正確學會 改訂新版 ディジタル回路と Verilog HDL

# Verilog 的16堂課

## 第 5 章

# 組合電路 II

# 本章重點

邏輯電路，可分為組合電路以及序向電路。第 4、5 章會把具有代表性的組合電路逐一做個說明。

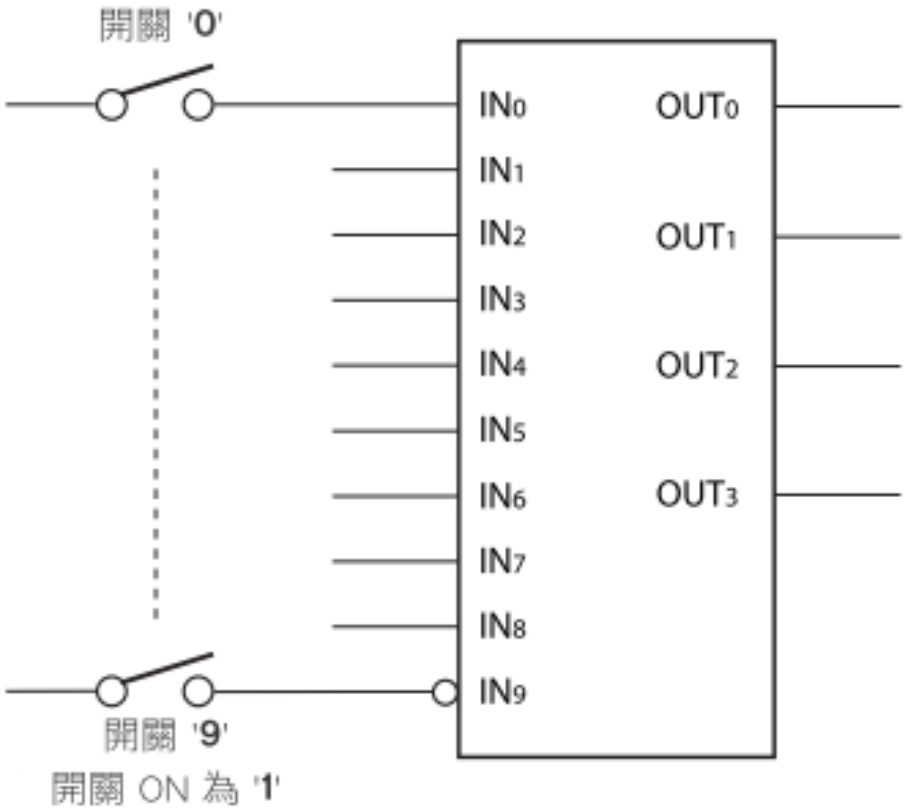
- 5.1 編碼器
- 5.2 解碼器
- 5.3 7 段顯示器的解碼器設計
- 5.4 同位電路
- 5.5 正邏輯與負邏輯

# 5.1 編碼器

- 編碼器 (編碼電路: encoder), 是根據輸入來做出相對應輸出的電路
- 解碼器 (解碼電路: decoder) 則是把輸入的信號還原為原始信號的電路

# 5.1.1 十進位對 BCD 編碼器

- 輸入為 10 進制, 輸出為 BCD 的編碼器

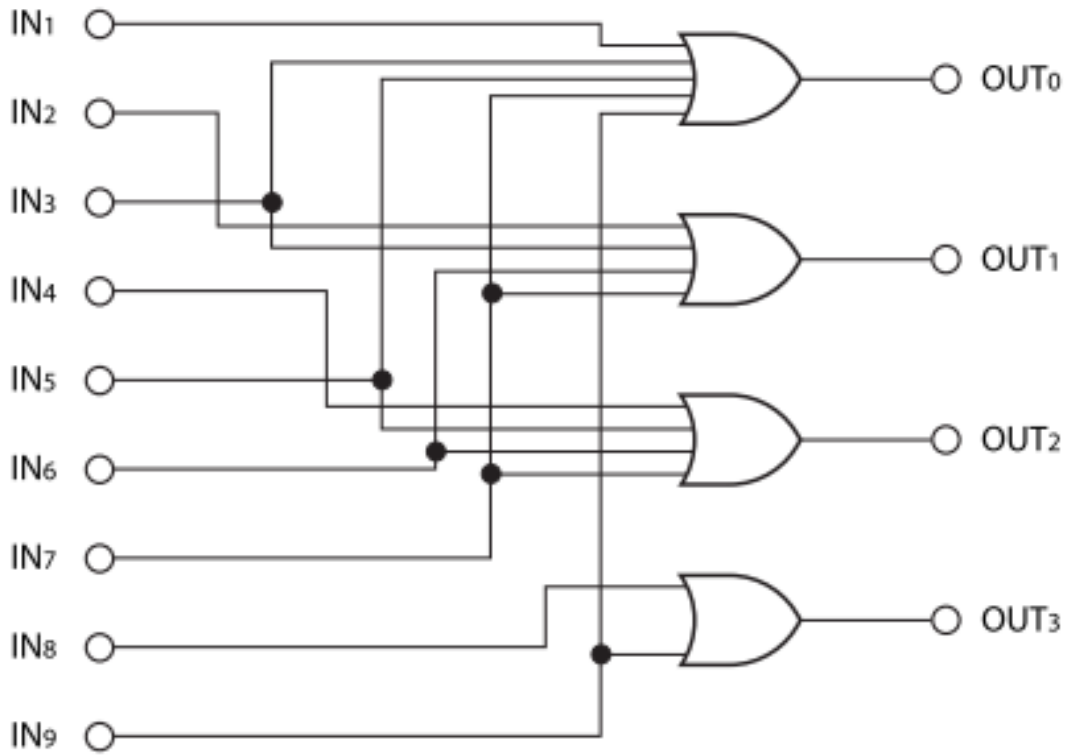



▲ 圖 5.1 十進位對 BCD 編碼器的方塊圖

- BCD 的輸出最低位元 (LSB) → 「OUT 0 」
- BCD 的輸出倒數第二低位元 → 「OUT 1 」
- BCD 的輸出倒數第三低位元 → 「OUT 2 」
- BCD 的輸出最高位元 (MSB) → 「OUT 3 」

| 開關 | $OUT_3$ | $OUT_2$ | $OUT_1$ | $OUT_0$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |

▲ 表 5.1　十進位對 BCD 編碼器的輸入輸出




▲ 圖 5.2　十進位對 BCD 編碼器

$$OUT_0 = IN_1 + IN_3 + IN_5 + IN_7 + IN_9$$
$$OUT_1 = IN_2 + IN_3 + IN_6 + IN_7$$
$$OUT_2 = IN_4 + IN_5 + IN_6 + IN_7$$
$$OUT_3 = IN_8 + IN_9$$

- 前述的電路有兩個問題：
  - 沒有使用到「$IN_0$」，無法分辨開關0的狀態
  - 無法處理同時按下兩個開關的狀態
- 下一小節將說明解決方法

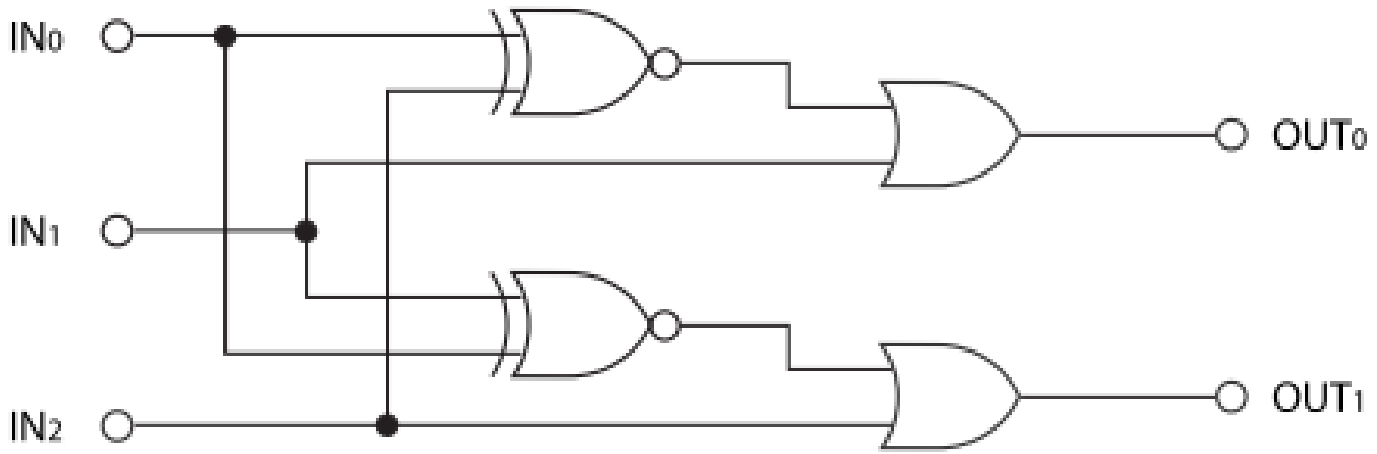
# 5.1.2 解決問題的編碼器

- 要解決前述問題，需要編碼器增加以上兩個狀態的處理方式：
  - 沒有開關按下或是多個開關按下時「$OUT_1$」 = '1',「$OUT_0$」 = '1'
  - 其餘狀態同前面的編碼器
- 對應的邏輯式：

$$OUT_0 = \overline{IN_2 \oplus IN_0} + IN_1$$
$$OUT_1 = \overline{IN_1 \oplus IN_0} + IN_2$$

| $IN_2$ | $IN_1$ | $IN_0$ | $OUT_1$ | $OUT_0$ | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | → 沒有按鈕被按下 |
| 0 | 0 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 0 | 1 | |
| 0 | 1 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 1 | 0 | |
| 1 | 0 | 1 | 1 | 1 | |
| 1 | 1 | 0 | 1 | 1 | |
| 1 | 1 | 1 | 1 | 1 | → 多個按鈕被按下 |

