

Topic 5. Switching algebra. Logic function minimisation.

Switching Algebra

Switching algebra consist of :

- 1) A set of elements $B = \{0,1\}$;
- 2) Logic operations AND, OR and NOT, that are defined as:

AND

$$0 * 0 = 0$$

$$0 * 1 = 0$$

$$1 * 0 = 0$$

$$1 * 1 = 1$$

OR

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

NOT

$$\overline{0} = 1$$

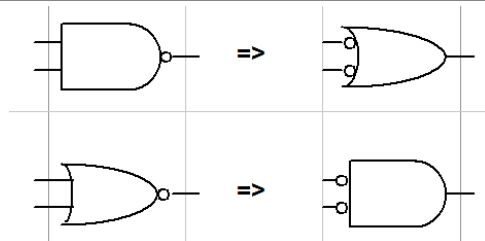
$$\overline{1} = 0$$

Single-variable Theorems (Axioms)

1. $X + 0 = X$ $X \cdot 1 = X$ - Identities
2. $X + 1 = 1$ $X \cdot 0 = 0$ - Null element
3. $X + X = X$ $X \cdot X = X$ - Idem potency
4. $\overline{\overline{X}} = X$ - Involution
5. $X + \overline{X} = 1$ $X \cdot \overline{X} = 0$ - Complements

Two and three variable Theorems

No	Logic expressions		Theorem
1.	$x_1 + x_2 = x_2 + x_1$	$x_1 \cdot x_2 = x_2 \cdot x_1$	<i>Commutativity</i>
2.	$(x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$	$(x_1 \cdot x_2) \cdot x_3 = x_1 \cdot (x_2 \cdot x_3)$	<i>Associativity</i>
3.	$x_1 \cdot x_2 + x_1 \cdot x_3 = x_1 \cdot (x_2 + x_3)$	$(x_1 + x_2) \cdot (x_1 + x_3) = x_1 + (x_2 \cdot x_3)$	<i>Distributivity</i>
4.	$x_1 + x_1 x_2 = x_1$	$x_1 \cdot (x_1 + x_2) = x_1$	<i>Covering</i>
	Proof: $x_1 + x_1 x_2 = x_1 \cdot 1 + x_1 x_2 = x_1(1 + x_2) = x_1 \cdot 1 = x_1$		
5.	$x_1 x_2 + x_1 \overline{x_2} = x_1$	$(x_1 + x_2) \cdot (x_1 + \overline{x_2}) = x_1$	<i>Combining</i>
	Proof: $x_1 x_2 + x_1 \overline{x_2} = x_1 \cdot (x_2 + \overline{x_2}) = x_1 \cdot 1 = x_1$		
6.	$\overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}$	$\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$	<i>De Morgan</i>



Obs. 1. The number of terms in theorems can be extended.

Obs. 2. In all theorems it is possible to replace each variable with an arbitrary logic expression.

We can substitute complex expressions using these theorems.

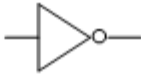



Logic signals and gates





In a logic function $y = f(x_1, x_2, \dots, x_n)$ variables and functions can take on only 2 values: 0 and 1.

A logic function of n variables are defined in $m=2^n$ points. As the value of a logic function in these points can be only 0 and 1, there are $N= 2^m$ such functions.

For $n=1 : N=4$

$f_i \backslash x$	0	1	Representation	<i>Name</i>
f_0	0	0	0	Constant 0
f_1	0	1	X	Variable x
f_2	1	0	X	Negation of x
f_3	1	1	1	Constant 1

Function	Logic gate	Symbol	Trooth tabel															
Negation $F = \bar{x}$	Invertor NOT		<table border="1"> <thead> <tr> <th>x</th> <th>f</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	x	f	0	1	1	0									
x	f																	
0	1																	
1	0																	
Identity $F = x$	BUFFER		<table border="1"> <thead> <tr> <th>x</th> <th>f</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> </tr> </tbody> </table>	x	f	0	0	1	1									
x	f																	
0	0																	
1	1																	
Logic multiplication $f = x_1 \cdot x_2$	AND		<table border="1"> <thead> <tr> <th>x₁</th> <th>x₂</th> <th>f</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	x ₁	x ₂	f	0	0	0	0	1	0	1	0	0	1	1	1
x ₁	x ₂	f																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
Negation of logic multiplication Shaffers function $f = \overline{x_1 \cdot x_2}$	NAND		<table border="1"> <thead> <tr> <th>x₁</th> <th>x₂</th> <th>f</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	x ₁	x ₂	f	0	0	1	0	1	1	1	0	1	1	1	0
x ₁	x ₂	f																
0	0	1																
0	1	1																
1	0	1																
1	1	0																

