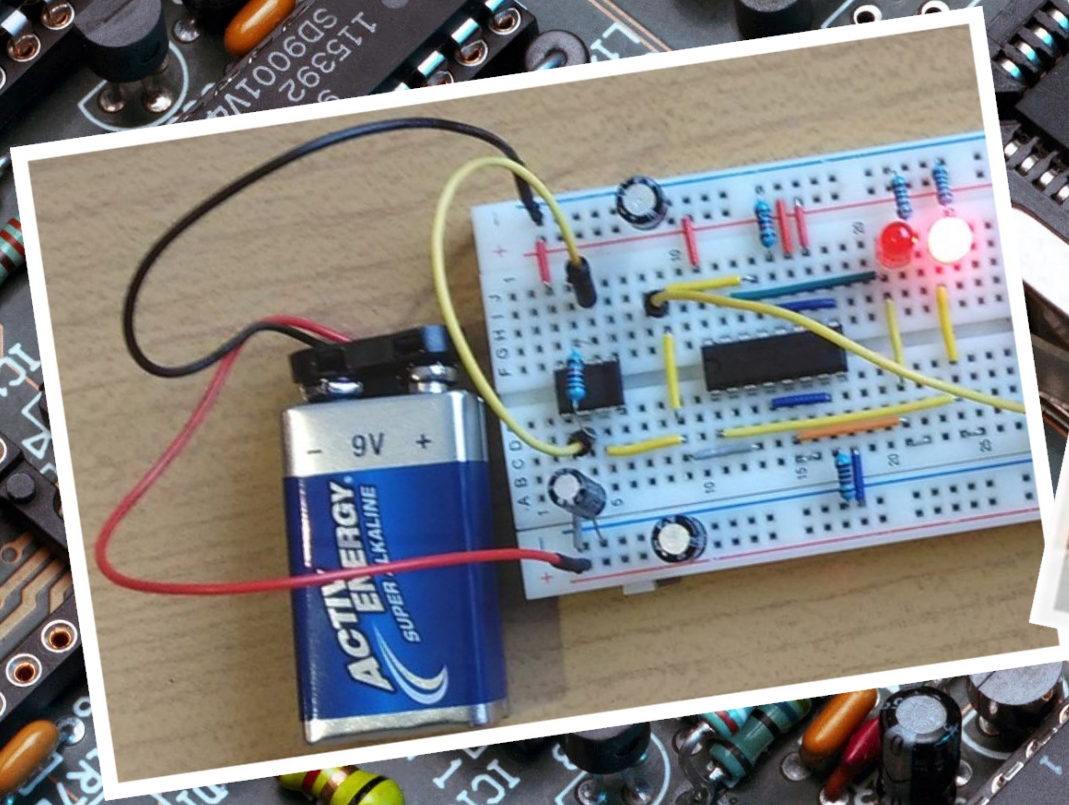
A digitális elektronika alapjai



3. Logikai kapuáramkörök – 2. rész

Felhasznált és ajánlott irodalom

- Gulyás Dénes: Számítógép architektúrák *(interaktív jegyzet)*
- Mike Gábor: A digitális elektronika alapjai *(jegyzet és videók)*
- Zalotay Péter: Digitális technika
- Végh János: Ismerkedés a digitális elektronikával
- Mészáros Miklós: Logikai algebra alapjai, logikai függvények I.
- Mingesz Róbert: Digitális technikai tananyagok
- F-alpha.net: Digital Electronics
- Electronics Tutorials: Logic Gates
- M. Morris Mano and Michael D. Ciletti: Digital Design - With an Introduction to the Verilog HDL, 5th. Edition

Logikai áramkör szimulátorok

- LogiSim szimulátor: www.cburch.com/logisim/
- Falstad.com: Circuit simulator
- CircuitVerse: Simulator
- University of Genoa: Deeds Simulator
- Gatecat: Breadboard Simulator v1.0
- Logic.ly: Logic.ly Simulator (online demo)

Ismétlés: további logikai műveletek

- A kétváltozós logikai függvények igazságtáblázatát az eddig tárgyalt **ÉS** és **VAGY** logikai műveleten kívül másképpen (összesen 16-féleképpen) is kitölthetjük. Ezeket, vagy ezek némelyikét újabb logikai műveletnek is tekinthetjük

Kétváltozós logikai függvények igazságtáblázatai

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

$F_0 = 0$
$F_1 = x \cdot y$
$F_2 = x \cdot y'$
$F_3 = x$

$F_4 = x' \cdot y$
$F_5 = y$
$F_6 = x \oplus y$
$F_7 = x + y$

$F_8 = (x + y)'$
$F_9 = (x \oplus y)'$
$F_{10} = y'$
$F_{11} = x + y'$

$F_{12} = x'$
$F_{13} = x' + y$
$F_{14} = (x \cdot y)'$
$F_{15} = 1$

Ismétlés: további logikai műveletek

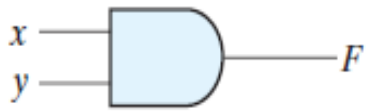
Boole függvény	Műveleti jel	Elnevezés	Megjegyzés
$F_0 = 0$		Zérus	konstans
$F_1 = xy$	$x \cdot y$	**AND (ÉS)**	x és y
$F_2 = xy'$	x/y	Inhibíció	x, de nem y
$F_3 = x$		Transzfer	x
$F_4 = x'y$	y/x	Inhibíció	y, de nem x
$F_5 = y$		Transzfer	y
$F_6 = xy' + x'y$	$x \oplus y$	**XOR**	x vagy y, de mindkettő nem
$F_7 = x + y$	$x + y$	**OR (VAGY)**	x vagy y
$F_8 = (x + y)'$	$x \downarrow y$	**NOR**	nem-VAGY
$F_9 = xy + x'y'$	$(x \oplus y)'$	**Ekvivalencia (XNOR)**	x egyenlő y-nal
$F_{10} = y'$	y'	**Komplemens**	nem y
$F_{11} = x + y'$	$x \subset y$	Implikáció	ha y, akkor x
$F_{12} = x'$	x'	**Komplemens**	nem x
$F_{13} = x' + y$	$x \supset y$	Implikáció	ha x, akkor y
$F_{14} = (xy)'$	$x \uparrow y$	**NAND**	nem-ÉS
$F_{15} = 1$		Egység	konstans

Megjegyzés: Az inhibíció és az implikáció nem kommutatív és nem asszociatív, nem érdemes használni.

Ismétlés: Logikai kapuk

- Az előző oldalakon felsorolt 16 függvény közül csak az alábbi nyolc terjedt el általánosan a digitális tervezés gyakorlatában

AND



$F = x \cdot y$

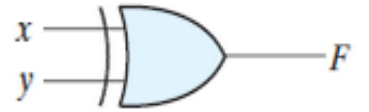
x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR

$F = x + y$

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

XOR



$F = xy' + x'y$
$= x \oplus y$

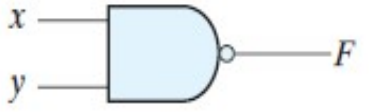
x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Buffer

$F = x$

x	F
0	0
1	1

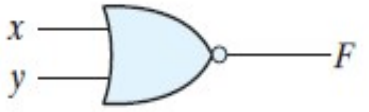
NAND



$F = (xy)'$

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

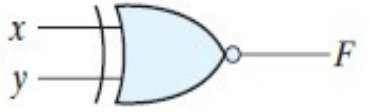
NOR



$F = (x + y)'$

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

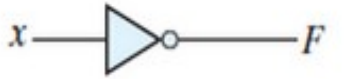
equivalence (XNOR)



$F = xy + x'y'$
$= (x \oplus y)'$

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

Inverter



$F = x'$

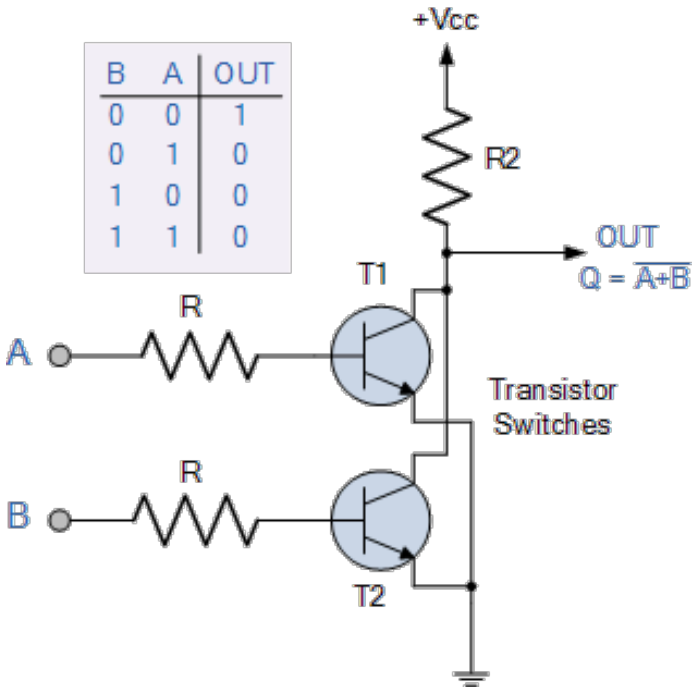
x	F
0	1
1	0

Resistor – Transistor Logics (RTL)

- A logikai kapuk egyik lehetséges megvalósítása az ellenállásokkal és tranzisztorokkal (mint kapcsolókkal) kivitelezett kapuáramkörök (angolul: Resistor – Transistor Logics, rövidítve: RTL)
- Néhány példa:

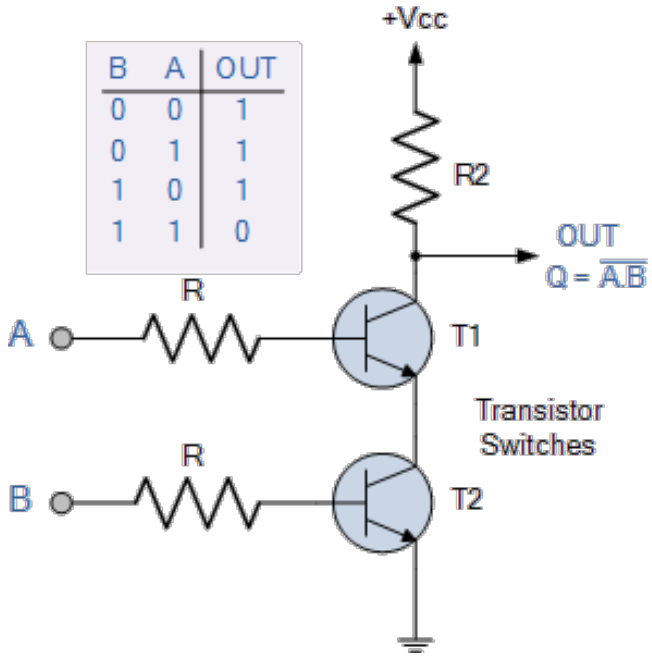
NOR (Nem VAGY) kapu
Bármelyik bemenet magas, valamelyik tranzisztor kinyit és alacsony lesz a kimenő feszültség

B	A	OUT
0	0	1
0	1	0
1	0	0
1	1	0



NAND (Nem ÉS) kapu
Bármelyik bemenet alacsony, a két sorbakötött tranzisztor nem vezet és magas lesz a kimeneti feszültség.

B	A	OUT
0	0	1
0	1	1
1	0	1
1	1	0

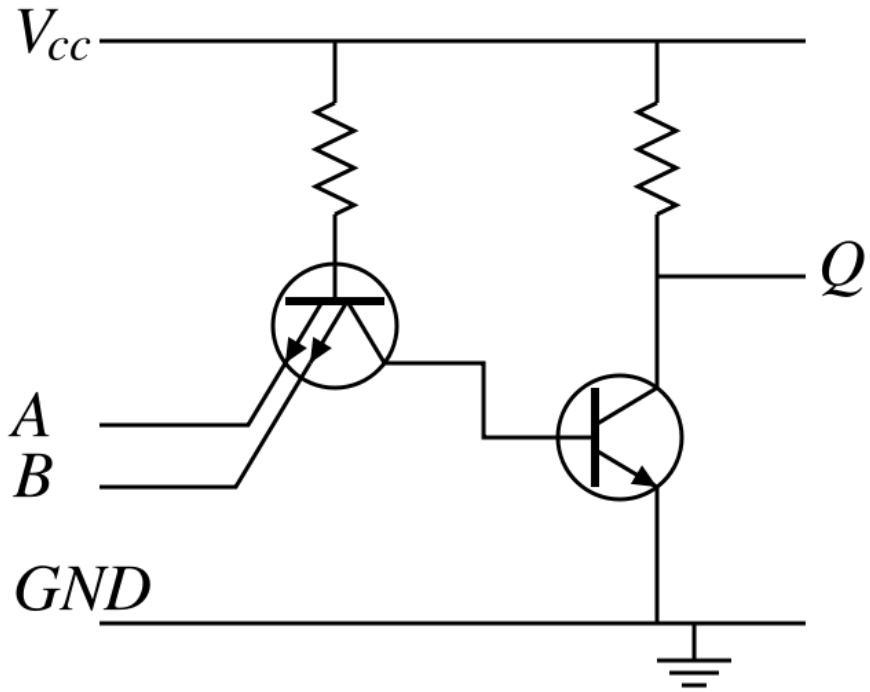


TTL Logikai áramkörök

A TTL (Transistor-Transistor Logic) kapcsolásban bipoláris tranzisztorok vannak, s a logikai bemenetek többemitteres tranzisztorok emitter kivezetései.

1. A több emitterrel rendelkező tranzisztorok működése egyenértékű azzal, ha több tranzisztor bázisát és kollektorát összekötjük.
2. Az ábrán látható kétbemenetű NEM-ÉS (NAND) áramkör ebben a leegyszerűsített formában nem mentes az RTL áramkörök ismert problémáitól.

   A probléma a következő oldalon bemutatott totem-pole kimenettel orvosolható
3. TTL felépítésűek a klasszikus **74xx sorozatú** logikai IC-k



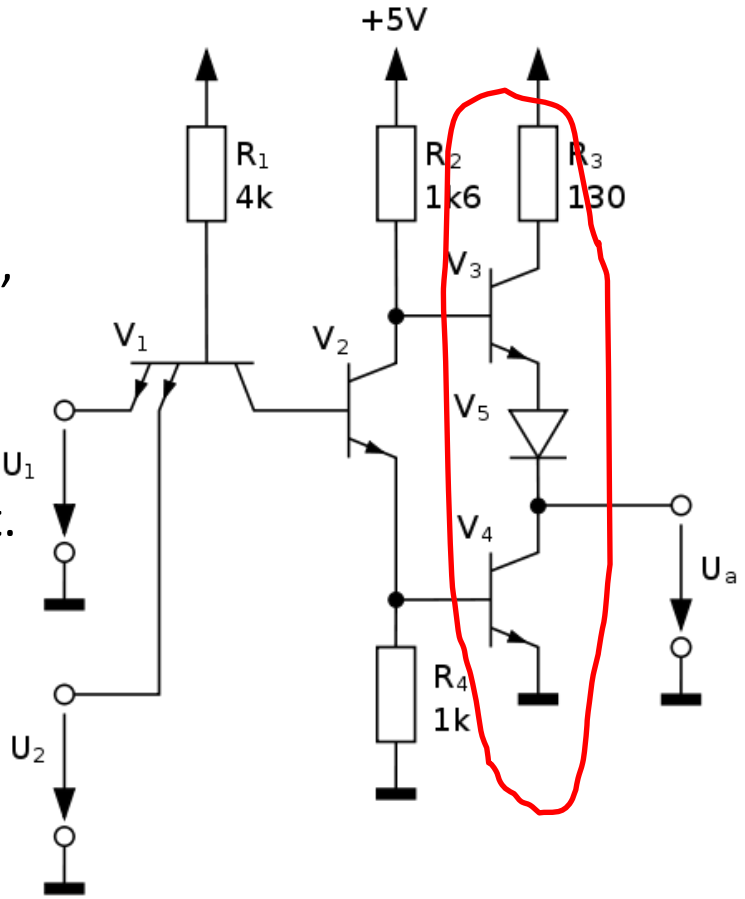
Két bemenetű NAND áramkör
Bármelyik bemenetet alacsony szintre húzzuk, a második tranzisztor lezár és a kimenet magas szintre kerül (NEM-ÉS kapu)

TTL kapu totem-pole kimenettel

Totem-pole = totem oszlop. Fából faragott oszlop, figurái egymás fölött helyezkednek el, „egymás fején ülnek”… A TTL kimenet elnevezése onnan ered, hogy itt is egymás tetejére ültetett alkatrészeket látunk (az ábrán V3, V5 és V4).

Kétbemenetű TTL NAND áramkör totem-pole kimenettel

1. Alaphelyzetben V2 és V4 vezet, a kimenet alacsony szintű.
2. Ha valamelyik bemenetet lehúzzuk, V2 és V4 nem vezet, V3 vezet: magas kimeneti szint.



Hátrány: viszonylag alacsony kimeneti magas szint (~3,5 V)

TTL Inverter szimulációja

A http://www.falstad.com/circuit/ címen elérhető áramkör szimulátor segítségével vizsgáljuk a kapcsolás működését!
A Circuits/Logic Families/TTL Inverter mintapéldát nézzük meg!

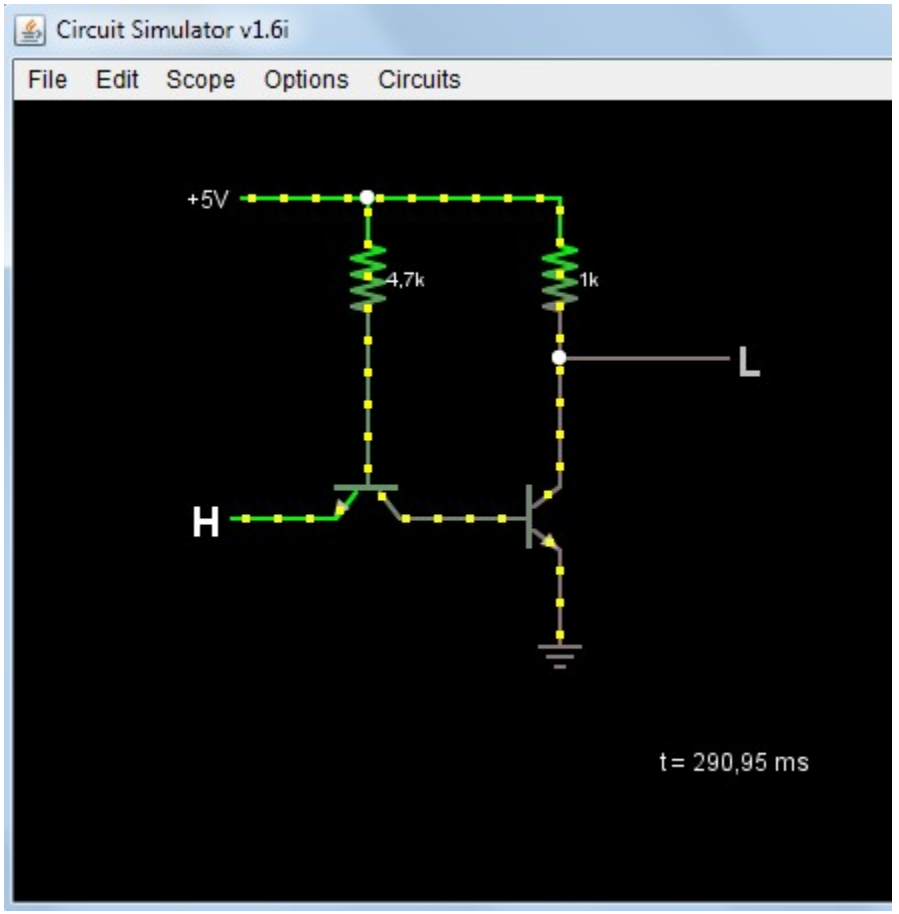
Baloldalt a bemenet állapota egérkattintással váltogatható:

H = magas szint (logikai 1)

L = alacsony szint (logikai 0)

Jobboldalt a kimenet állapota látható.

