DIGITAL DESIGN AND COMPUTER ORGANIZATION

Combinational Logic

Team DDCO

Department of Computer Science and Engineering

Introduction

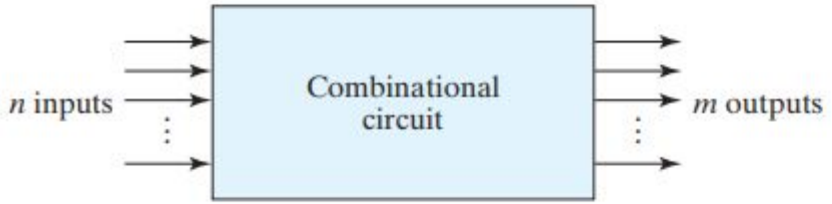
- There are two types of Logic Circuits:
  - Combinational
  - Sequential

- Combinational Circuits consists of logic gates whose output at any time is function of the **present inputs only**.

- Sequential Circuits along with logic gates use storage elements. The value stored in these elements is due to a previous combination of inputs. Thus the present output of a sequential circuits depends upon **the present and the past inputs.**

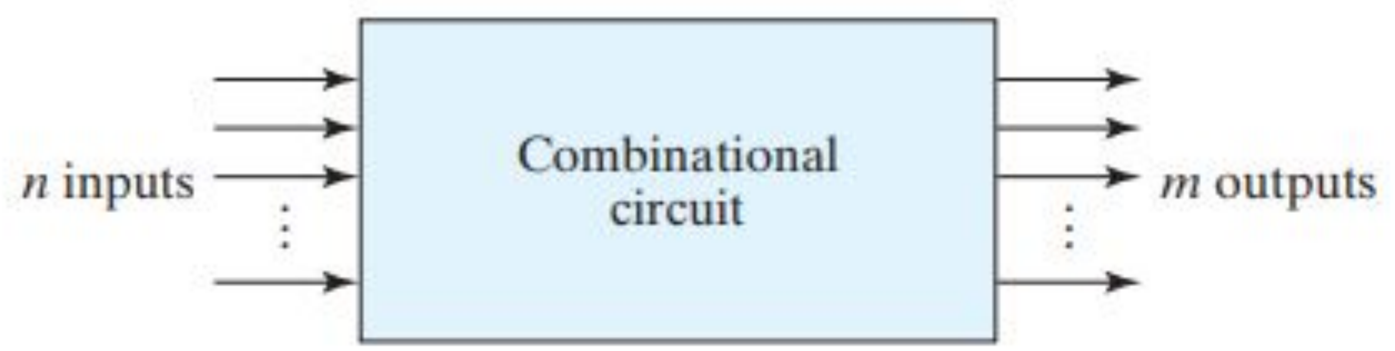
Combinational Circuits

- A combinational circuit consists of input variables,logic gates and output variables.
- For n input variables there will be a total **2^n input combinations** and the pattern of m outputs w.r.t the different combination can be listed in the form of a truth table.
- A combinational circuit can be represented by a truth table having **n inputs & m outputs** or it can be represented by a Boolean function having m equations one for each output variable.

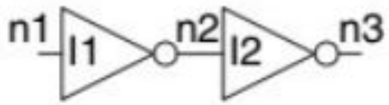


Combinational Circuits

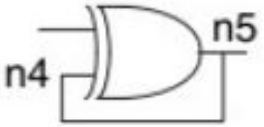
<u>Block diagram of combinational circuit</u>



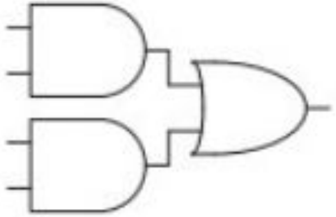
Identify the Circuit



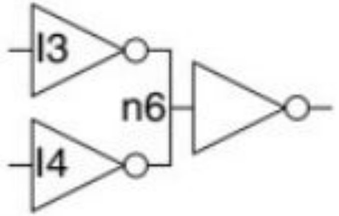
(a)



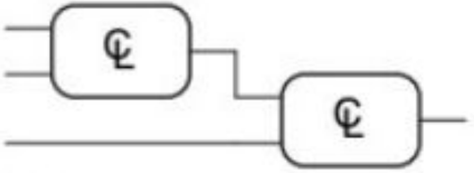
(b)



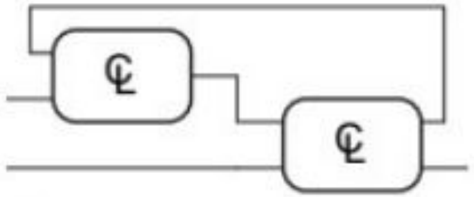
(c)



(d)



(e)



(f)

Identify the Circuit

A. is **combinational**. It is constructed from two combinational circuit elements (inverters I1 and I2). It has three nodes: n1, n2, and n3. n1 is an input to the circuit and to I1; n2 is an internal node, which is the output of I1 and the input to I2; n3 is the output of the circuit and of I2.

