Lab 1

Verilog Basics & Simulation (1/2)

助教：張瀚銓、江衍廷

Outline

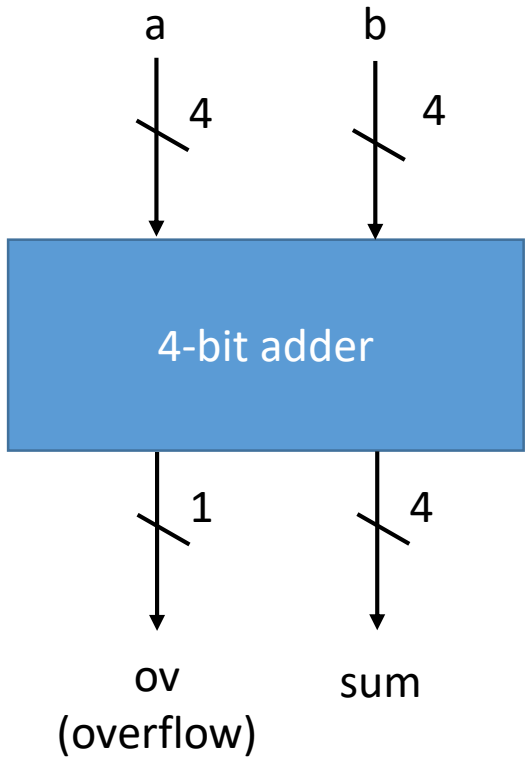
- 課程目的
- 硬體設計與Verilog
- Verilog基本模擬
- 作業

課程目的

- 基本的硬體設計概念
- 如何用Verilog (硬體描述語言) 設計硬體
- 如何完成設計模擬

基本概念

- 硬體設計的第一步，先清楚的定義好I/O後，再來描述input與output的關係
- 定義好I/O，Verilog已經完成一半了
- 如下面4-bit加法器為例:




```
module add2(a, b, sum, ov);

    input [3:0] a, b;
    output [3:0] sum;
    output ov;

    Describe behavior or structure
    of your design

endmodule
```

I/O

Verilog描述方法

- Continuous assignment
- Procedural assignment
- Structural modeling

Continuous Assignment

- output可以用簡單的一行指令描述出來，就用assign

```
module add2(a, b, sum, ov);

    input [3:0] a, b;
    output [3:0] sum;
    output ov;

    assign {ov, sum} = a + b;

endmodule
```

Procedural Assignment

- input沒有變化時，output需要等於原本的值，所以output的data type才需要是暫存器(reg)儲存原本的值，所以要把所有output(等號左邊)的變數都宣告成暫存器(reg)

```
module add2(a, b, sum, ov);

    input [3:0] a, b;
    output [3:0] sum;
    output ov;
    reg [3:0] sum;
    reg ov;

    always @(a or b) begin
        {ov, sum} = a + b;
    end

endmodule
```

Another Example :3-Operand Adder

```
module add3(a, b, c, sum, ov);

    input [3:0] a, b, c;
    output [3:0] sum;
    output ov;
    wire s5, s4;

    assign {s5, s4, sum} = a + b + c;
    assign ov = s5 || s4;

endmodule
```

