邏輯系統實習

實驗七

虛擬儀器(二) + Verilog語法介紹(四)：行為層次-序向電路

國立成功大學 電機系

2016

大綱

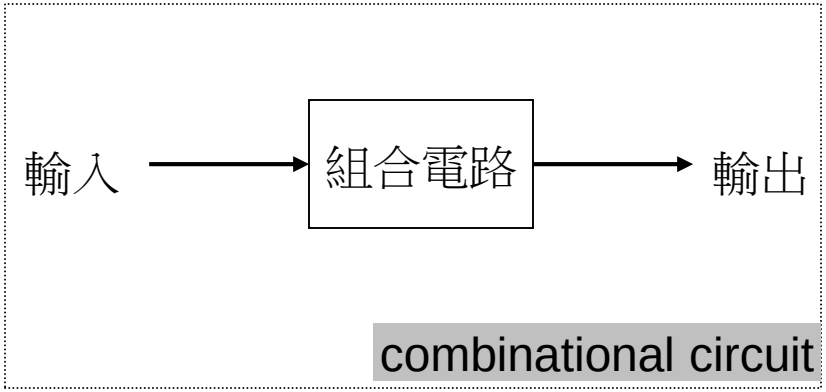
- Verilog 中的四種描述層次
- 組合電路與序向電路之比較
- 程序區塊
- 時序控制
  - 邊緣觸發
  - 準位觸發與邊緣觸發之比較
- 程序指定
  - 無阻礙指定
  - 阻礙指定與無阻礙指定之比較
- Verilog 編碼風格
- 除頻器
- 基礎題 (一)
  - 4bit計數器與七段顯示器
- 基礎題 (二)
  - 6bit串入並出移位器與七段顯示器
- 挑戰題
  - 計時器
- 實驗結報繳交

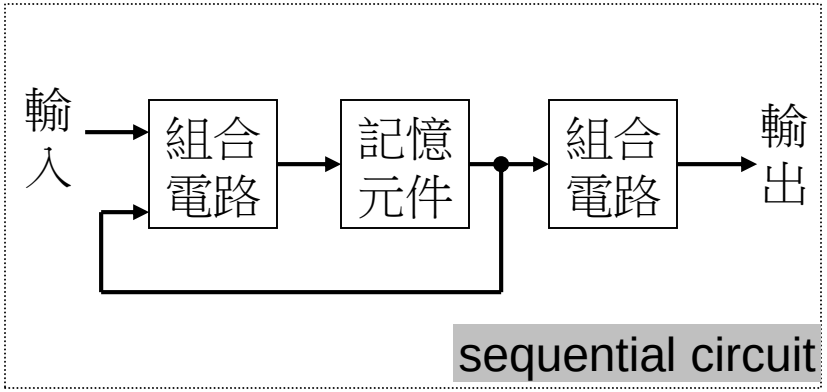
Verilog中的四種描述層次

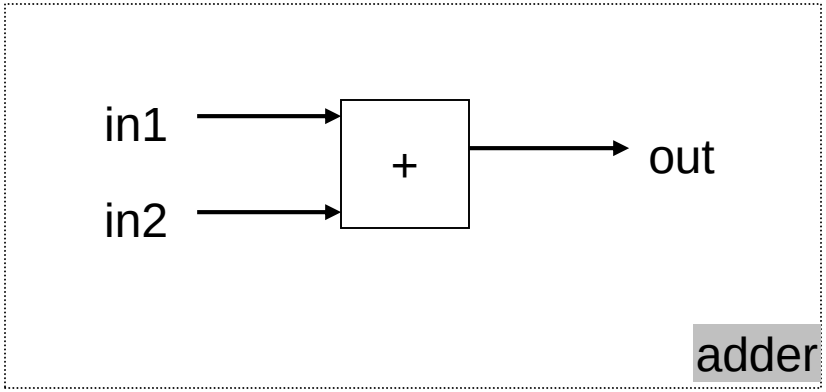
- 行為或演算法層次 (lab6~)
  - 在這個層次中，只需要考慮模組的功能或演算法，不需要考慮硬體方面的詳細電路是如何。
- 資料處理層次 (lab5)
  - 在這個層次中，只需要指名資料處理的方式，像是資料如何在暫存器中儲存與傳送以及資料在設計裡處理的方式。
- 邏輯閘層次 (lab4)
  - 在這個層次中，模組是邏輯閘連接而成，如同以前用邏輯閘描繪電路一樣。
- 低階交換層次 (x)
  - 在這個層次中，線路是由開關與儲存點構成，需要知道電晶體的元件特性。