Len a be- és kimeneti feszültségek oszcilloszkópos megjelenítése látható.



TTL Inverter (NEM áramkör)

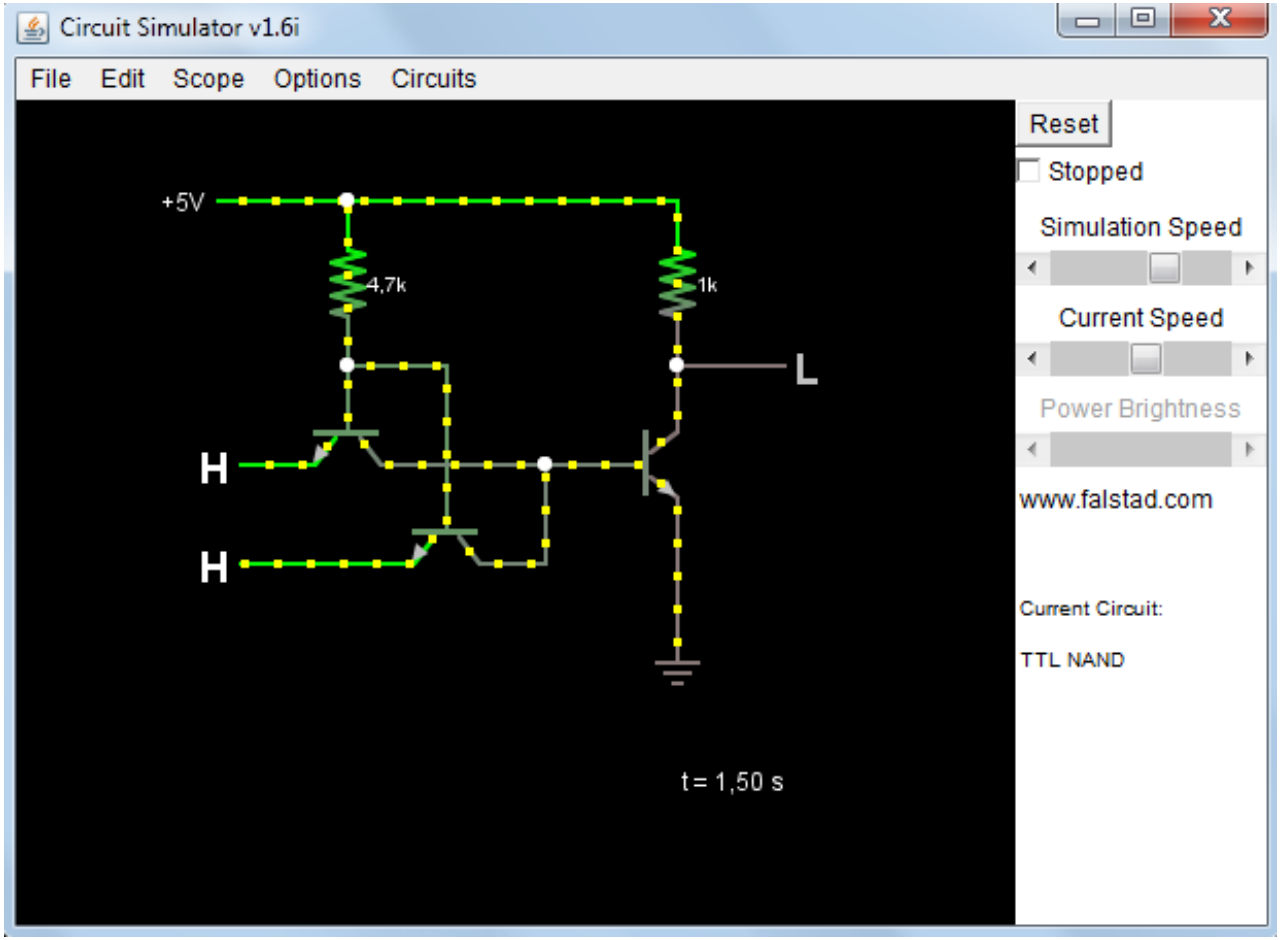
TTL NAND (NEM-ÉS)

A http://www.falstad.com/circuit/ címen elérhető áramkör szimulátor segítségével vizsgáljuk a kapcsolás működését!
A Circuits/Logic Families/TTL NAND mintapéldát nézzük meg!

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

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

A kimenet csak akkor alacsony, ha minden bemenet magas szinten van

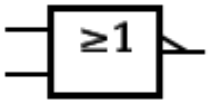


2 bemenetű TTL NAND (NEM-ÉS)

TTL NOR (NEM-VAGY)

A http://www.falstad.com/circuit/ címen elérhető áramkör szimulátor segítségével vizsgáljuk a kapcsolás működését!
A Circuits/Logic Families/TTL NOR mintapéldát nézzük meg!

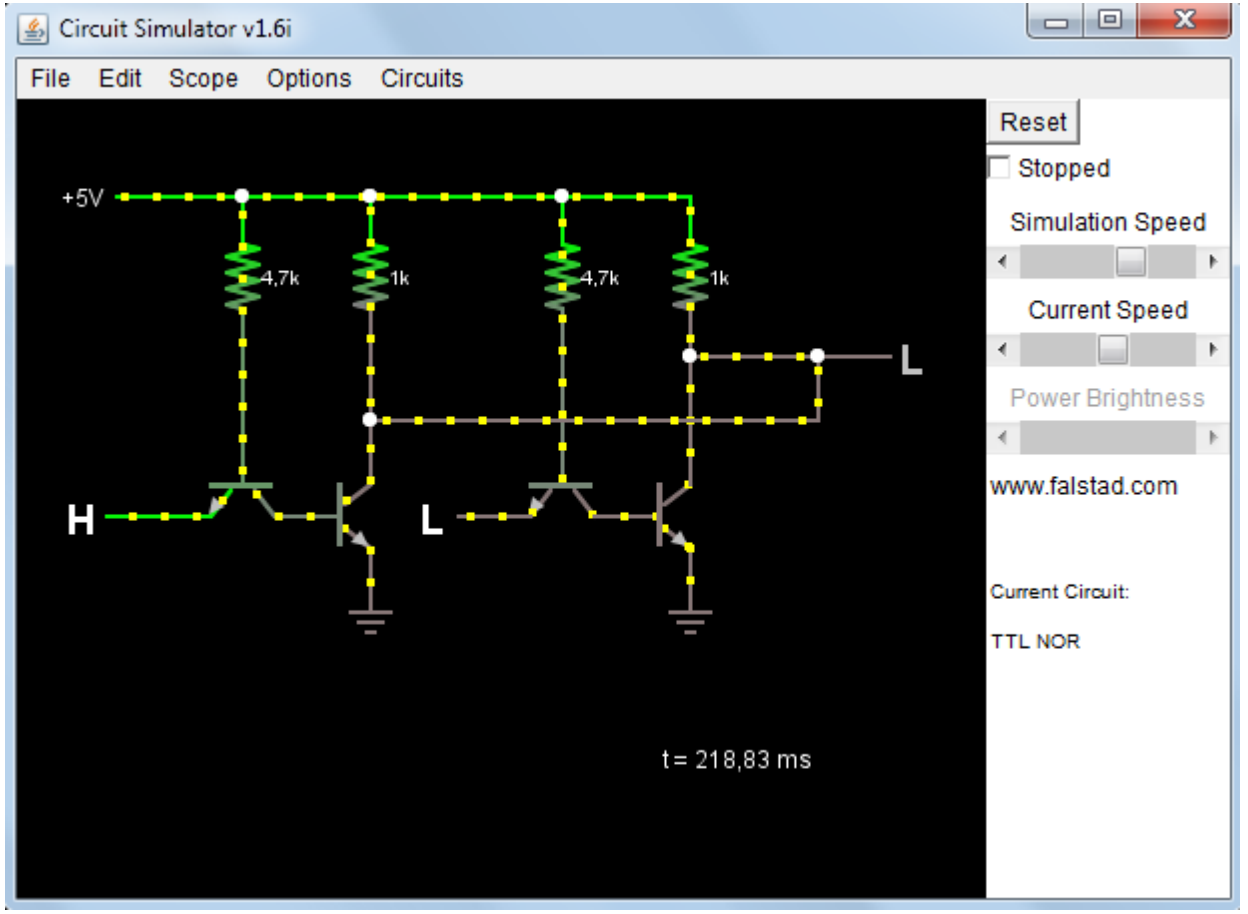
Rajzjele:



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

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

A kimenet csak akkor magas, ha minden bemenet alacsony szinten van

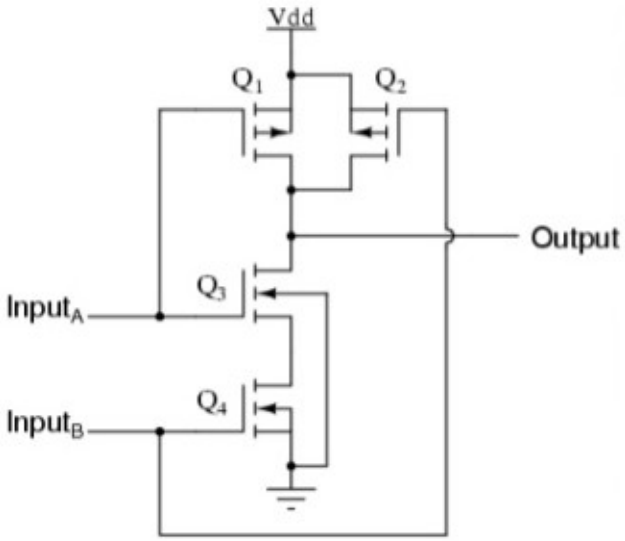


2 bemenetű TTL NOR (NEM-VAGY)

