



# 邏輯系統實習

## 實驗四之二

**Xilinx ISE之介紹與操作 + Verilog語法介紹(一)：邏輯閘層次**

國立成功大學 電機系

2016

# 大綱

- Verilog的發展歷史
- Verilog中的四種描述層次
- 模組的觀念
- 接線與暫存器
- 階層化模組的設計
  - 輸入埠與外部訊號的連接方法
- 輸出入埠的連接規定
- 數字規格
- 延遲與註解
- 系統任務
- 測試平台 (testbench)
- Verilog 編碼風格
- 數位電路設計流程
- Xilinx ISE操作
  - 建立新專案
  - 建立電路原始碼
  - 編譯電路
  - 建立測試平台原始碼
  - 檢查測試平台語法
  - 合成前模擬
  - 波形驗證
- 基礎題 (一)
  - 四對一多工器
- 基礎題 (二)
  - 全加器
- 挑戰題
  - 4bits 漣波加法器
- 實驗結報繳交

# Verilog的發展歷史

- 1984 Gateway Design Automation, Phil Moorby
- 1986 Verilog-XL: an efficient gate-level simulator
- 1988 Verilog logic synthesizer, Synopsys
- 1989 Cadence Data System Inc. acquired Gateway
- 1990 Verilog HDL is released to public domain
- 1991 Open Verilog International (OVI)
- 1994 IEEE 1364 Working group
- 1995 December : Verilog becomes an IEEE standard (IEEE Std. 1364)
- 2001 SystemVerilog

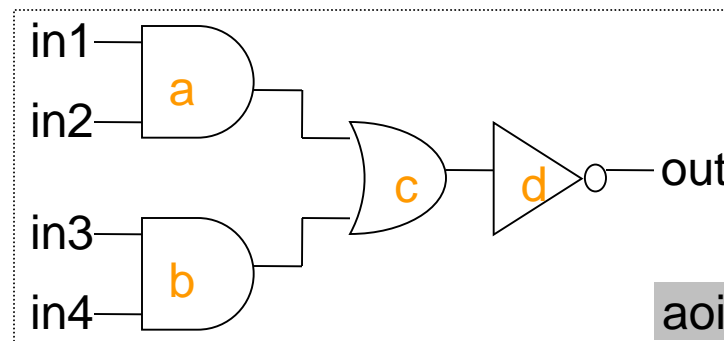
# Verilog中的四種描述層次

- 行為或演算法層次 (lab6~)
  - 在這個層次中，只需要考慮模組的功能或演算法，不需要考慮硬體方面的詳細電路是如何。
- 資料處理層次 (lab5)
  - 在這個層次中，只需要指名資料處理的方式，像是資料如何在暫存器中儲存與傳送以及資料在設計裡處理的方式。
- 邏輯閘層次 (lab4)
  - 在這個層次中，模組是邏輯閘連接而成，如同以前用邏輯閘描繪電路一樣。
- 低階交換層次 (x)
  - 在這個層次中，線路是由開關與儲存點構成，需要知道電晶體的元件特性。

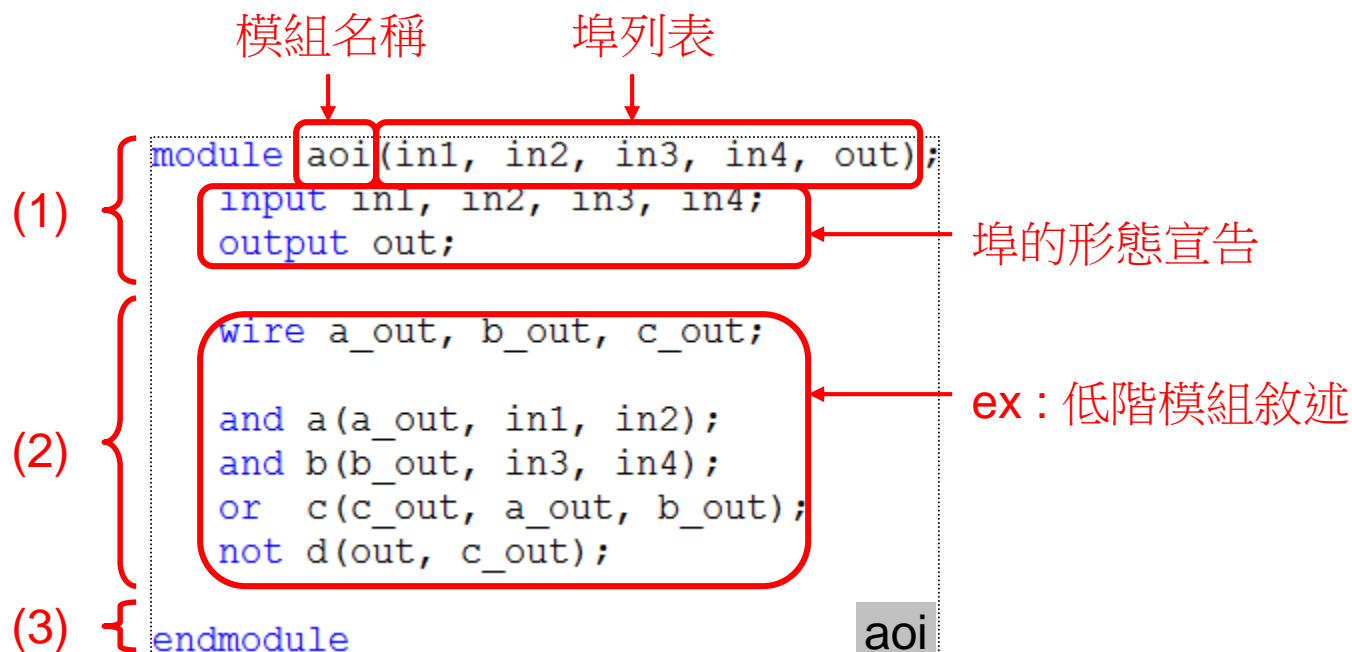
越上層設計層次越高

# 模組的觀念

- 1) 模組名稱、埠列表、埠的形態宣告、參數。
- 2) 接線(wire)、暫存器(reg)、變數、低階模組敘述、資料流敘述(assign)、行為模式敘述(initial、always)。



- 3) 結尾



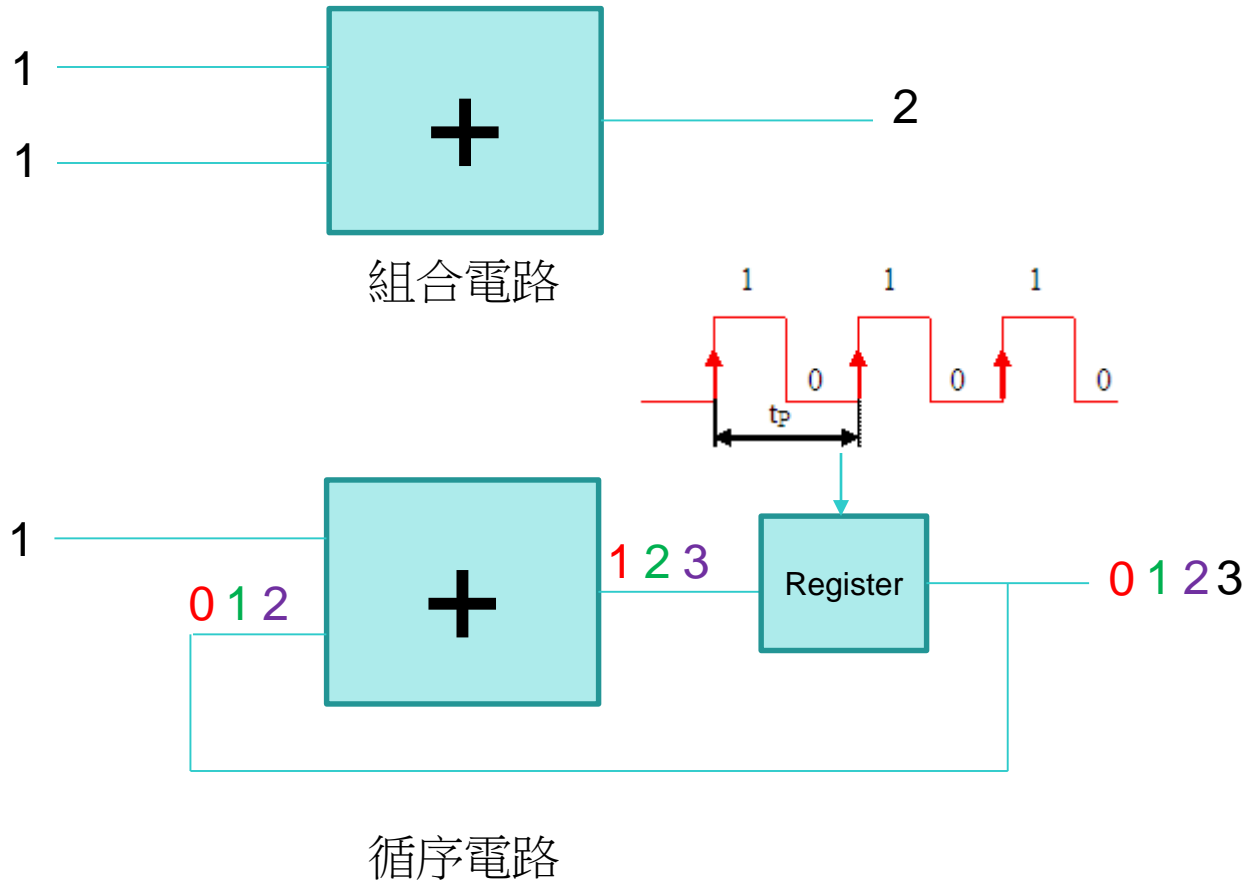
# 接線與暫存器

- 接線 (wire)
  - 接線是連接硬體元素的點，預設為一個bit。
- 暫存器 (reg)
  - 暫存器是用來表示資料儲存的元素，除非給定新值，否則暫存器內的數值會一直維持，預設為一個bit。
  - 此暫存器(reg)與硬體中的暫存器(register)不同。
- 向量
  - 接線與暫存器接可定義為向量，定義為[high# : low#]，左邊為MSB。
  - ex : wire [11:0]addr;
  - ex : reg [31:0]data;
- 向量子集合選取
  - 針對上面的例子，選取其中的一分子集合，範例如下。
  - ex : wire [15:0]data\_1 = data[15:0];
  - ex : wire [15:0]data\_2 = data[31:16];

# 接線與暫存器(2)

- 什麼時候該用wire?什麼時候又該用reg?
  - 但若此變數要放在begin...end內，該變數就須使用reg，在begin...end之外，則使用wire。
  - 另外使用wire時，須搭配assign；reg則不必。
  - input，output預設值都是wire。
  - 若wire和reg用錯地方，compiler都會提醒，所以不必太擔心。
- 一個很重要的觀念，在Verilog中使用reg，並不表示合成後就是暫存器(register)。
  - 若在組合電路中使用reg，合成後仍只是net，
  - 唯有在循序電路中使用reg，合成後才會以flip-flop形式表示成register。

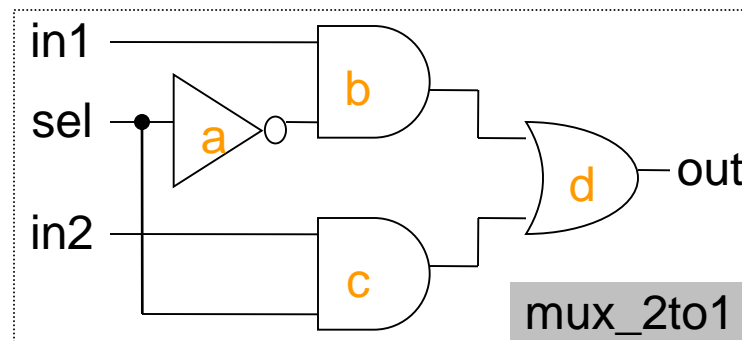
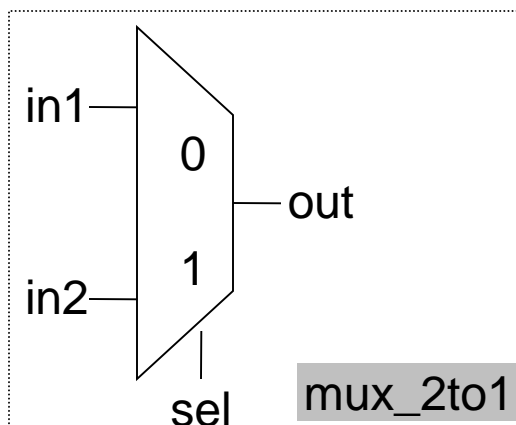
# 接線與暫存器(3)





# 階層化模組的設計 (1/3)

- 利用邏輯閘層次語法寫出一個二對一的多工器。
- 邏輯閘的種類。
  - Verilog事先定義好的邏輯模型。
    - not、and、nand、or、nor
    - xor、xnor
  - 當邏輯閘的輸入多於兩個時。
    - ex : nand n\_4i(out, in1, in2, in3, in4);



```
module mux_2to1(in1, in2, sel, out);
    input in1, in2, sel;
    output out;

    wire a_out, b_out, c_out;

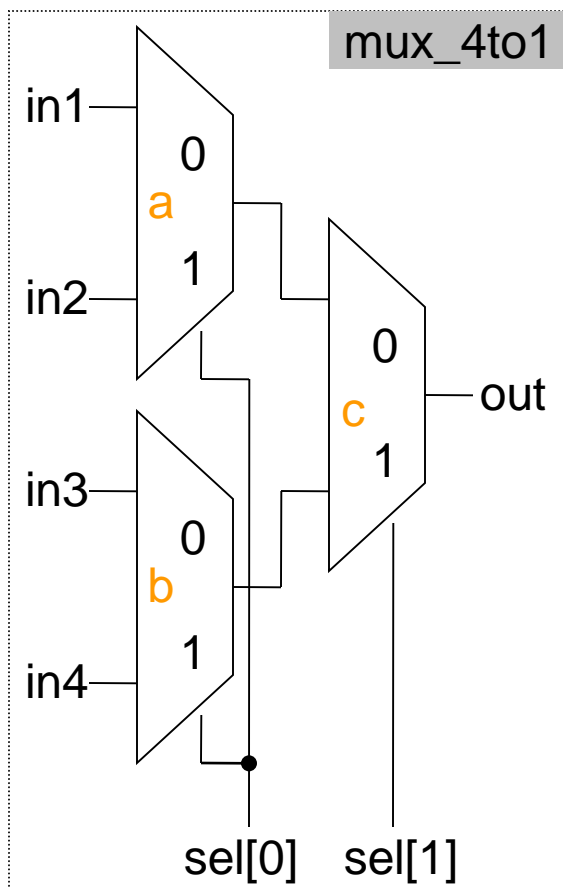
    not a(a_out, sel);
    and b(b_out, in1, a_out);
    and c(c_out, in2, a_out);
    or d(out, b_out, c_out);

endmodule
```

mux\_2to1

## 階層化模組的設計 (2/3)

- 利用二對一的多工器模組寫出四對一多工器。



```
module mux_2to1(in1, in2, sel, out);  
    input in1, in2, sel;  
    output out;  
  
    wire a_out, b_out, c_out;  
  
    not a(a_out, sel);  
    and b(b_out, in1, a_out);  
    and c(c_out, in2, sel);  
    or d(out, b_out, c_out);  
  
endmodule
```

mux\_2to1

```
module mux_4to1(in1, in2, in3, in4, sel, out);  
    input in1, in2, in3, in4;  
    input [1:0]sel;  
    output out;  
  
    wire a_out, b_out, c_out;  
  
    mux_2to1 a(in1, in2, sel[0], a_out);  
    mux_2to1 b(in3, in4, sel[0], b_out);  
    mux_2to1 c(a_out, b_out, sel[1], out);  
  
endmodule
```

