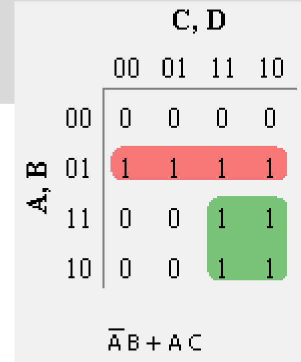
6. The Karnaugh Map

Aims

- to describe the ***Karnaugh map*** technique for circuit simplification
- to apply **Karnaugh** mapping as a technique
- to extend Karnaugh maps to include ***don't care*** states

6.0 Introduction



- Consider the truth table below : -

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

- In ***Sum of Products*** representation, the truth table can be expressed as : -

$$X = \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.C$$

$\overline{A}B\overline{C} \wedge \overline{A}BC \wedge A\overline{B}C \wedge ABC$

Using *Boolean algebra* the expression can be reduced as follows :-

Law name	AND form	OR form
Identity	$1A = A$	$0 + A = A$
Null	$0A = 0$	$1 + A = 1$
Idempotent	$AA = A$	$A + A = A$
Inverse	$A\bar{A} = 0$ $\bar{\bar{A}} = A$	$A + \bar{A} = 1$
Commutative	$AB = BA$	$A + B = B + A$
Associative	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption	$A(A + B) = A$	$A + AB = A$ $A + \bar{A}B = A + B$
De-Morgan's	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}.\bar{B}$

$$X = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC$$

$$X = \bar{A}B\bar{C} + \bar{A}BC + AC\left(\bar{B} + B\right)$$ ⇒ distributive **OR**

$$X = \bar{A}B\bar{C} + \bar{A}BC + AC(1)$$ ⇒ inverse **OR**

$$X = \bar{A}B\bar{C} + \bar{A}BC + AC$$ ⇒ identity **AND**

$$X = \bar{A}B\left(\bar{C} + C\right) + AC$$ ⇒ distributive **OR**

$$X = \bar{A}B(1) + AC$$ ⇒ inverse **OR**

$$X = \bar{A}B + AC$$ ⇒ identity **AND**

- Relatively straightforward for simple circuits.

  $\Rightarrow$ But ….. using Boolean algebra to reduce circuits can be difficult.

6.1 Karnaugh Maps

- ***Visual way of detecting redundancy in a*** ***Sum of Products*** ***expression***.

  ⇒ can easily be used for circuits with 2, 3, or 4 inputs.

- Consists of an array of cells, each representing a possible combination of inputs.

  ⇒ cells are arranged so that each cell's input combination differs from adjacent cells by only a single bit.

2-input

00	01
10	11

3-input

000	001	011	010
100	101	111	110

4-input

0000	0001	0011	0010
0100	0101	0111	0110
1100	1101	1111	1110
1000	1001	1011	1010

- Now lets look at empty Karnaugh Map tables – ready for logical inputs to be written in them.

2-input

		B	
		0	1
A	0		
	1		

3-input

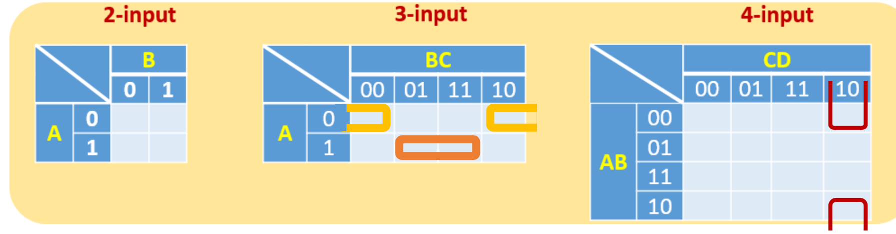
		BC			
		00	01	11	10
A	0				
	1				

4-input

		CD			
		00	01	11	10
AB	00				
	01				
	11				
	10				

NOTICE !!