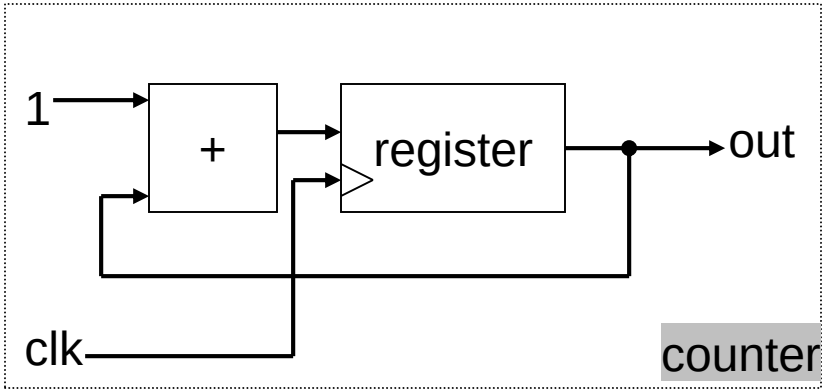
越上層設計層次越高

組合電路與序向電路之比較



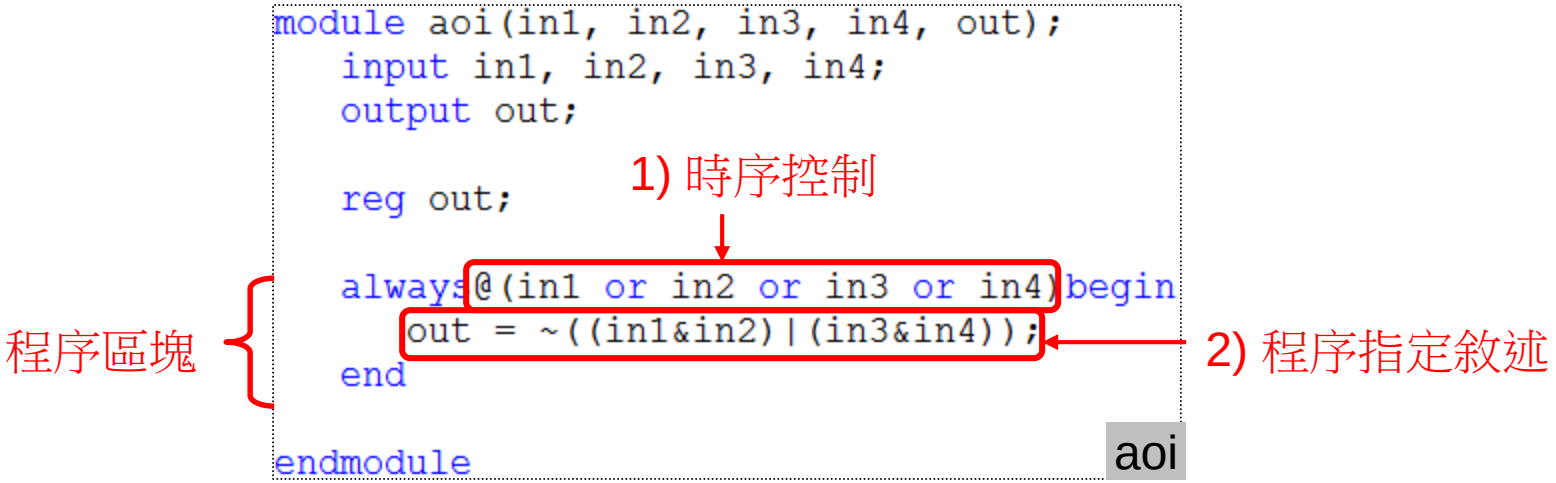






程序區塊

- 程序區塊包含兩個要素：
  - 1) 時序控制
  - 2) 程序指定敘述



時序控制 (1/3)

- 時序控制是用來設定某程序敘述在某時間被執行，共有三種方式。
  - 1) 簡易延遲 (lab6)
  - 2) 準位觸發 (lab6)
  - 3) 邊緣觸發

```
module alu_4bit(sel, in1, in2, result);
   input [2:0]sel;
   input [3:0]in1, in2;
   output [3:0]result;

   reg [3:0]result;

   always@(sel or in1 or in2)begin
      case(sel)
         3'b000: result=in1+in2;
         3'b001: result=in1-in2;
         3'b010: result=in1&in2;
         3'b011: result=in1|in2;
         3'b100: result=in1^in2;
         3'b101: result=in1>>in2;
         3'b110: result=in1<<in2;
         3'b111: result=in1>in2;
         default: result=4'b0;
      endcase
   end

endmodule
```

2) 準位觸發

alu_4bit

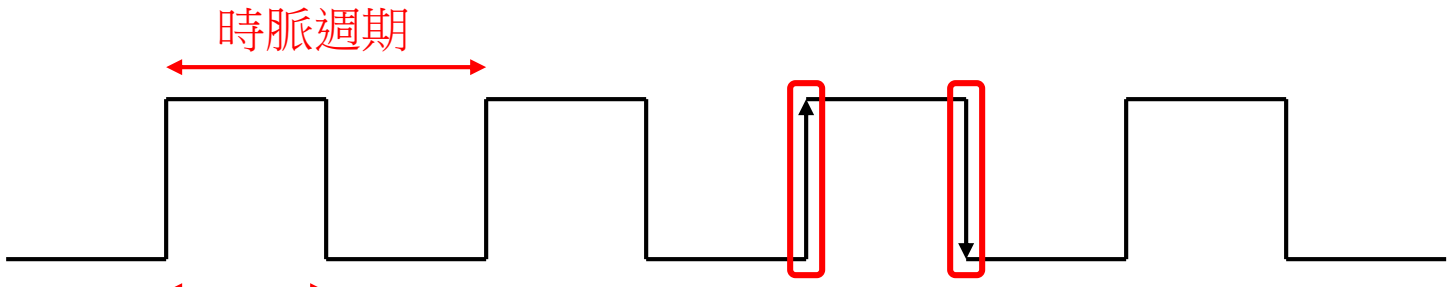
```
module counter_4bit(clk, rst, out);
   input clk, rst;
   output [3:0]out;

   reg [3:0]out;

   always@(posedge clk or posedge rst)begin
      if(rst)out<=4'b0;
      else out<=out+4'b1;
   end

endmodule
```

3) 邊緣觸發

counter_4bit

時序控制 (2/3)
邊緣觸發

- 邊緣觸發又稱為同步觸發，當信號產生正邊緣或負邊緣轉換時，敘述才會被執行。

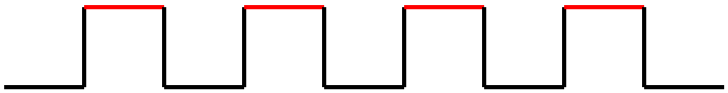


時脈(clock)是系統中所有元件辨識信號參考的準位

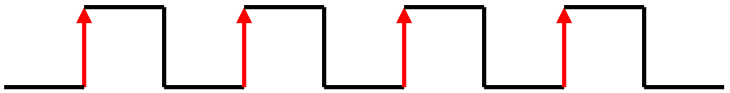
時脈頻率 = 1/時脈周期

時序控制 (3/3)

