Unit – 4
Combinational Logic

Combinational Logic circuit Implementation:

Combinational Logic Circuit: A combinational circuit is one in which the state of the output at any instant is entirely determined by the states of the inputs at that time. The output occurs immediately after a slight propagation delay once the input is given.

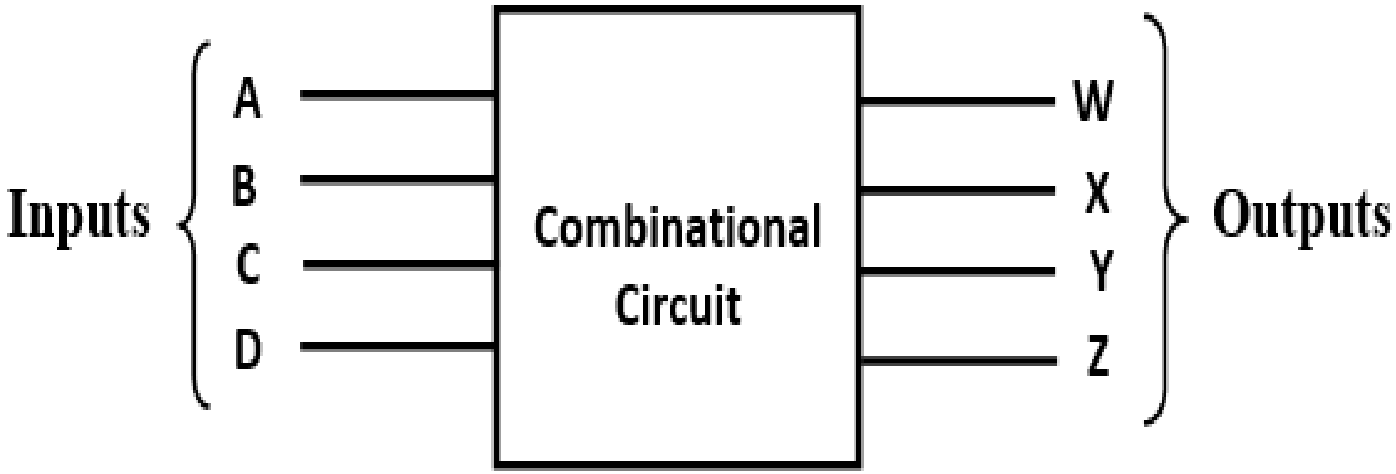
There is no memory in combinational circuits. There will be 2^n combinations of input variable for n inputs. The output will be different for each input combination. Adders, subtractors, decoders, encoders, XOR, XNOR gates, Multiplexer, De-multiplexer etc.

Combinational circuit design procedure:

1) Problem definition. That is problem is stated.
2) Determine the number of input and output variables.
3) Assign each input and output variable with letter symbol.
4) Find the relationship between input and output variables using truth table and logic functions.
5) Simplify the logic function using K-map or Boolean algebra.
6) Draw logic diagram for the simplified logic expression.

Example: Design a combinational circuit with four input lines that represent a decimal digit in BCD and four output lines that generate the 9's compliment of the input digit.

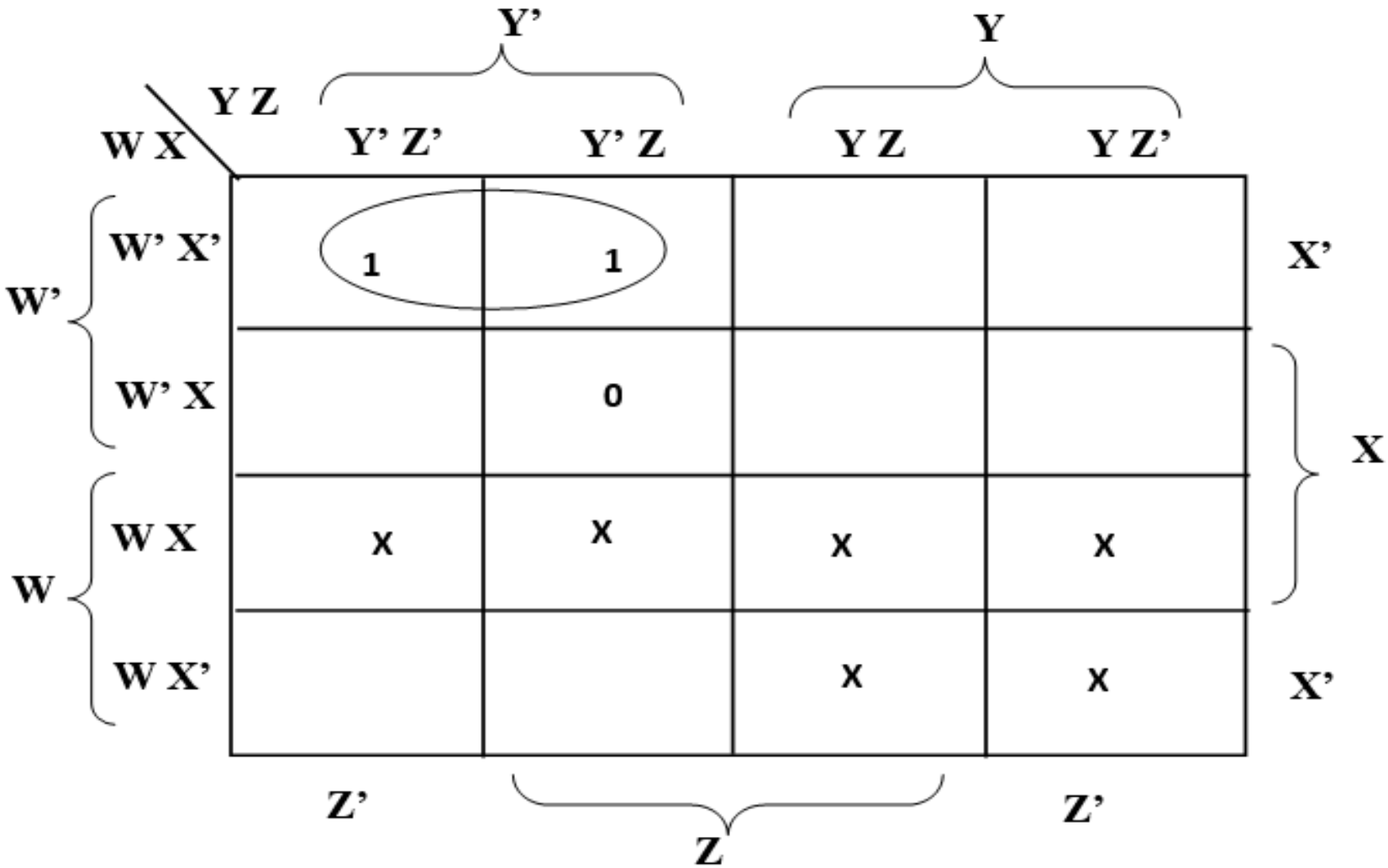
Solution:



Truth table:

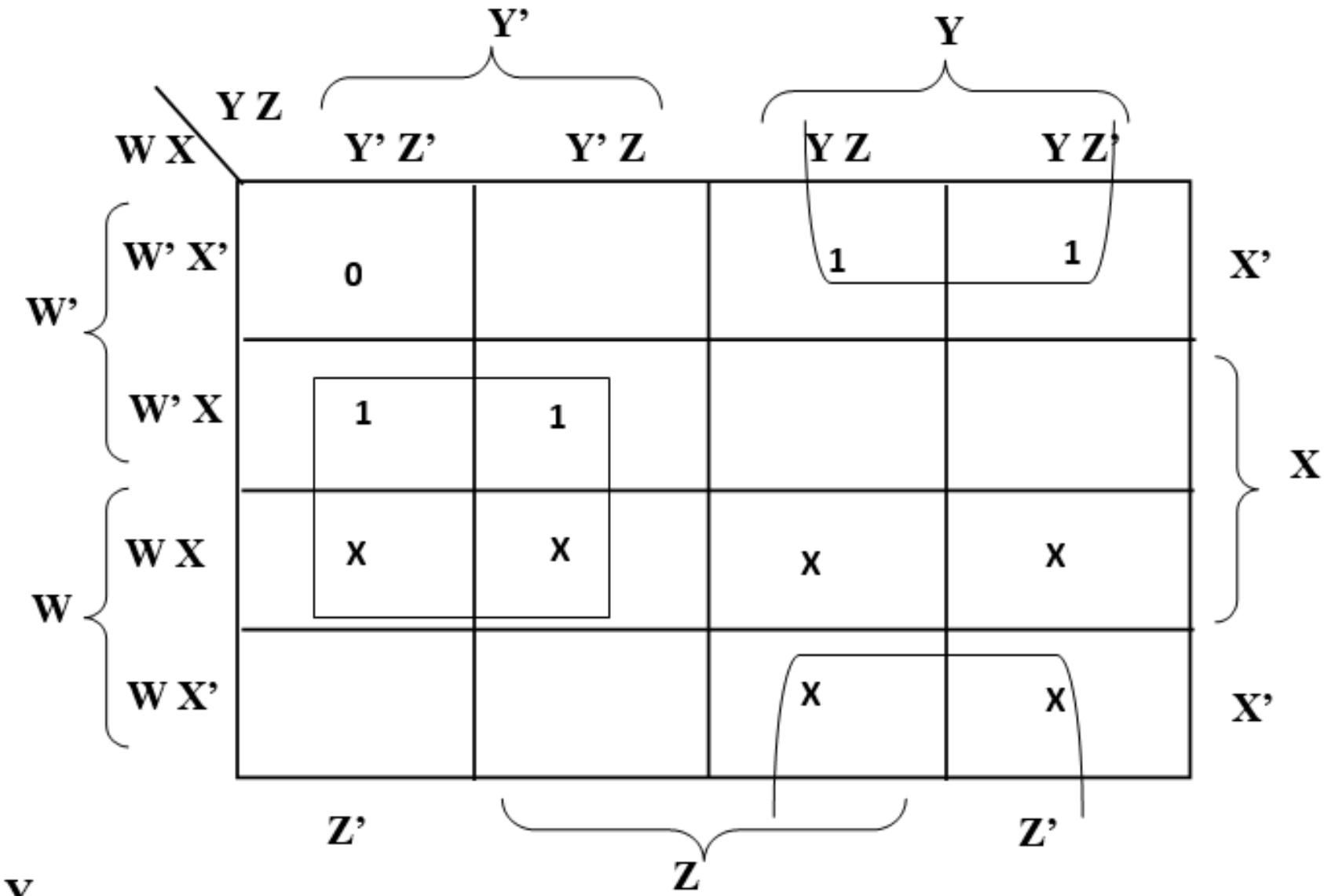
Inputs		Outputs
Decimal	BCD	9's complement
	W X Y Z	A B C D
0	0 0 0 0	1 0 0 1
1	0 0 0 1	1 0 0 0
2	0 0 1 0	0 1 1 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 0 1
5	0 1 0 1	0 1 0 0
6	0 1 1 0	0 0 1 1
7	0 1 1 1	0 0 1 0
8	1 0 0 0	0 0 0 1
9	1 0 0 1	0 0 0 0

K-Map for A:



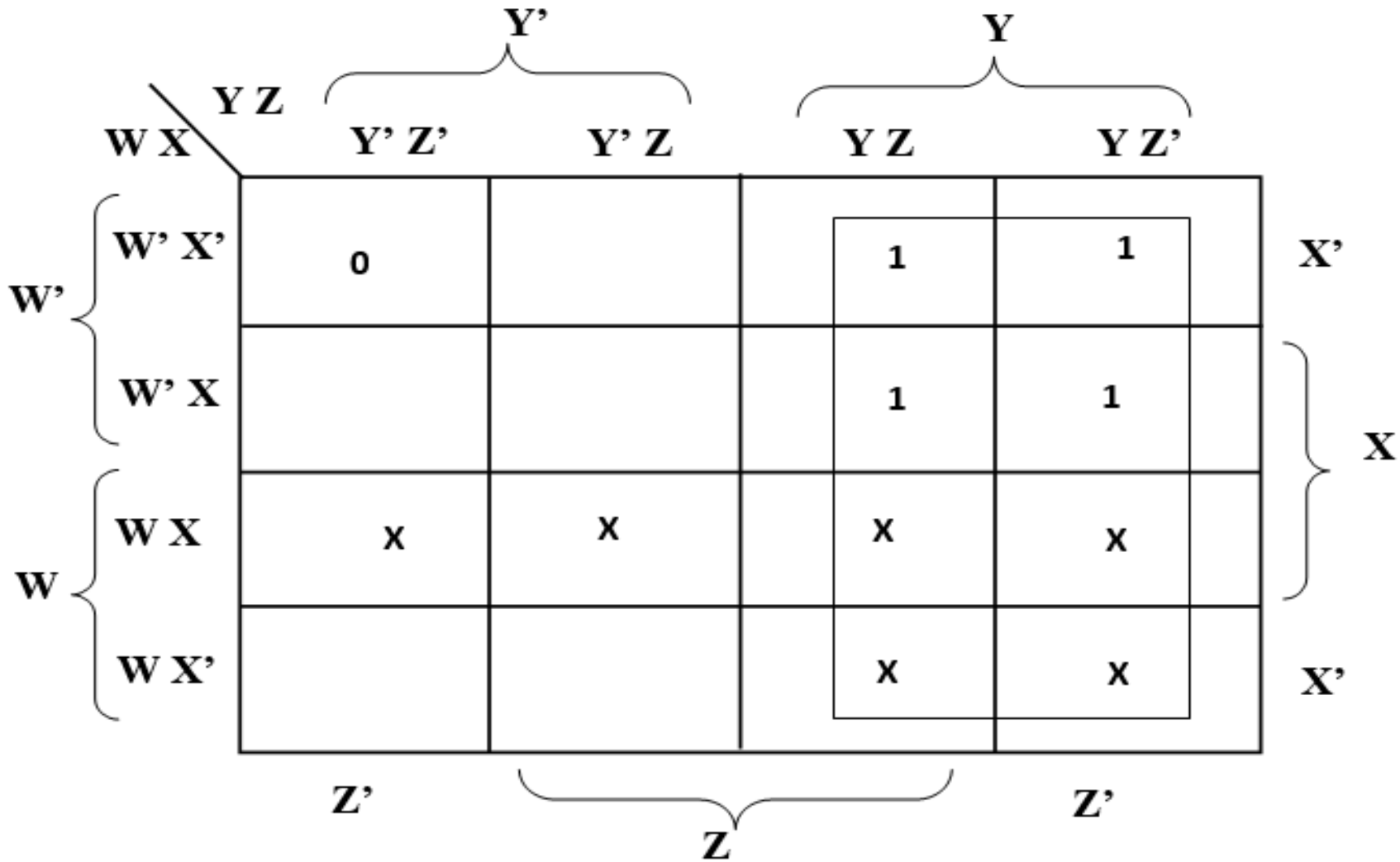
A = W' X' Y'

K-Map for B:



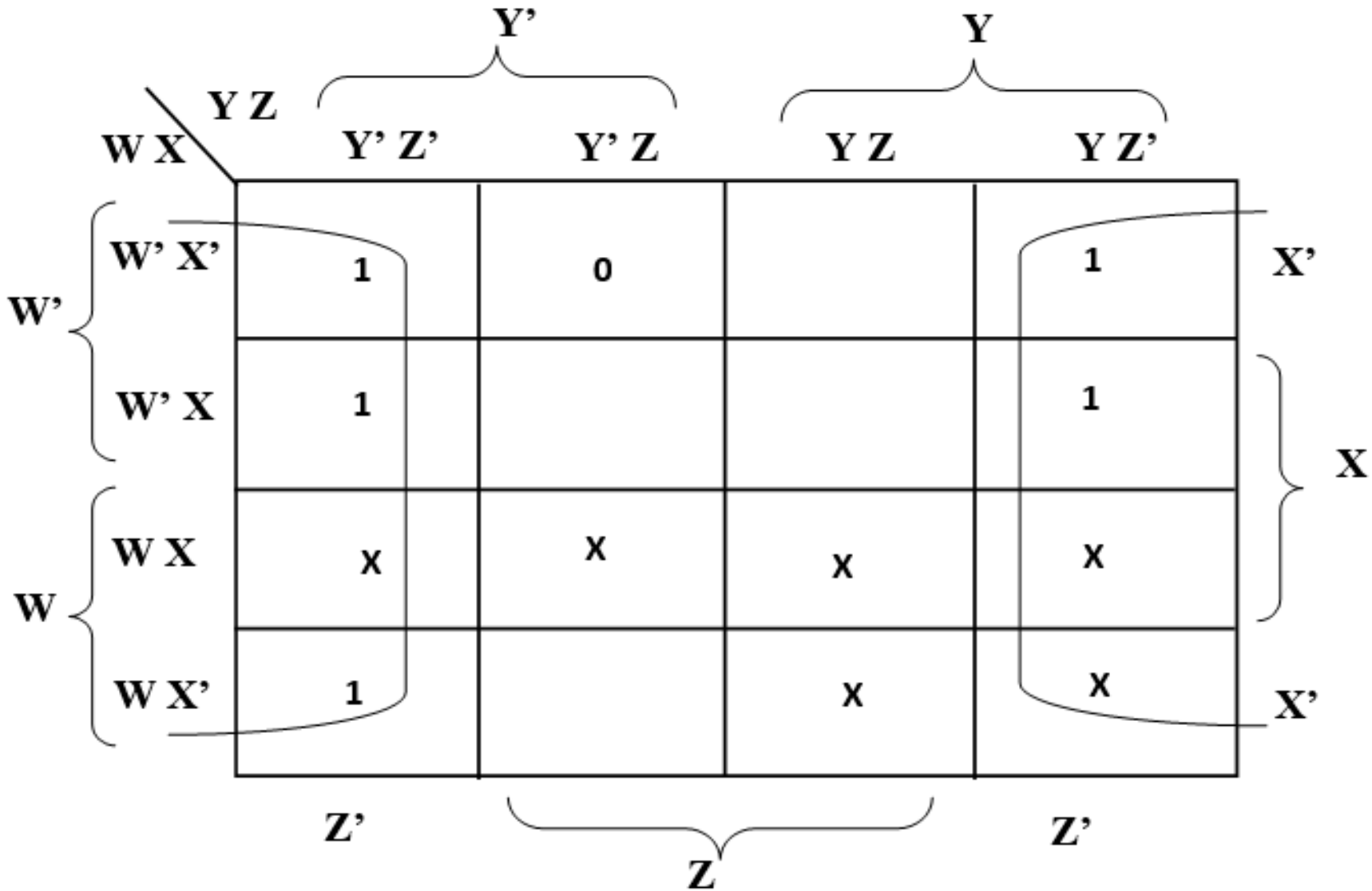
B = X Y' + X' Y

K-Map for C:



C = Y

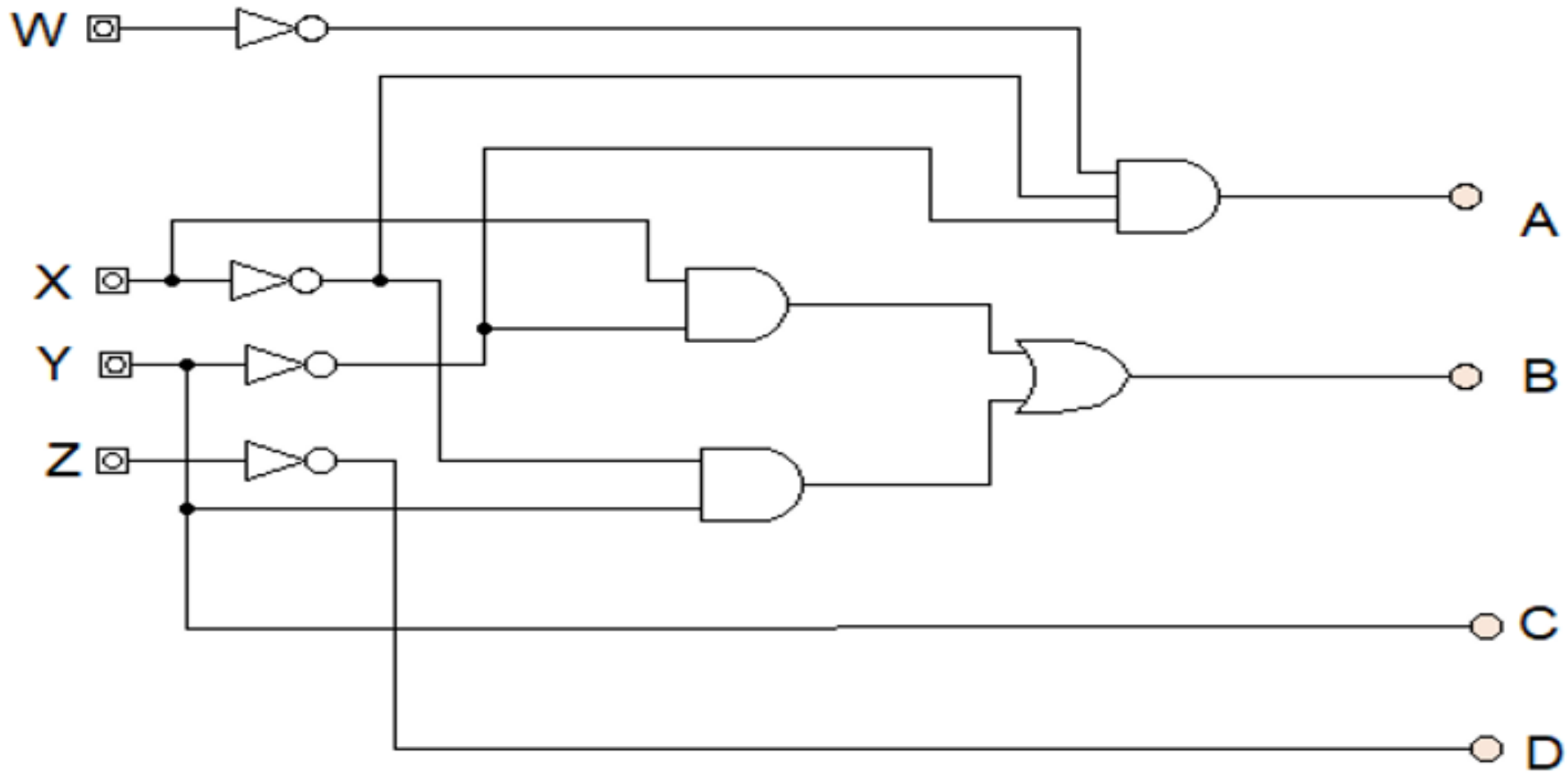
K-Map for D:



D = Z'

Logic diagram implementation:

$A = W' X' Y' \quad B = X Y' + X'Y \quad C = Y \quad D = Z'$



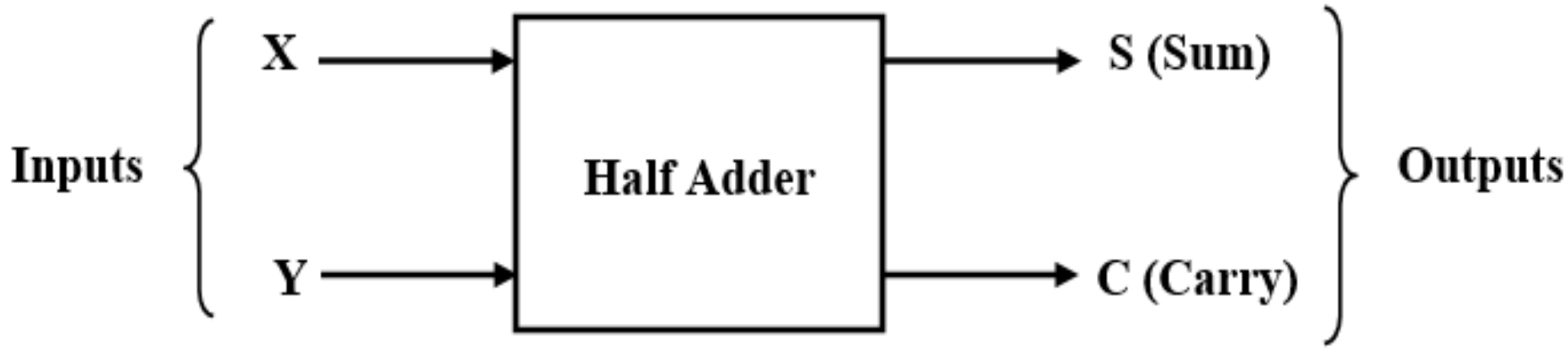
Adders: Adders are the combinational logic circuit, which is used to add two or more than two bits at a time.

Types of adders:

1) **Half Adder**
2) **Full Adder**

Half-Adder: It is a combinational logic circuit, which is used to find the sum of two binary digits at a time. Circuit needs two inputs and two outputs. The input variables designate the augend (x) and addend (y) bits; the output variables produce the sum (S) and carry (C).

Block diagram of Half-Adder:



Now we formulate a Truth table to identify the function of half-adder.

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

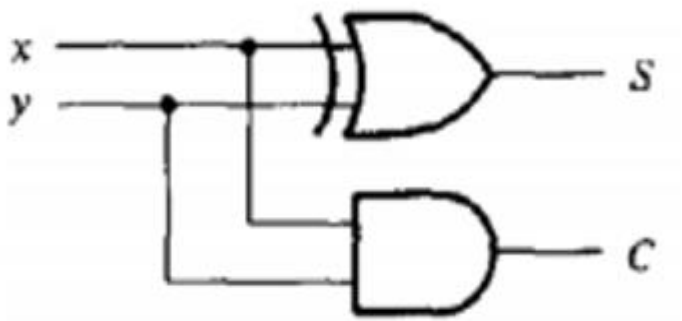
The simplified Boolean functions for the two outputs can be obtained directly from the truth table. The simplified **sum of products** expressions are:

$$S = x'y + xy' \quad \text{or} \quad S = x \oplus y$$

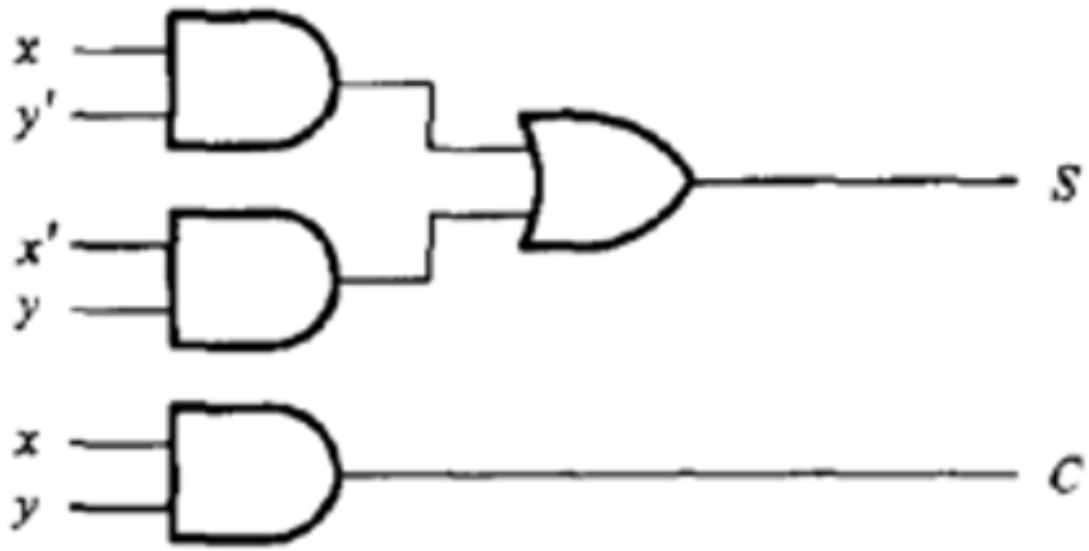
$$C = xy$$

Logic Diagram of Half Adder:

Or



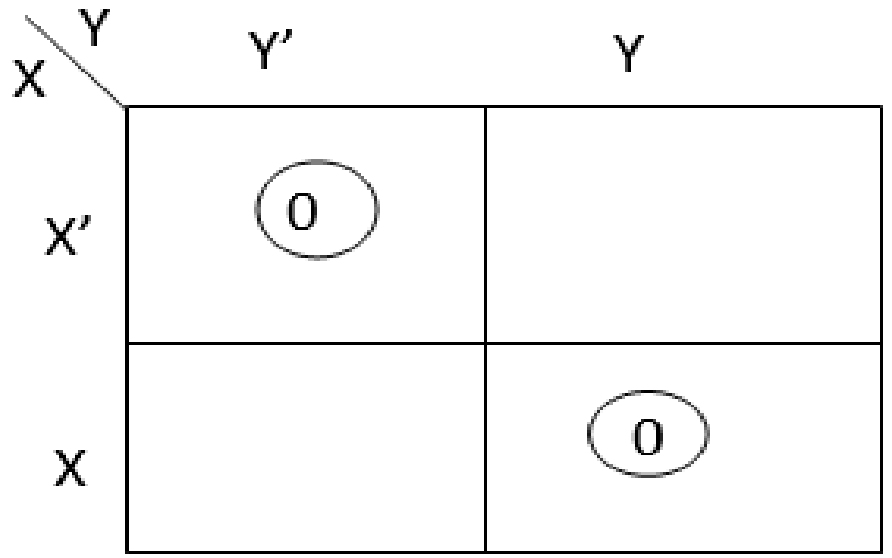
(c) $S = x \oplus y$
$C = xy$



(a) $S = xy' + x'y$
$C = xy$

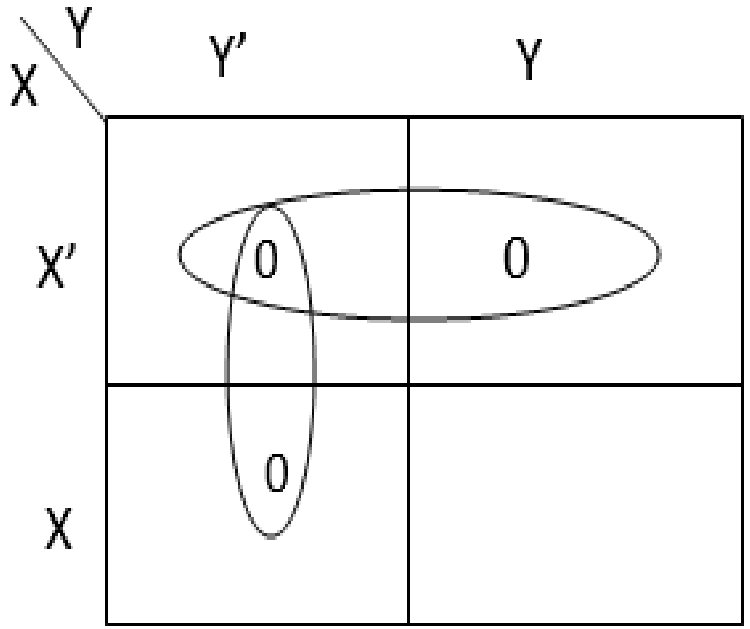
The Sum and Carry can be realized in Product of Sums form.