mux\_4to1

# 階層化模組的設計 (3/3)

## 輸入埠與外部訊號的连接方法

- 1) 依照定義模組時輸出入埠的列表順序來連接。
  - 如上一頁的mux\_4to1。
- 2) 用指定名稱的方法來連接。
  - 如下圖的mux\_4to1\_v2。

```
module mux_2to1(in1, in2, sel, out);  
    input in1, in2, sel;  
    output out;  
  
    wire a_out, b_out, c_out;  
  
    not a(a_out, sel);  
    and b(b_out, in1, a_out);  
    and c(c_out, in2, sel);  
    or d(out, b_out, c_out);  
  
endmodule
```

mux\_2to1

```
module mux_4to1_v2(in1, in2, in3, in4, sel, out);  
    input in1, in2, in3, in4;  
    input [1:0]sel;  
    output out;  
  
    wire a_out, b_out, c_out;  
  
    mux_2to1 a(.in1(in1), .in2(in2), .sel(sel[0]), .out(a_out));  
    mux_2to1 b(.in1(in3), .in2(in4), .sel(sel[0]), .out(b_out));  
    mux_2to1 c(.in1(a_out), .in2(b_out), .sel(sel[1]), .out(out));  
  
endmodule
```

mux\_4to1

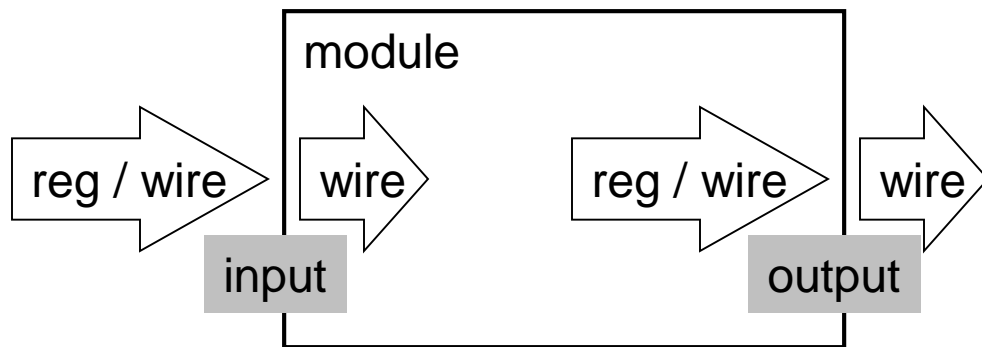
# 輸出入埠的連接規定

## ■ 輸入

- 模組內部：輸入永遠是接線。
- 模組外部：輸入可以是暫存器或接線。

## ■ 輸出

- 模組內部：輸出可以是暫存器或接線。
- 模組外部：輸出永遠是接線。



# 數字規格

- 規定長度之數字 `<size>'<base format><number>`
  - `<size>` : 以十進位表示數字的bit數。
  - `<base format>` : 定義此數為二進位('b或 'B)、八進位('o或 'O)、十進位('d或 'D)、十六進位('h或 'H)。
  - ex : `4'b0111`; //4bit的二進位數
  - ex : `12'habc`; //12bit的十六進位數
- 不定長度之數字
  - 若省略`<size>`等效於模擬器內定規格(ex : 32bit)。
  - 若省略`<base format>`等效於使用十進制。
  - ex : `23456`; //為32bit的十進位數
  - ex : `'hc3`; //為32bit的十六進位數
- 底線
  - 用於增加數值的可讀性。
  - ex : `12'1111_0000_1010`;

# 延遲與註解

## ■ 延遲

- ex :     a = 1'b0;

- #5 b = 1'b1; //在a被指定為0的5個單位時間之後，b才被指定為1。

## ■ 註解

- 可以為單行註解(//...)。

- 或多行註解(/\*...\*/)。

# 系統任務

## ■ \$display

- 用於顯示變數值或字串內容(只執行一次)，類似C語言中的printf。
- ex : \$display("Hello World");

## ■ \$monitor

- 用於監控訊號變化，當被監控的訊號有變化時，便會輸出新的值(觸發就執行)。
- ex : \$monitor("address = %b", addr);

## ■ \$stop

- 暫時終止模擬運算的進行。

## ■ \$finish

- 結束模擬運算。

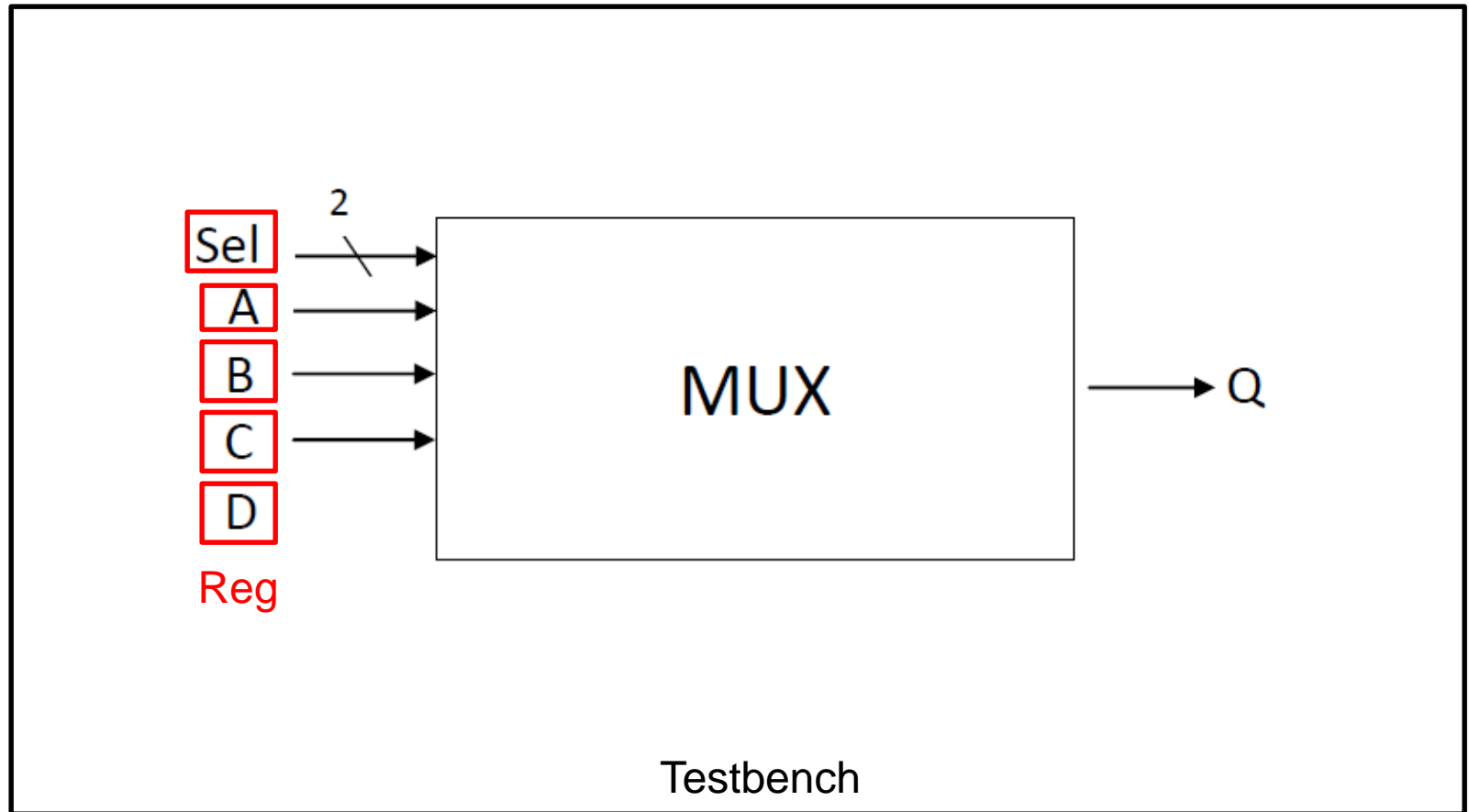
## ■ \$time

- 回傳目前的模擬時間。

格式定義表

%b 或 %B	二進制變數
%o 或 %O	八進制變數
%d 或 %D	十進制變數
%h 或 %H	十六進制變數
%f 或 %F	十進制實數變數
%c 或 %C	ASCII字元
%s 或 %S	字串

# 測試平台 (*testbench*)(1/3)





# 測試平台 (testbench)(2/3)

- 寫一個 mux\_4to1 的測試平台。

```
module testbench();  
    reg in1, in2, in3, in4;  
    reg [1:0] sel;  
    wire out;  
  
    mux_4to1 x1(in1, in2, in3, in4, sel, out);
```

行為層次的語法，一個 **initial** 區塊啟動於模擬時間零，而且僅執行一次

產生輸入訊號

```
    initial begin  
        sel=2'b00; in1=1'b1; in2=1'b0; in3=1'b0; in4=1'b0;  
        #1 sel=2'b00; in1=1'b0; in2=1'b1; in3=1'b1; in4=1'b1;  
  
        #1 sel=2'b01; in1=1'b0; in2=1'b1; in3=1'b0; in4=1'b0;  
        #1 sel=2'b01; in1=1'b1; in2=1'b0; in3=1'b1; in4=1'b1;  
  
        #1 sel=2'b10; in1=1'b0; in2=1'b0; in3=1'b1; in4=1'b0;  
        #1 sel=2'b10; in1=1'b1; in2=1'b1; in3=1'b0; in4=1'b1;  
  
        #1 sel=2'b11; in1=1'b0; in2=1'b0; in3=1'b0; in4=1'b1;  
        #1 sel=2'b11; in1=1'b1; in2=1'b1; in3=1'b1; in4=1'b0;  
  
        #1 $finish;  
    end
```

印出輸出結果

```
    initial begin  
        $monitor($time, " sel=%d, in1=%b, in2=%b, in3=%b, in4=%b, out=%b",  
            sel, in1, in2, in3, in4, out);  
    end
```

```
endmodule
```

testbench

# 測試平台 (testbench)(3/3)

```
module testbench();
    reg in1, in2, in3, in4;
    reg [1:0] sel;

0 sel=0, in1=1, in2=0, in3=0, in4=0, out=1
1 sel=0, in1=0, in2=1, in3=1, in4=1, out=0
2 sel=1, in1=0, in2=1, in3=0, in4=0, out=1 in1, in2, in3, in4, sel, out);
3 sel=1, in1=1, in2=0, in3=1, in4=1, out=0
4 sel=2, in1=0, in2=0, in3=1, in4=0, out=1 in
5 sel=2, in1=1, in2=1, in3=0, in4=1, out=0 b00; in1=1'b1; in2=1'b0; in3=1'b0; in4=1'b0;
6 sel=3, in1=0, in2=0, in3=0, in4=1, out=1 b00; in1=1'b0; in2=1'b1; in3=1'b1; in4=1'b1;
7 sel=3, in1=1, in2=1, in3=1, in4=0, out=0
    #1 sel=2'b01; in1=1'b0; in2=1'b1; in3=1'b0; in4=1'b0;
    #1 sel=2'b01; in1=1'b1; in2=1'b0; in3=1'b1; in4=1'b1;

    #1 sel=2'b10; in1=1'b0; in2=1'b0; in3=1'b1; in4=1'b0;
    #1 sel=2'b10; in1=1'b1; in2=1'b1; in3=1'b0; in4=1'b1;

    #1 sel=2'b11; in1=1'b0; in2=1'b0; in3=1'b0; in4=1'b1;
    #1 sel=2'b11; in1=1'b1; in2=1'b1; in3=1'b1; in4=1'b0;

    #1 $finish;
end

    { initial begin
        $monitor($time, " sel=%d, in1=%b, in2=%b, in3=%b, in4=%b, out=%b",
            sel, in1, in2, in3, in4, out);
    }
end

endmodule
```

模擬後，主控  
台的螢幕顯示  
結果

testbench

# Verilog 編碼風格

## ■ 命名風格

- 1) 使用小寫描述I/O埠、訊號、變數；使用大寫描述常數。
  - ex : input data\_in;
  - ex : `define WIDTH 32
- 2) 使用有意義的命名。
  - ex : data\_in代表資料輸入；a、b代表...意義不明？
- 3) 使用clk命名clock；使用rst命名reset。

## ■ 撰寫風格

- 1) 適當的使用縮排與空白，以增進原始碼的可讀性。

# 數位電路設計流程

- 1) 設計 (lab4 - Xilinx ISE)
- 2) 編譯 (lab4 - Xilinx ISE)
- 3) (合成前)模擬 (lab4 - ISim)
- 4) (合成後)驗證 (lab5 - VeriComm)
- 5) (合成後)驗證 (lab6 - VeriInstrument)

# Xilinx ISE操作 (1/10)

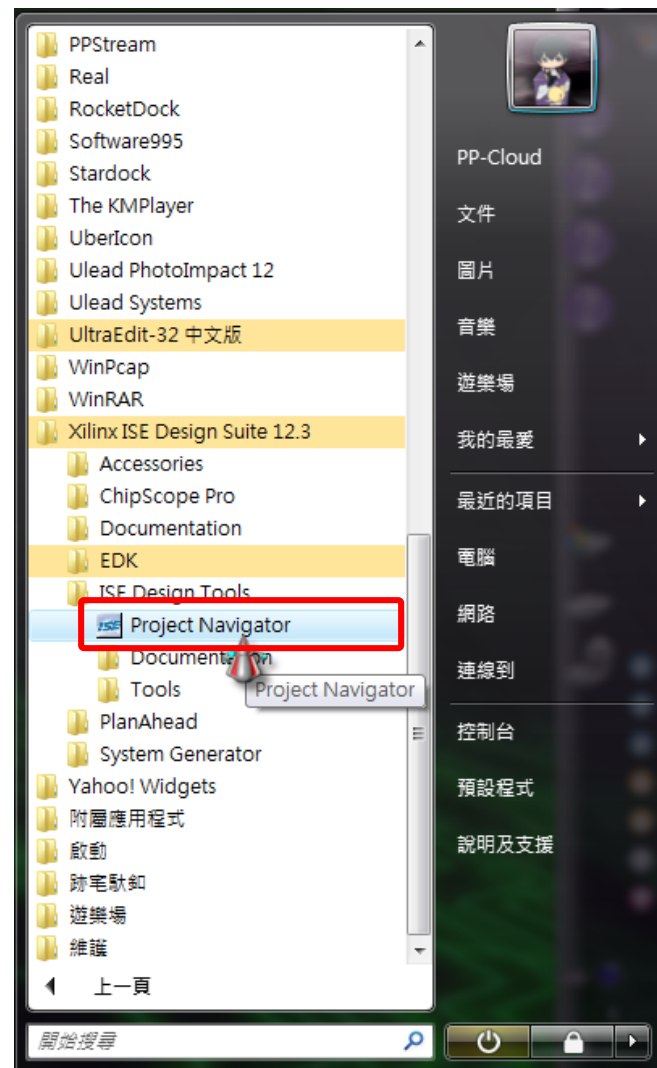
- 1) 請先在D槽建立一個你們的專屬資料夾。
  - ex : jou logic system lab wed或jou logic system lab thu。
- 2) 然後請在你們的專屬資料夾內建立lab04~lab12的專屬資料夾。
  - ex : lab04 ~ lab12。

