Lecture 13 – Combinational logic circuits 2

Dr. Aftab M. Hussain,

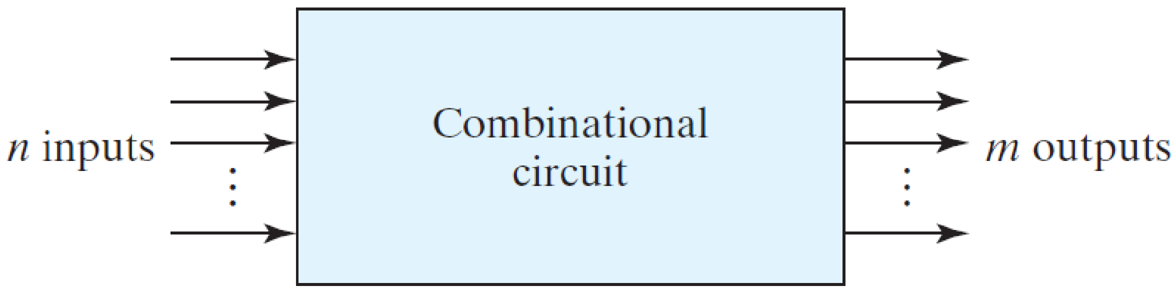
Assistant Professor, PATRIoT Lab, CVEST

Combinational circuits

- Logic circuits for digital systems may be combinational or sequential
- A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs
- A combinational circuit performs an operation that can be specified logically by a set of Boolean functions
- In contrast, sequential circuits employ storage elements in addition to logic gates
- Their outputs are a function of the inputs and the state of the storage elements
- Because the state of the storage elements is a function of previous inputs, the outputs of a sequential circuit depend not only on present values of inputs, but also on past inputs, and the circuit behavior must be specified by a time sequence of inputs and internal states

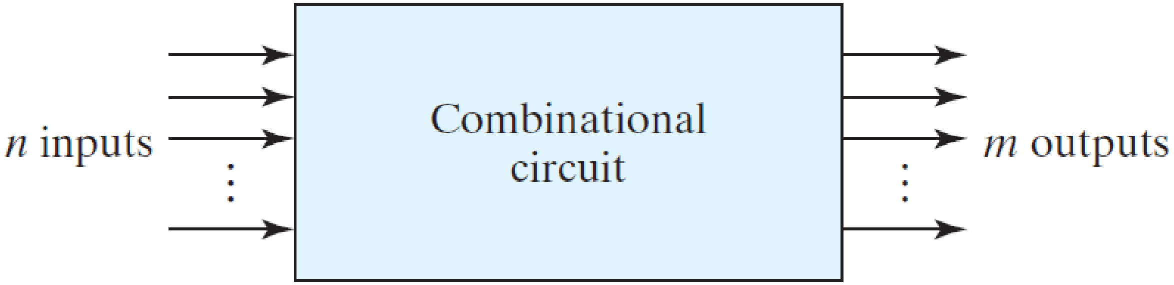
Combinational circuits

- A combinational circuit consists of an interconnection of logic gates
- Combinational logic gates react to the values of the signals at their inputs and produce the value of the output signal, transforming binary information from the given input data to a required output data
- Each input and output variable exists physically as an analog signal whose values are interpreted to be a binary signal that represents logic 1 and logic 0



Combinational circuits

- For *n* input variables, there are 2^n possible combinations of the binary inputs
- For each possible input combination, there is one possible value for each output variable
- Thus, a combinational circuit can be specified with a truth table that lists the output values for each combination of input variables
- A combinational circuit also can be described by *m* Boolean functions, one for each output variable
- Each output function is expressed in terms of the *n* input variables



Circuit design

- 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:

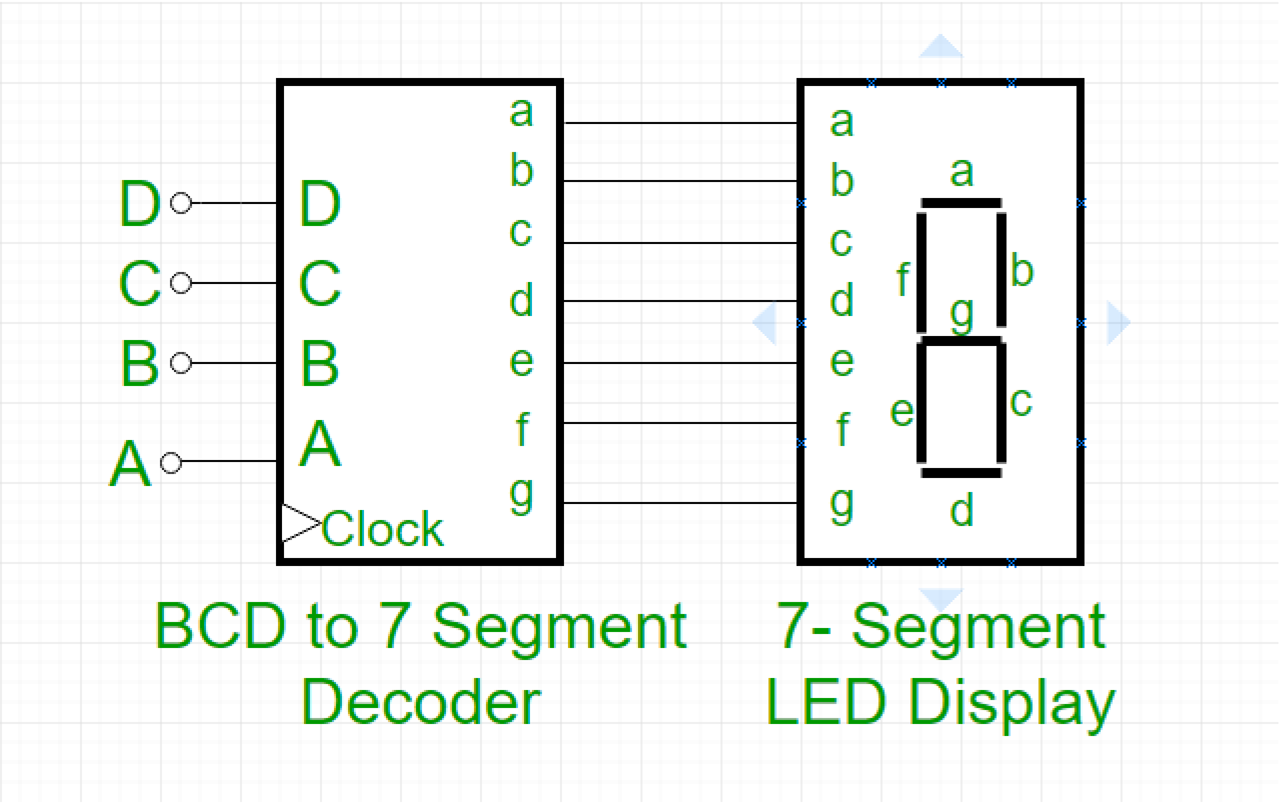
1. From the specifications of the circuit, determine the required number of inputs and outputs and assign a symbol to each
2. Derive the truth table that defines the required relationship between inputs and outputs
3. Make the K-map, if necessary
4. Obtain the simplified Boolean functions for each output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

