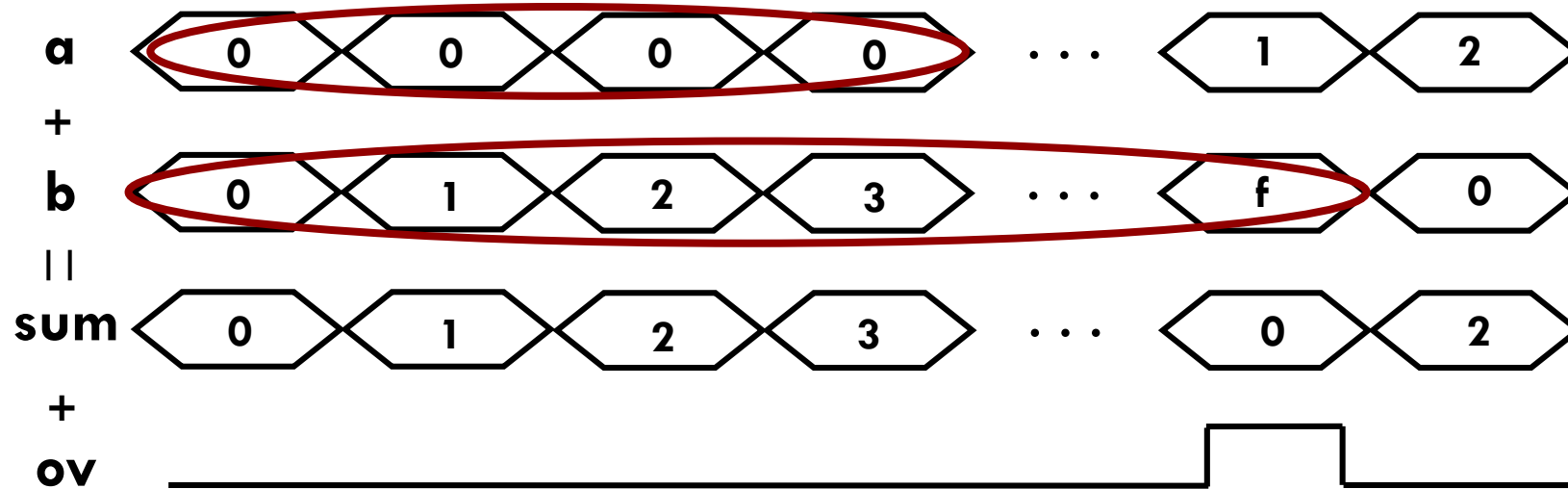
Waveform viewer (2/2)



1. 點選SST 區域中 testbench_add2 中的被測元件 (DUT)
2. 選取變數
3. 點擊 Insert
4. 為方便觀察，按步驟將資料格式改成 10 進位
5. 點擊 Zone Fit