K-map for simplified expression in POS for Sum:



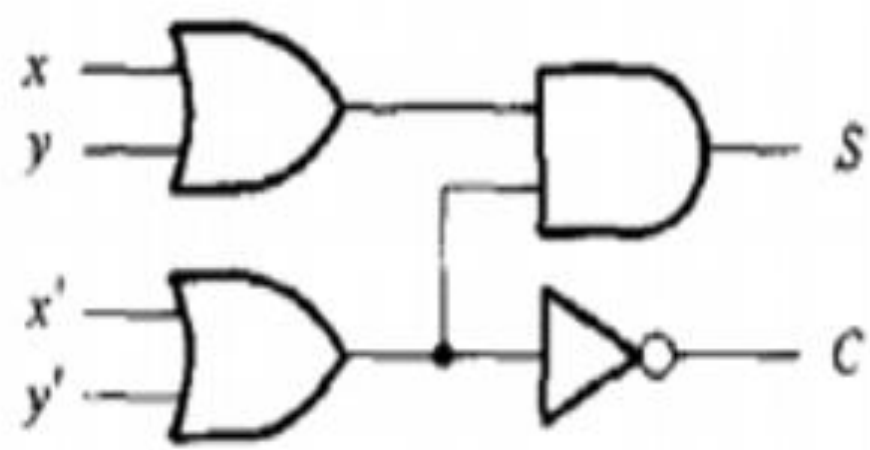
Sum (S) = (x + y) (x' + y')

K-Map for Carry:



Carry (C) = (x' + y')'

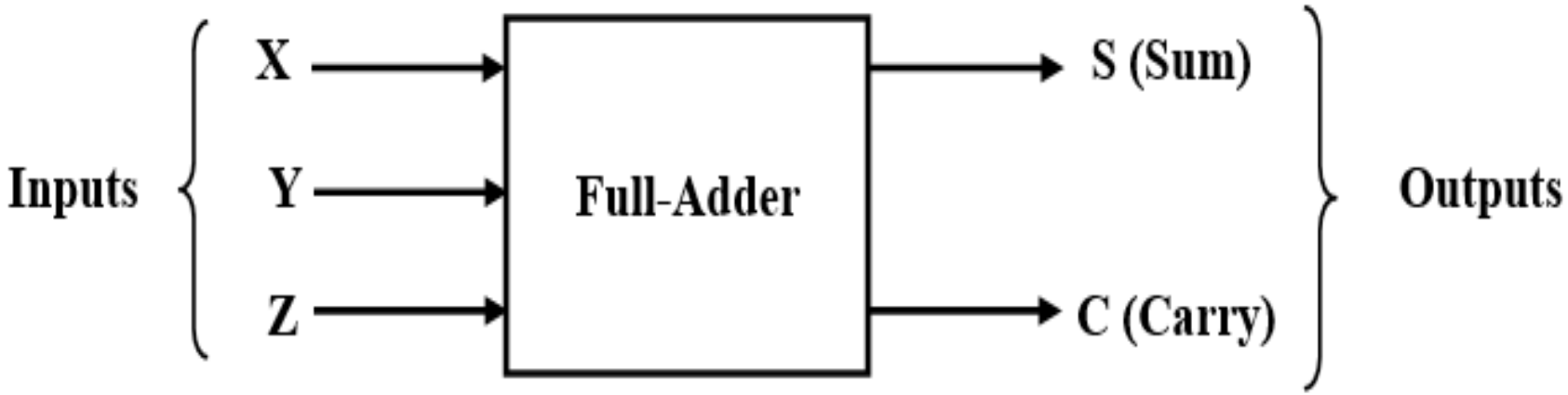
Logic diagram of Half-adder for sum and carry in POS:



$$S = (x + y)(x' + y')$$
$$C = (x' + y')'$$

Full Adder: Full-adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by x and y, represent the two significant bits to be added. The third input, z, represents the carry from the previous lower significant position.

Block diagram of Full-Adder:

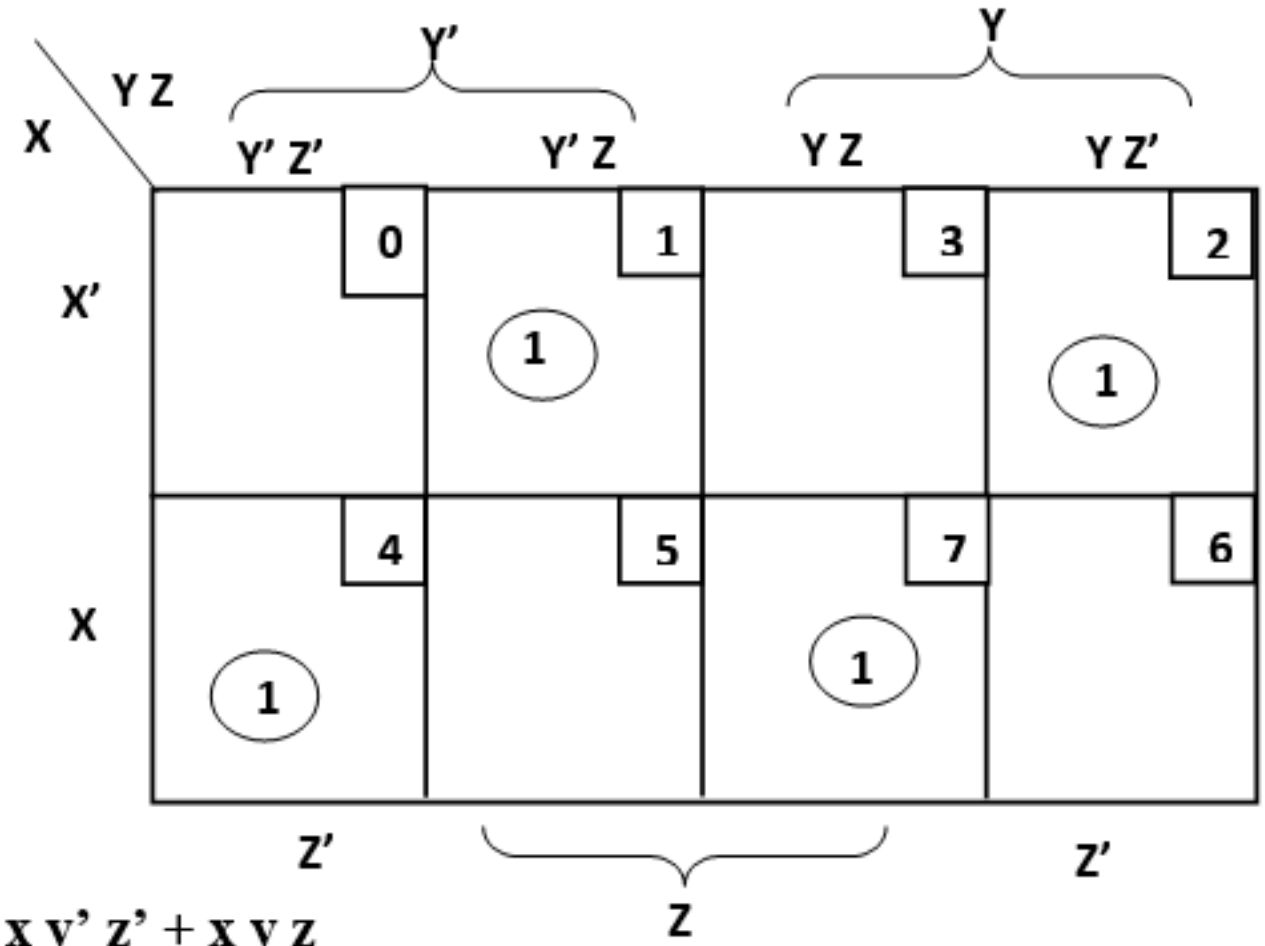


Now we formulate a Truth table to identify the function of Full-adder.

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

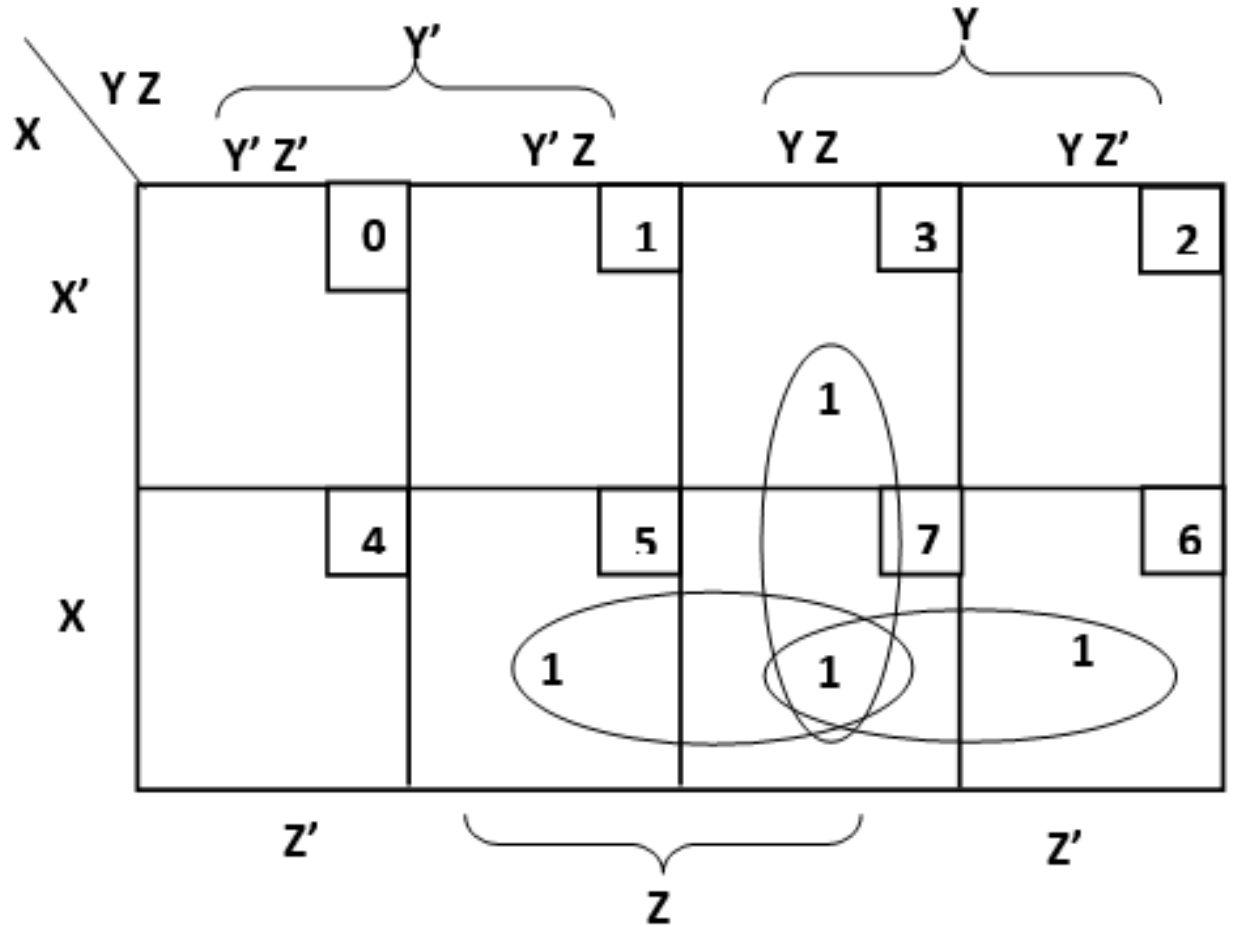
K-Map for simplified expression in SOP for full- adder:

For Sum:



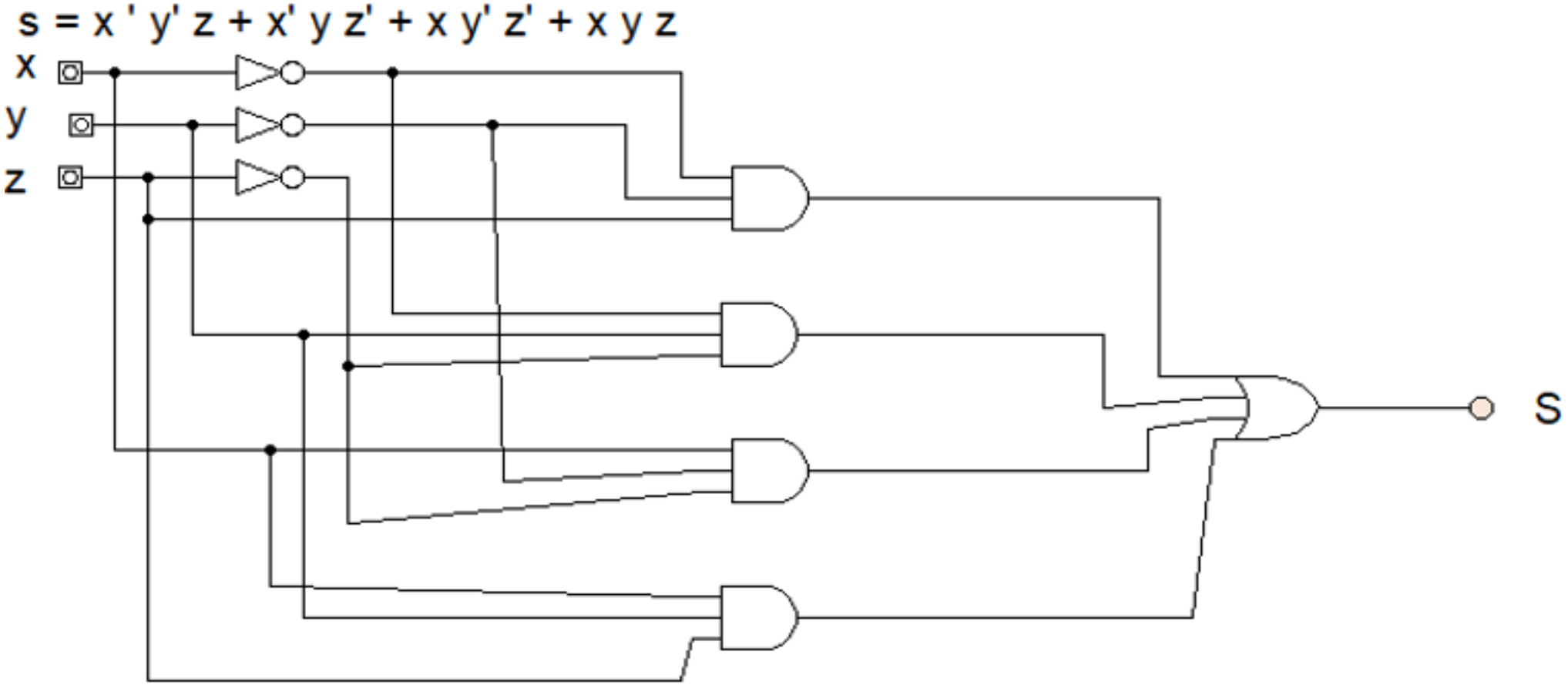
$S = x' y' z + x' y z' + x y' z' + x y z$

For Carry:



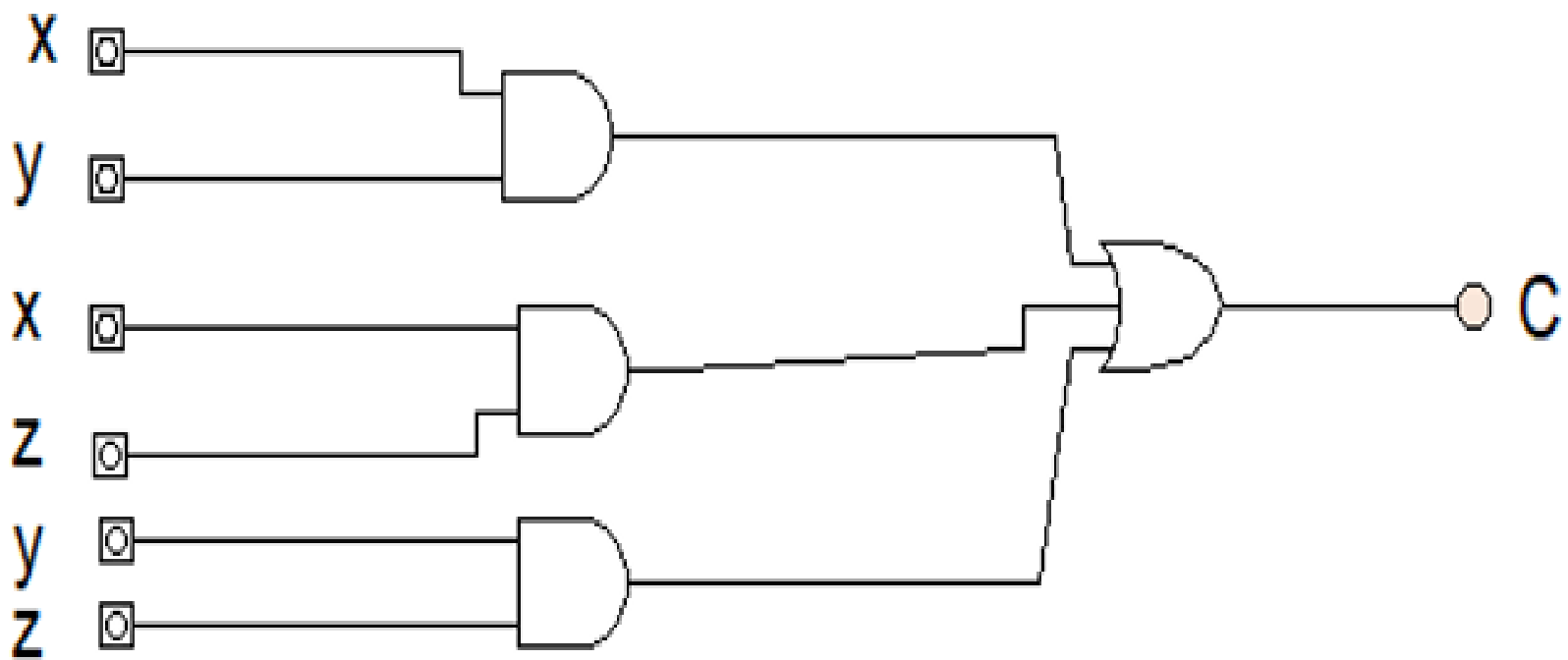
$$C = x\,z + y\,z + x\,y$$

Logic Diagram implementation for Full-Adder:



$$C = x\,y + x\,z + y\,z$$

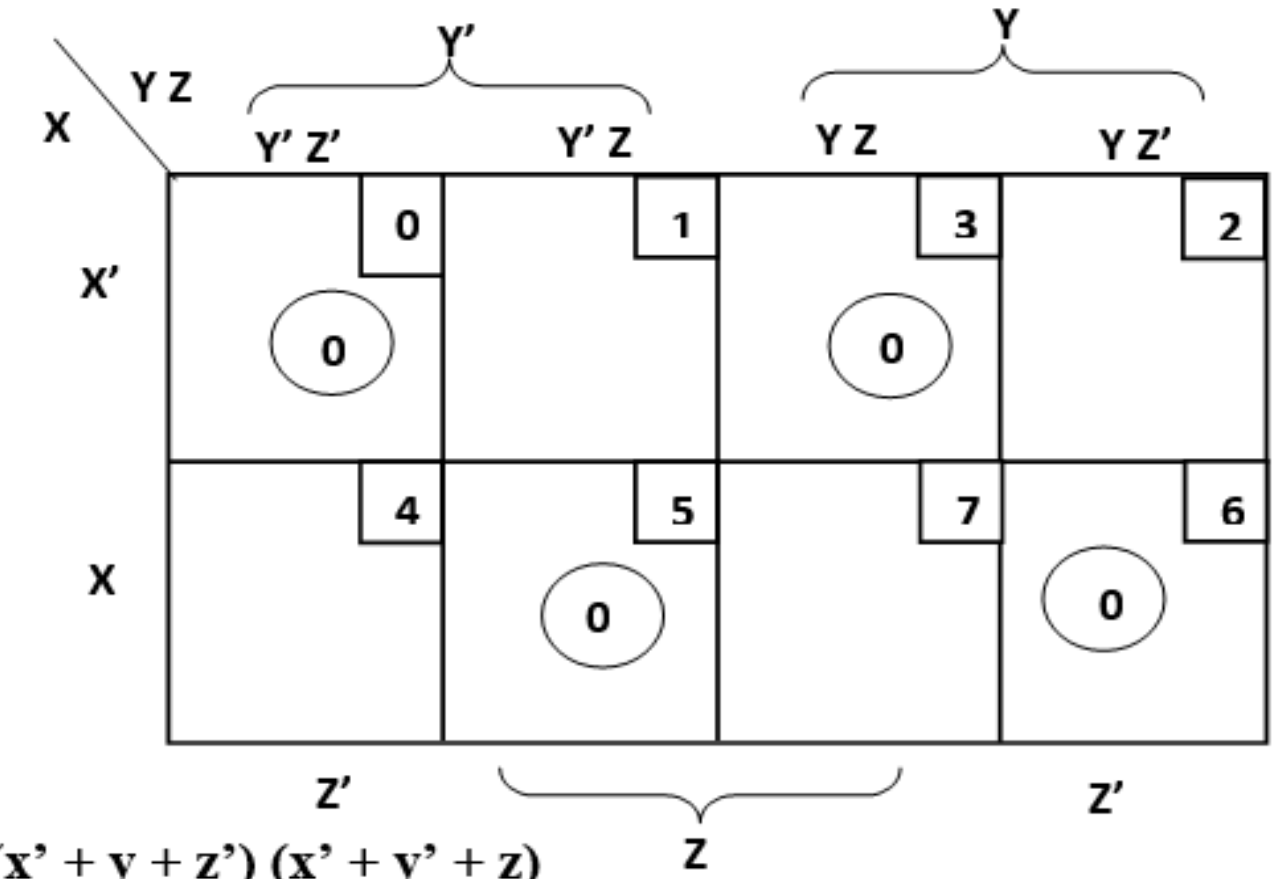
$$C = x\,y + x\,z + y\,z$$



Implementation of Full-adder in product of sums form:

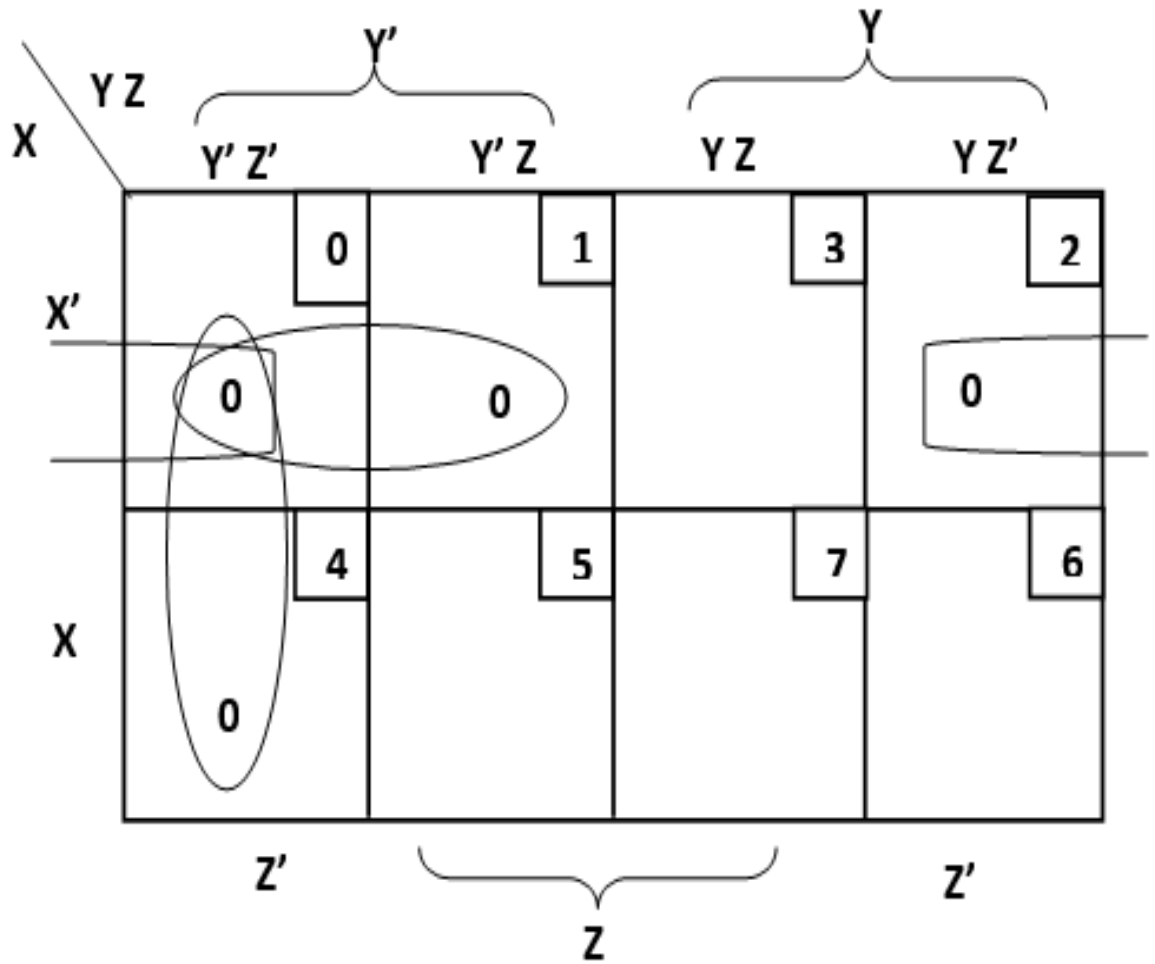
K-Map for simplified expression in POS for full adder:

For Sum:



S = (x + y + z) (x + y' + z') (x' + y + z') (x' + y' + z)

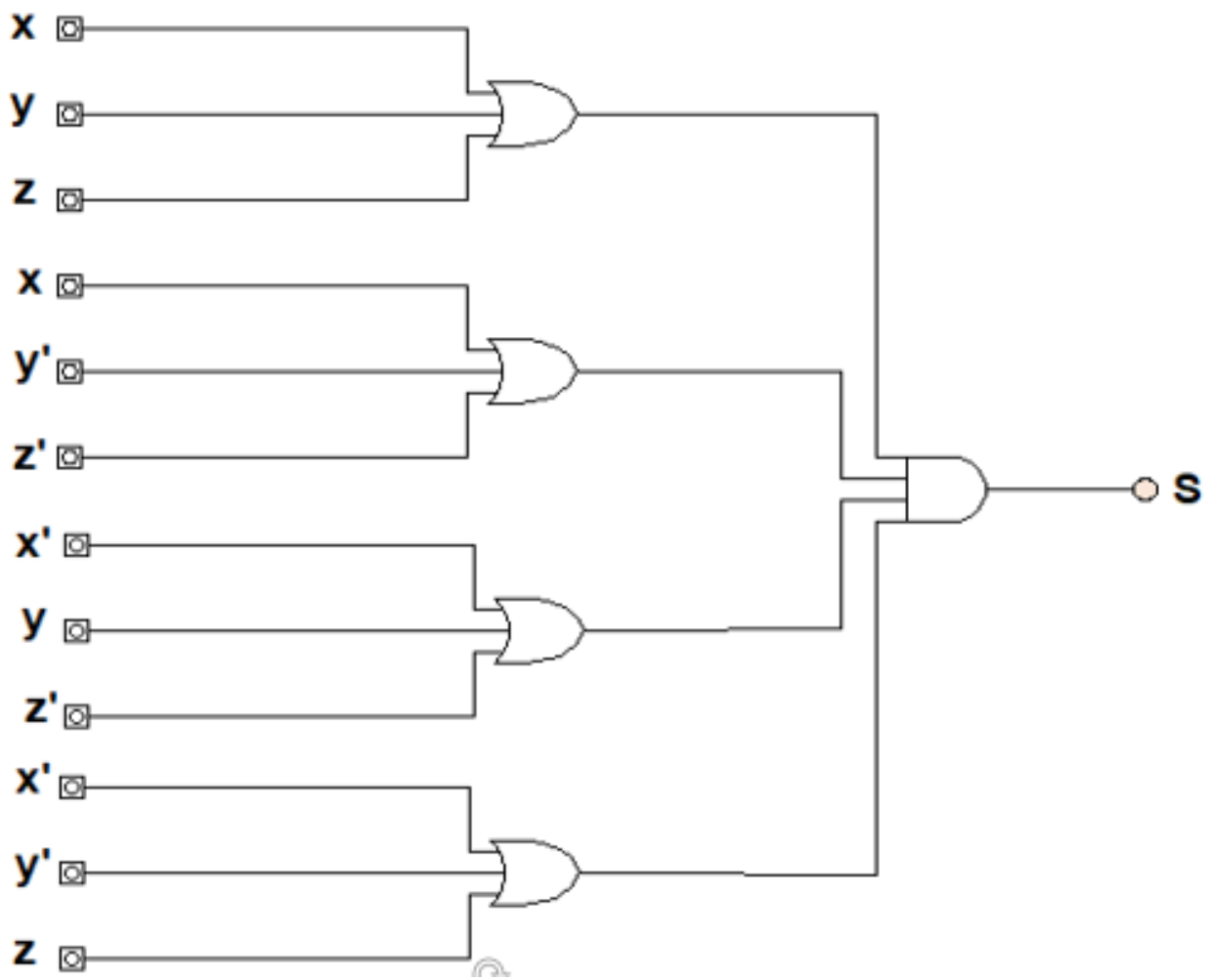
For Carry:



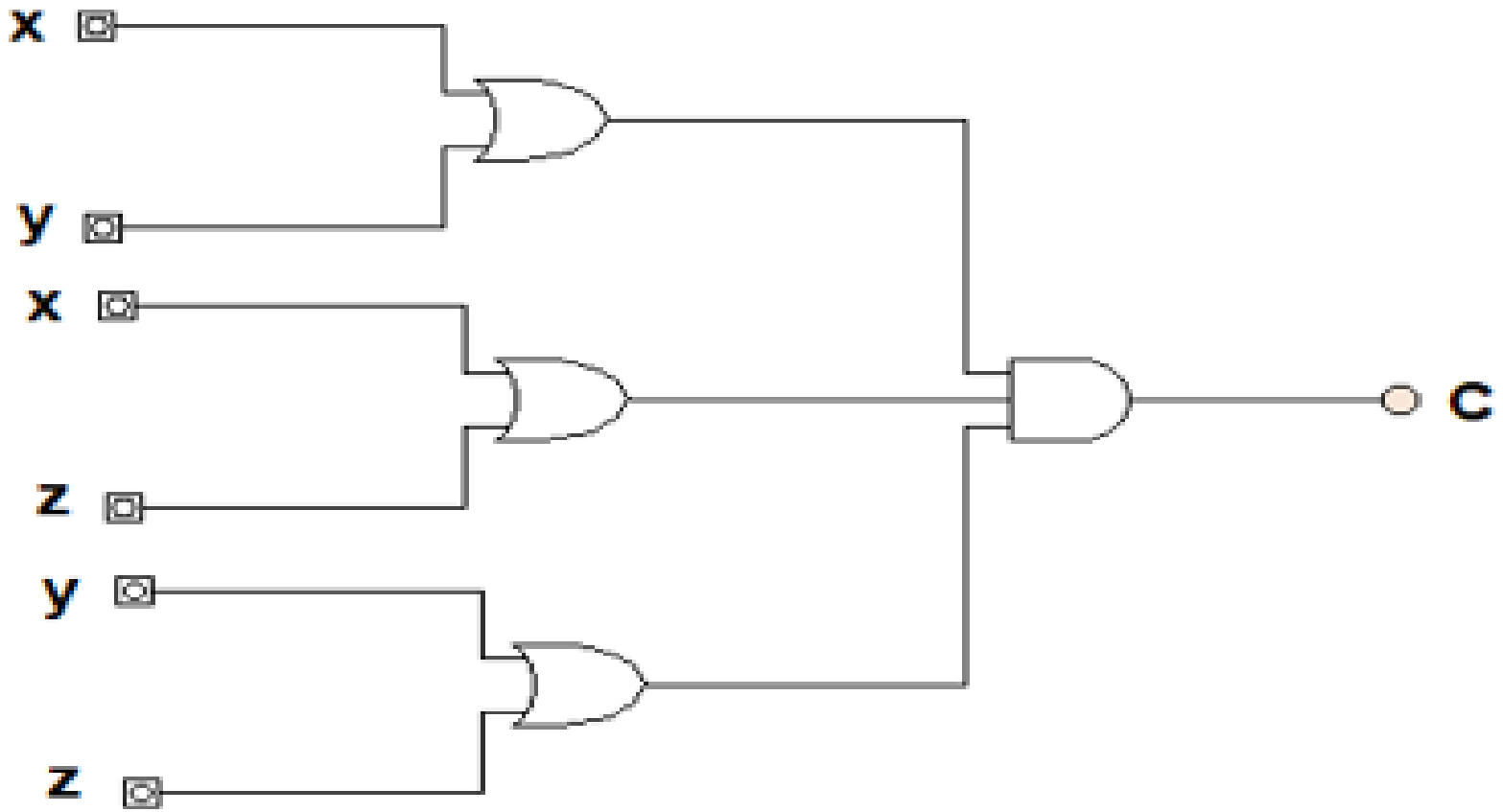
$C = (x + y)(x + z)(y + z)$

Implementation of full adder in POS with gates:

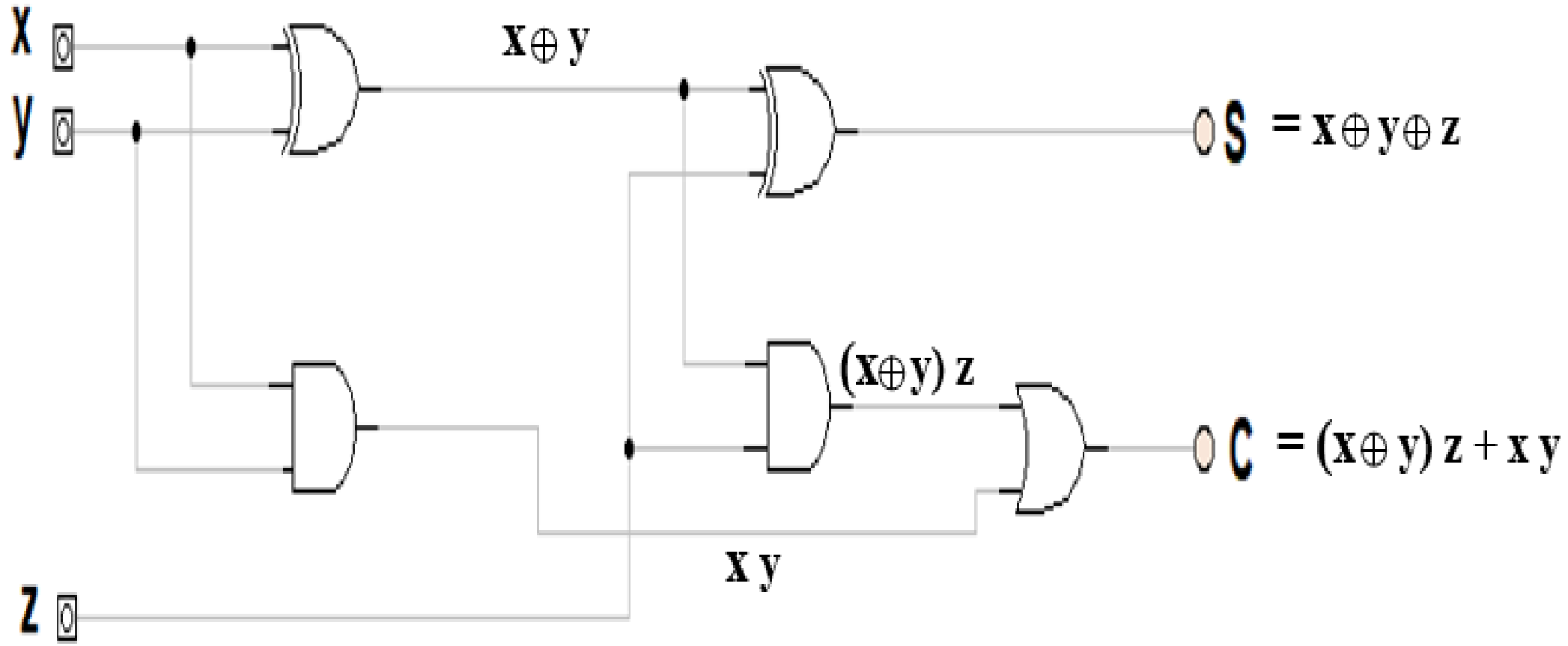
S = (x + y + z) (x + y' + z') (x' + y + z') (x' + y' + z)



$$C = (x + y)\,(x + z)\,(y + z)$$



Implementation of full adder using two half adder and one OR gate:



$S = x \oplus y \oplus z$

$= (x' y + x y') \oplus z$

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

$= \{(x' y)' (x y')'\} z + x' y z' + x y' z'$

$= \{(x + y') (x' + y)\} z + x' y z' + x y' z'$

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

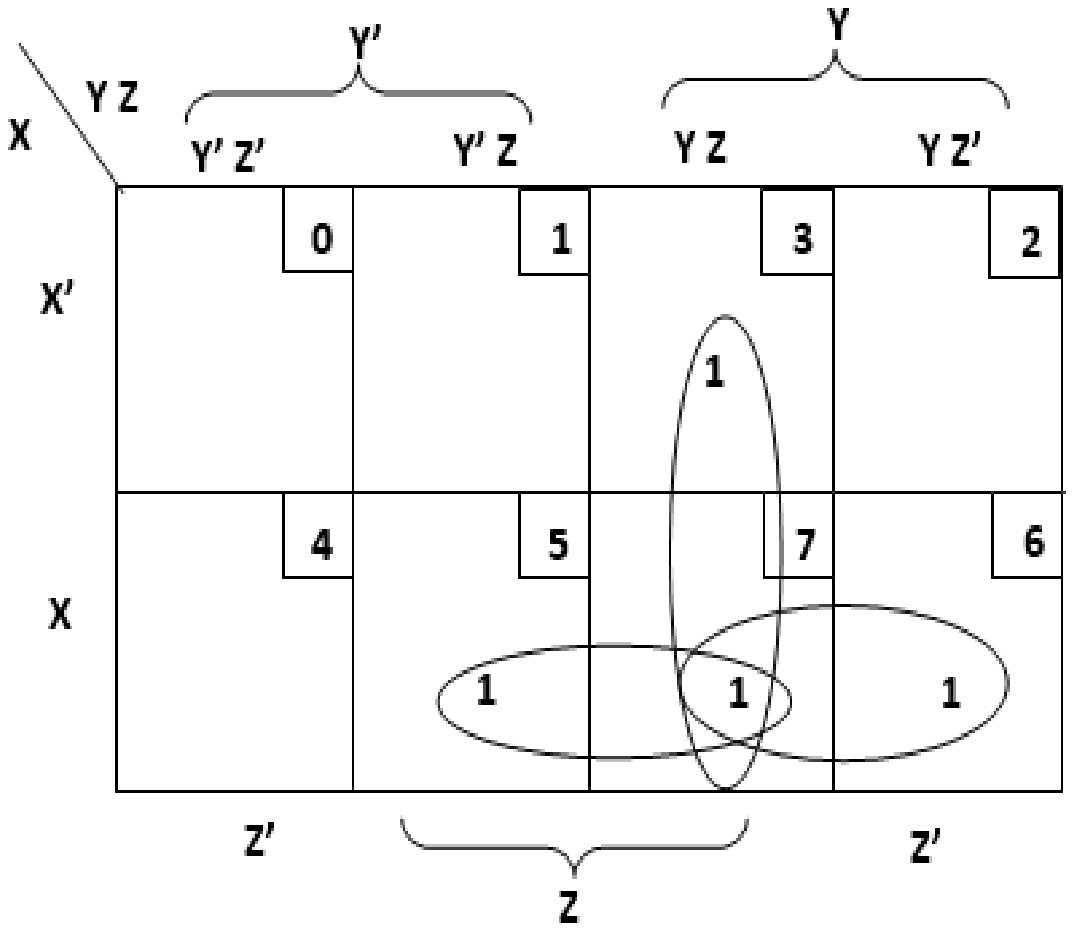
$= x y z + x' y' z + x' y z' + x y' z'$ **this expression cannot be simplified further**

$C = (x \oplus y) z + x y$

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

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

K – Map for carry



C = x z + x y + y z

Subtractor: Subtractor is the combinational circuit, which is used to subtract two or more than two binary digits at a time.

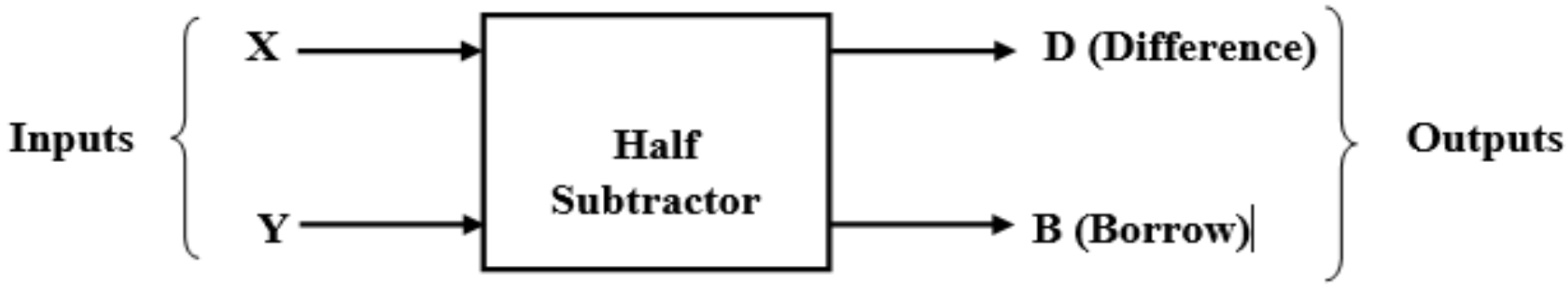
Types of subtractors:

1) **Half subtractor**
2) **Full subtractor**

Half Subtractor: A half-subtractor is a combinational circuit that subtracts two bits and produces their difference bit. Denoting minuend bit by x and the subtrahend bit by y. To perform x - y, we have to check the relative magnitudes of x and y:

If $x \geq y$, we have three possibilities: 0 - 0 = 0, 1 - 0 = 1, and 1 - 1 = 0. If $x < y$, we have 0 - 1, and it is necessary to borrow a 1 from the next higher stage. The half-subtractor needs two outputs, difference (D) and borrow (B).

Block diagram of Half-Subtractor:

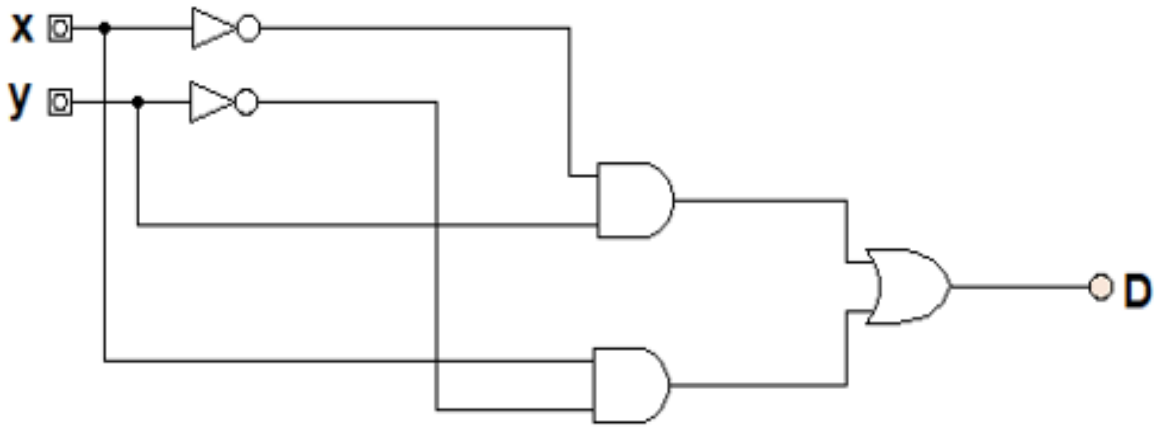


Truth table to identify the function of half-subtractor:

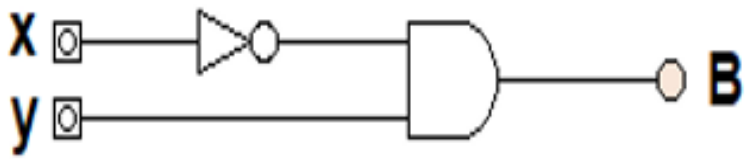
x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Logic Diagram of Half Subtractor:

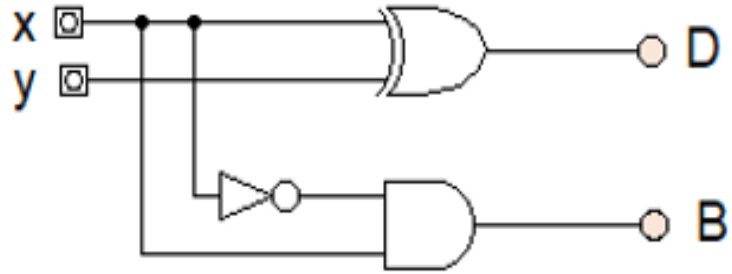
$D = x'y + xy'$



$B = x'y$



or

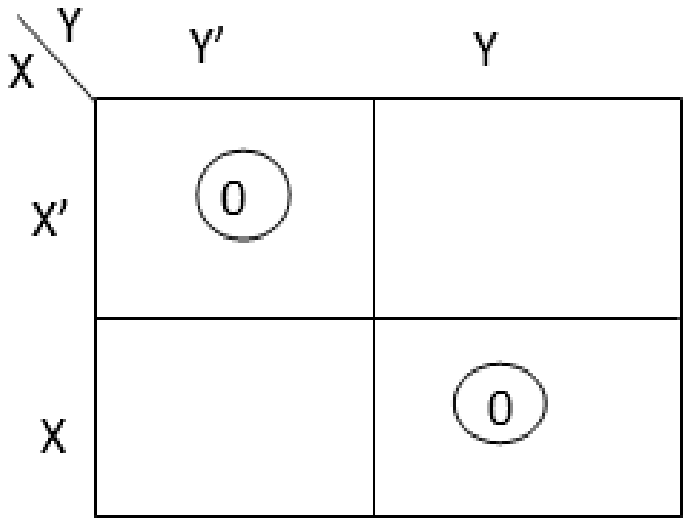


$D = x \oplus y$

$B = x' y$

Difference and borrow of Half- Subtractor can be implemented in POS form:

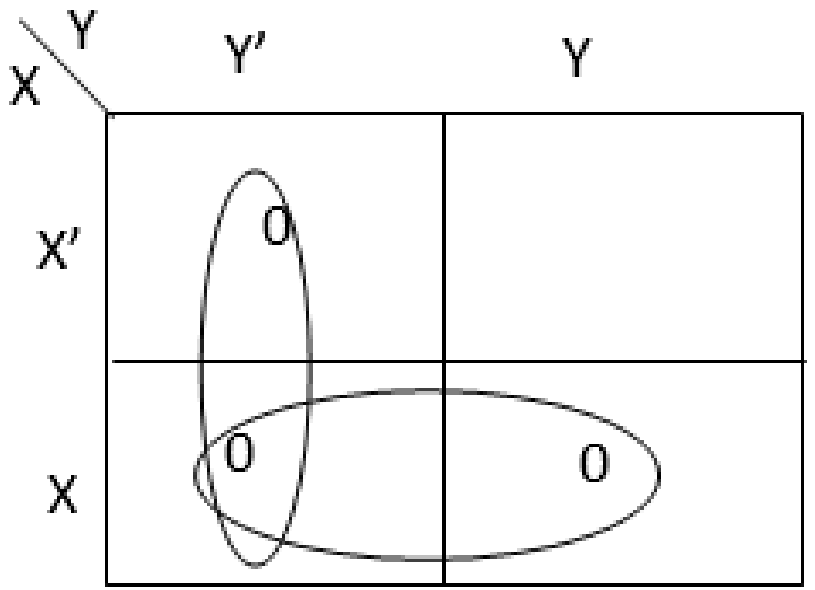
K-map for simplified expression for Difference in POS form:



Difference (D) = (x + y) (x' + y')

