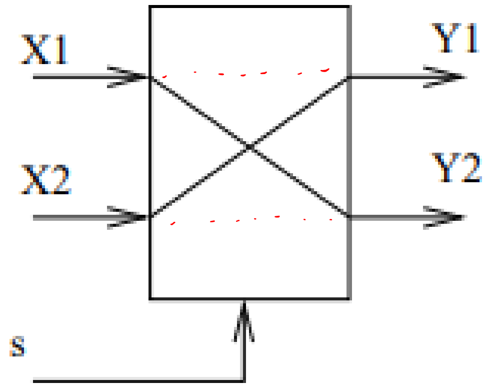
COE328
Digital Electronics

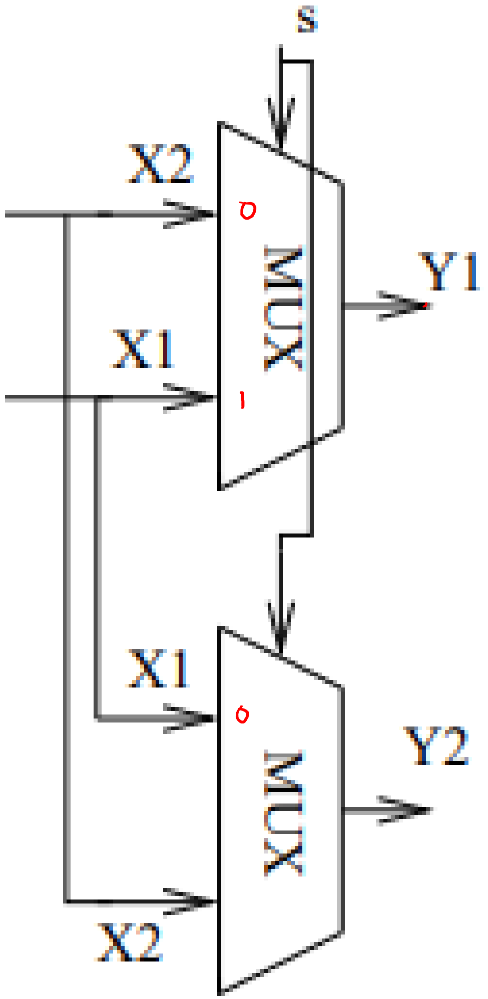
Lecture 11

Dr. Shazzat Hossain

Multiplexer Application: Crossbar switch



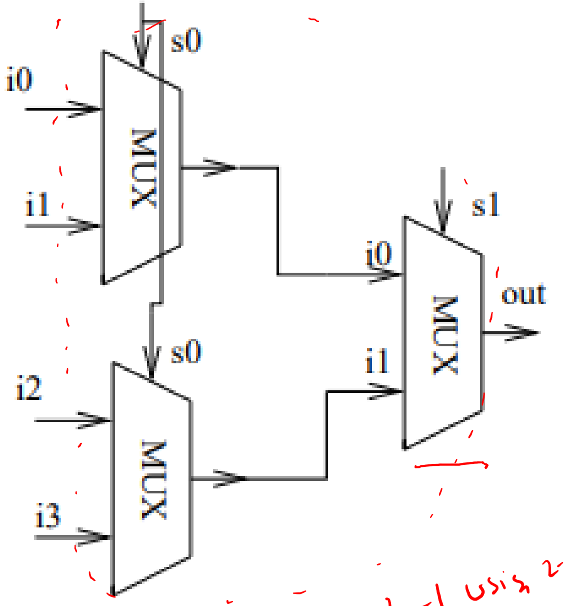
If s=0 then Y1=X2, Y2=X1
If s=1 then Y1=X1, Y2=X2



