第四章

組合邏輯

4-1 組合電路

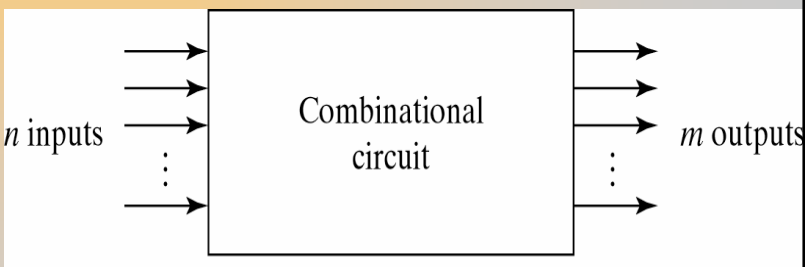


Fig. 4-1 Block Diagram of Combinational Circuit

4-3 設計步驟

★組合電路的設計

1.由電路的敘述，決定所需的輸入與輸出的個數並且對每一個輸入與輸出安排一個變數符號。

2.導出真值表並定義輸入與輸出間的關係。

3.對每一個輸出求出以輸入變數爲函數之簡化的布林函數。

4.畫出邏輯圖並且證明設計的正確性。

BCD碼到超3碼轉換器

表 4-2　碼轉換例題的真值表

BCD 輸入				超 3 碼輸出			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

BCD到超3碼卡諾圖

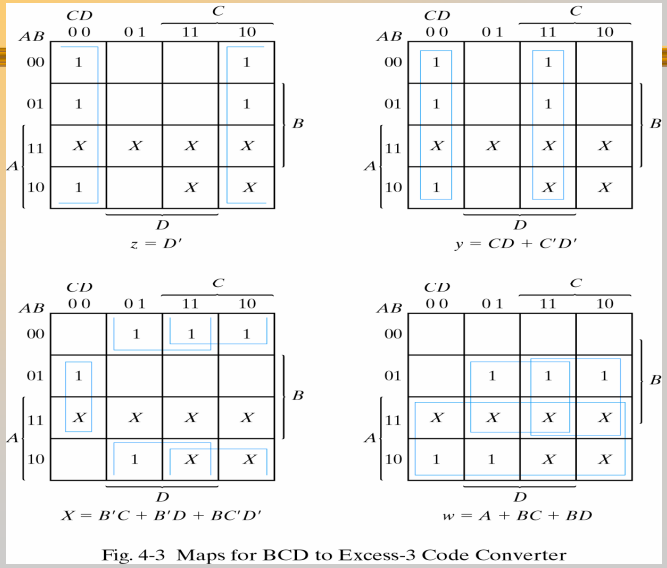


Fig. 4-3 Maps for BCD to Excess-3 Code Converter

BCD到超3碼電路圖

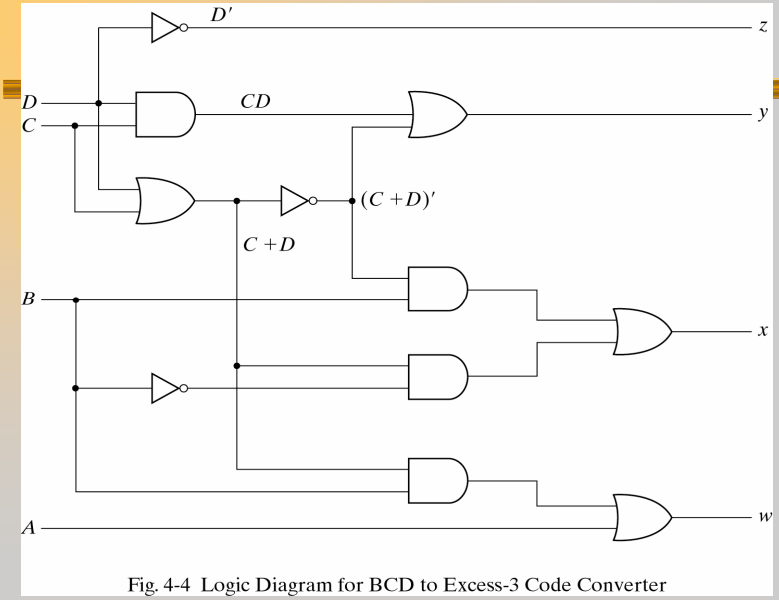


Fig. 4-4 Logic Diagram for BCD to Excess-3 Code Converter

4-4 二進位加法器---減法器

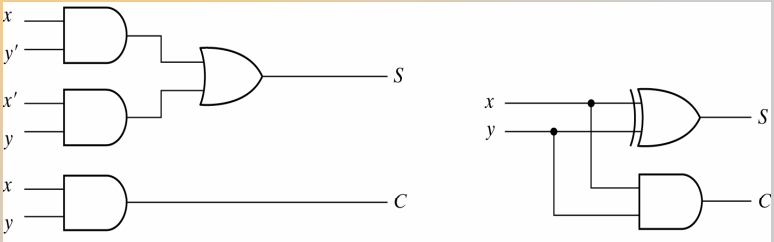
★半加法器(half adder)

x	y	S	C
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

$S = x'y + xy'$ $C = xy$

半加法器

★電路



(a) $S = xy' + x'y$, $C = xy$ (b) $S = x \oplus y$, $C = xy$

Fig. 4-5 Implementation of Half-Adder

全加法器

- 真值表

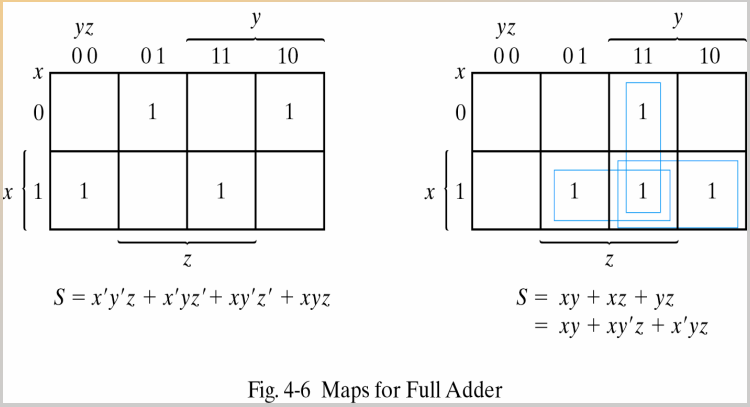
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$C = xy + yz + xz$$

全加法器

- 卡諾圖



$S = x'y'z + x'yz' + xy'z' + xyz$

$S = xy + xz + yz$
$= xy + xy'z + x'yz$

Fig. 4-6 Maps for Full Adder

全加法器

★電路

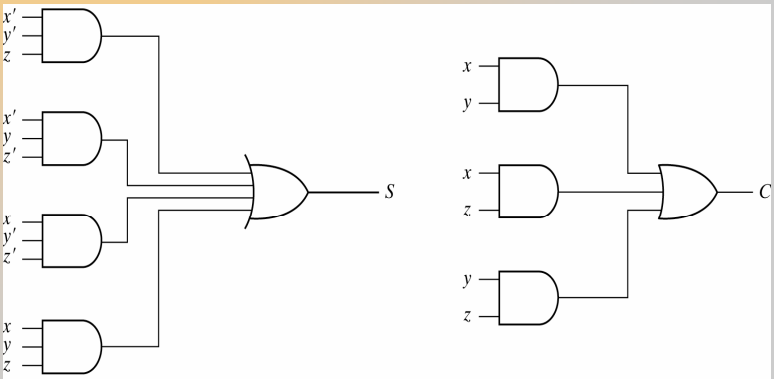


Fig. 4-7 Implementation of Full Adder in Sum of Products

二進位加法器

$A = 1011$ $B = 0011$

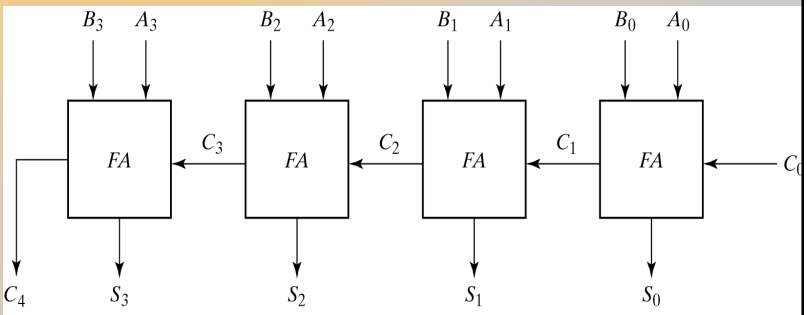


Fig. 4-9 4-Bit Adder

$S = 1110$

進位傳播

進位產生 G_i　　進位傳播 P_i

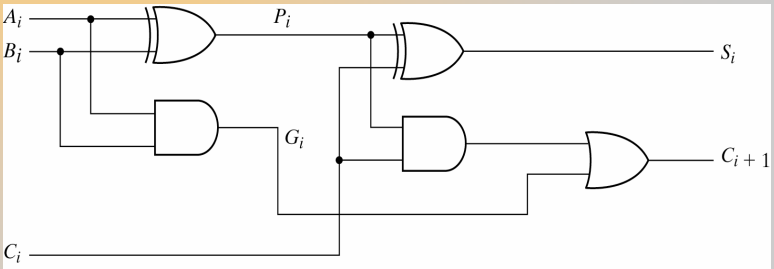


Fig. 4-10 Full Adder with P and G Shown

$$P_i = A_i \oplus B_i \qquad S_i = P_i \oplus C_i$$

$$G_i = A_i B_i \qquad C_{i+1} = G_i + P_i C_i$$

進位遞迴產生器

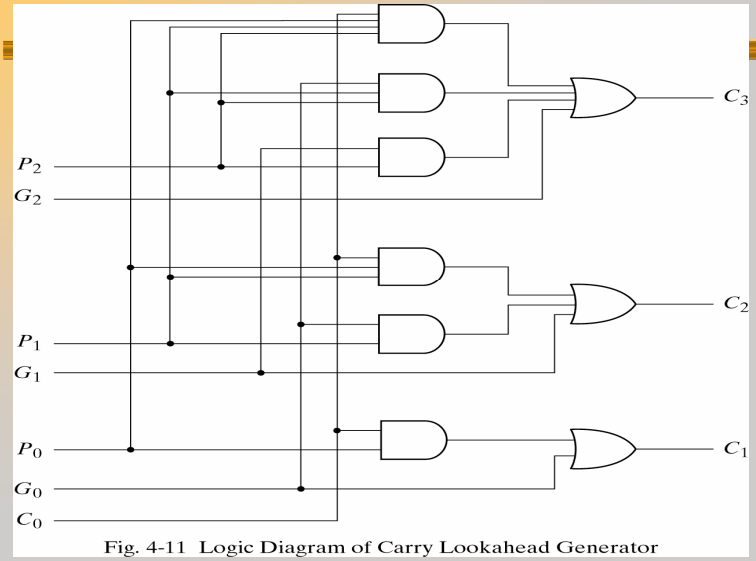


Fig. 4-11 Logic Diagram of Carry Lookahead Generator

具有進位遞迴的4位元加法器

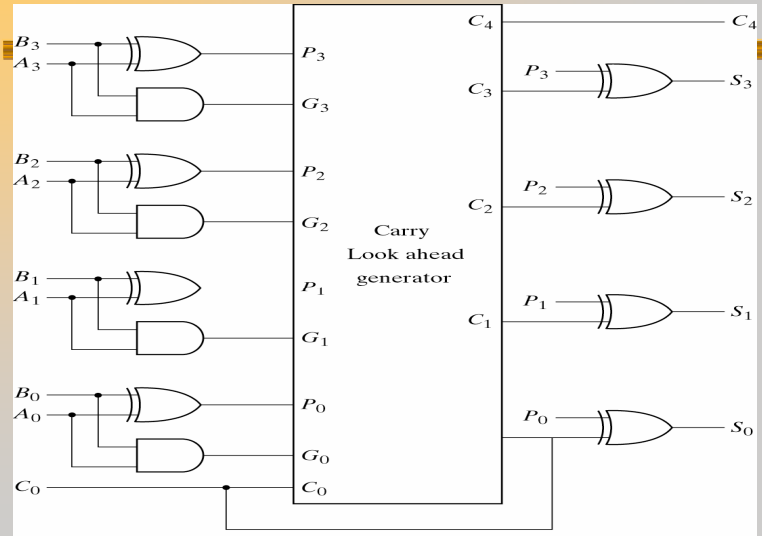


Fig. 4-12 4-Bit Adder with Carry Lookahead

二進位減法器

✶4位元加法器---減法器

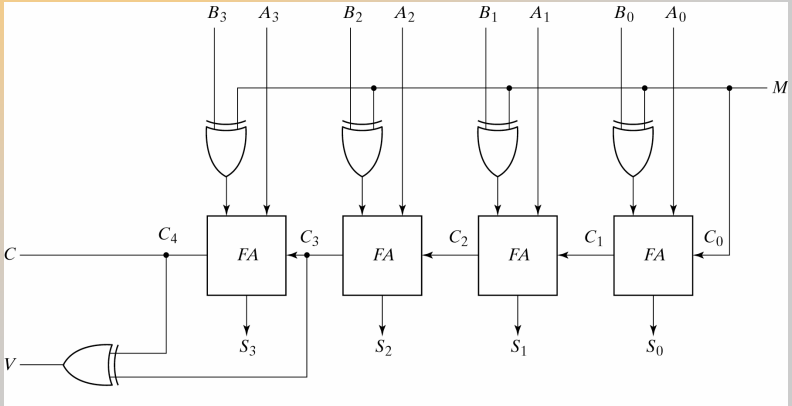


Fig. 4-13 4-Bit Adder Subtractor

4-5 十進位加法/器

★BCD加法器的推導

表 4-5　BCD 加法器的推導

二進位和					BCD 和					十進位
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

BCD加法器電路

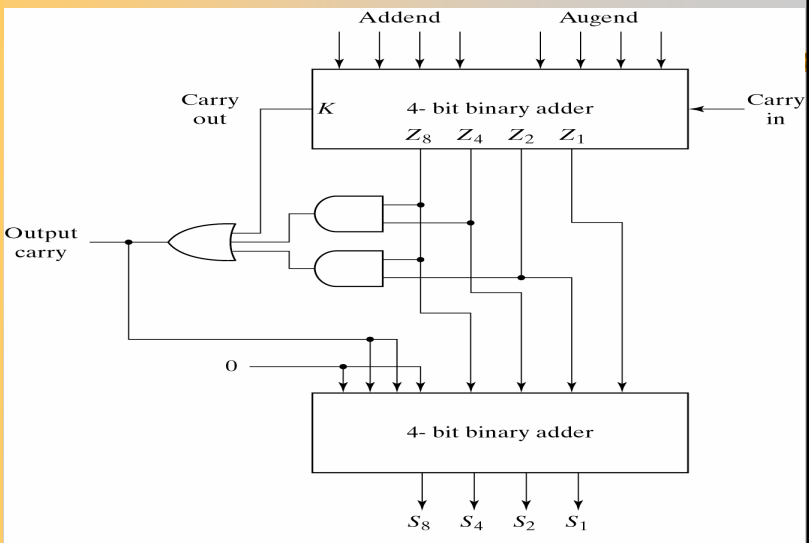


Fig. 4-14 Block Diagram of a BCD Adder

4-6 二進位乘法器

★2位元乘2位元

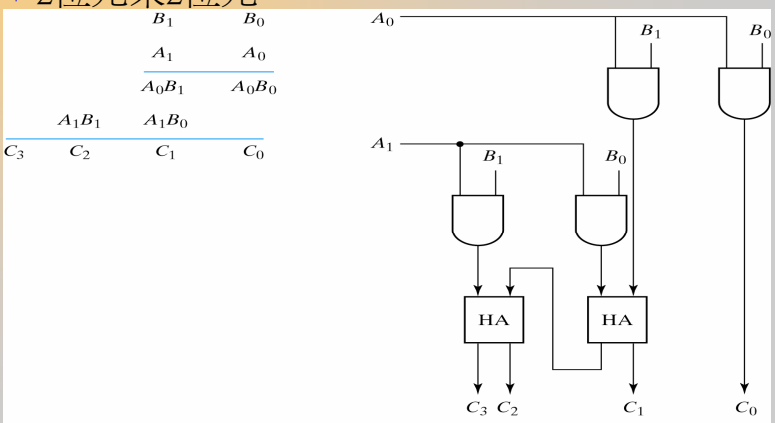


Fig. 4-15 2-Bit by 2-Bit Binary Multiplier

4位元乘3位元之二進位乘法器

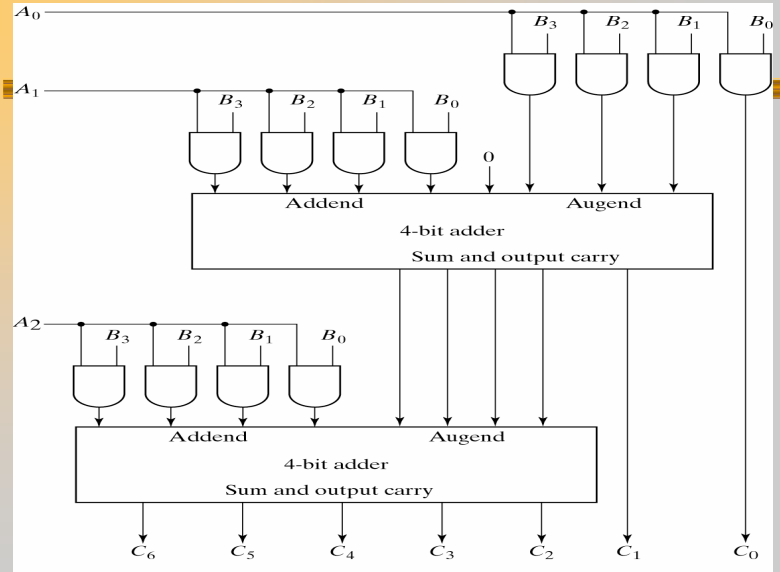