Logic addition $f = x_1 + x_2$	OR		<table border="1"> <thead> <tr> <th>x_1</th> <th>x_2</th> <th>f</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	x_1	x_2	f	0	0	0	0	1	1	1	0	1	1	1	1
x_1	x_2	f																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Negation of logic addition Pirs function $f = \overline{x_1 + x_2}$	NOR		<table border="1"> <thead> <tr> <th>x_1</th> <th>x_2</th> <th>f</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	x_1	x_2	f	0	0	1	0	1	0	1	0	0	1	1	0
x_1	x_2	f																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Additiion modulo 2 $f = x_1 \oplus x_2$	Exclusive OR XOR		<table border="1"> <thead> <tr> <th>x_1</th> <th>x_2</th> <th>f</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	x_1	x_2	f	0	0	0	0	1	1	1	0	1	1	1	0
x_1	x_2	f																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
Equivalence $f = \overline{x_1 \oplus x_2}$	Exclusive NOR XNOR		<table border="1"> <thead> <tr> <th>x_1</th> <th>x_2</th> <th>f</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	x_1	x_2	f	0	0	1	0	1	0	1	0	0	1	1	1
x_1	x_2	f																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Standard Representation of Logic Functions

Logic functions can be represented in 3 modes:

- graphically (truth table , Karnaugh map, logic circuit and timing diagram);
- analytically (canonical sum, canonical product, minimal sum and minimal product);
- as a list (minterm list, maxterm list).

Truth table

Traditionally the input combinations are arranged in rows in ascending binary counting order, and the corresponding output values are written in a column next to the rows. The general structure of a 3-variable truth table is shown below:

Row	X	Y	Z	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

The TT for n-variable logic function has 2^n rows.

Karnaugh Maps

A **Karnaugh map** is a graphical representation of a logic function's TT. Karnaugh maps for logic functions of 2,3 and 4 variables:

$x_1 \backslash x_2$	0	1
0	0	2
1	1	3

$x_1x_2 \backslash x_3$	00	01	11	10
0	0	2	6	4
1	1	3	7	5

$x_1x_2 \backslash x_3x_4$	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

The columns and rows are labeled using Gray code or reflected code – 00, 01, 11, 10.

Each cell, because of the Gray code, corresponds to an input combination that differs from each of its immediately adjacent neighbors in only one variable. E.g.: cells 5 and 13 in the 4-variable map differ only in the value of x .

Karnaugh Maps

	x1	x2	x3	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

		x1x2			
		00	01	11	10
x3	0	000	010	110	100
	1	001	011	111	101

		x1x2			
		00	01	11	10
x3	0	0	2	6	4
	1	1	3	7	5

		x1x2			
		00	01	11	10
x3	0			1	1
	1	1	1		

Canonical forms

Canonical sum

A n-variable **minterm** is a normal product term with n literals. There are 2^n such product terms. The literals are written in the minterm according to the following rule:

$$x_i = \begin{cases} x_i & \text{if } x_i = 1 \\ \bar{x}_i & \text{if } x_i = 0 \end{cases} \quad x_1 \bar{x}_2 x_3$$

The **canonical sum** of a logic function is a sum of the minterms corresponding to TT rows (input combination) for which the function produces a 1 output.

$$F = \bar{x}_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 x_3$$

Canonical product

A n-variable **maxterm** is a normal sum term with n literals. There are 2^n such sum terms. The literals are written in the maxterm according to the following rule:

$$x_i = \begin{cases} x_i & \text{if } x_i = 0 \\ \bar{x}_i & \text{if } x_i = 1 \end{cases} \quad \bar{x}_1 + x_2 + \bar{x}_3$$

The **canonical product** of a logic function is the product of the maxterms, corresponding to input combinations for which the function produces a 0 output.

$$F = (x_1 + x_2 + x_3) \cdot (x_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + x_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3)$$

	x1	x2	x3	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Implementation of logic functions

