COM 204 – Digital Logic Systems

Assist. Prof. Özge ÖZTİMUR KARADAĞ

- Assist. Prof. Özge ÖZTİMUR KARADAĞ
  - Email: ozge.karadag@alanya.edu.tr

- Asistant: Gürkan Çelik
  - Email: gurkan.celik@alanya.edu.tr

Digital Logic Systems– COM 204

- Prerequisite:Digital Logic Design

- Book: Morris Mano, Digital Design, 4th edition

- **Evaluation:**
  - Midterm Quantity:1 Percentage: 40%
  - Final : Quantity:1 Percentage : 60%

Weekly Plan

Week	
21 February	Introduction, Material Related to Combinational Circuit
28 February	Synchronous Sequential Logic: Analysis of Clocked Sequential Circuits
7 March	Synchronous Sequential Logic: State Reduction and Assignment, Flip-Flop Excitation Tables
14 March	Design of Counters, Problems
21 March	Registers, Shift Registers
28 March	Ripple Counters, Synchronous Counters
4 April	Timing, Problems
10-16 April	Midterm (tentative)
18 April	Random Access Memory (RAM), Memory Decoding, Error Detection and Correction
25 April	ADM (Algorithmic State Machine), Realization of the Control Circuit
2 May	ADM
9 May	ADM: Design with Multiplexers
16 May	ADM: PLA (Programmable Logic Array)
23 May	Asynchronous Sequential Logic: Analysis
30 May	Asynchronous Sequential Logic: Design Procedure

The plan is tentative. Possible changes will be announced during lectures.

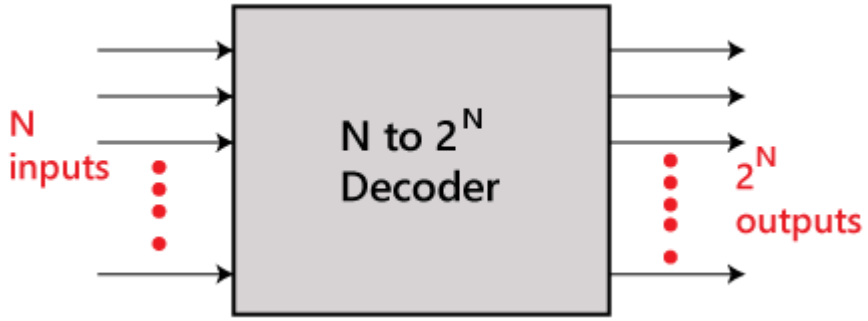
- Lab will start in April. Follow the announcements.
- Questions??

