Laboratory 6

VeriLite 實驗電路板之介紹與操作

Department of Electrical Engineering
National Cheng Kung University

實驗目的

- 了解FPGA的基本概念
- 學習使用 VeriLite FPGA的工具來實踐邏輯電路並且驗證
  - HDL Auto Assign Pin
  - ISE Project Navigator
  - SMIMS FPGA programmer

使用器材

- 桌上型電腦
- Xilinx FPGA 板
- LED * 2
- 七段顯示器 * 1
- 杜邦線 * 12
- 排針

VeriLite FPGA

以下會簡單介紹VeriLite FPGA與 相關FPGA的知識

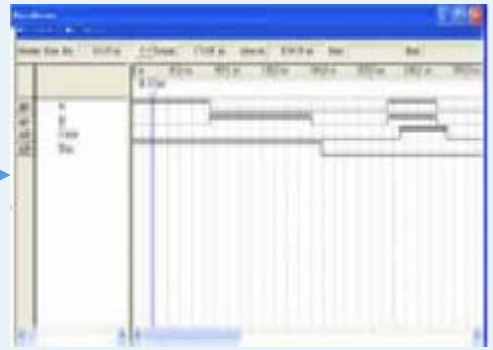
Design

HDL Model

Compilation

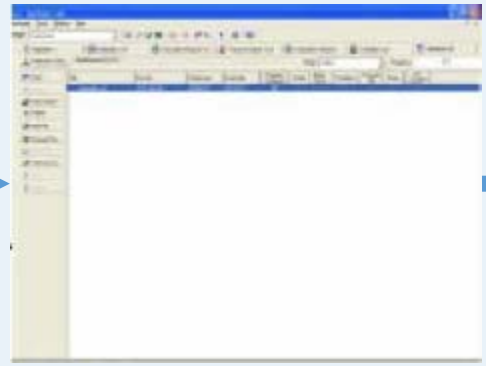
Compiler

Simulation

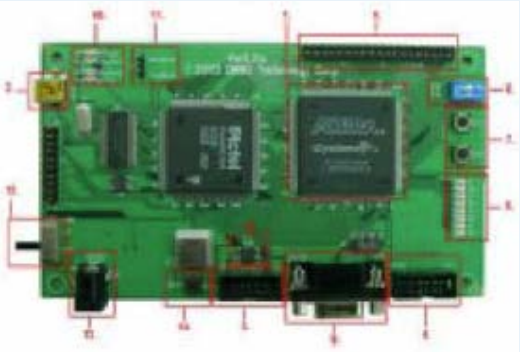


Timing Analysis

Verification



Program FPGA



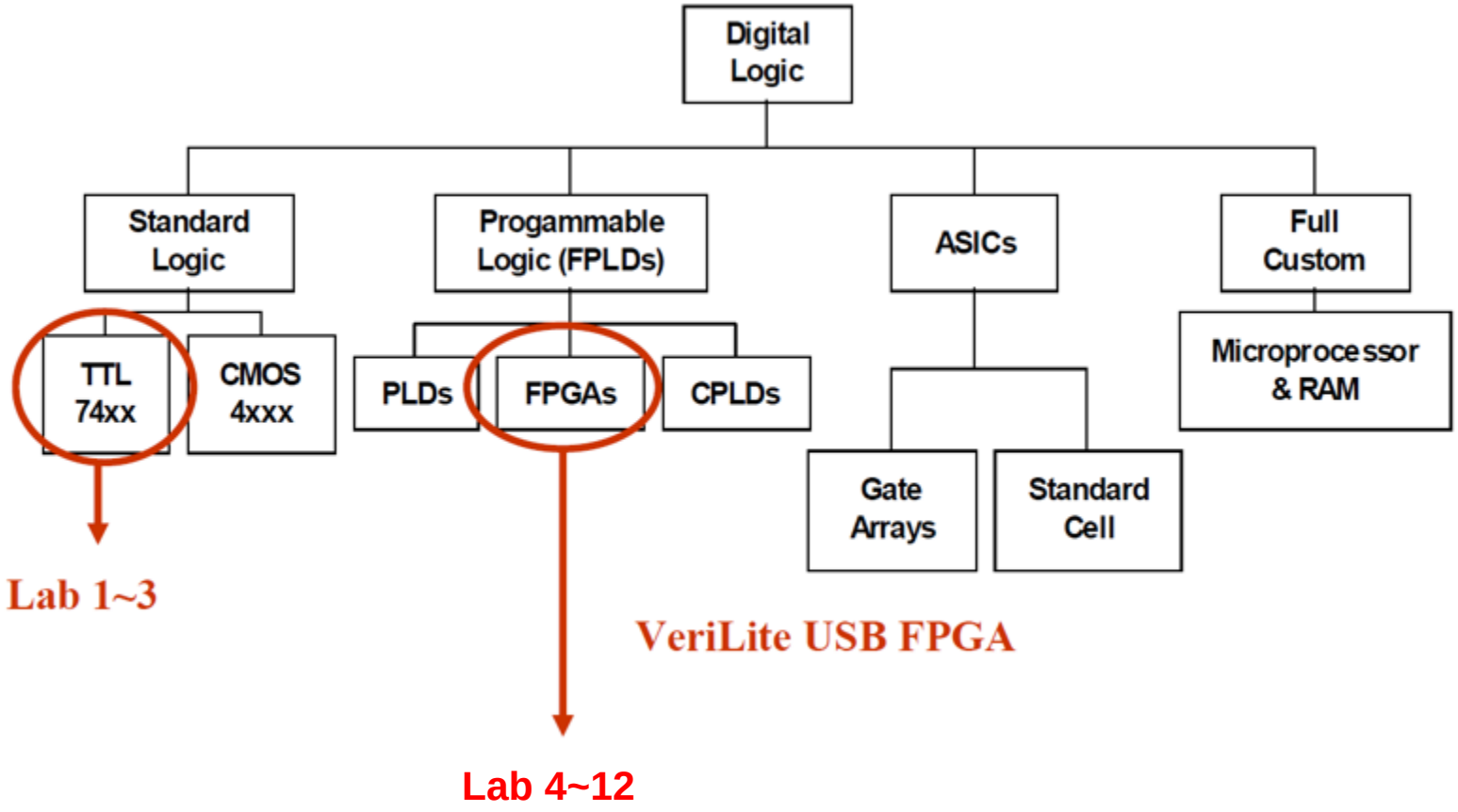
SMIMS VeriLite

FPGA 簡介

- 又稱**現場可程式邏輯閘陣列**，是一個可供使用者設定的邏輯閘元件。
- 目前以硬體描述語言(verilog或VHDL)所完成的電路設計，可以燒錄至FPGA上進行測試，是現代IC設計驗證的主流。
- 這些可編輯元件可以被用來實現基本的邏輯閘電路(ex:AND，OR，XOR，NOT)或者更複雜的功能，如解碼器或數學運算。
- 系統設計師可以依據需要通過可編輯的線路把FPGA內部的邏輯塊連接起來。

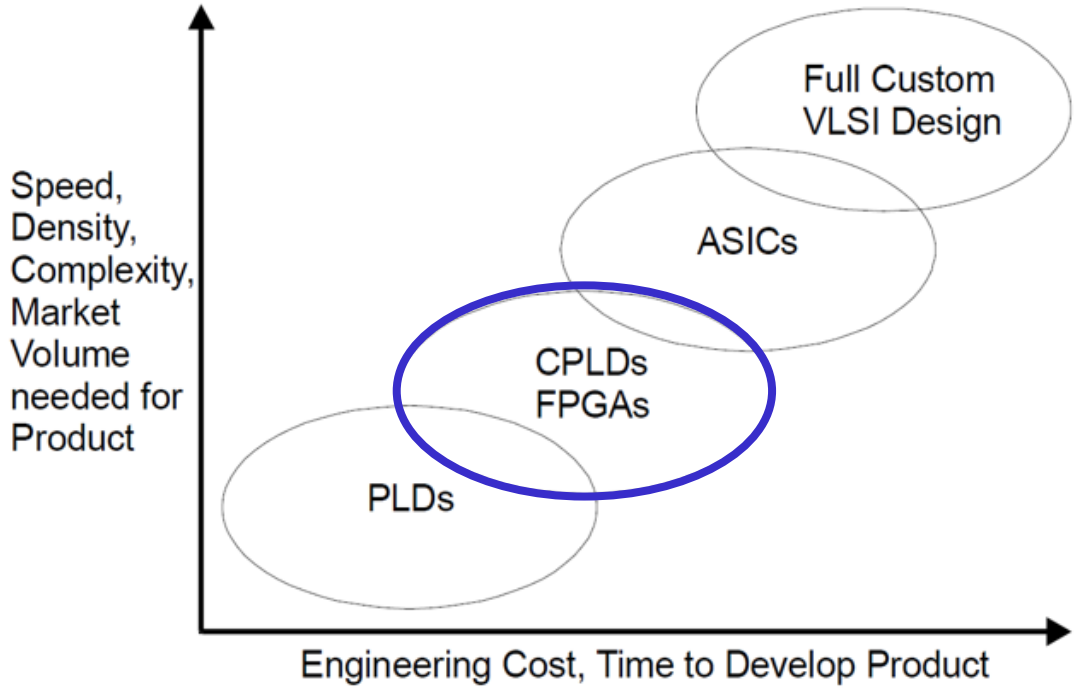
數位邏輯科技 (1/2)

➢本學期所相對應的學習地圖

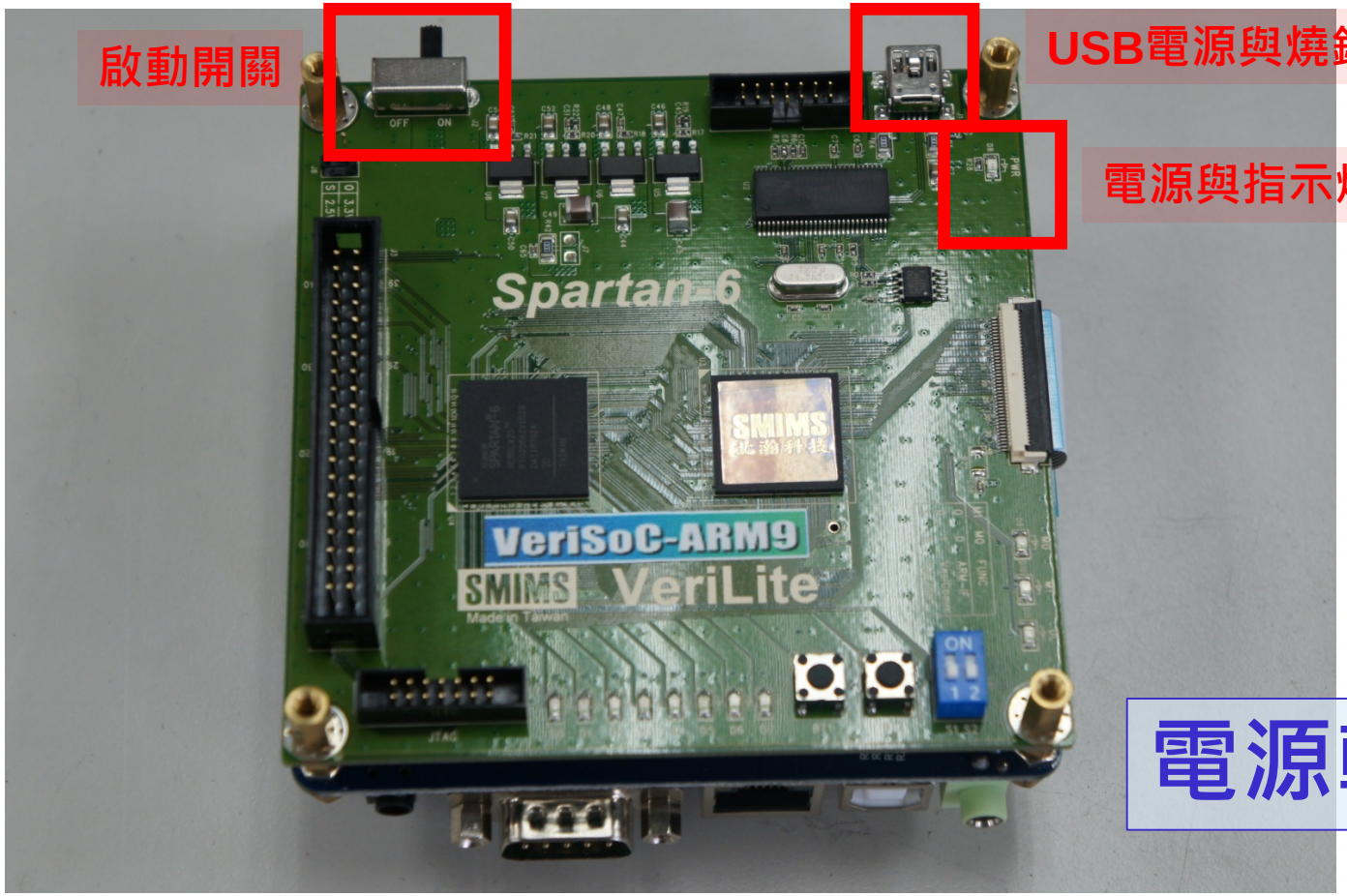


數位邏輯科技 (2/2)

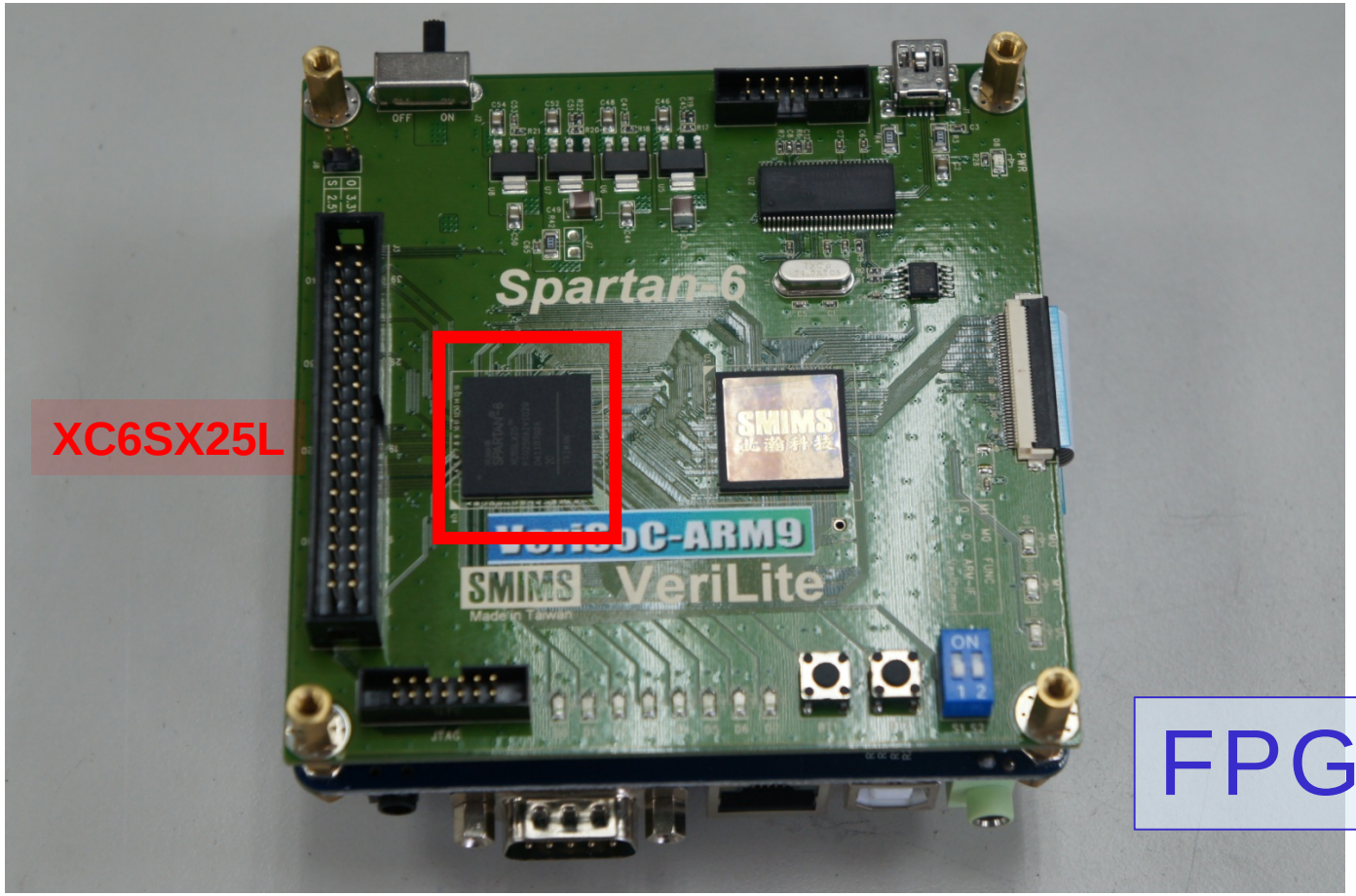
- FPGA提供數位元件設計者一個成本較低(相較於洗晶片)、且能快速產生成品的開發環境



VeriLite FPGA板 (1/4)

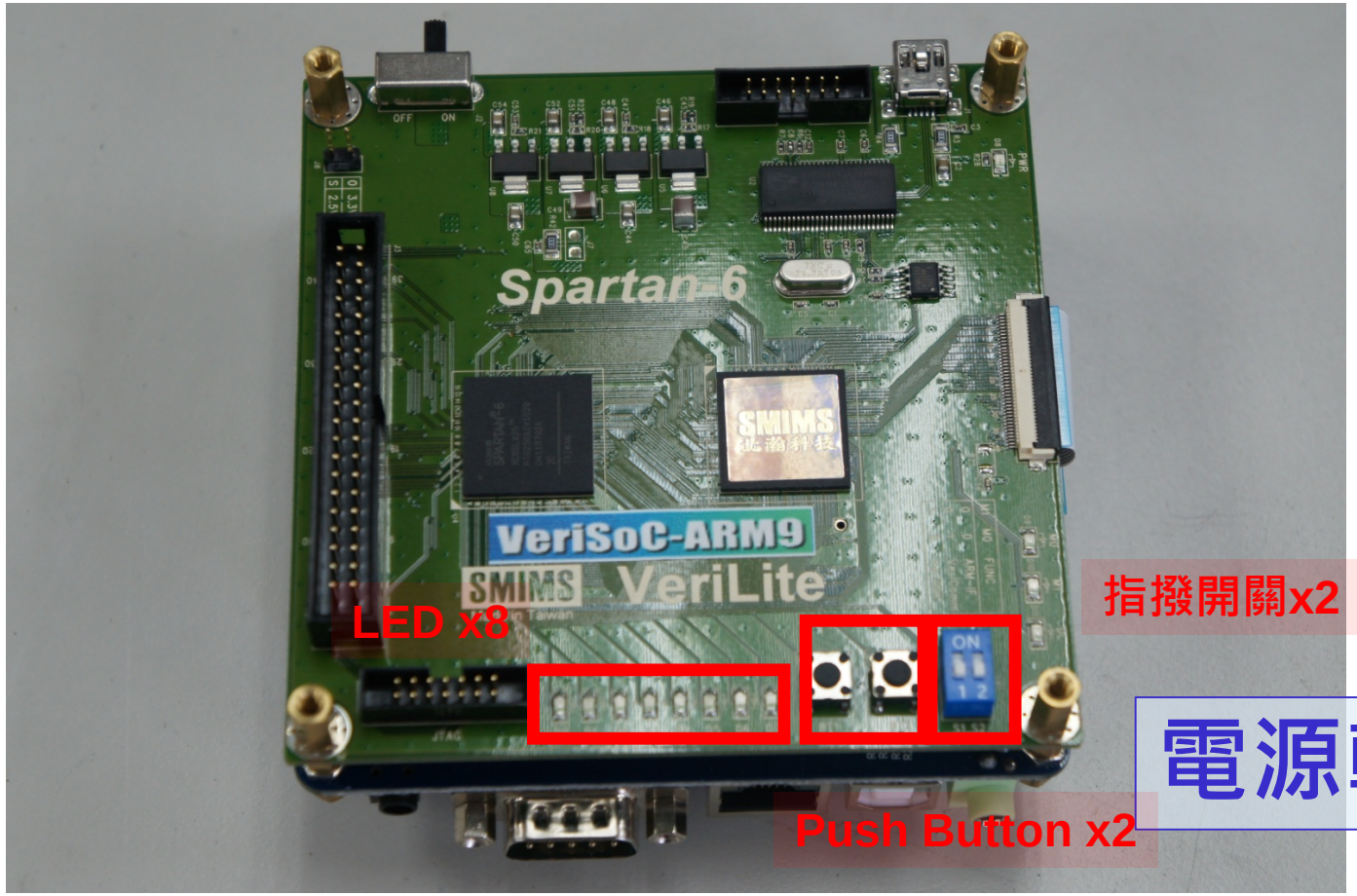


VeriLite FPGA板 (2/4)



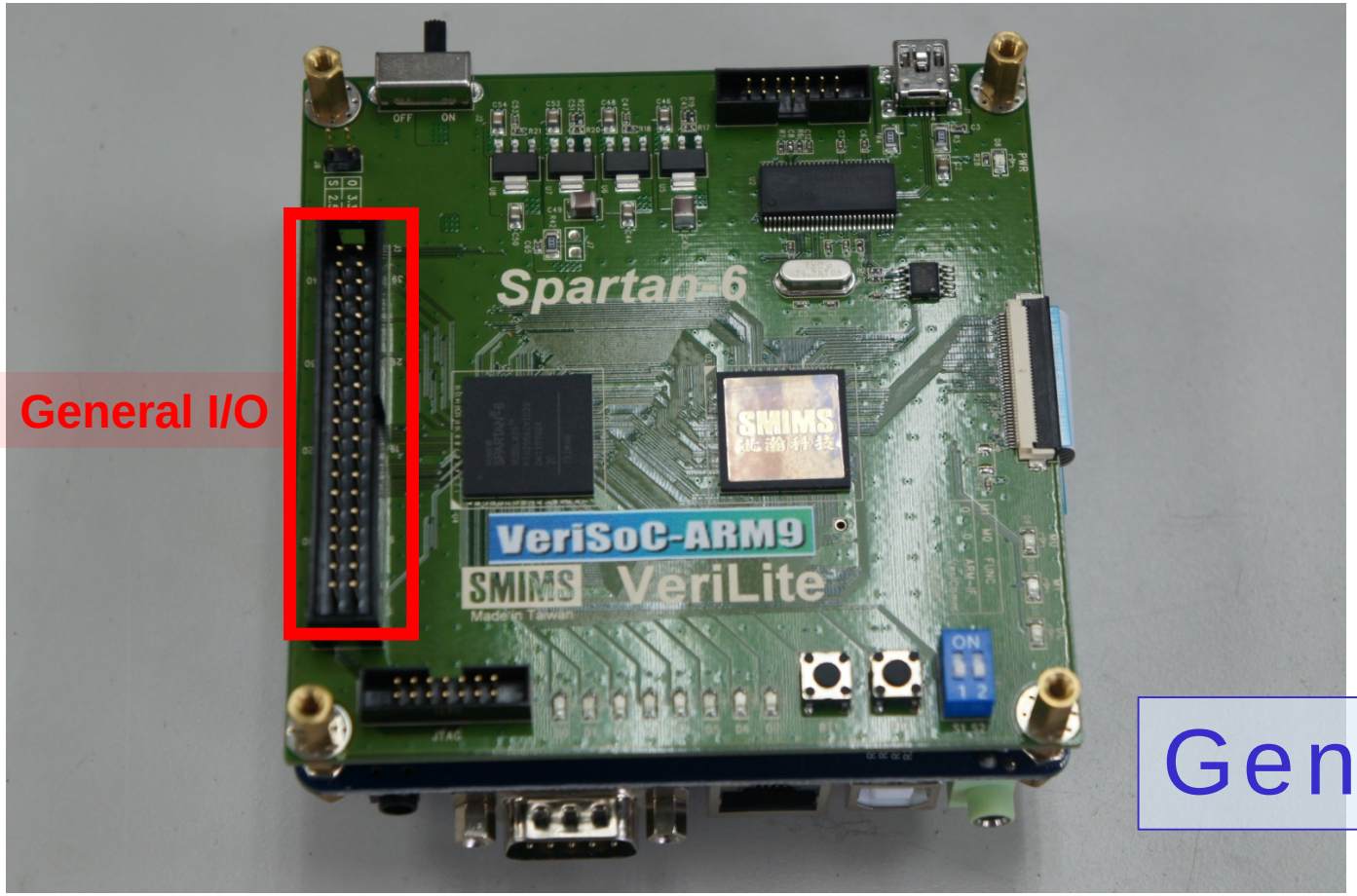
FPGA晶片

VeriLite FPGA板 (3/4)



電源輸入與開關

VeriLite FPGA板 (4/4)



General I/O

實作題

本次實作分為兩個基本實作與一個挑戰實作

實作題(一) 指撥開關與LED之使用

➢將邏輯電路燒入FPGA中並且驗證，主要分四個步驟

Step1. 設計邏輯電路

Step2. 產生腳位檔

Step3. 產生bit檔

Step4. 將bit檔燒入FPGA中執行

➢這個實作將會使用到下列三個工具:

- HDL Auto Assign pin: 根據板子的型號來設計與產生腳位檔
- ISE Project Navigator: 產生可燒入FPGA中的bit檔
- SMIMS FPGA programmer: 將產生的bit檔燒入FPGA中執行

實作題(一) Step 1 (1/2)

Step1. 設計邏輯電路

1. 請在 “c:\logiclab\<自己的學號>” 的路徑下新增一資料夾 lab6_1
2. 開啟 ISE ，並開啟一新 project
3. 在 working directory 的地方選擇 “c:\logiclab\<自己的學號>\lab6_1”
4. 將此 project 命名為 FPGA_practice
5. 專案設定如同上次實驗 (Family設為Spartan6, Device設為XC6SLX25)

實作題(一) Step 1 (2/2)

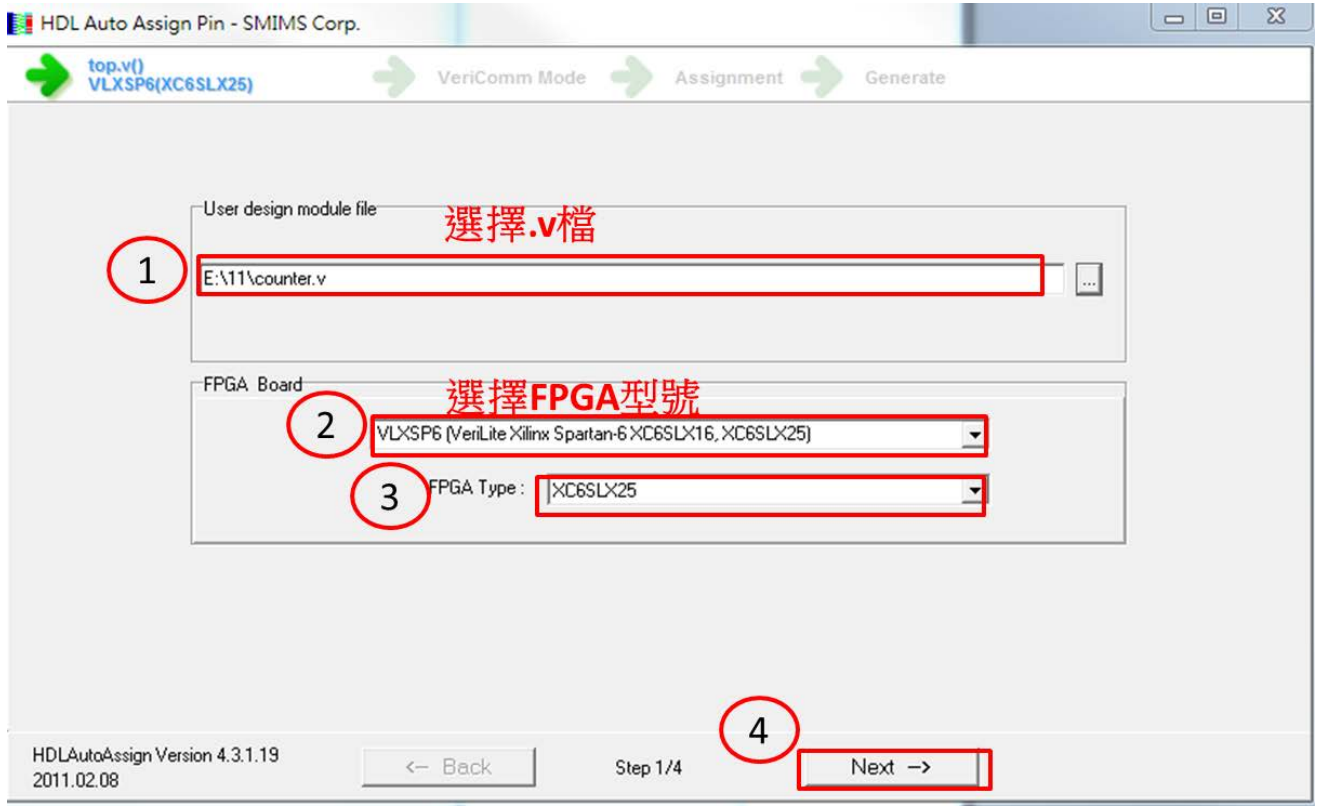
6. 在 File → New 下，選擇 Verilog HDL File，開啟一新 verilog 檔案。
7. 將以下內容鍵入此 Verilog 文件中，並儲存之(可隨意命名)

```
module FPGA_pratice(led0,led1,led2,led3,in0,in1);

    input in0,in1;
    output led0,led1,led2,led3;

    // low-active
    assign led0 = {in0,in1} == 2'b11 ? 1'b0 : 1'b1;
    assign led1 = {in1,in0} == 2'b10 ? 1'b0 : 1'b1;
    assign led2 = {in1,in0} == 2'b01 ? 1'b0 : 1'b1;
    assign led3 = {in1,in0} == 2'b00 ? 1'b0 : 1'b1;

endmodule
```

實作題(一) Step 2 (1/4)

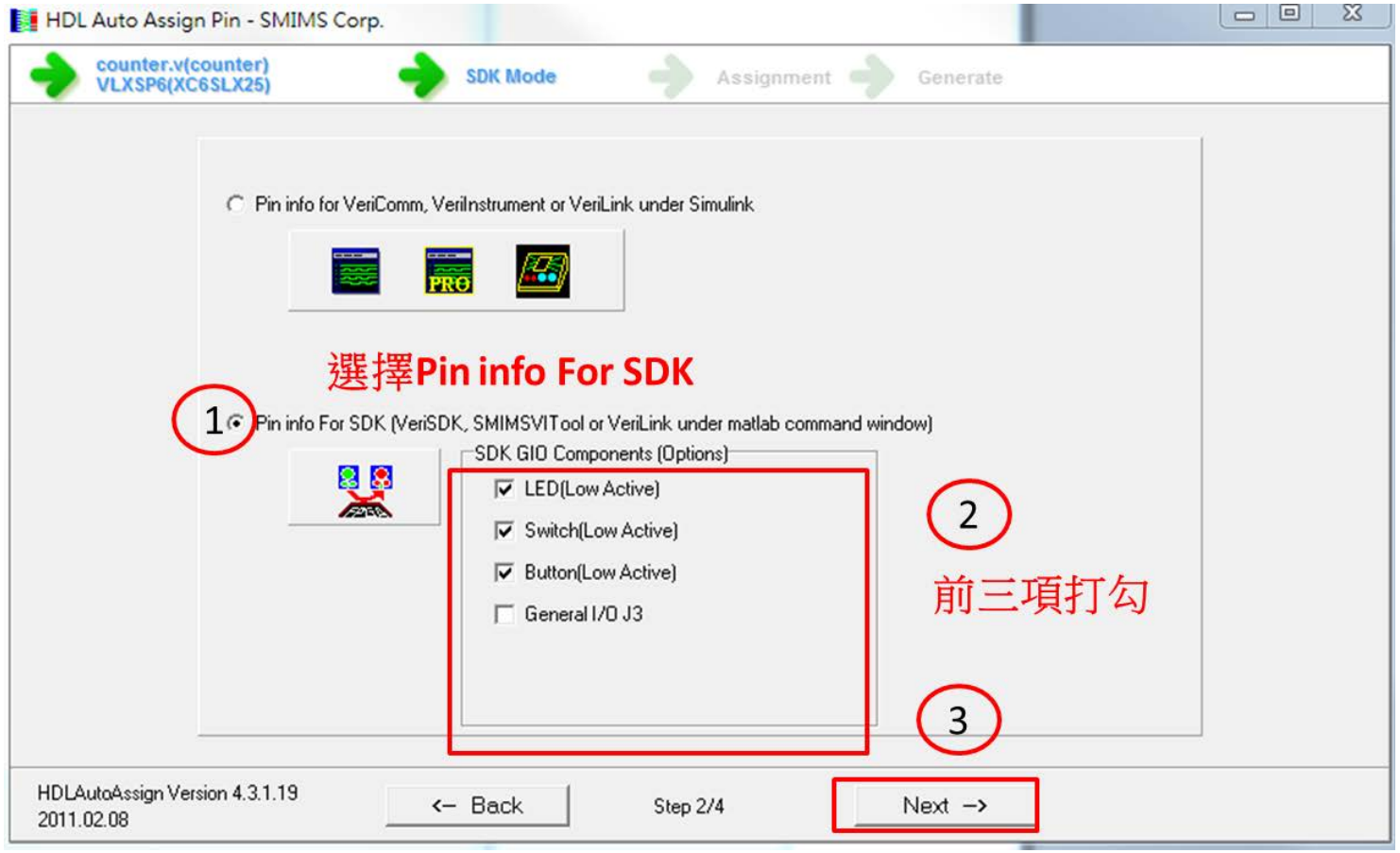
Step2. 透過HDL Auto Assign Pin產生腳位檔

1. 選取指定邏輯電路的Top Module 與 FPGA型號



實作題(一) Step 2 (2/4)

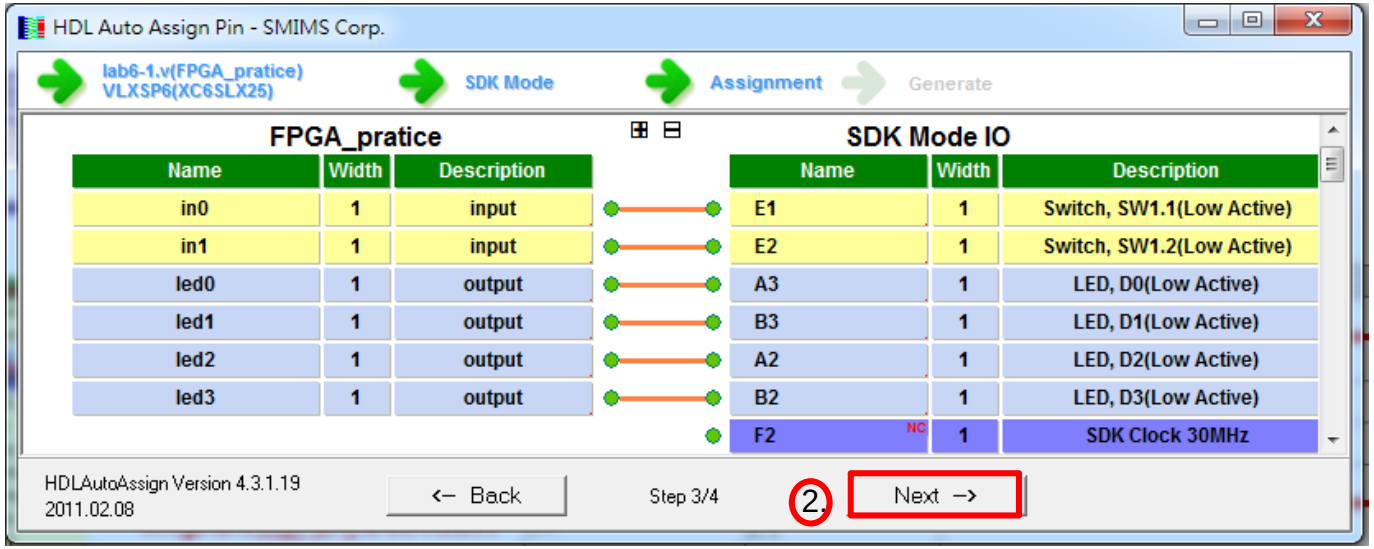
2. 選擇需要使用到的FPGA腳位



實作題(一) Step 2 (3/4)

3. 連接邏輯電路與FPGA的腳位

① **Input連到Switch**
Output連到LED



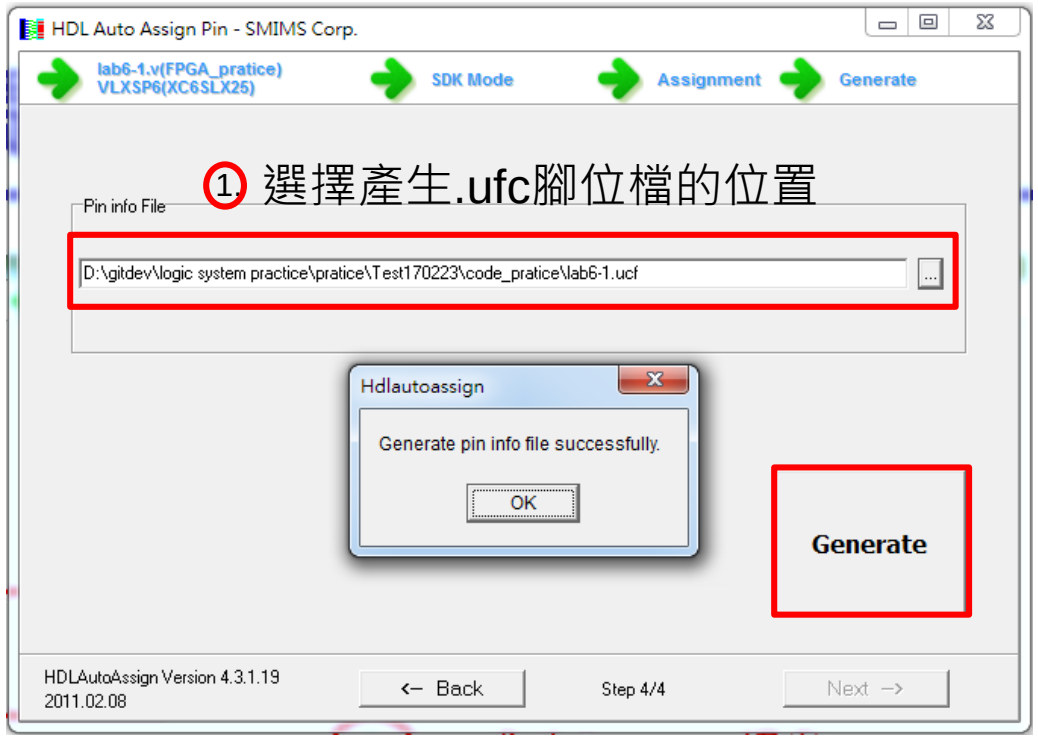
	LED (Low Active)	
D0	A3	LED on PCB
D1	B3	
D2	A2	
D3	B2	
D4	B1	
D5	C1	
D6	C2	
D7	C3	

	Switch (Low Active)
SW1.1	E1
SW1.2	E2

	Button (Low Active)	
BT1	D1	
BT2	D3	

實作題(一) Step 2 (4/4)

4. 產生腳位檔



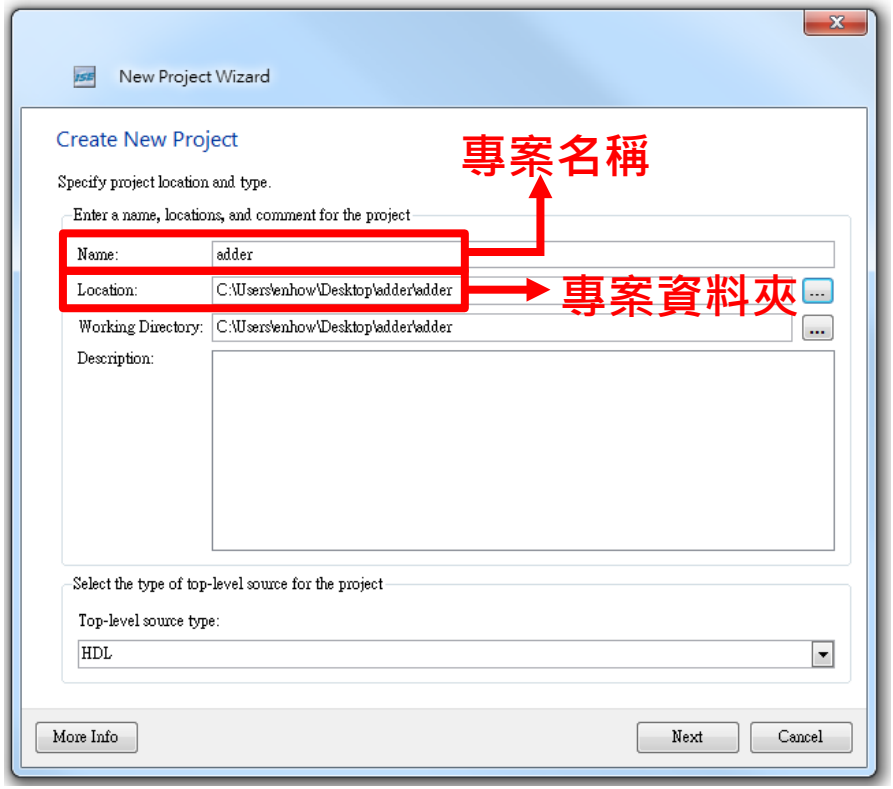

```
#PACE: Start of Constraints generated by PACE

#PACE: Start of PACE I/O Pin Assignments
NET "in0" LOC = "E1" | IOSTANDARD = LVCMOS33 ;
NET "in1" LOC = "E2" | IOSTANDARD = LVCMOS33 ;
NET "led0" LOC = "A3" | IOSTANDARD = LVCMOS33 ;
NET "led1" LOC = "B3" | IOSTANDARD = LVCMOS33 ;
NET "led2" LOC = "A2" | IOSTANDARD = LVCMOS33 ;
NET "led3" LOC = "B2" | IOSTANDARD = LVCMOS33 ;
#PACE: Start of PACE Area Constraints

#PACE: Start of PACE Prohibit Constraints

#PACE: End of Constraints generated by PACE
```

② 按下Generate產生腳位檔，
該檔案紀錄Top Module的I/O
如何對應到FPGA的I/O

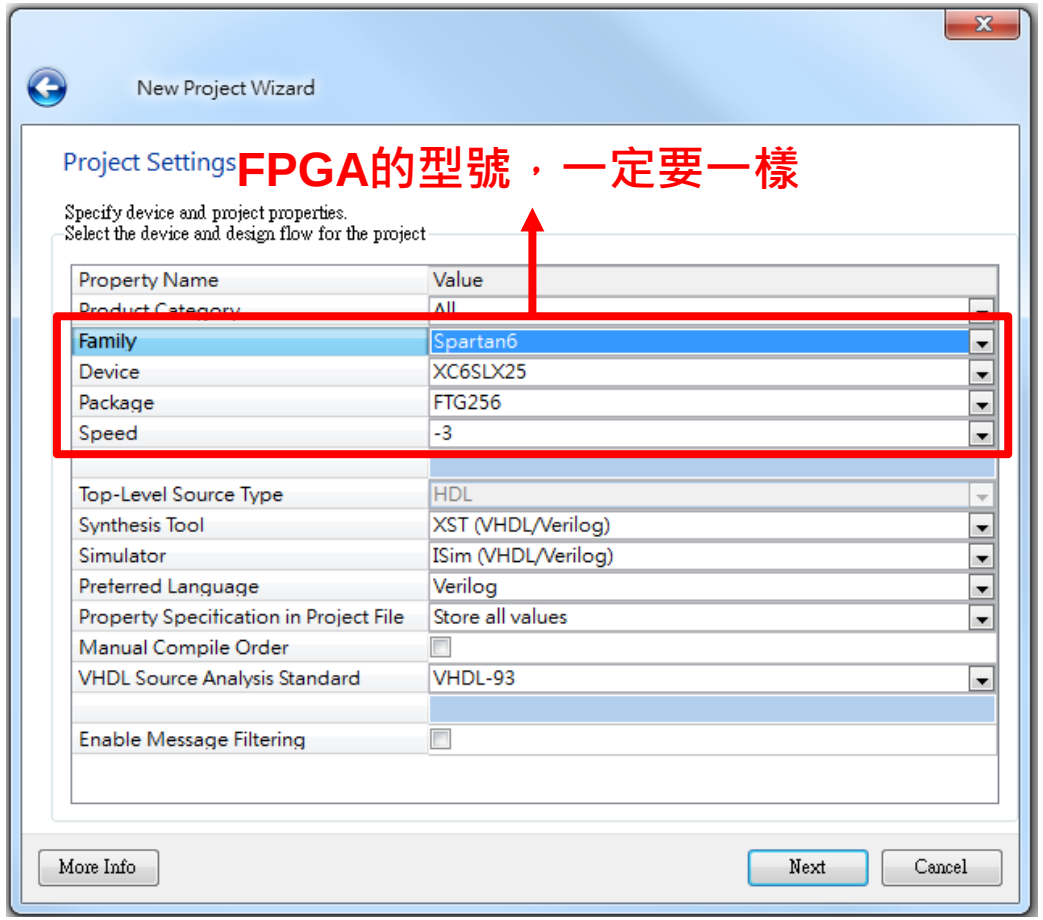
實作題(一) Step 3 (1/4)

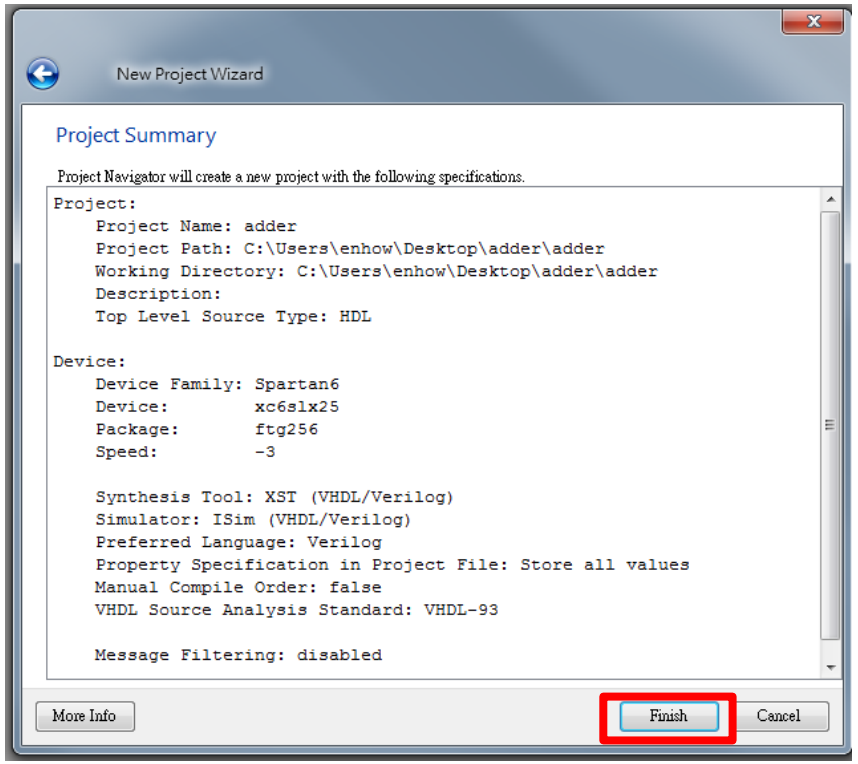
Step3. 透過ISE Project Navigator產生可燒入FPGA的bit檔

1. 開啟 ISE ，並開啟一新 project (前面有做的話可以跳過)



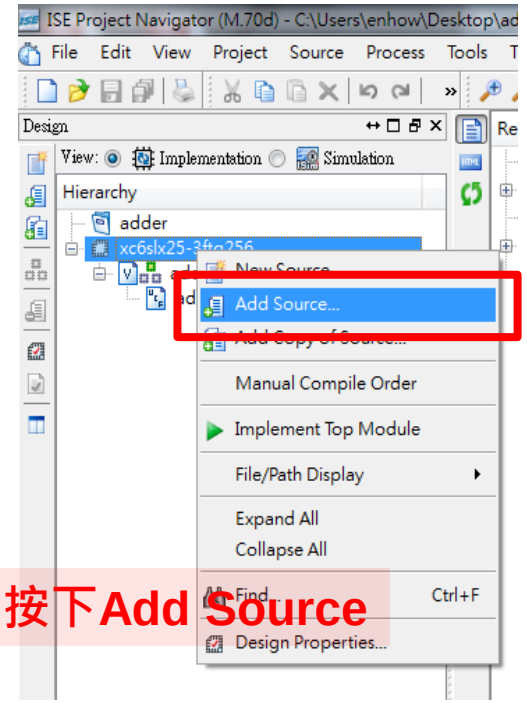
實作題(一) Step 3 (2/4)

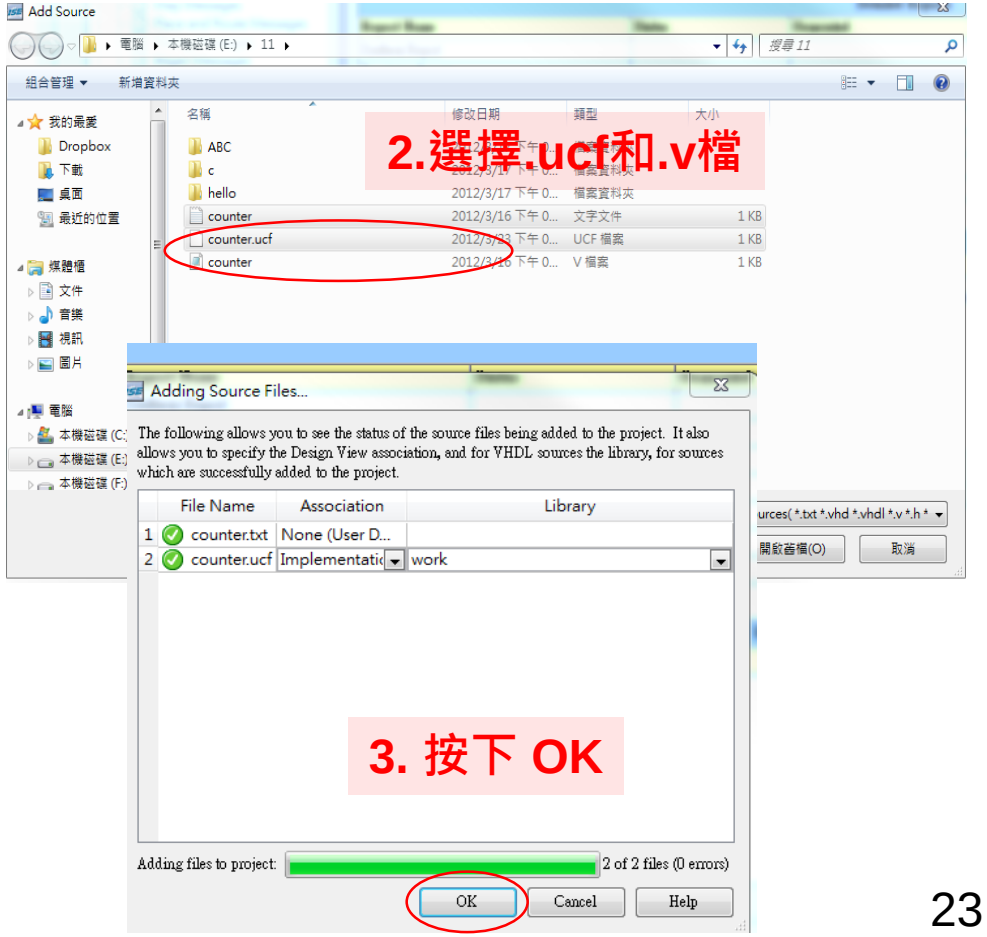




實作題(一) Step 3 (3/4)

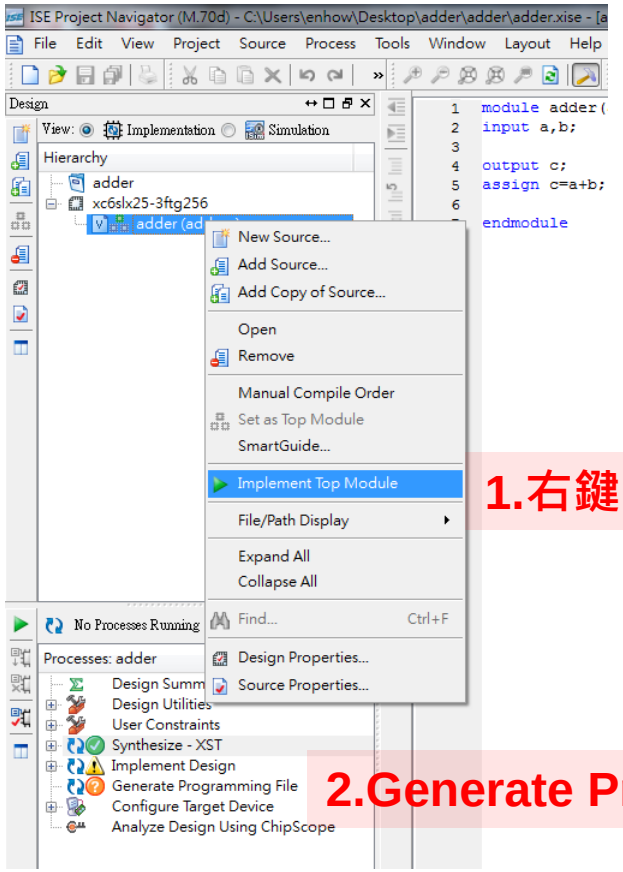
2. 加入Verilog檔與ucf檔





實作題(一) Step 3 (4/4)

3. 編譯並且產生bit檔

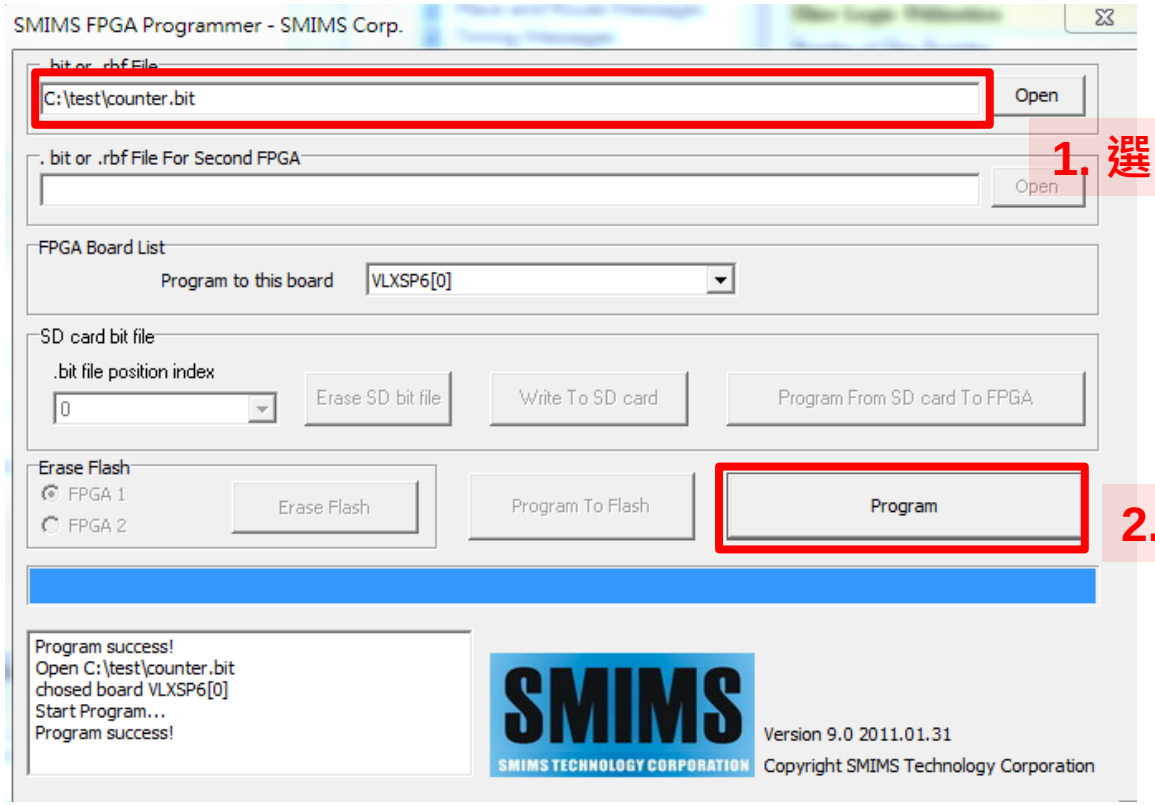


1.右鍵按下，並選擇Implement Top Module

2.Generate Programming File → 按右鍵 → run

實作題(一) Step 4

Step4. 透過SMIMS FPGA programmer將bit檔燒入FPGA

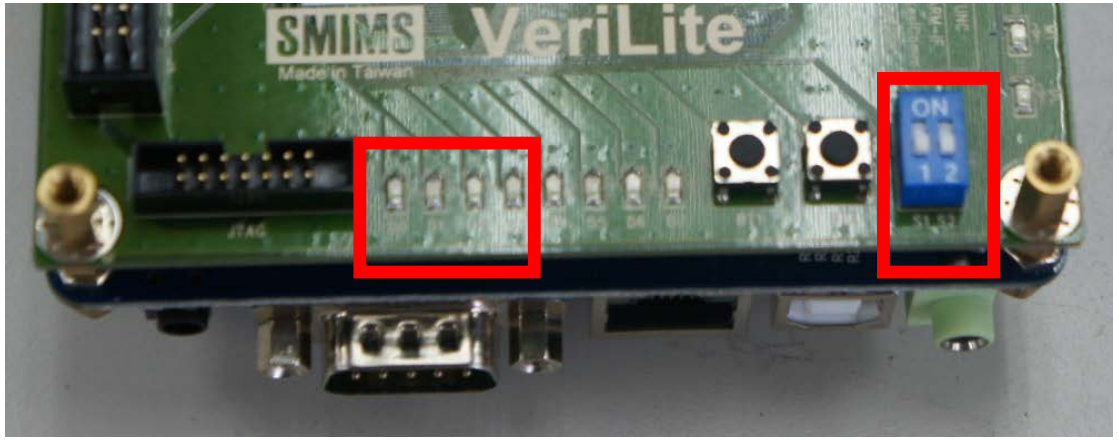


1. 選擇 bit檔

2.”Program”

實作題(一)指撥開關與LED之使用

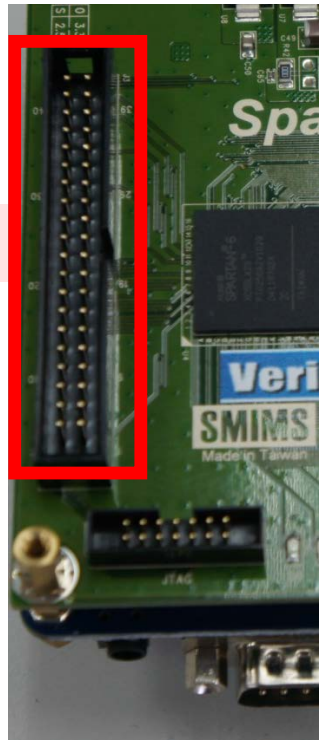
- 燒入的邏輯電路為解碼器
- 驗證時，請注意Switch與LED燈下方的編號是否與預期結果相符



實作題(二) General I/O (1/2)

- 燒入FPGA的邏輯電路除了可以利用FPAG上原有的元件，也可以透過General I/O來與外部的實體電路直接連接
- VeriLite FPGA 的General I/O放置在J3腳位群
  - 使用時請注意左右側的編號
  - 其中有些腳位為VCC與GND

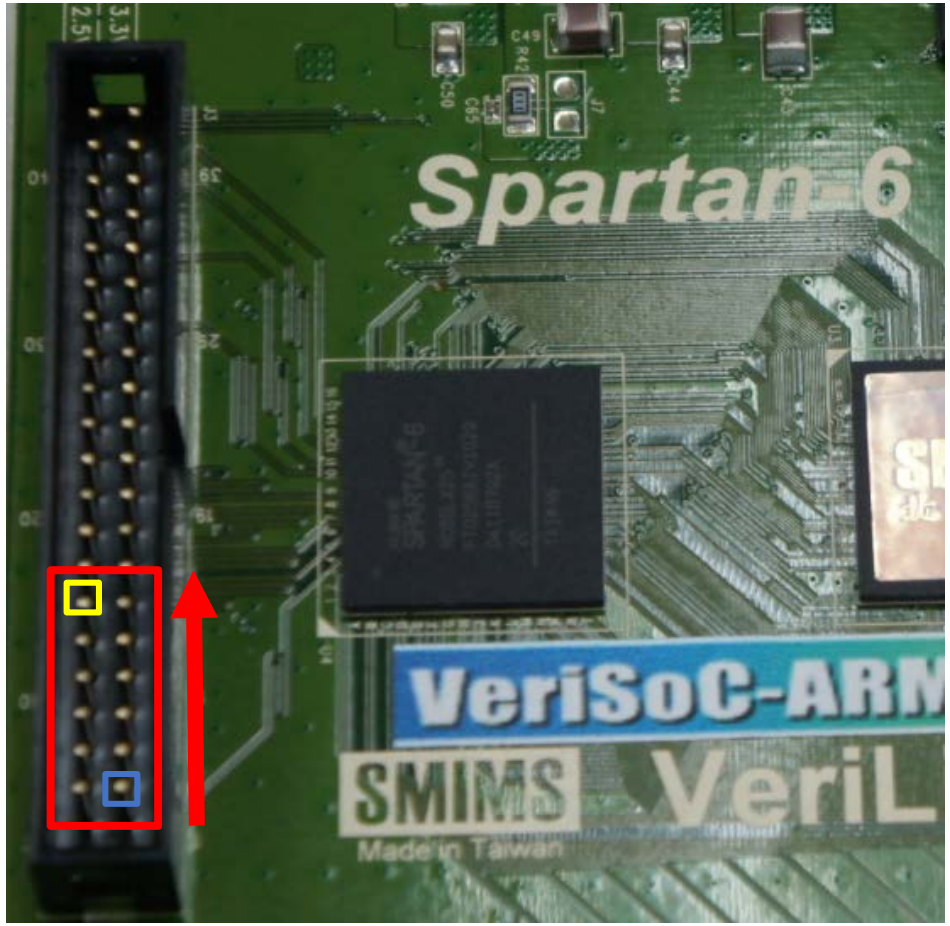
    (請勿把VCC與GND短接，會燒壞板子)



實作題(二) General I/O (2/2)

		General I/O	J3
D5	J3.1	J3.2	B5
C5	J3.3	J3.4	A5
F7	J3.5	J3.6	D6
E6	J3.7	J3.8	C6
E7	J3.9	J3.10	E8
VCC5V	J3.11	J3.12	GND

(功能,腳位代號) (腳位代號,功能)



實作題(二) Step 1 (1/2)

Step1. 設計邏輯電路

1. 請在 “c:\logiclab\<自己的學號>” 的路徑下新增一資料夾 lab6_2
2. 開啟 ISE ，並開啟一新 project
3. 在 working directory 的地方選擇 “c:\logiclab\<自己的學號>\lab6_2”
4. 將此 project 命名為 FPGA_practice2
5. 專案設定如同上次實驗 (Family設為Spartan6, Device設為XC6SLX25)

實作題(二) Step 1 (2/2)

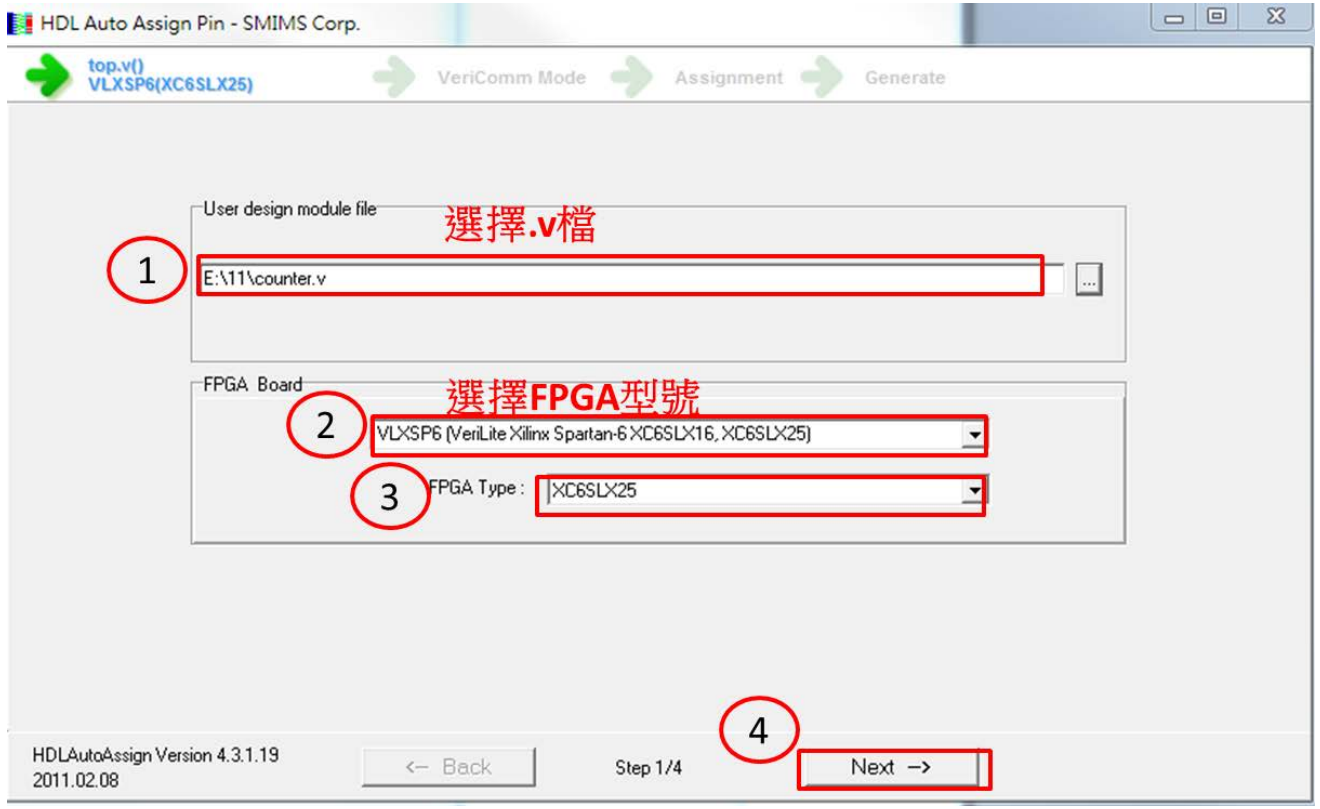
6. 在 File → New 下，選擇 Verilog HDL File，開啟一新 verilog 檔案。
7. 將以下內容鍵入此 Verilog 文件中，並儲存之(可隨意命名)

```
module FPGA_pratice2(led0,led1,in0,in1,in2);

    input in0,in1,in2;// low-active
    output led0,led1;

    assign led0 = ~in2 | (in2 & in1 & ~in0);
    assign led1 = ~in2 | (in2 & ~in1);
endmodule
```

實作題(二) Step 2 (1/4)

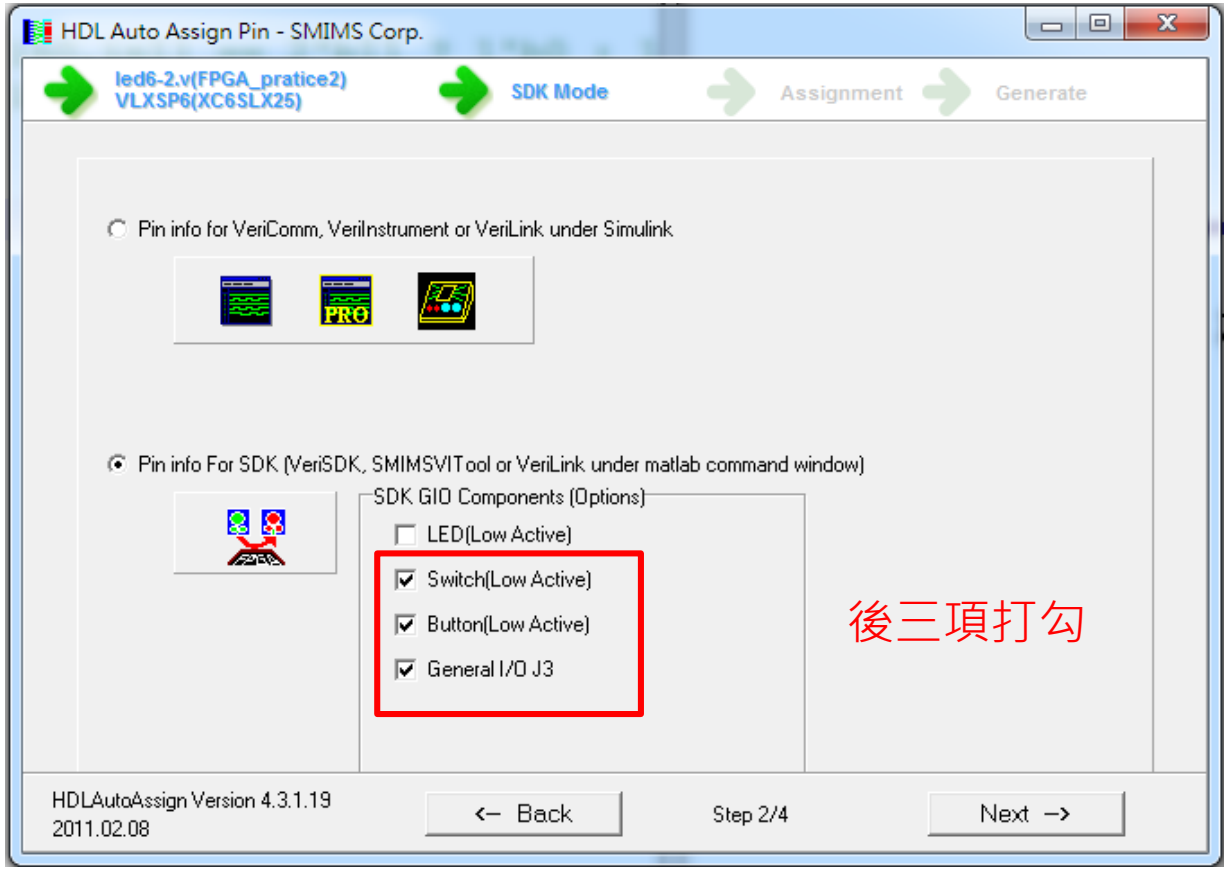
Step2. 透過HDL Auto Assign Pin產生腳位檔

1. 選取指定邏輯電路的Top Module 與 FPGA型號



實作題(二) Step 2 (2/4)

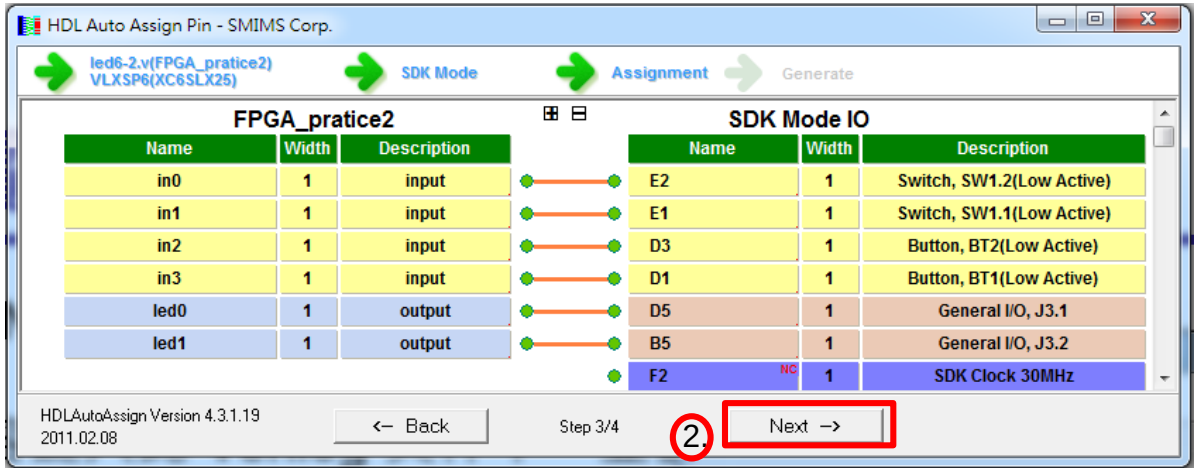
2. 選擇需要使用到的FPGA腳位



實作題(二) Step 2 (3/4)

3. 連接邏輯電路與FPGA的腳位

① **Input連到Switch**
Output連到General I/O
(注意編號)



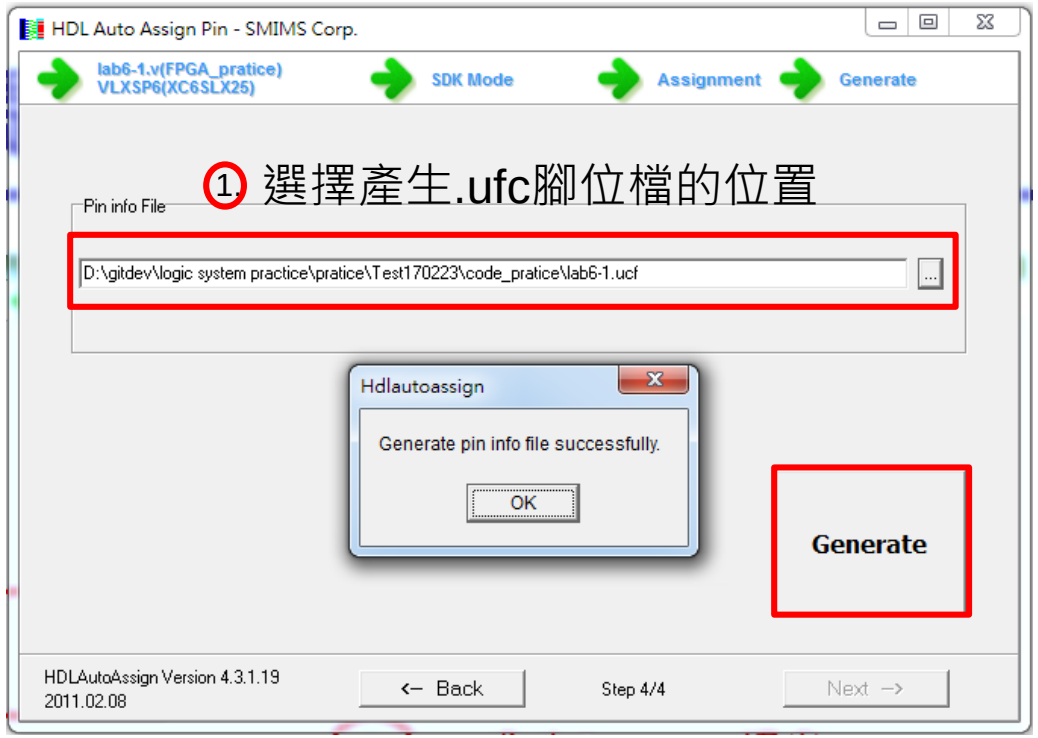
		General I/O J3	
D5	J3.1	J3.2	B5
C5	J3.3	J3.4	A5
F7	J3.5	J3.6	D6
E6	J3.7	J3.8	C6
E7	J3.9	J3.10	E8
VCC5V	J3.11	J3.12	GND

Switch (Low Active)	
SW1.1	E1
SW1.2	E2

Button (Low Active)	
BT1	D1
BT2	D3

實作題(二) Step 2 (4/4)

4. 產生腳位檔



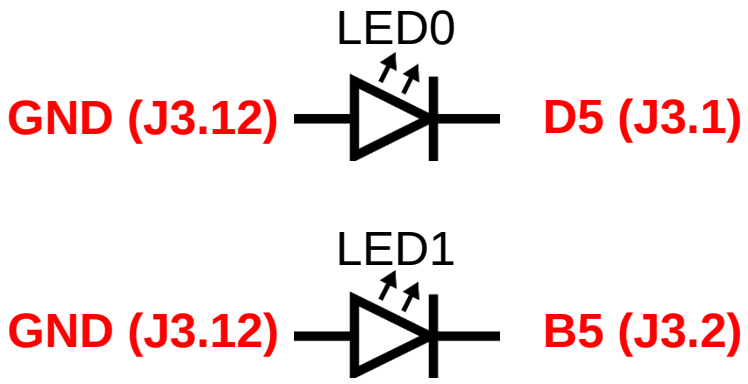
② 按下Generate產生腳位檔

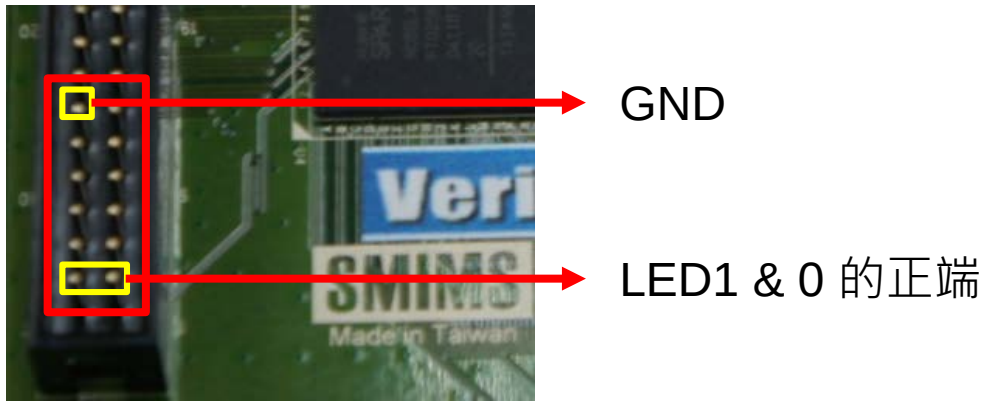
實作題(二) Step 3 & 4

Step3. 透過ISE Project Navigator產生可燒入FPGA的bit檔

Step4. 透過SMIMS FPGA programmer將bit檔燒入FPGA

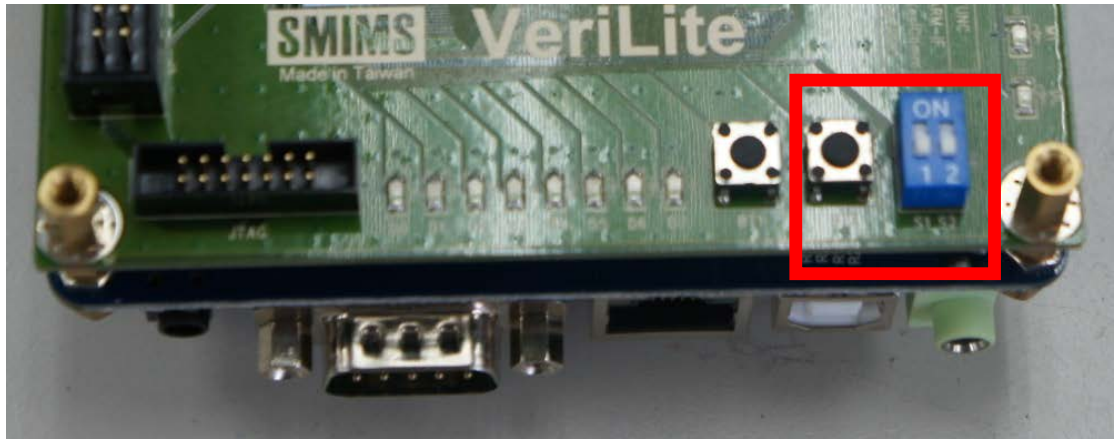
燒入完成後，將FPGA板與LED電路相接





實作題(二) General I/O

- 燒入的邏輯電路為編碼器
- 驗證時，請注意Switch與Button下方的編號是否與預期結果相符



挑戰題(一) 七段顯示器 (1/4)

- 將七段顯示器的解碼電路燒入FPGA中，設定腳位檔，再與實體七段顯示器相接驗證

- 請參考前面實驗的步驟與後面的提示完成本實作

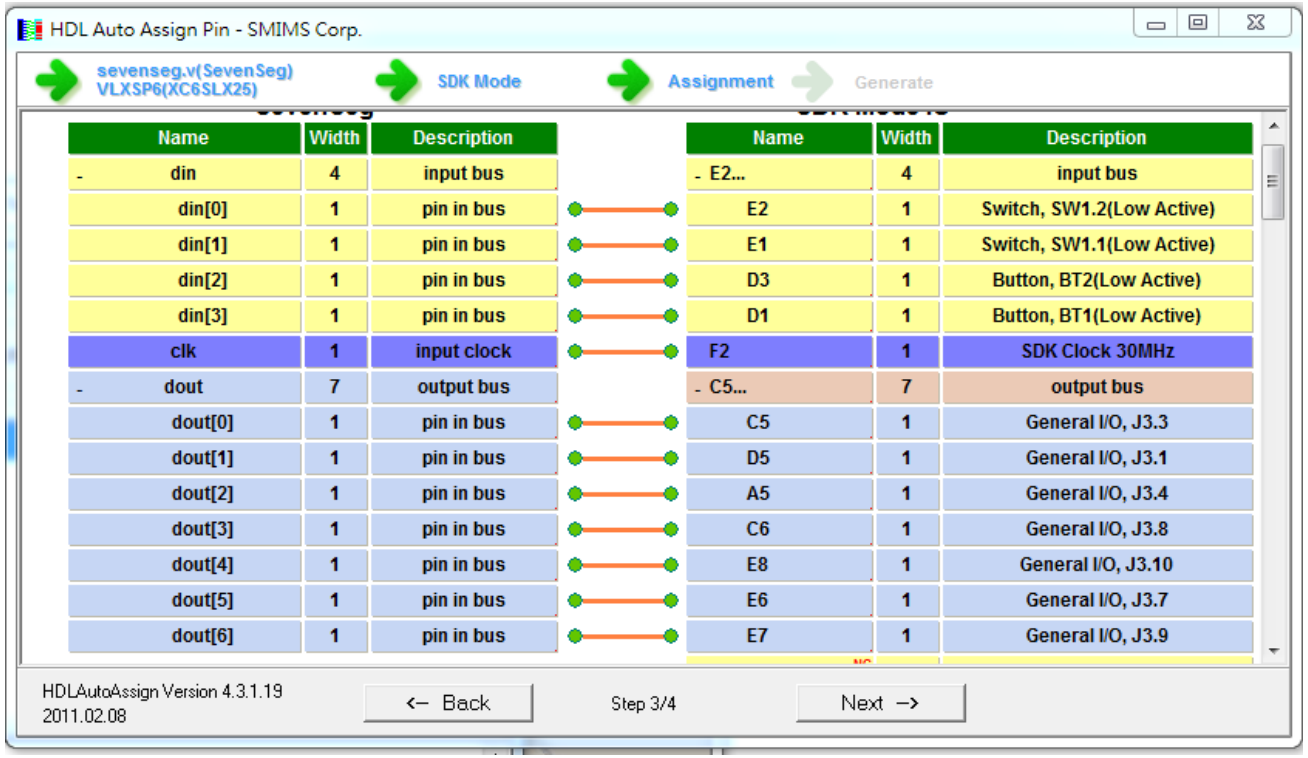
挑戰題(一) 七段顯示器 (2/4)

Hint1: 七段顯示器的解碼器電路

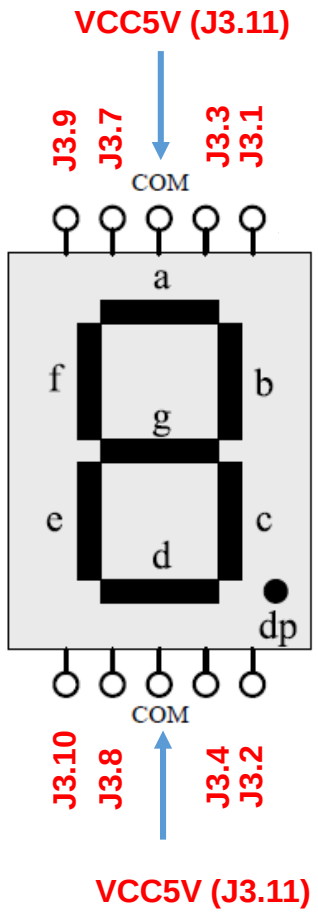
```
module SevenSeg(dout, din, clk);//lab6-3
    input    [3:0]    din;
    input             clk;
    output reg[6:0]      dout;

    always@(din)
    begin
        case(din)
            4'b1111:dout=7'b1000_000;
            4'b1110:dout=7'b1111_001;
            4'b1101:dout=7'b0100_100;
            4'b1100:dout=7'b0110_000;

            4'b1011:dout=7'b0011_001;
            4'b1010:dout=7'b0010_010;
            4'b1001:dout=7'b0000_010;
            4'b1000:dout=7'b1111_000;

            4'b0111:dout=7'b0000_000;
            4'b0110:dout=7'b0010_000;
            4'b0101:dout=7'b0001_000;
            4'b0100:dout=7'b0000_011;

            4'b0011:dout=7'b1000_110;
            4'b0010:dout=7'b0100_001;
            4'b0001:dout=7'b0000_110;
            4'b0000:dout=7'b0001_110;
            default:dout=7'b1111_111;
        endcase
    end
endmodule
```

挑戰題(一) 七段顯示器 (3/4)

Hint2: 腳位設定與硬體接線



Name	Width	Description	Name	Width	Description
- din	4	input bus	- E2...	4	input bus
din[0]	1	pin in bus	E2	1	Switch, SW1.2(Low Active)
din[1]	1	pin in bus	E1	1	Switch, SW1.1(Low Active)
din[2]	1	pin in bus	D3	1	Button, BT2(Low Active)
din[3]	1	pin in bus	D1	1	Button, BT1(Low Active)
clk	1	input clock	F2	1	SDK Clock 30MHz
- dout	7	output bus	- C5...	7	output bus
dout[0]	1	pin in bus	C5	1	General I/O, J3.3
dout[1]	1	pin in bus	D5	1	General I/O, J3.1
dout[2]	1	pin in bus	A5	1	General I/O, J3.4
dout[3]	1	pin in bus	C6	1	General I/O, J3.8
dout[4]	1	pin in bus	E8	1	General I/O, J3.10
dout[5]	1	pin in bus	E6	1	General I/O, J3.7
dout[6]	1	pin in bus	E7	1	General I/O, J3.9



挑戰題(一) 七段顯示器 (4/4)

- 燒入的電路為七段顯示器的解碼器
- 驗證時，請注意Switch與Button下方的編號是否與預期結果相符