PDS0101

Introduction to Digital Systems

Boolean Algebra and Logic Simplification

MMU
MULTIMEDIA UNIVERSITY

Lecture outcome

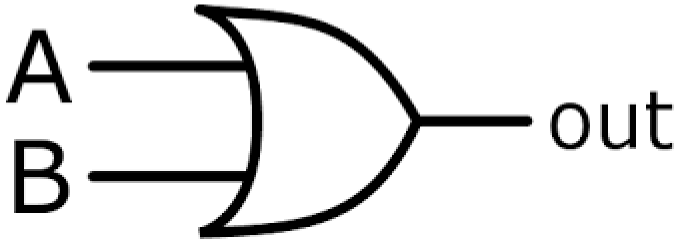
- By the end of today’s lecture you should know
  - the basic rules on boolean algebra
  - DeMorgan’s theorems
  - how to perform simplification of logic circuit diagrams and boolean expressions using the above

Boolean algebra

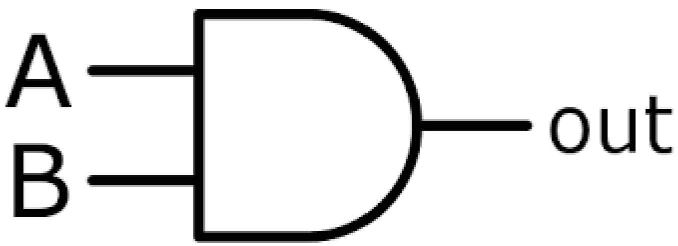
- Boolean algebra is the mathematics of digital systems.
- A basic knowledge of boolean algebra is indispensable to the study and analysis of logic circuits.
- *Variable*, *complement* and *literal* are terms used in boolean algebra.
- A literal is a variable or its complement
  - A variable is a symbol used to represent logical quantity.
    - Any single variable can have a 1 or a 0 value.
  - The complement represents the inverse of a variable and is indicated with an overbar. Thus, the complement of A is $\overline{A}$.
  - To simplify entry, some use the apostrophe to indicate complement → the complement of B is B'

Boolean addition and multiplication

- Boolean addition is equivalent to OR operation



- Boolean multiplication is equivalent to AND operation



Boolean algebra rules

- In algebra, we learned rules or laws
- For example the *commutative law of addition*

$$A + B = B + A$$

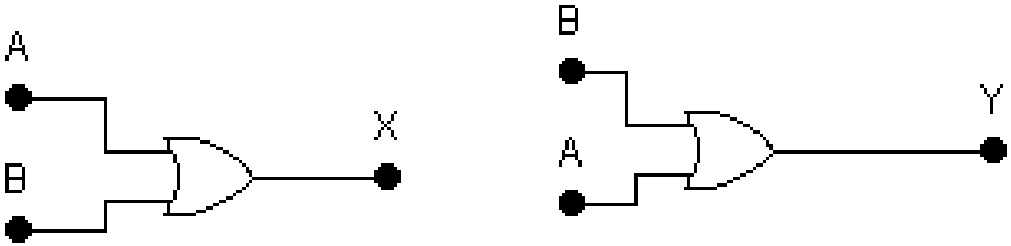
  - where A and B are any whole number (in 6th grade) or A and B are any real number (in 9th grade)

- In 1860 George Boole developed an algebra where A and B were only allowed to be true or false → this is called *Boolean algebra* and is used in digital electronics.
- Boolean algebra laws and rules are similar to the algebra, but only 1 or 0 is allowed for the values in variables

- The following slides show basic rules of boolean algebra

Commutative law of addition

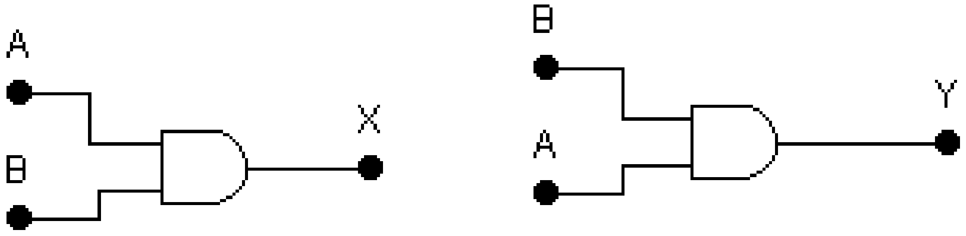
- A+B = B+A
- The order of OR-ing does not matter



X = Y

Commutative law of multiplication

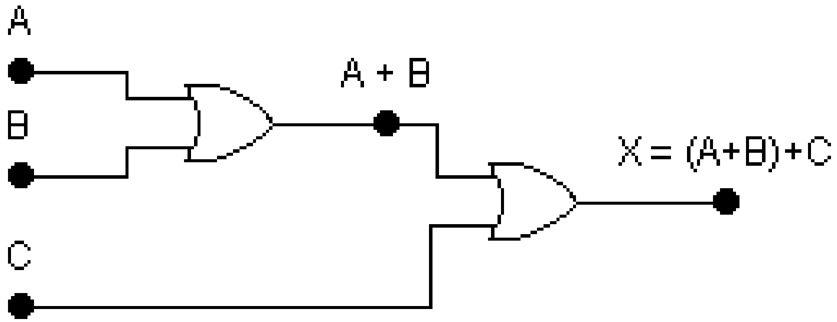
- AB = BA
- The order of ANDing does not matter.

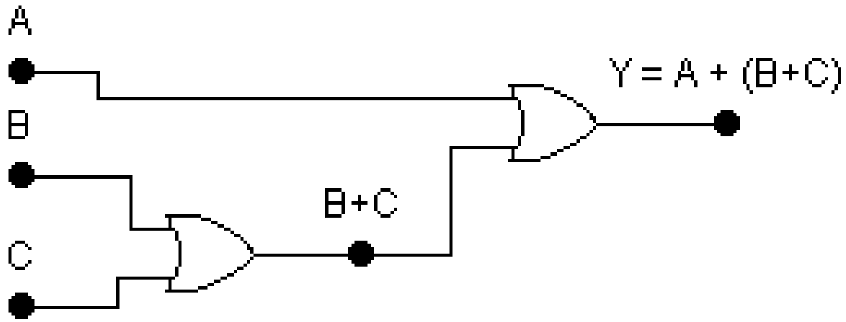


X = Y

Associative law of addition

- A + (B + C) = (A + B) + C
- The grouping of ORed variables does not matter

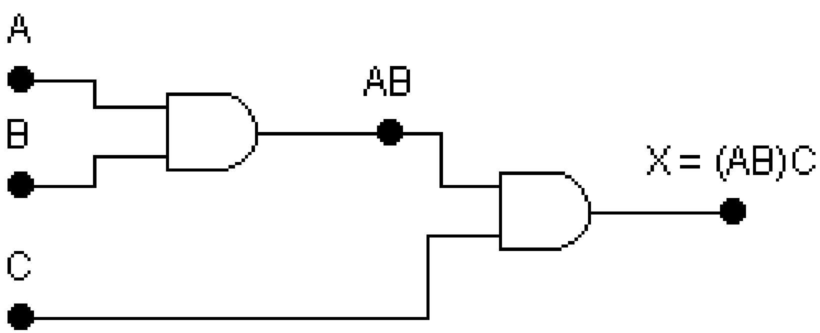


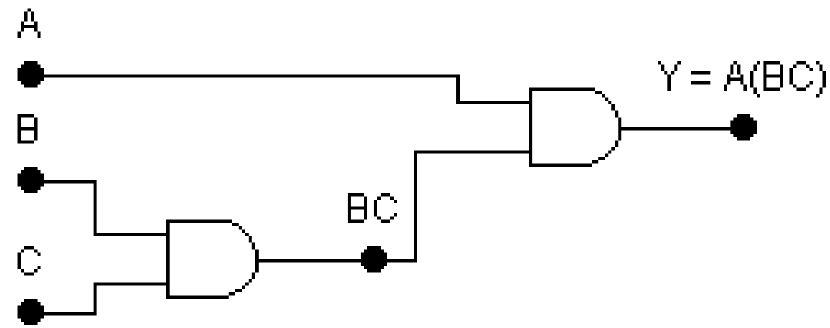


X = Y

Associative law of multiplication

- A(BC) = (AB)C
- The grouping of ANDed variables does not matter

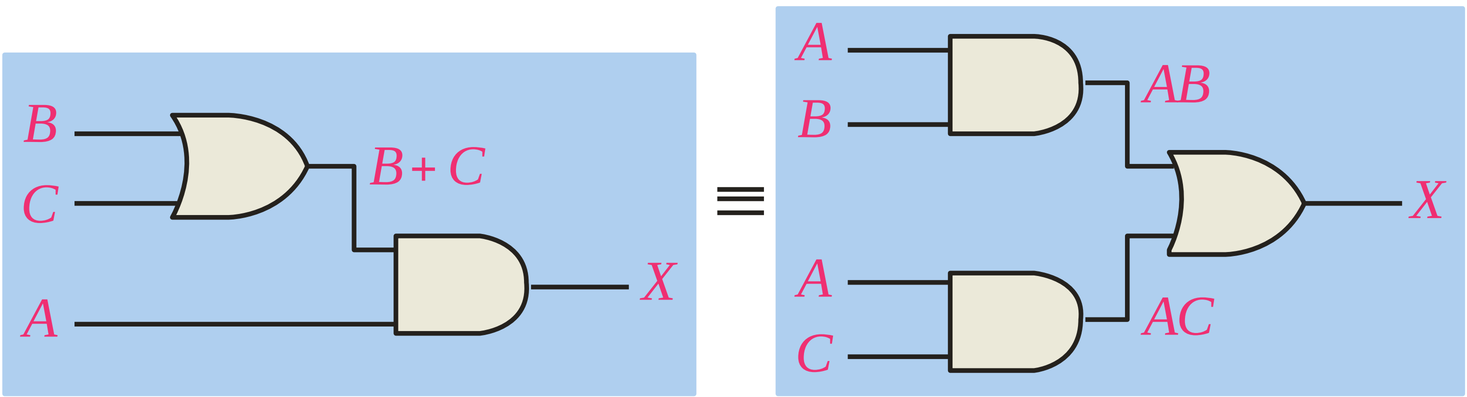




X = Y

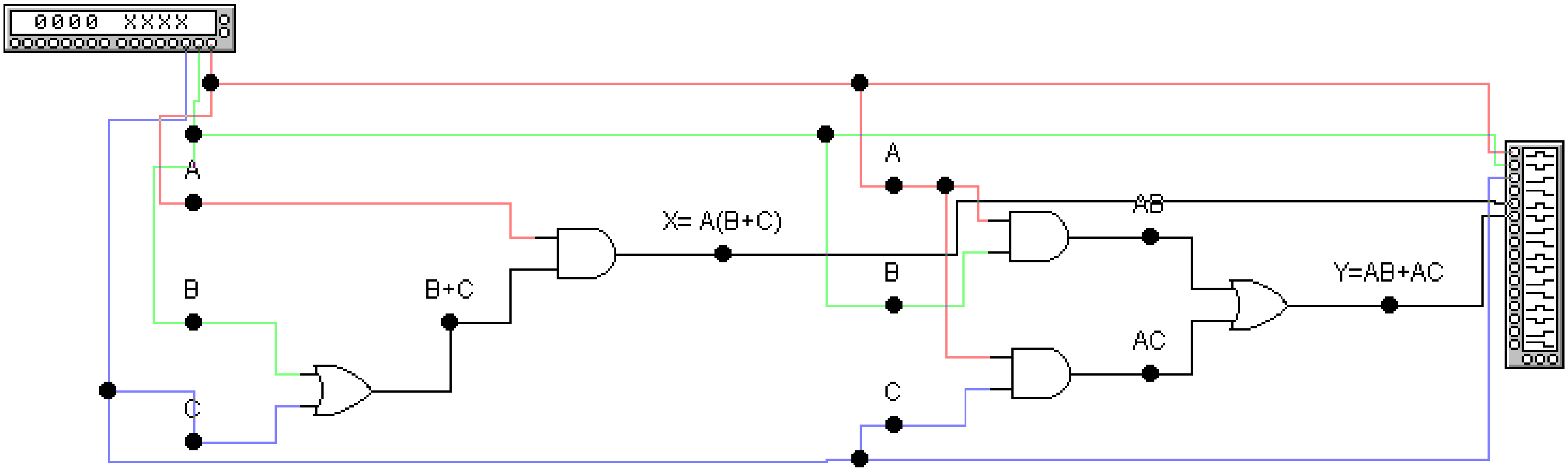
Distributive Law 1

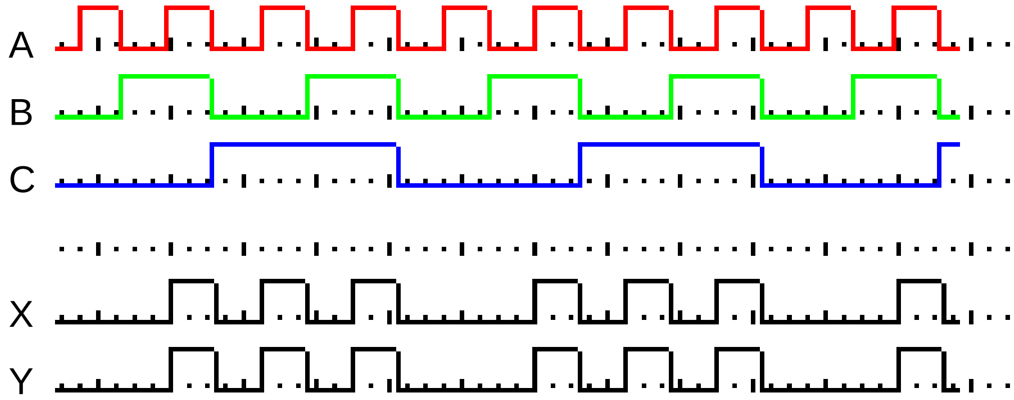
- The distributive law is the factoring law. A common variable can be factored from an expression just as in ordinary algebra
- A(B + C) = AB + AC



$A(B + C)$ $\qquad$ $AB + AC$

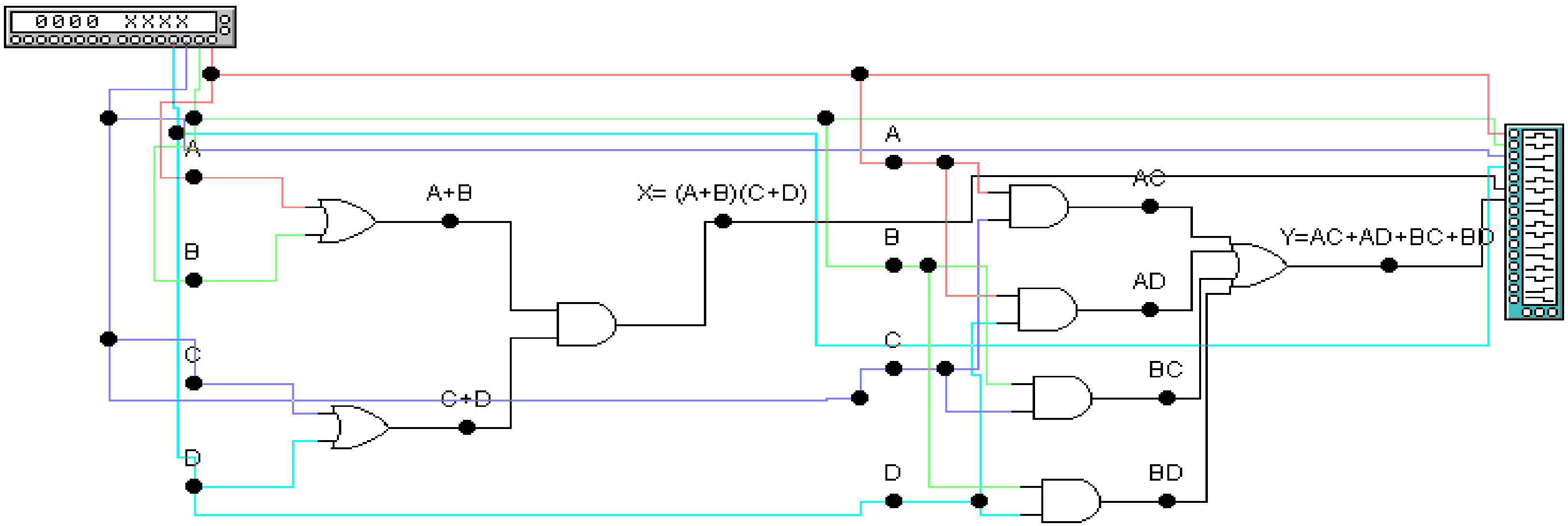
Proof





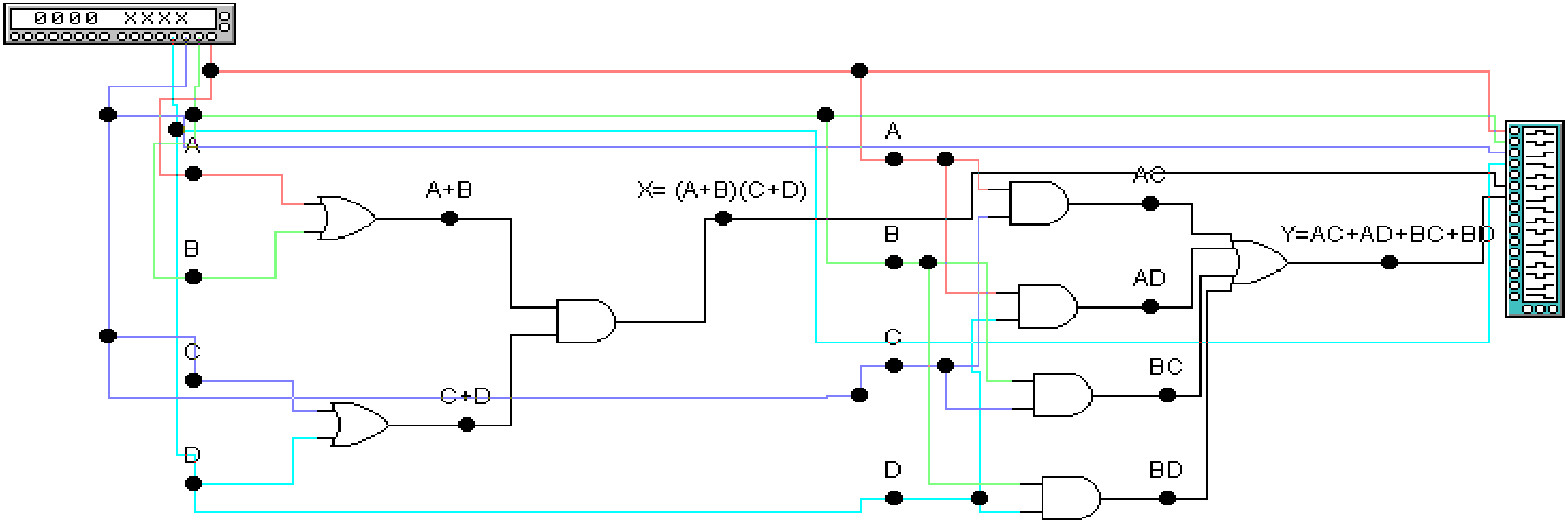
Distributive Law 2

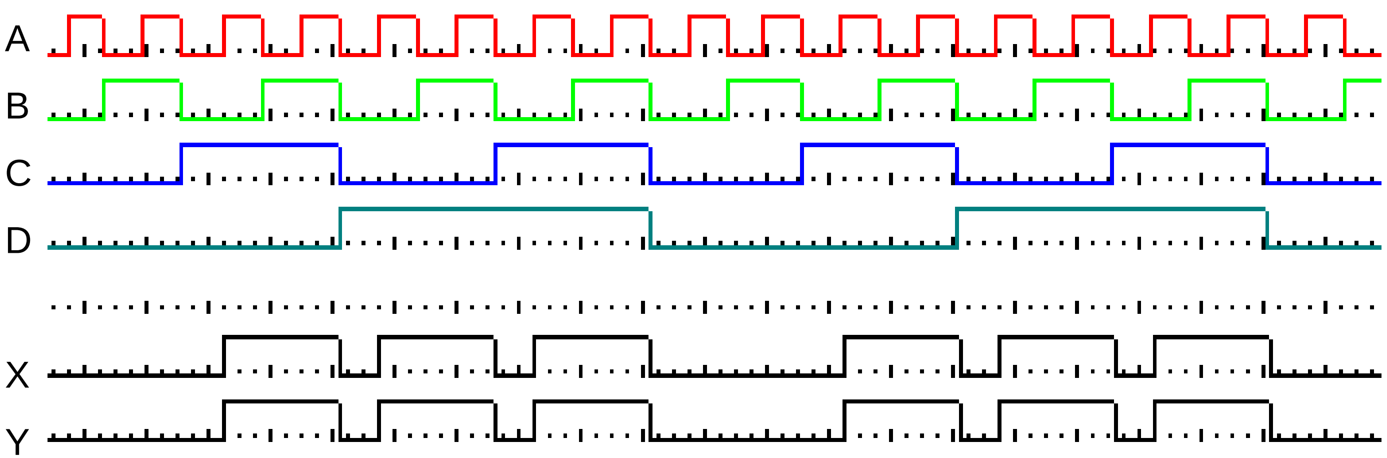
- (A+B)(C+D) = AC + AD + BC + BD



X = Y

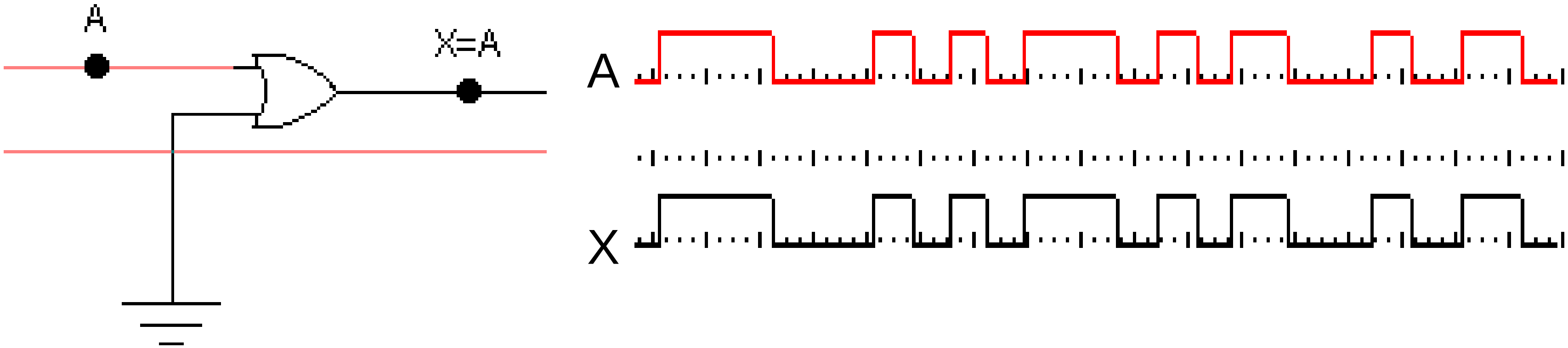
Proof





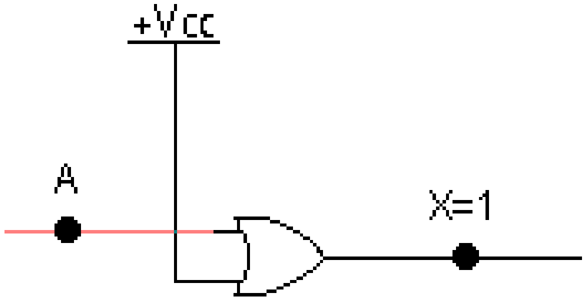
A+0=A

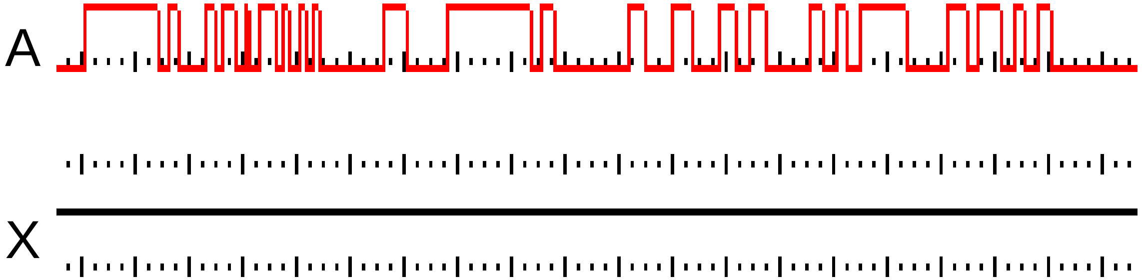
- In math if you add 0 you have changed nothing
- In Boolean Algebra ORing with 0 changes nothing



A+1=1

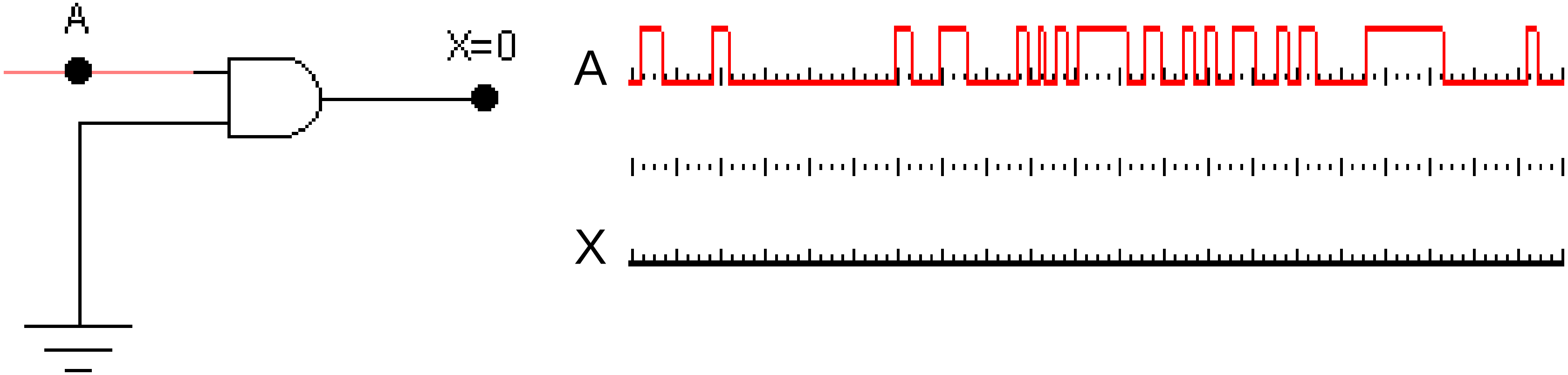
- OR-ing with 1 must give a 1 since if any input is 1 to an OR gate will give a 1





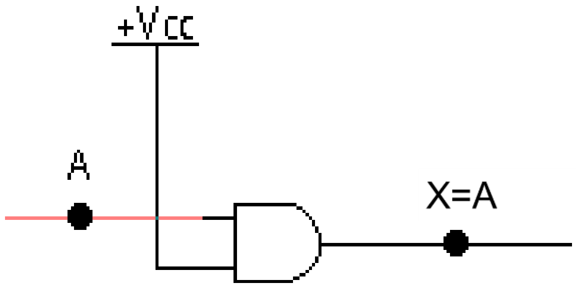
A•0=0

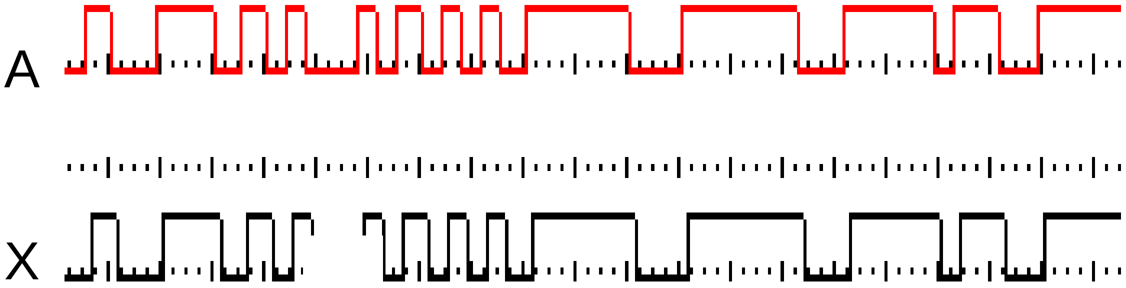
- In math if 0 is multiplied with anything you get 0.
- So if you AND anything with 0 you get 0



A•1 =A

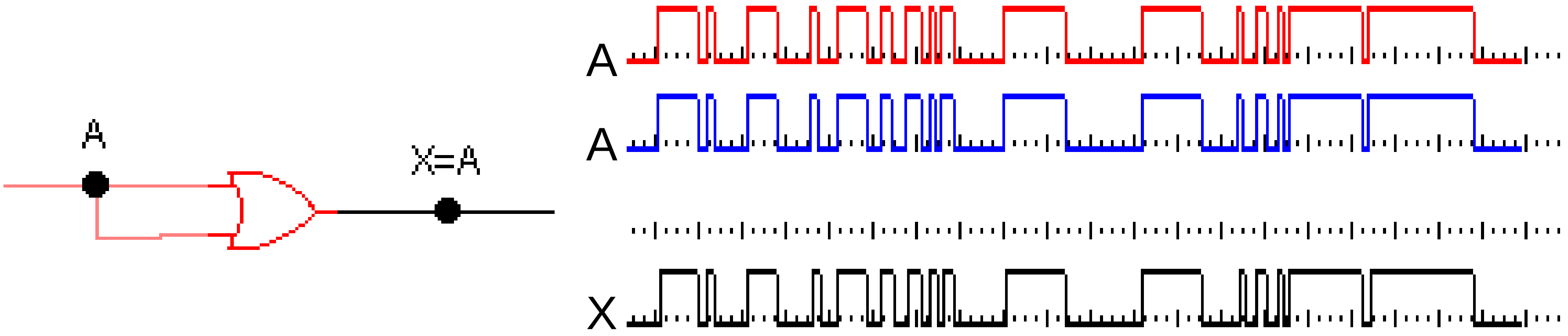
- ANDing anything with 1 will yield back anything





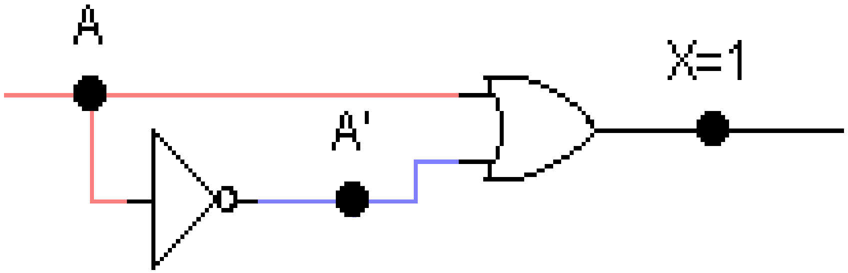
A+A = A

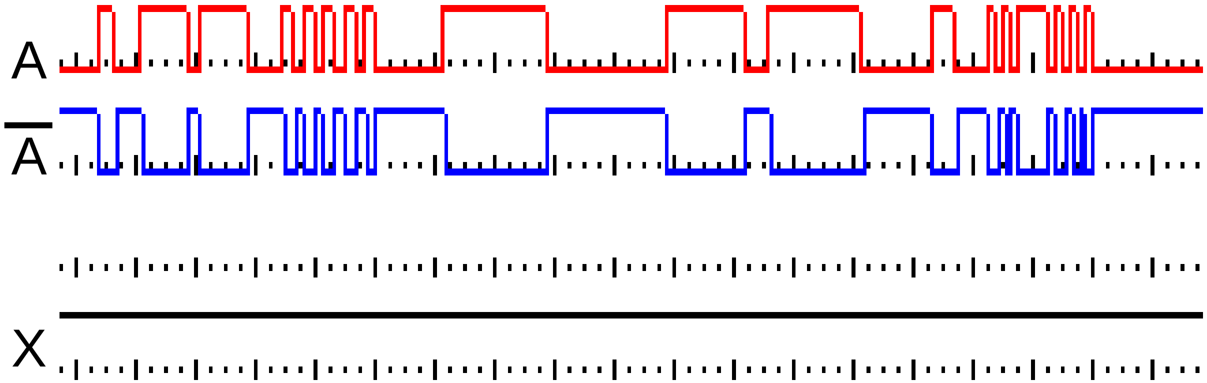
- ORing with itself will give the same result



$A+\overline{A}=1$

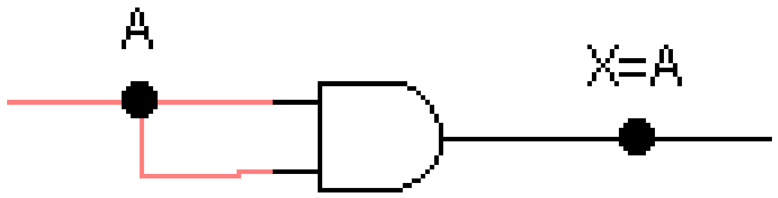
- Either A or $\overline{A}$ must be 1 so $A + \overline{A} = 1$

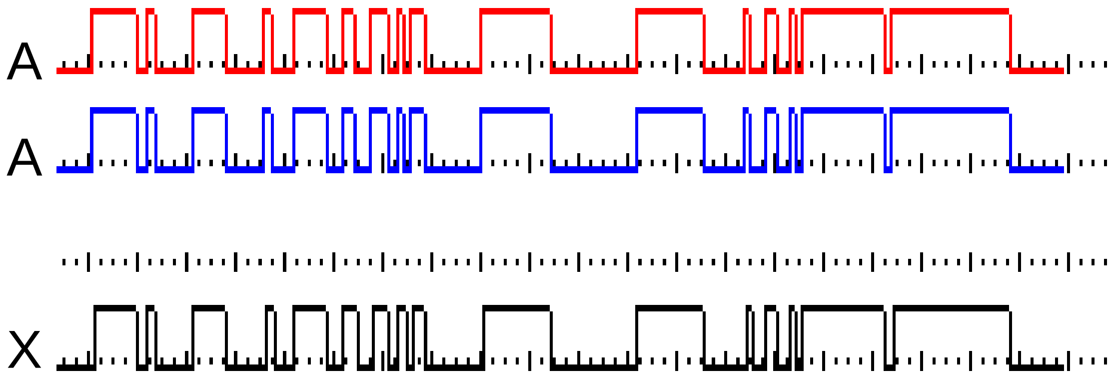




A•A = A

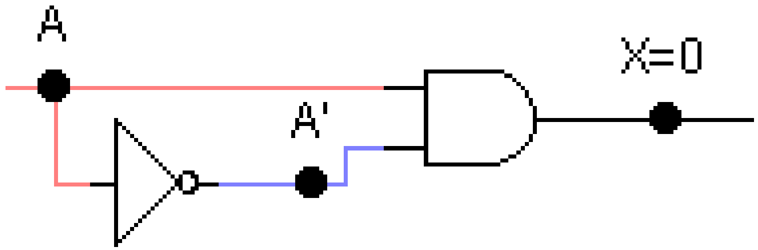
- AND-ing with itself will give the same result

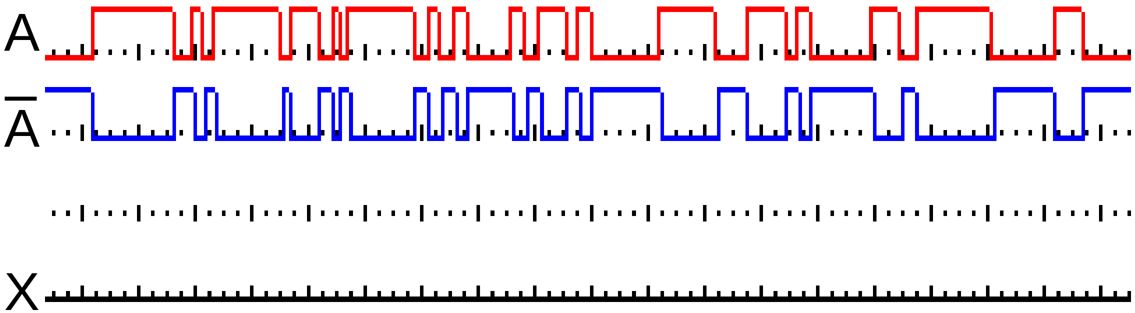




$A \bullet \overline{A} = 0$

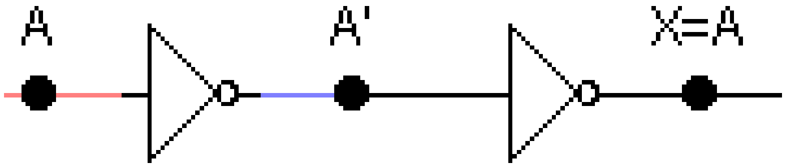
- In digital logic if A =1 then A'=0, i.e. either A or A' = 0, so AA'=0 since one of the inputs must be 0.

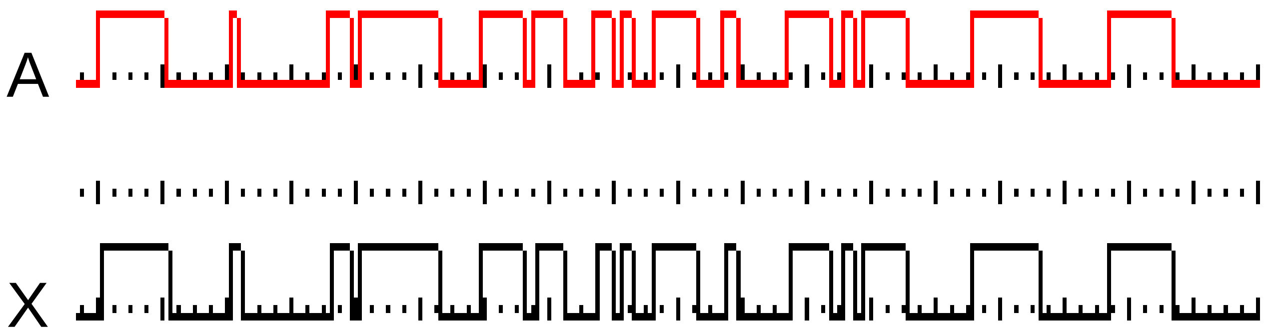




$\overline{\overline{A}} = A$

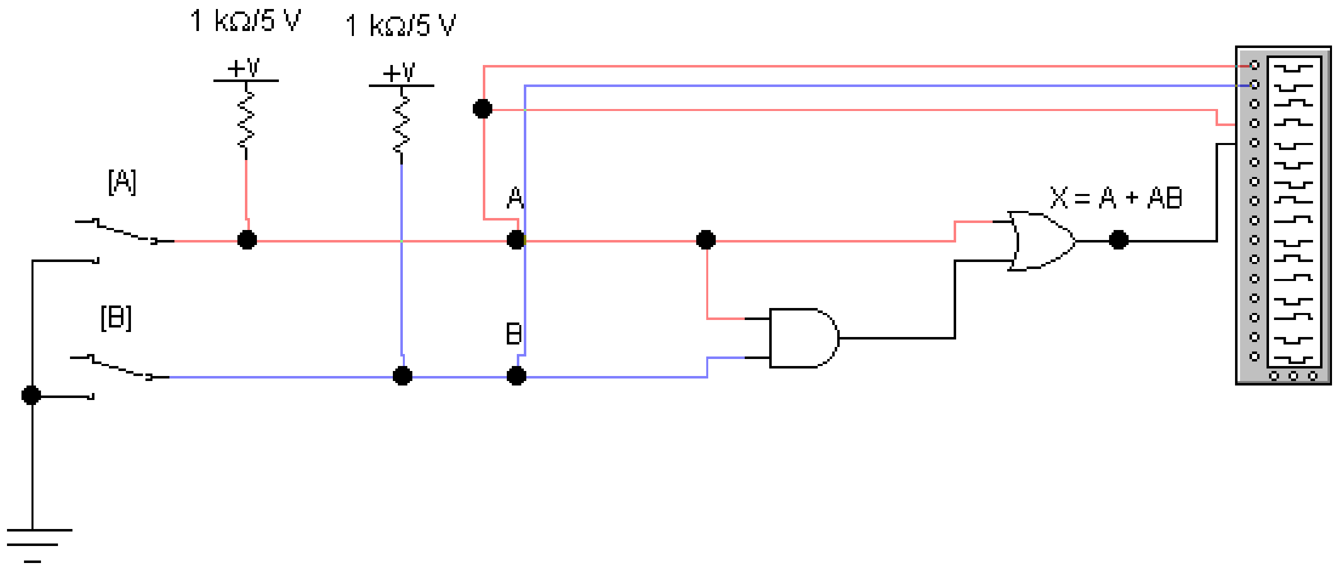
- If you invert any input twice you are back to the beginning

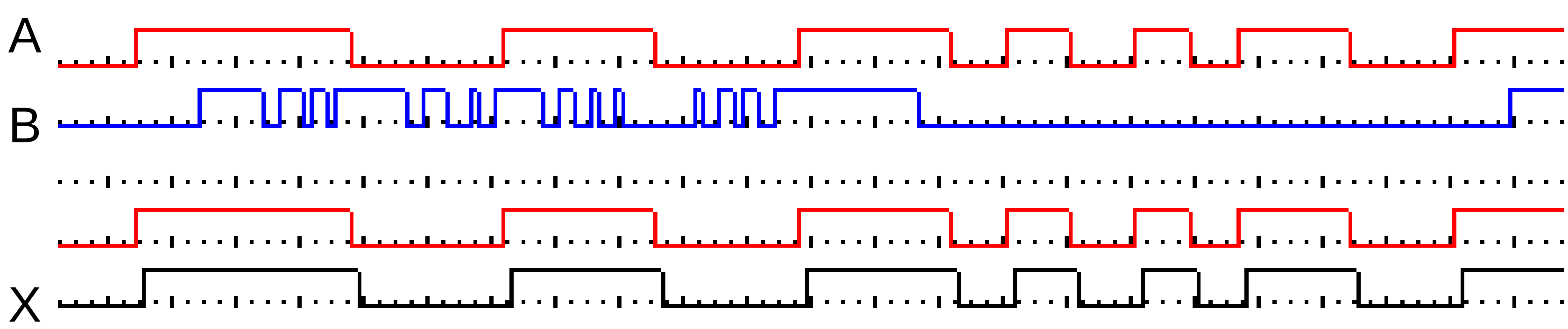




A + AB = A

- $A + AB = A(1 + B) = A(1) = A$





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

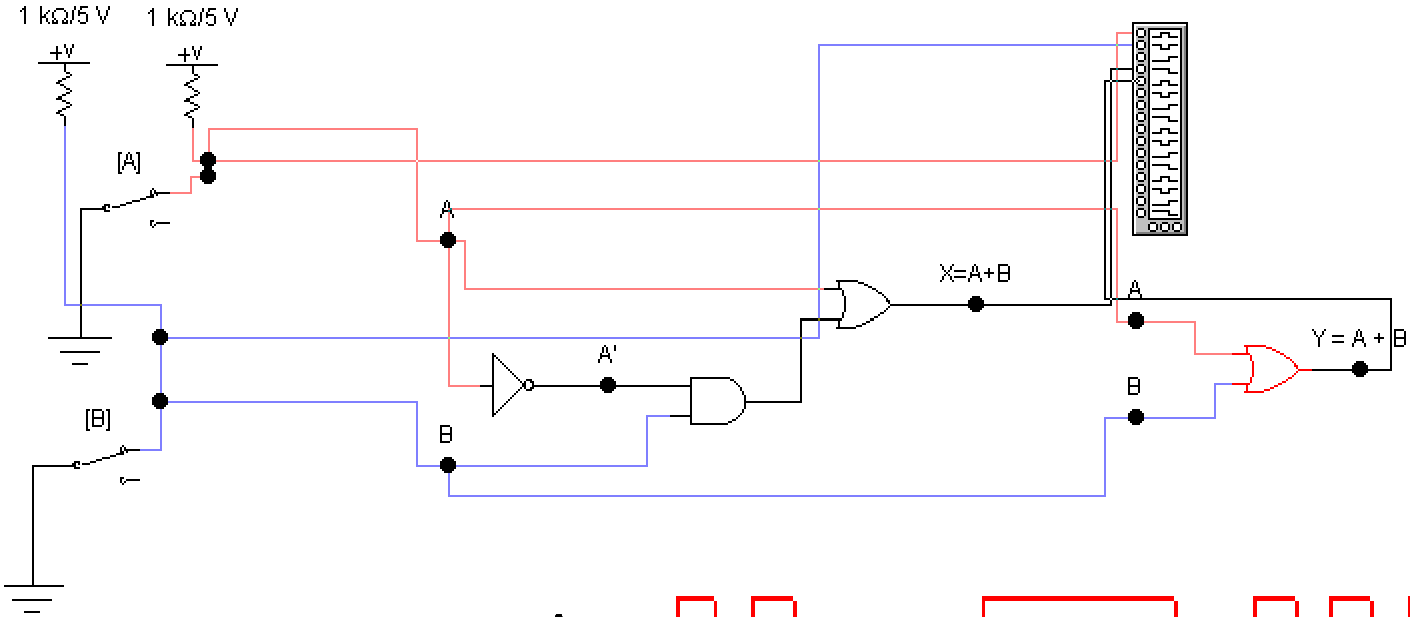
- If A is 1 the output is 1, If A is 0 the output is B

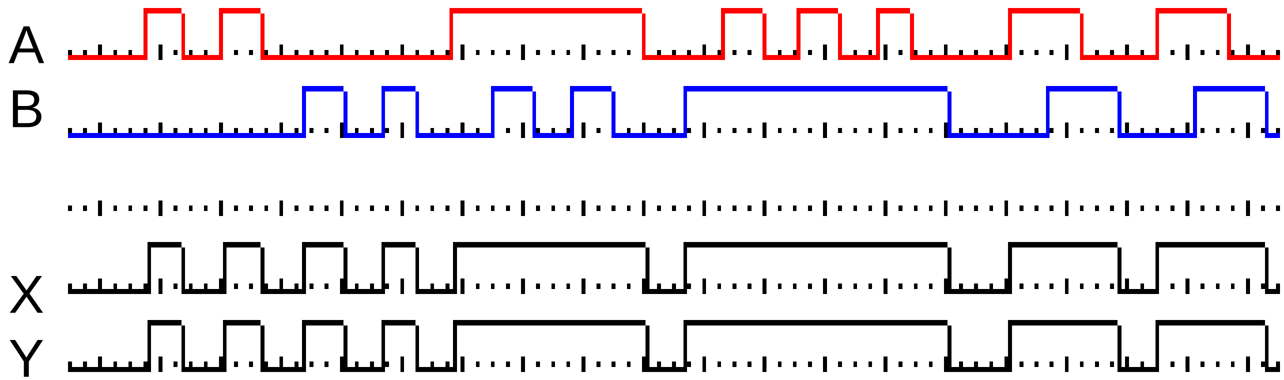
$$
\begin{aligned}
A + \overline{A}B &= (A + AB) + \overline{A}B && \leftarrow \text{rule 10} \\
&= (AA + AB) + \overline{A}B && \leftarrow \text{rule 7} \\
&= AA + AB + A\overline{A} + \overline{A}B && \leftarrow \text{rule 8} \\
&= (A+\overline{A})(A+B) && \leftarrow \text{rule 12} \\
&= 1.\ (A+B) = A + B && \leftarrow \text{rule 4}
\end{aligned}
$$

- What about A’ + AB?

$$
\begin{aligned}
A' + AB &= A'(B + B') + AB \\
&= A'B + A'B' + AB \\
&= (A' + A)B + A'B' \\
&= B + A'B' \\
&= A' + B
\end{aligned}
$$

Proof





$(A + B)(A + C) = A + BC$

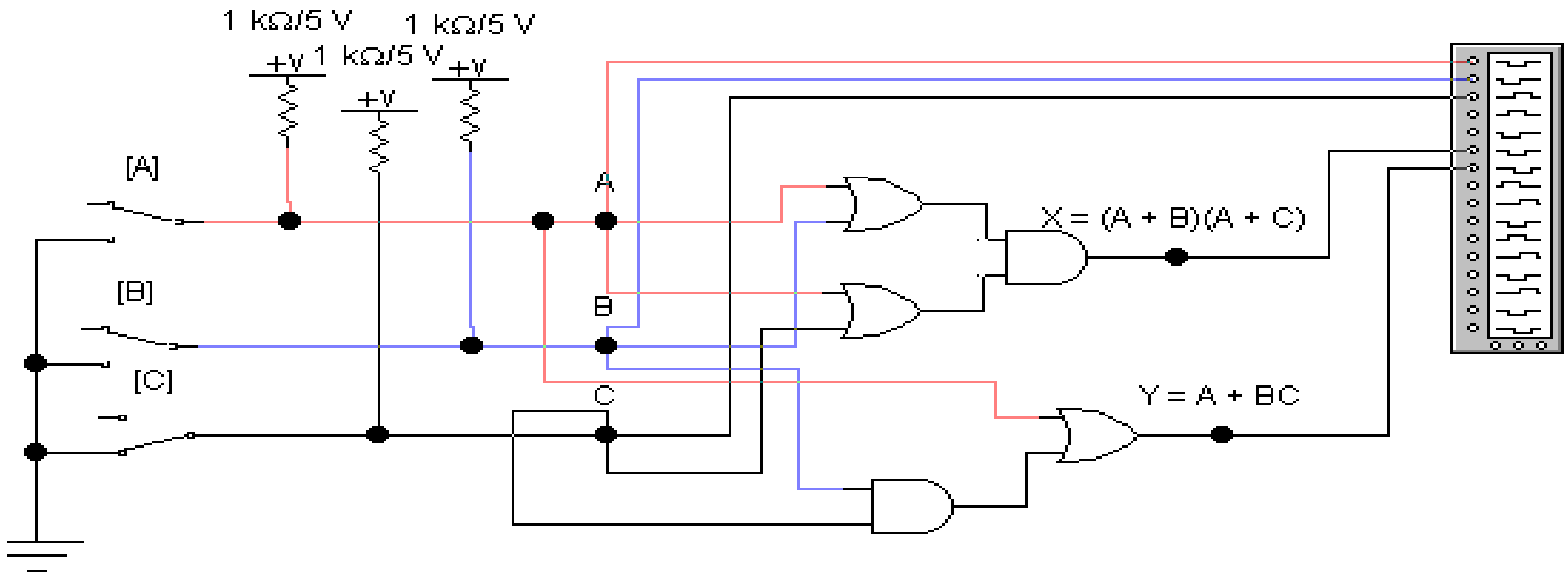
- $(A + B)(A + C) \quad = AA + AC + BA + BC$

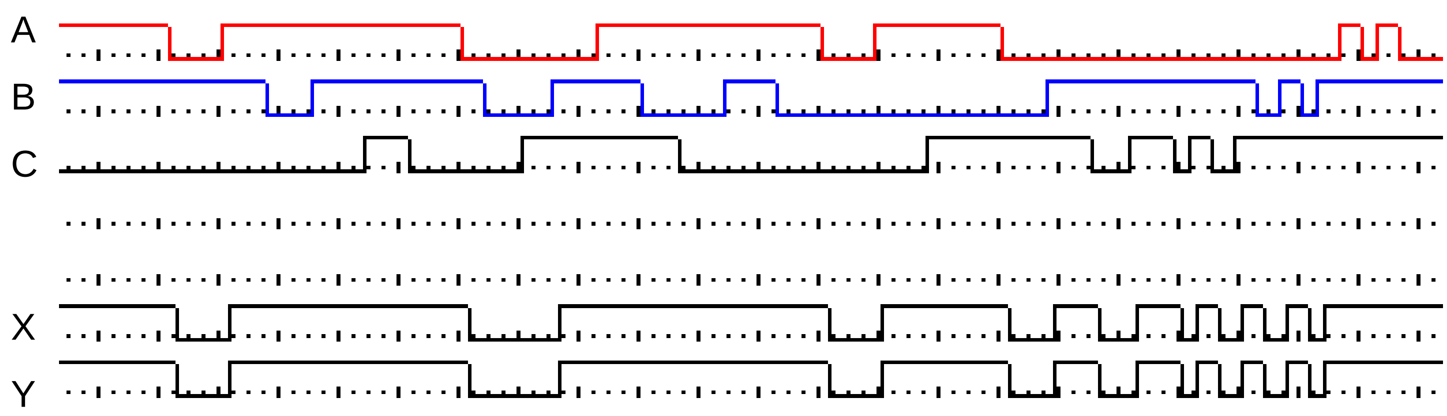
  $= A + AC + BA + BC$

  $= A (1 + C + B) + BC$

  $= A + BC$

Proof





DeMorgan's Theorem

- DeMorgan's theorems provide mathematical verification of the equivalency of the NAND gate and negative-OR gates and equivalency of the NOR and negative- AND gates.
- These theorems are extremely useful in simplifying expressions in which a product or sum of variables is inverted
  - DeMorgan will help to simplify digital circuits using NORs and ANDs his theorem states

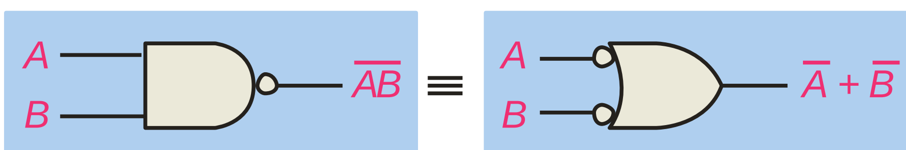
DeMorgan's Theorem

- De Morgan's first theorem:

  The complement of a product of variables is equal to the sum of the complements of the variables.

- Stated another way,
- The complement of two or more variables ANDed is equivalent to the OR of the complements of the individual variables.
- The formula for expressing this theorem for two variables is:

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



NAND   Negative-OR

Inputs		Output	
A	B	$\overline{AB}$	$\overline{A} + \overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

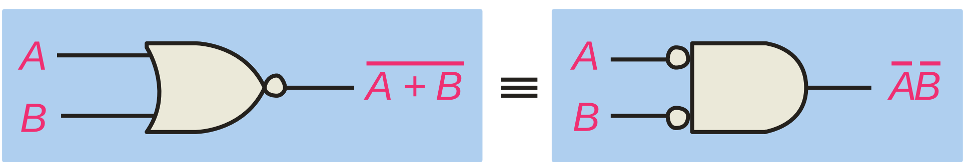
DeMorgan's Theorem

- Second theorem:

  The complement of a sum of variables is equal to the product of the complements of the variables.

- Stated in another way,
- The complement of two or more variables ORed is equivalent to the AND of the complements of the individual variables
- The formula for expressing this theorem:

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



NOR Negative-AND

Inputs		Output	
A	B	$\overline{A + B}$	$\overline{A}\overline{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Rules and theorems of Boolean algebra

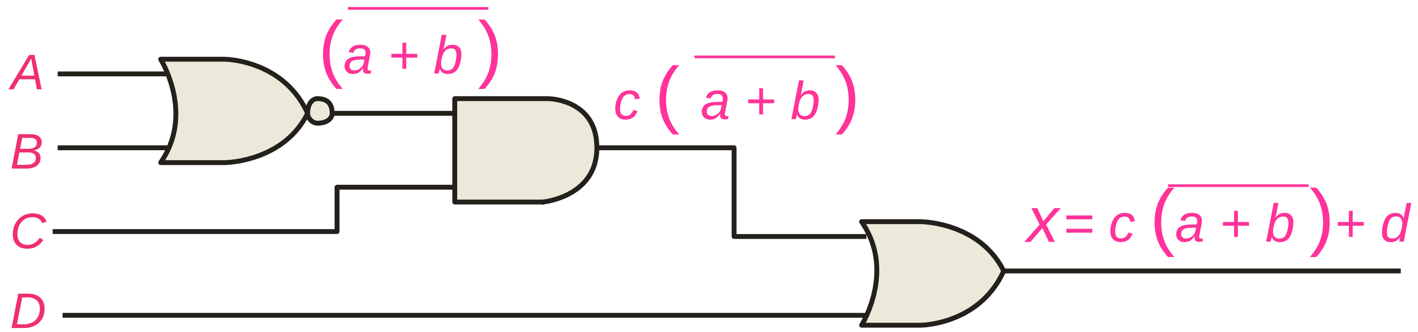
1. $A+0 = A$
2. $A+1 = 1$
3. $A.0 = 0$
4. $A.1 = A$
5. $A+A = A$
6. $A+A' = 1$ $(\overline{A} + A = 1)$
7. $A.A = A$
8. $A'.A = 0$ $(\overline{A}.A = 0)$
9. $A'' = A$ $(\overline{\overline{A}} = A)$
10. $A+AB = A$
11. $A+A'B = A+B$ $(A + \overline{A}B = A+B)$
12. $(A+B)(A+C) = A+BC$
13. $(AB)' = A' + B'$ $(\overline{AB} = \overline{A} + \overline{B})$
14. $(A + B)' = A'B'$ $(\overline{A+B} = \overline{A}.\overline{B})$

Boolean Analysis of Logic Circuits

- The purpose of this section is to practice changing gates to simplified Boolean Algebra expressions.
- Combinational logic circuits can be analyzed by writing the expression for each gate and combining the expressions according to the rules for Boolean algebra
  - Simplification of the Boolean is also done using the Boolean Laws and rules.

Examples

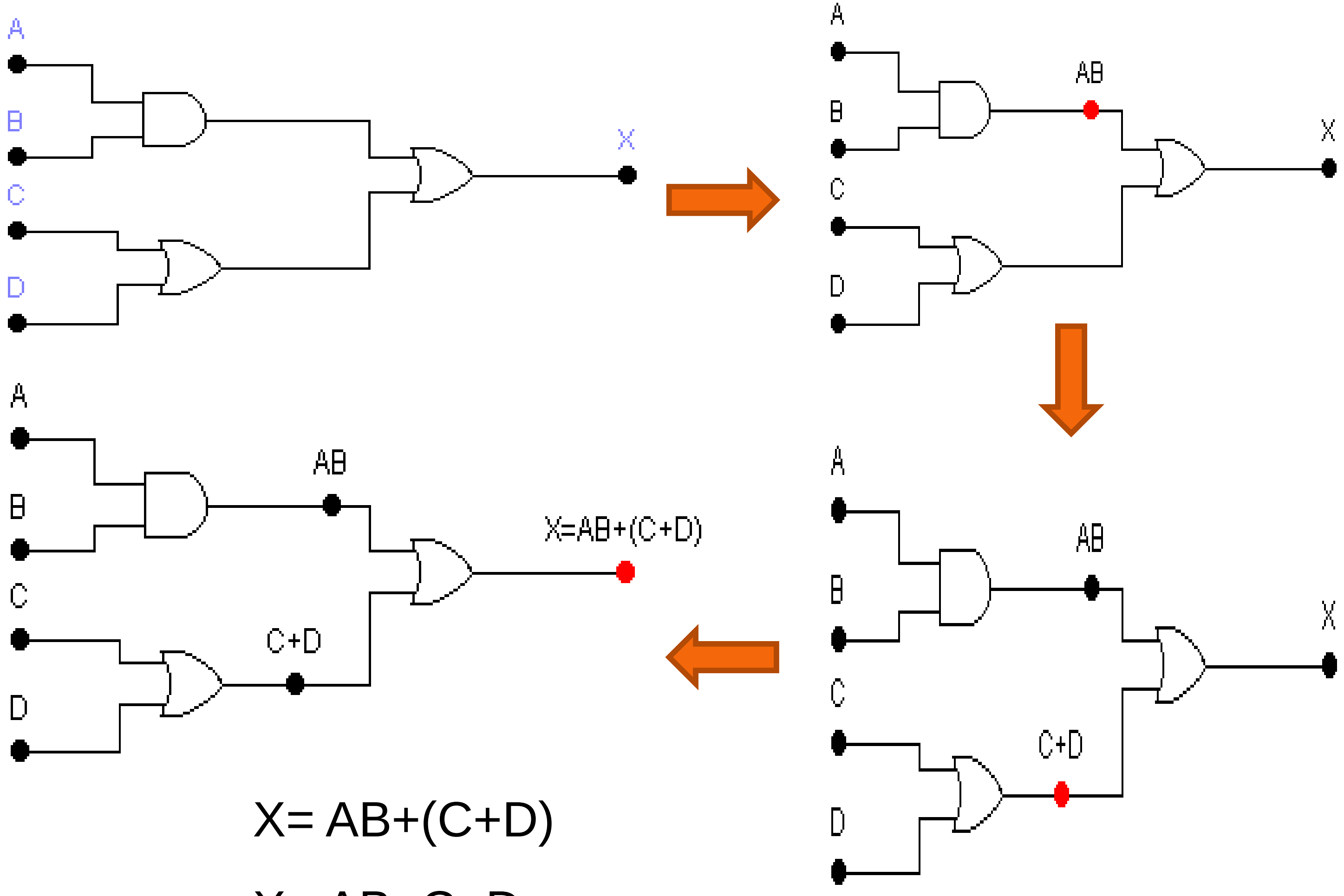
- Example 1
  Apply Boolean algebra to derive the expression for X.



- Solution
  Write the expression for each gate

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

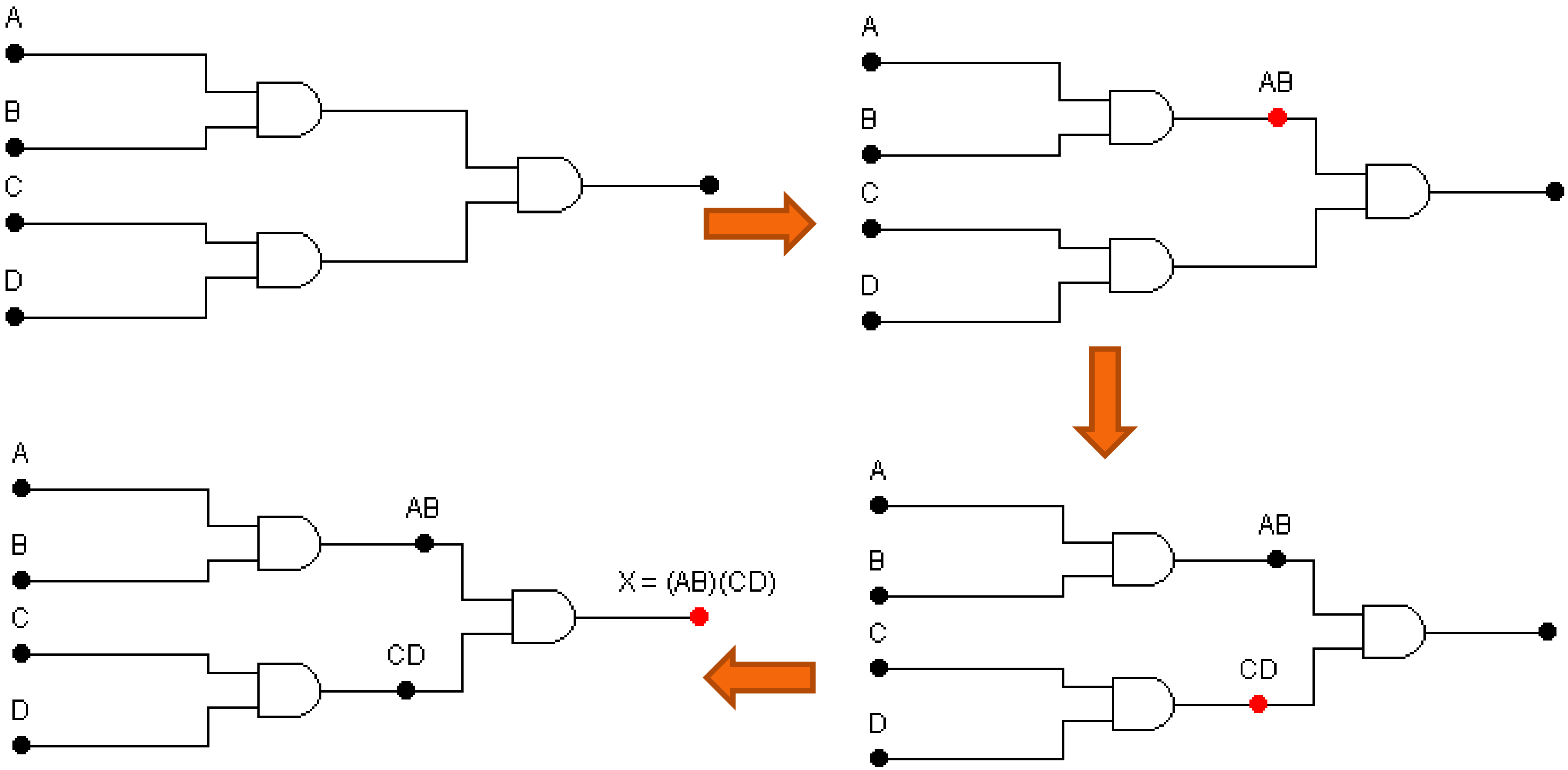
Example 2



$X = AB + (C+D)$

$X = AB + C + D$

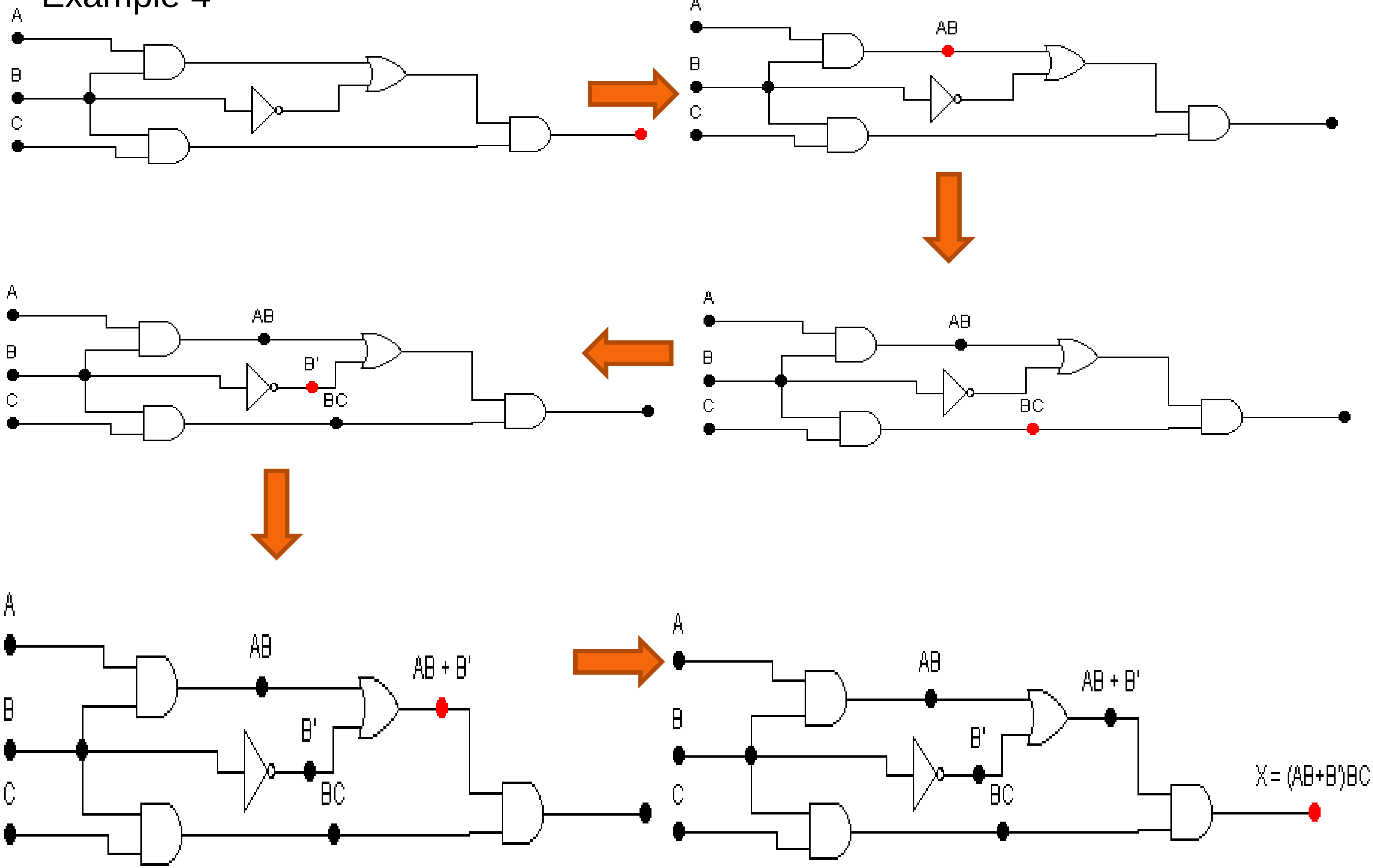
Example 3



X = (AB)(CD)

X = ABCD

Example 4



Boolean logic simplification

- We will see the circuit from the slide before can be simplified with the following circuit and see they do the same thing

$X = (AB+\overline{B})BC$

using distributive law

$X = ABBC + \overline{B}BC$

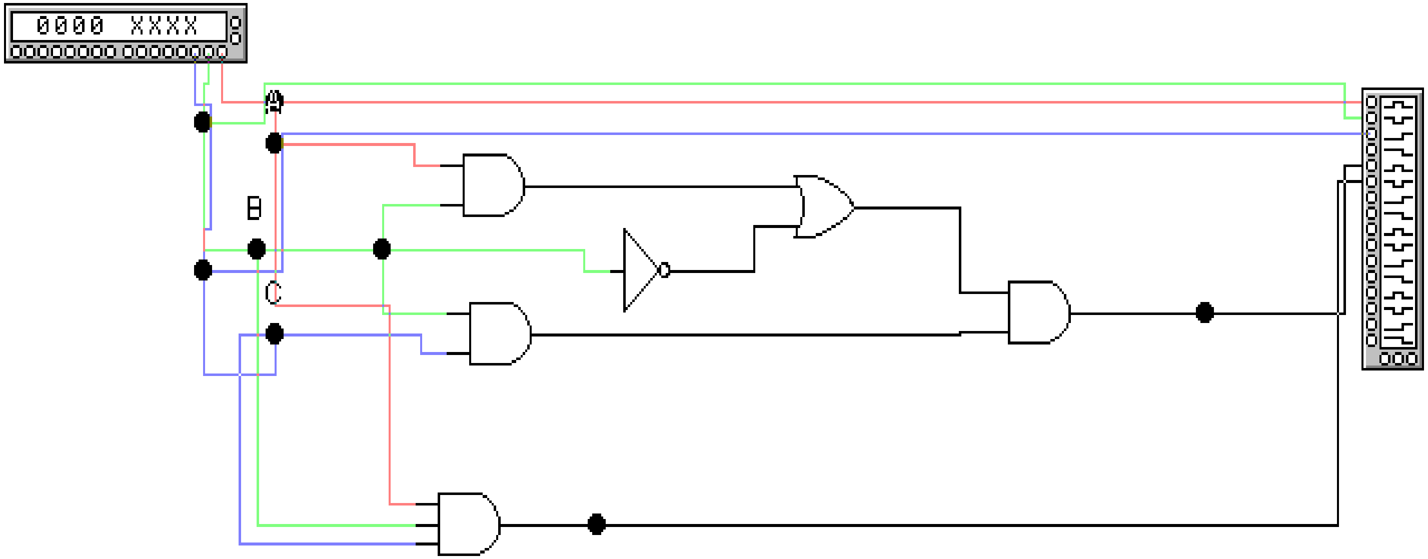
$X = ABC + \overline{B}BC$

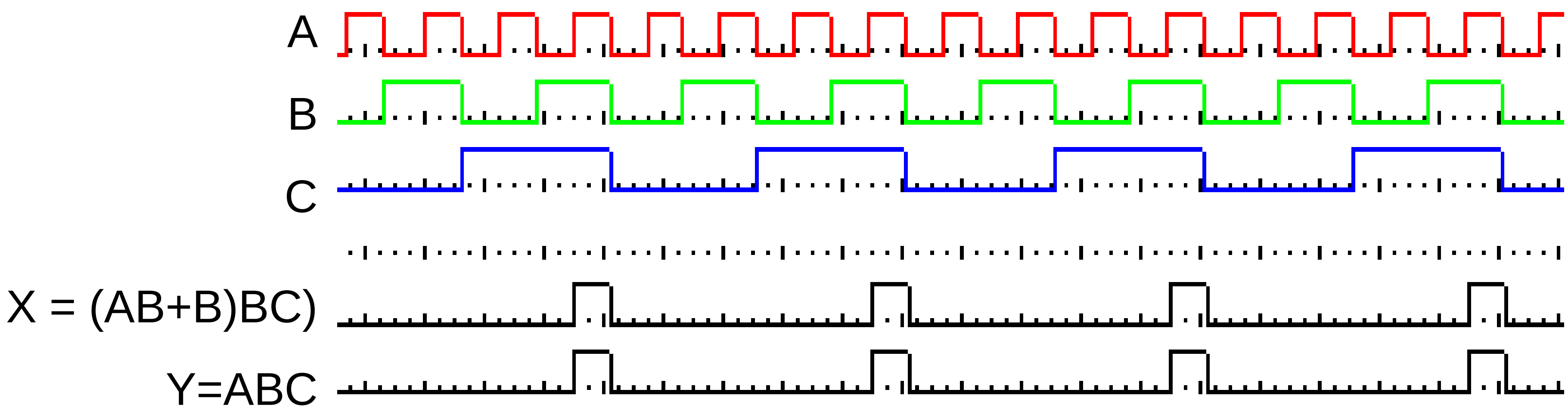
$X = ABC + 0 \bullet C$

$X = ABC + O$

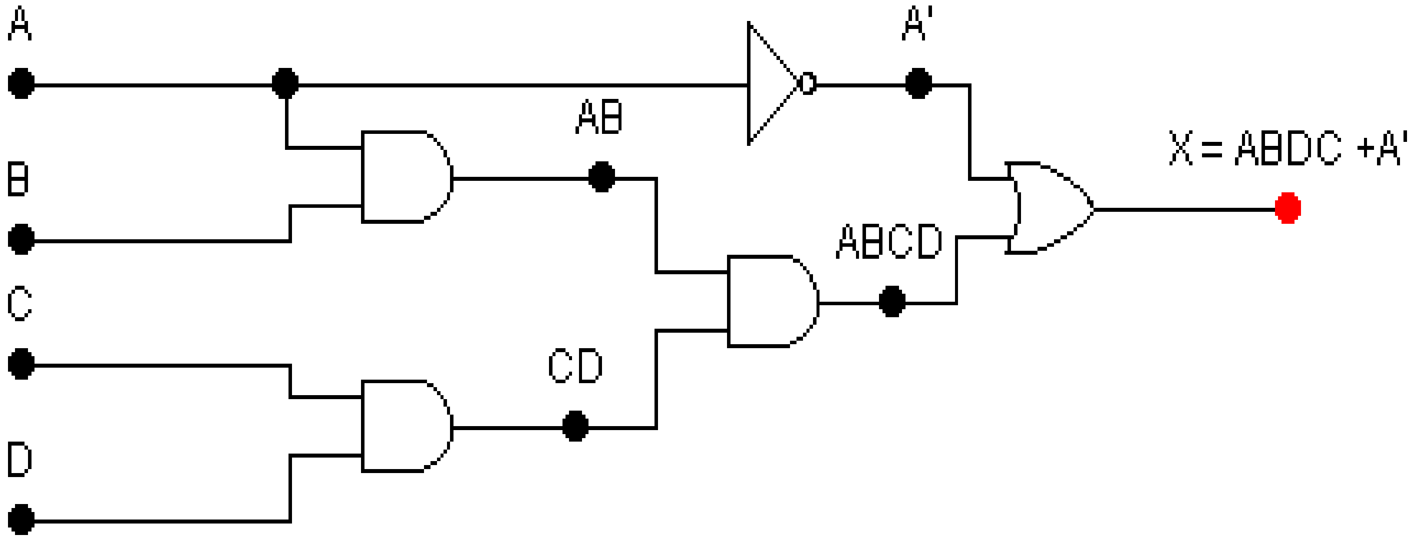
$X = ABC$

Proof





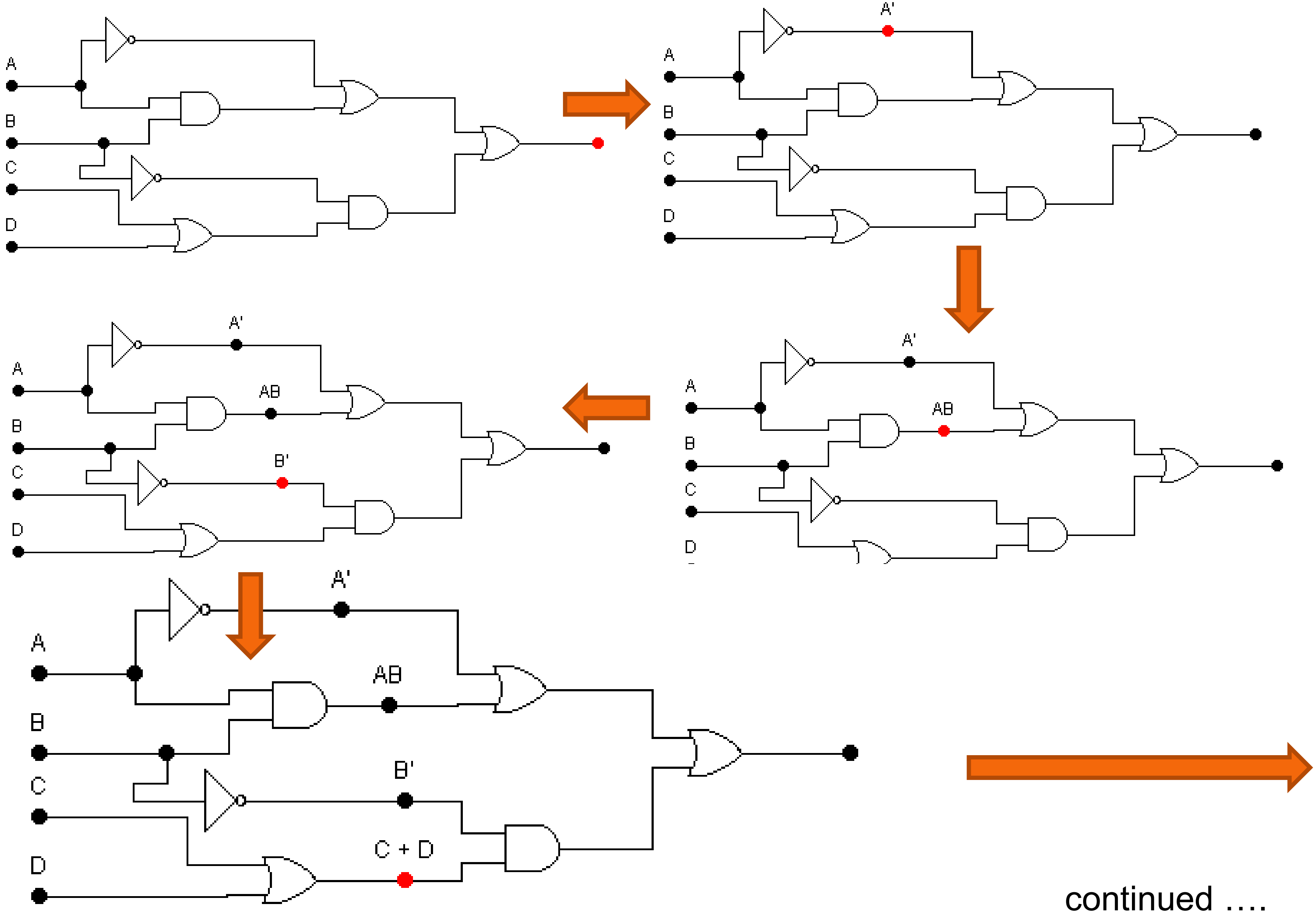
Using example 3 from earlier slides, simplify the equation obtained



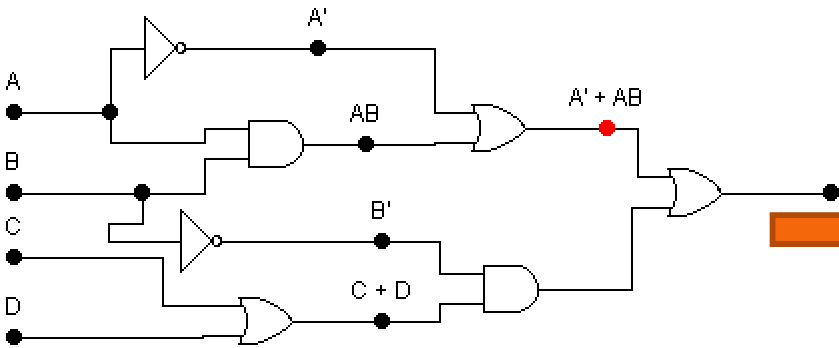
$$X = ABCD + \overline{A}$$

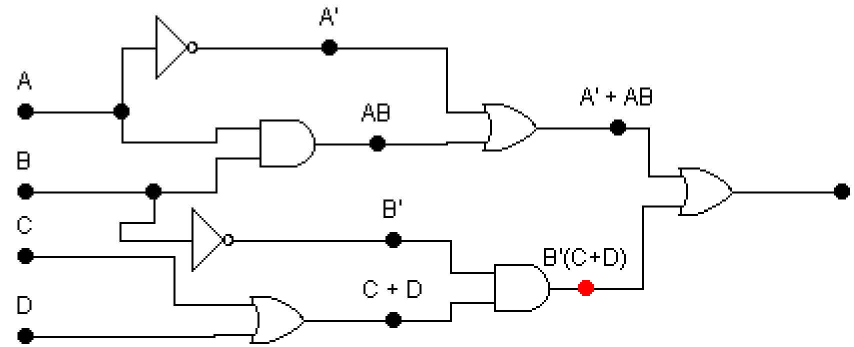
$$= \overline{A} + BCD$$

Example 5 – Simplify the logic diagram below



continued ….





$$X = (\overline{A} + AB) + (\overline{B}(C + D))$$

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

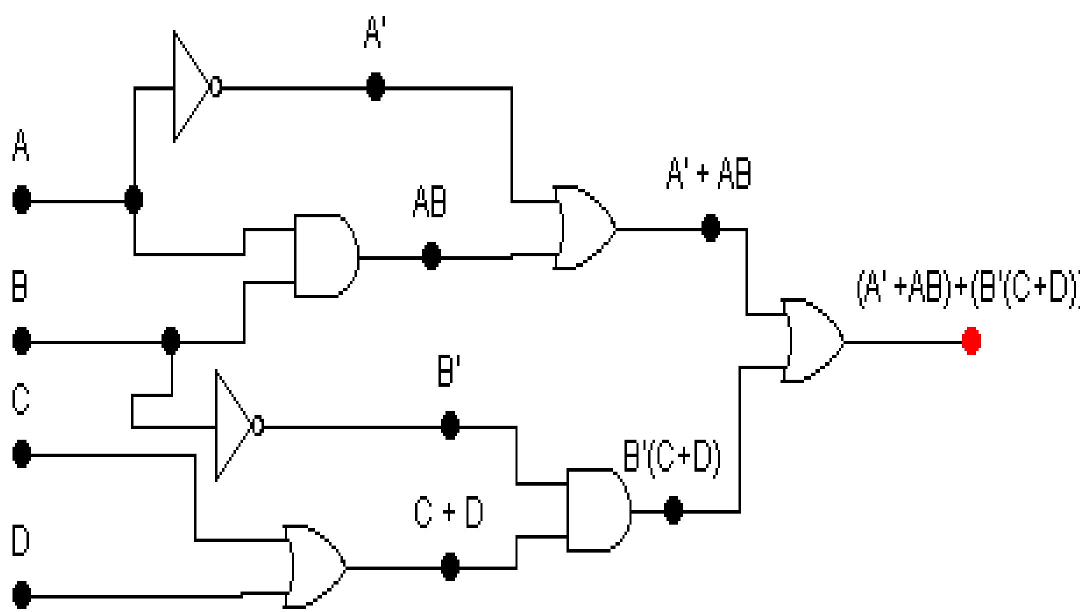
$$X = (\overline{A} + B) + (\overline{B}C + \overline{B}D)$$

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

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

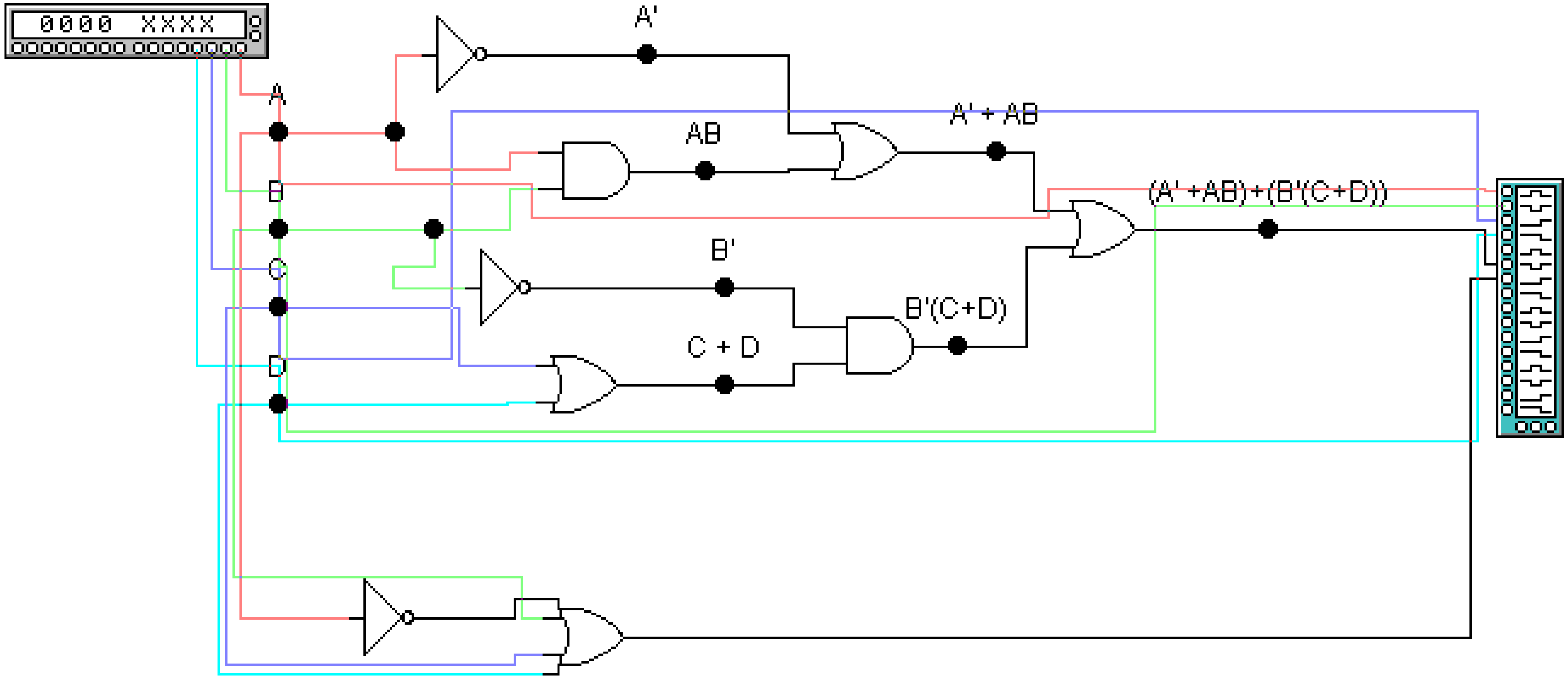
$$X = \overline{A} + B + C + \overline{B}D$$

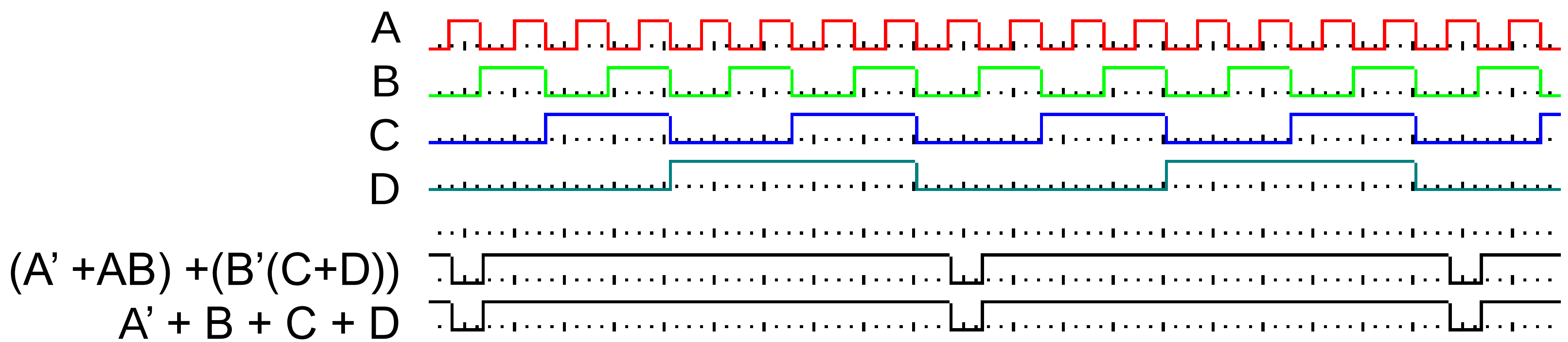
$$X = \overline{A} + B + C + D$$



Proof

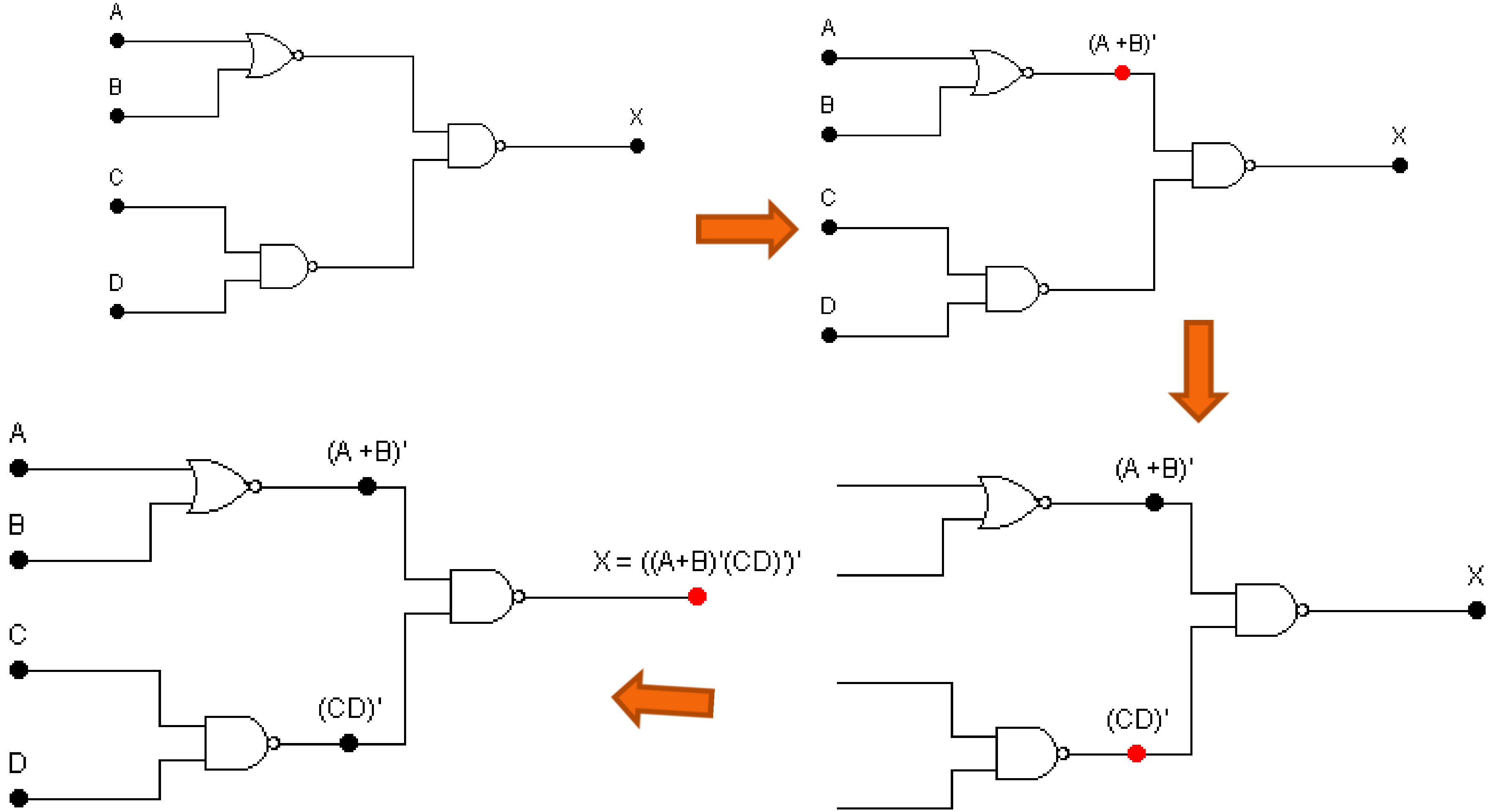
- The circuits are different but the outputs are the same

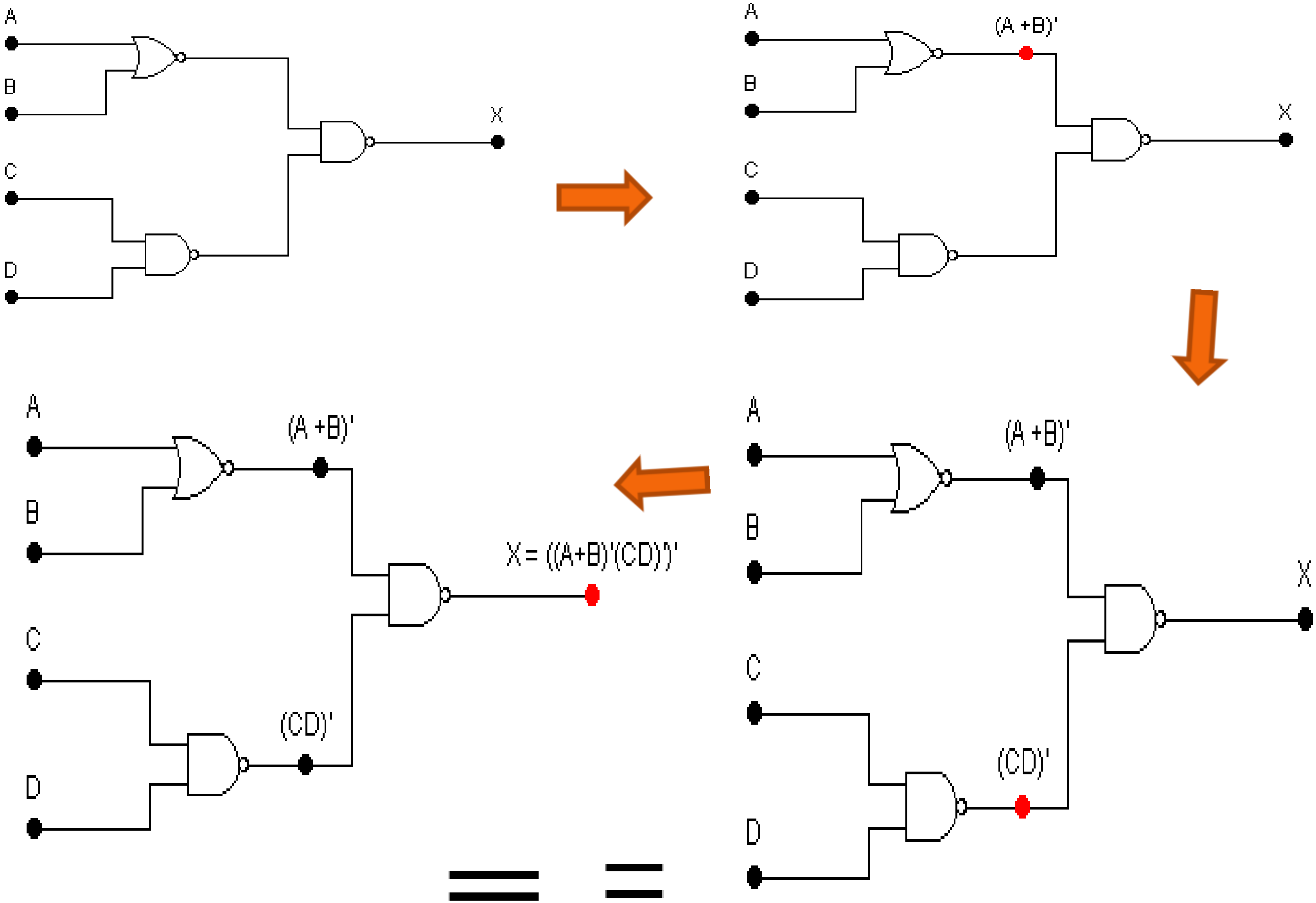




Another simplification example

- Simplify the logic diagram below

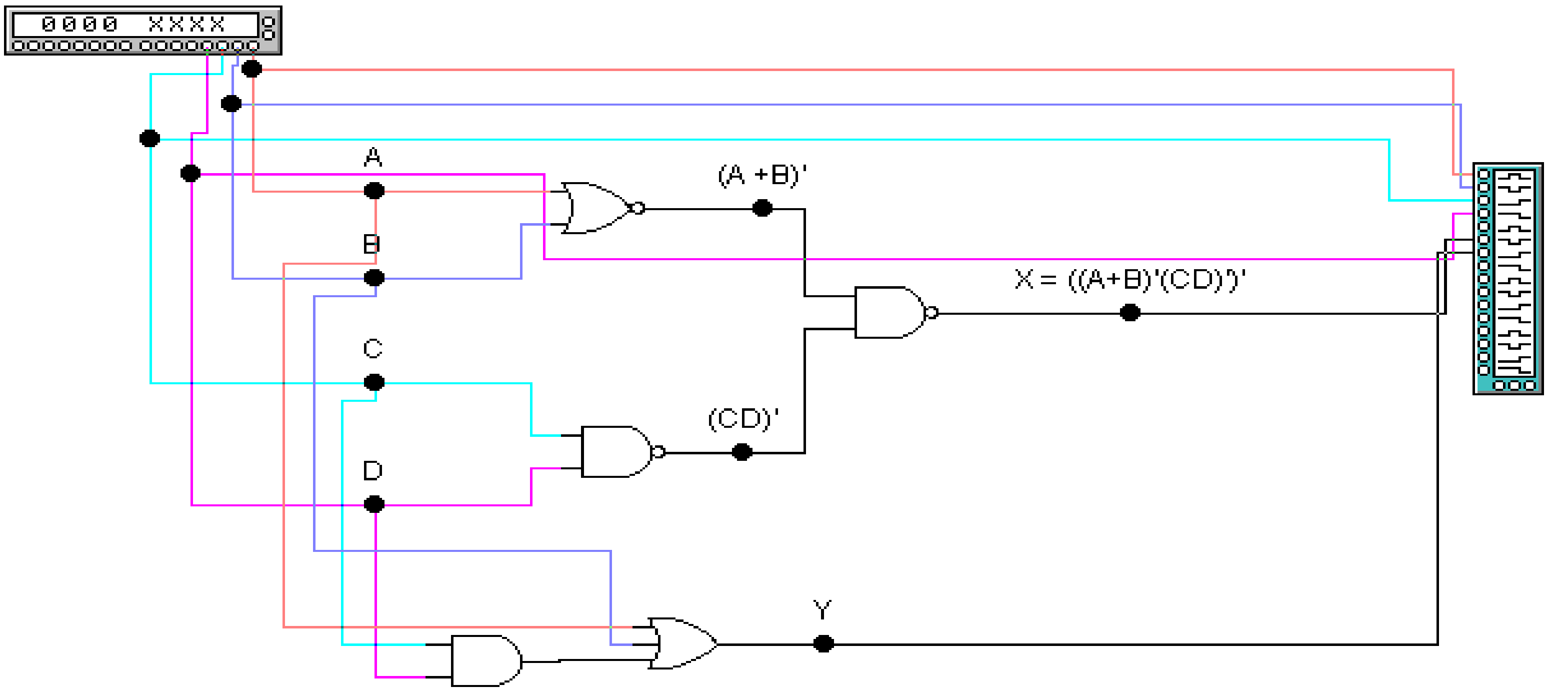


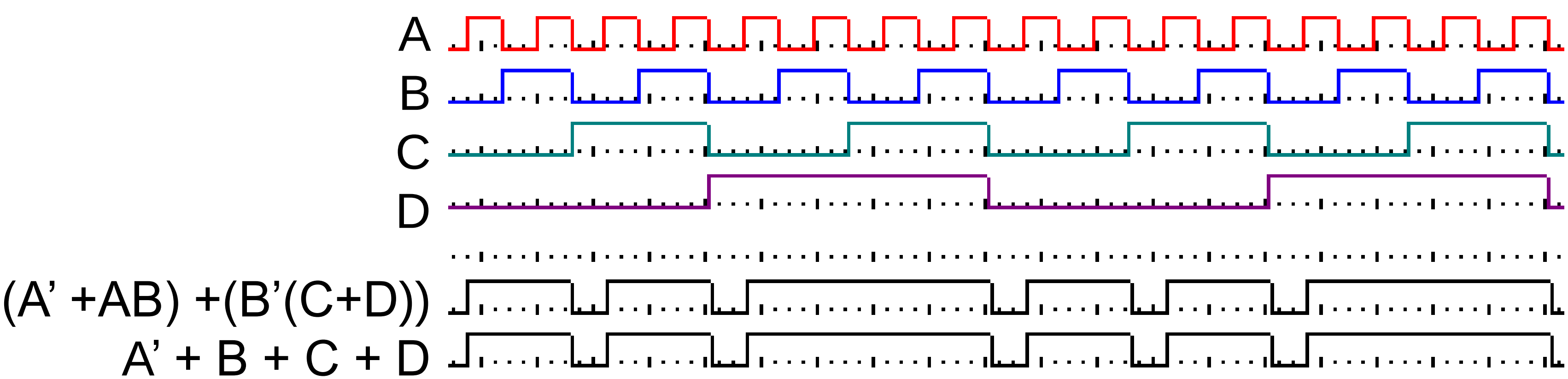


$$((A + B)'(CD)')' = (A + B) + CD$$

$$= A + B + CD$$

Proof

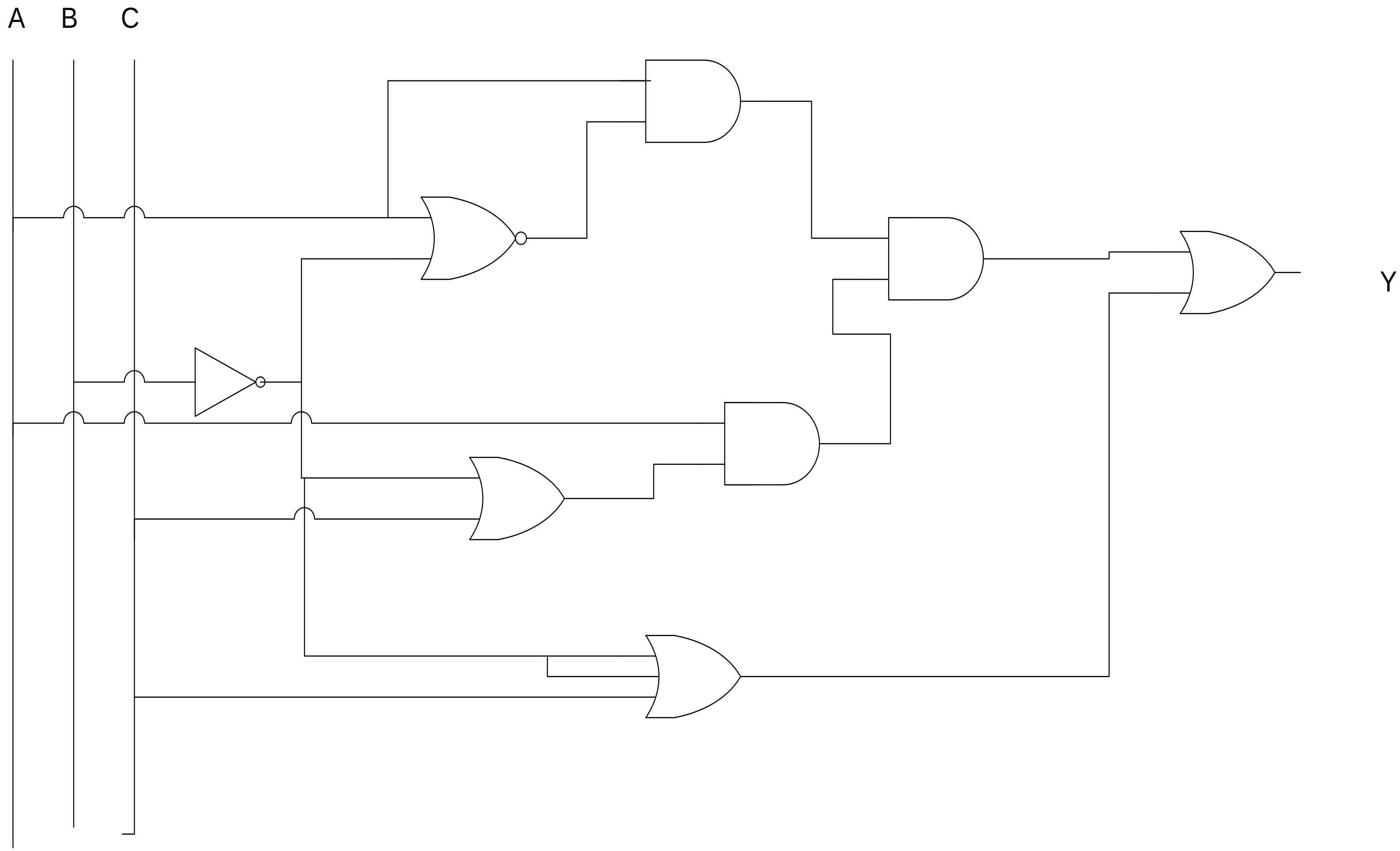




Example 6

Draw the Logic circuit for the boolean expression Y

$$Y= [(A+B')'A][(B'+C)A]+[B'+C+ B']$$



Example 6

Simplify Y using boolean algebra rules + theorems and draw the resulting logic diagram for Y

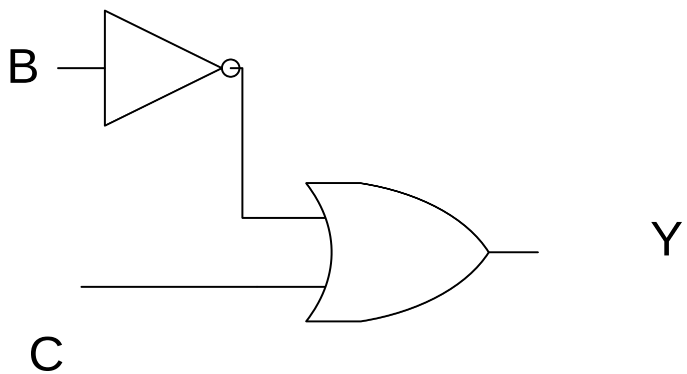
$$Y = \left[\overline{(A+\bar{B})}A\right][(\bar{B}+C)A] + [\bar{B}+C+\bar{B}]$$

$$= \left[\bar{A}\bar{\bar{B}}A\right][A\bar{B}+AC] + [\bar{B}+C]$$

$$= 0[A\bar{B}+AC] + [\bar{B}+C]$$

$$= 0 + [\bar{B}+C]$$

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



Truth tables

- Truth tables for boolean expressions are similar to truth tables for individual gates – each gate is individually evaluated, then combined (using boolean algebra) repeatedly until the output is acheived
- Example : Create the truth table for the given Boolean expression

$$Y = [(A+B')'A][(B'+C)A]+[B'+C+B']$$

INPUT						X_1		X_2			OUTPUT
A	B	C	B'	A+B'	(A+B')'	(A+B')'A	B'+C	(B'+C)A	X_1X_2	B'+C+B'	Y
0	0	0	1	1	0	0	1	0	0	1	1
0	0	1	1	1	0	0	1	0	0	1	1
0	1	0	0	0	1	0	0	0	0	0	0
0	1	1	0	0	1	0	1	0	0	1	1
1	0	0	1	1	0	0	1	1	0	1	1
1	0	1	1	1	0	0	1	1	0	1	1
1	1	0	0	1	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1	1	0	1	1

Validation

- The boolean expression earlier was simplified to Y = B’ + C
- Truth tables can be used to validate the simplification

INPUT						X_1		X_2			OUTPUT
A	B	C	B’	A+B’	(A+B’)’	(A+B’)’A	B’+C	(B’+C)A	X_1X_2	B’+C+B’	Y
0	0	0	1	1	0	0	1	0	0	1	1
0	0	1	1	1	0	0	1	0	0	1	1
0	1	0	0	0	1	0	0	0	0	0	0
0	1	1	0	0	1	0	1	0	0	1	1
1	0	0	1	1	0	0	1	1	0	1	1
1	0	1	1	1	0	0	1	1	0	1	1
1	1	0	0	1	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1	1	0	1	1

Summary

- The Boolean sum of two or more literals equivalent to an OR operation.
- The Boolean product of two or more literals equivalent to an AND operation
- Boolean algebra has its roots in conventional algebra → 12 rules in total
- Boolean expressions (and their corresponding logic diagrams) can be simplified using algebra rules and DeMorgan's theorems to create shorter expressions (and smaller logic diagrams)