- Karnaugh Map Numbers do not follow a binary sequence



⇒ Rows and columns are arranged so that only a single bit changes between neighbours

⇒ …. this applies at the edges too

Building a Karnaugh Map from a Truth Table

Starting with an empty ***Karnaugh map …***

- locate the ***1's*** in the function's truth table output

- place them in the cells corresponding to the same inputs

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

		BC			
		00	01	11	10
A	0	0	0	1	1
	1	0	1	1	0

Karnaugh Map RULES

1. Group adjacent 1's together in ***square*** *or* ***rectangular groups*** *of* ***2***, ***4***, ***8***, *or* ***16***,
such that the total number of groups and isolated 1's is minimised,
… while making the largest groups as possible

1. Groups may overlap
so that a particular cell may be included in more than one group

3. Remember that adjacency wraps around the edges of the map

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

- Considering the grouping, notice the following :

  1. B changes but the output does not
  - Therefore ***B*** is redundant in this group

  2. C changes but the output does not
  - Therefore ***C*** is redundant in this group

A \ B, C	00	01	11	10
0	0	0	0	1
1	0	1	0	0

$\overline{A}B\overline{C} + A\overline{B}C$

- **Boolean expressions are written for each group, leaving out the redundant terms**

$$X = \overline{A}.B + A.C$$

- Process of converting a truth table output into a ***Karnaugh map*** can be summarised into a basic set of rules :

1. Each cell with a **1** must be included in at least one group
2. Form the **largest possible groups**
3. Form as **few groups as possible**
4. Groups may be sizes that are powers of **2**
5. Groups may be **square** or **rectangular** only

   (including wrap-around at the map edges)

6. The **larger a group is**, **the more redundant inputs there are** : -

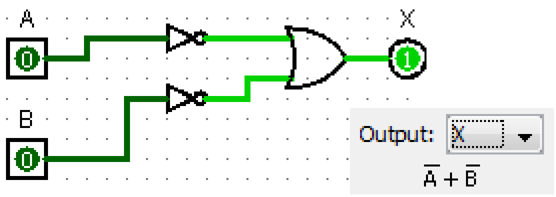
• A group	of	**1**	has	**no**	redundant inputs
• A group	of	**2**	has	**1**	redundant input
• A group	of	**4**	has	**2**	redundant inputs
• A group	of	**8**	has	**3**	redundant inputs
• A group	of	**16**	has	**4**	redundant inputs

2-Input

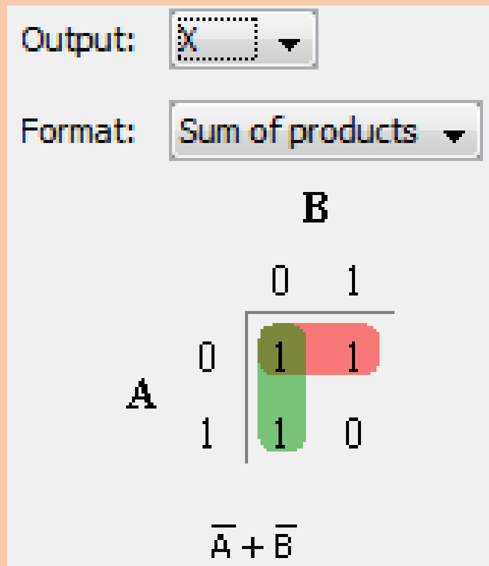
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

		B = 0	B = 1
A	0	1	1
	1	1	

$$X = \overline{B} + \overline{A}$$



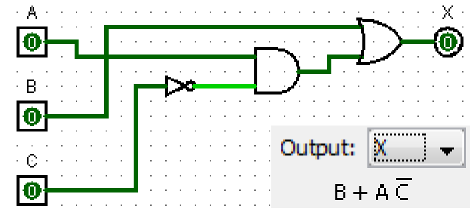
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0