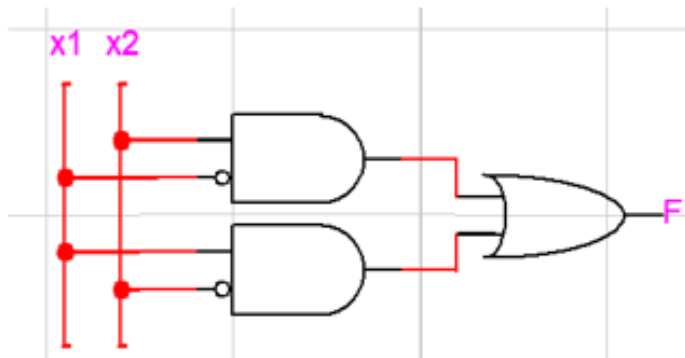
- Implementation of logic function means its realization with logic gates.
- The most used forms of implementation are NAND and NOR.

Cost (C)- the number of all inputs in logic gates. It is measured in Quines (Q).

Delay time (T_d) – The number of levels in the circuit or the maximal number of logic gates that cross the signal from input to output. It is measured in τ .

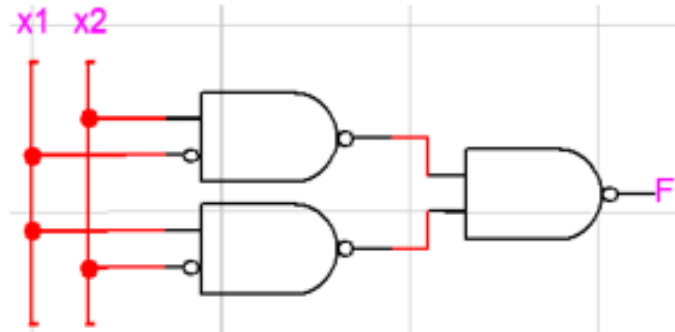
Examples

Ex: $F_{FDM} = \overline{x_1}x_2 + x_1\overline{x_2}$



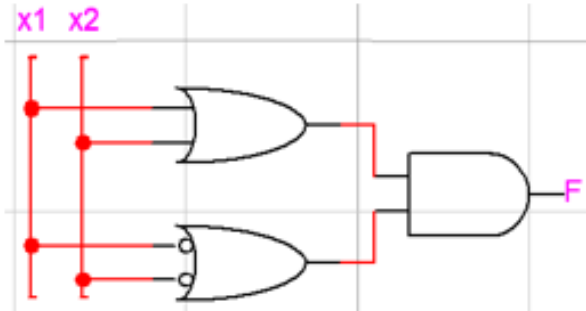
$C=6 \lambda$
 $T_d=2 \tau$

Ex: $F_{FDM} = \overline{\overline{x_1}x_2} + \overline{x_1\overline{x_2}} = \overline{x_1}x_2 \cdot x_1\overline{x_2}$



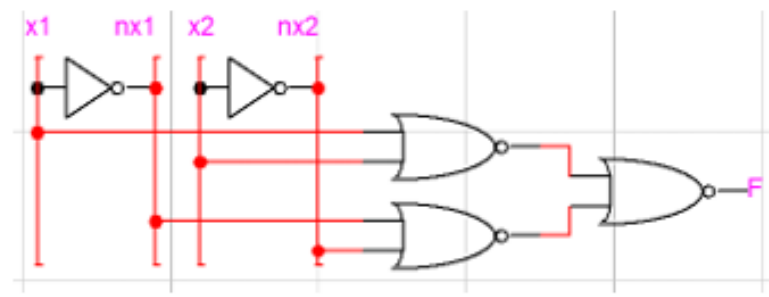
$C=6 \lambda$
 $T_d=2 \tau$

Ex. $F_{FCM} = (x_1+x_2) \cdot (\overline{x_1} + \overline{x_2})$



$C=6 \lambda$
 $T_d=2 \tau$

$$F_{FCM} = \overline{\overline{(x_1+x_2)} \cdot \overline{(\overline{x_1} + \overline{x_2})}} = \overline{\overline{(x_1+x_2)} + \overline{(\overline{x_1} + \overline{x_2})}}$$



$C=8 \lambda$
 $T_d=3 \tau$

Karnaugh Maps

Each input combination with a “1” in the truth table corresponds to a minterm in the logic function’s canonical sum. Since pairs of adjacent “1” cells in the Karnaugh map have minterms that differ in only one variable, the minterm pairs can be combined into a single product term (Theorem 5).

