EEE 3101: Digital Logic and Circuits

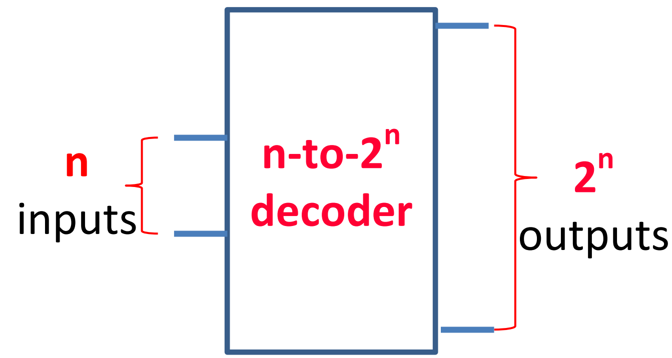
Decoder & Encoder

Course Teacher: Nafiz Ahmed Chisty

Associate Professor, Department of EEE & CoE
Head (UG), Department of EEE
Faculty of Engineering
Room# DNG03, Ground Floor, D Building
Email: chisty@aiub.edu
Website: http://engg.aiub.edu/
Website: www.nachisty.com

Decoder

A **n-to-2^n decoder** takes an **n-bit** input and produces **2^n** outputs. The n inputs represent a binary number that determines which of the 2^n outputs is *uniquely* true.



Example:

- Reception counter: When you reach an Academic Institute

- Receptionist asks: Which Dept. to go?

- Based on your Specific answer, Receptionist redirects you to the specific building.

The job of the Decoder is to **Decode!**

-It knows what to do for a fixed question.

Use:

- Memory addressing

- Address to a particular location.

Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

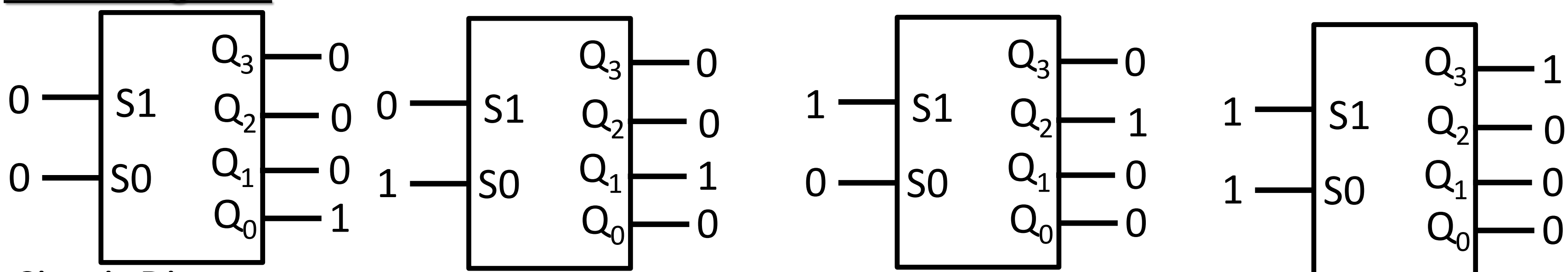
2-to-4 decoder

- Truth table:

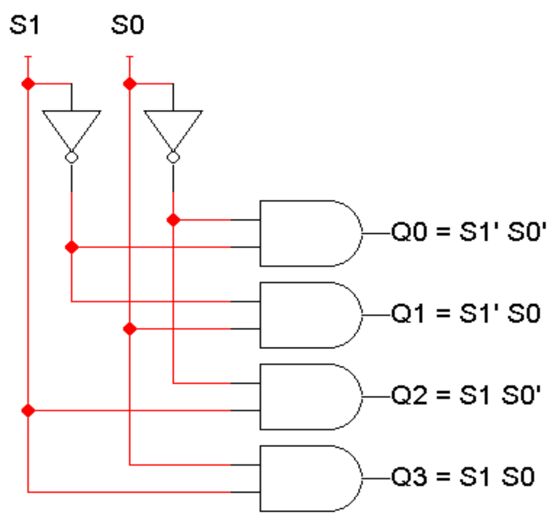
Input		Output			
S_1	S_0	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

For instance, if the input **S1 S0 = 10** (decimal 2), then output **Q2 is true**, and **Q0, Q1, Q3 are all false**. This circuit "decodes" a binary number into a "one-of-four" code.

- Block Diagram:



- Circuit Diagram:



Q0 = S1' S0'

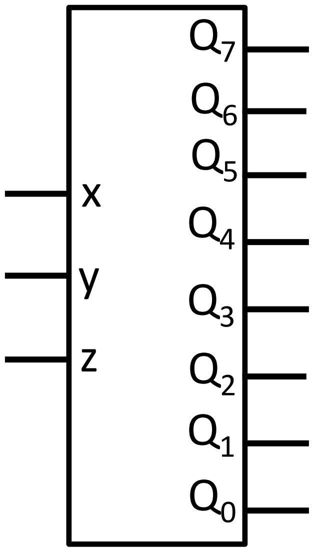
Q1 = S1' S0

Q2 = S1 S0'

Q3 = S1 S0

3-to-8 decoder

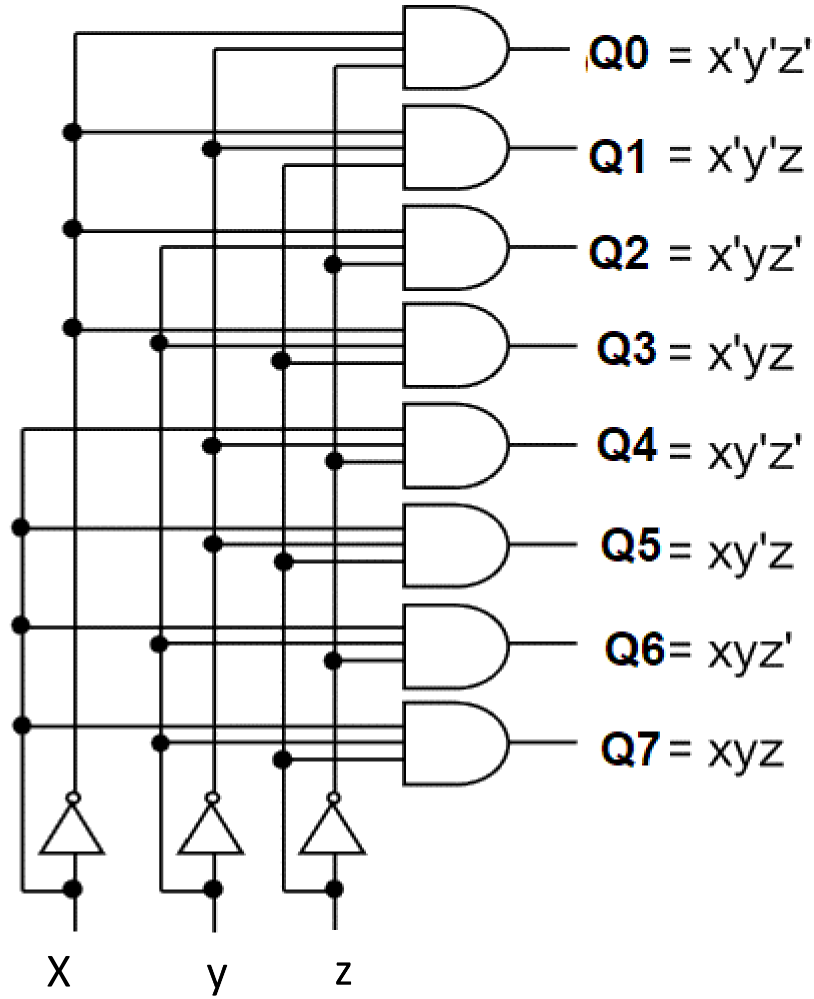
- Block Diagram:



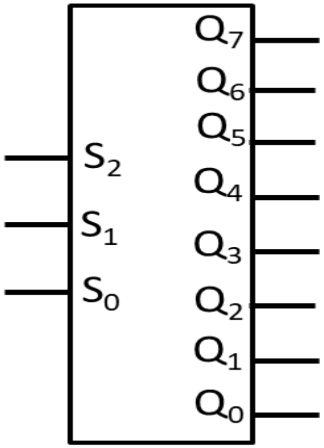
- Truth table:

Input			Output							
x	y	z	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_5	Q_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

- Circuit Diagram:

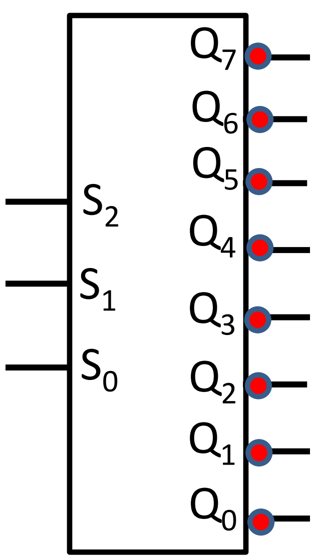


3-to-8 decoder (Active High Output)



Input			Output							
S_2	S_1	S_0	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_5	Q_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

3-to-8 decoder (Active Low Output)



Input			Output							
S_2	S_1	S_0	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_5	Q_7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

- Decoding ONLY a specific sequence:

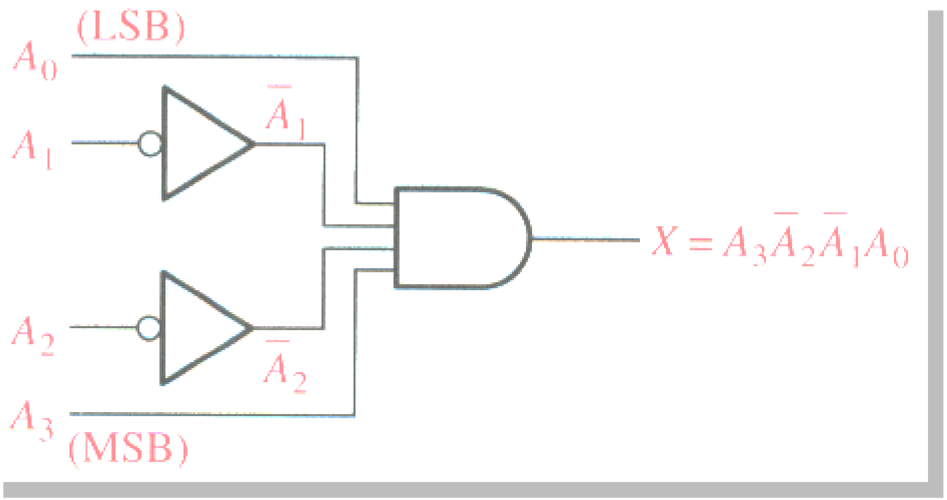
The output is 1 only when:

$A_0 = 1$

$A_1 = 0$

$A_2 = 0$

$A_3 = 1$

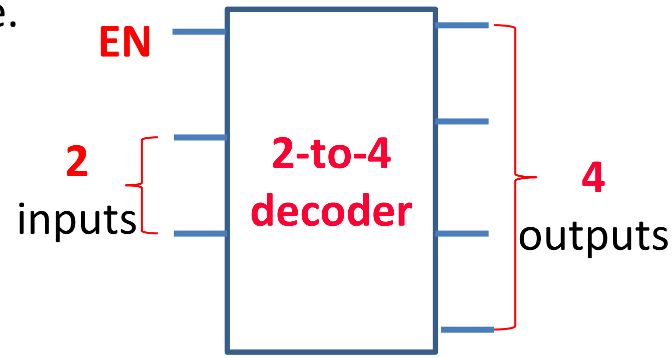


Use:

1) Encryption system,
2) Counter decoding…etc.

Enable inputs

- Many devices have an additional **enable input**, which is used to "**activate**" or "**deactivate**" the device.



- For a decoder,
  - EN=0 "deactivates" the decoder. By convention, that means *all* of the decoder's outputs are 0.
  - EN=1 activates the decoder, so it behaves as specified earlier. Exactly one of the outputs will be 1.

EN	S1	S0	Q0	Q1	Q2	Q3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
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

EN	S1	S0	Q0	Q1	Q2	Q3
0	x	x	0	0	0	0
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

