**Lab 2 110060027朱豐蔚**

1. **exp\_1 (4-bit binary-to-Gray-code converter)**

**Design Specification**

graycoder

a

b

c

d

w

x

y

z

Input: a, b, c, d.

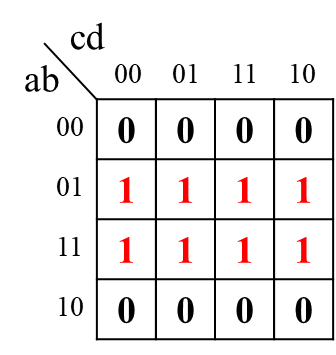
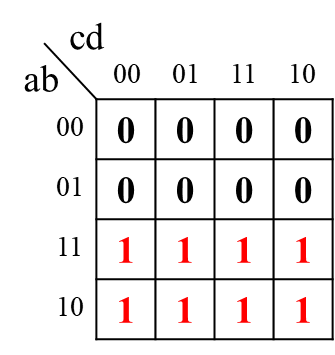
Output: w, x, y, z.

**Design Implementation**

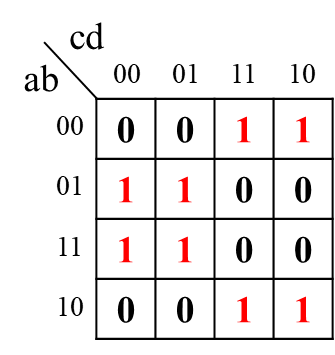
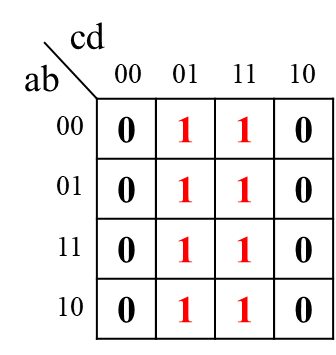
Use truth table and K-map to design the functions:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** |  | **w** | **x** | **y** | **z** |
| 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |  | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |  | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |  | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 |  | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 |  | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |  | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |  | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |  | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |  | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |  | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |  | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |  | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |  | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |  | 1 | 0 | 0 | 0 |

w = a x = ab’+a’b = a⊕b



y = bc’+b’c = b⊕c z = cd’+c’d = c⊕d



Code:

assign w = a;

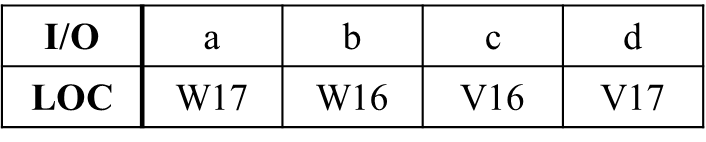
assign x = a ^ b;

assign y = b ^ c;

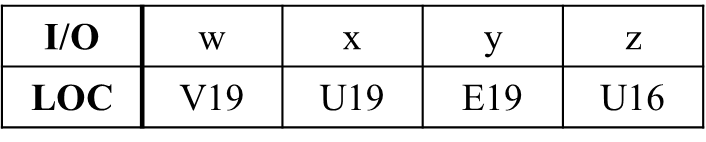
assign z = c ^ d;

Emulation parameters:

Input:



Output:



1. **exp\_2 (7-segment display)**

**Design Specification**

4-to-7-segment- display decoder

i[3:0]

d[3:0]

4

4

8

D[7:0]

Input: i[3:0].

Output: D[7:0], d[3:0].

**Design Implementation**

七段顯示器不適合使用K-map來找Boolean function，由於幾乎沒有規則去簡化，因此在此題使用truth table搭配case的寫法會較為簡單明瞭:

(d[3:0] = i[3:0]，不另外使用case)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **i[3]** | **i[2]** | **i[1]** | **i[0]** |  | **D[7]** | **D[6]** | **D[5]** | **D[4]** | **D[3]** | **D[2]** | **D[1]** | **D[0]** |
| 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |  | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 |  | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |  | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 |  | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |  | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 |  | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 |  | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |  | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |  | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |  | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |  | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |  | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Code:

Reg [7:0] D;

assign d[3:0] = i[3:0];

always @(i)

begin

case (i)

4’d0: D[7:0] = 8’b00000011;

4’d1: D[7:0] = 8’b10011111;

4’d2: D[7:0] = 8’b00100101;

4’d3: D[7:0] = 8’b00001101;

4’d4: D[7:0] = 8’b10011001;

4’d5: D[7:0] = 8’b01001001;

4’d6: D[7:0] = 8’b01000001;

4’d7: D[7:0] = 8’b00011111;

4’d8: D[7:0] = 8’b00000001;

4’d9: D[7:0] = 8’b00011001;

4’d10: D[7:0] = 8’b00010001;

4’d11: D[7:0] = 8’b11000001;

4’d12: D[7:0] = 8’b01100011;

4’d13: D[7:0] = 8’b10000101;

4’d14: D[7:0] = 8’b01100001;

