COM 205 - Digital Logic Design
Boolean Algebra and Logic Gates -V

Assist. Prof. Özge ÖZTİMUR KARADAĞ

ALKÜ

Previously

- K-Map method
  - Two-variable

m_o	m_1
m_2	m_3

  - Three-variable

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

  - Four-variable

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

K-Map Method

- Five-variable Map

A=0

E

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

C

B

D

A=1

	00	01	11	10
00	m_{16}	m_{17}	m_{19}	m_{18}
01	m_{20}	m_{21}	m_{23}	m_{22}
11	m_{28}	m_{29}	m_{31}	m_{30}
10	m_{24}	m_{25}	m_{27}	m_{26}

- Each square in the A=0 map is adjacent to the corresponding square in the A=1 map, ie. m_4 is adjacent to m_{20}

K-Map

- In a n-variable map for k=0,1,2,…,n any 2^k adjacent squares represent an area that gives a term of n-k literals. n must be greater than k.
- When n=k, the entire area of the map is cobined to give the identity function.
- The relationship between the number of adjacent squares and the number of literals in the term:

k	2^k	n=2	n=3	n=4	n=5
0	1	2	3	4	5
1	2	1	2	3	4
2	4	0	1	2	3
3	8		0	1	2
4	16			0	1
5	32				0

K-Map

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

	00	01	11	10
00	m_{16}	m_{17}	m_{19}	m_{18}
01	m_{20}	m_{21}	m_{23}	m_{22}
11	m_{28}	m_{29}	m_{31}	m_{30}
10	m_{24}	m_{25}	m_{27}	m_{26}

- Simplify the following Boolean Function:
- $F(A,B,C,D,E)=\sum(0,2,4,6,9,13,21,23,25,29,31)$

A'B'E'

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

BD'E

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

ACE

F=A'B'E'+BD'E+ACE

Product of Sums Simplification

- 1s placed in the squares of the map represent →minterms of the function
- 0s placed in the squares of the map represent →Complement of the function, F’.
- The complement of F’ gives F. By DeMorgan’s theorem the function obtained in this way is in product of sums form.

Product of Sums Simplification

F=?

F'=?

- Simplify the following Boolean Function into
  a. Sum of products form
  b. Product of sums form
  - $F(A,B,C,D) = \sum(0,1,2,5,8,9,10)$

	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

a. $F=B'D'+B'C'+A'C'D$

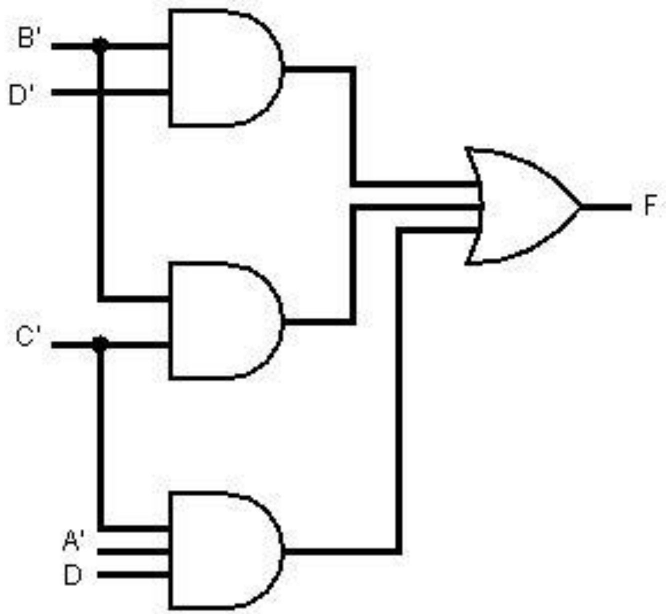
$F'=AB+CD+BD'$

DeMorgan

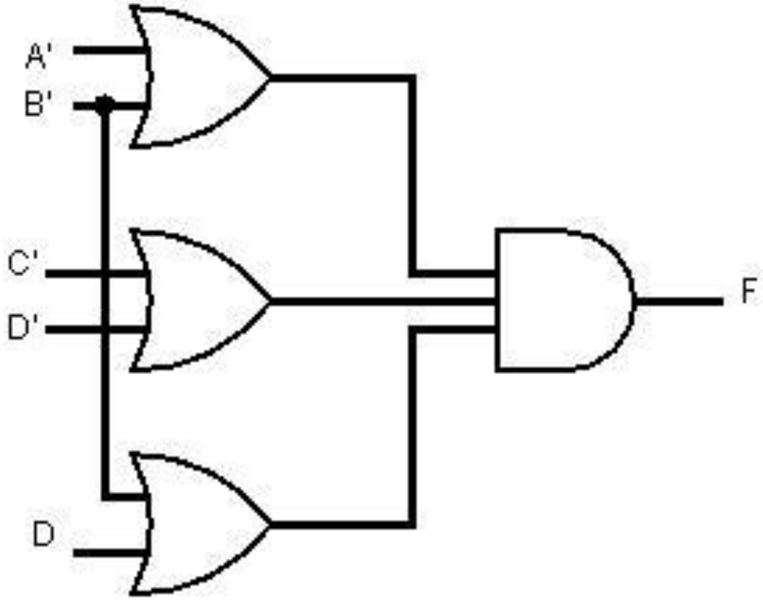
b. $F=(A'+B')(C'+D')(B'+D)$

Product of Sums Simplification

a. F=B’D’+B’C’+A’C’D



b. F=(A’+B’)(C’+D’)(B’+D)



Implementation of the function in a standard form is said to be a two-level implementation

Product of Sums Simplification

- Given the truth table of F, Express the function in sum of minterms form and product of maxterms form.

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$F(x,y,z)=\sum(?)$ 1,3,4,5,6

$F(x,y,z)=\prod(?)$ 0,2,7

	00	01	11	10
0	0	1	1	0
1	1	1	0	1

Sum of products: ? F=xz'+x'z+y'z

Product of sums: ? F'=x'y'z'+x'yz'+xyz

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

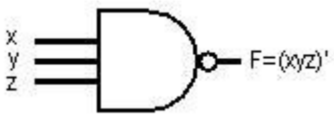
Product of Sums Simplification

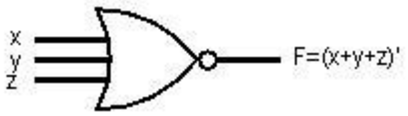
- To enter a function expressed in product of sums form into the map, use the complement of the function to find the squares that are to be marked by 0’s.
- Ex: F=(A’+B’+C’)(B+D)
- F’=ABC+B’D’
  - Put 0’s to those minterms and 1’s to remaining squares.

	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	1	1	0	0
10	0	1	1	0

NAND and NOR Implementation

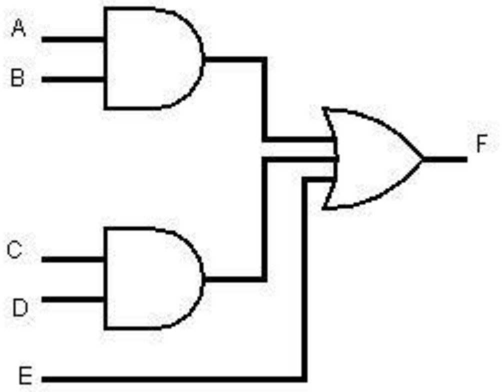
- Digital circuits are frequently constructed with NAND or NOR gates rather than with AND and OR gates. Because NAND and NOR gates are easier to fabricate with electronic components.
- So it is important that an AND-OR-NOT implementation can be converted to an equivalent NAND or NOR implementation.
- Graphical symbols:

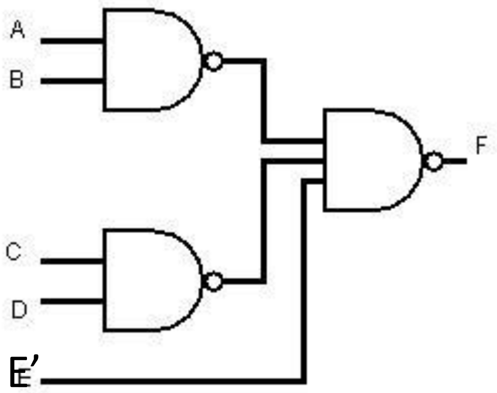




NAND Circuits

- To implement a Boolean function with NAND gates obtain the simplified Boolean function in terms of Boolean operators and then convert the function to NAND logic.
- F=AB+CD+E





NAND Circuits

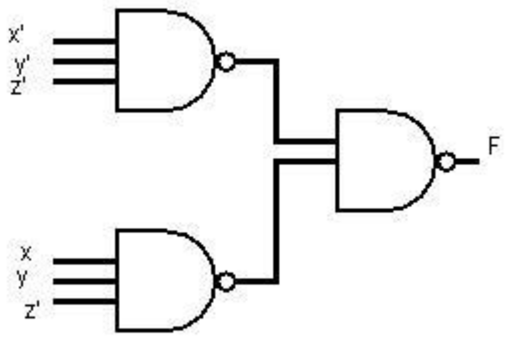
- Steps to follow to implement a Boolean Function by NAND gates:
- WAY 1:
    - Simplify the function in sum of products form.
    - For each product term with at least two literals put a NAND gate and connect the literals as NAND input.
    - Using AND-invert or invert-OR gates, connect the output of level-one to the input of level two with NAND gates.
    - Implement a single literal by means of an inverter or use its complement to connect as an input to second level NAND gate.

- WAY 2:
  - Simplify the function in product of sums form. Implement F’ using WAY 1. On the third level take its complement to obtain F.

- Ex: Implement the following function using NAND gates $F(x,y,z)=\sum(0,6)$
  - WAY 1:

	00	01	11	10
0	1	0	0	0
1	0	0	0	1

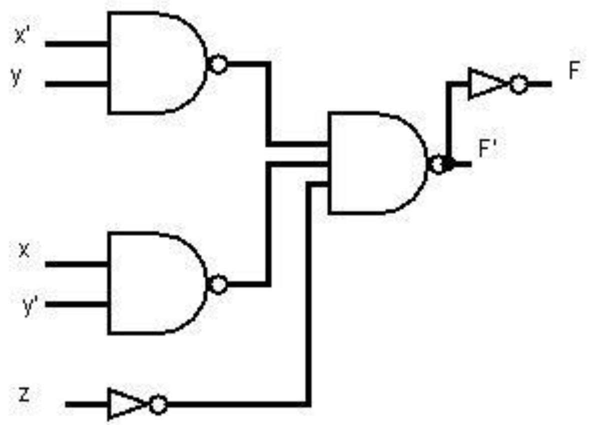
F=x’y’z’+xyz’



- Ex: Implement the following function using NAND gates F(x,y,z)=$\sum(0,6)$
  - WAY 2:

	00	01	11	10
0	1	0	0	0
1	0	0	0	1

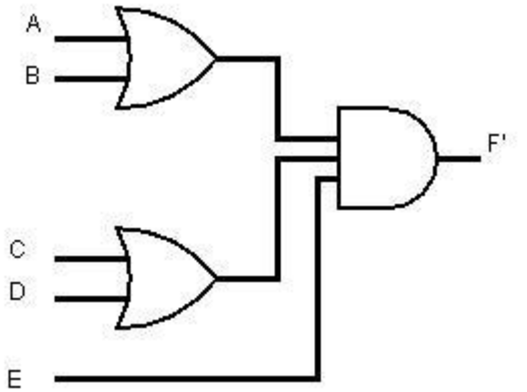
F’=x’y+xy’+z

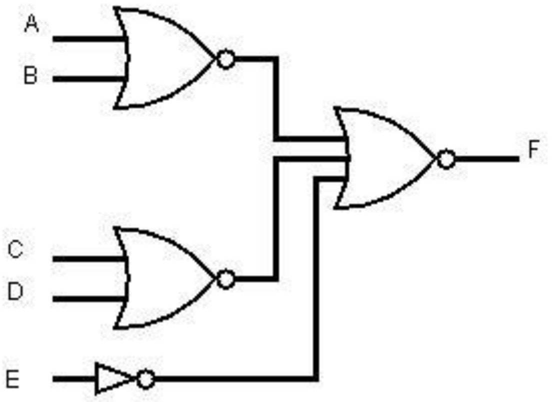


NOR Implementation

- The NOR Operation is the dual of the NAND Operation. Therefore, all procedures and rules for NOR logic are the duals of the corresponding procedures and rules developed for NAND logic. In order to implement a Boolean function by NOR gates,

1. The simplified function is represented in product of sums form.
2. OR, AND and Invert gates are converted to NOR-NOR gates.





NOR Implementation

- Way 1:
  - Simplify the function in product of sums form.
  - For each sum term put a NOR gate and connect the literals into its input.
  - Implement the second level by OR-invert or invert-AND gates by connecting the output of level-one as input.
  - For each term with a single literal either use a single input NOR gate or an inverter or take its complement and connect it directly to the second level.

NOR Implementation

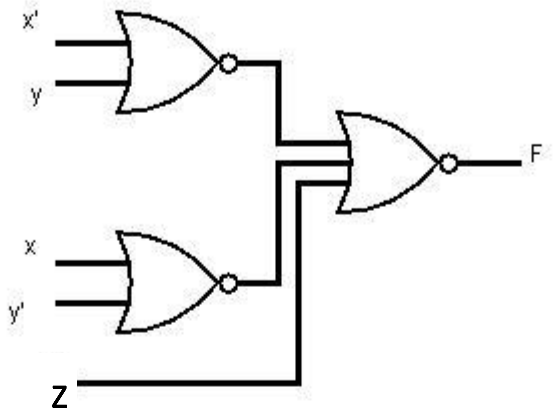
- Way 2:
  - Complement the function in product of sums form. (to obtain it from map take 0s and take the complement of function)
  - F’: two levels, F: three levels
  - In order to obtain F’ in product of sums form from map take 1’s from map.

NOR Implementation

- Ex: F(x,y,z)=$\sum(0,6)$ Implement the function using NOR gates.

	00	01	11	10
0	1			
1				1

- F'=x'y+xy'+z
- F=(x+y')(x'+y)z'

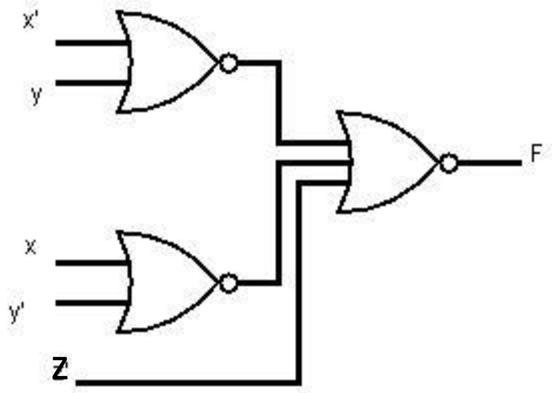


NOR Implementation

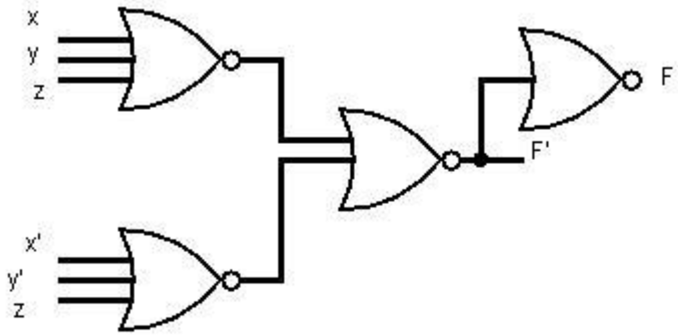
- Ex: F(x,y,z)=$\sum(0,6)$ Implement the function using NOR gates.

	00	01	11	10
0	1			
1				1

- F’=x’y+xy’+z
- F=(x+y’)(x’+y)z’



Alternatively:
F=x’y’z’+xyz’
F’=(x+y+z)(x’+y’+z)



Rules for NAND and NOR Implementations

	Function to be simplified	Standard form to be used	How to obtain	Implementation	#levels for F
a	F	SOP	Grouping of 1s in map	NAND	2
b	F’	SOP	Grouping of 0s in map	NAND	3
c	F	POS	Complement of F’ in (b)	NOR	2
d	F’	POS	Complement of F in (a)	NOR	3