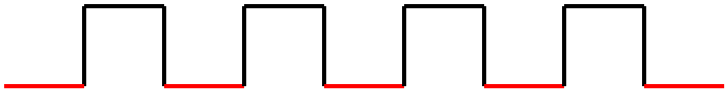
準位觸發與邊緣觸發之比較



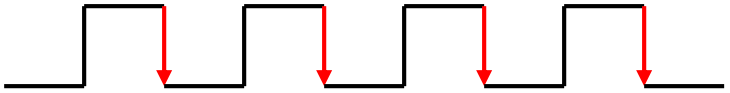
正準位觸發



正邊緣觸發



負準位觸發



負邊緣觸發

```
module latch(clk, rst, in, out);
   input clk, rst, in;
   output out;

   reg out;

   always@(clk or rst or in)begin
      if(clk)begin
         if(rst)out=1'b0;
         else out=in;
      end
   end
endmodule
```

positive level-triggered latch

```
module flipflop(clk, rst, in, out);
   input clk, rst, in;
   output out;

   reg out;

   always@(posedge clk or posedge rst)begin
      if(rst)out<=1'b0;
      else out<=in;
   end
endmodule
```

positive edge-triggered flip flop

程序指定 (1/2)
無阻礙指定

- 程序指定可以用來更新暫存器等變數。被指定的變數值將被保持到下次被更新為止。
  - 在程序指定的左邊必須是暫存器(reg)或其他變數等等；而在敘述右邊的運算元可以是任意運算式。
- 程序指定有兩種形態：
  - 1) 阻礙指定 (lab6)
  - 2) 無阻礙指定

無阻礙指定(<=)敘述的執行順序，不會受到敘述位置前後的影響

```
module counter_4bit(clk, rst, out);
   input clk, rst;
   output [3:0]out;

   reg [3:0]out;

   always@(posedge clk or posedge rst)begin
      if(rst)out<=4'b0;
      else out<=out+4'b1;
   end

endmodule
```

counter_4bit

程序指定 (2/2)

阻礙指定與無阻礙指定之比較

阻礙指定(=)的敘述，先求出等式右邊的結果，然後立即更新等式左邊的值

```
module testbench();
   reg [3:0]a, b;

   initial begin
          a=5;
          b=9;
      #10 a=b;
          b=a;
      #10 $finish;
   end

   initial begin
      $monitor($time, " a=%d, b=%d", a, b);
   end

endmodule
```

testbench

阻礙指定(=)敘述會依照其在程序區塊中的位置依序執行(循序執行)

```
0 a= 5, b= 9
10 a= 9, b= 9
```

無阻礙指定(<=)的敘述，在時間一開始時先求出等式右邊的結果，然後在時間結束時才更新等式左邊的值

```
module testbench();
   reg [3:0]a, b;

   initial begin
          a<=5;
          b<=9;
      #10 a<=b;
          b<=a;
      #10 $finish;
   end

   initial begin
      $monitor($time, " a=%d, b=%d", a, b);
   end

endmodule
```

testbench

無阻礙指定(<=)敘述的執行順序，不會受到敘述位置前後的影響(並行執行)

```
0 a= 5, b= 9
10 a= 9, b= 5
```

Verilog 編碼風格 (1/2)

- Verilog編撰時，經常引起合成前後模擬不一致的情形，且不容易即時發現，因此建議遵守以下準則：

- 1) 在編寫時序控制時
  - 應避免在同一個always區塊中混合使用準位觸發與邊緣觸發敘述。
  - 應避免混合使用正緣觸發與負緣觸發的正反器。

- 2) 在編寫程序指定時
  - 組合電路以阻礙指定(=)來編輯設計。
  - 序向電路以無阻礙指定(<=)來編輯設計。
  - 應避免在同一個always區塊中混合使用阻礙與無阻礙指定敘述。

- 3) 分開撰寫組合電路與序向電路。

Verilog 編碼風格 (2/2)

```
module counter_4bit(clk, rst, out_7seg);
   input clk, rst;
   output [6:0]out_7seg;

   reg [3:0]out;
   reg [6:0]out_7seg;

   always@(posedge clk or posedge rst)begin
      if(rst)out<=4'b0;
      else out<=out+4'b1;

      case(out)
         4'b0000: out_7seg<=7'b0111111;
         4'b0001: out_7seg<=7'b0000110;
         4'b0010: out_7seg<=7'b1011011;
         4'b0011: out_7seg<=7'b1001111;
         4'b0100: out_7seg<=7'b1100110;
         4'b0101: out_7seg<=7'b1101101;
         4'b0110: out_7seg<=7'b1111101;
         4'b0111: out_7seg<=7'b0000111;
         4'b1000: out_7seg<=7'b1111111;
         4'b1001: out_7seg<=7'b1101111;
         4'b1010: out_7seg<=7'b1110111;
         4'b1011: out_7seg<=7'b1111100;
         4'b1100: out_7seg<=7'b0111001;
         4'b1101: out_7seg<=7'b1011110;
         4'b1110: out_7seg<=7'b1111001;
         4'b1111: out_7seg<=7'b1110001;
        default: out_7seg<=7'b0000000;
      endcase
   end
endmodu
```

counter_4bit (含bin_to_7seg)