電路圖



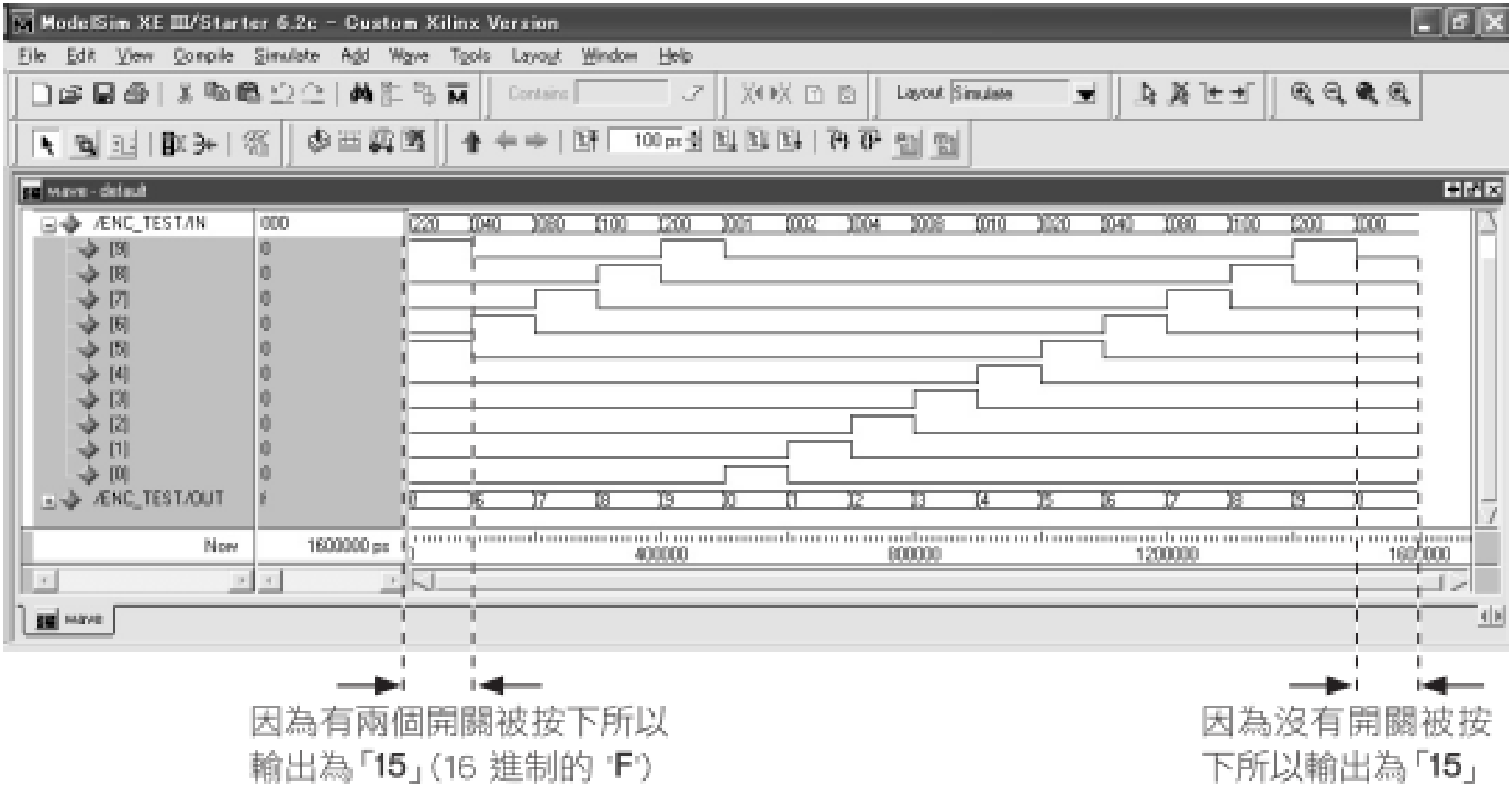
# 5.1.3 十進位對 BCD 編碼器的模擬

```
initial     begin
            j = { 5'b10001, 9'b0, 1'b1 };   ←—— 運用連接運算子
            for ( i = 0; i <= 15; i = i + 1 )
                  begin
                        j = j << 1;         ←—— 把 j 的值左移一個位元
                        IN = j[15:6];       ←—— 把 j 的 15~6 位元的值
                        #100;                   代入到 IN 裡
                  end
        $finish;
end
```

程式 5.2　十進位對 BCD 編碼器的 Verilog HDL 描述 (行為層)　LST

```
/* Data Difinision     */
`define                SW_IN0      10'b0000000001
`define                SW_IN1      10'b0000000010
`define                SW_IN2      10'b0000000100
`define                SW_IN3      10'b0000001000
`define                SW_IN4      10'b0000010000
`define                SW_IN5      10'b0000100000
`define                SW_IN6      10'b0001000000
`define                SW_IN7      10'b0010000000
`define                SW_IN8      10'b0100000000
`define                SW_IN9      10'b1000000000
```