Here. We. Go.

The 7-Segment Decoder

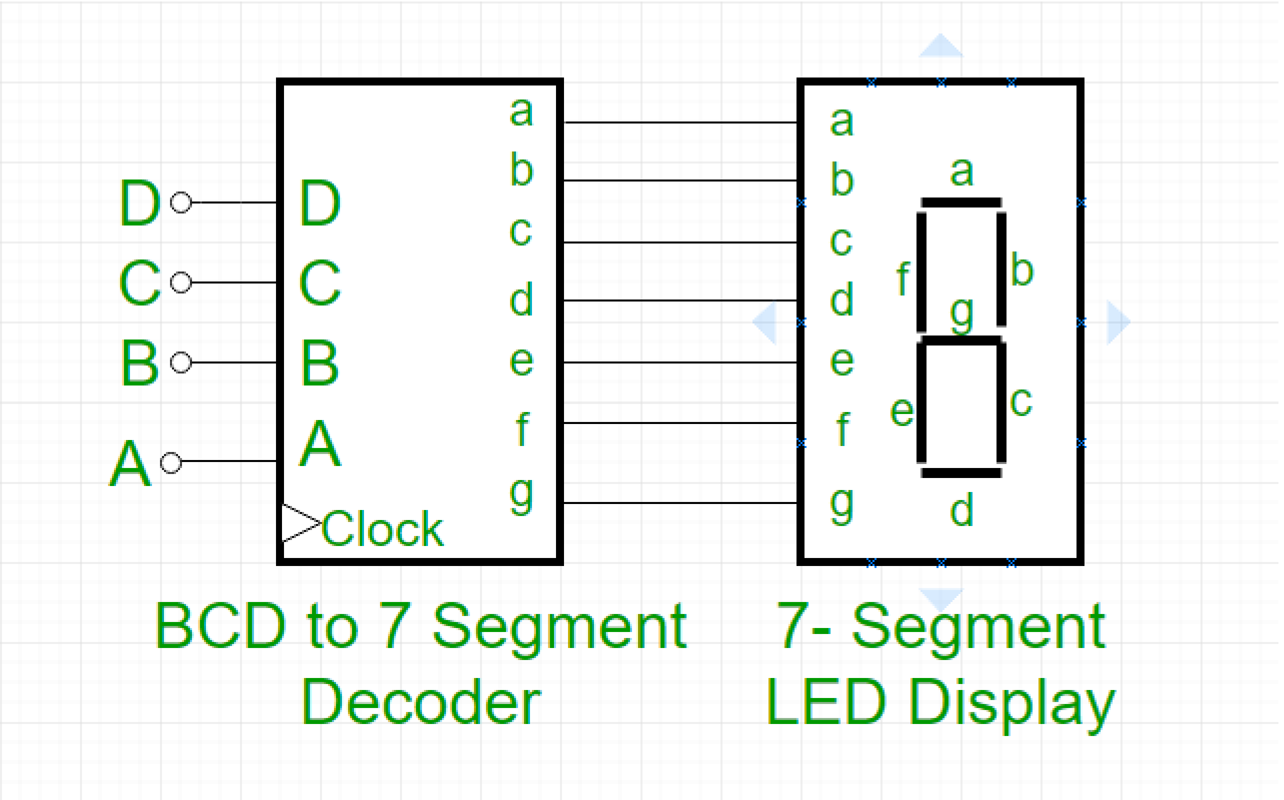
7-segment decoder

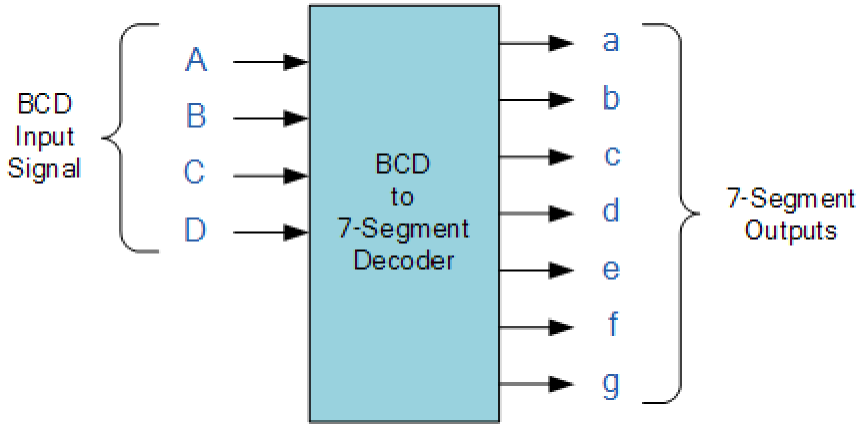
- We discussed many ways of representing symbols using binary variables – signed magnitude, 2's complement, BCD, ASCII etc.
- Going from one representation may be considered as an encoding/decoding operation
- Consider the practical problem of displaying binary numbers using a 7-segment LED display
- For this, we need to "decode" the binary information into the display inputs