Multiplexer Extension

Implement a 4-to-1 MUX using 2-to-1 MUX

inputs o/p



4to1 MUX

No. of multiplexers

1st stage $= \frac{4}{2} = 2$

2nd stage $= \frac{2}{2} = 1$

8-to-1 using 2-to-1

1st $\frac{8}{2} = 4$

2nd $4/2 = 2$

8-to-1 4 to-1

1st $8/4 = 2$

2nd $2/4 = 0.5$

1st stage $= \frac{16}{4} = 4$

2nd stage $= \frac{4}{4} = 1$

16-to-1 using 2 to-1

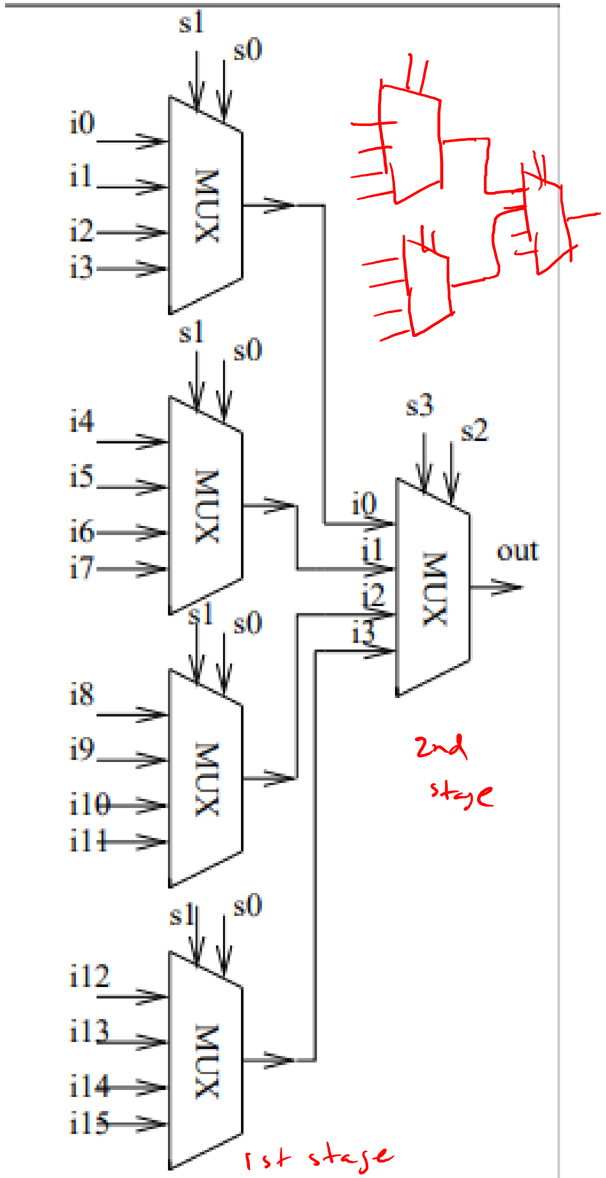
1st stage $= \frac{16}{2} = 8$

2nd " $= \frac{8}{2} = 4$

3rd stage: $\frac{4}{2} = 2$

4th stg $= \frac{2}{2} = 1$

Implement a 16-to-1 MUX using 4-to-1 MUX



16to1 MUX

Function Implementation using LUT

2 I/P LUT

Implement $f = \bar{w}_1\bar{w}_3 + w_1 w_3 + w_1 w_2$

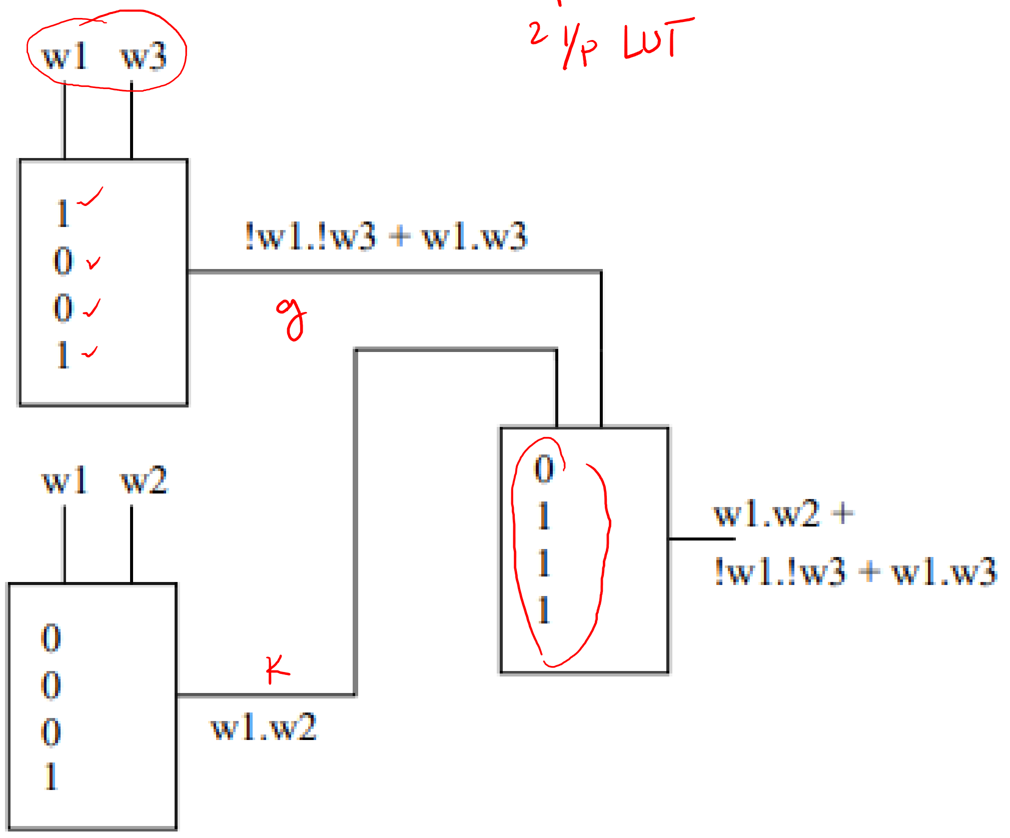
g_1 g_2 g_3

$g + K$

LUT for $g = \bar{w}_1\bar{w}_3 + w_1 w_3$