序向電路以無阻礙指定(<=)來編輯設計

沒有分開撰寫組合電路與序向電路，會多合成出out_7seg這個暫存器，然而這並不是我們所想要的

組合電路以阻礙指定(=)來編輯設計

```
module counter_4bit(clk, rst, out_7seg);
   input clk, rst;
   output [6:0]out_7seg;

   reg [3:0]out;
   reg [6:0]out_7seg;

always@(posedge clk or posedge rst)begin
   if(rst)out<=4'b0;
   else out<=out+4'b1;
end

always@(out)begin
   case(out)
      4'b0000: out_7seg=7'b0111111;
      4'b0001: out_7seg=7'b0000110;
      4'b0010: out_7seg=7'b1011011;
      4'b0011: out_7seg=7'b1001111;
      4'b0100: out_7seg=7'b1100110;
      4'b0101: out_7seg=7'b1101101;
      4'b0110: out_7seg=7'b1111101;
      4'b0111: out_7seg=7'b0000111;
      4'b1000: out_7seg=7'b1111111;
      4'b1001: out_7seg=7'b1101111;
      4'b1010: out_7seg=7'b1110111;
      4'b1011: out_7seg=7'b1111100;
      4'b1100: out_7seg=7'b0111001;
      4'b1101: out_7seg=7'b1011110;
      4'b1110: out_7seg=7'b1111001;
      4'b1111: out_7seg=7'b1110001;
      default: out_7seg=7'b0000000;
   endcase
end
en
```

counter_4bit (含bin_to_7seg)

除頻器 (1/2)

- 由於FPGA 板的時脈頻率太高(48MHz)，可能不符我們的需求，因此需要降低其頻率。
- 所以在程式中加入除頻電路，來降低時脈頻率，並且方便使用者觀察 VeriInstrument 軟體中虛擬元件的輸出變化。

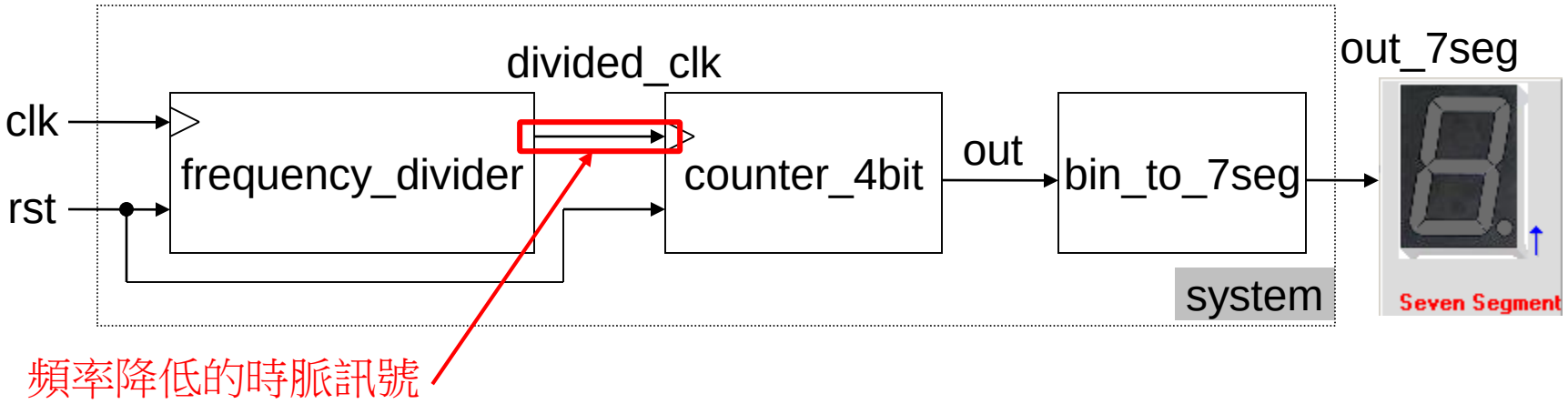
```
module frequency_divider(clk, rst, divided_clk);
   input clk, rst;
   output divided_clk;

   reg divided_clk;
   reg [24:0]count;

   always@(posedge clk or posedge rst)begin
      if(rst)begin
         divided_clk<=1'b0;
         count<=25'b0;
      end
      else if(count==25'd24000000)begin
         divided_clk<=~divided_clk;
         count<=25'b0;
      end
      else begin
         count<=count+25'b1;
      end
   end

endmodule
```

frequency_divider

因為FPGA內部的時脈頻率為48MHz

利用一個計數器來達到時脈除頻的效果，將原本FPGA的時脈頻率降到1Hz

除頻器 (2/2)

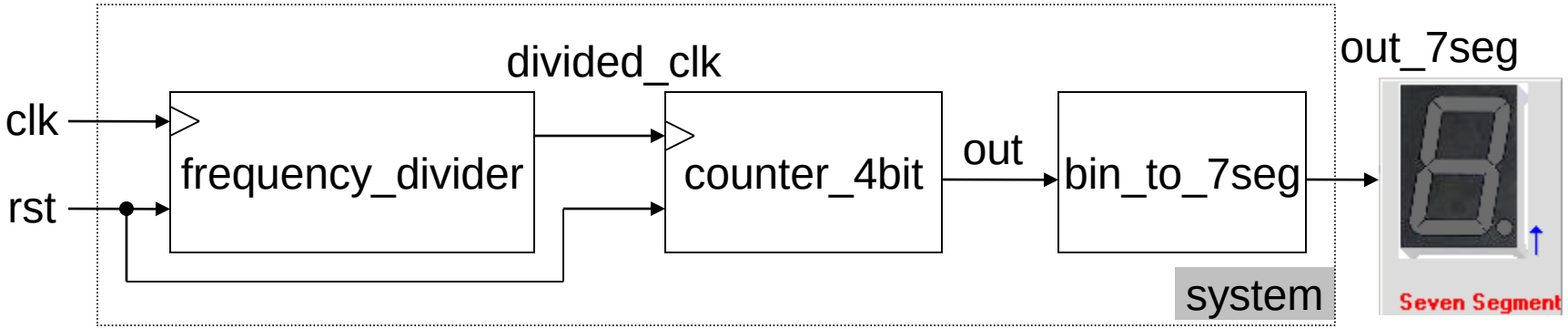



```
module system(clk, rst, out_7seg);
   input clk, rst;
   output [6:0]out_7seg;

   wire divided_clk;
   wire [3:0]out;

   frequency_divider FD(clk, rst, divided_clk);
   counter_4bit C(divided_clk, rst, out);
   bin_to_7seg BT7(out, out_7seg);

endmodule
```