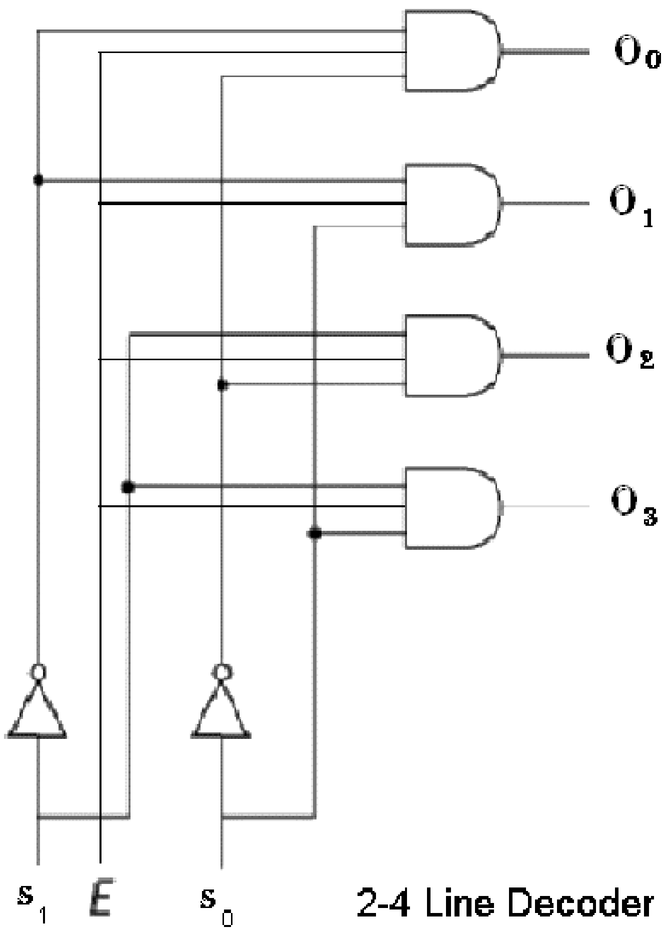
$Q0 = S1'\, S0'\, \mathbf{EN}$

$Q1 = S1'\, S0\, \mathbf{EN}$

$Q2 = S1\, S0'\, \mathbf{EN}$

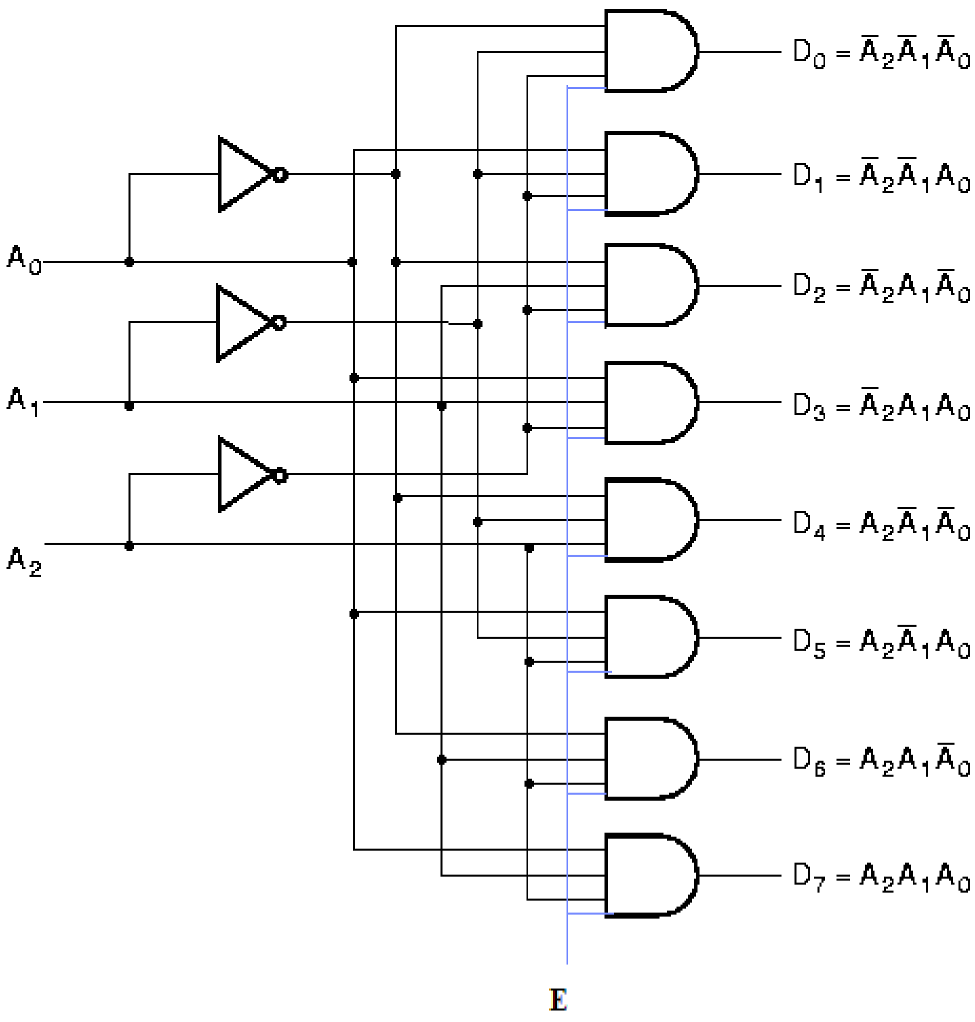
$Q3 = S1\, S0\, \mathbf{EN}$

2-to-4



2-4 Line Decoder

3-to-8



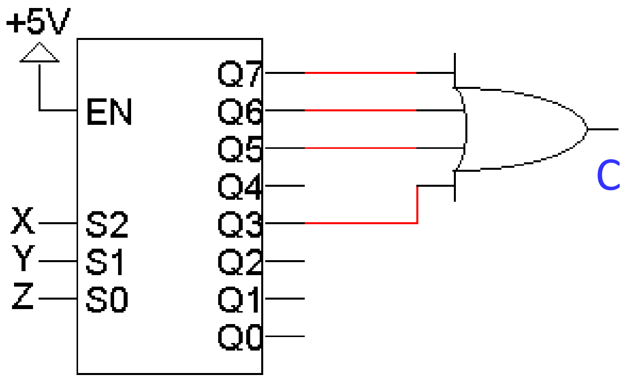
Implementing Functions using Decoders

Design example: addition

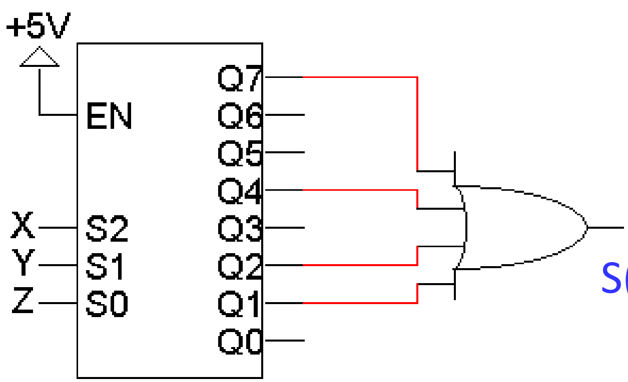
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$C(X,Y,Z) = \Sigma\ (3,5,6,7)$