w_1	W_3	$\bar{w}_1\bar{w}_3 + w_1 w_3$
0	0	1
0	1	0
1	0	0
1	1	1

$w_1 = 0$
$w_3 = 1$



LUT for $k = w_1 w_2$

w_1	W_2	$w_1 w_2$
0	0	~~1~~ 0
0	1	0
1	0	0
1	1	1

g	k	f
0	0	0
0	1	1
1	0	1
1	1	1

LUT for $f = g + k$

0
1
1
1

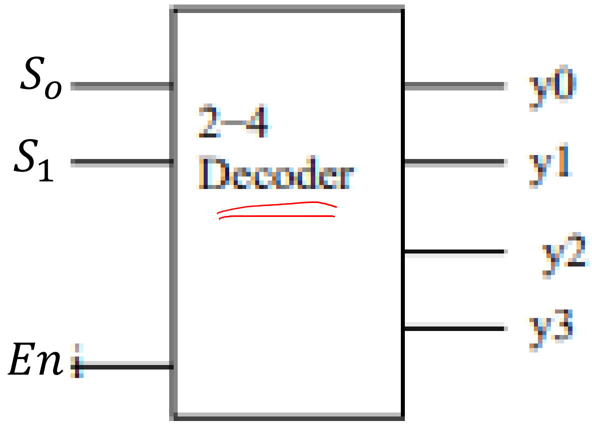
Demultiplexer

Multiple outputs from a single input, selected by select line(s).
Could use a decoder, with En=input.

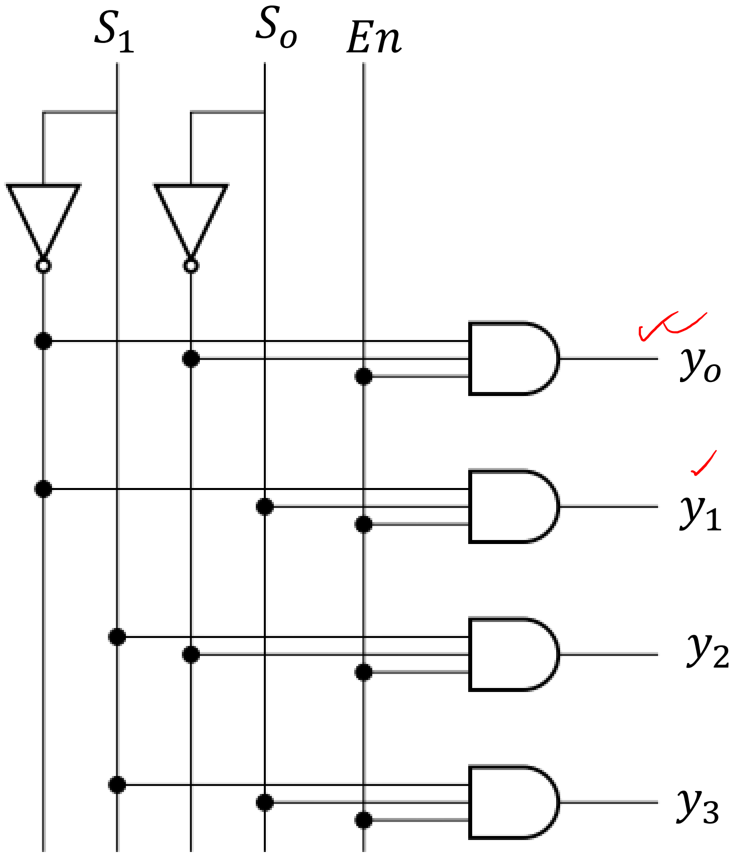
$n \rightarrow$ Select Lines
$2^n \rightarrow$ I/p
$1 \rightarrow$ o/p $\rightarrow$

DEMUX
$n \leftarrow$ select lines
$1 \longrightarrow$ I/p
$2^n \rightarrow$ o/p

S_1	S_o	Output
0	0	y_o
0	1	y_1
1	0	y_2
1	1	y_3



DeMultiplexer



Encoder

From n inputs, one is active and the output is the code for the specific input.