We can simplify a logic function by combining pairs of adjacent 1-cells (minterms) whenever possible and write a sum of product terms that covers all of the 1-cells.

In general, 2^i 1-cells may be combined to form a product term containing $n-i$ literals, where n is the number of variables in the function.

Graphically this rule means that we can circle rectangular sets of 2^i 1’s. We can determine the literals of the corresponding product terms directly from the map, for each variable we make the following determination.

- If a circle covers only areas of the map where the variable is “0” then the variable is complemented in the product term.
- If a circle covers only areas of the map where the variable is “1” – uncomplemented.
- If a circle covers both areas of the map “0” and “1”, then the variable does not appear in the product term.

Obs. The number of circles must be minimal and the number of 1’s in each circle -maximal.

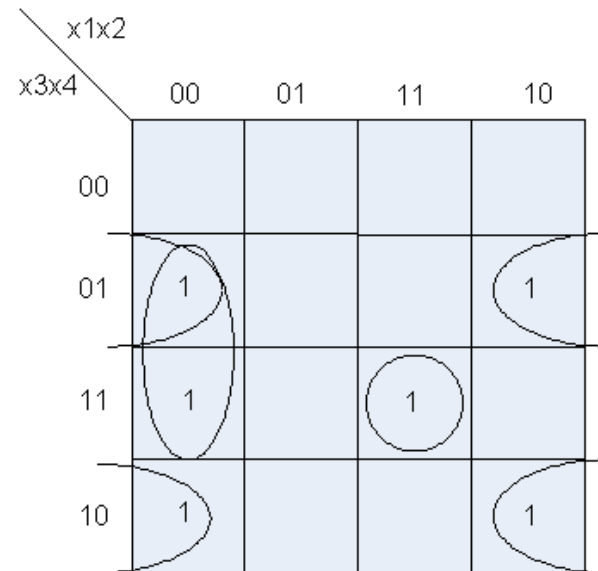
Examples

Example 1

$$F(x_1, x_2, x_3, x_4) = \sum (1, 2, 3, 9, 10, 15)$$

Minimal sum

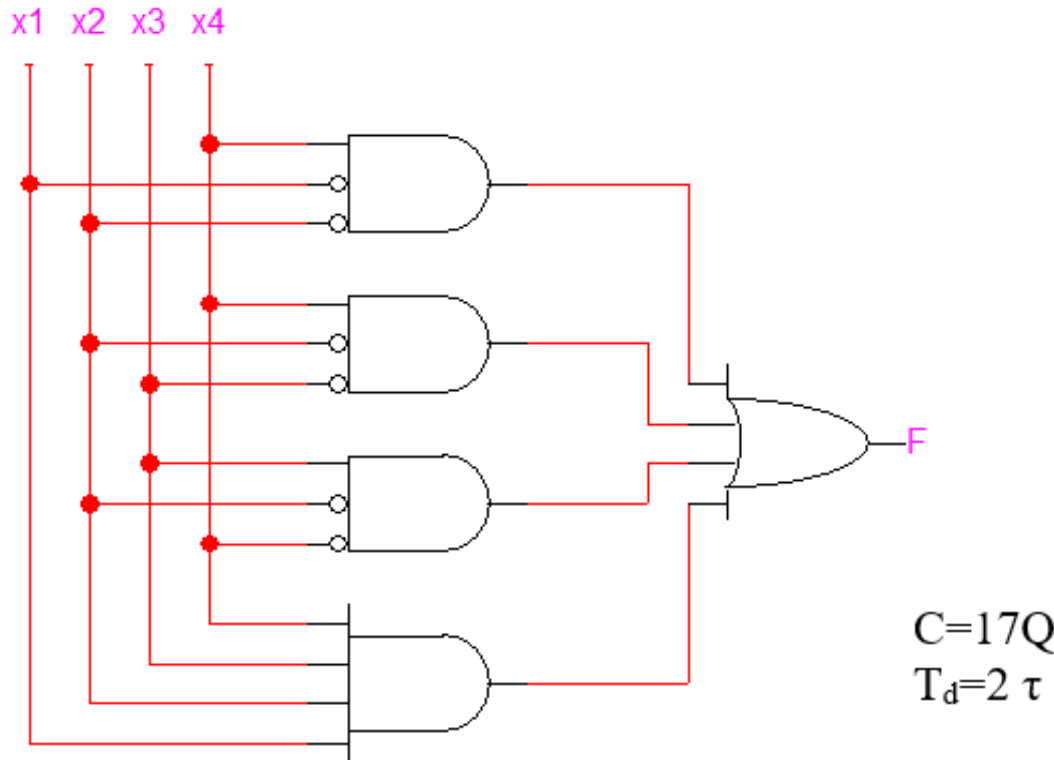
No	X1	X2	X3	X4	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1



