Digital System Design

Lecture 4
Modeling Combinational Circuits in VHDL

Reading Assignment:

- Brown, "Fundamentals of Digital Logic with VHDL", pp. 60 - 65, pp. 318 – 364
- Brown, "Appendix A: VHDL Reference", pp. 791 - 811

Learning Objective:

- Design a VHDL model for a combinational logic circuit using concurrent modeling techniques
  - signal assignments and logical operators
  - conditional signal assignments
  - selected signal assignments
- Design a VHDL model for a combinational logic circuit using sequential modeling techniques
- Design a VHDL model for a combinational logic circuit using a structural design approach

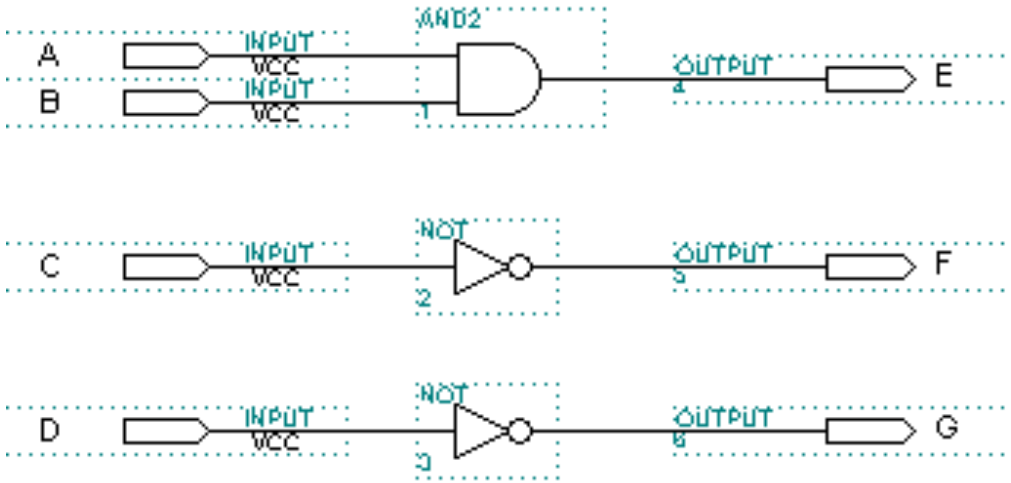
Concurrent Signal Assignments

- A simple concurrent signal assignments are accomplished by using the <= operator after the begin statement in the architecture.

signal_name <= expression ;

- Each individual assignment will be executed concurrently and synthesized as separate logic circuits.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity con_vhdl is
port (A,B,C,D: in STD_LOGIC;
       E,F,G   : out STD_LOGIC);
end con_vhdl;

architecture a of con_vhdl is
begin
  E <= A and B;
  F <= not C;
  G <= not D;
end a;
```



Selected and Conditional Signal Assignments

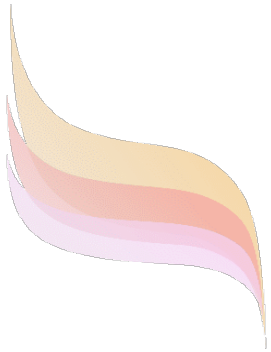
- A selected signal assignment is used to set the value of a signal to one of several alternatives based on a selection criterion. The general form is

```
WITH expression SELECT
    signal_name <= expression WHEN constant_value{,
                   expression WHEN constant_value} ;
```

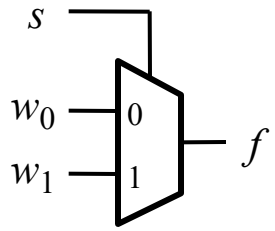
- Similar to the selected signal assignment, the conditional signal assignment is used to set a signal to one of several alternative values. The general form is

