

數位系統導論

Lab5 Verilog Basics & Simulation

助教：林子淵 zihyuanlin@gmail.com

Outline

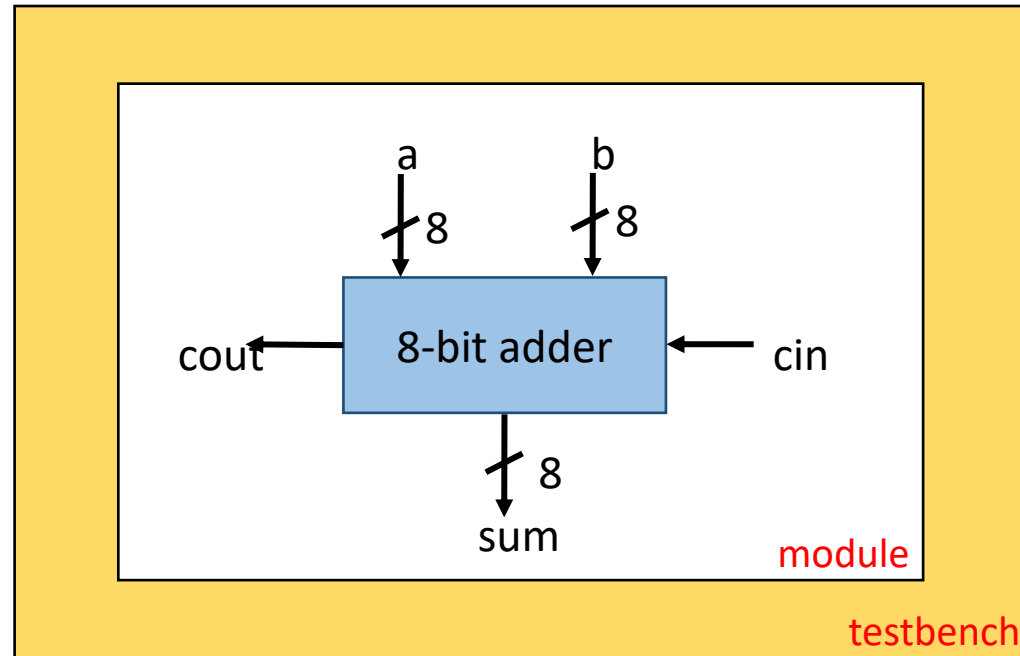
- 課程目的
- Verilog簡介與範例
- Verilog模擬與工具
- Lab說明及評分方式

課程目的

- 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
- 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

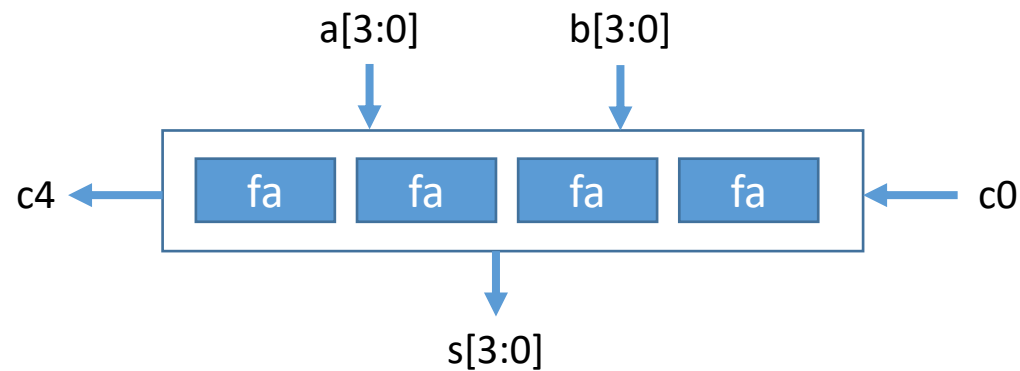
Verilog 簡介 (1/4)

- Verilog是一種硬體描述語言，透過程式碼描述硬體的結構和行為，完成電路設計
- 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



Verilog 簡介 (2/4)

- Verilog電路的建模方式主要分為 Structural Modeling 與 Behavioral Modeling
- Structural Modeling：以邏輯閘、預定義模組及連結方式明確地描述數位電路



```
module fa(a, b, ci, s, co);  
  input a, b, ci;  
  output s, co;  
  assign {co,s}=a+b+ci;  
endmodule
```

```
module adder(a, b, c0, s, c4);  
  input [3:0] a, b;  
  input c0;  
  output [3:0] s;  
  output c4;  
  wire c1, c2, c3;
```

```
  fa bit0(a[0], b[0], c0, s[0], c1);  
  fa bit1(a[1], b[1], c1, s[1], c2);  
  fa bit2(a[2], b[2], c2, s[2], c3);  
  fa bit3(a[3], b[3], c3, s[3], c4);  
endmodule
```

Verilog 簡介 (3/4)

- Behavioral Modeling：以行為描述數位電路，常用於設計 testbench，可分為 Continuous Assignment 與 Procedural Assignment
- Continuous Assignment：以 assign 描述硬體的架構連結，位於 always 與 initial 外，且等式的左邊須為線

Continuous Assignment

```
module adder(a, b, c0, s, c4);  
    input [3:0] a, b;  
    input c0;  
    output [3:0] s;  
    output c4;  
  
    assign {c4,s}=a+b+c0;  
endmodule
```

Verilog 簡介 (4/4)

