數位邏輯設計-HW5

鄭凱文 104062223

HW5-1:

Decoder: input 多少,output 的 1 就往左移多少 bit

6-64→3-8:

首先把 6-bit 的 input 平分成兩段,分別透過 decoder 0 和 decoder 1 得到兩個 8-bit 的 one-hot,X & Y。對最後的 output 來說:

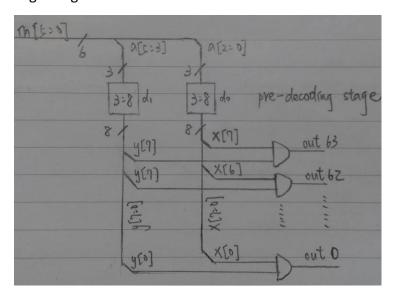
X[j] = j;

Y[i] = 8 * i;

Out[i*8+j] = Y[i] & X[j] , if input = (i*8 + j) , output[i*8+j] = 1

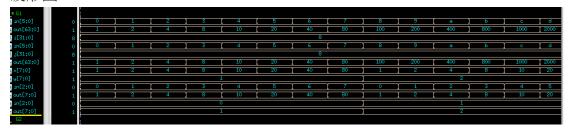
Testbench 選擇:從 0~63 全部跑一遍

Logic diagram:



Simulation:

波形圖:



HW5-2:

Binary Priority Encoder:

為了 rotate 回來方便,取 in_d[15:0] & pri_d[3:0]:

in_d = {in, in} ←延伸為兩倍的 input,如此一來操作時不用 rotate 回 in[0] pri d = {1'b0, pri} ←變成 3-bit,方便之後作加法進位

利用 for 迴圈,從 pri + 8(即在延伸的 in_d 的原本 pri 位置)開始,依序向右找第一個出現的 1(i=i-1),若有找到則 zero = 0,剩餘的 for 迴圈將自動跳出

for(i=8;i>=1&&zero!=0;i=i-1) //start from pri, right forward, if zero==0 then break.

out = pri + i;

zero = 1'b0;

output 即為從 pri 開始,往右找到的第一個 1 的位置。

Testbench 選擇:

in = 8'b00000000, pri = 4: 測試input==0時, output = 0, zero = 1

in = 8'b00000001, pri = 5: 測試input==1時, output = 0, 但zero = 0(代表其不為0)

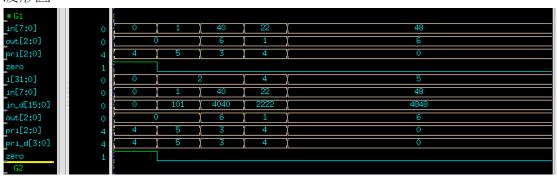
in = 8'b01000000, pri = 3: 測試會不會正確的rotate回來

其他兩個測資為input中有多個1的情況。

Simulation:

```
in = 00000000, pri = 4 | out = 0, zero = 1
in = 00000001, pri = 5 | out = 0, zero = 0
in = 01000000, pri = 3 | out = 6, zero = 0
in = 00100010, pri = 4 | out = 1, zero = 0
in = 01001000, pri = 0 | out = 6, zero = 0
```

波形圖:



HW5-3:

16-bit 的 input, 拆成兩部分放入 encoder:

使用兩個 encoder,input[7:0]放入 enc0,input[15:8]放入 enc1,而傳入的 pri 必 須是 3-bit。

pri 的選擇:如果 pri<=7,代表從 input[7:0]開始找起,再"rotate"回 input[15:8]内找,因此 enc1 的 pri 應一律設為"7"(等於從 input[15]開始找);反之若 pri>7,代表 enc0 的 pri 也應設為"7"(從 input[7]開始找)。

wire tmp_pri = 7 – pri[2:0] \leftarrow 取 7 和 pri%7 的相差值,方便待會補成 7 另外 zero_1, zero_2 用來判斷各自部分是否皆為零,zero = zero_1 & zero_2。因此:

hw5_2 enc0(in[7:0], tmp_pri[2:0]*pri[3]+pri[2:0], out_1, zero_1) ←若 pri[3]=1, 代表 pri>7,enc0 應從 in[7]找起(tmp_pri*1+pri[2:0] = 7)

hw5_2 enc1(in[15:8], tmp_pri[2:0]*~pri[3]+pri[2:0], out_2, zero_2) ←若 pri[3]=0, 代表 pri<=7,enc1 應從 in[15]找起(tmp_pri * ~0 + pri[2:0] = 7)