B. is **not combinational**, because there is a cyclic path: the output of the XOR feeds back to one of its inputs.

C. is **combinational**.

D. is **not combinational**, because node n6 connects to the output terminals of both I3 and I4.

E. is **combinational**, illustrating two combinational circuits connected to form a larger combinational circuit.

F. is **not combinational** ,does not obey the rules of combinational composition because it has a cyclic path through the two elements.

Analysis Procedure

- The analysis starts with a given logic diagram and culminates with
  - **A set of Boolean functions**
  - **A Truth Table**
  - **or, possibly, an explanation of the circuit operation**
- The first step in the analysis is to make sure that the given circuit is **combinational and not sequential.**
- The diagram of a **combinational** circuit has logic gates with **no feedback paths or memory elements**.
- A feedback path is a connection from the output of one gate to the input of a second gate whose output forms part of the input to the first gate.

Analysis Procedure

To obtain the output Boolean functions from a logic diagram, we proceed as follows:

1. Label all gate outputs that are a function of input variables with arbitrary symbols— but with meaningful names. Determine the Boolean functions for each gate output.
2. Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean functions for these gates.
3. Repeat the process outlined in step 2 until the outputs of the circuit are obtained.
4. By repeated substitution of previously defined functions, obtain the output Boolean functions in terms of input variables.

Analysis Procedure

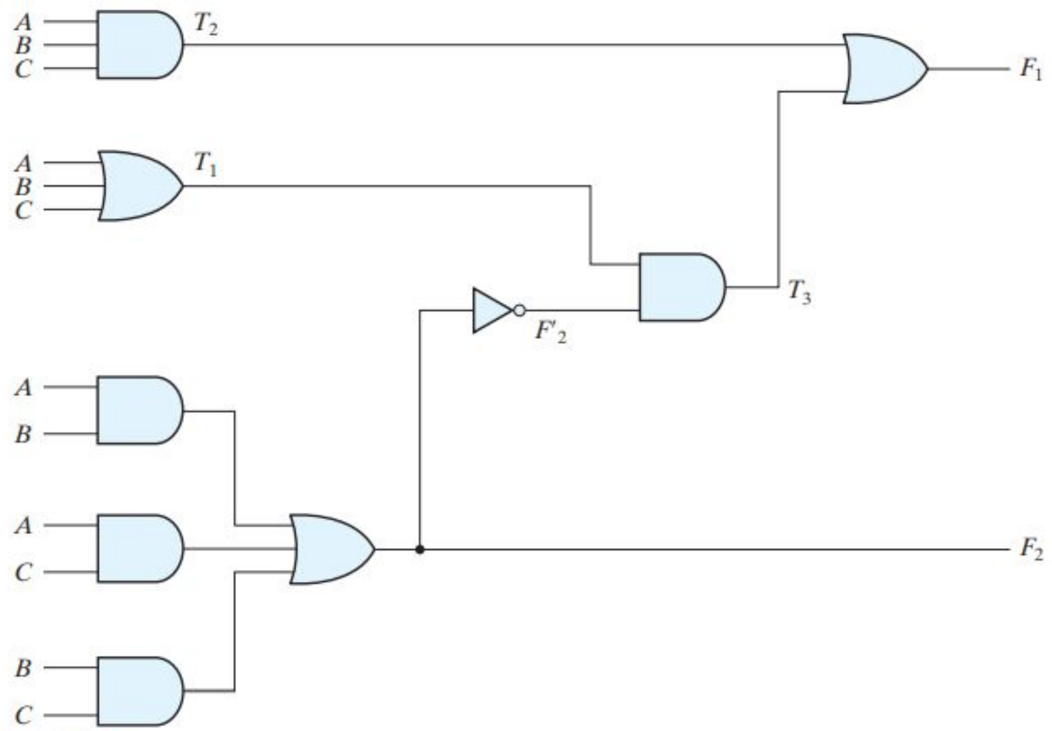


FIGURE 4.2
Logic diagram for analysis example

Analysis Procedure

The analysis of the combinational circuit of Fig. 4.2 illustrates the proposed procedure. We note that the circuit has three binary inputs—A, B, and C—and two binary outputs—F_1 and F_2. The outputs of various gates are labeled with intermediate symbols. The outputs of gates that are a function only of input variables are T_1 and T_2. Output F_2 can easily be derived from the input variables. The Boolean functions for these three outputs are

$$
\begin{aligned}
F_2 &= AB + AC + BC \\
T_1 &= A + B + C \\
T_2 &= ABC
\end{aligned}
$$