Fig. 4-16 4-Bit by 3-Bit Binary Multiplier

4-7 大小比較器

$$A = A_3A_2A_1A_0$$
$$B = B_3B_2B_1B_0$$

$$x_i = A_iB_i + A_i'B_i'$$

$$(A = B) = x_3x_2x_1x_0$$

$$(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$

$$(A < B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$$

4位元大小比較器

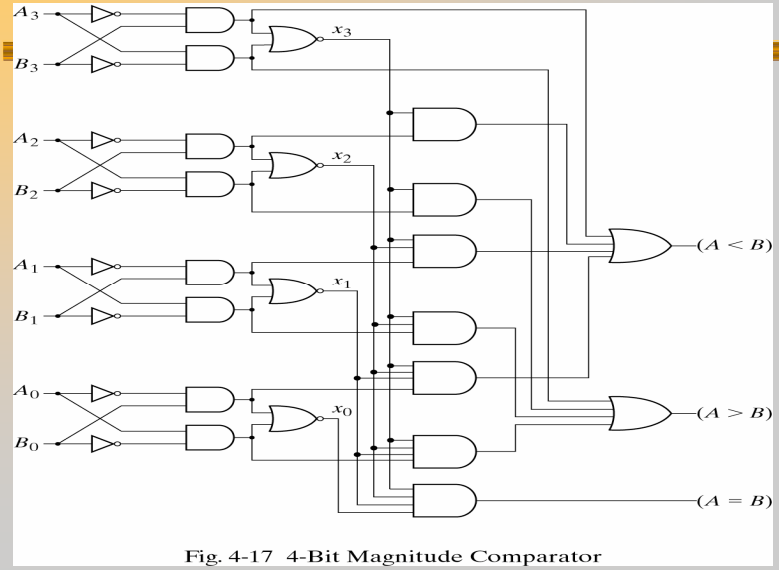


Fig. 4-17 4-Bit Magnitude Comparator

4-8 解碼器

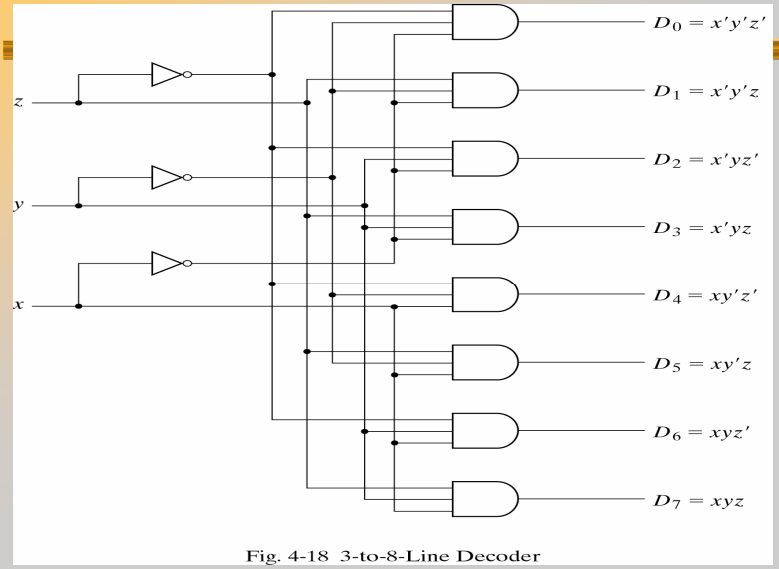
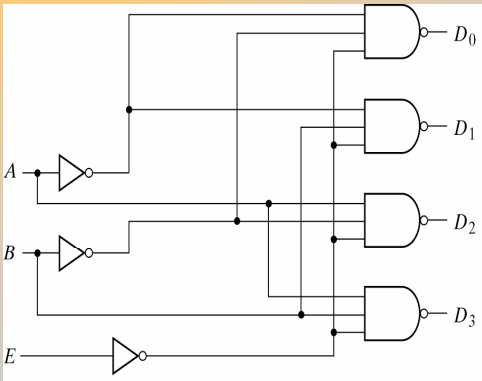


Fig. 4-18 3-to-8-Line Decoder

具有致能輸入之2對4線解碼器



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(a) Logic diagram (b) Truth table

