Lecture 10 – Combinational Logic – Part II

Rami Abielmona
University of Ottawa
February 10, 2015
ITI 1100 B
Digital Systems I

Presentation Outline

- Implementation Using Decoders
- Implementation Using NAND Gates
- Active-High/Active-Low Decoders
- Encoders
- Priority Encoders
- Multiplexers/Demultiplexers
- Key terms

Implementing Boolean Functions Using Decoders

- Any combinational circuit can be constructed using decoders and OR gates! Why?

→Here is an example:

Implement a full adder circuit with a decoder and two OR gates.

Recall full adder equations, and let X, Y, and Z be the inputs:

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

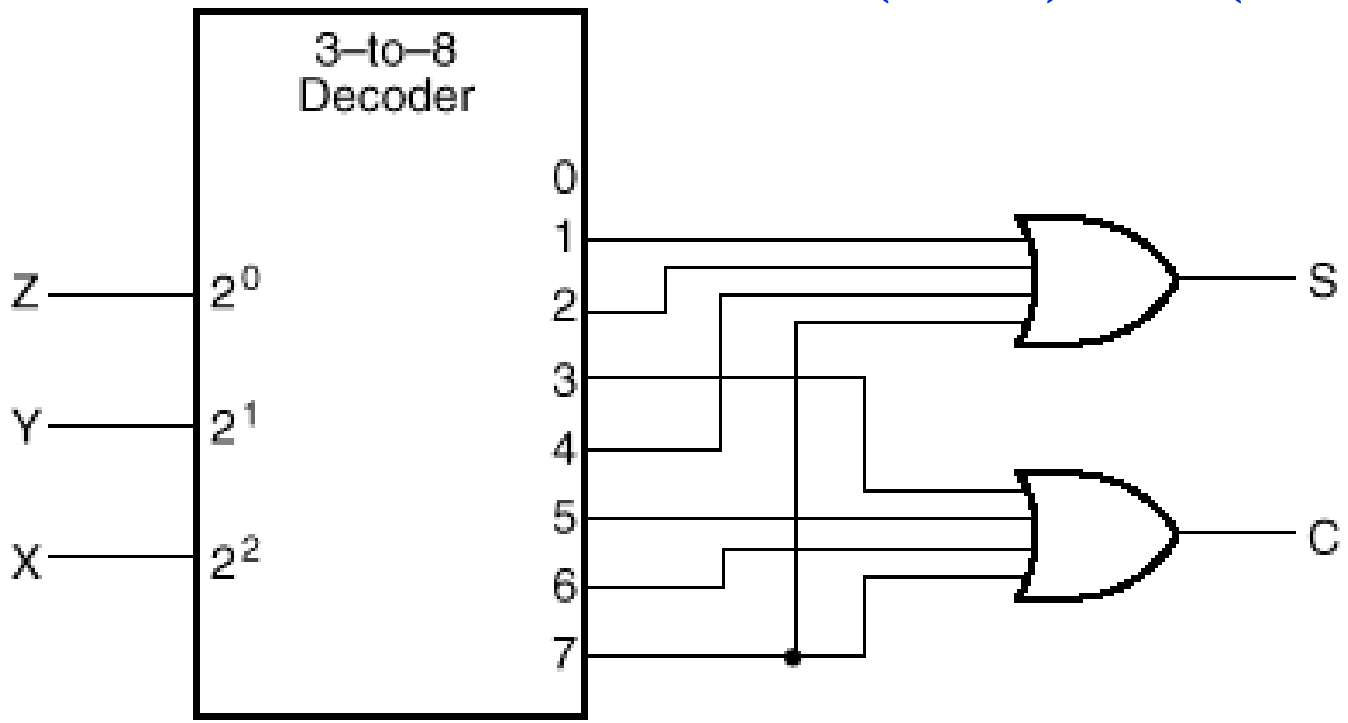
$C(X,Y,Z) = \Sigma m(3, 5, 6, 7)$.

•Since there are 3 inputs and a total of 8 minterms, we need a 3-to-8 decoder.

Implementing Boolean Functions Using Decoders

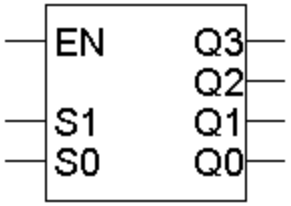
$S(X,Y,Z) = \Sigma m(1,2,4,7)$

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



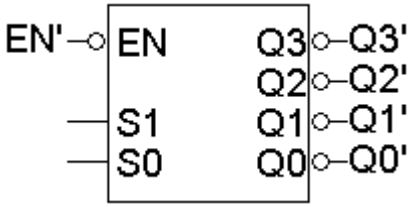
Implementing a decoder with NAND gates

- The decoders studied so far are active-high decoders (i.e. implemented with AND gates)



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

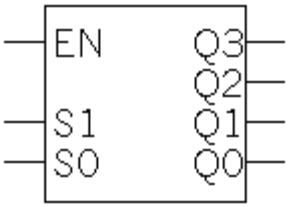
- Active-low decoders are implemented using NAND gates (i.e. with an inverted EN input and inverted outputs).



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

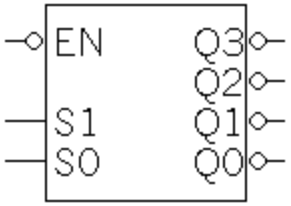
Active-High and Active-Low decoders

- Active-high decoders generate *minterms*, as we have already seen.



Q3 = S1 S0
Q2 = S1 S0'
Q1 = S1' S0
Q0 = S1' S0'

- Active-low decoders generate *Maxterms*.



Q3' = (S1 S0)' = S1' + S0'
Q2' = (S1 S0')' = S1' + S0
Q1' = (S1' S0)' = S1 + S0'
Q0' = (S1' S0')' = S1 + S0

Building a 3-to-8 decoder with two 2-to-4 decoders

- Another way to design a 3-to-8 decoder is to use two 2-to-4 decoders.
- from the truth table of 3-8 decoder we can notice some patterns:
  - When S2 = 0, outputs Q0-Q3 are generated as in a 2-to-4 decoder.
  - When S2 = 1, outputs Q4-Q7 are generated as in a 2-to-4 decoder.

