邏輯系統實習

Lab9 PYNQ + 七段顯示器

國立成功大學 電機系

2023

大綱

- Verilog 補充 - 條件敘述與多路徑分支
- Verilog 補充 - *Priority mux v.s. Parallel mux*
- Verilog 補充 -運算子的優先順序
- FPGA 與 PYNQ簡介
- PYNQ 七段顯示器
- PYNQ 七段顯示器編譯燒錄進板子
- 防彈跳電路(Debounce circuit)

- 基礎題 *(一)*
  - 4bits輸入七段顯示器
- 基礎題 *(二)*
  - Debounce circuit
- 挑戰題
  - *2bits*ALU

Verilog 補充 -條件敘述與多路徑分支 (1/5)

```
module mux_4to1(in1, in2, in3, in4, sel, out);
   input in1, in2, in3, in4;
   input [1:0]sel;
   output out;

   reg out;

   always@(sel or in1 or in2 or in3 or in4)begin
      if(sel==2'b00)out=in1;
      else if(sel==2'b01)out=in2;
      else if(sel==2'b10)out=in3;
      else out=in4;
   end

endmodule
```

mux_4to1

if-else敘述需考慮priority，即前面的條件不符合才判斷下一個條件，因此會合成出較複雜的電路。

case敘述通常會合成出多工器電路

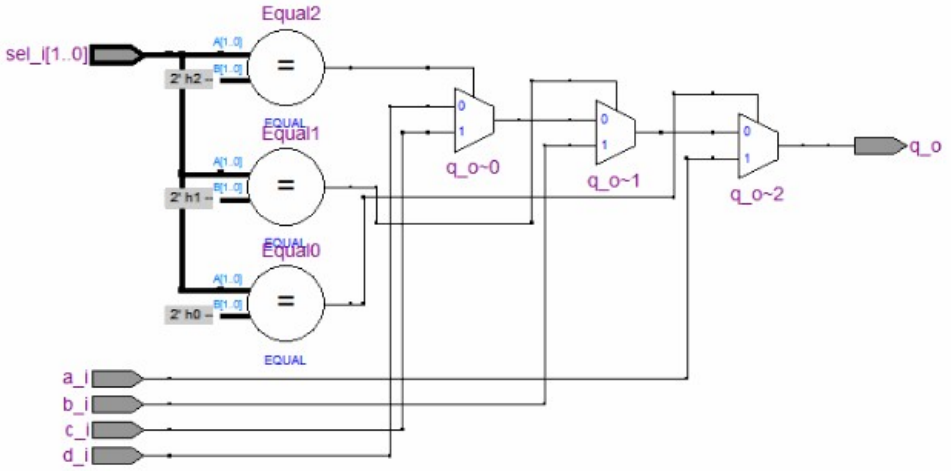
```
module mux_4to1(in1, in2, in3, in4, sel, out);
   input in1, in2, in3, in4;
   input [1:0]sel;
   output out;

   reg out;

   always@(sel or in1 or in2 or in3 or in4)begin
      case(sel)
         2'b00: out=in1;
         2'b01: out=in2;
         2'b10: out=in3;
         2'b11: out=in4;
         default:out=1'b0;
      endcase
   end

endmodule
```

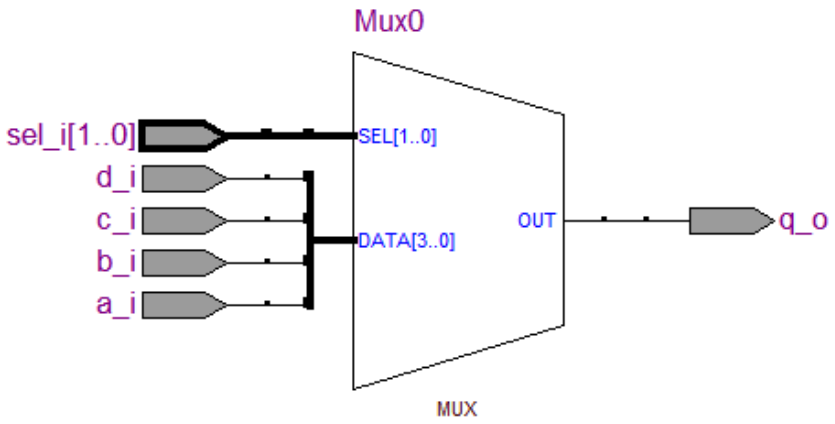
mux_4to1

Verilog 補充 -條件敘述與多路徑分支 (2/5)

```
module mux_4to1(in1, in2, in3, in4, sel, out);
   input in1, in2, in3, in4;
   input [1:0]sel;
   output out;

   reg out;

   always@(sel or in1 or in2 or in3 or in4)begin
      if(sel==2'b00)out=in1;
      else if(sel==2'b01)out=in2;
      else if(sel==2'b10)out=in3;
      else out=in4;
   end

endmodule
```