3-Input

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

		BC			
		00	01	11	10
A	0			1	1
	1	1		1	1



$$X = A.\overline{C} + B$$

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Output: X

Format: Sum of products

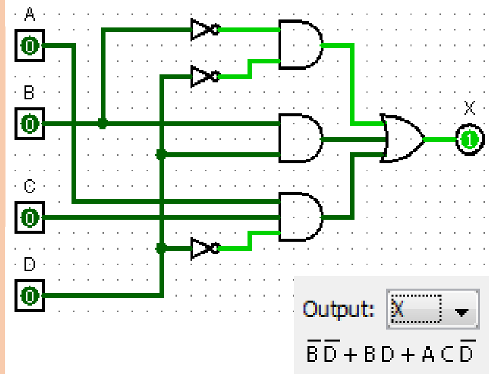
		B, C			
		00	01	11	10
A	0	0	0	1	1
	1	1	0	1	1

B + A $\overline{C}$

4-Input

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

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



$$X = \overline{B}.\overline{D} + B.D + A.C.\overline{D}$$

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	$\boxed{0}$
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Output: X

Format: Sum of products

A, B \ C, D	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	1
10	1	0	0	1

$\overline{B}\,\overline{D} + B\,D + A\,C\,\overline{D}$

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

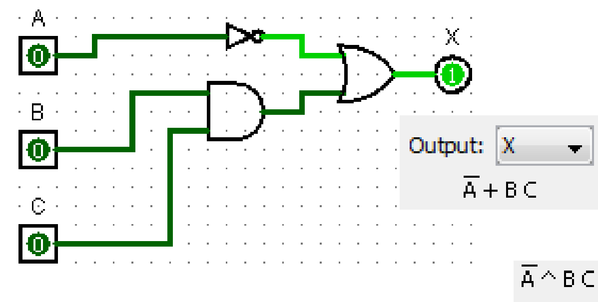
Output: X

Format: Sum of products

A \ B, C	00	01	11	10
0	1	1	1	1
1	0	0	1	0

$\overline{A} + B\,C$

$$X = \overline{A} + B.C$$



A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

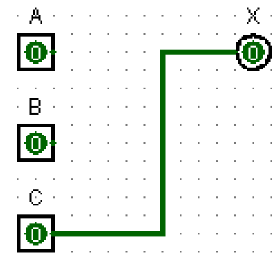
Output: X

Format: Sum of products

A \ B, C	00	01	11	10
0	0	1	1	0
1	0	1	1	0

C

$$X = C$$



A	B	C	D	X
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

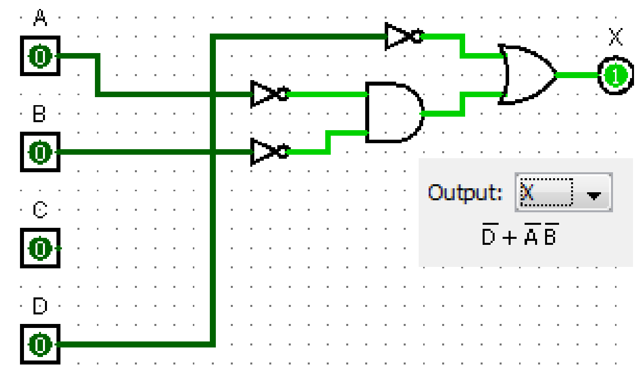
Output: X

Format: Sum of products

A, B \ C, D	00	01	11	10
00	1	1	1	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

$\overline{A}\,\overline{B} + \overline{D}$

$$X = \overline{D} + \overline{A}.\overline{B}$$



7.1 Don’t Care Logic States

We have seen how : -

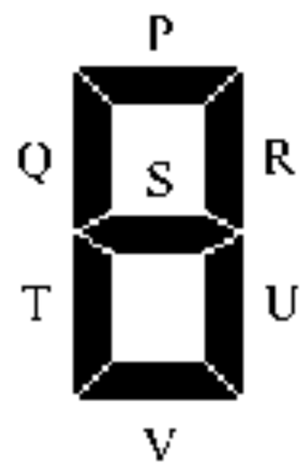
Karnaugh maps are a visual way of detecting redundancy in a Sum of Products representation

⇒ We sometimes have cases where the output of a circuit does not matter

⇒ Certain input combinations for which we ***do not care*** what the output is

Don’t Care States

- Consider a seven-segment display:



- Only the numbers 0 to 9 can be displayed.