CMOS kapuk

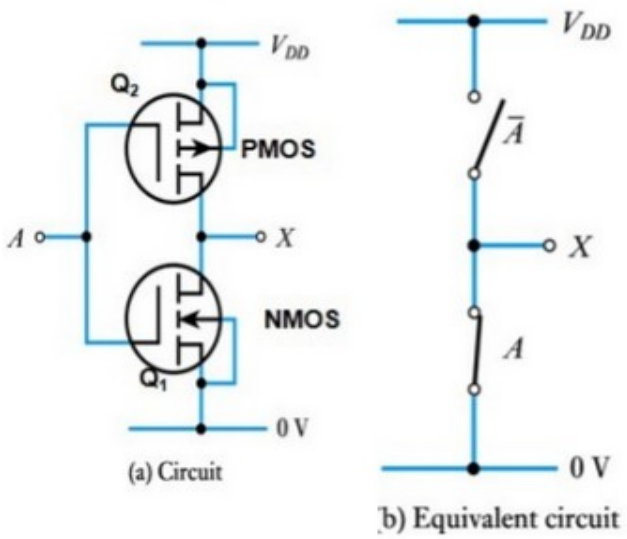
CMOS = Complementary MOS, p- és n-csatornás FET-ekből épülnek fel.

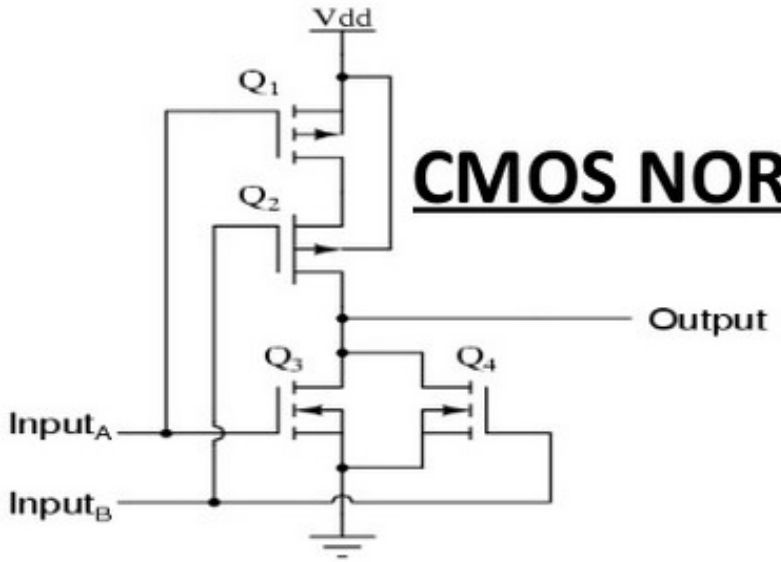
CMOS NAND



Input		Transistor				O/P
A	B	Q1	Q2	Q3	Q4	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

CMOS inverter



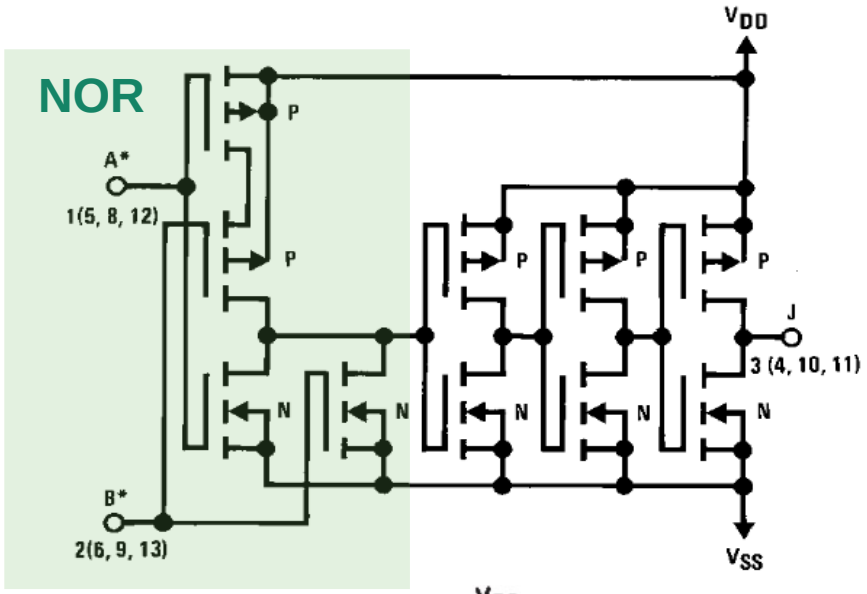


CMOS NOR

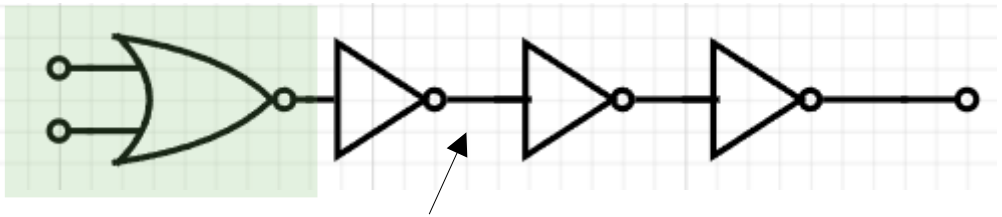
Input		Transistor	Output
A	B	Q1 Q2 Q3 Q4	Y
0	0	ON ON OFF OFF	1
0	1	ON OFF OFF ON	0
1	0	OFF ON ON OFF	0
1	1	OFF OFF ON ON	0

