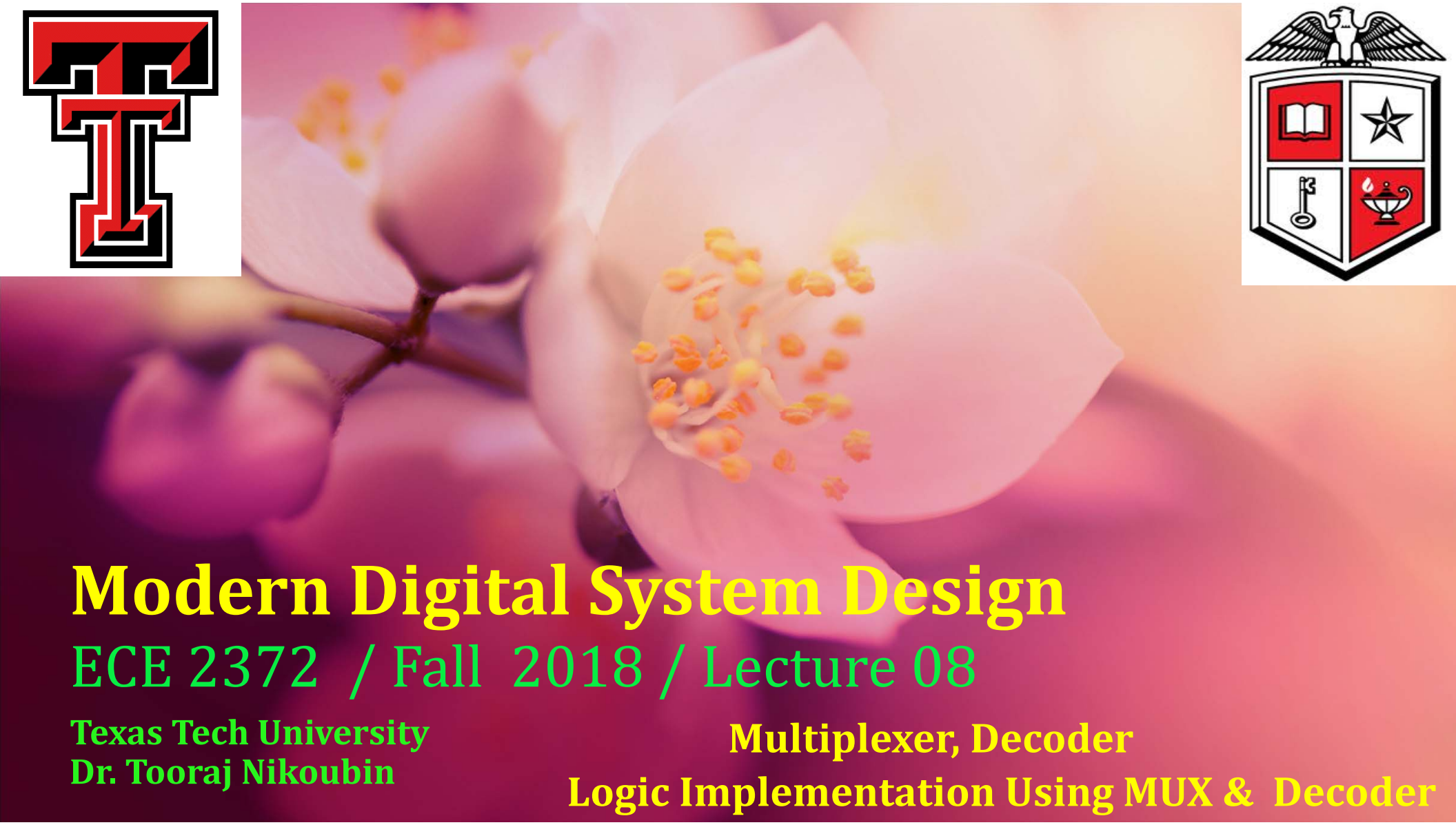
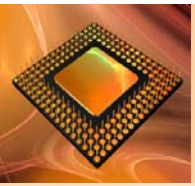


Modern Digital System Design

ECE 2372 / Fall 2018 / Lecture 08

Texas Tech University
Dr. Tooraj Nikoubin

Multiplexer, Decoder
Logic Implementation Using MUX & Decoder

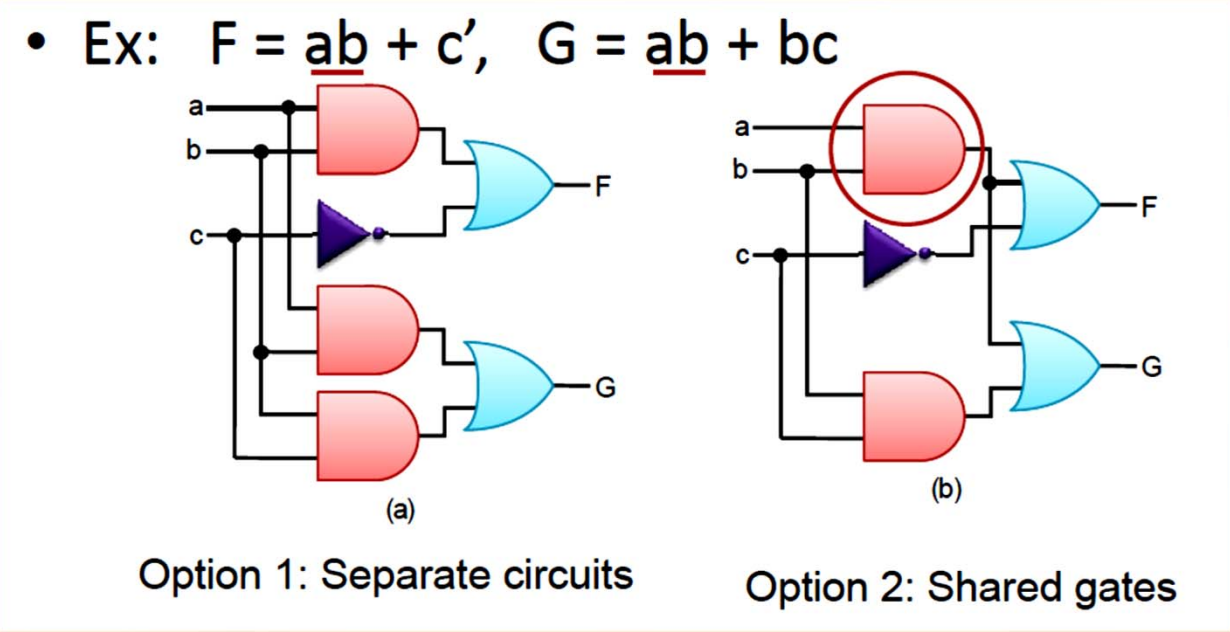


Outline

Study of Components
– Multiplexor, Decoder
– Logic Implementation Using MUX & Decoder

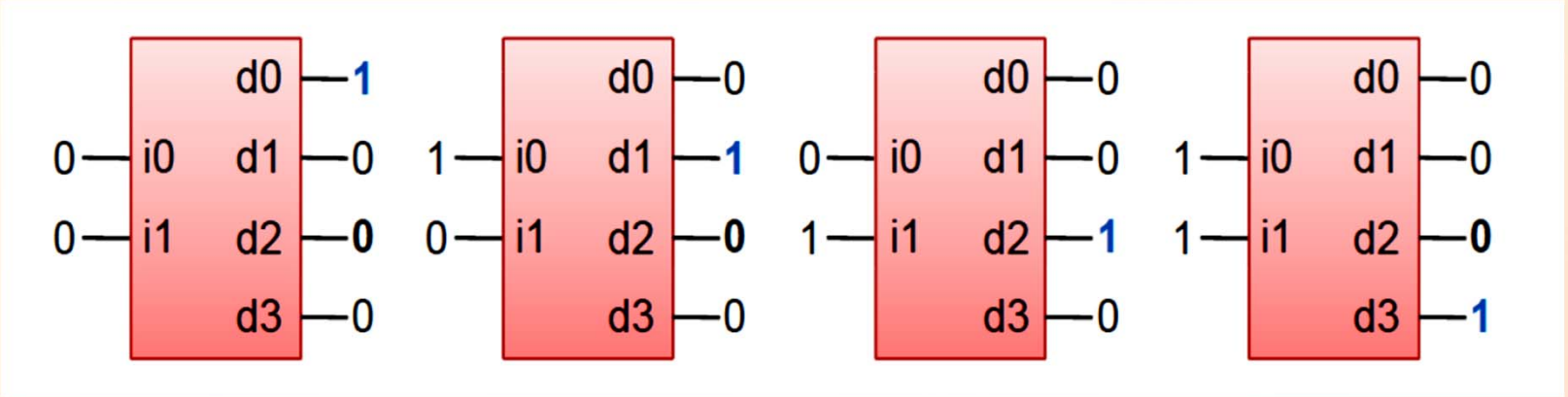
Multiple-Output Circuits

- Many circuits have more than one output
- Can give each a separate circuit, or can share gates
- Ex: F = ab + c', G = ab + bc

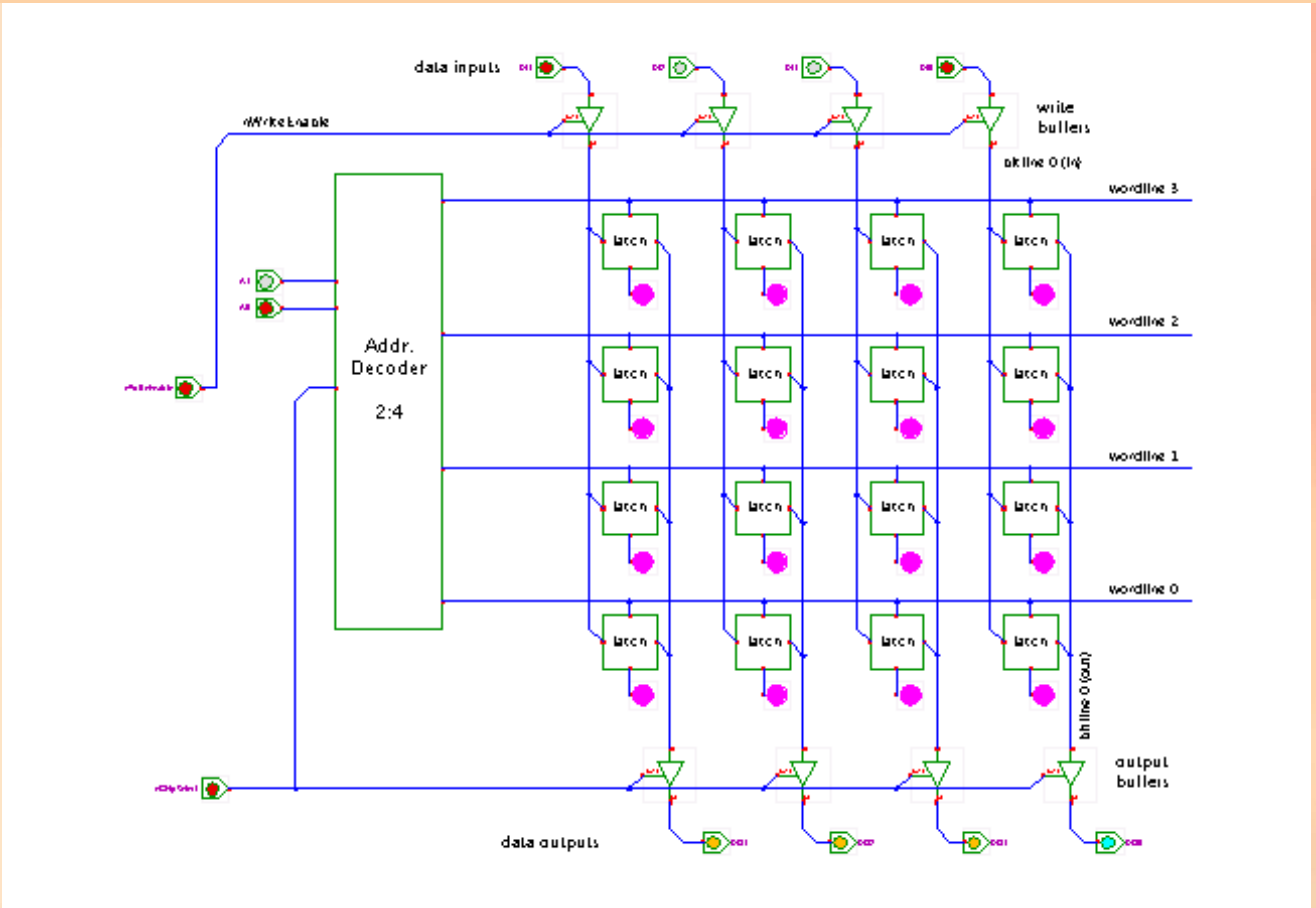


Decoder

- Decoder : knows what to do with this: Decode
- N input: 2N output
- Memory Addressing
  - Address to a particular location
- **Decoder**: Popular combinational logic building block, in addition to logic gates
  - Converts input binary number to one high output
- 2-input decoder: four possible input binary numbers
  - So has four outputs, one for each possible input binary number

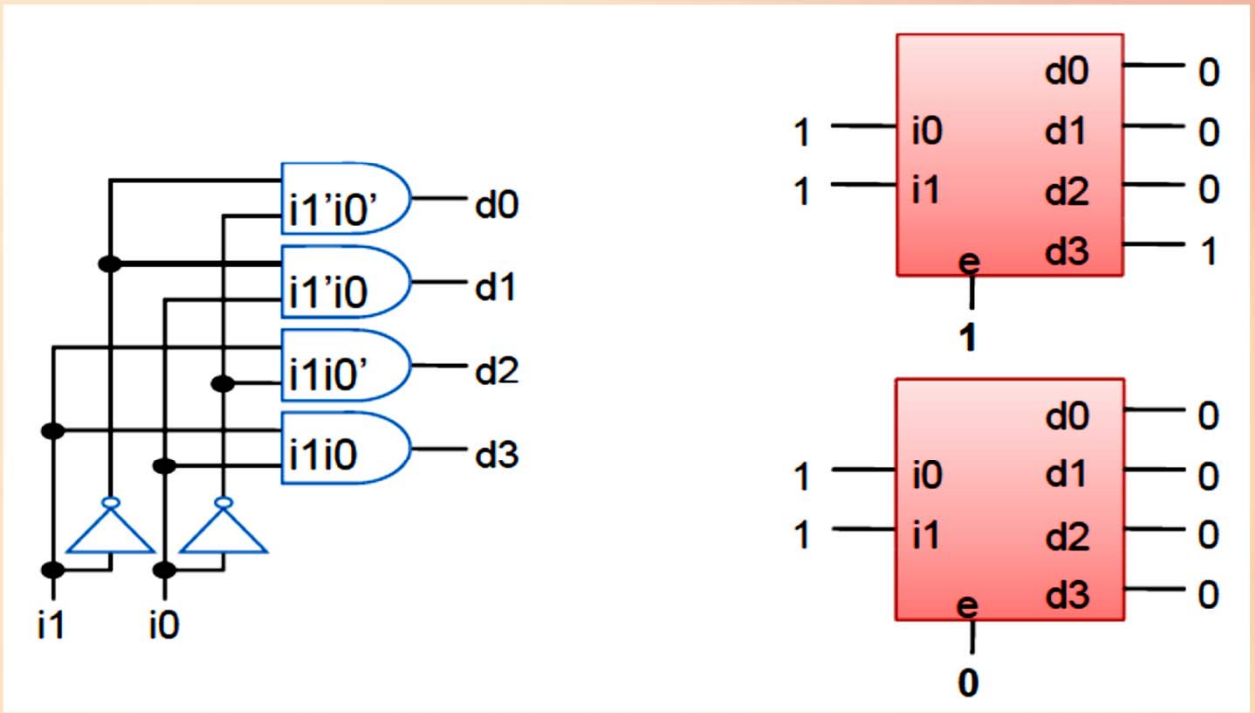


Decoder in Memory structures

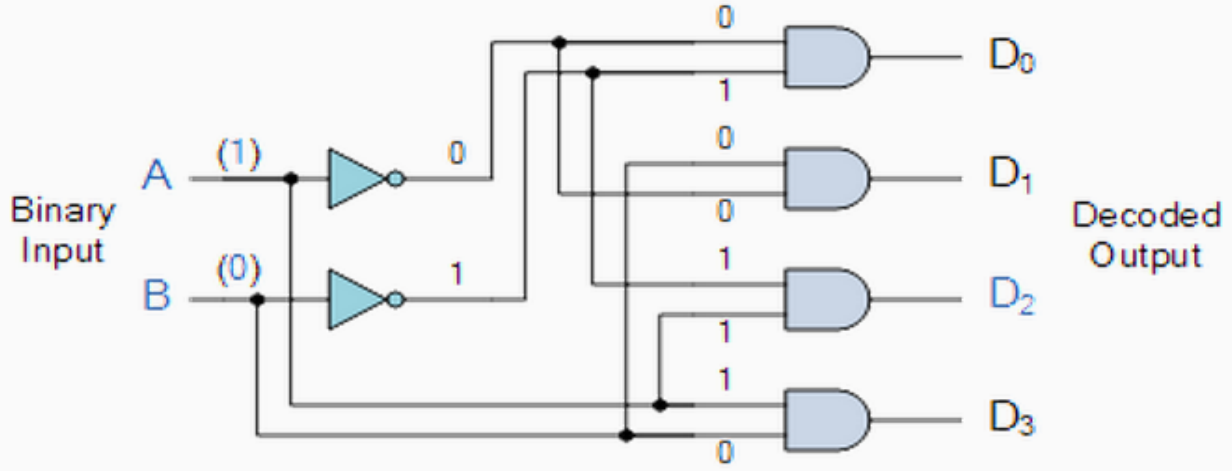


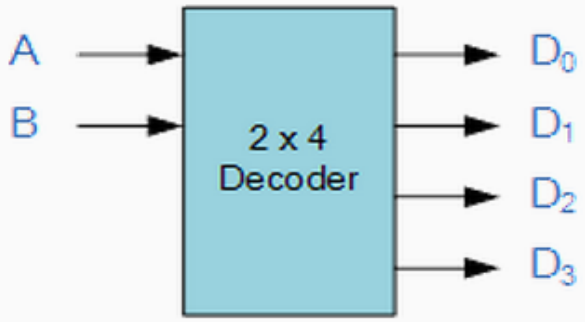
Decoders and Multiplexers (Muxes)

- Internal design
  - AND gate for each output to detect input combination
- Decoder with enable e
  - Outputs all 0 if e=0, Regular behavior if e=1
- n-input decoder: 2n outputs



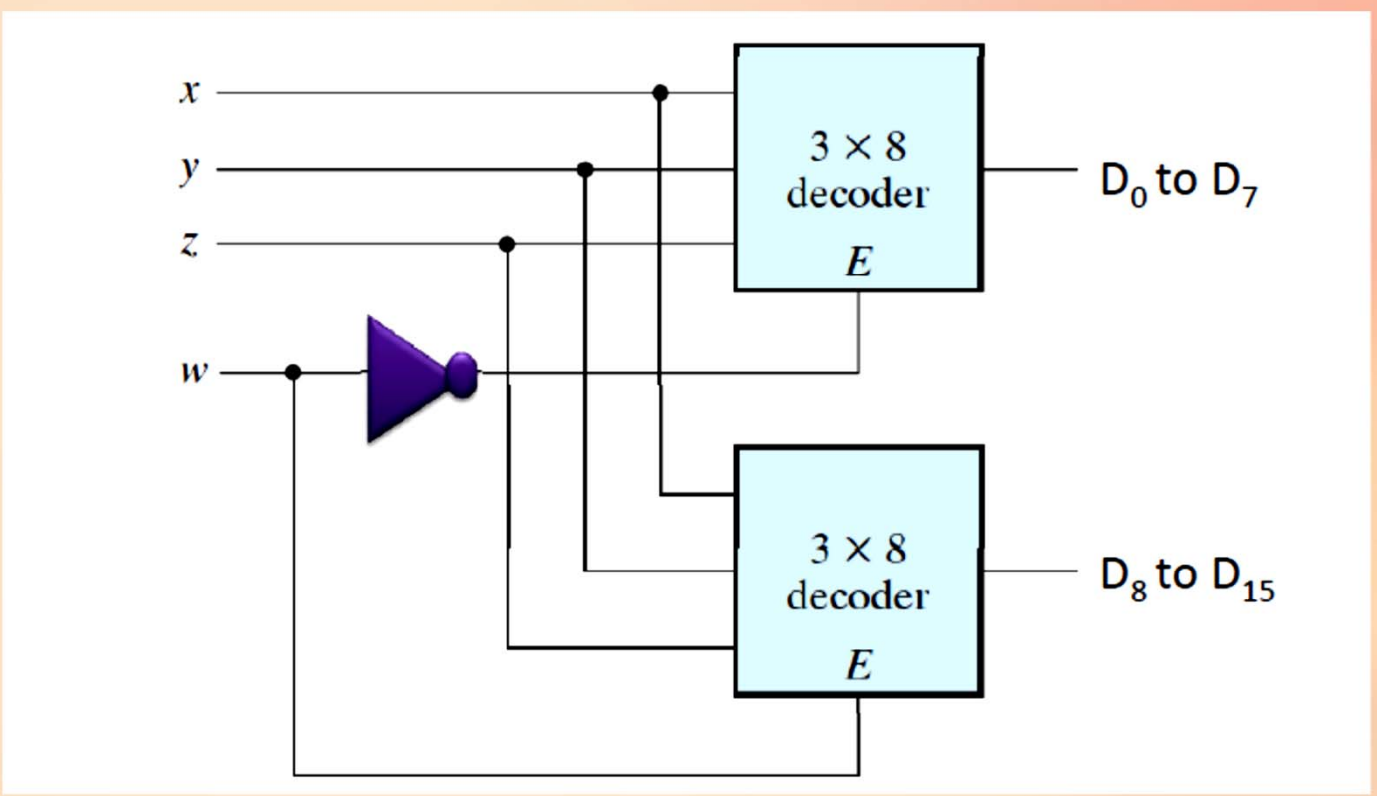
Decoder

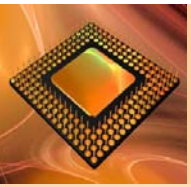




A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

4-to-16 Decoder using two 3-to-8 Decoders





Boolean Function Implementation using Decoders

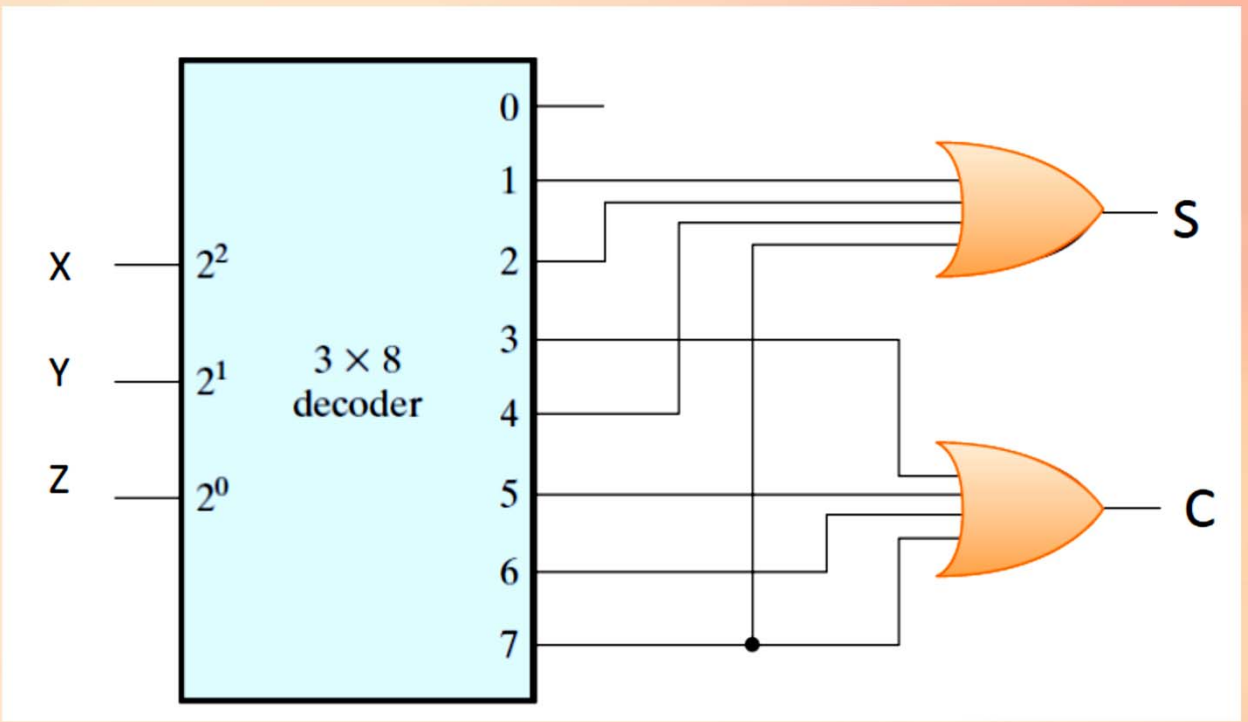
- Using a n-to-2n decoder and OR gates any functions of n variables can be implemented.