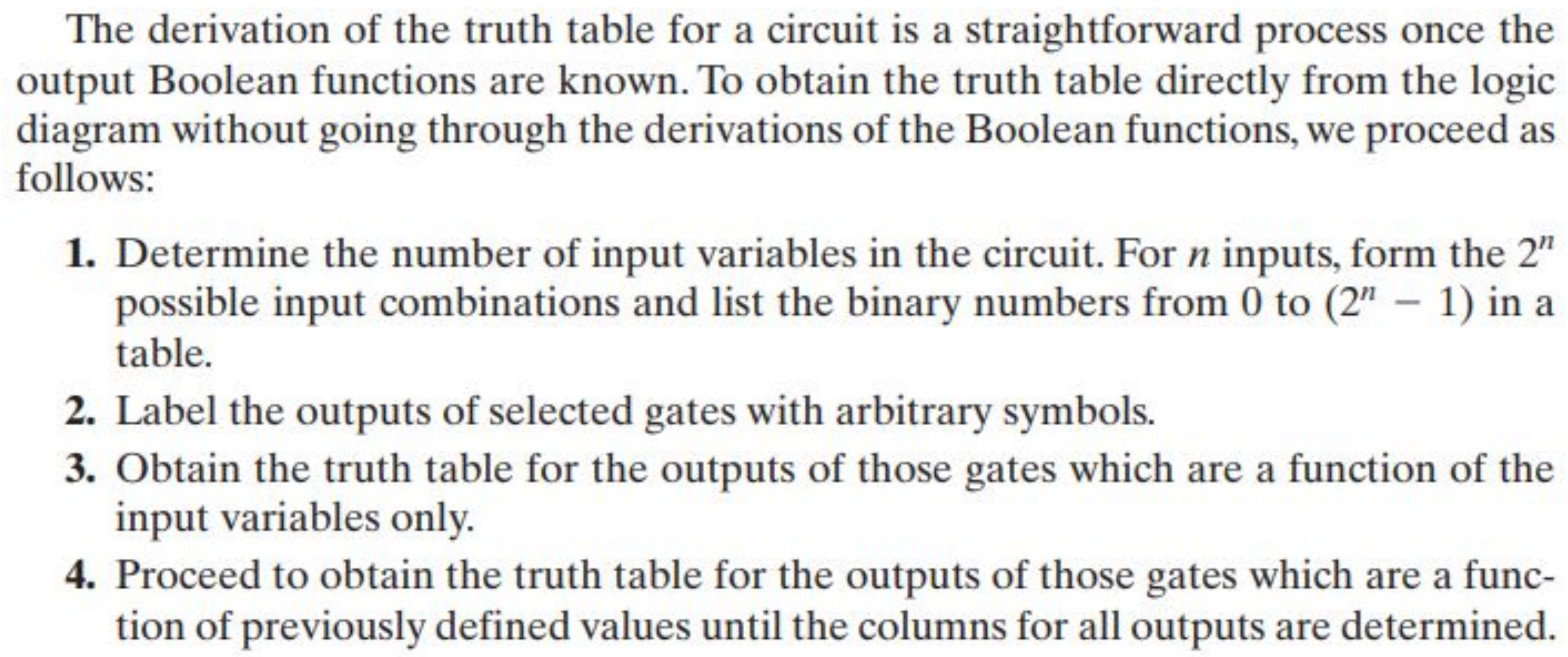
Next, we consider outputs of gates that are a function of already defined symbols:

$$
\begin{aligned}
T_3 &= F'_2 T_1 \\
F_1 &= T_3 + T_2
\end{aligned}
$$

To obtain F_1 as a function of A, B, and C, we form a series of substitutions as follows:

$$
\begin{aligned}
F_1 = T_3 + T_2 = F'_2 T_1 + ABC &= (AB + AC + BC)'(A + B + C) + ABC \\
&= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC \\
&= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC \\
&= A'BC' + A'B'C + AB'C' + ABC
\end{aligned}
$$

Analysis Procedure



The derivation of the truth table for a circuit is a straightforward process once the output Boolean functions are known. To obtain the truth table directly from the logic diagram without going through the derivations of the Boolean functions, we proceed as follows:

1. Determine the number of input variables in the circuit. For n inputs, form the 2^n possible input combinations and list the binary numbers from 0 to $(2^n - 1)$ in a table.
2. Label the outputs of selected gates with arbitrary symbols.
3. Obtain the truth table for the outputs of those gates which are a function of the input variables only.
4. Proceed to obtain the truth table for the outputs of those gates which are a function of previously defined values until the columns for all outputs are determined.

Analysis Procedure

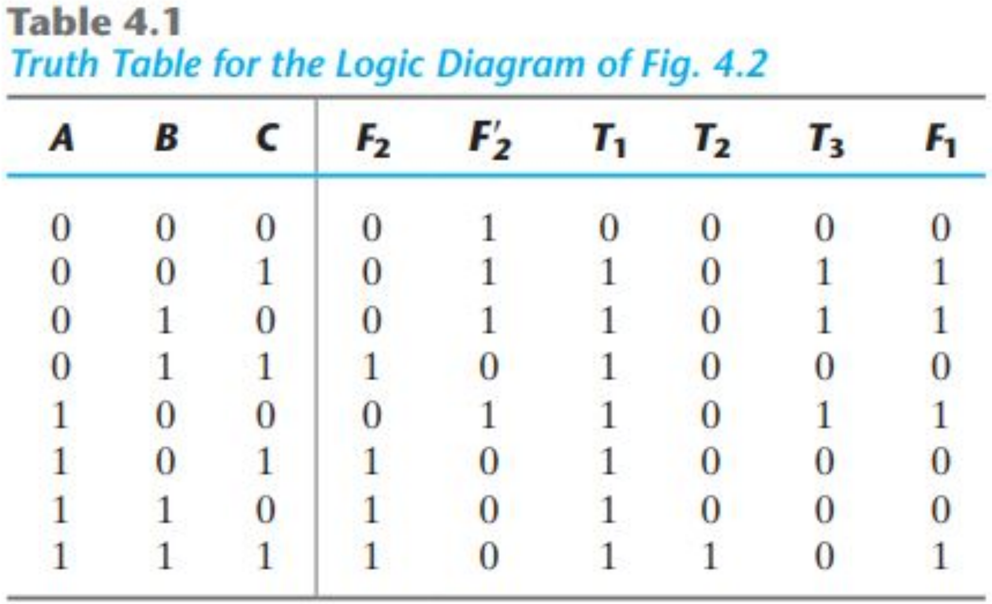


Table 4.1
Truth Table for the Logic Diagram of Fig. 4.2

A	B	C	F_2	F'_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

Design Procedure

The design of combinational circuits starts from the specification of the design objective and culminates in a logic circuit diagram or a set of Boolean functions from which the logic diagram can be obtained. The procedure involves the following steps:

1. From the specifications of the circuit, determine the required number of inputs and outputs and assign a symbol to each.
2. Derive the truth table that defines the required relationship between inputs and outputs.
3. Obtain the simplified Boolean functions for each output as a function of the input variables.
4. Draw the logic diagram and verify the correctness of the design (manually or by simulation).

Code Conversion Example

It is sometimes necessary to use the **output of one system as the input to another.** A conversion circuit must be inserted between the two systems if each uses different codes for the same information. Thus, a code converter is a circuit that makes the two systems compatible even though each uses a different binary code

Code Conversion Example

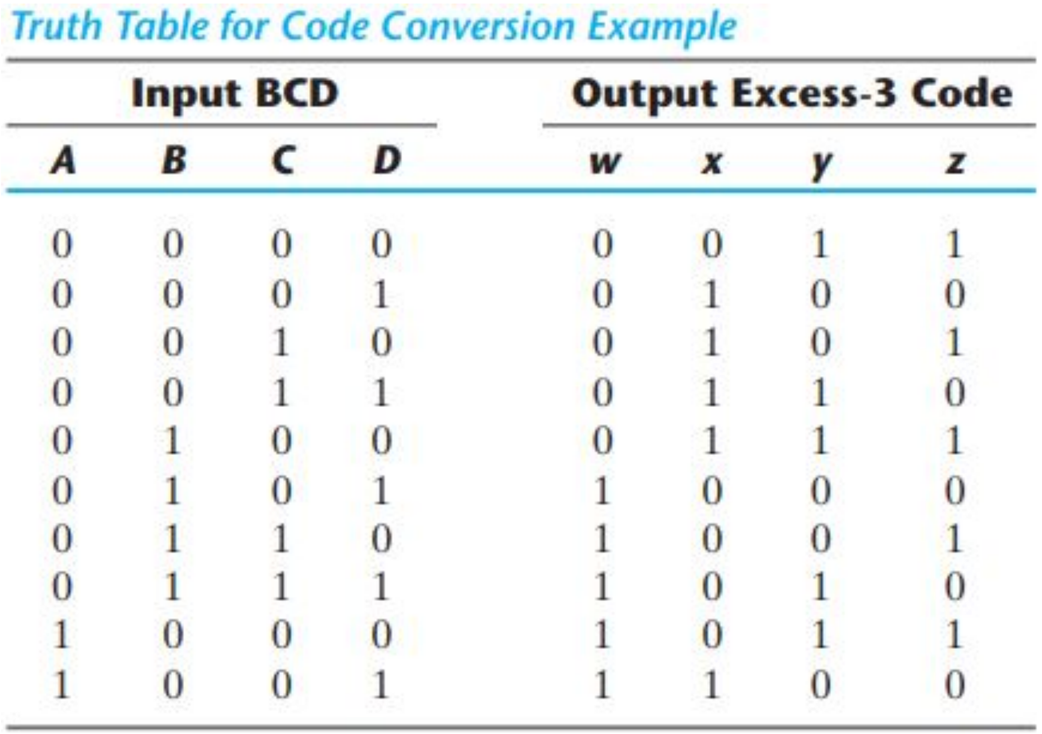
Design a circuit that converts from a BCD code to excess-3 code.