Observation of circuit wave

- 波形會以時序圖的方式呈現，接著就可以觀察結果是否與你設計相符

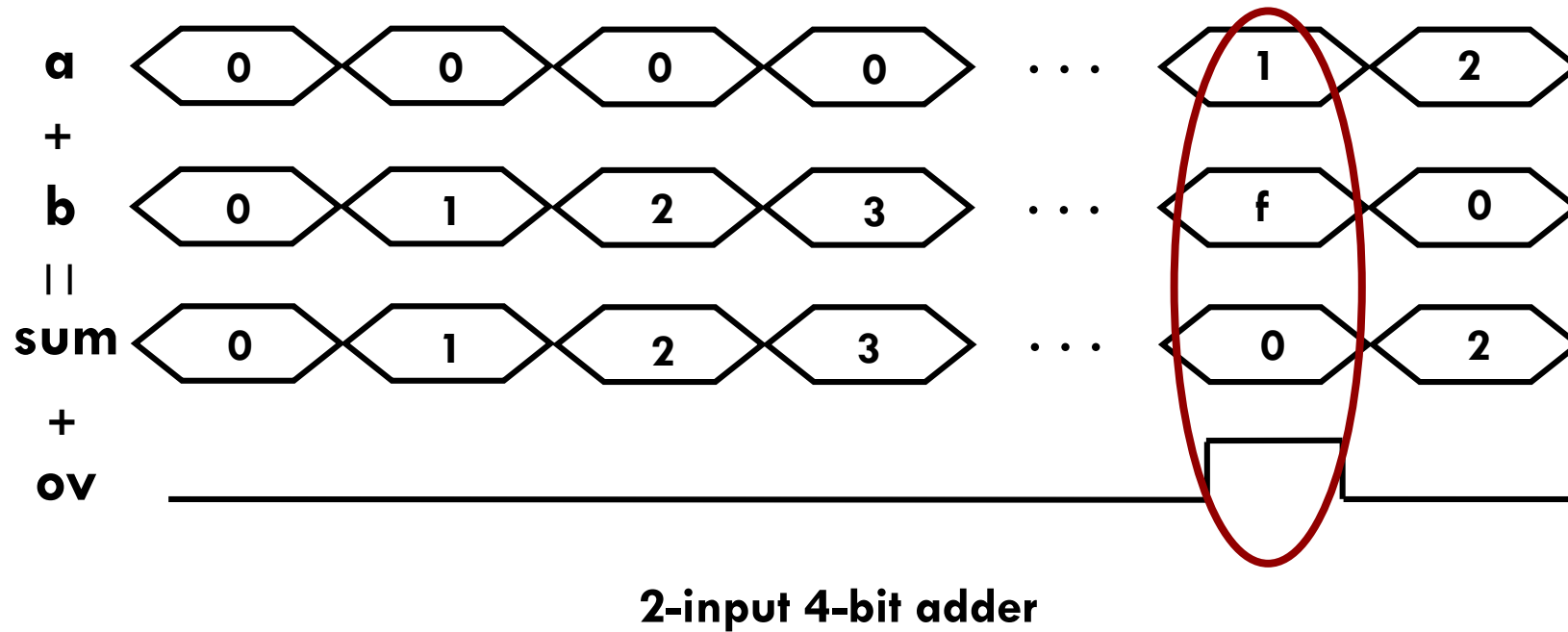


2-input 4-bit adder

0 _{hex} = 0 _{dec} = 0 _{oct}	0	0	0	0
1 _{hex} = 1 _{dec} = 1 _{oct}	0	0	0	1
2 _{hex} = 2 _{dec} = 2 _{oct}	0	0	1	0
3 _{hex} = 3 _{dec} = 3 _{oct}	0	0	1	1
4 _{hex} = 4 _{dec} = 4 _{oct}	0	1	0	0
5 _{hex} = 5 _{dec} = 5 _{oct}	0	1	0	1
6 _{hex} = 6 _{dec} = 6 _{oct}	0	1	1	0
7 _{hex} = 7 _{dec} = 7 _{oct}	0	1	1	1
8 _{hex} = 8 _{dec} = 10 _{oct}	1	0	0	0
9 _{hex} = 9 _{dec} = 11 _{oct}	1	0	0	1
A _{hex} = 10 _{dec} = 12 _{oct}	1	0	1	0
B _{hex} = 11 _{dec} = 13 _{oct}	1	0	1	1
C _{hex} = 12 _{dec} = 14 _{oct}	1	1	0	0
D _{hex} = 13 _{dec} = 15 _{oct}	1	1	0	1
E _{hex} = 14 _{dec} = 16 _{oct}	1	1	1	0
F _{hex} = 15 _{dec} = 17 _{oct}	1	1	1	1

