Lecture 9 – Combinational Logic – Part I

Rami Abielmona
University of Ottawa

February 07, 2015

ITI 1100 B

Digital Systems I

Presentation Outline

- Introduction
- Design Procedure
- Adder-Subtractor
- Decoder
- 2-to-4 and 3-to-8 Decoders
- Key terms

Combinational logic circuits

- → outputs logical functions of inputs
- → new outputs appear shortly after changed inputs (propagation delay)
- → no feedback loops
- → no clock

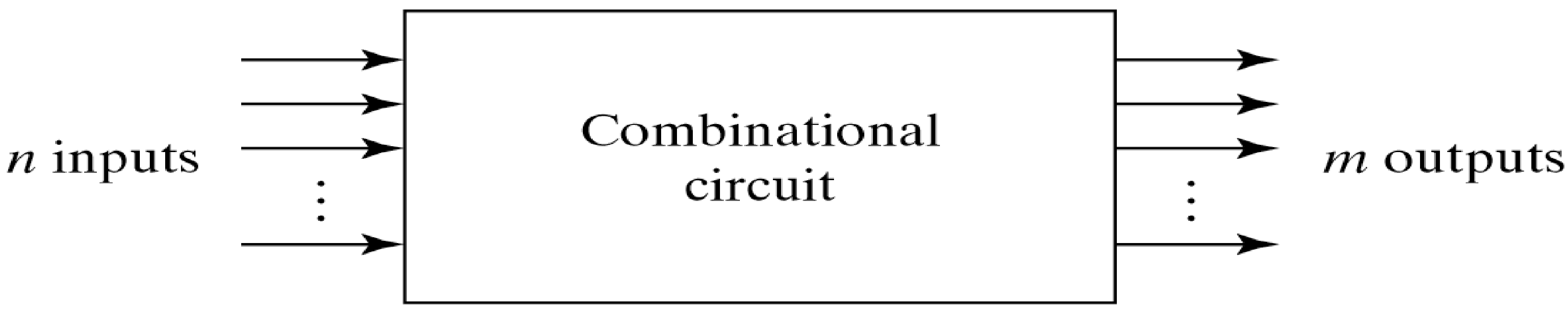


Fig. 4-1 Block Diagram of Combinational Circuit

Design Procedure

Design a circuit from a specification.

1. Determine number of required inputs and outputs.

2. Derive truth table

3. Obtain simplified Boolean functions

4. Draw logic diagram and verify correctness

Example: Code Converter

– A circuit that translates one binary code to another

- Example 3-2: BCD to Excess-3 Code Converter
  – Excess-3 code: decimal code + 3
  – BCD inputs 1010 to 1111 are don't care conditions

Decimal Digit	Input BCD				Output Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Simplification with K-map