- Example:

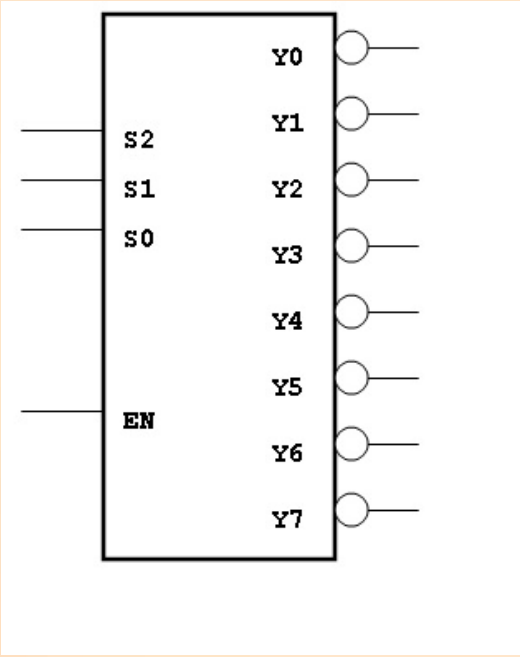
$S(x,y,z)= \Sigma(1,2,4,7)$, $C(x,y,z)=\Sigma(3,5,6,7)$

- Functions S and C can be implemented using a 3-to-8 decoder and two 4-input OR gates

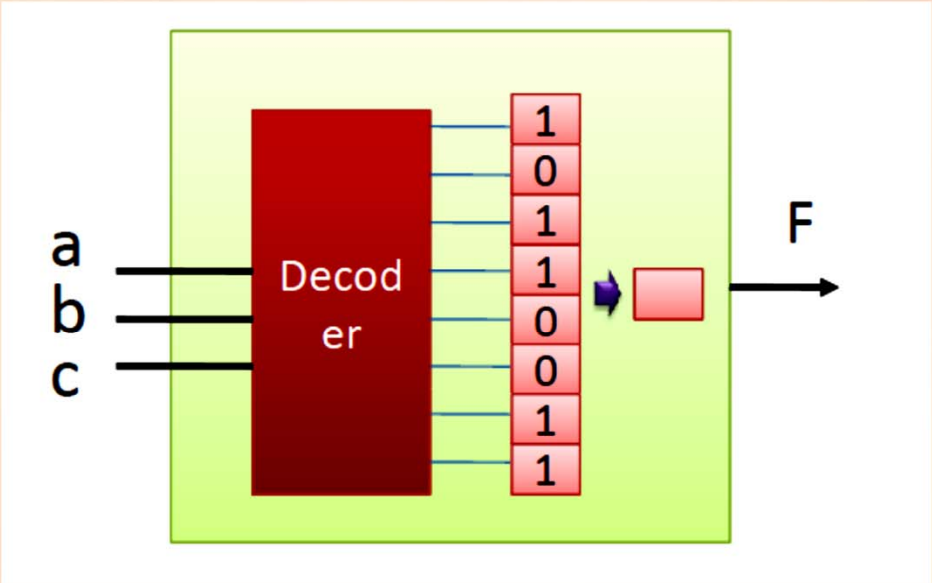
Implementation of S and C



(Active Low Outputs)Decoder



Configurable Function using Decoder & Memory

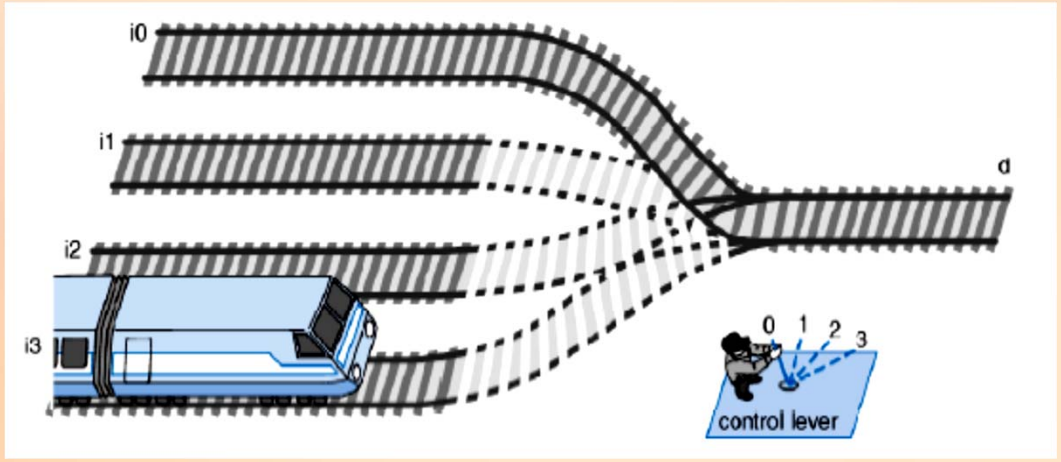


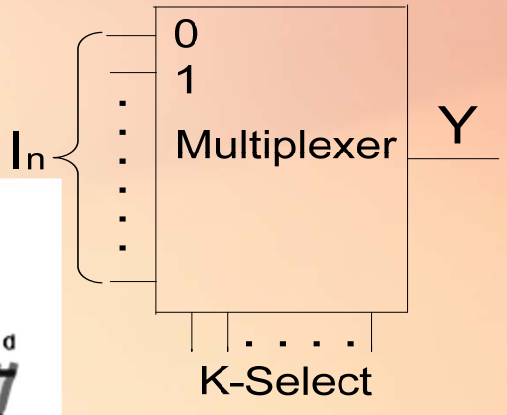
Any Function can be implemented

Function depend on Memory Elements, Direct correspondence to Truth Table

Multiplexor (Mux)

- Mux: Another popular combinational building block
  - Routes one of its N data inputs to its one output, based on binary value of select inputs
- **4** input mux → needs **2** select inputs to indicate which input to route through
- **8** input mux → **3** select inputs
- **N** inputs → Log_2 (**N**) selects
  - Like a rail yard switch





$$Mux\ 2 \times 1$$

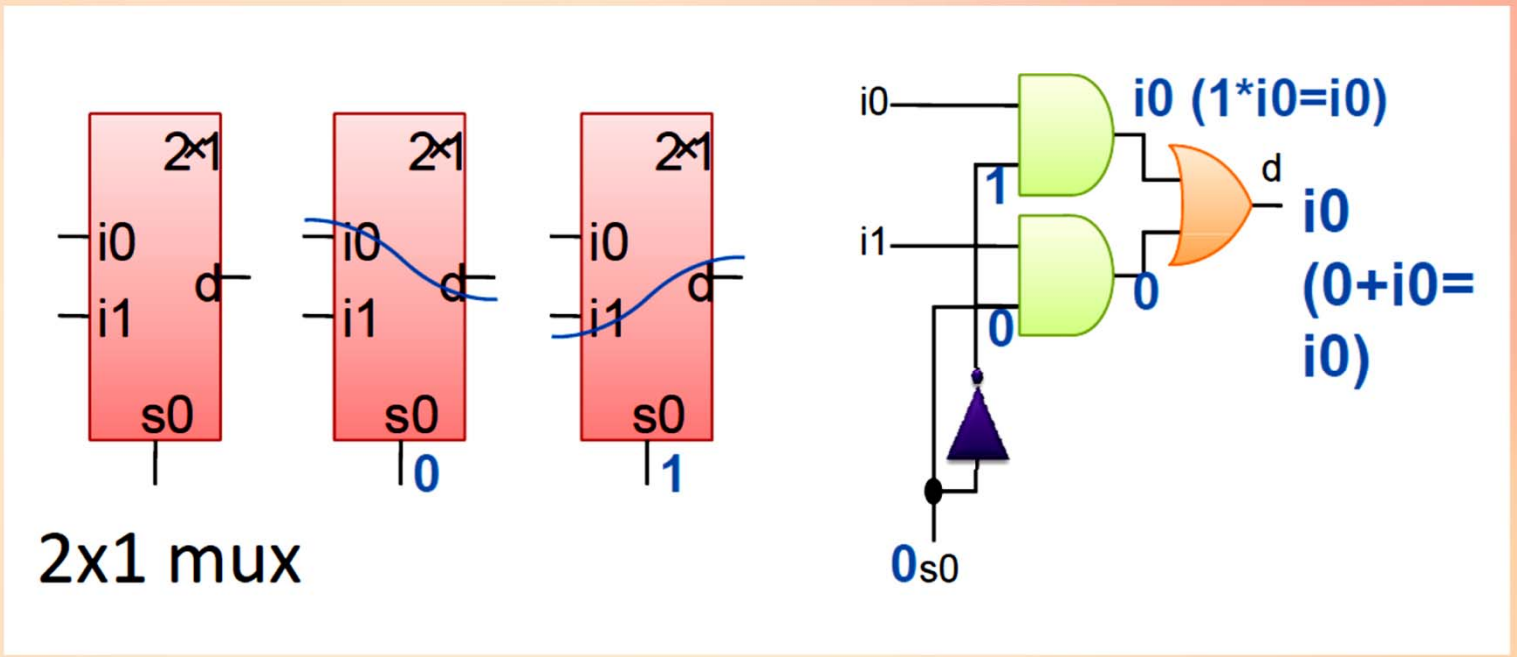
$$Mux\ 4 \times 1$$

$$Mux\ 8 \times 1$$

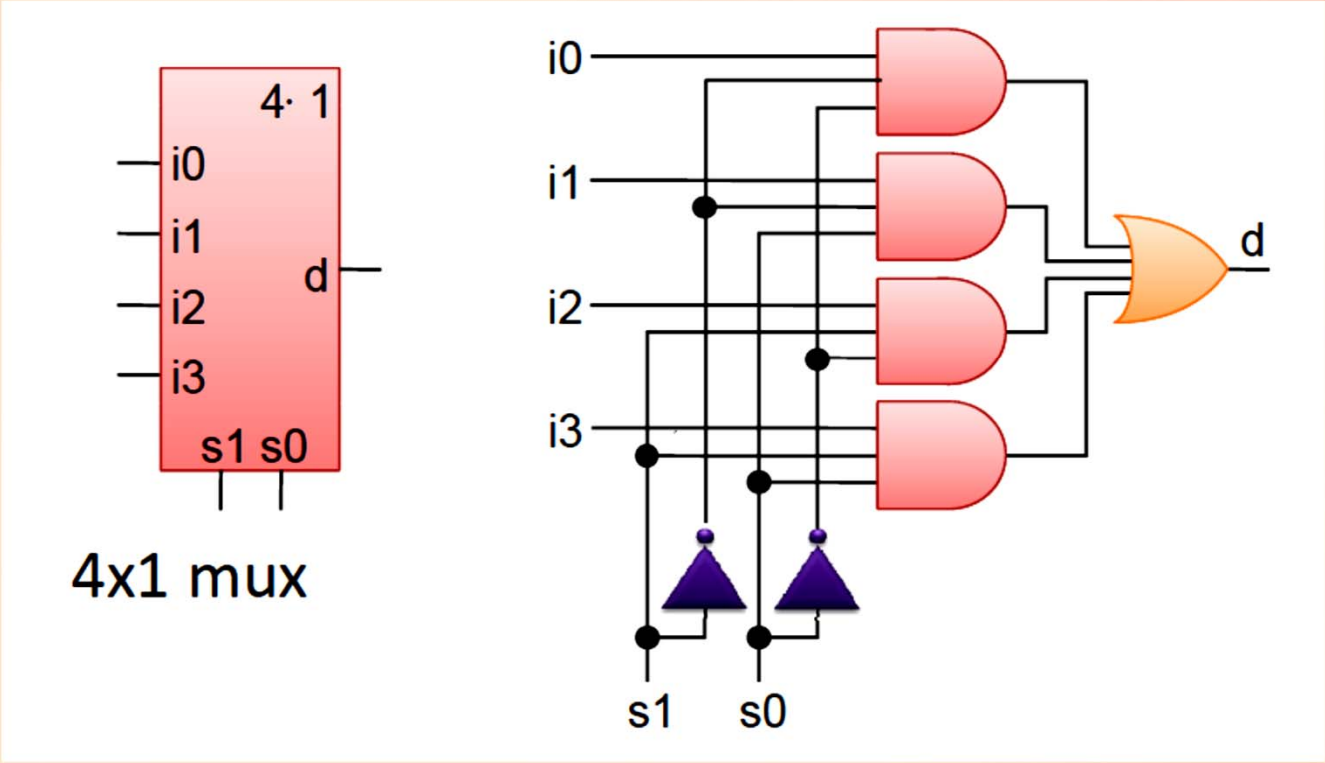
$$Mux\ 2^{k} \times 1$$

$$k = 1, 2, 3, \ldots.$$

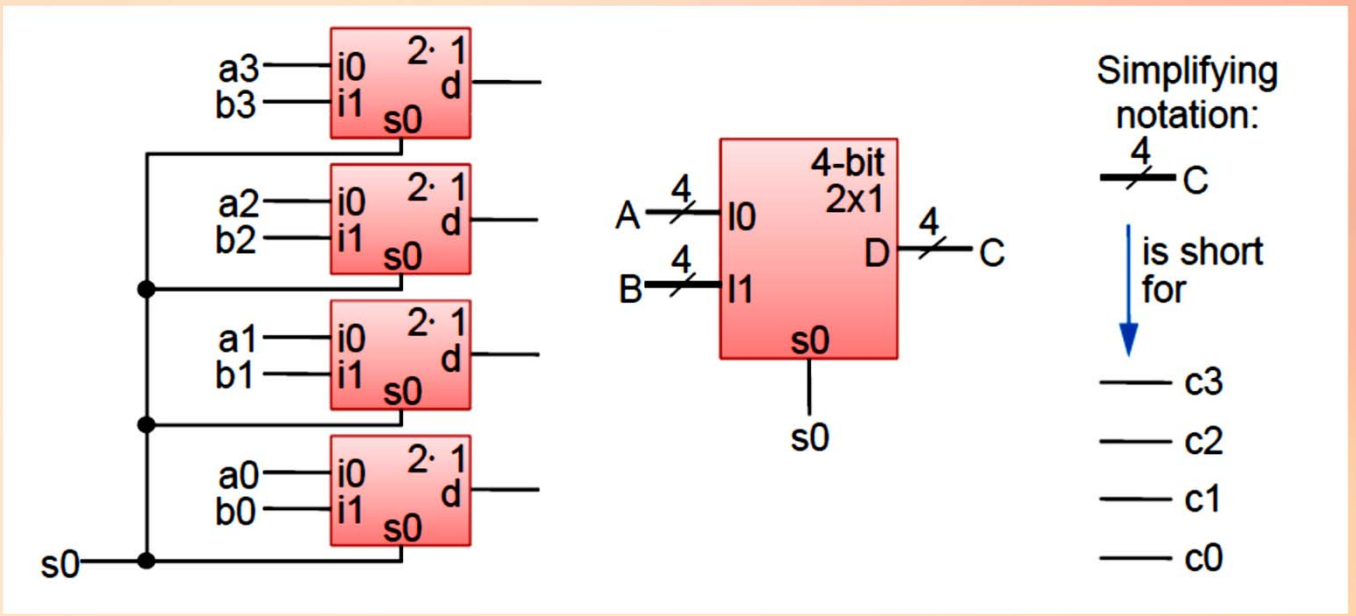
Mux Internal Design



Mux Internal Design

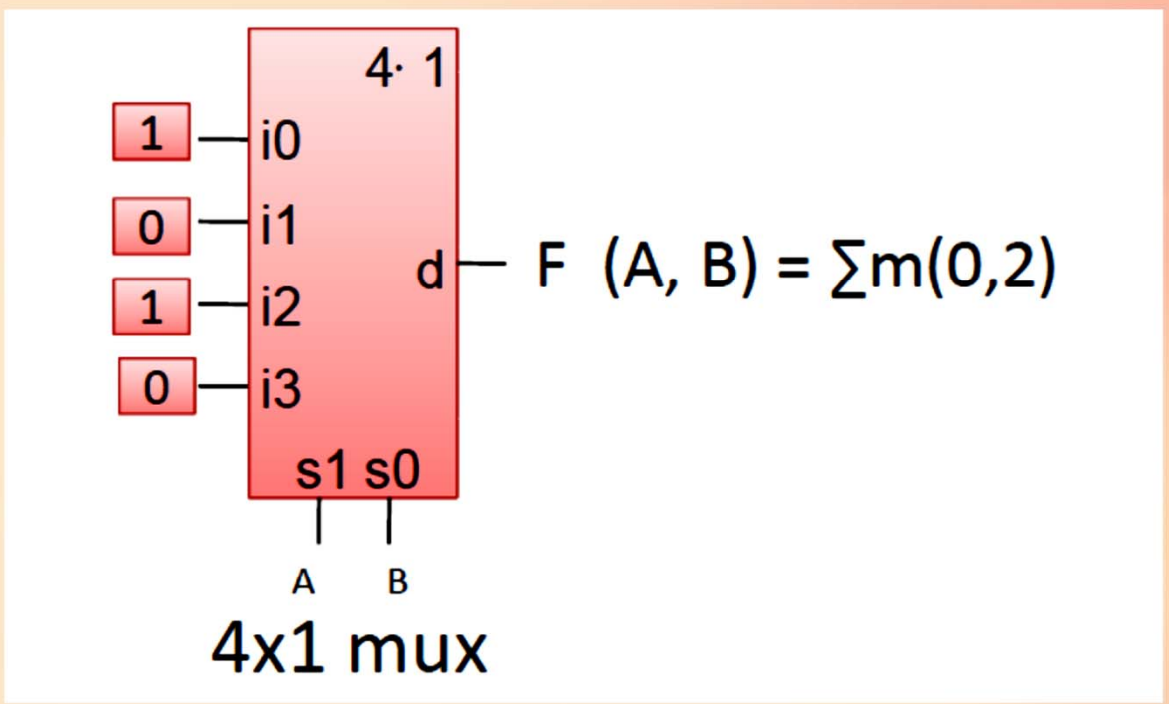


Muxes Commonly Together -- N-bit Mux

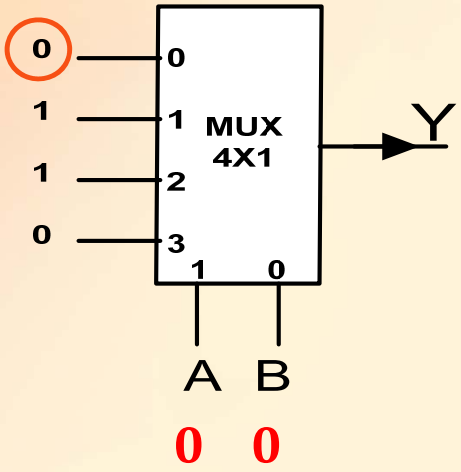


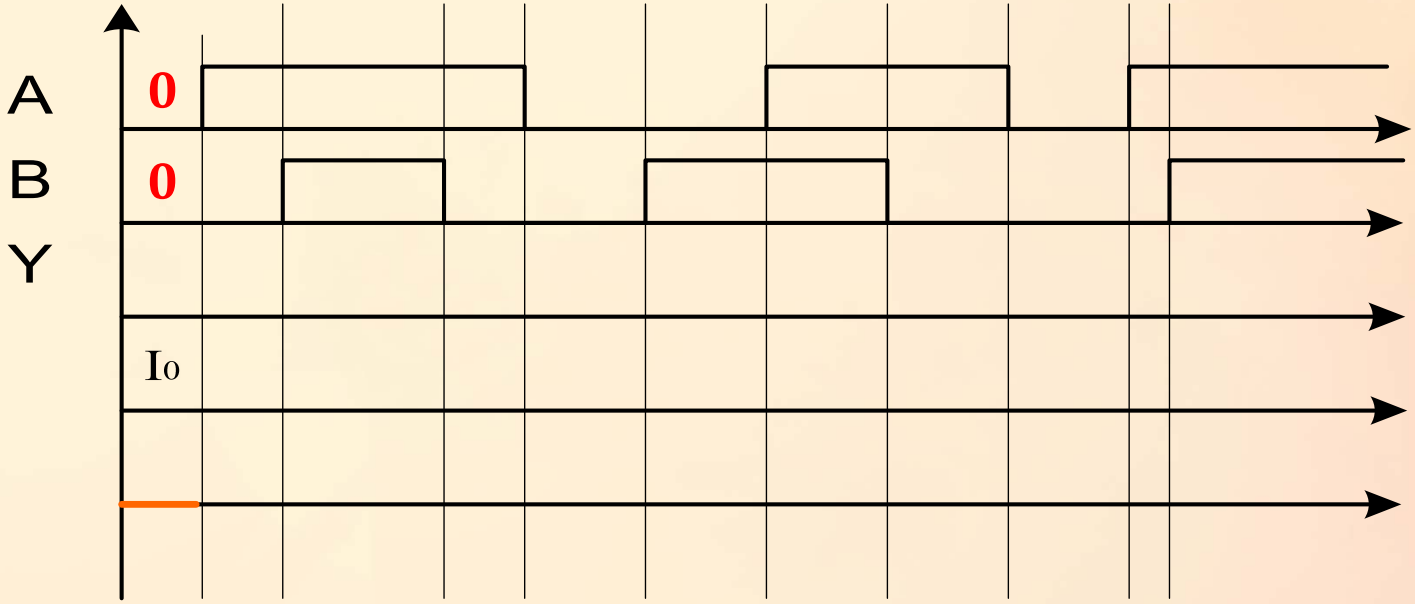
• Ex: Two 4-bit inputs, A (a3 a2 a1 a0), and B (b3 b2 b1 b0)
– 4-bit 2x1 mux (just four 2x1 muxes sharing a select line) can select between A or B