CD4071B (OR) áramköri megvalósítások

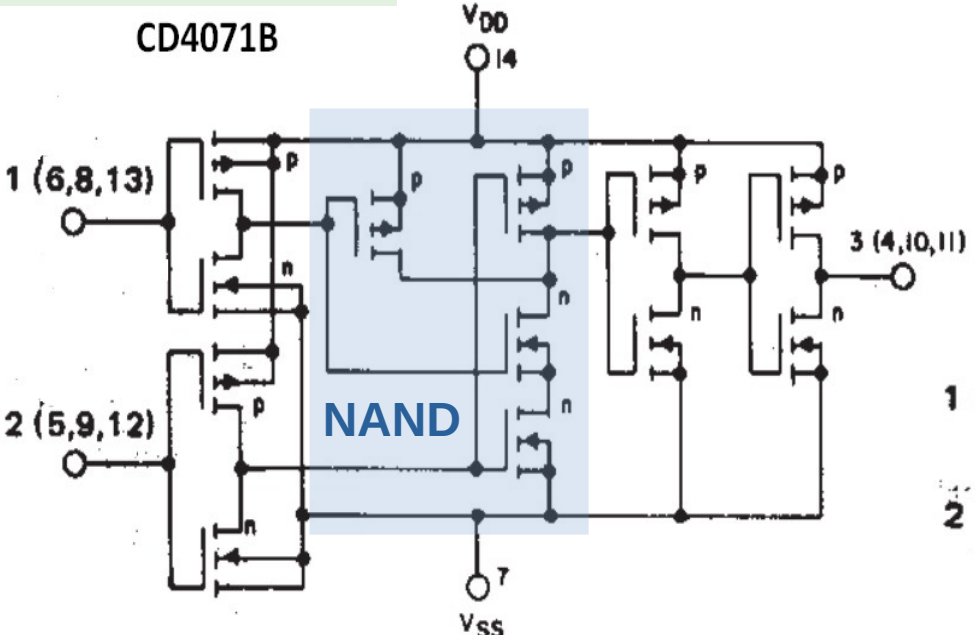
CD4071B



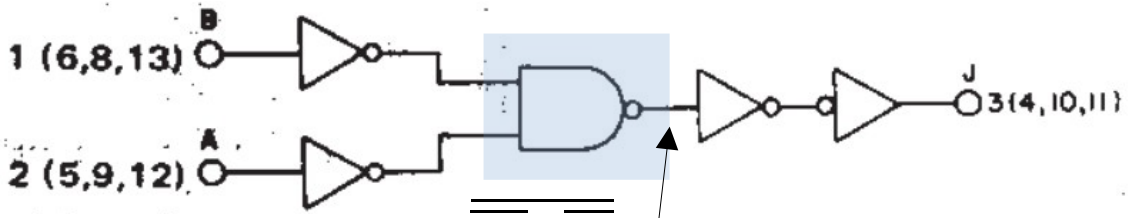
FAIRCHILD SEMICONDUCTOR®



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



TEXAS INSTRUMENTS

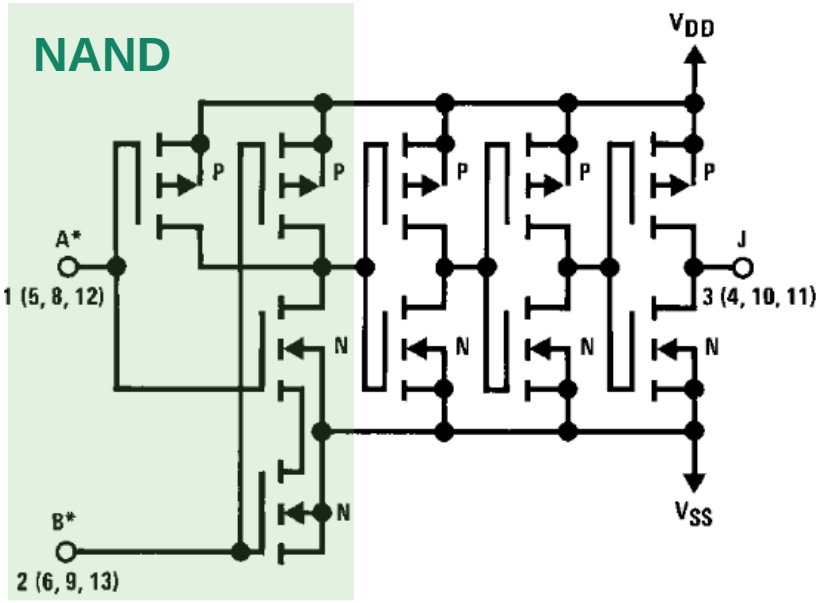


$$\overline{\overline{A}\cdot\overline{B}}=A+B$$

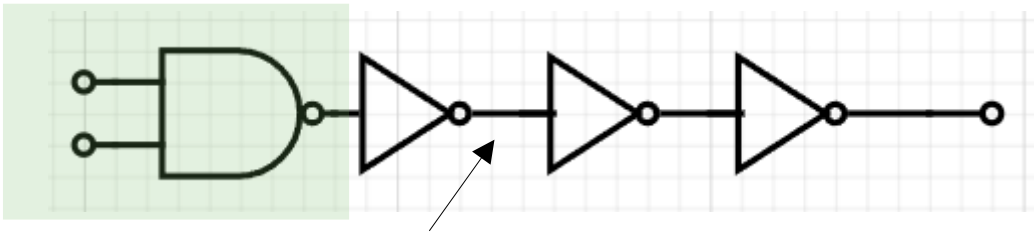
CD4081B (AND) áramköri megvalósítások

CD4081B

NAND



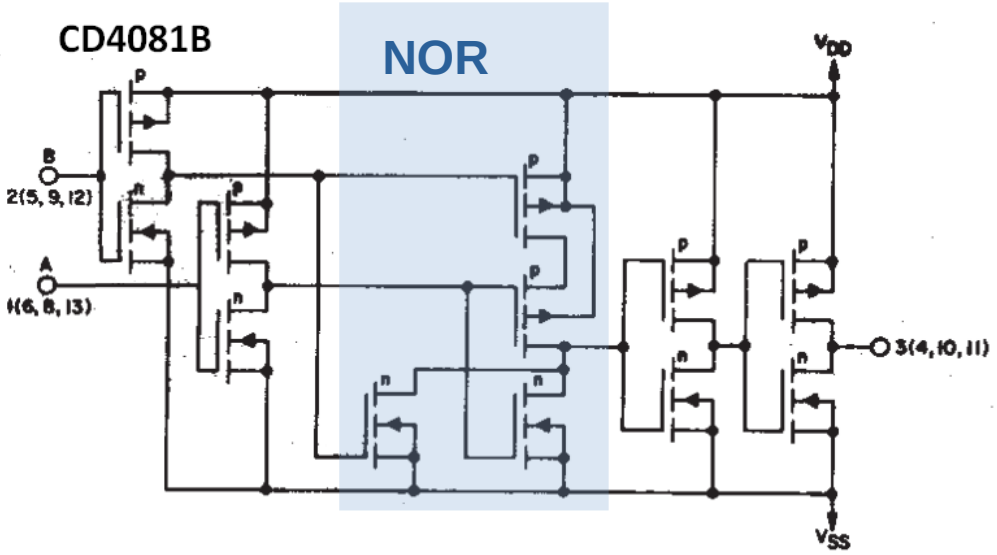
FAIRCHILD SEMICONDUCTOR®



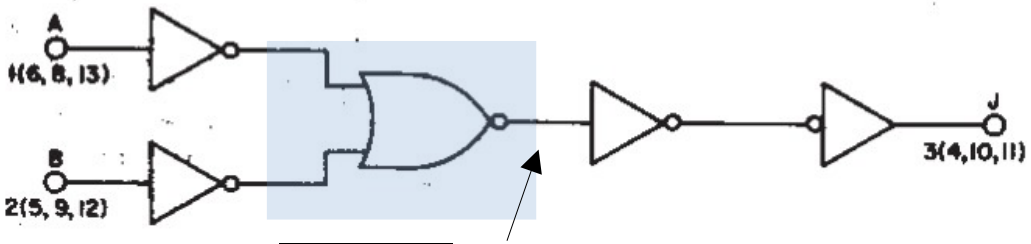
$$\overline{\overline{A \cdot B}} = A \cdot B$$

CD4081B

NOR



TEXAS INSTRUMENTS



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

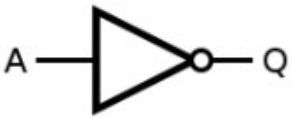
Funkcionális teljesség

- A logikában a **funkcionális teljesség** azt jelenti, hogy logikai műveletek egy halmazából minden lehetséges igazságtáblázat megvalósítható a halmaz műveleteinek kombinálásával
- **Funkcionálisan teljes** például az **{AND, NOT},** halmaz, viszont az **{AND, OR}** halmaz funkcionálisan hiányos, mivel hiányzik belőle a komplemensképzés (**NOT**)
- Különleges tulajdonságú a **NAND** vagy a **NOR** logikai művelet, mivel ezek önmagukban is funkcionálisan teljes halmazt képeznek. Az ilyen műveletet áramköri megvalósítását **univerzális kapunak** nevezzük
- Konkrétan az összes logikai kapu (és ezek tetszőleges hálózata) összeállítható csupán bináris **NAND** kapukból, vagy csupán bináris **NOR** kapukból. Az előbbit NAND logikának, az utóbbit NOR logikának nevezzük
- Annak bizonyításához, hogy a logikai műveletek egy adott halmaza **funkcionálisan teljes,** elegendő megmutatni, hogy az első előadásban definiált alapműveletek (**AND, OR** és **NOT** – azaz **ÉS, VAGY** és **NEM**) műveletek megvalósíthatók az adott halmaz műveleteivel

NAND logika

- A **NAND** kapu **univerzális kapu,** tehát bármelyik más logikai kapu megvalósítható **NAND** kapuk összekapcsolásával

NOT

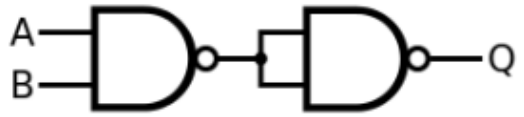


Q = NOT(**A**)

= **A** NAND **A**

AND

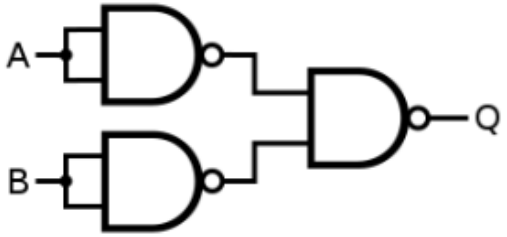
Q = **A** AND **B**



= (**A** NAND **B**) NAND (**A** NAND **B**)

OR

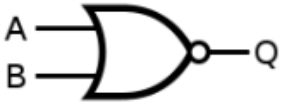
Q = **A** OR **B**



= (**A** NAND **A**) NAND (**B** NAND **B**)

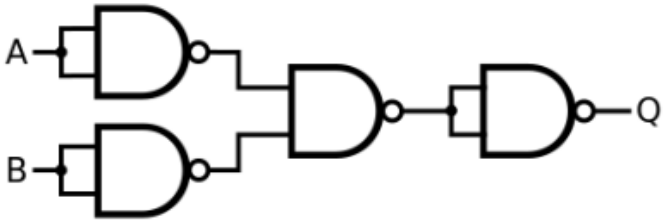
$\overline{\overline{A}\cdot\overline{B}} = A + B$

NOR



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

Q = **A** NOR **B**

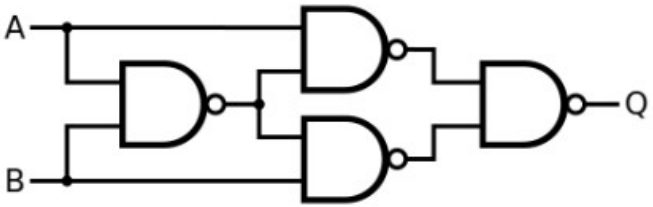


= [(**A** NAND **A**) NAND (**B** NAND **B**)] NAND [(**A** NAND **A**) NAND (**B** NAND **B**)]

XOR

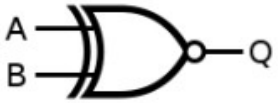
$A\cdot\overline{B} + \overline{A}\cdot B$

Q = **A** XOR **B**



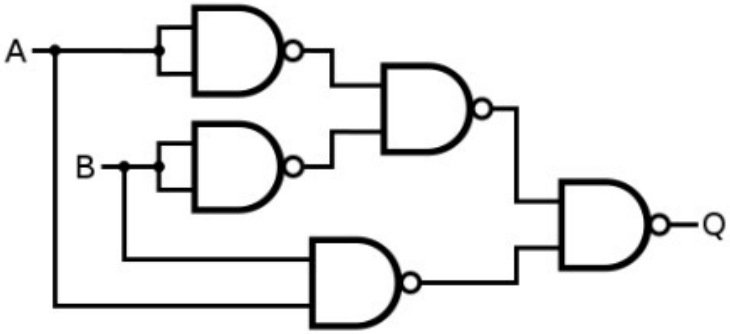
= [**A** NAND (**A** NAND **B**)] NAND [**B** NAND (**A** NAND **B**)]