$S(X,Y,Z) = \Sigma\ (1,2,4,7)$



$C(X,Y,Z) = \Sigma(3,5,6,7)$



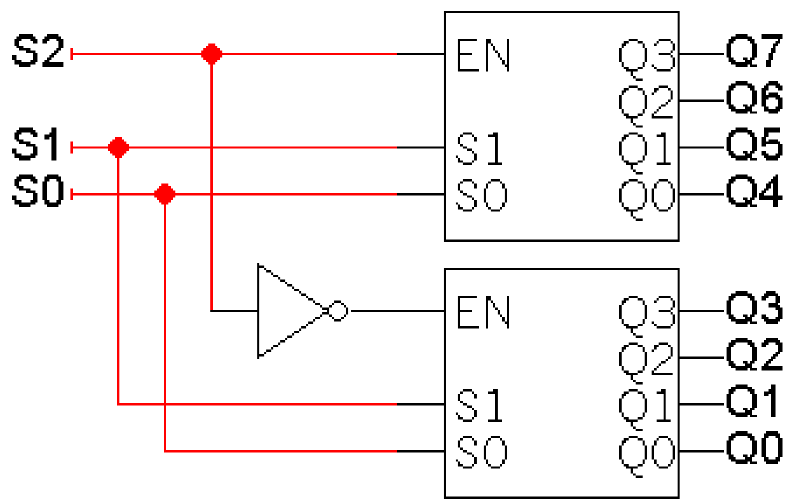
$S(X,Y,Z) = \Sigma(1,2,4,7)$

Decoder expansion

- Combine two or more small decoders with **enable** inputs to form a larger decoder.
- Here a 3-to-8 decoder has been constructed from **two** 2-to-4 decoders:

S2	S1	S0	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
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

EN	S1	S0	Q0	Q1	Q2	Q3
0	x	x	0	0	0	0
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



Use two 3 to 8 decoders to make 4 to 16 decoder

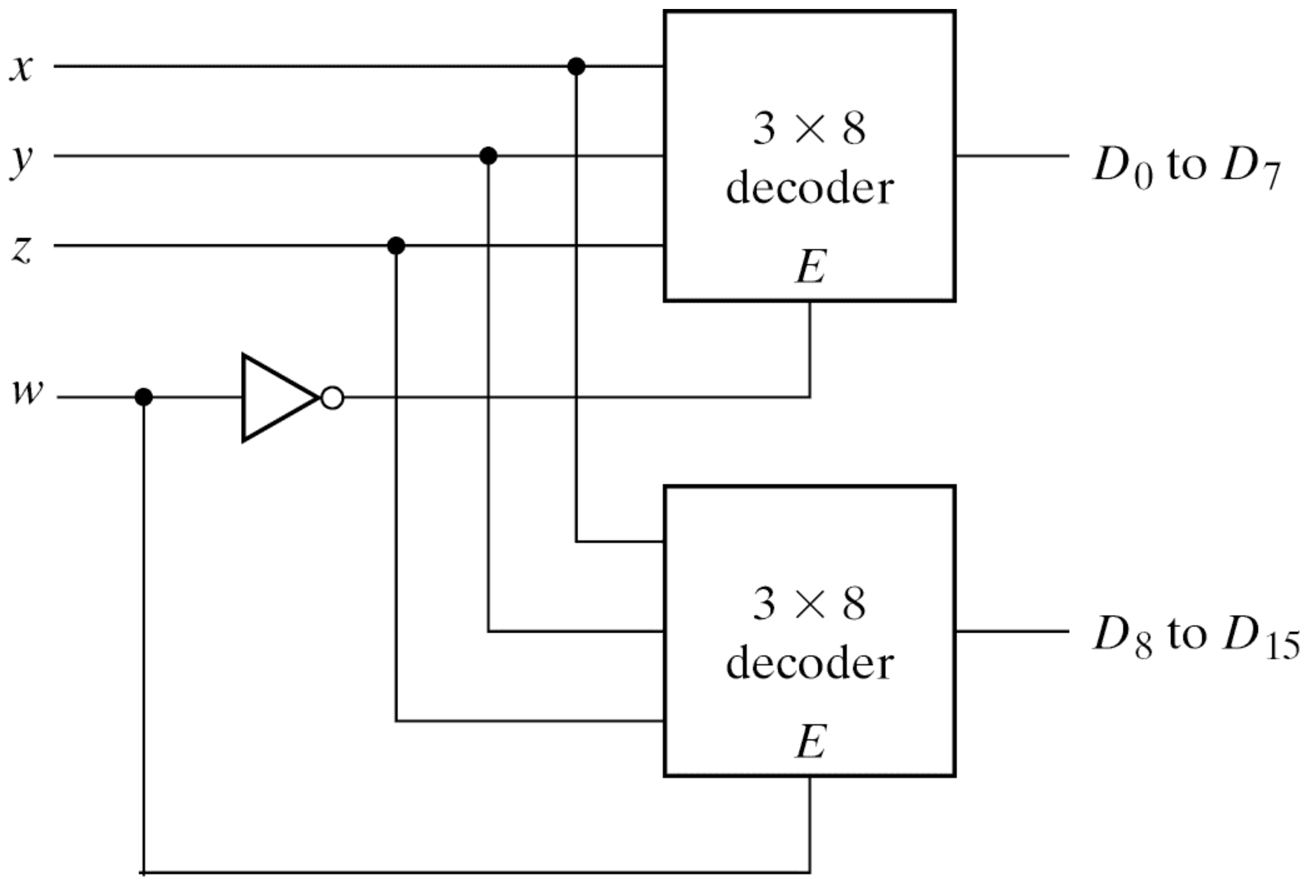
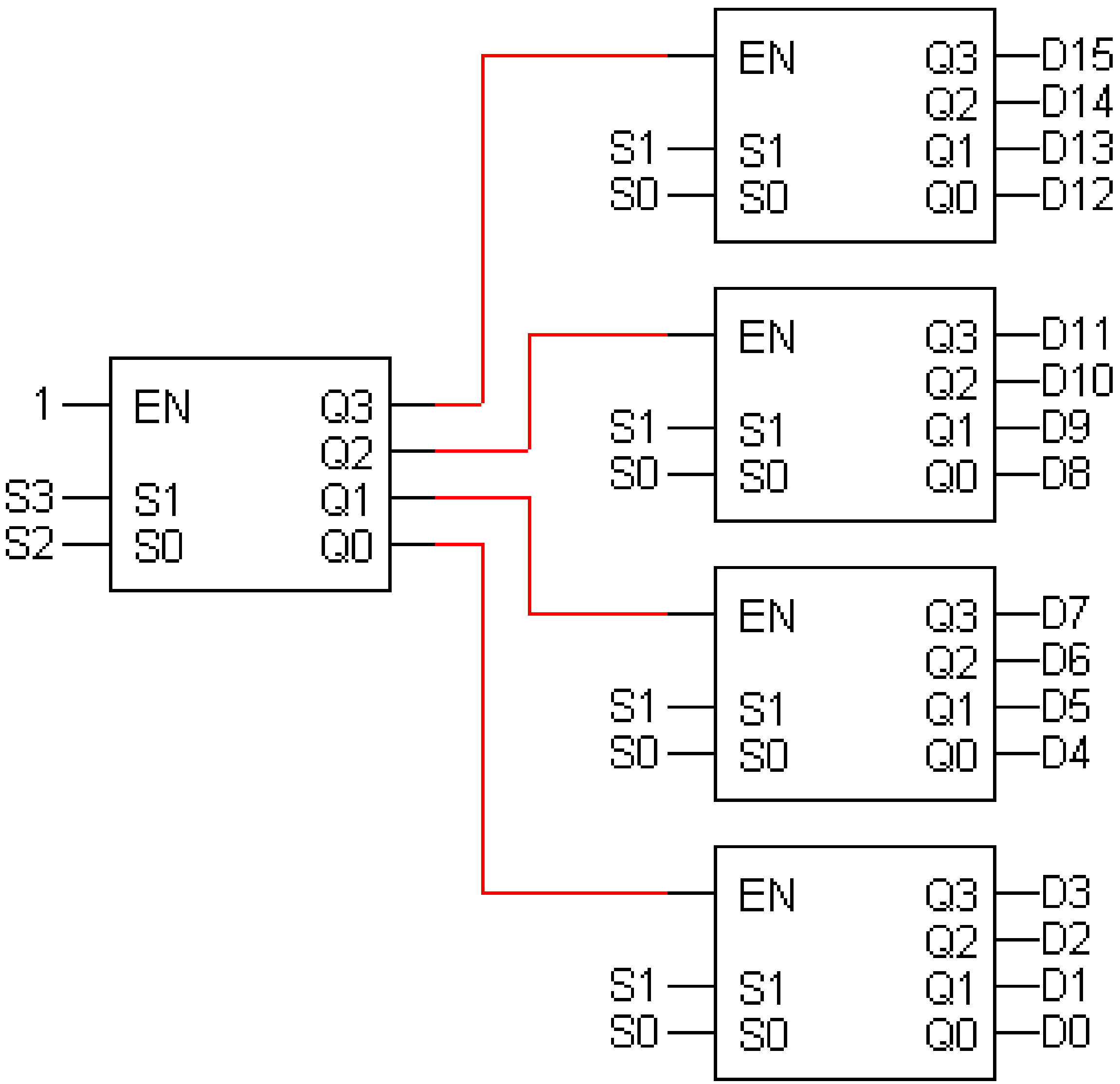


Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders

4-to-16 decoder using only 2-to-4 decoders (no gates)

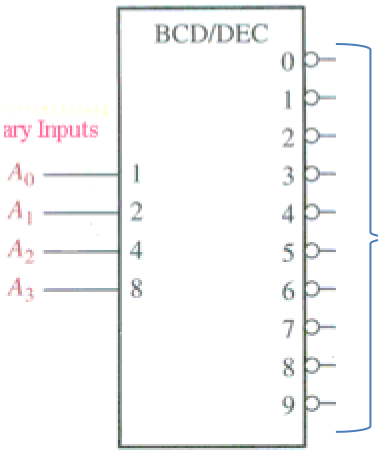
Input		Output			
S_1	S_0	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Input				Output
S_3	S_2	S_1	S_0	Q
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7
-	-	-	-	-



BCD-to-decimal decoder

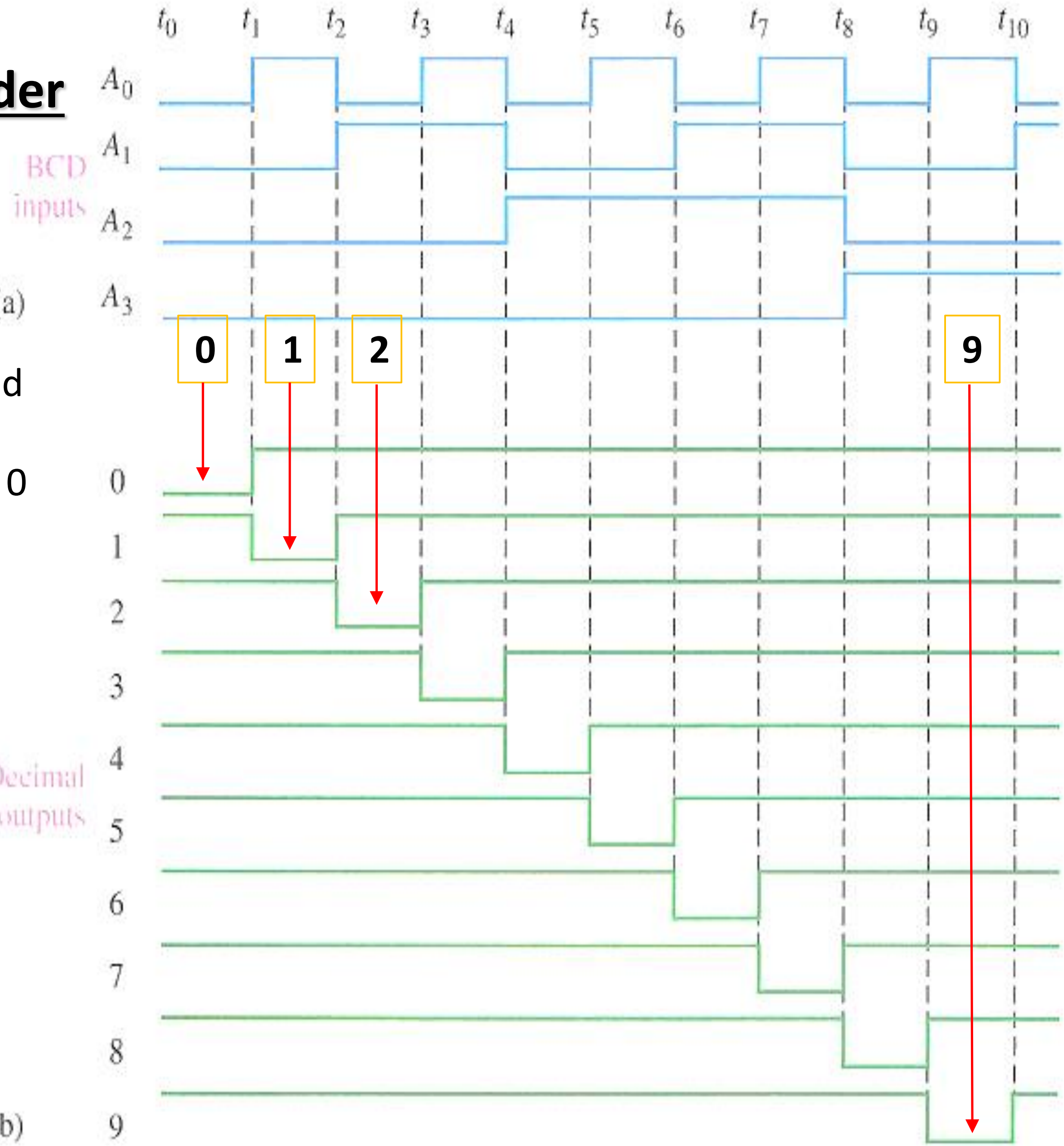
(Active Low Output)



Selected o/p will be 0

(a)

DECIMAL DIGIT	BCD CODE A_3	A_2	A_1	A_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

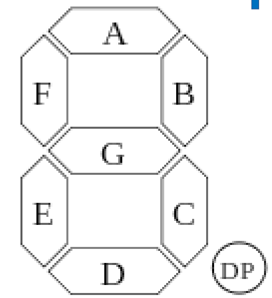


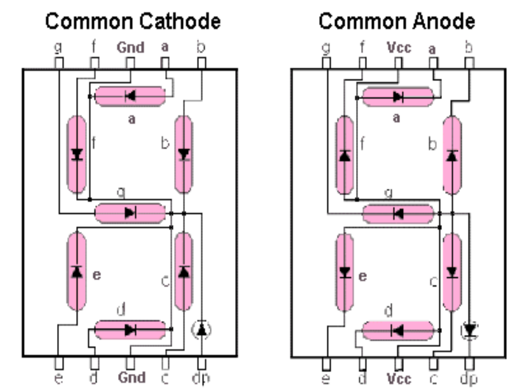
(b)

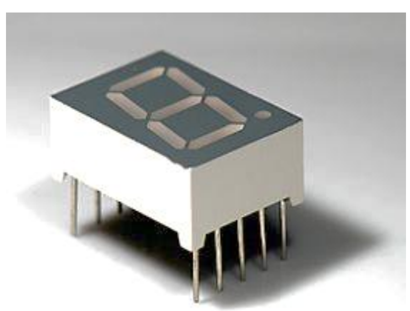
Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

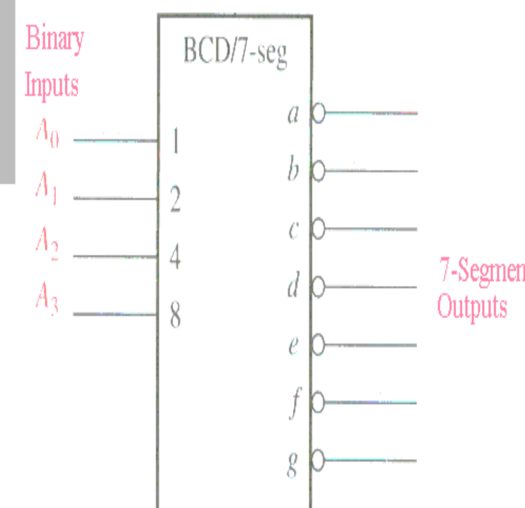
BCD-to-7-segement decoder

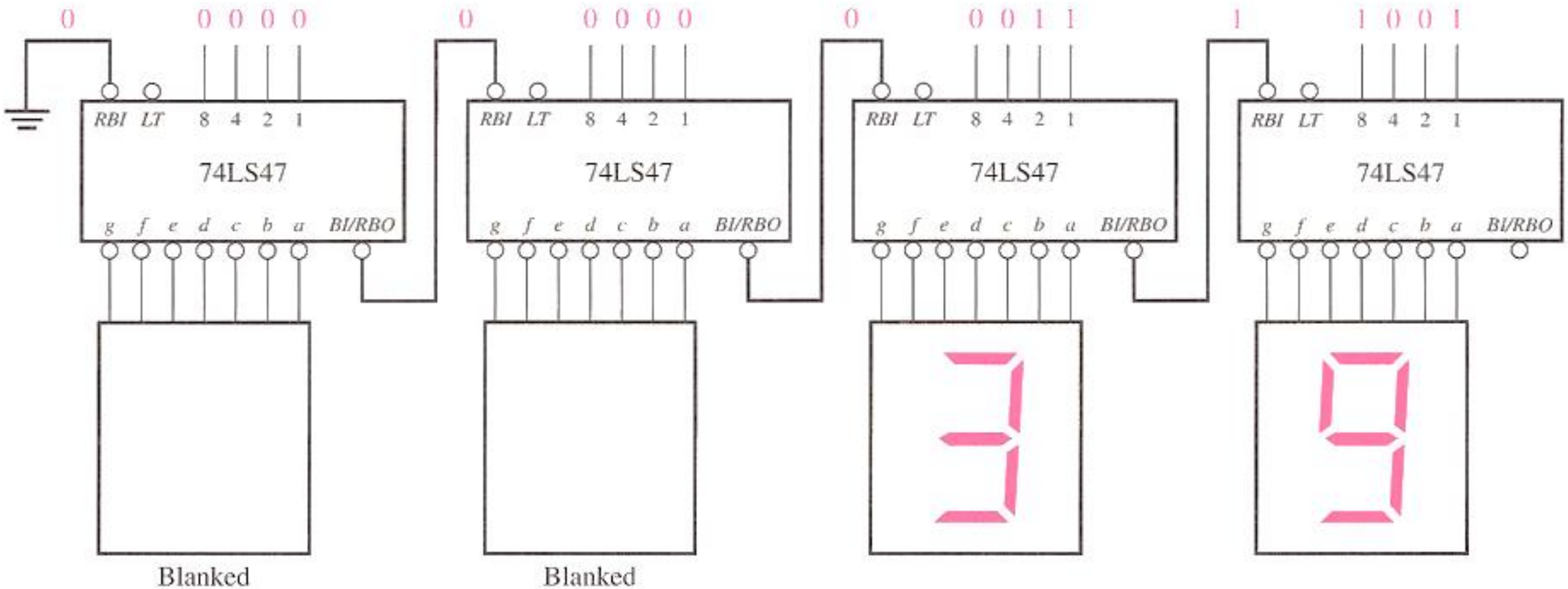
DECIMAL DIGIT	INPUTS				SEGMENT OUTPUTS						
	D	*C*	*B*	*A*	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X










0
0 0 0 0
RBI LT 8 4 2 1
74LS47
g f e d c b a BI/RBO
Blanked
0
0 0 0 0
RBI LT 8 4 2 1
74LS47
g f e d c b a BI/RBO
Blanked
0
0 0 1 1
RBI LT 8 4 2 1
74LS47
g f e d c b a BI/RBO
1
1 0 0 1
RBI LT 8 4 2 1
74LS47
g f e d c b a BI/RBO

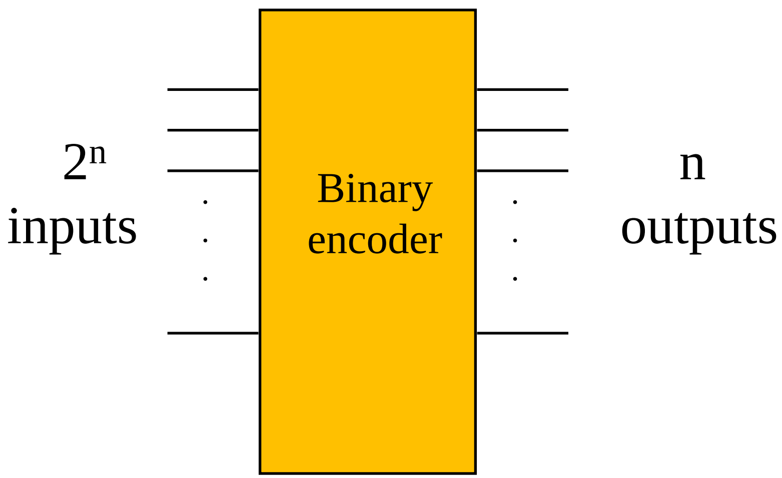
Encoders

An **Encoder** is a combinational logic circuit that performs a "**reverse**" decoder function.

An Encoder accepts an active level on one of ots inputs representing a digit, such as a decimal or octal digit, and converts it to a coded output, such as BCD or binary.

Encoders can also be devised to encode various symbols and alphabetic characters. The process of converting from familiar symbols or numbers to a coded format is called **Encoding**.

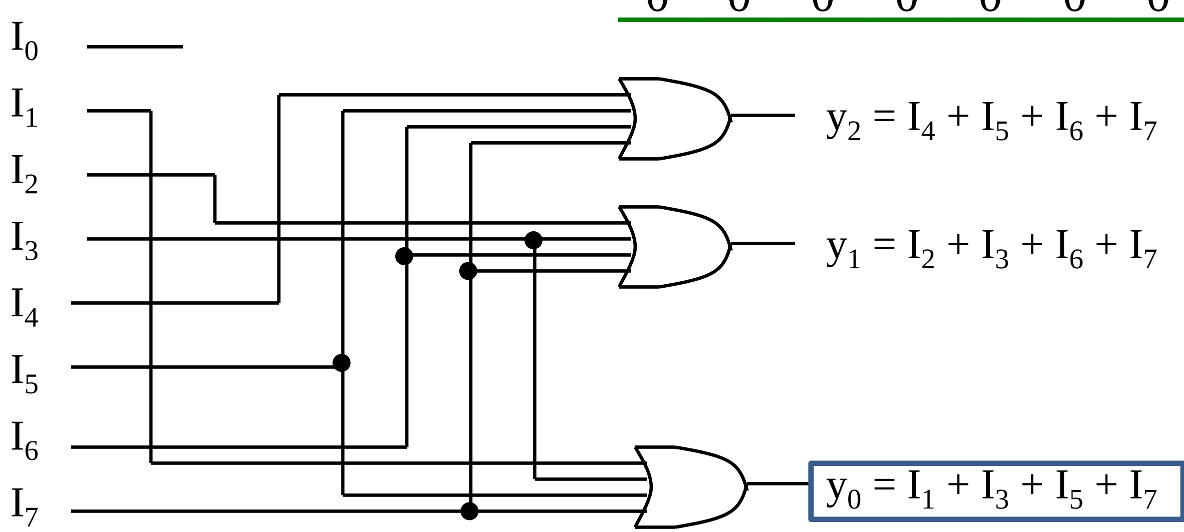
2^n-to-n Encoder:

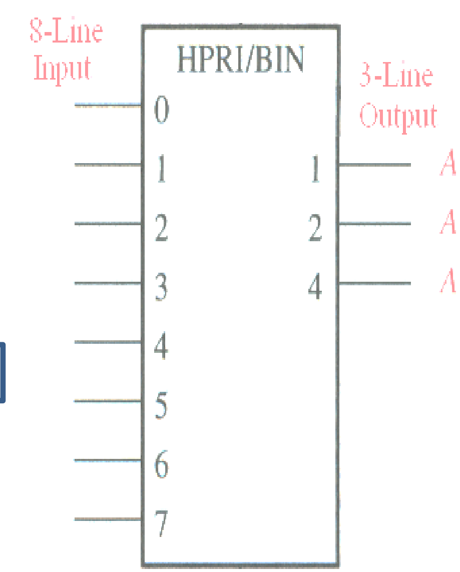


8-to-3 Binary Encoder

At any one time, only one input line has a value of 1.

Inputs								Outputs		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	y_2	y_1	y_0
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





8-to-3 Priority Encoder

- What if more than one input line has a value of 1?

Example:

- For the above mentioned problem, let's give priority to higher bits
- Ignore "lower priority" inputs.
- The sequence is:

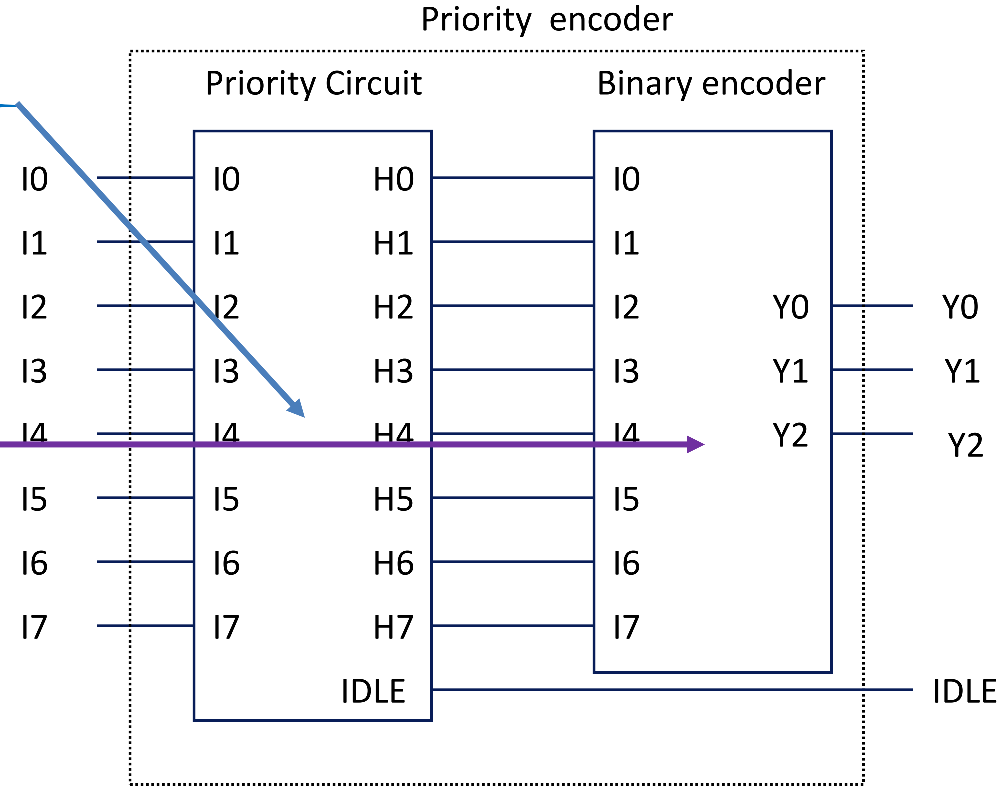
7>6>5>4>3>2>1>0

- Idle indicates that no input is a 1.

Inputs								Outputs			
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	y_2	y_1	y_0	Idle
0	0	0	0	0	0	0	0	x	x	x	1
1	0	0	0	0	0	0	0	0	0	0	0
X	1	0	0	0	0	0	0	0	0	1	0
X	X	1	0	0	0	0	0	0	1	0	0
X	X	X	1	0	0	0	0	0	1	1	0
X	X	X	X	1	0	0	0	1	0	0	0
X	X	X	X	X	1	0	0	1	0	1	0
X	X	X	X	X	X	1	0	1	1	0	0
X	X	X	X	X	X	X	1	1	1	1	0

Priority Encoder (8 to 3 encoder)

- **Assign priorities to the inputs**
- **When more than one inputs are asserted, the output generates the code of the input with the highest priority: 7>6>5>4>3>2>1>0**
- **Priority Encoder :**
  - H7=I7 (Highest Priority)
  - H6=I6.I7'
  - H5=I5.I6'.I7'
  - H4=I4.I5'.I6'.I7'
  - H3=I3.I4'.I5'.I6'.I7'
  - H2=I2.I3'.I4'.I5'.I6'.I7'
  - H1=I1. I2'.I3'.I4'.I5'.I6'.I7'
  - H0=I0.I1'. I2'.I3'.I4'.I5'.I6'.I7'
  - IDLE= I0'.I1'. I2'.I3'.I4'.I5'.I6'.I7'
- **Encoder**
  - Y0 = I1 + I3 + I5 + I7
  - Y1 = I2 + I3 + I6 + I7
  - Y2 = I4 + I5 + I6 + I7



Nafiz Ahmed Chisty| Head (UG) & Associate Professor, Dept. of EEE, FE, AIUB|

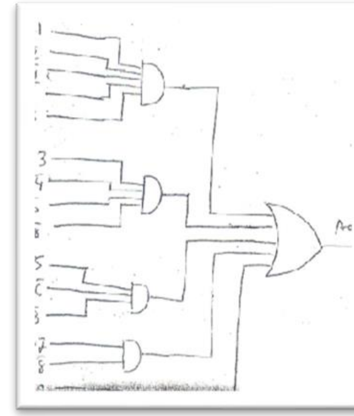
Optional

The Decimal to - BCD Priority Encder:

Let Priority is given to the higher order digits. Requirements to activate A0:

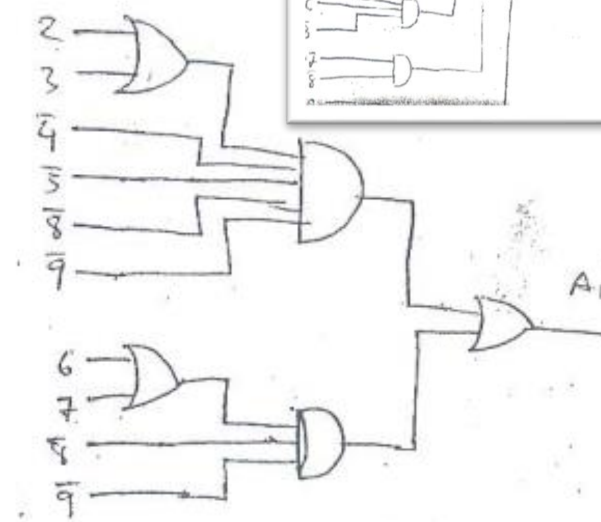
1) A_0 is HIGH if 1 is HIGH and 2,4,6,8 LOW
   A_0 is HIGH if 3 is HIGH and 4,6,8 LOW
   A_0 is HIGH if 5 is HIGH and 6,8 LOW
   A_0 is HIGH if 7 is HIGH and 8 LOW
   A_0 is HIGH if 9 is HIGH

Therefore, $A_0 = 1.2'.4'.6'.8' + 3.4'.6'.8' + 5.6'.8' + 7.8' + 9$



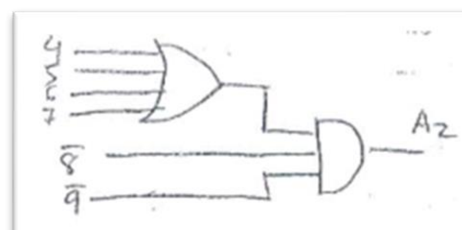
2) A_1 is HIGH if 2 is HIGH and 4,5,8,9 LOW
   A_1 is HIGH if 3 is HIGH and 4,5,8,9 LOW
   A_1 is HIGH if 6 is HIGH and 8,9 LOW
   A_1 is HIGH if 7 is HIGH and 8,9 LOW

Therefore, $A_1 = (2+3)4'.5'.8'.\ 9' + (6+7)8'.\ 9'$



3) A_2 is HIGH if 4 is HIGH and 8,9 LOW
   A_2 is HIGH if 5 is HIGH and 8,9 LOW
   A_2 is HIGH if 6 is HIGH and 8,9 LOW
   A_2 is HIGH if 7 is HIGH and 8,9 LOW

Therefore, $A_2 = (4+5+6+7)8'.\ 9'$



4) A_3 is HIGH if 8 &9 are HIGH

Therefore, $A_3 = 8+9$

Reference:

[1] Thomas L. Floyd, "Digital Fundamentals" 11th edition, Prentice Hall.
[2] M. Morris Mano, "Digital Logic & Computer Design" Prentice Hall.
[3] Mixed contents from Vahid And Howard.

Thanks