7-segment decoder

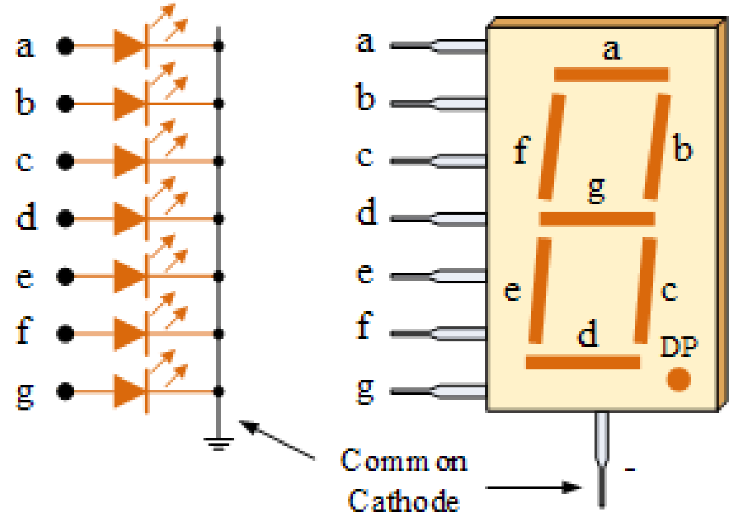
- First, we realize that there are 4 inputs and 7 outputs!
- Thus, the truth-table will be a 16 row table with 7 different columns for 7 outputs
- Hence, there will be 7 different functions we will be implementing to make this decoder
- The other thing we need to realize is whether a particular output should be HIGH or LOW for the LED to glow?

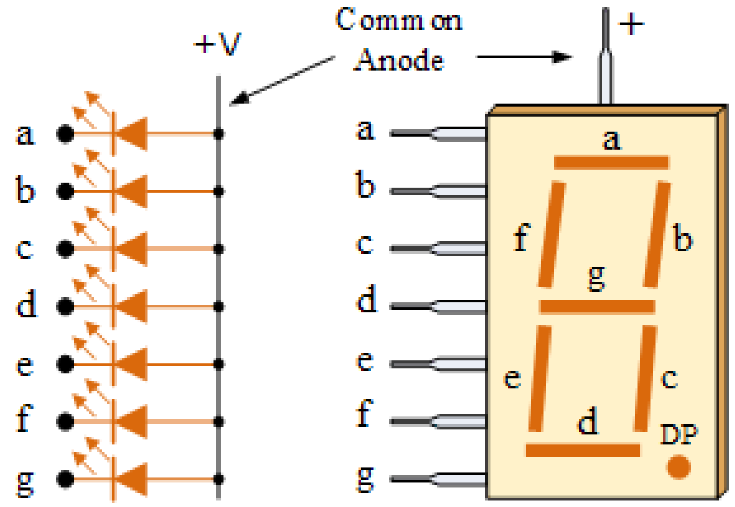




7-segment decoder

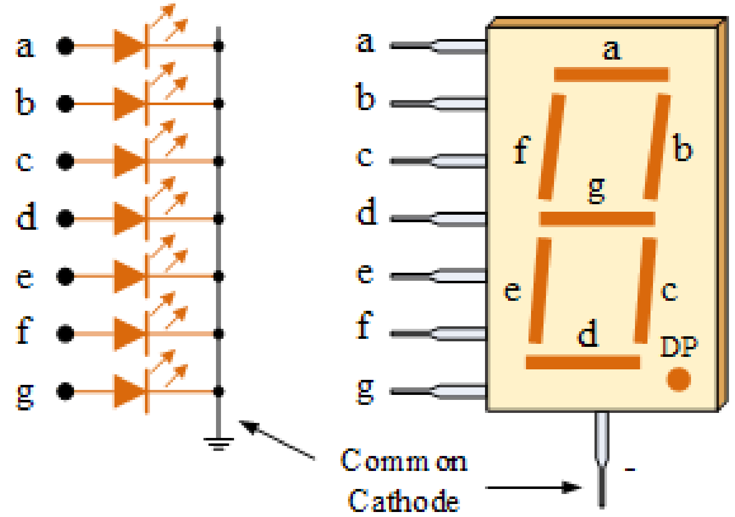
- The other thing we need to realize is whether a particular output should be HIGH or LOW for the LED to glow?
- To know the answer to this, we see that there are two types of 7-segment LED displays – common anode and common cathode
- The common cathode connects all LED cathodes to a common ground, thus, when input is high LED glows
- Conversely, the common anode connects all LED anodes to +V$_{CC}$, thus, when input is low, LED glows

