

Lesson Objectives

In this lesson, you will learn about

- What are combinational circuits
- Design procedure of combinational circuits
- Examples of combinational circuit design

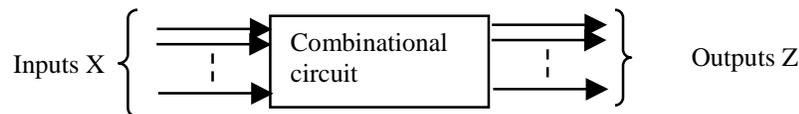
Combinational Circuits

Logic circuit can be classified into two types. Combinational circuit, which consists of logic gates whose outputs at any time are determined by combining the values of the applied inputs using logic operations, and sequential circuits, which will be studied later.

In combinational circuits, the output at any time is a direct function of the applied external inputs. In other words,

$$Z = F(X)$$

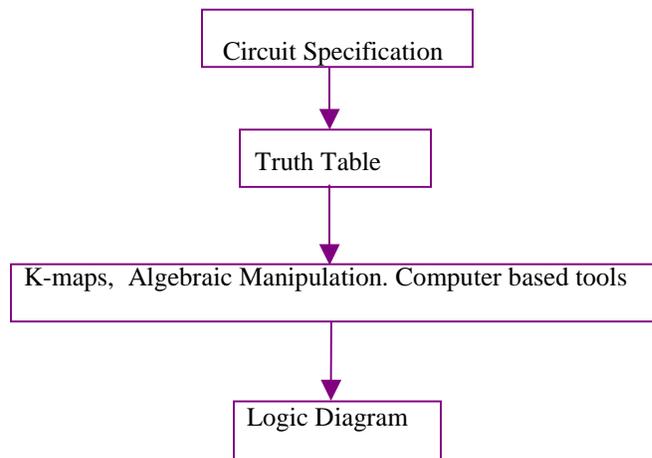
That is, the outputs depend only on present inputs



A combinational circuit can be specified either by a truth table

Design procedure

The design of a combinational circuit starts from the specification of the problem, which leads to the truth table. Using the output values in the truth table, the logic equation for output function is found and simplified using K maps, or Algebraic manipulation or computer base tools. The equation of the output functions, the corresponding circuit is found



Let us state these steps formally.

1) The first step is to find the truth table from circuit specification. This involves two sub-steps.

- The first is to determine the required number of inputs and outputs from the specification or verbal description of the problem. Then, assign a letter symbol to each input.
- Then, derive the truth table that defines the required relationship between inputs and outputs

2) Using the truth table, obtain the simplified Boolean expression for each output as a function of the input variables. The simplified equations can then be obtained using algebraic manipulation, K-maps, or computer-based tools

ain the simplified Boolean expression for each output as of the input variables using

3) Once the simplified equations are found, the corresponding logic diagram can be derived

A practical design must consider constraints such as:

- Number of gates used.
- Number of gate inputs (*Fan-in*).
- Maximum number of gates an output signal can drive (*Fan-out*).
- Speed (propagation delay) requirements.

Example

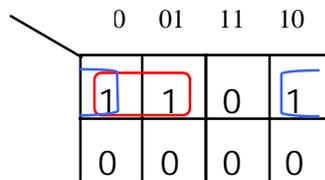
Design a combinational circuit which has 3-bit input number and a single output (F). The output signal F is specified as follows:

- $F = 1$ when the input number is less than $(3)_{10}$
- $F = 0$ otherwise.
- Implement F using only NAND gates

Let the three inputs be called X, Y, and Z. X is the most significant variable and Z is the least significant variable. The output F goes high, that is, the output produces logic 1 value if the input is less than 011, equivalent to a decimal value of three. This means that the output will be logic one for input combinations 000, 001, and 010. For other input combinations, which are 011 upto 111, the output is logic zero.