# Xilinx ISE操作 (2/10)

- Xilinx ISE允許我們使用圖形、狀態圖、VHDL與Verilog等方式來設計電路。

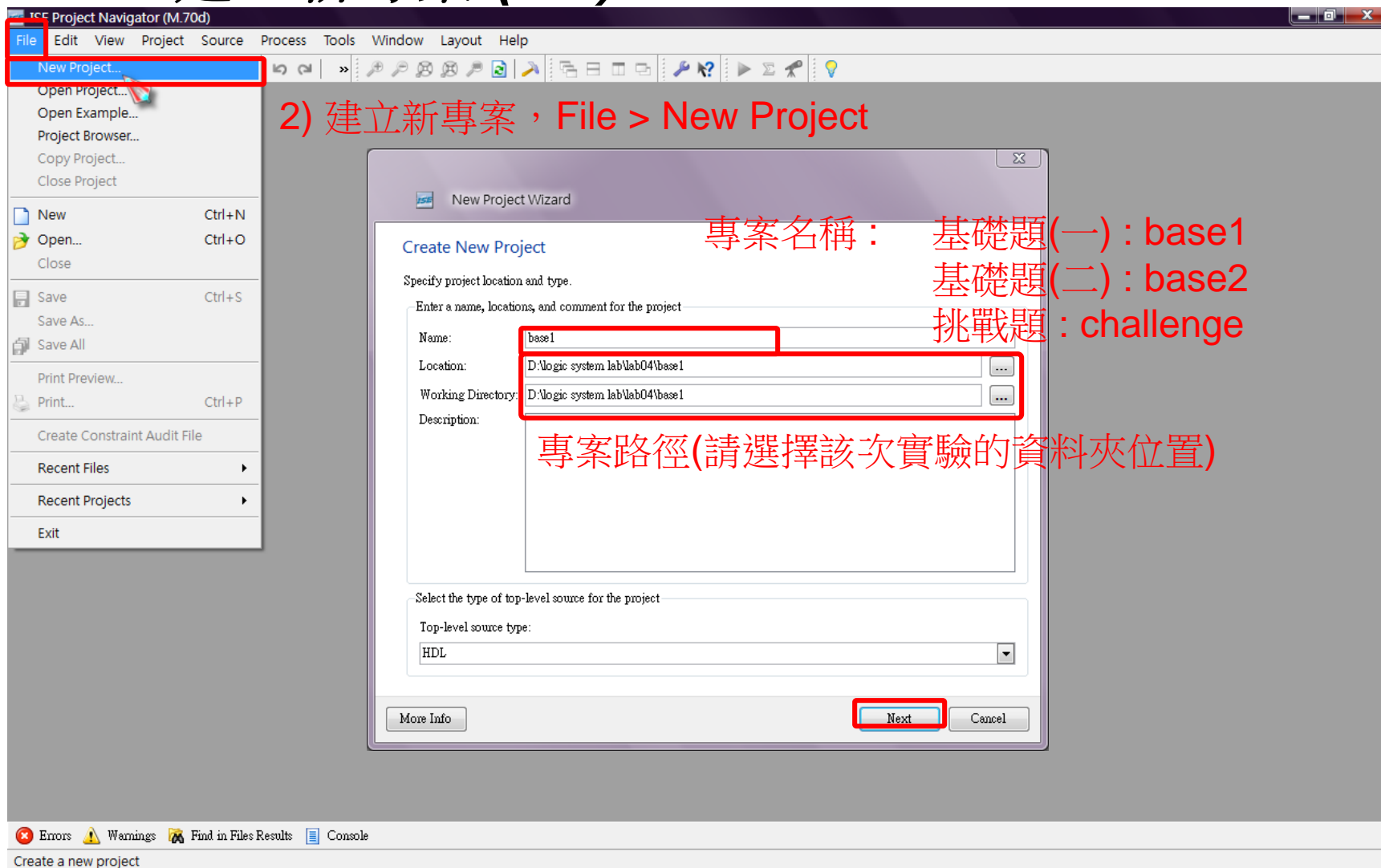


1) 開始 > Xilinx ISE Design Suite12.3 > ISE Design Tools > Project Navigator



# Xilinx ISE操作 (3/10)

## 建立新專案 (1/3)



# Xilinx ISE操作 (3/10)

## 建立新專案 (2/3)



New Project Wizard

Project Settings

Specify device and project properties.  
Select the device and design flow for the project

Property Name	Value
Product Category	All
Family	Spartan6
Device	XC6SLX25
Package	FTG256
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	<input type="checkbox"/>

3) 選擇FPGA板型號：  
VeriLite Spartan-6 XC65LX25  
FTG256

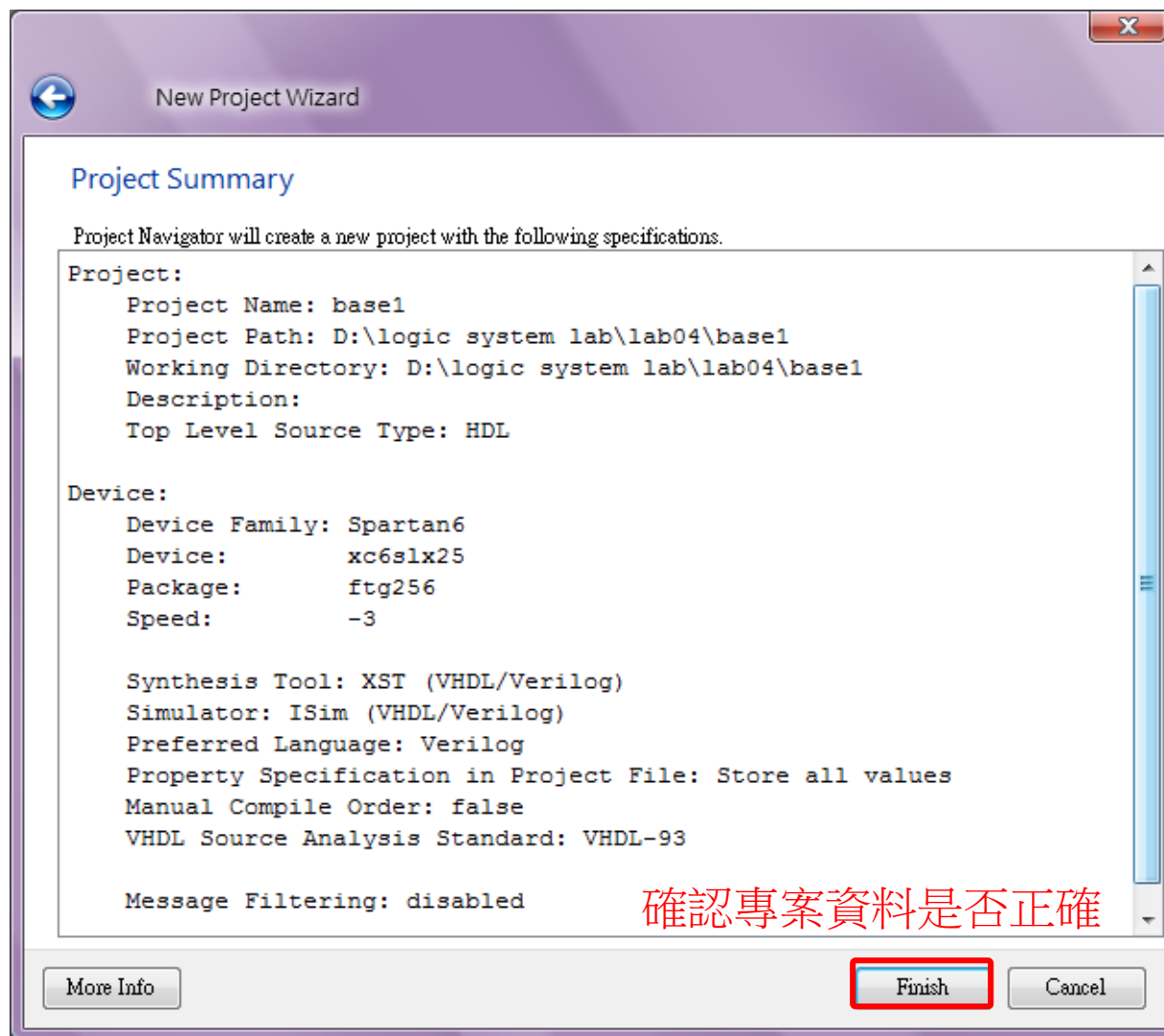
其他的也確認看看有沒有被改過

More Info Next Cancel



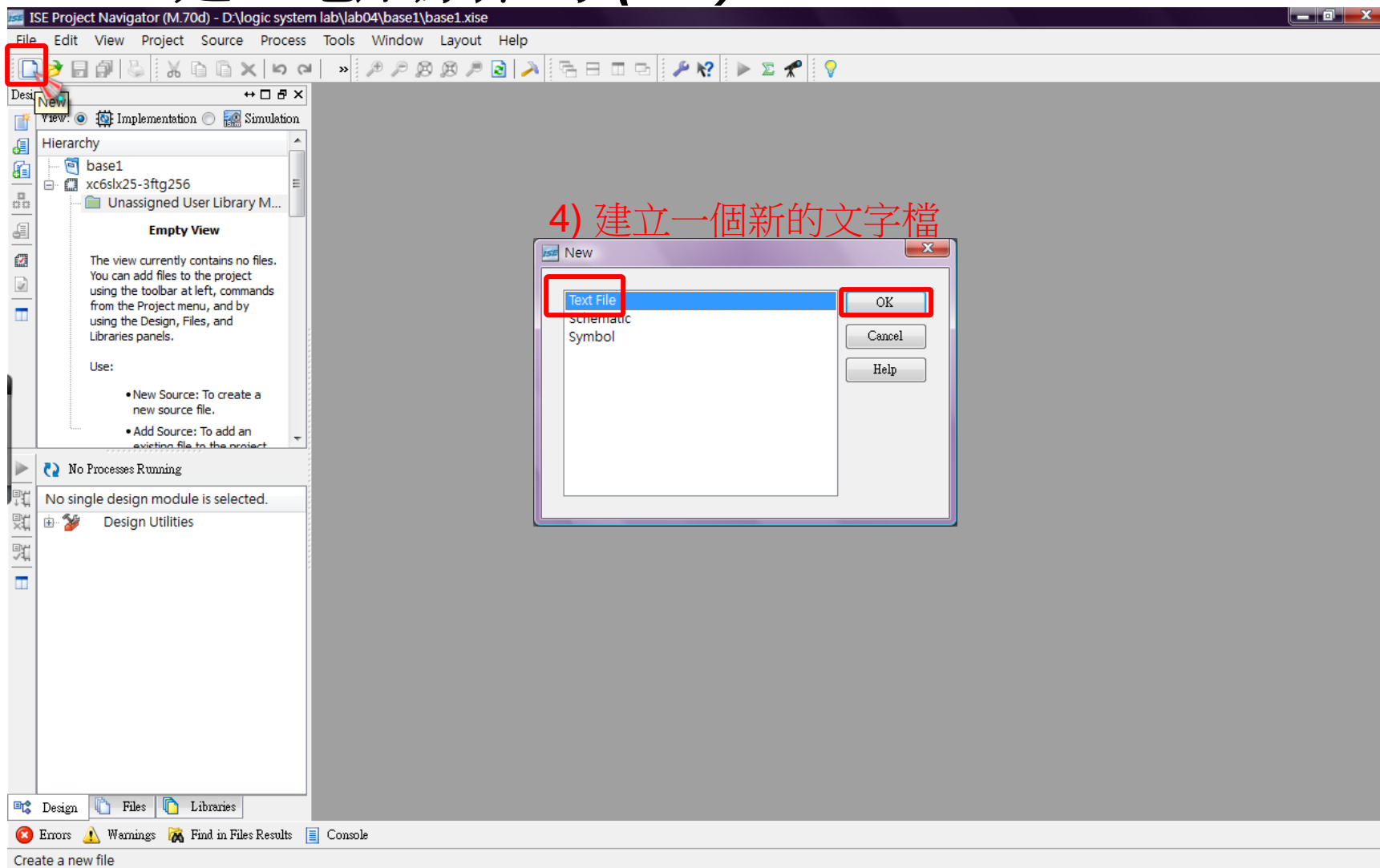
# Xilinx ISE操作 (3/10)

## 建立新專案 (3/3)



# Xilinx ISE操作 (4/10)

## 建立電路原始碼 (1/3)



# Xilinx ISE操作 (4/10)

## 建立電路原始碼 (2/3)

5) 打完原始碼後按存檔，存檔時檔名必須跟模組名稱相同，注意加上.v結尾  
ex : mux\_2to1.v

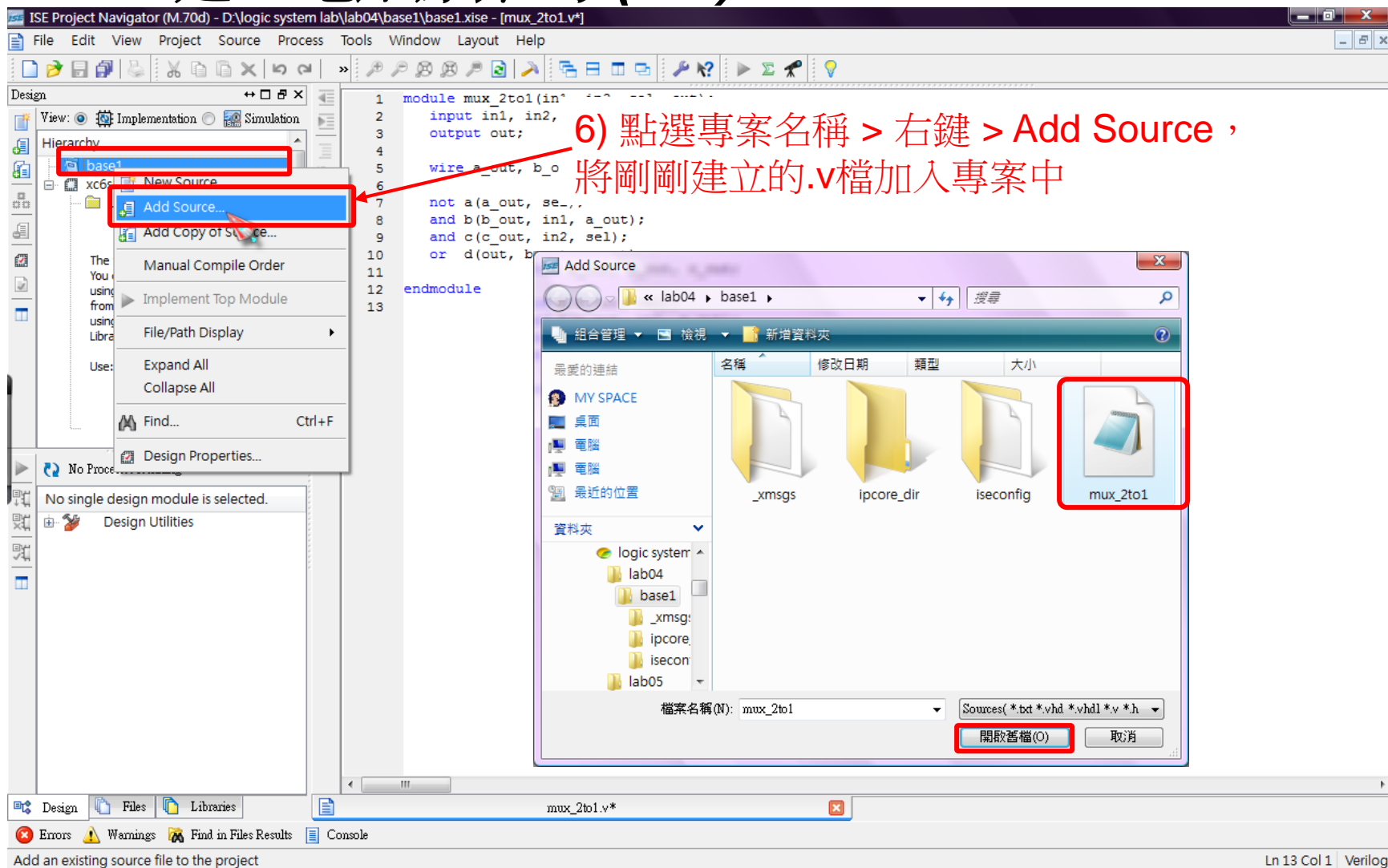
```
1 module mux_2to1(in1, in2, sel, out);
2   input in1, in2, sel;
3   output out;
4
5   wire a_out, b_out, c_out;
6
7   not a(a_out, sel);
8   and b(b_out, in1, a_out);
9   and c(c_out, in2, sel);
10  or d(out, b_out, c_out);
11
12 endmodule
13
```

檔案名稱(N): mux\_2to1.v  
存檔類型(T): Verilog (\*.v \*.vf \*.tf \*.tft \*.tftw \*.veo \*.tft \*.vh)