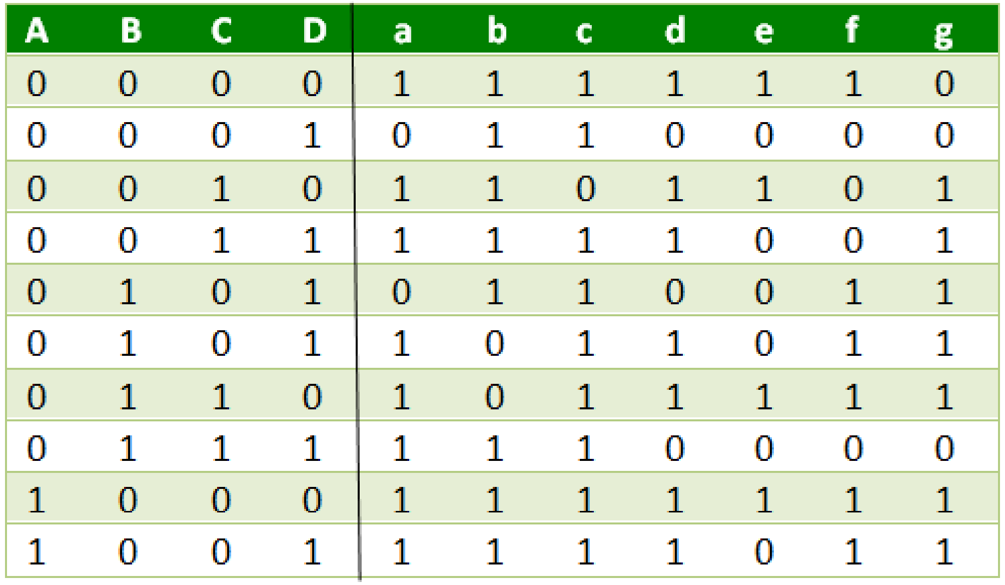




7-segment decoder

- Let us choose common cathode LED display to make the function
- Thus, we can make the truth-table
- Obviously, the BCD system only goes from 0 to 9, while we have 16 rows for 4 inputs
- What about the other rows?
- In the other rows, we fill all the outputs as don’t care, because we are sure that these are not going to be input anyway (trust the engineer before you)
- Thus, we have six don’t care conditions

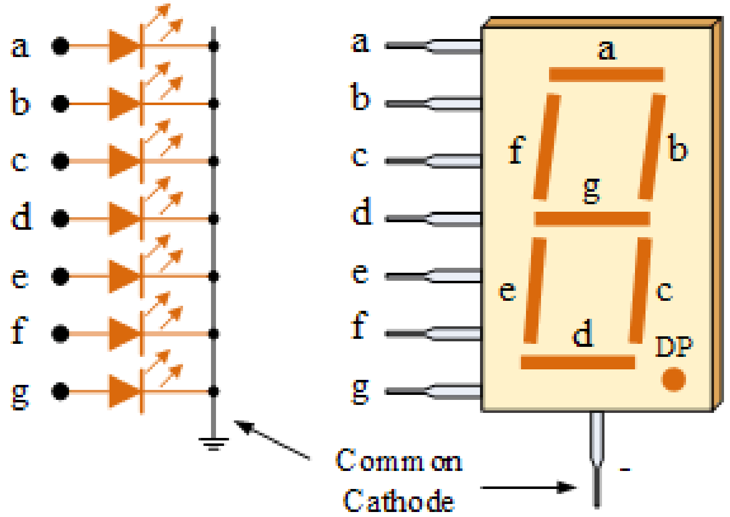




A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	1	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

7-segment decoder

- Let us choose common cathode LED display to make the function
- Thus, we can make the truth-table
- Obviously, the BCD system only goes from 0 to 9, while we have 16 rows for 4 inputs
- In the other rows, we fill all the outputs as don’t care, because we are sure that these are not going to be input anyway (trust the engineer before you)
- Thus, we have six don’t care conditions



Common Cathode

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	1	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

7-segment decoder

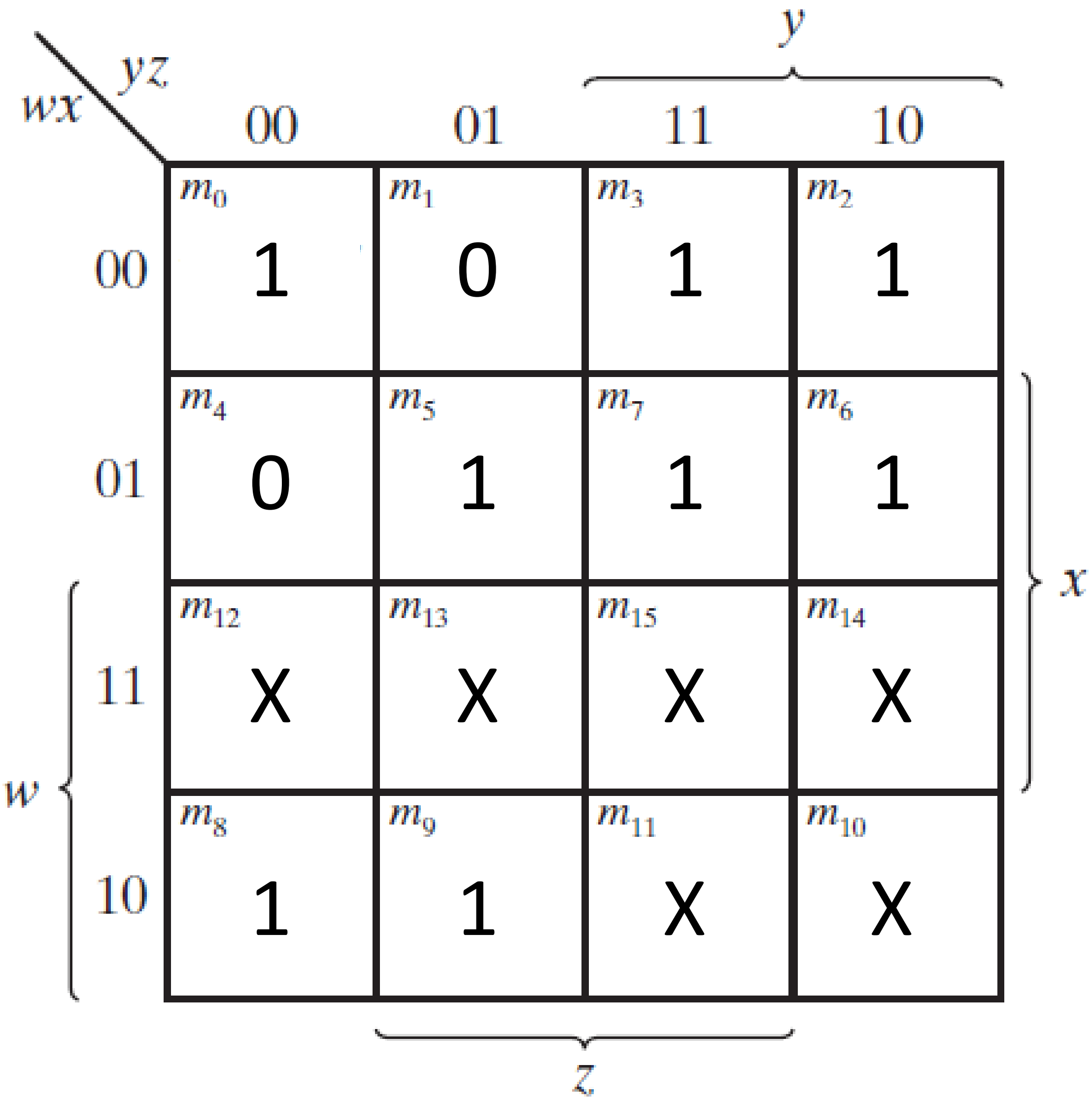
- Thus, the K-map for output a will look like this
- We have two clusters of 8: y and w
- Two clusters of four: xz and $x'z'$
- Thus, the logic function for a can be:

$$F_a = y + w + xz + x'z'$$

- Or we can have PoS as:
- One cluster of two: $xy'z'$
- One min-term: $w'x'y'z$
- Thus,

$$F_a = (x' + y + z)(w + x + y + z')$$

- This can be done for all the other outputs

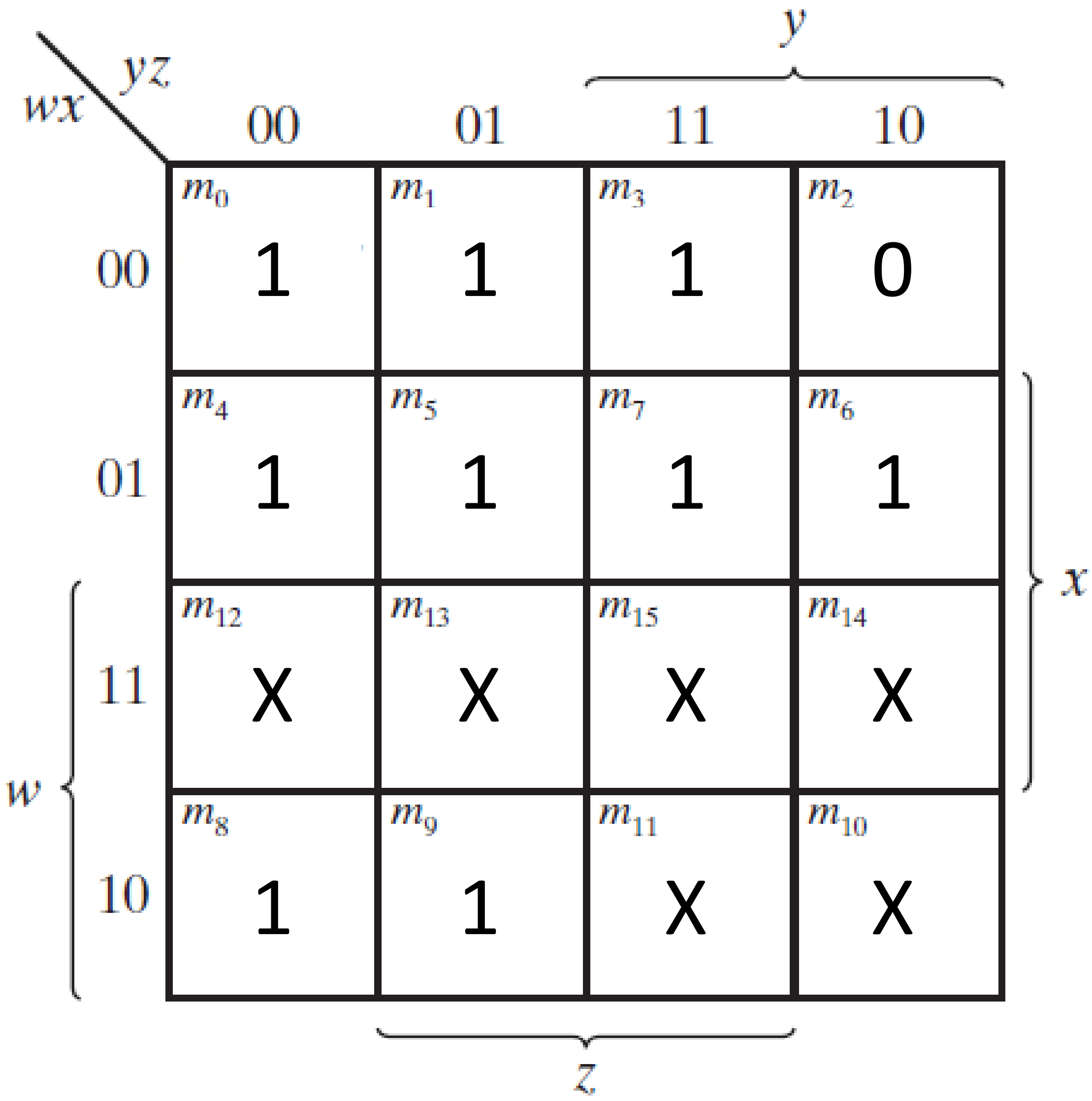


$wx \backslash yz$	00	01	11	10
00	m_0 1	m_1 0	m_3 1	m_2 1
01	m_4 0	m_5 1	m_7 1	m_6 1
11	m_{12} X	m_{13} X	m_{15} X	m_{14} X
10	m_8 1	m_9 1	m_{11} X	m_{10} X