Example: Binary encoder

Priority encoder:

Postal code

M4B 2K3

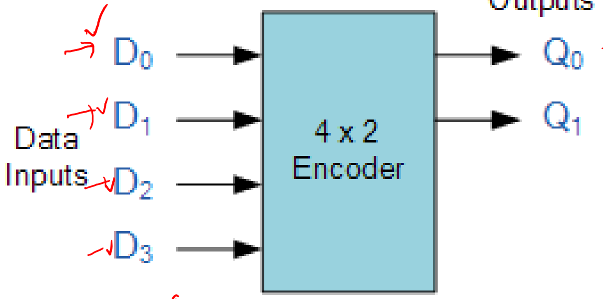
code

$A \rightarrow$ ASCII

$9 \rightarrow$ BCD

n

2^n



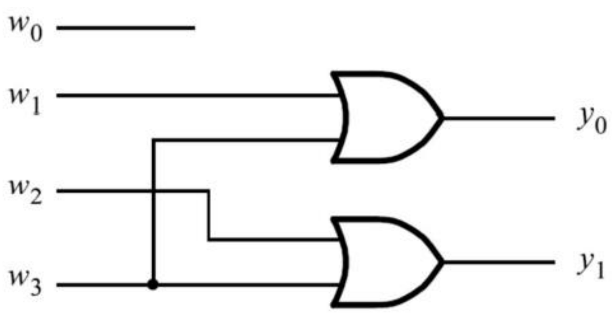
0

7

Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_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
0	0	0	0	x	x

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

Figure 6.23 A 4-to-2 binary encoder.

Inputs				Output	
D_3	D_2	D_1	D_0	B	A
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

$$A = w_3 + \overline{w}_3\overline{w}_2 w_1$$
$$B = w_3 + \overline{w}_3 w_2$$

Encoder

Hexadecimal to Binary encoder: Coverts a hexadecimal number into a four-bit binary