Code Conversion Example

- Some circuits use excess-3 code and we need to feed this circuit with the right input.
- In excess-3 code, numbers are represented as decimal digits, and each digit is represented by four bits as the digit value plus 3

BCD stands for **Binary Coded Decimal**. Binary coded decimal (BCD) is a system of writing numerals that assigns a four-digit binary code to each digit 0 through 9 in a decimal (base-10) numeral.
BCD code is also known as the 8 4 2 1 code.
In truth table 10-15 is don’t care output.

Code Conversion Example



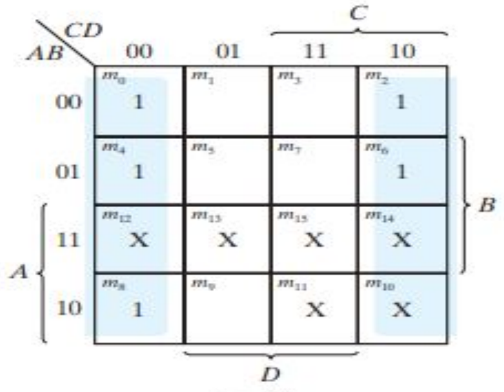
Truth Table for Code Conversion Example

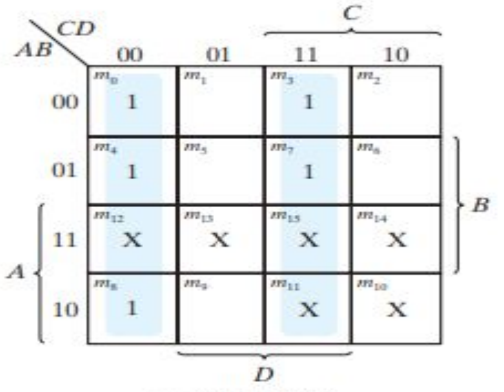
Input BCD				Output Excess-3 Code			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

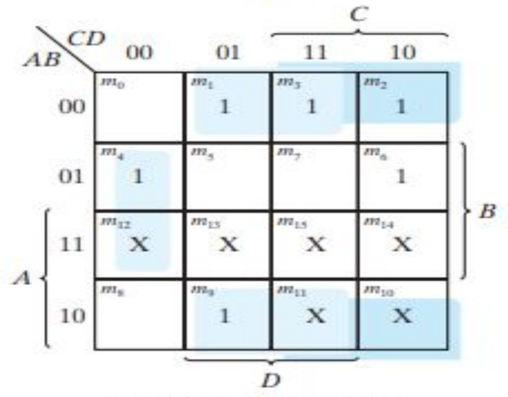
Code Conversion Example

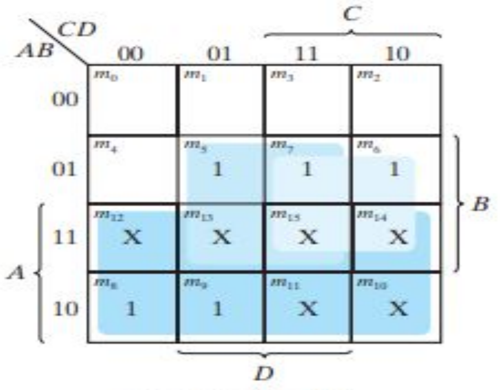
Maps for BCD-to-excess-3 code converter

$$z = D'$$
$$y = CD + C'D' = CD + (C + D)'$$
$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$
$$= B'(C + D) + B(C + D)'$$
$$w = A + BC + BD = A + B(C + D)$$