$$F = \bar{x}_1\bar{x}_2x_4 + \bar{x}_2\bar{x}_3x_4 + \bar{x}_2x_3\bar{x}_4 + x_1x_2x_3x_4$$

Logic circuit

$$F = \bar{x}_1\bar{x}_2x_4 + \bar{x}_2\bar{x}_3x_4 + \bar{x}_2x_3\bar{x}_4 + x_1x_2x_3x_4$$



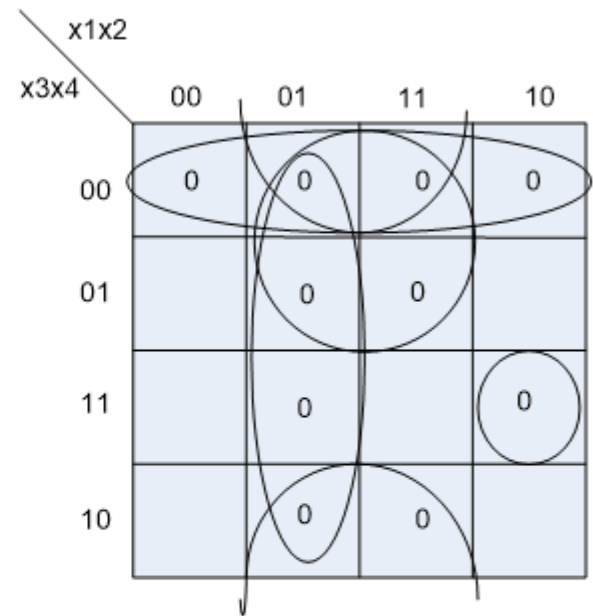
Cost (C)- the number of all inputs in logic gates. It is measured in Quines (Q).

Delay time (T_d) – The number of levels in the circuit or the maximal number of logic gates that cross the signal from input to output. It is measured in τ .