⇒ **four bits** are required : - **0000** represents **zero**
1001 represents **nine**

- **Four bits** can actually represents values from **0** (0000) to **15** (1111).

  ⇒ when the input bit values are from **10** (1010) to **15** (1111)

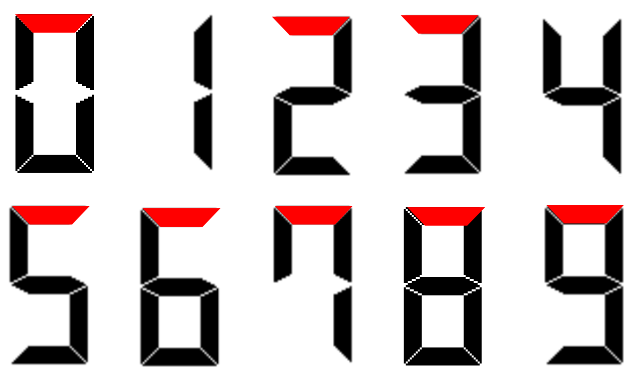
  we do not care what the output is

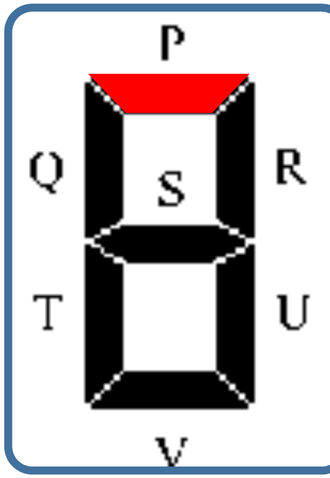
  ⇒ inputs to the circuits will never have those values so ***we do not care*** what the circuit does if it does happen

- Consider the segment labelled ***P***.

  ⇒ assume that a value of **1** activates the segment and a value of **0** turns it off

  ⇒ the top segment will be lit for values **0**, **2**, **3**, **5**, **6**, **7**, **8**, **9**



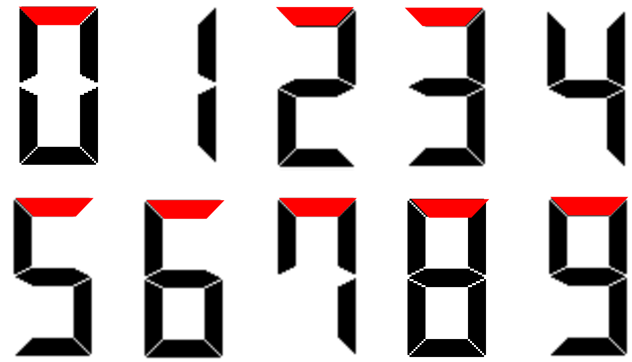


  ⇒ We ***do not care*** what it does if the inputs are in the range **10** to **15**

- In both the truth table and the Karnaugh map, a don't care state is represented with a '**d**'. They are drawn on the Karnaugh map along with the **1**'s

'X' is also a common 'don't care' notation

For segment ‘P’



A	B	C	D	P
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	d
1	0	1	1	d
1	1	0	0	d
1	1	0	1	d
1	1	1	0	d
1	1	1	1	d