Dec. #	X	Y	Z	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

Since SOP expressions are directly implementable as 2-Level implementation of NAND gates, we consider the 1's of the function as shown in the K-map.



$$F = X' Y' + X' Z'$$

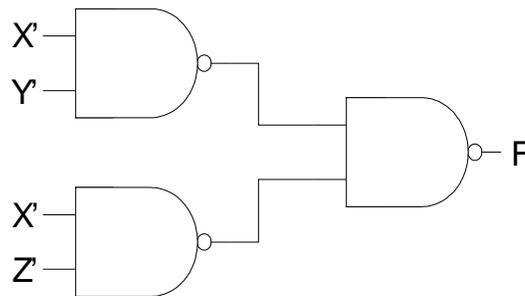


Figure: NAND-NAND- implementation for $F = X' Y' + X' Z'$

Code Converters

- Code converters are circuits which translate information from one binary code to another
- The inputs to the circuit provide the bit combination belonging to the first code, while the outputs constitute the corresponding combination belonging to the second code

- The combinational circuit performs the transformation from one code to another

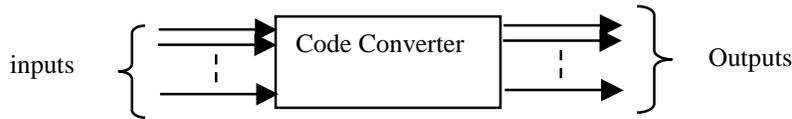


Figure shows the general structure of a code converter, containing the inputs, the code converter circuit, and the outputs. Consider, for example, a binary BCD to Excess-3 code converter

Example: BCD to excess-3 Code Converter

In this problem, the input is a BCD codeword. Since this is a 4-bit code that represents a decimal digit (0-to-9), there will be 4 input bits which will be represented by four input variables A,B,C, and D. Output is a 4-bit excess-3 code (W, X, Y ,Z)

Having defined the inputs and outputs, we proceed to build the truth table for this code converter. The truth table, lists the values of the output (that is the excess-3 code) for all possible combinations of the binary code. Note that, these codes are codes for decimal digits 0-9. In other words, even though the 4 bits of the input can represent up to 16 different combinations, ONLY 10 combinations are used to represent the 10 decimal digits.

Thus, a total of 6 input combinations are not likely to occur. Since these inputs will never occur, we use Don't cares for the corresponding output codes.

	BCD input				Ex-3 output			
Decimal #	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	1	0	1	0	0	1
6	0	1	1	1	1	0	1	0
7	1	0	0	0	1	0	1	1
8	1	0	0	1	1	1	0	0
9	0	1	1	0	1	0	0	1
10 - 15	All other inputs				X	X	X	X

Follow implementation procedure

As the procedure for simplification of a Boolean function suggests, we will minimize the four output functions using K-maps. Thus we will be having four K-maps, one for each output function. Each of these K-maps are given below

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

$$W = A + BC + BD$$

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

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

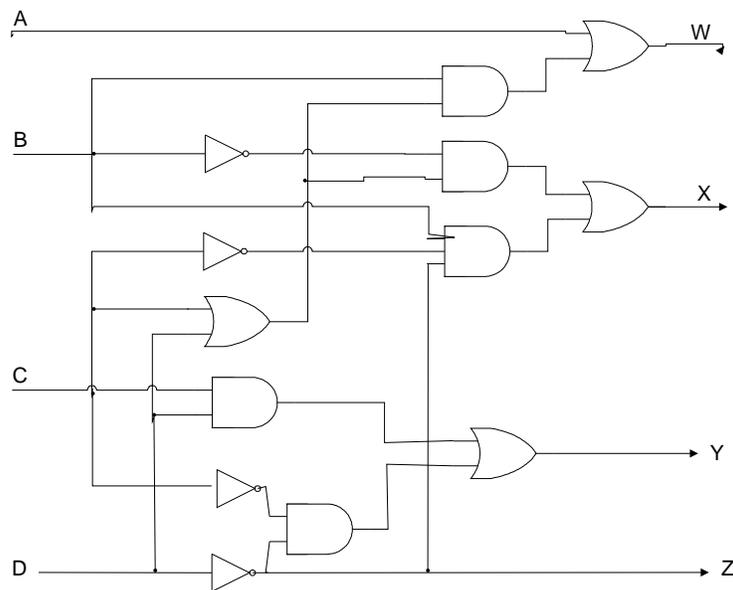
AB \ CD		CD			
		00	01	11	10
AB	00	1	0	1	0
	01	1	0	1	0
	11	X	X	X	X
	10	1	0	X	X