Example 2 $F(x_1, x_2, x_3, x_4) = \prod (0,4,5,6,7,8,11,12,13,14)$

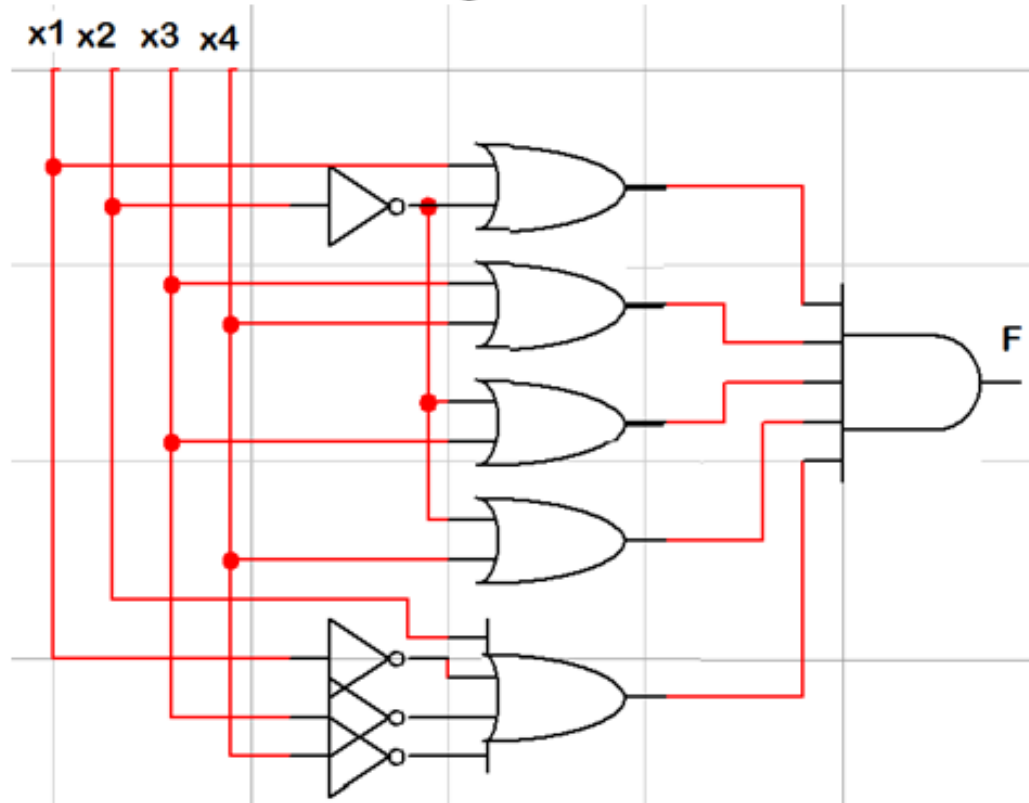
Minimal product

No	X1	X2	X3	X4	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1



$$F = (x_1 + \bar{x}_2) \cdot (x_3 + x_4) \cdot (\bar{x}_2 + x_3) \cdot (\bar{x}_2 + x_4) \cdot (\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4)$$

Logic circuit



“Don’t Care “ Input Combinations

Sometimes the specification of a combinational circuit is so that its output doesn’t matter for certain input combinations, called “don’t cares”. This may be true because the outputs really don’t matter when these input combinations occur or because these input combinations never occur in normal operation.

- Obs. 1. Allow d’s to be included when circling sets of 1’s to make the sets as large as possible. This reduces the number of variables.
- Obs. 2. Do not circle any sets that contain only d’s. It would unnecessarily increase its cost.

E.g.: $F(x_1, x_2, x_3, x_4) = \sum_{x_1, x_2, x_3, x_4} (2, 5, 9, 10) + *(0, 1, 4, 7, 8, 11, 14)$

x1x2

x3x4

	00	01	11	10
00	*	*		*
01	*	1		1
11		*		*
10	1		*	1

$$F = \bar{x}_1 \bar{x}_3 + x_1 \bar{x}_2 + \bar{x}_2 \bar{x}_4$$

Multiple Output Minimization

$$F_1 = V_1(0,1,2,4,6,7,8,10,15)$$

$$F_2 = V_1(0,2,4,7,8,9,10,15)$$

x_1x_2	x_3x_4 00	01	11	10
00	1	1		1
01	1			
11		1	1	
10	1	1		1

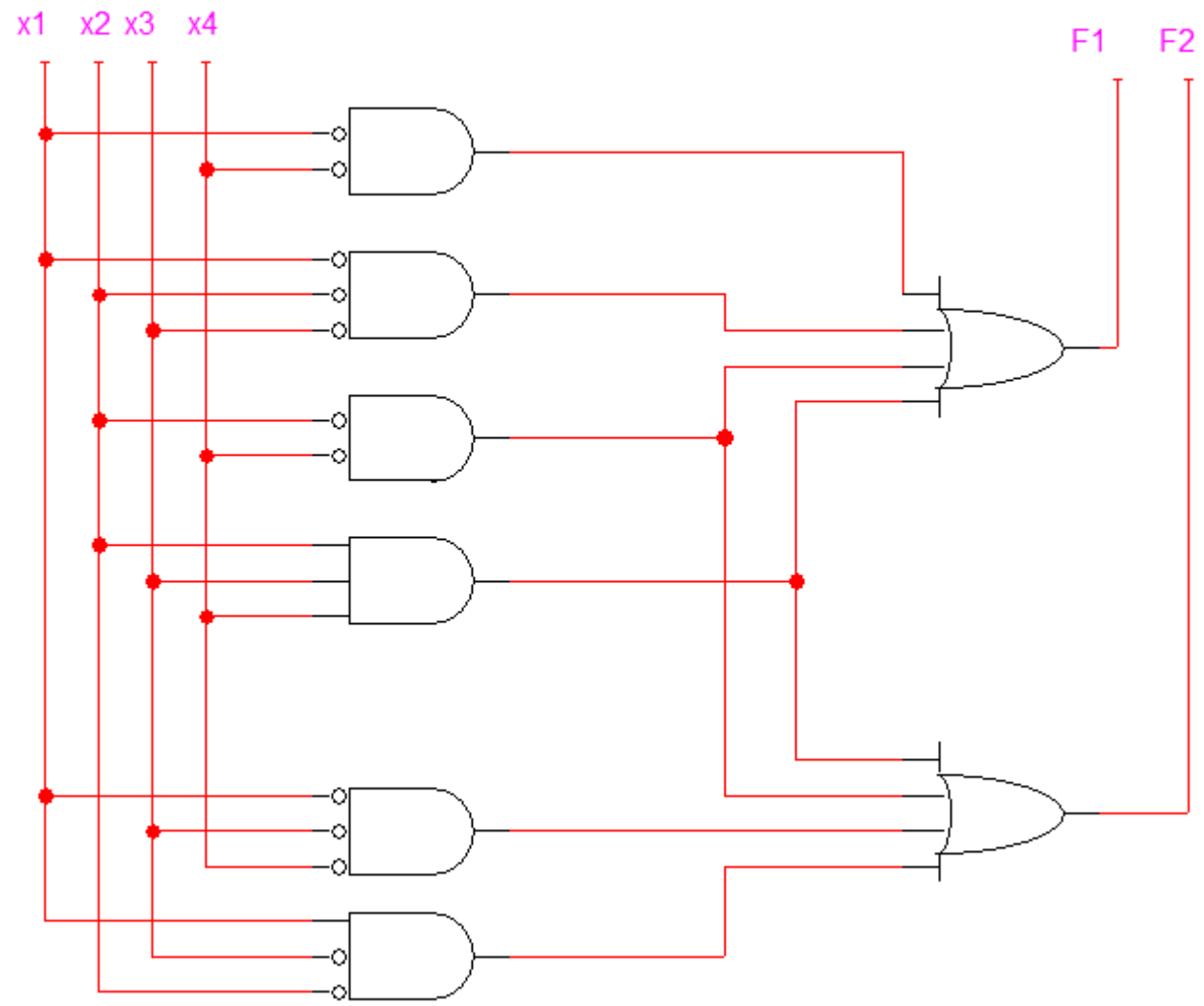
$$F_1 = \bar{x}_2\bar{x}_4 + \bar{x}_1\bar{x}_4 + x_2x_3x_4 + \bar{x}_1\bar{x}_2\bar{x}_3$$

x_1x_2	x_3x_4 00	01	11	10
00	1	1		1
01				1
11		1	1	
10	1			1

$$F_2 = \bar{x}_2\bar{x}_4 + \bar{x}_1\bar{x}_3\bar{x}_4 + x_2x_3x_4 + x_1\bar{x}_2\bar{x}_3$$

$$F_1 = \bar{x}_2\bar{x}_4 + \bar{x}_1\bar{x}_4 + x_2x_3x_4 + \bar{x}_1\bar{x}_2\bar{x}_3$$

$$F_2 = \bar{x}_2\bar{x}_4 + \bar{x}_1\bar{x}_3\bar{x}_4 + x_2x_3x_4 + x_1\bar{x}_2\bar{x}_3$$



Example

Diagrama Karnaugh pentru funcții de 5 variabile

		x1x2x3							
		000	001	011	010	110	111	101	100
x4x5	00	0	4	12	8	24	28	20	16
	01	1	5	13	9	25	29	21	17
	11	3	7	15	11	27	31	23	19
	10	2	6	14	10	26	30	22	18

		x1x2x3							
		000	001	011	010	110	111	101	100
x4x5	00	1			1	1			1
	01	1		1			1		1
	11		1		1	1	1	1	
	10	1	1		1	1		1	1

		x1x2x3							
		000	001	011	010	110	111	101	100
x4x5	00	1			1	1			1
	01	1		1			1		1
	11		1		1	1	1	1	
	10	1	1		1	1		1	1

Diagrama Karnaugh colorată și etichetată pentru gruparea termenilor:

- 5: Gruparea de 2 celule (001, 011) în rândul 01.
- 1: Gruparea de 2 celule (010, 110) în rândul 00.
- 3: Gruparea de 2 celule (000, 010) în coloana 00.
- 6: Gruparea de 2 celule (001, 101) în coloana 01.
- 4: Gruparea de 2 celule (011, 111) în coloana 01.
- 2: Gruparea de 2 celule (110, 100) în coloana 11.

$$F = \bar{x}_3 \bar{x}_5 + \bar{x}_2 x_3 x_4 + \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_2 \bar{x}_3 x_4 + x_2 x_3 \bar{x}_4 x_5 + x_1 x_2 x_3 x_5$$