system

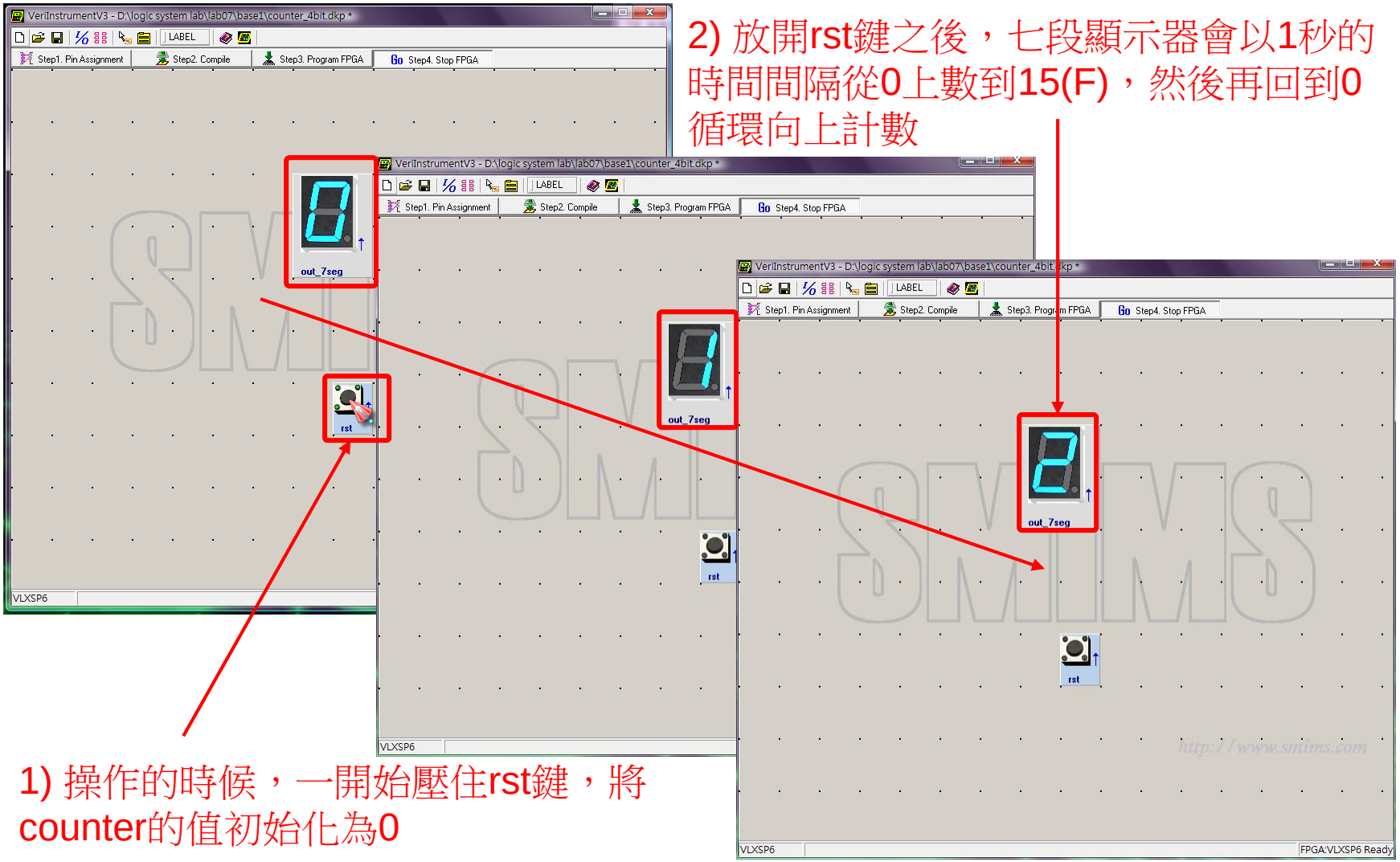
基礎題 (一)
4bit計數器與七段顯示器 (1/2)

- 請參考counter_4bit.v的範例原始碼與操作流程，寫出以下的電路。
  - 設計時，請寫出以下的電路模組，並且確認編譯後沒有error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。)
  - 驗證時，請觀察虛擬儀器之運作是否如期望一般。



基礎題 (一)

4bit計數器與七段顯示器 (2/2)