4’d15: D[7:0] = 8’b01110001;

default: D[7:0] = 8’b11111110; // .

endcase

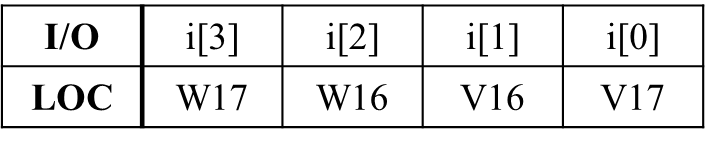
end

Testbench:

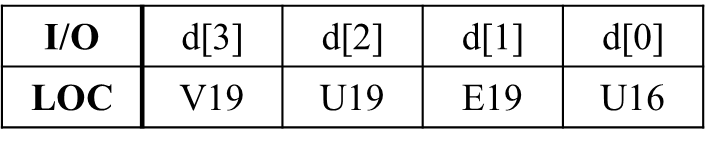
從i[3:0] = 4’d0開始，每過10單位時間(#10)就讓i加1，執行到i[3:0] = 4’d15為止，觀察d[3:0]與D[7:0]是否與預期相符合。

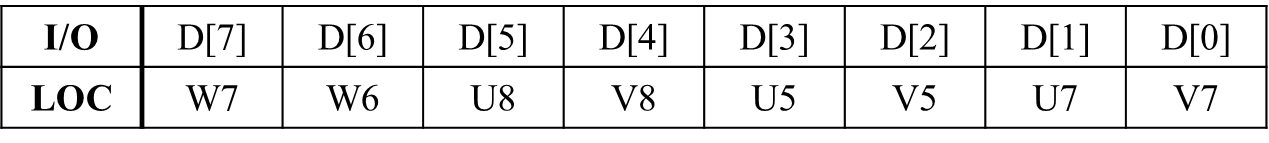
Emulation parameters:

Input:



Output:



****

1. **exp\_3 (Bulls and Cows)**

**Design Specification**

****

Input: a[7:0], b[7:0].

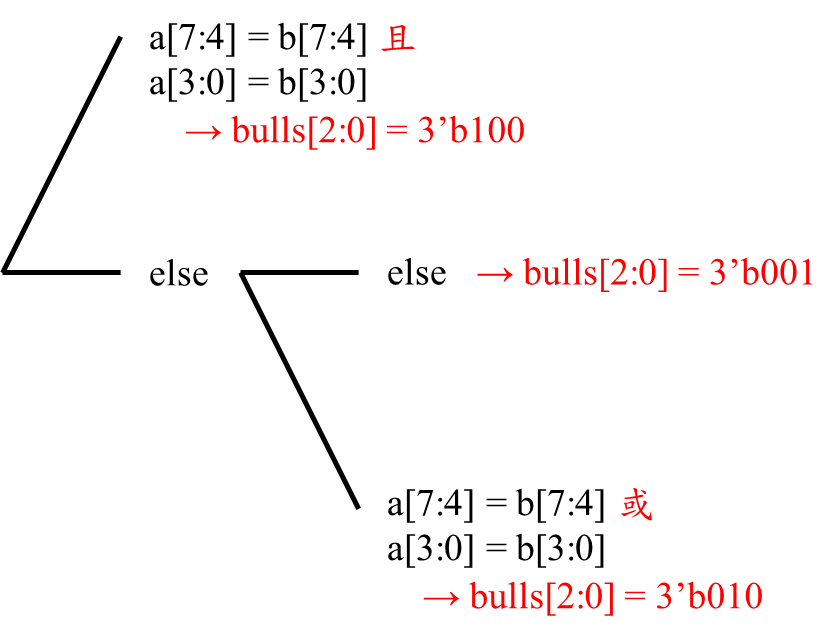
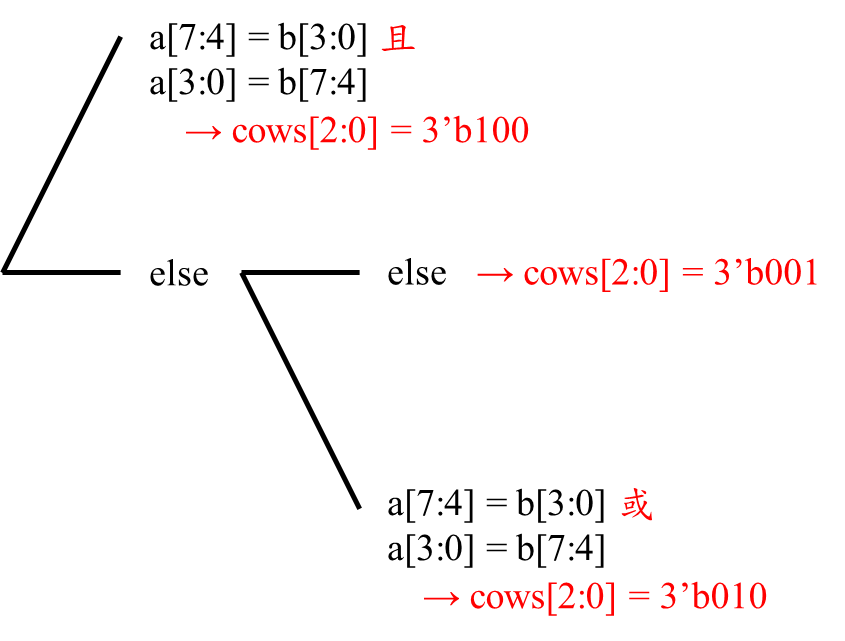
Output: bulls[2:0], cows[2:0]**.**