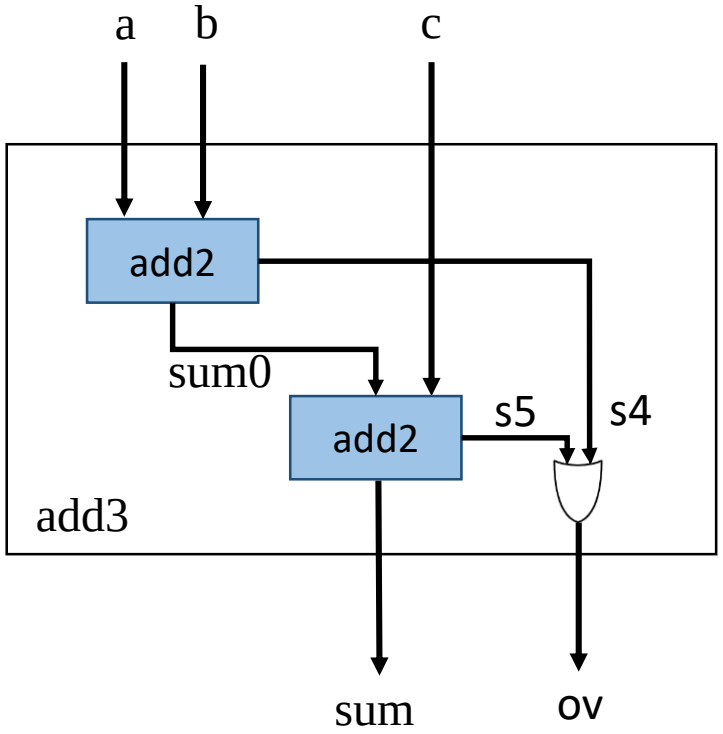
Continuous Assignment

```
module add3(a, b, c, sum, ov);

    input [3:0] a, b, c;
    output [3:0] sum;
    output ov;
    reg [3:0] sum;
    reg ov;
    wire s5, s4;

    always @(a or b or c) begin
        {s5, s4, sum} <= a + b + c;
        ov <= s5 || s4;
    end

endmodule
```

Procedural Assignment

Structural Modeling

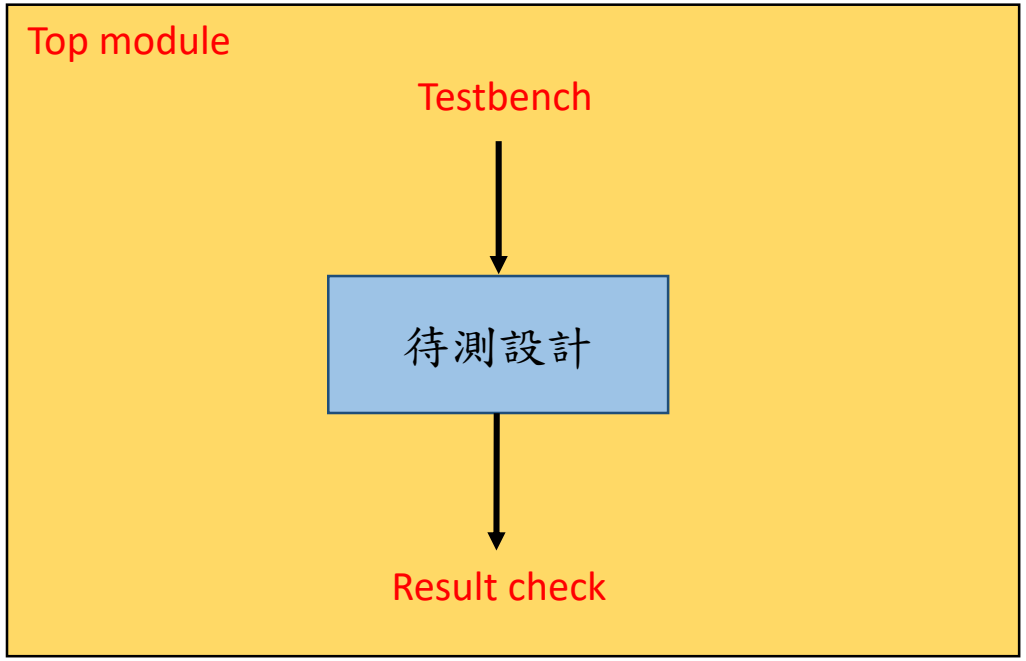
- structural modeling是透過I/O與其他模組連接起來
- 用前面寫好的兩個2-operand adder組成3-operand adder




```
module add3(a, b, c, sum, ov);

    input [3:0] a, b, c;
    output [3:0] sum;
    output ov;
    wire s5, s4;
    wire [3:0] sum0;

    add2 d0(a, b, sum0, s4);
    add2 d1(sum0, c, sum, s5);

    assign ov = s5 || s4;

endmodule

module add2(a, b, sum, ov);

    input [3:0] a, b;
    output [3:0] sum;
    output ov;

    assign {ov, sum} = a + b;

endmodule
```

Verilog 基本模擬

- 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
- 這堂課還不需要會寫testbench，同學只需要大概了解就好



