MSI Circuits

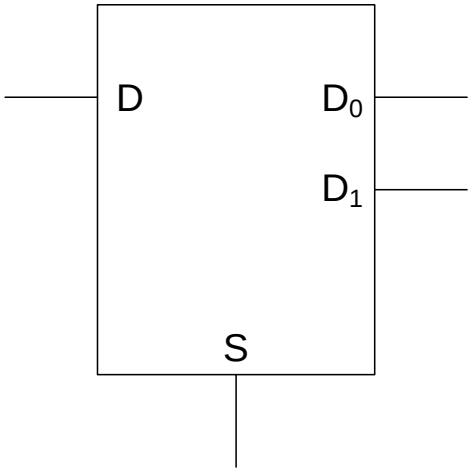
- Multiplexers (Mux)
  - 2x1, 4x1, and 8x1 muxes
  - 74x151, 74x153, 74x157 devices
- Demultiplexers (Demux), Decoders, and Encoders
  - 74x138 and 74x139 decoders
  - Encoder and priority encoder
  - BCD to 7-segment decoder and the 7447 device
  - Logic functions using muxes and decoders
- Adders and Comparators
  - Half, full and ripple carry adders
  - The 74x83 devices
  - Comparator and the 74x85 devices

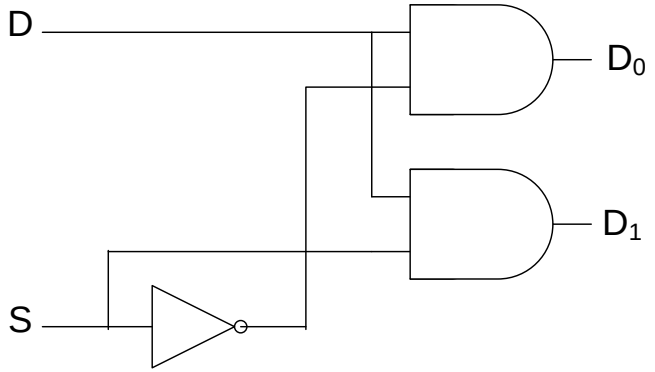
Demultiplexers (Demux)

- Reverse of the multiplexing function
- Takes data from one line and distributes to a given number of output lines
- Demux can be designed as 1x2, 1x4, 1x8, etc.

Demux (cont.)

- 1x2 Demux





Function:

D_0 = D, D_1 = 0 when S = 0

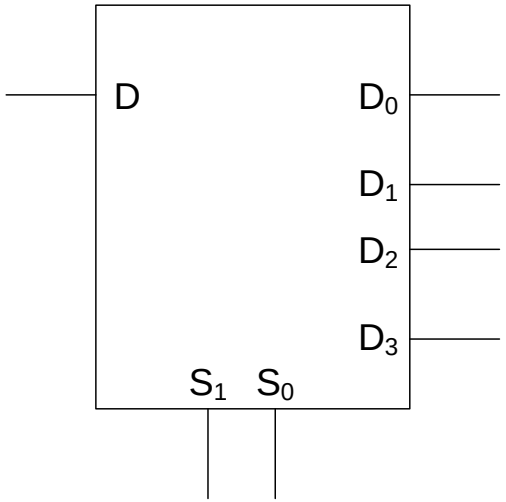
D_1 = D, D_0 = 0 when S = 1

$$D_0 = D \cdot \overline{S}$$

$$D_1 = D \cdot S$$

Demux (cont.)

- 1x4 Demux



$$D_0 = D \cdot \overline{S_1} \cdot \overline{S_0}$$
$$D_1 = D \cdot \overline{S_1} \cdot S_0$$
$$D_2 = D \cdot S_1 \cdot \overline{S_0}$$
$$D_3 = D \cdot S_1 \cdot S_0$$

Can you draw the logic circuit?