用編譯指示子「`define」來定義巨集




▲ 圖 5.4　十進位對 BCD 編碼器的模擬結果

# 編譯指示子「`define」

- `define 語法為

```
`define    巨集名稱    巨集文字列
```

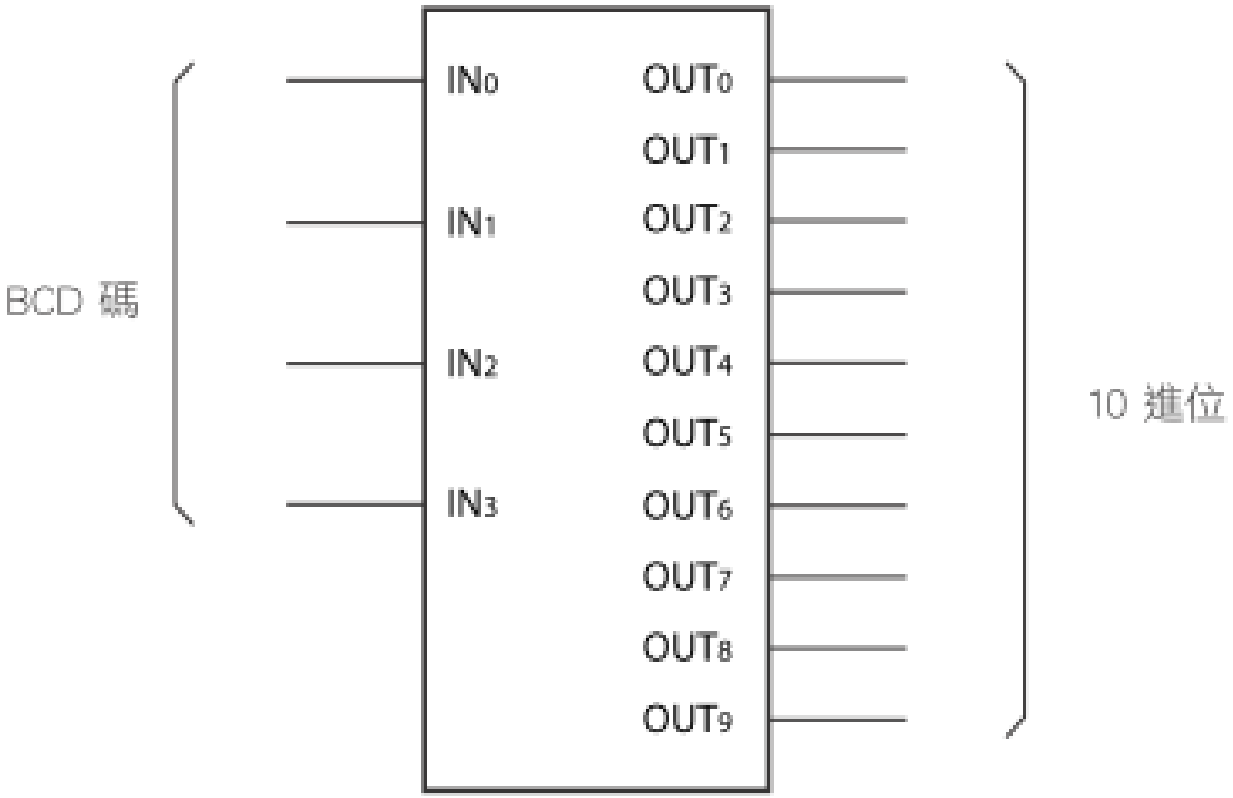
- 也可簡化為

```
`巨集名稱
```

# 5.2 解碼器

- 解碼器 (解碼電路: decoder) 是把編碼器進行編碼之後的信號還原回來的電路

# 5.2.1 BCD 對十進位解碼器




▲ 圖 5.5　BCD 對十進位解碼器的方塊圖

| $IN_3$ | $IN_2$ | $IN_1$ | $IN_0$ | $OUT_9$ | $OUT_8$ | $OUT_7$ | $OUT_6$ | $OUT_5$ | $OUT_4$ | $OUT_3$ | $OUT_2$ | $OUT_1$ | $OUT_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | — | — | — | — | — | — | — | — | — | — |
| 1 | 0 | 1 | 1 | — | — | — | — | — | — | — | — | — | — |
| 1 | 1 | 0 | 0 | — | — | — | — | — | — | — | — | — | — |
| 1 | 1 | 0 | 1 | — | — | — | — | — | — | — | — | — | — |
| 1 | 1 | 1 | 0 | — | — | — | — | — | — | — | — | — | — |
| 1 | 1 | 1 | 1 | — | — | — | — | — | — | — | — | — | — |

▲ 表 5.2　BCD 對十進位解碼器的真值表

— : don't care

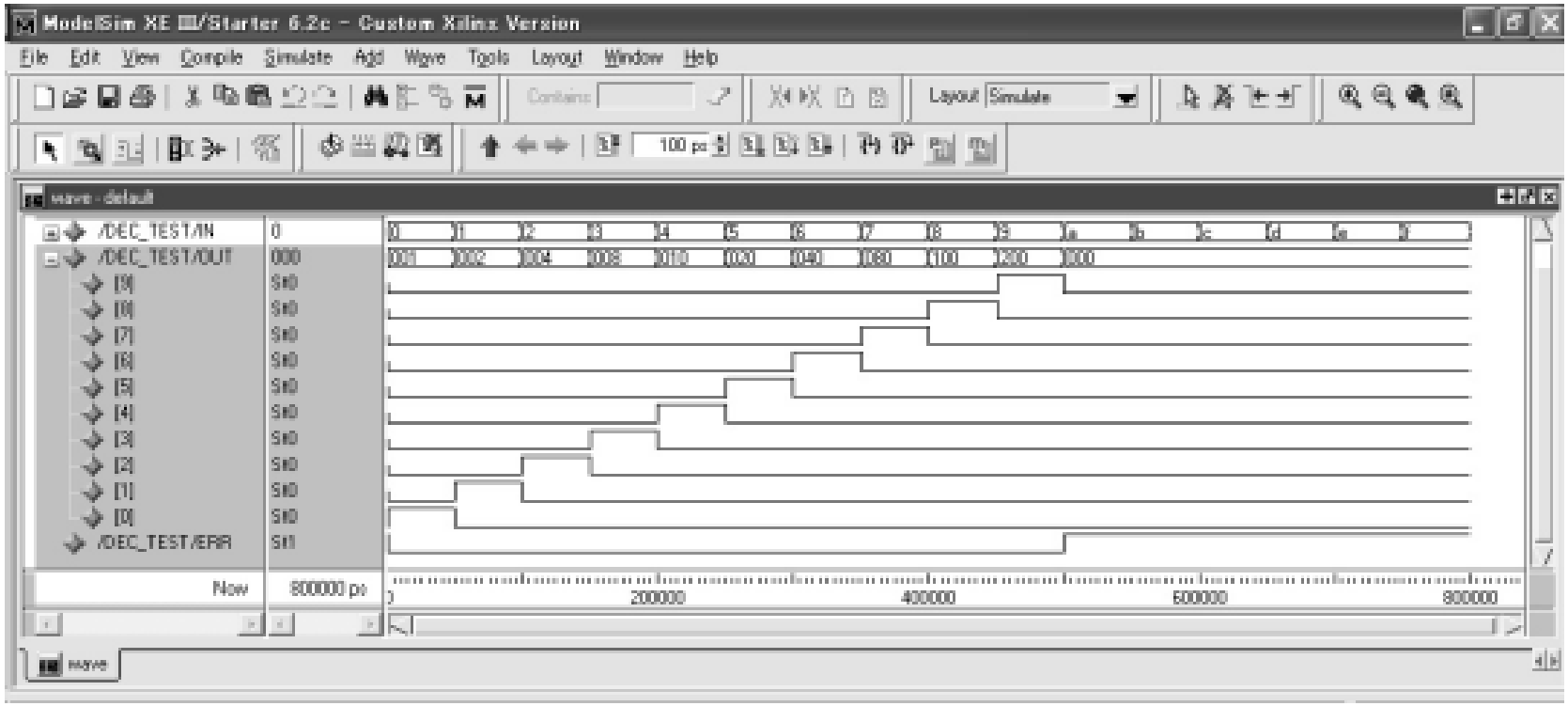
$$OUT_0 = IN_3 \cdot IN2_2 \cdot IN1_1 \cdot IN_0$$

- BCD 碼不使用「10」~「15」，邏輯式為：

$$OUT_0 = \overline{IN_3} \cdot \overline{IN_2} \cdot \overline{IN_1} \cdot \overline{IN_0}$$
$$OUT_1 = \overline{IN_3} \cdot \overline{IN_2} \cdot \overline{IN_1} \cdot IN_0$$
$$OUT_2 = \overline{IN_2} \cdot IN_1 \cdot \overline{IN_0}$$
$$OUT_3 = \overline{IN_2} \cdot IN_1 \cdot IN_0$$
$$OUT_4 = IN_2 \cdot \overline{IN_1} \cdot \overline{IN_0}$$
$$OUT_5 = IN_2 \cdot \overline{IN_1} \cdot IN_0$$
$$OUT_6 = IN_2 \cdot IN_1 \cdot \overline{IN_0}$$
$$OUT_7 = IN_2 \cdot IN_1 \cdot IN_0$$
$$OUT_8 = IN_3 \cdot \overline{IN_0}$$
$$OUT_9 = IN_3 \cdot IN_0$$

# 5.2.2 BCD 對十進位解碼器的模擬

- 前面提到的 don' t care 部分用輸出「ERR」= '1' 來表示
- 此處模擬的結果跟 P5-12 電路圖 5.6 等效,輸入為 'F', 根據模擬結果的「$OUT_0$」~「$OUT_9$」為 '0', 但與圖 5.8 (或公式5.6) 的$OUT_9$不吻合