Example: add2

- 我們使用 testbench (testbench_4bit_adder.v) 作為驗證手段，testbench 將輸入信號傳送到4-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
- 利用monitor觀察input和output的結果

```
add2 DUT(a, b, sum, ov);

initial
begin
    clk <= 0;
    rst <= 0;
    a <= 0;
    b <= 0;
end

always@(posedge clk or negedge rst)
begin
    if(a < 16 && b < 16) begin
        a <= a + 1;
    end
    if(a >= 15)begin
        a <= 0;
        b <= b + 1;
    end

end
```

```
initial begin

    $monitor("%4dns monitor: a=%d b=%d sum=%d ov=%d", $stime, a, b, sum, ov);
end
```

```
C:\Users\User\Desktop\iverilog\iverilog\bin>vvp test
   0ns monitor: a= 0 b= 0 sum= 0 ov=0
   1ns monitor: a= 1 b= 0 sum= 1 ov=0
   2ns monitor: a= 2 b= 0 sum= 2 ov=0
   3ns monitor: a= 3 b= 0 sum= 3 ov=0
   4ns monitor: a= 4 b= 0 sum= 4 ov=0
   5ns monitor: a= 5 b= 0 sum= 5 ov=0
   6ns monitor: a= 6 b= 0 sum= 6 ov=0
   7ns monitor: a= 7 b= 0 sum= 7 ov=0
   8ns monitor: a= 8 b= 0 sum= 8 ov=0
   9ns monitor: a= 9 b= 0 sum= 9 ov=0
  10ns monitor: a=10 b= 0 sum=10 ov=0
  11ns monitor: a=11 b= 0 sum=11 ov=0
  12ns monitor: a=12 b= 0 sum=12 ov=0
  13ns monitor: a=13 b= 0 sum=13 ov=0
  14ns monitor: a=14 b= 0 sum=14 ov=0
  15ns monitor: a=15 b= 0 sum=15 ov=0
  16ns monitor: a= 0 b= 1 sum= 1 ov=0
  17ns monitor: a= 1 b= 1 sum= 2 ov=0
  18ns monitor: a= 2 b= 1 sum= 3 ov=0
  19ns monitor: a= 3 b= 1 sum= 4 ov=0
  20ns monitor: a= 4 b= 1 sum= 5 ov=0
  21ns monitor: a= 5 b= 1 sum= 6 ov=0
  22ns monitor: a= 6 b= 1 sum= 7 ov=0
  23ns monitor: a= 7 b= 1 sum= 8 ov=0
  24ns monitor: a= 8 b= 1 sum= 9 ov=0
  25ns monitor: a= 9 b= 1 sum=10 ov=0
  26ns monitor: a=10 b= 1 sum=11 ov=0
  27ns monitor: a=11 b= 1 sum=12 ov=0
  28ns monitor: a=12 b= 1 sum=13 ov=0
  29ns monitor: a=13 b= 1 sum=14 ov=0
  30ns monitor: a=14 b= 1 sum=15 ov=0
```