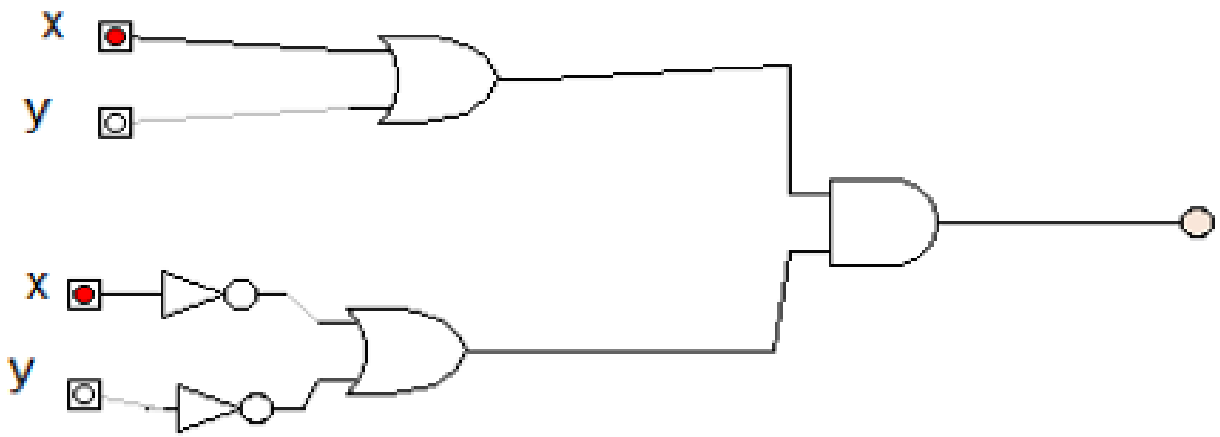
K-map for simplified expression for Borrow in POS for Sum:

B = (x + y')'



Logic diagram implementation of Half-subtractor in POS form:

D = (x + y) (x' + y')



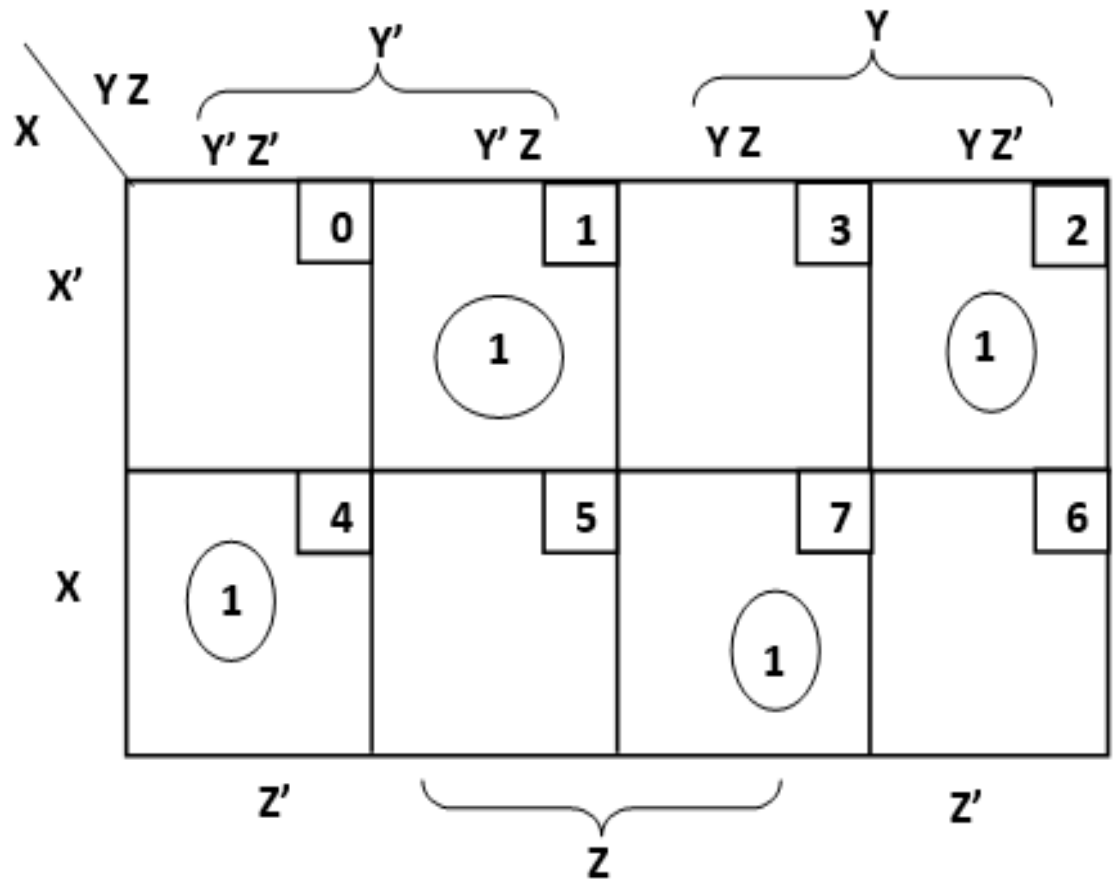
B = (x + y')'

Full Subtractor: Used to subtract three binary digits at a time. Inputs x, y and z, outputs Difference (D) and Borrow (B).

Truth table:

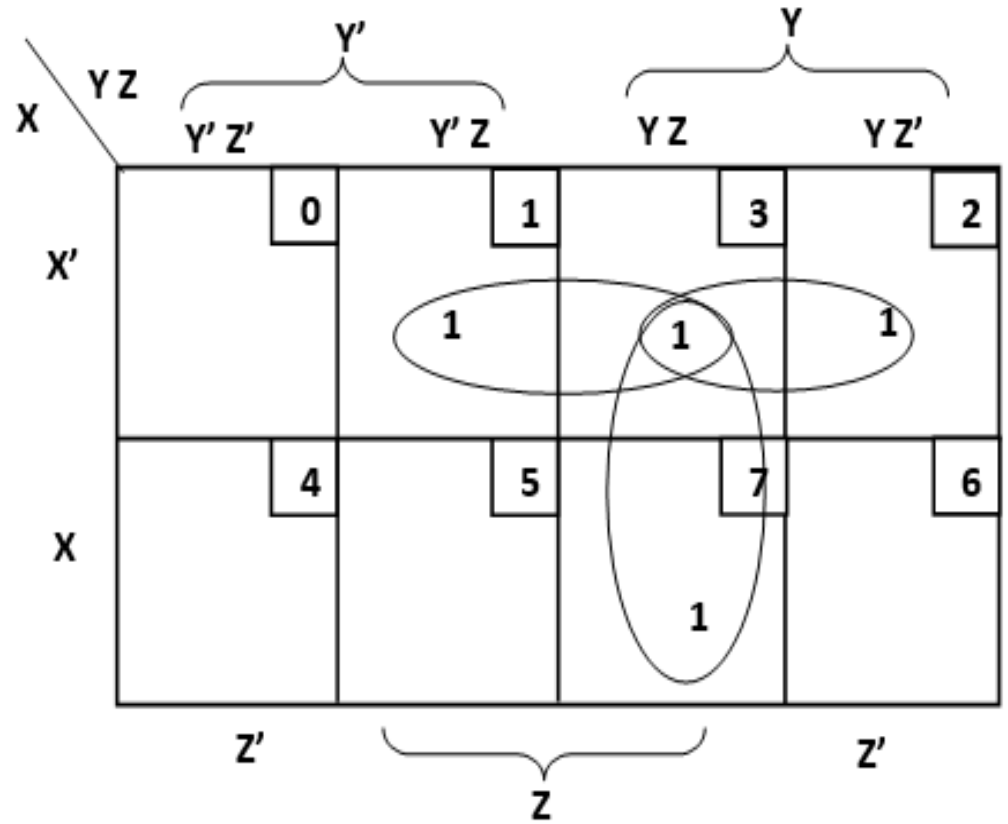
X Y Z	D B
0 0 0	0 0
0 0 1	1 1
0 1 0	1 1
0 1 1	0 1
1 0 0	1 0
1 0 1	0 0
1 1 0	0 0
1 1 1	1 1

K-map for simplified expression in SOP for Difference (D):



D = x' y' z + x' y z' + x y' z' + x y z

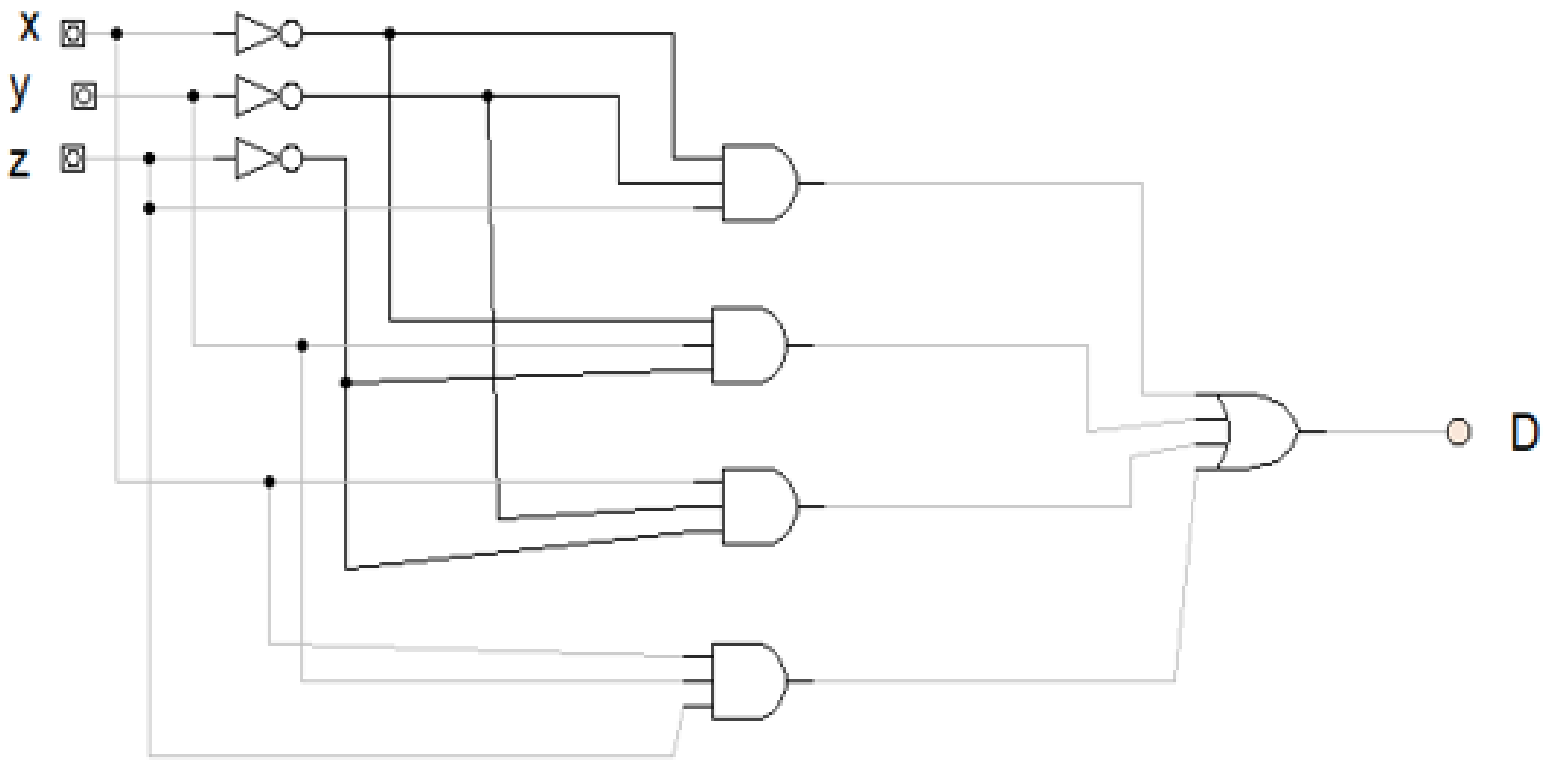
K-map for simplified expression in SOP for Borrow (B):



B= x' z + x' y + y z

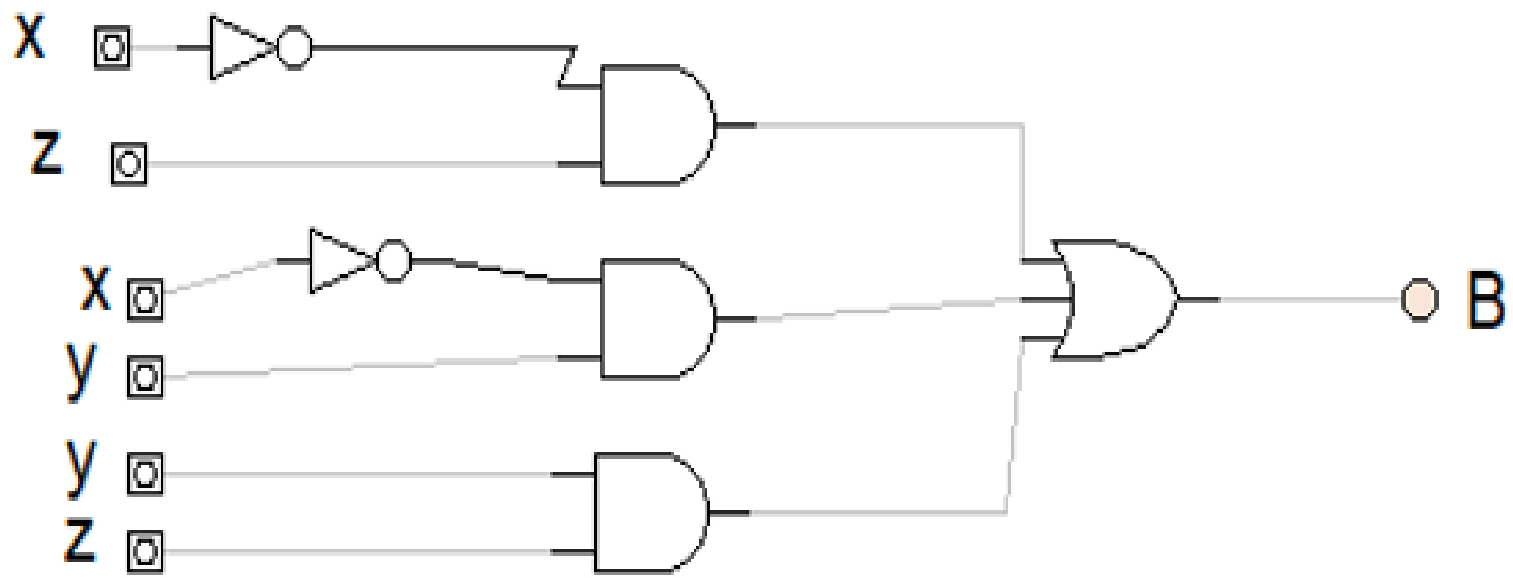
Logic diagram implementation of Full Subtractor in SOP form:

D = x' y' z + x' y z' + x y' z' + x y z



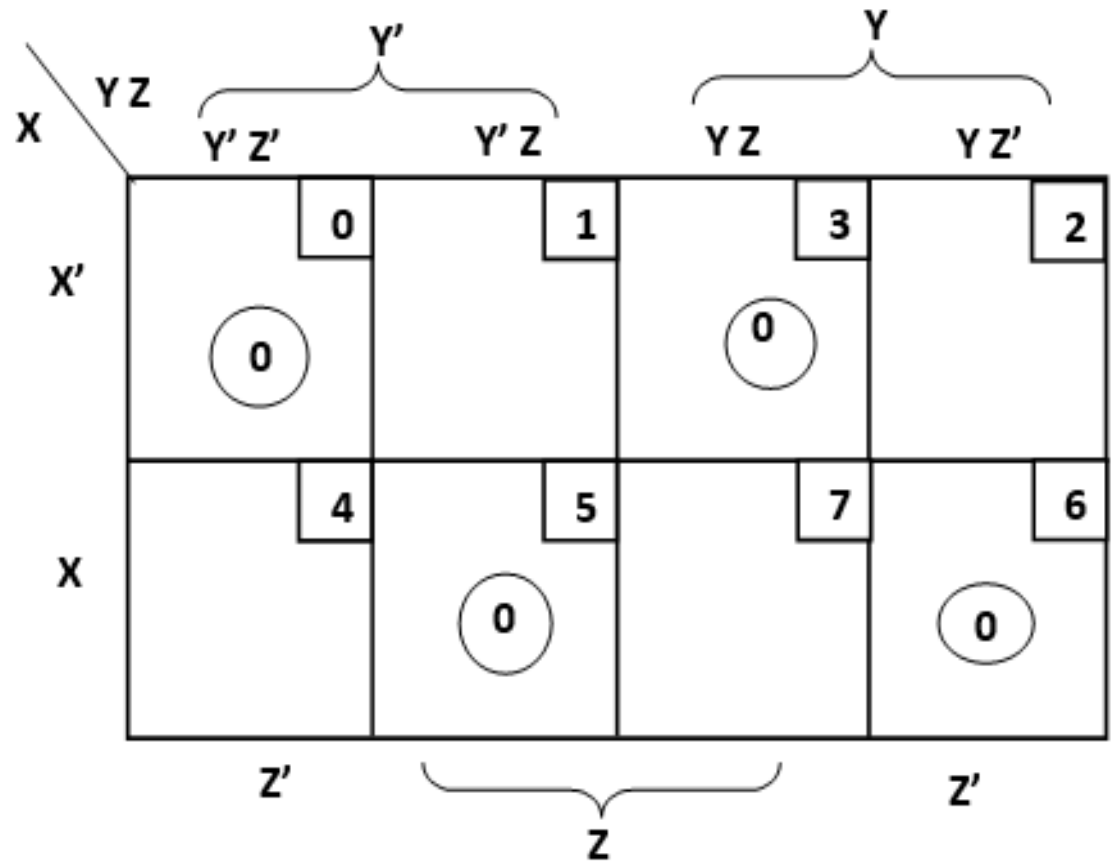
Logic diagram implementation of Full Subtractor in SOP form:

$$B = x' z + x' y + yz$$



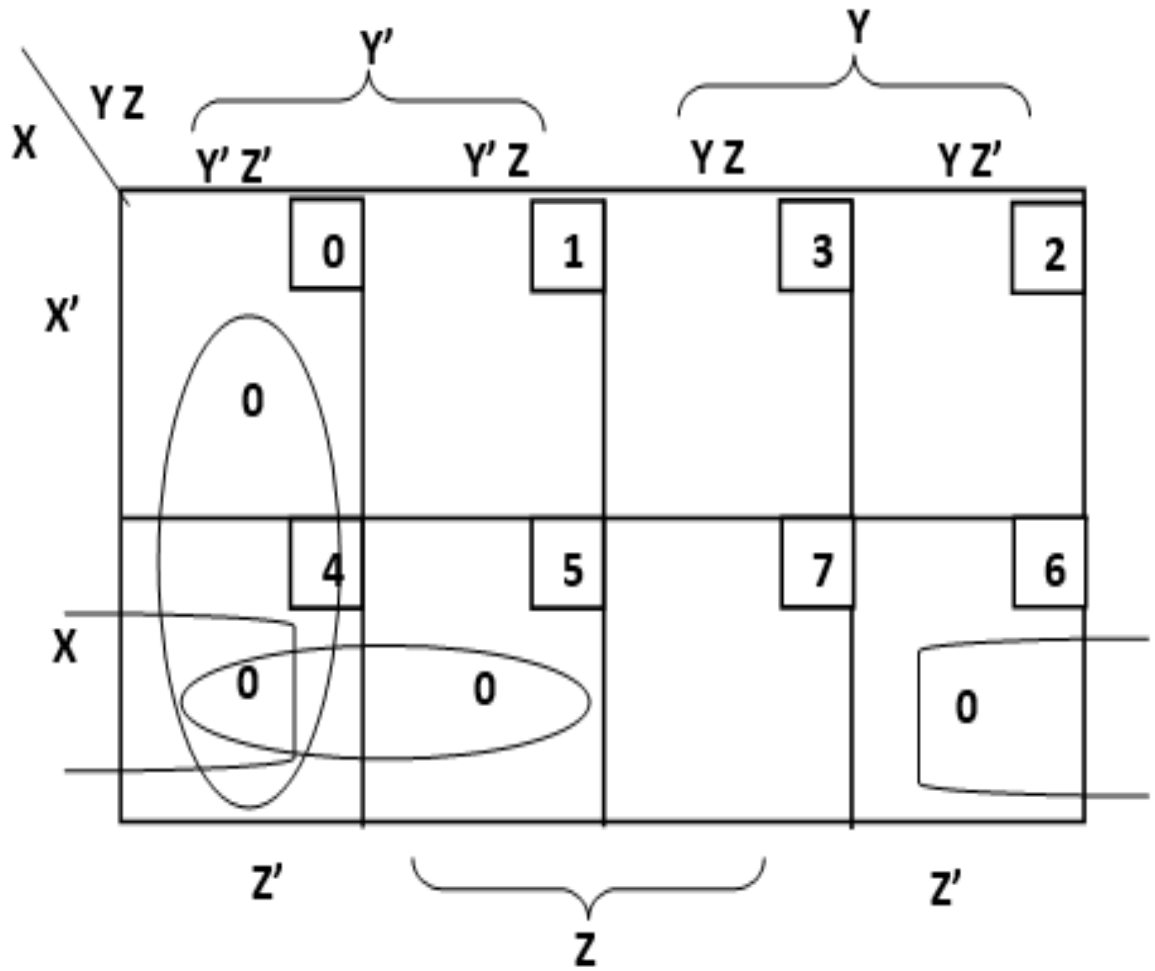
POS implementation of Full-Subtractor:

K-Map for D:



$$D = (x + y + z)\,(x + y' + z')\,(x' + y + z')\,(x' + y' + z)$$

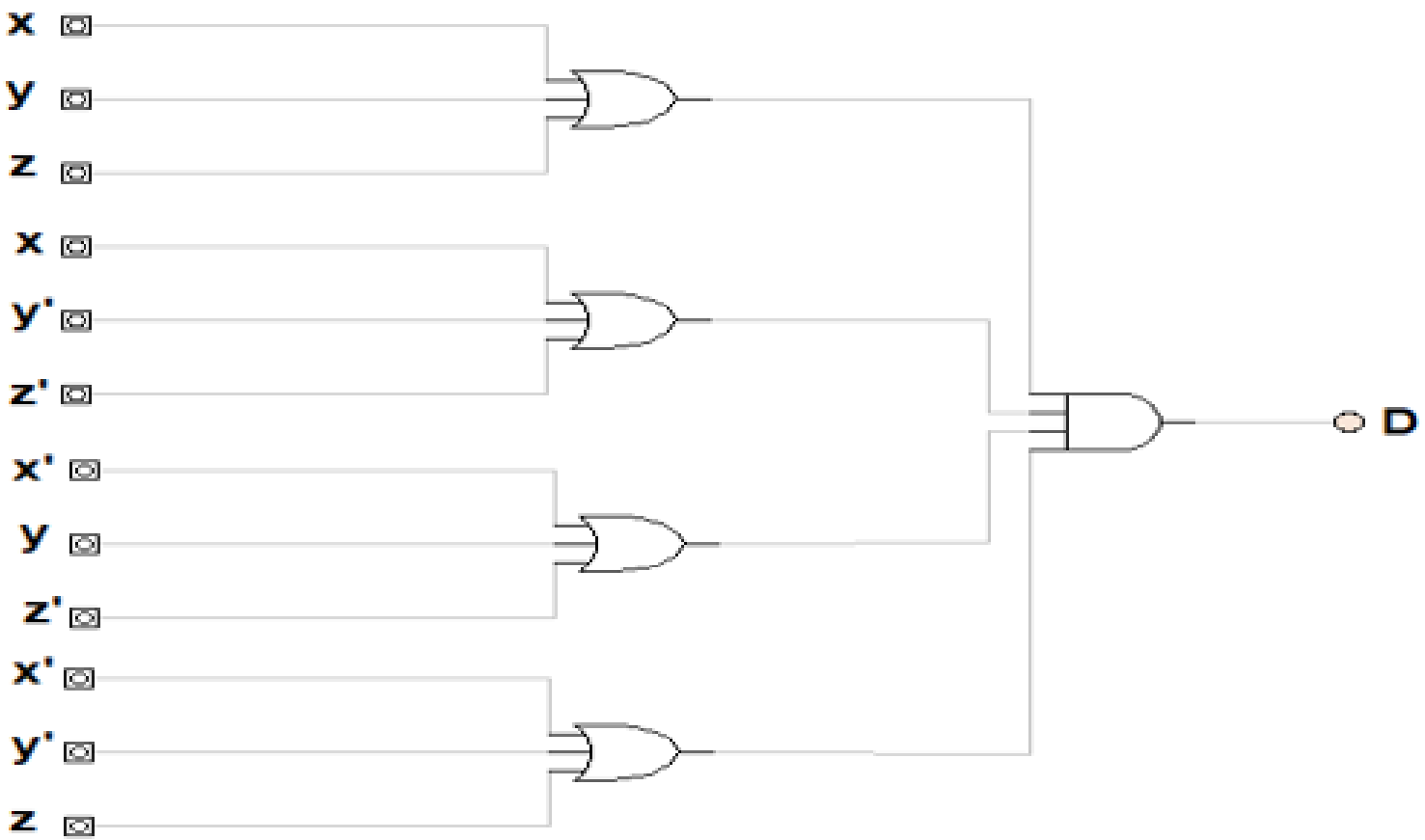
K-Map for Borrow(B)



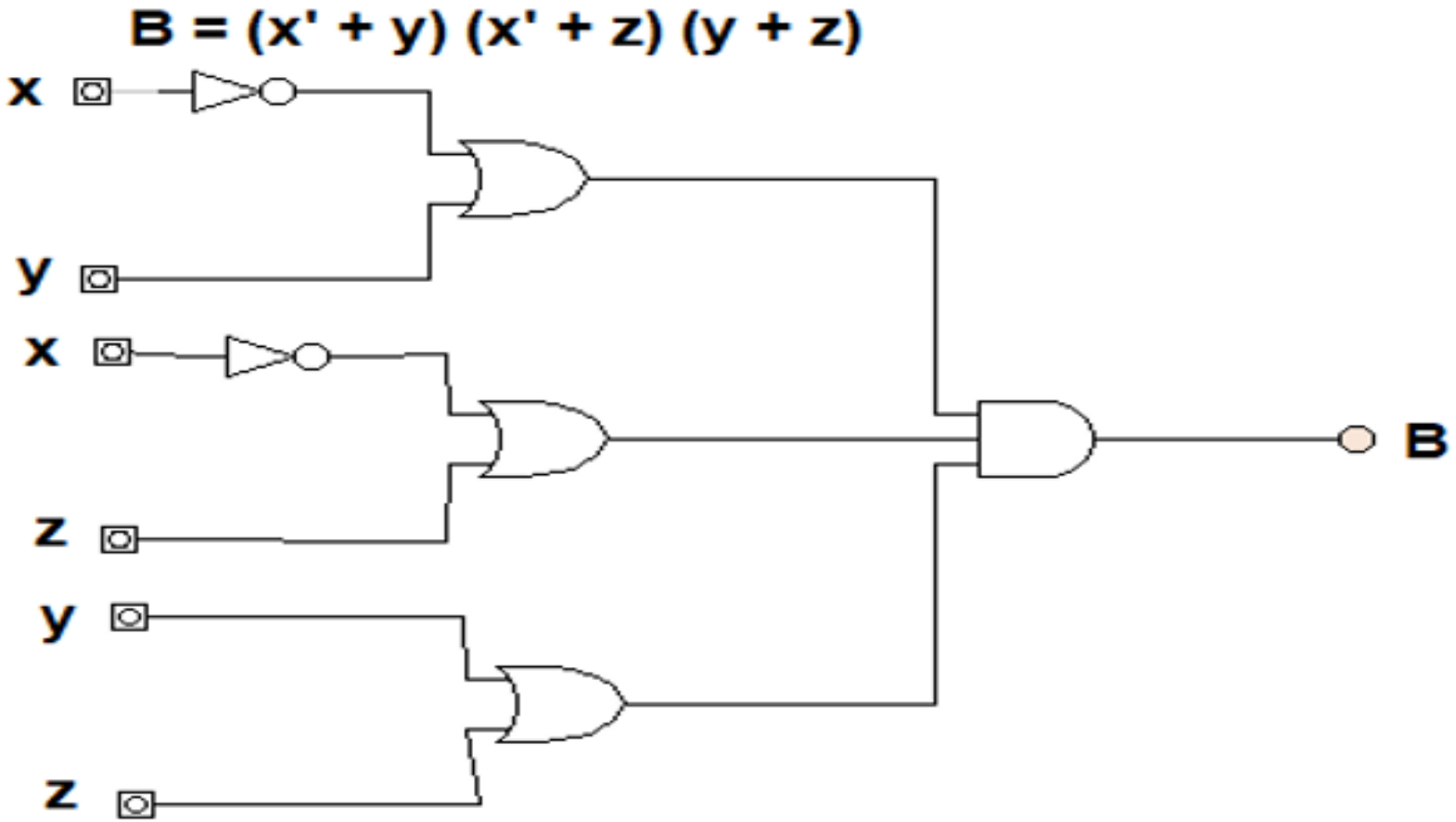
B = (y + z) (x' + y) (x' + z)

Logic diagram implementation of Full- Subtractor in POS form:

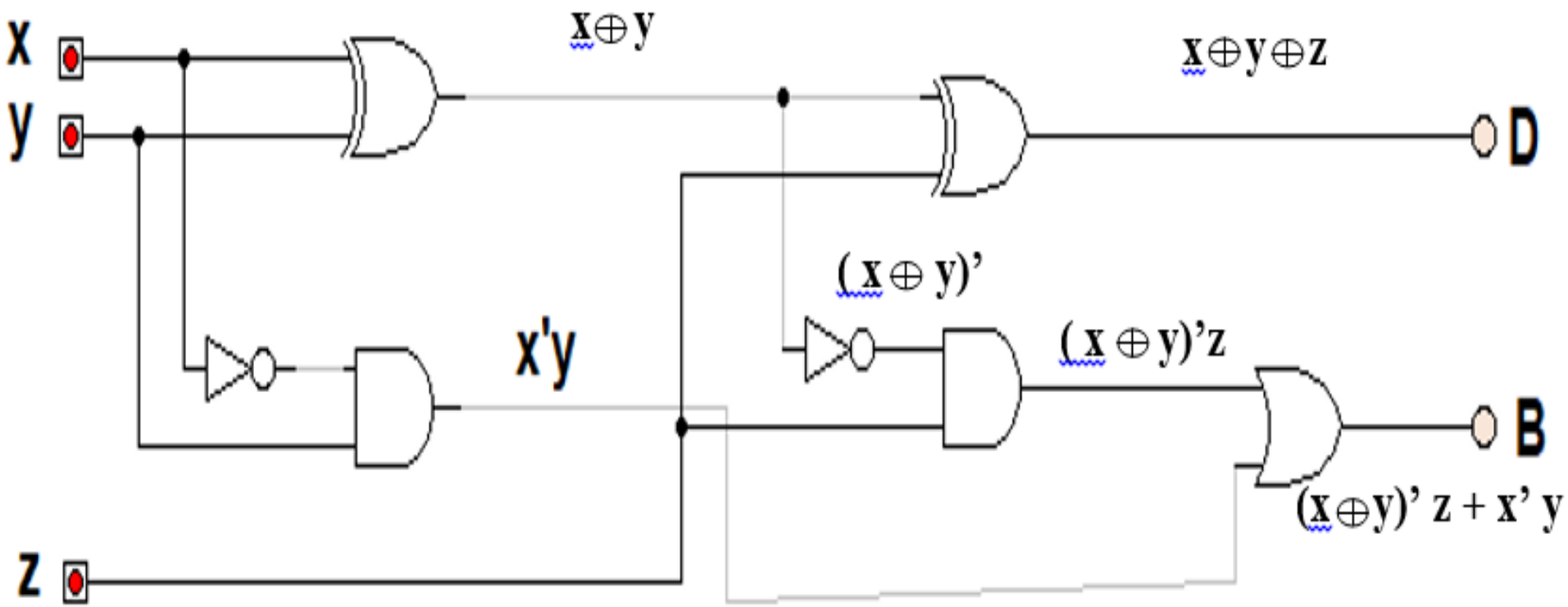
$$D = (x + y + z)\,(x + y' + z')\,(x' + y + z')\,(x' + y' + z)$$



Logic diagram implementation of Full- Subtractor in POS form:



Implementation of Full-Subtractor using two half-subtractor and one OR gate:



$D = S \oplus x \oplus y \oplus z$

$= (x' y + x y') \oplus z$

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

$= \{(x' y)' (x y')'\} z + x' y z' + x y' z'$

$= \{ (x + y') (x' + y)\} z + x' y z' + x y' z'$

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

$= x y z + x' y' z + x' y z' + x y' z'$ **This expression cannot be simplified further.**