7-segment decoder

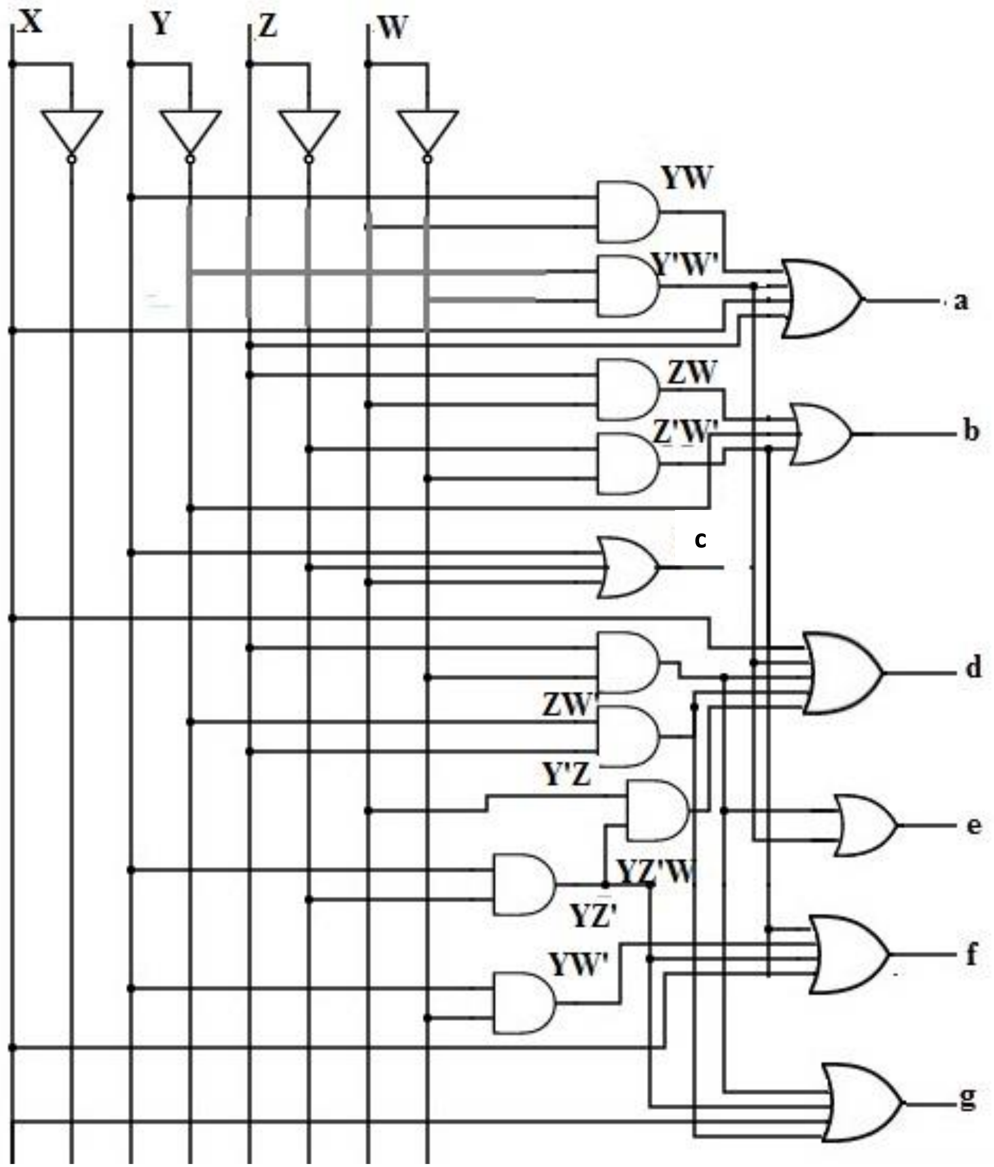
- Thus, the K-map for output c will look like this
- We have only one zero, so we go the PoS route
- The max term cluster of two is represented by yz’x’
- In PoS form:

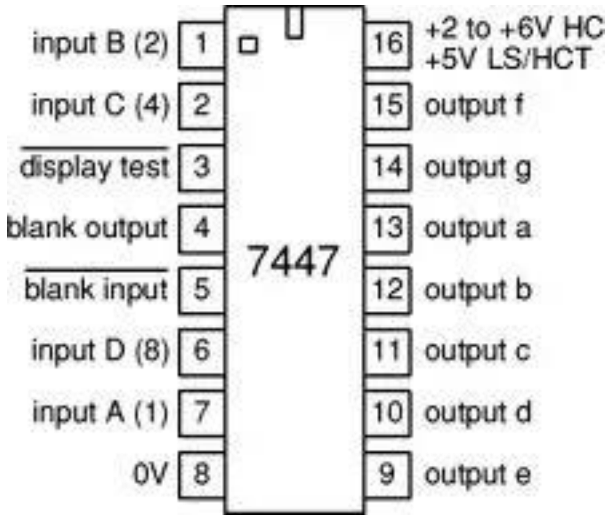
$$c = y' + z + x$$



$wx \backslash yz$	00	01	11	10
00	m_0 1	m_1 1	m_3 1	m_2 0
01	m_4 1	m_5 1	m_7 1	m_6 1
11	m_{12} X	m_{13} X	m_{15} X	m_{14} X
10	m_8 1	m_9 1	m_{11} X	m_{10} X