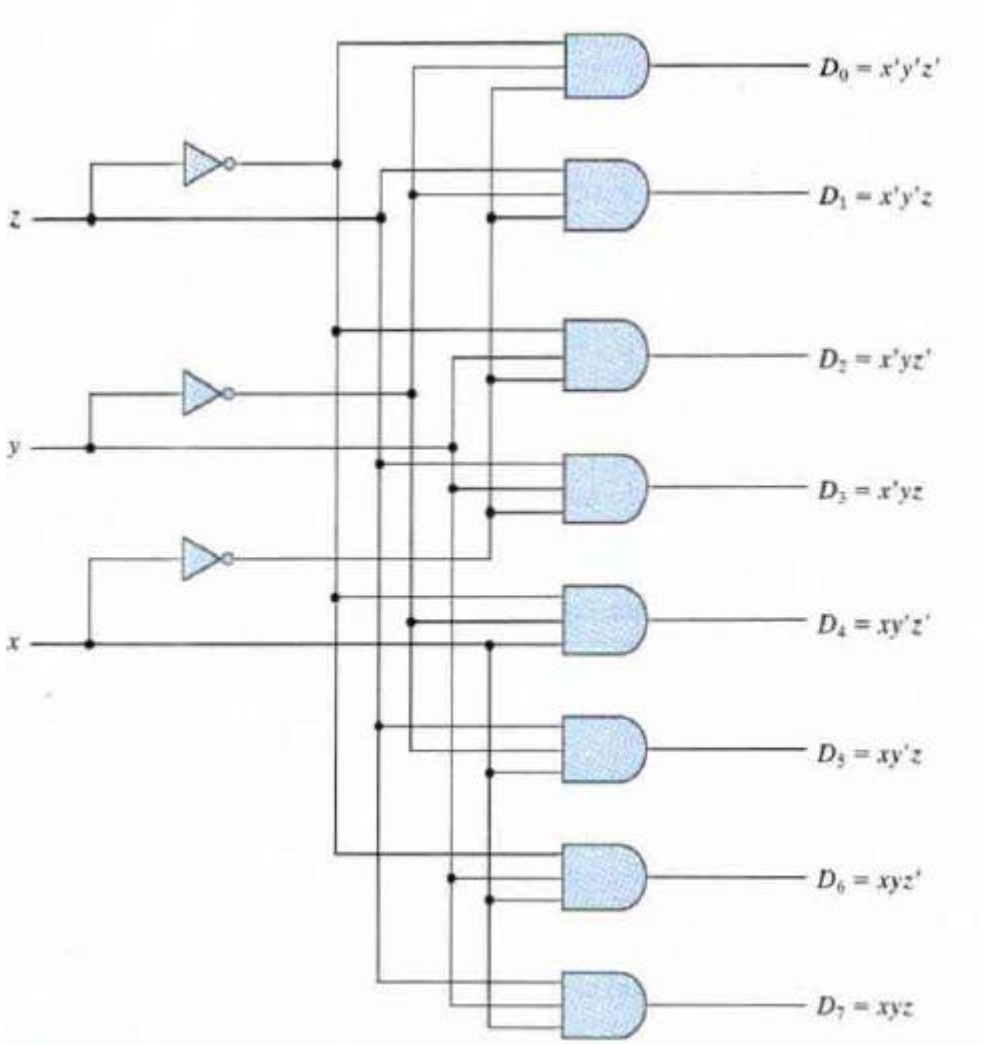
Digital Logic Design

- Binary System
- Boolean Algebra and Logic Gates
- Combinational Circuits
  - Binary adder-subtractor
  - Decoder
  - Encoder
  - Multiplexer

DECODERS

- A decoder is a combinational circuit that converts Binary information from n input lines to a maximum of 2^n unique output lines.
- n-to-m line decoders, where $m \leq 2^n$
- Ex: three-to-eight line decoder





DECODERS

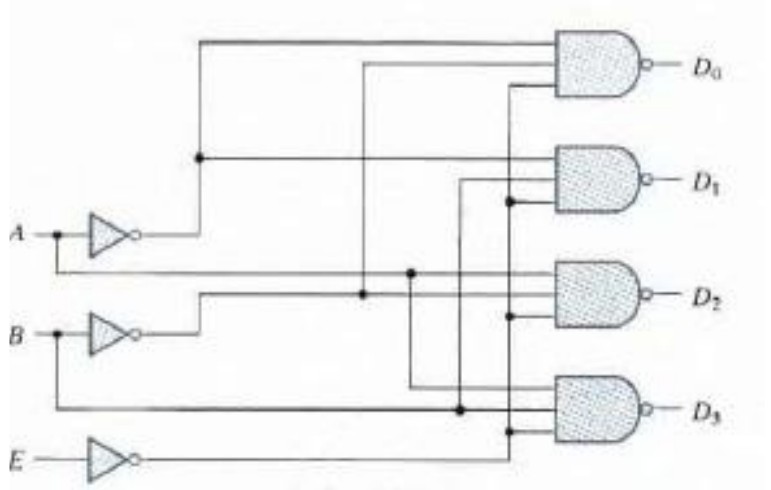
- Three-to-eight line decoder (Binary-to-octal conversion)

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_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

DECODERS

- Ex: two-to-four line decoder with **enable** input
  - Circuit is disabled when E=1