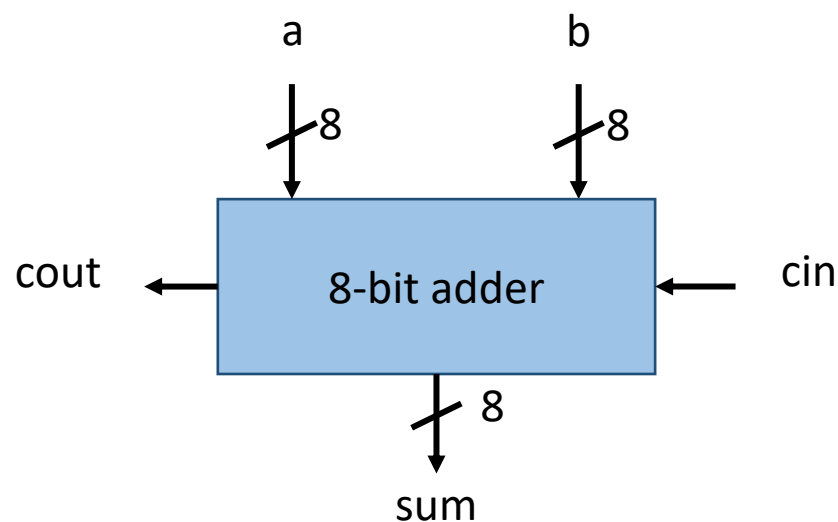
- Procedure Assignment：Verilog 的 Behavioral code 通常位於 Procedure Block 內，該Assignment 賦值於型態為 reg、integer、時間變數...等的變數，但不能賦值於線 (wire)，Procedure Block 的兩種型式為 always 與 initial

Procedural Assignment

```
module adder(a, b, c0, s, c4);  
    input [3:0] a, b;  
    input c0;  
    output [3:0] s;  
    output c4;  
    reg [3:0] s;  
    reg c4;  
  
    always@(a or b or c0)  
        {c4,s}=a+b+c0;  
endmodule
```

範例練習一 – 8-bit adder

- 在本實驗中同學將透過8-bit adder的範例，熟悉Continuous Assignment和Procedural Assignment，並藉由Icarus Verilog 進行編譯



範例練習一 – 8-bit adder

- 依照所設計的block diagram 編寫 Verilog 程式碼 (cont_8bit_adder.v)
(proc_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
```

Continuous Assignment

```
1  `timescale 1ns / 1ps
2  /*宣告8bit adder module名稱,輸出入名稱*/
3  module proc_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      /*在procedural assignment下,等號的左邊需為reg型態*/
10     reg [7:0] sum;
11     reg cout;
12
13     /*將a、b、cin相加,而cout為第8bit值,sum則為0~7bit值*/
14     always@(a or b or cin)
15         {cout, sum} = a + b + cin;
16
17 endmodule
```

Procedural Assignment

範例練習一 – Test Bench (1/2)

- 我們使用 testbench (testbench_cont_8bit_adder.v) 作為驗證手段，testbench 將輸入信號傳送到8-bit adder，再將運算結果傳回來，確認輸出結果是否符合設計功能
- 紅框處，在testbench開始時，對a、b、cin 進行初始化

```
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
```

範例練習一 – Test Bench (2/2)

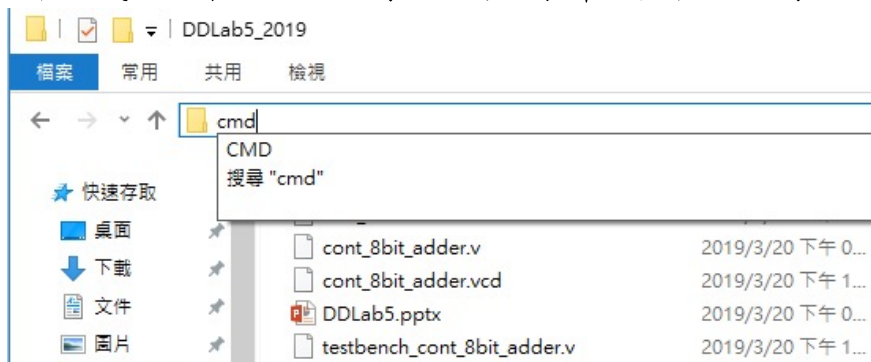
- 對三個input (a、b、cin) 打入15個pattern，最後透過\$monitor印出所有input/output的數值變化，藉此觀察該16筆運算結果是否正確

```
27  always #1
28  begin
29      a = ( a<<1 | cin );/*a左移1bit後再與cin做or*/
30      /*使用$monitor印出所有input、output數值變化*/
31      $monitor("%4dns monitor: a=%d b=%d cin=%d sum=%d cout=%d", $stime, a, b, cin, sum, cout);
32  end
33
34  always #2 b = b >>2;/*每經過2ns右移2bit*/
35  always #3 cin = ~cin;/*每經過3ns反向一次*/
36  initial #15 $finish;/*經過15ns後結束程式*/
37  endmodule
```

範例練習一 – 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**

```
D:\DDLAb5_2019>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

範例練習一 – 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

```
D:\VDDLab5_2019>vvp 8bit_adder.out
VCD info: dumpfile cont 8bit adder.vcd opened for output.
1ns monitor: a=225 b=255 cin=1 sum=225 cout=1
2ns monitor: a=195 b= 63 cin=1 sum= 3 cout=1
3ns monitor: a=134 b= 63 cin=0 sum=197 cout=0
4ns monitor: a= 12 b= 15 cin=0 sum= 27 cout=0
5ns monitor: a= 24 b= 15 cin=0 sum= 39 cout=0
6ns monitor: a= 49 b= 3 cin=1 sum= 53 cout=0
7ns monitor: a= 99 b= 3 cin=1 sum=103 cout=0
8ns monitor: a=199 b= 0 cin=1 sum=200 cout=0
9ns monitor: a=142 b= 0 cin=0 sum=142 cout=0
10ns monitor: a= 28 b= 0 cin=0 sum= 28 cout=0
11ns monitor: a= 56 b= 0 cin=0 sum= 56 cout=0
12ns monitor: a=113 b= 0 cin=1 sum=114 cout=0
13ns monitor: a=227 b= 0 cin=1 sum=228 cout=0
14ns monitor: a=199 b= 0 cin=1 sum=200 cout=0
15ns monitor: a=142 b= 0 cin=0 sum=142 cout=0
```

```
27 always #1
28 begin
29     a = ( a<<1 | cin );/*a左移1bit後再與cin做or*/
30     /*使用$monitor印出所有input、output數值變化*/
31     $monitor("%4dns monitor: a=%d b=%d cin=%d sum=%d",
32     end
```

```
34 always #2 b = b >>2;/*每經過2ns右移2bit*/
35 always #3 cin = ~cin;/*每經過3ns反向一次*/
36 initial #15 $finish;/*經過15ns後結束程式*/
```


範例練習一 – 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

```
D:\DDLlab5_2019>gtkwave cont_8bit_adder.vcd
```

```
GTKWave Analyzer v3.3.48 (w)1999-2013 BSI
```

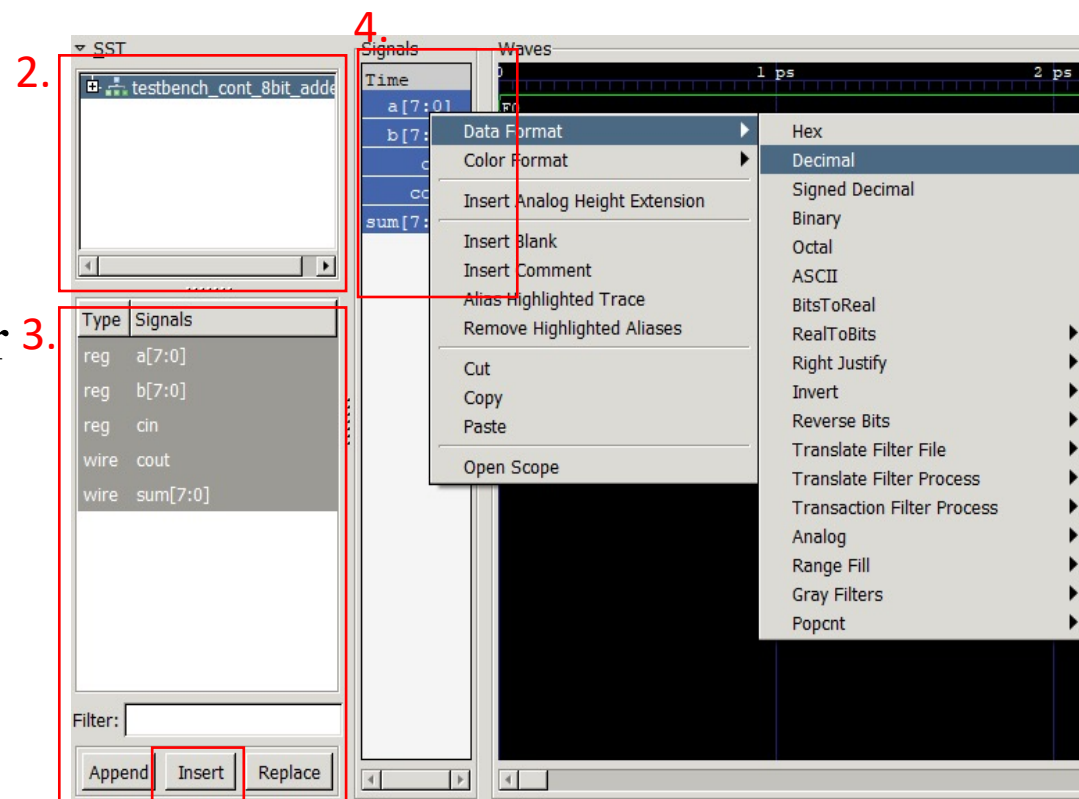
2. 點擊SST區域中testbench_cont_8bit_adder

3. 選取變數並且點擊Insert

4. 在Signals區域選取變數，

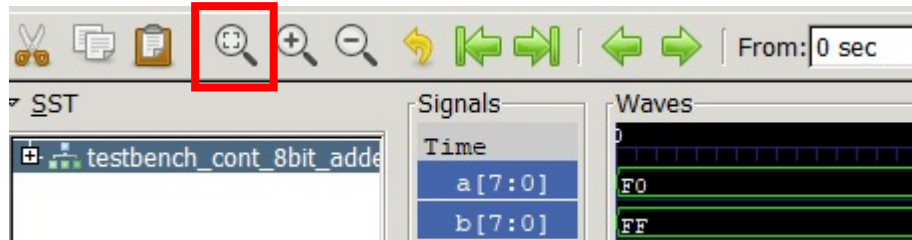
點擊右鍵並選擇Data Format的Decimal

※為了方便同學觀察，以十進位觀看波形圖

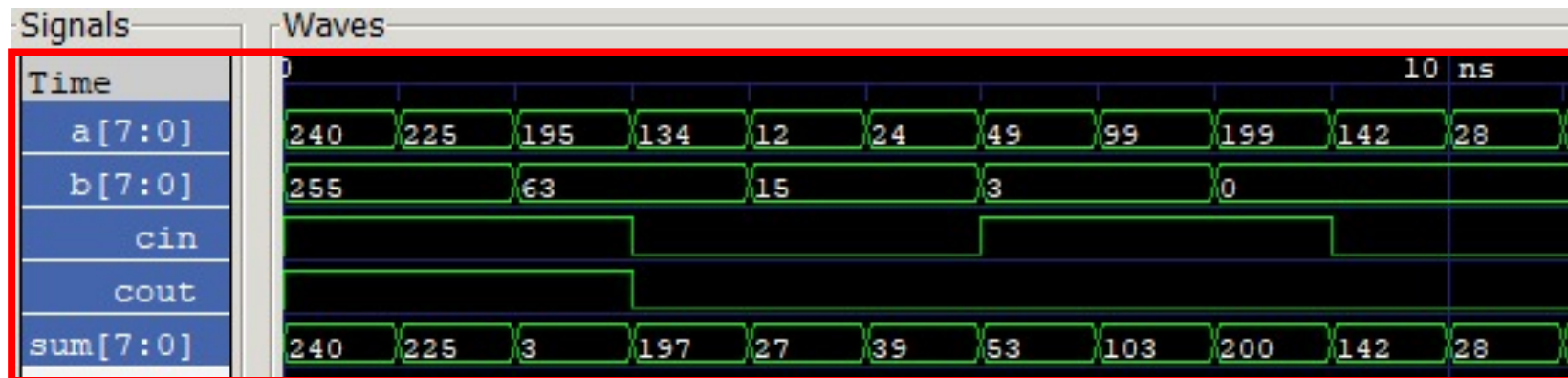


範例練習一 – 查看波形圖 (2/2)

5. 點擊Zoom Fit



6. 看到生成的波形圖，同學可以確認設計是否正確



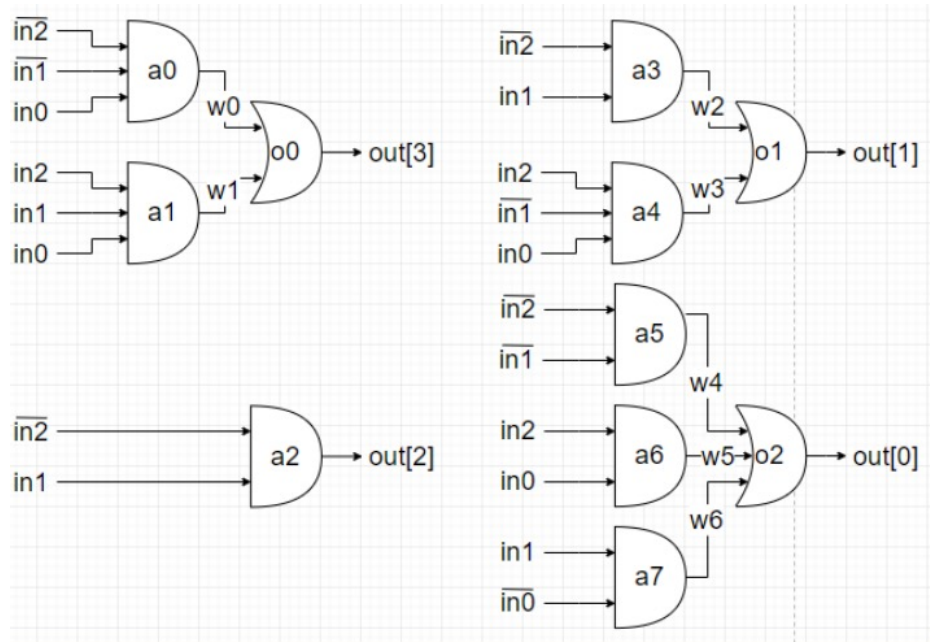
範例練習二 – Structural Modeling (1/3)

- 將生日碼產生器以 Structural Modeling 完成，並以 testbench 驗證19760319
- 請同學開啟課程壓縮包內的 DDLab5_tbstructuralex_2019.v

本機 > Win10_OS (C:) > 使用者 > user > 下載 > DDLab5_2019 > orig > structural example				
名稱	修改日期	類型	大小	
DDLab5_structuralex_2019.v	2019/3/28 下午 1...	V 檔案	1 KB	
DDLab5_tbstructuralex_2019.v	2019/3/28 下午 1...	V 檔案	1 KB	
lab5.fsdb	2019/3/28 下午 1...	FSDb 檔案	2 KB	
test	2019/3/28 下午 1...	檔案	5 KB	

範例練習二 – Structural Modeling (2/3)

1. 將生日碼產生器的邏輯圖內各個邏輯閘與線命名
2. 宣告輸入埠、輸出埠與線
3. 以 Verilog 內建的邏輯閘模組描述電路



```
module lab5(in, out); //模組 lab5(對應的輸入埠與輸出埠)

    //宣告區段
    input [2:0] in; //宣告輸入埠 in, 其形態為 3bit wire
    output [3:0] out; //宣告輸出埠 out, 其形態為 4bit wire
    wire nin2, nin1, nin0; //宣告線
    wire w0, w1, w2, w3, w4, w5, w6;

    //邏輯區段
    //使用 not 將in[2]的補數訂為線nin2
    not n2(nin2, in[2]),
        n1(nin1, in[1]),
        n0(nin0, in[0]);

    //使用 and 描述電路圖上的 and gate 與線的關係
    and a0(w0, nin2, nin1, in[0]),
        a1(w1, in[2], in[1], in[0]),
        a2(out[2], nin2, in[1]),
        a3(w2, nin2, in[1]),
        a4(w3, in[2], nin1, in[0]),
        a5(w4, nin2, nin1),
        a6(w5, in[2], in[0]),
        a7(w6, in[1], nin0);

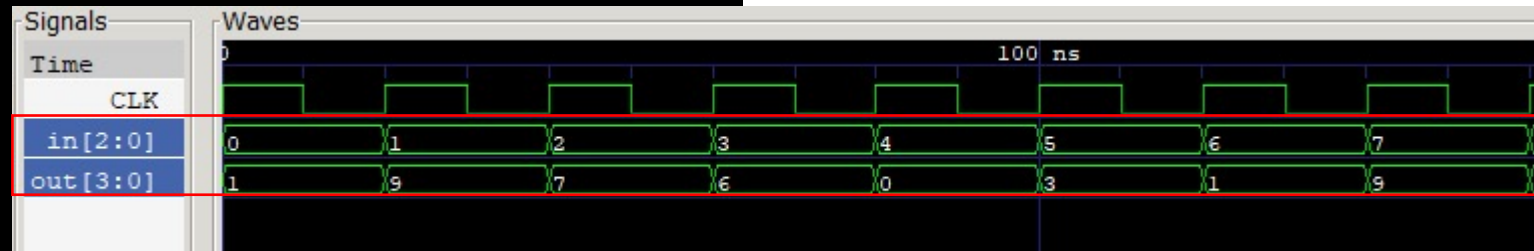
    //使用 or 描述電路圖上的 or gate 與線的關係
    or o0(out[3], w0, w1),
        o1(out[1], w2, w3),
        o2(out[0], w4, w5, w6);

endmodule //結束模組
```

範例練習二 – Structural Modeling (3/3)

4. 開啟命令提示字元並進入當前資料夾
5. 輸入指令 “iverilog -o test DDLab5_tbstructuralex_2019.v” 編譯程式
6. 輸入指令 ” vvp test” 執行程式並確認程式正確地依序輸出19760319
7. 輸入指令 “gtkwave lab5.fsdb” 觀察波形圖

```
C:\Users\user\Downloads\DDLab5_2019\orig\structural example>iverilog -o test DDLab5_tbstructuralex_2019.v  
C:\Users\user\Downloads\DDLab5_2019\orig\structural example>vvp test  
VCD info: dumpfile lab5.fsdb opened for output.  
count = 0, out = 1  
count = 1, out = 9  
count = 2, out = 7  
count = 3, out = 6  
count = 4, out = 0  
count = 5, out = 3  
count = 6, out = 1  
count = 7, out = 9
```



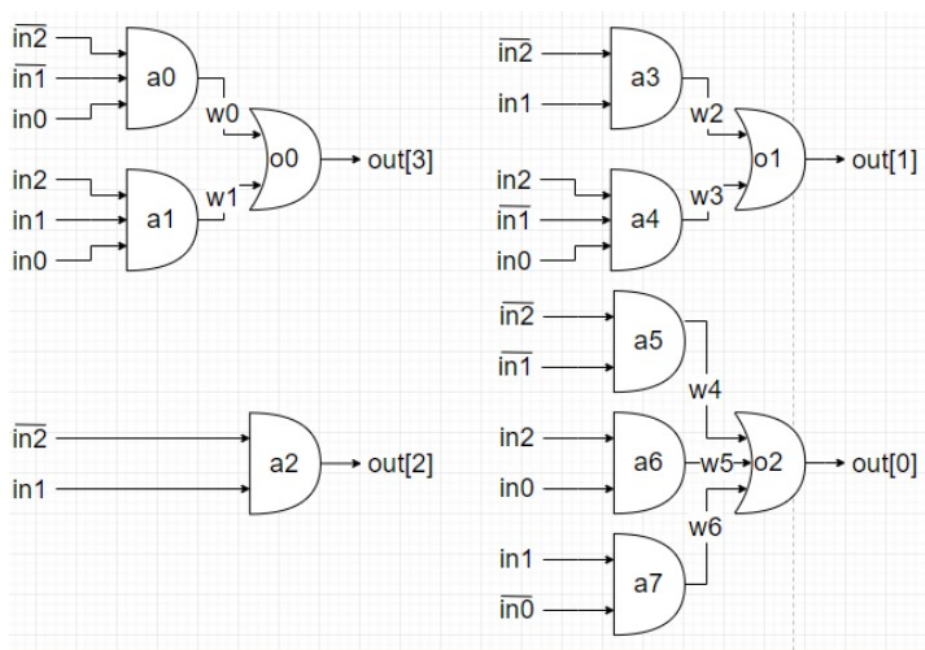
範例練習三 – Continuous Assignment (1/3)

- 將生日碼產生器以 Behavioral Modeling 的 Continuous Assignment 完成，並以 testbench 驗證19760319
- 請同學開啟課程壓縮包內的 DDLab5_continuousex_2019.v

fin10_OS (C:) > 使用者 > user > 下載 > DDLab5_2019 > orig > behavioral example > continuous assignment				
名稱	修改日期	類型	大小	
 DDLab5_continuousex_2019.v	2019/3/28 下午 1...	V 檔案	1 KB	
 DDLab5_tbcontinuousex_2019.v	2019/3/28 下午 1...	V 檔案	1 KB	
 lab5.fsdb	2019/3/28 下午 1...	FSDB 檔案	1 KB	
 test	2019/3/28 下午 1...	檔案	7 KB	

範例練習三 – Continuous Assignment (2/3)

1. 宣告輸入與輸出埠
2. 將生日碼產生器的 SOP 表示式以 assign 及 verilog 運算子描述



```
module lab5(in, out); //模組 lab5 (對應的輸入埠與輸出埠)

    //宣告區段
    input [2:0]in; //宣告輸入埠 in ,其形態為 3bit wire
    output [3:0]out; //宣告輸出埠 out ,其形態為 4bit wire

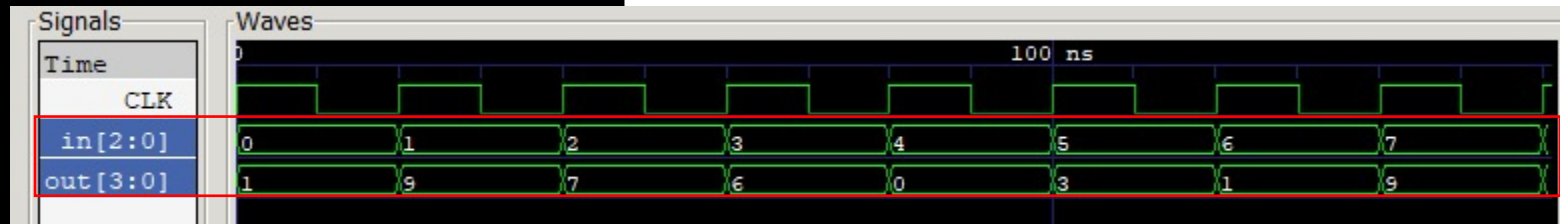
    //邏輯區段
    assign out[3] = (!in[2] & !in[1] & in[0]) | (in[2] & in[1] & in[0]),
           out[2] = (!in[2] & in[1]),
           out[1] = (!in[2] & in[1]) | (in[2] & !in[1] & in[0]),
           out[0] = (!in[2] & !in[1]) | (in[2] & in[0]) | (in[1] & !in[0]);
    //用 assign 敘述描述輸出埠 out 的各個位數值

endmodule //結束模組
```

範例練習三 – Continuous Assignment (3/3)

3. 開啟命令提示字元並進入當前資料夾
4. 輸入指令編譯程式 “iverilog -o test DDLab5_tbcontinuousex_2019.v”
5. 輸入指令執行程式 ” vvp test” 並確認程式正確地依序輸出19760319
6. 輸入指令觀察波形圖 “gtkwave lab5.fsdb”





```
C:\Users\user\Downloads\DDLab5_2019\orig\behavioral example\continuous assignment>iverilog -o test DDLab5_tbcontinuousex_2019.v  
C:\Users\user\Downloads\DDLab5_2019\orig\behavioral example\continuous assignment>vvp test  
VCD info: dumpfile lab5.fsdb opened for output.  
count = 0, out = 1  
count = 1, out = 9  
count = 2, out = 7  
count = 3, out = 6  
count = 4, out = 0  
count = 5, out = 3  
count = 6, out = 1  
count = 7, out = 9
```



範例練習四 – Procedural Assignment (1/3)

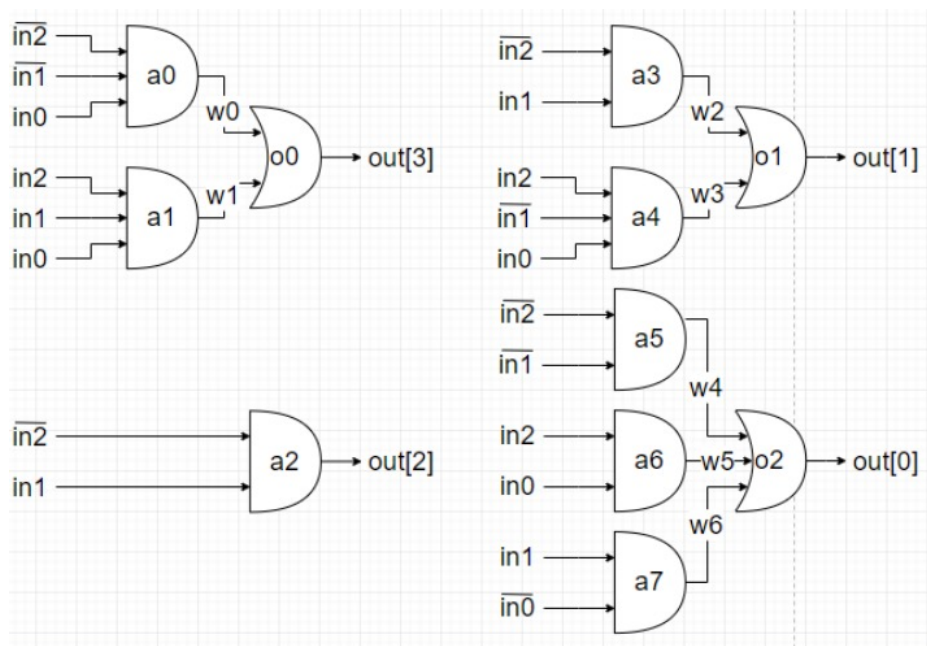
- 將生日碼產生器以 Behavioral Modeling 的 Procedural Assignment 完成，並以 testbench 驗證19760319
- 請同學開啟課程壓縮包內的DDLlab5_alwaysex_2019.v

📁 > 下載 > DDLab5_2019 > orig > behavioral example > procedural assignment

名稱	修改日期	類型	大小
 DDLab5_alwaysex_2019.v	2019/4/1 上午 11...	V 檔案	1 KB
 DDLab5_tbalwaysex_2019.v	2019/3/29 上午 1...	V 檔案	1 KB
 lab5.fsdb	2019/4/1 上午 11...	FSDB 檔案	1 KB
 test	2019/4/1 上午 11...	檔案	6 KB

範例練習四 – Procedural Assignment (2/3)

1. 確認輸入與輸出之間的關係
2. 因always敘述內僅能對 reg 型態賦值，宣告輸入與輸出埠時還須令 out 為 reg
3. 藉由 always 敘述，將structural modeling的電路以 minimal SOP化簡並利用 verilog 運算子描述



```
module lab5(in, out); //模組 lab5(對應的輸入埠與輸出埠)

    //宣告區段
    input [2:0]in; //宣告輸入埠 in ,其形態為 3bit wire
    output reg [3:0]out; //宣告輸出埠 out ,其形態為 4bit reg

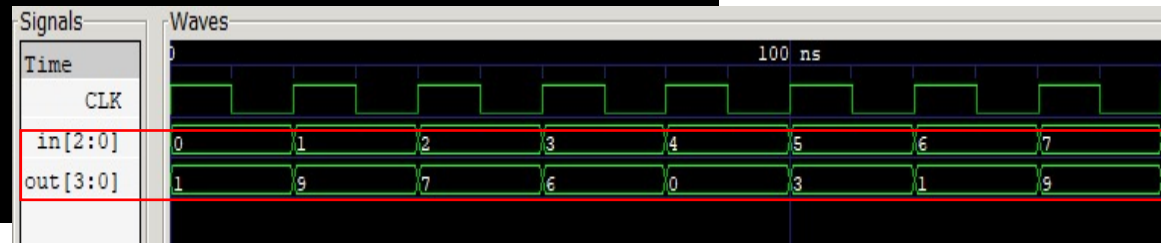
    //邏輯區段
    always @(*)
    //當 always 讀取的變數或線改變時，再次執行 always 內容
    begin //開始描述 always 內容
        out[3] = (!in[2] & !in[1] & in[0]) | (in[2] & in[1] & in[0]);
        //計算輸出埠 out 的第3個位數
        out[2] = (!in[2] & in[1]);
        out[1] = (!in[2] & in[1]) | (in[2] & !in[1] & in[0]);
        out[0] = (!in[2] & !in[1]) | (in[2] & in[0]) | (in[1] & !in[0]);
    end //結束描述 always 內容

endmodule //結束模組
```

範例練習四 – Procedural Assignment (3/3)

4. 開啟命令提示字元並進入當前資料夾
5. 輸入指令 “iverilog -o test DDLab5_tbalwaysex_2019.v” 編譯程式
6. 輸入指令 ” vvp test” 執行程式並確認程式正確地依序輸出19760319
7. 輸入指令 “gtkwave lab5.fsdb” 觀察波形圖

```
C:\Users\user\Downloads\DDLab5_2019\orig\behavioral example\procedural assignment>iverilog -o test DDLab5_tbalwaysex_2019.v  
C:\Users\user\Downloads\DDLab5_2019\orig\behavioral example\procedural assignment>vvp test  
VCD info: dumpfile lab5.fsdb opened for output.  
count = 0, out = 1  
count = 1, out = 9  
count = 2, out = 7  
count = 3, out = 6  
count = 4, out = 0  
count = 5, out = 3  
count = 6, out = 1  
count = 7, out = 9
```



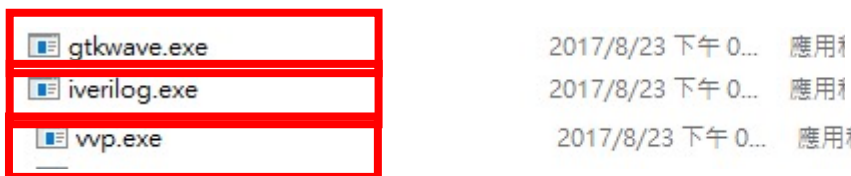
模擬工具 – Icarus Verilog

- 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. 檢查bin資料夾是否有執行檔：iverilog.exe、vvp.exe、gtkwave.exe



MAC版安裝教學：<http://easonchang.logdown.com/posts/649863>

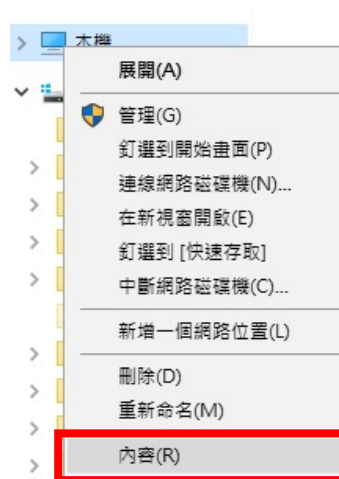
模擬工具 – 設定環境變數 (1/2)

- 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

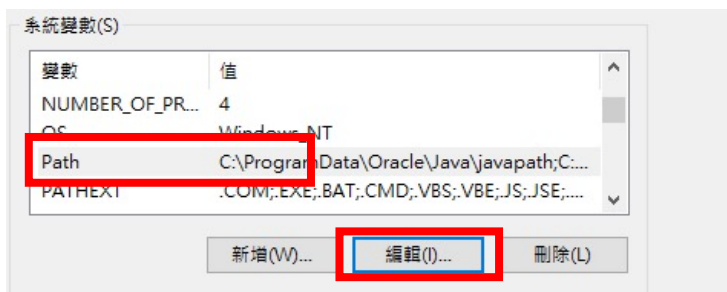


模擬工具 – 設定環境變數 (2/2)

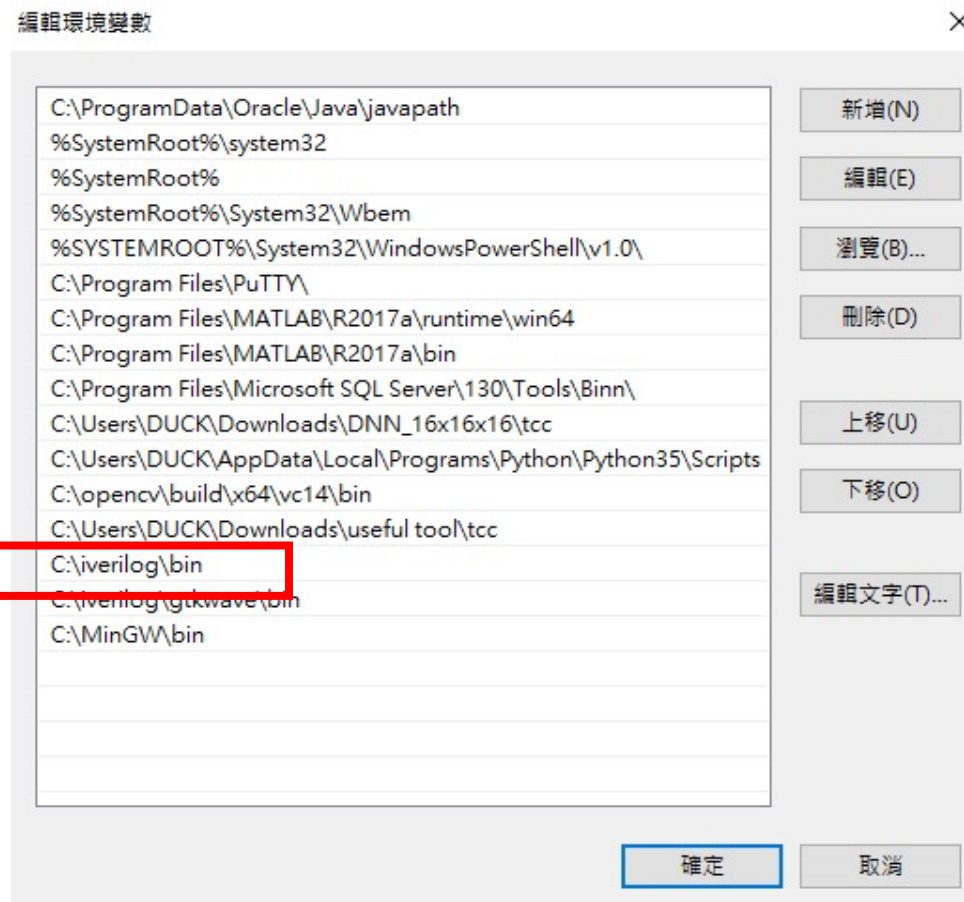
4. 點擊環境變數



5. 選擇系統變數 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

編輯工具 – Notepad++

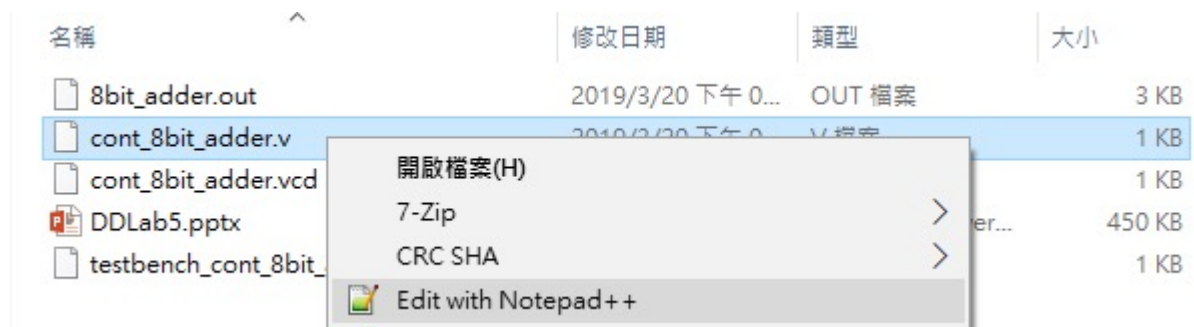


- <https://notepad-plus-plus.org/download/v7.6.4.html>
- 下載Notepad++並解壓縮

Download 64-bit x64

- **Notepad++ Installer 64-bit x64:** Take this one if you have no idea which one you should take.
- **Notepad++ zip package 64-bit x64:** Don't want to use installer? Check this one (zip format).
- **Notepad++ 7z package 64-bit x64:** Don't want to use installer? 7z format.
- **Notepad++ minimalist package 64-bit x64:** No theme, no plugin, no updater, quick download and play directly. 7z format.
- **SHA-256/SHA-1/MD5 digests for binary packages:** Check it to verify the integrity of your Notepad++ download.

- 利用Notepad++編輯.v檔



LAB

- 題目：參考範例練習，以 minimal SOP 完成生日產生器並透過 testbench 顯示自己的生日 (驗收時請附上生日證明文件)
- Demo 需要 Structural Modeling、Continuous Assignment、Procedural Assignment 三個版本

課程評分

- Demo 時間：四梯次時間分別為 19:30、19:50、20:10 與 20:30
- Demo 梯次：依E-Course公布為準
- Demo 地點：工一館 206
- 評分方式：
 - (1)範例一~四成功執行 40%
 - (2)完成Lab作業 40%
 - (3)隨堂練習 20%