Fig. 4-19 2-to-4-Line Decoder with Enable Input

利用3x8解碼器建立4x16解碼器

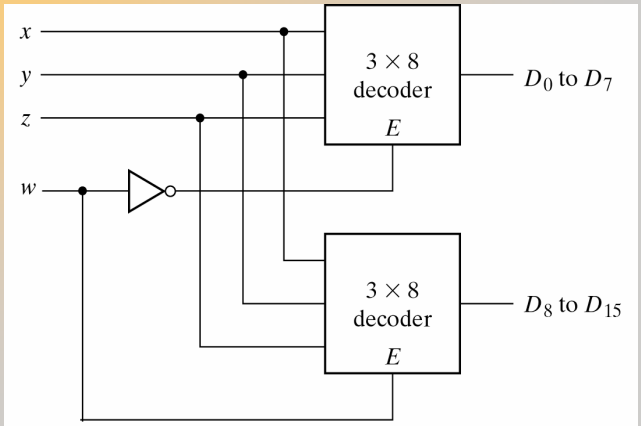


Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders

利用解碼器實現組合邏輯電路

$$S(x,y,z) = \sum(1,2,4,7)$$

$$C(x,y,z) = \sum(3,5,6,7)$$

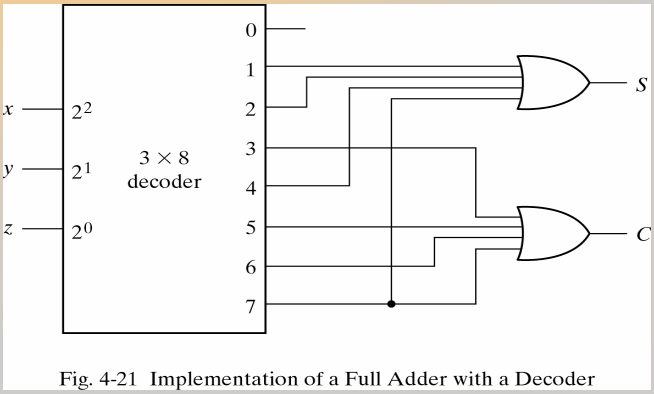


Fig. 4-21 Implementation of a Full Adder with a Decoder

4-9 編碼器

表 4-7　八進制對二進制編碼器之真值表

輸入								輸出		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

優先權編碼器之真值表

表 4-8　優先權編碼器之真值表

輸入				輸出		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

優先權編碼器卡諾圖

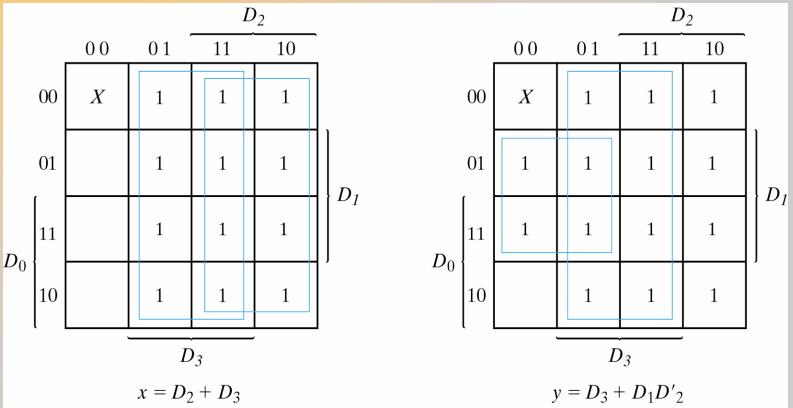


Fig. 4-22 Maps for a Priority Encoder

4輸入優先權編碼器

$$x = D_2 + D_3$$

$$y = D_3 + D_1D_2'$$

$$V = D_0 + D_1 + D_2 + D_3$$

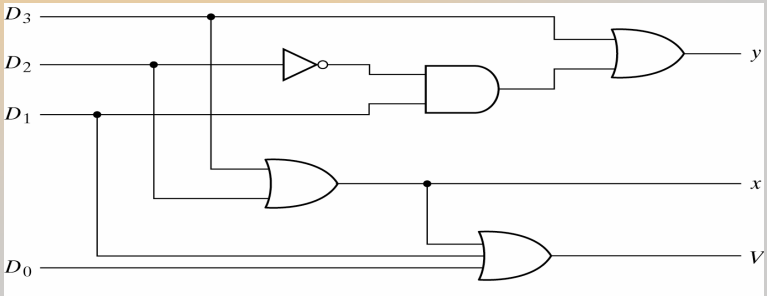
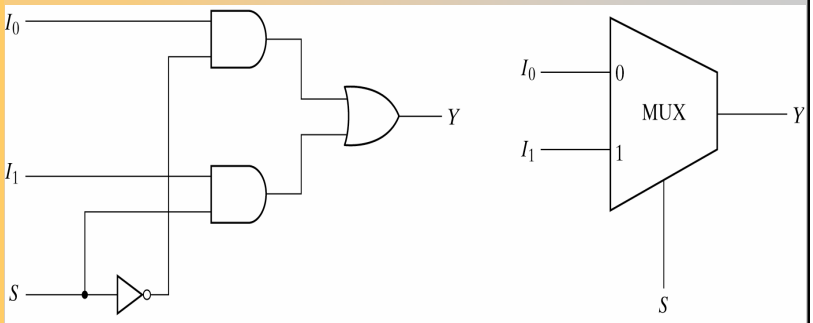


Fig. 4-23 4-Input Priority Encoder

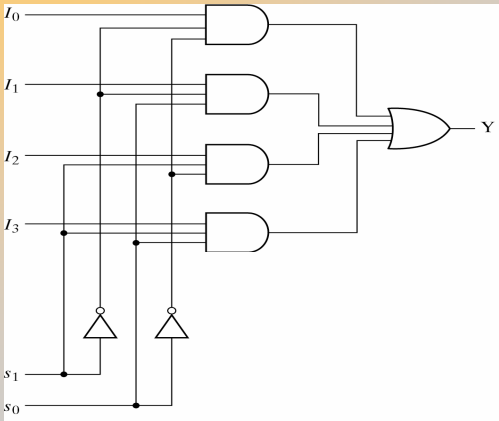
4-10 多工器(Multiplexers)



(a) Logic diagram (b) Block diagram

Fig. 4-24 2-to-1-Line Multiplexer

4對1線 多工器



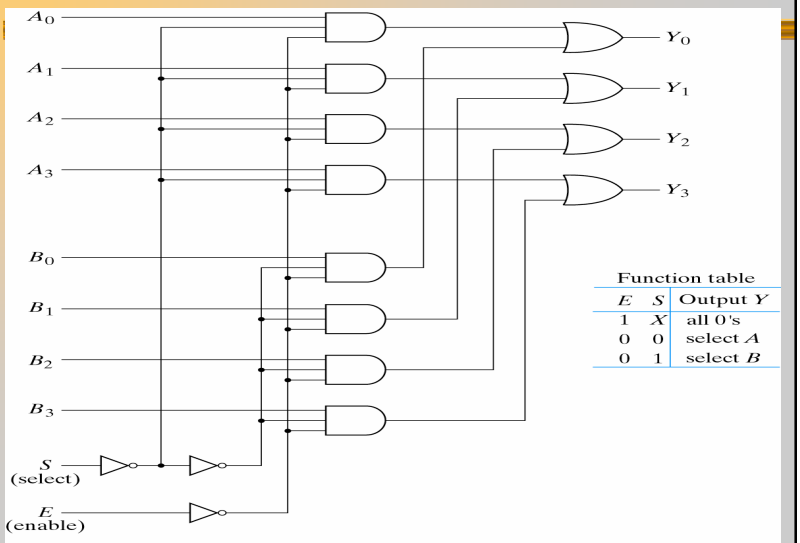
s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