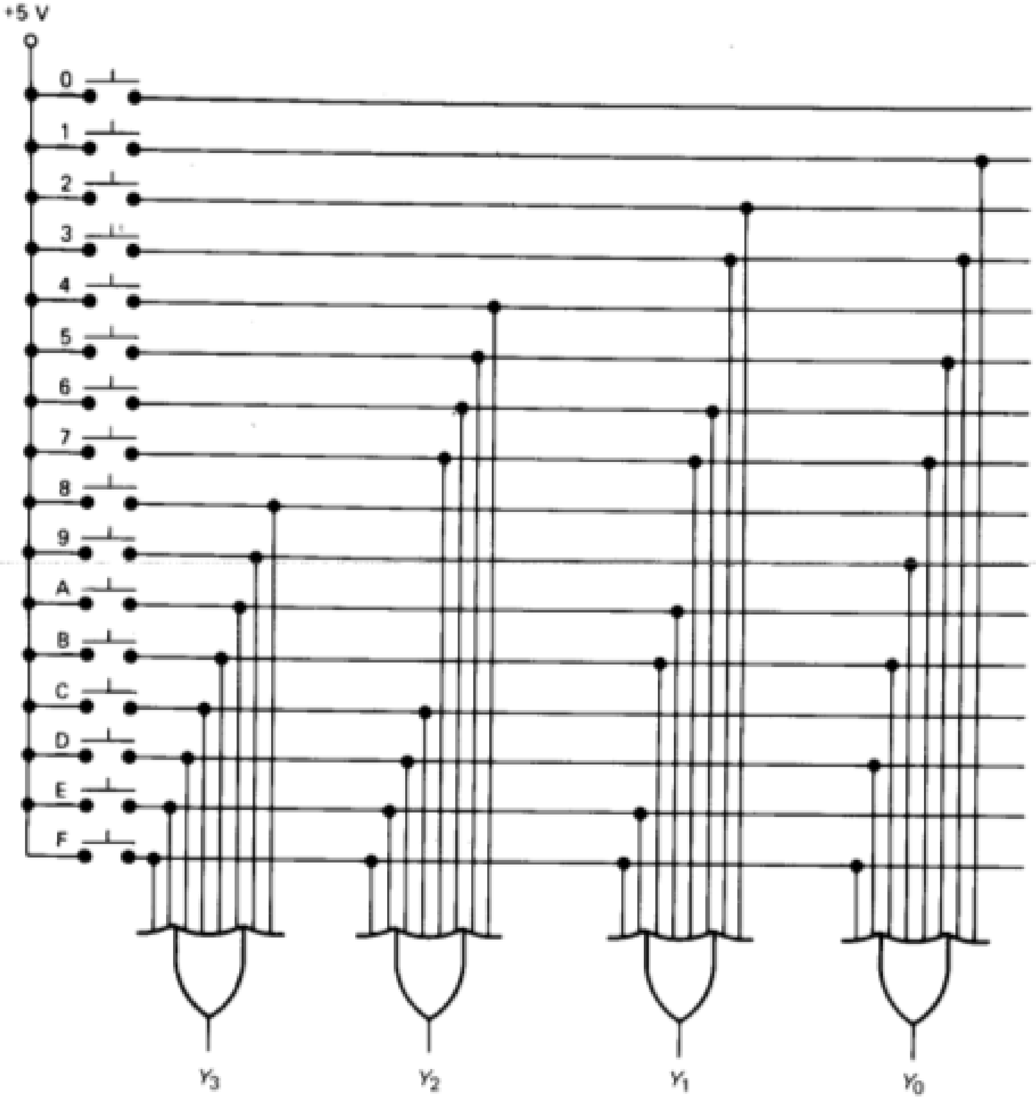


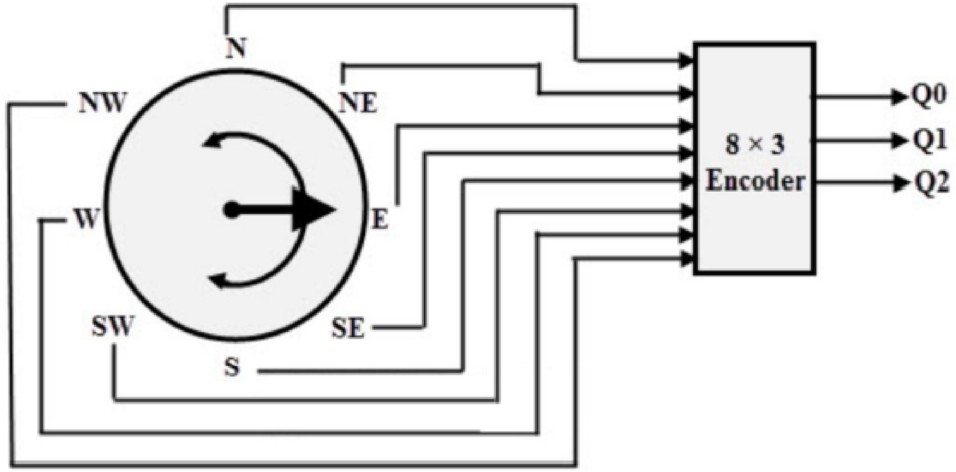
Fig. 2-18 Hexadecimal encoder.

Application of Encoder

Encoder

Positional Encoders (Source: electronicshub.org)

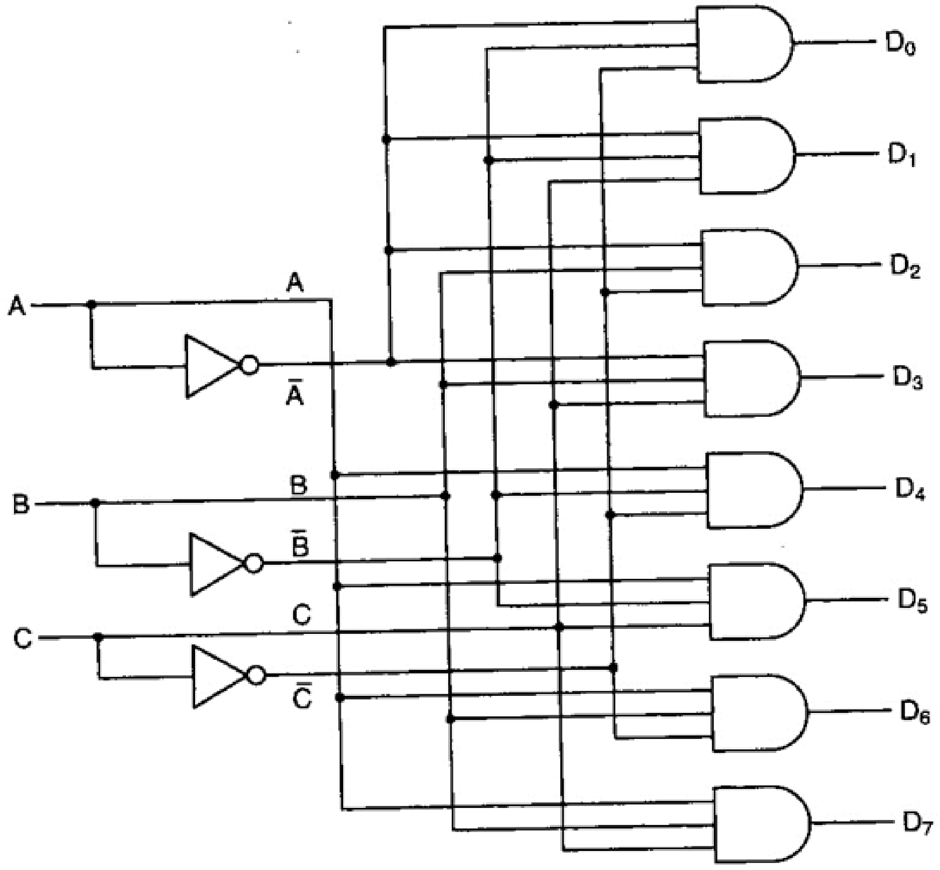
A magnetic positional control is another common application of priority encoders. Such control is used in robotic arm positioning and ship navigations. In such cases, encoder converts the rotary or angular position of a compass to a digital code. Then this code is input to the computer so that the navigational data is provided. Below figure shows the simple compass encoder that converts the 8 positions to 3 bit output. For this type of input –output configuration, a 74LS148 IC is used which is an 8-to-3 line priority encoder. For indicating the compasses angular position, generally reed switches and magnets are used. application of priority encoder block diagram application of priority encoder