$z = D'$


$y = CD + C'D'$


$x = B'C + B'D + BC'D'$


$w = A + BC + BD$

Code Conversion Example

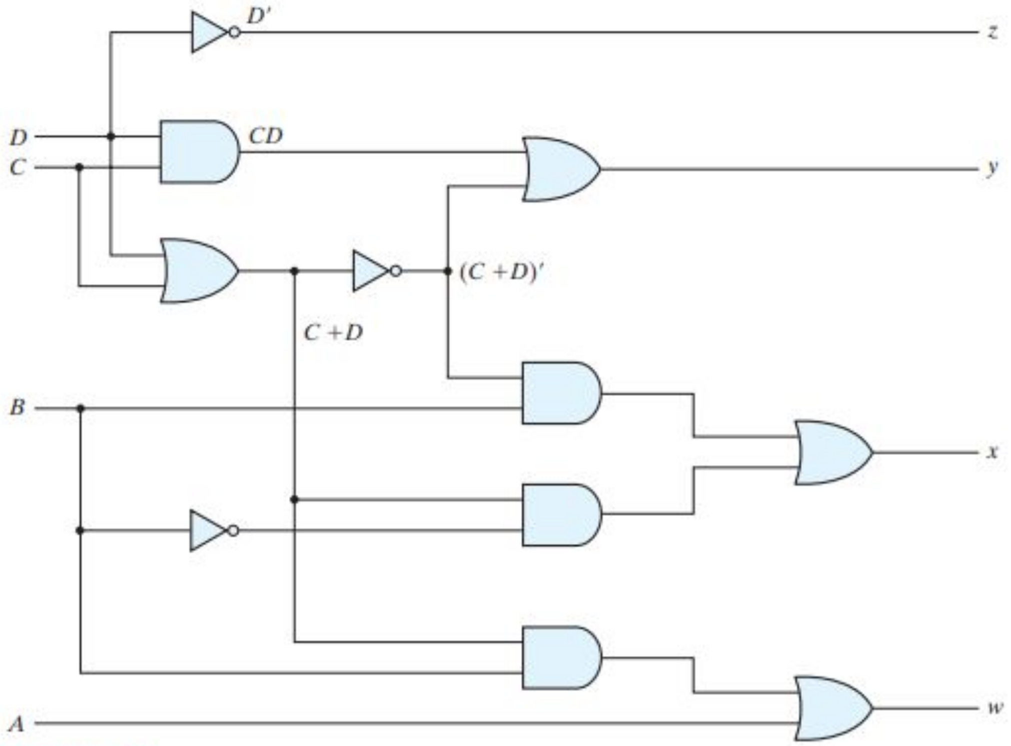


FIGURE 4.4
Logic diagram for BCD-to-excess-3 code converter

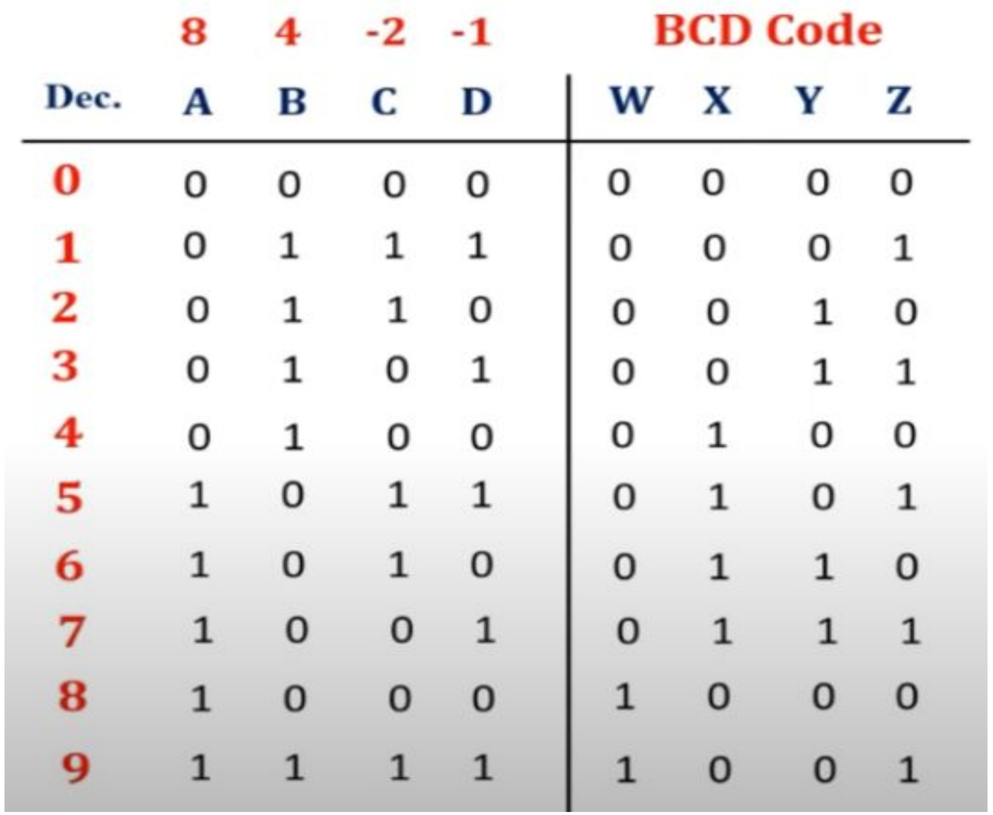
Code Conversion Example

Not counting input inverters, the implementation in sum-of-products form requires seven AND gates and three OR gates(without simplification).