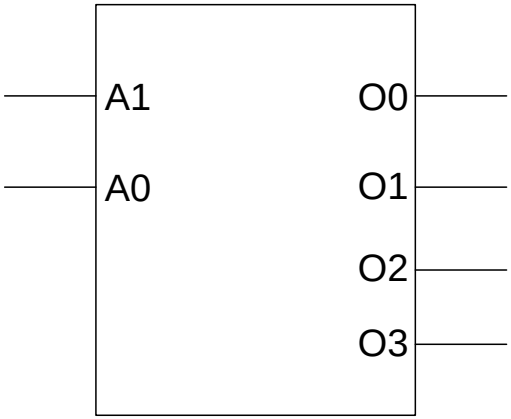
Data at D is distributed to D_0 to D_3 depending on select S_1 and S_0

Decoders

- Detect the presence of a specified combination of bits (code) on its inputs and to indicate the presence of that code by a specified output level
- Standard decoders are designed as 1x2, 2x4, 3x8, 4x16, etc
- Other decoders include the BCD to 7-segment LED decoder, gray code decoder, etc.

Decoders

- 2x4 Decoder: only one output is active at one time



Active high output decoder

Function:

$O_3O_2O_1\mathbf{O_0}$ = 000**1** when $\mathbf{A_1A_0 = 00}$ $\longrightarrow$ $O_0 = \overline{A_1} \cdot \overline{A_0}$

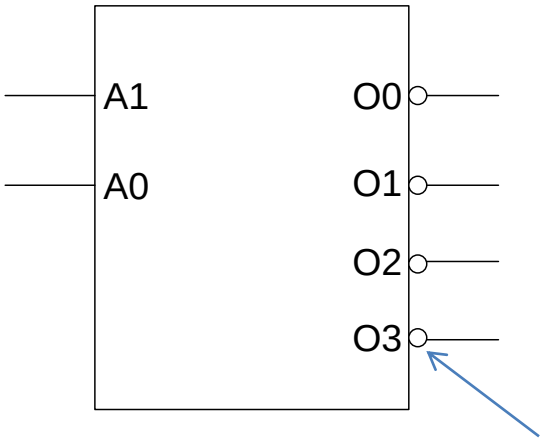
$O_3O_2\mathbf{O_1}O_0$ = 00**1**0 when $\mathbf{A_1A_0 = 01}$ $\longrightarrow$ $O_1 = A_1 \cdot \overline{A_0}$

$O_3\mathbf{O_2}O_1O_0$ = 0**1**00 when $\mathbf{A_1A_0 = 10}$ $\longrightarrow$ $O_2 = A_1 \cdot A_0$

$\mathbf{O_3}O_2O_1O_0$ = **1**000 when $\mathbf{A_1A_0 = 11}$ $\longrightarrow$ $O_3 = A_1 \cdot A_0$

Decoders (cont.)

- Decoders are typically designed as active low



Bubble at output denotes active low output

Function:

$O_3O_2O_1\mathbf{O_0}$ = 1110 when $\mathbf{A_1A_0 = 00}$ $\longrightarrow$ $O_0 = A_1 + \overline{A_0}$

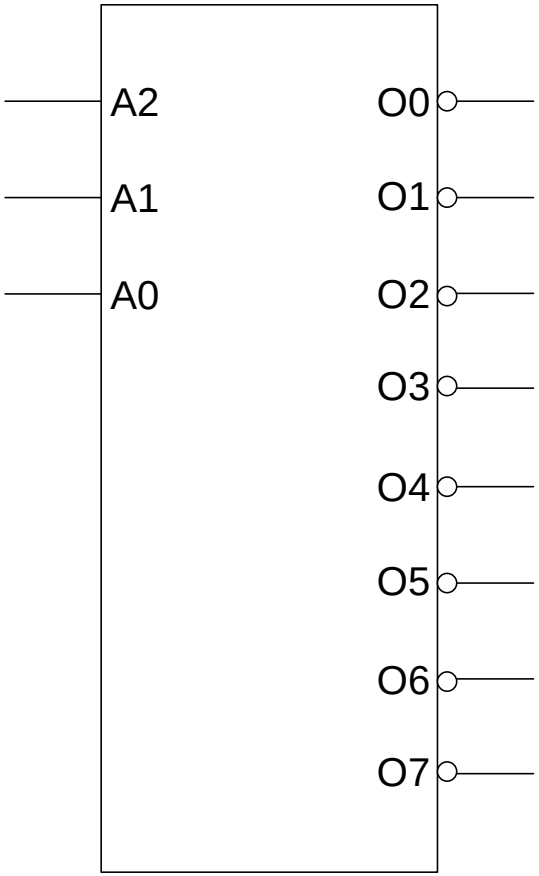
$O_3O_2\mathbf{O_1}O_0$ = 1101 when $\mathbf{A_1A_0 = 01}$ $\longrightarrow$ $O_1 = \overline{A_1} + A_0$