Example: add3

- 將add2的兩個數相加的testbench改成三個數相加的testbench

```
add3 DUT(a, b, c, sum, ov);

initial
begin
    clk <= 0;
    rst <= 0;
    a <= 0;
    b <= 0;
    c <= 0;
end


always@(posedge clk or negedge rst)
begin
    if(a <= 15) begin
        a <= a + 1;
    end
    if(a == 15) begin
        b <= b + 1;
    end
    if(b == 15 && a == 15) begin
        b <= b + 1;
        c <= c + 1;
    end

end
```

```
initial begin
    $monitor("%4dns monitor: a=%d b=%d c=%d sum=%d ov=%d", $stime, a, b, c, sum, ov);
end
```

```
C:\Users\User\Desktop\iverilog\iverilog\bin>vvp test
   0ns monitor: a= 0 b= 0 c= 0 sum= 0 ov=0
   1ns monitor: a= 1 b= 0 c= 0 sum= 1 ov=0
   2ns monitor: a= 2 b= 0 c= 0 sum= 2 ov=0
   3ns monitor: a= 3 b= 0 c= 0 sum= 3 ov=0
   4ns monitor: a= 4 b= 0 c= 0 sum= 4 ov=0
   5ns monitor: a= 5 b= 0 c= 0 sum= 5 ov=0
   6ns monitor: a= 6 b= 0 c= 0 sum= 6 ov=0
   7ns monitor: a= 7 b= 0 c= 0 sum= 7 ov=0
   8ns monitor: a= 8 b= 0 c= 0 sum= 8 ov=0
   9ns monitor: a= 9 b= 0 c= 0 sum= 9 ov=0
  10ns monitor: a=10 b= 0 c= 0 sum=10 ov=0
  11ns monitor: a=11 b= 0 c= 0 sum=11 ov=0
  12ns monitor: a=12 b= 0 c= 0 sum=12 ov=0
  13ns monitor: a=13 b= 0 c= 0 sum=13 ov=0
  14ns monitor: a=14 b= 0 c= 0 sum=14 ov=0
  15ns monitor: a=15 b= 0 c= 0 sum=15 ov=0
  16ns monitor: a= 0 b= 1 c= 0 sum= 1 ov=0
  17ns monitor: a= 1 b= 1 c= 0 sum= 2 ov=0
  18ns monitor: a= 2 b= 1 c= 0 sum= 3 ov=0
  19ns monitor: a= 3 b= 1 c= 0 sum= 4 ov=0
  20ns monitor: a= 4 b= 1 c= 0 sum= 5 ov=0
  21ns monitor: a= 5 b= 1 c= 0 sum= 6 ov=0
  22ns monitor: a= 6 b= 1 c= 0 sum= 7 ov=0
  23ns monitor: a= 7 b= 1 c= 0 sum= 8 ov=0
  24ns monitor: a= 8 b= 1 c= 0 sum= 9 ov=0
  25ns monitor: a= 9 b= 1 c= 0 sum=10 ov=0
  26ns monitor: a=10 b= 1 c= 0 sum=11 ov=0
  27ns monitor: a=11 b= 1 c= 0 sum=12 ov=0
  28ns monitor: a=12 b= 1 c= 0 sum=13 ov=0
  29ns monitor: a=13 b= 1 c= 0 sum=14 ov=0
  30ns monitor: a=14 b= 1 c= 0 sum=15 ov=0
```

Step-by-Step 範例

1. 將要執行的design和testbench準備好
2. 修改testbench，選擇要對input丟甚麼數字
3. 輸入指令進行編譯
4. 執行程式察看結果

Step 1

- 將寫好的design(add3.v)和testbench(testbench_4bit_adder.v)放在同一個資料夾

Step 2

- 修改testbench，選擇要對input丟甚麼數字
- 這堂課還不需要會寫testbench，只要知道要怎麼對input丟想要的輸入就好
- 這裡我們列出所有可能的輸入

```
add3 DUT(a, b, c, sum, ov);
```

在testbench呼叫我們的design

```
initial
begin
    clk <= 0;
    rst <= 0;
    a <= 0;
    b <= 0;
    c <= 0;
end

always@(posedge clk or negedge rst)
begin
    if(a <= 15) begin
        a <= a + 1;
    end
    if(a == 15) begin
        b <= b + 1;
    end
    if(b == 15 && a == 15) begin
        b <= b + 1;
        c <= c + 1;
    end
end
```