The implementation of the figure in the previous slide requires four AND gates, four OR gates, and one inverter(after simplification)

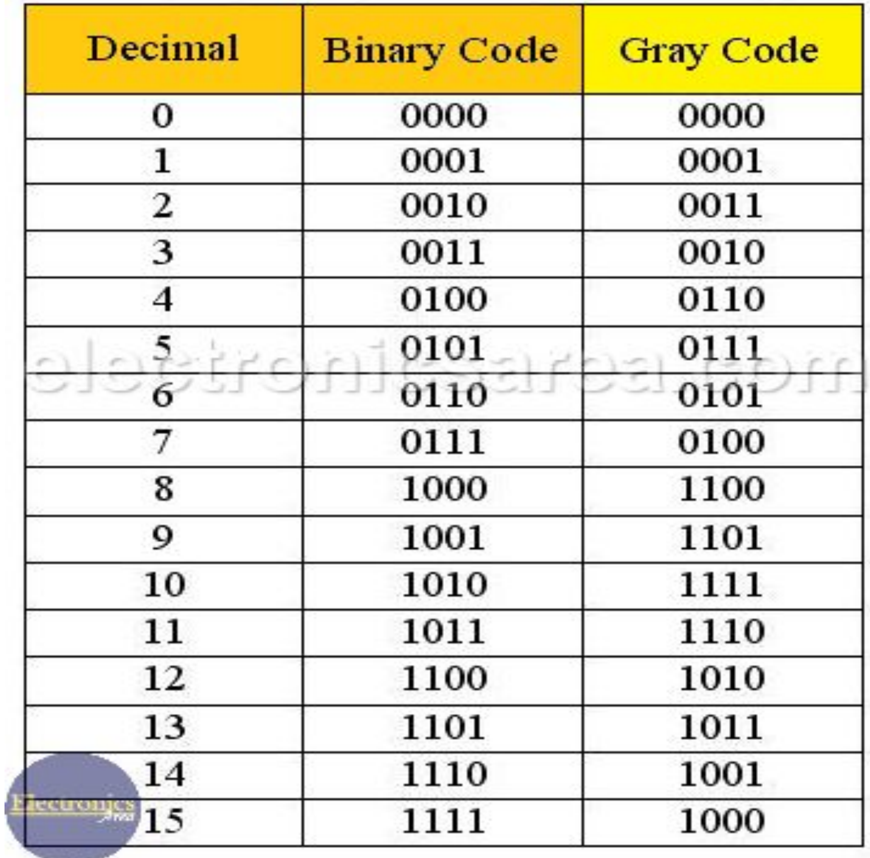
Thus, the three-level logic circuit requires fewer gates, all of which in turn require no more than two inputs.

8 4 -2 -1 Code and BCD Code



	8	4	-2	-1	BCD Code			
Dec.	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1
2	0	1	1	0	0	0	1	0
3	0	1	0	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	1	0	1	1	0	1	0	1
6	1	0	1	0	0	1	1	0
7	1	0	0	1	0	1	1	1
8	1	0	0	0	1	0	0	0
9	1	1	1	1	1	0	0	1

Gray code



Decimal	Binary Code	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Think about it

In Gray code, **only one bit** changes at a time compared to binary. What is the **Gray code equivalent** of the binary number 1101?

A) 1111
B) 1011
C) 1000
D) 1010

Think about it

In Gray code, **only one bit** changes at a time compared to binary. What is the **Gray code equivalent** of the binary number 1101?

A) 1111
B) 1011
C) 1000
D) 1010

Answer: B) 1011

Explanation:

Binary bit position	Binary bit	Rule	Gray bit
1 (MSB)	1	Copy as-is	1
2	1	$1 \oplus 1 = 0$	0
3	0	$1 \oplus 0 = 1$	1
4 (LSB)	1	$0 \oplus 1 = 1$	1

Think about it

A combinational logic circuit is designed to convert a **4-bit BCD** input into **Gray code**. If the BCD input is restricted to digits 0–9, how many valid Gray code outputs are expected?

A) 10
B) 15
C) 9
D) 16

Think about it

A combinational logic circuit is designed to convert a **4-bit BCD** input into **Gray code**. If the BCD input is restricted to digits 0–9, how many valid Gray code outputs are expected?

A) 10
B) 15
C) 9
D) 16

Answer: A) 10

Explanation:
Since BCD digits range from 0000 to 1001 (0 to 9), only **10 valid inputs** exist. So, the circuit will produce 10 valid Gray code outputs corresponding to these.