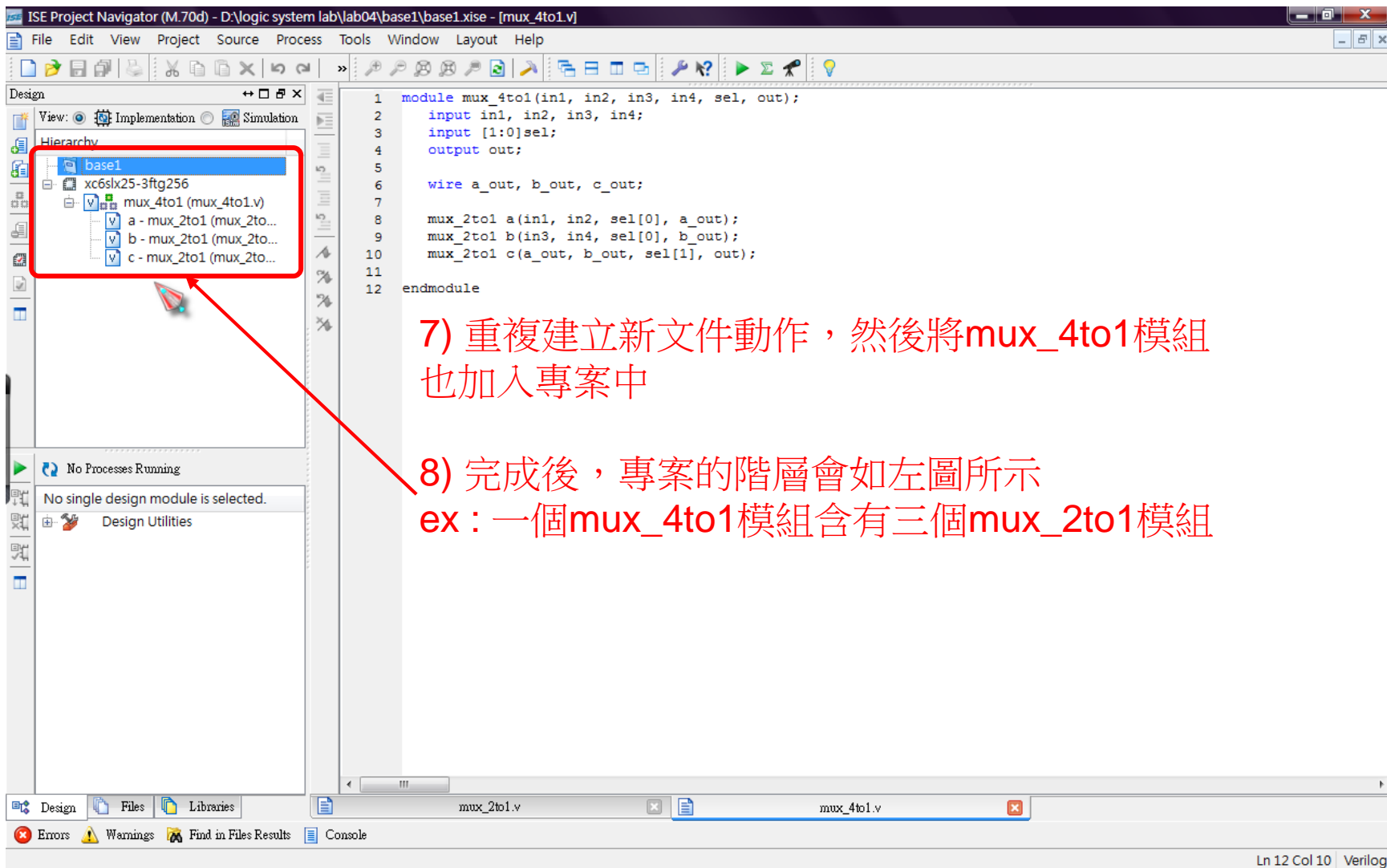
注意：一個.v檔只能放一個模組

# Xilinx ISE操作 (4/10)

## 建立電路原始碼 (3/3)



# Xilinx ISE操作 (5/10)



The screenshot displays the Xilinx ISE Project Navigator interface. The left pane shows the project hierarchy under 'base1', which includes a 'xc6slx25-3ftg256' device and a 'mux\_4to1 (mux\_4to1.v)' module. This module is further expanded to show three sub-modules: 'a - mux\_2to1 (mux\_2to1.v)', 'b - mux\_2to1 (mux\_2to1.v)', and 'c - mux\_2to1 (mux\_2to1.v)'. A red box highlights this hierarchy, and a red arrow points from the text below to it. The right pane shows the Verilog code for the 'mux\_4to1' module, which defines four inputs (in1, in2, in3, in4), a 2-bit select input (sel), and a single-bit output (out). It uses three 'mux\_2to1' modules to implement the 4-to-1 multiplexer logic. The bottom status bar indicates the current file is 'mux\_4to1.v' and the line/col is 'Ln 12 Col 10'.

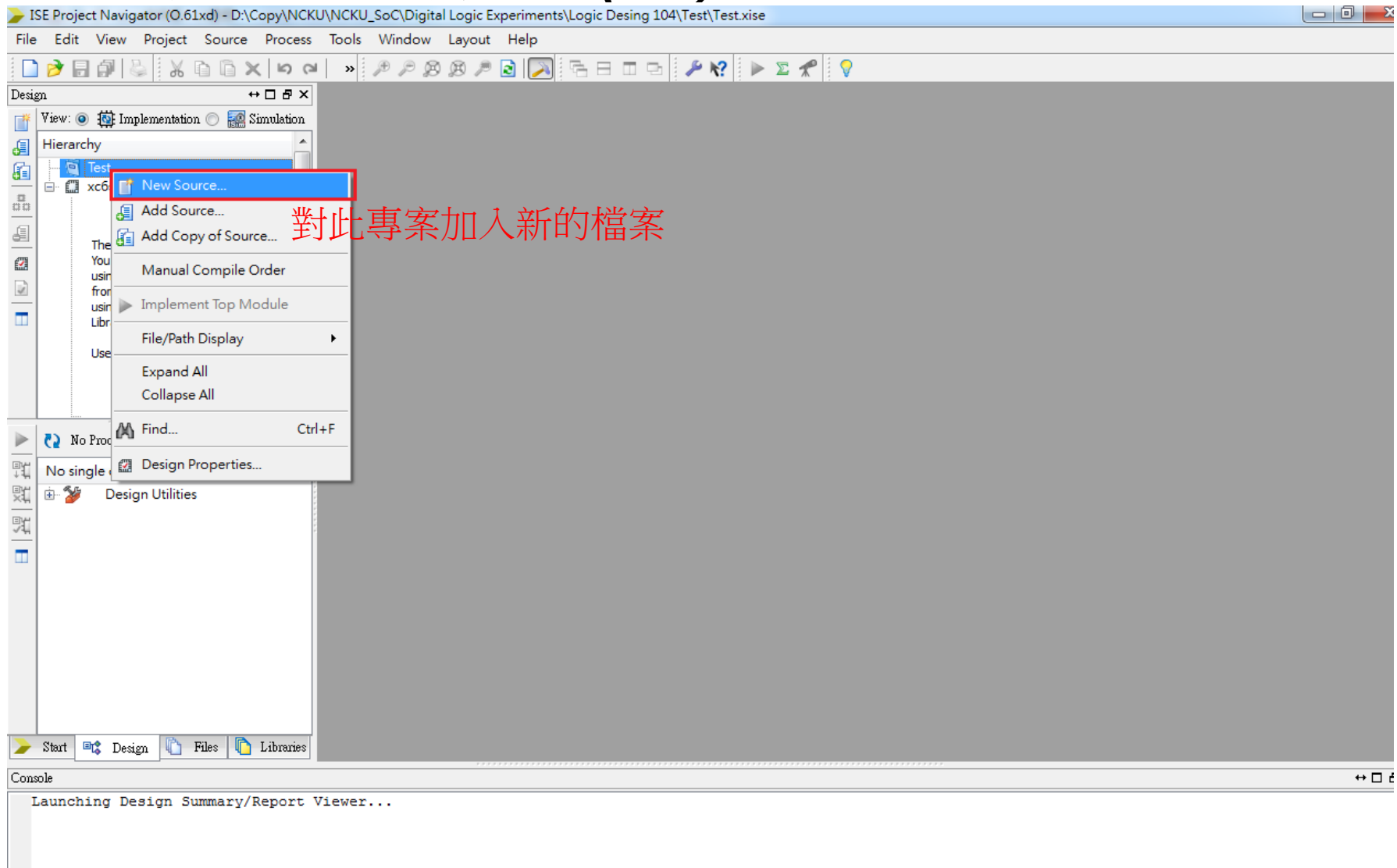
```
1 module mux_4to1(in1, in2, in3, in4, sel, out);
2   input in1, in2, in3, in4;
3   input [1:0]sel;
4   output out;
5
6   wire a_out, b_out, c_out;
7
8   mux_2to1 a(in1, in2, sel[0], a_out);
9   mux_2to1 b(in3, in4, sel[0], b_out);
10  mux_2to1 c(a_out, b_out, sel[1], out);
11
12 endmodule
```

7) 重複建立新文件動作，然後將mux\_4to1模組也加入專案中

8) 完成後，專案的階層會如左圖所示  
ex：一個mux\_4to1模組含有三個mux\_2to1模組

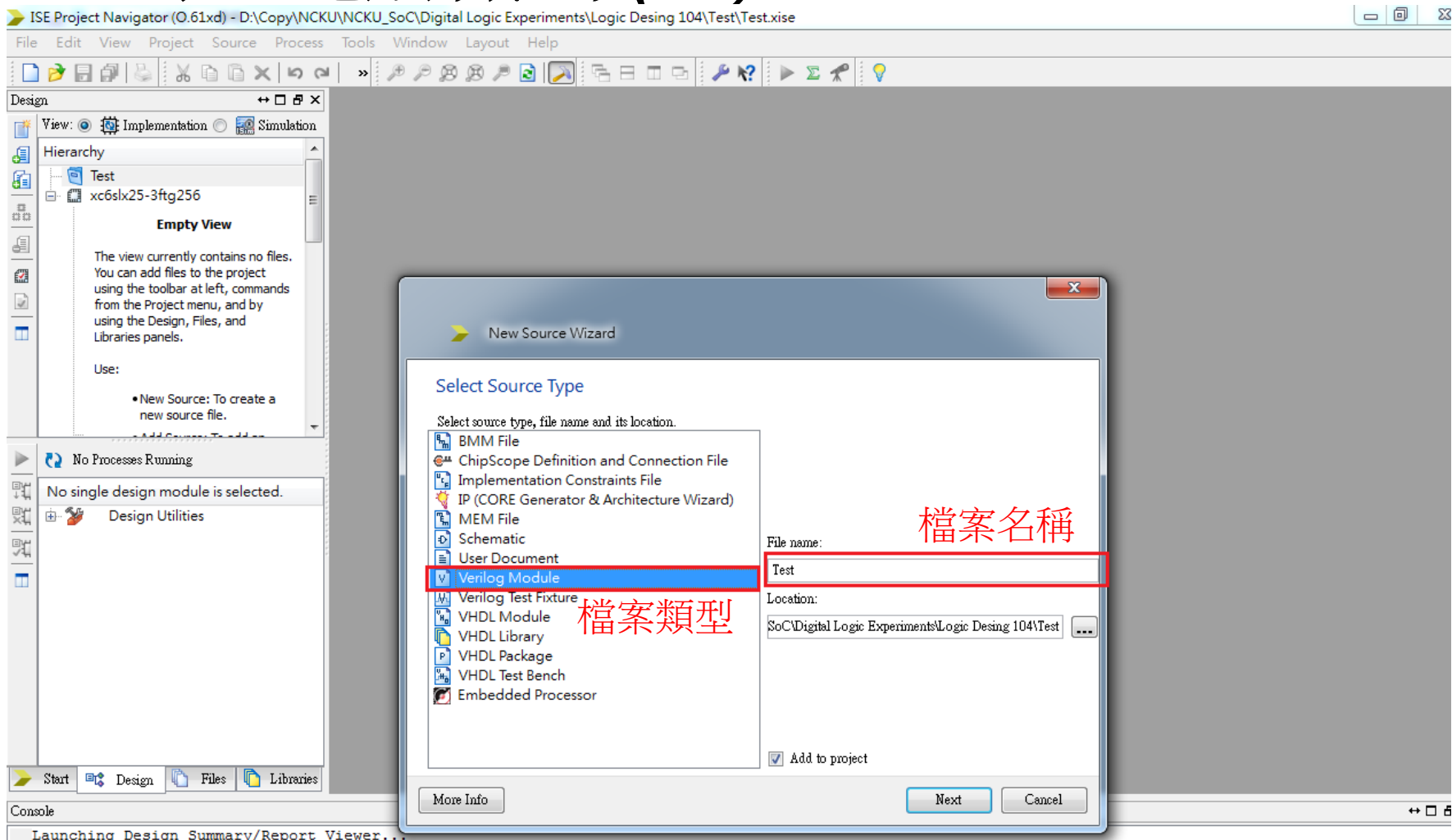
# Xilinx ISE操作 (補充)

## 建立電路原始碼 (1/5)



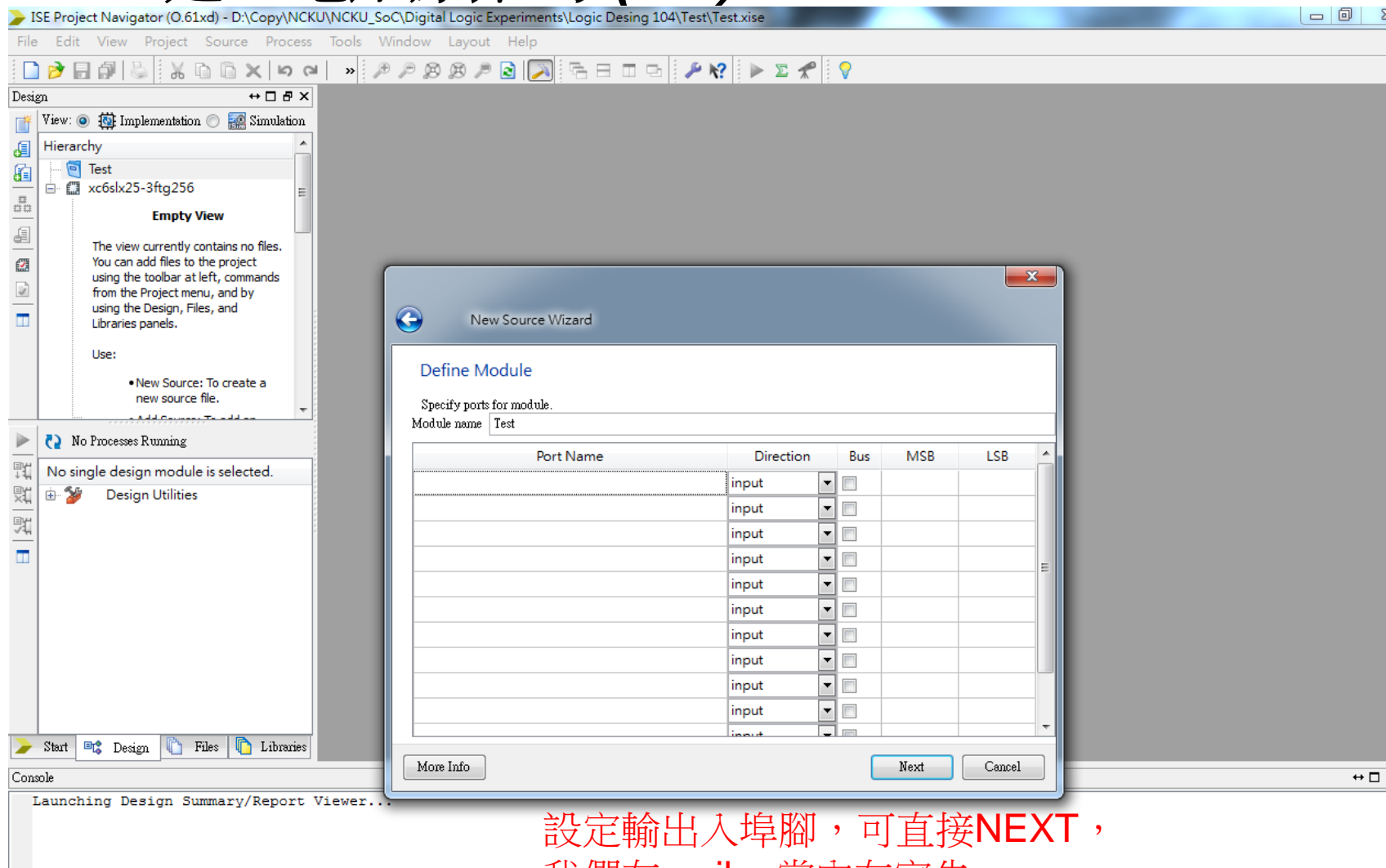
# Xilinx ISE操作 (補充)