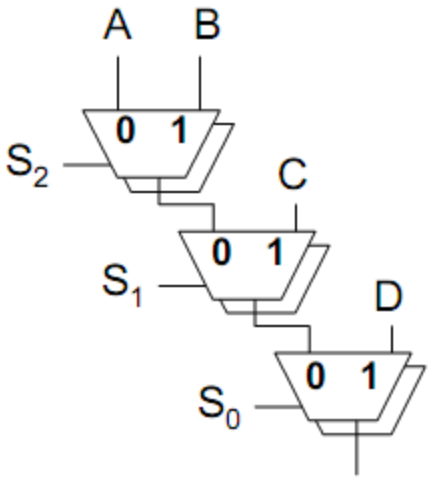
```
module mux_4to1(in1, in2, in3, in4, sel, out);
   input in1, in2, in3, in4;
   input [1:0]sel;
   output out;

   reg out;

   always@(sel or in1 or in2 or in3 or in4)begin
      case(sel)
         2'b00: out=in1;
         2'b01: out=in2;
         2'b10: out=in3;
         2'b11: out=in4;
         default:out=1'b0;
      endcase
   end

endmodule
```



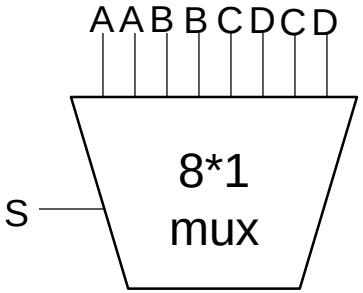


Verilog 補充 - *Priority mux v.s. Parallel mux*

- ***Priority mux***
  - if (S0 == 1'b1)
    - Out=D
  - else if(S1== 1'b1)
    - Out = C;
  - else if(S2 == 1'b1)
    - Out = B;
  - else
    - Out = A;



- ***Parallel mux***
  - Case({S0,S1,S2})
  - 3'b000=A;
  - 3'b001=A;
  - 3'b010=B;
  - 3'b011=B;
  - 3'b100=C
  - 3'b101=D;
  - 3'b110=C;
  - 3'b111=D;
  - endcase



Verilog 補充 -條件敘述與多路徑分支 (1/3)

```
module mux_4to1(in1, in2, in3, in4, sel, out);
   input in1, in2, in3, in4;
   input [1:0]sel;
   output out;

   reg out;

   always@(sel or in1 or in2 or in3 or in4)begin
      if(sel==2'b00)out=in1;
      else if(sel==2'b01)out=in2;
      else if(sel==2'b10)out=in3;
      else out=in4;
   end

endmodule
```

mux_4to1

if-else敘述記得要有個else

case敘述記得要有default，
除非已經列出所有case。

```
module mux_4to1(in1, in2, in3, in4, sel, out);
   input in1, in2, in3, in4;
   input [1:0]sel;
   output out;

   reg out;

   always@(sel or in1 or in2 or in3 or in4)begin
      case(sel)
         2'b00: out=in1;
         2'b01: out=in2;
         2'b10: out=in3;
         2'b11: out=in4;
         default:out=1'b0;
      endcase
   end

endmodule
```

mux_4to1

這兩種敘述都必須要做完整的定義，
才能使合成的工作正常運作

因為如果定義不完整，合成軟體也
許會合成出latch。

Verilog 補充 -條件敘述與多路徑分支 (2/3)

```
module example(in1, in2, sel, out1, out2);
   input in1, in2, sel;
   output out1, out2;

   reg out1, out2;

   always@(*)begin
      if(sel==1'b1)begin
         out1=in1&in2;
         //out2=??
      end
      else begin
         //out1=??
         out2=in1|in2;
      end
   end

endmodule
```

example

WARNING : Found 1-bit latch for signal <out2>

WARNING : Found 1-bit latch for signal <out1>

務必將每一項敘述補齊，例如補上 1'b0或1'bx (don't care)等

通常建議補上1'bx，因為可以讓合成軟體替你做最佳化，而減少電路面積

```
module example(in1, in2, sel, out1, out2);
   input in1, in2, sel;
   output out1, out2;

   reg out1, out2;

   always@(*)begin
      if(sel==1'b1)begin
         out1=in1&in2;
         out2=1'b0;
      end
      else begin
         out1=1'b0;
         out2=in1|in2;
      end
   end

endmodule
```

example

Verilog 補充 -條件敘述與多路徑分支 (3/3)

- 使用條件運算子(?:), if-else和case語法的不同
  - if-else 和 ?: 語法需考慮priority，因此會合成較複雜的電路。
  - case則會合成單純的多工器，但必須確保一次只有一個條件符合。

- ?: 和 if-else的不同
  - ?: 使用在continuous assignment

```
assign a = (b)? c : d;
```

  - if-else 使用在procedural block

```
always@(*) begin
        if(b) a = c;
        else a = d;
end
```

  - 當判斷條件出現unknown時，使用 ?: 語法會直接出現unknown，但if-else則會出現else的結果。

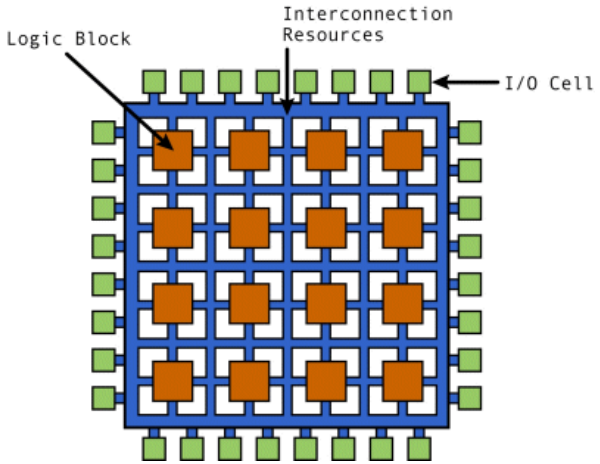
FPGA簡介

- 前幾次Lab中我們所完成的Verilog電路，若要被製作成真的電路，還必須經過幾個步驟。

  1. 邏輯合成(Logic Synthesis)，是將電路轉換為製造商可支援的標準元件(standard cell)型式。
  2. 布局和布線(Place & Route)，將轉換完的元件擺放和排線。

- 現場可程式化邏輯閘陣列 (Field Programmable Gate Array, FPGA)，是由許多的Logic block和接點組成的電路，透過燒錄的方式改變接點的開關，將其轉變為我們設計的電路。

- 過去幾周Lab所使用的Vivado，除了可以模擬電路，最主要的功能其實是將電路合成和PR後，燒錄至FPGA。

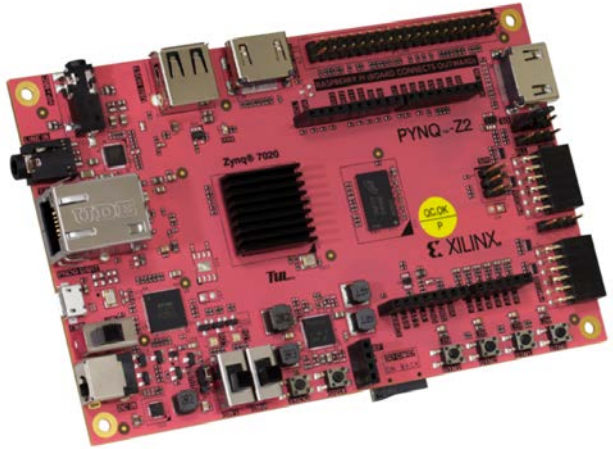


PYNQ簡介

- 本學期的實驗課，我們使用PYNQ-Z2作為主要的開發版。
- PYNQ (Python + Zynq)，是根據Xilinx原有的Zynq FPGA並加上python library, python overlays及jupyter notebook等功能的FPGA。

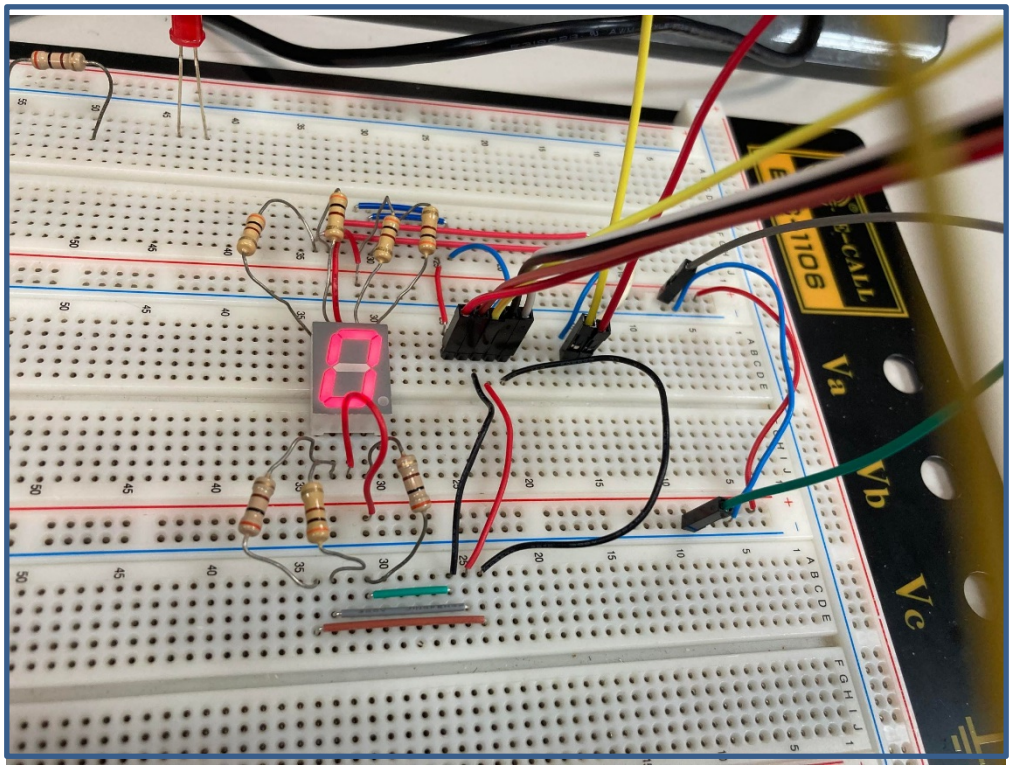
- 除了programmable logic，另外提供dual-core ARM Cortex-A9處理器及其他周邊設備。

- 本次lab，便是要使用on board switch, button輸入給我們設計的電路，並輸出給on board led燈和外接的七段顯示器。



PYNQ 七段顯示器(1/3)

- 先用麵包版接好七段顯示器
  - 先確認好七段顯示每個led(a,b,c,d,e,f,g)都可正常顯示。
  - 再接的時候每個led都要先接好電阻(橙黑棕300Ω)。
  - 七段顯示器是共陽極。



PYNQ 七段顯示器(2/3)

- 打開名稱為lab8_7segment的project
- 點開top.v和seg7.v

```
23  module top(
24      input sysclk,
25      input [1:0] sw,
26      input [3:0] btn,
27      output wire [3:0] led,
28      output wire led4_b,
29      output wire led4_g,
30      output wire led4_r,
31      output wire led5_b,
32      output wire led5_g,
33      output wire led5_r,
34      //gpio
35      output wire [7:0] ar
36      );
37  //wire [2:0] color_led4;  //[r,g,b]
38  //wire [2:0] color_led5;  //[r,g,b]
39  seg7 seg7(.clk(sysclk),.rst(sw[0]),.a(btn[3]),.out(ar[6:0]));
40
41  endmodule
```

top.v

```
22  module seg7(clk,rst,a,out);
23  input clk;
24  input rst ;
25  input a ;
26
27  output reg [6:0] out ;
28  reg [3:0] sel ;
29  wire dout ;
30
31  // debounce debounce_(.din(a),.dout(dout),.clk(clk
32  always@(posedge a or posedge rst)begin
33      if(rst)begin
34          sel <= 4'd0 ;
35      end
36      else begin
37          sel <= sel + 4'd1 ;
38      end
39  end
```

```
41  always@(*)begin
42          case(sel)
43              4'b0000: out = 7'b1000000;
44              4'b0001: out = 7'b1111001;
45              4'b0010: out = 7'b0100100;
46              4'b0011: out = 7'b0110000;
47              4'b0100: out = 7'b0011001;
48              4'b0101: out = 7'b0010010;
49              4'b0110: out = 7'b0000010;
50              4'b0111: out = 7'b1111000;
51              4'b1000: out = 7'b0000000;
52              4'b1001: out = 7'b0010000;
53              4'b1010: out = 7'b0001000;
54              4'b1011: out = 7'b0000011;
55              4'b1100: out = 7'b1000110;
56              4'b1101: out = 7'b0100001;
57              4'b1110: out = 7'b0000110;
58              4'b1111: out = 7'b0001110;
59          endcase
60  end
61  endmodule
```

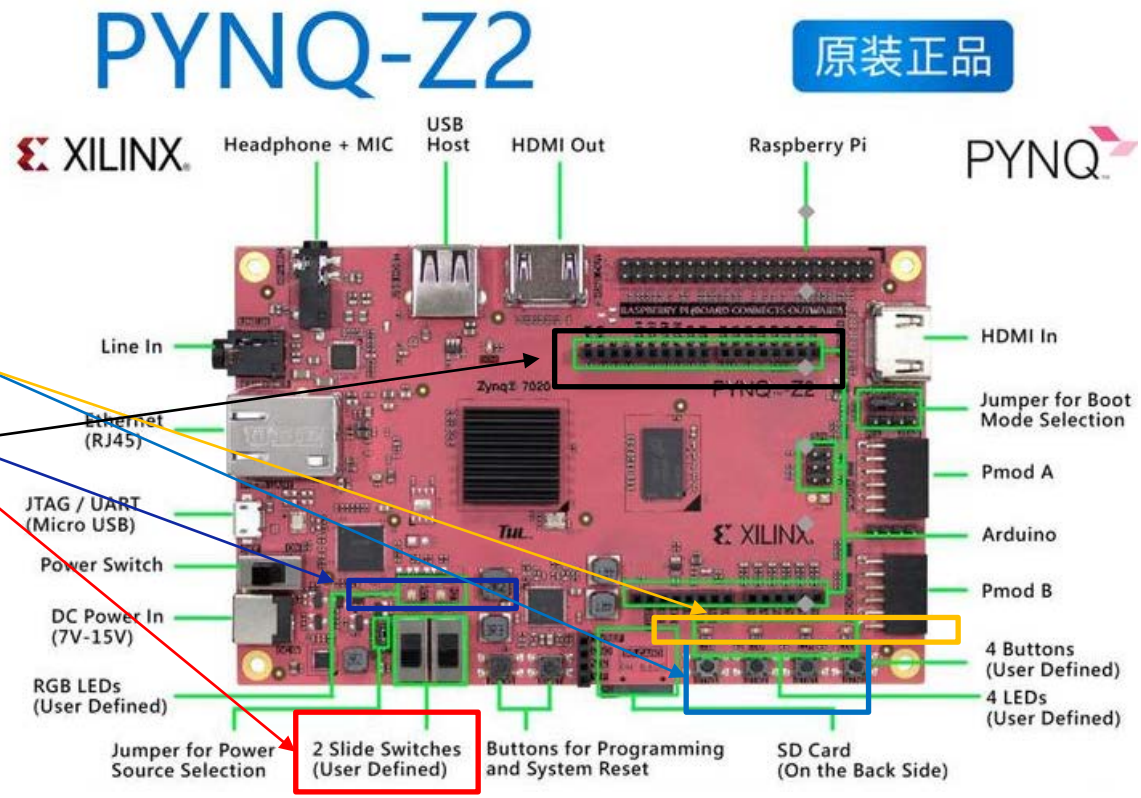
seg7.v

PYNQ 七段顯示器(3/3)

- top.v的腳位對應

```
23  module top(
24      input sysclk,
25      input [1:0] sw,
26      input [3:0] btn,
27      output wire [3:0] led,
28      output wire led4_b,
29      output wire led4_g,
30      output wire led4_r,
31      output wire led5_b,
32      output wire led5_g,
33      output wire led5_r,
34      //gpio
35      output wire [7:0] ar
36      );
37  //wire [2:0] color_led4;  //[r,g,b]
38  //wire [2:0] color_led5;  //[r,g,b]
39  seg7 seg7(.clk(sysclk),.rst(sw[0]),.a(btn[3]),.out(ar[6:0]));
40
41  endmodule
```

top.v

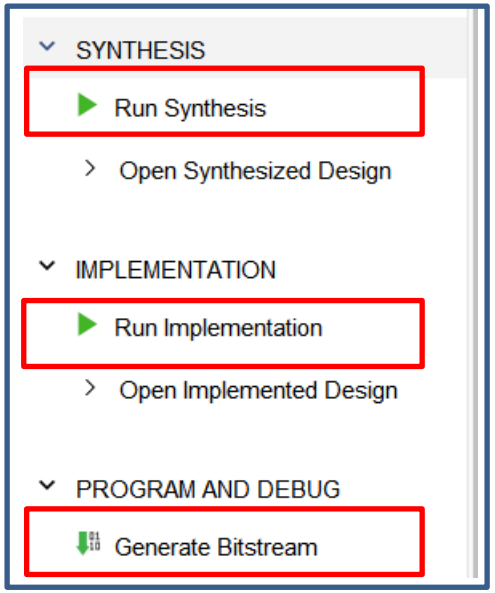


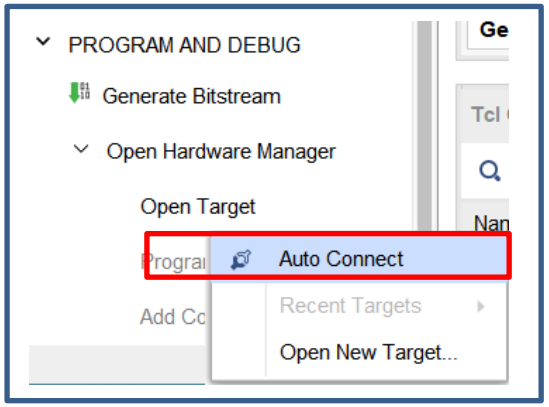
Constraints

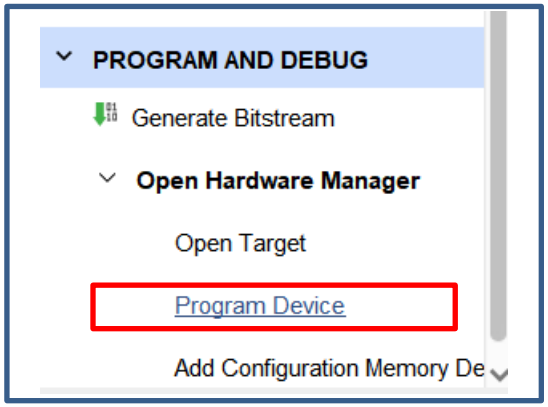
- 當我們需要使用外部的I/O時，我們需要讓Vivado知道電路中哪些訊號對應到板子上的哪些I/O pin。
- 資料夾中的PYNQ-Z2 v1.0.xdc 就是用來讓Vivado得知這些資訊，以及其他燒錄所需要的資訊，這種檔案被稱為constraint file。
- Constraints file助教都已經提供，同學們直接使用即可。
- 本次Lab都已經提供專案檔給同學，同學們可以不用重新新增專案。

PYNQ 七段顯示器編譯燒錄進板子(1/2)

- 點左邊Run Synthesis
- 點左邊Run Implemented
- 點左邊Generate Bitstream
- 點開左邊Open Hardware Manager
  - 點Open Target的Auto connect
- 確定連到板子後點下面的Program Device
- 確認功能

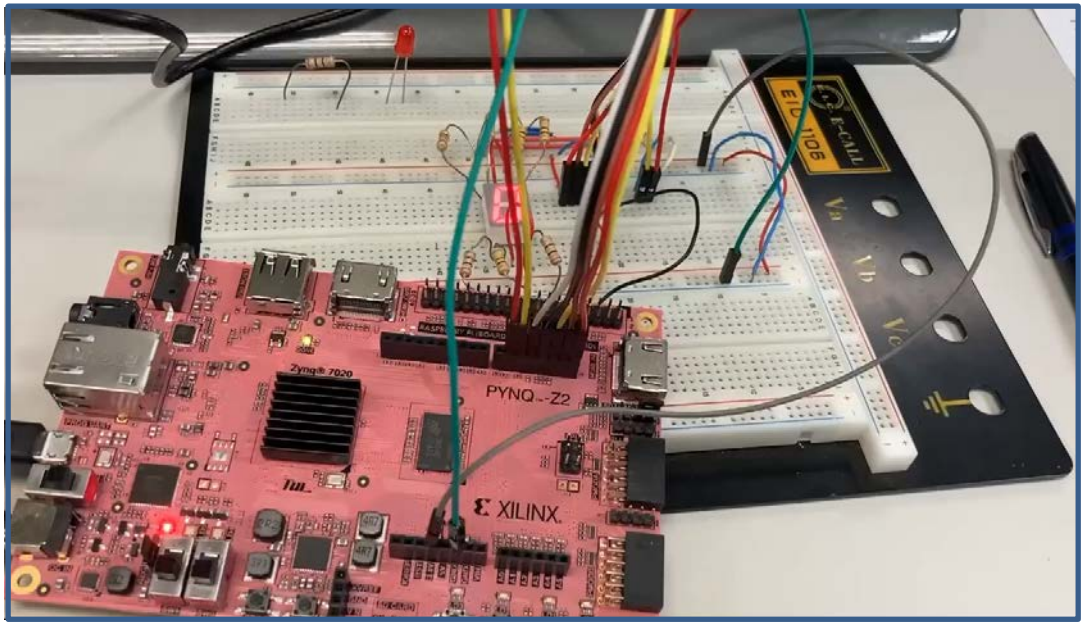




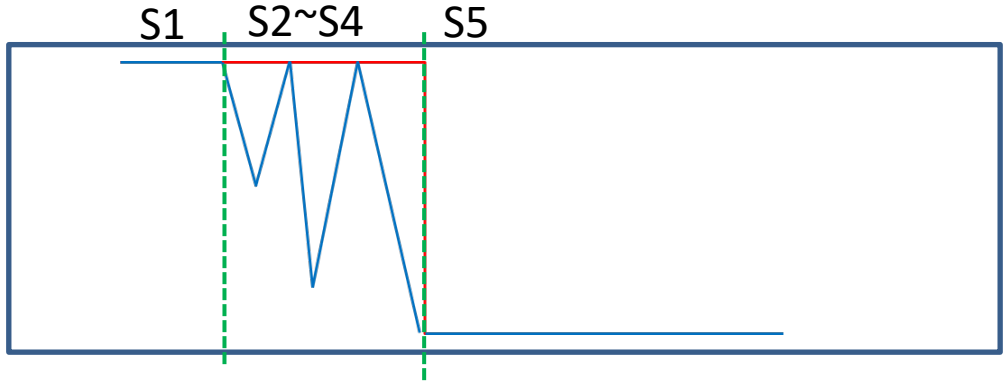


PYNQ 七段顯示器編譯燒錄進板子(2/2)

- 點左邊Run Synthesis
- 點左邊Run Implemented
- 點左邊Generate Bitstream
- 點開左邊Open Hardware Manager
  - 點Open Target的Auto connect
- 確定連到板子後點下面的Program Device
- 確認功能

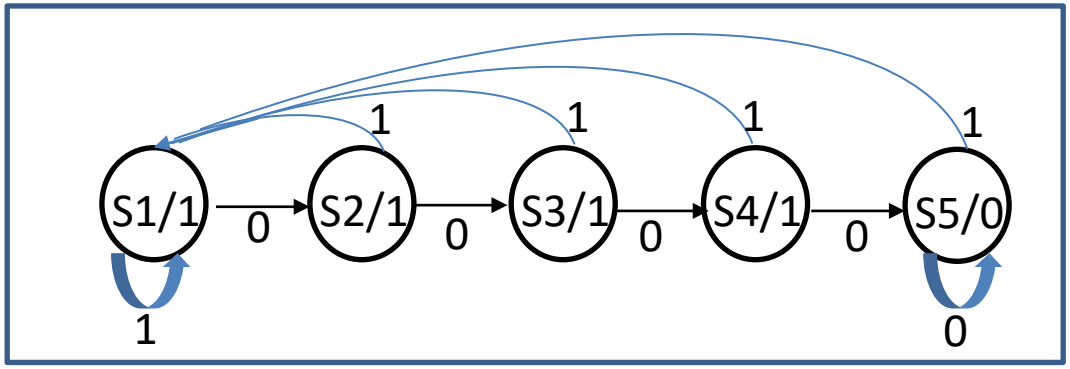


防彈跳電路(Debounce circuit)



波形

- 為moore machine
  - 線上為輸入，state下的值為輸出



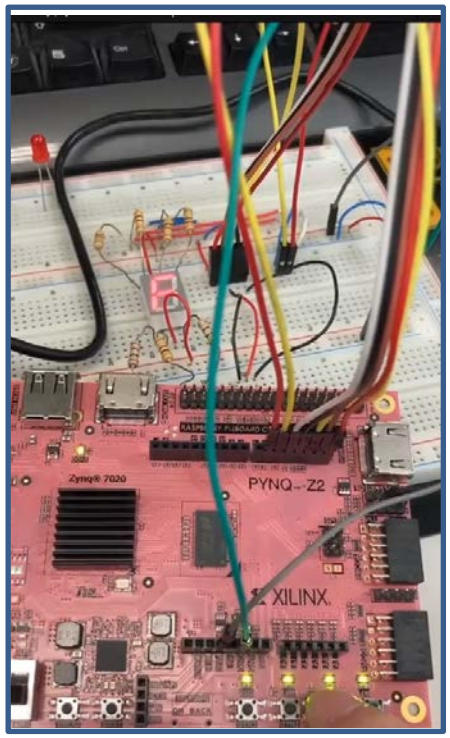
狀態圖

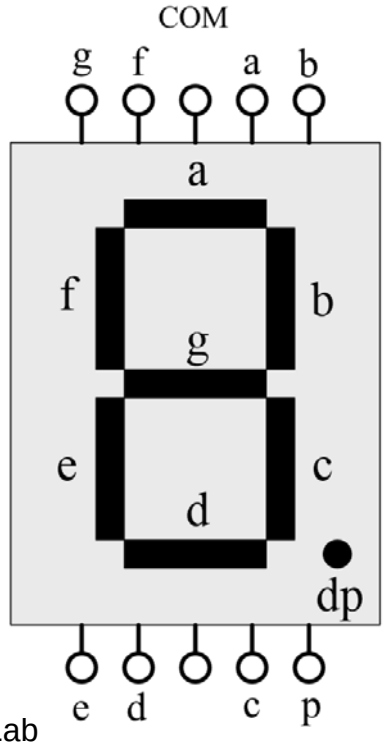
基礎題 (一)

*4bits*輸入七段顯示器

顯示 (0、 1、2、3、4、5、6、7、8、9、A、b、C、d、E、F)。

- BTN[3:0] – 數字輸入， BTN[3] 為最高位元。
- SW[0] – reset，非同步的 reset 訊號，在高準位時會重置目前的數字。
- LED[3:0] – 數字顯示，當相應的 BTN 按下時， LED 的亮暗會被切換。
- ar[6:0] – 七段顯示器輸出，分別對應七端顯示器的 g、f、e、d、c、b、a。