```
signal_name <= expression WHEN logic_expression ELSE
               {expression WHEN logic_expression ELSE}
               expression ;
```

Multiplexers



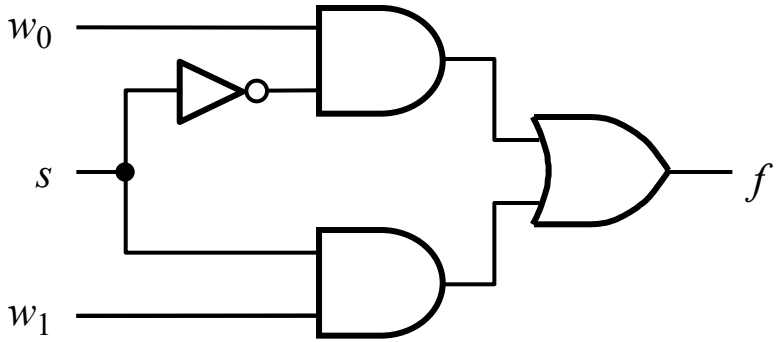
A 2-to-1 multiplexer



(a) Graphical symbol

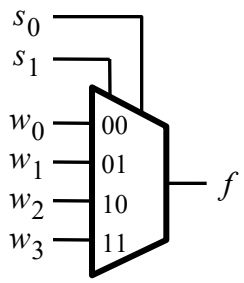
s	f
0	w_0
1	w_1

(b) Truth table



(c) Sum-of-products circuit

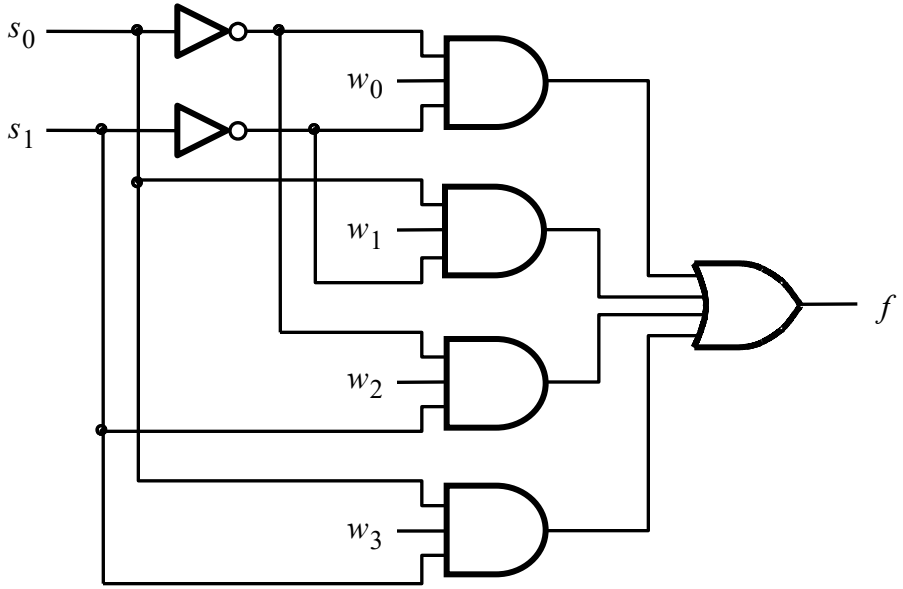
A 4-to-1 multiplexer



(a) Graphic symbol

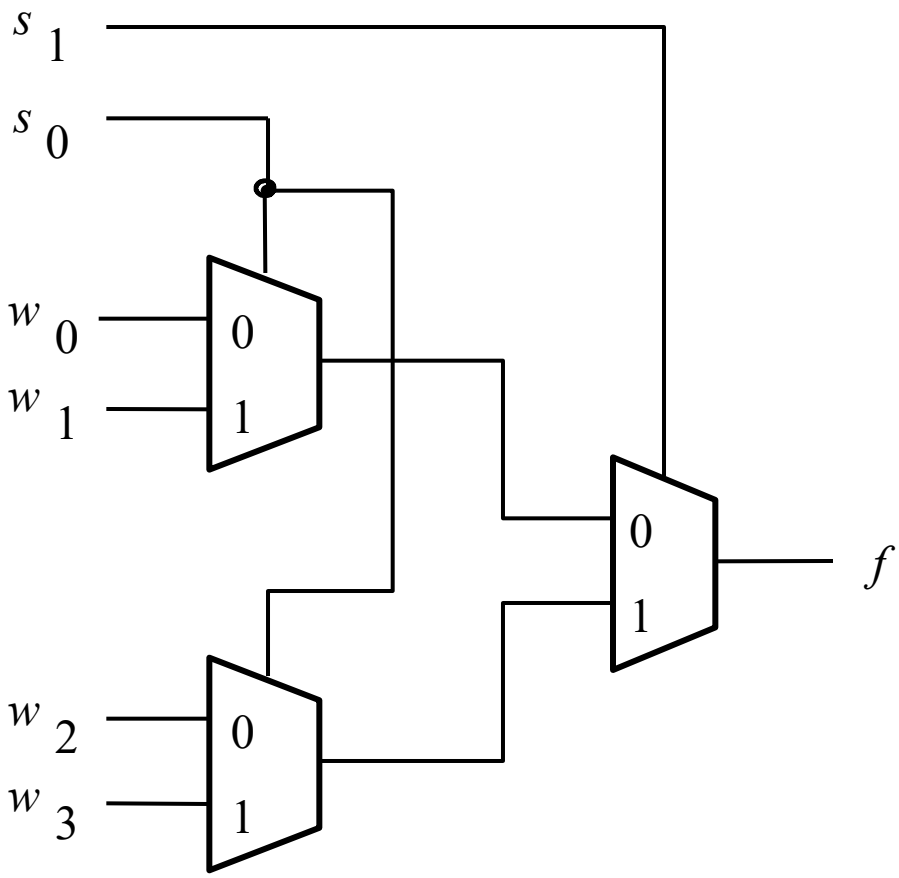
s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

(b) Truth table

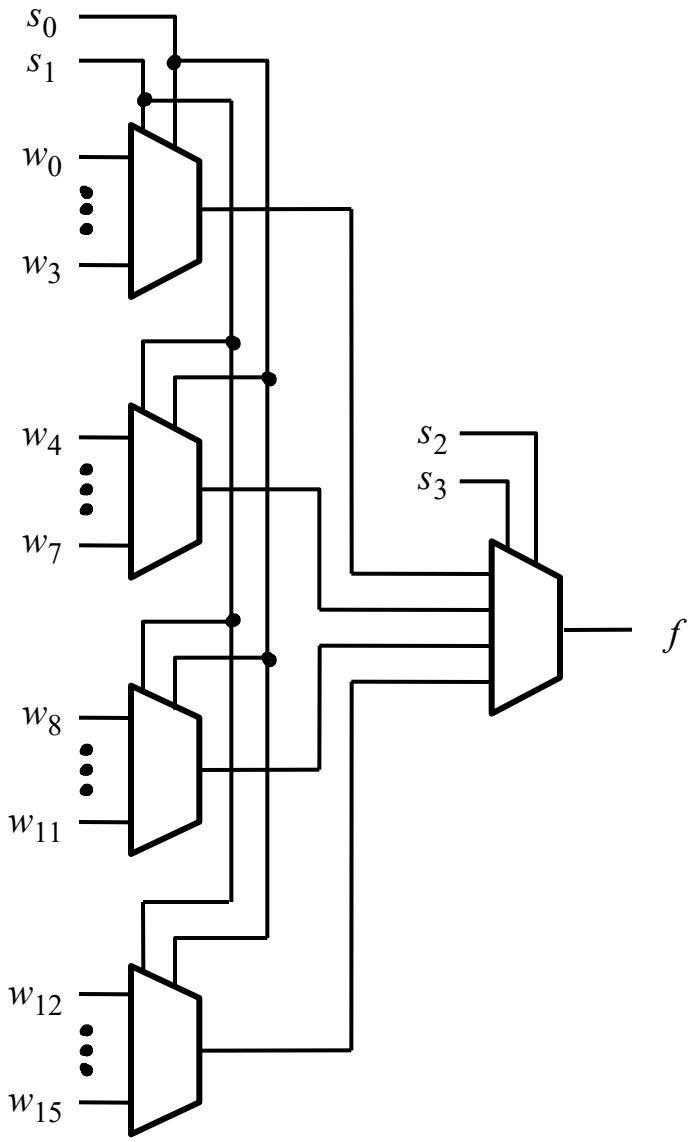


(c) Circuit

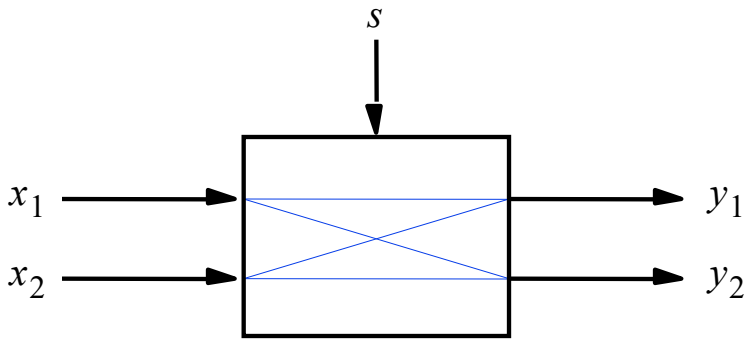
Using 2-to-1 multiplexers to build a 4-to-1 multiplexer



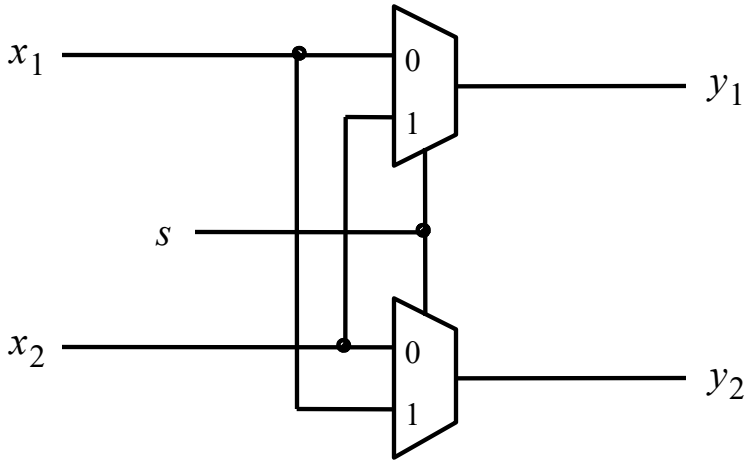
Using 4-to-1 multiplexers to build a 16-to-1 multiplexer



A practical application of multiplexers



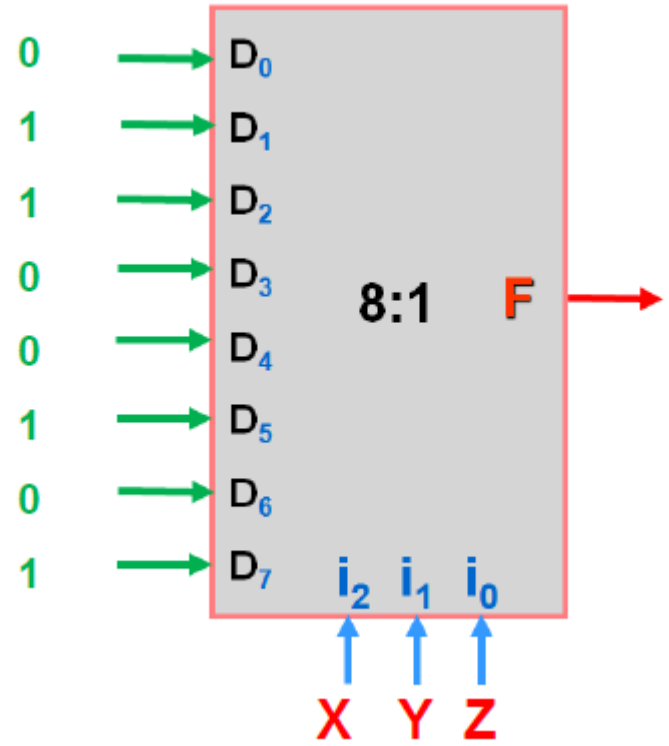
(a) A 2x2 crossbar switch



(b) Implementation using multiplexers

Multiplexer Function Realization

Determine the multiplexer data input values for realizing the function $F(X,Y,Z) = X \cdot Z + X' \cdot (Y \oplus Z)$