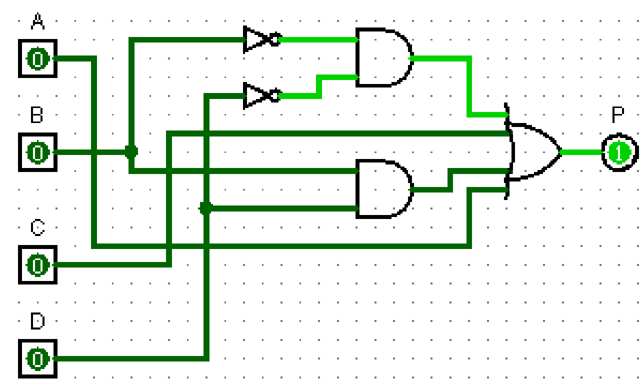
AB \ CD	00	01	11	10
00	1		1	1
01		1	1	1
11	d	d	d	d
10	1	1	d	d

$$P = A + C + B.D + \overline{B}.\overline{D}$$

A	B	C	D	P
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

A, B \ C, D	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	x	x	x	x
10	1	1	x	x

$\overline{B}\,\overline{D} + C + BD + A$



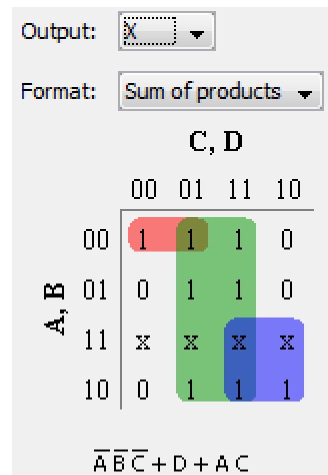
Example

A	B	C	D	X	Y	Z
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	1	1	0
0	1	1	0	0	0	1
0	1	1	1	1	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	d	d	d
1	1	0	1	d	d	d
1	1	1	0	d	d	d
1	1	1	1	d	d	d

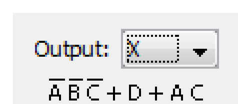
A	B	C	D	X	Y	Z
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	1	1	0
0	1	1	0	0	0	1
0	1	1	1	1	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	x	x	x
1	1	0	1	x	x	x
1	1	1	0	x	x	x
1	1	1	1	x	x	x

X

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



$$X = \overline{A}.\overline{B}.\overline{C} + D + A.C$$



Example

A	B	C	D	X	Y	Z
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	1	1	0
0	1	1	0	0	0	1
0	1	1	1	1	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	d	d	d
1	1	0	1	d	d	d
1	1	1	0	d	d	d
1	1	1	1	d	d	d

A	B	C	D	X	Y	Z
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	1	1	0
0	1	1	0	0	0	1
0	1	1	1	1	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	x	x	x
1	1	0	1	x	x	x
1	1	1	0	x	x	x
1	1	1	1	x	x	x

Y

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

Output: Y

Format: Sum of products

A, B \ C, D	00	01	11	10
00	1	0	1	0
01	1	1	1	0
11	x	x	x	x
10	1	0	1	0

$\overline{C}\,\overline{D} + C\,D + B\,\overline{C}$

$$Y = \overline{C}.\overline{D} + C.D + B.D$$

Output: Y

$\overline{C}\,\overline{D} + C\,D + B\,\overline{C}$

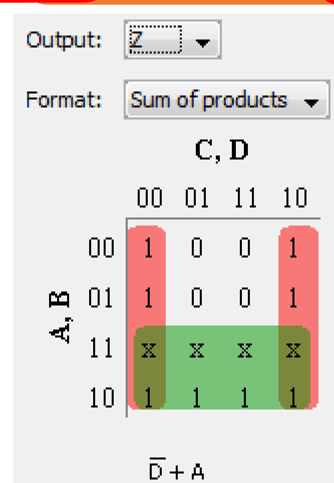
Example

A	B	C	D	X	Y	Z
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	1	1	0
0	1	1	0	0	0	1
0	1	1	1	1	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	d	d	d
1	1	0	1	d	d	d
1	1	1	0	d	d	d
1	1	1	1	d	d	d