XNOR



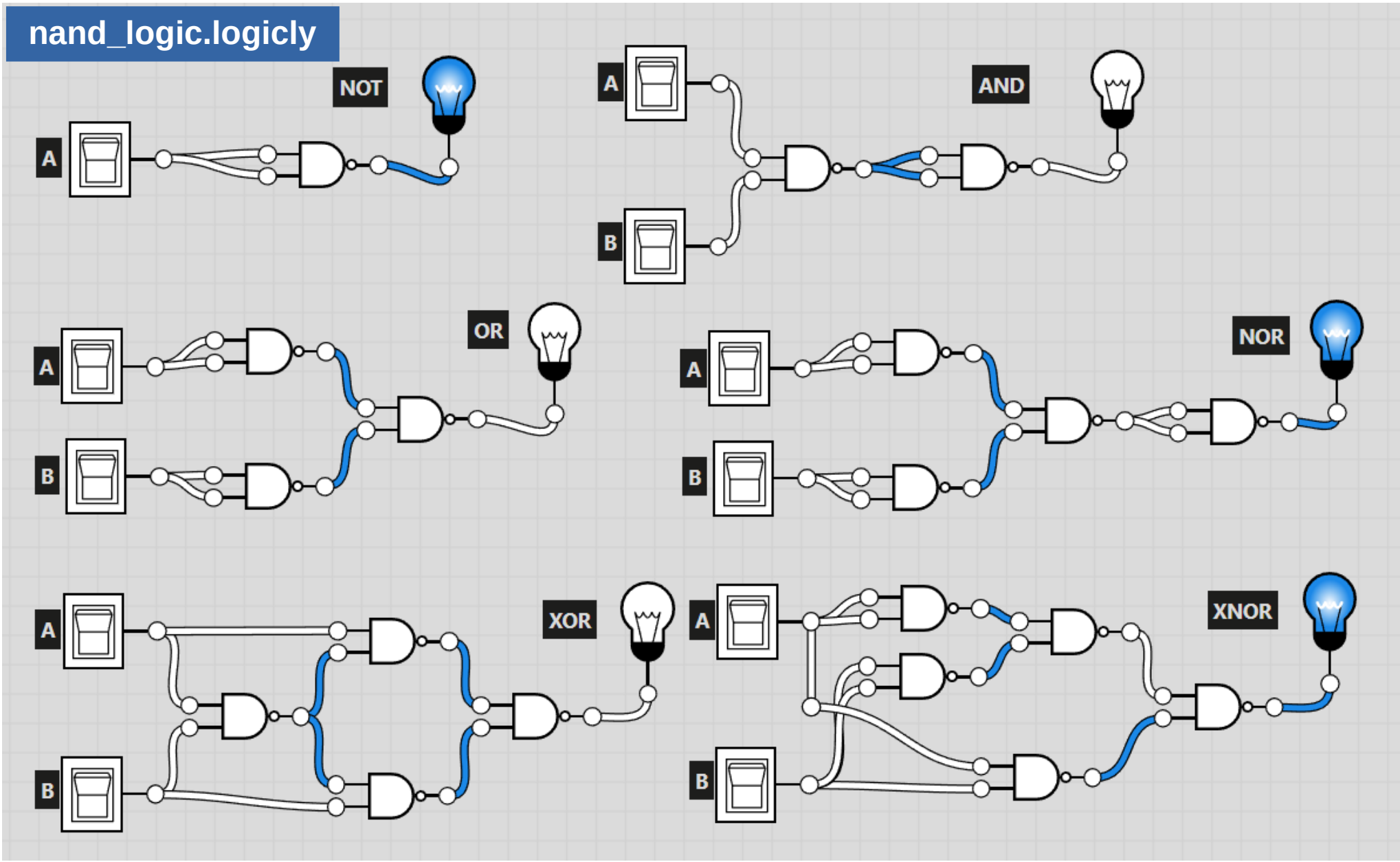
$A\cdot B + \overline{A}\cdot\overline{B}$

Q = **A** XNOR **B**



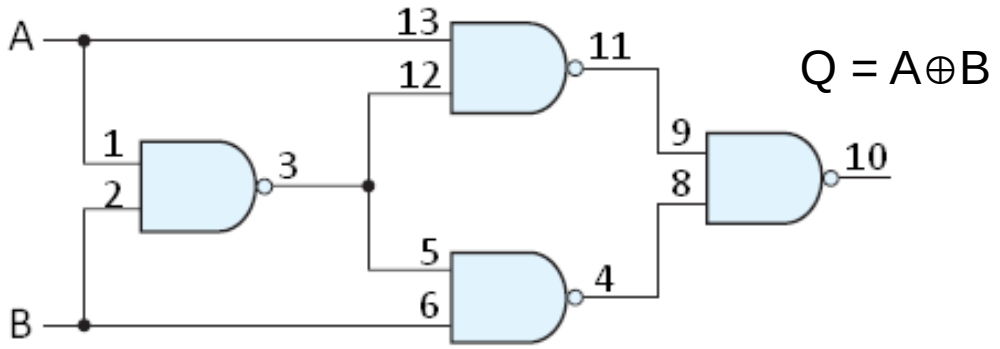
= [(**A** NAND **A**) NAND (**B** NAND **B**)] NAND (**A** NAND **B**)

NAND logika

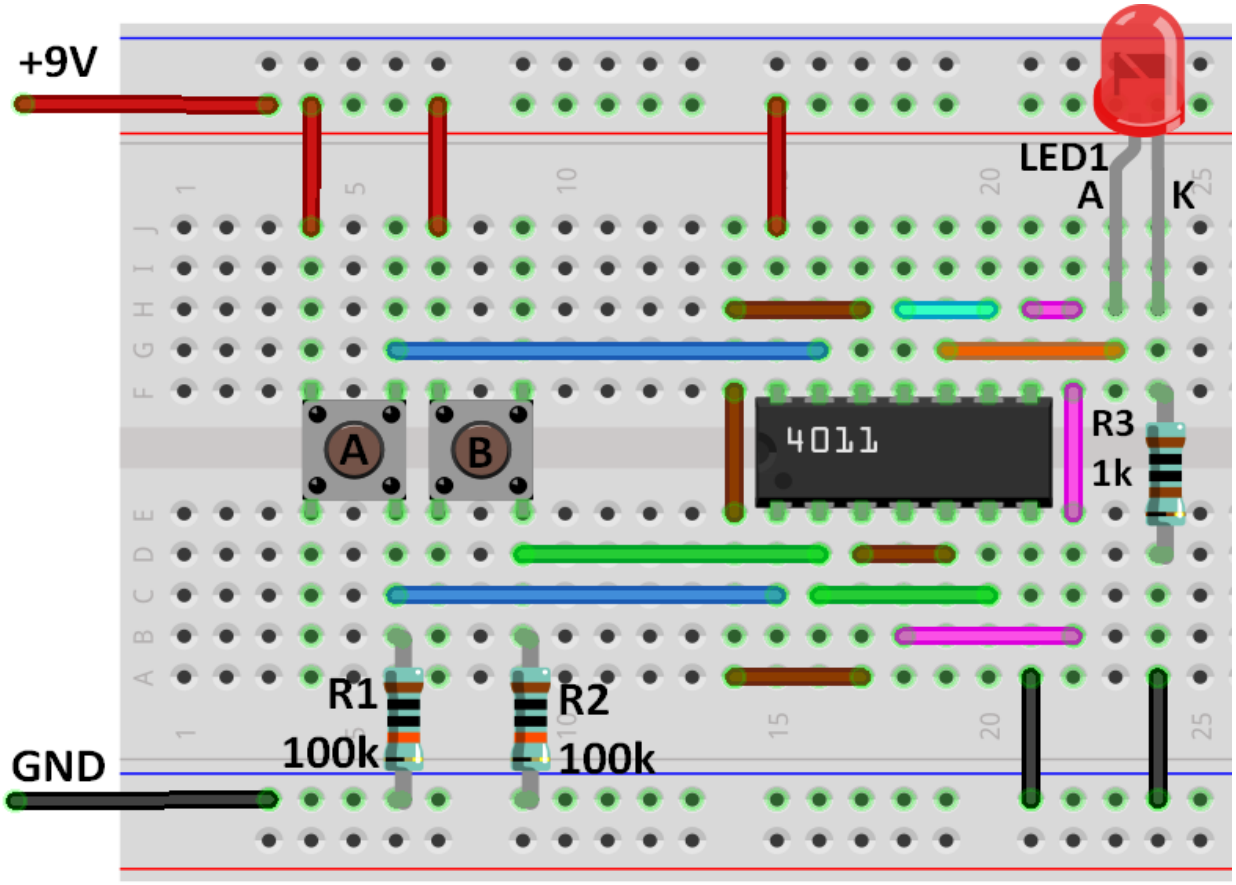


XOR kapu NAND áramkörökből

Építsük meg az alábbi kapcsolást, s igazoljuk, hogy
az igazságtáblázata a XOR műveletnek felel meg!



A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0



XOR kapu, mint inverter

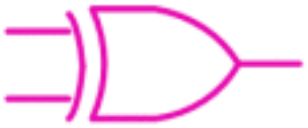
Az **XOR** kapu kimenentén nincs invertálás, mégis felhasználhatjuk inverterként. Az igazságtáblázatából kiolvashatjuk, hogy:

- Amikor a **B** bemenet nulla, az **Y** kimenet az **A** bemenettel megegyezik.
- Amikor a **B** bemenet '1', akkor az **Y** kimenet az **A** bemenettel ellentétes.

Az **XNOR** kapu esetében a **B** bemenet szerepe megfordul:

- Amikor a **B** bemenet nulla, az **Y** kimenet az **A** bemenettel ellentétes.
- Amikor a **B** bemenet '1', akkor az **Y** kimenet az **A** bemenettel megegyezik.