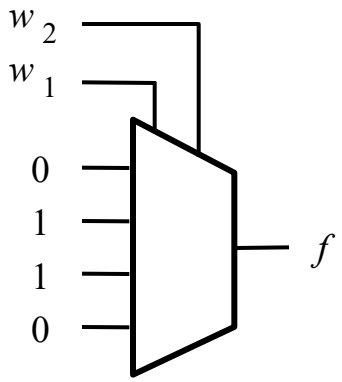
$$F(X,Y,Z) = X \cdot Z + X' \cdot (Y \oplus Z)$$
$$= X \cdot Z + X' \cdot (Y' \cdot Z + Y \cdot Z')$$
$$= X \cdot Z + X' \cdot Y' \cdot Z + X' \cdot Y \cdot Z'$$

X \ YZ	00	01	11	10
0	0	1	0	1
1	0	1	1	0

$$F(X,Y,Z) = \Sigma_{X,Y,Z}(1,2,5,7)$$

Synthesis of a logic function using multiplexers

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

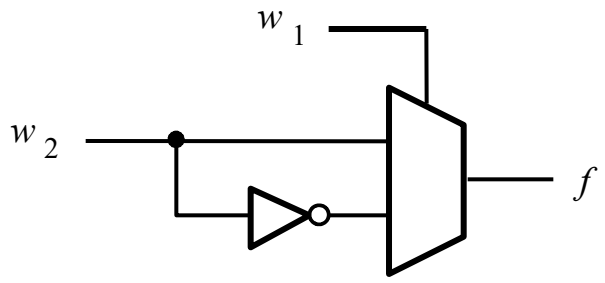


(a) Implementation using a 4-to-1 multiplexer

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

w_1	f
0	w_2
1	$\overline{w}_2$

(b) Modified truth table



(c) Circuit

VHDL code for a 2-to-1 multiplexer

Use selected signal assignment

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux2to1 IS
      PORT (   w0, w1, s       : IN       STD_LOGIC ;
               f               : OUT      STD_LOGIC ) ;
END mux2to1 ;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
      WITH s SELECT
            f <= w0 WHEN '0',
                 w1 WHEN OTHERS ;
END Behavior ;
```

