



[illegible]

### Binary Priority Encoder:

`in_d = {in, in}` ← 延伸為兩倍的 input，如此一來操作時不用 rotate 回 `in[0]`

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

```
if(in_d[pri_d+i] == 1'b1)
```

```
zero = 1'b0;
```

Testbench 選擇：

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

其他兩個測資為input中有多個1的情況。

```
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
```

Variable	0	1	40	22	48
in[7:0]	0	1	40	22	48
out[2:0]	0	0	6	1	6
pri[2:0]	4	4	5	3	4
zero	1				0
i[31:0]	0	0	2	4	5
in[7:0]	0	0	1	40	22
in_d[15:0]	0	0	101	4040	2222
out[2:0]	0	0	6	1	6
pri[2:0]	4	4	5	3	4
pri_d[3:0]	4	4	5	3	4
zero	1				0

### 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  $pri \leq 7$ ，代表從 input[7:0]開始找起，再"rotate"回 input[15:8]內找，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] ←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  $pri \leq 7$ ，enc1 應從 in[15]找起( $tmp\_pri * \sim 0 + pri[2:0] = 7$ )

再來是 output 的判斷：

如果 zero==1，代表 input 為 0。如果 zero!=1，則有  $pri \leq 7$  or  $> 7$  兩種可能。

若  $pri \leq 7$ ，則優先從 out\_1 考慮：

(1)  $out\_1 \leq pri$  &&  $zero\_1 \neq 1$ : 代表 out\_1 在 pri 的"右邊"，output 即為

out\_1( $zero\_1 \neq 1$  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  $zero\_2 == 0 \rightarrow input[15:8] == 0$ ，因此 output 仍為 out\_1

(3)  $out\_1 > pri$  &&  $zero\_2 \neq 0$ : 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  $out\_2 > pri$  &&  $zero\_1 \neq 0$  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 內 1 的中間，或是 input 中有多個 1 的情況。

```

in = 000000000000000000, pri = 10 | out = 0, zero = 1
in = 000000000000000001, pri = 10 | out = 0, zero = 0
in = 000000000000000010, pri = 10 | out = 1, zero = 0
in = 000000000000000100, pri = 10 | out = 2, zero = 0
in = 000000000000001000, pri = 10 | out = 3, zero = 0
in = 000000000000010000, pri = 10 | out = 4, zero = 0
in = 000000000000100000, pri = 10 | out = 5, zero = 0
in = 000000000010000000, pri = 10 | out = 6, zero = 0
in = 000000000100000000, pri = 10 | out = 7, zero = 0
in = 000000001000000000, pri = 10 | out = 8, zero = 0
in = 000000010000000000, pri = 10 | out = 9, zero = 0
in = 000000100000000000, pri = 10 | out = 10, zero = 0
in = 000001000000000000, pri = 10 | out = 11, zero = 0
in = 000010000000000000, pri = 10 | out = 12, zero = 0
in = 000100000000000000, pri = 10 | out = 13, zero = 0
in = 001000000000000000, pri = 10 | out = 14, zero = 0
in = 010000000000000000, pri = 10 | out = 15, zero = 0
in = 100000000000000000, pri = 10 | out = 5, zero = 0
in = 000000000001000001, pri = 10 | out = 13, zero = 0
in = 001000001000000000, pri = 3 | out = 0, zero = 0
in = 001000000000000001, pri = 7 | out = 9, zero = 0
in = 000000010000000001, pri = 13 | out = 5, zero = 0
in = 0100000100101000, pri = 7 | out = 5, zero = 0

```

[illegible]