Implementing logic Function using MUX

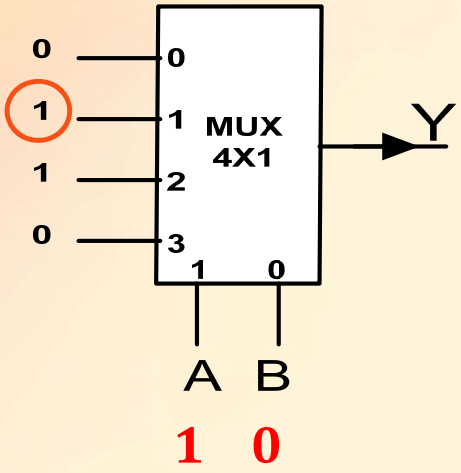


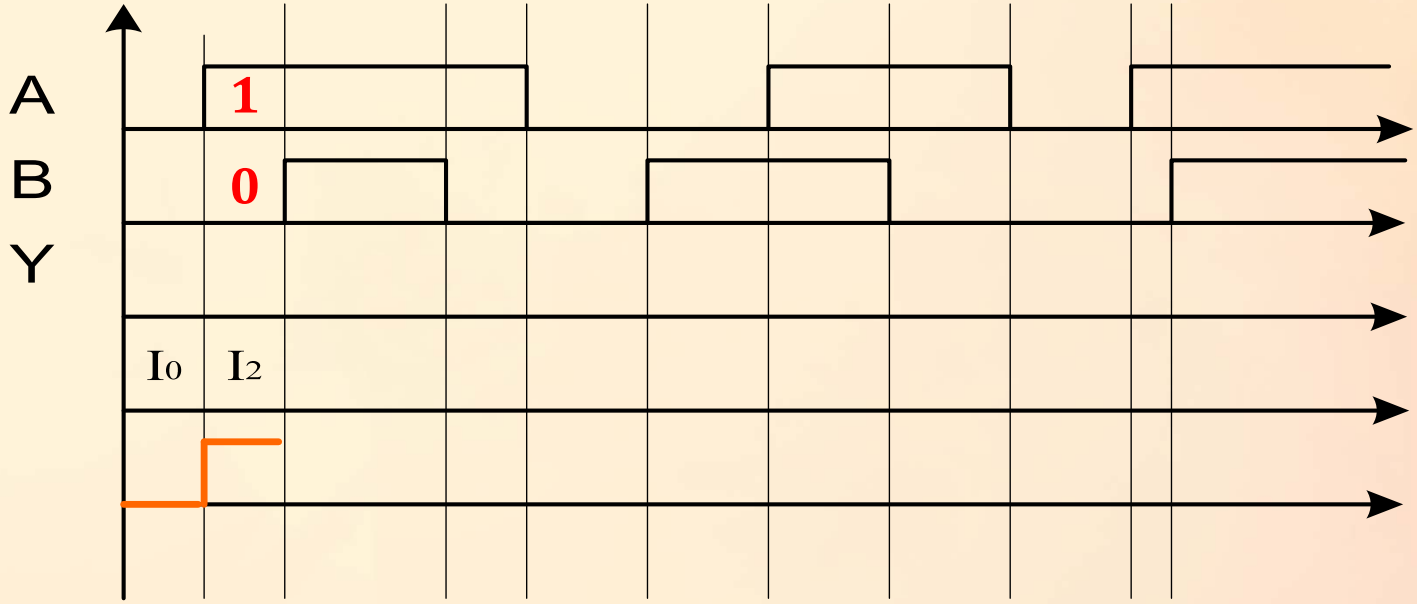
Implementing logic Function using MUX



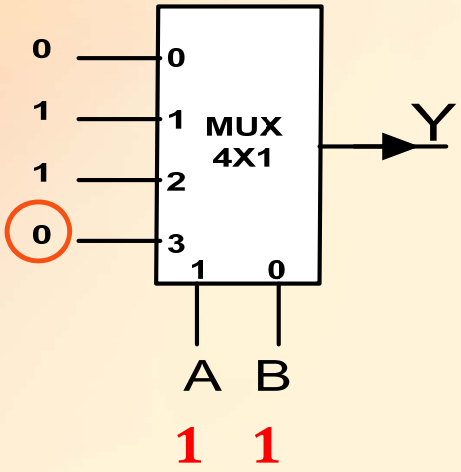


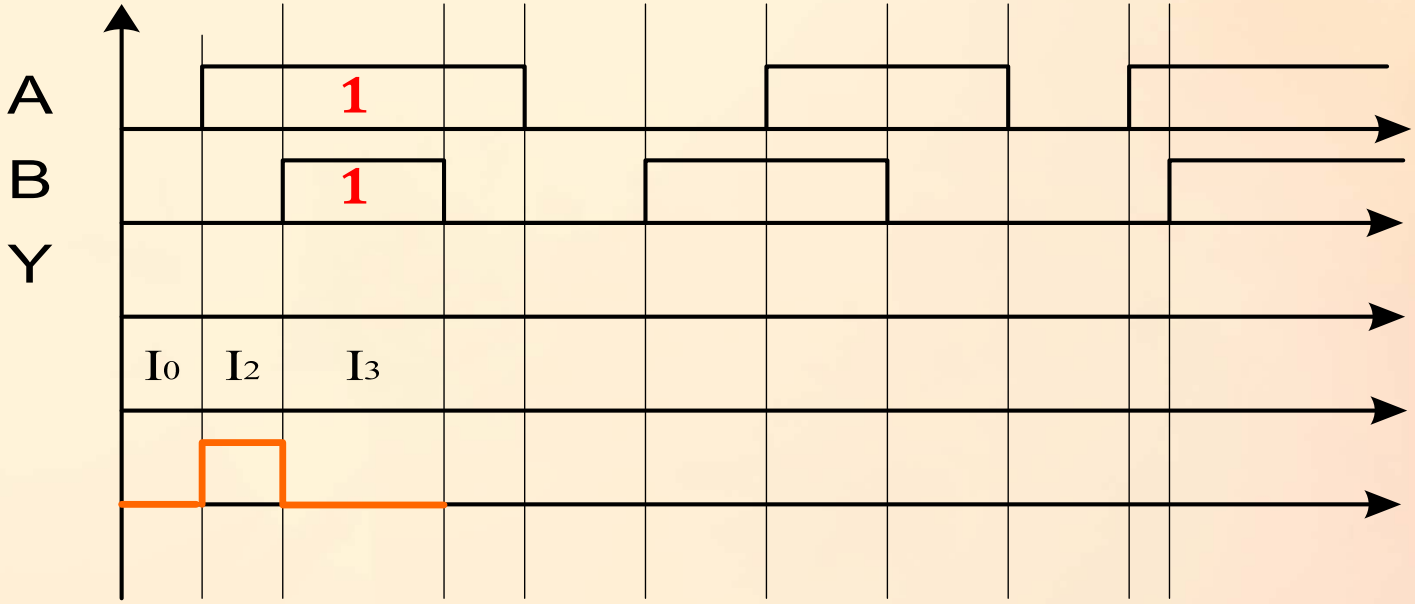
Implementing logic Function using MUX



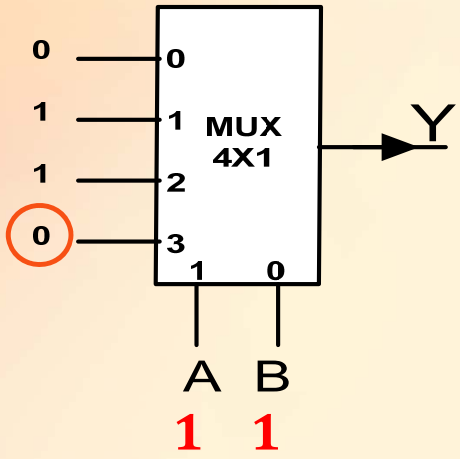


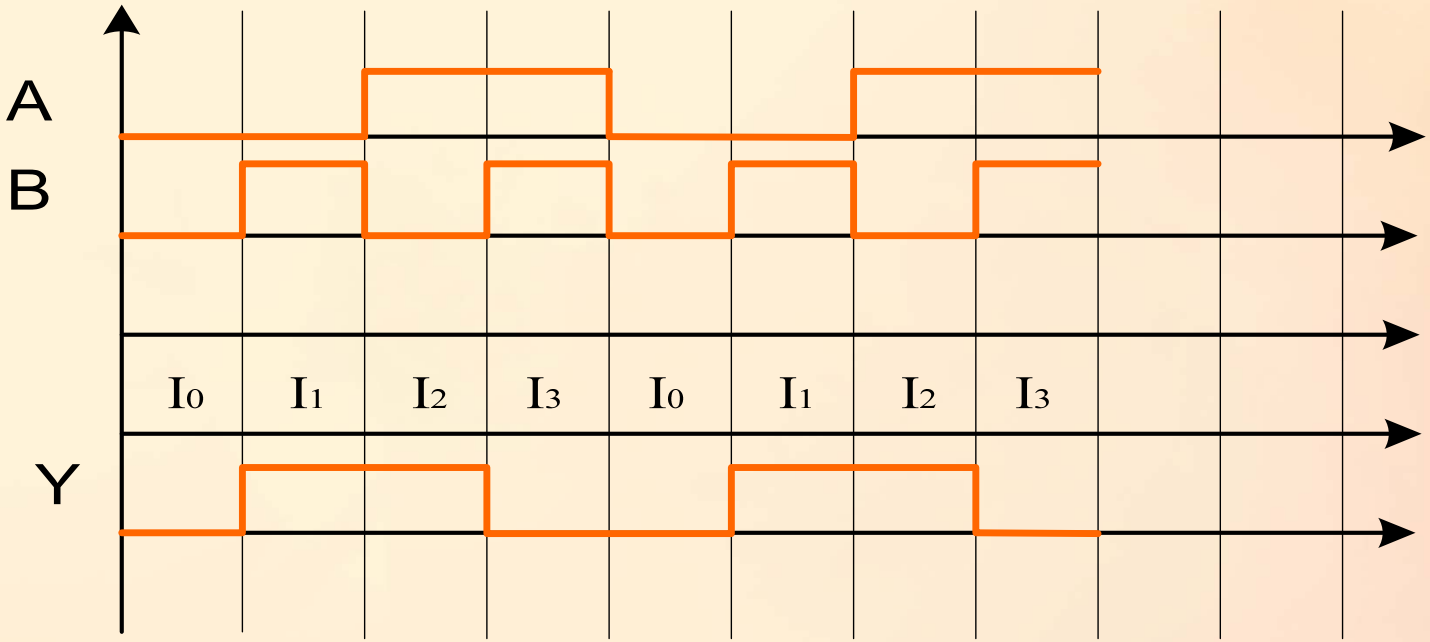
Implementing logic Function using MUX



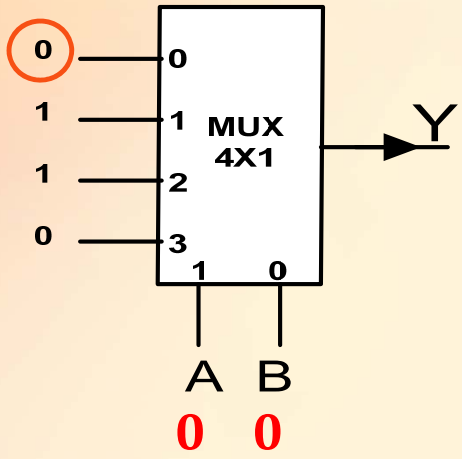


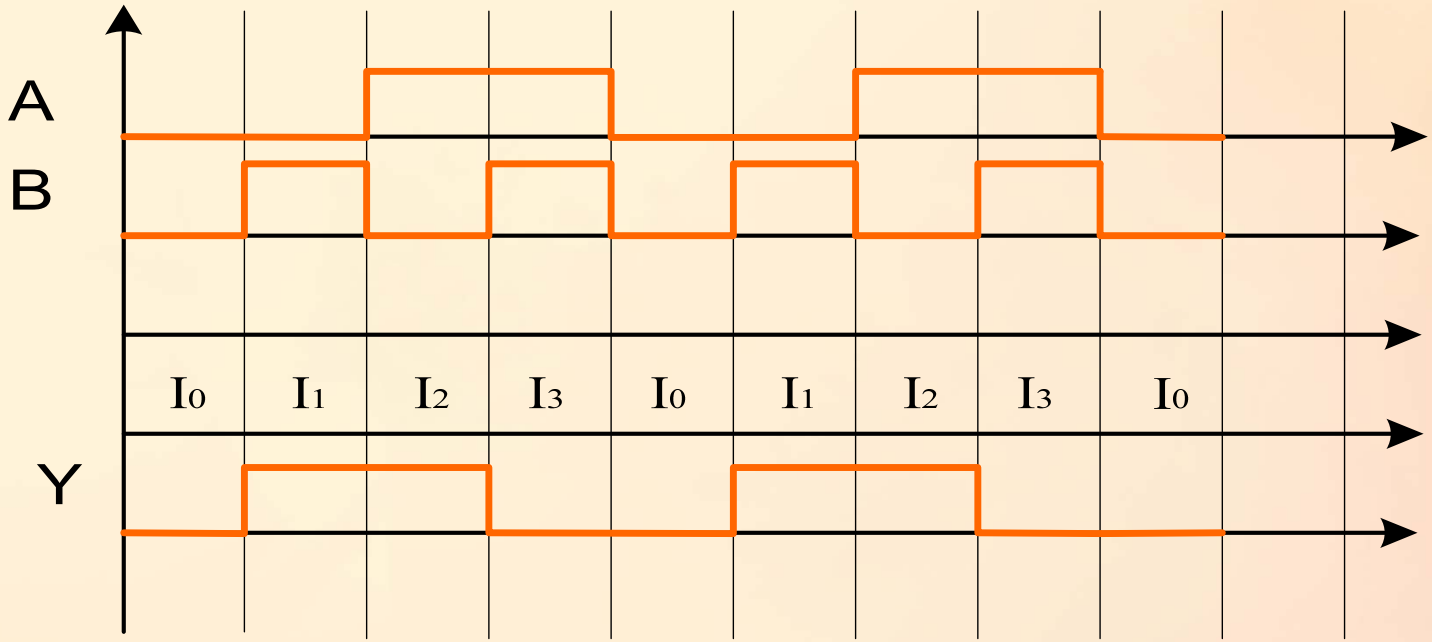
Implementing logic Function using MUX



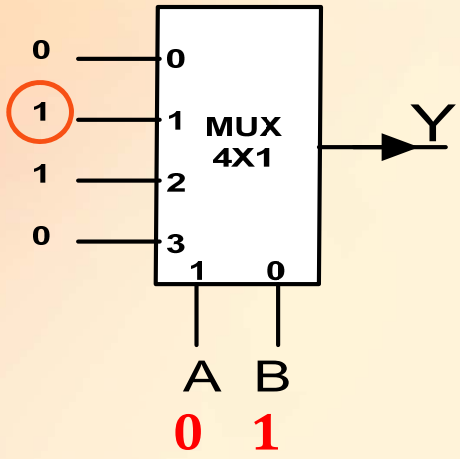


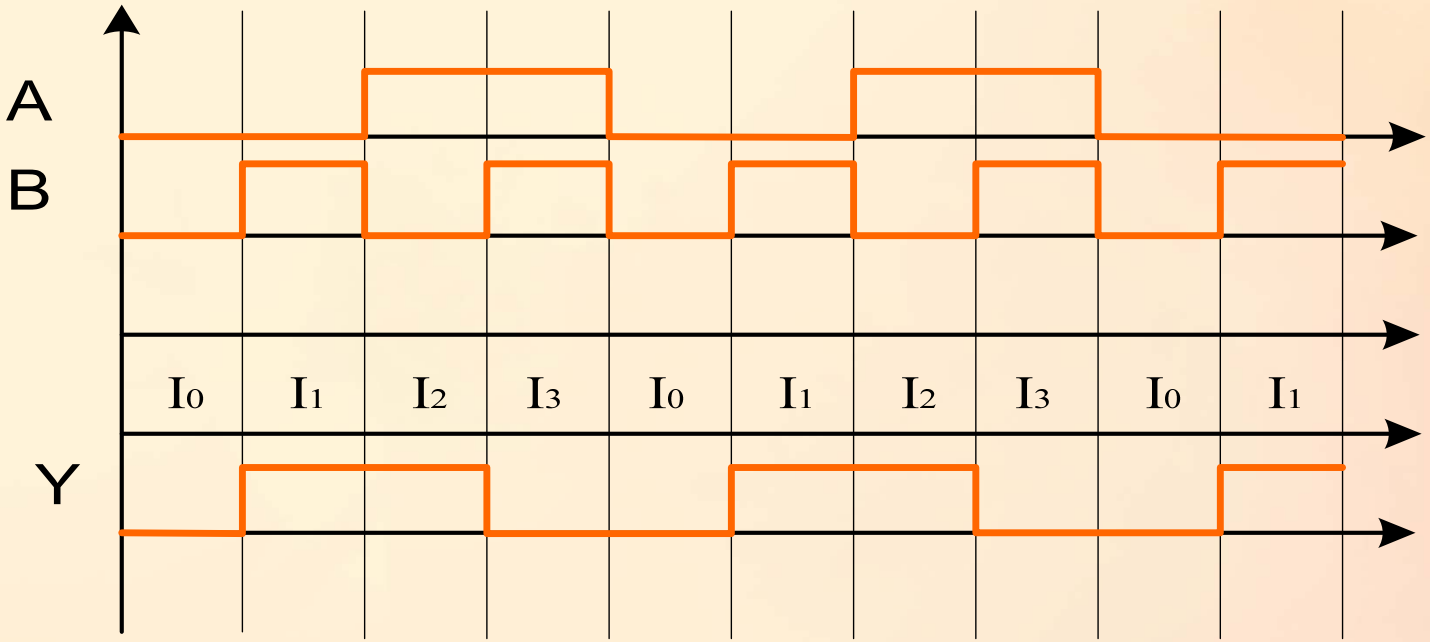
Implementing logic Function using MUX



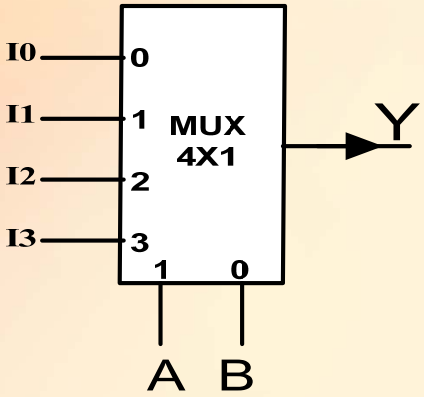


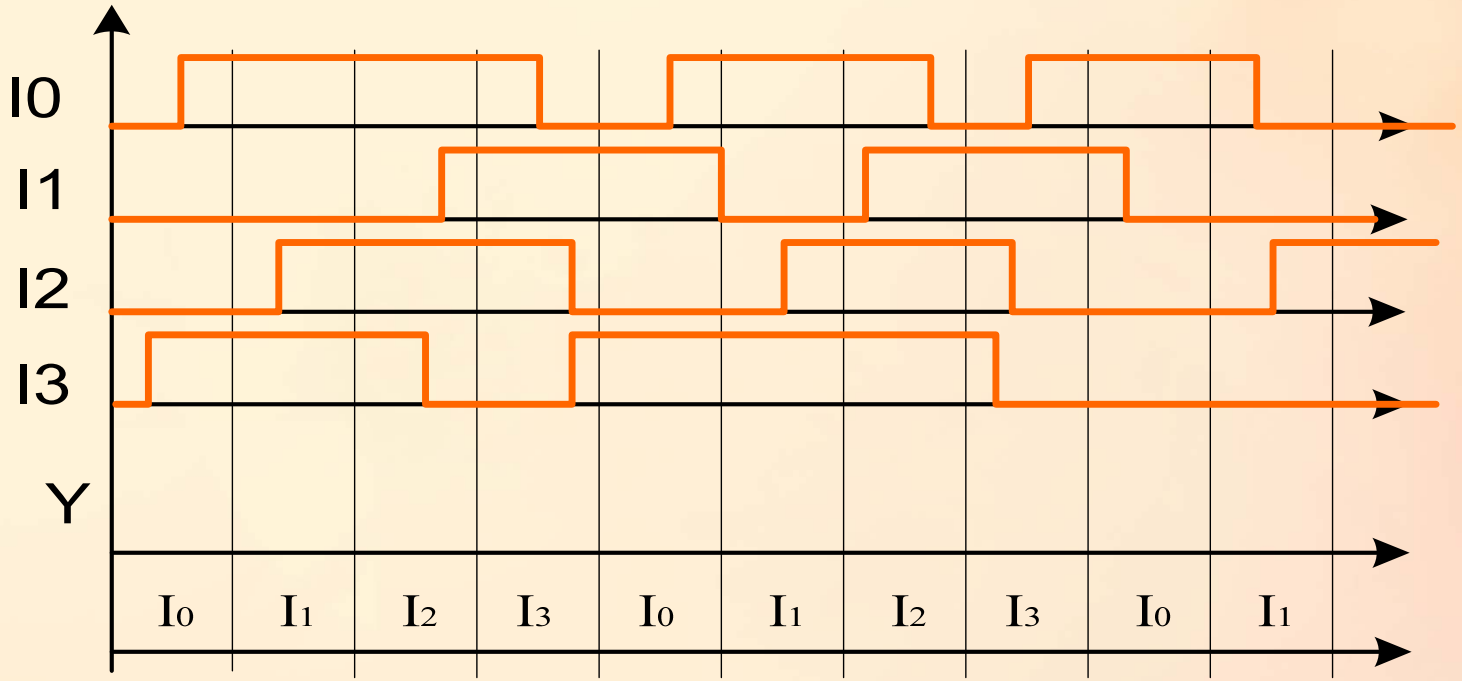
Implementing logic Function using MUX



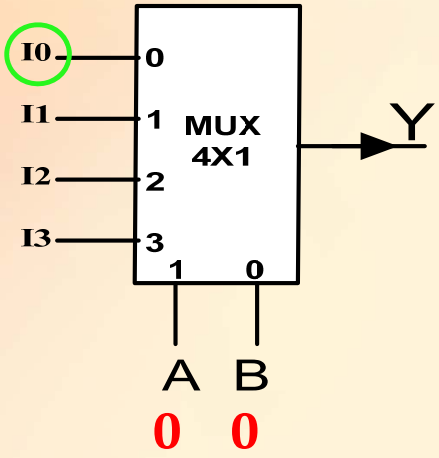


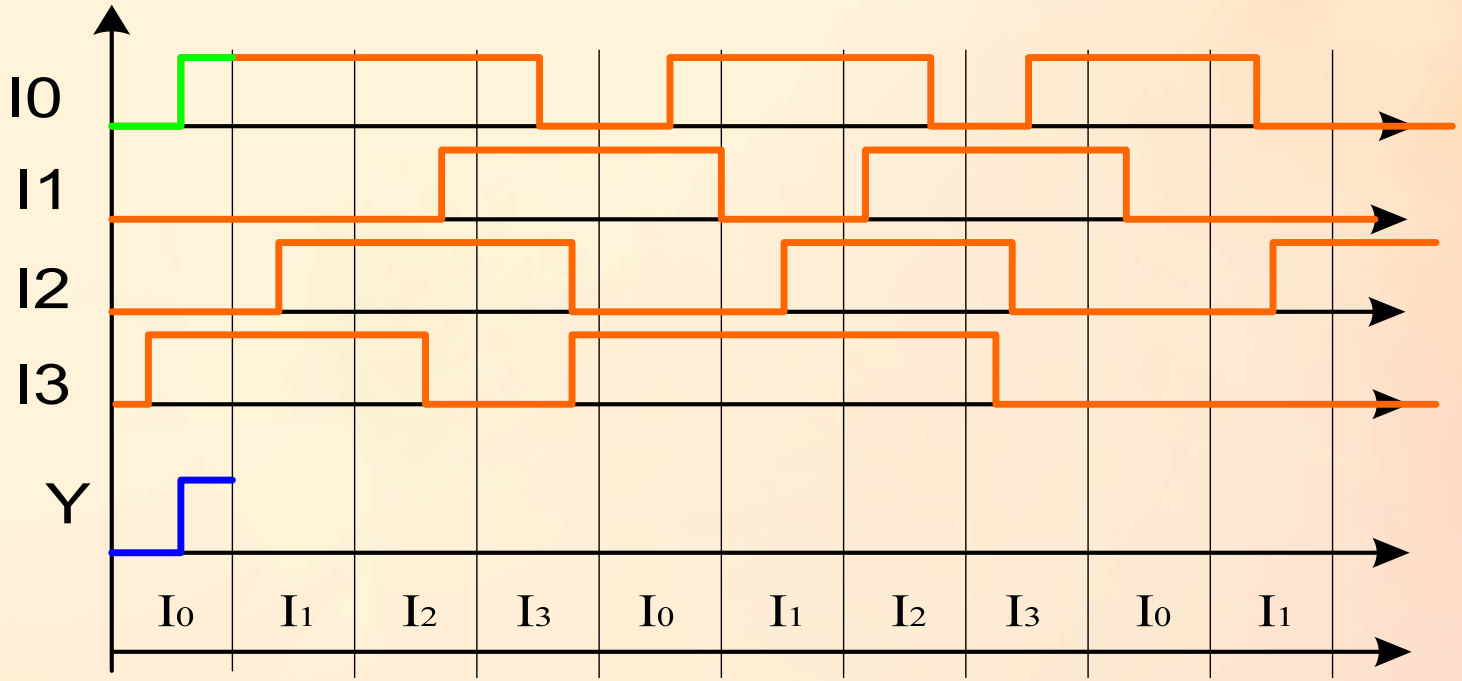
Implementing logic Function using MUX



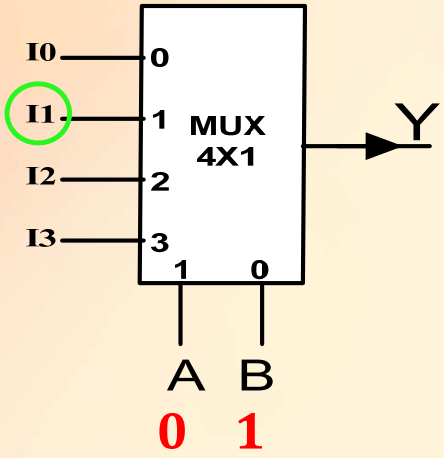