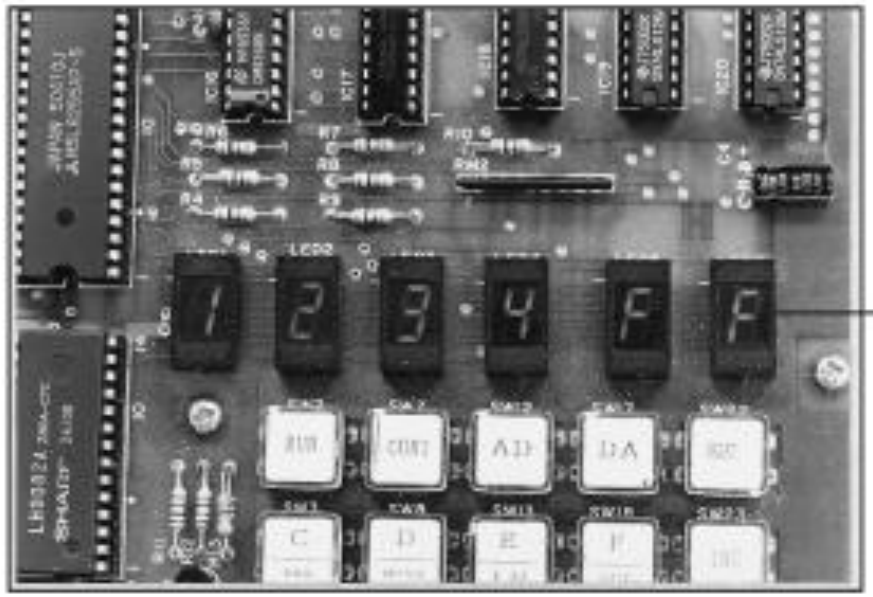

▲ 圖 5.9　BCD 對十進位解碼器的模擬結果

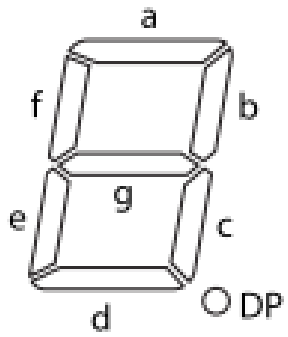
# 5.3 7 段顯示器的解碼器設計

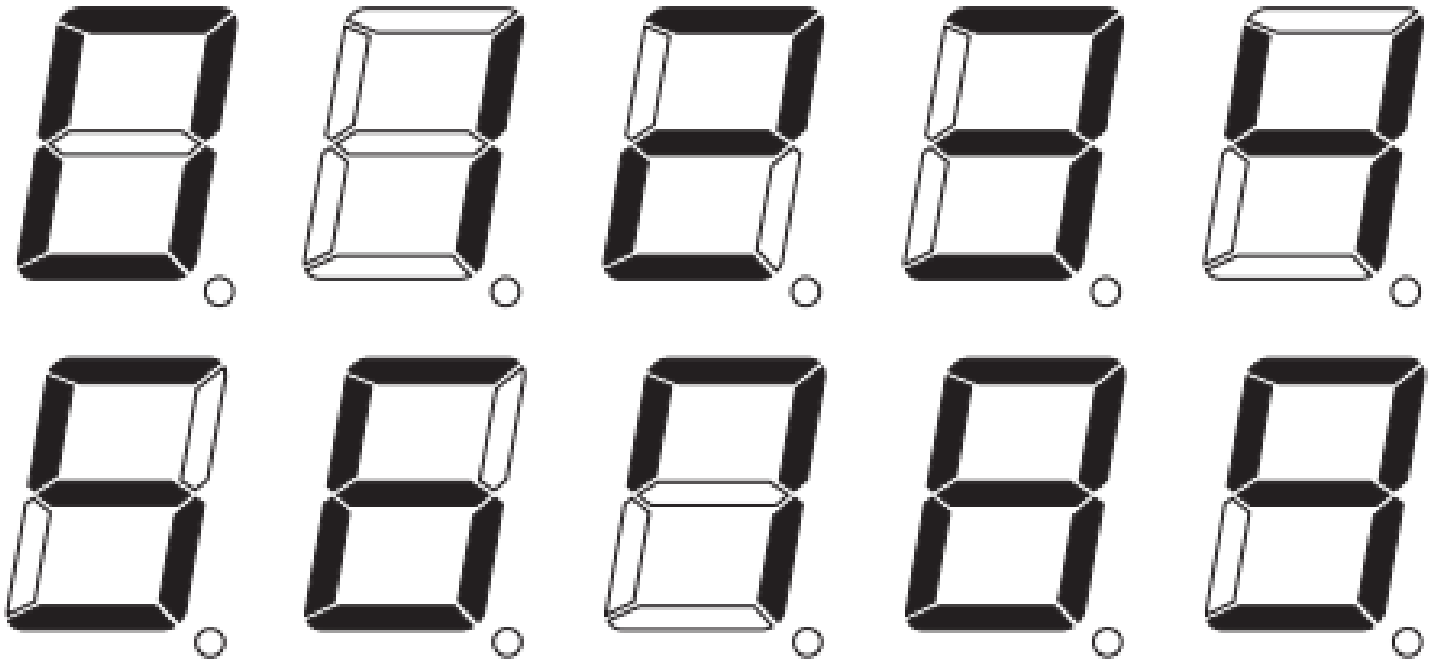
- 7 段顯示器, 是計算機或時鐘常用的表示, 透過 7 段顯示器 (含 DP 的話則為 8 段) 的 ON/OFF 來表示數字與一部份的英文字

# 5.3.2 真值表, 卡諾圖, 邏輯式, 電路圖

- 7 段顯示器的解碼器是用來將 BCD 碼或者 16 進制的值解碼成 7 段顯示器可以表示的值
- 透過 a~g 來表示數字 ‘0’ ~ ‘9‘, 輸入值超過 10, 一律會顯示 'E' (error)




a
f
b
g
e
c
d
DP



# 7 段顯示器解碼器的真值表

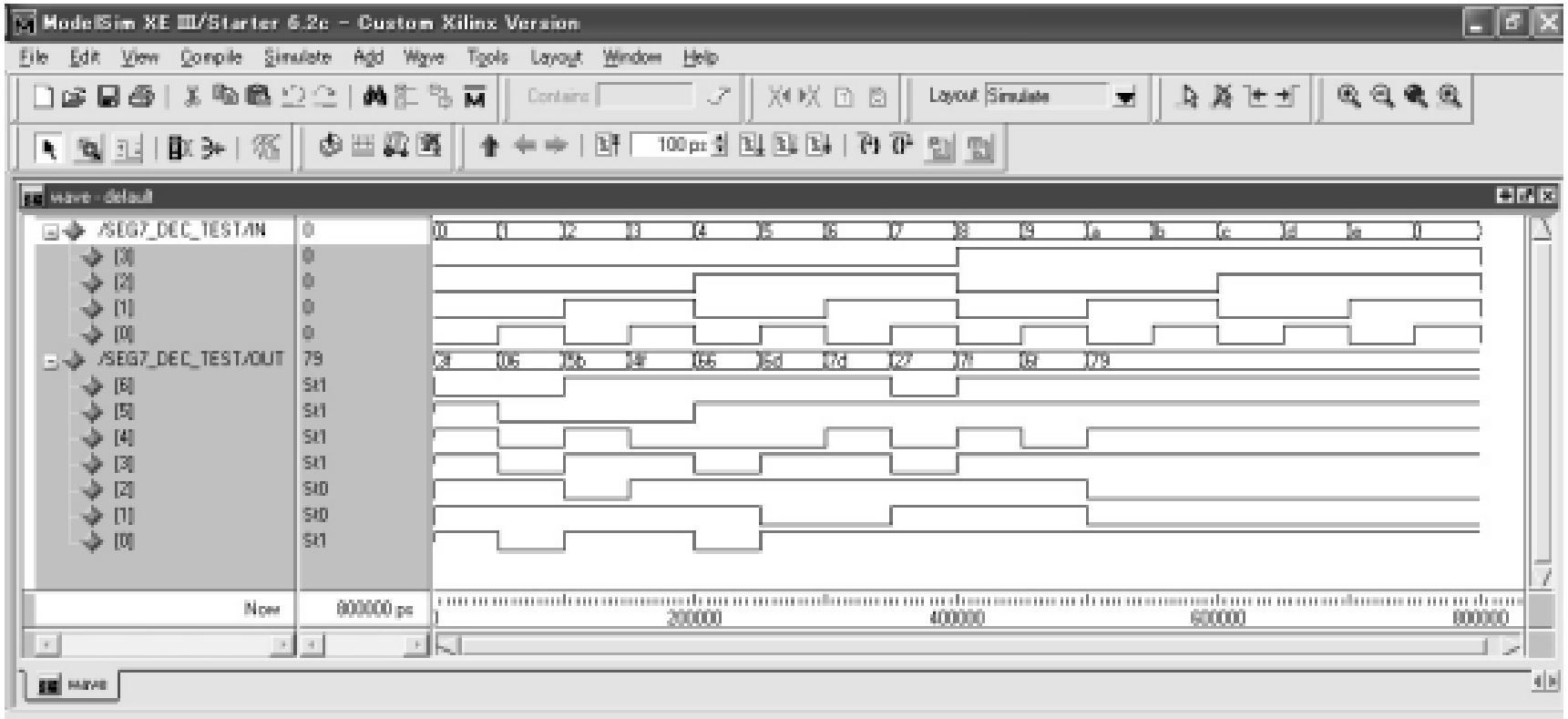
| $IN_3$ | $IN_2$ | $IN_1$ | $IN_0$ | SEG_g | SEG_f | SEG_e | SEG_d | SEG_c | SEG_b | SEG_a |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

「10」以上顯示為 'E'

# 7 段顯示器解碼器的邏輯式

$$
\begin{aligned}
SEG\_a &= IN_1 + IN_3 + \overline{IN_2 \oplus IN_0} \\
SEG\_b &= \overline{IN_3} \cdot \overline{IN_2} + \overline{IN_2} \cdot \overline{IN_1} + \overline{IN_3} \cdot \overline{(IN_1 \oplus IN_0)} \\
SEG\_c &= \overline{IN_3} \cdot IN_2 + \overline{IN_2} \cdot \overline{IN_1} + \overline{IN_3} \cdot IN_0 \\
SEG\_d &= IN_3 + IN_1 \cdot \overline{IN_0} + \overline{IN_2} \cdot IN_1 + \overline{IN_2} \cdot \overline{IN_0} + IN_2 \cdot \overline{IN_1} \cdot IN_0 \\
SEG\_e &= IN_3 \cdot IN_2 + IN_1 \cdot \overline{IN_0} + IN_3 \cdot IN_1 + \overline{IN_2} \cdot \overline{IN_0} \\
SEG\_f &= IN_3 + IN_2 + \overline{IN_1} \cdot \overline{IN_0} \\
SEG\_g &= IN_3 + IN_2 \cdot \overline{IN_0} + (IN_2 \oplus IN_1)
\end{aligned}
$$

# 5.3.3 7 段顯示器解碼器的模擬




▲ 圖 5.13　7 段顯示器解碼器的模擬結果

# 5.4 同位電路

- 資料在傳輸中遭遇到雜訊之類的干擾, 同位 (parity) 的功能是為了檢查出資料有否錯誤
- 計算資料中有幾個 '1', 以及把事先決定好的奇同位/偶同位加入到資料的稱為同位元產生器
- 檢查資料中 '1' 有幾個的動作稱為同位元檢查

# 5.4.1 4 位元同位元產生器

- 同位元分為偶同位和奇同位

# 偶同位的情況

- 4 位元偶數同位元產生器

  輸入 4 位元 → A, B, C, D

  輸出 → [ODD_OUT]

  → A~D 的 '1' 的個數如果是奇數個的話, [ODD_OUT] → '1'

  如果是偶數個的話, [ODD_OUT → '0'

# 偶同位的真值表和邏輯式

| D | C | B | A | ODD_OUT |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

'A' ~ 'D' 的 '1' 如果是奇數個 → '1'

'A' ~ 'D' 的 '1' 如果是偶數個 → '0'

$$ODD\_OUT = D \oplus C \oplus B \oplus A$$

▲ 表 5.4　4 位元偶數同位元產生器的真值表

# 奇同位的情況

輸出 → [EVEN_OUT]

A~D 的 '1' 的個數如果是奇數個的話, [ODD_OUT] →'0'

如果是偶數個的話, [ODD_OUT] →'1'

- 奇同位的邏輯式

$$EVEN\_OUT = \overline{D \oplus C \oplus B \oplus A}$$

# 5.4.2 5 位元同位元偵錯器 – 偶同位

- 信號名稱如下所示

輸入 4 位元　→A, B, C, D, [ODD_OUT]
輸出　→[P_ERR]
A~D, [ODD_OUT]的 '1' 的個數如果是奇數的話, [P_ERR]　→'1'
(同位錯誤)
如果是偶數的話, [P_ERR]　→'0'

- 其動作為「所有輸入互相做 XOR, 如果輸入有奇數個 '1' 輸出就是 '1'」
- 其邏輯式如下

$$P\_ERR = ODD\_OUT \oplus D \oplus C \oplus B \oplus A$$

# 5 位元同位元偵錯器 – 奇同位

- 信號名稱如下所示

輸入 4 位元 → A, B, C, D, [EVEN_OUT]

輸出 → [P_ERR]

→ A~D, [EVEN_OUT]的 '1' 的個數如果是奇數的話, [P_ERR] →'0'

如果是偶數的話, [P_ERR] →'1'

(同位錯誤)

- 其邏輯式為

$$P\_ERR = \overline{EVEN\_OUT \oplus D \oplus C \oplus B \oplus A}$$

# 5.4.3 4 位元偶數同位元產生器的模擬

- 資料流層可根據公式 5.9 用 assign 將邏輯式用位元運算子描述如下

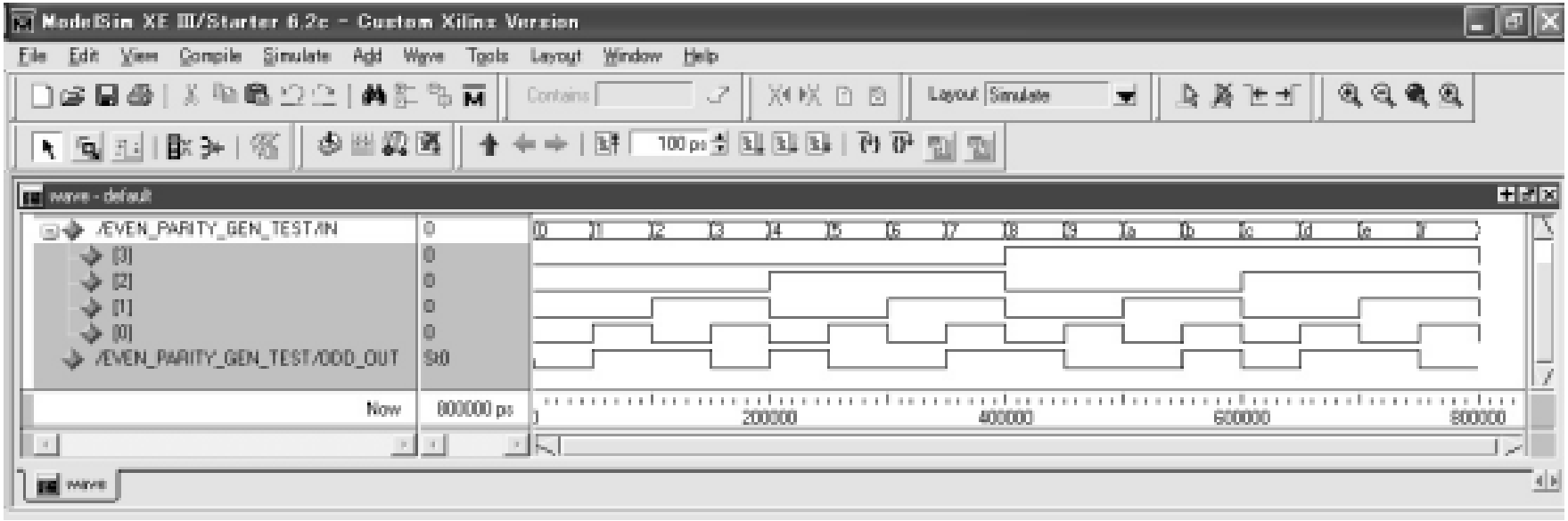
```
ODD_OUT=IN[3] ^ IN[2] ^ IN[1] ^ IN[0];
```

- 也可以用簡化運算子描述如下

```
ODD_OUT = ^ (IN);
```

- 本電路也可以用 for 來做行為層的描述；一開始把 '0' 代入「FUNC_GEN」，透過 for，一邊遞增整數型變數 'i',一邊把「IN[i]」代入「FUNC_GEN」做 XOR。

# 模擬結果




▲ 圖 5.19　4 位元偶數同位元產生器的模擬結果

# 5.5 正邏輯與負邏輯

- 正邏輯:
  高電壓 (H) 為「真 (1)」, 低電壓 (L) 為「偽 (0)」

- 負邏輯:
  高電壓 (H) 為「偽 (0)」, 低電壓 (L) 為「真 (1)」

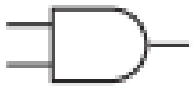
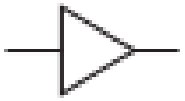
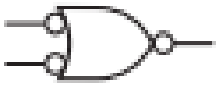
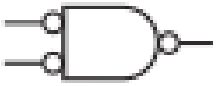
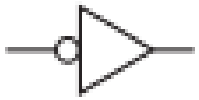
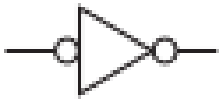
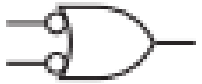
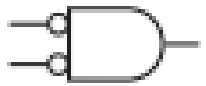
# 5.5.1 互斥或電路

- 透過「笛摩根定律」可以把 AND/OR 變成 NAND/NAND

$$
\begin{aligned}
Y &= \overline{A \cdot \overline{B} + \overline{A} \cdot B} \\
&= \overline{\overline{A \cdot \overline{B}} \cdot \overline{\overline{A} \cdot B}} \\
&= \overline{A \cdot \overline{B}} \cdot \overline{\overline{A} \cdot B}
\end{aligned}
$$

- 標明出正邏輯/負邏輯的電路圖，都是以加一個 '○' 來識別負邏輯信號

- 1. 輸入為正邏輯的時候 AND/OR 的邏輯不變
  輸入為負邏輯的時候 AND 跟 OR 會互換
- 2. 對 AND 電路, OR 電路與緩衝器電路來說,
  輸入為正邏輯的時候輸出也是正邏輯
  輸入為負邏輯的時候輸出也是負邏輯
  輸入輸出之間的邏輯不會改變
- 3. NAND 電路, NOR 電路與 NOT 電路
  輸入為正邏輯的時候輸出是負邏輯
  輸入為負邏輯的時候輸出是正邏輯
  輸入輸出之間的邏輯剛好會相反

| | | | | | | |
|---|---|---|---|---|---|---|
| 正邏輯輸入 |  |  |  |  |  |  |
| 負邏輯輸入 |  |  |  |  |  |  |

▲ 表 5.5　正邏輯/負邏輯輸入的邏輯符號