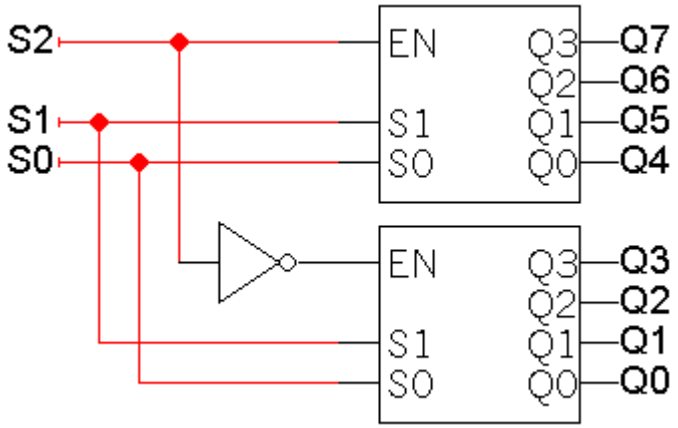
  → S2 can be used as an Enable input that activates/deactivates the 2-to-4 decoders.

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

Q0 = S2' S1' S0' = m_0
Q1 = S2' S1' S0 = m_1
Q2 = S2' S1 S0' = m_2
Q3 = S2' S1 S0 = m_3
Q4 = S2 S1' S0' = m_4
Q5 = S2 S1' S0 = m_5
Q6 = S2 S1 S0' = m_6
Q7 = S2 S1 S0 = m_7

Building a 3-to-8 decoder with two 2-to-4 decoders

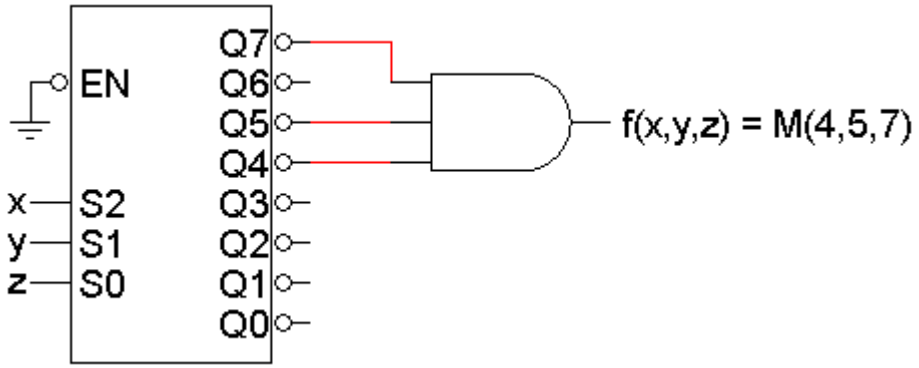
- We can use enable inputs to string decoders together. 3-to-8 decoder constructed from two 2-to-4 decoders:



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

Active-low decoder example

- We can use active-low decoders to implement arbitrary functions as a product of maxterms.
- For example, here is an implementation of the function f(x,y,z) = ΠM(4,5,7), using an active-low decoder.



- The "ground" symbol connected to EN represents logical 0, so this decoder is always enabled.
- We need an AND gate for a product of sums.

Decoder Expansions

- Larger decoders can be constructed using a number of smaller ones.

→ HIERARCHICAL design

- *Example:*
  A 6-to-64 decoder can be designed using four 4-to-16 and one 2-to-4 decoders. How? (tip: Use the 2-to-4 decoder to generate the enable signals to the four 4-to-16 decoders).

4-input tree decoder

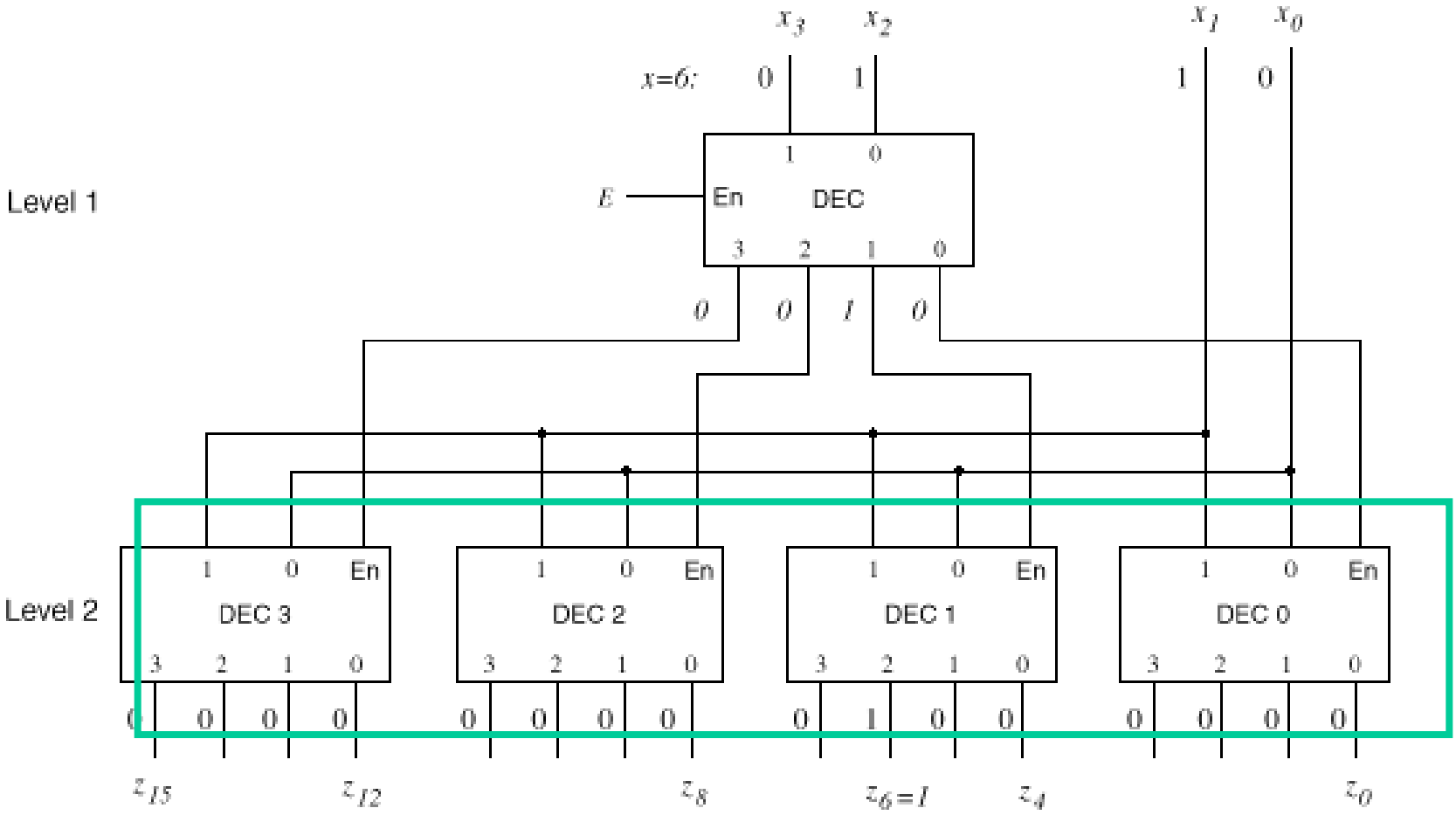


Figure 9.8: 4- nput tree decoder

Encoder

- An encoder has a number of input lines, only one of which is activated at a given time.
- The opposite of the decoding process.

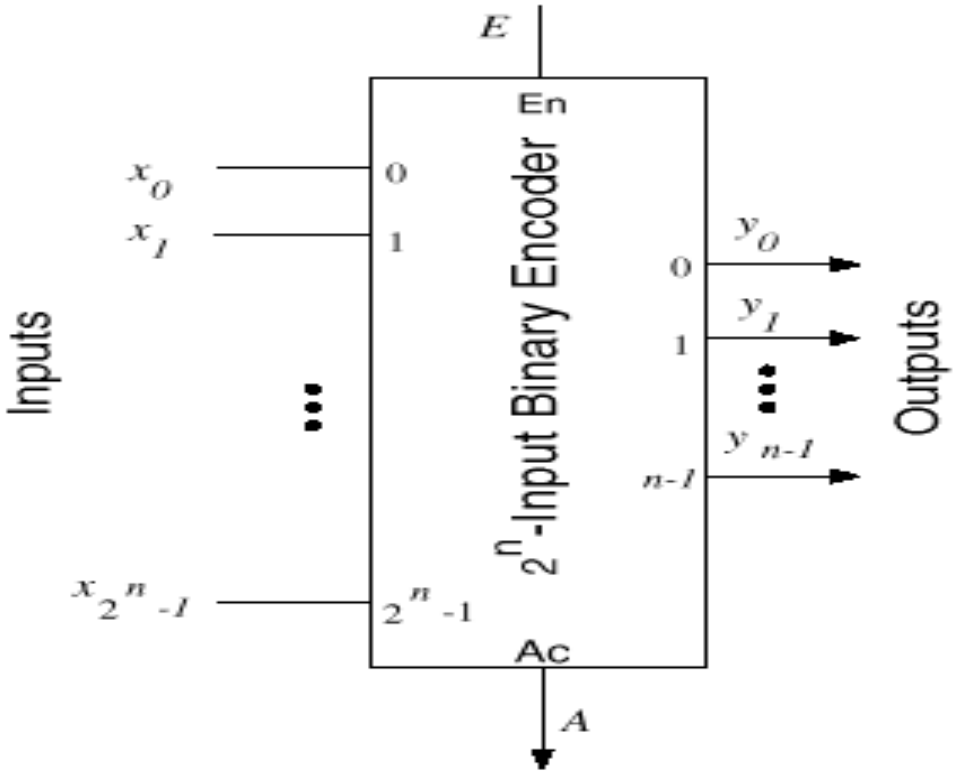


Figure 9.12: 2−-input binary encoder.

Encoder

Example: **8-to-3 Encoder**

x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	y_2	y_1	y_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