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
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



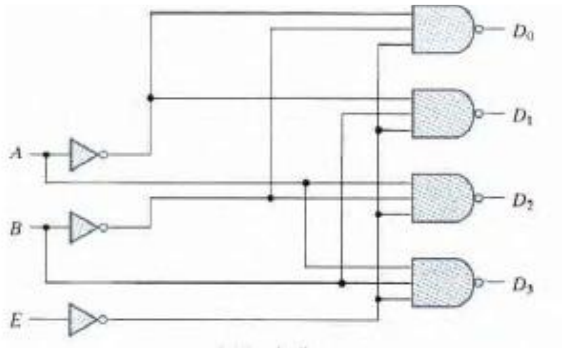
DECODERS

- A decoder may operate with complemented or uncomplemented outputs.
- The enable input may be activated with a 0 or a 1 signal.
- Some decoders have two or more enable inputs that must satisfy a given logic condition in order to enable the circuit.

DECODERS

- Demultiplexer: A circuit that receives information from a single line and directs it to one of 2^n possible output lines. The selection of the specific output is controlled by the bit combination of n selection lines.
- A decoder with enable input can function as a demultiplexer.
  - In the figure, one-to-four line demultiplexer
    - E is the input
    - A and B are selection inputs
    - Ex: if selection AB=10, output D_2 will be same as E.

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
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



DECODERS

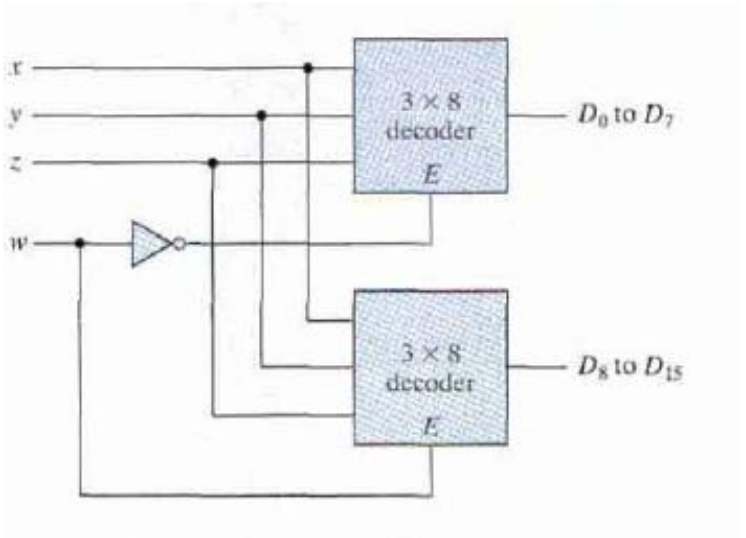
- Decoders with enable inputs can be connected to form a larger decoder circuit.
- Ex: 3-to-8 line decoders with enable inputs connected to form a 4-to-16 line decoder.
  - w=0: top decoder is enabled, the other is disabled

    bottom decoder outputs are all 0's.

    top eight outputs generate minterms 0000 to 0111
  - w=1: top decoder is disabled, the other is enabled,

    the bottom decoder outputs generate minterms 1000 to 1111

    the outputs of top decoder are all 0's



DECODERS

- Any Boolean Function can be represented in Sum of Products form.
- A decoder provides all minterms as its outputs.
- So any Boolean function with n inputs and m outputs can be implemented using a n-to-2^n line decoder and m OR gates.