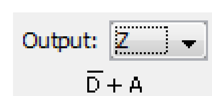
A	B	C	D	X	Y	Z
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	1	1	0
0	1	1	0	0	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	x	x	x
1	1	0	1	x	x	x
1	1	1	0	x	x	x
1	1	1	1	x	x	x

Z

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

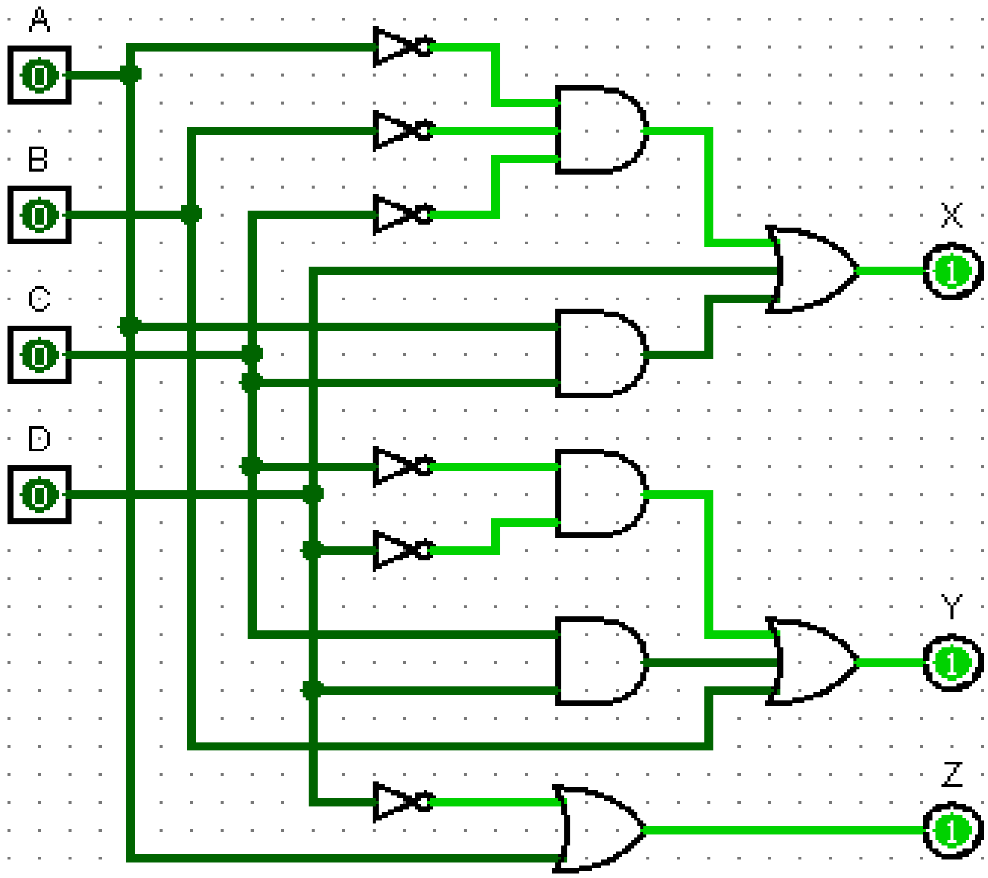


$$Z = \overline{D} + A$$



The circuit for X , Y , Z

A	B	C	D	X	Y	Z
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	1	1	0
0	1	1	0	0	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	x	x	x
1	1	0	1	x	x	x
1	1	1	0	x	x	x
1	1	1	1	x	x	x



$$X = \overline{A}.\overline{B}.\overline{C} + D + A.C$$

$$Y = \overline{C}.\overline{D} + C.D + B.D$$

$$Z = \overline{D} + A$$