## 建立電路原始碼 (2/5)



# Xilinx ISE操作 (補充)

## 建立電路原始碼 (3/5)

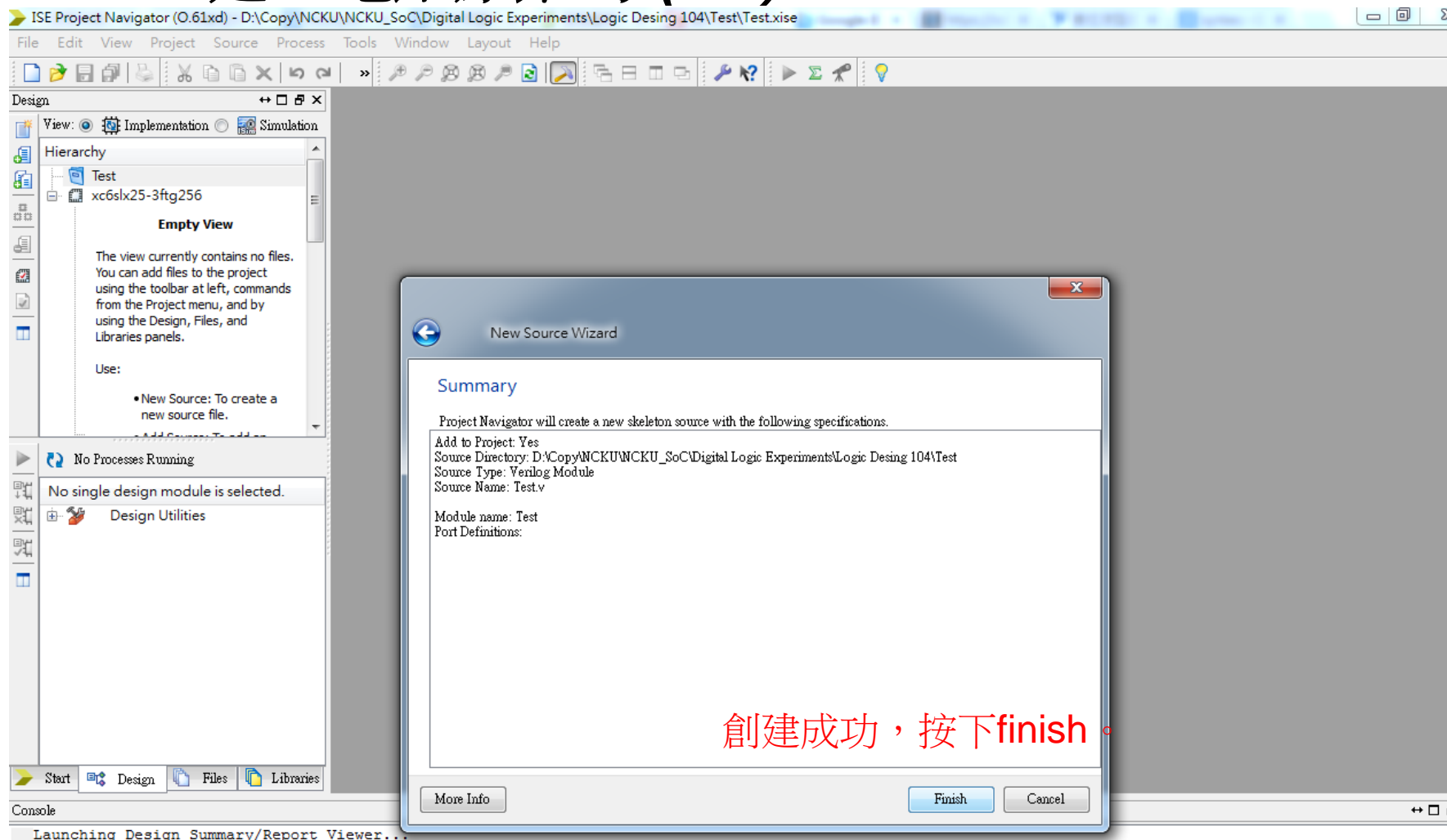


設定輸出入埠腳，可直接NEXT，  
我們在verilog黨內在宣告。



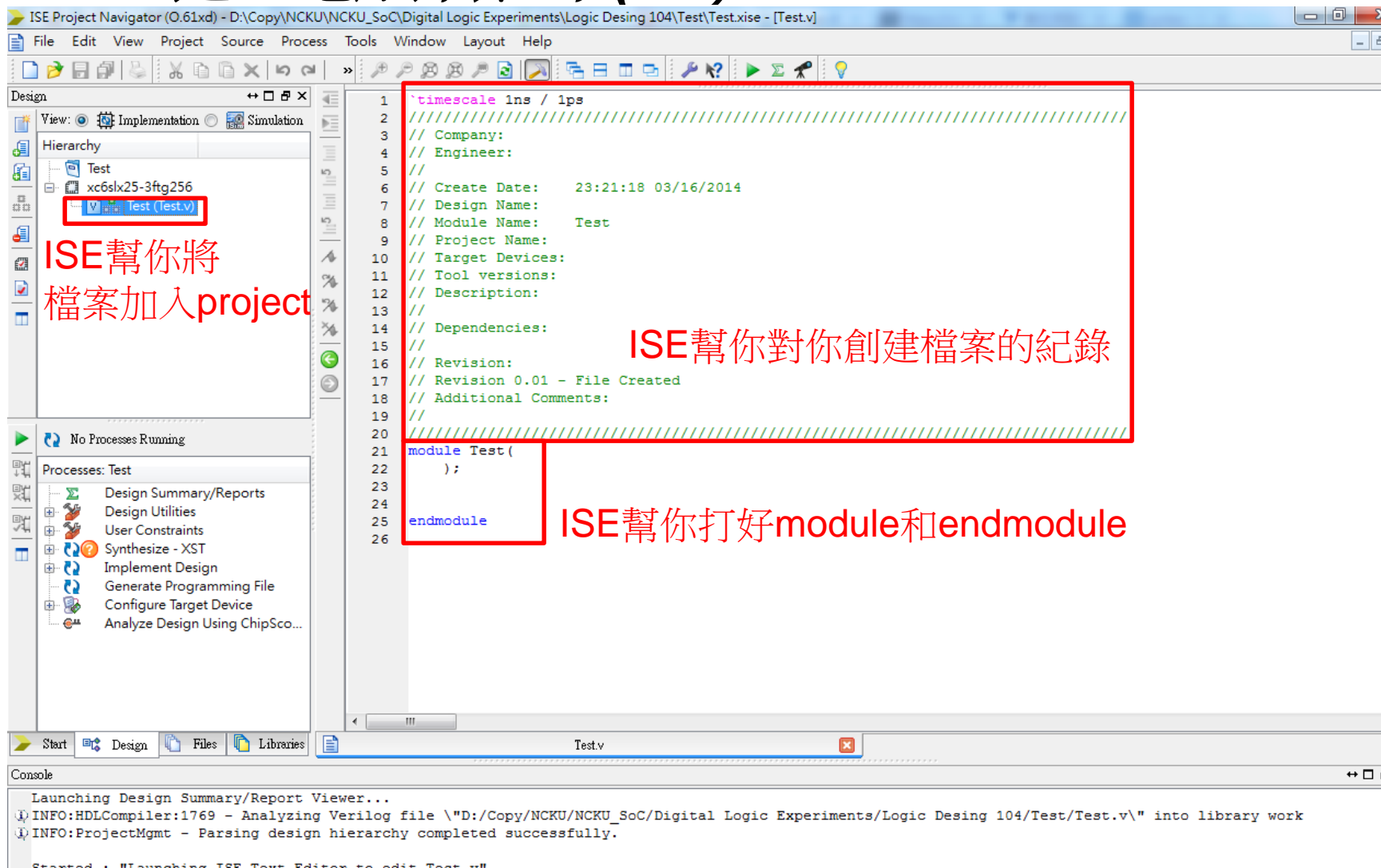
# Xilinx ISE操作 (補充)

## 建立電路原始碼 (4/5)



# Xilinx ISE操作 (補充)

## 建立電路原始碼 (5/5)



The screenshot displays the Xilinx ISE Project Navigator interface. On the left, the 'Design' pane shows the project hierarchy with 'Test' selected. The 'Processes' pane lists various design steps, including 'Synthesize - XST' and 'Implement Design'. The main editor window shows the Verilog code for 'Test.v', which includes a header section with project metadata and a module definition for 'Test'.

ISE 幫你將檔案加入project

ISE 幫你對你創建檔案的紀錄

ISE 幫你打好module和endmodule

```
1 `timescale 1ns / 1ps
2 //////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date:    23:21:18 03/16/2014
7 // Design Name:
8 // Module Name:    Test
9 // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////
21 module Test(
22 );
23
24
25 endmodule
26
```

Console

Launching Design Summary/Report Viewer...

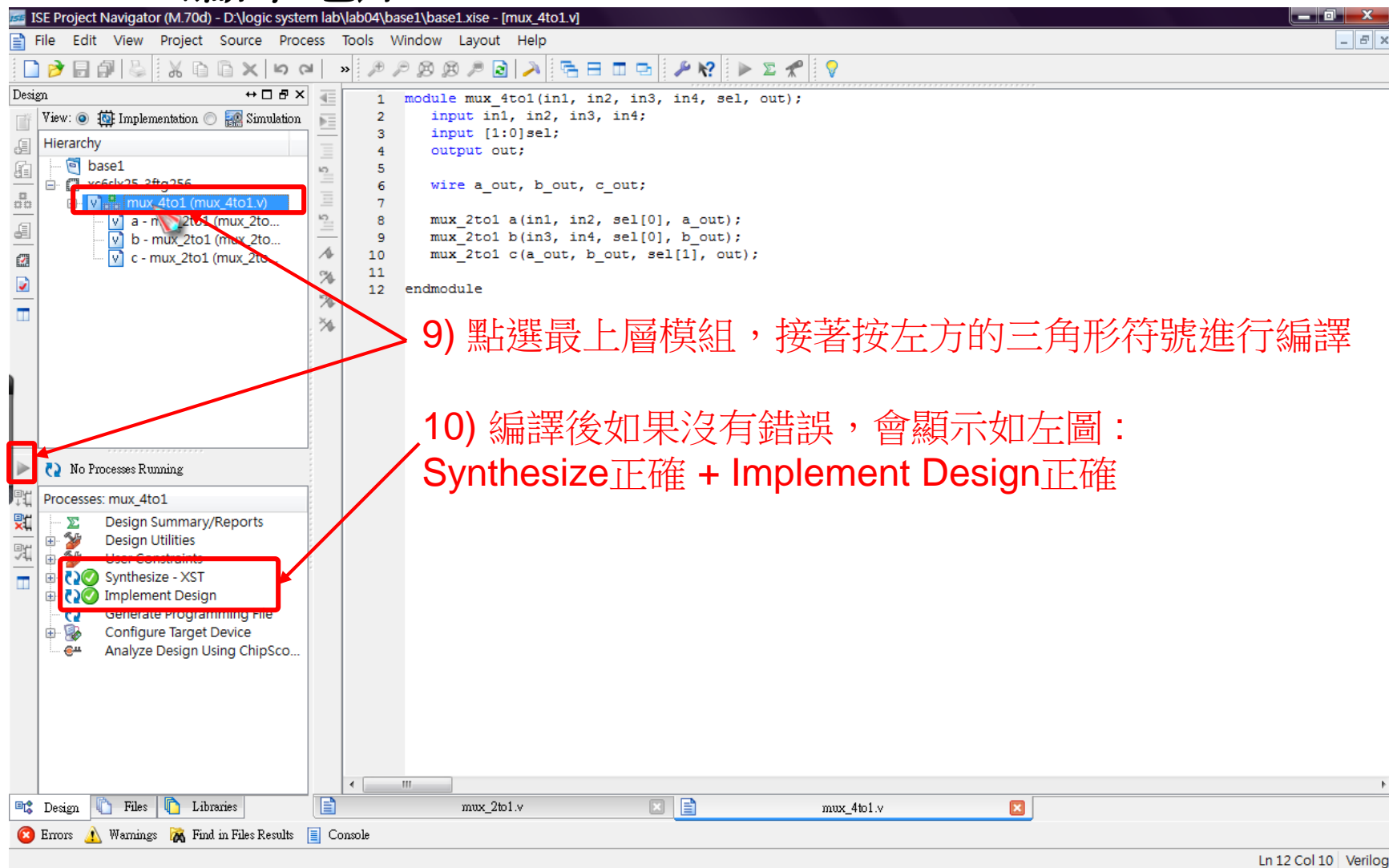
INFO:HDLCompiler:1769 - Analyzing Verilog file "D:/Copy/NCKU/NCKU\_SoC/Digital Logic Experiments/Logic Desing 104/Test/Test.v" into library work

INFO:ProjectMgmt - Parsing design hierarchy completed successfully.

Started : "Launching ISE Text Editor to edit Test.v"

# Xilinx ISE操作 (6/10)

## 編譯電路



ISE Project Navigator (M.70d) - D:\logic system lab\lab04\base1\base1.xise - [mux\_4to1.v]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- base1
  - xc6kx25-3ftg256
    - mux\_4to1 (mux\_4to1.v)**
      - a - mux\_2to1 (mux\_2to1.v)
      - b - mux\_2to1 (mux\_2to1.v)
      - c - mux\_2to1 (mux\_2to1.v)

```
1 module mux_4to1(in1, in2, in3, in4, sel, out);
2     input in1, in2, in3, in4;
3     input [1:0]sel;
4     output out;
5
6     wire a_out, b_out, c_out;
7
8     mux_2to1 a(in1, in2, sel[0], a_out);
9     mux_2to1 b(in3, in4, sel[0], b_out);
10    mux_2to1 c(a_out, b_out, sel[1], out);
11
12 endmodule
```

9) 點選最上層模組，接著按左方的三角形符號進行編譯

10) 編譯後如果沒有錯誤，會顯示如左圖：  
**Synthesize正確 + Implement Design正確**

No Processes Running

Processes: mux\_4to1

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST**
- Implement Design**
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipSco...

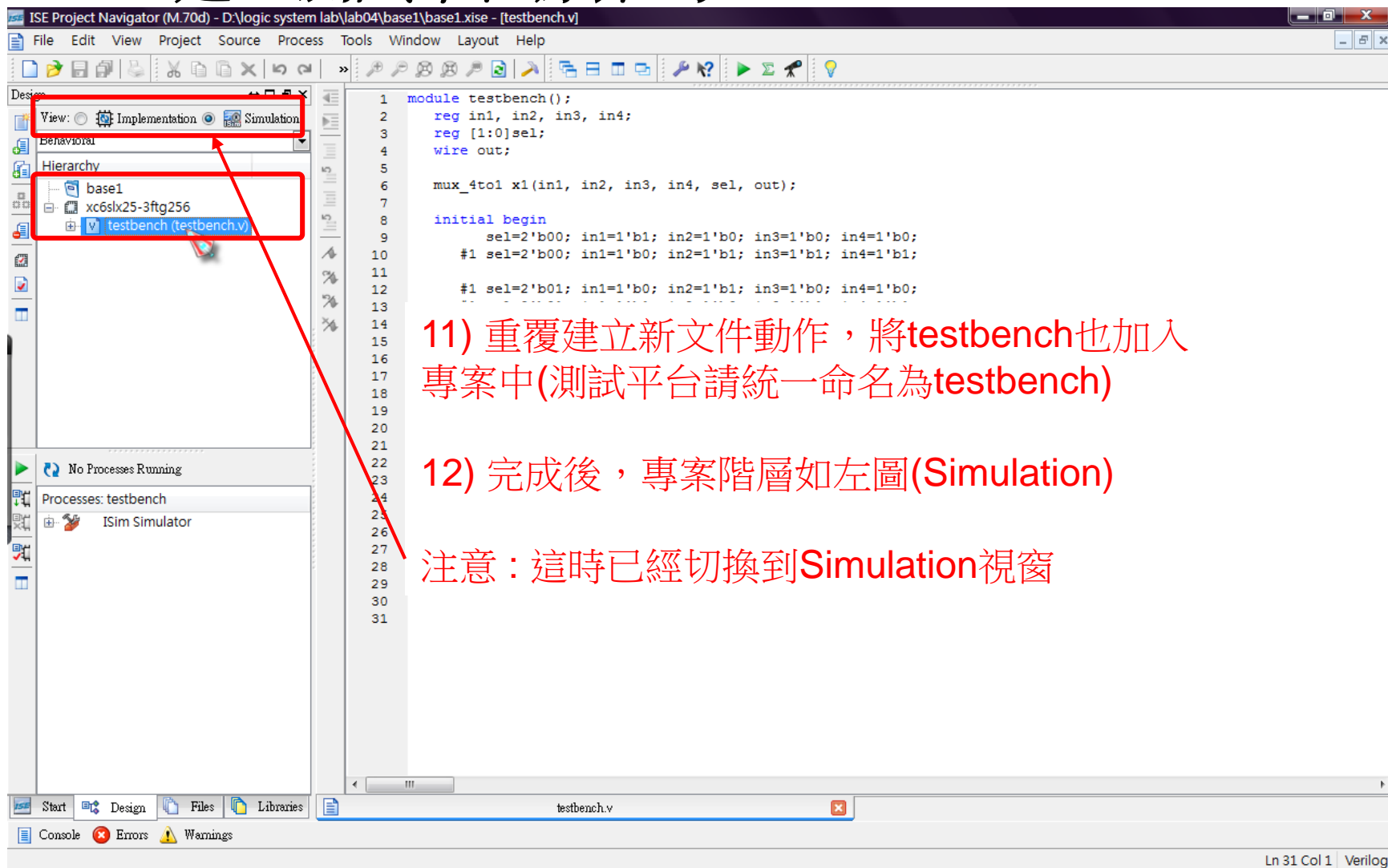
Design Files Libraries

Errors Warnings Find in Files Results Console

Ln 12 Col 10 Verilog

# Xilinx ISE操作 (7/10)

## 建立測試平台原始碼



The screenshot shows the Xilinx ISE Project Navigator interface. The 'View' dropdown is set to 'Simulation'. In the 'Hierarchy' pane, the project 'base1' is expanded, showing 'xc6slx25-3ftg256' and 'testbench (testbench.v)'. The 'testbench (testbench.v)' file is selected. The main editor displays the Verilog code for the testbench. The code includes module declarations, register and wire declarations, and a series of initial values and assignments for the testbench.

```
1 module testbench();
2     reg in1, in2, in3, in4;
3     reg [1:0] sel;
4     wire out;
5
6     mux_4to1 x1(in1, in2, in3, in4, sel, out);
7
8     initial begin
9         sel=2'b00; in1=1'b1; in2=1'b0; in3=1'b0; in4=1'b0;
10        #1 sel=2'b00; in1=1'b0; in2=1'b1; in3=1'b1; in4=1'b1;
11
12        #1 sel=2'b01; in1=1'b0; in2=1'b1; in3=1'b0; in4=1'b0;
13    end
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

11) 重覆建立新文件動作，將testbench也加入專案中(測試平台請統一命名為testbench)

12) 完成後，專案階層如左圖(Simulation)

注意：這時已經切換到Simulation視窗

# Xilinx ISE操作 (補充)

## 建立測試平台原始碼(1/6)

**ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4\_1\Lab4.xise - [Design Summary]**

File Edit View Project Source Process Tools Window Layout Help

**Design Overview**

- Summary
- IOB Properties
- Module Level Utilization
- Timing Constraints
- Pinout Report
- Clock Report
- Static Timing
- Errors and Warnings
- Parser Messages
- Synthesis Messages
- Translation Messages
- Map Messages
- Place and Route Messages
- Timing Messages
- Bitgen Messages
- All Implementation Messages
- Detailed Reports
- Synthesis Report
- Translation Report
- Map Report
- Place and Route Report
- Post-PAR Static Timing Report

**Design Properties**

- Enable Message Filtering
- Optional Design Summary Contents
- Show Clock Report
- Show Failing Constraints
- Show Warnings
- Show Errors

**Design Utilities**

- No Processes Running
- No single design module is selected.
- Design Utilities

**Design Properties**

- Find... Ctrl+F
- Design Properties...

**Design Summary**

**mux\_4to1 Project Status**

<b>Project File:</b>	Lab4.xise	<b>Parser Errors:</b>	No Errors
<b>Module Name:</b>	mux_4to1	<b>Implementation State:</b>	Placed and Routed
<b>Target Device:</b>	xc6slx25-3ftg256	<b>Errors:</b>	No Errors
<b>Product Version:</b>	ISE 13.2	<b>Warnings:</b>	No Warnings
<b>Design Goal:</b>	Balanced	<b>Routing Results:</b>	All Signals Completely Routed
<b>Design Strategy:</b>	Xilinx Default (unlocked)	<b>Timing Constraints:</b>	
<b>Environment:</b>	System Settings	<b>Final Timing Score:</b>	0 (Timing Report)

**Device Utilization Summary**

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	0	30,064	0%	
Number of Slice LUTs	1	15,032	1%	
Number used as logic	1	15,032	1%	
Number using O6 output only	1			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as ROM	0			
Number used as Memory	0	3,664	0%	
Number of occupied Slices	1	3,758	1%	
Number of LUT Flip Flop pairs used	1			
Number with an unused Flip Flop	1	1	100%	
Number with an unused LUT	0	1	0%	
Number of fully used LUT-FF pairs	0	1	0%	
Number of slice register sites lost to control set restrictions	0	30,064	0%	
Number of bonded IOBs	7	186	3%	
Number of RAMB16BWRs	0	52	0%	

**Console**

```

INFO:HDLCompiler:1769 - Analyzing Verilog file "D:/Copy/NCKU/NCKU_SoC/Digital Logic Experiments/Logic Desing 104/Lab/Lab4_1/mux_2to1.v" into library wor
INFO:HDLCompiler:1769 - Analyzing Verilog file "D:/Copy/NCKU/NCKU_SoC/Digital Logic Experiments/Logic Desing 104/Lab/Lab4_1/mux_4to1.v" into library wor
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
INFO:HDLCompiler:1769 - Analyzing Verilog file "D:/Copy/NCKU/NCKU_SoC/Digital Logic Experiments/Logic Desing 104/Lab/Lab4_1/mux_2to1.v" into library wor
INFO:HDLCompiler:1769 - Analyzing Verilog file "D:/Copy/NCKU/NCKU_SoC/Digital Logic Experiments/Logic Desing 104/Lab/Lab4_1/mux_4to1.v" into library wor
  
```

# Xilinx ISE操作(補充)

## 建立測試平台原始碼(2/6)

The screenshot shows the ISE Project Navigator interface. The 'New Source Wizard' dialog is open, prompting the user to 'Select Source Type'. The list of source types includes BMM File, ChipScope Definition and Connection File, Implementation Constraints File, IP (CORE Generator & Architecture Wizard), MEM File, Schematic, User Document, Verilog Module, Verilog Test Fixture (selected), VHDL Module, VHDL Library, VHDL Package, VHDL Test Bench, and Embedded Processor. The 'File name:' field is set to 'testbench', and the 'Location:' field is set to 'D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4\_1'. The background shows the project hierarchy on the left and the 'mux\_4to1 Project Status' window on the right, which indicates that the project is 'Placed and Routed' with 'No Errors' and 'No Warnings'.

# Xilinx ISE操作(補充)

## 建立測試平台原始碼(3/6)

ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4\_1\Lab4.xise - [Design Summary]

File Edit View Project Source Process Tools Window Layout Help

Design Overview

- Summary
- IOB Properties
- Module Level Utilization
- Timing Constraints
- Pinout Report
- Clock Report
- Static Timing
- Errors and Warnings
- Parser Messages

mux\_4to1 Project Status

Project File:	Lab4.xise	Parser Errors:	No Errors
Module Name:	mux_4to1	Implementation State:	Placed and Routed
Target Device:	xc6slx25-3ftg256	Errors:	No Errors
Product Version:	ISE 13.2	Warnings:	No Warnings
Design Goal:	Balanced	Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	Timing Constraints:	
Environment:	Custom Settings	Final Timing Score:	0 (Timing Report)

Design

View: Implementation Simulation

Behavioral

Hierarchy

- Lab4
  - xc6slx25-3ftg256
    - mux\_4to1 (mux\_4to1.v)
      - a - mux\_2to1 (mux\_2to1.v)
      - b - mux\_2to1 (mux\_2to1.v)
      - c - mux\_2to1 (mux\_2to1.v)

No Processes Running

No single design module is selected.

Design Utilities

Start Design Files Libraries

Console

INFO:HDLCompiler:1769 - Analyzing Verilog file \D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4\_1\mux\_2to1.v\" into library worl  
INFO:HDLCompiler:1769 - Analyzing Verilog file \D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4\_1\mux\_4to1.v\" into library worl  
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.  
INFO:HDLCompiler:1769 - Analyzing Verilog file \D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4\_1\mux\_2to1.v\" into library worl  
INFO:HDLCompiler:1769 - Analyzing Verilog file \D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4\_1\mux\_4to1.v\" into library worl

New Source Wizard

Associate Source

Select a source with which to associate the new source.

- mux\_4to1
- mux\_2to1

可先選擇要模擬的檔案

More Info Next Cancel

available	Utilization	Note(s)
30,064	0%	
15,032	1%	
15,032	1%	
3,664	0%	
3,758	1%	
1	100%	
1	0%	
1	0%	
30,064	0%	
186	3%	
52	0%	



# Xilinx ISE操作 (補充)

## 建立測試平台原始碼(4/6)

ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4\_1\Lab4.xise - [Design Summary]

File Edit View Project Source Process Tools Window Layout Help

Design Overview

- Summary
- IOB Properties
- Module Level Utilization
- Timing Constraints
- Pinout Report
- Clock Report
- Static Timing
- Errors and Warnings
- Parser Messages

Design

View: Implementation Simulation

Behavioral

Hierarchy

- Lab4
  - xc6slx25-3ftg256
    - mux\_4to1 (mux\_4to1.v)
      - a - mux\_2to1 (mux\_2to1.v)
      - b - mux\_2to1 (mux\_2to1.v)
      - c - mux\_2to1 (mux\_2to1.v)

Design Utilities

No single design module is selected.

Design Utilities

Summary

Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes  
 Source Directory: D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4\_1  
 Source Type: Verilog Test Fixture  
 Source Name: testbench.v  
 Association: mux\_4to1

More Info Finish Cancel

mux\_4to1 Project Status

Project File:	Lab4.xise	Parser Errors:	No Errors
Module Name:	mux_4to1	Implementation State:	Placed and Routed
Target Device:	xc6slx25-3ftg256	Errors:	No Errors
Product Version:	ISE 13.2	Warnings:	No Warnings
Design Goal:	Balanced	Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	Timing Constraints:	
Environment:	System Settings	Final Timing Score:	0 (Timing Report)

available	Utilization	Note(s)
30,064	0%	
15,032	1%	
15,032	1%	
3,664	0%	
3,758	1%	
1	100%	
1	0%	
1	0%	
30,064	0%	
186	3%	
52	0%	

Console

```

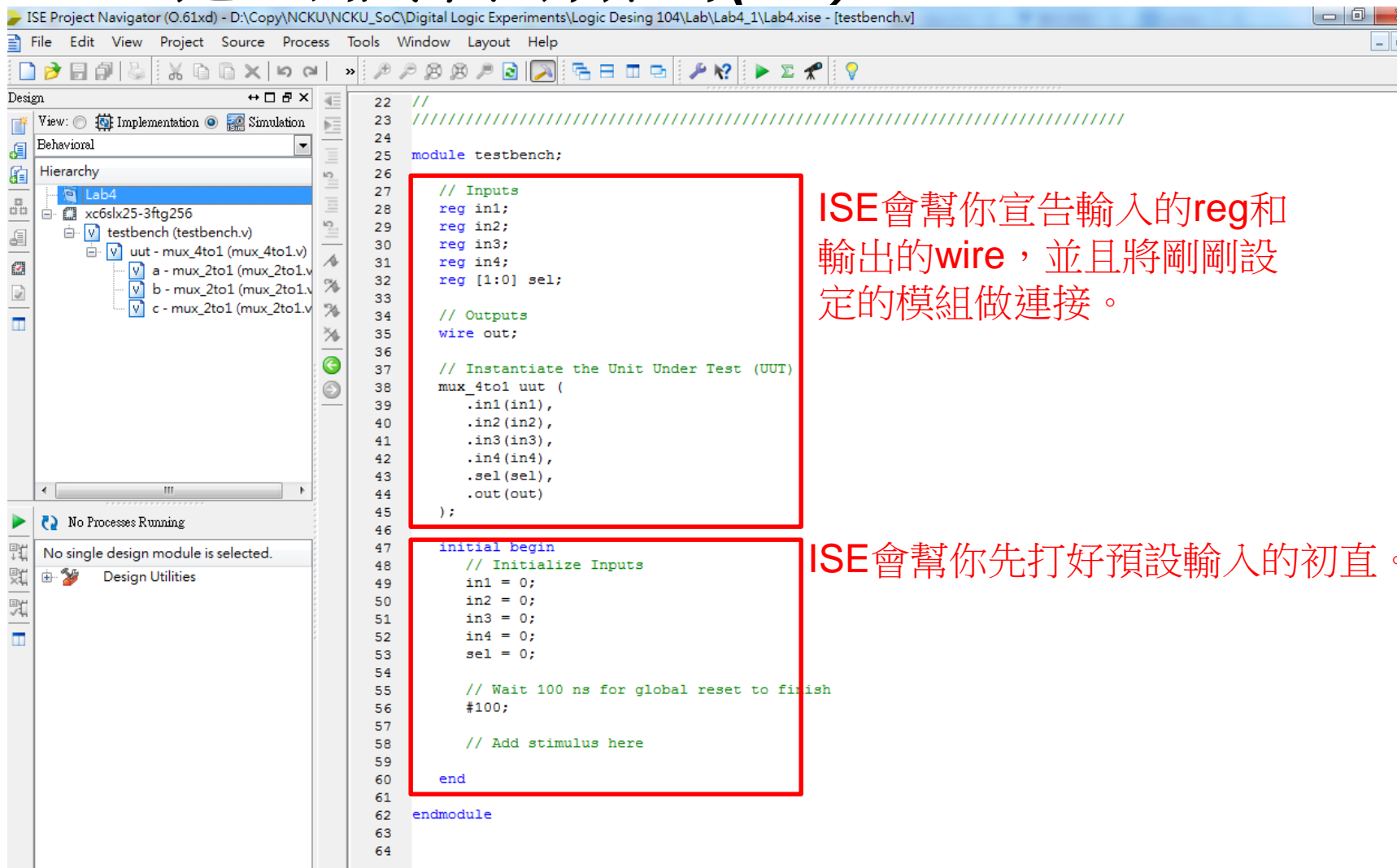
INFO:HDLCompiler:1769 - Analyzing Verilog file "D:\Copy\NCKU\NCKU_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4_1\mux_2to1.v" into library worl
INFO:HDLCompiler:1769 - Analyzing Verilog file "D:\Copy\NCKU\NCKU_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4_1\mux_4to1.v" into library worl
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
INFO:HDLCompiler:1769 - Analyzing Verilog file "D:\Copy\NCKU\NCKU_SoC\Digital Logic Experiments\Logic Desing 104\Lab\Lab4_1\mux_2to1.v" into library worl
  
```

創建成功



# Xilinx ISE操作 (補充)

## 建立測試平台原始碼(5/6)



ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Lab4\Lab4\_1\Lab4.xise - [testbench.v]

File Edit View Project Source Process Tools Window Layout Help

Design

View: Implementation Simulation

Behavioral

Hierarchy

xc6slx25-3ftg256

testbench (testbench.v)

uut - mux\_4to1 (mux\_4to1.v)

a - mux\_2to1 (mux\_2to1.v)

b - mux\_2to1 (mux\_2to1.v)

c - mux\_2to1 (mux\_2to1.v)

No Processes Running

No single design module is selected.

Design Utilities

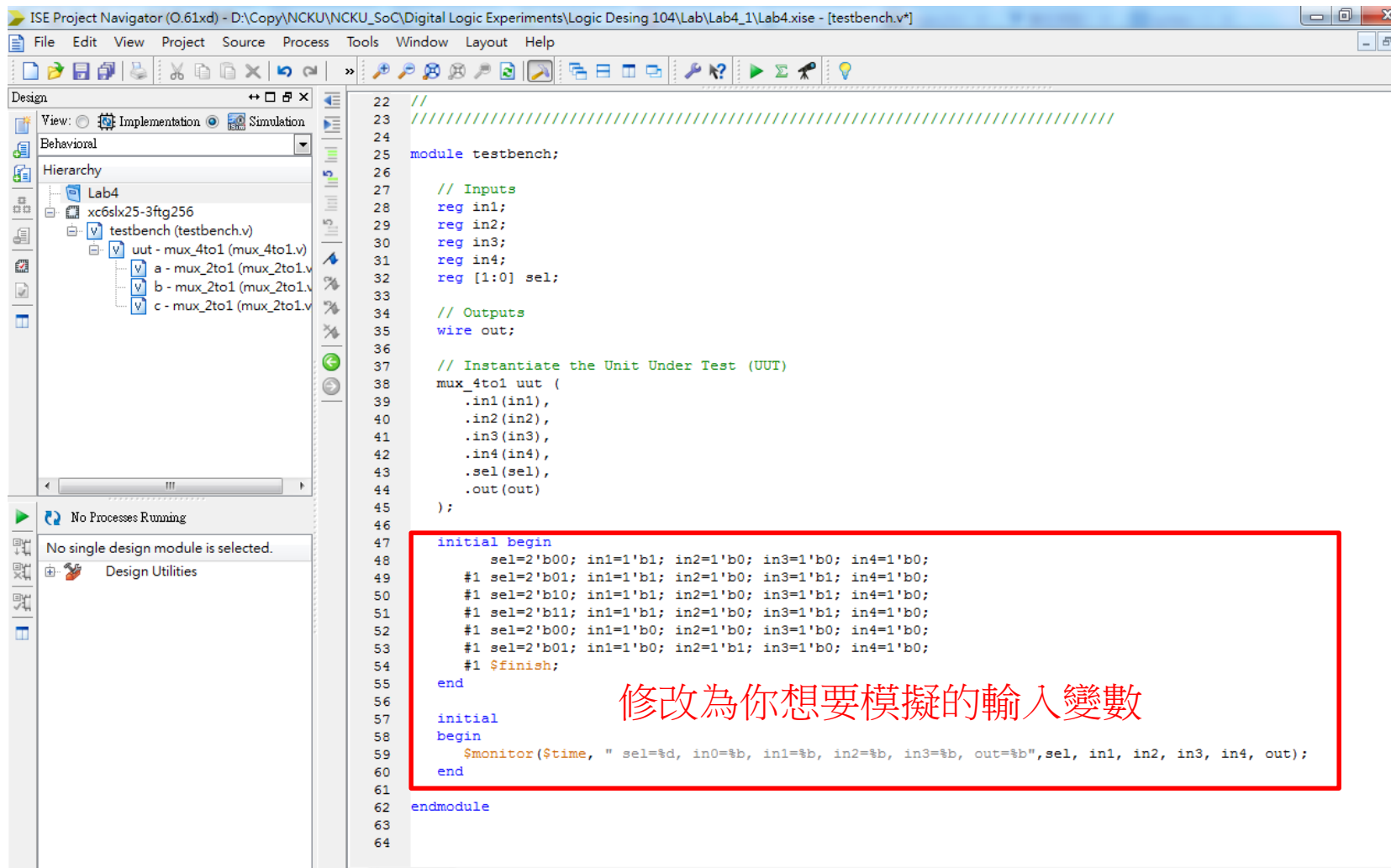
```
22 //
23 ///////////////////////////////////////////////////
24
25 module testbench;
26
27 // Inputs
28 reg in1;
29 reg in2;
30 reg in3;
31 reg in4;
32 reg [1:0] sel;
33
34 // Outputs
35 wire out;
36
37 // Instantiate the Unit Under Test (UUT)
38 mux_4to1 uut (
39     .in1(in1),
40     .in2(in2),
41     .in3(in3),
42     .in4(in4),
43     .sel(sel),
44     .out(out)
45 );
46
47 initial begin
48     // Initialize Inputs
49     in1 = 0;
50     in2 = 0;
51     in3 = 0;
52     in4 = 0;
53     sel = 0;
54
55     // Wait 100 ns for global reset to finish
56     #100;
57
58     // Add stimulus here
59
60 end
61
62 endmodule
63
64
```

ISE會幫你宣告輸入的reg和輸出的wire，並且將剛剛設定的模組做連接。

ISE會幫你先打好預設輸入的初值。

# Xilinx ISE操作 (補充)

## 建立測試平台原始碼(6/6)

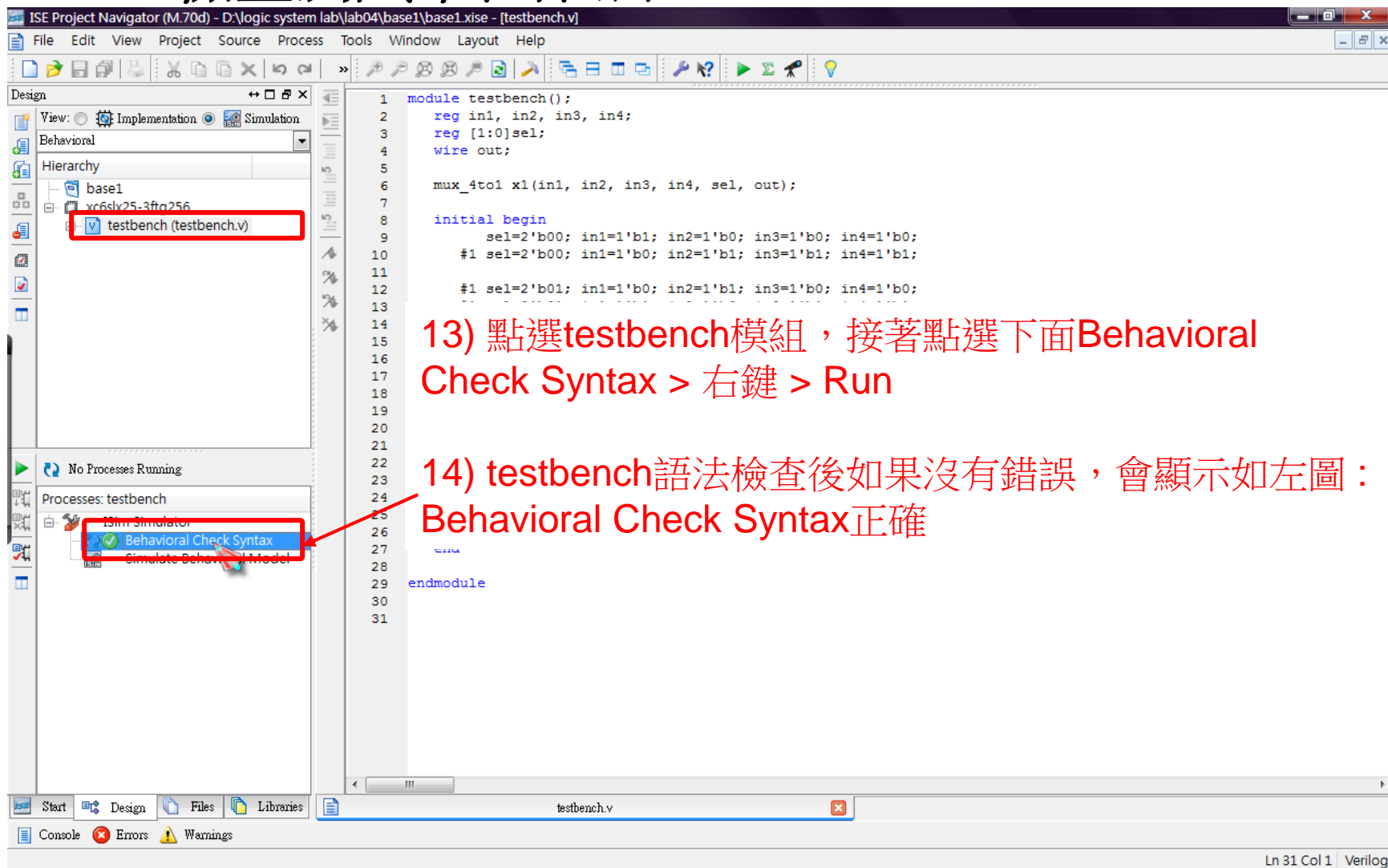


