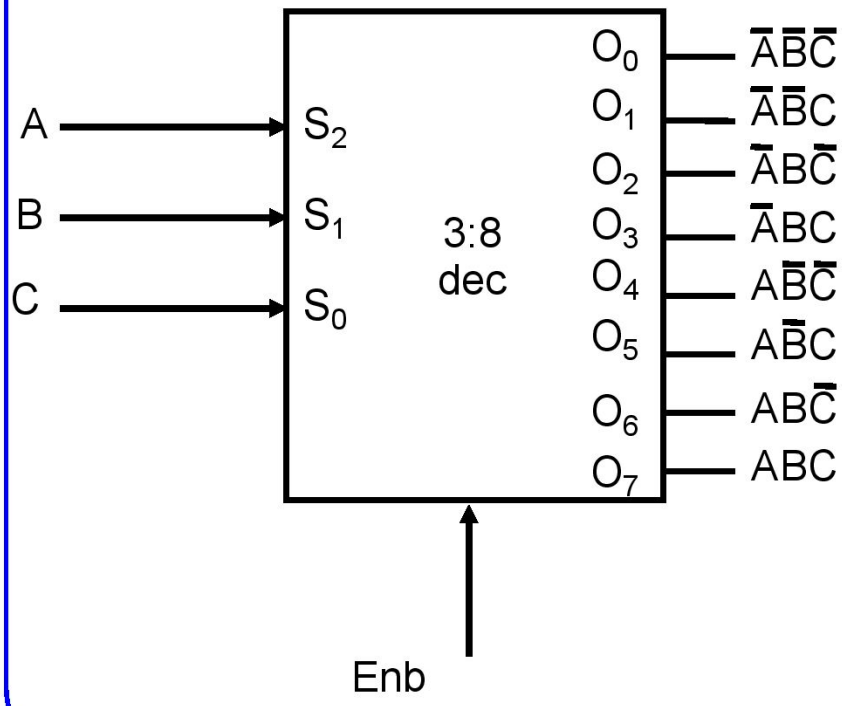
Decoder

Decoder

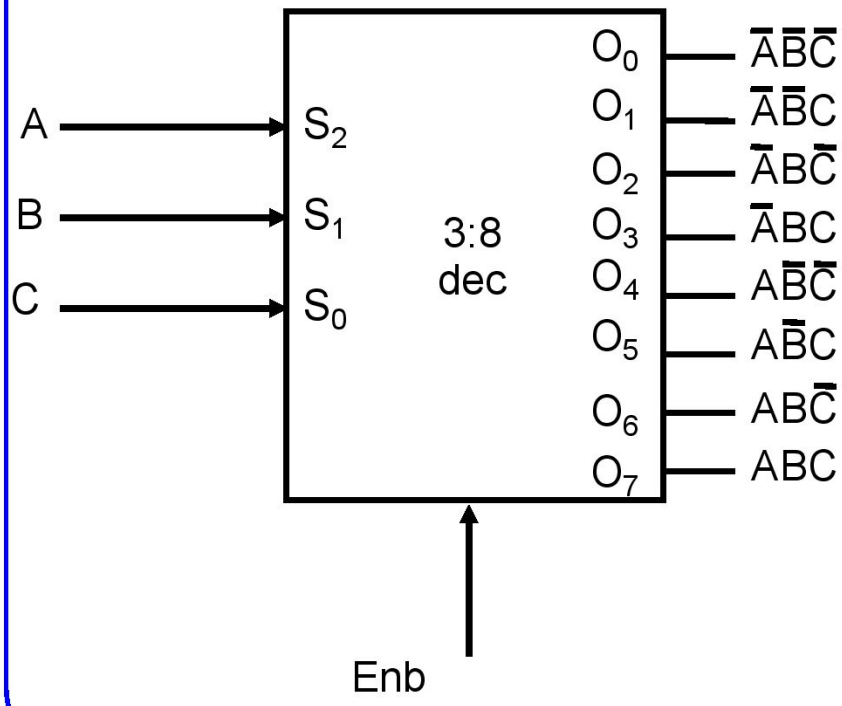
- 2-to-4,
- 3-to-8,
- …
- n-to-2^n



A	B	C	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

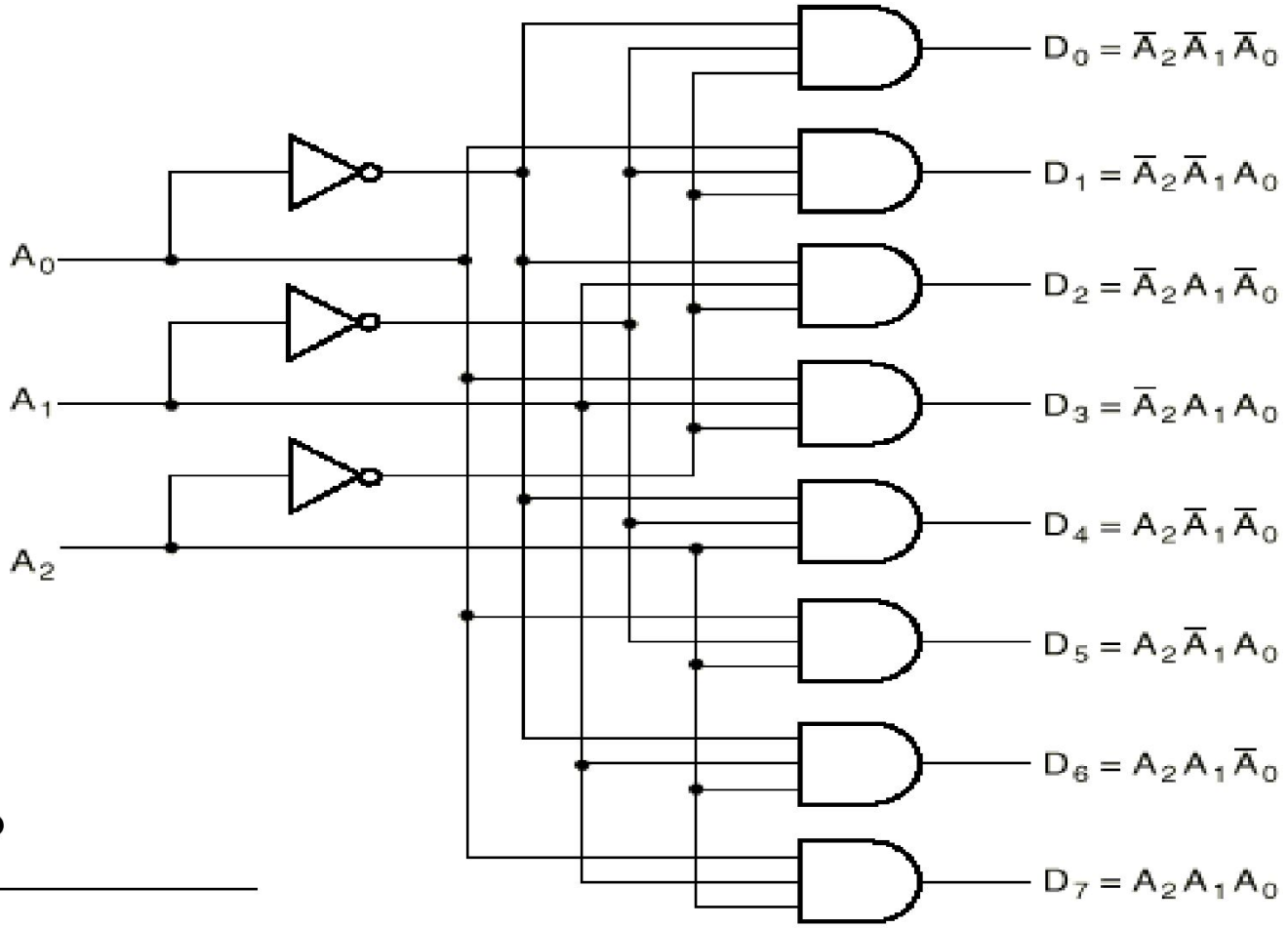
Decoder

- 2-to-4,
- 3-to-8,
- …
- n-to-2^n



E	A	B	C	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	X	X	X	0	0	0	0	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	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

Decoder

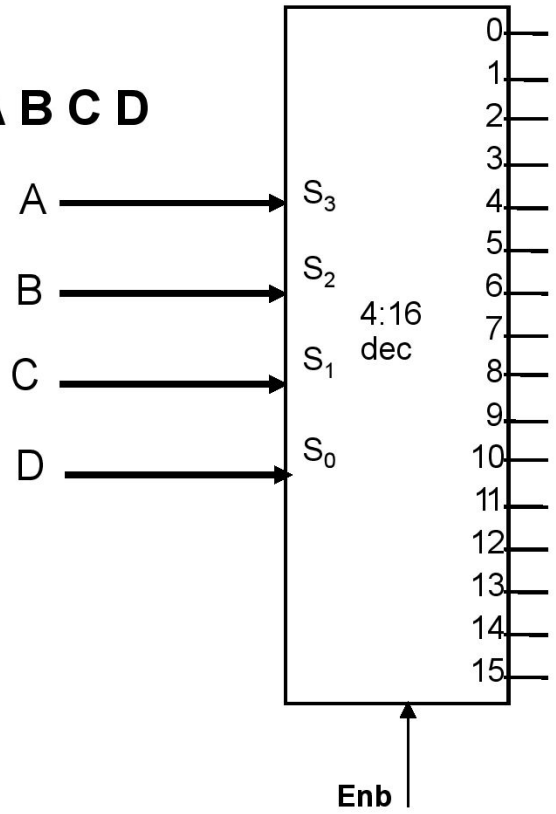

A0
A1
A2
D0 = Ā2Ā1Ā0
D1 = Ā2Ā1A0
D2 = Ā2A1Ā0
D3 = Ā2A1A0
D4 = A2Ā1Ā0
D5 = A2Ā1A0
D6 = A2A1Ā0
D7 = A2A1A0

E?