Example Combinational Logic Implementation

- Implement a full adder using a decoder and OR gates.
- Truth table for full adder?

x	y	Z(carry from the previous lower significant position)	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

$$S = \sum(1,2,4,7)$$

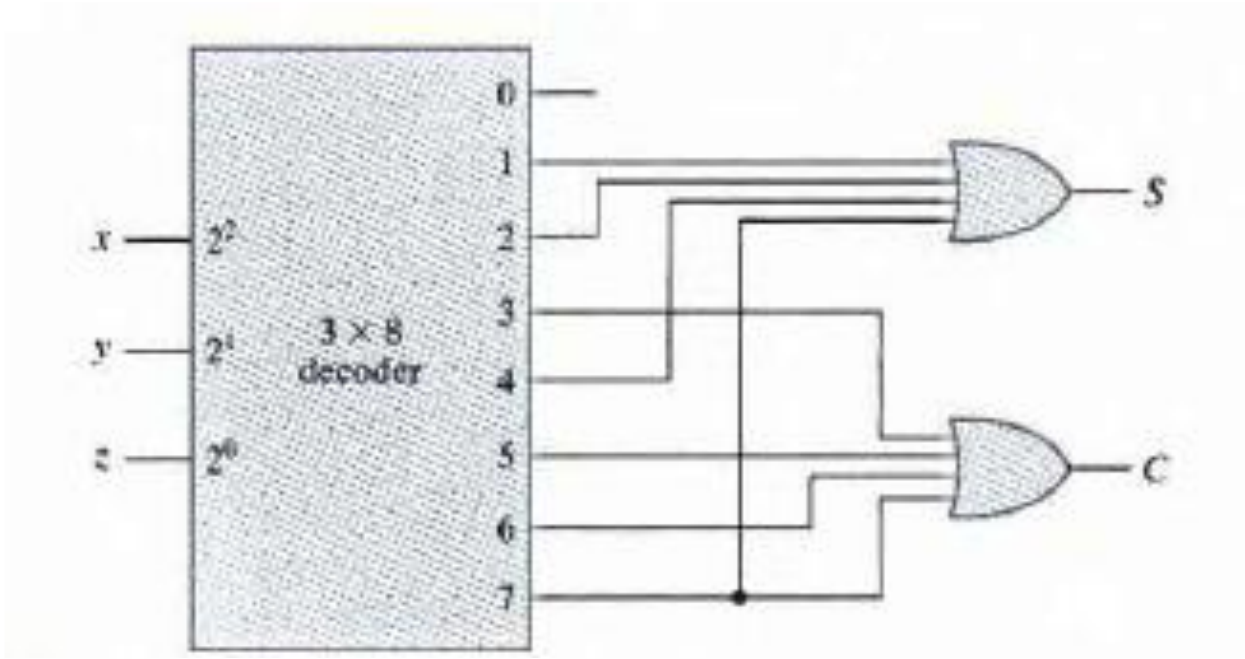
$$C = \sum(3,5,6,7)$$

Example …

$$S = \sum(1,2,4,7)$$

$$C = \sum(3,5,6,7)$$

- What do we need?
  - 3-to-8 line decoder
  - 2 OR gates



ENCODERS

- Digital Circuit that performs the inverse operation of a decoder.
- Has 2^n (or fewer) inputs and n output lines.
- The output lines generate the binary code corresponding to the input value.
- Ex: Octal-to-Binary encoder with the following truth table (it is assumed that only one input has a value of 1 at any given time.)

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
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

  - If two inputs are 1. How to resolve the ambiguity?
    - Define a priority rule for inputs.
  - How to distinguish the case all inputs are 0 and D0=1?
    - Define an additional output which will be set when at least one input is 1.

ENCODERS

- Priority Encoder: If two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

- V(valid bit) is set when one or more inputs are equal to 1.

ENCODERS

- Priority Encoder example:

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Karnauph maps?

	00	01	11	10
00	X	1	1	1
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

	00	01	11	10
00				
01				
11				
10				

	00	01	11	10
00				
01				
11				
10				

ENCODERS

- Priority Encoder example:

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Karnauph maps?