$B = (x \oplus y)' z + x' y$

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

$= \{ (x' y)' (x y')'\} z + x' y$

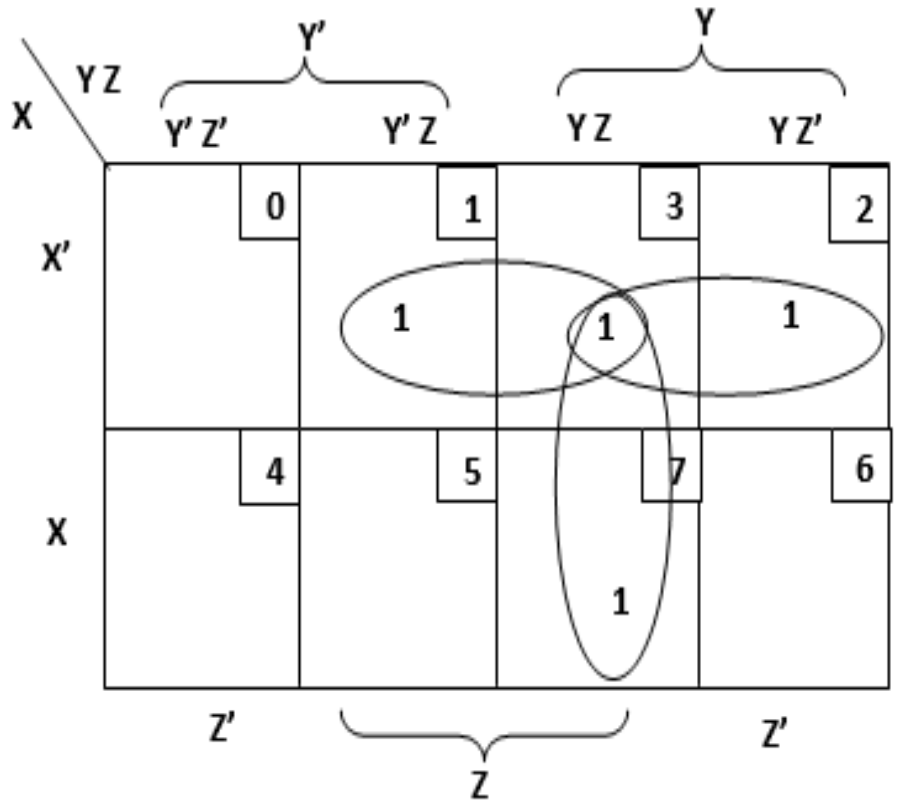
$=\{ (x + y') (x' + y)\} z + x' y$

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

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

$$B = x\,y\,z + x'\,y'\,z + x'\,y$$

K-Map for B:



$$B = x'\,z + x'\,y + y\,z$$

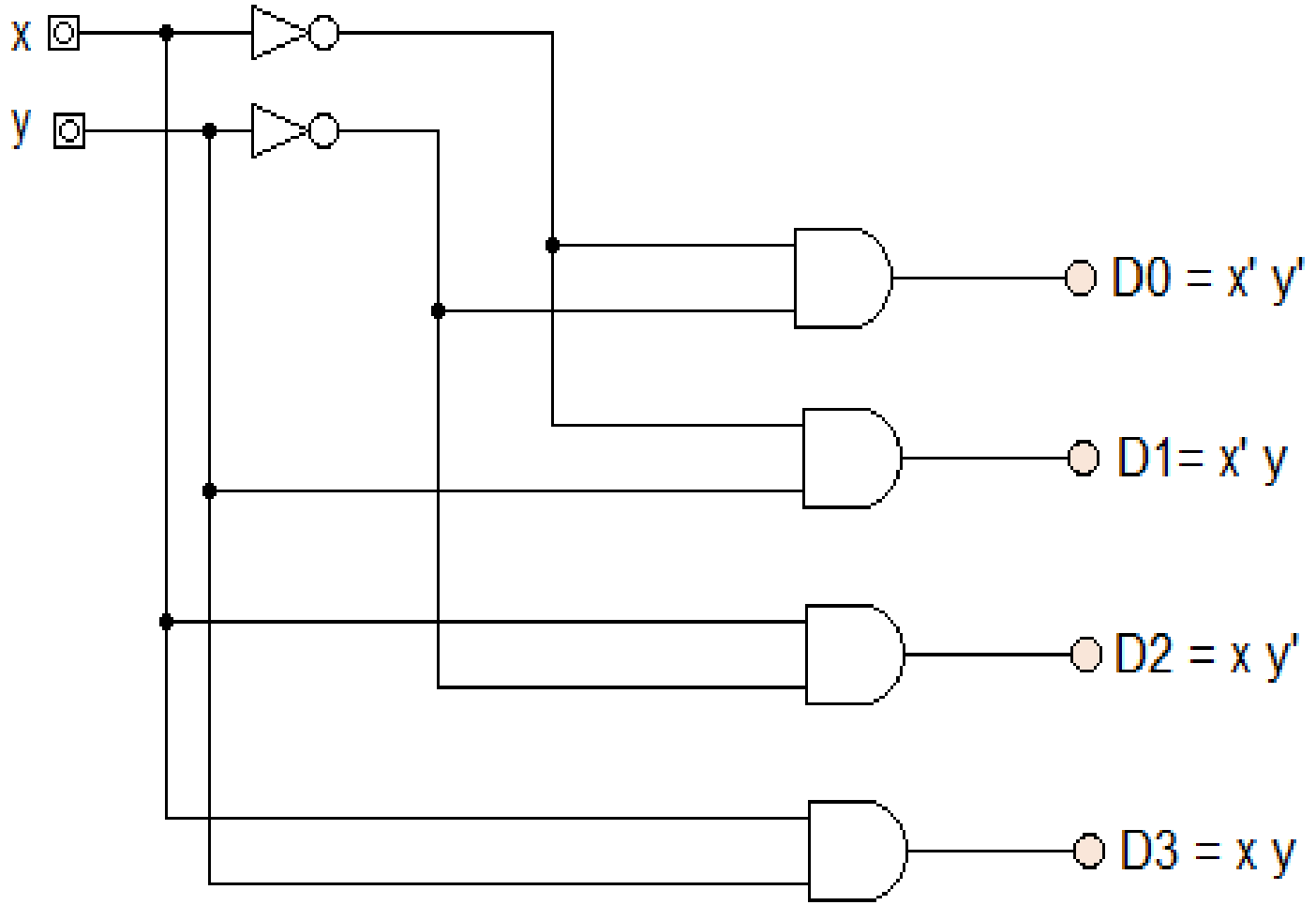
Decoder: Decoder is a combinational circuit that converts binary information from n input lines to maximum of 2^n unique output lines. If the n-bit decoded information has unused or don't care combinations, the decoder output will have less than 2^n outputs. There should be only one output line of decoder high at a time.

2 – to – 4 line decoder: It consists of two inputs and four output lines.

Truth Table:

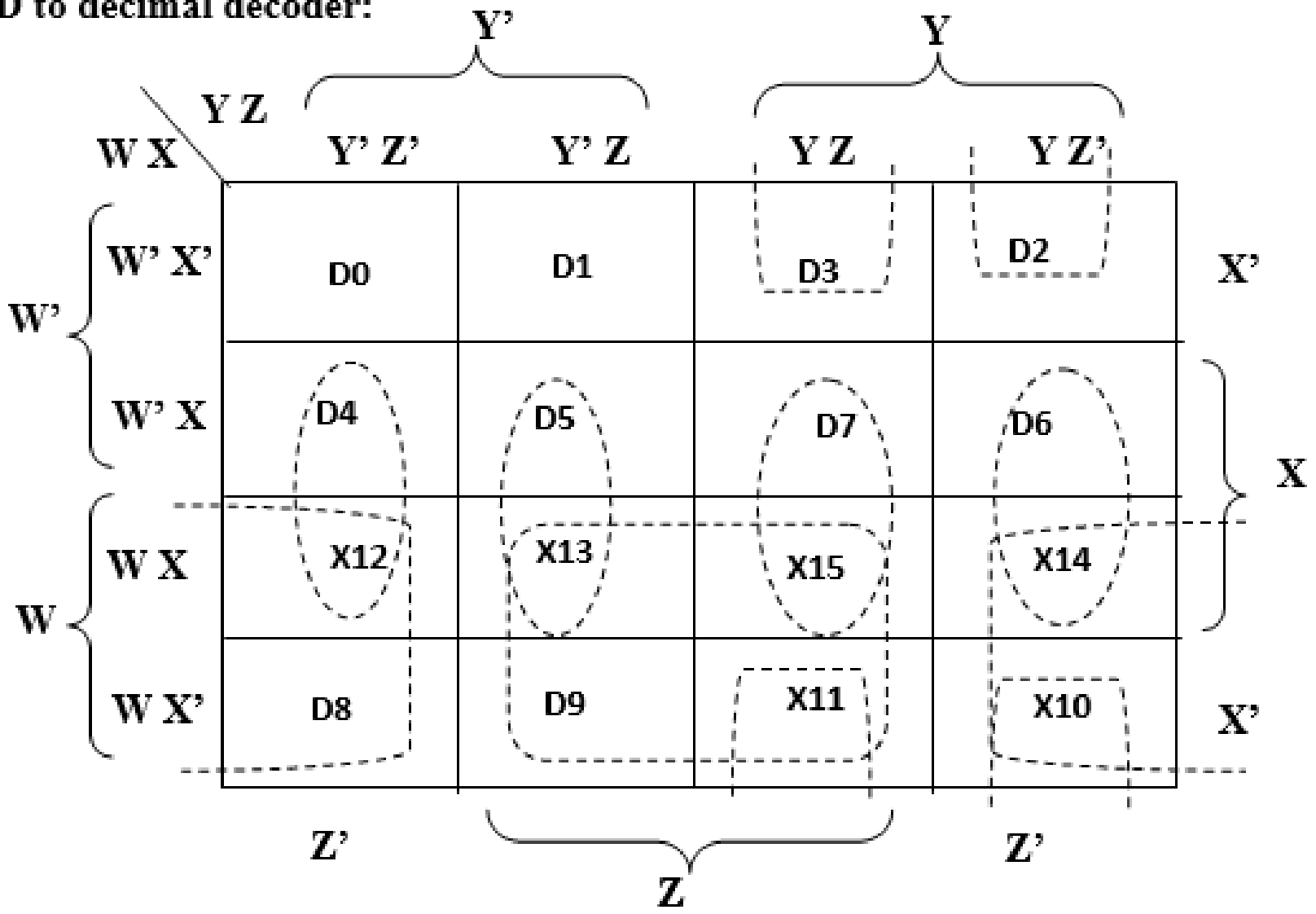
Inputs	Outputs
X Y	D_0 D_1 D_2 D_3
0 0	1 0 0 0
0 1	0 1 0 0
1 0	0 0 1 0
1 1	0 0 0 1

Logic diagram of 2 – to – 4 line decoder:



BCD – to Decimal Decoder: It consists of 4 inputs and 10 output lines: The decoder which converts binary coded decimal code (BCD) into decimal values is called BCD to decimal decoder. The BCD code uses four bits and therefore there should be 2^4 input combinations. This produces 16 different output signals, but the decimal digits are from 0 to 9 (equal to 10 different combinations). Hence other 6 input combinations are not used in decimal system. Therefore we can use don't cares to represent 10 to 15 and use K-Map to simplify the circuit of BCD to decimal decoder.

K-Map for BCD to decimal decoder:



Simplified expressions for different outputs:

D0 = w' x' y' z'

D1 = w' x' y' z

D2 = x' y z'

D3 = x' y z

D4 = x y' z'

D5 = x y' z

D6 = x y z'

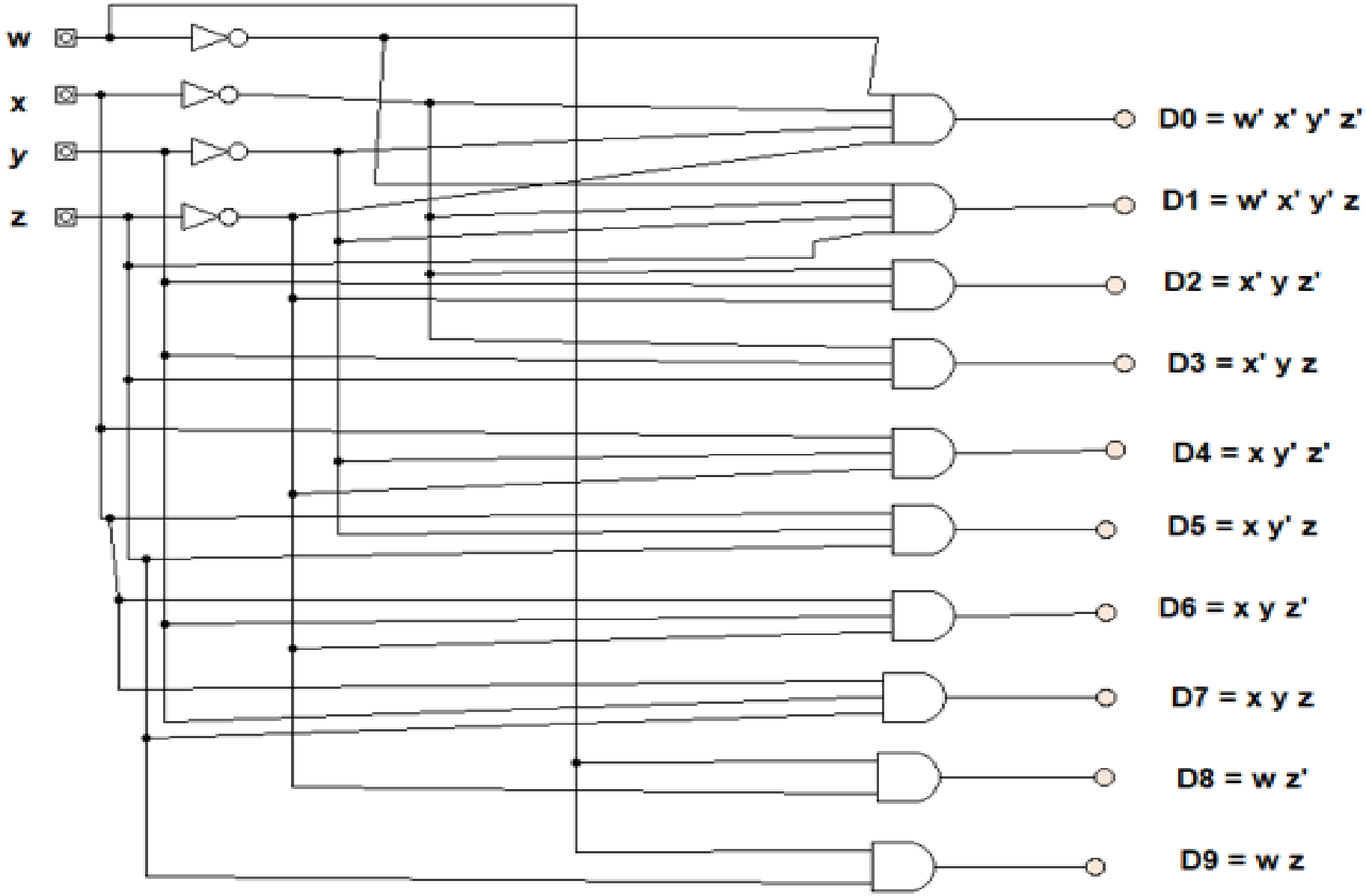
D7 = x y z

D8 = w z'

D9 = w z

- In the above K-Map D0 and D1 cannot be combined with any don't cares.
- D2 can be combined with don't care X10
- D3 with don't care X11
- D4 with don't care X12
- D5 with don't care X13
- D6 with don't care X14
- D7 with don't care X15
- D8 with don't cares X10, X12 and X14
- D9 with don't care X11, X13 and 15

Logic diagram of BCD to Decimal Decoder:



Truth Table of BCD to decimal decoder:

W	X	Y	Z	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Implementation of Full-Adder circuit using 3-to-8 line decoder and two OR gates:

Truth Table of Full-Adder:

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

Implementation of Full-Adder circuit using 3-to-8 line decoder and two OR gates:

Logic Circuit: