數位系統導論實驗
Lab5 Verilog

Outline

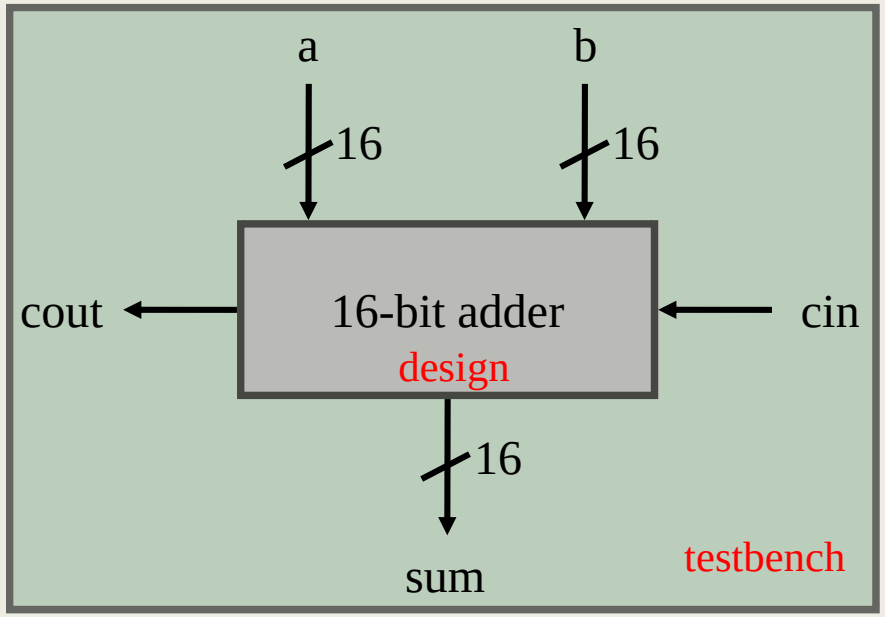
- 課程目的
- Verilog簡介
- 實驗環境
- 範例練習
- 作業說明及評分方式

課程目的

- Verilog是一種硬體描述語言，我們能透過程式碼描述硬體的結構和行為，完成電路設計
- 接下來幾週的課程，將會帶領同學練習使用Verilog設計電路，最後實現 DNN 的硬體設計
- 在本課程中將簡介Verilog，使用Icarus Verilog中的指令觀察硬體的執行狀況，讓同學更熟悉Verilog

Verilog簡介

- 在Verilog中我們建構各個模組 (module) ，採「由上而下」階層方式設計硬體
- 利用測試平台（Testbench）驗證設計的功能是否符合需求
- 能夠描述多種層次電路，例如：描述模組功能的行為層次 (Behavioral level) 、描述邏輯閘連接形式的邏輯層次 (Gate level) 等

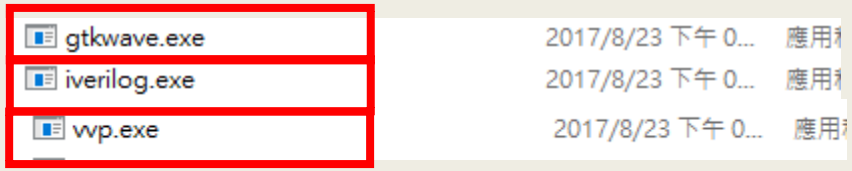


實驗環境 – Icarus Verilog

- 在本次實驗課，同學將使用 Icarus Verilog 的 iverilog、vvp、gtkwave 來模擬及觀測 8-bit adder 的執行結果和波形

1. 將附檔解壓縮後打開bin資料夾

2. 依序安裝執行檔：iverilog.exe、vvp.exe、gtkwave.exe



實驗環境 – 設定環境變數 (1/2)

- 避免同學將程式全放在bin資料夾編譯、執行，請同學依照下面步驟操作：

1. 打開檔案總管

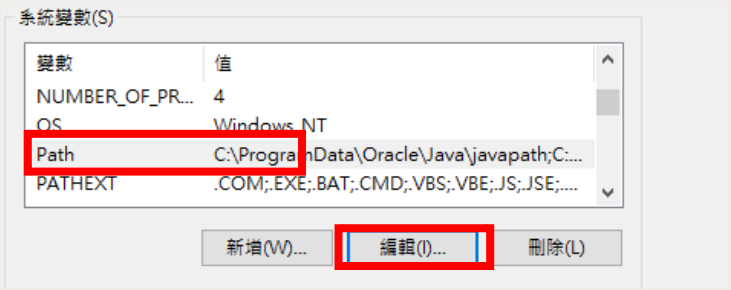
2. 在本機圖示點擊右鍵，選擇內容

3. 點擊進階系統設定

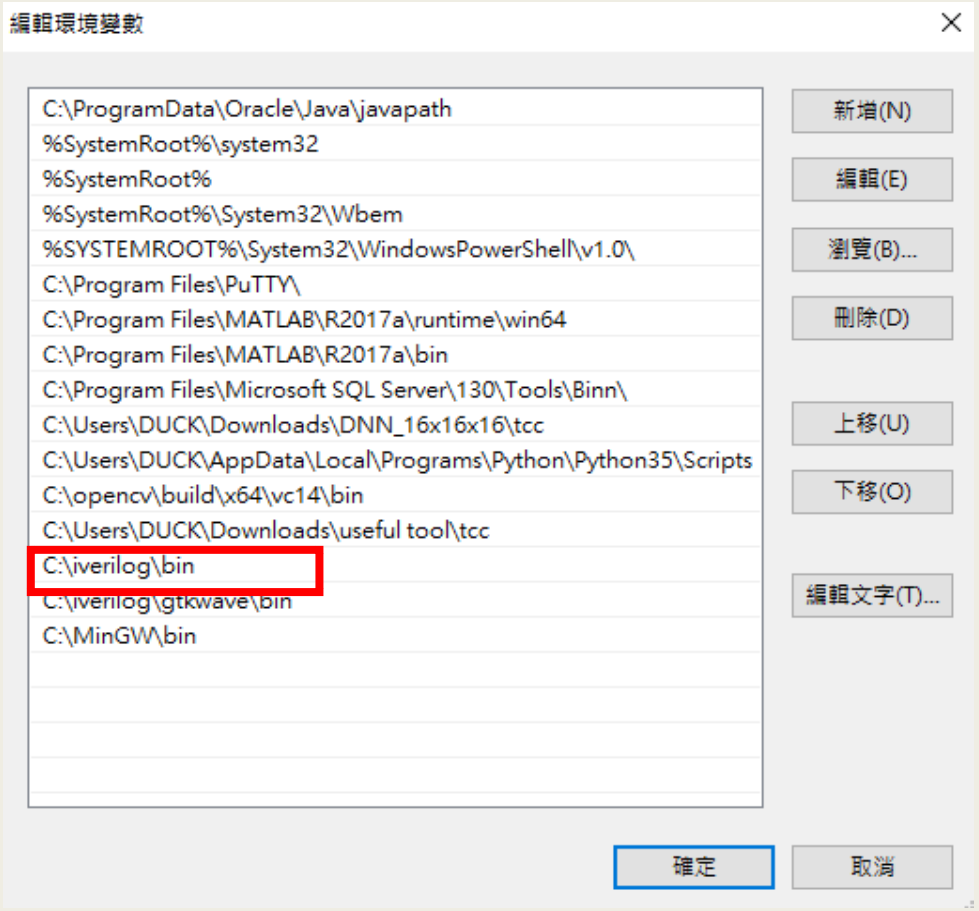
實驗環境 – 設定環境變數 (2/2)

4. 點擊環境變數

5. 點擊path並按下編輯



6. 新增並輸入bin資料夾路徑，按下確定

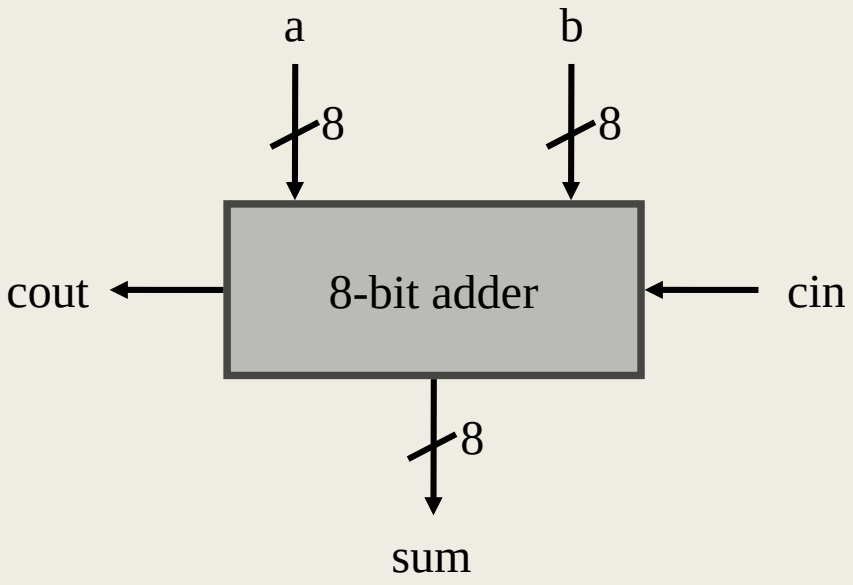


※路徑為iverilog與gtkwave下的bin資料夾，

助教已將資料放置同處，同學只需新增一個環境變數

範例練習 – 8-bit adder

- 在本實驗中同學將透過8-bit adder的範例，熟悉Verilog 基本結構並藉由 Icarus Verilog 進行編譯、模擬驗證
- 依照所設計的block diagram 編寫 Verilog 程式碼 (cont_8bit_adder.v)




```
1  `timescale 1ns / 1ps
2  /*宣告8-bit adder module名稱,輸出入名稱*/
3  module cont_8bit_adder(sum, cout, a, b, cin);
4      /*定義port,包含input、output*/
5      input [7:0] a, b;
6      input cin;
7      output [7:0] sum;
8      output cout;
9      /*將a、b、cin相加,而cout為第8bit值，sum則為第0bit~7bit值*/
10     assign {cout,sum} = a+b+cin;
11 endmodule
```

範例練習 – Test Bench (1/2)

- 我們使用 test bench 作為驗證手段，test bench 將輸入信號傳送到16-bit adder，再將運算結果傳回來
- 依照所需測試的範圍，撰寫 test bench 測試程式(testbench_cont_16bit_adder.v)，驗證設計之 8-bit adder 輸出結果是否符合設計功能

```
1  `timescale 1ns / 1ps
2  /*宣告testbench module*/
3  module testbench_cont_8bit_adder;
4  /*定義資料型態*/
5  reg [7:0] a, b;
6  reg cin;
7  wire [7:0] sum;
8  wire cout;
9  /*呼叫8-bit adder module*/
10 cont_8bit_adder DUT(sum, cout, a, b, cin);
```

```
11 // behavior description
12 initial begin
13     $dumpfile("cont_8bit_adder.vcd");//繪製波形檔
14     $dumpvars;//繪製波形檔
15 end
16 /*初始化設定*/
17 initial
18 begin
19     a = 8'b11110000;
20     b = 8'b11111111;
21     cin = 1'b1;
22 end
```

範例練習 – Test Bench (2/2)

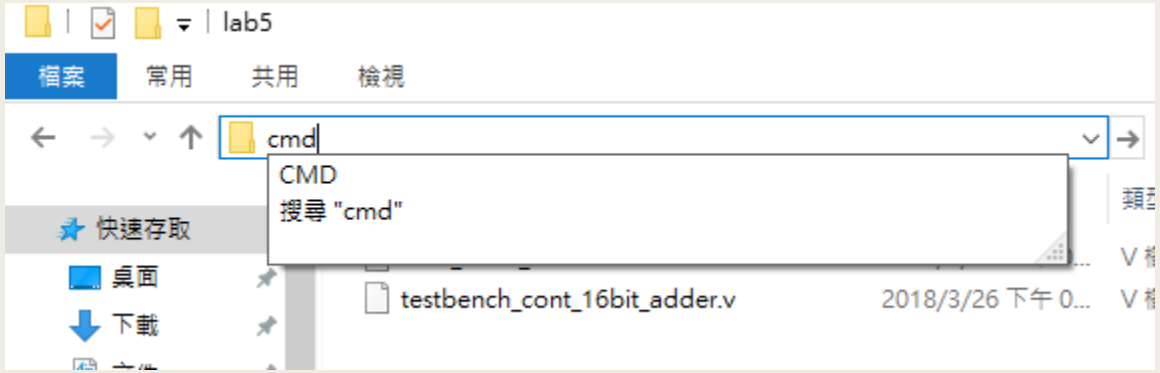
- 我們使用 test bench 作為驗證手段，test bench 將輸入信號傳送到16-bit adder，再將運算結果傳回來
- 依照所需測試的範圍，撰寫 test bench 測試程式(testbench_cont_16bit_adder.v)，驗證設計之 8-bit adder 輸出結果是否符合設計功能

```
23  /*每經過1ns執行一次*/
24  always #1
25  begin
26      a = ( a << 1 | cin );
27      /*使用$monitor 印出所有input、output數值變化*/
28      $monitor("%4dns monitor: a=%d b=%d cin=%d sum=%d cout=%d",$stime, a, b, cin, sum, cout);
29  end
30
31  always #2 b = b >> 2;/*每經過2ns執行一次*/
32  always #3 cin = ~cin ;/*每經過3ns執行一次*/
33  initial #15 $finish; /*經過15ns後結束程式*/
34  endmodule
```

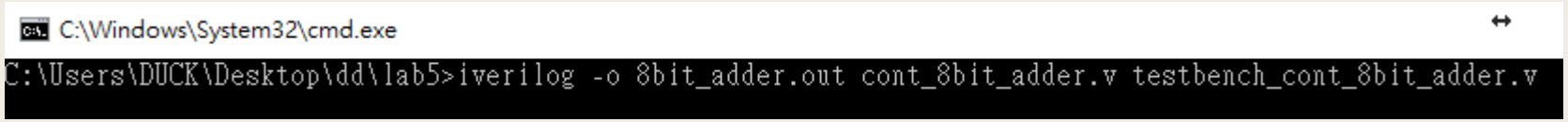
範例練習 – 編譯範例

- 在本實驗中同學將透過 8-bit adder 的範例，藉由 Icarus Verilog 進行編譯、模擬驗證
- 使用Icarus Verilog的 iverilog、vvp 兩個指令，進行編譯及模擬

1. 在程式檔案路徑欄位打開命令提示字元



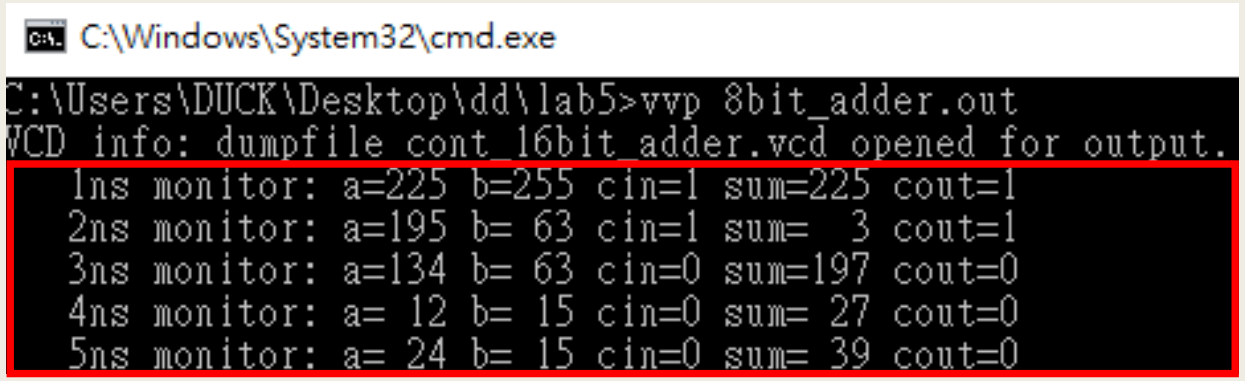
2. 輸入以下指令進行編譯：
   - iverilog –o 8bit_adder.out cont_8bit_adder.v testbench_cont_8bit_adder.v



※8bit_adder.out為編譯後產生的檔案

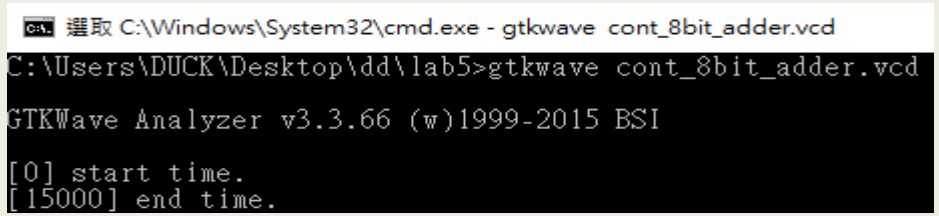
範例練習－執行範例

- 輸入指令執行程式，檢視設計之 8-bit adder 功能是否有錯誤：
  - vvp 8bit_adder.out



範例練習 – 查看波形圖 (1/2)

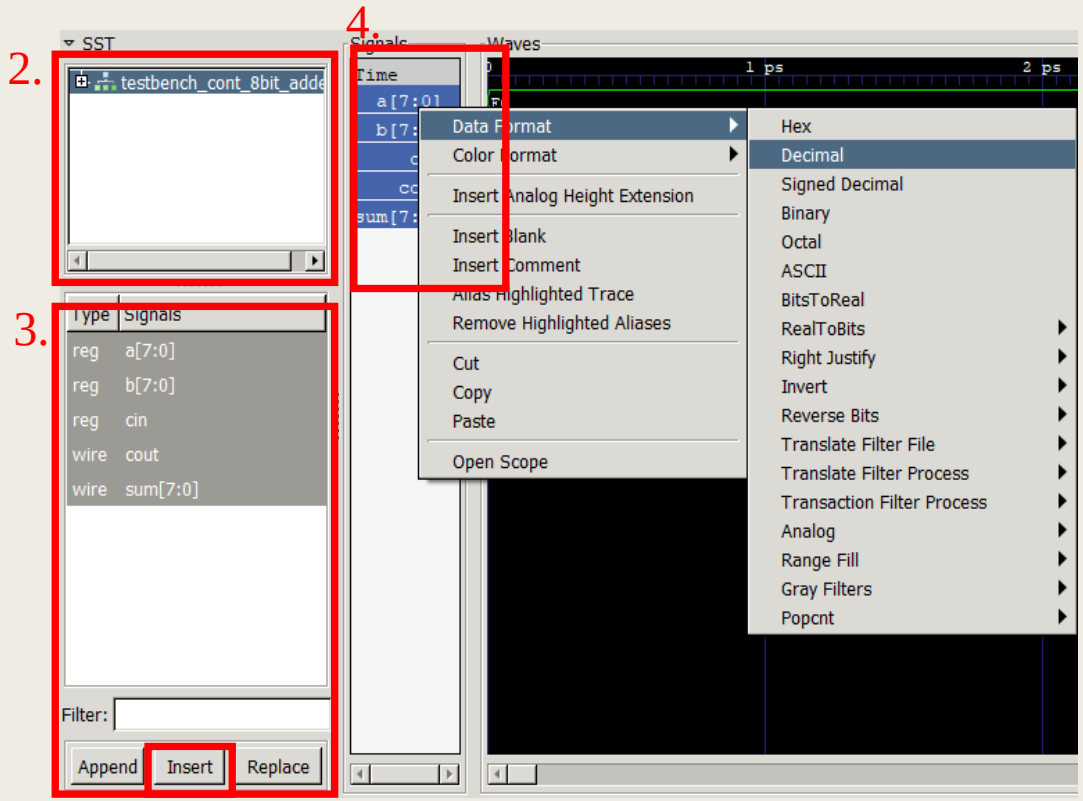
1. 輸入指令，查看 test bench 產生的波形檔：
   - gtkwave cont_8bit_adder.vcd



2. 點擊SST區域中testbench_cont_8bit_adder
3. 選取變數並且點擊Insert
4. 在Signals區域選取變數，

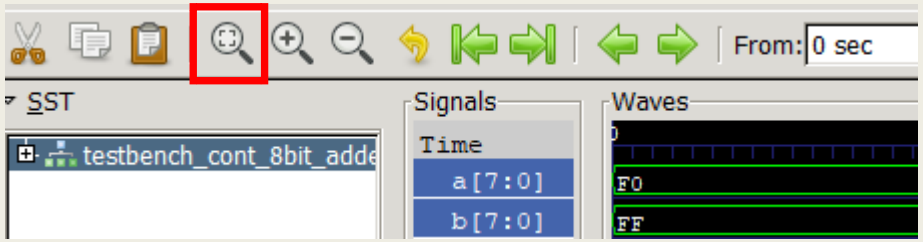
   點擊右鍵並選擇Data Format的Decimal

   ※為了方便同學觀察，以十進位觀看波形圖

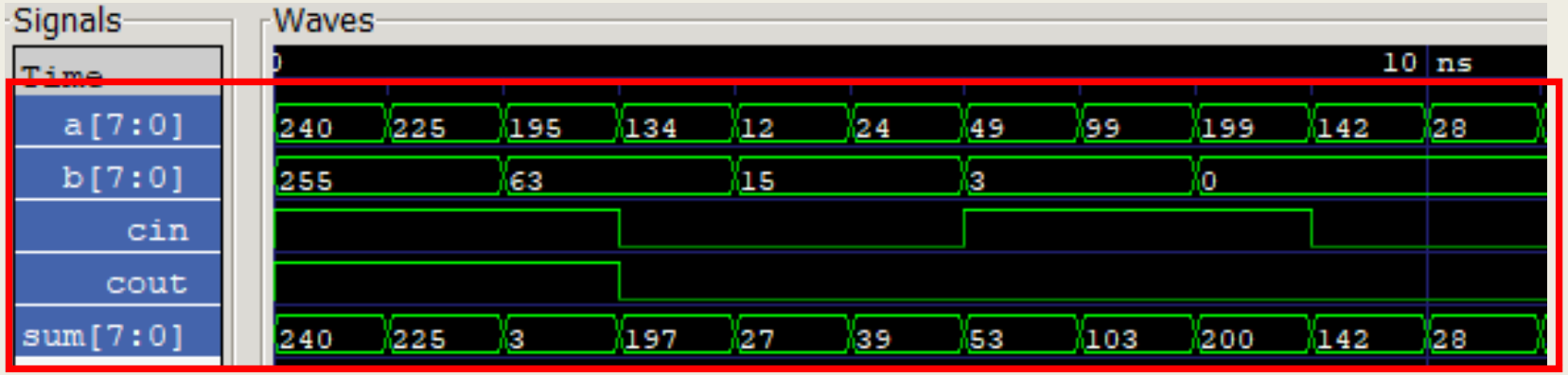


範例練習 – 查看波形圖 (2/2)

5. 點擊Zoot Fit



6. 看到生成的波形圖，同學可以確認設計是否正確



作業 – part 1

- 題目：完成範例練習，使用命令提示字元及GTKWave觀察所有變數

作業 – part 2

- 題目：使用testbench測試a和b可能的全部加法(256的平方，共35536筆)，使用GTKWave顯示波型

作業 – part 3

- 題目：承接作業 part2 ，利用testbench判斷加法器運算是否正確，並顯示運算後的數值與預期答案

作業說明及課程評分

- Demo 時間：測驗時間共分四梯次，分別為 19:30、19:50、 20:10 與 20:30
- Demo 梯次：與 Lab 1 相同
- Demo 地點：計中217
- 評分方式：完成part1 40%，part2 40%，part3 20%