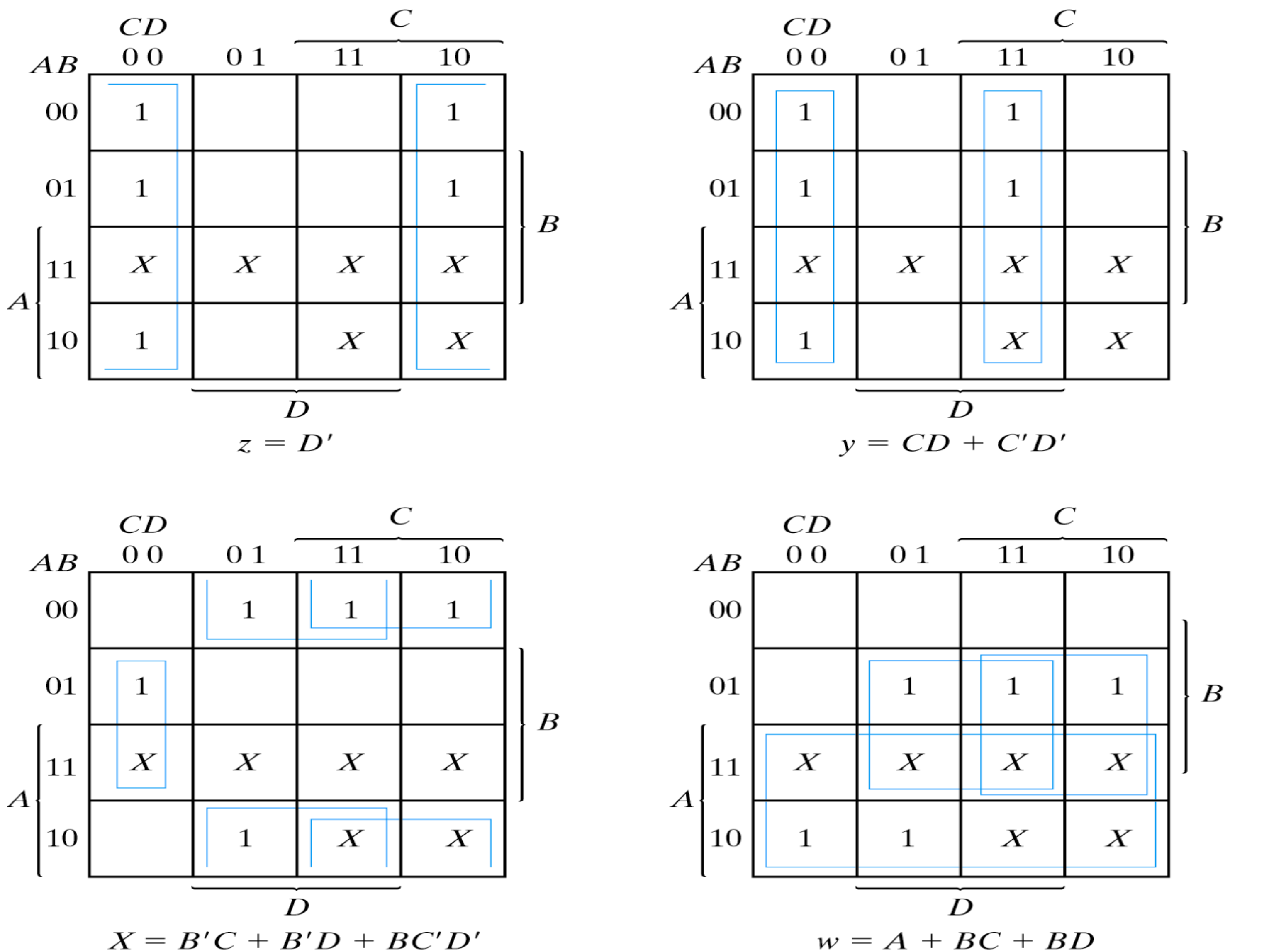


Fig. 4-3 Maps for BCD to Excess-3 Code Converter

Further manipulation of simplified expressions

- Two-level AND-OR implementation for the circuit can be obtained directly from the Boolean expression derived from the K-MAPS.
- Further manipulation can be done on the function to allow use of common gates for multiple-output circuits.
- Thus there are several possibilities for the implementation. The following shows the implementation with 3 levels of gates.

$$W = A + BC + BD = A + B(C + D)$$

$$X = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$

$$= B'(C+D) + B(C+D)'$$

$$Y = CD + C'D' = CD + (C + D)'$$

$$Z = D'$$

Three Level Implementation

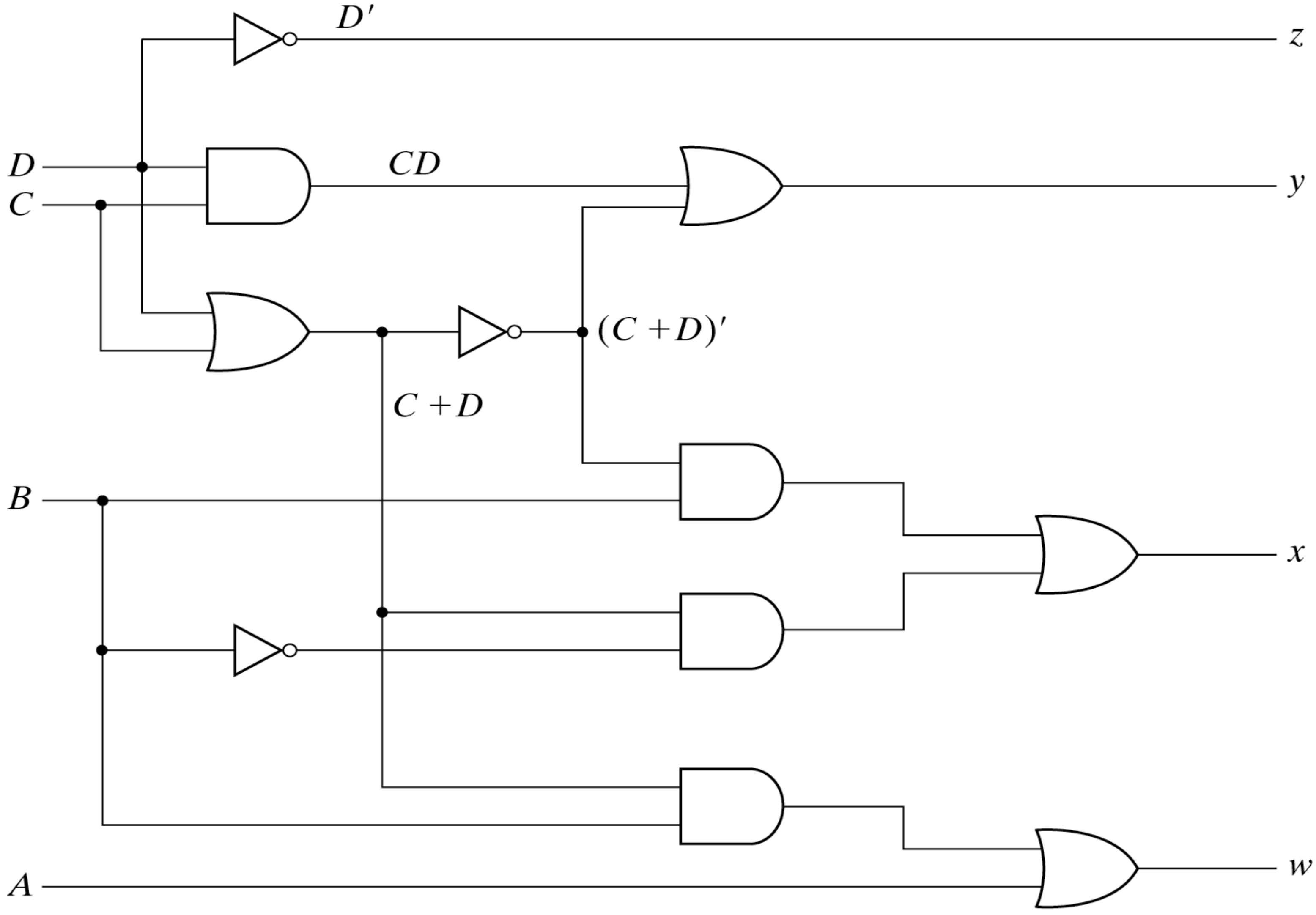


Fig. 4-4 Logic Diagram for BCD to Excess-3 Code Converter

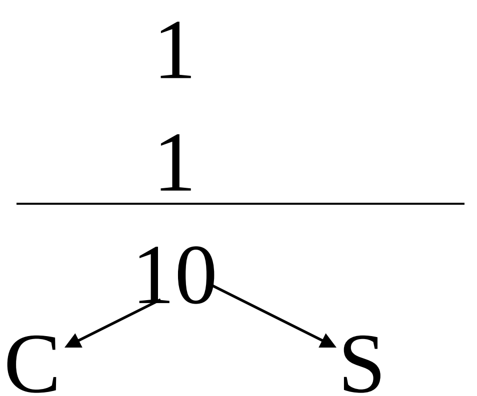
Binary Adder -Subtractor

Half Adder

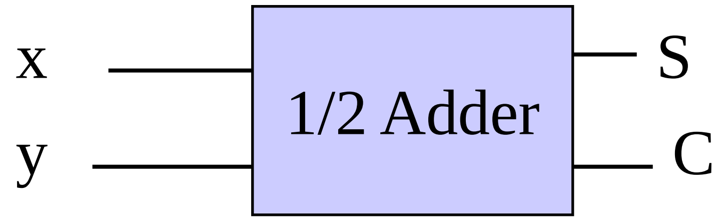
→The half-adder accepts two binary digits on its inputs and produces two binary digits on its outputs: a sum bit and a carry bit.

Truth Table

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Half-Adder (see other implementations in Chap. 2)



Truth Table

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Some of products

S = x'y + xy'

C = xy

Half-Adder-Implementation

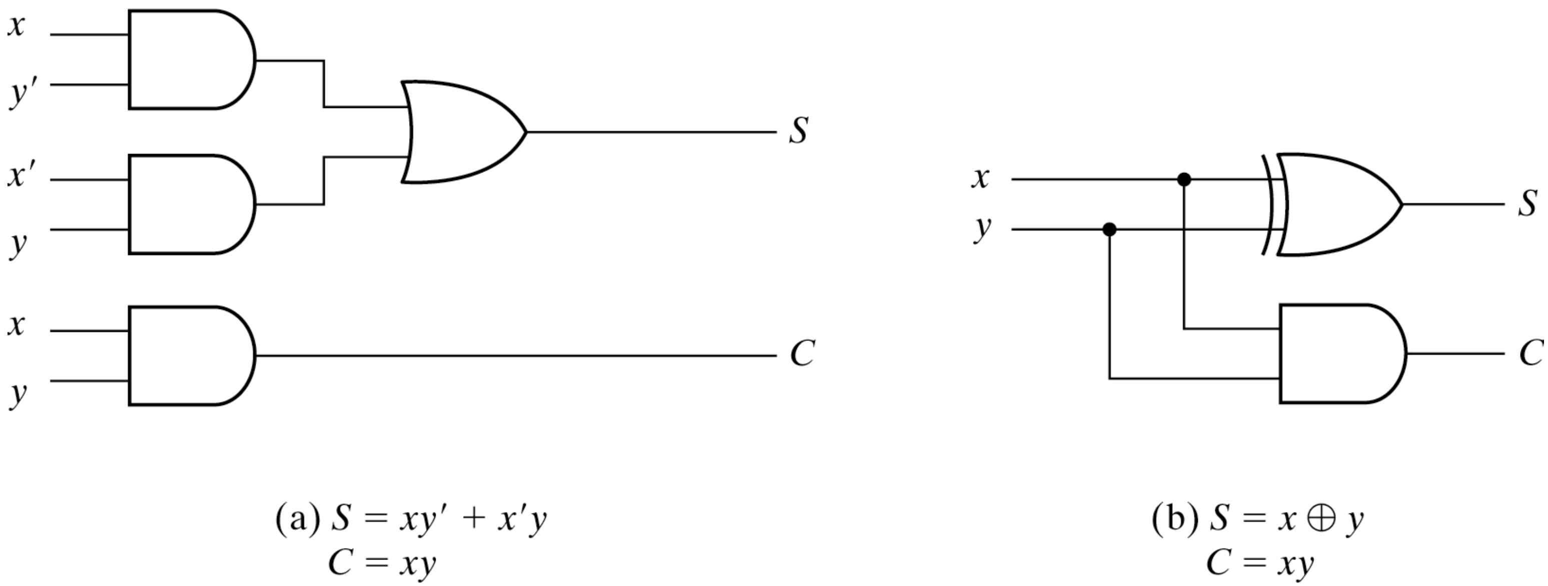


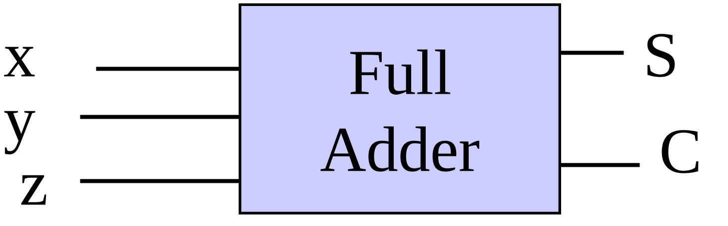
Fig. 4-5 Implementation of Half-Adder

Full Adder (see other implementations in Chap. 2)

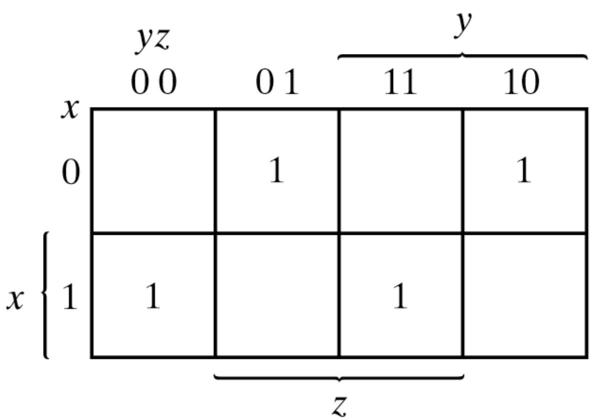
Truth Table

x	y	z	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

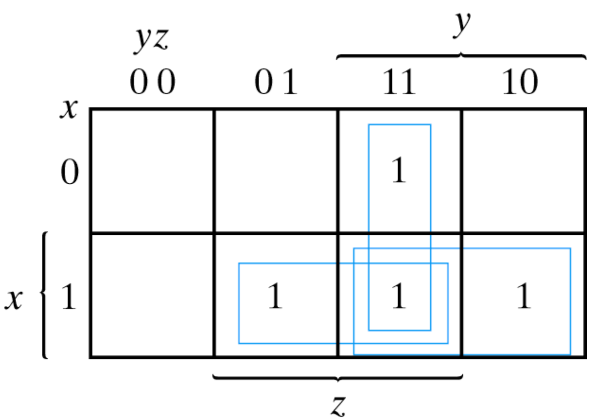
→ The Full-adder is combinational circuit



Full Adder (see other implementations in Chap. 2)



$S = x'y'z + x'yz' + xy'z' + xyz$



$C = xy + xz + yz$

Fig. 4-6 Maps for Full Adder

Full Adder (see other implementations in Chap. 2)

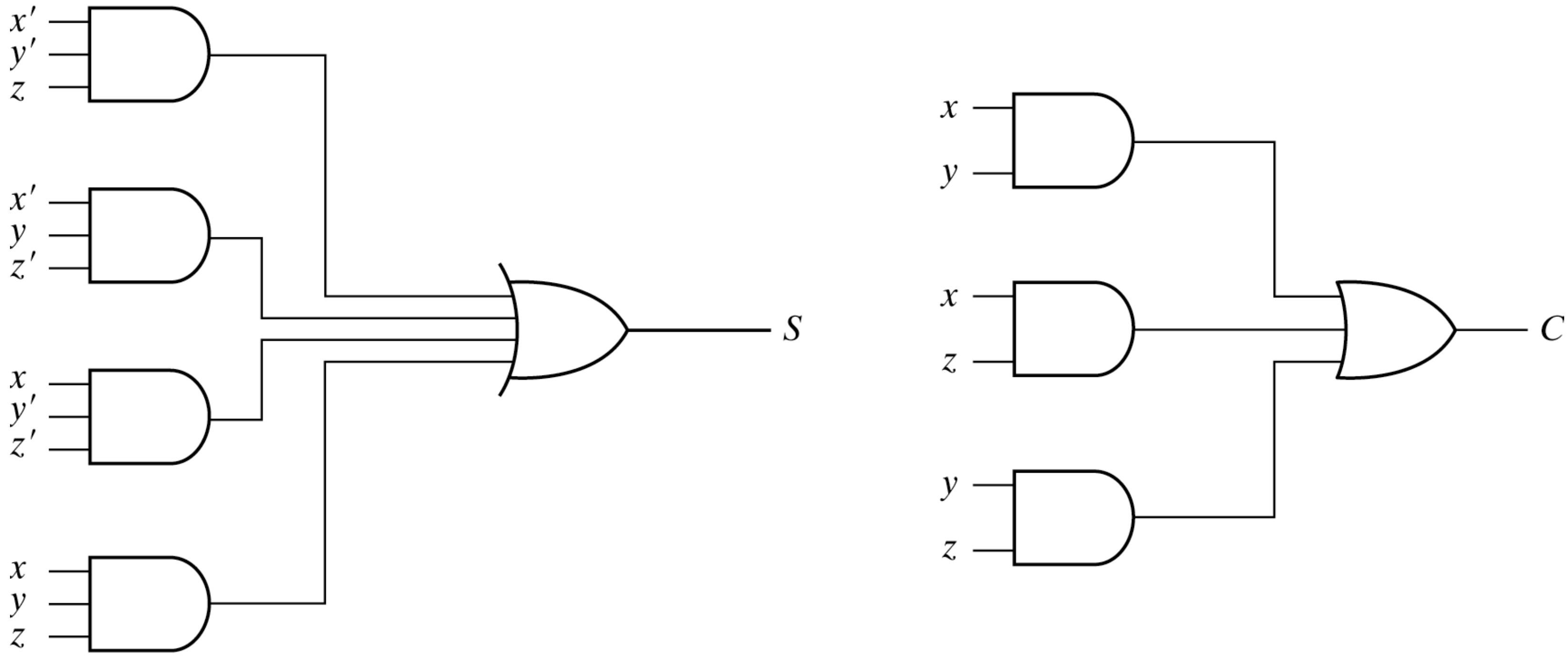


Fig. 4-7 Implementation of Full Adder in Sum of Products

Full Adder (same as in Chap. 2)

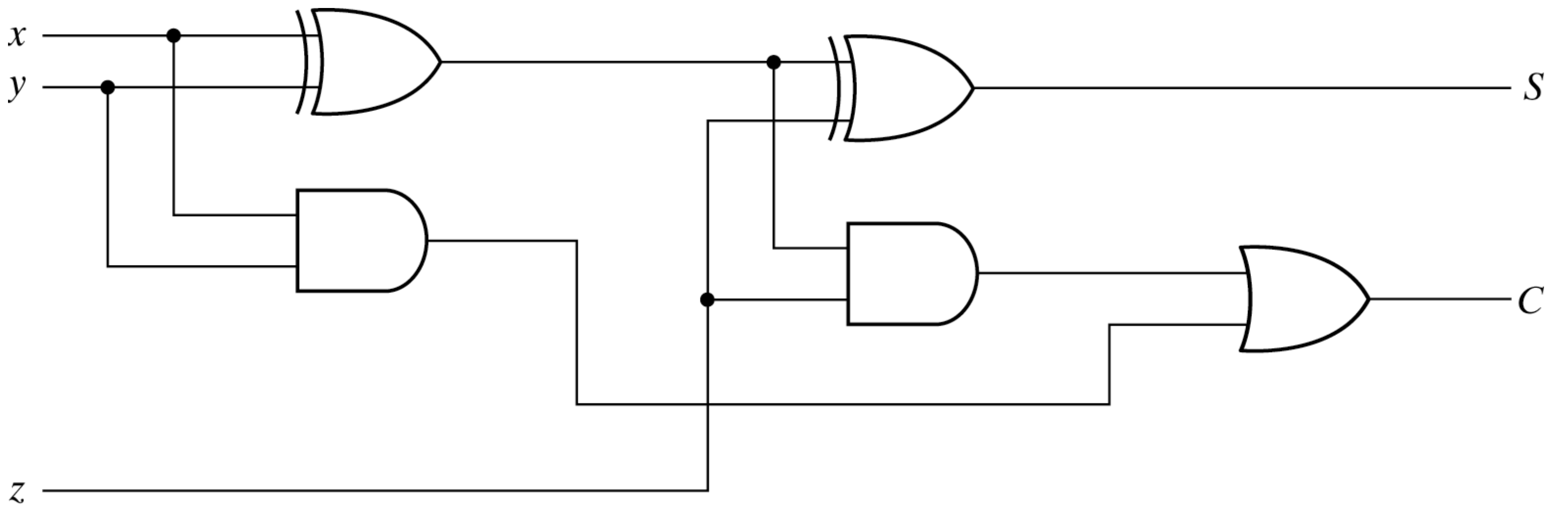
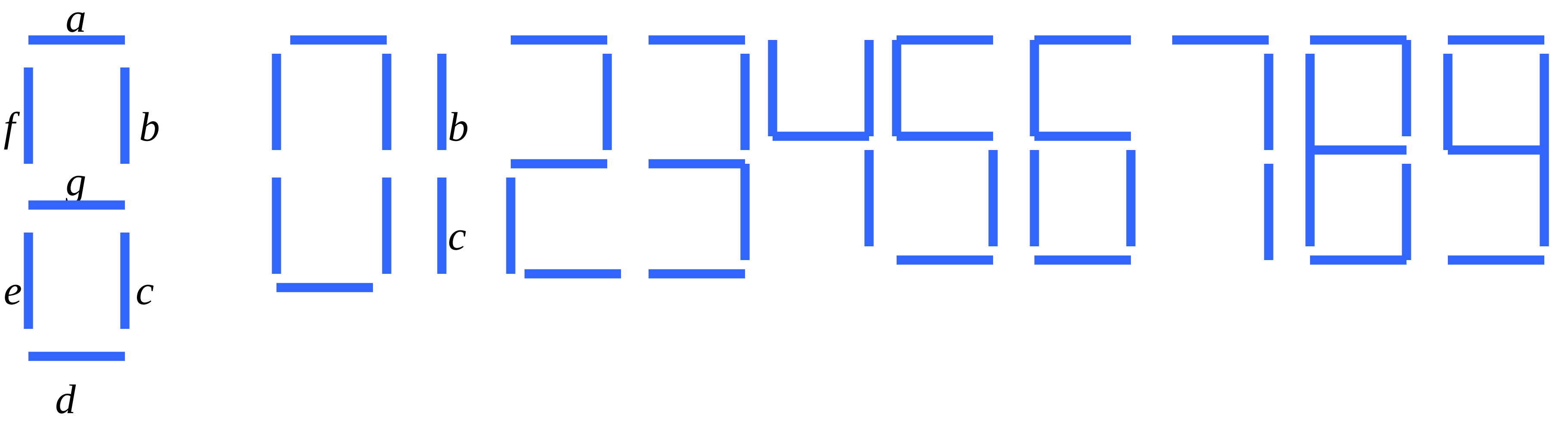


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

Decoder -Example

a BCD to Seven Segment Decoder inputs data in BCD form and converts it to a seven segment output



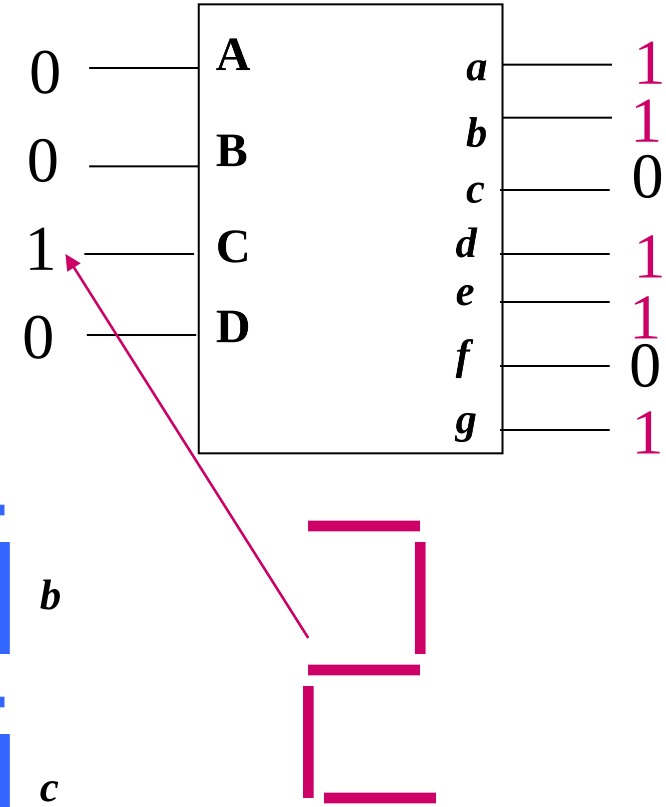
(a) Segment designation

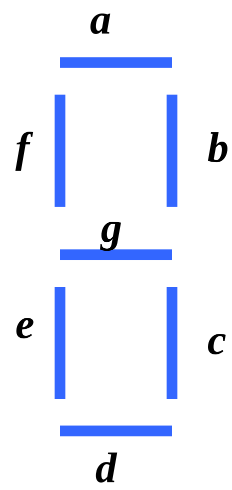
(b) Numerical designation for display

A- BCD to Seven Segment Decoder

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x

Don't care terms





K-MAP

AB \ CD	00	01	11	10
00	1		1	1
01		1	1	1
11				
10	1	1		

***a* =**

AB \ CD	00	01	11	10
00	1	1	1	1
01	1		1	
11				
10	1	1		

***b* =**

AB \ CD	00	01	11	10
00	1	1	1	
01	1	1	1	1
11				
10	1	1		

***c* =**

AB \ CD	00	01	11	10
00	1		1	1
01		1		1
11				
10	1	1		

***d* =**

a

AB \ CD	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	d	d	d	d
10	1	1	d	d

a = A + BD + B'D'+ C

b

AB \ CD	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	d	d	d	d
10	1	1	d	d

b = CD + C'D' + B'

c

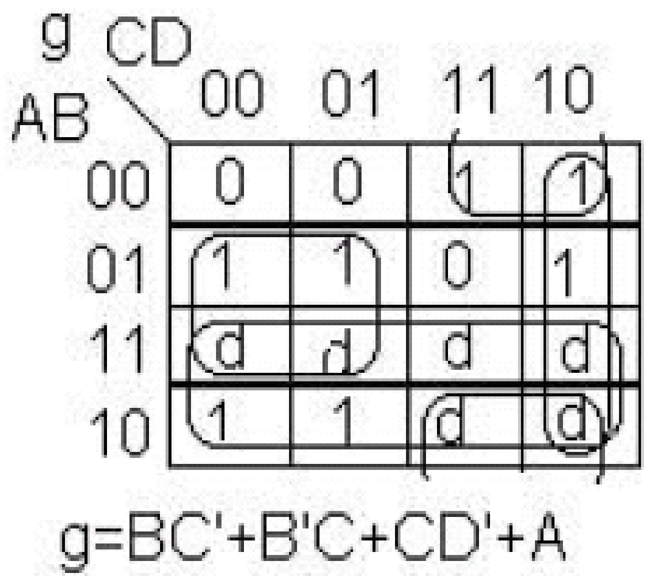
AB \ CD	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	d	d	d	d
10	1	1	d	d

c= B + C'+ D

d

AB \ CD	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	d	d	d	d
10	1	0	d	d

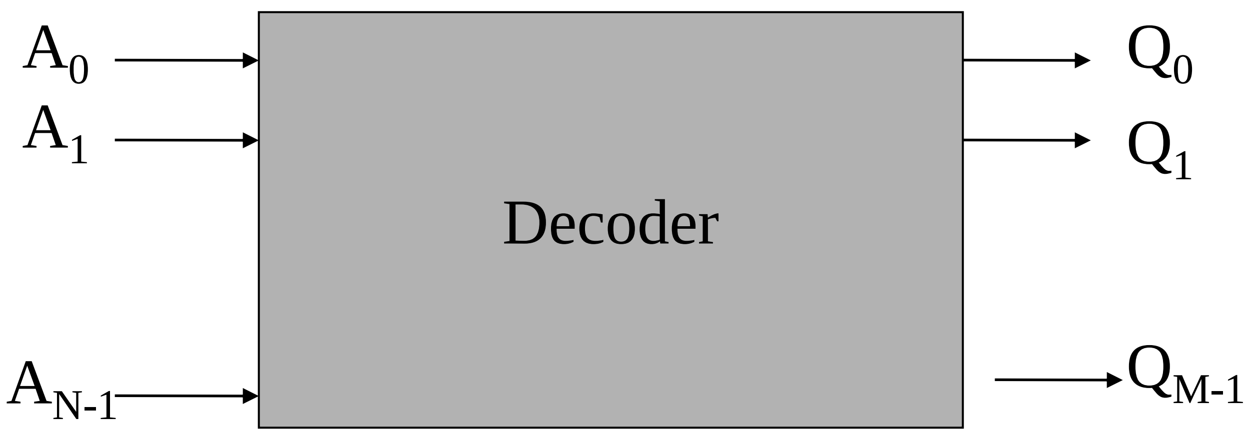
d=BC'D+B'C+B'D'+CD'


g
CD
AB
00
01
11
10
00
01
11
10
0
0
1
1
1
1
0
1
d
d
d
d
1
1
d
d
g=BC'+B'C+CD'+A

Decoder

- 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-* 2^n decoder
- If the n-bit coded information has unused combinations → less than 2^n outputs.

→ *n-to-m* decoder, $m \leq 2^n$, Example: BCD-to-7-segment decoder, where *n=4* and *m=7*



2-to-4 Decoder

→A 2-to-4 decoder operates according to the following truth table.

- The 2-bit input is called S1S0, and the four outputs are Q0-Q3.
- If the input is the binary number i, then output Qi is uniquely true.

S1	S0	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- For example, if the input S1 S0 = 10 (decimal 2), then output Q2 is true, and Q0, Q1, Q3 are all false.
- This logic circuit "decodes" a binary number into a "one-of-four" code.

Building a 2-to-4 decoder?

- Same deign procedure as for the combinational logic circuit (see previous slides). From the truth table, we can derive equations for each of the four outputs (Q0-Q3), based on the two inputs (S0-S1).

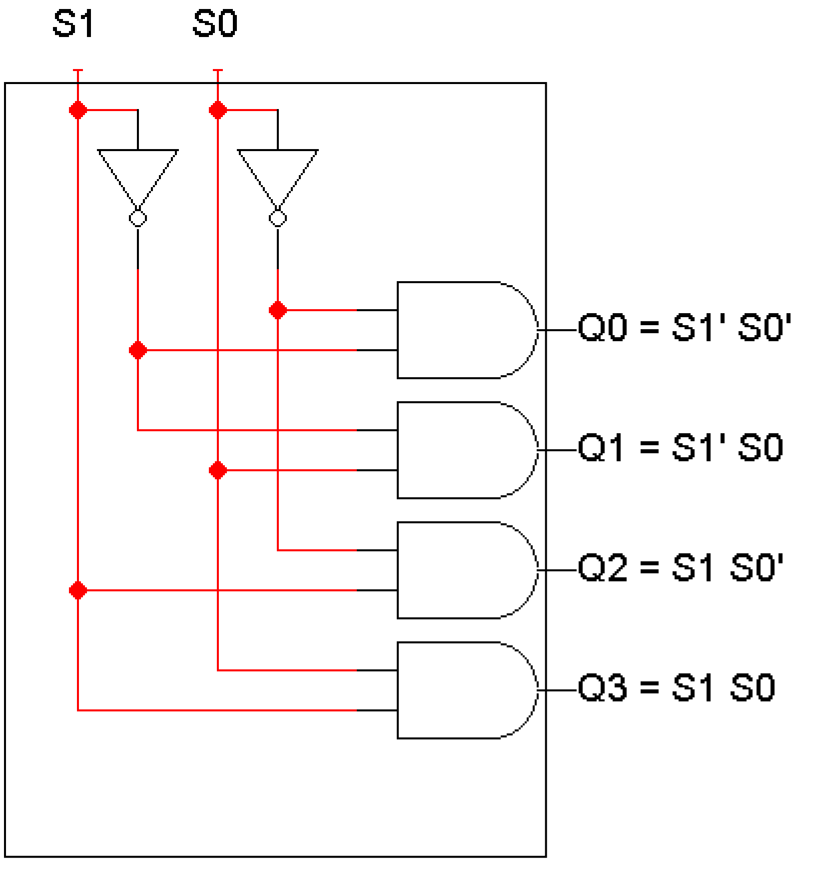
S1	S0	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- There is no much to be simplified. the equations are:

$$Q0 = S1' \, S0'$$
$$Q1 = S1' \, S0$$
$$Q2 = S1 \, S0'$$
$$Q3 = S1 \, S0$$

Implementation of 2-to-4 decoder

S1	S0	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Enable inputs

- Many devices have an additional enable input, which is used to "activate" or "deactivate" the device.
- For a decoder,
  - EN=1 activates the decoder, so it behaves as specified earlier. Exactly one of the outputs will be 1.
  - EN=0 "deactivates" the decoder. By convention, that means *all* of the decoder's outputs are 0.
- We can include this additional input in the decoder's truth table:

EN	S1	S0	Q0	Q1	Q2	Q3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	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

abbreviated truth tables

- In this table, note that whenever EN=0, the outputs are always 0, ***regardless*** of inputs S1 and S0.

EN	S1	S0	Q0	Q1	Q2	Q3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	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

- We can abbreviate the table by writing x's in the input columns for S1 and S0.

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

Decoder- a Minterm Generator

S1	S0	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$Q0 = S1' \; S0'$

$Q1 = S1' \; S0$

$Q2 = S1 \; S0'$

$Q3 = S1 \; S0$

- Decoders are sometimes called minterm generators.
  - For each of the input combinations, exactly one output is true.
  - Each output equation contains all of the input variables.
  - These properties hold for all sizes of decoders.
- Therefore we can implement arbitrary functions with decoders.

→from a sum of minterms equation for a function, we can use a decoder (a minterm generator) to implement that function.

3- to- 8 line Decoder

- Three inputs, x, y, z, are decoded into eight outputs, D0 through D7
- Each output D*i* represents one of the minterms of the 3 input variables.
- D*i* = 1 when the binary number xyz = *i*
- Shorthand: D*i* = m*i*
- The output variables are *mutually exclusive*; exactly one output has the value 1 at any time, and the other seven are 0.

3- to- 8 line Decoder

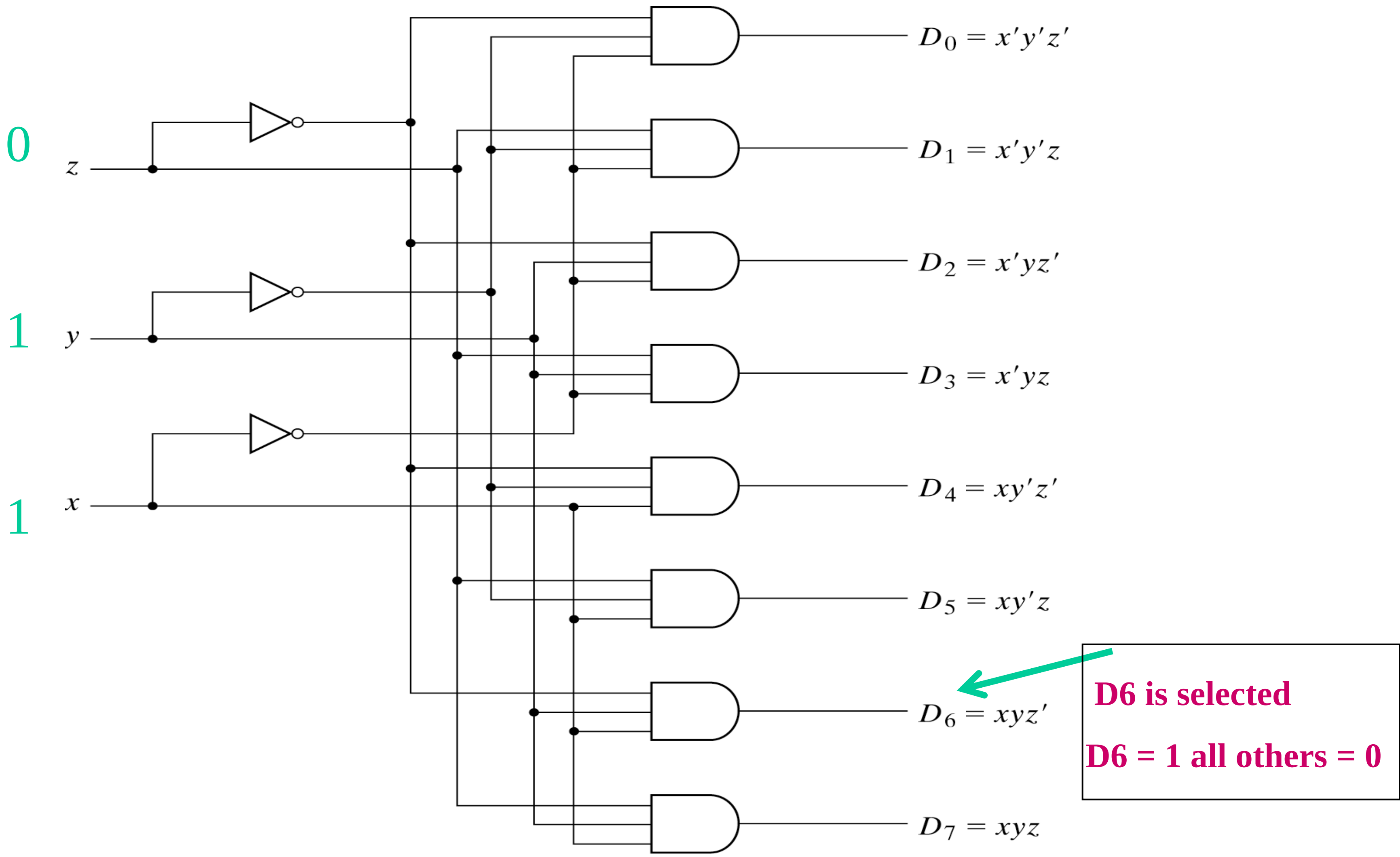


Fig. 4-18 3-to-8-Line Decoder

Key Terms and Review Points

- 2-to-4 Decoder
- 3-to-8 Decoder
- Abbreviated Truth Tables
- Code Converter
- Combinational Circuit
- Enable Input
- Minterm Generator
- Multiple Output circuit
- Mutually Exclusive Outputs
- Propagation Delay

- References: Dr. Karmouch ITI 1100 Slides