Observation of circuit wave

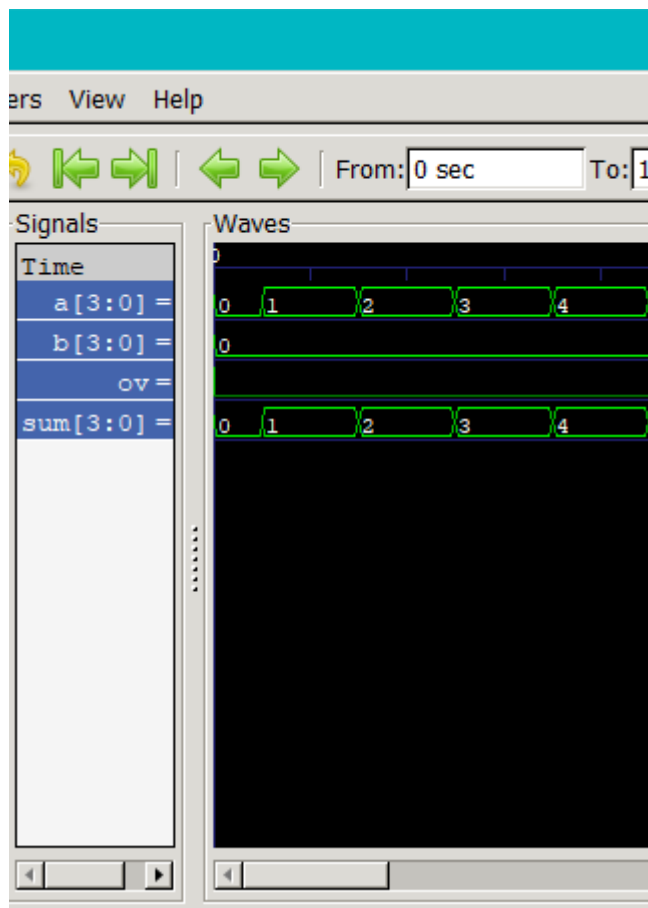
- 波形會以時序圖的方式呈現，接著就可以觀察結果是否與你設計相符



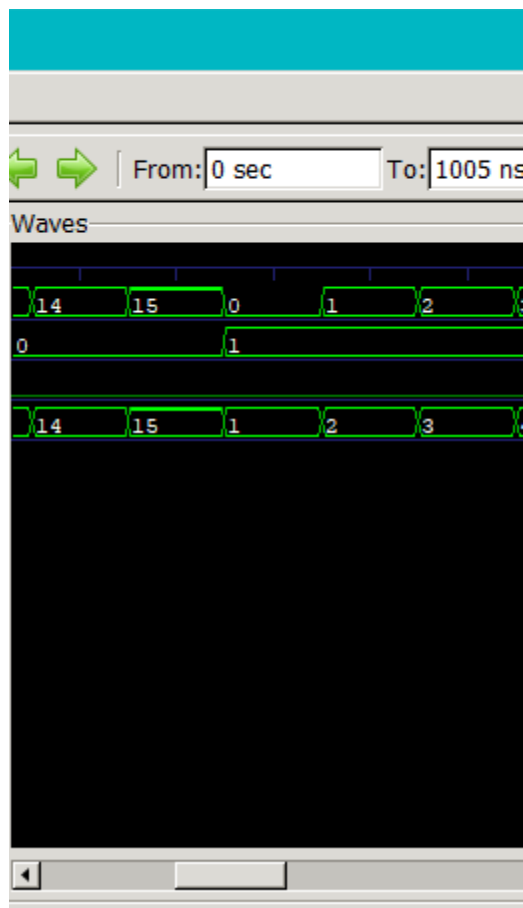
0 _{hex} = 0 _{dec} = 0 _{oct}	0	0	0	0
1 _{hex} = 1 _{dec} = 1 _{oct}	0	0	0	1
2 _{hex} = 2 _{dec} = 2 _{oct}	0	0	1	0
3 _{hex} = 3 _{dec} = 3 _{oct}	0	0	1	1
4 _{hex} = 4 _{dec} = 4 _{oct}	0	1	0	0
5 _{hex} = 5 _{dec} = 5 _{oct}	0	1	0	1
6 _{hex} = 6 _{dec} = 6 _{oct}	0	1	1	0
7 _{hex} = 7 _{dec} = 7 _{oct}	0	1	1	1
8 _{hex} = 8 _{dec} = 10 _{oct}	1	0	0	0
9 _{hex} = 9 _{dec} = 11 _{oct}	1	0	0	1
A _{hex} = 10 _{dec} = 12 _{oct}	1	0	1	0
B _{hex} = 11 _{dec} = 13 _{oct}	1	0	1	1
C _{hex} = 12 _{dec} = 14 _{oct}	1	1	0	0
D _{hex} = 13 _{dec} = 15 _{oct}	1	1	0	1
E _{hex} = 14 _{dec} = 16 _{oct}	1	1	1	0
F _{hex} = 15 _{dec} = 17 _{oct}	1	1	1	1