Think about it

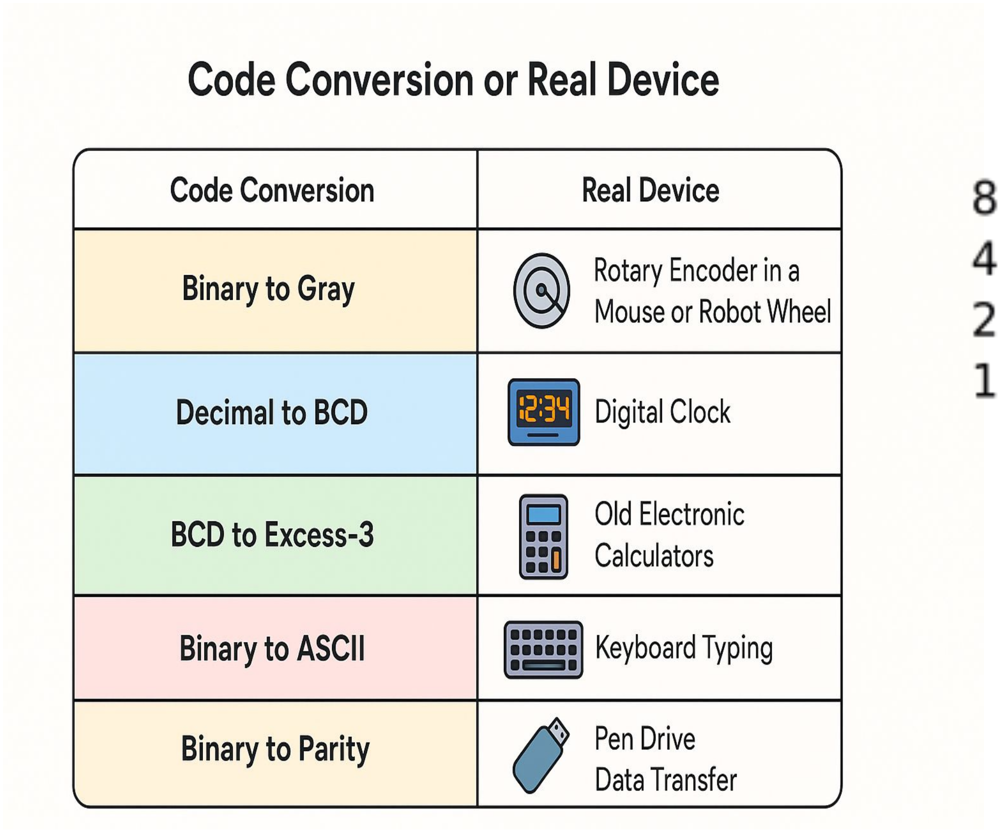
In a BCD to Excess-3 code converter, what is the output for the BCD input **0101 (5)**?

A) 1000
B) 0110
C) 1010
D) 1100

Think about it

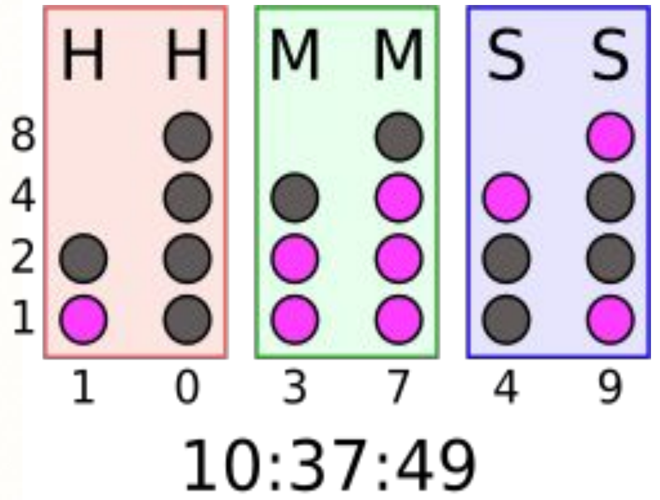
In a BCD to Excess-3 code converter, what is the output for the BCD input **0101 (5)**?
A) 1000
B) 0110
C) 1010
D) 1100
Answer: A) 1000
Explanation:
Excess-3 code = BCD + 0011.
BCD(0101) + 0011 = 1000 (8).
So, the output is **1000**.

Applications



Code Conversion or Real Device

Code Conversion	Real Device
Binary to Gray	Rotary Encoder in a Mouse or Robot Wheel
Decimal to BCD	Digital Clock
BCD to Excess-3	Old Electronic Calculators
Binary to ASCII	Keyboard Typing
Binary to Parity	Pen Drive Data Transfer



THANK YOU

Team DDCO

Department of Computer Science and Engineering