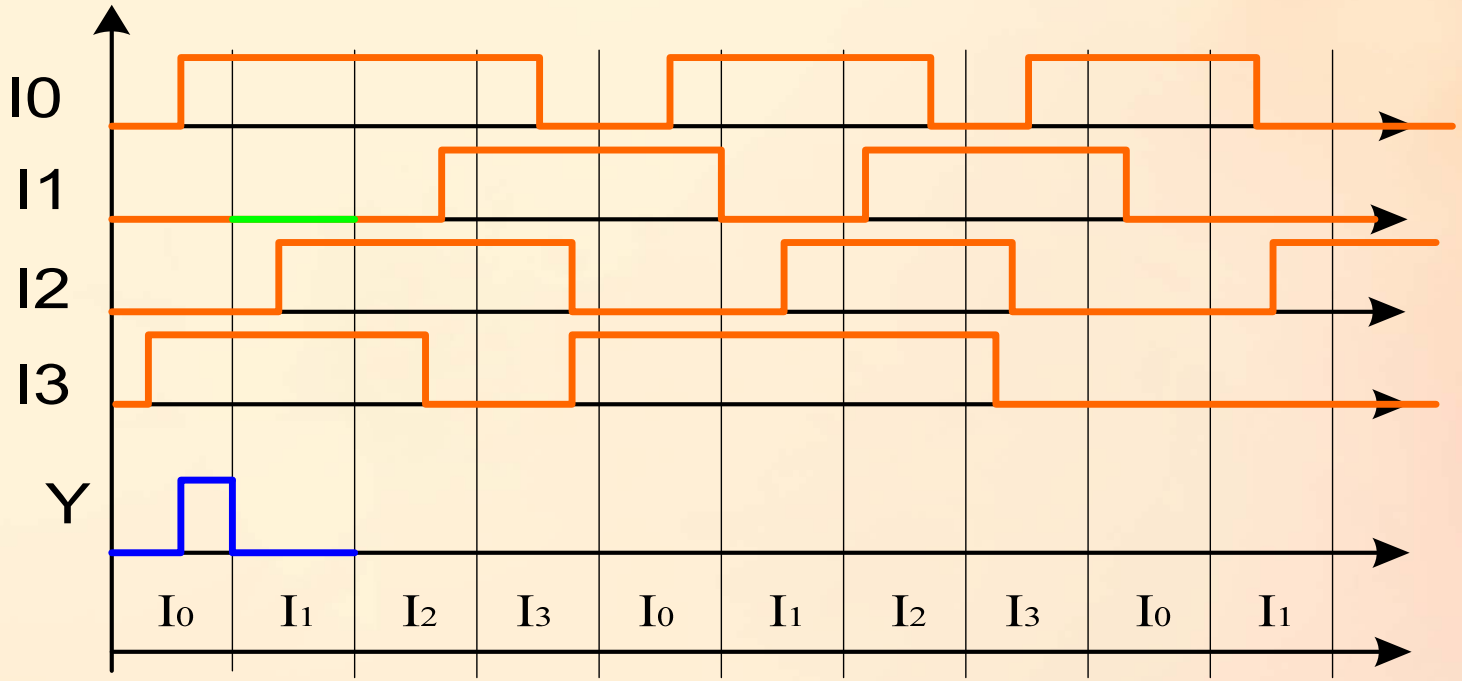
Implementing logic Function using MUX



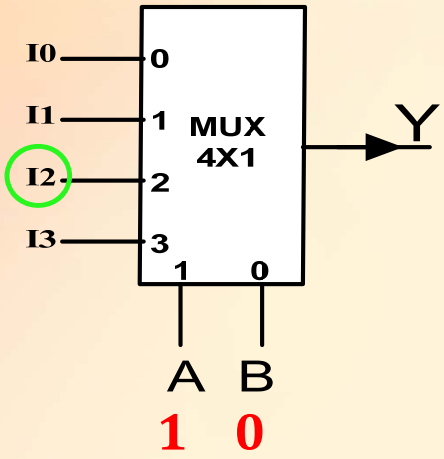


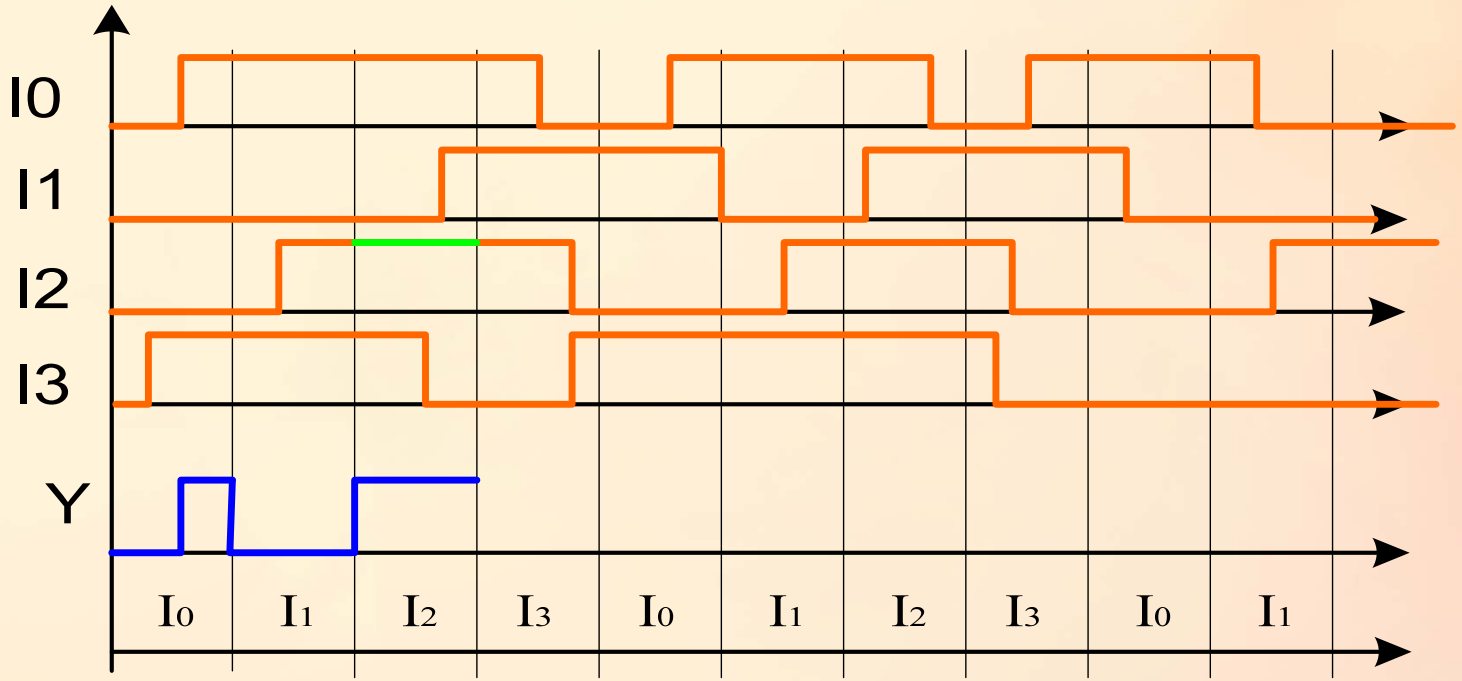
Implementing logic Function using MUX



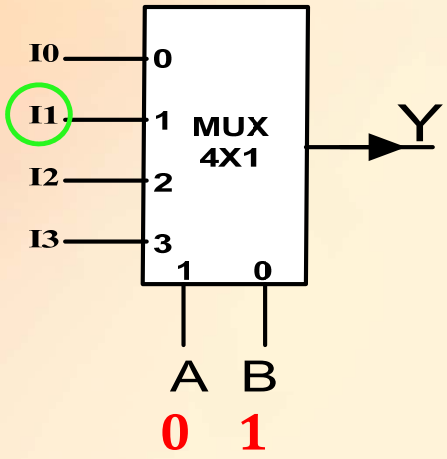


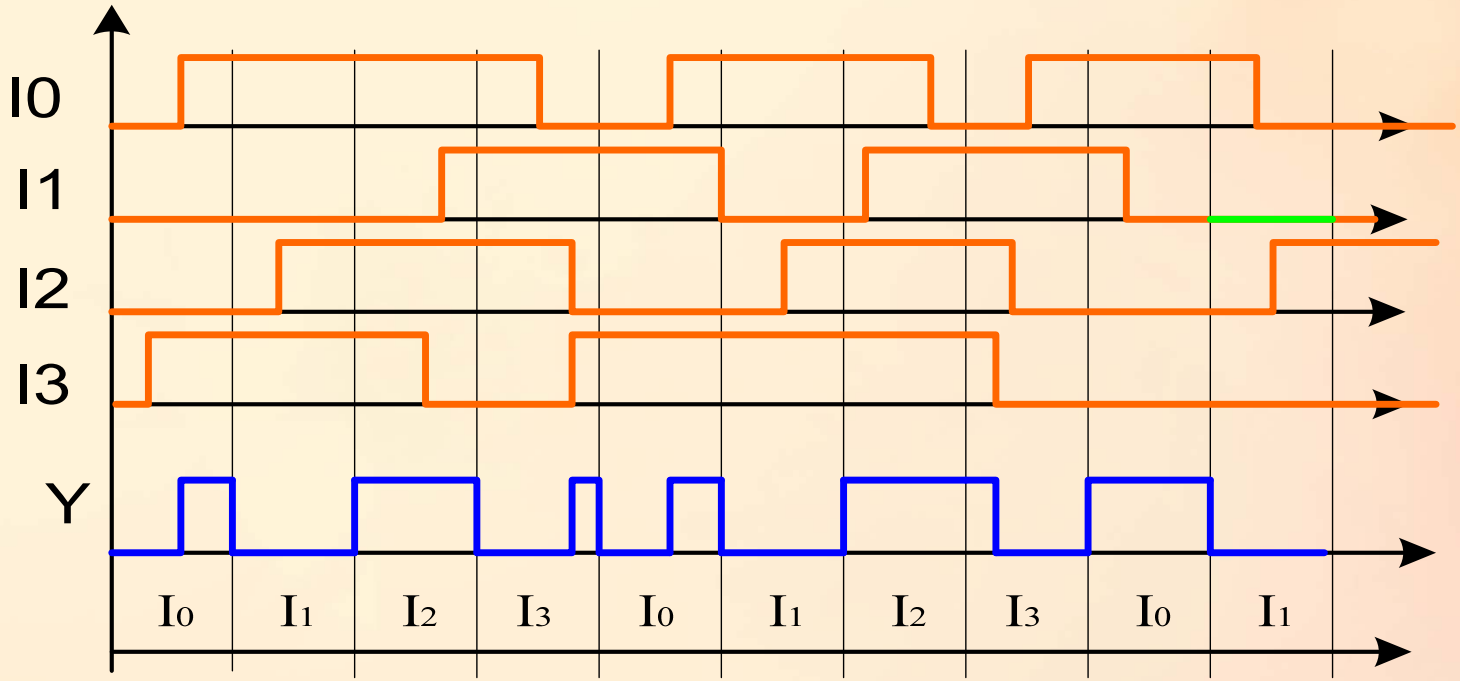
Implementing logic Function using MUX



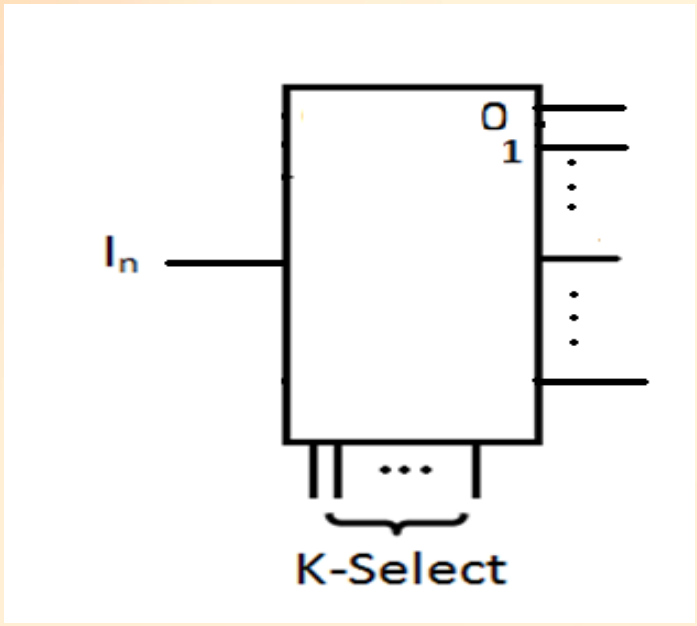


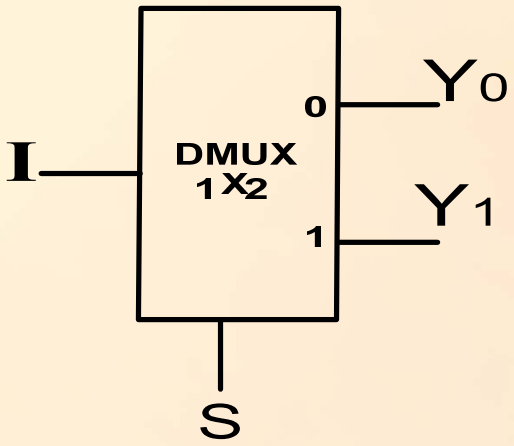
Implementing logic Function using MUX





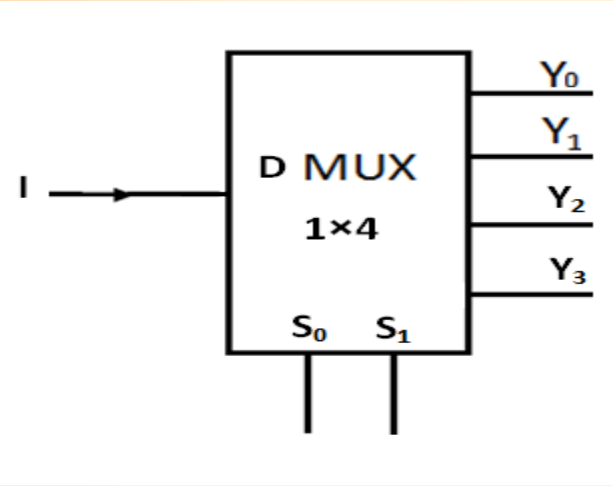
(De Multiplexer) DEMUX



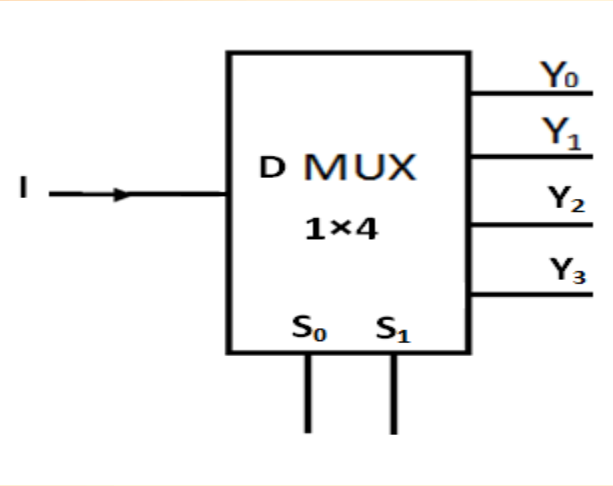


S	y_0	y_1
0	I	0
1	0	I

(De Multiplexer) DEMUX

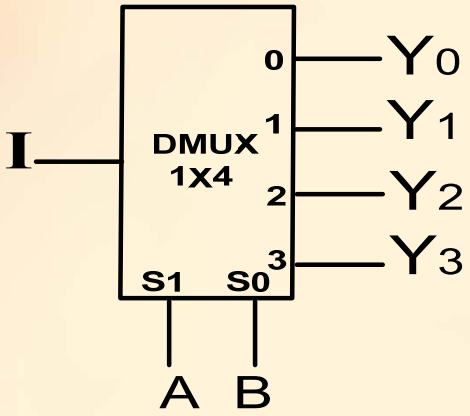


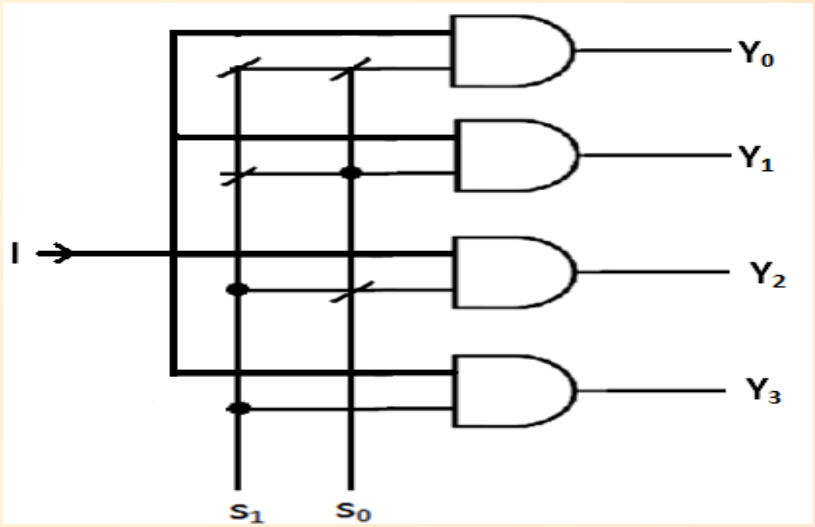
(De Multiplexer) DEMUX



S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

(De Multiplexer) DEMUX

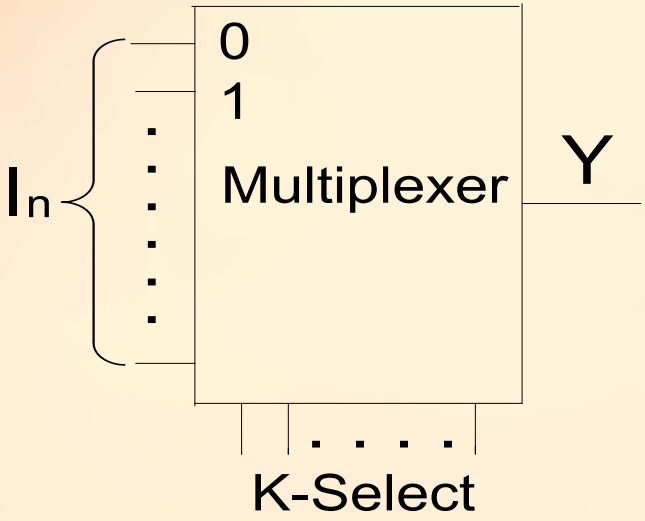




S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

Multiplexer

Multiplexer (MUX)



$$Mux\ 2 \times 1$$

$$Mux\ 4 \times 1$$

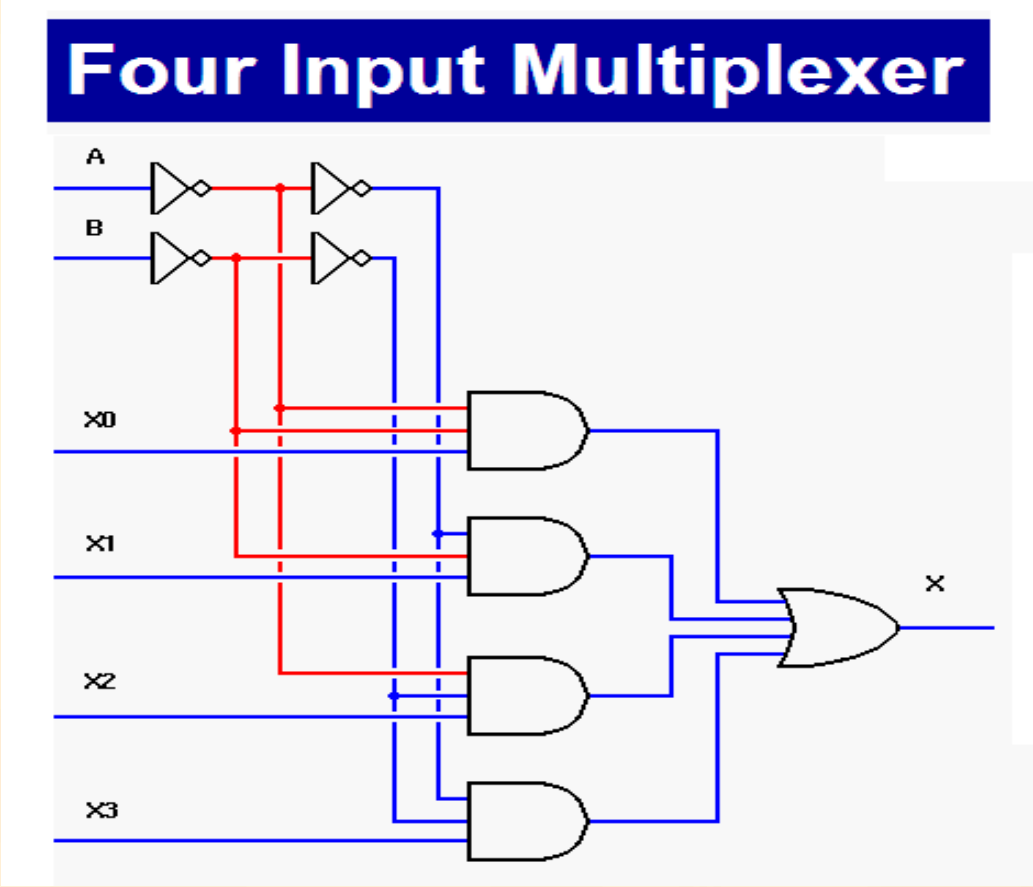
$$Mux\ 8 \times 1$$

$$\vdots$$

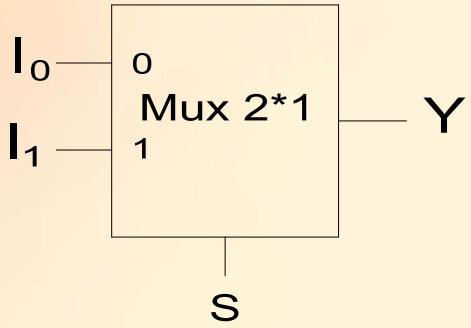
$$Mux\ 2^{k} \times 1$$

$$k = 1, 2, 3, \ldots.$$

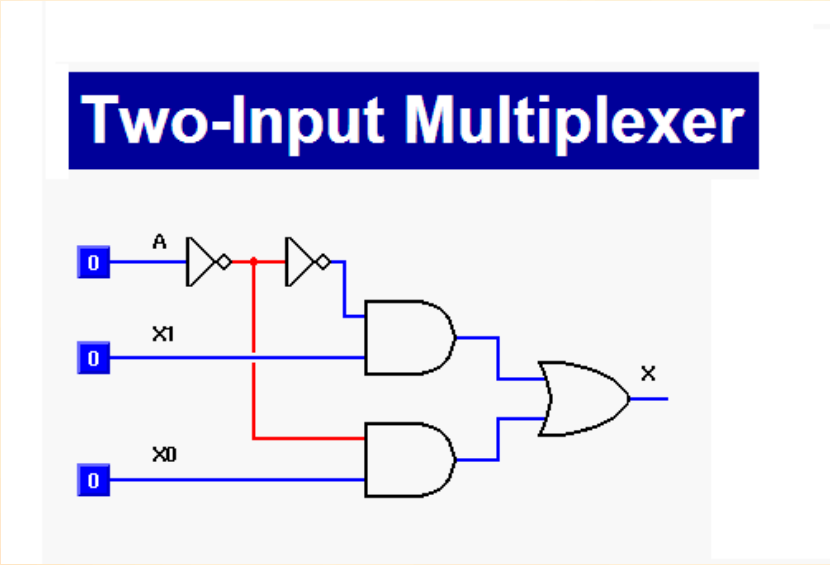
Multiplexer

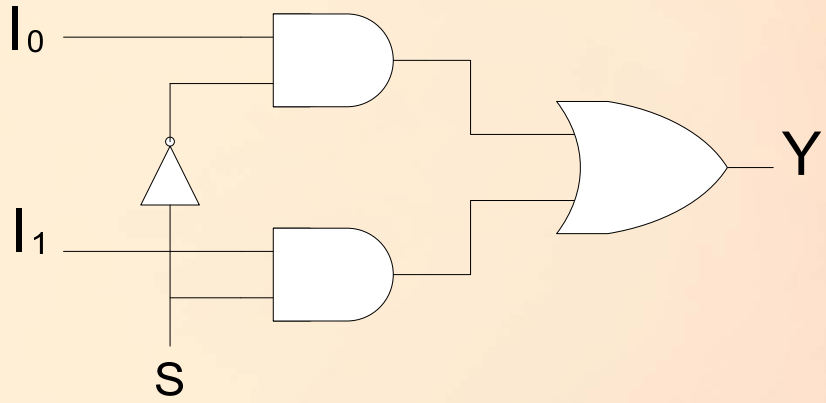


Multiplexer



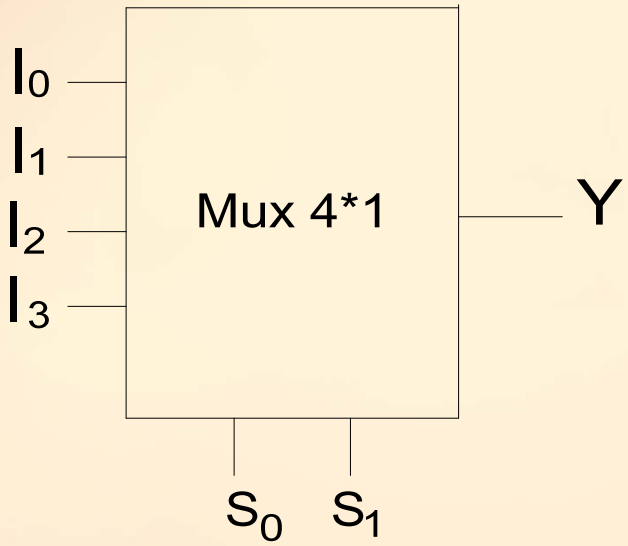
S	Y
0	I_0
1	I_1





$$Y = \overline{S} I_0 + S I_1$$

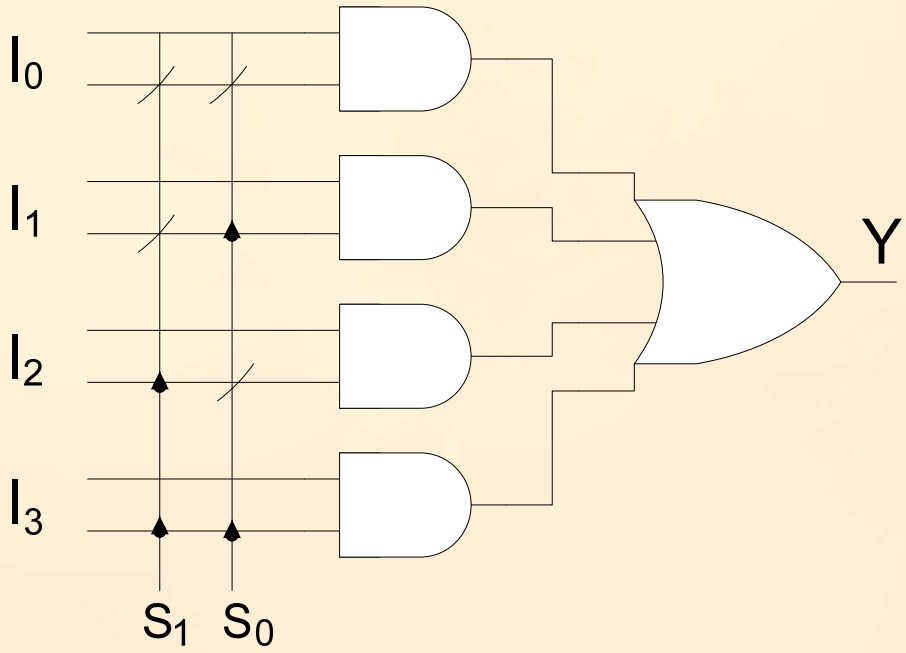
(De Multiplexer) DEMUX



S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

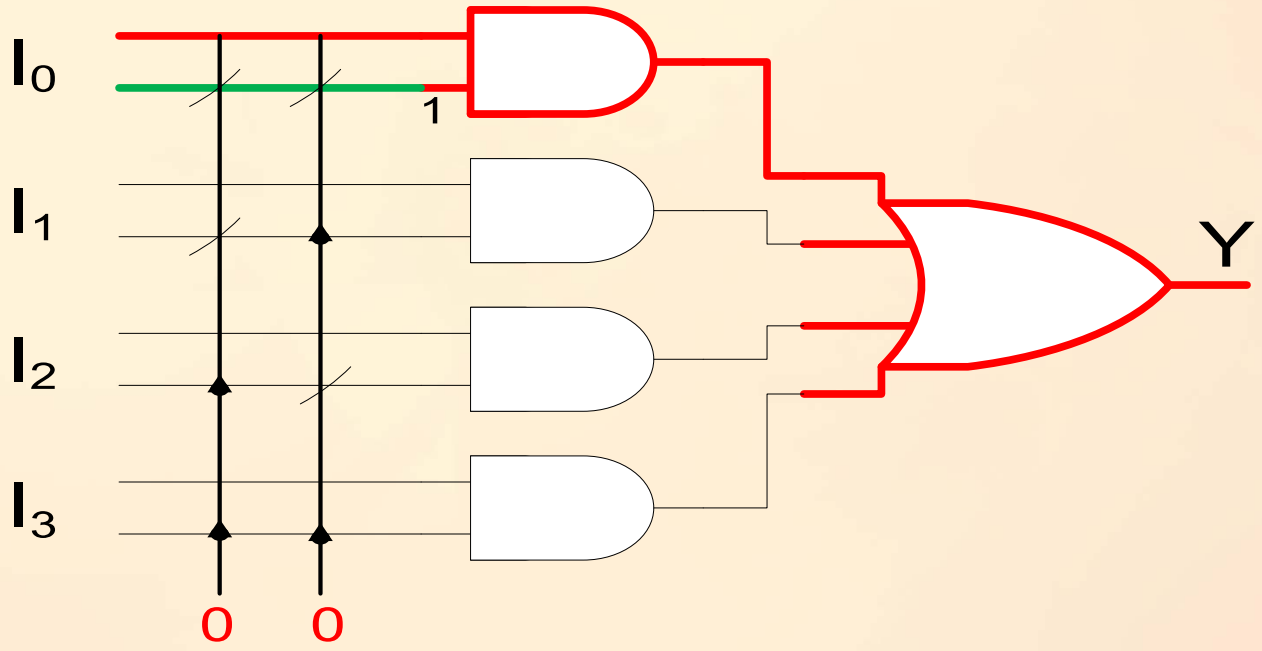
$$F = S0'\,S1'\,.I0 + S0\,S1'.\,I1 + S0'S1\,.I2 + S0\,S1\,.I3$$

Four Input Multiplexer

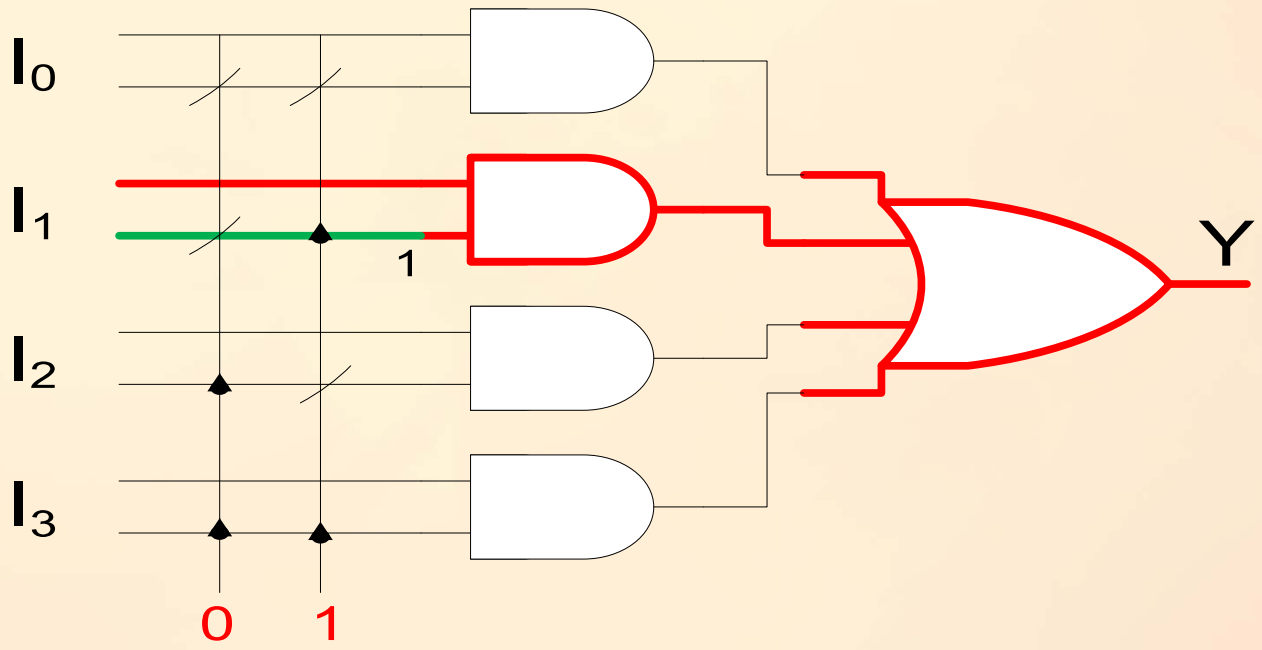


$$F = S0'\,S1'\,.I0 + S0\,S1'.\,I1 + S0'S1\,.I2 + S0\,S1\,.I3$$

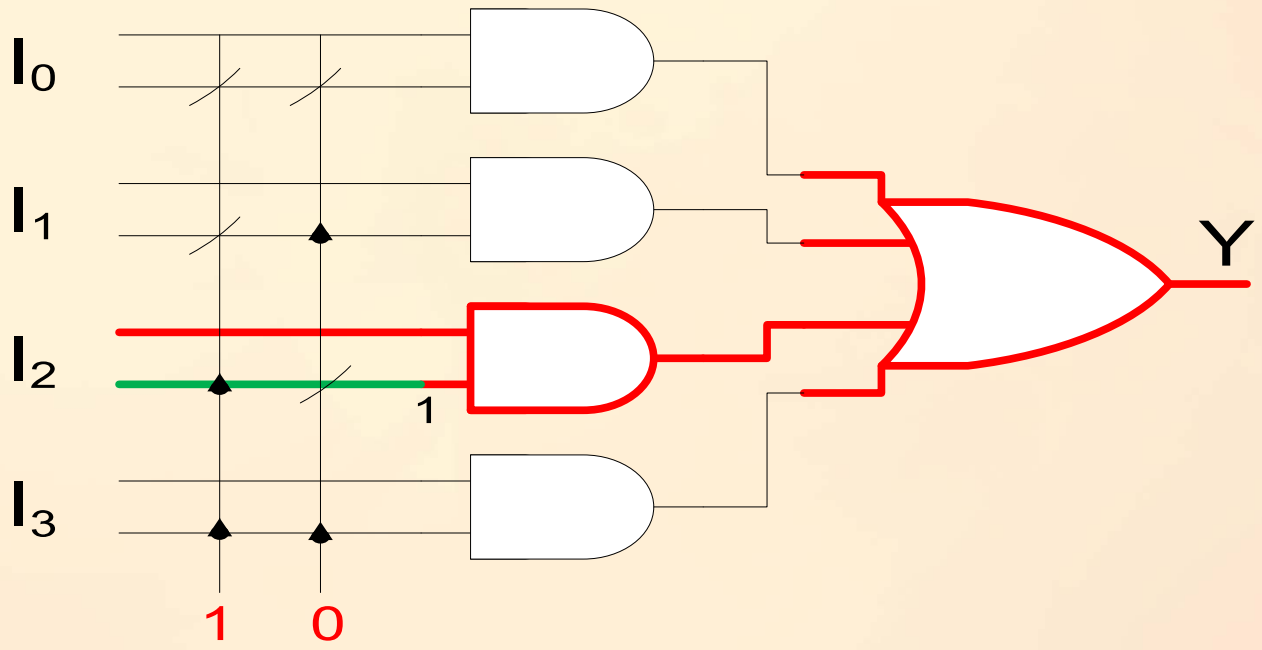
Four Input Multiplexer



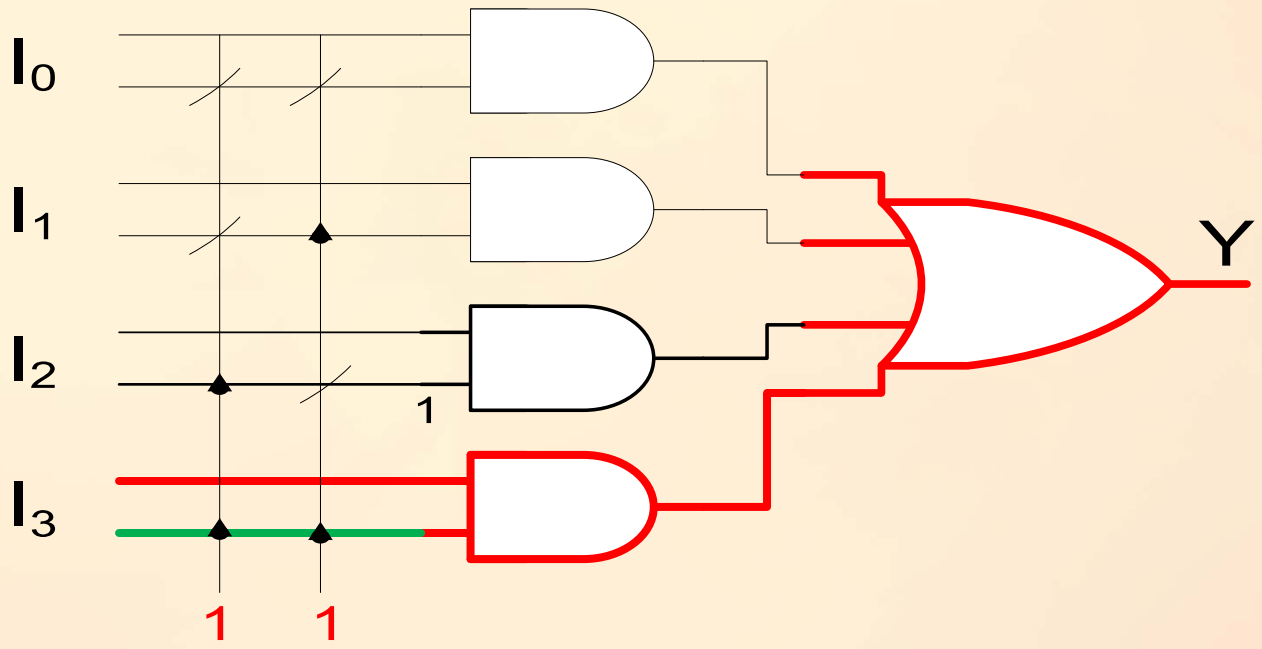
Four Input Multiplexer



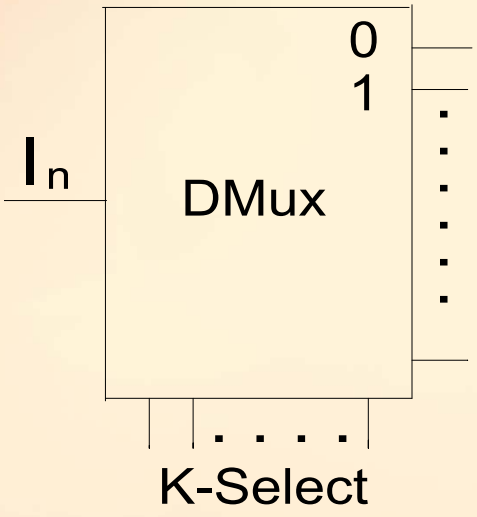
Four Input Multiplexer



Four Input Multiplexer



(De Multiplexer) DEMUX



$$DMux\ 1 \times 2$$

$$DMux\ 1 \times 4$$

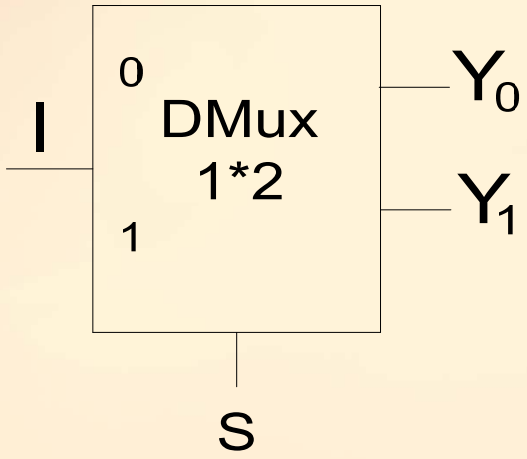
$$DMux\ 1 \times 8$$

$$\vdots$$

$$DMux\ 1 \times 2^{k}$$

$$k = 1, 2, 3, \ldots.$$

(De Multiplexer) DEMUX

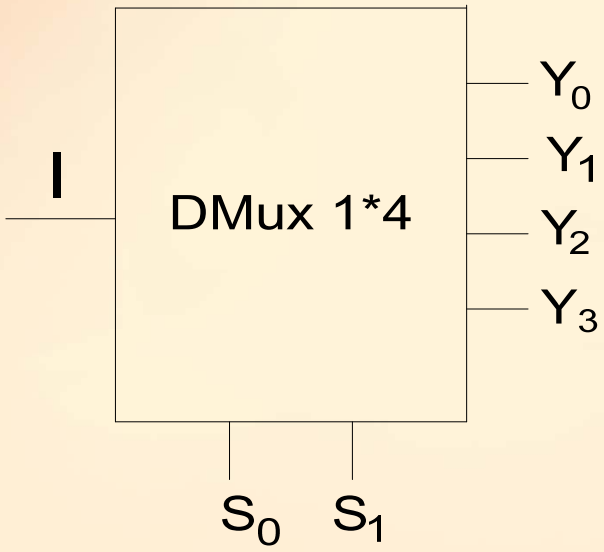


S	Y_0	Y_1
0	0	I
1	I	0

$Y0 = S'I$

$Y1 = S\,I$

(De Multiplexer) DEMUX



S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

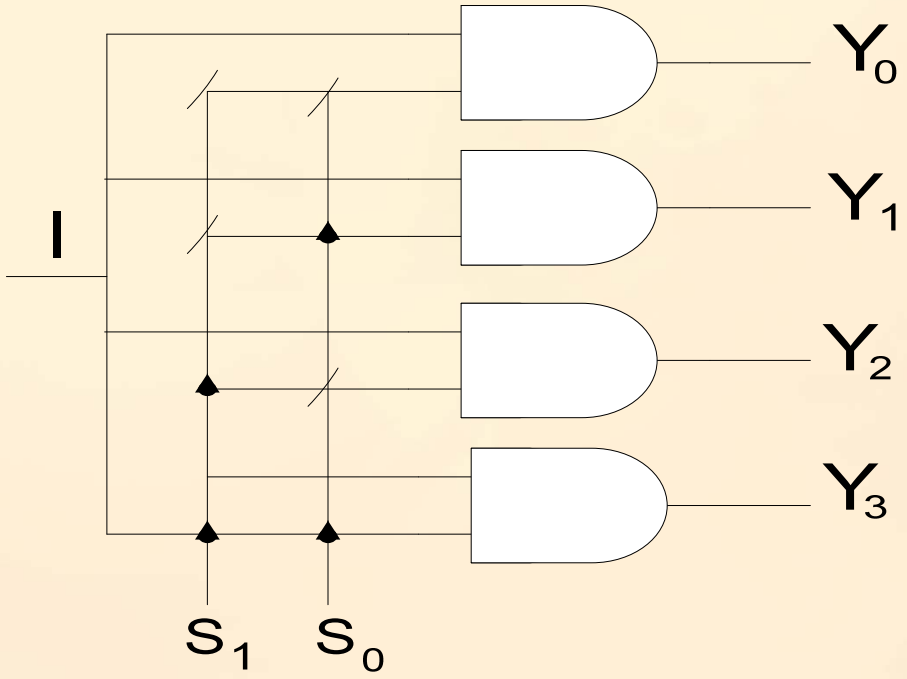
$$Y0 = S0' \; S1' . I$$

$$Y1 = S0 \quad S1' . I$$

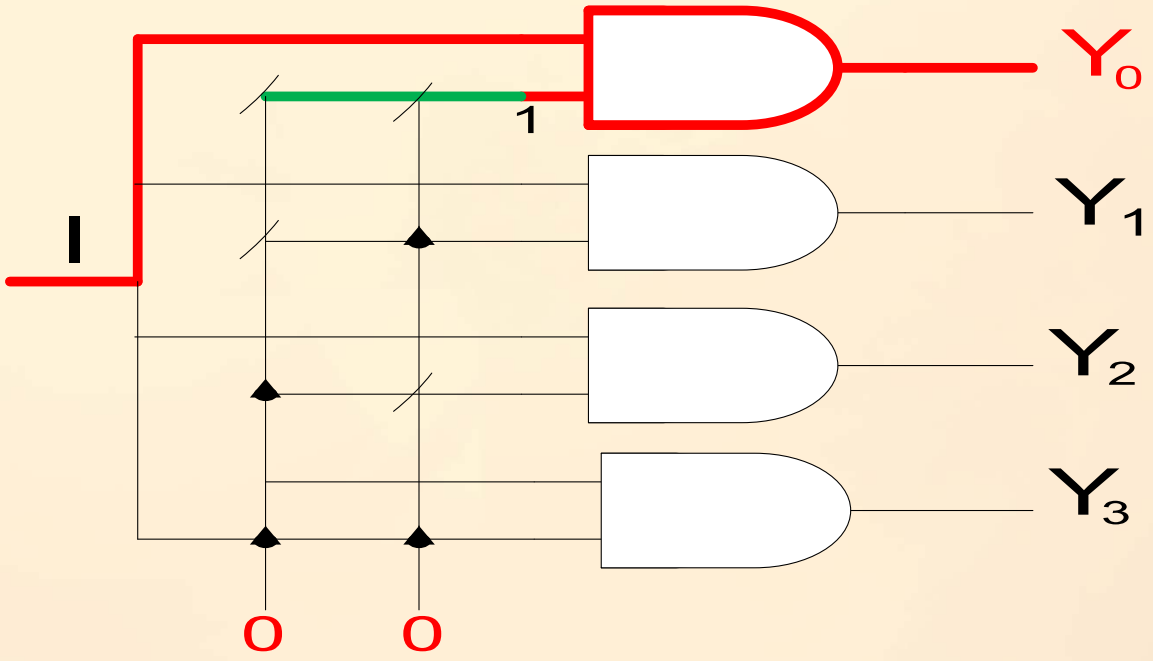
$$Y2 = S0' \, S1 \quad . I$$

$$Y3 = S0 \quad S1 \quad . I$$

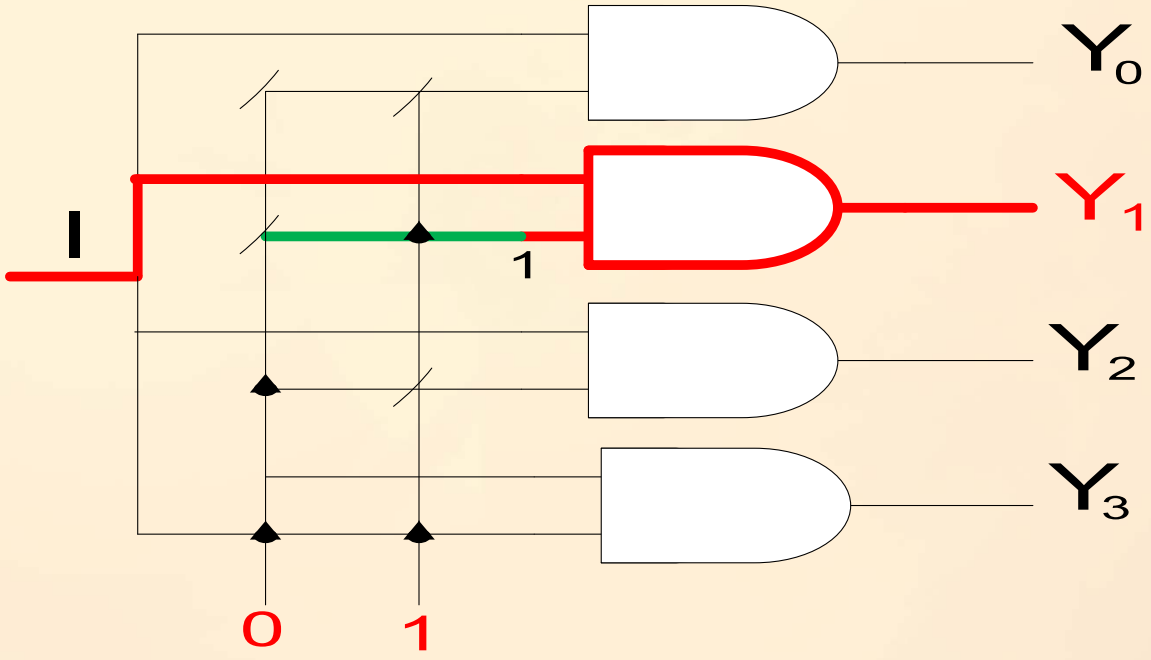
(De Multiplexer) DEMUX



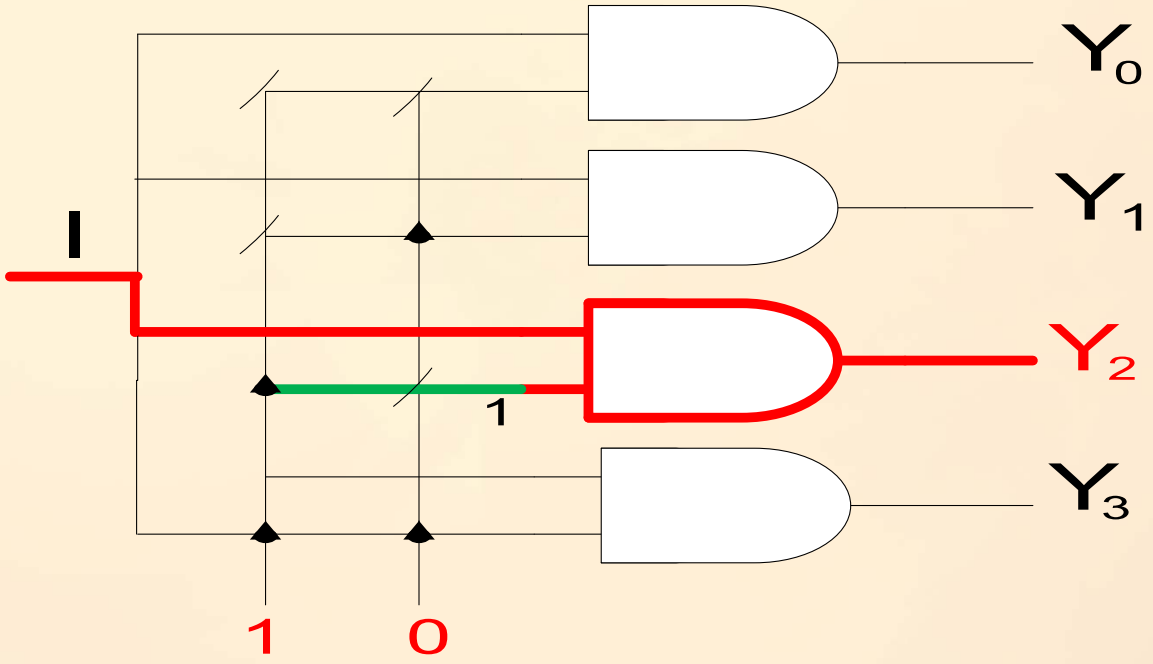
(De Multiplexer) DEMUX



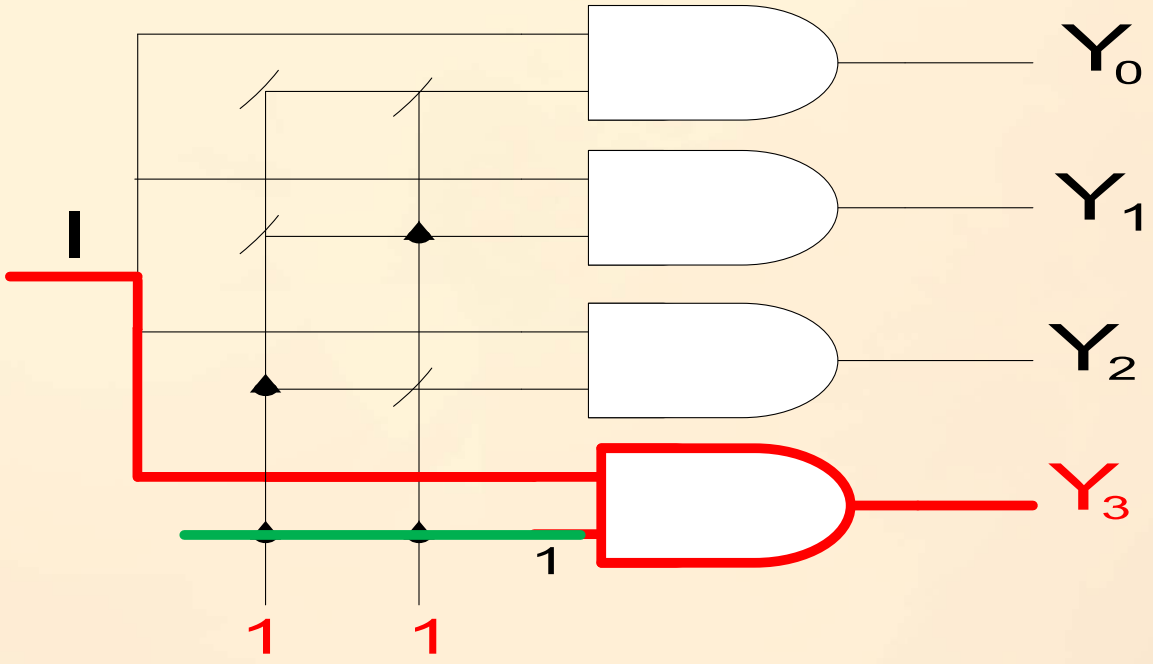
(De Multiplexer) DEMUX



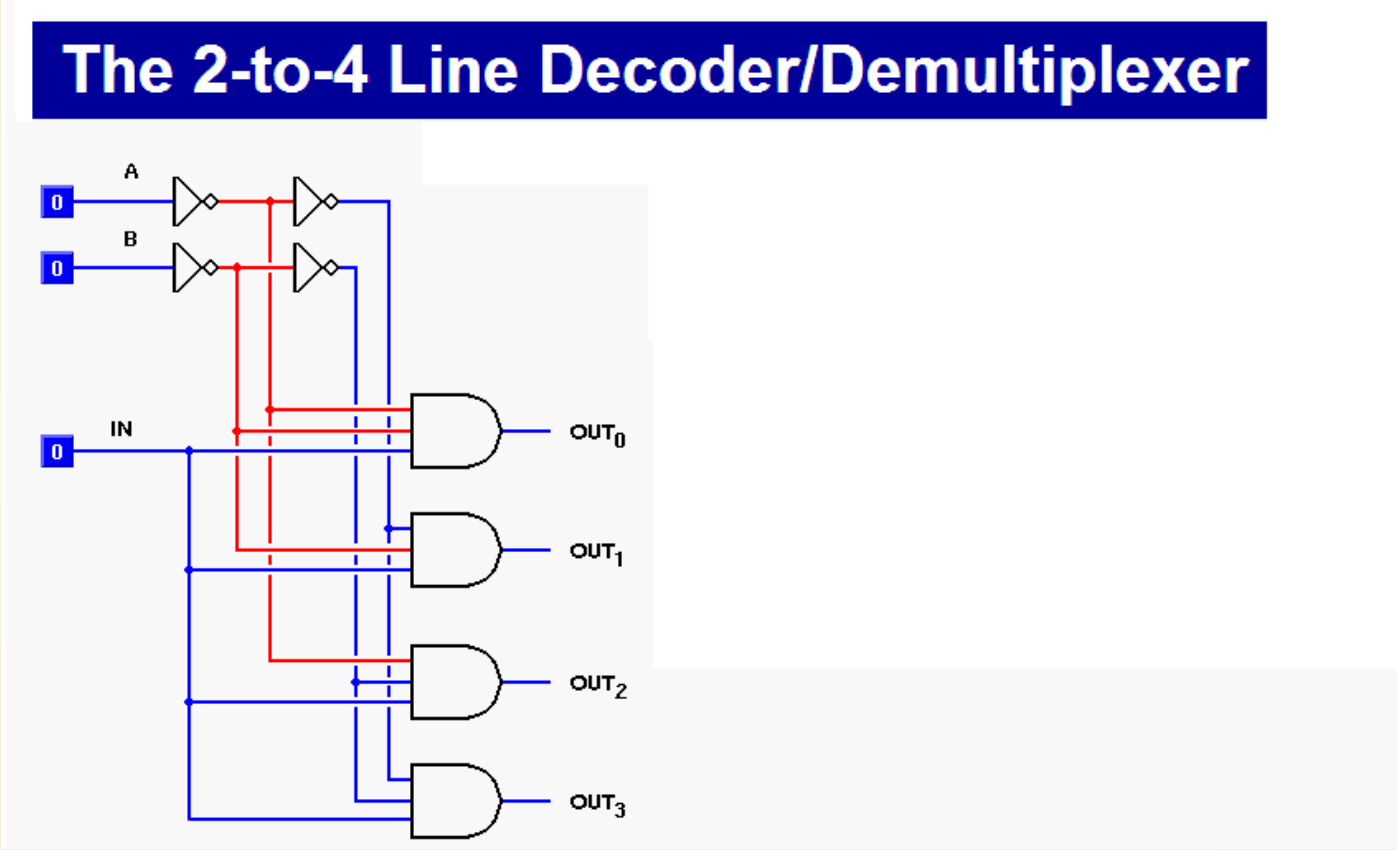
(De Multiplexer) DEMUX



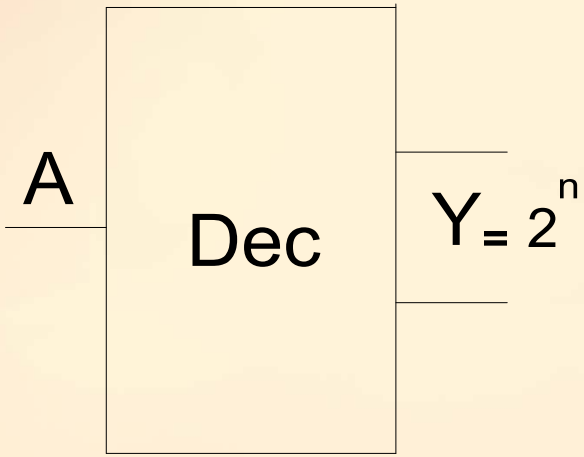
(De Multiplexer) DEMUX



2 to 4 Decoder



Decoder



$$Dec1 \times 2$$

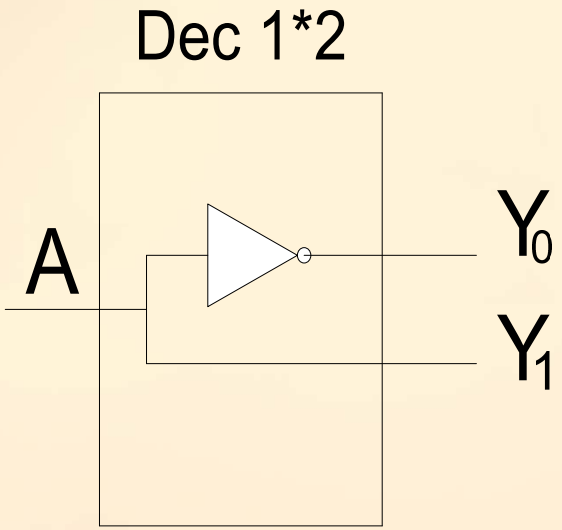
$$Dec2 \times 4$$

$$Dec3 \times 8$$

$$\vdots$$

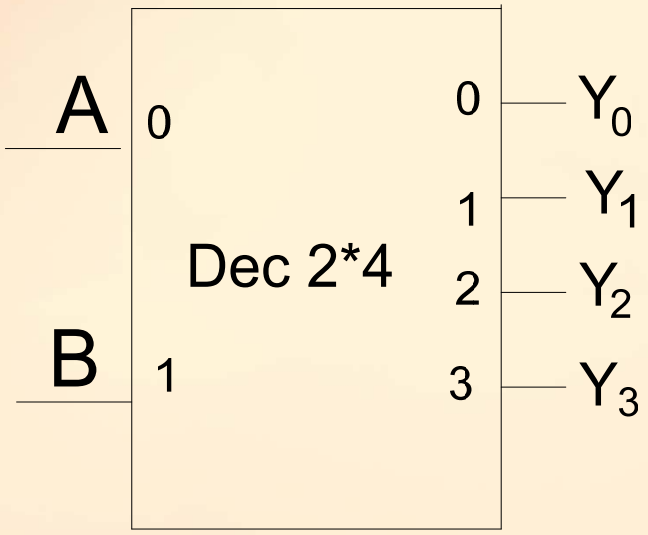
$$Decn \times 2^n \qquad n = 1, 2, 3, \ldots.$$