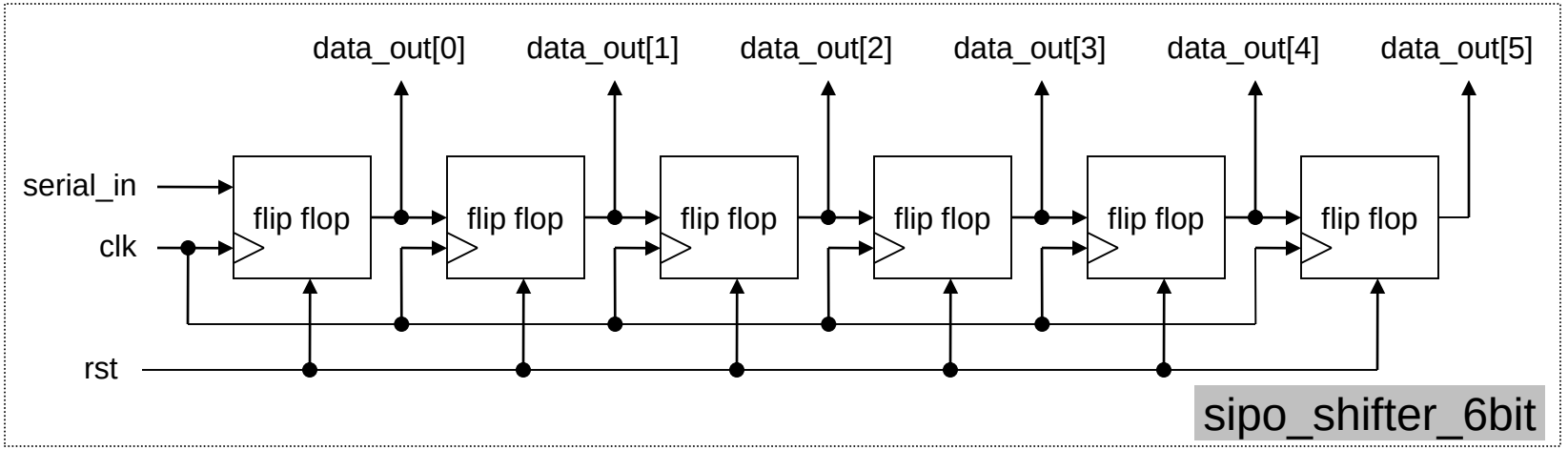


2) 放開rst鍵之後，七段顯示器會以1秒的時間間隔從0上數到15(F)，然後再回到0循環向上計數

1) 操作的時候，一開始壓住rst鍵，將counter的值初始化為0

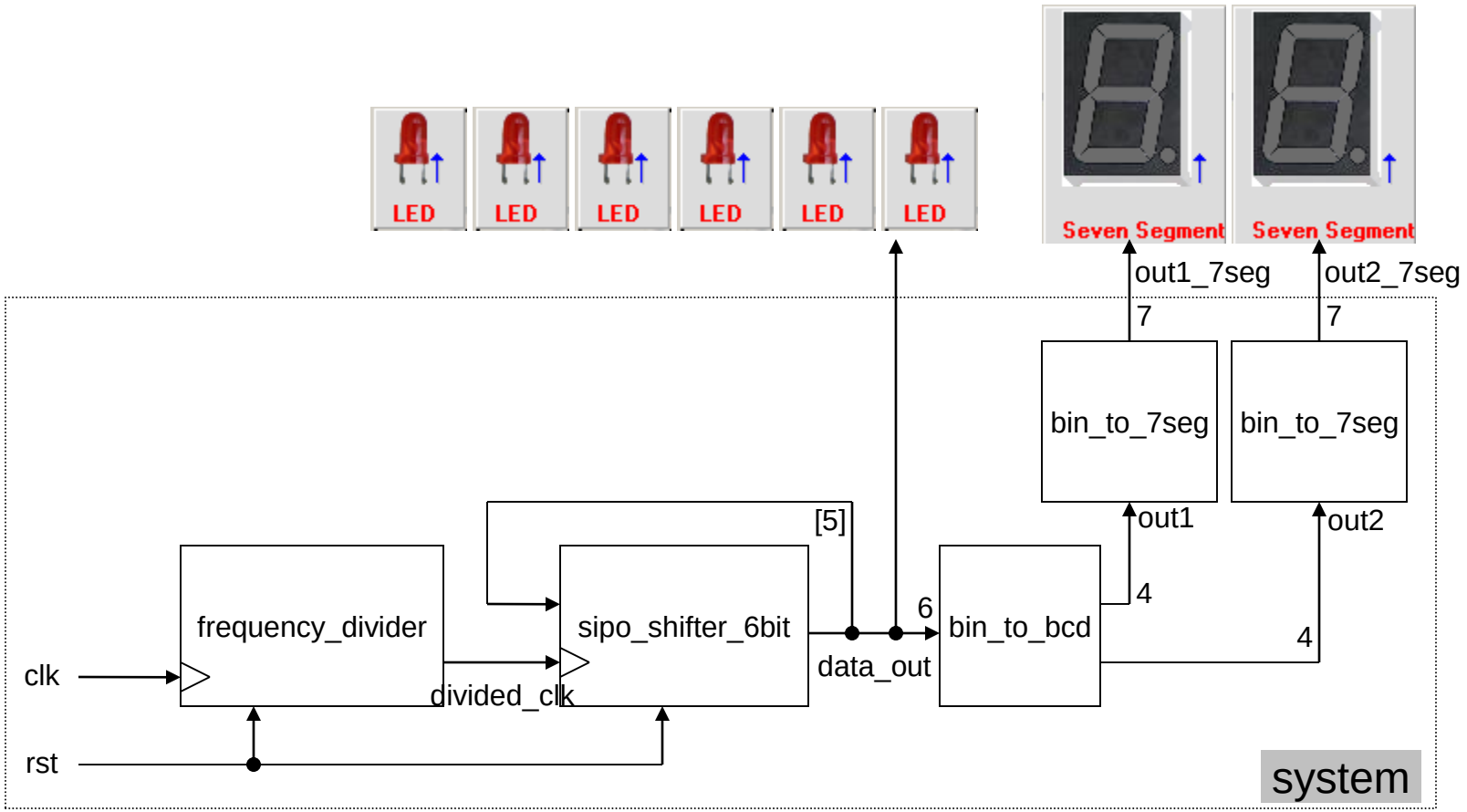
基礎題 (二)
6bit串入並出移位器與七段顯示器 (1/3)