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

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

$$Z = D'$$

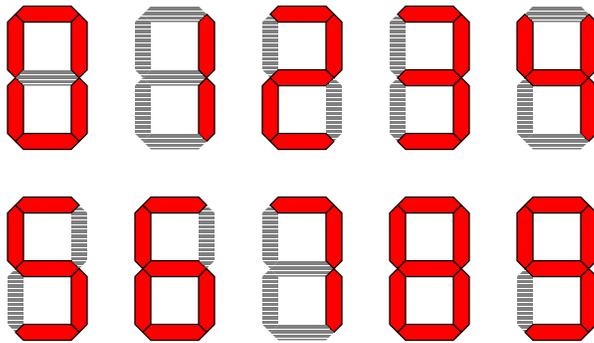
The Circuit



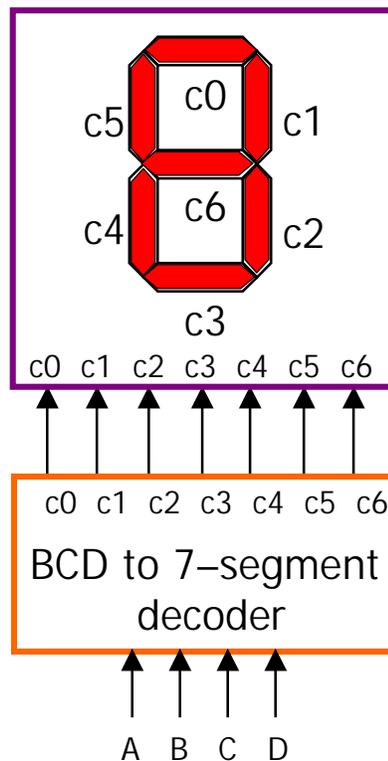
Example: BCD to 7-segment display controller

Let's take another example. We will design a BCD to seven-segment decoder. Before proceeding, let's first understand what a 7-segment display is?

You might have noticed a digital watch, where the digits from 0 to 9 are displayed. These digits can be displayed using seven Light emitting diode segments (or LED's) arranged to look like digit 8 as shown in figure. By controlling which segment is ON and which is OFF we can display illuminated patterns that correspond to the 10 decimal digits 0 to 9. For example, digit 8 can be displayed by illuminating all the segments.



The objective is to design a circuit that will take a BCD number as input, and produces the control signals C0 to C6 which allow illuminating the corresponding segments in the 7-segment display.



Thus, the input is a 4-bit BCD digit A,B,C, and D; A being the most significant while D being the least significant.

The seven segments, which are actually seven output signals, are numbered C0 to C6 that control the illumination of the 7-segment display.

Each of the segment is a Light-Emitting Diode (LED) which is illuminated if current passes through it or dimmed if no current passes through it. For example, digit zero can be displayed by illuminating all the segments except segment C6. Digit 1 can be displayed by ONLY illuminating segments C1 and C2.

Having defined the format of inputs and outputs, let us find out the truth table for this circuit. In the truth table, each input BCD code and its corresponding 7-segment output is shown. The truth table assumes that a logic-1 illuminates a segment while a logic-0 turns the segment off.

Decimal #	BCD input				Outputs for 7-segments						
	A	B	C	D	C0	C1	C2	C3	C4	C5	C6
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10-15	All other inputs				0	0	0	0	0	0	0

Even though the 4 bits of the input can represent up to 16 different combinations, ONLY 10 input combinations representing the 10 decimal digits are considered Valid.

We will design the controller such that the Invalid Input combinations would turn-off all segments. Thus all 7 segments are turned off for input codes beyond 1001.

Now, we are ready to build the seven K-maps, one for each output segment.

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

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

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

$$C2 = A' B + B' C' + A' C' + A' D$$

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

$$C3 = A' C D' + A' B' C + B' C' D' + A B' C' + A' B C' D$$

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

$$C4 = A' C D' + B' C' D'$$

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

$$C5 = A' B C' + A' C' D' + A' B D' + A B' C'$$

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

$$C6 = A' C D' + A' B' C + A' B C' + A B' C'$$