再來是 output 的判斷:

如果 zero==1,代表 input 為 0。如果 zero!=1,則有 pri<=7 or >7 兩種可能。若 If pri<=7,則優先從 out 1考慮:

- (1) out_1<=pri && zero_1!=1: 代表 out_1 在 pri 的"右邊", output 即為out_1(zero_1!=1 表示排除 input[7:0]==0 的可能)
- (2) out_1>pri && zero_2==0: 雖然 out_1 在 pri 的"左邊",代表有 rotate 過,但由於 zero_2==0 → input[15:8]==0,因此 output 仍為 out_1
- (3) out_1>pri && zero_2!=0: out_1 在 pri 的"左邊",且 input[15:8]!=0,代表有 rotate 過且在 input[15:8]內已經碰到 1,因此 output=out_2 + 8(out_2 原為對 8 取的餘數,輸出時應補回 8)

反之亦然, if pri>7, 只有在 out 2>pri && zero 1!=0 時才考慮 out 1。

Testbench 選擇:

- (1) Input=0, 16-bit 全部的 one-hot, pri=10: 全部跑一遍確認程式無誤,以及 input=0 時 zero 是否為 1。
- (2) Input=16'b0000000000100001, pri = 10; in = 16'b0010000100000000, pri = 3: 確認如果碰到上述狀況(3)的時候,1 集中在其中一側,pri 在另一側時,是否可以得出正確結果。
- (3) 其他為 pri 介於 input 内 1 的中間,或是 input 中有多個 1 的情況。

Simulation:

```
in = 00000000000000000, pri
in = 0000000000000001, pri = 10
                                      out =
                                              \theta, zero = \theta
in = 0000000000000010, pri = 10
                                      out =
                                              1, zero = \theta
in = 0000000000000100, pri = 10
                                      out =
                                              2, zero = 0
in = 0000000000001000, pri = 10
                                      out =
                                              3, zero = \theta
in = 0000000000010000, pri = 10
                                      out =
                                             4, zero = 0
in = 0000000000100000, pri = 10
                                      out =
                                              5, zero = \theta
in = 0000000001000000, pri = 10
in = 0000000010000000, pri = 10
in = 0000000100000000, pri = 10
                                      out =
                                              6, zero = \theta
                                      out =
                                              7, zero =
                                      out =
                                              8, zero =
                                                         Θ
                                      out = 9, zero =
in = 0000001000000000, pri =
                                10
                                                         Θ
                                      out = 10, zero = 0
in = 0000010000000000, pri =
                                10
in = 0000100000000000, pri = 10
                                      out = 11, zero = \theta
in = 0001000000000000, pri = 10
                                      out = 12, zero = 0
                                      out = 13, zero = \theta
in = 0010000000000000, pri = 10
in = 0100000000000000, pri = 10
                                      out = 14, zero = 0
                                      out = 15, zero = 0
in = 1000000000000000, pri = 10
in = 00000000000100001, pri = 10
                                      out = 5, zero = 0
in = 0010000100000000, pri =
                                      out = 13, zero =
                                 3
in = 0010000000000001, pri =
                                      out =
                                             0, zero =
                                                         Θ
                                      out =
in = 0000001000000001, pri = 13
                                              9, zero =
                                                         Θ
                                              5,
in = 0100000100101000, pri
                                 7
                                      out
                                                 zero =
                                          Θ
```

波形圖:

