Digital Logic Design

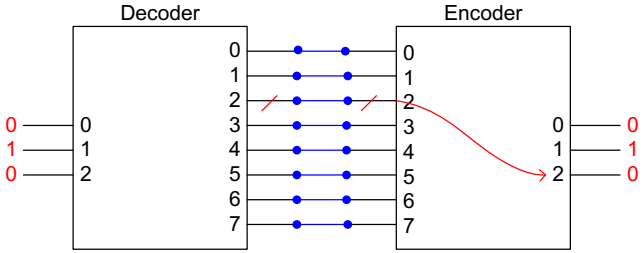
Hajar Falahati

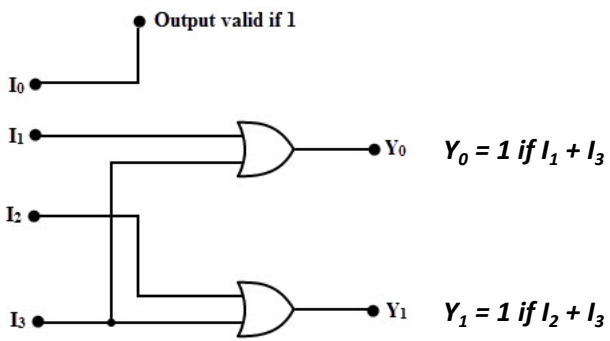
Department of Computer Engineering
IRAN University of Science and Technology

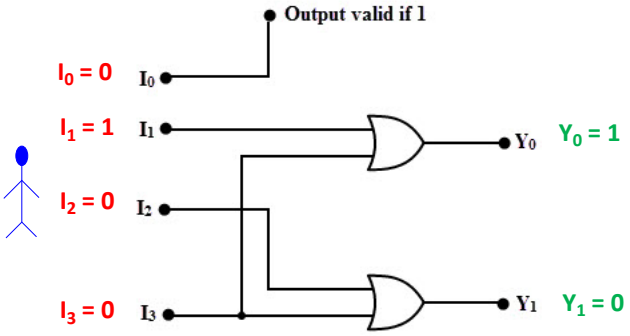
hfalahati@iust.ac.ir

Encoder

- 2^n inputs and n outputs
- At each time only one input can be active
- **Generates the binary code corresponding to the input values**







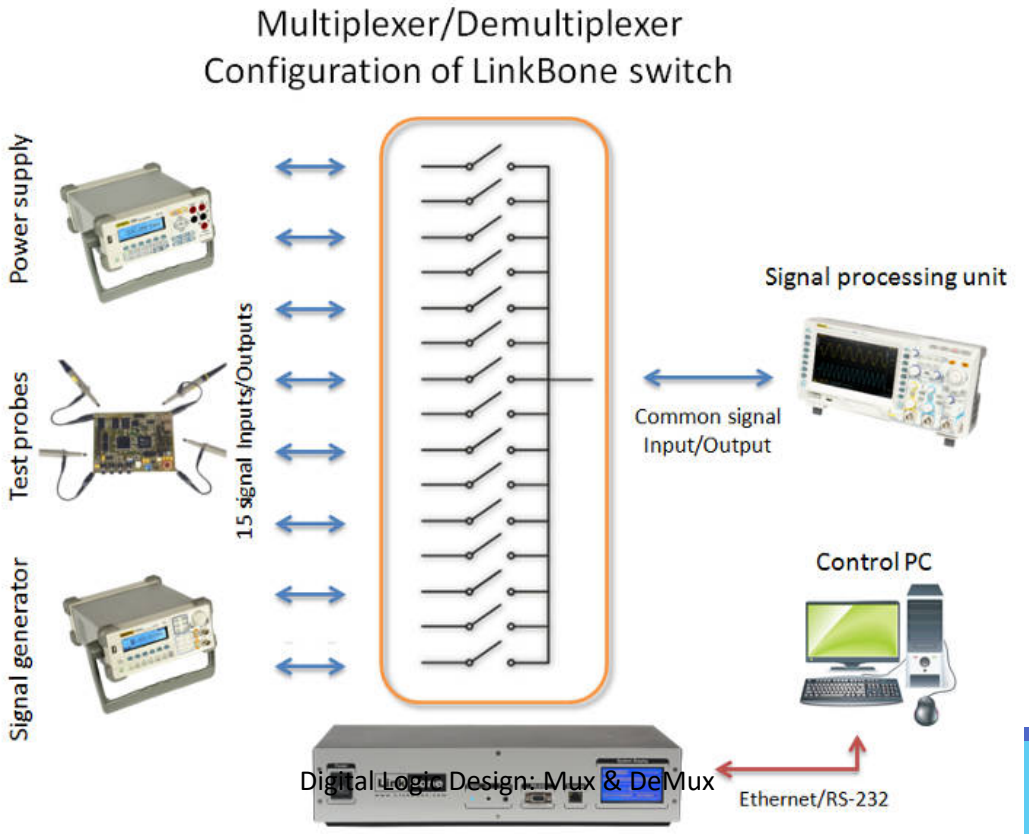
Outline

- Multiplexer
- Demultiplexer

Multiplexer

Selecting

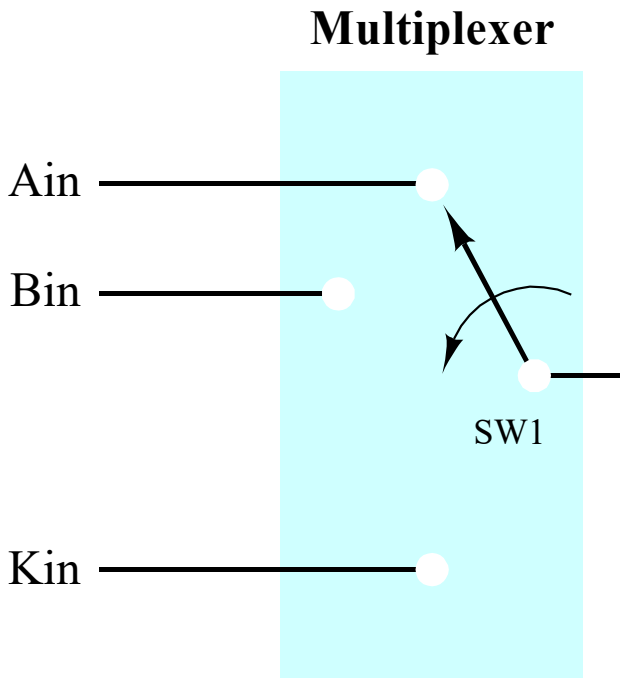
- Suppose a logic has more than one candidates to its input
- Logic should select one of candidates
- Which candidate does logic select as its input?



Selection Logic

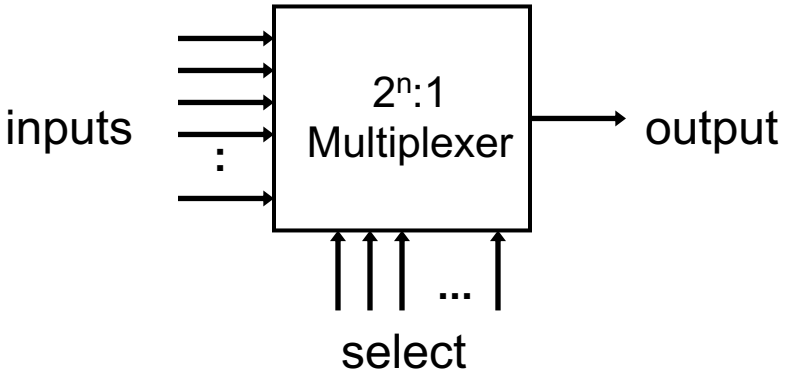
- Characteristics of selection logic
  - A set of information inputs from which the selection is made
  - A set of control lines for making the selection
  - A single output

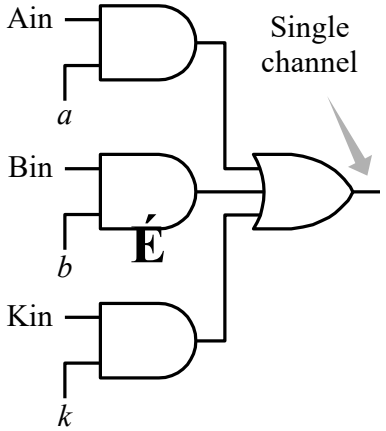
- Realization of selection logic
  - Multiplexer, a.k.a., MUX



Multiplexer

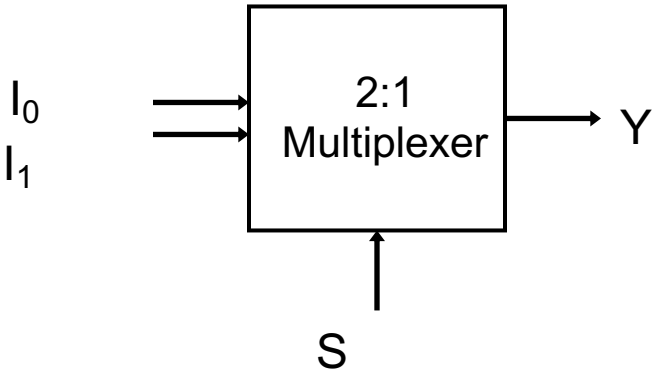
- Input
  - n selection lines
  - 2^n input lines
- Output
  - 1 output line
- Function
  - Uses ***n* selection** lines
  - Selects **one** of 2^n inputs to a single output line





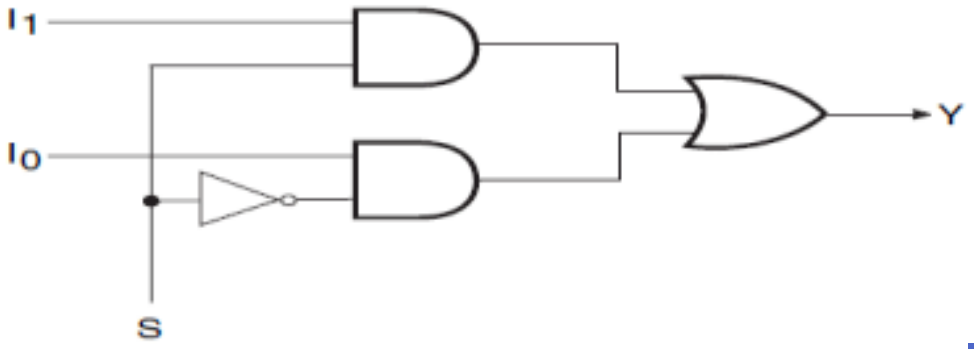
Multiplexer: 2-1

- Input
  - 1 selection lines; S
  - 2 input lines ; I_0 and I_1
- Output
  - S =0 ; output = I_0
  - S =1 ; output = I_1