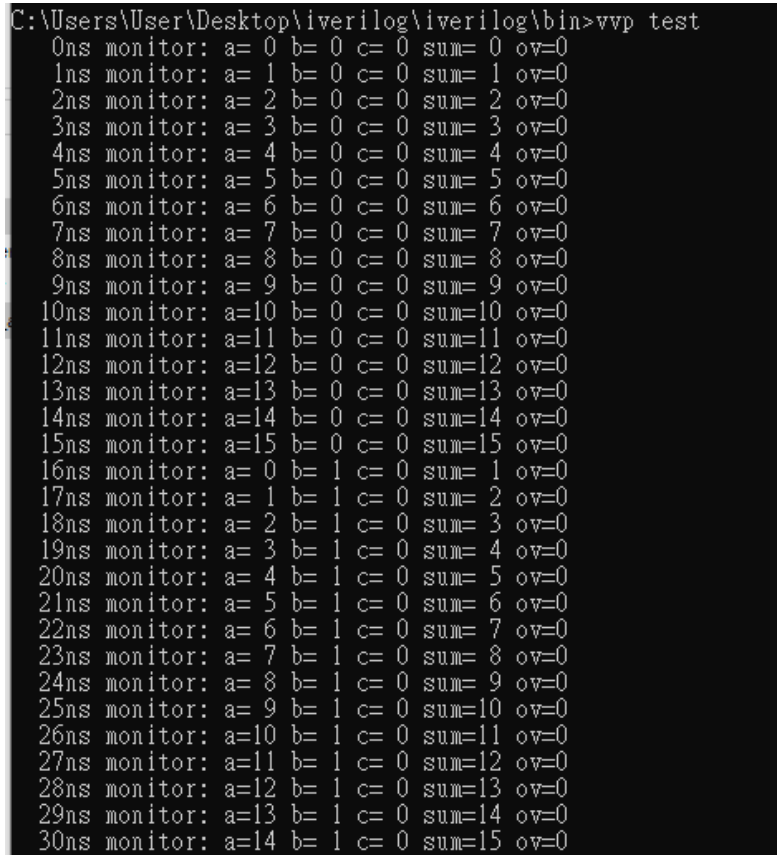
我們要丟入的input

Step 3

- 在程式檔案路徑欄位打開命令提示字元，輸入指令進行編譯
  - **iverilog –o test add3.v testbench_4bit_adder.v**

編譯後生成的檔案

- 執行程式察看結果
  - **vvp test**



LAB

- 題目：參考範例練習，修改成4個4-bit數相加並透過我們提供的testbench 顯示答案
- Demo時需要按照以下格式秀出程式碼，並且執行成功畫面

```
module adder(sum, a, b, c, d, vo);
    input [3:0] a, b, c, d;
    output [3:0] sum;
    output vo;

    Describe behavior or structure of
    your design

endmodule
```

```
C:\Users\User\Desktop\iverilog\iverilog\bin>vvp test
   0ns monitor: a= 0 b= 0 c= 0 d= 0 sum= 0 ov=0
   1ns monitor: a= 1 b= 0 c= 0 d= 4 sum= 5 ov=0
   2ns monitor: a= 2 b= 0 c= 0 d= 1 sum= 3 ov=0
   3ns monitor: a= 3 b= 0 c= 0 d= 9 sum=12 ov=0
   4ns monitor: a= 4 b= 0 c= 0 d= 3 sum= 7 ov=0
   5ns monitor: a= 5 b= 0 c= 0 d=13 sum= 2 ov=1
   6ns monitor: a= 6 b= 0 c= 0 d=13 sum= 3 ov=1
   7ns monitor: a= 7 b= 0 c= 0 d= 5 sum=12 ov=0
   8ns monitor: a= 8 b= 0 c= 0 d= 2 sum=10 ov=0
   9ns monitor: a= 9 b= 0 c= 0 d= 1 sum=10 ov=0
  10ns monitor: a=10 b= 0 c= 0 d=13 sum= 7 ov=1
  11ns monitor: a=11 b= 0 c= 0 d= 6 sum= 1 ov=1
  12ns monitor: a=12 b= 0 c= 0 d=13 sum= 9 ov=1
  13ns monitor: a=13 b= 0 c= 0 d=13 sum=10 ov=1
  14ns monitor: a=14 b= 0 c= 0 d=12 sum=10 ov=1
  15ns monitor: a=15 b= 0 c= 0 d= 9 sum= 8 ov=1
  16ns monitor: a= 0 b= 1 c= 0 d= 6 sum= 7 ov=0
  17ns monitor: a= 1 b= 1 c= 0 d= 5 sum= 7 ov=0
  18ns monitor: a= 2 b= 1 c= 0 d=10 sum=13 ov=0
  19ns monitor: a= 3 b= 1 c= 0 d= 5 sum= 9 ov=0
  20ns monitor: a= 4 b= 1 c= 0 d= 7 sum=12 ov=0
  21ns monitor: a= 5 b= 1 c= 0 d= 2 sum= 8 ov=0
  22ns monitor: a= 6 b= 1 c= 0 d=15 sum= 6 ov=1
  23ns monitor: a= 7 b= 1 c= 0 d= 2 sum=10 ov=0
  24ns monitor: a= 8 b= 1 c= 0 d=14 sum= 7 ov=1
  25ns monitor: a= 9 b= 1 c= 0 d= 8 sum= 2 ov=1
  26ns monitor: a=10 b= 1 c= 0 d= 5 sum= 0 ov=1
  27ns monitor: a=11 b= 1 c= 0 d=12 sum= 8 ov=1
  28ns monitor: a=12 b= 1 c= 0 d=13 sum=10 ov=1
  29ns monitor: a=13 b= 1 c= 0 d=13 sum=11 ov=1
  30ns monitor: a=14 b= 1 c= 0 d= 5 sum= 4 ov=1
```

課程評分

- Demo 時間：依E-Course公布為準
- Demo 梯次：依E-Course公布為準
- Demo 地點：依E-Course公布為準

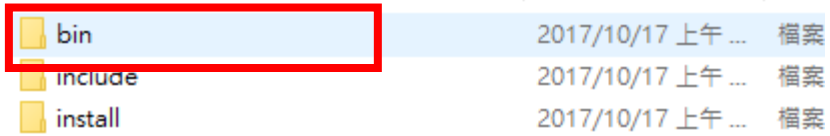
- 評分方式
  1. 成功執行step-by-step 範例程式40%
  2. 完成Lab作業 60%

環境安裝

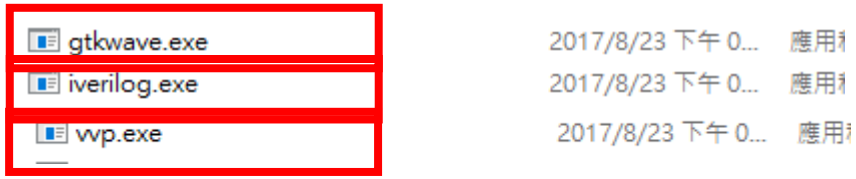
模擬工具 – Icarus Verilog

- 在本次實驗課，同學將使用 Icarus Verilog 的 iverilog、vvp、gtkwave 來模擬及觀測 4-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

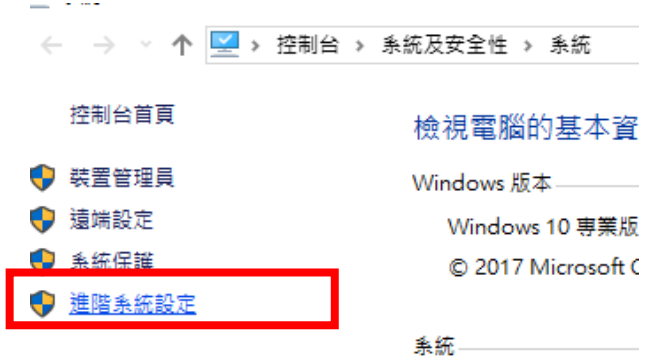
模擬工具 – 設定環境變數 (1/2)