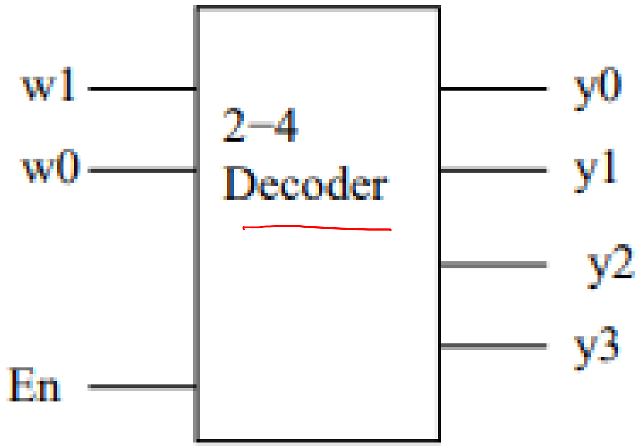


Compass Direction	Binary Output		
	Q_0	Q_1	Q_2
North	0	0	0
North - East	0	0	1
East	0	1	0
South - East	0	1	1
South	1	0	0
South - West	1	0	1
West	1	1	0
North - West	1	1	1

Decoders → important

N inputs generate 2n outputs, with only one =true based on value of n.
For 2 inputs, we have 4 outputs.