7-segment decoder

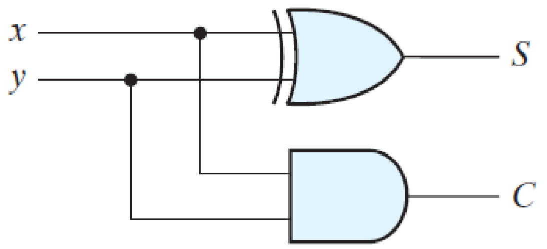


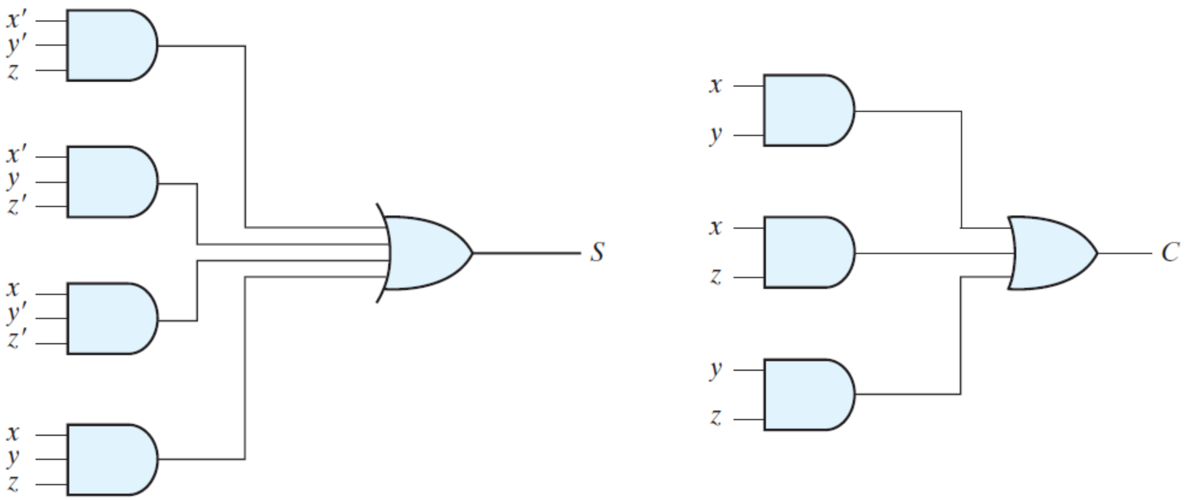


The Binary Adder

Binary adder

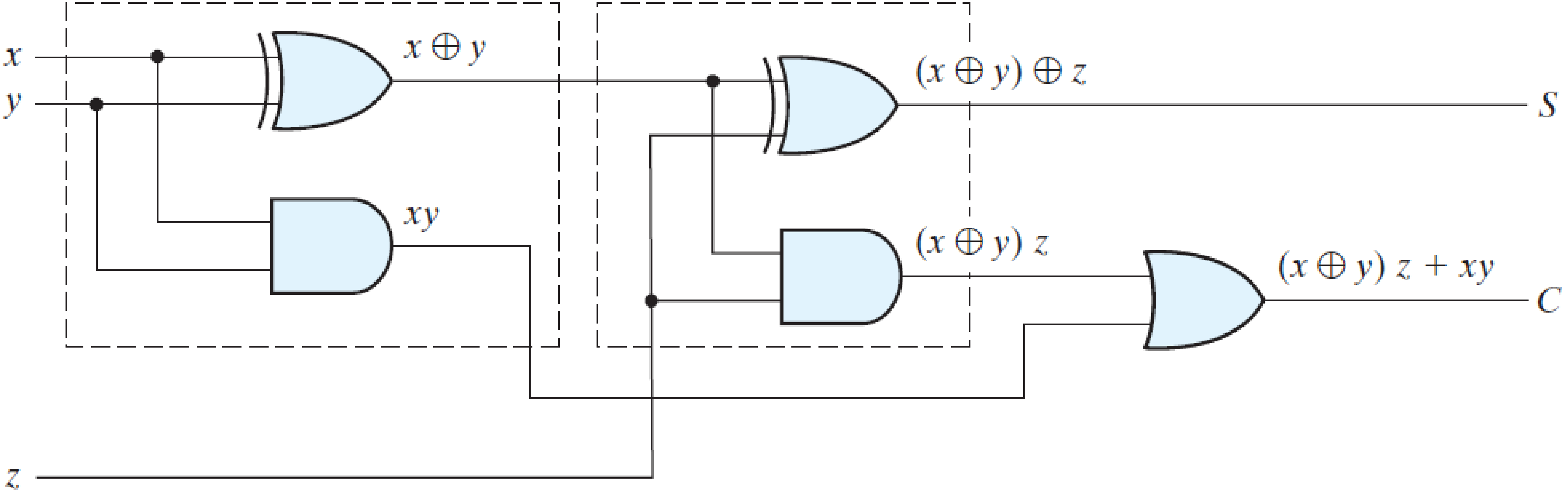
- Digital computers perform a variety of information-processing tasks
- Among the functions encountered are the various arithmetic operations
- The most basic arithmetic operation is the addition of two binary digits
- A combinational circuit that performs the addition of two bits is called a *half adder*.
- One that performs the addition of three bits (two significant bits and a previous carry) is a *full adder*