Encoder

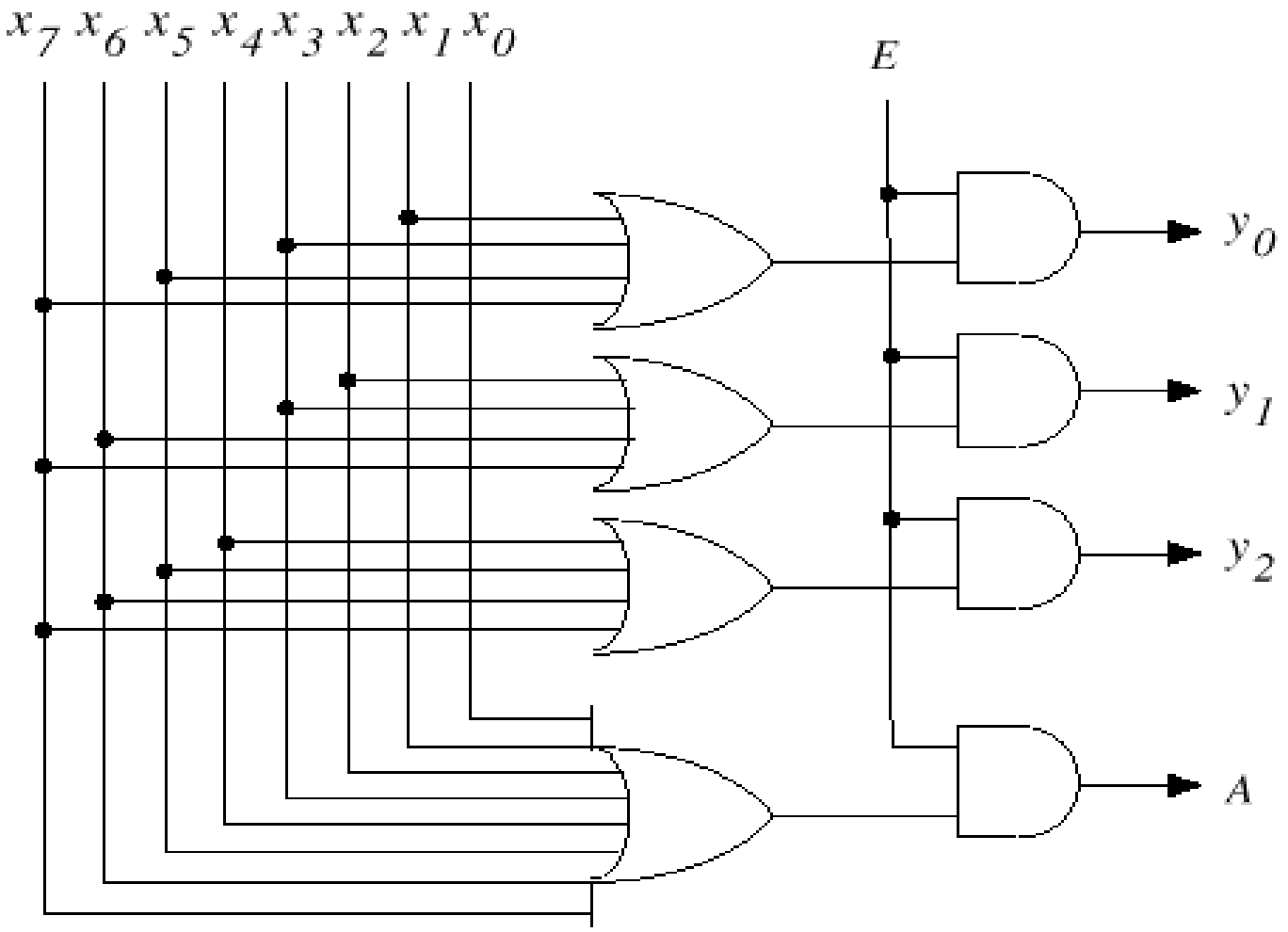


Figure 9.13: mplementation of an 8-input binary encoder.

Encoder

- In the previous truth table each line selected 0 through 7 generates it's own binary code such as a 1 is a 001, 5 is a 101 and so on.
- Boolean functions for Outputs

$$y2 = x4 + x5 + x6 + x7$$

$$y1 = x2 + x3 + x6 + x7$$

$$y0 = x1 + x3 + x5 + x7$$

Simple Encoder Design Issues

- There are two ambiguities associated with the design of a simple encoder:
  - 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 x3 and x6 are 1 simultaneously, the output of the encoder will be 111. (100 and 011)
  - An output with all 0's can be generated when all the inputs are 0's,or when X0 is equal to 1.

Encoder

Example: **8-to-3 Encoder**

x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	y_2	y_1	y_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

Priority Encoders

- Solves the ambiguities multiple assigned inputs are allowed; one has priority over all others.
- Separate indication of not assigned inputs (all inputs are 0s).

Example: 4-to-2 Priority Encoder Truth Table

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

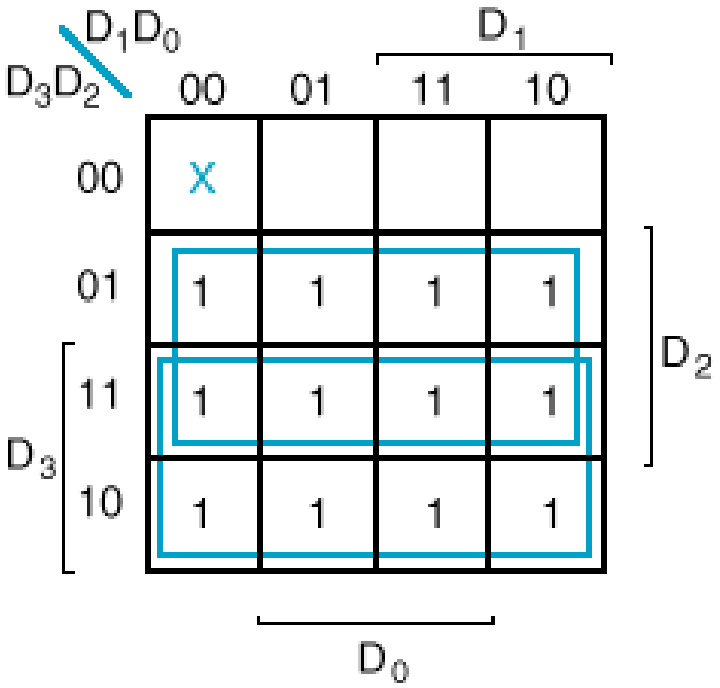
4-to-2 Priority Encoder (cont.)

- The operation of the priority encoder is such that:

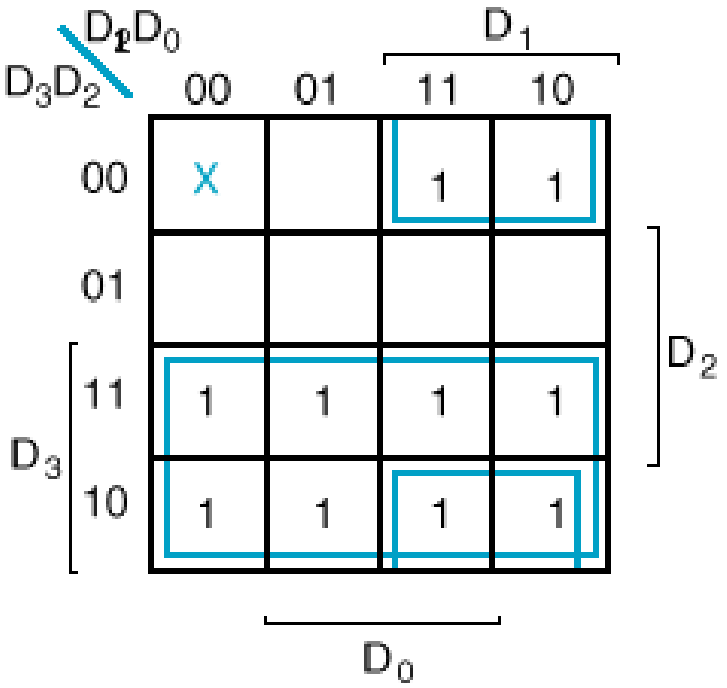
→If two or more inputs are equal to 1 at the same time, the input in the highest-numbered position will take precedence.