	00	01	11	10
00	X	1	1	1
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

$x=D_2+D_3$

	00	01	11	10
00	x	1	1	
01	1	1	1	
11	1	1	1	
10		1	1	

$y=D_3+D_1D_2'$

	00	01	11	10
00	0			
01				
11				
10				

$V=D_0+D_1+D_2+D_3$

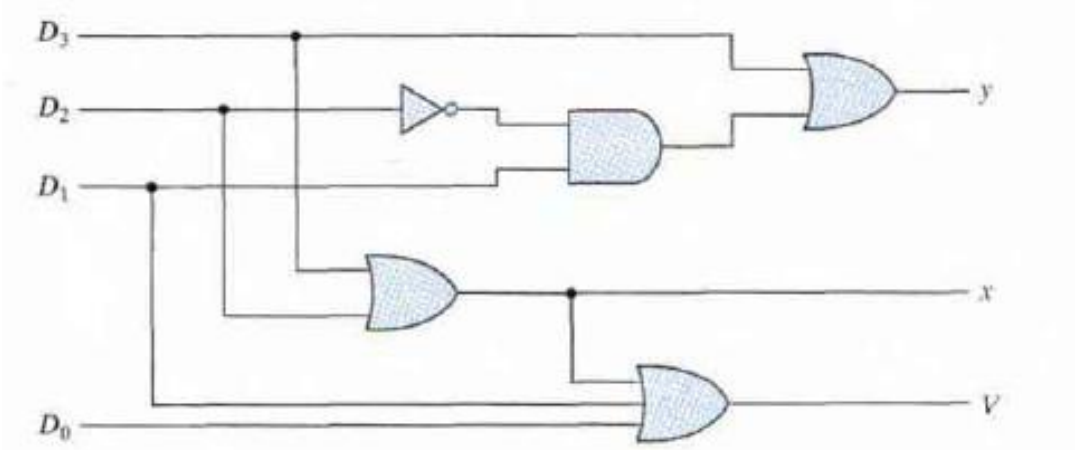
ENCODERS

- Priority Encoder

$x=D_2+D_3$

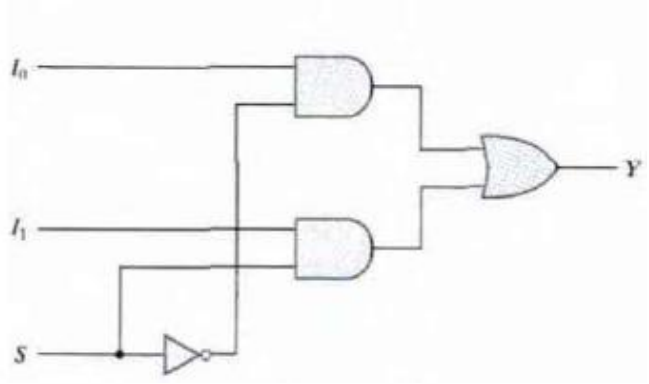
$y=D_3+D_1D_2'$

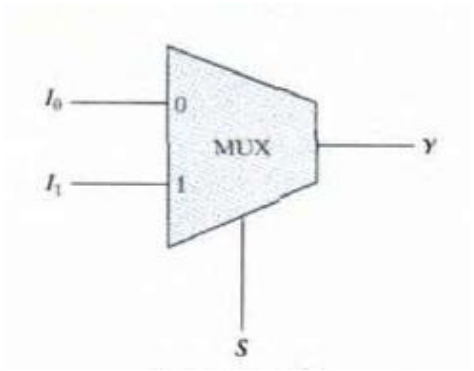
$V=D_0+D_1+D_2+D_3$



Multiplexers

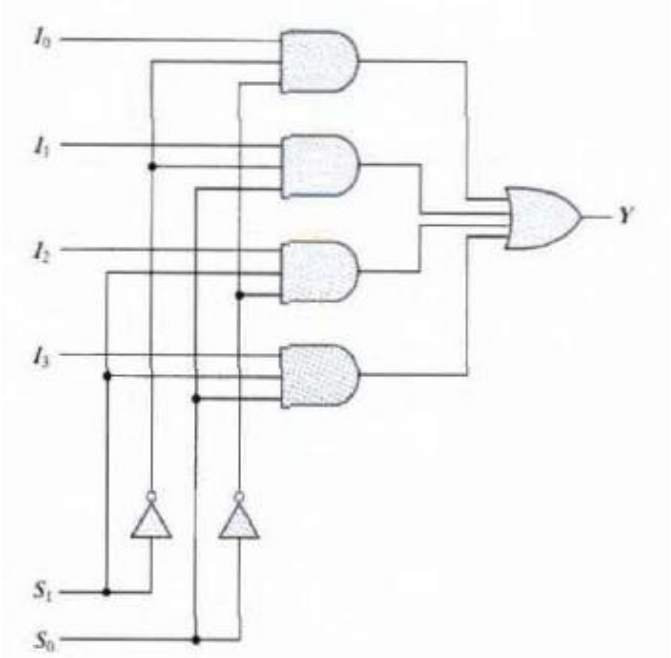
- A multiplexer is a combinational circuit that selects Binary information from one of many input lines and directs it to a single output line.
- The selection of a particular input line is controlled by a set of selection lines.
- There are 2^n input lines and n selection lines whose bit combinations determine which input is selected.
- Ex: a two-to-one-line multiplexer and its block diagram:





Multiplexers

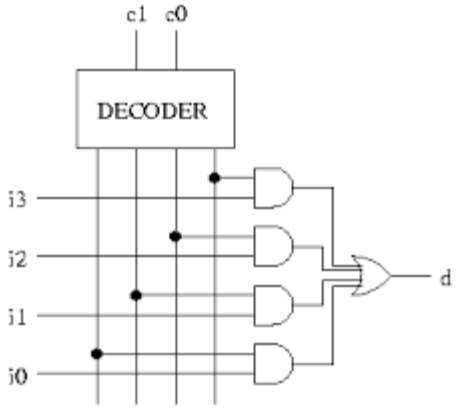
- Ex: A four-to-one-line multiplexer's logic diagram and function table:



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

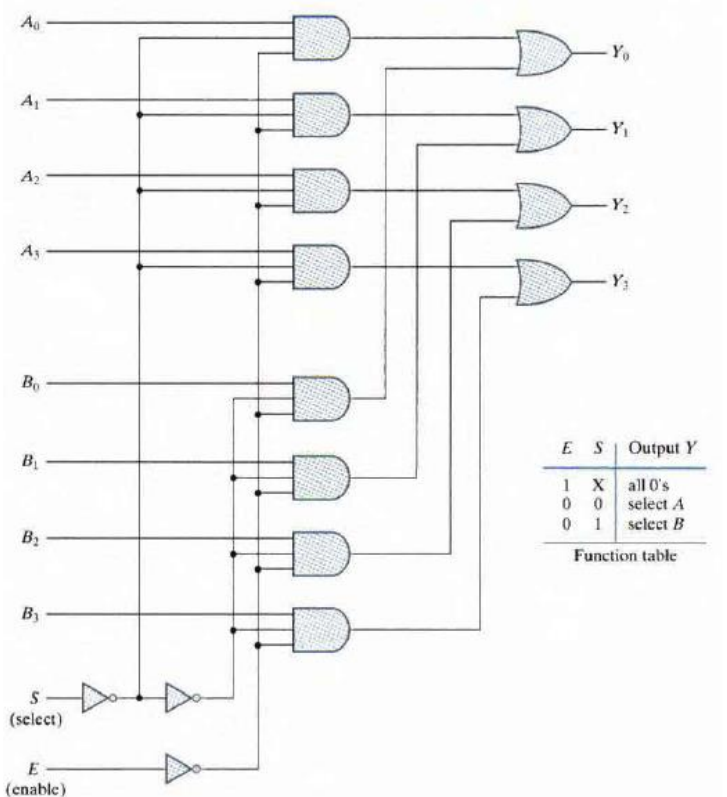
Multiplexers

- A multiplexer is also called a selector, since it selects one of many inputs.
- A multiplexer may have an enable input to control the operation of the unit.
- A multiplexer can be constructed using a decoder.
  - Ex: a 4-to-1-line multiplexer can be constructed using a 2-to-4 line decoder and gates. How?



Multiplexers

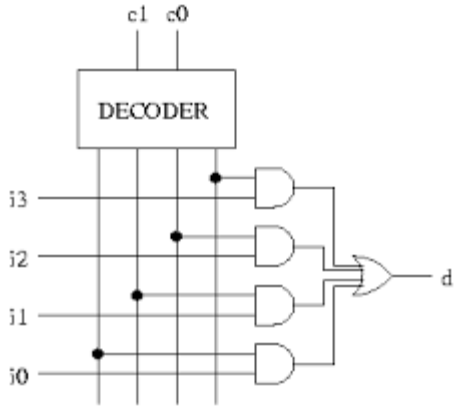
- Multiplexer circuits can be combined with common selection inputs to provide multiple-bit selection logic.
- Ex: A quadruple 2-to-1 line multiplexer's logic diagram and function table:



E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B

Function table

Multiplexers



- Boolean Function implementation:
  - Previously we have seen how to use a decoder to implement a boolean function.
  - A multiplexer is a decoder that includes the OR gate within the unit.
    - The minterms of the function are generated in a multiplexer by the circuit associated with the selection inputs.
    - The individual minterms can be selected by the data inputs.
    - A boolean function with n variables can be implemented with a multiplexer that has n selection inputs and 2^n data inputs, one for each minterm.

Multiplexers

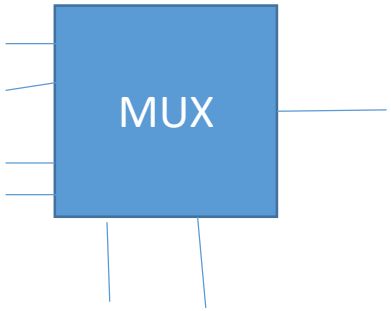
- Boolean Function implementation:
  - Boolean function of n variables can be implemented with a multiplexer that has n-1 selection inputs.
  - The n-1 variables of the function are connected to the selection inputs of the multiplexer.
  - The remaining single variable of the function is used for the data inputs.
  - If the sigle variable is z, each data of the multiplexer will be z, z’, 1 or 0.

Multiplexers

- Boolean Function implementation:
  - Ex: F(x,y,z)= $\sum(1,2,6,7)$

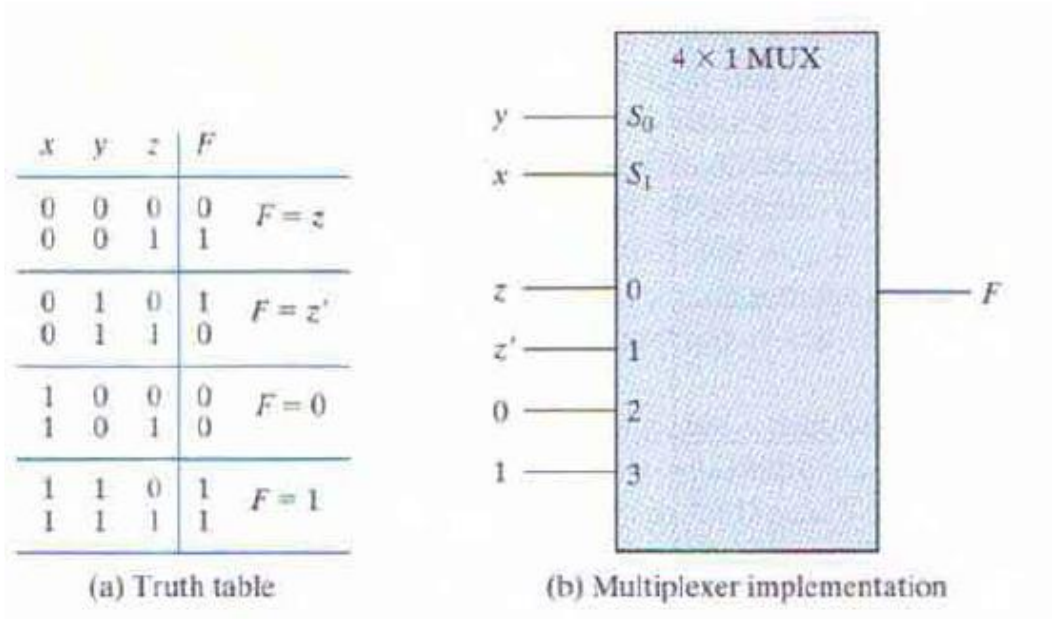
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