Design Using Decoder

- **Applications:**
  - Implementing General Logic
    - Any combinational circuit can be constructed using decoders and OR gates!
- **Example:**

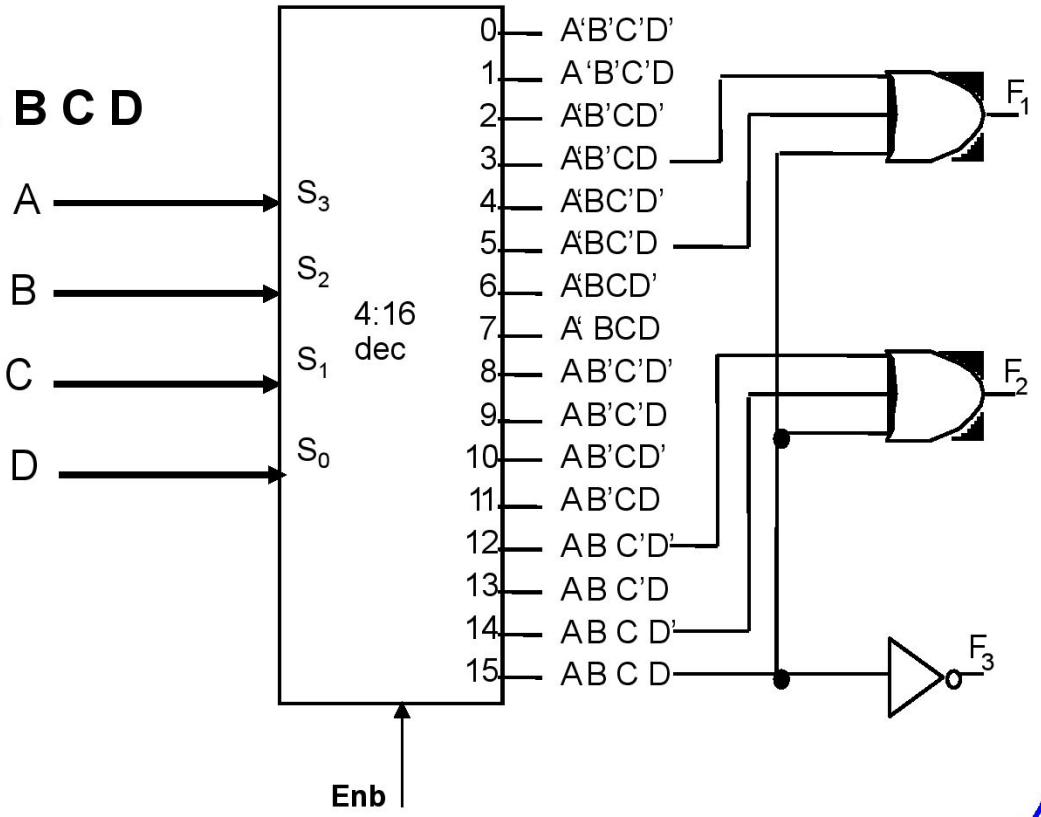
F1 = A' B C' D + A' B' C D + A B C D
F2 = A B C' D' + A B C
F3 = (A' + B' + C' + D')



Design Using Decoder

- **Applications:**
  - Implementing General Logic
    - Any combinational circuit can be constructed using decoders and OR gates!
- **Example:**

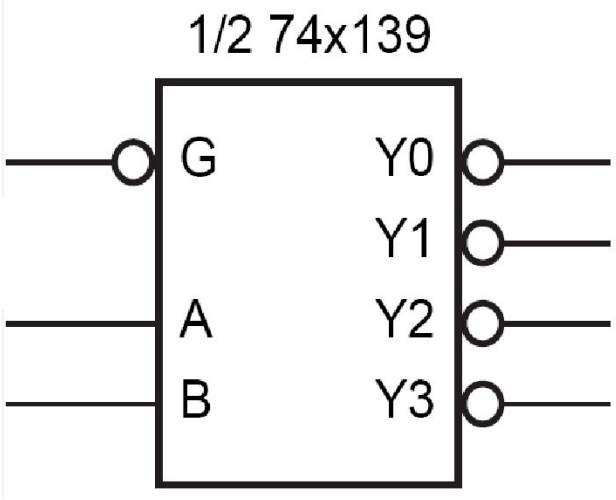
F1 = A' B C' D + A' B' C D + A B C D
F2 = A B C' D' + A B C
F3 = (A' + B' + C' + D')



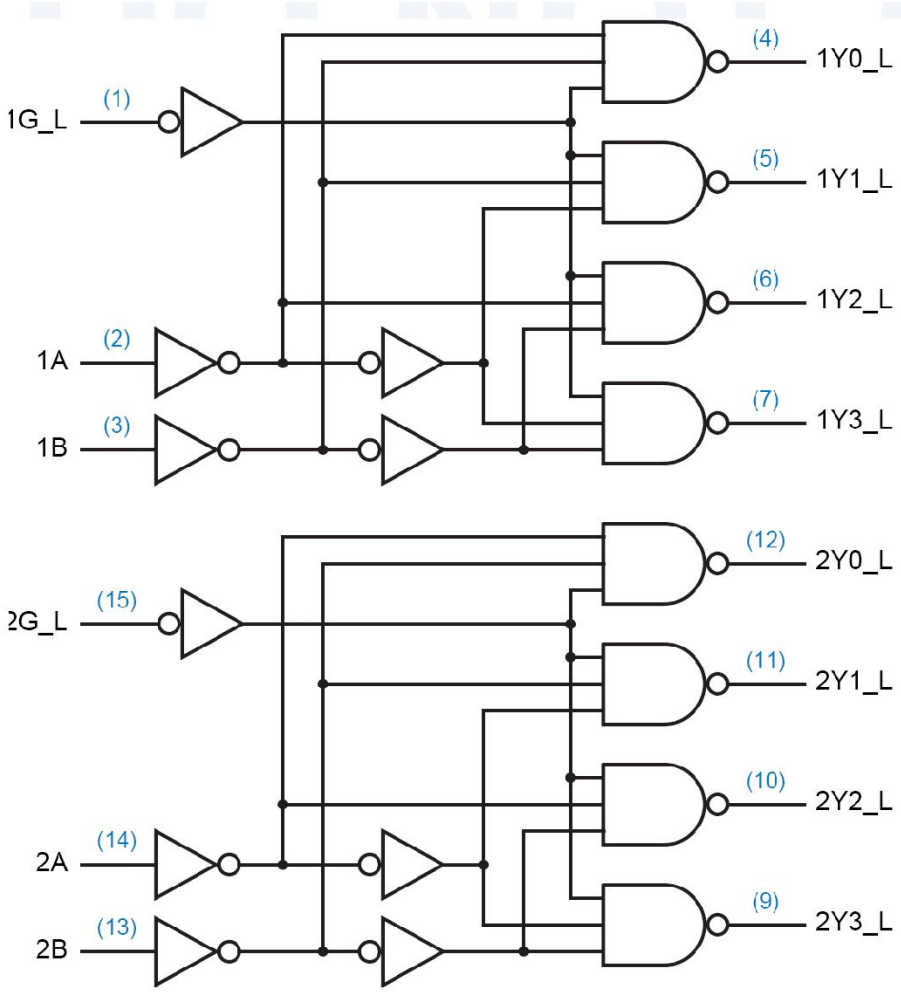
Active Low

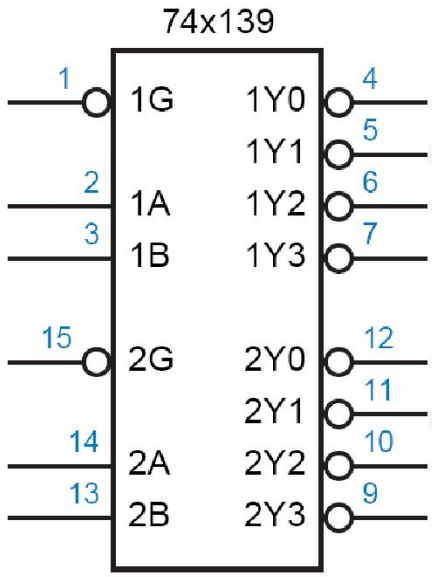
- **Decoder with**
  - Active Low Enable
  - Active Low Outputs

G	A	B	Y_0	Y_1	Y_2	Y_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



74x139 dual 2-to-4 decoder

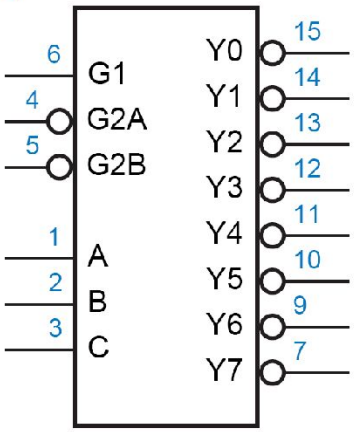




Inputs			*Outputs*			
G_L	B	A	Y3_L	Y2_L	Y1_L	Y0_L
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

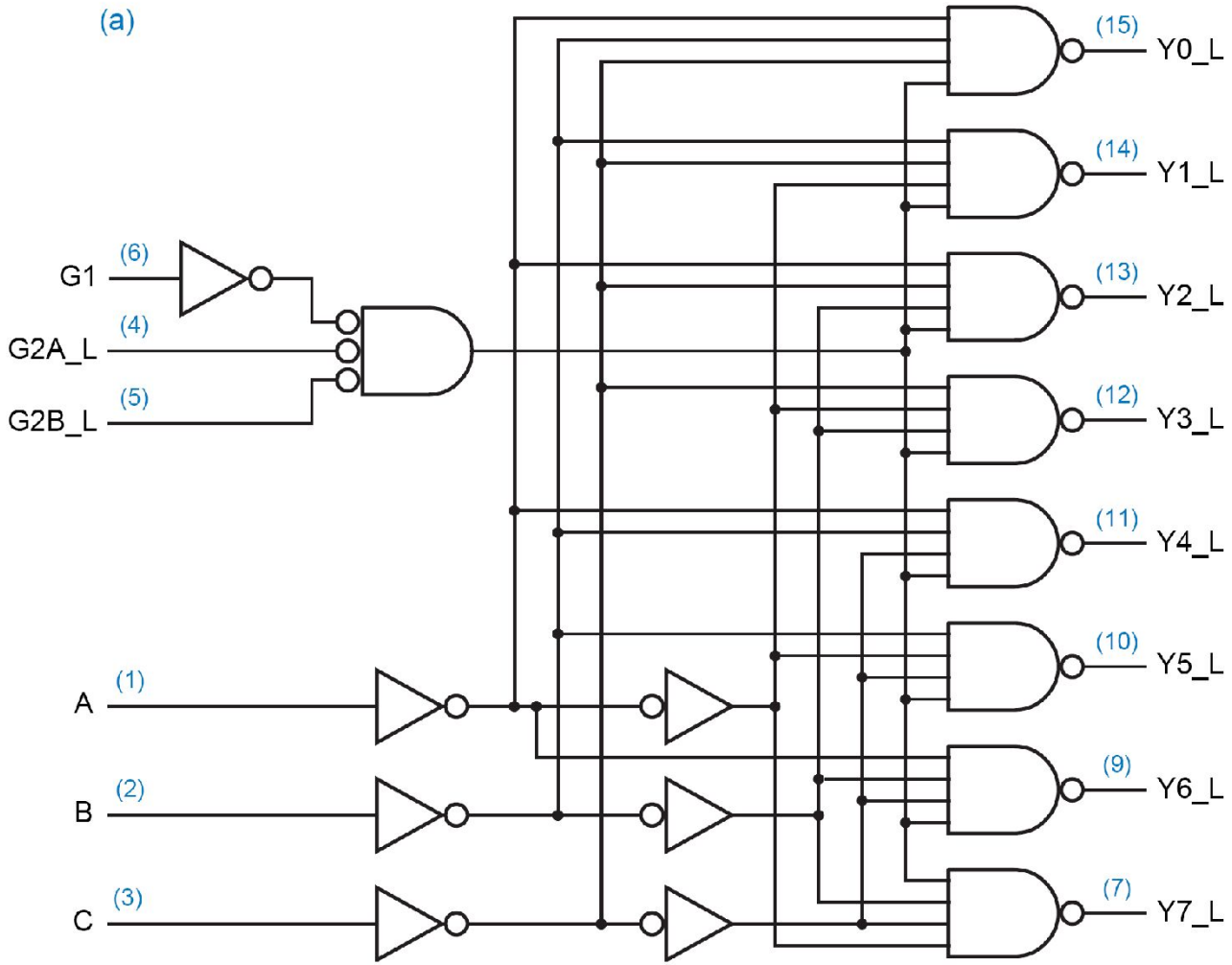
74x138 3-8 Decoder

(b)



Inputs						*Outputs*							
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

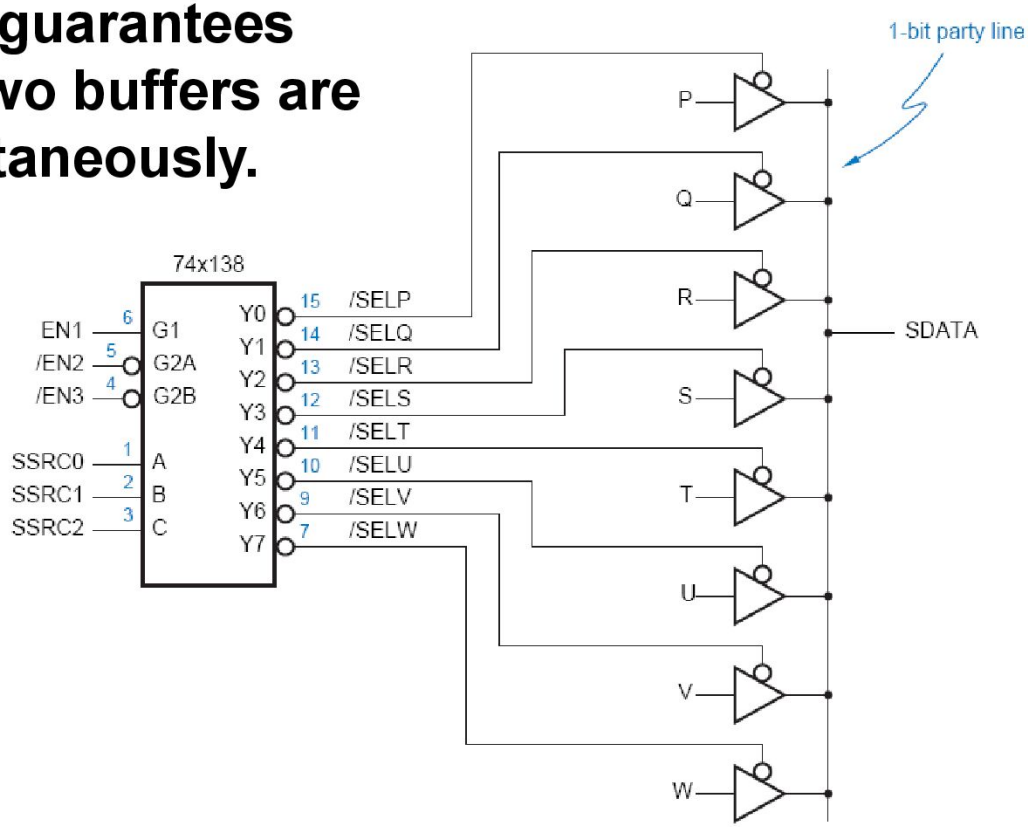
74x138 3-8 Decoder



Using 3-State Buffers

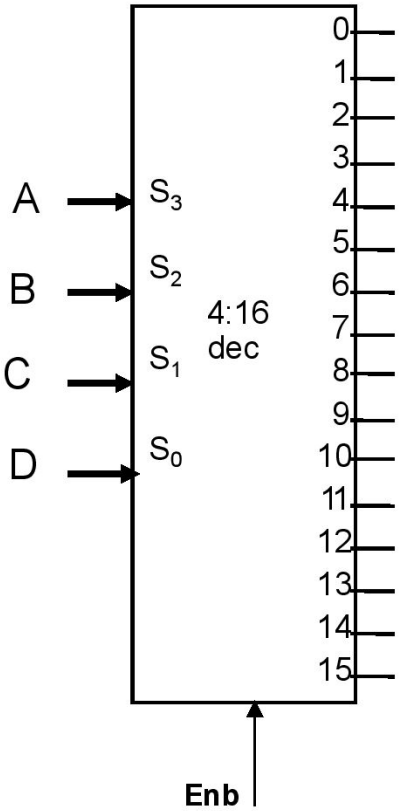
- Can use 3-state buffers to share a single line for several devices.

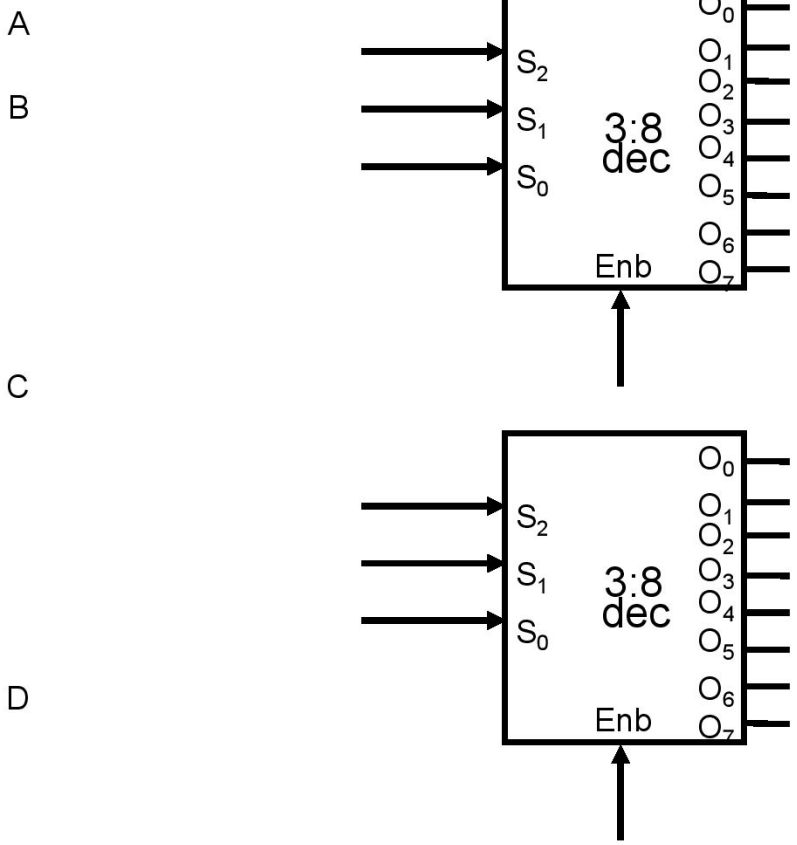
Decoder guarantees that no two buffers are on simultaneously.



Decoders

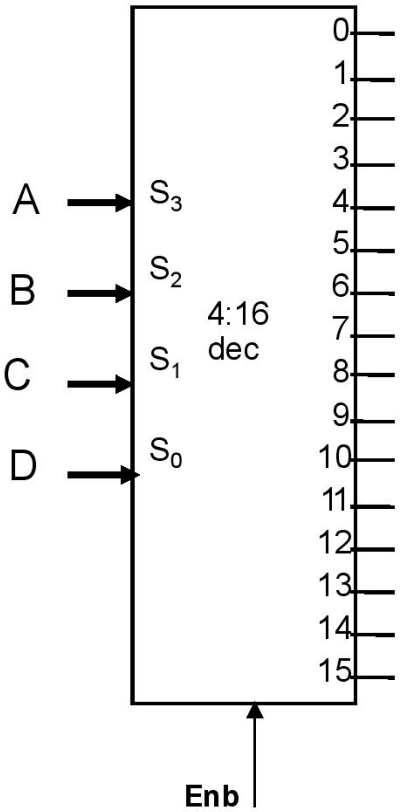
- Can build a decoder by smaller decoders

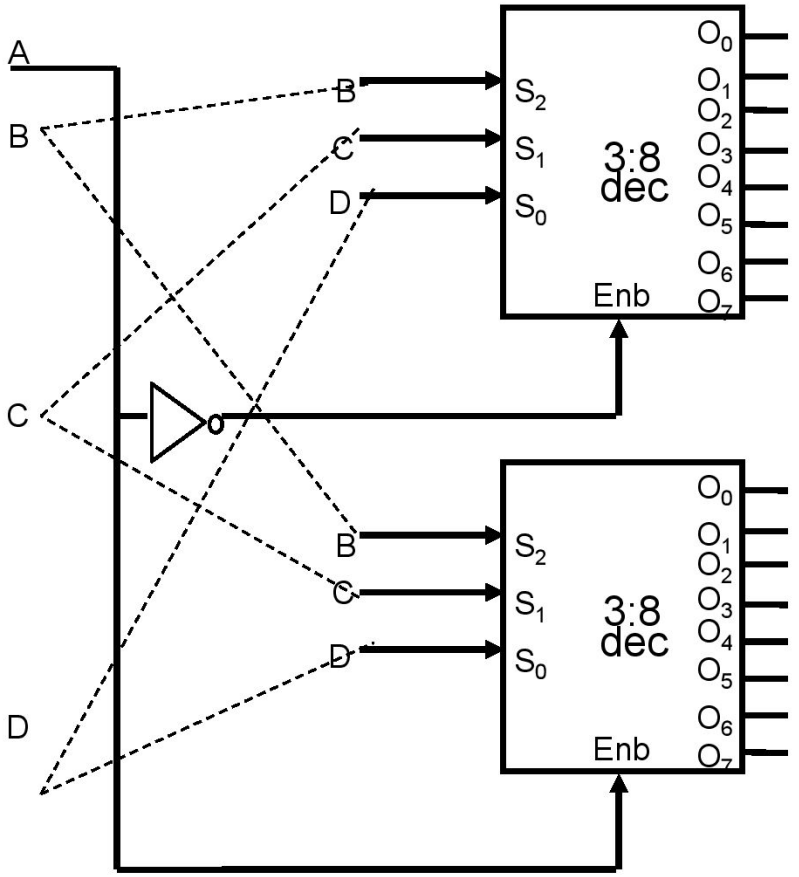




Decoders

- Can build a decoder by smaller decoders





Decoders

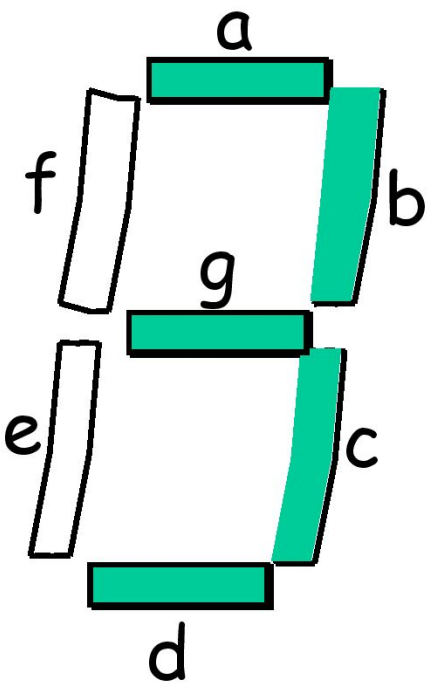
- How to build a 5-32 decoder by using 4-16 and 2-4 decoders?

Decoders

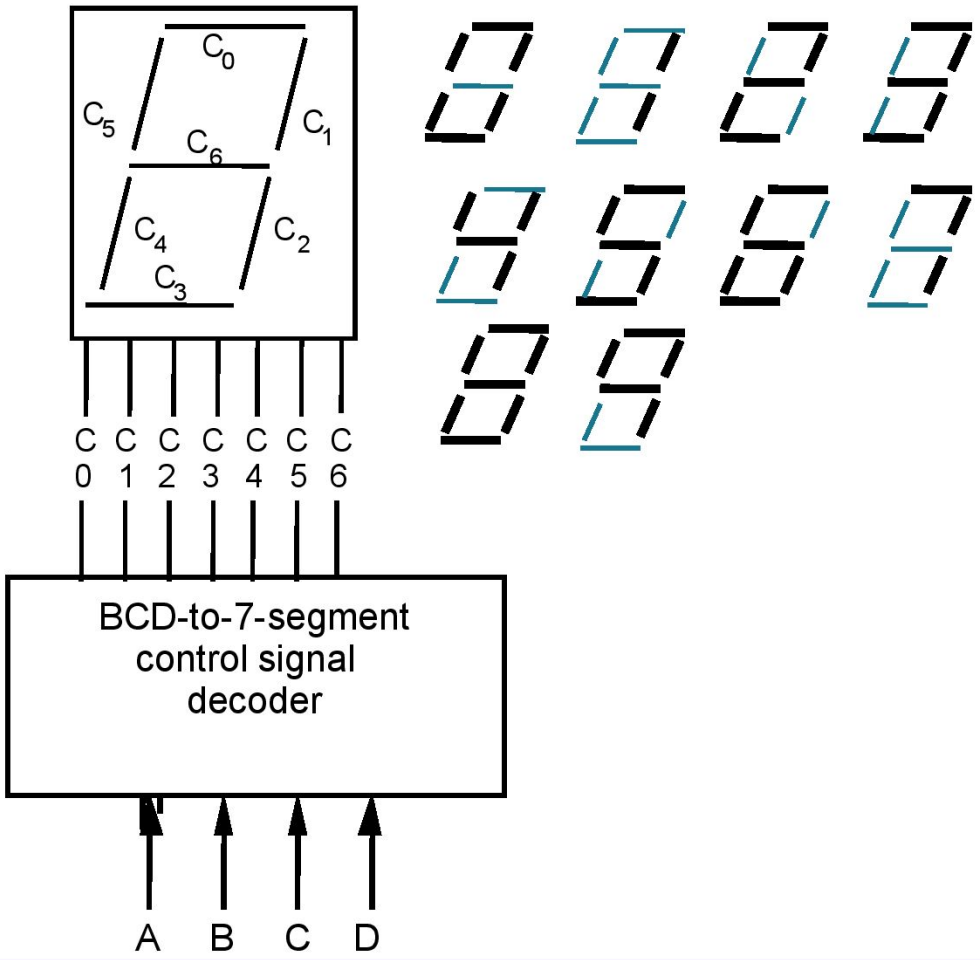
- **Decoder: a more general term**
  - Our focus was on “binary decoders”

7-Segment Decoder

- **Seven-segment display:**
  - 7 LEDs (light emitting diodes), each one controlled by an input
  - 1 means “on”, 0 means “off”
  - Display digit “3”?
    - Set a, b, c, d, g to 1
    - Set e, f to 0

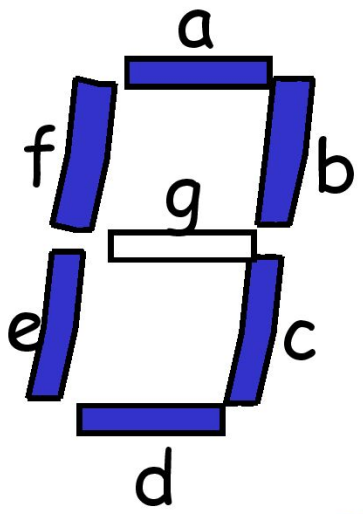


7-Segment Decoder



7-Segment Decoder

- **7-Segment Decoder:**
  - Input is a 4-bit BCD code → 4 inputs (A, B, C, D).
  - Output is a 7-bit code (a,b,c,d,e,f,g) that allows for the decimal equivalent to be displayed.
- **Example:**
  - Input: 0000_{BCD}
  - Output: 1111110 (a=b=c=d=e=f=1, g=0)

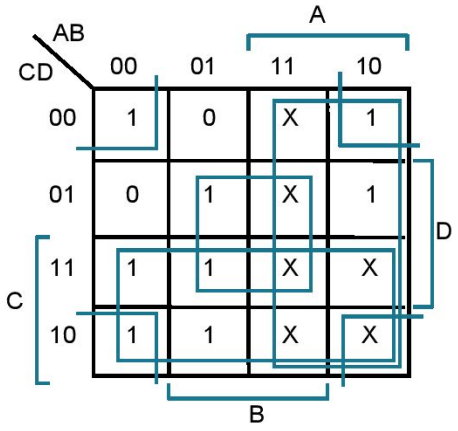


BCD-to-7Segment Truth Table

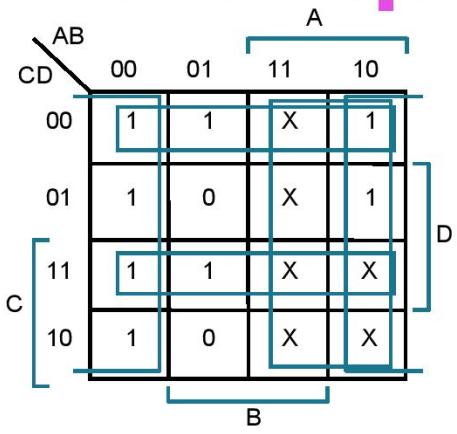
Digit	ABCD	abcdefg
0	0000	1111110
1	0001	0110000
2	0010	1101101
3	0011	1111001
4	0100	0110011
5	0101	1011011
6	0110	X011111
7	0111	11100X0

Digit	ABCD	abcdefg
8	1000	1111111
9	1001	111X011
	1010	XXXXXXX
	1011	XXXXXXX
	1100	XXXXXXX
	1101	XXXXXXX
	1110	XXXXXXX
	1111	XXXXXXX

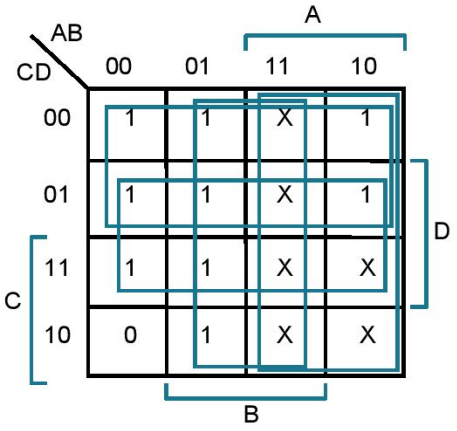
K-maps



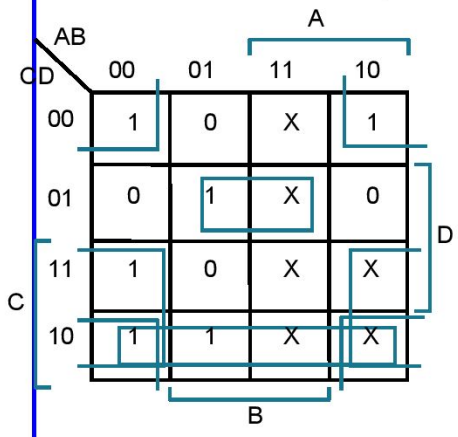
K-map for a



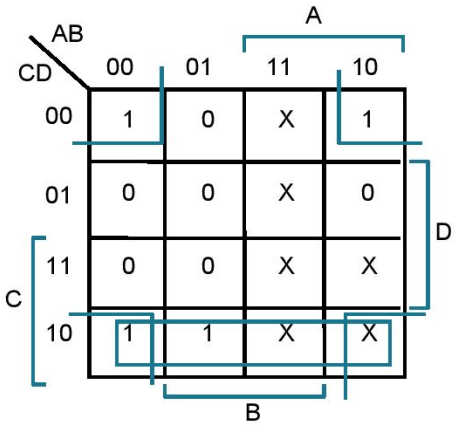
K-map for b



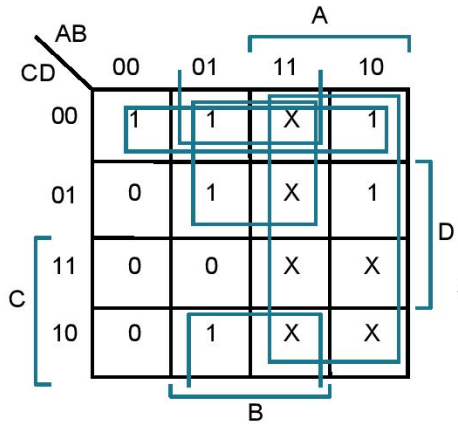
K-map for c



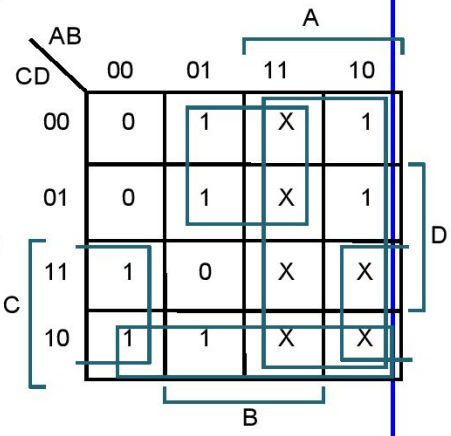
K-map for d



K-map for e



K-map for f



K-map for g

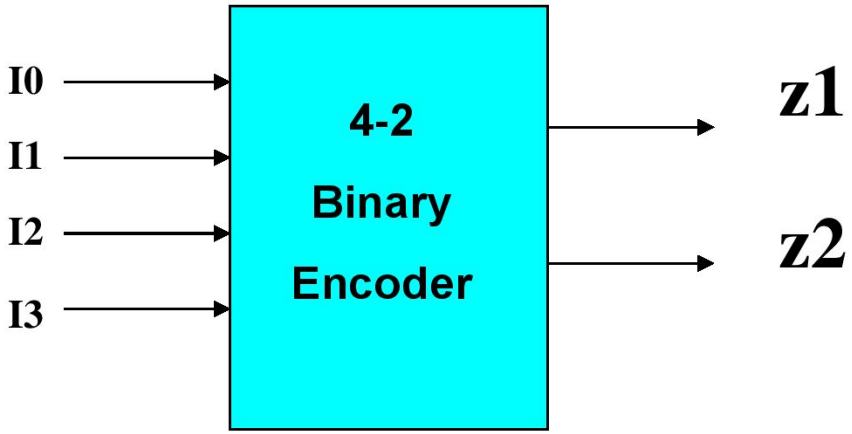
a = A + B D + C + B' D'
b = A + C' D' + C D + B'
c = A + B + C' + D

d = B' D' + C D' + B C' D + B' C
e = B' D' + C D
f = A + C' D' + B D' + B C'
g = A + C D' + B C' + B' C

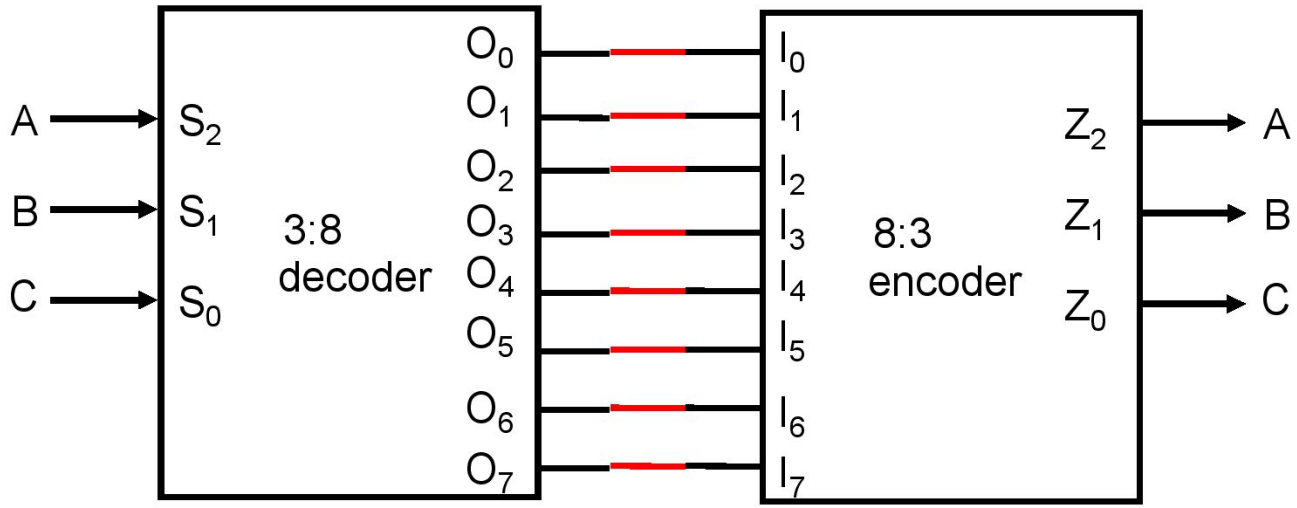
Encoder

Encoder

- **Encoder:**
  - the inverse operation of a decoder.
  - Has 2^n input lines and n output lines.
  - The output lines generate the binary equivalent of the input line whose value is 1.



Encoder


A
B
C
S_2
S_1
S_0
3:8
decoder
O_0
O_1
O_2
O_3
O_4
O_5
O_6
O_7
I_0
I_1
I_2
I_3
I_4
I_5
I_6
I_7
8:3
encoder
Z_2
Z_1
Z_0
A
B
C

Encoder Circuit Design

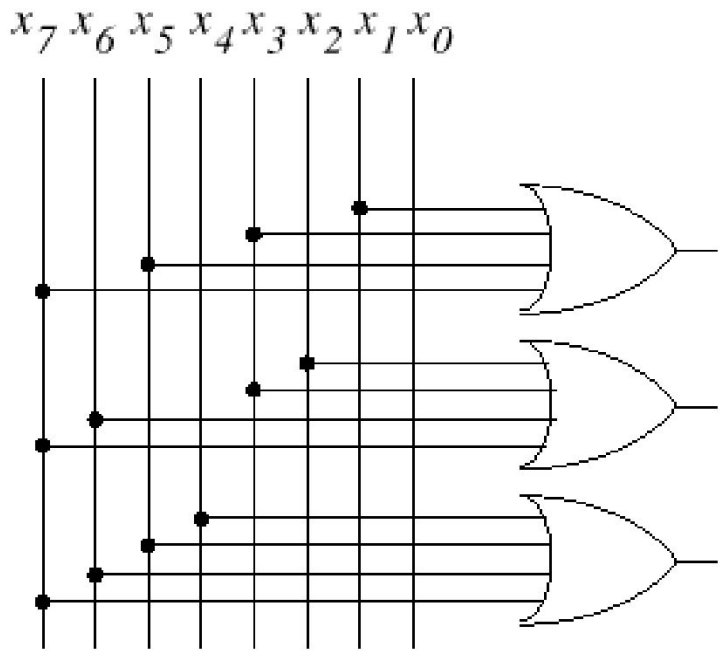
- **Example:**
  - 8-3 Binary Encoder

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$A_0 = D_1 + D_3 + D_5 + D_7$$
$$A_1 = D_2 + D_3 + D_6 + D_7$$
$$A_2 = D_4 + D_5 + D_6 + D_7$$

Encoder Circuit

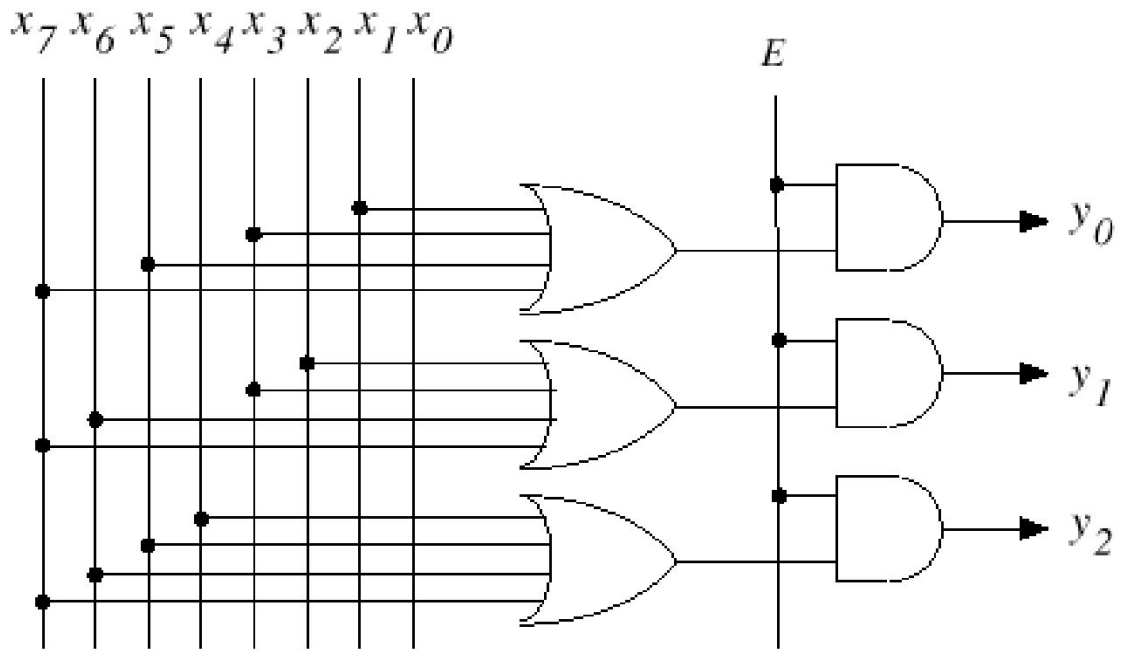
With Enable



With Acknowledge

Encoder Circuit

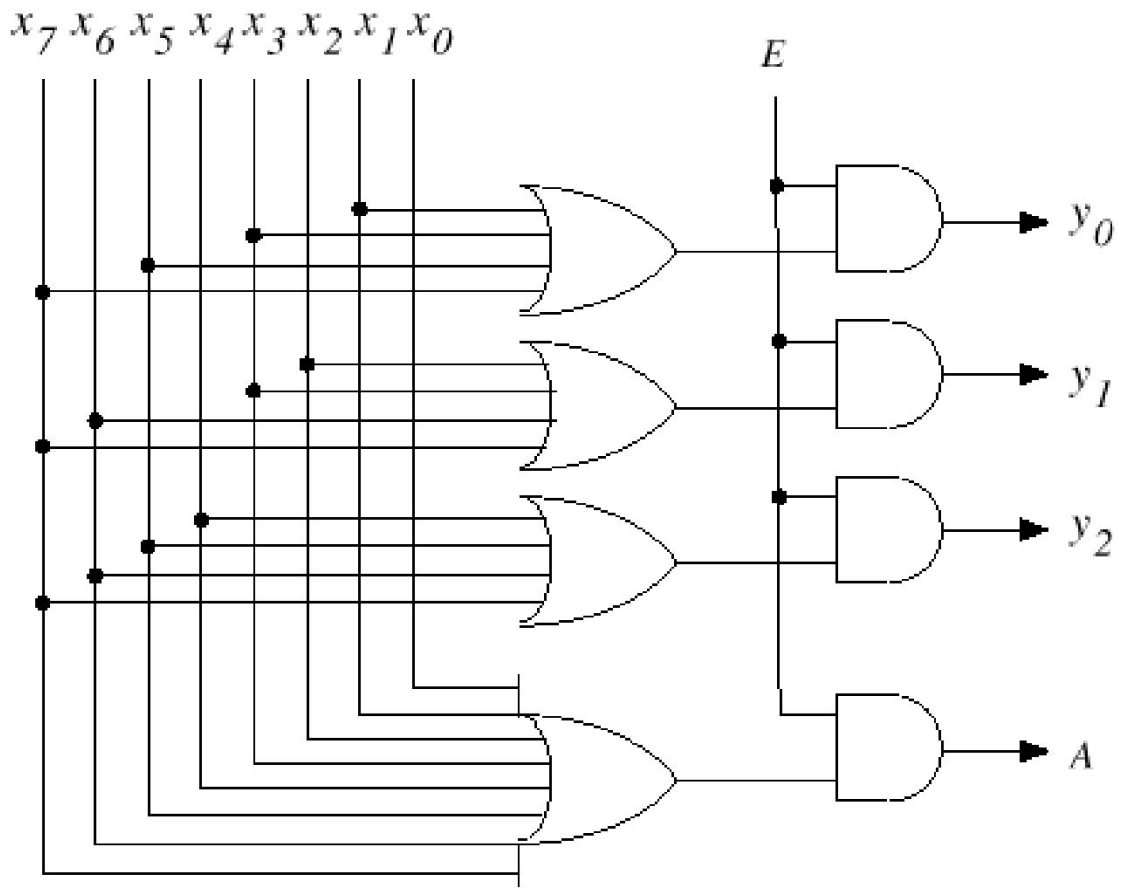
With Enable



With Acknowledge

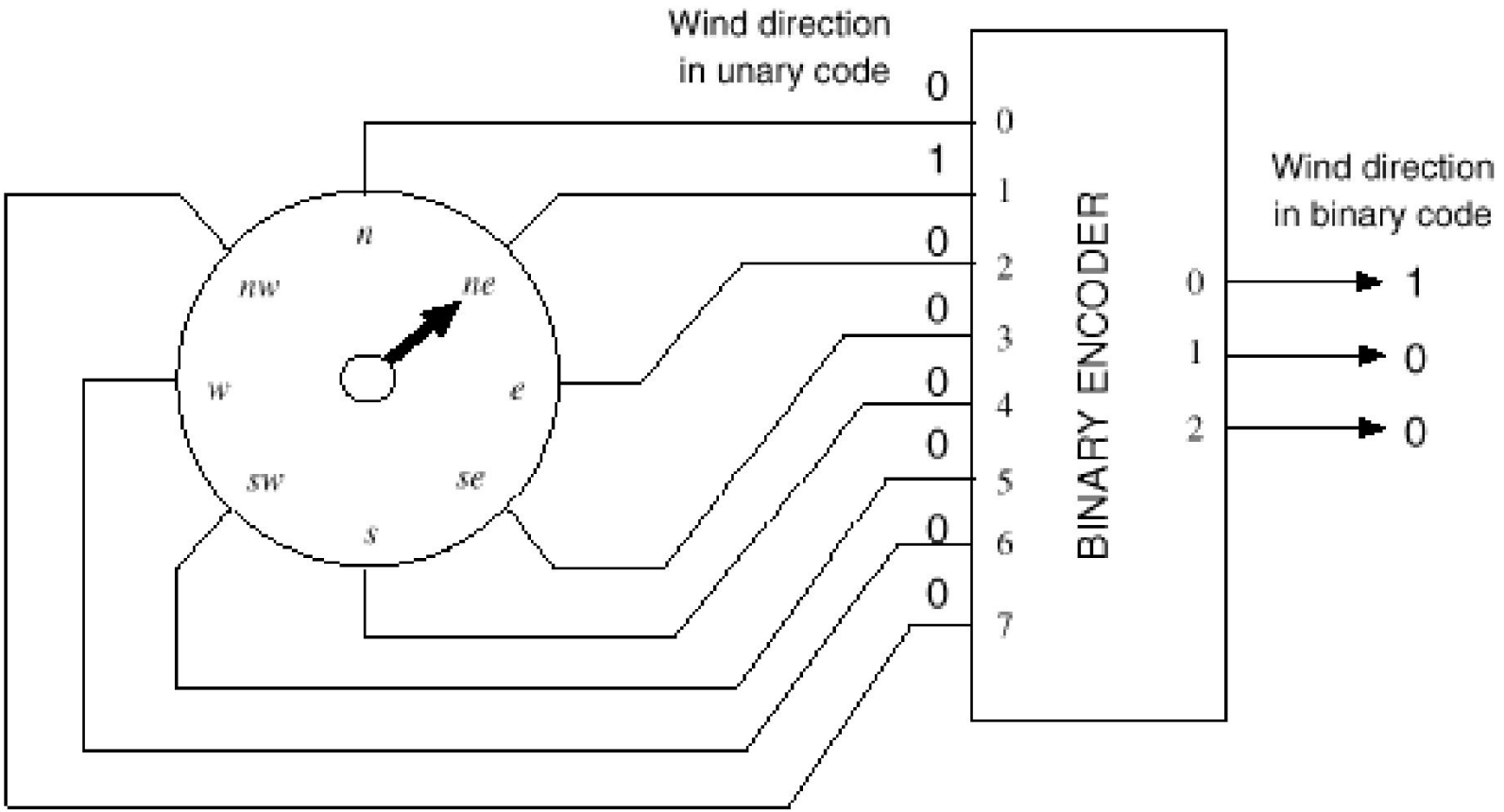
Encoder Circuit

With Enable



With Acknowledge

Application



The number of inputs: large

→ fewer lines

Encoder Design Issues

- Only one input can be active at any given time.
  - If two inputs are active simultaneously, the output produces an undefined combination
    - (for example, if D_3 and D_6 are 1 simultaneously, the output of the encoder will be 111.

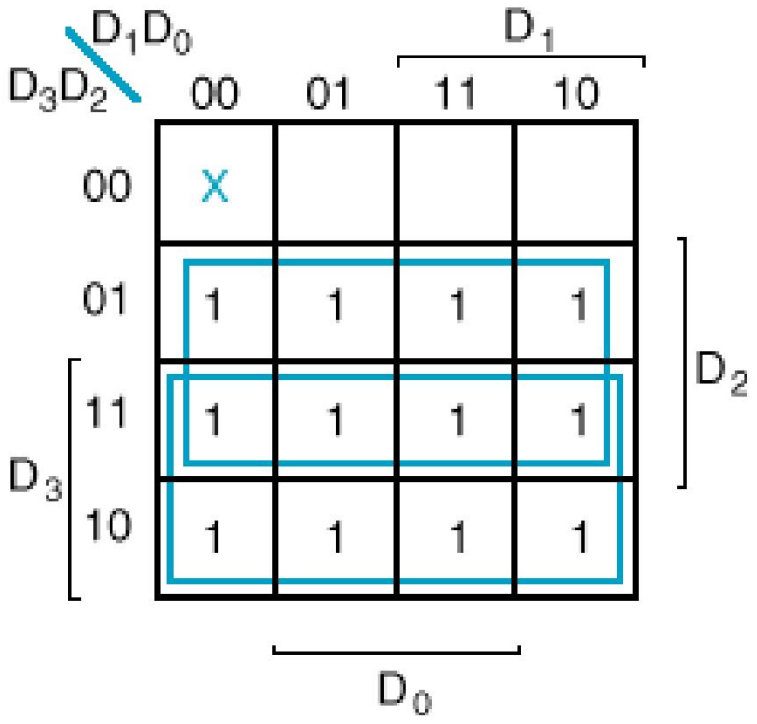
$$A_0 = D_1 + D_3 + D_5 + D_7$$
$$A_1 = D_2 + D_3 + D_6 + D_7$$
$$A_2 = D_4 + D_5 + D_6 + D_7$$

Priority Encoder

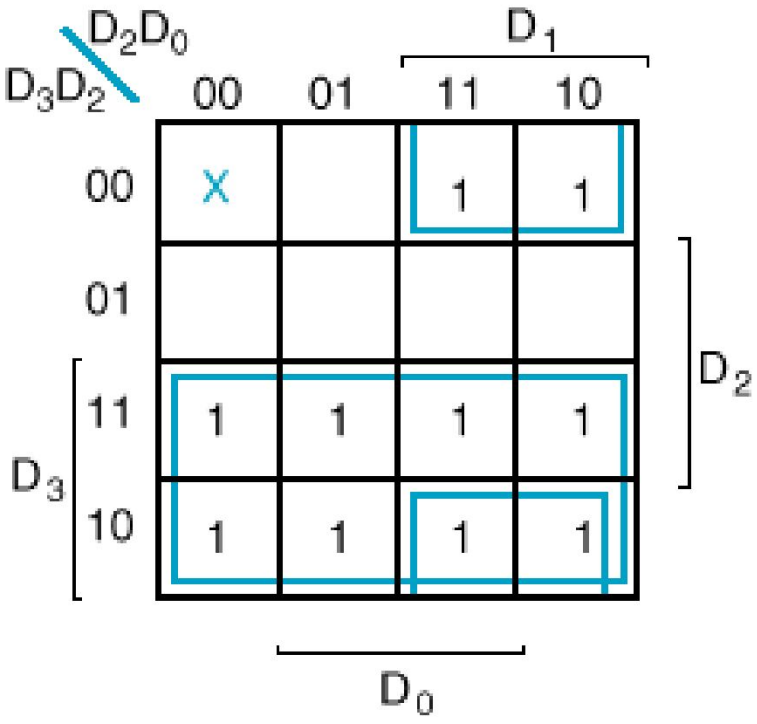
- Multiple asserted inputs are allowed; one has priority over all others.

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	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

K-Maps

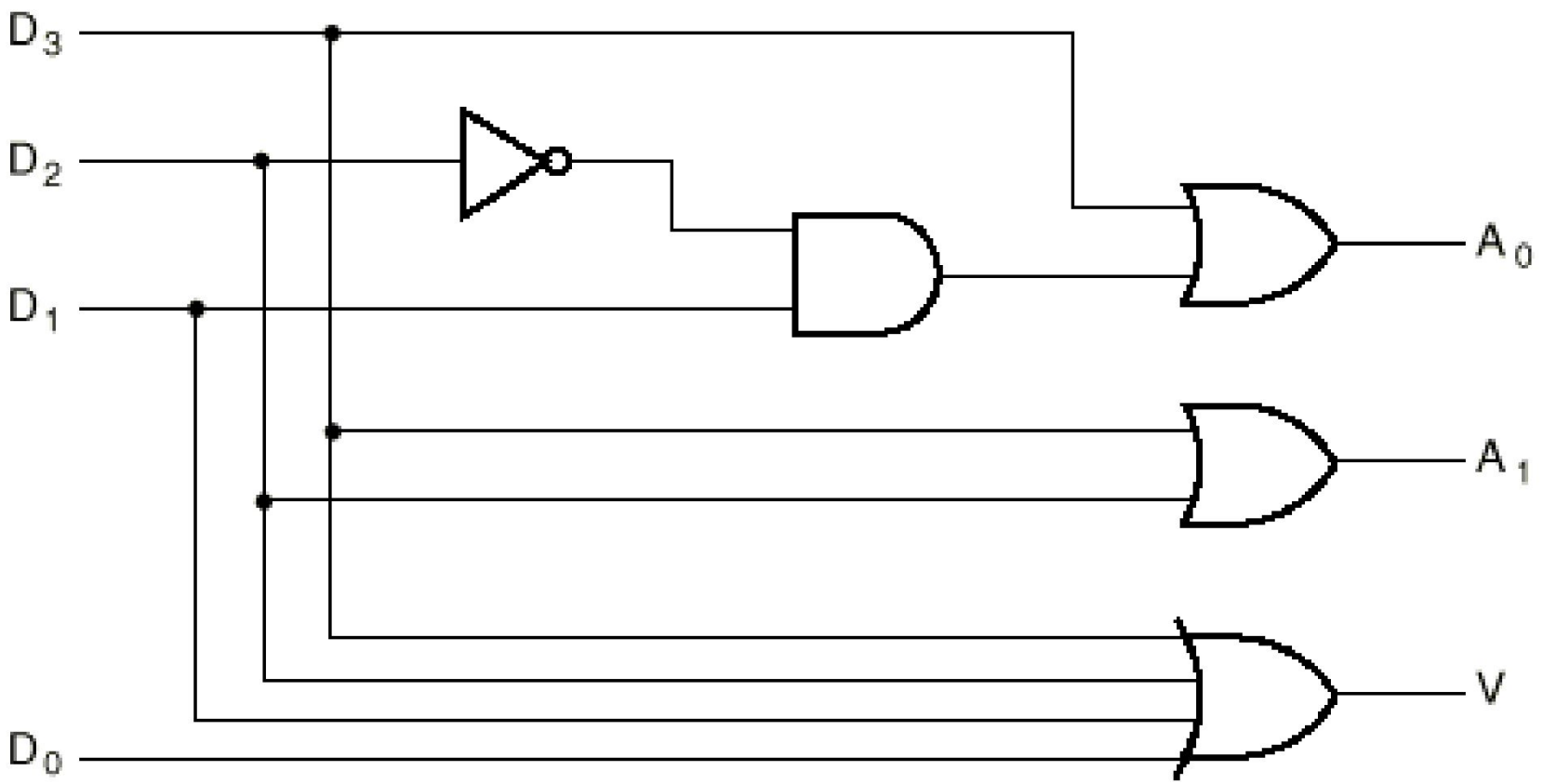


$A_1 = D_2 + D_3$



$A_0 = D_3 + D_1\overline{D}_2$

Circuit

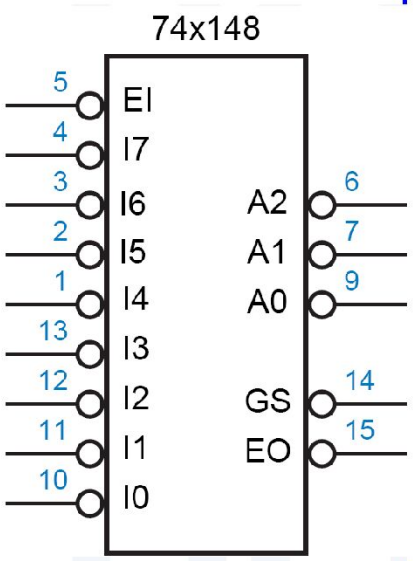


8-3 Priority Encoder

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Z_0	Z_1	Z_2	NR
0	0	0	0	0	0	0	0	X	X	X	1
X	X	X	X	X	X	X	1	1	1	1	0
X	X	X	X	X	X	1	0	1	1	0	0
X	X	X	X	X	1	0	0	1	0	1	0
X	X	X	X	1	0	0	0	1	0	0	0
X	X	X	1	0	0	0	0	0	1	1	0
X	X	1	0	0	0	0	0	0	1	0	0
X	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0

74x148

- **Features:**
  - inputs and outputs are active low.
  - EI_L must be asserted for any of its outputs to be asserted.
  - GS_L is asserted when the device is enabled and one or more of the request inputs is asserted. (“Group Select” or “Got Something.”)
  - EO_L is an enable output designed to be connected to the EI_L input of another ’148 that handles lower-priority requests.
    - It is asserted if EI_L is asserted but no request input is asserted; thus, a lower-priority ’148 may be enabled.



74x148 Truth Table

Inputs									Outputs				
/EI	/I0	/I1	/I2	/I3	/I4	/I5	/I6	/I7	/A2	/A1	/A0	/GS	/EO
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
0	x	x	x	x	x	0	1	1	0	1	0	0	1
0	x	x	x	x	0	1	1	1	0	1	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	0	1
0	x	x	0	1	1	1	1	1	1	0	1	0	1
0	x	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0

Datasheets

- http://users.otenet.gr/~athsam/database.htm
- Some sample datasheets in the course site.