VHDL code for a 4-to-1 multiplexer

Use selected signal assignment

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux4to1 IS
    PORT (  w0, w1, w2, w3    : IN     STD_LOGIC ;
            s                 : IN     STD_LOGIC_VECTOR(1 DOWNTO 0) ;
            f                 : OUT    STD_LOGIC ) ;
END mux4to1 ;

ARCHITECTURE Behavior OF mux4to1 IS
BEGIN
    WITH s SELECT
        f <= w0 WHEN "00",
             w1 WHEN "01",
             w2 WHEN "10",
             w3 WHEN OTHERS ;
END Behavior ;
```

Alternate VHDL code for a 2-to-1 multiplexer

Use conditional signal assignment

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux2to1 IS
        PORT (    w0, w1, s        : IN        STD_LOGIC ;
                  f                : OUT       STD_LOGIC ) ;
END mux2to1 ;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
        f <= w0 WHEN s = '0' ELSE w1 ;
END Behavior ;
```

Alternate VHDL code for a 4-to-1 multiplexer

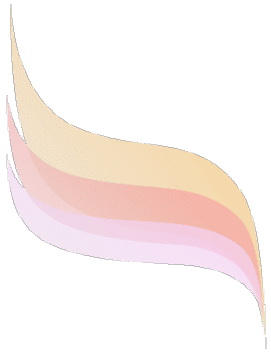
Use conditional signal assignment

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux4to1 IS
    PORT (   w0, w1, w2, w3          : IN      STD_LOGIC ;
                  s          : IN      STD_LOGIC_VECTOR(1 DOWNTO 0) ;
                  f          : OUT     STD_LOGIC ) ;
END mux4to1 ;

ARCHITECTURE Behavior OF mux4to1 IS
BEGIN
    f <= w0 WHEN s = "00" ELSE
         w1 WHEN s = "01" ELSE
         w2 WHEN s = "10" ELSE
         w3;
END Behavior ;
```

VHDL code for a 16-to-1 multiplexer

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux16to1 IS
        PORT (     w      : IN        STD_LOGIC_VECTOR(0 TO 15) ;
                   s      : IN        STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                   f      : OUT       STD_LOGIC ) ;
END mux16to1 ;

ARCHITECTURE Structure OF mux16to1 IS
        SIGNAL m : STD_LOGIC_VECTOR(0 TO 3) ;
       COMPONENT mux4to1
                   PORT (     w0, w1, w2, w3   : IN        STD_LOGIC ;
                              s                : IN        STD_LOGIC_VECTOR(1 DOWNTO 0) ;
                              f                : OUT       STD_LOGIC ) ;
        END COMPONENT ;

BEGIN
        Mux1: mux4to1 PORT MAP ( w(0), w(1), w(2), w(3), s(1 DOWNTO 0), m(0) ) ;
        Mux2: mux4to1 PORT MAP ( w(4), w(5), w(6), w(7), s(1 DOWNTO 0), m(1) ) ;
        Mux3: mux4to1 PORT MAP ( w(8), w(9), w(10), w(11), s(1 DOWNTO 0), m(2) ) ;
        Mux4: mux4to1 PORT MAP ( w(12), w(13), w(14), w(15), s(1 DOWNTO 0), m(3) ) ;
        Mux5: mux4to1 PORT MAP ( m(0), m(1), m(2), m(3), s(3 DOWNTO 2), f ) ;
END Structure ;
```

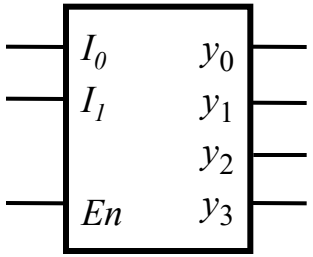
Decoders/Encoders



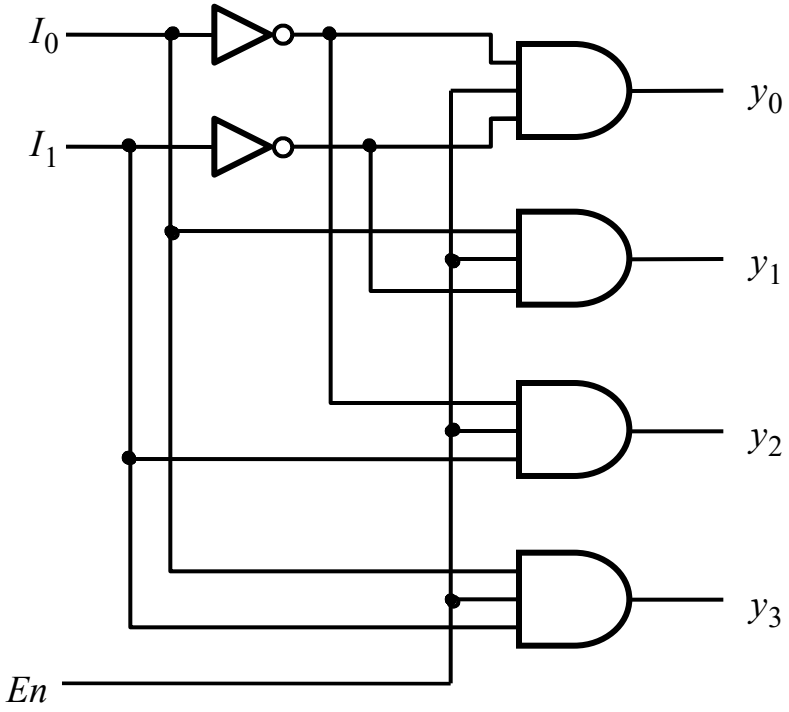
A 2-to-4 decoder

En	I_1	I_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

(a) Truth table

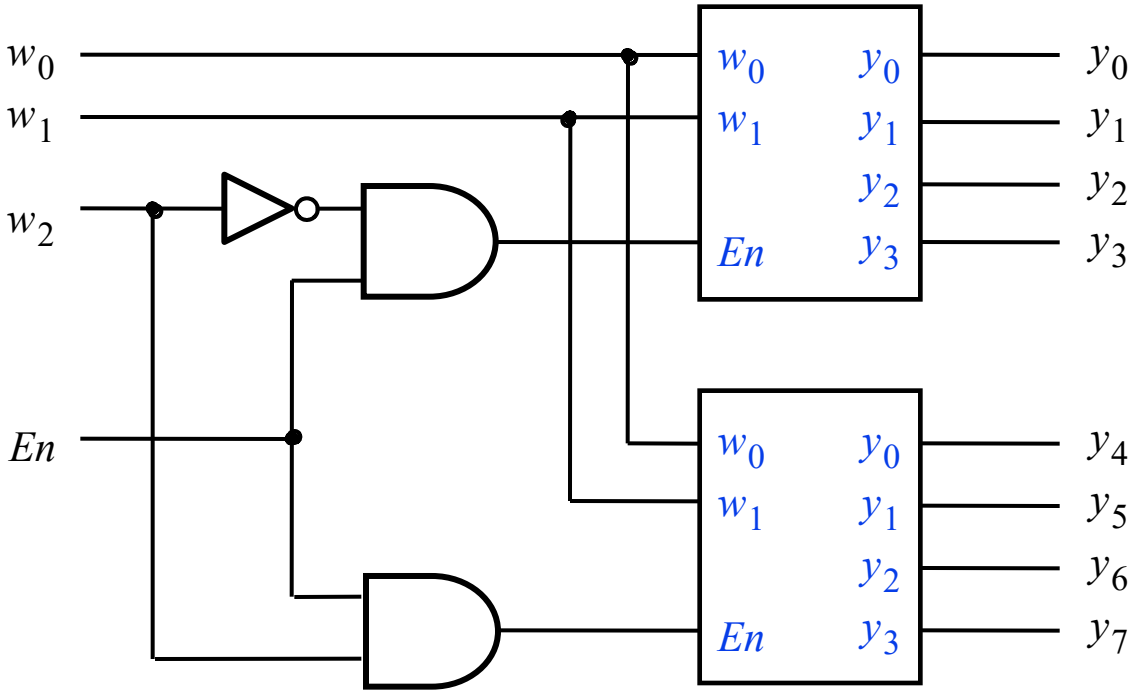


(b) Graphic symbol

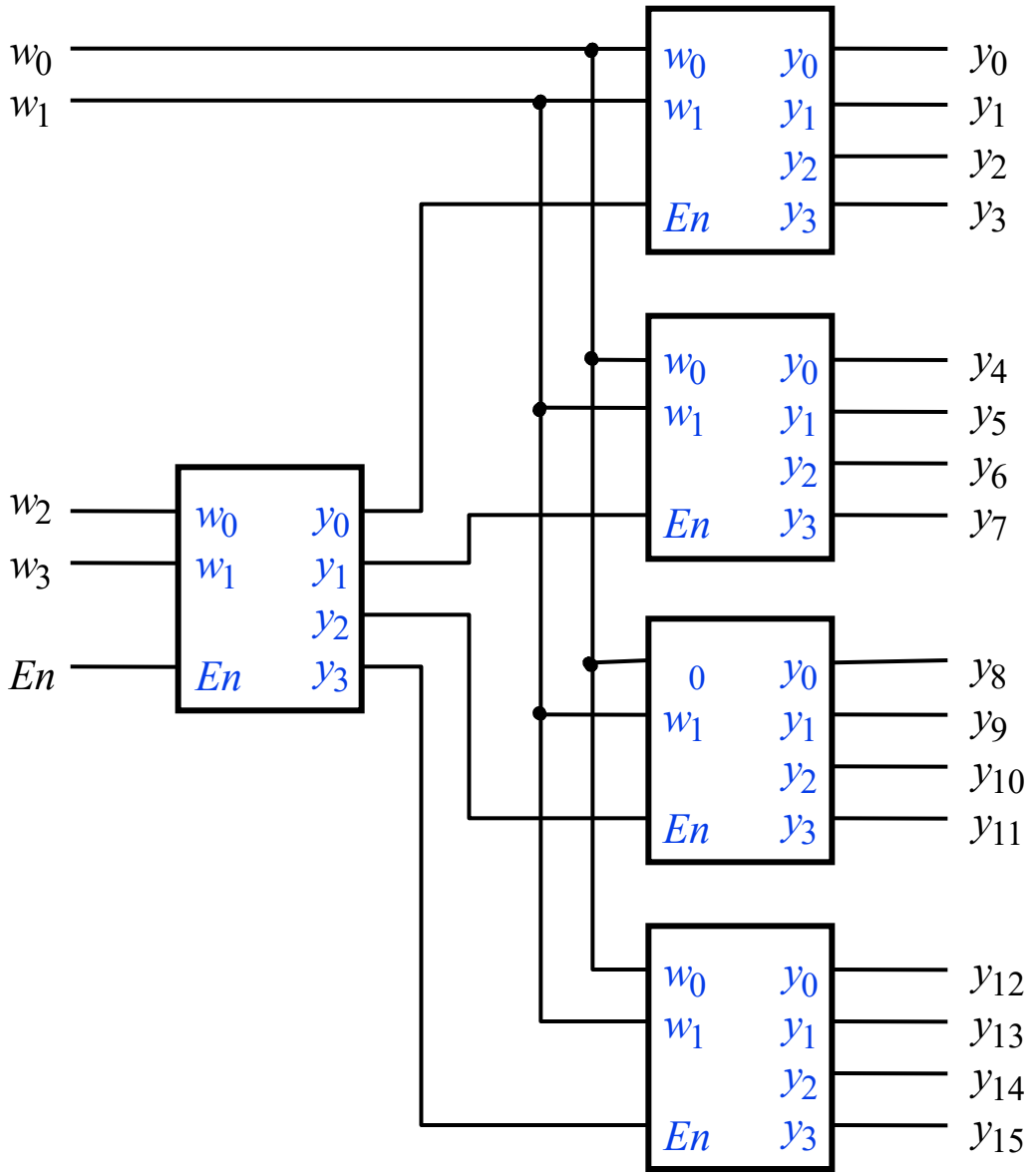


(c) Logic circuit

A 3-to-8 decoder using two 2-to-4 decoders



A 4-to-16 decoder built using a decoder tree



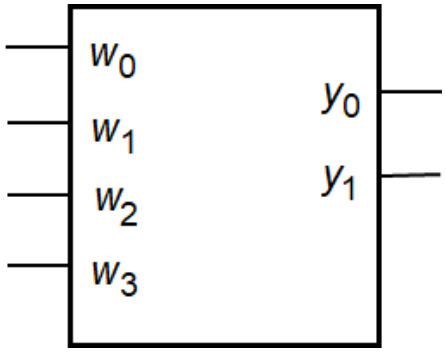
VHDL code for a 2-to-4 binary decoder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY dec2to4 IS
      PORT (   w     : IN      STD_LOGIC_VECTOR(1 DOWNTO 0) ;
               En    : IN      STD_LOGIC ;
               y     : OUT     STD_LOGIC_VECTOR(0 TO 3) ) ;
END dec2to4 ;

ARCHITECTURE Behavior OF dec2to4 IS
      SIGNAL Enw : STD_LOGIC_VECTOR(2 DOWNTO 0) ;
BEGIN
      Enw <= En & w ;
      WITH Enw SELECT
               y <= "1000" WHEN "100",
                    "0100" WHEN "101",
                    "0010" WHEN "110",
                    "0001" WHEN "111",
                    "0000" WHEN OTHERS ;
END Behavior ;
```

Alternate VHDL code for a 2-to-4 binary decoder

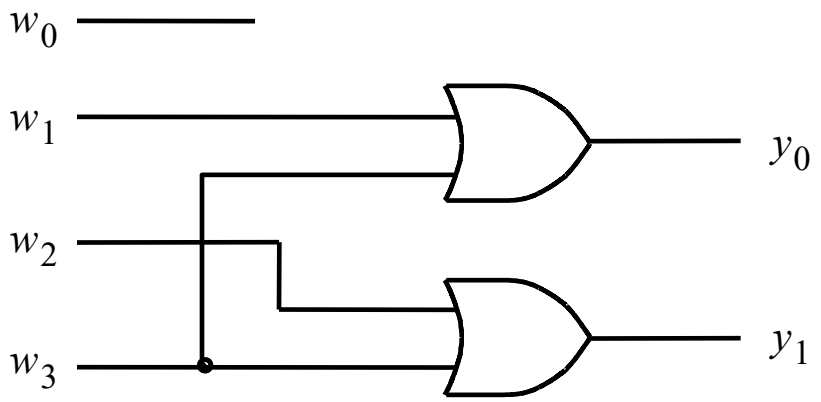
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY dec2to4 IS
      PORT (   w     : IN      STD_LOGIC_VECTOR(1 DOWNTO 0) ;
               En    : IN      STD_LOGIC ;
               y     : OUT     STD_LOGIC_VECTOR(0 TO 3) ) ;
END dec2to4 ;

ARCHITECTURE Behavior OF dec2to4 IS
      SIGNAL Enw : STD_LOGIC_VECTOR(2 DOWNTO 0) ;
BEGIN
      Enw <= En & w ;
      y <= "1000" WHEN  Enw = "100"   ELSE
           "0100" WHEN  Enw = "101"   ELSE
           "0010" WHEN  Enw = "110"   ELSE
           "0001" WHEN  Enw = "111"   ELSE
           "0000" ;
END Behavior ;
```

A 4-to-2 binary encoder



w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

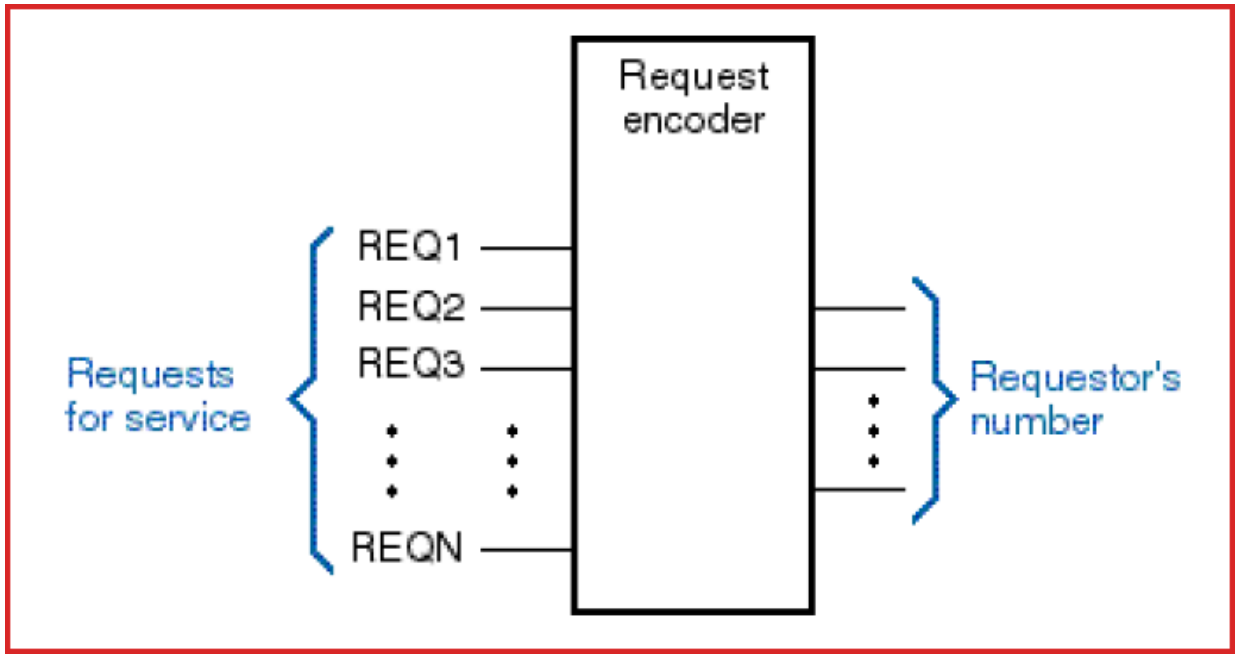
(a) Truth table



(b) Circuit

Priority Encoders

- A common application is to encode the number of a device requesting service from a microprocessor-based system



Problem: More than one device may be requesting service at any given time

Priority Encoders

- **Solution**: Assign ***priority*** to the input lines, such that when multiple inputs are asserted simultaneously, the ***highest priority*** (i.e. ***highest numbered***) input "wins" – such a device is called a ***priority encoder***

Truth table for a 4-to-2 priority encoder

w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

$$i_0 = \overline{w}_3\overline{w}_2\overline{w}_1 w_0$$
$$i_1 = \overline{w}_3\overline{w}_2 w_1$$
$$i_2 = \overline{w}_3 w_2$$
$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$
$$y1 = i_2 + i_3$$
$$z = i_0 + i_1 + i_2 + i_3$$

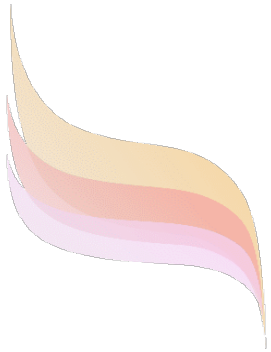
Less Efficient code for a priority encoder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY priority IS
      PORT (   w : IN      STD_LOGIC_VECTOR(3 DOWNTO 0) ;
               y : OUT     STD_LOGIC_VECTOR(1 DOWNTO 0) ;
               z : OUT     STD_LOGIC ) ;
END priority ;

ARCHITECTURE Behavior OF priority IS
BEGIN
      WITH w SELECT
            y <=   "00" WHEN "0001",
                   "01" WHEN "0010",
                   "01" WHEN "0011",
                   "10" WHEN "0100",
                   "10" WHEN "0101",
                   "10" WHEN "0110",
                   "10" WHEN "0111",
                   "11" WHEN OTHERS ;
      WITH w SELECT
            z <=   '0' WHEN "0000",
                   '1' WHEN OTHERS ;
END Behavior ;
```

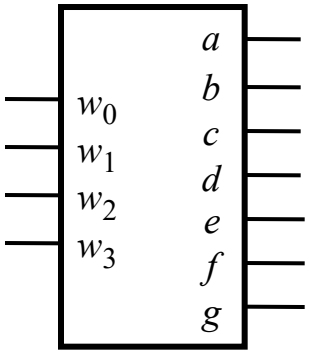
Better VHDL code for a priority encoder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY priority IS
    PORT (  w   : IN    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            y   : OUT   STD_LOGIC_VECTOR(1 DOWNTO 0) ;
            z   : OUT   STD_LOGIC ) ;
END priority ;

ARCHITECTURE Behavior OF priority IS
BEGIN
    y <=    "11" WHEN w(3) = '1' ELSE
            "10" WHEN w(2) = '1' ELSE
            "01" WHEN w(1) = '1' ELSE
            "00" ;
    z <= '0' WHEN w = "0000" ELSE '1' ;
END Behavior ;
```

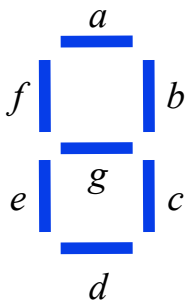
Other Combinational Circuits



A BCD-to-7-segment display code converter



(a) Code converter

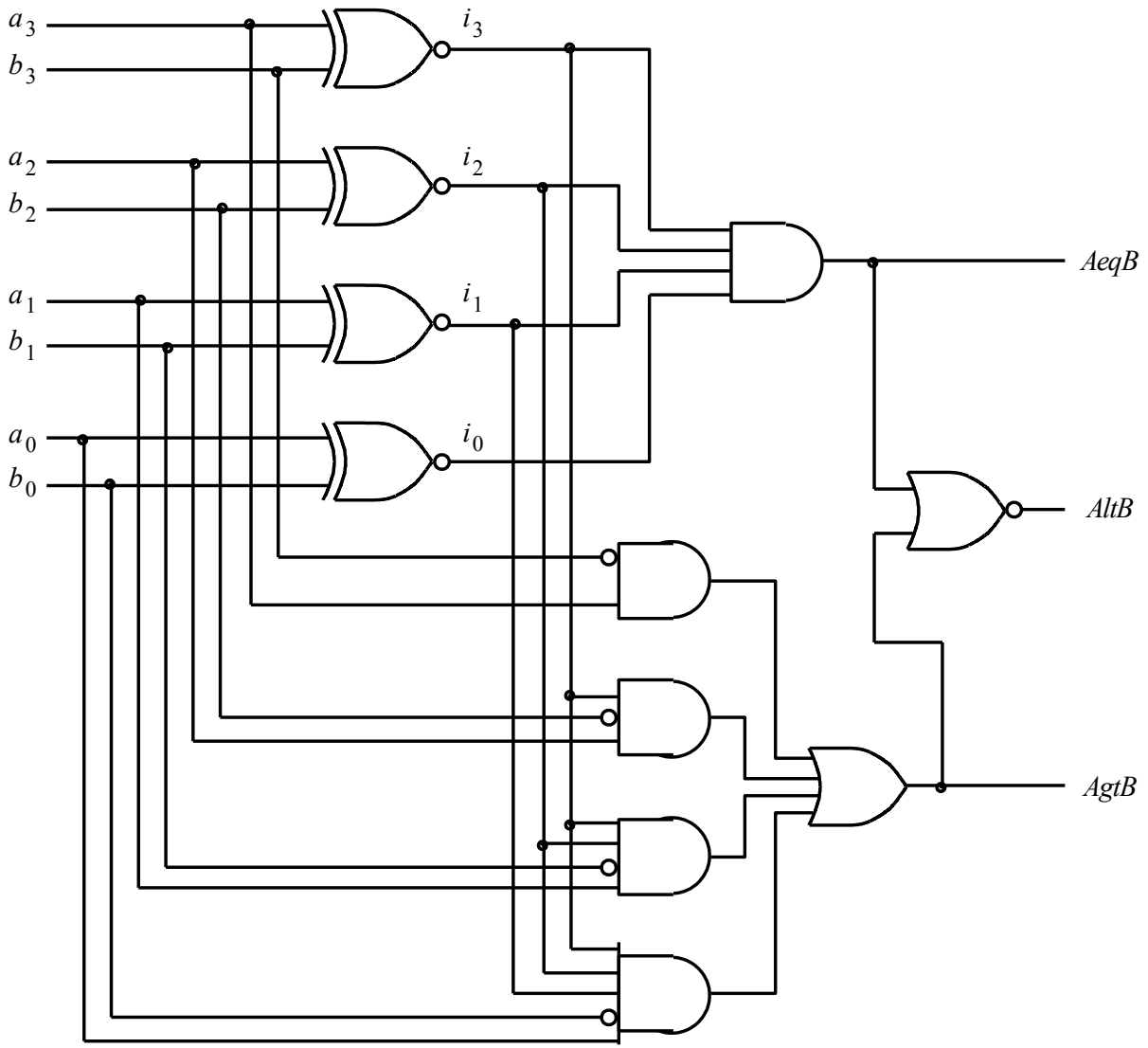


(b) 7-segment display

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

(c) Truth table

Comparison Circuits



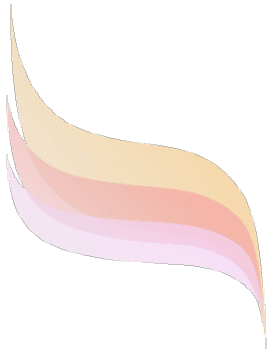
VHDL code for a 4-bit comparator

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY compare IS
      PORT (   A, B                  : IN     STD_LOGIC_VECTOR(3 DOWNTO 0) ;
               AeqB, AgtB, AltB      : OUT    STD_LOGIC ) ;
END compare ;

ARCHITECTURE Behavior OF compare IS
BEGIN
      AeqB <= '1' WHEN A = B ELSE '0' ;
      AgtB <= '1' WHEN A > B ELSE '0' ;
      AltB <= '1' WHEN A < B ELSE '0' ;
END Behavior ;
```

VHDL code for a 4-bit comparator

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_arith.all ;

ENTITY compare IS
      PORT (   A, B                  : IN     SIGNED(3 DOWNTO 0) ;
               AeqB, AgtB, AltB      : OUT    STD_LOGIC ) ;
END compare ;

ARCHITECTURE Behavior OF compare IS
BEGIN
      AeqB <= '1' WHEN A = B ELSE '0' ;
      AgtB <= '1' WHEN A > B ELSE '0' ;
      AltB <= '1' WHEN A < B ELSE '0' ;
END Behavior ;
```

Process Statements



Concurrent vs. Sequential

- All previous statements are called *concurrent assignment statements* because order does not matter.
- When order matters, the statements are called *sequential assignment statements*.
- All sequential assignment statements are placed within a *process statement*.

Process Statement

- Begins with PROCESS keyword followed by a *sensitivity list*.
- For a combinational circuit, sensitivity list includes all input signals used in the process.
- Process executed whenever there is a change on a signal in the sensitivity list.
- Statements executed in sequential order.
- No assignments are visible until all statements in the process have been executed.
- If multiple assignments, only last has an effect.

A 2-to-1 multiplexer specified using if-then-else

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux2to1 IS
        PORT (      w0, w1, s       : IN        STD_LOGIC ;
                    f               : OUT       STD_LOGIC ) ;
END mux2to1 ;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
        PROCESS ( w0, w1, s )
        BEGIN
                IF s = '0' THEN
                        f <= w0 ;
                ELSE
                        f <= w1 ;
                END IF ;
        END PROCESS ;
END Behavior ;
```

Alternative code for a 2-to-1 multiplexer

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux2to1 IS
        PORT (      w0, w1, s          : IN        STD_LOGIC ;
                    f                  : OUT       STD_LOGIC ) ;
END mux2to1 ;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
        PROCESS ( w0, w1, s )
        BEGIN
                f <= w0 ;
                IF s = '1' THEN
                        f <= w1 ;
                END IF ;
        END PROCESS ;
END Behavior ;
```

要 s 改變才會 改變 f 的值 ($w_0 \to w_1$, $w_1 \to w_0$) → 形成 multiplexer

↳ 記憶性

VHDL code with implied Memory

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY implied IS
        PORT (      A, B        : IN        STD_LOGIC ;
                    AeqB        : OUT       STD_LOGIC ) ;
END implied ;

ARCHITECTURE Behavior OF implied IS
BEGIN
        PROCESS ( A, B )
        BEGIN
                IF A = B THEN
                        AeqB <= '1' ;
                END IF ;
        END PROCESS ;
END Behavior ;
```

A priority encoder specified using if-then-else

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY priority IS
    PORT (  w   : IN    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            y   : OUT   STD_LOGIC_VECTOR(1 DOWNTO 0) ;
            z   : OUT   STD_LOGIC ) ;
END priority ;

ARCHITECTURE Behavior OF priority IS
BEGIN
    PROCESS ( w )
    BEGIN
        IF w(3) = '1' THEN
            y <= "11" ;
        ELSIF w(2) = '1' THEN
            y <= "10" ;
        ELSIF w(1) = '1' THEN
            y <= "01" ;
        ELSE
            y <= "00" ;
        END IF ;
    END PROCESS ;
    z <= '0' WHEN w = "0000" ELSE '1' ;
END Behavior ;
```

Alternative code for the priority encoder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY priority IS
    PORT (  w   : IN    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            y   : OUT   STD_LOGIC_VECTOR(1 DOWNTO 0) ;
            z   : OUT   STD_LOGIC ) ;
END priority ;

ARCHITECTURE Behavior OF priority IS
BEGIN
    PROCESS ( w )
    BEGIN
        y <= "00" ;
        IF w(1) = '1' THEN y <= "01" ; END IF ;
        IF w(2) = '1' THEN y <= "10" ; END IF ;
        IF w(3) = '1' THEN y <= "11" ; END IF ;

        z <= '1' ;
        IF w = "0000" THEN z <= '0' ; END IF ;
    END PROCESS ;
END Behavior ;
```

A 2-to-1 multiplexer specified using a CASE statement

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux2to1 IS
        PORT (      w0, w1, s       : IN      STD_LOGIC ;
                    f               : OUT     STD_LOGIC ) ;
END mux2to1 ;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
        PROCESS ( w0, w1, s )
        BEGIN
                CASE s IS
                        WHEN '0' =>
                                f <= w0 ;
                        WHEN OTHERS =>
                                f <= w1 ;
                END CASE ;
        END PROCESS ;
END Behavior ;
```

A 2-to-4 binary decoder specified using a CASE statement

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY dec2to4 IS
    PORT (   w    : IN     STD_LOGIC_VECTOR(1 DOWNTO 0) ;
             En   : IN     STD_LOGIC ;
             y    : OUT    STD_LOGIC_VECTOR(0 TO 3) ) ;
END dec2to4 ;

ARCHITECTURE Behavior OF dec2to4 IS
BEGIN
    PROCESS ( w, En )
    BEGIN
        IF En = '1' THEN
            CASE w IS
                WHEN "00" =>        y <= "1000" ;
                WHEN "01" =>        y <= "0100" ;
                WHEN "10" =>        y <= "0010" ;
                WHEN OTHERS =>  y <= "0001" ;
            END CASE ;
        ELSE
            y <= "0000" ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

A BCD-to-7-segment decoder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY seg7 IS
        PORT (      bcd      : IN        STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                    leds     : OUT       STD_LOGIC_VECTOR(1 TO 7) ) ;
END seg7 ;
ARCHITECTURE Behavior OF seg7 IS
BEGIN
        PROCESS ( bcd )
        BEGIN
                CASE bcd IS   --                                abcdefg
                        WHEN "0000"       => leds   <=    "1111110" ;
                        WHEN "0001"       => leds   <=    "0110000" ;
                        WHEN "0010"       => leds   <=    "1101101" ;
                        WHEN "0011"       => leds   <=    "1111001" ;
                        WHEN "0100"       => leds   <=    "0110011" ;
                        WHEN "0101"       => leds   <=    "1011011" ;
                        WHEN "0110"       => leds   <=    "1011111" ;
                        WHEN "0111"       => leds   <=    "1110000" ;
                        WHEN "1000"       => leds   <=     "1111111" ;
                        WHEN "1001"       => leds   <=     "1110011" ;
                        WHEN OTHERS       => leds   <=     "-------" ;
                END CASE ;
        END PROCESS ;
END Behavior ;
```

The functionality of the 74381 ALU

Operation	Inputs s_2 s_1 s_0	Outputs F
Clear	0 0 0	0 0 0 0
B−A	0 0 1	$B - A$
A−B	0 1 0	$A - B$
ADD	0 1 1	$A + B$
XOR	1 0 0	A XOR B
OR	1 0 1	A OR B
AND	1 1 0	A AND B
Preset	1 1 1	1 1 1 1

Code that represents the functionality of the 74381 ALU

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;
ENTITY alu IS
      PORT (    s       : IN      STD_LOGIC_VECTOR(2 DOWNTO 0) ;
                A, B    : IN      STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                F       : OUT     STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END alu ;

ARCHITECTURE Behavior OF alu IS
BEGIN
      PROCESS ( s, A, B )
      BEGIN
            CASE s IS
                  WHEN "000" => F <= "0000" ;
                  WHEN "001" => F <= B - A ;
                  WHEN "010" => F <= A - B ;
                  WHEN "011" => F <= A + B ;
                  WHEN "100" => F <= A XOR B ;
                  WHEN "101" => F <= A OR B ;
                  WHEN "110" => F <= A AND B ;
                  WHEN OTHERS =>
                        F <= "1111" ;
            END CASE ;
      END PROCESS ;
END Behavior ;
```