- 請利用下圖的6bit串入並出移位器，撰寫出下一頁的電路圖。
  - 設計時，請寫出下一頁的電路模組，並且確認編譯後沒有error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。)
  - 驗證時，請觀察虛擬儀器之運作是否如期望一般。



基礎題 (二)

6bit串入並出移位器與七段顯示器 (2/3)



基礎題 (二)

6bit串入並出移位器與七段顯示器 (3/3)

挑戰題
計時器 (1/5)

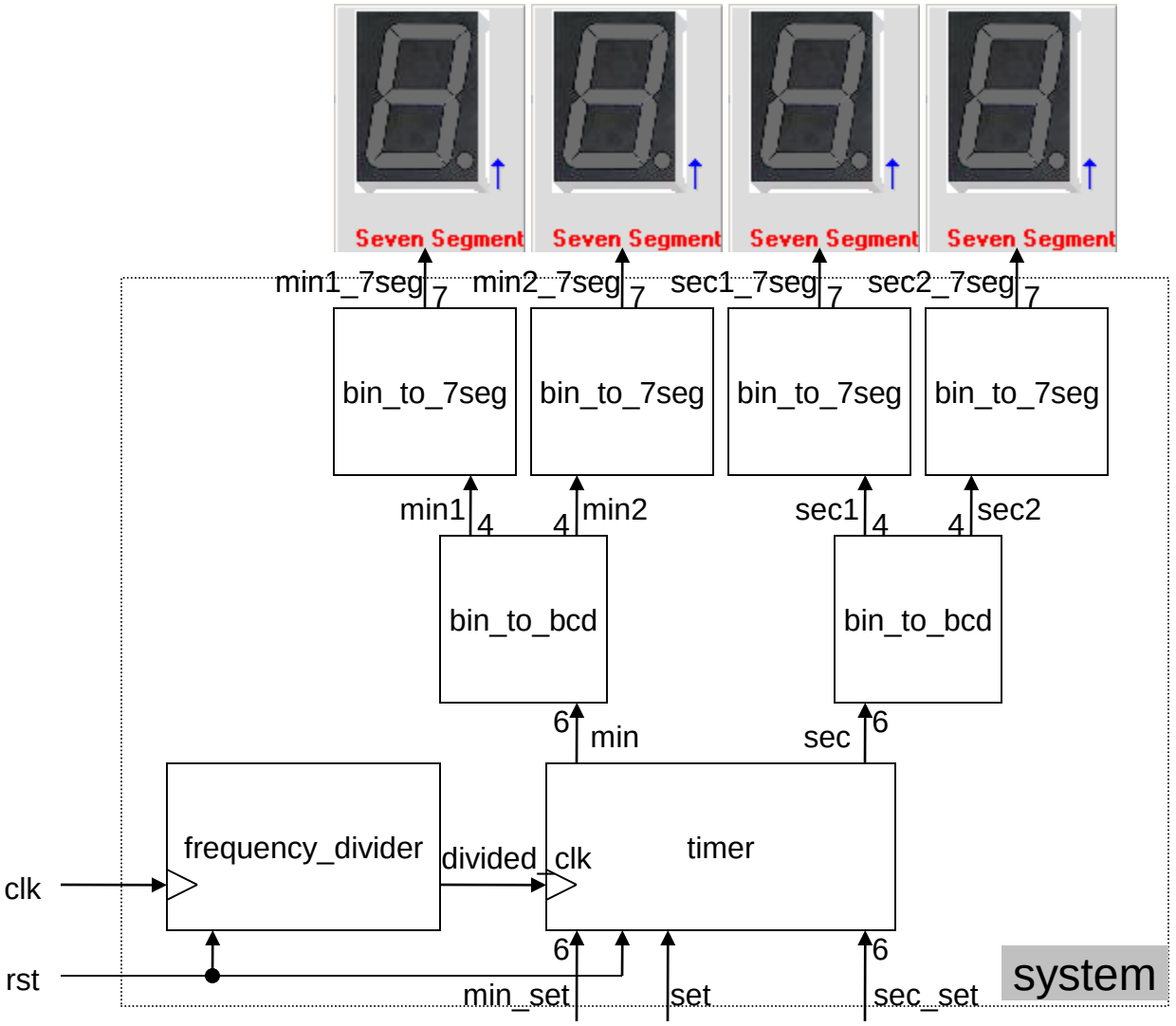
- 請寫出具有下一頁功能的電路。
  - 設計時，請寫出下一頁的電路模組，並且確認編譯後沒有error產生。
  - (模擬時，請自行撰寫testbench.v，並觀察波形結果或主控台的螢幕顯示結果是否如期望一般。)
  - 驗證時，請觀察虛擬儀器之運作是否如期望一般。

挑戰題
計時器 (2/5)

- 輸入
  - clk : 計時器時脈。
  - rst : 當rst為1時，將輸出min與sec歸零。
  - set : 當set為1時，將min設定為set_min、將sec設定為set_sec。
  - set_min : 用來設定輸出min的值。
  - set_sec : 用來設定輸出sec的值。
- 輸出
  - min : 計時器的”分”部分，範圍為0~59，數滿時進位。
  - sec : 計時器的”秒”部分，範圍為0~59，數滿時進位。
- 功能
  - 計數器每一個時脈會向上加一。
  - 當set為1時，將min設定為set_min、將sec設定為set_sec。
  - 當計時器的”秒”部分數滿時(sec=59)，下一時脈會進位(min++, sec=0)。
  - 當計時器的”分”部分數滿時(min=59, sec=59)，下一時脈會進位(min=0, sec=0)。