$O_3\mathbf{O_2}O_1O_0$ = 1011 when $\mathbf{A_1A_0 = 10}$ $\longrightarrow$ $O_2 = \overline{A_1} + \overline{A_0}$

$\mathbf{O_3}O_2O_1O_0$ = 0111 when $\mathbf{A_1A_0 = 11}$ $\longrightarrow$ $O_3 = \overline{A_1} + \overline{A_0}$

Decoders (cont.)

- 3x8 Decoder



Function:

$O_7O_6O_5O_4O_3O_2O_1O_0$ = 1111 1110 when $A_2A_1A_0$ = 000
$O_7O_6O_5O_4O_3O_2O_1O_0$ = 1111 1101 when $A_2A_1A_0$ = 001
$O_7O_6O_5O_4O_3O_2O_1O_0$ = 1111 1011 when $A_2A_1A_0$ = 010
$O_7O_6O_5O_4O_3O_2O_1O_0$ = 1111 0111 when $A_2A_1A_0$ = 011
$O_7O_6O_5O_4O_3O_2O_1O_0$ = 1110 1111 when $A_2A_1A_0$ = 100
$O_7O_6O_5O_4O_3O_2O_1O_0$ = 1101 1111 when $A_2A_1A_0$ = 101
$O_7O_6O_5O_4O_3O_2O_1O_0$ = 1011 1111 when $A_2A_1A_0$ = 110
$O_7O_6O_5O_4O_3O_2O_1O_0$ = 0111 1111 when $A_2A_1A_0$ = 111

How does the logic circuit looks like?
- Can we use K-Map?

Decoders (cont.)

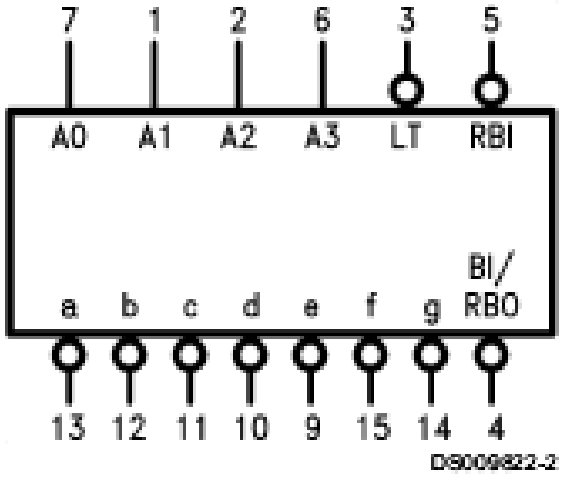
- Decoder IC's
  - 2x4 Decoder ➔ 74139
  - 3x8 Decoder ➔ 74138

BCD to 7 Segment Decoder

- 74247 BCD to 7 Segment Decoder

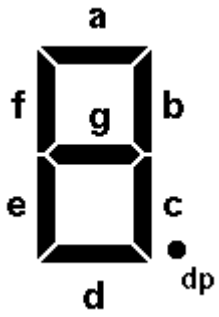
Converts 4-bit BCD (A_3, A_2, A_1, A_0) to 7-segment LED (a,b,c,d,e,f,g)

Logic Symbol



V_{CC} = Pin 16
GND = Pin 8

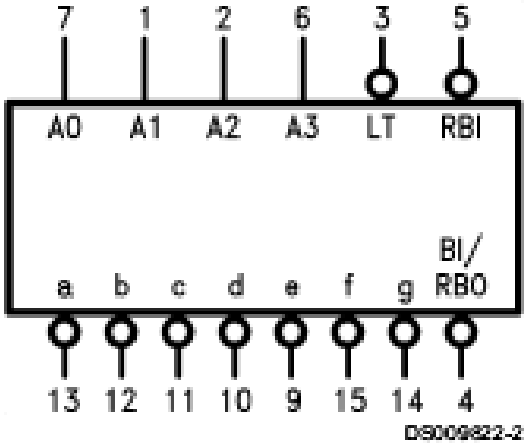
Input: A_3, A_2, A_1, A_0
Control Input: LT, RBI, RBO
Output: a, b, c, d, e, f, g
**** note that output is active low**



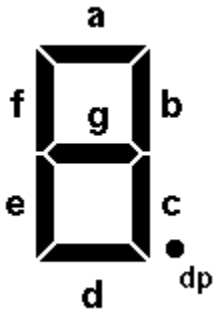
BCD to 7 Segment Decoder (cont.)

- What is the output of 74247 if input $(A_3A_2A_1A_0)$=0110?

Logic Symbol



V_{CC} = Pin 16
GND = Pin 8

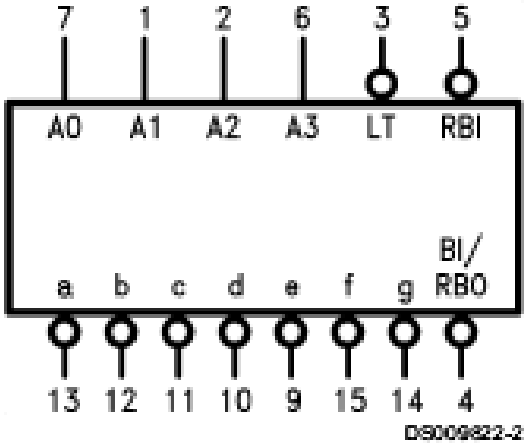


a b c d e f g
0 1 0 0 0 0 0

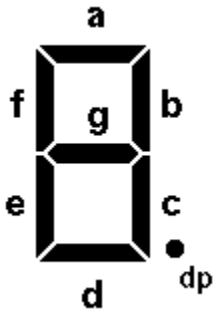
BCD to 7 Segment Decoder (cont.)

- What is the output of 74247 if input $(A_3A_2A_1A_0)$=0011?

Logic Symbol



V_{CC} = Pin 16
GND = Pin 8

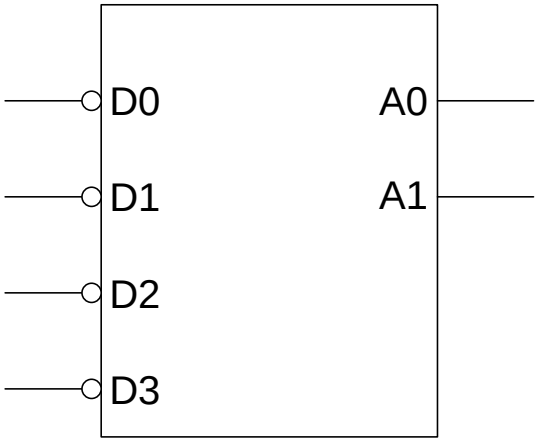


a	b	c	d	e	f	g
0	0	0	0	1	1	0

Encoders

- Performs reverse decoder function
- Only one input can be active at one time

4x2 Encoder



Function:

$A_1A_0 = 00$ when $D_3D_2D_1D_0 = 1110$

$A_1A_0 = 01$ when $D_3D_2D_1D_0 = 1101$

$A_1A_0 = 10$ when $D_3D_2D_1D_0 = 1011$

$A_1A_0 = 11$ when $D_3D_2D_1D_0 = 0111$

Which implies,

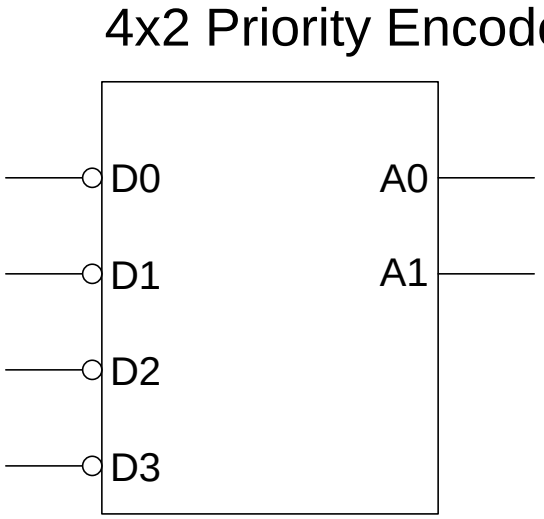
$$A_1 = D_3\overline{D_2}D_1D_0 + \overline{D_3}D_2D_1D_0$$

$$A_0 = D_3D_2\overline{D_1}D_0 + \overline{D_3}D_2D_1D_0$$

What happens if more than one input is '0' ($D_0 = 0$ and $D_1 = 0$)?
- output is invalid, we need a priority encoder

Encoders (cont.)

- Priority Encoder: Output depends on the largest active input



Function:

$A_1A_0 = 00$ when $D_3D_2D_1D_0 = 1110$

$A_1A_0 = 01$ when $D_3D_2D_1D_0 = 110x$

$A_1A_0 = 10$ when $D_3D_2D_1D_0 = 10xx$

$A_1A_0 = 11$ when $D_3D_2D_1D_0 = 0xxx$

Can you derive the truth table of the priority encoder?

What happens if more than 1 input is '0' ($D_0 = 0$ and $D_1 = 0$)?

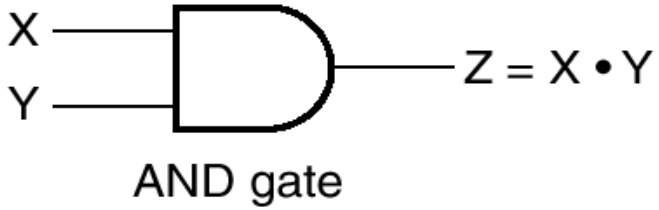
- output $A_1A_0 = 01$

Logic function using Mux

- Any logic function (AND, OR, decoder, encoder, etc) can be realized using mux. All we need is a truth table
- This is the concept widely being used today in FPGA (Field Programmable Gate Array), where any logic functions can be rapidly implemented using Mux

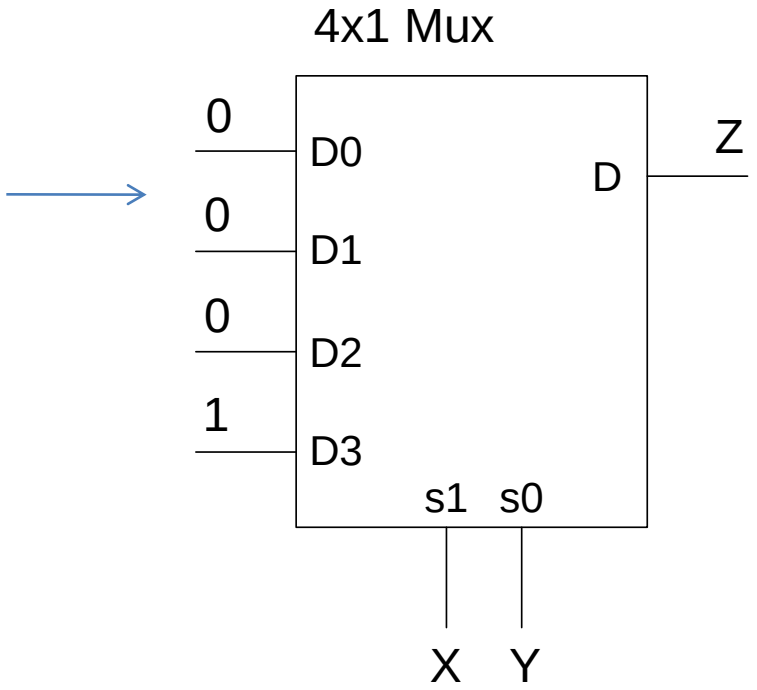
Logic function using Mux (cont.)

- 2-input AND gate using 4x1 Mux



AND gate

X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

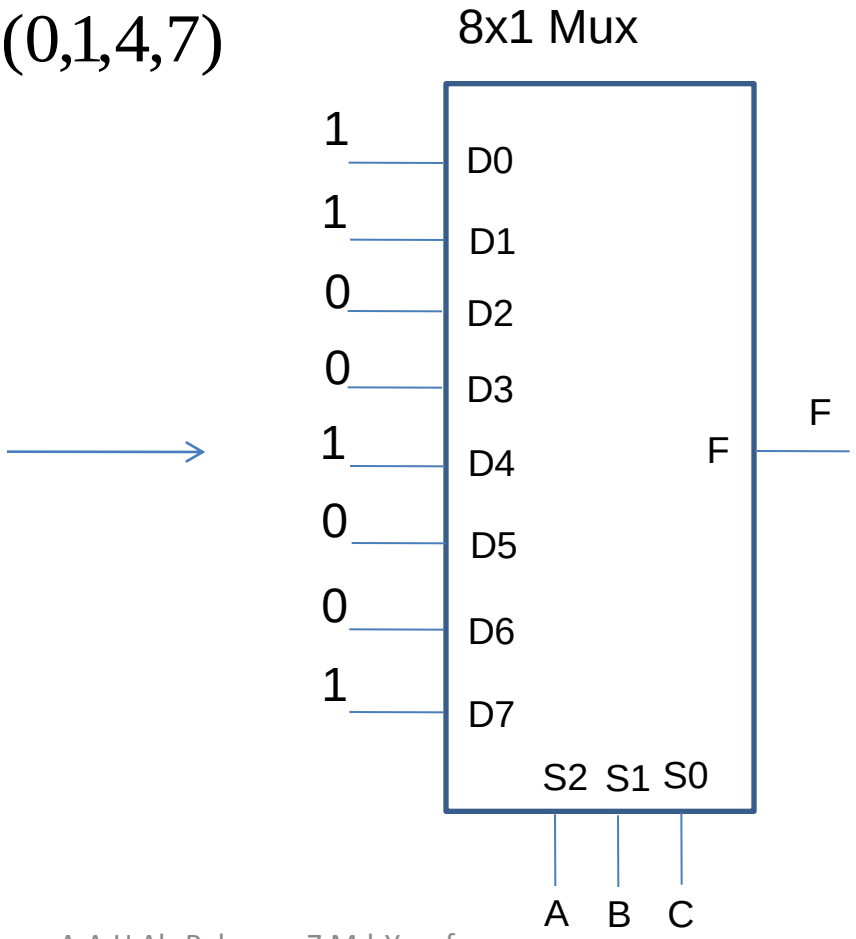


Logic function using Mux (cont.)

- Implement the following using Mux

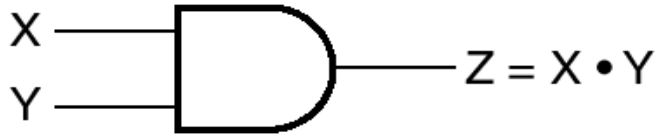
$$F(A,B,C) = \sum m(0,1,4,7)$$

A B C	F
0 0 0	1
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1



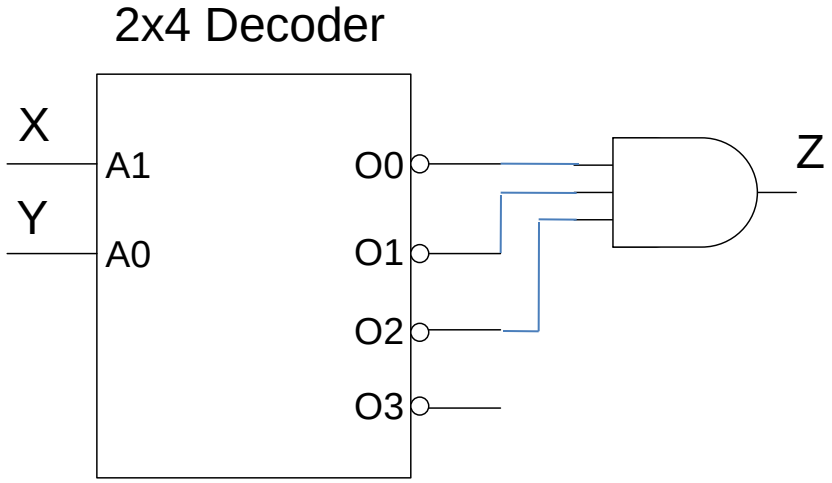
Logic function using Decoder

- Similar to Mux, any logic function can be realized using decoders
- 2-input AND gate using 2x4 decoder



AND gate

X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

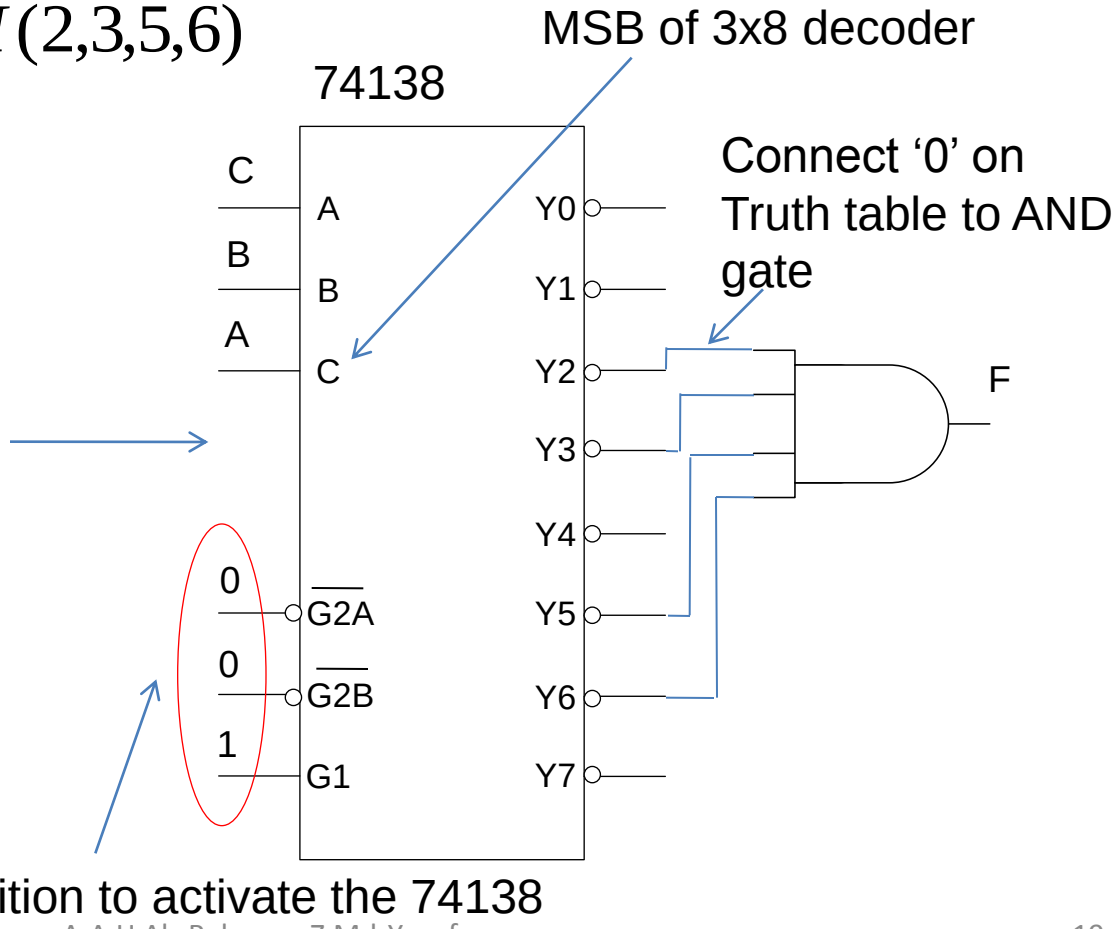


Logic function using Decoder (cont.)

- Implement the following using 74138

$$F(A,B,C) = \prod M(2,3,5,6)$$

A B C	F
0 0 0	1
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1



Logic function using Decoder(cont.)

- Logic function of decoders can also be implemented using a NAND gate

$$F(A,B,C) = \prod M(2,3,5,6)$$

A B C	F
0 0 0	1
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1