$Y = A \oplus B$ **XOR**

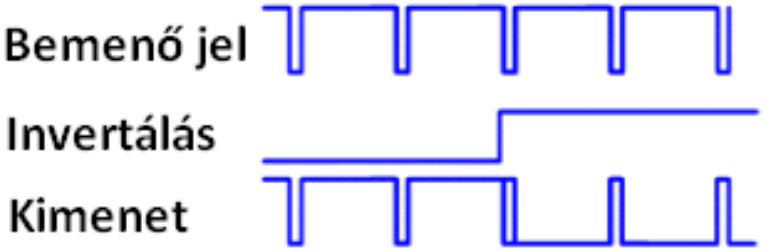


A	B	Y
0	0	0
1	0	1
0	1	1
1	1	0

XNOR

$Y = \overline{A \oplus B}$

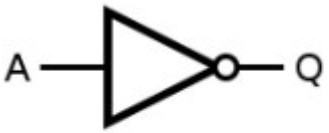
A	B	Y
0	0	1
1	0	0
0	1	0
1	1	1



NOR logika

- A **NOR** kapu is **univerzális kapu,** tehát bármelyik más logikai kapu megvalósítható **NOR** kapuk összekapcsolásával is

NOT

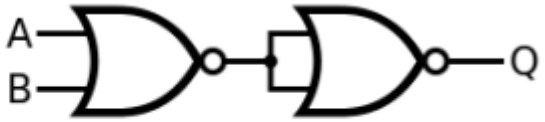


Q = NOT(**A**)

= **A** NOR **A**

OR

Q = **A** OR **B**

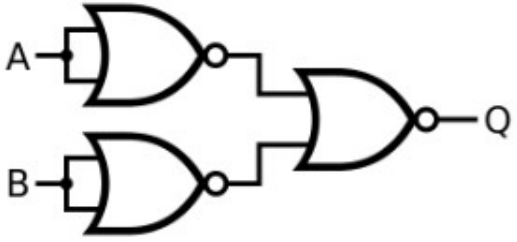


= (**A** NOR **B**) NOR (**A** NOR **B**)

AND

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

Q = **A** AND **B**

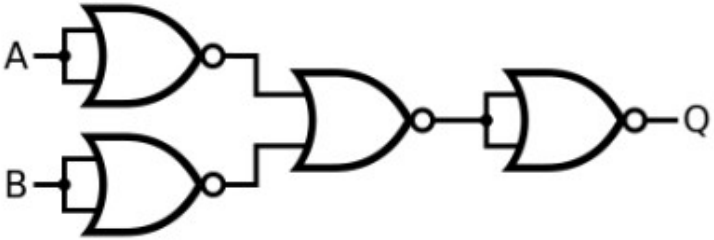


= (**A** NOR **A**) NOR (**B** NOR **B**)

NAND

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

Q = **A** NAND **B**

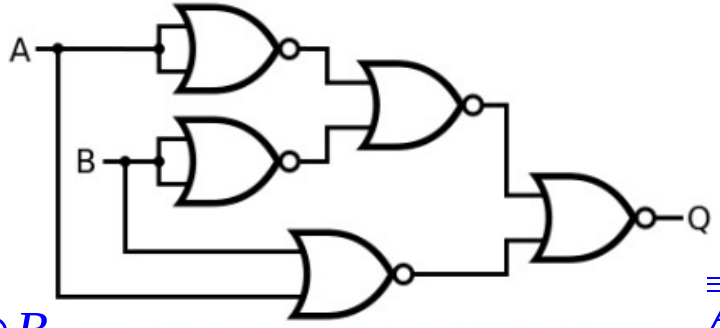


= [(**A** NOR **A**) NOR (**B** NOR **B**)] NOR [(**A** NOR **A**) NOR (**B** NOR **B**)]

XOR

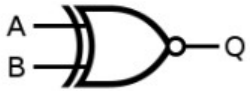
$\overline{A\cdot B+\overline{A+B}}=A\oplus B$

Q = **A** XOR **B**



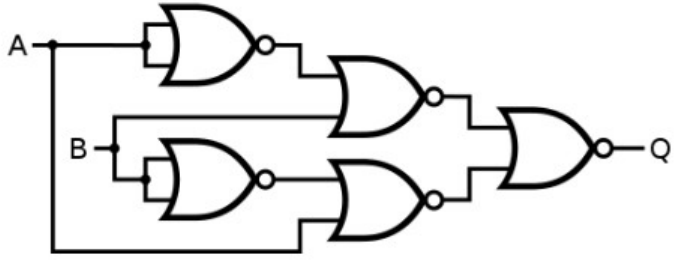
= [(**A** NOR **A**) NOR (**B** NOR **B**)] NOR (**A** NOR **B**)

XNOR



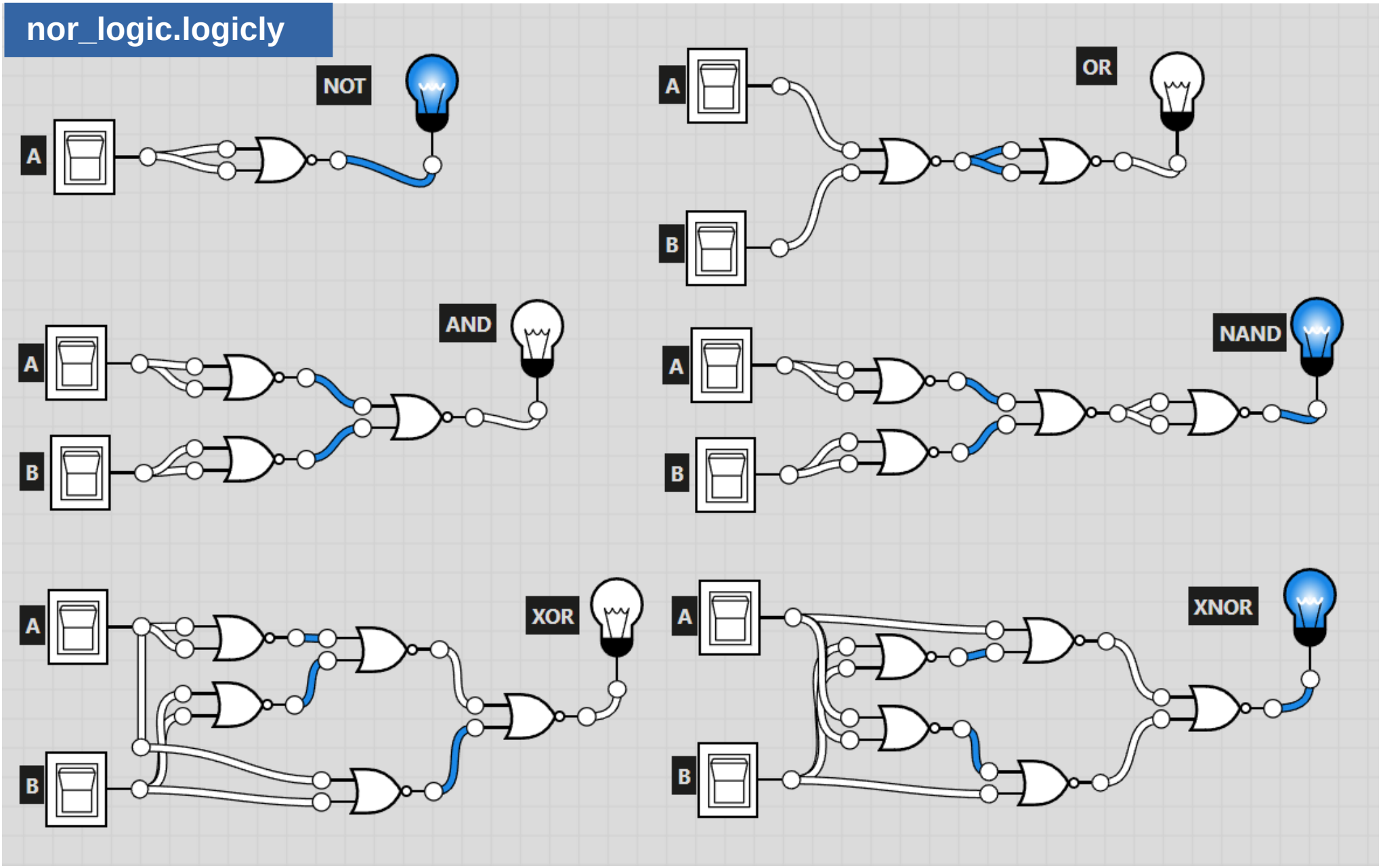
$\overline{\overline{\overline{A}+B}+\overline{A+\overline{B}}}=\overline{A\oplus B}$

Q = **A** XNOR **B**



= [**B** NOR (**A** NOR **A**)] NOR [**A** NOR (**B** NOR **B**)]