挑戰題

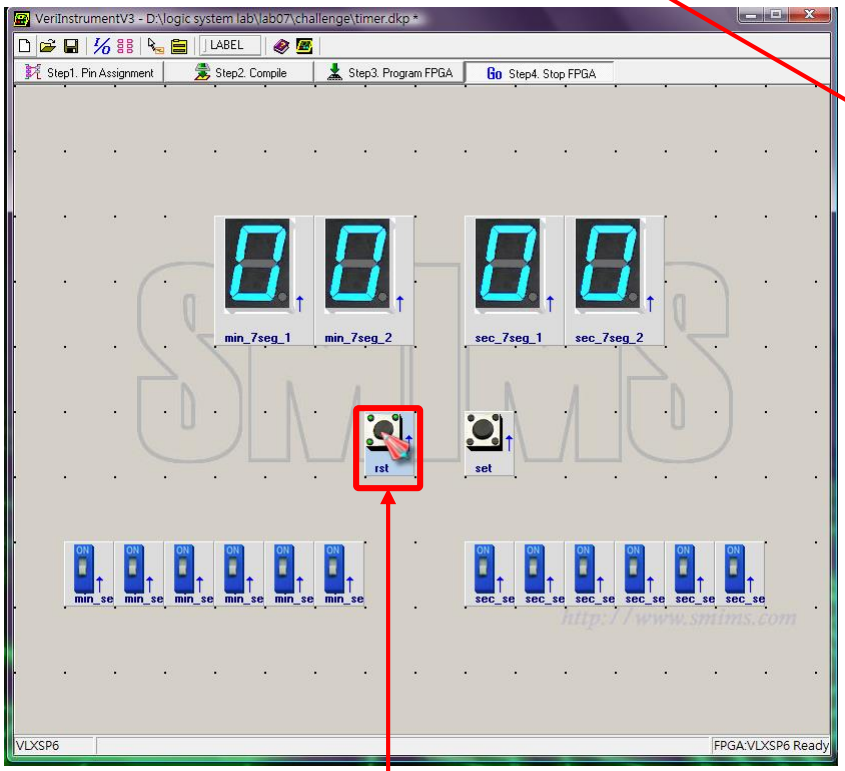
計時器 (3/5)

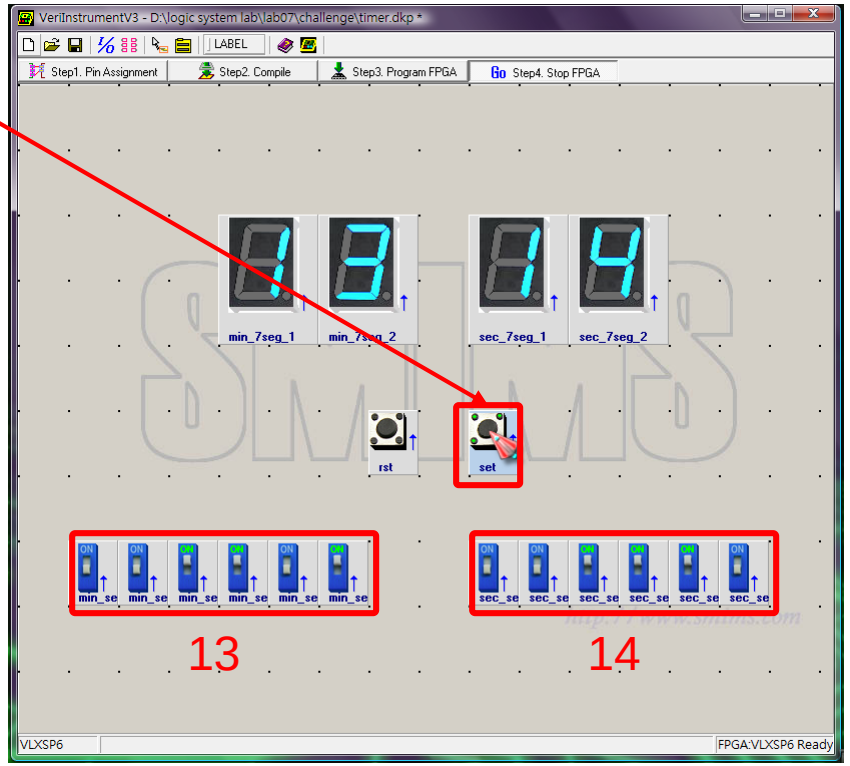

Seven Segment
Seven Segment
Seven Segment
Seven Segment
min1_7seg
7
min2_7seg
7
sec1_7seg
7
sec2_7seg
7
bin_to_7seg
bin_to_7seg
bin_to_7seg
bin_to_7seg
min1
4
4
min2
sec1
4
4
sec2
bin_to_bcd
bin_to_bcd
6
min
sec
6
frequency_divider
divided
clk
timer
clk
rst
6
6
min_set
set
sec_set
system

挑戰題

計時器 (4/5)

2) 若壓住set鍵，可以將指撥開關上的值設定進去min與sec中





1) 操作的時候，一開始壓住rst鍵，將min與sec的值初始化為0

挑戰題

計時器 (5/5)

實驗結報繳交

- 基礎題 (一)
  - 請附上原始碼、(模擬結果擷圖)、驗證結果擷圖與解釋。
- 基礎題 (二)
  - 請附上原始碼、(模擬結果擷圖)、驗證結果擷圖與解釋。
- 挑戰題
  - 請附上原始碼、(模擬結果擷圖)、驗證結果擷圖與解釋。
- 各自之心得報告