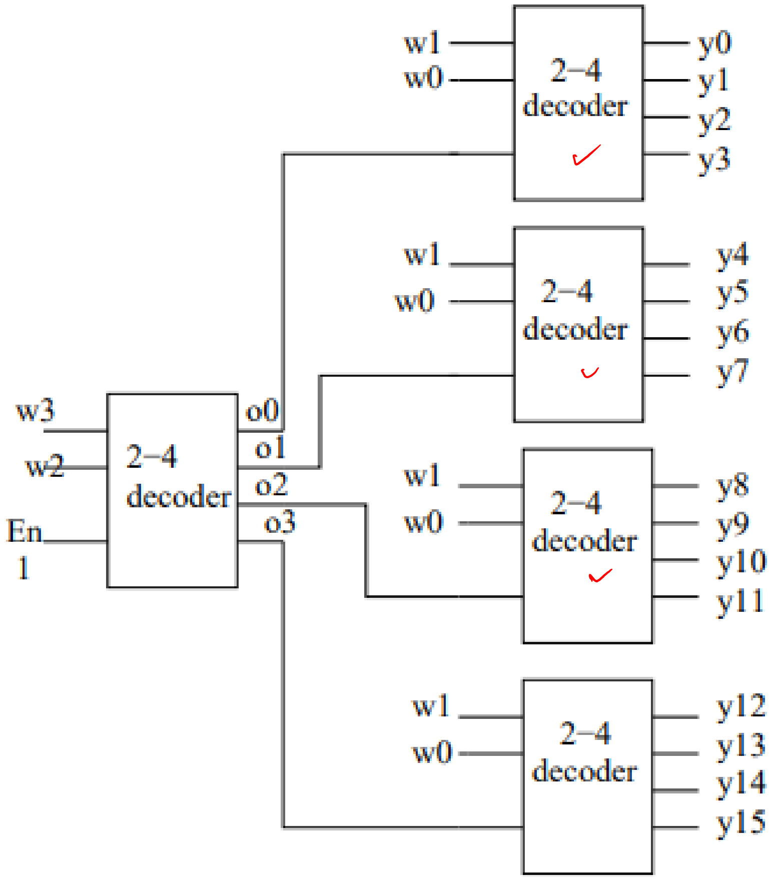




Symbol

Decoder Expansion

Design a 16-output decoder using 4 output decoders



Decoder Expansion

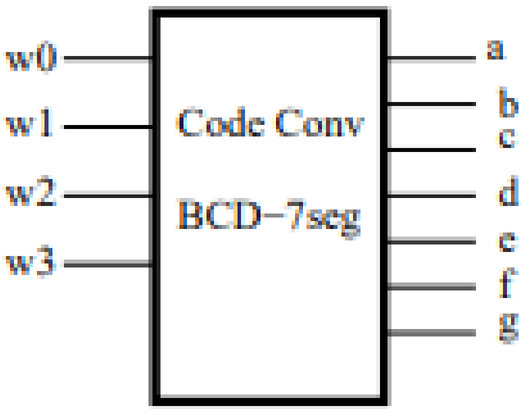
CODE Conversion

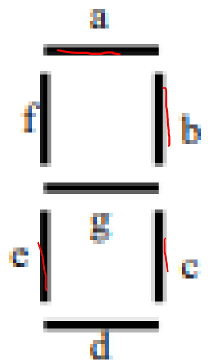
From BCD to 7 segment display

Segment a =0 + 2 + 3 + 5 + 6 + 7 + 8 + 9

Use a 4 to 16 decoder, then OR gate for each segment

$w3$	$w2$	$w1$	$w0$	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

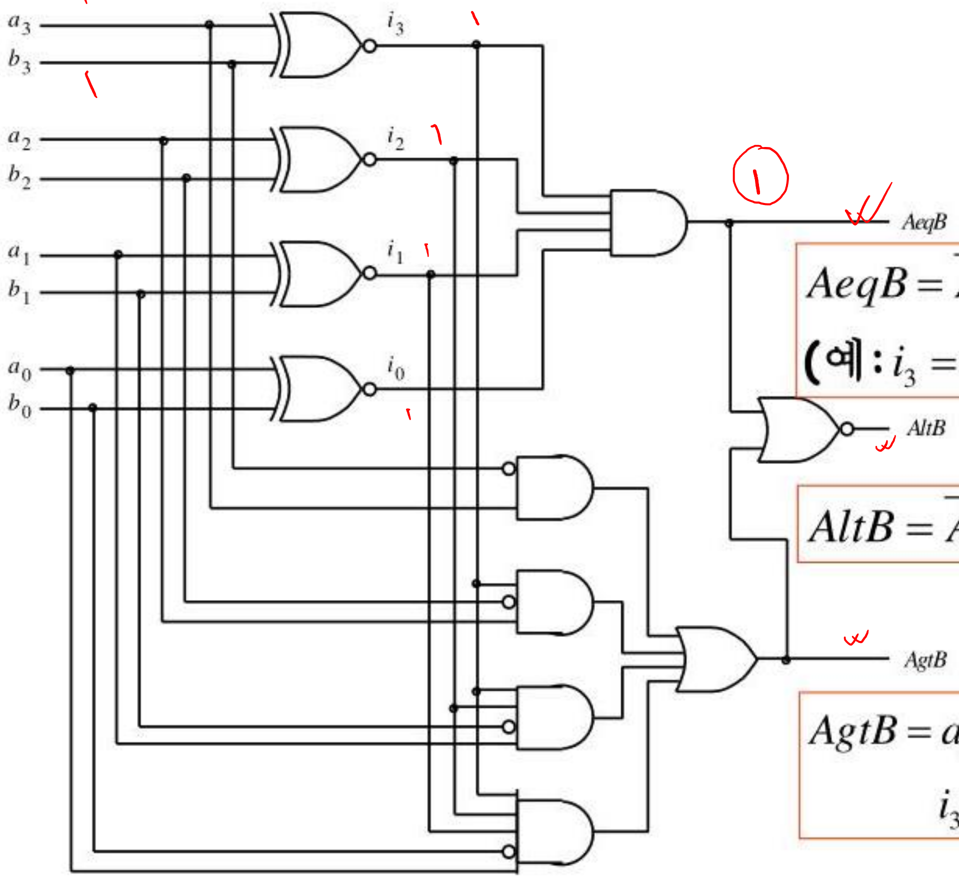




Code Converter

Arithmetic Comparator

5 ARITHMETIC COMPARISIONS



$$AeqB = \overline{A \oplus B} = i_3 i_2 i_1 i_0$$

(예 : $i_3 = \overline{a_3 \oplus b_3}$)

$$AltB = \overline{AeqB + AgtB}$$

$$AgtB = a_3\overline{b_3} + i_3 a_2 \overline{b_2} + i_3 i_2 a_1 \overline{b_1} + i_3 i_2 i_1 a_0 \overline{b_0}$$

Figure 6.26. A four-bit comparator circuit.