```
22 //
23 //////////////////////////////////////////////////
24
25 module testbench;
26
27 // Inputs
28 reg in1;
29 reg in2;
30 reg in3;
31 reg in4;
32 reg [1:0] sel;
33
34 // Outputs
35 wire out;
36
37 // Instantiate the Unit Under Test (UUT)
38 mux_4to1 uut (
39     .in1(in1),
40     .in2(in2),
41     .in3(in3),
42     .in4(in4),
43     .sel(sel),
44     .out(out)
45 );
46
47 initial begin
48     sel=2'b00; in1=1'b1; in2=1'b0; in3=1'b0; in4=1'b0;
49     #1 sel=2'b01; in1=1'b1; in2=1'b0; in3=1'b1; in4=1'b0;
50     #1 sel=2'b10; in1=1'b1; in2=1'b0; in3=1'b1; in4=1'b0;
51     #1 sel=2'b11; in1=1'b1; in2=1'b0; in3=1'b1; in4=1'b0;
52     #1 sel=2'b00; in1=1'b0; in2=1'b0; in3=1'b0; in4=1'b0;
53     #1 sel=2'b01; in1=1'b0; in2=1'b1; in3=1'b0; in4=1'b0;
54     #1 $finish;
55 end
56
57 initial
58 begin
59     $monitor($time, " sel=%d, in0=%b, in1=%b, in2=%b, in3=%b, out=%b",sel, in1, in2, in3, in4, out);
60 end
61
62 endmodule
63
64
```

修改為你想要模擬的輸入變數

# Xilinx ISE操作 (8/10)

## 檢查測試平台語法

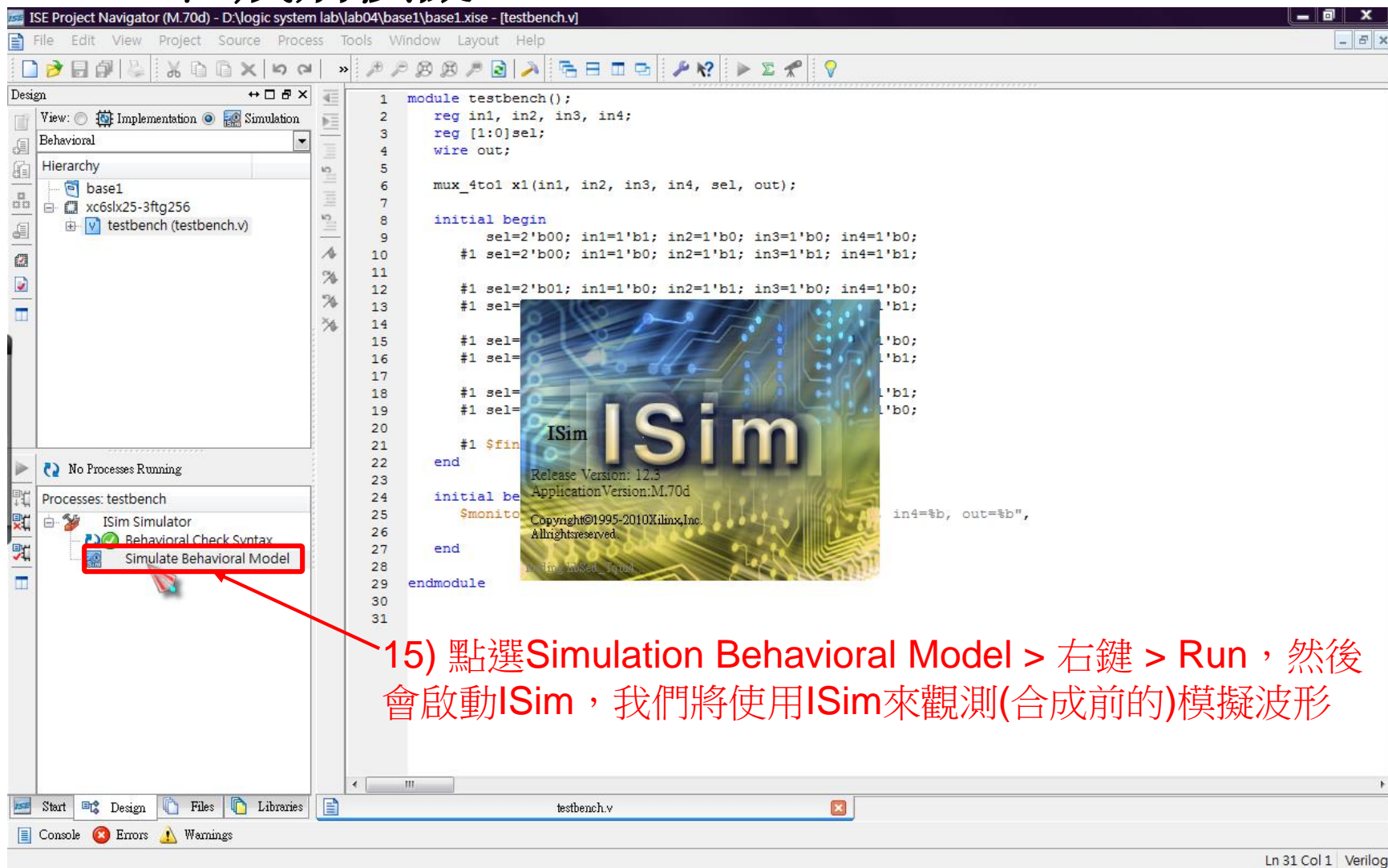


13) 點選testbench模組，接著點選下面Behavioral Check Syntax > 右鍵 > Run

14) testbench語法檢查後如果沒有錯誤，會顯示如左圖：Behavioral Check Syntax正確

# Xilinx ISE操作 (9/10)

## 合成前模擬



15) 點選Simulation Behavioral Model > 右鍵 > Run，然後會啟動ISim，我們將使用ISim來觀測(合成前的)模擬波形

# Xilinx ISE操作 (10/10)

## 波形驗證 (1/7)

The screenshot displays the Xilinx ISE simulation environment. The top window, titled "ISim (M.70d) - [Default.wcfg\*]", contains a variable list on the left and a waveform viewer on the right. The variable list includes "sel[1:0]" with a value of "11", and inputs "in1", "in2", "in3", "in4" all set to "1", and output "out" set to "0". The waveform viewer shows a time axis from 1 ns to 9 ns. A red box highlights the zoom controls in the toolbar, with a text box stating "適當的縮放視窗". Another red box highlights the waveform signals, with a text box stating "輸入、輸出、接線、暫存器等訊號波形". The bottom window, titled "Console", shows the simulation log. A red box highlights the log content, with a text box stating "\$display與\$monitor的列印結果會顯示在這裡". The log text includes the simulator version, time resolution, initialization process, and a list of signal states at each time step from 0 to 7 ns. The simulation stopped at 8 ns. The status bar at the bottom right indicates "Sim Time: 8,000 ps".

適當的縮放視窗

適當的調整I/O  
順序，可以增進  
對訊號的理解

輸入、輸出、接線、暫存器等訊號波形

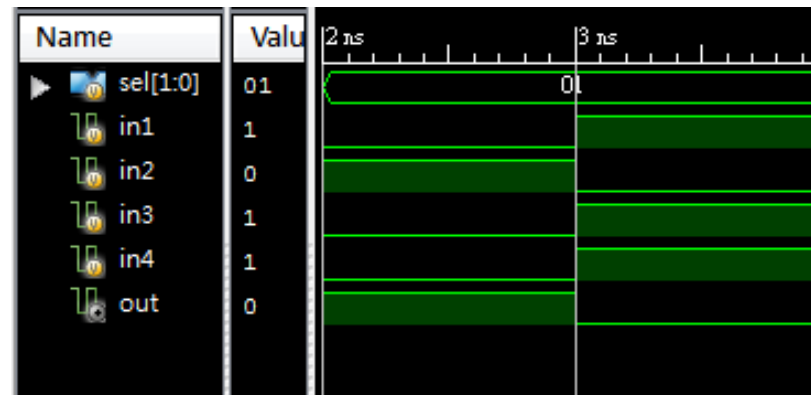
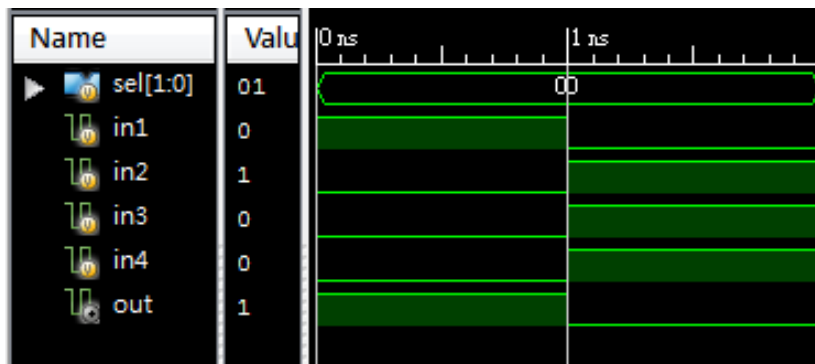
\$display與\$monitor的列印結果會顯示在這裡

ISim M.70d (signature 0x36e8438f)  
This is a Full version of ISim.  
Time resolution is 1 ps  
Simulator is doing circuit initialization process.  
Finished circuit initialization process.  
0 sel=0, in1=1, in2=0, in3=0, in4=0, out=1  
1 sel=0, in1=0, in2=1, in3=1, in4=1, out=0  
2 sel=1, in1=0, in2=1, in3=0, in4=0, out=1  
3 sel=1, in1=1, in2=0, in3=1, in4=1, out=0  
4 sel=2, in1=0, in2=0, in3=1, in4=0, out=1  
5 sel=2, in1=1, in2=1, in3=0, in4=1, out=0  
6 sel=3, in1=0, in2=0, in3=0, in4=1, out=1  
7 sel=3, in1=1, in2=1, in3=1, in4=0, out=0  
Stopped at time : 8 ns : File "D:\logic system lab\lab04\base1\testbench.v" Line 21  
ISim>

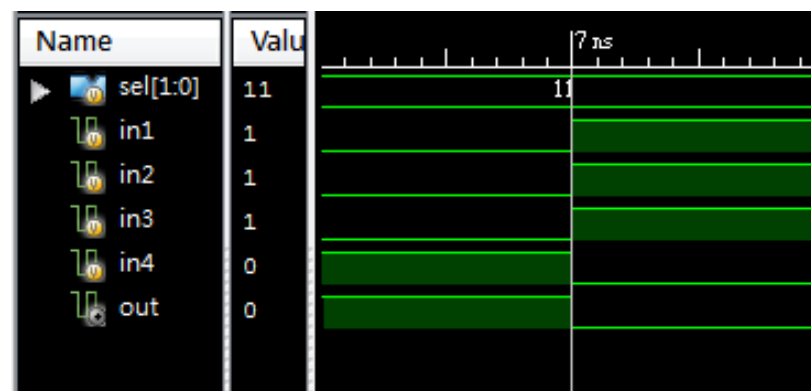
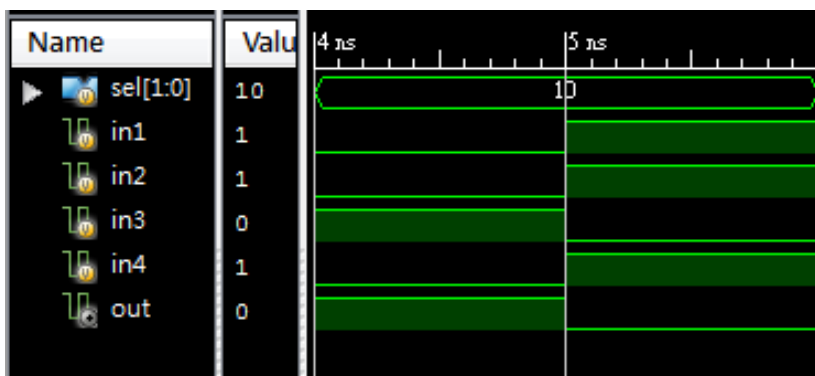
Sim Time: 8,000 ps

# Xilinx ISE操作 (10/10)

## 波形驗證 (2/7)



16) 接著逐一的檢查波形是否如所期望一般



# Xilinx ISE操作 (10/10)

## 波形驗證 (3/7)

17) 另外如果想要觀測電路內部的訊號，可以操作如下  
View > Panels > Instances and Processes

Console

ISim M.70d (signature 0x36e8438f)  
This is a Full version of ISim.  
Time resolution is 1 ps  
Simulator is doing circuit initialization process.  
Finished circuit initialization process.

0 sel=0, in1=1, in2=0, in3=0, in4=0, out=1  
1 sel=0, in1=0, in2=1, in3=1, in4=1, out=0  
2 sel=1, in1=0, in2=1, in3=0, in4=0, out=1  
3 sel=1, in1=1, in2=0, in3=1, in4=1, out=0  
4 sel=2, in1=0, in2=0, in3=1, in4=0, out=1  
5 sel=2, in1=1, in2=1, in3=0, in4=1, out=0  
6 sel=3, in1=0, in2=0, in3=0, in4=1, out=1  
7 sel=3, in1=1, in2=1, in3=1, in4=0, out=0

Stopped at time : 8 ns : File "D:\logic system lab\lab04\base1\testbench.v" Line 21  
ISim>

Sim Time: 8,000 ps

# Xilinx ISE操作 (10/10)

## 波形驗證 (4/7)

The screenshot shows the Xilinx ISE simulation environment. On the left, the 'Instances and Processes' pane lists the design hierarchy. The 'testbench' module contains an instance 'x1' of the 'mux\_4to1' module. A right-click context menu is open over the 'mux\_4to1' instance, with the option 'Add To Wave Window' highlighted. A red box and arrow point to this option. The main window displays a table of signals and their values, and a waveform viewer showing the timing of these signals. The console at the bottom shows the simulation log.

Name	Value
sel[1:0]	11
in1	1
in2	1
in3	1
in4	0
out	0

18) 開啟Panel後，點選你想要觀測的電路模組，右鍵 > Add to Wave Window，來將電路內部訊號加入到模擬波形中

ISim M.70d (signature 0x36e8438f)  
This is a Full version of ISim.  
Time resolution is 1 ps  
Simulator is doing circuit initialization process.  
Finished circuit initialization process.  
0 sel=0, in1=1, in2=0, in3=0, in4=0, out=1  
1 sel=0, in1=0, in2=1, in3=1, in4=1, out=0  
2 sel=1, in1=0, in2=1, in3=0, in4=0, out=1  
3 sel=1, in1=1, in2=0, in3=1, in4=1, out=0  
4 sel=2, in1=0, in2=0, in3=1, in4=0, out=1  
5 sel=2, in1=1, in2=1, in3=0, in4=1, out=0  
6 sel=3, in1=0, in2=0, in3=0, in4=1, out=1  
7 sel=3, in1=1, in2=1, in3=1, in4=0, out=0  
Stopped at time : 8 ns : File "D:\logic system lab\lab04\base1\testbench.v" Line 21  
ISim>

Add To Wave Configuration

Sim Time: 8,000 ps



# Xilinx ISE操作 (10/10)

## 波形驗證 (5/7)

The screenshot shows the Xilinx ISE waveform viewer interface. The top menu bar includes File, Edit, View, Simulation, Window, Layout, and Help. The toolbar contains various icons for file operations, editing, and simulation. The main window displays a timing diagram with a grid. The left pane shows a list of signals: sel[1:0], in1, in2, in3, in4, out, and a 'New Divider' menu option. The right pane shows the waveform for these signals. A red box highlights the 'New Divider' menu option, and a red arrow points to it. A text box with red text explains the function of this menu option.

19) 此外你也可以在訊號上，按右鍵 > New Divider，來插入一行分隔線，分隔線可以用來劃分訊號間的位置

Console

```
ISim M.70d (signature 0x36e8438f)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
0 sel=0, in1=1, in2=0, in3=0, in4=0, out=1
1 sel=0, in1=0, in2=1, in3=1, in4=1, out=0
2 sel=1, in1=0, in2=1, in3=0, in4=0, out=1
3 sel=1, in1=1, in2=0, in3=1, in4=1, out=0
4 sel=2, in1=0, in2=0, in3=1, in4=0, out=1
5 sel=2, in1=1, in2=1, in3=0, in4=1, out=0
6 sel=3, in1=0, in2=0, in3=0, in4=1, out=1
7 sel=3, in1=1, in2=1, in3=1, in4=0, out=0
Stopped at time : 8 ns : File "D:\logic system lab\lab04\base1\testbench.v" Line 21
ISim>
```

Sim Time: 8,000 ps

# Xilinx ISE操作 (10/10)

## 波形驗證 (6/7)

The screenshot displays the Xilinx ISE simulation environment. The top toolbar contains various simulation controls, with a red box highlighting the 'Run All' button (a blue square with a white play icon). A red arrow points from the text box to this button. The main window shows a waveform with a time scale from 0 to 9 ns. A red box highlights a section of the waveform from approximately 4 ns to 8 ns. The left pane shows a list of signals: sel[1:0], in1, in2, in3, in4, out, New Divider, in1, in2, sel, out, a\_out, b\_out, and c\_out. The bottom pane shows the console output, which includes the command '# restart', '# run all', and a list of simulation results for the 'New Divider' block. The console output is as follows:

```
ISim>
# restart
ISim>
# run all
Simulator is doing circuit initialization process.
Finished circuit initialization process.
0 sel=0, in1=1, in2=0, in3=0, in4=0, out=1
1 sel=0, in1=0, in2=1, in3=1, in4=1, out=0
2 sel=1, in1=0, in2=1, in3=0, in4=0, out=1
3 sel=1, in1=1, in2=0, in3=1, in4=1, out=0
4 sel=2, in1=0, in2=0, in3=1, in4=0, out=1
5 sel=2, in1=1, in2=1, in3=0, in4=1, out=0
6 sel=3, in1=0, in2=0, in3=0, in4=1, out=1
7 sel=3, in1=1, in2=1, in3=1, in4=0, out=0
Stopped at time : 8 ns : File "D:/logic system lab/lab04/base1/testbench.v" Line 21
ISim>
```

The console output shows a list of simulation results for the 'New Divider' block, including the values of sel, in1, in2, in3, in4, and out for each time step. The simulation is stopped at 8 ns. The bottom status bar shows 'Run All' and 'Sim Time: 8,000 ps'.

20) 加入新的訊號後，需要重新模擬，  
Restart(左邊按鈕) > Run All(右邊按鈕)

# Xilinx ISE操作 (10/10)

## 波形驗證 (7/7)

The screenshot shows the Xilinx ISE simulation environment. On the left, a project tree lists signals: carry\_in, in1[3:0], in2[3:0], carry\_c, and sum[3:0]. A context menu is open over the 'sum[3:0]' signal, showing options like Cut, Copy, Paste, Delete, Rename, Find..., Select All, Expand, Collapse, Ungroup, Name, Radix, Signal Color, Divider Color, and Reverse Bit Order. The 'Radix' submenu is open, showing options: Default, Binary, Hexadecimal, Unsigned Decimal (highlighted with a red box), Signed Decimal, Octal, and ASCII. The main window displays a waveform with time markers from 0 ns to 14 ns. The waveform shows three signals: in1[3:0] (green), in2[3:0] (green), and sum[3:0] (green). The sum[3:0] signal is currently in binary format. A text box in the center of the waveform area contains the text: 21) 最後你也可以轉換訊號的基底，以增加訊號的可讀性，選取訊號 > 右鍵 > Radix > (其他基底). The bottom status bar shows 'Uses unsigned decimal representation' and 'Sim Time: 15,000 ps'.

21) 最後你也可以轉換訊號的基底，以增加訊號的可讀性，選取訊號 > 右鍵 > Radix > (其他基底)

(其他範例)

# 基礎題 (一)

## 四對一多工器

- 請參考mux\_4to1.v與mux\_2to1.v的範例原始碼與操作流程。
  - 設計時，請利用Verilog邏輯閘層次語法寫出二對一多工器模組，然後利用二對一多工器模組，組合出四對一多工器模組，並且確認編譯後沒有error或warning產生。
  - 模擬時，請參考四對一多工器模組的testbench.v的範例原始碼與操作流程，並觀察波形結果與主控台的螢幕顯示結果。

### 邏輯閘層次

```
module mux_2to1(in1, in2, sel, out);  
  input in1, in2, sel;  
  output out;  
  
  wire a_out, b_out, c_out;  
  
  not a(a_out, sel);  
  and b(b_out, in1, a_out);  
  and c(c_out, in2, sel);  
  or d(out, b_out, c_out);  
  
endmodule
```

mux\_2to1

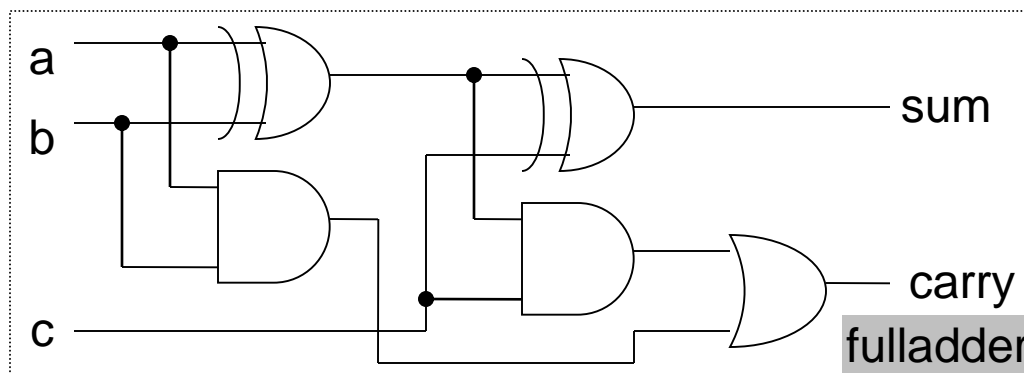
```
module mux_4to1_v2(in1, in2, in3, in4, sel, out);  
  input in1, in2, in3, in4;  
  input [1:0]sel;  
  output out;  
  
  wire a_out, b_out, c_out;  
  
  mux_2to1 a(.in1(in1), .in2(in2), .sel(sel[0]), .out(a_out));  
  mux_2to1 b(.in1(in3), .in2(in4), .sel(sel[0]), .out(b_out));  
  mux_2to1 c(.in1(a_out), .in2(b_out), .sel(sel[1]), .out(out));  
  
endmodule
```

mux\_4to1

# 基礎題(二)

## 全加器

- 請寫出全加器。
  - 設計時，請利用Verilog邏輯閘層次語法寫出全加器模組，並且確認編譯後沒有error或warning產生。
  - 模擬時，請自行撰寫testbench.v，並觀察波形結果或主控台的螢幕顯示結果是否如期望一般。



# 挑戰題

## 4bits漣波加法器

- 請寫出4bits的漣波加法器。
  - 設計時，請利用Verilog邏輯閘層次語法寫出全加器模組，然後利用全加器模組，組合出4bits漣波加法器模組，並且確認編譯後沒有error或warning產生。
  - 模擬時，請自行撰寫testbench.v，並觀察波形結果或主控台的螢幕顯示結果是否如期望一般。