How to represent F using inputs?



Multiplexers

- Boolean Function implementation:
  - Ex: F(x,y,z)= $\sum(1,2,6,7)$



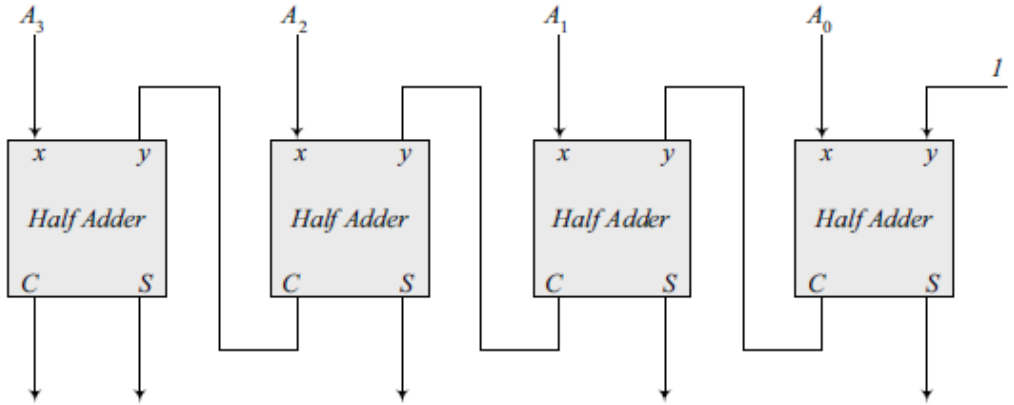
x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table

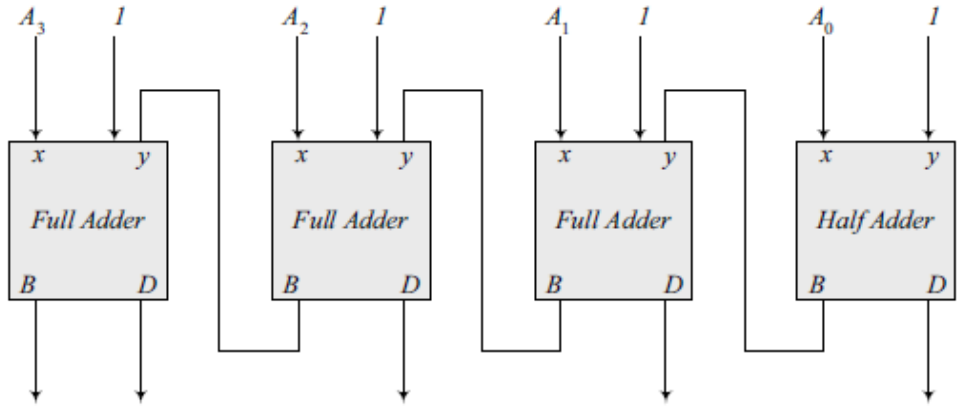
(b) Multiplexer implementation

Review questions

- Using half adders:
  - Design a four-bit combinational circuit incrementer (a circuit that adds 1 to a four-bit binary number)
  - Design a four-bit combinational circuit decrementer (a circuit that subtracts 1 from a four-bit binary number)



incrementer



decrementer

Review questions

- Construct a 16x1 multiplexer with two 8x1 and one 2x1 multiplexers. Use block diagrams.