A , B

A > B

A < B

A = B

	a_3	a_2	a_1	a_0
A →	1	0	1	1
	b_3	b_2	b_1	b_0
B →	1	0	1	1

x	y	f
0	0	1
0	1	0
1	0	0
1	1	1

VHDL Codes

```
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;
```

VHDL Codes

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

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

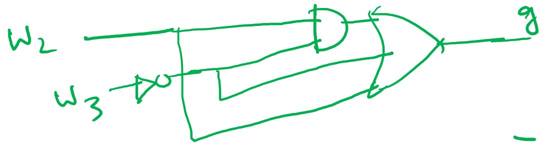
VHDL Code

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

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

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

Example



6.2. Use a 3-to-8 decoder and an OR gate to implement

$$f = \sum m\,(1, 2, 3, 5, 6)$$

$g = \bar{w}_3 + w_2\bar{w}_3 + w_2$

$= \bar{w}_2(\bar{w}_3) + w_2(1)$

$1 + \bar{x} + x = 1$

6.3. Use 2-to-1 MUX to implement $f = \bar{w}_1\bar{w}_3 + w_2\bar{w}_3 + \bar{w}_1 w_2$

only

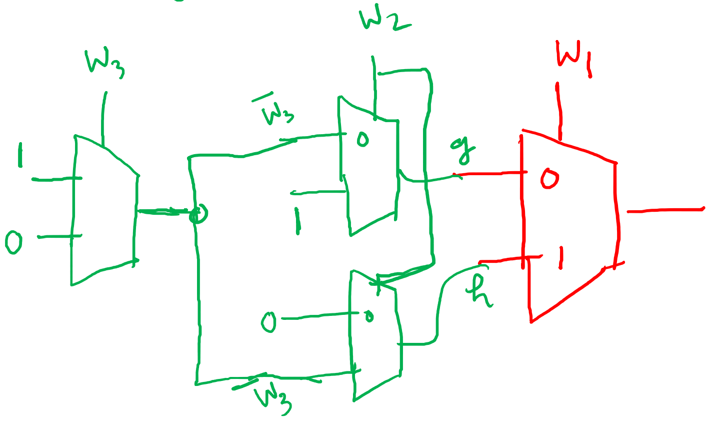
has no w_1

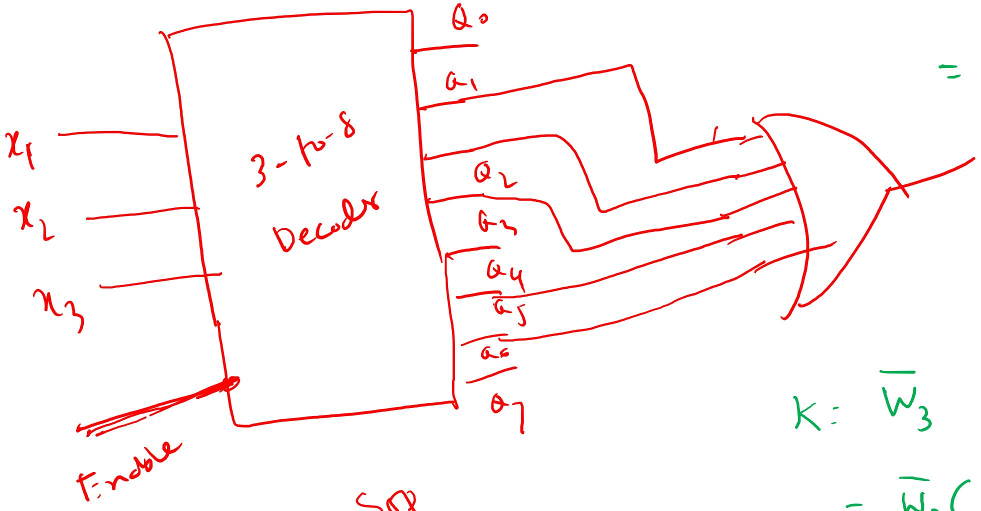
$h = w_2\bar{w}_3$

$= \bar{w}_2(0) + w_2(\bar{w}_3)$

$f = \bar{w}_1(\bar{w}_3 + w_2\bar{w}_3 + w_2) + w_1(w_2\bar{w}_3)$

$= \bar{w}_1 g + w_1 h$





SOP

$K = \bar{w}_3$

$= \bar{w}_3(1) + w_3(0)$