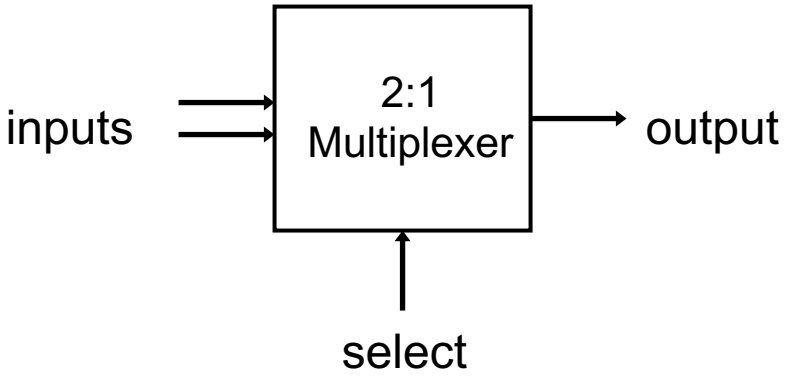
S	Y
0	I_0
1	I_1

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

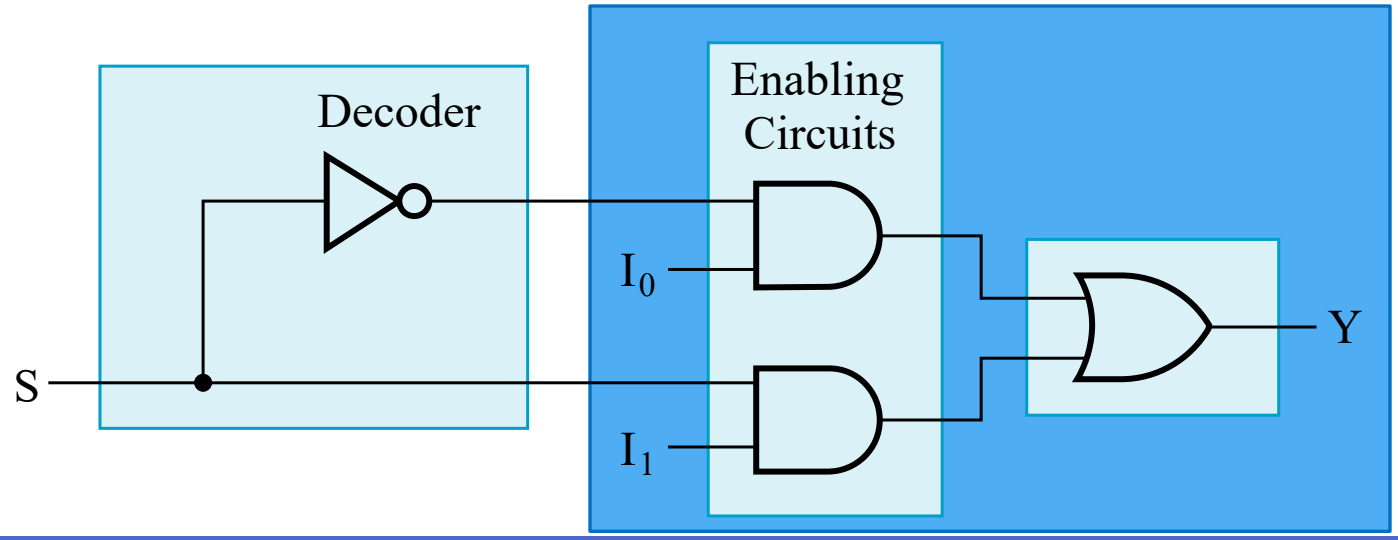


Multiplexer 2-1: More Details

- Three parts
  - 1-2 Decoder
  - 2 enabling circuits
  - 2-input OR gats

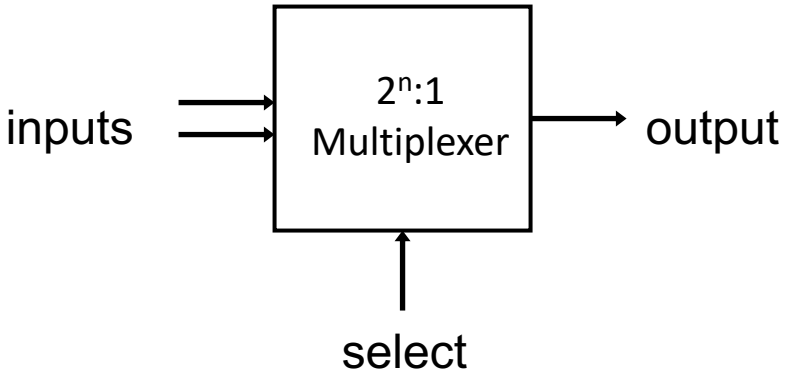


- 2 × 2 AND-OR circuit



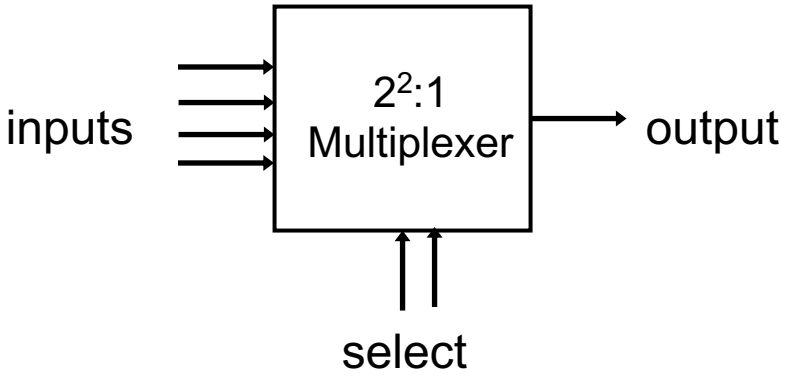
Multiplexer: 2^n -1

- Three parts
  - n-2^n Decoder
  - $2^n \times 2$ AND-OR circuit



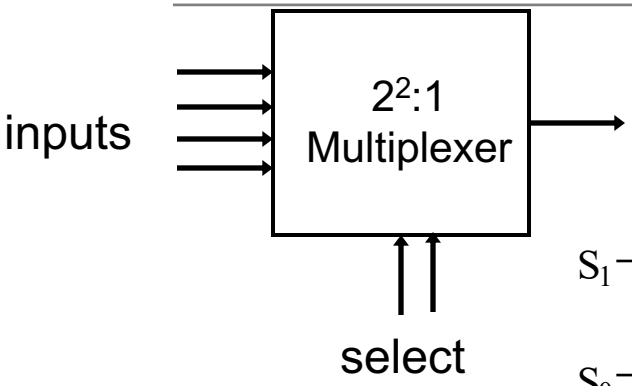
Multiplexer: 2^2- 1

- Three parts
  - 2-2^2 Decoder
  - $2^2 \times 2$ AND-OR circuit

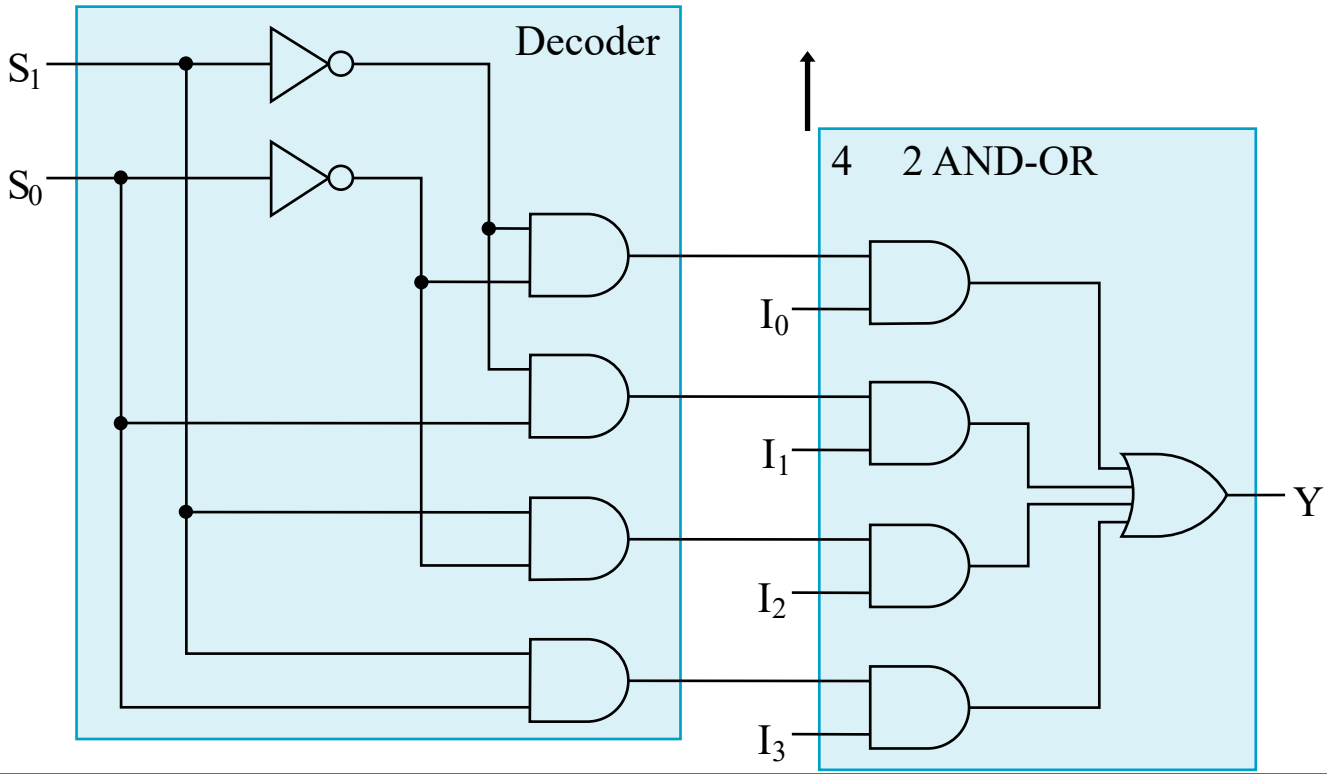


S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

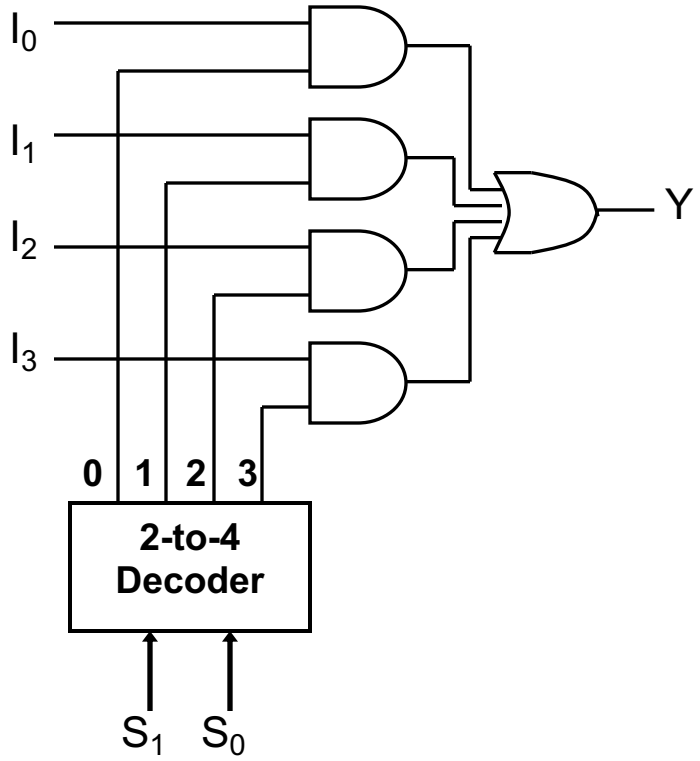
Multiplexer 2^2-1: Realization



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

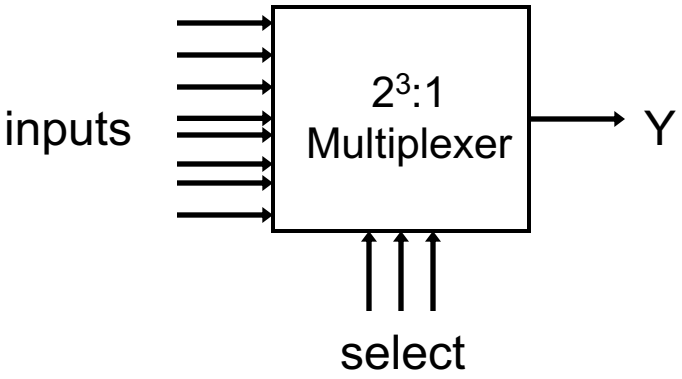


Multiplexer 2^2-1: Matrix Simplification



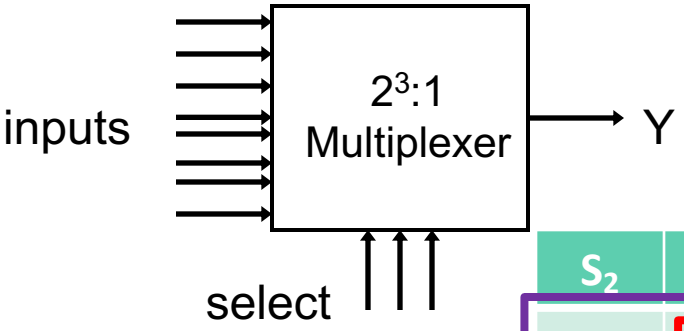
Expansion

- Constructed from smaller MUXs
- Sample: Construct an 8-1 MUX

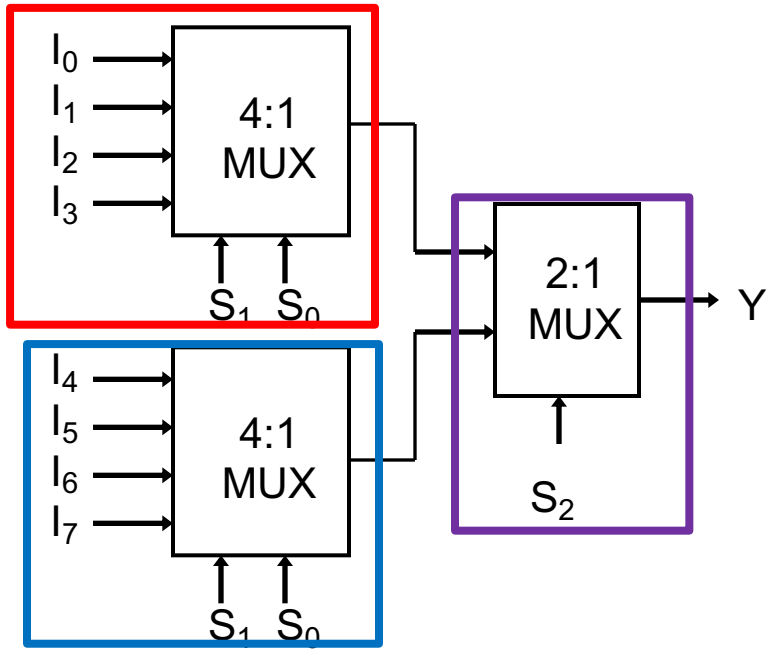


S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Expansion: 8-1 MUX

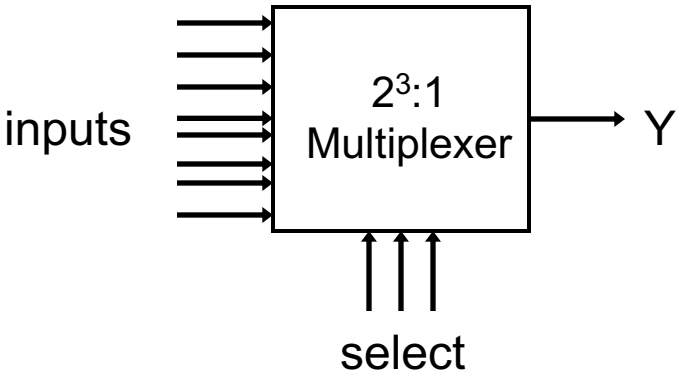


S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



Sample 1

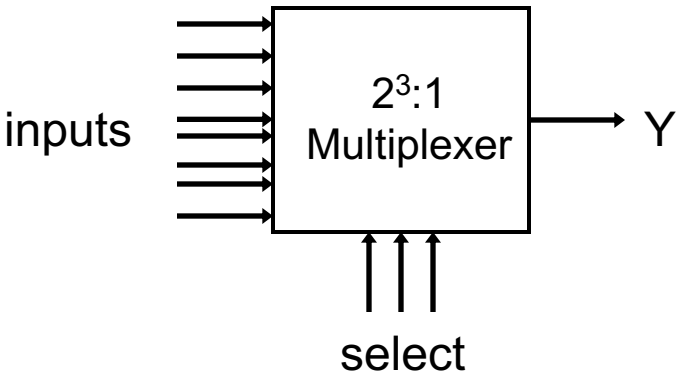
- Construct a 8-1 MUX using 2-1 MUX



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

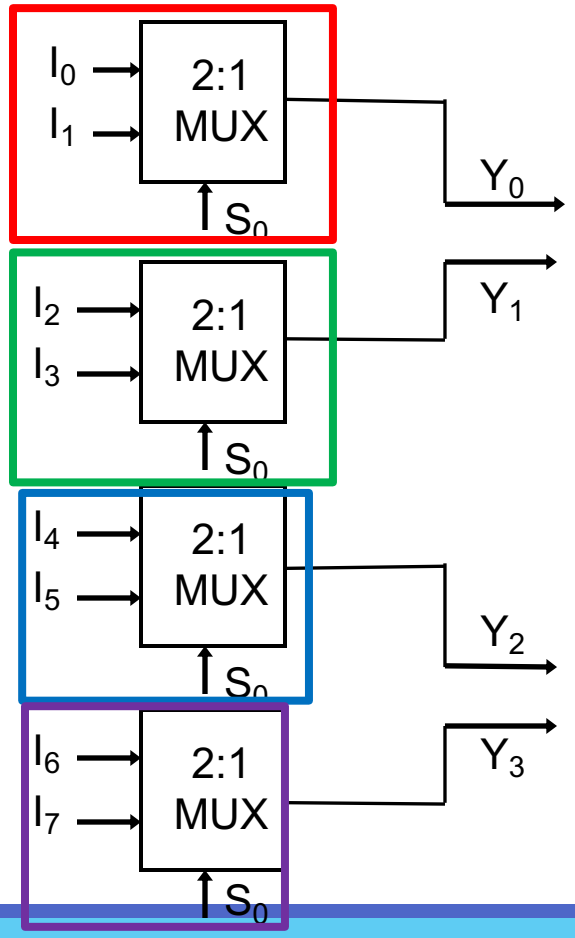
Sample 1 (cont'd)

- Construct a 8-1 MUX using 2-1 MUX



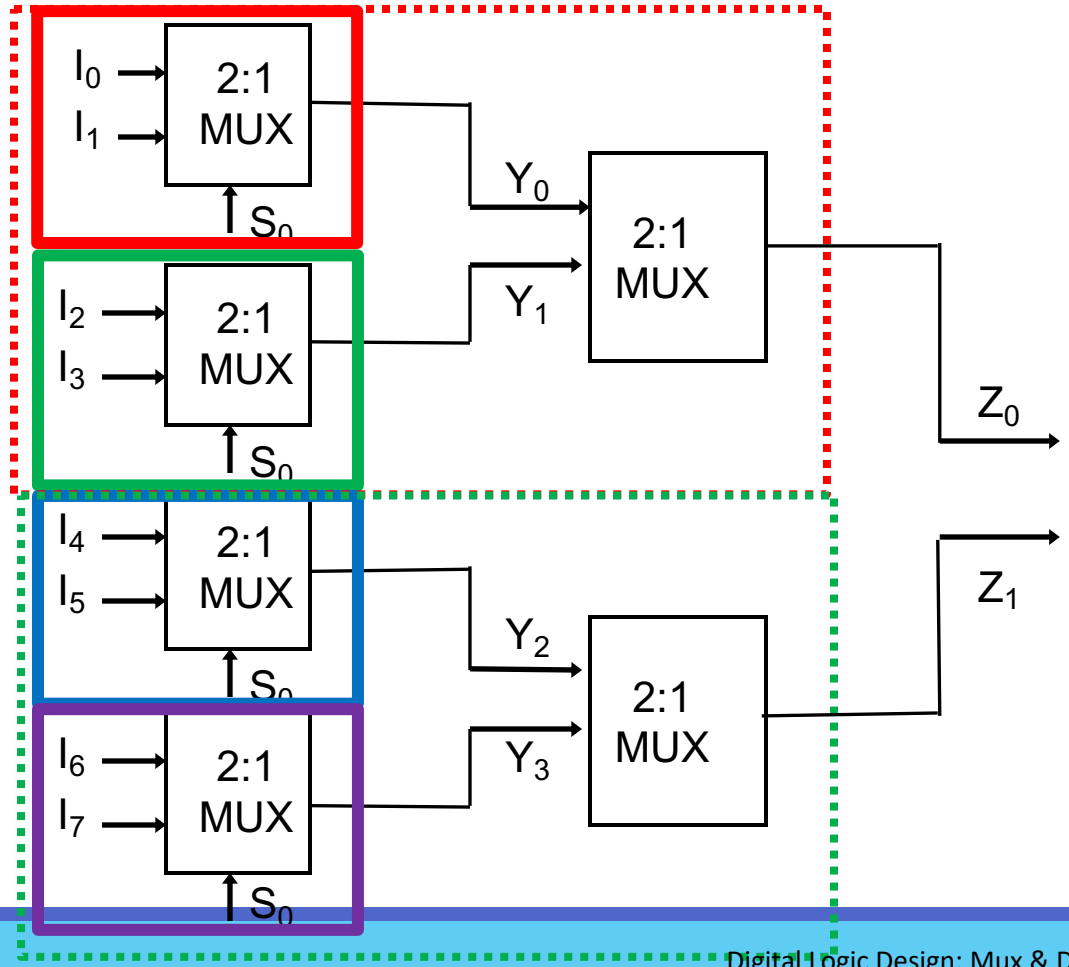
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Sample 1 (cont'd)



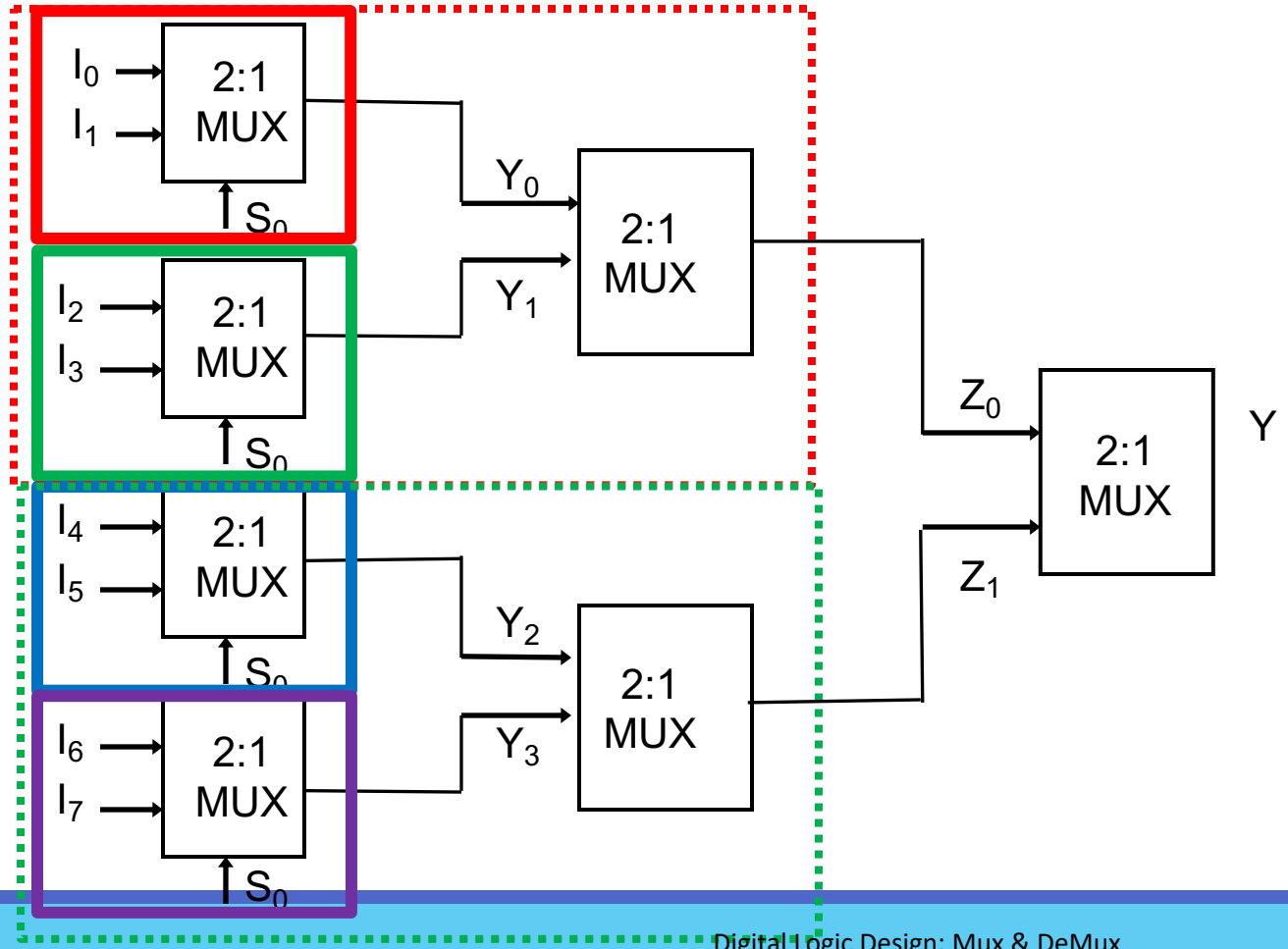
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Sample 1 (cont'd)



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

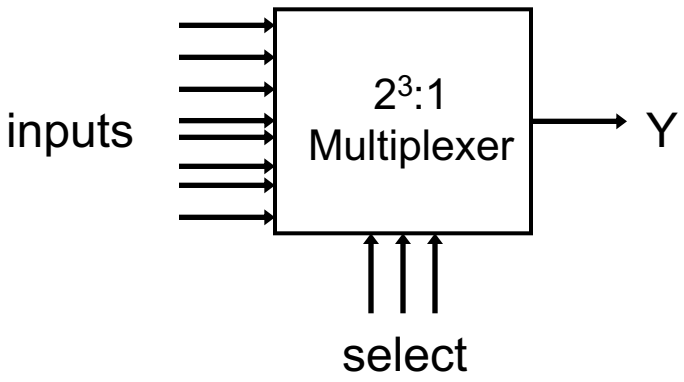
Sample 1 (cont'd)



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

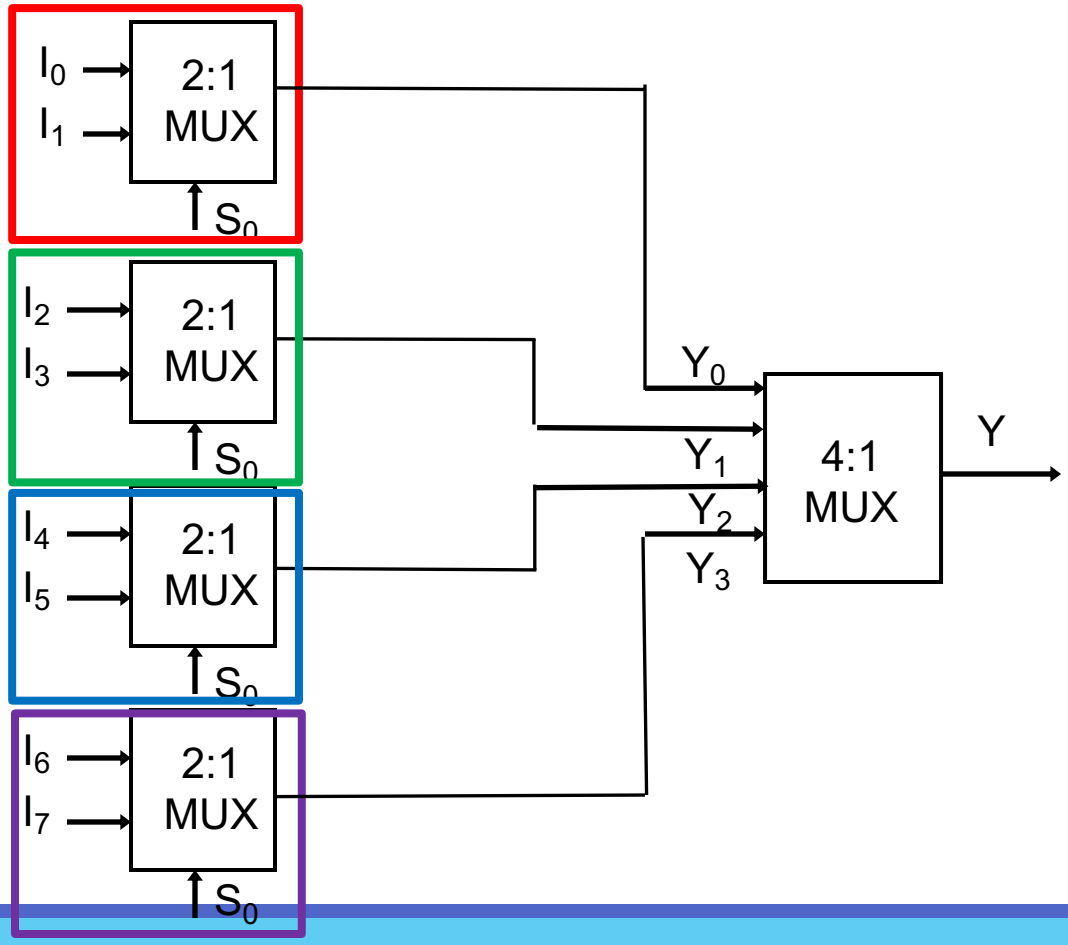
Sample 2

- Construct a 8-1 MUX
  - 4 2-1 MUX
  - 1 4-1 MUX



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

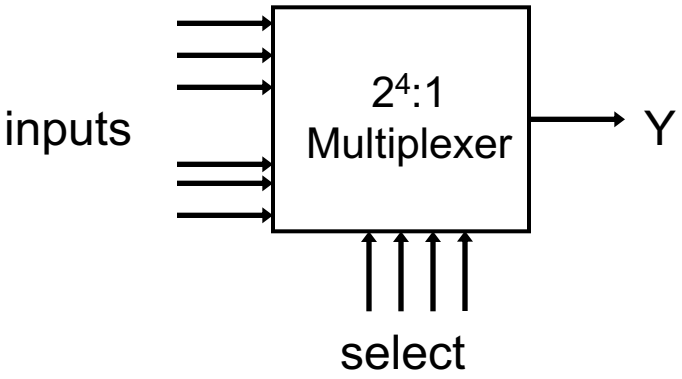
Sample 2 (cont'd)



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

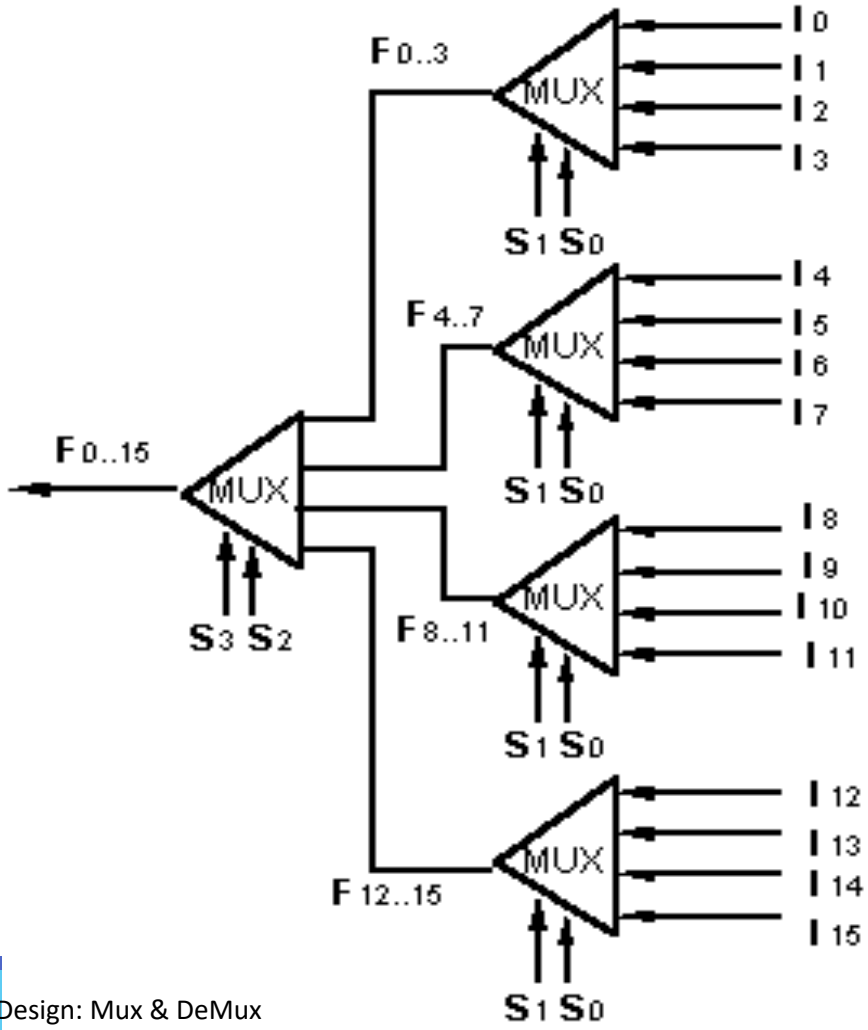
Sample 3

- Construct a 16-1 MUX
  - 4-1 MUX



Sample 3 (cont'd)

- Construct a 16-1 MUX
  - 4-1 MUX

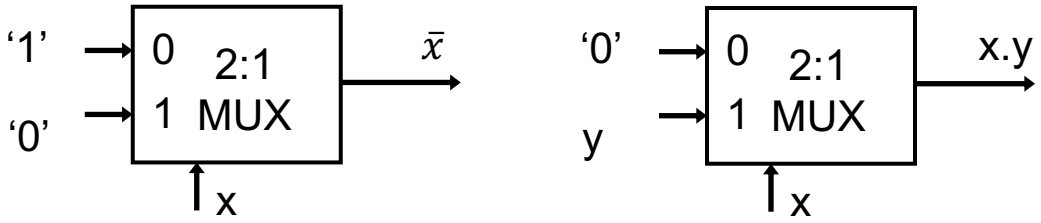


MUX & Universal Logic?

- Prove Mux is a UNIVERSAL logic

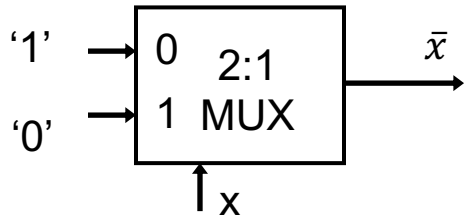
MUX Is a UNIVERSAL

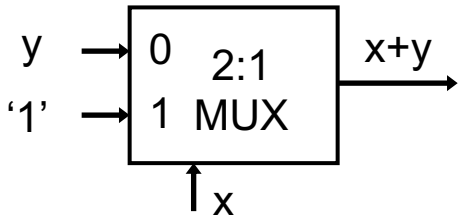
- Implement **and** and **not**



MUX Is a UNIVERSAL

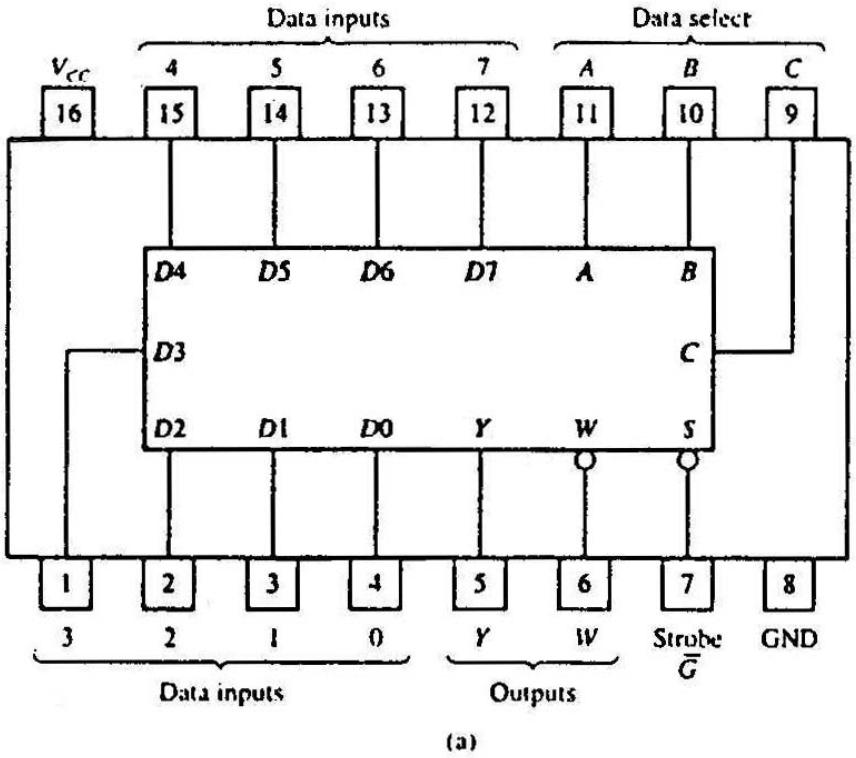
- Implement **or** and **not**

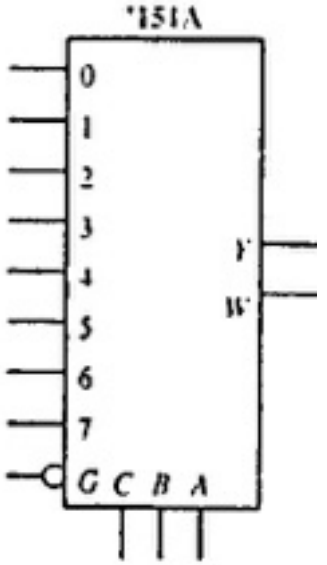




Multiplexer: Chip

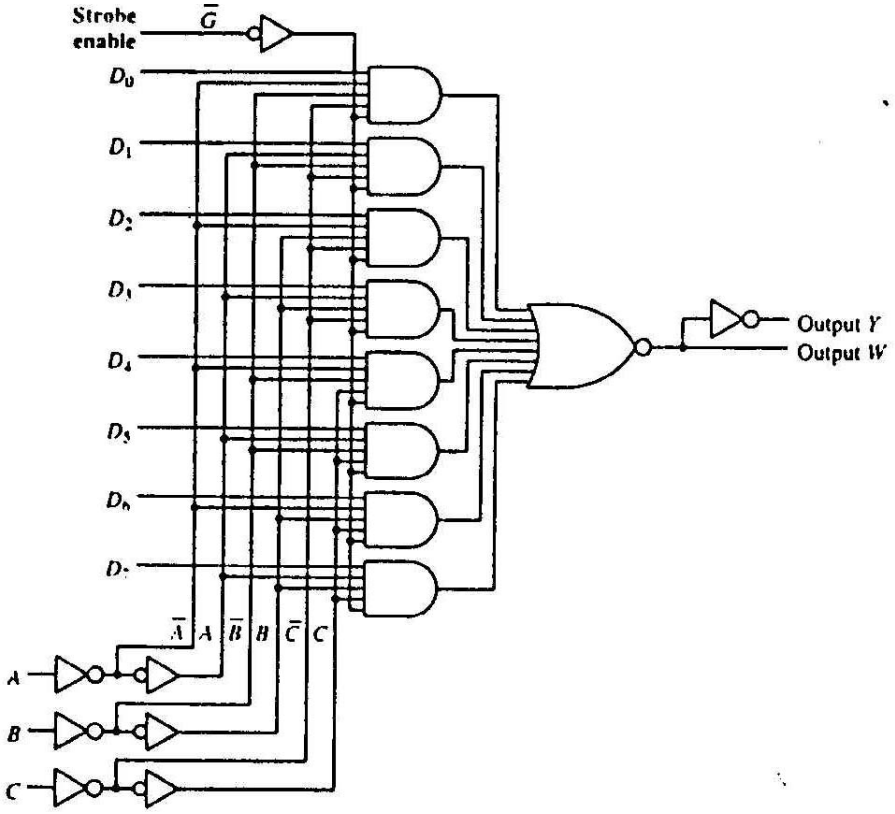
- 74151A
  - 8-1 MUX





Multiplexer: Chip (cont'd)

- 74151A
  - 8-1 MUX

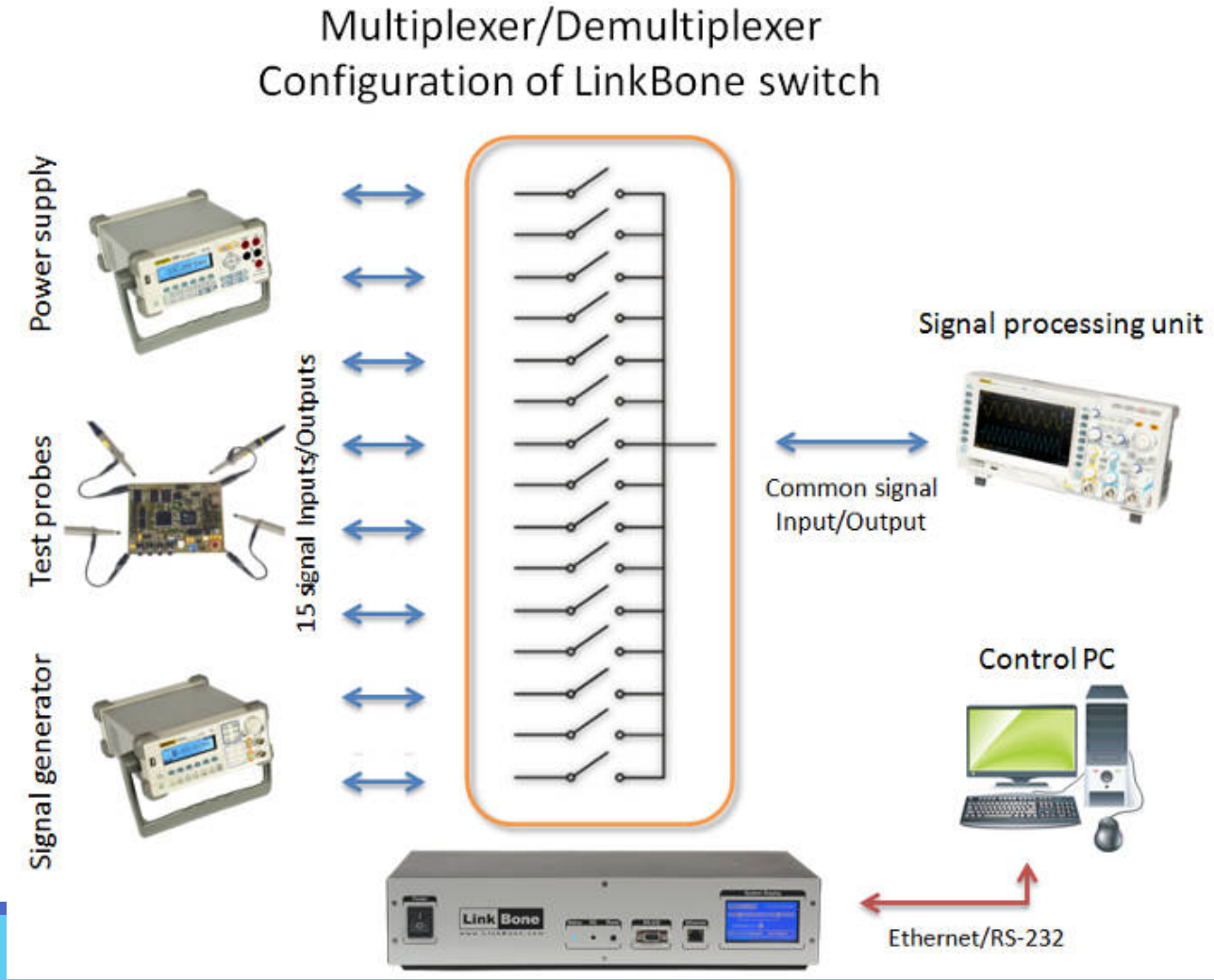


Inputs				Outputs	
Select			Strobe		
C	B	A	$\overline{G}$	Y	W
x	x	x	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

(b)

Application

MUX Application



MUX Application

- Implement Boolean functions
- Inputs are **Product Terms (PT)**
- Selection lines are **variables**
- Steps
  - Present **SOP** form
  - Connect **n** variables to the **n** selection lines
  - Connect input lines to **'0'**
  - For each **PT**; connect corresponding input line to **'1'**

$$F = \sum_{i=0}^{i=n} m_i I_i$$

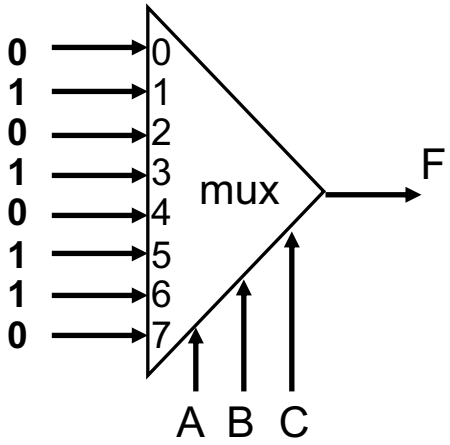
Sample 5

- Implement the function F using MUX
  - F(A,B,C) = A'B'C + A'BC + AB'C + ABC’

Sample 5: Realization

- Implement the function F using MUX
  - F(A,B,C) = A'B'C + A'BC + AB'C + ABC’

F(A,B,C) = A'B'C + A'BC + AB'C + ABC’ = Σ m(1,3,5,6)



Sample 6

- Implement the function F using 74151A MUX
  - $F(x_1, x_2, x_3) = \sum m(0,2,3,5)$

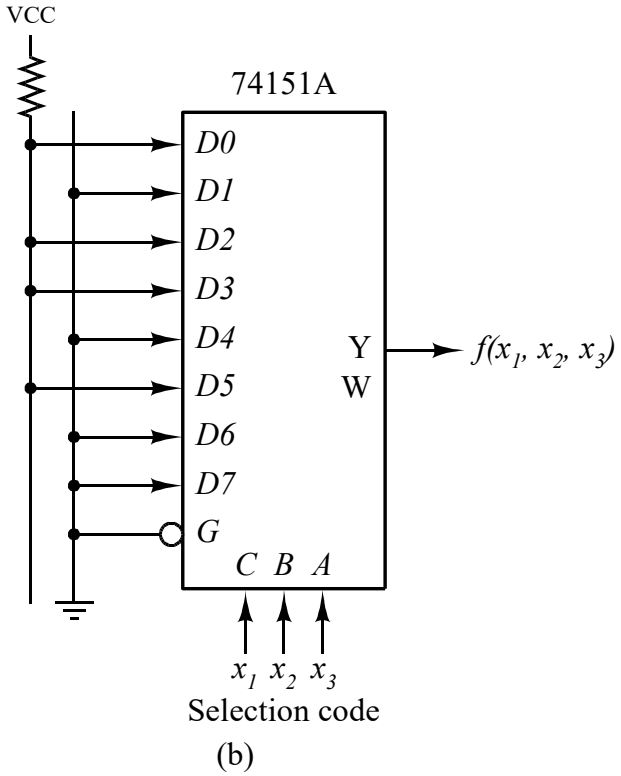
Inputs				Outputs	
Select			Strobe		
C	B	A	$\overline{G}$	Y	W
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

Sample 6: Realization

- Implement the function F using 74151A MUX
  - $F(x_1, x_2, x_3) = \sum m(0,2,3,5)$

i	C	B	A	Y
	x_1	x_2	x_3	f
0	0	0	0	$1\,D_0 = 1$
1	0	0	1	$0\,D_1 = 0$
2	0	1	0	$1\,D_2 = 1$
3	0	1	1	$1\,D_3 = 1$
4	1	0	0	$0\,D_4 = 0$
5	1	0	1	$1\,D_5 = 1$
6	1	1	0	$0\,D_6 = 0$
7	1	1	1	$0\,D_7 = 0$

(a)



(b)

Sample 7

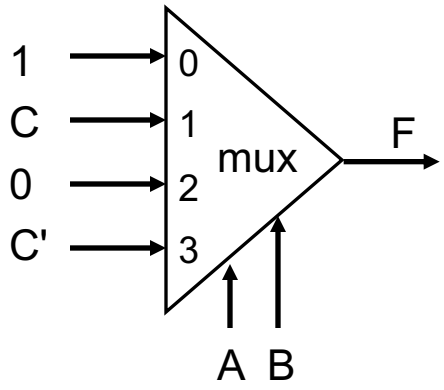
- Implement the function F using 4-1 MUX
  - F(A,B,C) = A'B'C + A'BC + AB'C + ABC’

Sample 7: Realization

- Implement the function F using 4-1 MUX
  - F(A,B,C) = Σ m(0,1,36)
  - F(A,B,C) = A'B'C’ + A'B’C + A’BC + ABC’
  - = A’B’ + A’B C + AB C’
  - => AB = 00 => F(A,B,C) = 1
  - => AB = 01 => F(A,B,C) = C
  - => AB = 10 => F(A,B,C) = 0
  - => AB = 11 => F(A,B,C) = C’

Sample 7: Realization (cont'd)

- Implement the function F using 4-1 MUX
  - F(A,B,C) = Σ m(0,1,36)
  - F(A,B,C) = A'B'C' + A'B'C + A'BC + ABC'
  - = A'B' + A'B C + AB C'
  - => AB = 00 => F(A,B,C) = 1
  - => AB = 01 => F(A,B,C) = C
  - => AB = 10 => F(A,B,C) = 0
  - => AB = 11 => F(A,B,C) = C'



Realization with Smaller MUX

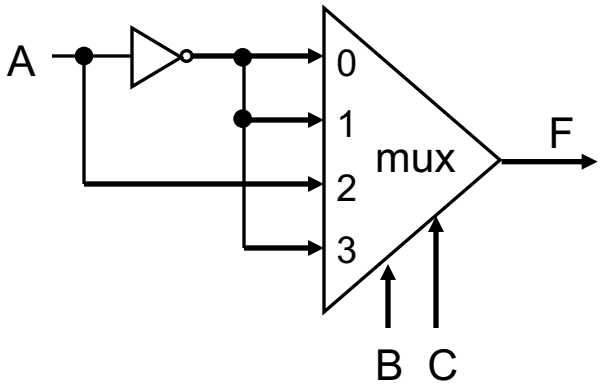
- Implement n variable function with $2^{n'}$:1 MUX; n' = n -1
- Steps
  - Present **SOP** form
  - Connect **one variable** to the **input** line of MUX
  - Connect **other variables** to the **n-1 selection lines**
  - Draw the **truth table** for function
    - Grouping inputs by selection line values
    - Determine inputs

Sample 8

- Implement the function F using 4-1 MUX
- Consider A as input line
  - $F(A,B,C) = \Sigma\, m(0,1,36)$

Sample 8: Realization

- $F(A,B,C) = \Sigma\, m(0,1,36)$
  - $F(A,B,C) = A'B'C' + A'B'C + A'BC + ABC'$
  - $= B'C' (A') + B'C (A') + BC' (A) + BC(A')$
  - => BC = 00 => $F(A,B,C) = A'$
  - => BC = 01 => $F(A,B,C) = A'$
  - => BC = 10 => $F(A,B,C) = A$
  - => BC = 11 => $F(A,B,C) = A'$



Sample 9

- Implement the function F using 74151A MUX (8-1 MUX)
- $f(x_1,x_2,x_3,x_4) = \sum m(0,1,2,3,4,9,13,14,15)$

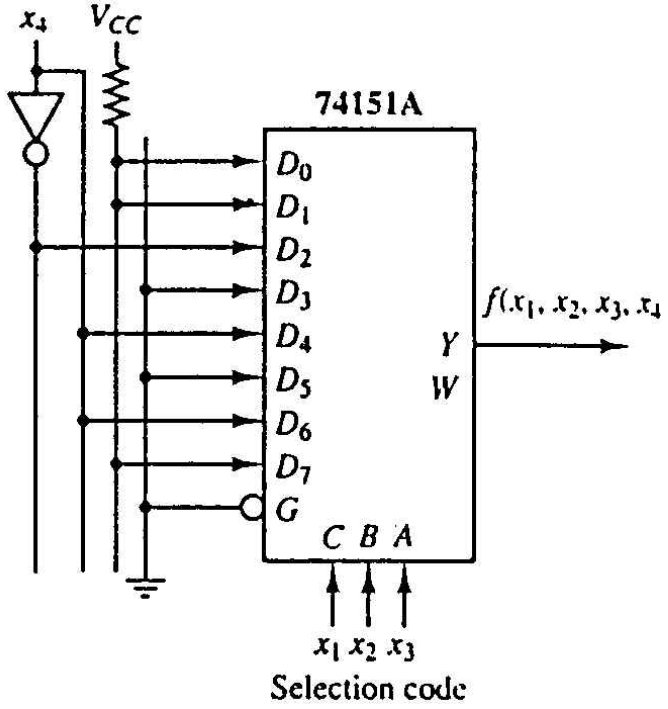
Inputs				Outputs	
Select			Strobe		
C	B	A	$\bar{G}$	Y	W
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

(b)

Sample 9: Realization

- Implement the function F using 74151A MUX (8-1 MUX)
- $f(x_1,x_2,x_3,x_4) = \sum m(0,1,2,3,4,9,13,14,15)$

	C	B	A				Y
i	X_1	X_2	X_3	X_4	f	f	
0	0	0	0	0	1		
	0	0	0	1	1	1	$D_0 = 1$
1	0	0	1	0	1		
	0	0	1	1	1	1	$D_1 = 1$
2	0	1	0	0	1		
	0	1	0	1	0	$\bar{X}_4$	$D_2 = \bar{X}_4$
3	0	1	1	0	0		
	0	1	1	1	0	0	$D_3 = 0$
4	1	0	0	0	0		
	1	0	0	1	1	X_4	$D_4 = X_4$
5	1	0	1	0	0		
	1	0	1	1	0	0	$D_5 = 0$
6	1	1	0	0	0		
	1	1	0	1	1	X_4	$D_6 = X_4$
7	1	1	1	0	1		
	1	1	1	1	1	1	$D_7 = 1$

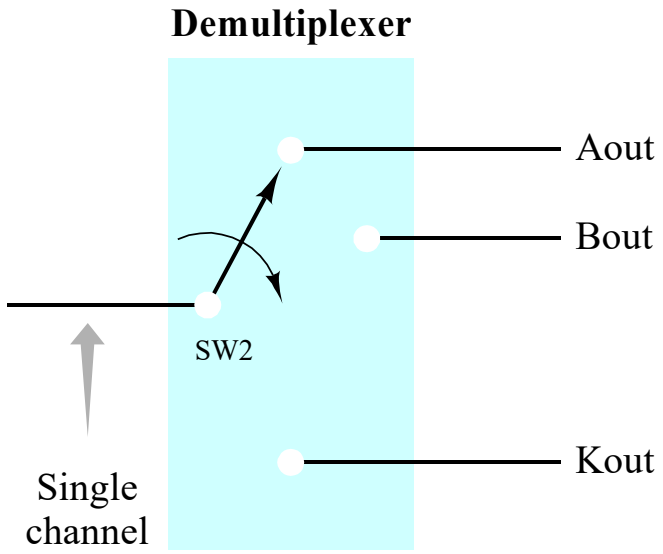


Demultiplexer

Demultiplexing

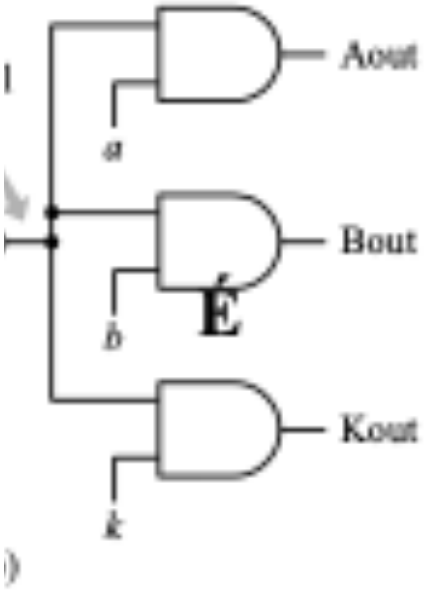
- Suppose a logic has **one input data** and **some candidates** on its **output**
- Logic should **select** one of candidates

- **Selection logic**
  - Select output line
  - Send data to the selected output line

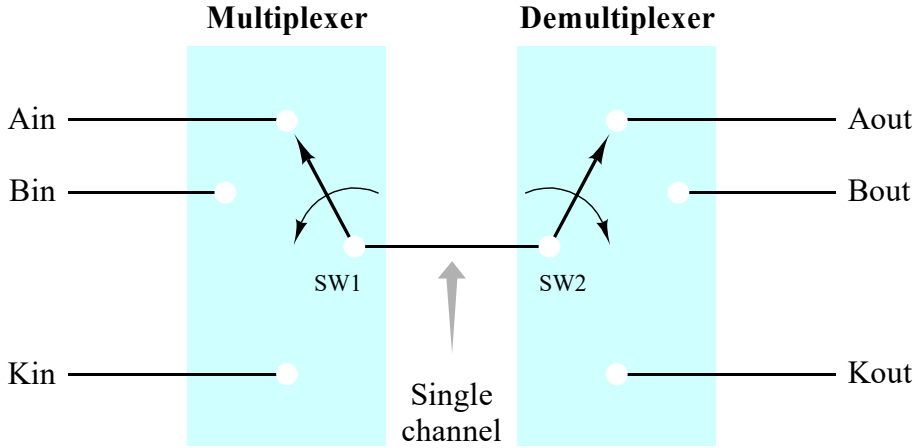


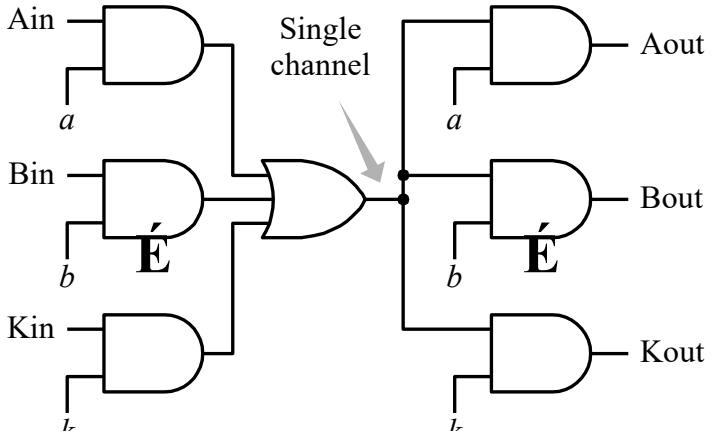
Demultiplexer

- Input
  - n selection lines
  - 1 input line
- Output
  - 2^n output lines
- Function
  - Uses ***n* selection** lines
  - Selects **one** of 2^n outputs



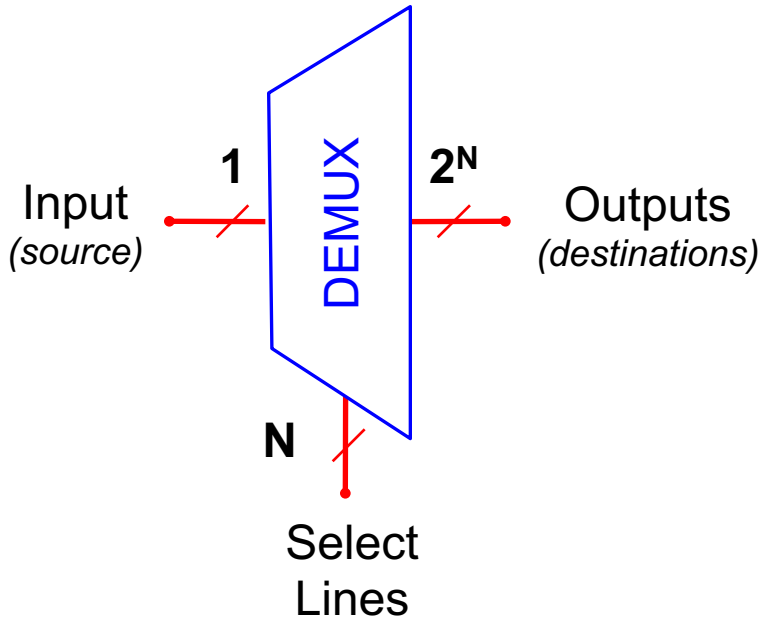
Multiplexer VS. Demultiplexer





Demultiplexer: 1-4

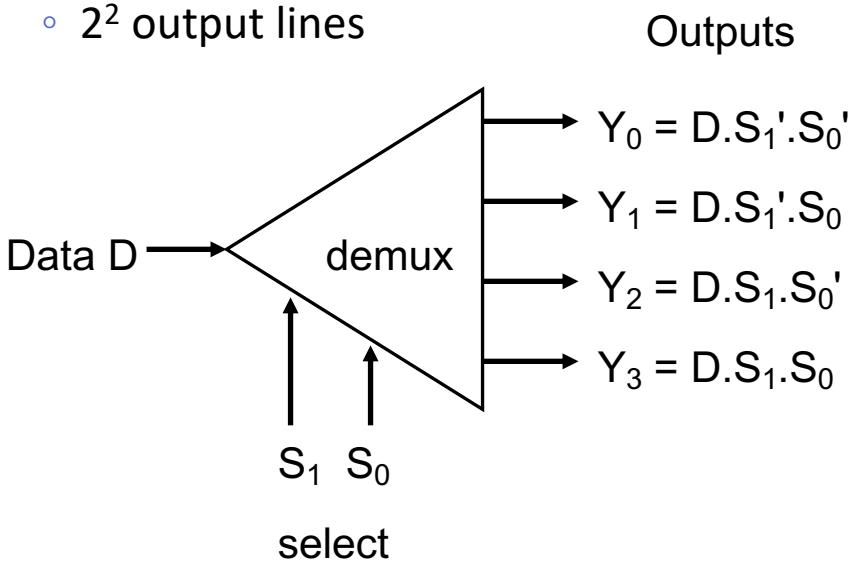
- Input
  - 2 selection lines
  - 1 input line
- Output
  - 2^2 output lines



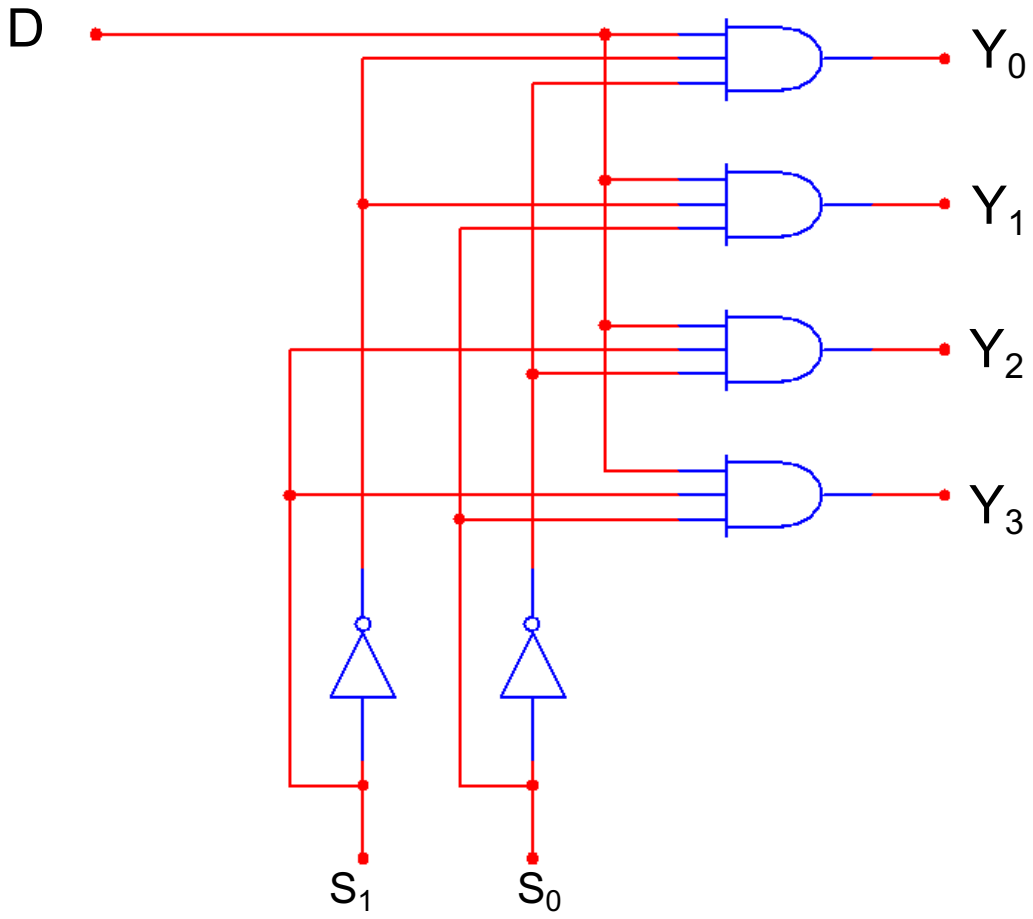
Demultiplexer: 1-4 (cont'd)

- Input
  - 2 selection lines
  - 1 input line
- Output
  - 2^2 output lines

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



Demultiplexer: 1-4 (cont'd)

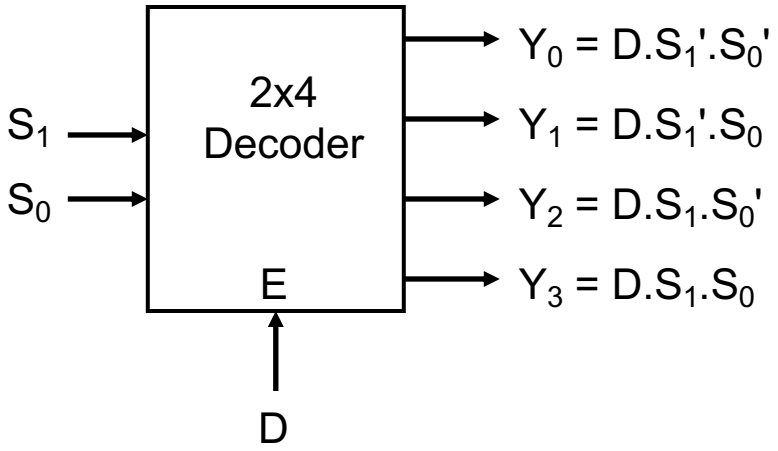


S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

Demultiplexer Vs. Decoder

Demultiplexer = An Enabled Decoder

- A decoder with **enable** signal
- Inputs
  - Selection lines of demultiplexer
- Enable Input
  - Input line of demultiplexer
- Output
  - 2^2 output lines of demultiplexer



Thank You