基礎題 (二)
*4bits*輸入七段顯示器

使用 BTN[3:0] 作為輸入，並且幫按鈕加上 debounce circuit 。

挑戰題

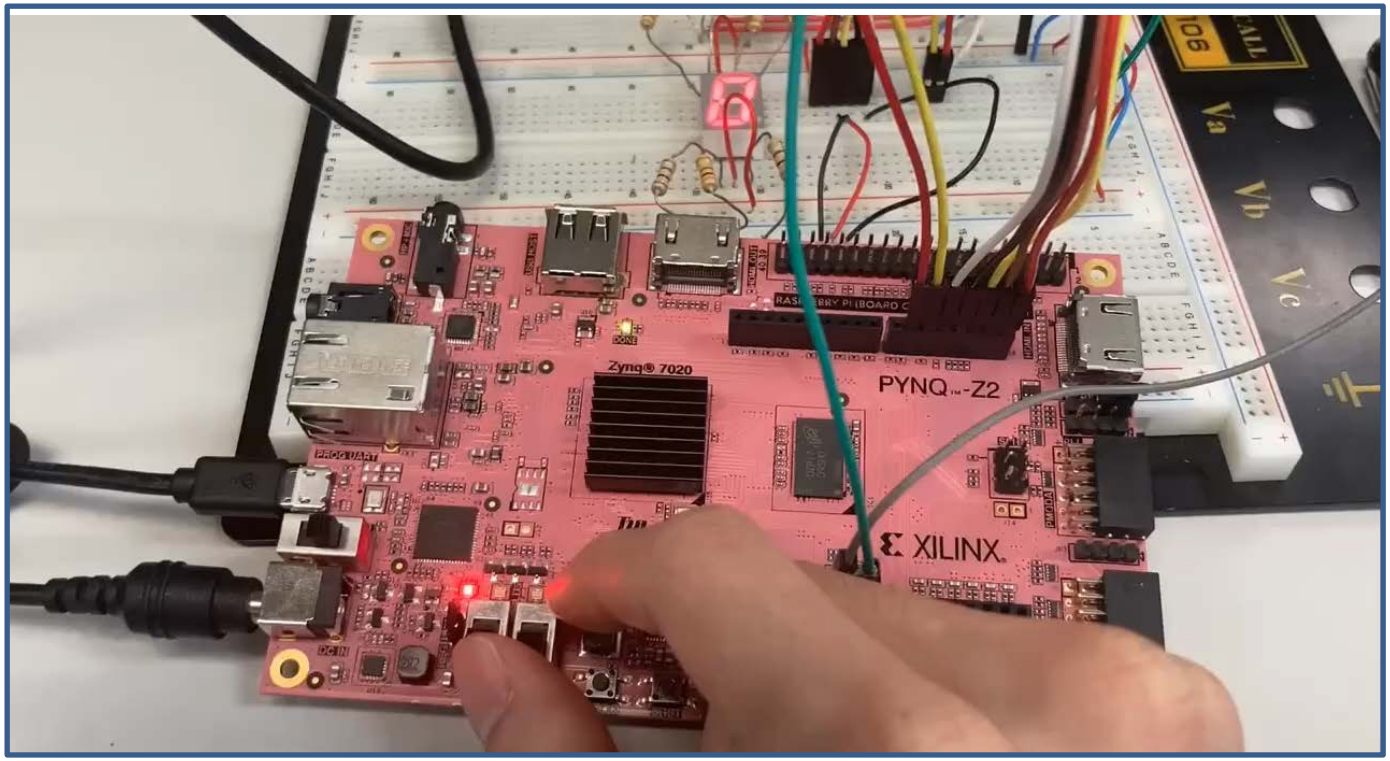
2bitsALU

功能有 add 、 sub 、 and 、 or的ALU。

- SW[1] – mode 輸入：
  - Mode 有四種可能 0(add)、1(sub)、2(and)、3(or)。
  - mode 會需要在 SW[1] 的正緣增加 1 ，當 mode 為 3 時再收到 SW[1] 的正緣訊號則需要讓 mode 回到 0。
- LED5 – mode 顯示：
  - 當 mode 為 0 時 LED5 全滅、為 1 亮紅色 、為 2 亮綠色、為 3 亮藍色。
- BTN[3:0] – 運算元輸入
  - BTN[3:2] 和 BTN[1:0] 分別視為兩個 operand ， BTN[3] 和 BTN[1] 為 MSB 。
- SW[0] – reset
  - 非同步的 reset 訊號，在高準位時會重置目前的數字。
- LED[3:0] – 數字顯示
  - 當相應的 BTN 按下時，LED 的亮暗會被切換。
- ar[6:0] – 運算結果輸出
  - 將運算結果透過七段顯示器輸出，分別對應七端顯示器的 g - a 腳位。

挑戰題

2bitsALU



課堂檢查與結報內容

- 課間檢查
  - ✓ 基礎題 (一) - 程式碼與執行過程。
  - ✓ 基礎題 (二) - 程式碼與執行過程。
  - ✓ 挑戰題 - 程式碼與執行過程。

- 結報繳交
  - 基礎題 (一) - 請附上程式碼、解釋。
  - 基礎題 (二) - 請附上程式碼、解釋。
  - 挑戰題 - 請附上程式碼、解釋。
  - 各自之心得報告

Reference

- https://www.fpga4student.com/2017/04/simple-debouncing-verilog-code-for.html
- https://www.youtube.com/watch?v=6fuHQuqil4&ab_channel=MerakChannel%E5%A4%A9%E7%92%87
- https://forum.digikey.com/t/debounce-logic-circuit-verilog/13196