(a) Logic diagram

Fig. 4-25 4-to-1-Line Multiplexer

四重2對1線 多工器



Function table

E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B

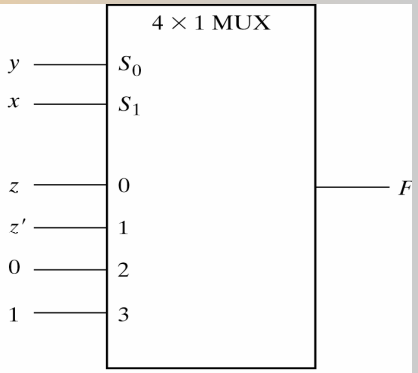
Fig. 4-26 Quadruple 2-to-1-Line Multiplexer

利用多工器實現布林函數

$$F(x,y,z) = \sum(1,2,6,7)$$

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table



(b) Multiplexer implementation

Fig. 4-27 Implementing a Boolean Function with a Multiplexer

利用多工器實現一個4輸入函數

$$F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$$

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	

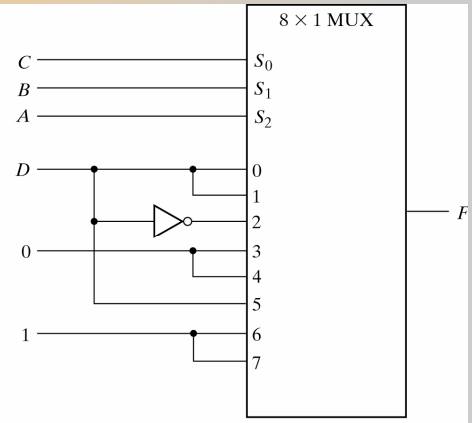


Fig. 4-28 Implementing a 4-Input Function with a Multiplexer

三態閘

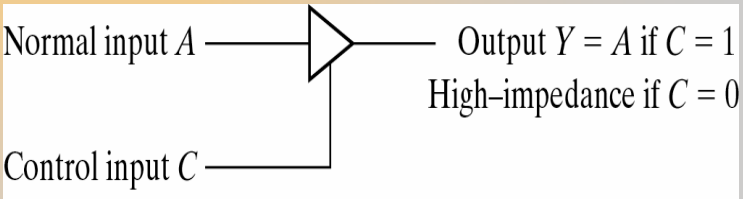


Fig. 4-29 Graphic Symbol for a Three-State Buffer

利用三態閘構成多工器

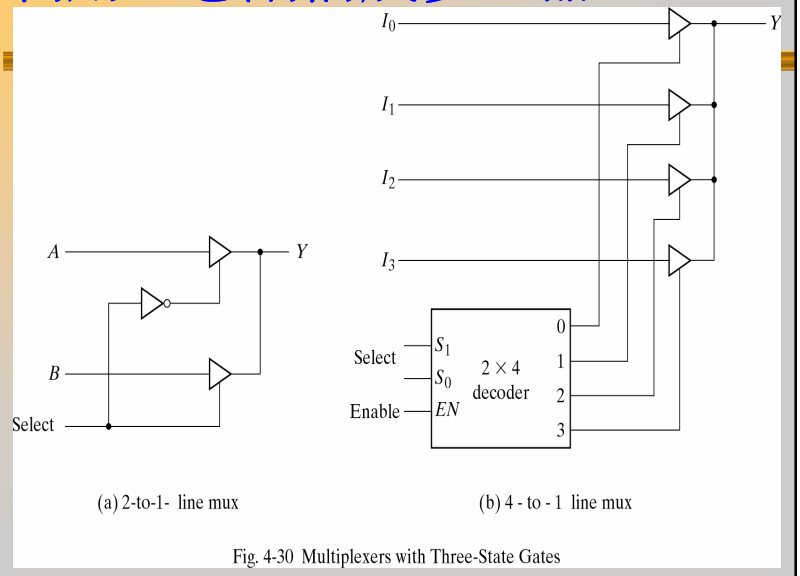


Fig. 4-30 Multiplexers with Three-State Gates

4-11 組合電路的硬體描述語言

★ 閘階層模型

關鍵字 and 、nand 、or 、nor 、xor 、xnor 、not 、buf

表 4-9 預先定義原始閘的真值表

and	0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

or	0	1	x	z
0	0	1	x	x
1	1	1	1	1
x	x	1	x	x
z	x	1	x	x

xor	0	1	x	z
0	0	1	x	x
1	1	0	x	x
x	x	x	x	x
z	x	x	x	x

not	輸入	輸出
	0	1
	1	0
	x	x
	z	x

HDL 範例 4-1
(2對4線解碼器的閘階層描述)

```
//Gate-level description of a 2-to-4-line decoder
//Figure 4-19
module decoder_gl (A,B,E,D);
  input A,B,E;
  output [0:3]D;
  wire Anot,Bnot,Enot;
  not
    n1 (Anot,A),
    n2 (Bnot,B),
```

HDL 範例 4-1
(2對4線解碼器的閘階層描述)

```
    n3 (Enot,E);
  nand
    n4 (D[0],Anot,Bnot,Enot),
    n5 (D[1],Anot,B,Enot),
    n6 (D[2],A,Bnot,Enot),
    n7 (D[3],A,B,Enot);
endmodule
```

HDL 範例 4-2
(4位元加法器之底部向上層次化描述)

```
//Gate-level hierarchical description of 4-bit adder
// Description of half adder (see Fig 4-5b)
module  halfadder (S,C,x,y);
   input  x,y;
   output  S,C;
//Instantiate primitive gates
   xor  (S,x,y);
   and (C,x,y);
```

HDL 範例 4-2
(4位元加法器之底部向上層次化描述)

```
endmodule
//Description of full adder (see Fig 4-8)
module  fulladder (S,C,x,y,z);
   input  x,y,z;
   output  S,C;
   wire  S1,D1,D2;
//Outputs of first XOR and two AND gates
//Instantiate the halfadder
    halfadder  HA1 (S1,D1,x,y),
```

HDL 範例 4-2
(4位元加法器之底部向上層次化描述)

```
        HA2 (S,D2,S1,z);
  or  g1(C,D2,D1);
endmodule
//Description of 4-bit adder (see Fig 4-9)
module  _4bit_adder (S,C4,A,B,C0);
  input  [3:0] A,B;
  input  C0;
```

HDL 範例 4-2
(4位元加法器之底部向上層次化描述)

```
  output  [3:0] S;
  output  C4;
  wire  C1,C2,C3;   //Intermediate carries
//Instantiate the fulladder
  fulladder  FA0 (S[0],C1,A[0],B[0],C0),
             FA1 (S[1],C2,A[1],B[1],C1),
             FA2 (S[2],C3,A[2],B[2],C2),
             FA3 (S[3],C4,A[3],B[3],C3);
endmodule
```

三態閘

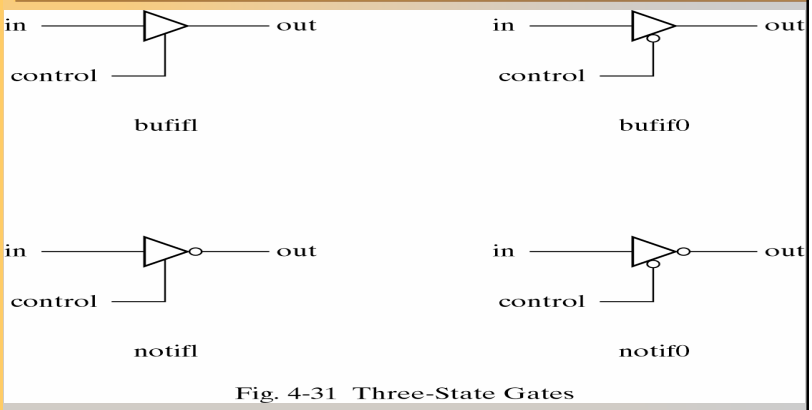


Fig. 4-31 Three-State Gates

bufif1 (OUT, A, control);
notif0 (Y, B, enable);

具有三態緩衝器之2對1線多工器

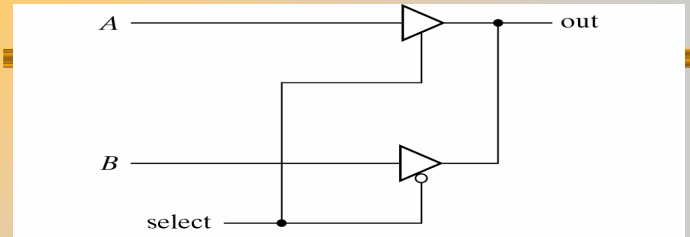


Fig. 4-32 2-to-1-Line Multiplexer with Three-State Buffers

```
module muxtri (A, B, select, OUT);
    input A, B, select;
    output OUT;
    tri OUT;
    bufif1 (OUT, A, select);
    bufif0 (OUT, B, select);
endmodule
```

資料流程模型

★Verilog HDL運算子

表 4-10　Verilog HDL 運算子

符號	運算
+	binary addition；二進位加法
−	binary subtraction；二進位減法
&	bit-wise AND；位元的及運算
|	bit-wise OR；位元的或運算
^	bit-wise XOR；位元的互斥或運算
~	bit-wise NOT；位元的反相運算
==	equality；全等
>	greater than；大於
<	less than；小於
{ }	concatenation；連結
?:	conditional；條件式

HDL 範例 4-3
(2對4線解碼器的資料流程描述)

```
//Dataflow description of a 2-to-4-line decoder
//See Fig.4-19
module  decoder_df (A,B,E,D);
  input  A,B,E;
  output  [0:3] D;
  assign  D[0] = ~(~A & ~B & ~E),
          D[1] = ~(~A & B & ~E),
          D[2] = ~(A & ~B & ~E),
          D[3] = ~(A & B & ~E);
endmodule
```

HDL 範例 4-4
(4位元加法器之資料流程描述)

```
//Dataflow description of 4-bit adder
module  binary_adder (A,B,Cin,SUM,Cout);
  input  [3:0] A,B;
  input  Cin;
  output  [3:0] SUM;
  output  Cout;
  assign  {Cout,SUM} = A + B + Cin;
endmodule
```

HDL 範例 4-5
(大小比較器之資料流程描述)

```
//Dataflow description of a 4-bit comparator.
module  magcomp (A,B,ALTB,AGTB,AEQB);
  input  [3:0] A,B;
  output  ALTB,AGTB,AEQB;
  assign  ALTB = (A < B),
          AGTB = (A > B),
          AEQB = (A = = B);
endmodule
```

HDL 範例 4-6
(2對1線多工器使用條件式運算子描述)

```
//Dataflow description of 2-to-1-line multiplexer
module  mux2x1_df (A,B,select,OUT);
   input  A,B,select;
   output  OUT;
   assign  OUT = select ? A : B;
endmodule
```

行爲模型

✶HDL 範例 4-7
(2對1線多工器使用條件式運算子描述)

```
//Behavioral description of 2-to-1-line multiplexer
module  mux2x1_bh(A,B,select,OUT);
   input  A,B,select;
   output  OUT;
   reg  OUT;
   always @ (select  or  A  or  B)
        if  (select == 1)  OUT = A;
        else  OUT = B;
endmodule
```

HDL 範例 4-8
(4對1線多工器之行爲描述)

```
//Behavioral description of 4-to-1- line multiplexer
//Describes the function table of  Fig. 4-25(b).
module  mux4x1_bh (i0,i1,i2,i3,select,y);
   input  i0,i1,i2,i3;
   input  [1:0] select;
   output  y;
   reg  y;
```

HDL 範例 4-8
(4對1線多工器之行爲描述)

```
   always  @ (i0 or i1 or i2 or i3 or select)
         case  (select)
            2'b00: y = i0;
            2'b01: y = i1;
            2'b10: y = i2;
            2'b11: y = i3;
         endcase
endmodule
```

測試平台

- 一個模擬模組是具有下列形式的HDL程式
  module 測試名稱。
  宣告特有的reg和wire識別字。
  在測試下例示設計模組。
  利用initial及always敘述產生模擬。
  顯示輸出響應。
  endmodule。

系統功能

- **$display**--顯示具有 end-of-line return之變數或字串的一次值。
- **$write**—與 **$display**相同，但是沒有到下一行。
- **$monitor**—當在模擬期間值改變時及顯示變數。
- **$time**--顯示模擬時間。
- **$finish**--結束模擬。

模擬與設計的交互模組

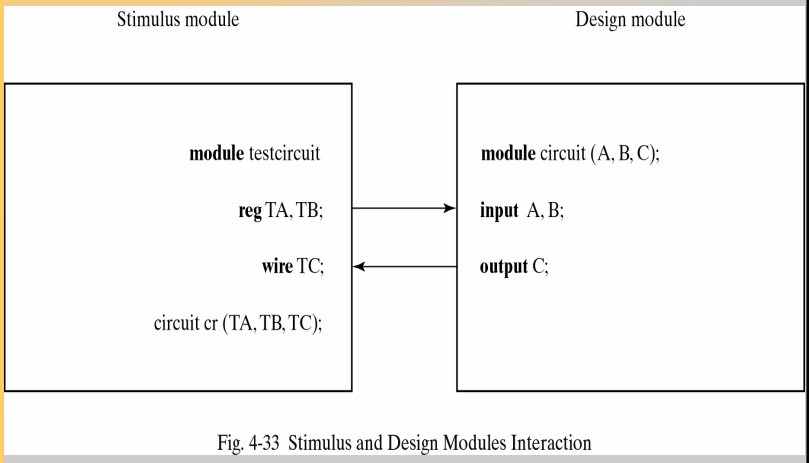


Fig. 4-33 Stimulus and Design Modules Interaction

HDL 範例 4-9
(測試範例4-6所描述的2對1多工器)

```
//Stimulus for mux2x1_df.
module  testmux;
   reg  TA,TB,TS;  //inputs for mux
   wire  Y;       //output from mux
   mux2x1_df mx (TA,TB,TS,Y);
//   instantiate mux
    initial
```

HDL 範例 4-9
(測試範例4-6所描述的2對1多工器)

```
begin
        TS = 1; TA = 0; TB = 1;
  #10 TA = 1; TB = 0;
  #10 TS = 0;
  #10 TA = 0; TB = 1;
end
```

HDL 範例 4-9
(測試範例4-6所描述的2對1多工器)

```
initial
  $monitor("select = %b  A = %b  B = %b
            OUT = %b  time = %0d",
            TS, TA, TB, Y, $time);
endmodule
```

HDL 範例 4-9
(測試範例4-6所描述的2對1多工器)

```
//Dataflow description of 2-to-1-line multiplexer
//from Example 4-6
module  mux2x1_df (A,B,select,OUT);
  input  A,B,select;
  output  OUT;
  assign  OUT = select ? A : B;
endmodule
```

HDL 範例 4-9
(測試範例4-6所描述的2對1多工器)

Simulatiom log:

select = 1 A=0 B=1 OUT=0 time=0

select = 1 A=1 B=0 OUT=1 time=10

select = 0 A=1 B=0 OUT=0 time=20

select = 0 A=0 B=1 OUT=1 time=30

HDL 範例 4-10
(全加法器之多階電路的閘階層描述)

```
//Gate-level description of circuit of Fig. 4-2
module  analysis (A,B,C,F1,F2);
  input   A,B,C;
  output  F1,F2;
  wire    T1,T2,T3,F2not,E1,E2,E3;
  or   g1 (T1,A,B,C);
  and  g2 (T2,A,B,C);
```

HDL 範例 4-10
(全加法器之多階電路的閘階層描述)

```
  and  g3 (E1,A,B);
  and  g4 (E2,A,C);
  and  g5 (E3,B,C);
  or   g6 (F2,E1,E2,E3);
  not  g7 (F2not,F2);
  and  g8 (T3,T1,F2not);
  or   g9 (F1,T2,T3);
endmodule
```

HDL 範例 4-10
(全加法器之多階電路的閘階層描述)

```
//Stimulus to analyze the circuit
 module  test_circuit;
    reg  [2:0]D;
    wire  F1,F2;
    analysis fig42(D[2],D[1],D[0],F1,F2);
    initial
      begin
```

HDL 範例 4-10
(全加法器之多階電路的閘階層描述)

```
        D = 3'b000;
        repeat(7)
          #10 D = D + 1'b1;
        end
     initial
        $monitor ("ABC = %b  F1 = %b  F2 =%b ",
                          D, F1, F2);
endmodule
```

HDL 範例 4-10
(全加法器之多階電路的閘階層描述)

Simulation log:

ABC=000 F1=0 F2=0
ABC=001 F1=1 F2=0
ABC=010 F1=1 F2=0
ABC=011 F1=0 F2=1
ABC=100 F1=1 F2=0
ABC=101 F1=0 F2=1
ABC=110 F1=0 F2=1
ABC=111 F1=1 F2=1