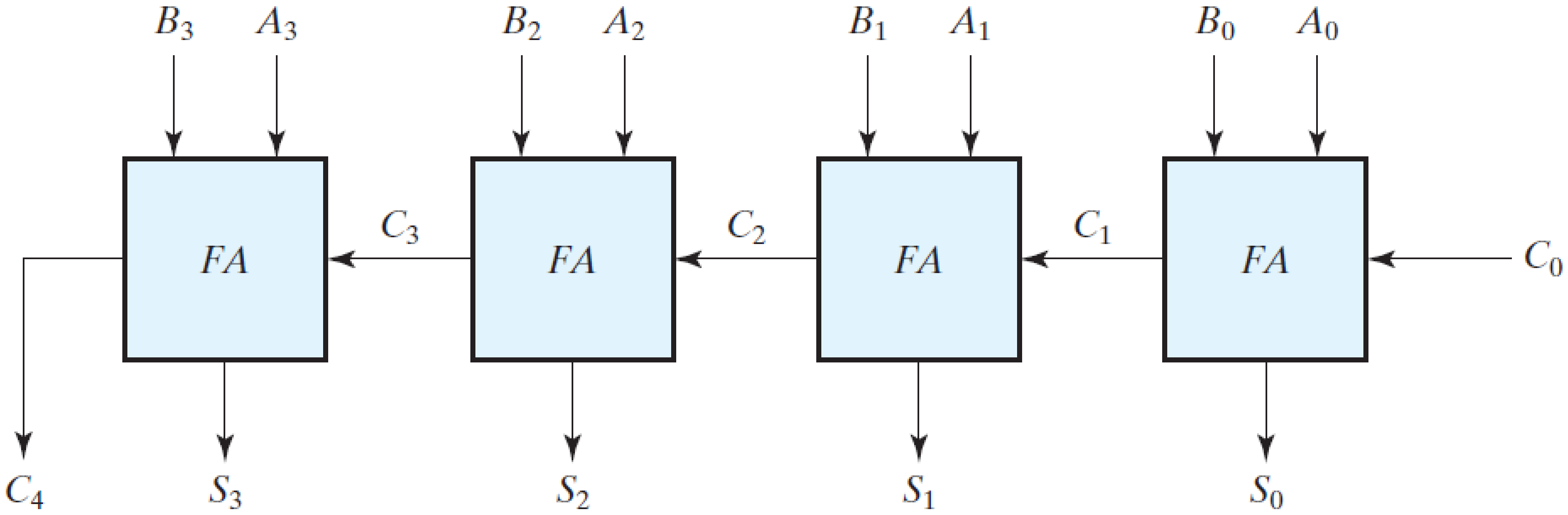
Binary adder

- We can use two half adders to create a full adder



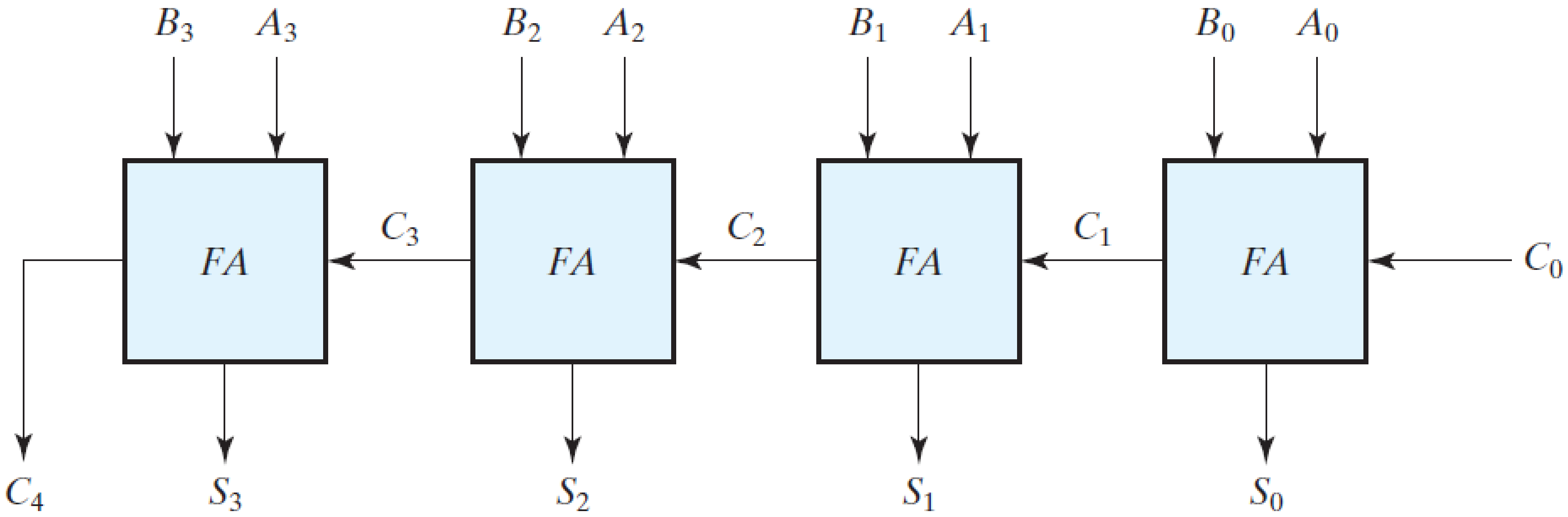
n-bit binary adder

- Addition of *n*-bit numbers requires a chain of *n* full adders or a chain of one-half adder and *n*-1 full adders
- Consider a four bit adder. The augend bits of *A* and the addend bits of *B* are designated by subscript numbers from right to left, with subscript 0 denoting the least significant bit
- The carries are connected in a chain through the full adders. The input carry to the adder is C_0, and it ripples through the full adders to the output carry C_4.
- The *S* outputs generate the required sum bits



n-bit binary adder

- Can we make this circuit through the normal route?
- Note that the classical method would require a truth table (and K-map) with 2^9 = 512 entries, since there are nine inputs to the circuit
- By using an iterative method of cascading a standard function, it is possible to obtain a simple and straightforward implementation



B_3
A_3
B_2
A_2
B_1
A_1
B_0
A_0
FA
C_3
FA
C_2
FA
C_1
FA
C_0
C_4
S_3
S_2
S_1
S_0