1-Input 2-Output Decoder



A	Y0	Y1
0	1	0
1	0	1

Decoder



B	A	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
0	0	0	0	0	1

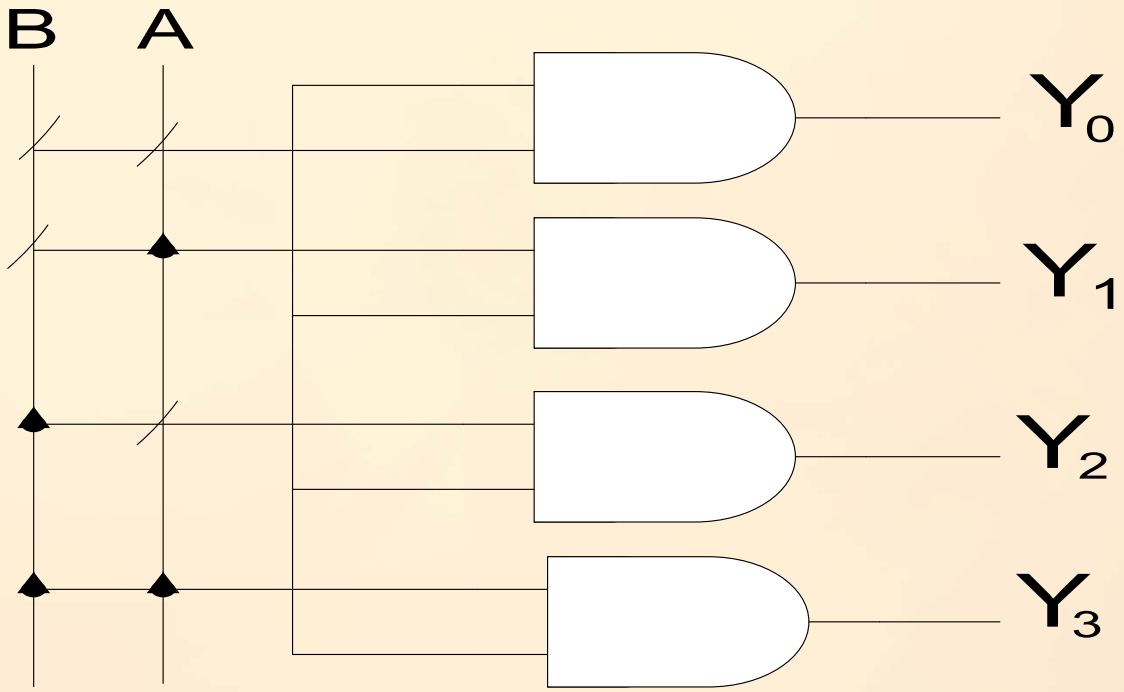
$$Y0 = A'\ B'$$

$$Y1 = A\ B'$$

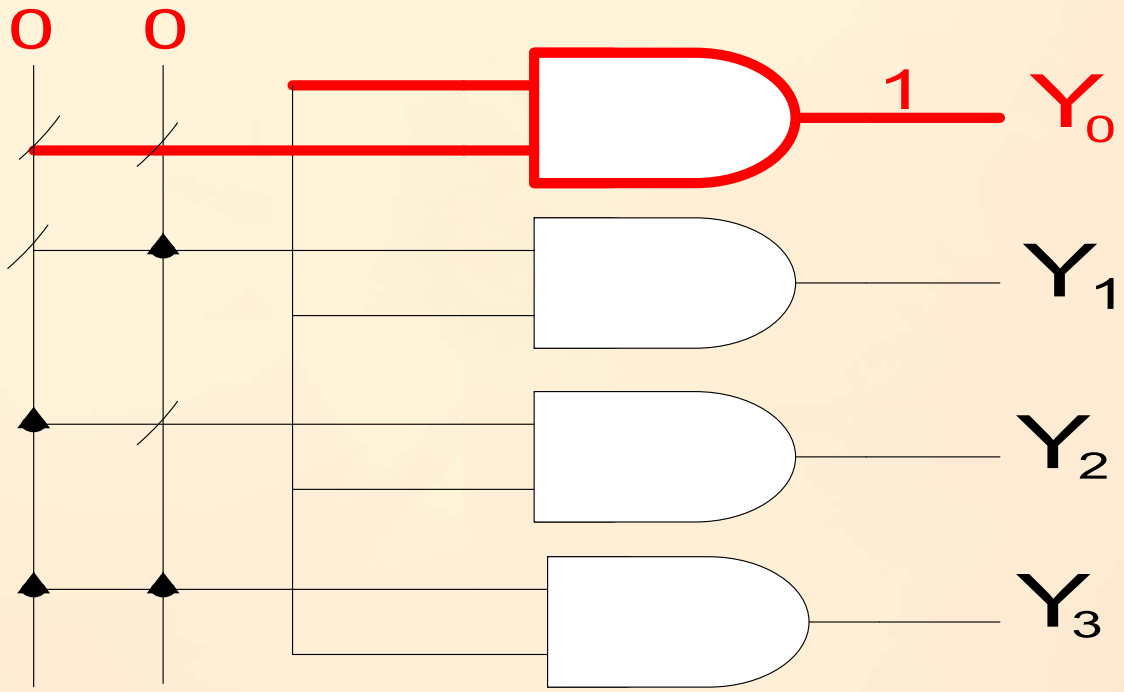
$$Y2 = A'\ B$$

$$Y3 = A\ B$$

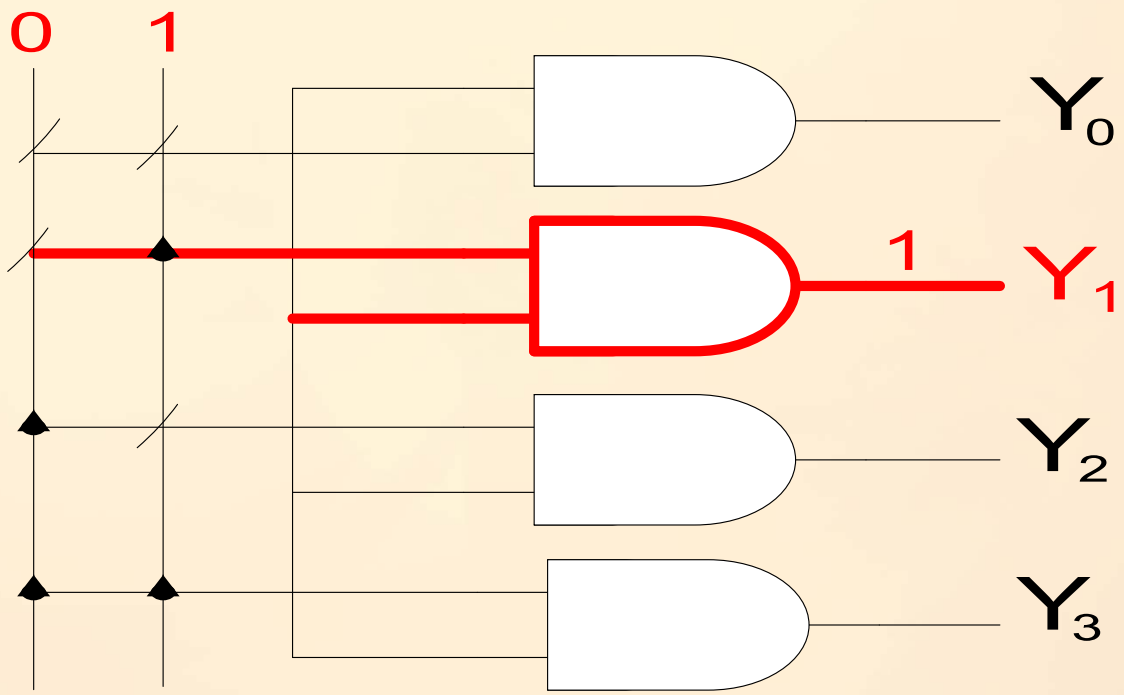
The 2 to 4 Line Decoder



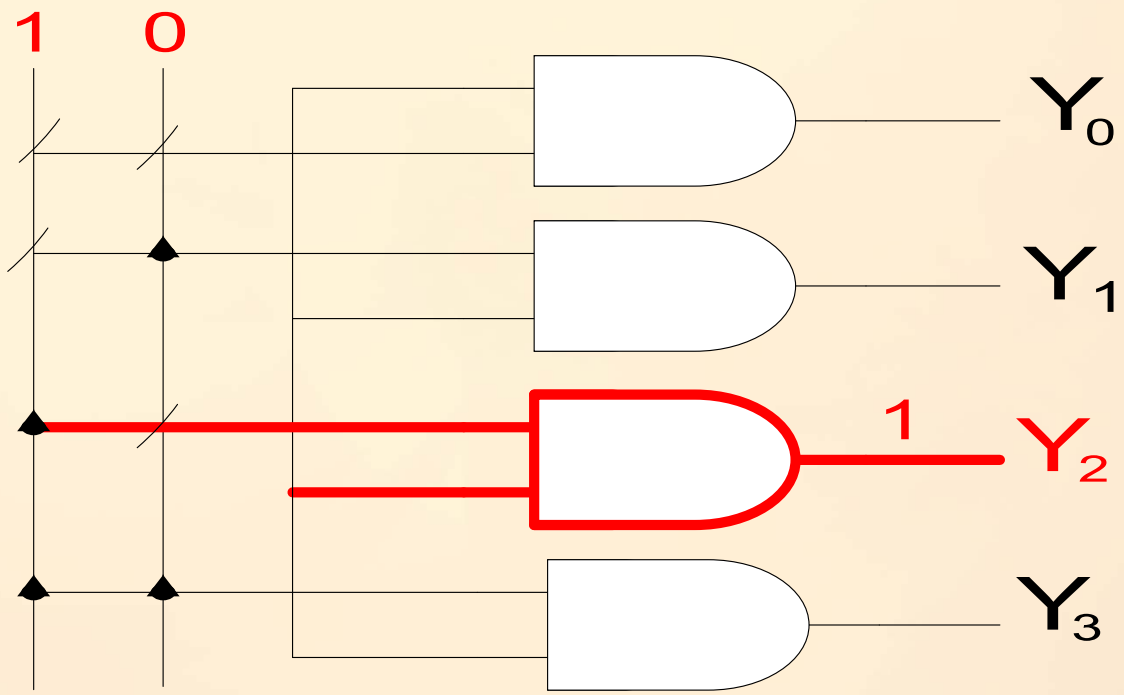
The 2 to 4 Line Decoder



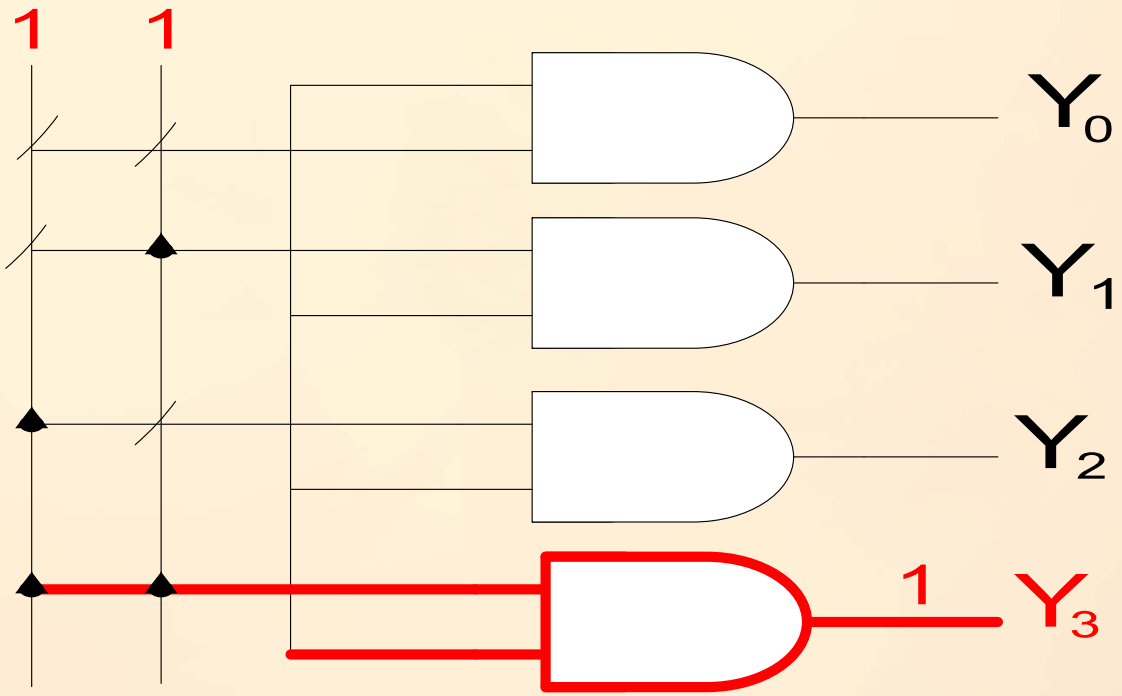
The 2 to 4 Line Decoder



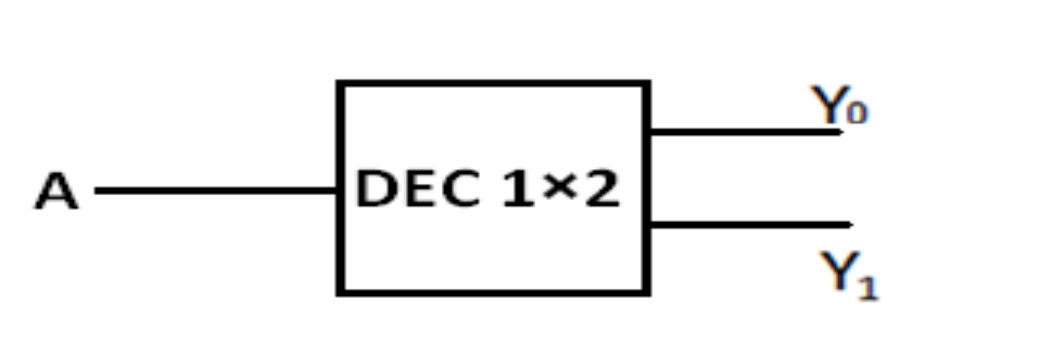
The 2 to 4 Line Decoder



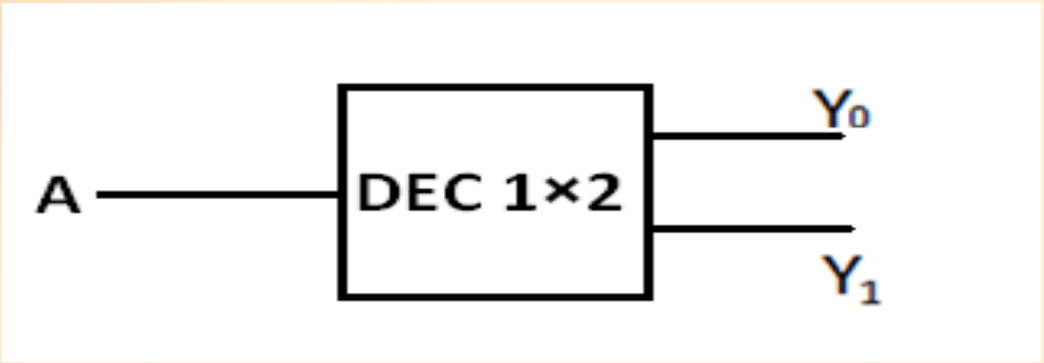
The 2 to 4 Line Decoder



(De Multiplexer) DEMUX

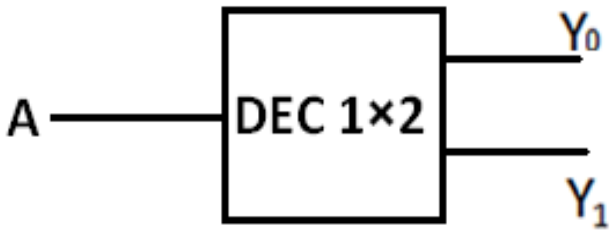


(De Multiplexer) DEMUX



A	Y_0	Y_1
0	I	0
1	0	I

(De Multiplexer) DEMUX

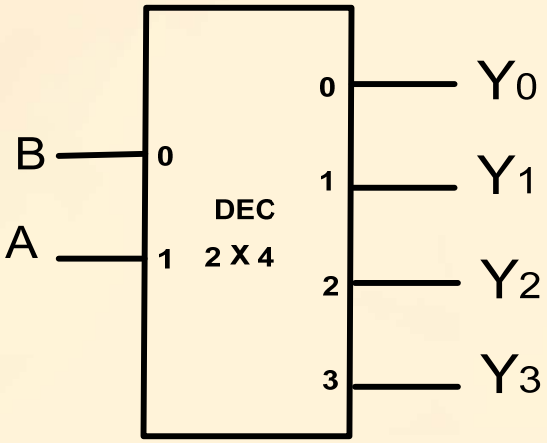


$$Y_1 = A$$

$$Y_0 = \overline{A}$$

A	Y_0	Y_1
0	I	0
1	0	I

The 2 to 4 Line Decoder



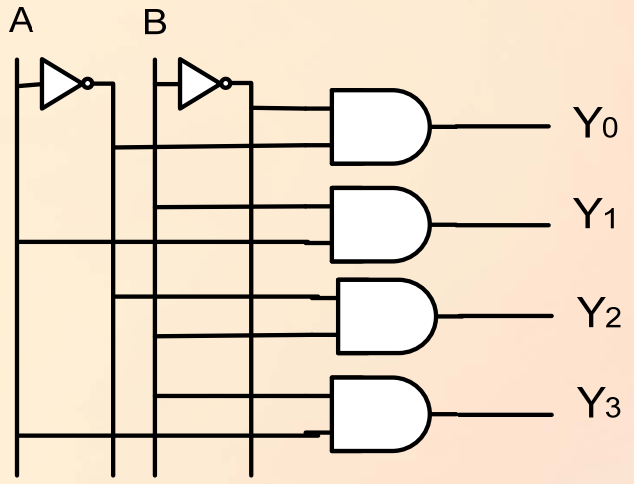
B	A	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$$Y_1 = \bar{A}B$$

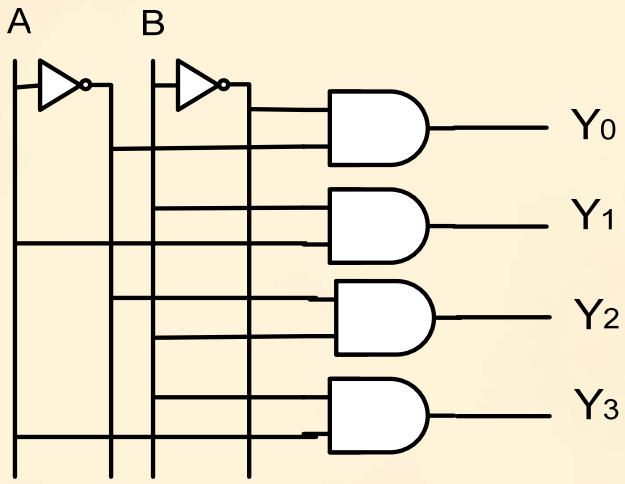
$$Y_0 = A\bar{B}$$

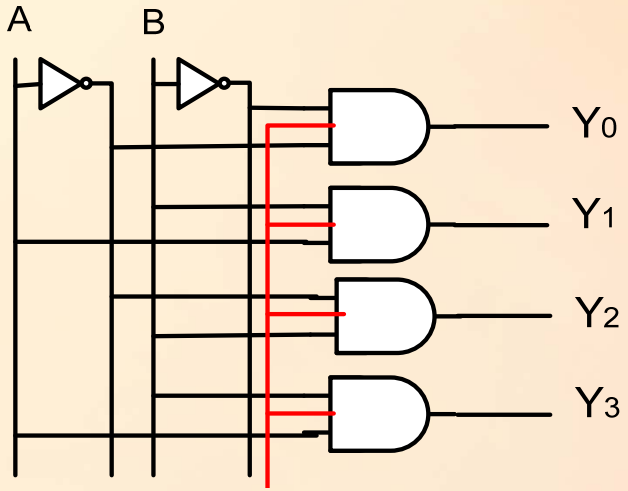
$$Y_2 = \bar{A}B$$

$$Y_3 = AB$$

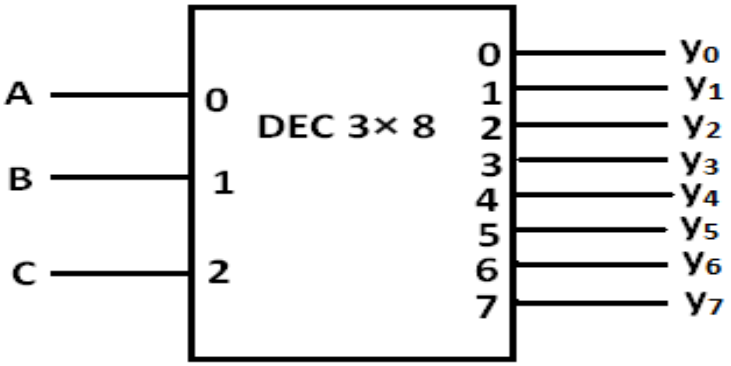


The 2 to 4 Line Decoder

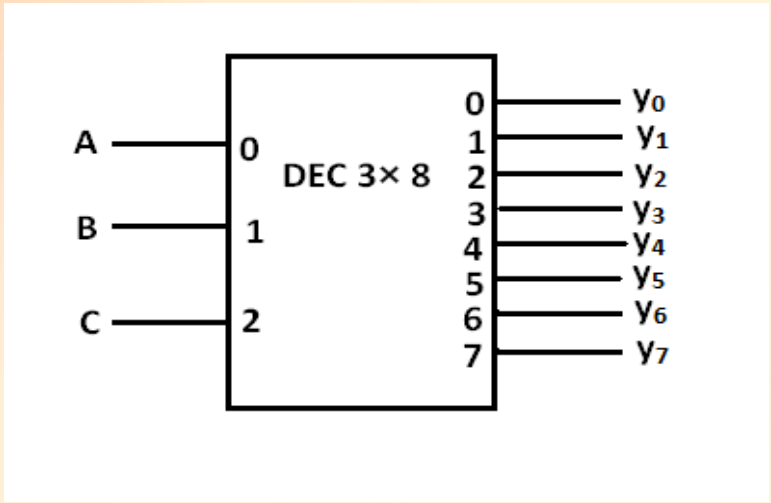




The 3 to 8 Line Decoder

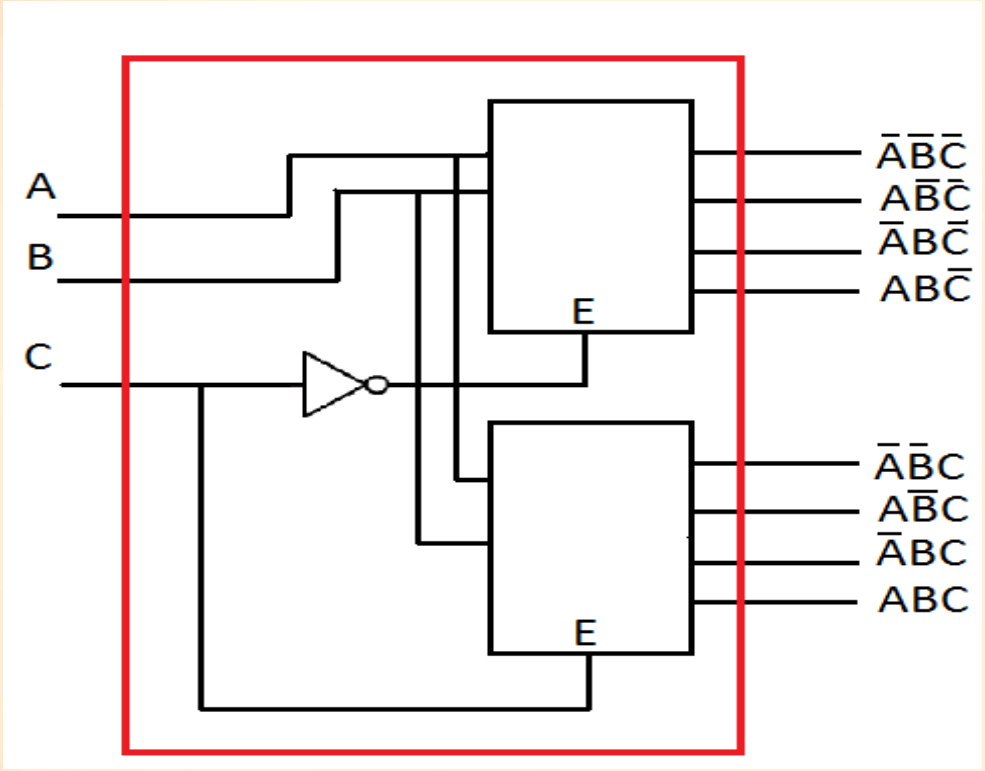


The 3 to 8 Line Decoder



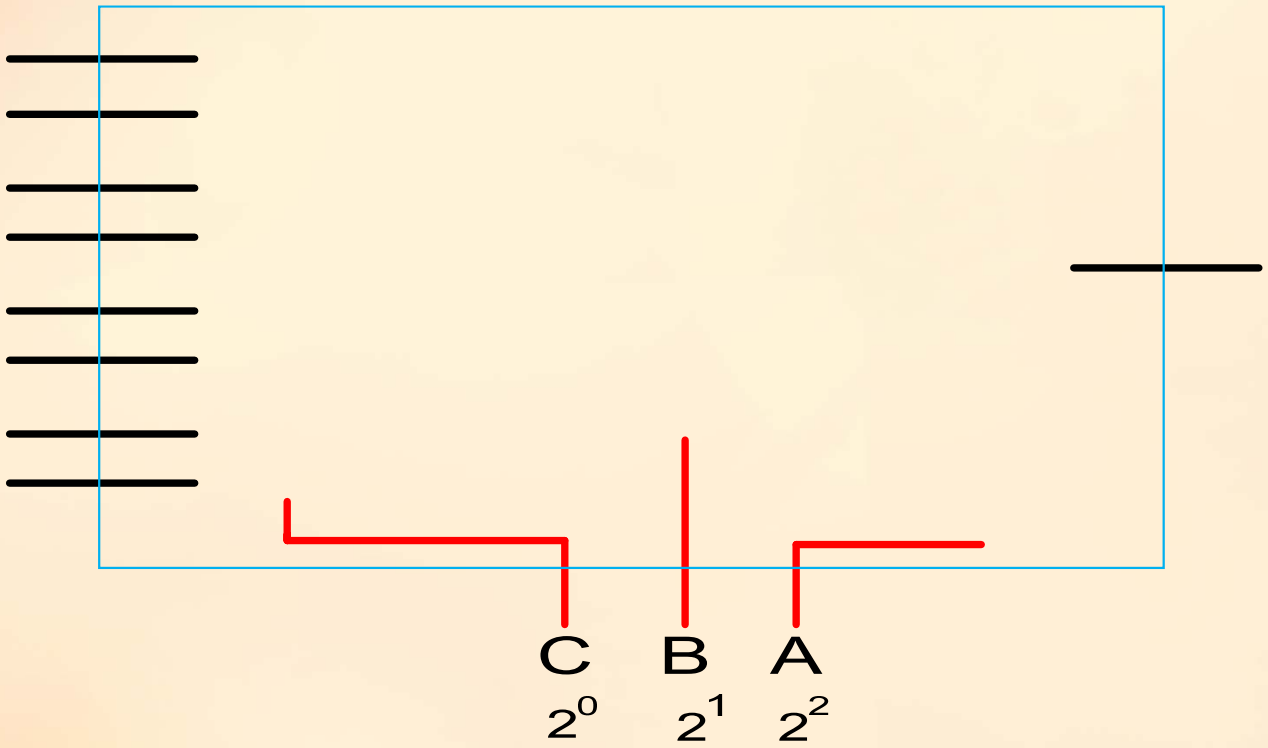
C	B	A	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

The 3 to 8 Line Decoder

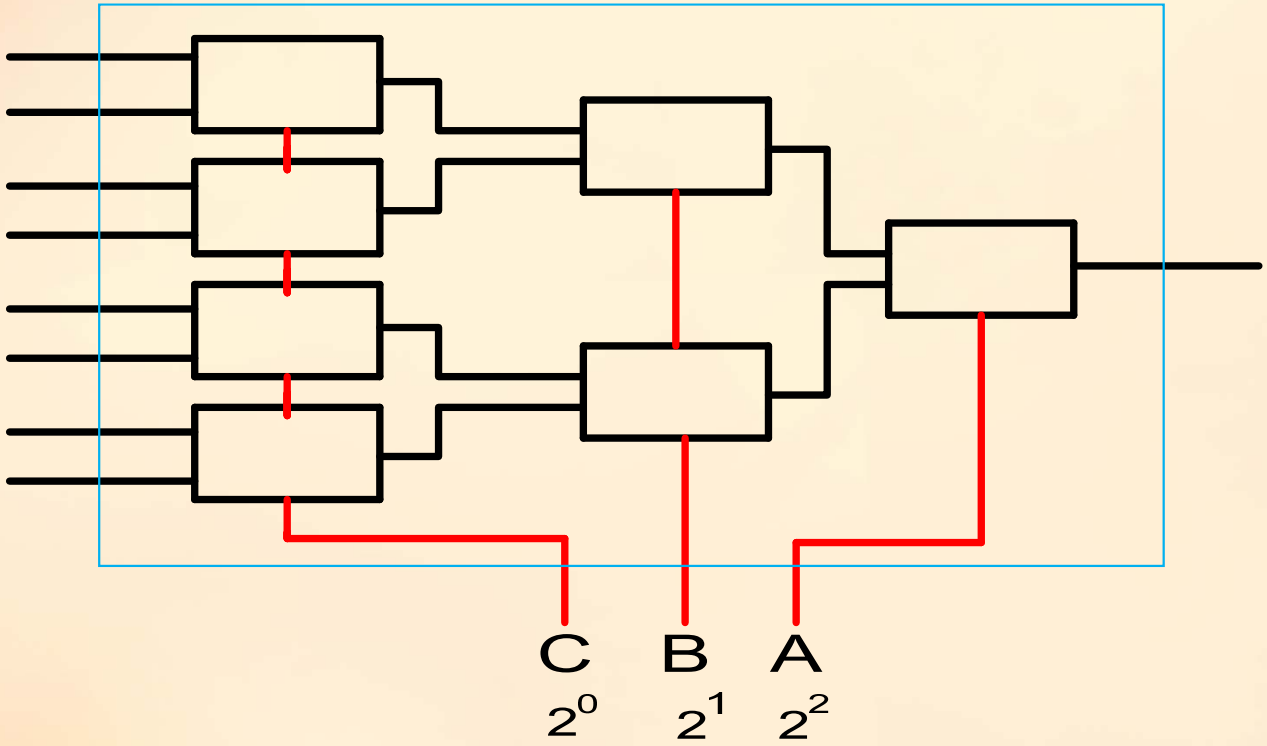


C	
0	1
1	2

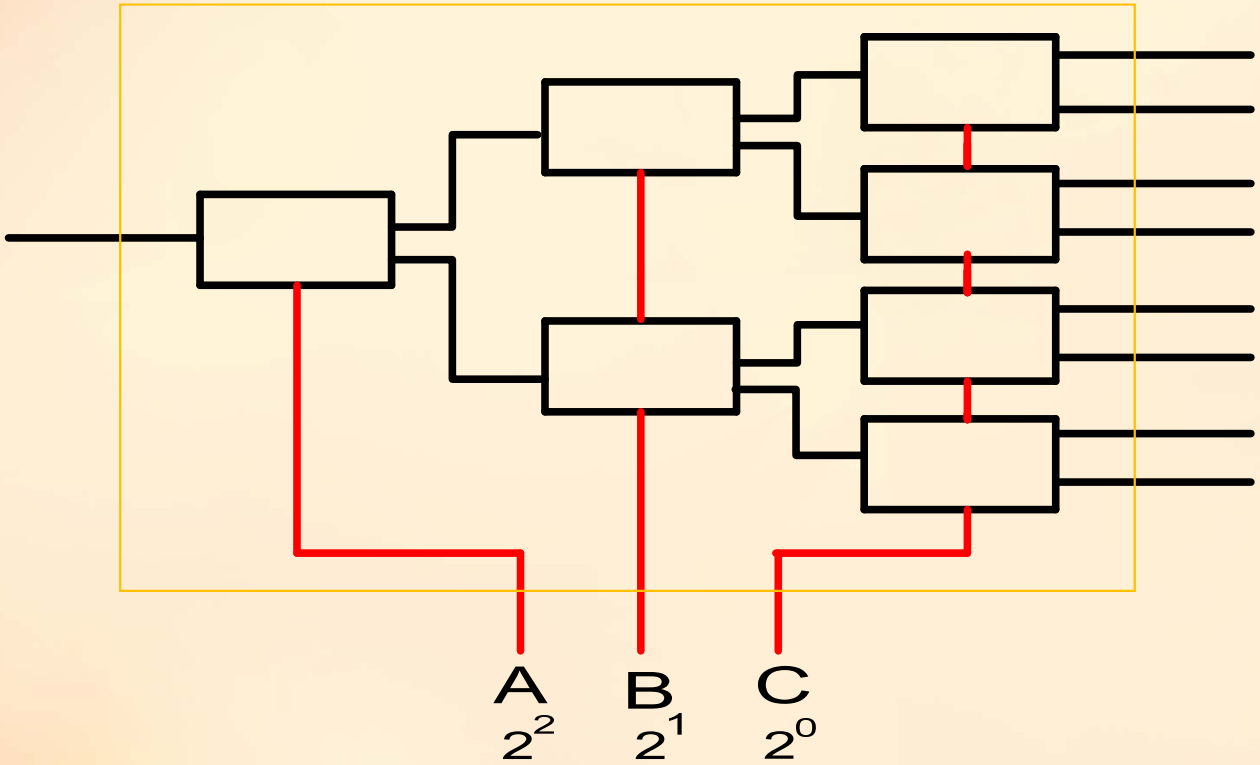
8-to-1 MUX using 2-to-1 MUX



8-to-1 MUX using 2-to-1 MUX



8-to-1 MUX using 2-to-1 MUX



8-to-1 MUX using 2-to-1 MUX

$\overline{B}$

B

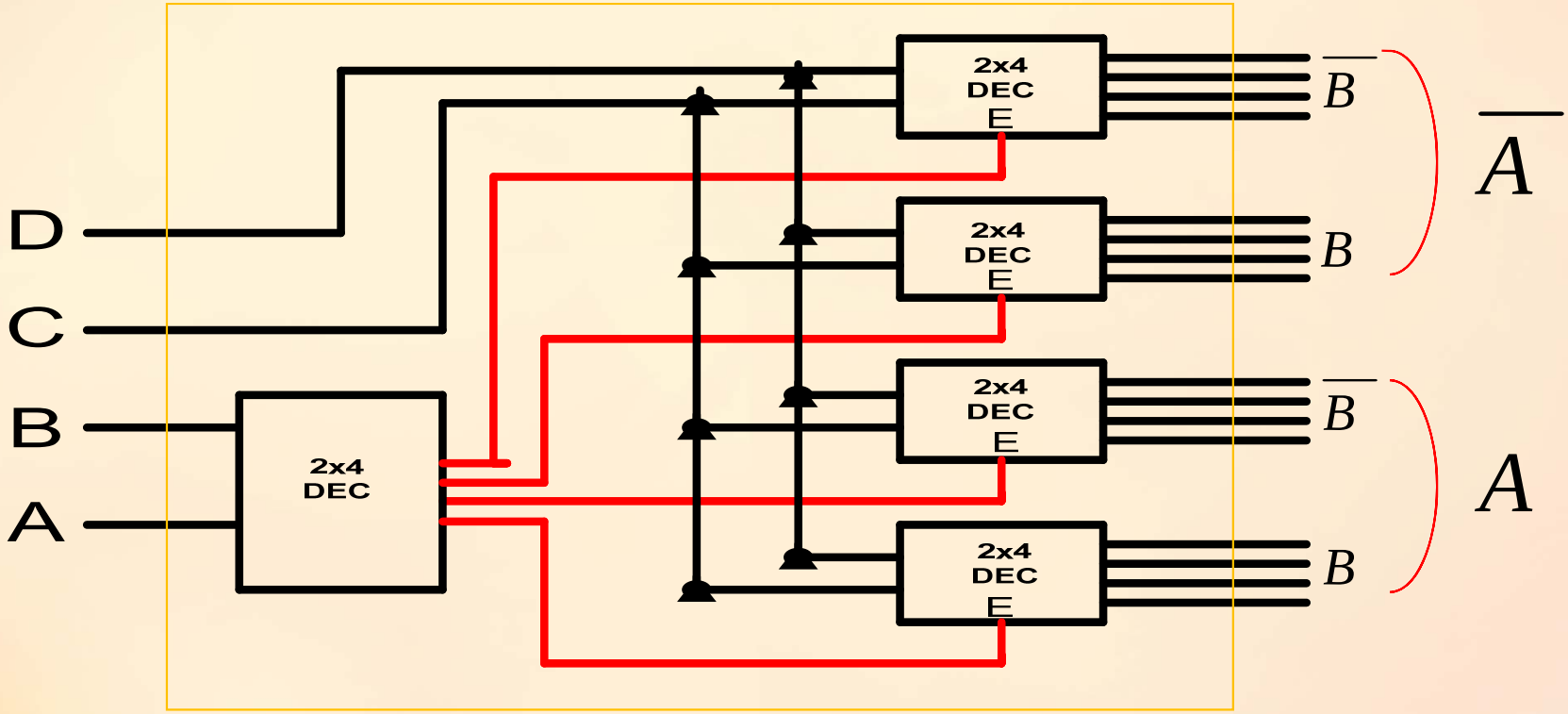
$\overline{A}$

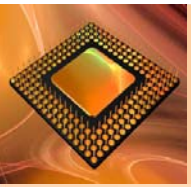
$\overline{B}$

B

A

4-to-16 Decoder using 2-to-4 Decoders





MUX and DEC Design

E#1: 4-to-1 Multiplexer using 2-to-1 Multiplexers

E#2: 8-to-1 Multiplexer using 2-to-1 Multiplexers

E#3:16-to-1 Multiplexer using 2-to-1 Multiplexers

E#4:64-to-1 Multiplexer using 2-to-1 Multiplexers

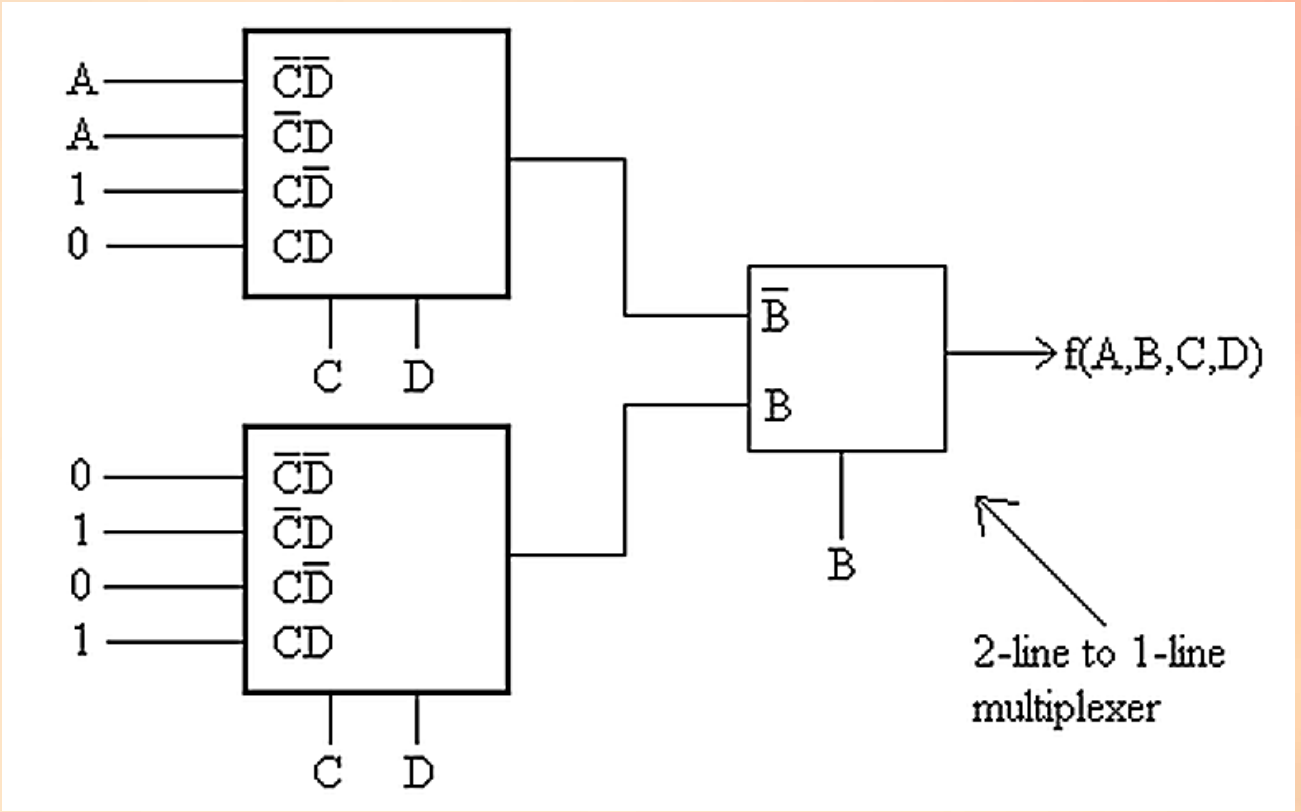
E#5:16-to-1 Multiplexer using 4-to-1 Multiplexers

E#6:32-to-1 Multiplexer using 4-to-1 Multiplexers

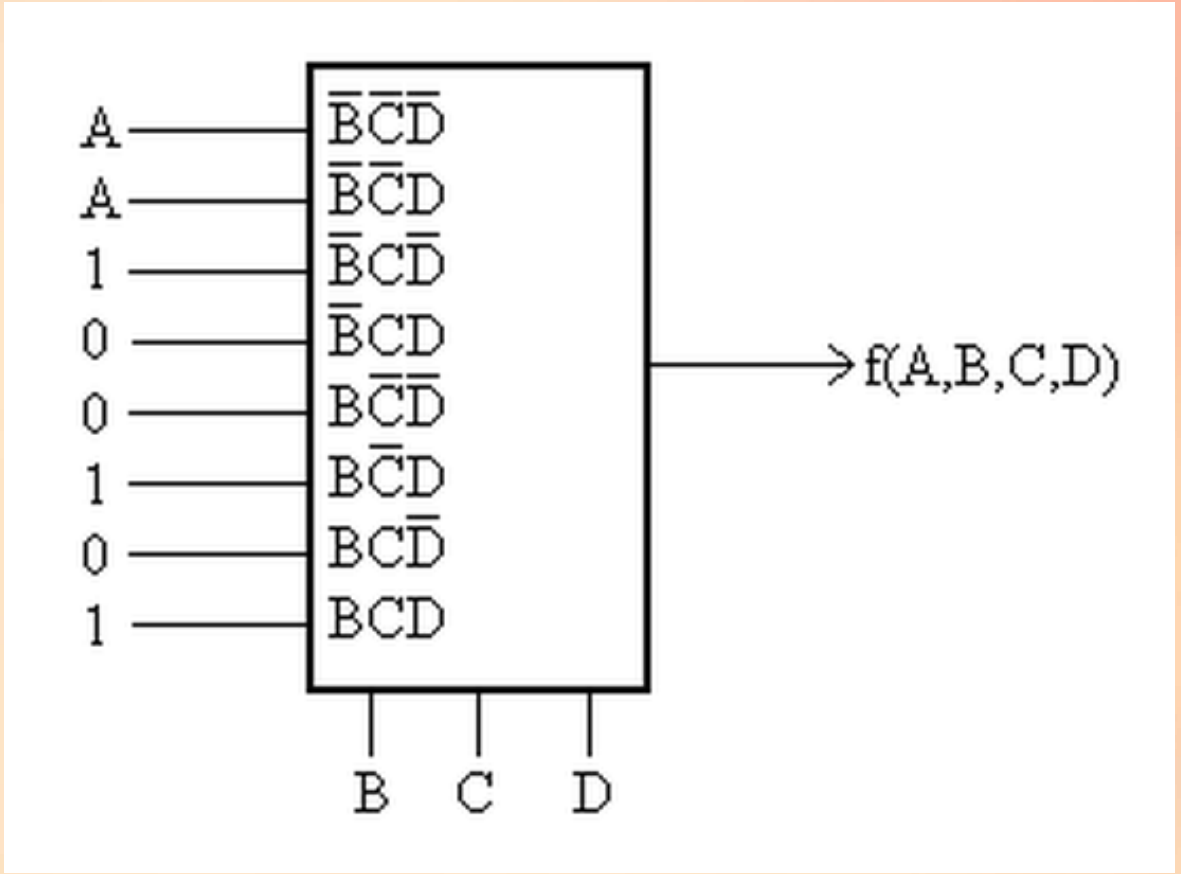
E#6:3-to-8 Decoder using 2-to-4 Decoders with Enable

E#7:4-to-16 Decoder using 2-to-4 Decoders with Enable

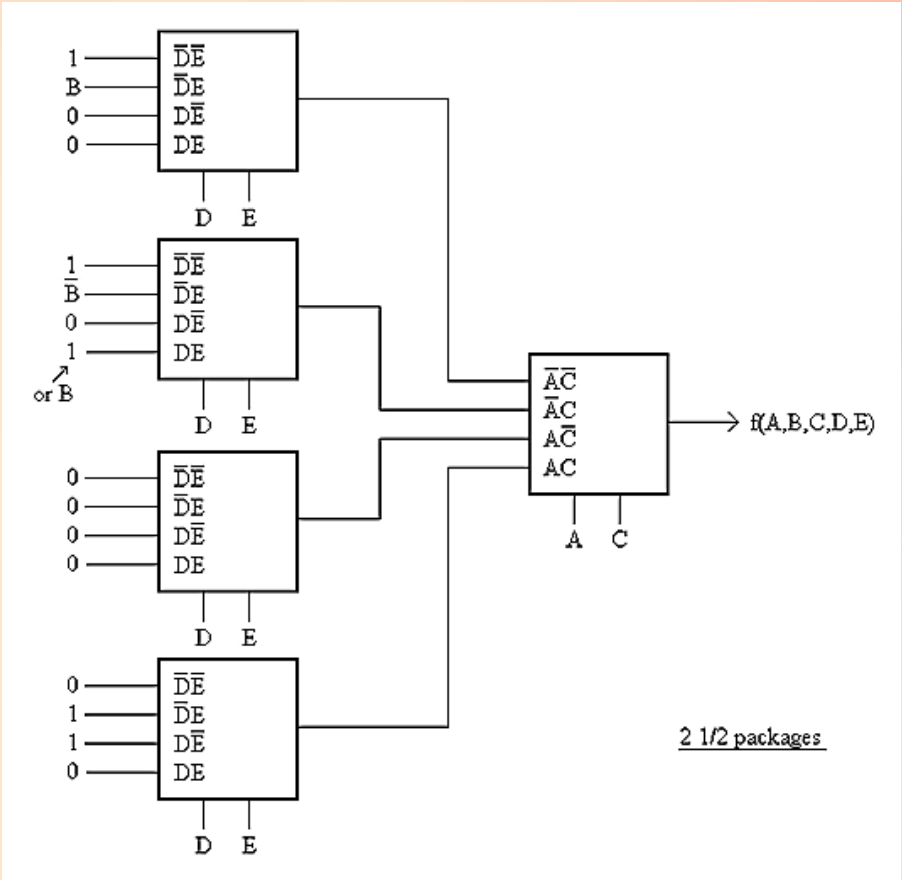
MUX design


A
A
1
0
$\bar{C}\bar{D}$
$\bar{C}D$
$C\bar{D}$
CD
C
D
$\bar{B}$
B
f(A,B,C,D)
B
0
1
0
1
$\bar{C}\bar{D}$
$\bar{C}D$
$C\bar{D}$
CD
C
D
2-line to 1-line multiplexer

Implementing logic Function using MUX

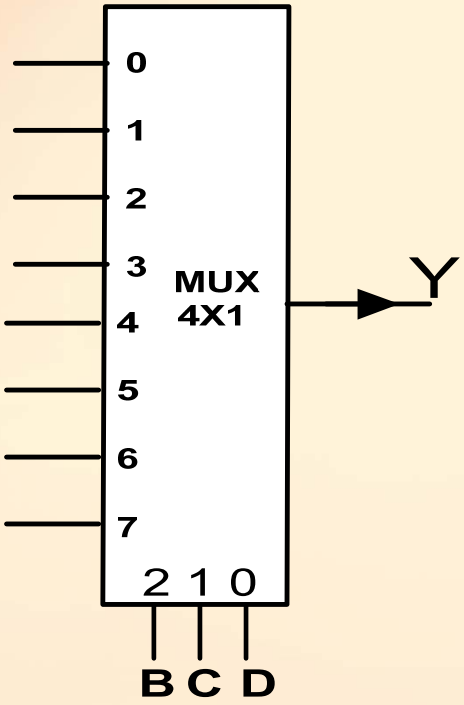


Implementing logic Function using MUX

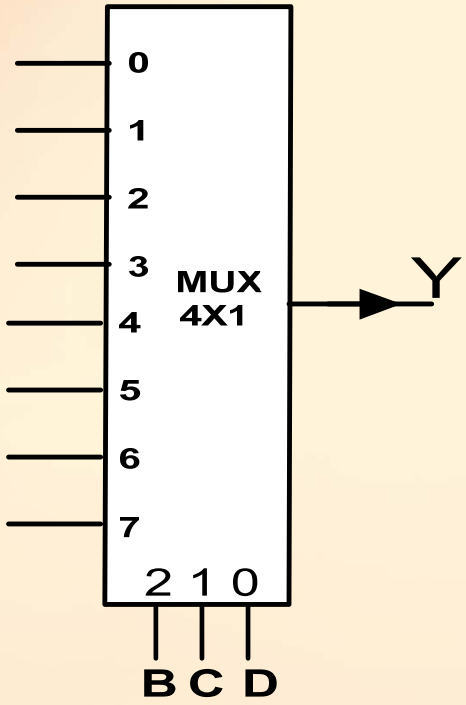


Implementing logic Function using MUX

$$Y=\sum m(0,1,4,5,8,9,12,13,15) + d(3,10,11)$$



Boolean Function Implementation using Multiplexers



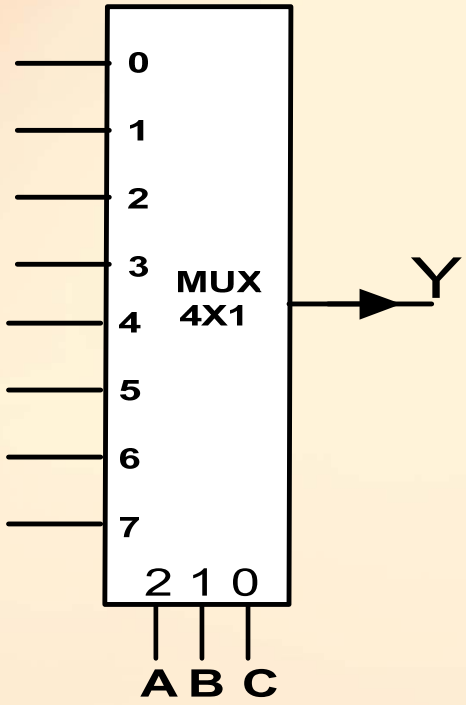
$$Y=\sum m(0,1,4,5,8,9,12,13,15) + d(3,10,11)$$

D	0	1	0	1	0	1	0	1
C	0	0	1	1	0	0	1	1
B	0	0	0	0	1	1	1	1

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A’								
A								

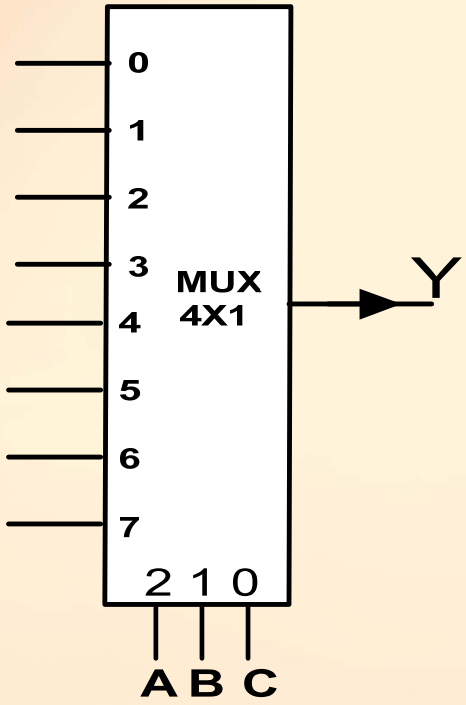
Boolean Function Implementation using Multiplexers

$$Y=\sum m(0,1,4,5,8,9,12,13,15) + d(3,10,11)$$



B	0	1	0	1	0	1	0	1
D	0	0	1	1	0	0	1	1
A	0	0	0	0	1	1	1	1
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
C'	0	4	1	5	8	12	9	13
C	2	6	3	7	10	14	11	15

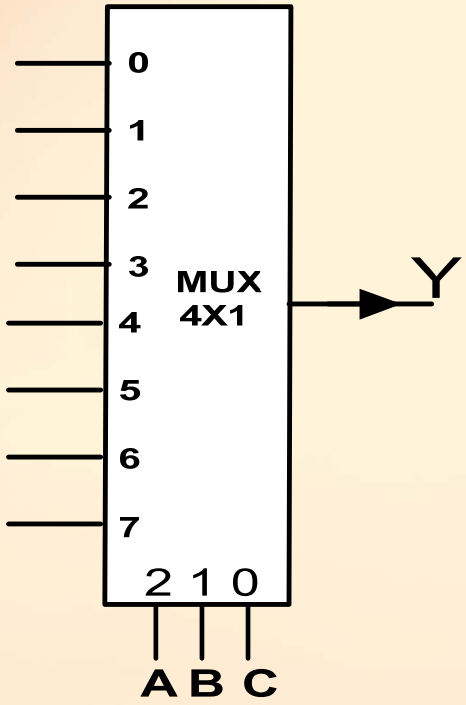
Boolean Function Implementation using Multiplexers



$$Y=\sum m(0,1,4,5,8,9,12,13,15) + d(3,10,11)$$

B	0	1	0	1	0	1	0	1
D	0	0	1	1	0	0	1	1
A	0	0	0	0	1	1	1	1
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
C'	0	4	1	5	8	12	9	13
C	2	6	3	7	10	14	11	15

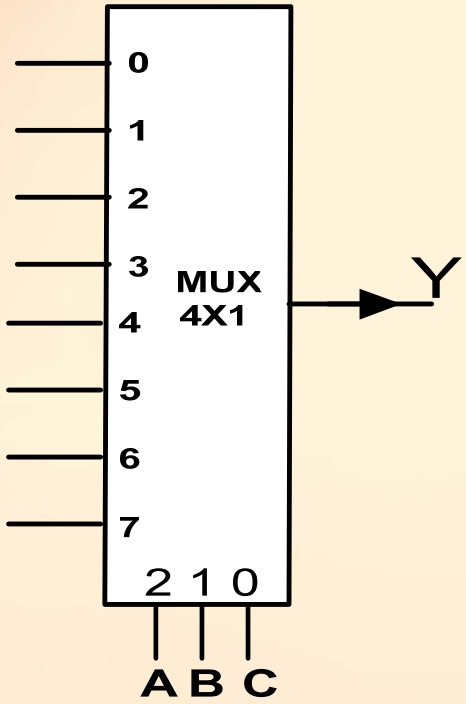
Boolean Function Implementation using Multiplexers



$$Y=\sum m(0,1,4,5,8,9,12,13,15) + d(3,10,11)$$

B	0	1	0	1	0	1	0	1
D	0	0	1	1	0	0	1	1
A	0	0	0	0	1	1	1	1
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
C'	0	4	1	5	8	12	9	13
C	2	6	3	7	10	14	11	15

Boolean Function Implementation using Multiplexers

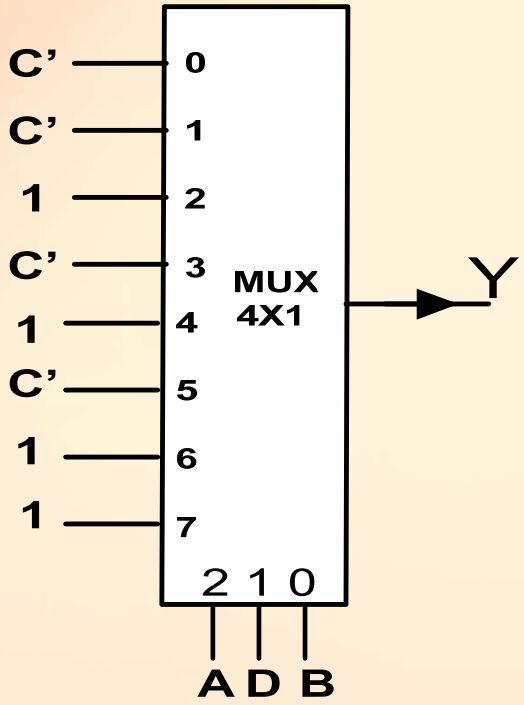


$$Y=\sum m(0,1,4,5,8,9,12,13,15) + d(3,10,11)$$

B	0	1	0	1	0	1	0	1
D	0	0	1	1	0	0	1	1
A	0	0	0	0	1	1	1	1
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
C'	0	4	1	5	8	12	9	13
C	2	6	3	7	10	14	11	15
	C'	C'	1	C'	1	C'	1	1

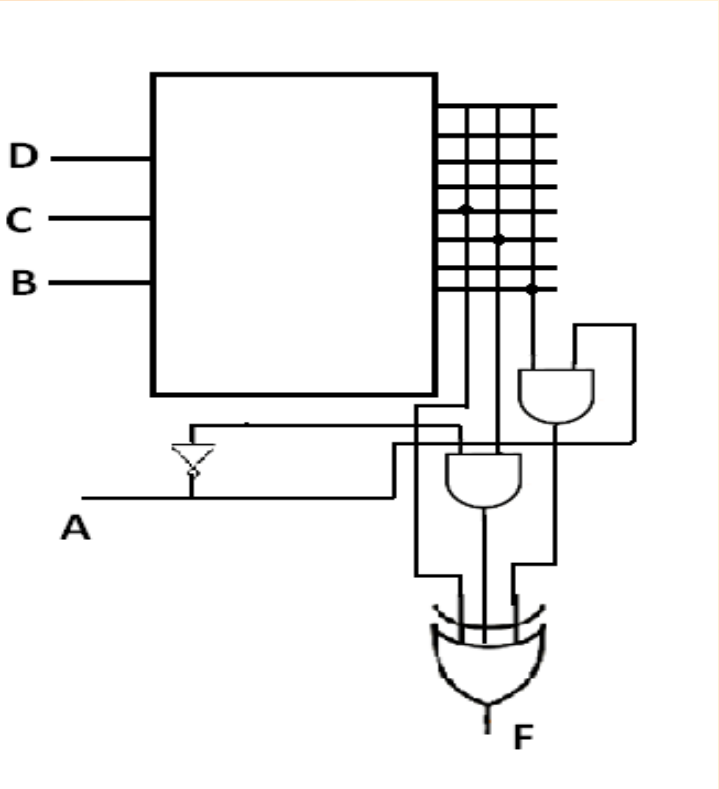
Boolean Function Implementation using Multiplexers

$$Y=\sum m(0,1,4,5,8,9,12,13,15) + d(3,10,11)$$



B	0	1	0	1	0	1	0	1
D	0	0	1	1	0	0	1	1
A	0	0	0	0	1	1	1	1
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
C'	0	4	1	5	8	12	9	13
C	2	6	3	7	10	14	11	15
	C'	C'	1	C'	1	C'	1	1

Boolean Function Implementation using Multiplexers



	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15

Thank You