→A ***valid output indicator***, designated by V, is set to 1 only when one or more inputs are equal to 1. $V = D_3 + D_2 + D_1 + D_0$ by inspection.

4-to-2 Priority Encoder (cont.)

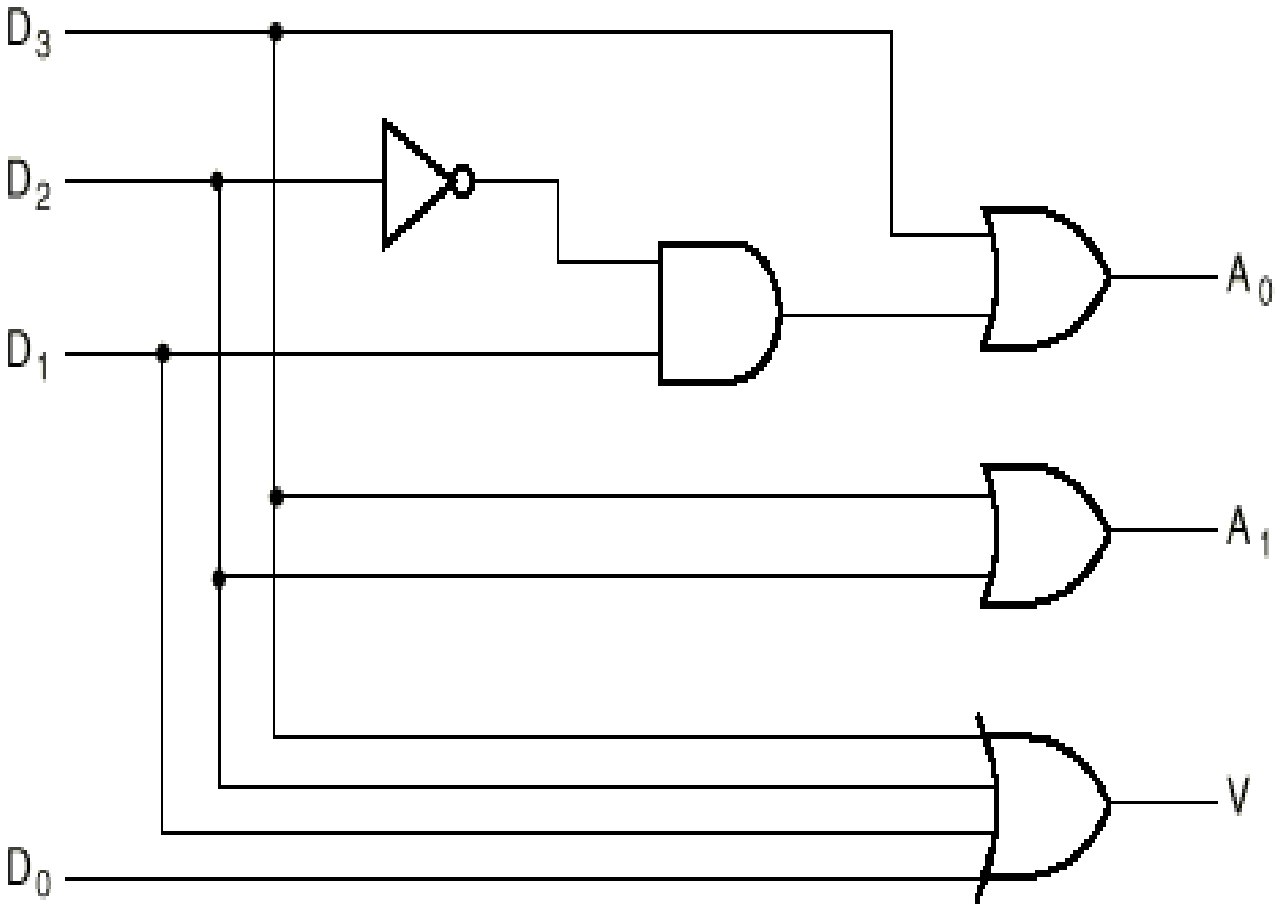


$A_1 = D_2 + D_3$



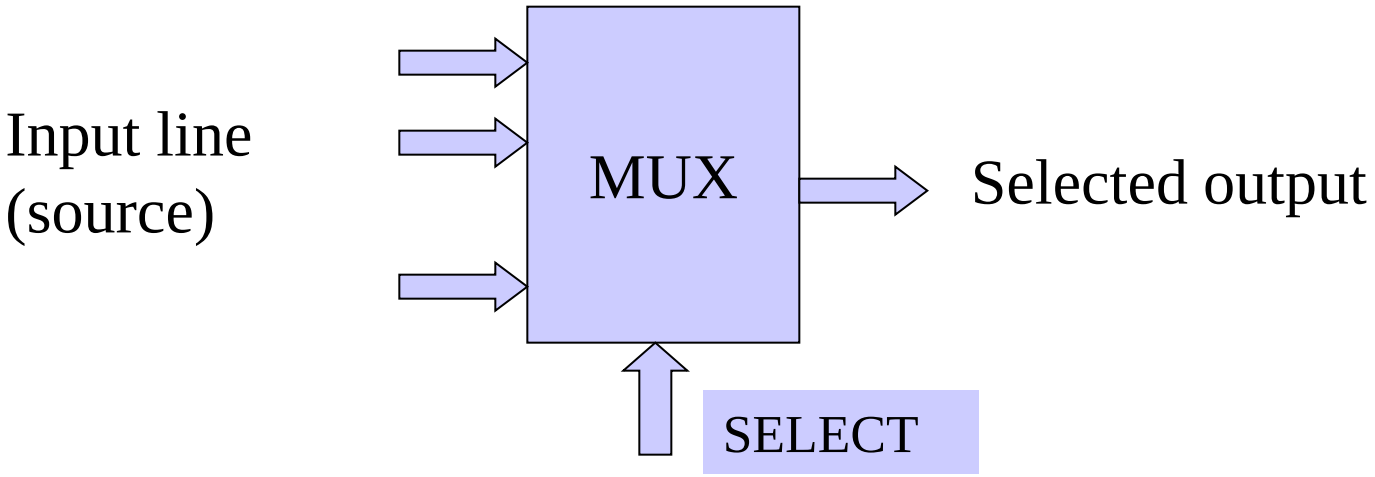
$A_0 = D_3 + D_1\overline{D}_2$

4-to-2 Priority Encoder (cont.)



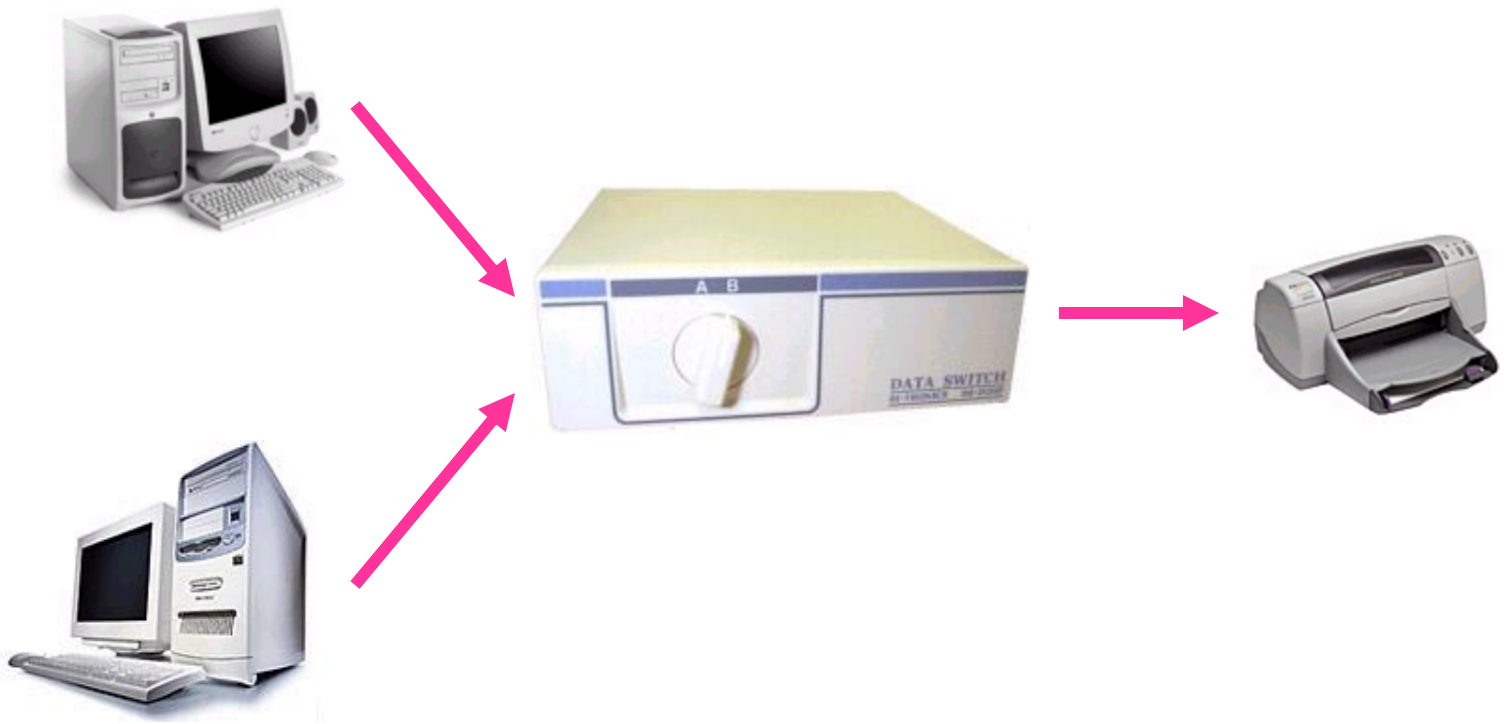
Multiplexer (MUX): Data Selectors

- A multiplexer selects one of several input signals and passes it on to the output.
- Routing of selected data input to the output is controlled by SELECT inputs.



Multiplexer

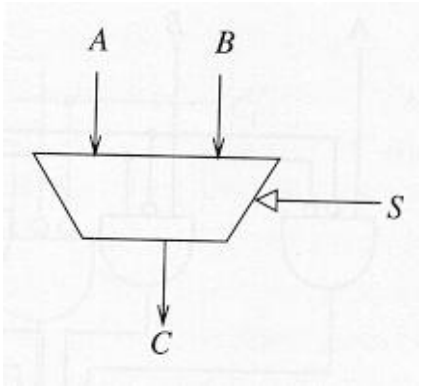
- Multiplexers, or muxes, are used to choose between resources.
- A real-life example: in the old days before networking, several computers could share one printer through the use of a switch.



Multiplexer (MUX): Data Selectors [2]

- A combinational circuit with 2^n *data inputs*, 1 data output and a number of bit *control input* that select one of the data inputs

C takes the value of A or B depending on the value of S



2-to-1 Multiplexer

S	A	B	C
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

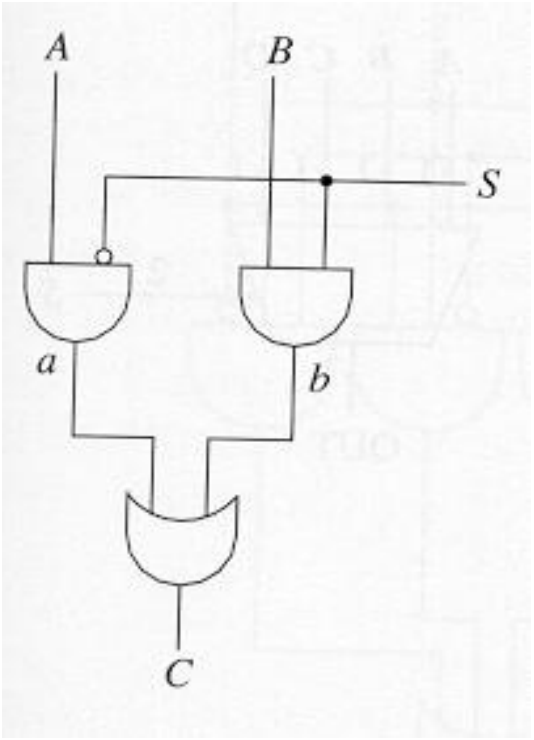
2-to-1 Multiplexer

When S= 0 → C= A, when S= 1 → C = B

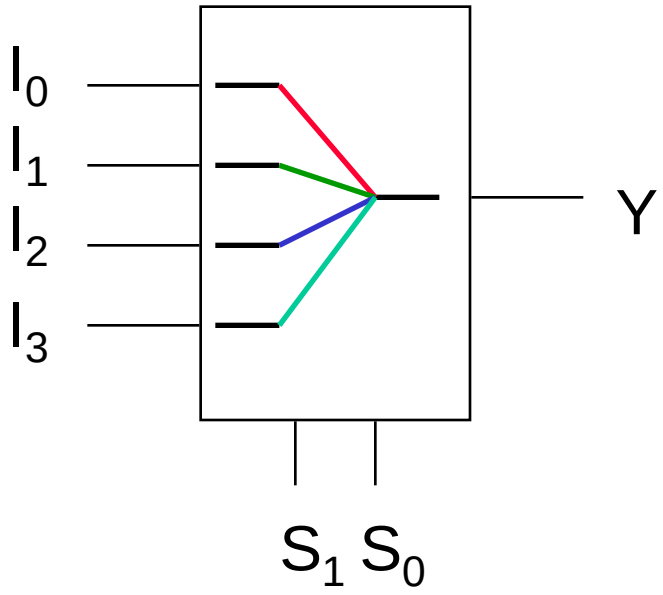
$$C = \bar{S}.A.\bar{B} + \bar{S}.A.B + S.\bar{A}.B + S.A.B$$
$$= \bar{S}.A + S.B$$

Inputs			Output
S	**A**	**B**	**C**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Two level implementation



4 – 1 Multiplexer (MUX)



if $(S_1S_0)_2 = 0$ Then $Y = I_0$

$$Q = \overline{S_0}.\overline{S_1}.I_O$$

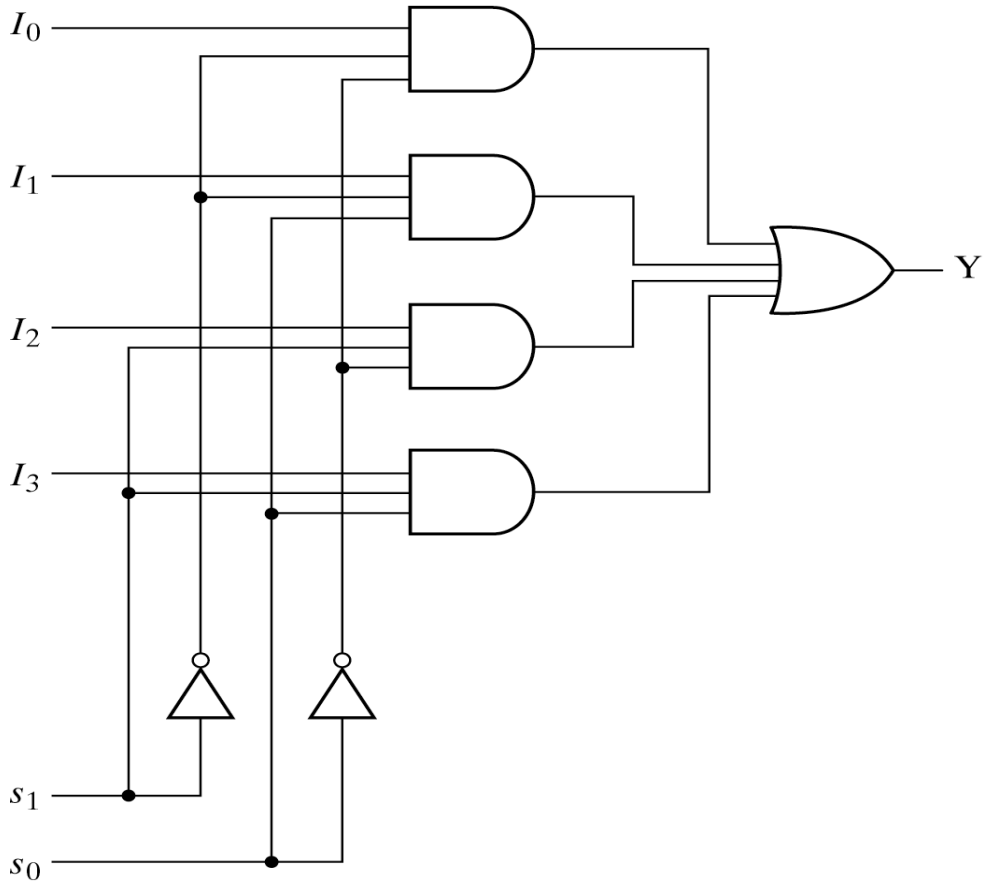
if $(S_1S_0)_2 = 1$ Then $Y = I_1$

$$Q = S_0.\overline{S_1}.I_1$$

and so on, thus we have

$$Y = \overline{S_1}.\overline{S_0}.I_0 + \overline{S_1}.S_0.I_1 + S_1.\overline{S_0}.I_2 + S_1.S_0.I_3$$

4-1 MUX- Two level Implementation



(a) Logic diagram

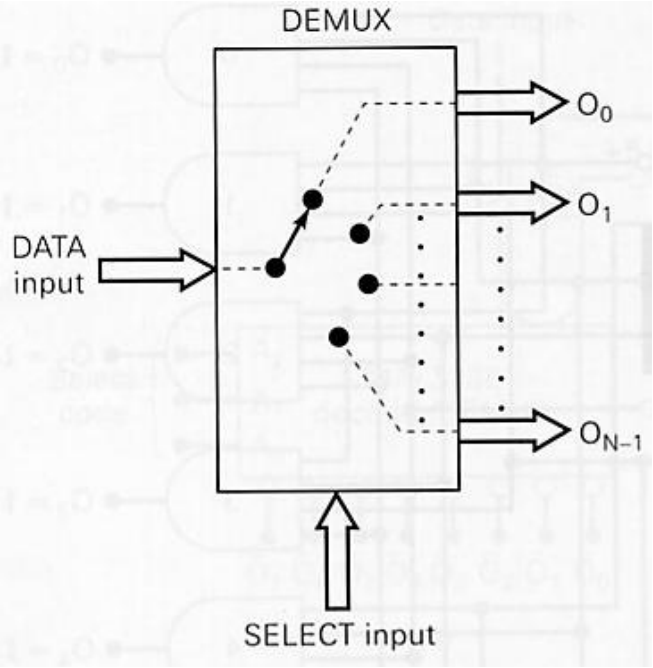
s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

Fig. 4-28 4-to-1 Line Multiplexer

Demultiplexer (DEMUX): Data Distributor

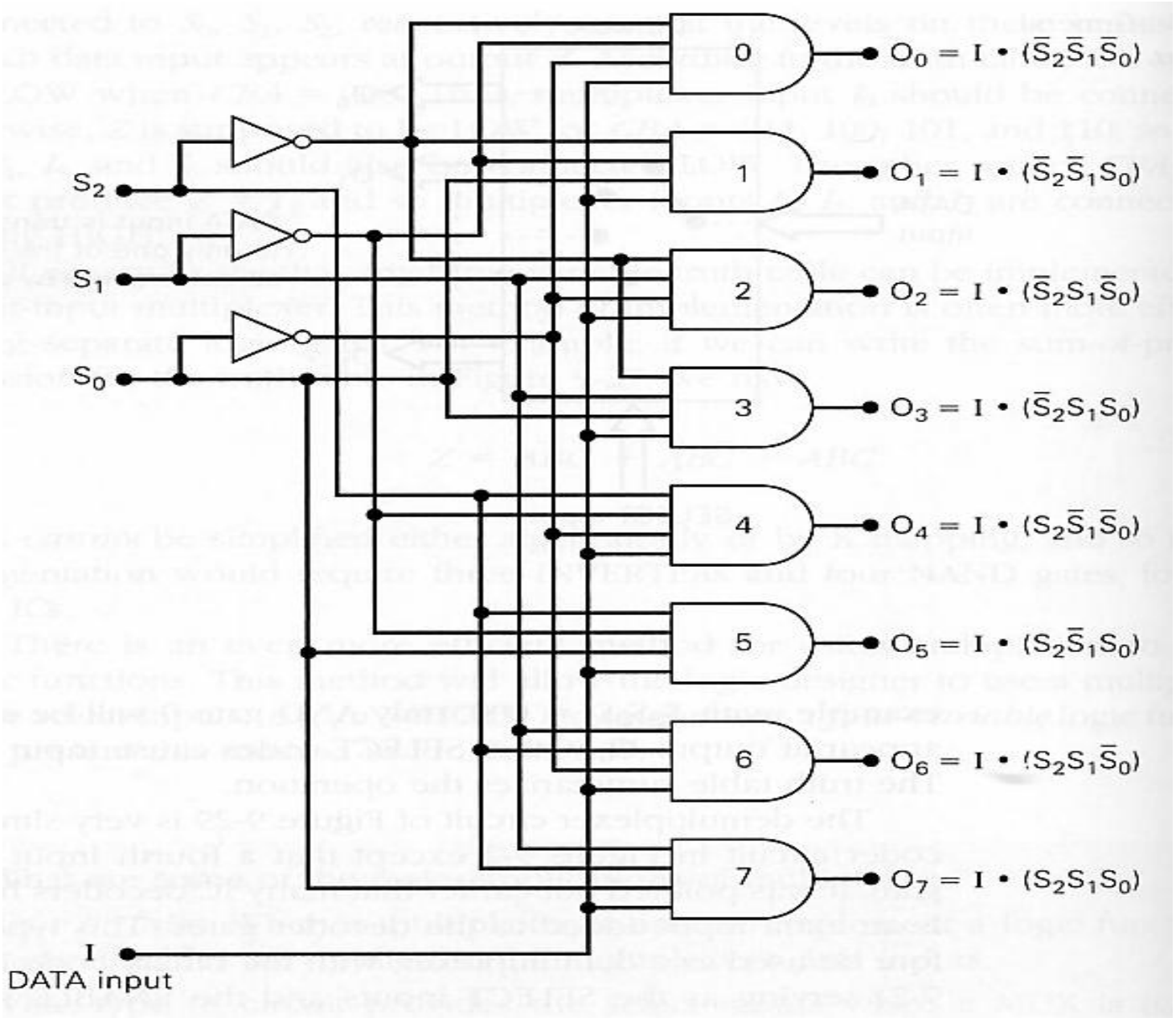
- Demultiplexer (DEMUX) takes a single input and distributes it over several outputs.



1-line-to-8-line Demultiplexer

SELECT code			OUTPUTS							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

1-line-to-8-line Demultiplexer



Key Terms and Review Points

- 1-to-8 Demultiplexer
- 8-to-3 Encoder
- 4-to-2 Priority Encoder
- Active-High Decoders
- Active-Low Decoders
- Decoder Expansions
- Demultiplexer
- Encoders
- Multiplexer
- Priority Encoders

- References: Dr. Karmouch ITI 1100 Slides