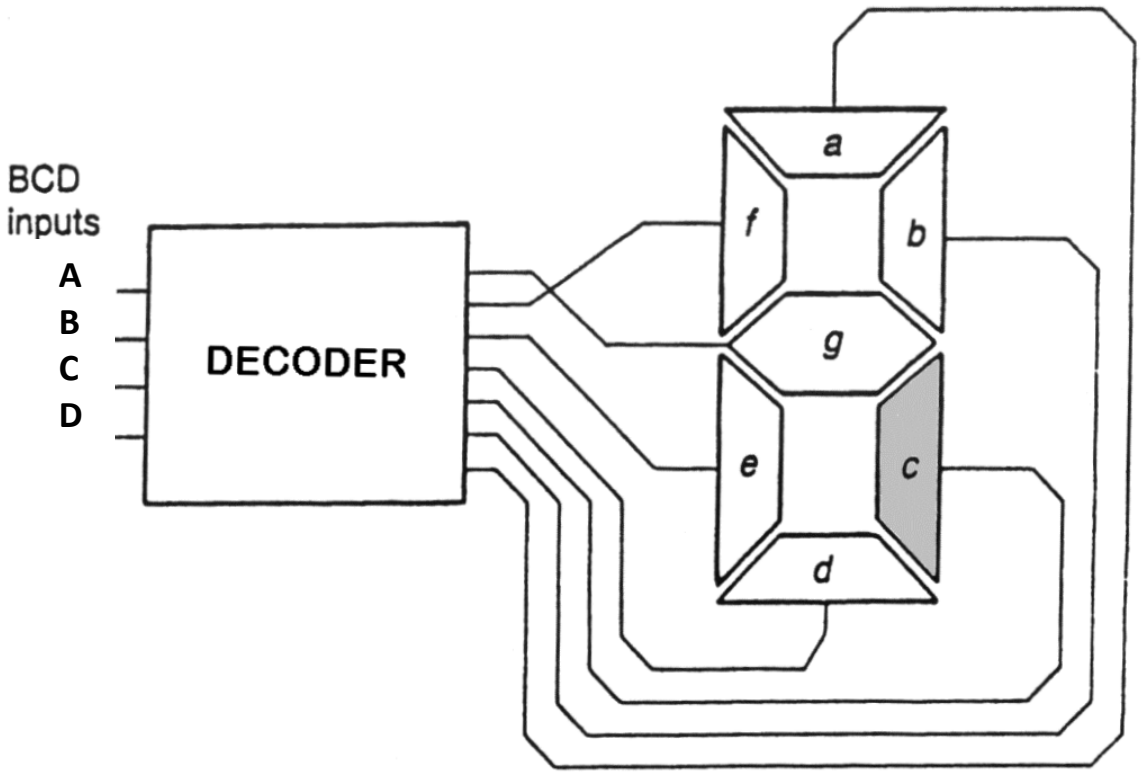
BCD to 7-segment display conversion



Recall: Binary Coded Decimal (BCD)

- SIMPLE CONVERSION TO DECIMAL used for driving a human interfaces e.g. Displays.
- **EACH DIGIT** OF DECIMAL NUMBER IS REPLACED BY **4 BITS** IN BINARY CODE (normal BCD):

e.g. $\boxed{4}\boxed{5}_{10}$

0100 0101

Hence $45_{10} = 01000101_{BCD}$

In our assignment, we need to perform the conversion of digits from 0 to 9

Recall: Binary Coded Decimal (BCD)

- **EACH DIGIT** OF DECIMAL NUMBER IS REPLACED BY **4 BITS** IN BINARY CODE (normal BCD):

$$\boxed{5}_{10}$$

0101

4-bits (4-variable) BCD input ➡ $0101_{BCD} = 5_{10}$ ⬅ Image on the 7-segment display

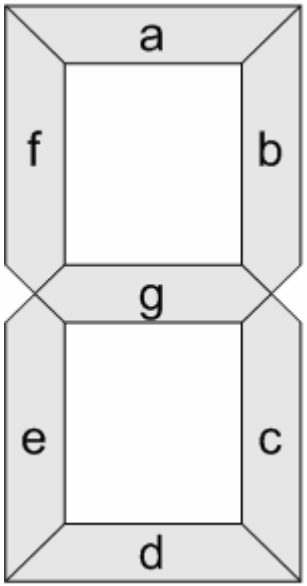
In our assignment, we need to perform the conversion of digits from 0 to 9

Useful and not useful combinations of input variables

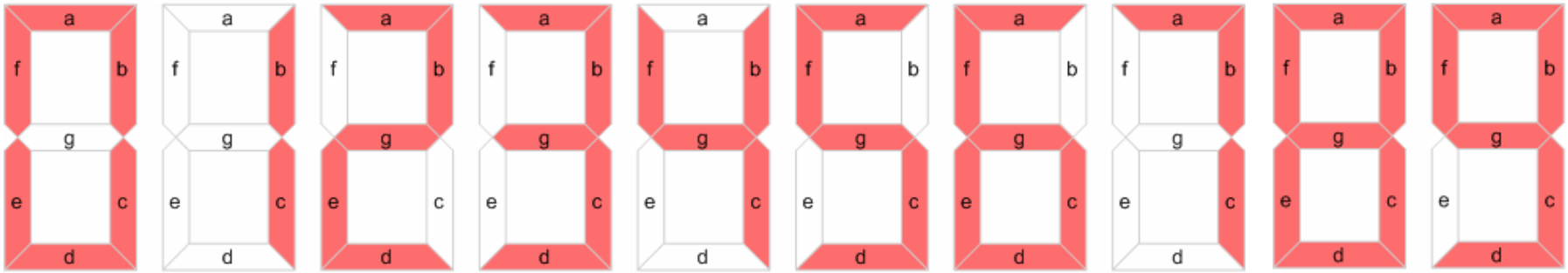
Decimal	BCD				
	W	X	Y	Z	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
	1	0	1	0	Don't care .....
	1	0	1	1	Don't care .....
	1	1	0	0	Don't care .....
	1	1	0	1	Don't care .....
	1	1	1	0	Don't care .....
	1	1	1	1	Don't care .....

These combinations are never used

Digits convention



Assume following representation of the digits:



Check the condition of the segment '**C**' for each digit!

Composing truth table

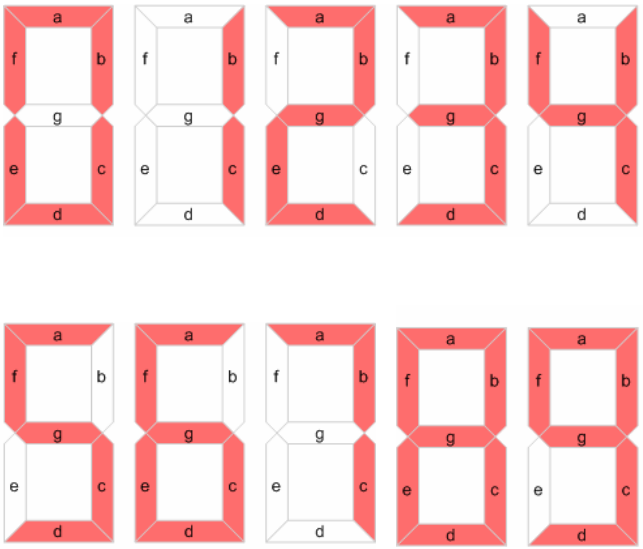
Truth table for segment '**C**' :

When this segment is lit up the output F is '1', when not the output F is '0'.

Decimal digit	BCD inputs				Output for segment c
	A	B	C	D	F
0	0	0	0	0	**1**
1	0	0	0	1	**1**
2	0	0	1	0	**0**
3	0	0	1	1	**1**
4	0	1	0	0	**1**
5	0	1	0	1	**1**
6	0	1	1	0	**1**
7	0	1	1	1	**1**
8	1	0	0	0	**1**
9	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**

Composing truth table

Decimal digit	BCD inputs				Output for segment c
	A	B	C	D	F
0	0	0	0	0	**1**
1	0	0	0	1	**1**
2	0	0	1	0	**0**
3	0	0	1	1	**1**
4	0	1	0	0	**1**
5	0	1	0	1	**1**
6	0	1	1	0	**1**
7	0	1	1	1	**1**
8	1	0	0	0	**1**
9	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**



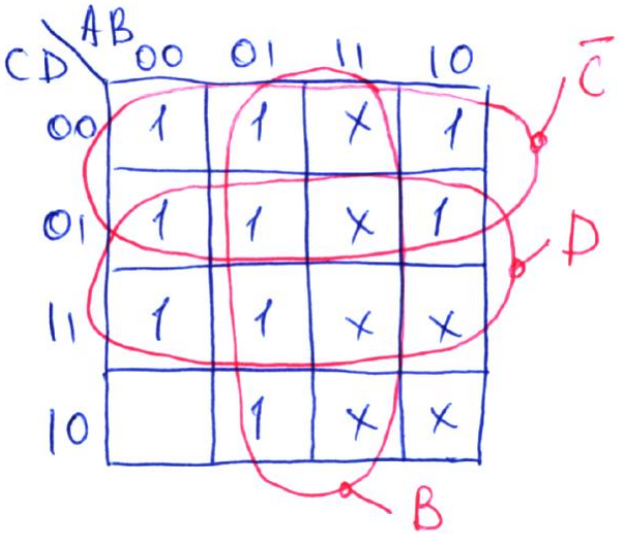
Segment '**C**' is lit up for all digits except '2'.

Karnaugh map for the 1-st canonical form

• Load all ‘1’ and ‘x’ from the truth table output into the relevant cells of the 4-variable Karnaugh map.

•Keep the ABCD order as in the truth table.

Decimal digit	BCD inputs				Output for segment c
	A	B	C	D	F
0	0	0	0	0	**1**
1	0	0	0	1	**1**
2	0	0	1	0	**0**
3	0	0	1	1	**1**
4	0	1	0	0	**1**
5	0	1	0	1	**1**
6	0	1	1	0	**1**
7	0	1	1	1	**1**
8	1	0	0	0	**1**
9	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**

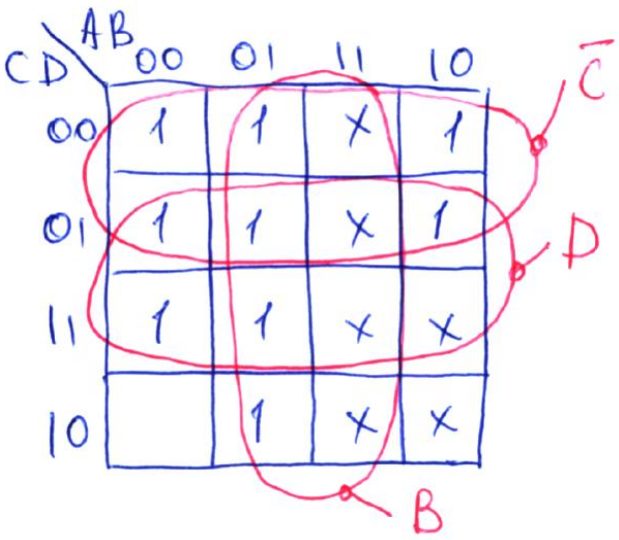


Karnaugh map for the 1-st canonical form

• Loop around the groups of ‘1’s and ‘x’s where appropriate. Aim at making larger groups, but remember that each group can consist of 2, 4, 8, or 16 terms (‘1’ and ‘x’). Groups of 3 or 6 terms are not allowed.

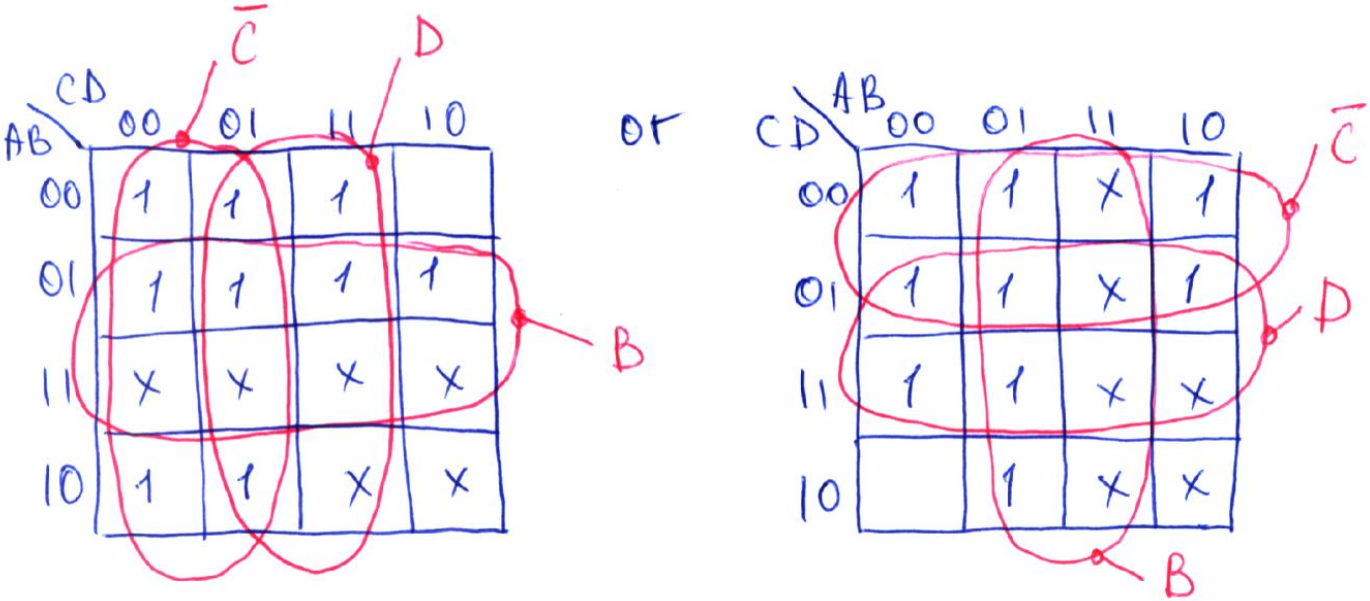
• Eliminate those loops, which do not have unique elements (i.e. completely overlapped by other loops). Remember that you don’t need to use all of ‘don’t care’ terms (‘x’) in the looping.

• Minimise the expression inside each loop, following the general rule.



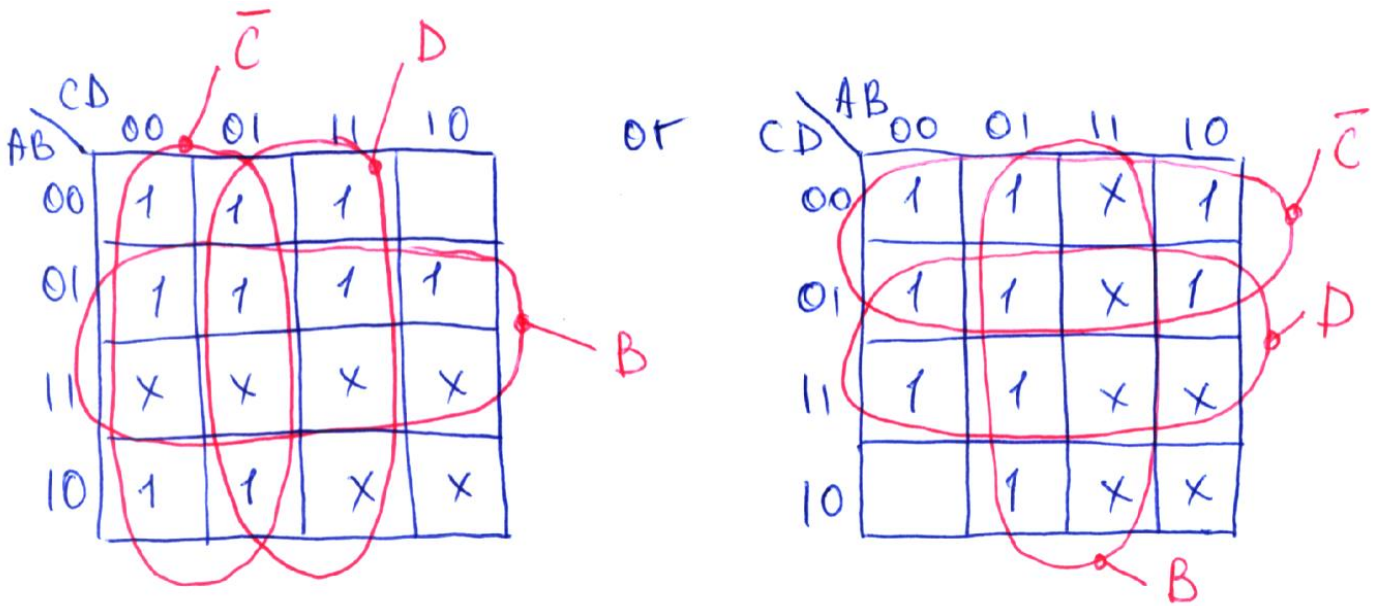
Karnaugh map for the 1-st canonical form

- There could be several options for building the Karnaugh map



1-st canonical form

- Compose sum of products (SOP) out of all loops minimised expressions.
- This is the final minimised expression.



$$F = B + \bar{C} + D$$

SOP into NAND gates

Transfer the obtained SOP expression in the form suitable for NAND gate implementation. For that apply the de Morgan's theorems twice:

$$F = B + \overline{C} + D$$

$$F' = (B + \overline{C} + D)' = \overline{B} \cdot C \cdot \overline{D}$$

$$F'' = F = (\overline{B} \cdot C \cdot \overline{D})'$$

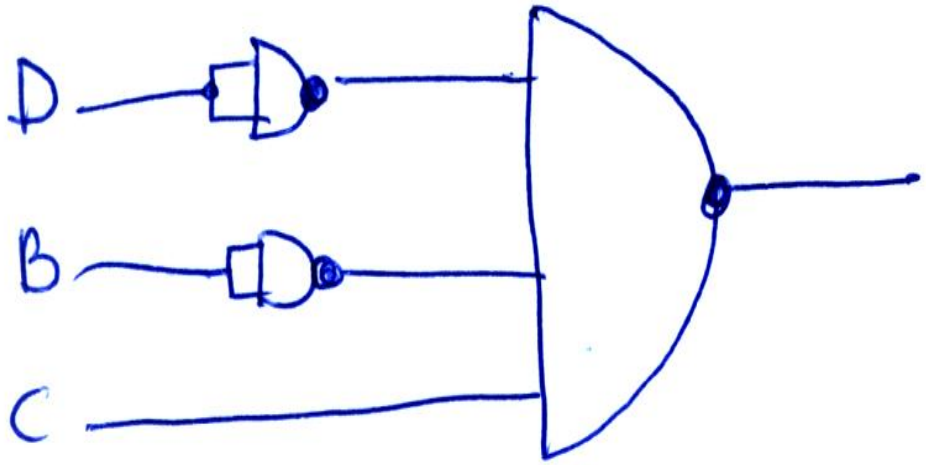
SOP into NAND gates

Draw the circuit diagram using NAND gates only.

$$F = B + \bar{C} + D$$

$$F' = (B + \bar{C} + D)' = \bar{B} \cdot C \cdot \bar{D}$$

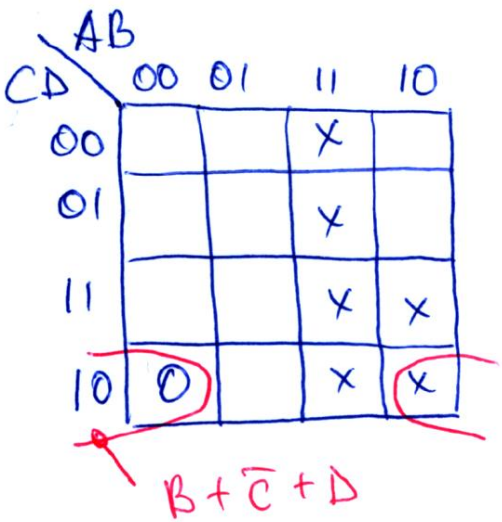
$$F'' = F = (\bar{B} \cdot C \cdot \bar{D})'$$



Karnaugh map for the 2-nd canonical form

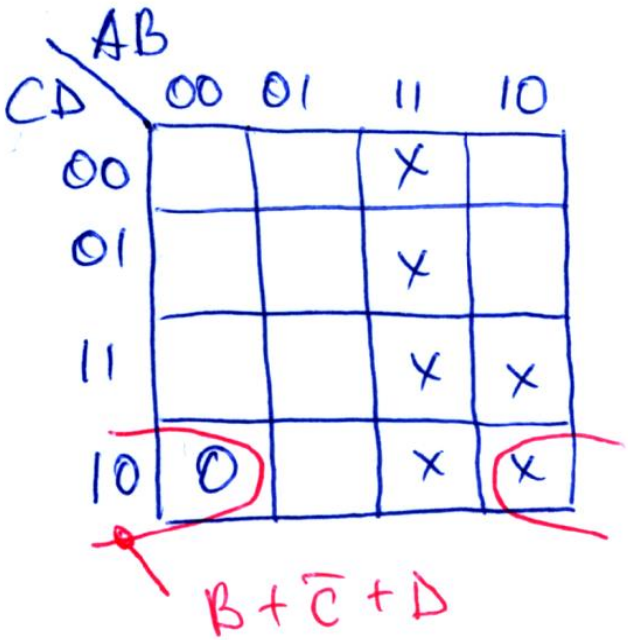
• Collect ‘0’ and ‘x’ in the standard 4-variable map. Loop around the ‘0’ s and ‘x’ as appropriate. Remember that you don’t need to use all ‘x’.

Decimal digit	BCD inputs				Output for segment c
	A	B	C	D	F
0	0	0	0	0	**1**
1	0	0	0	1	**1**
2	0	0	1	0	**0**
3	0	0	1	1	**1**
4	0	1	0	0	**1**
5	0	1	0	1	**1**
6	0	1	1	0	**1**
7	0	1	1	1	**1**
8	1	0	0	0	**1**
9	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**



2-nd canonical form

- Minimize expression inside each loop using the standard rules for the 2-nd canonical form.
- Form product of sum (POS) expression out of individual minimised expressions.



- This is the final minimised expression for the 2-nd form: $F = B + \bar{C} + D$

2-nd canonical form

- There could be several options for building the Karnaugh map

AB \ CD	00	01	11	10
00				0
01				
11	x	x	x	x
10			x	x

$B+\bar{C}+D$

or

CD \ AB	00	01	11	10
00			x	
01			x	
11			x	x
10	0		x	x

$B+\bar{C}+D$

$$F = B+\bar{C}+D$$

POS into NOR gates

Transfer the obtained minimised POS expression in a form suitable for the NOR gate implementation. For that, apply the de Morgan's theorem twice:

$$F = B + \bar{C} + D$$

$$F' = (B + \bar{C} + D)'$$

$$F'' = ((B + \bar{C} + D)')'$$

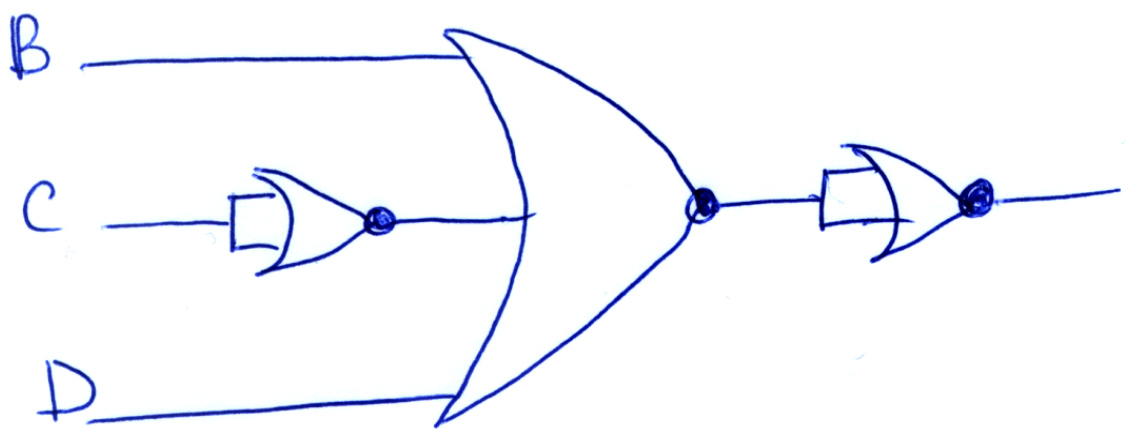
POS into NOR gates

$$F = B + \bar{C} + D$$

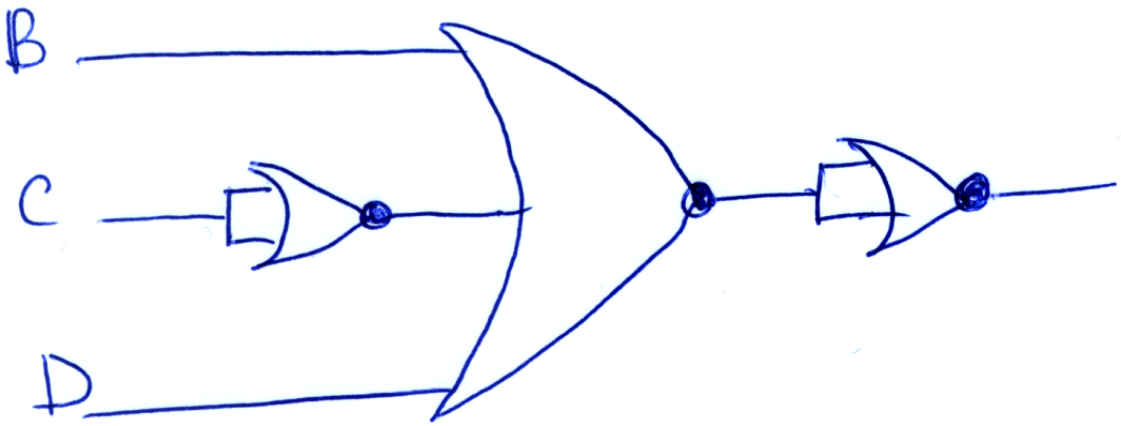
$$F' = (B + \bar{C} + D)'$$

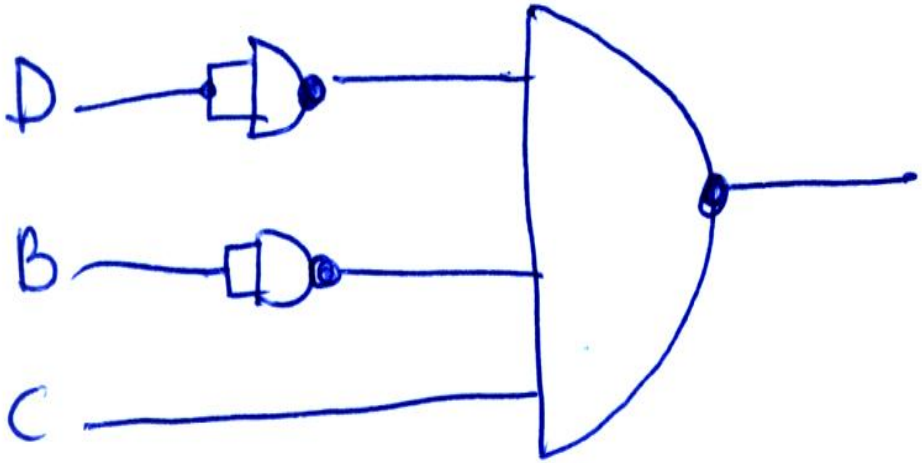
$$F'' = ((B + \bar{C} + D)')'$$

Draw the circuit diagram:



Compare NAND and NOR circuits



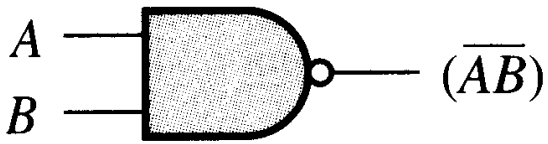


Conversion to one type of gates

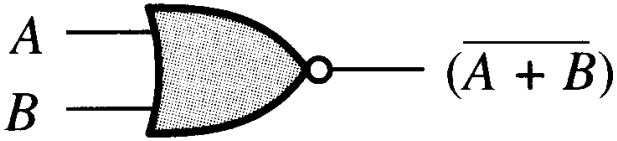
One of the important implications of De Morgan's theorems is that any logic function can be implemented solely with **NAND gates** (AND + inverter)

or

with **NOR gates** (OR + inverter).



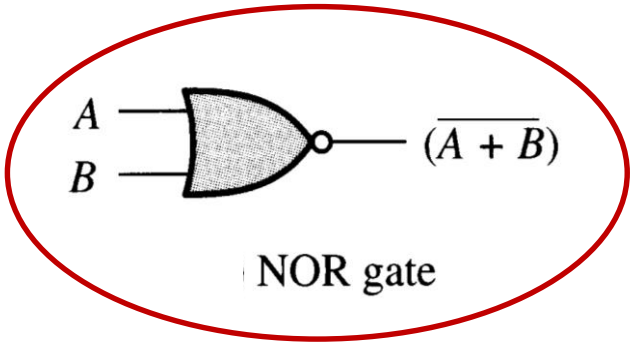
(a) NAND gate



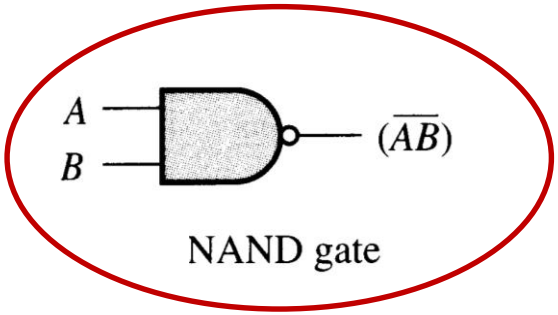
(b) NOR gate

Conversion to one type of gates

How is this done in practice and which type of the gate is preferable?



or



?

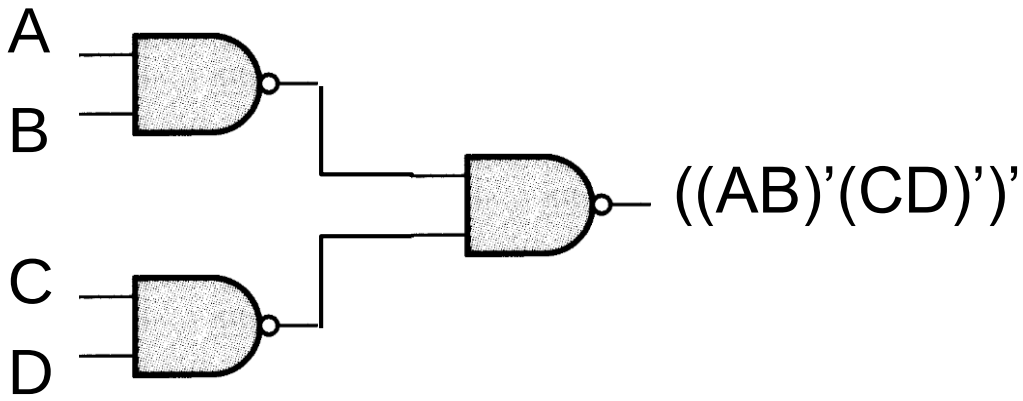
Conversion to one type of gates: SOP

First canonical form or sum of products (**SOP**) can be conveniently represented with **NAND** gates only:

$$F = AB + CD$$

Applying De Morgan's theorem: $F' = (AB)'(CD)'$ Applying De Morgan's theorem again:

$$F'' = F = ((AB)'(CD)')'$$

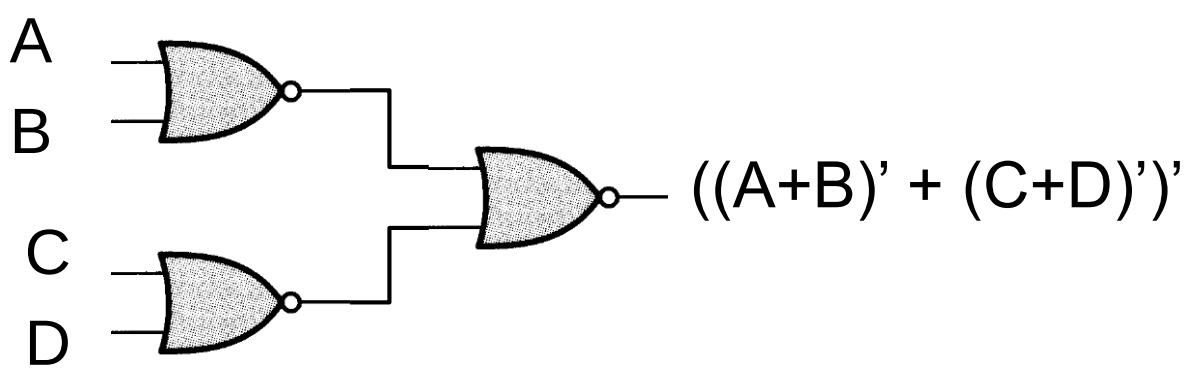


Conversion to one type of gates: POS

Second canonical form or product of sum (**POS**) can be conveniently represented with **NOR** gates only:

$$F = (A+B)(C+D)$$

Applying De Morgan's theorem: $F' = (A+B)' + (C+D)'$

Applying De Morgan's theorem again:

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