Result

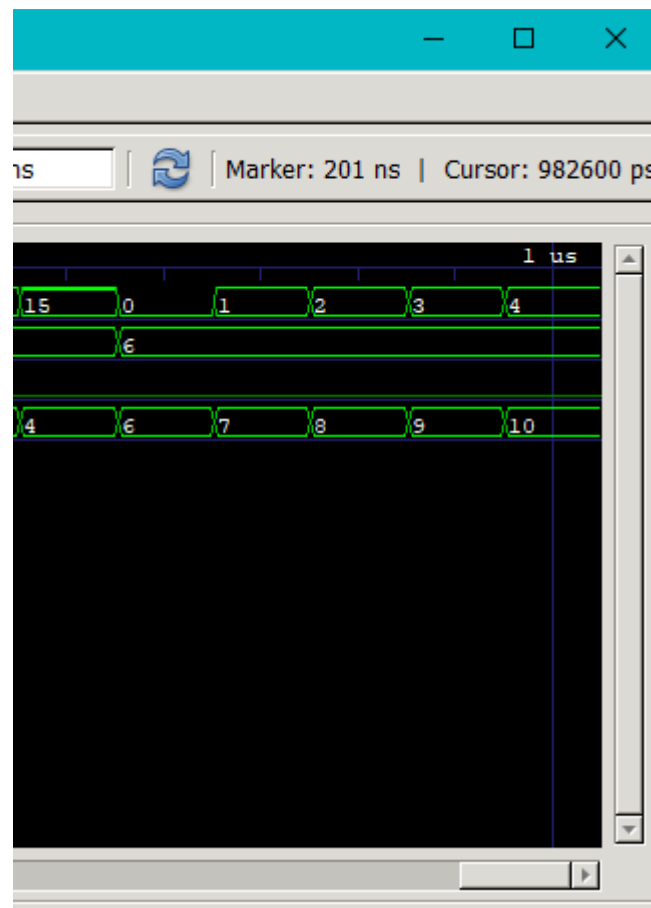
查看生成的波形圖，可以確認設計是否正確



...



...



Homework

- 題目：完成 4-operand adder 自動結果比對，請同學參考 Example 將輸入從零到十五所有組合當作輸入的 pattern，並 show 出結果
- 我們會拿左下角的 design，用同學們做的 testbench 驗證，result 為結果
- 請同學打開波形圖，並說明任意一組 overflow 發生的原因

```
add4.v
1  module add4(a, b, c, d, sum, ov);
2
3      input [3:0] a, b, c, d;
4      output [4:0] sum;
5      output ov;
6
7      assign sum = a + b + c + d;
8      assign ov = 1'b0;
9
10 endmodule
```

add4.v

```
C:\Windows\System32\cmd.exe - vvp test
D:\Work_Space\DD\Lab 教材\Lab2\作業\改>vvp test
VCD info: dumpfile add4.vcd opened for output.
Test    0
CORRECT! No overflow.
  0 + 0 + 0 + 0 = ?
your answer: ov = 0, sum = 0
correct answer: ov = 0, sum = 0

Test    1
CORRECT! No overflow.
  1 + 0 + 0 + 0 = ?
your answer: ov = 0, sum = 1
correct answer: ov = 0, sum = 1

Test    2
CORRECT! No overflow.
  2 + 0 + 0 + 0 = ?
your answer: ov = 0, sum = 2
correct answer: ov = 0, sum = 2

Test    3
CORRECT! No overflow.
  3 + 0 + 0 + 0 = ?
your answer: ov = 0, sum = 3
correct answer: ov = 0, sum = 3

Test    4
CORRECT! No overflow.
```

```
4 + 0 + 0 + 0 = ?  
your answer: ov = 0, sum = 4  
correct answer: ov = 0, sum = 4
```

•
•

當 a 循環結束時，進入下一個循環

```
Test 15  
CORRECT! No overflow.  
15 + 0 + 0 + 0 = ?  
your answer: ov = 0, sum = 15  
correct answer: ov = 0, sum = 15
```

```
Test 16  
CORRECT! No overflow.  
0 + 1 + 0 + 0 = ?  
your answer: ov = 0, sum = 1  
correct answer: ov = 0, sum = 1
```

```
Test 17  
CORRECT! No overflow.  
1 + 1 + 0 + 0 = ?  
your answer: ov = 0, sum = 2  
correct answer: ov = 0, sum = 2
```

•
•

```
Test 65281  
CORRECT! No overflow.  
1 + 0 + 15 + 15 = ?  
your answer: ov = 0, sum = 31  
correct answer: ov = 0, sum = 31
```

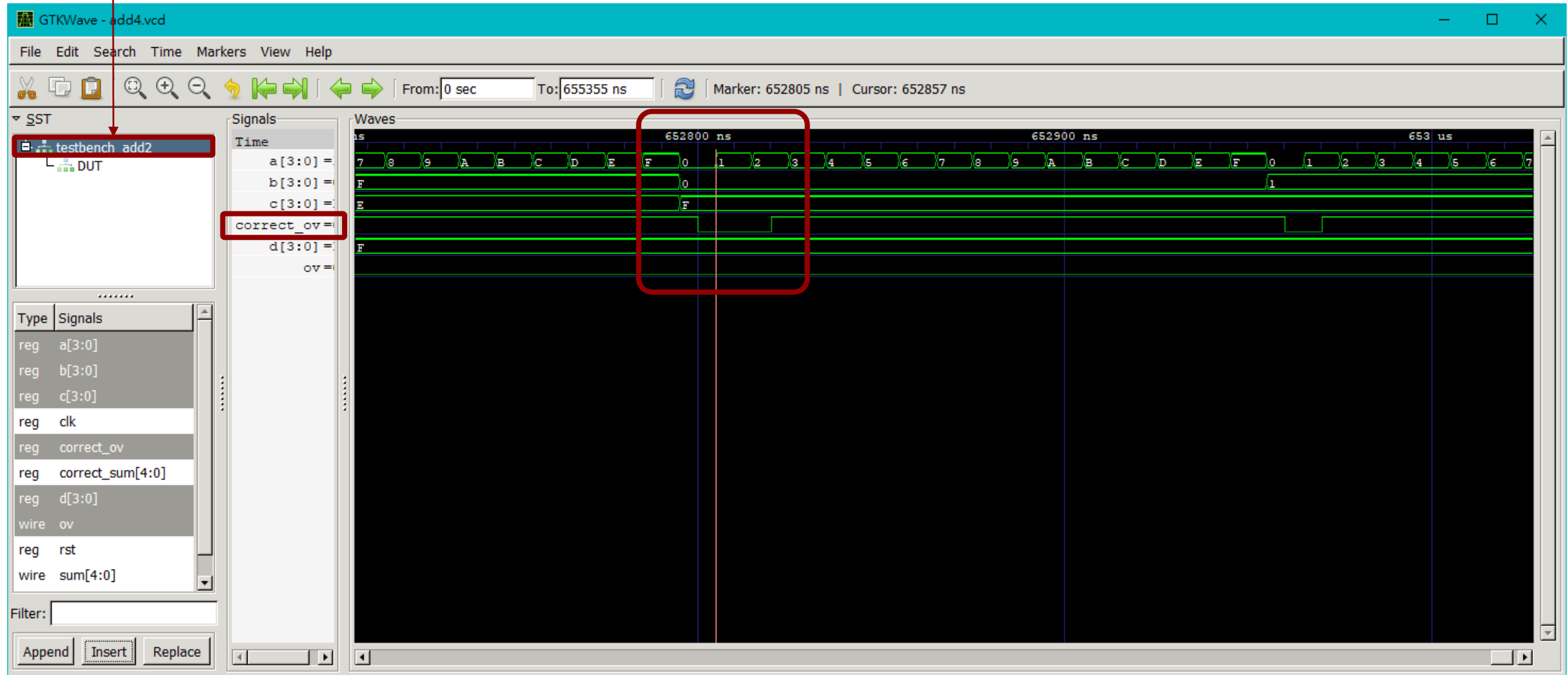
```
Test 65282  
////////////////////  
//////// Fail! //////////  
/// Occur Overflow! ///  
////////////////////  
2 + 0 + 15 + 15 = ?  
your answer: ov = 0, sum = 0  
correct answer: ov = 1, sum = 0
```

•
•

```
Test 65535  
////////////////////  
//////// Fail! //////////  
/// Occur Overflow! ///  
////////////////////  
15 + 15 + 15 + 15 = ?  
your answer: ov = 0, sum = 28  
correct answer: ov = 1, sum = 28
```

從一組沒有 correct_ov 的到有 correct_ov 時，
在波形圖可見其變化

點選 testbench_add2 可選 testbench 裡定義的變數，如 correct_ov



add4.vcd

Demo

- Demo 地點：EA 501A
- Demo 時間：依時間表為準
- 可提前5分鐘入場準備，其餘時間違規進入，一次扣總成績3分
- 安排時段內無法展示請即刻離場，違者一次扣總成績5分
- 可攜帶自己的筆電 Demo
- 請以隨身碟攜帶檔案 (不要用雲端)