**Design Implementation**

為使用if來做撰寫，先畫出樹狀圖整理bulls, cows和a, b之間的關係:

(因為或的判斷在第二層if，所以可以避掉重複and的問題)

bulls: cows:



Code:

reg [2:0] bulls, cows;

always @(a or b)

begin

if ((a[7:4] == b[7:4]) && (a[3:0] == b[3:0]))

bulls[2:0] = 3’b100;

else if ((a[7:4] == b[7:4]) || (a[3:0] == b[3:0]))

bulls[2:0] = 3’b010;

else

bulls[2:0] = 3’b001;

if ((a[7:4] == b[3:0]) && (a[3:0] == b[7:4]))

cows[2:0] = 3’b100;

else if ((a[7:4] == b[3:0]) || (a[3:0] == b[7:4]))

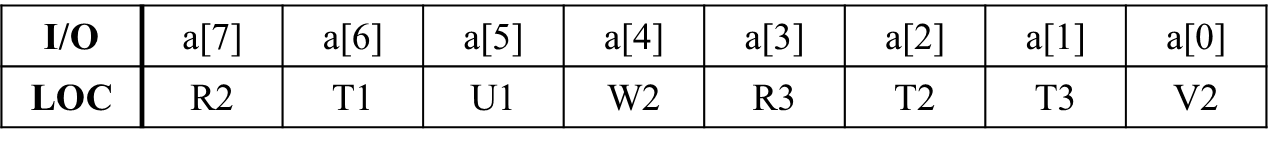
cows[2:0] = 3’b010;

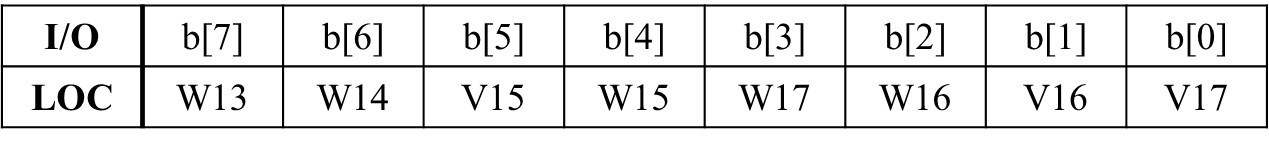
else

cows[2:0] = 3’b001;  
 end

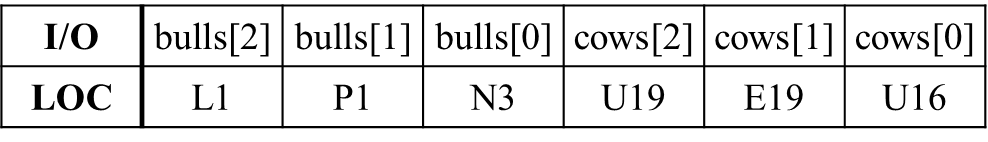
Emulation parameters:

Input:





Output:



**======================================================================**

**Discussion**

第一題如上週一樣，加上題目也有指定parameter，於是在撰寫上我沒有遇到問題。

而在第二題的撰寫中，由於題目沒有說七段顯示器的parameter，因此我在撰寫時透過查詢老師第二週的上課簡報與Basys3 Reference Manual中的接口位置，才順利的將七段顯示器完成。

在第三題的bulls與cows燈號選擇中，原先我是選擇鄰近的6個燈位，但後來我將bulls選為最左方的三個燈位，cows選為最右方的三個燈位，以方便讀取時不會將兩種不同的燈號混在一起讀取，這是我在第三題加入的小巧思。

**Conclusion**

這次的Lab對我來說很新鮮，由於是第一次碰到FPGA板，於是很期待能夠透過Vivado來讓板子實行我們想要的功能與解題，而最後當親眼看見FPGA上的七段顯示器真的如預期般實行，那樣的成就感是很令人開心的。

而另一方面，我也認知到了Basys3 Reference Manual對於我們的重要性，在未來的Lab、抑或是project的撰寫，我們一定會需要大量查詢Basys3 Reference Manual中各個接口的名稱、規定等，於是很感謝教授在第二週就安排如第二題的題目，讓我們能提前熟悉Basys3 Reference Manual的查詢。

**References**

7段顯示器的輸入接口查詢:(from 第二週上課講義)

Bulls and Cows:(from 邏輯設計期中)