- 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

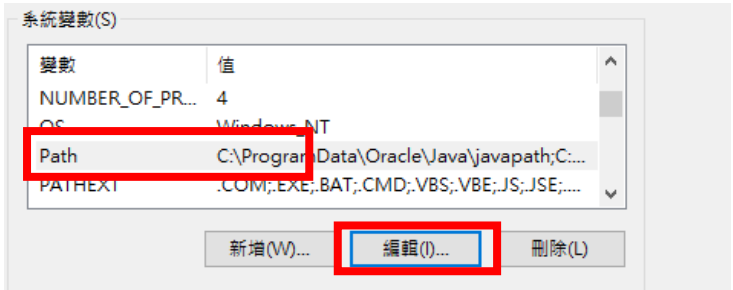
3. 點擊進階系統設定



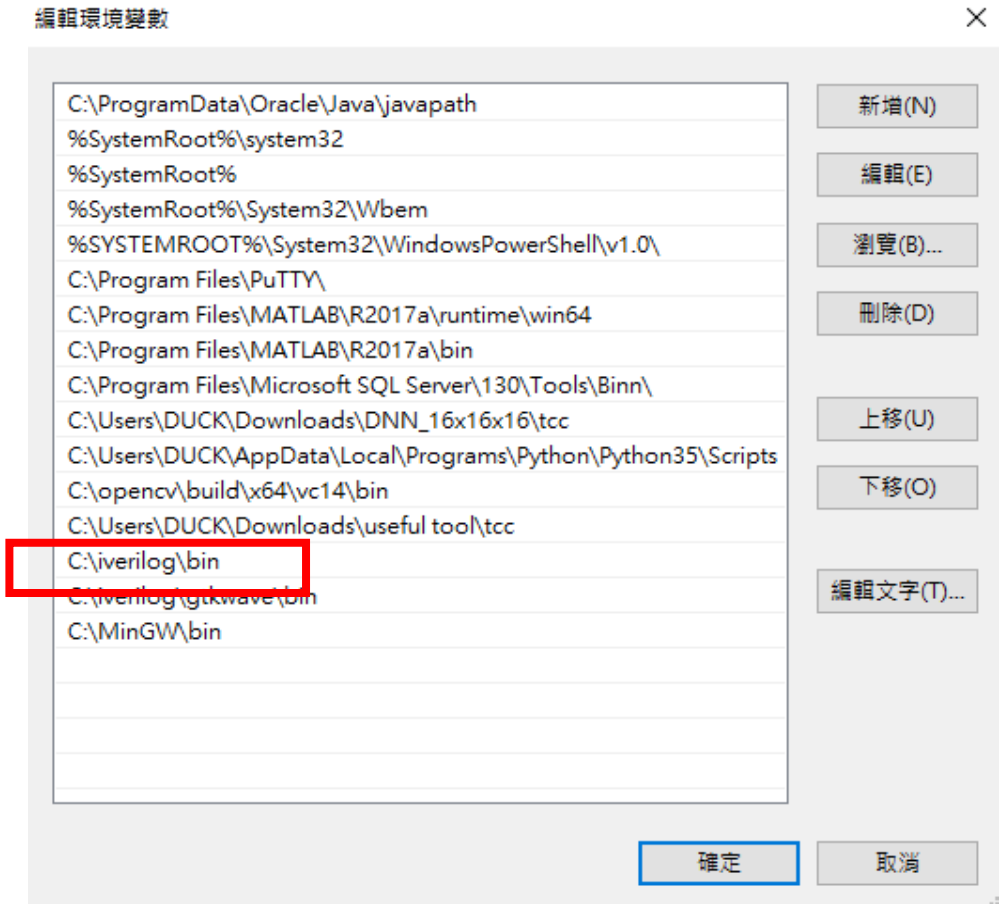
模擬工具 – 設定環境變數 (2/2)

4. 點擊環境變數

5. 選擇系統變數 點擊path並按下編輯



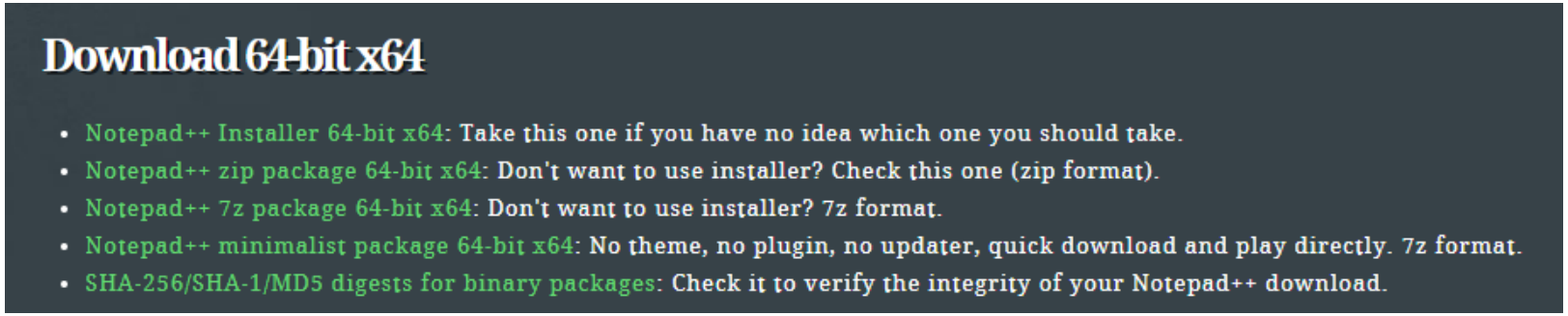
6. 新增並輸入bin資料夾路徑，按下確定



※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



- 利用Notepad++編輯.v檔