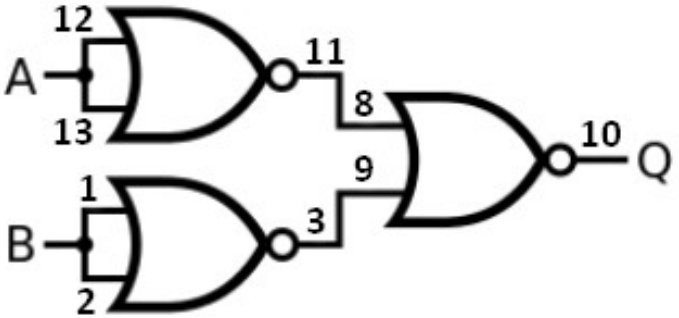
NOR logika



ÉS kapu építése NOR kapukból

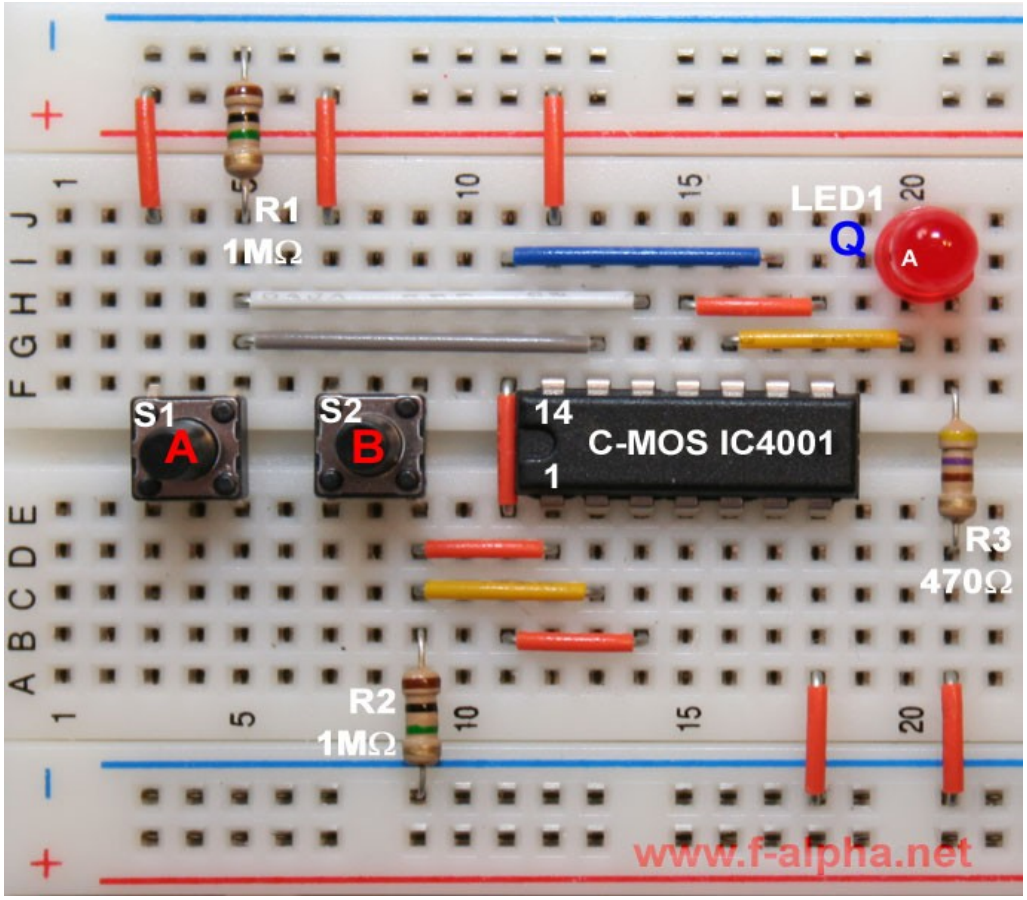
Építsünk ÉS kaput NOR (nem-VAGY) kapukból, és ellenőrizzük a működést!

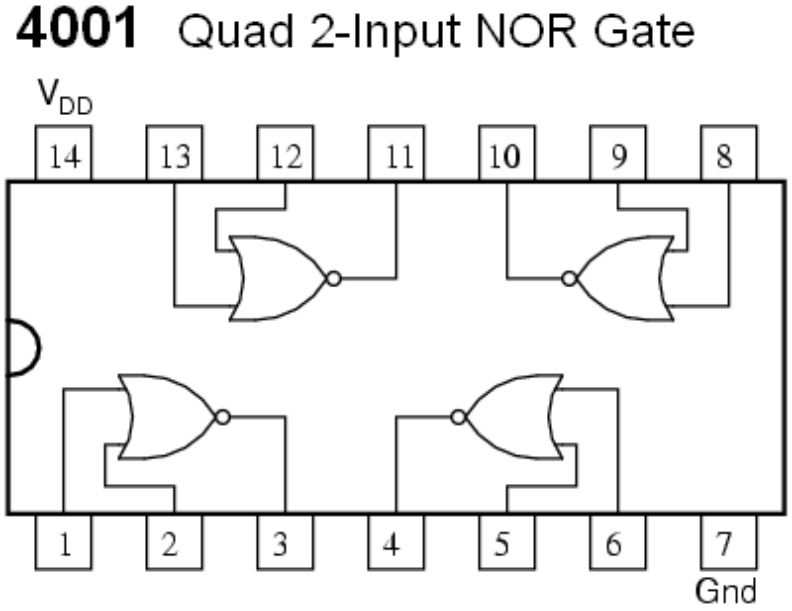
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



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

(De Morgan szabály alapján)

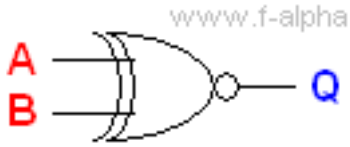




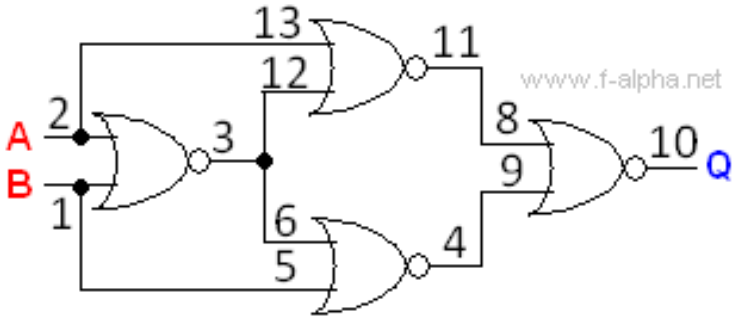
Forrás: en.f-alpha.net/electronics/digital-electronics/boolean-logic/lets-go/experiment-10-logic-with-nor-gates-ii/

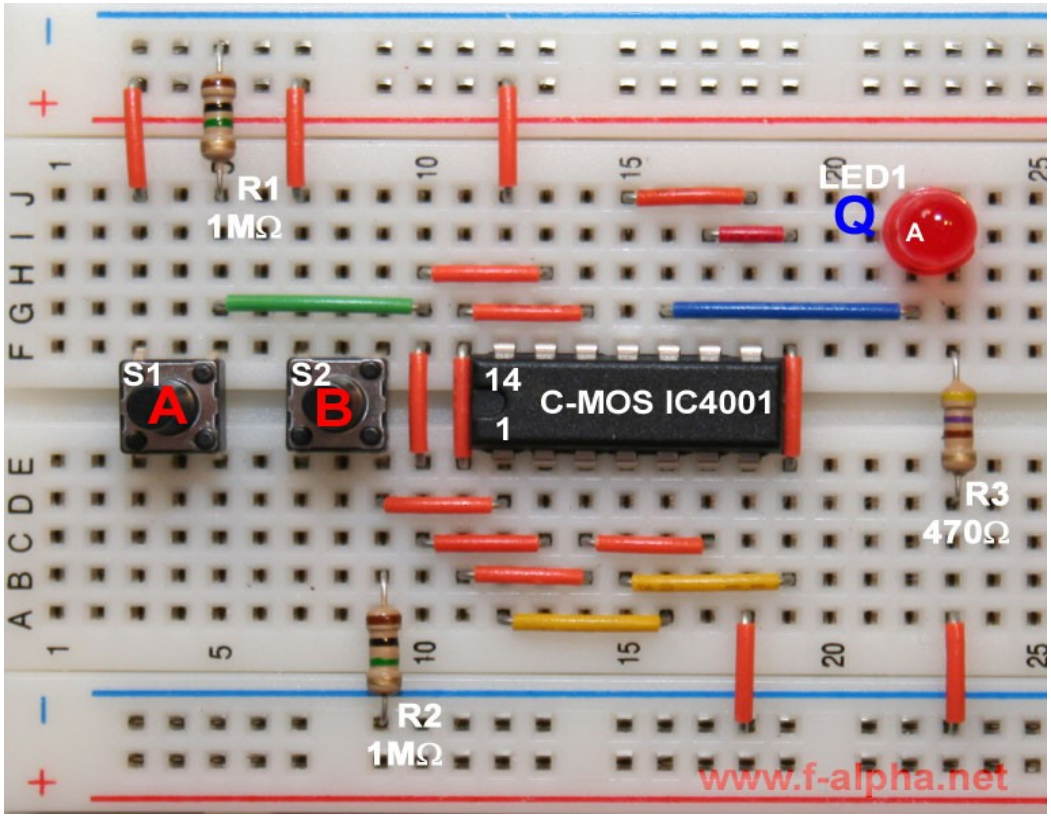
XNOR kapu építése NOR kapukból

XNOR = nem-XOR
(ellentettje az XOR-nak)

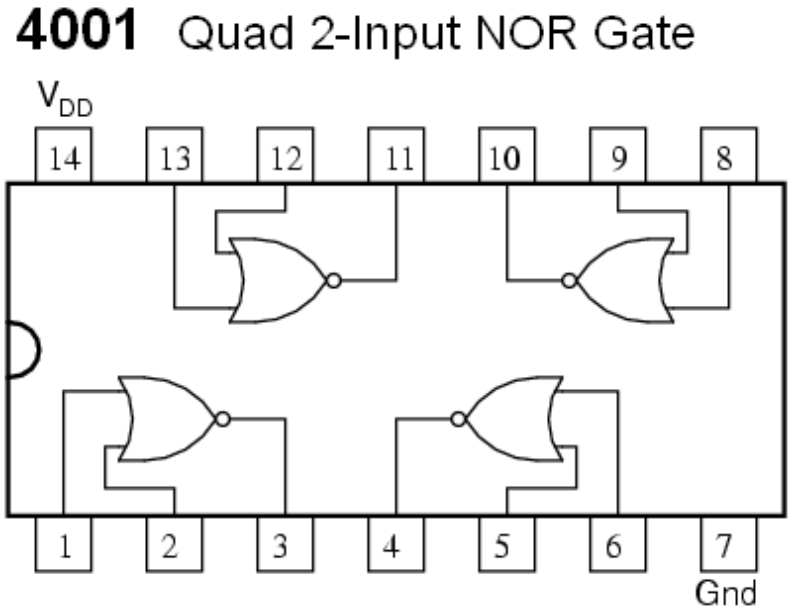


A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1



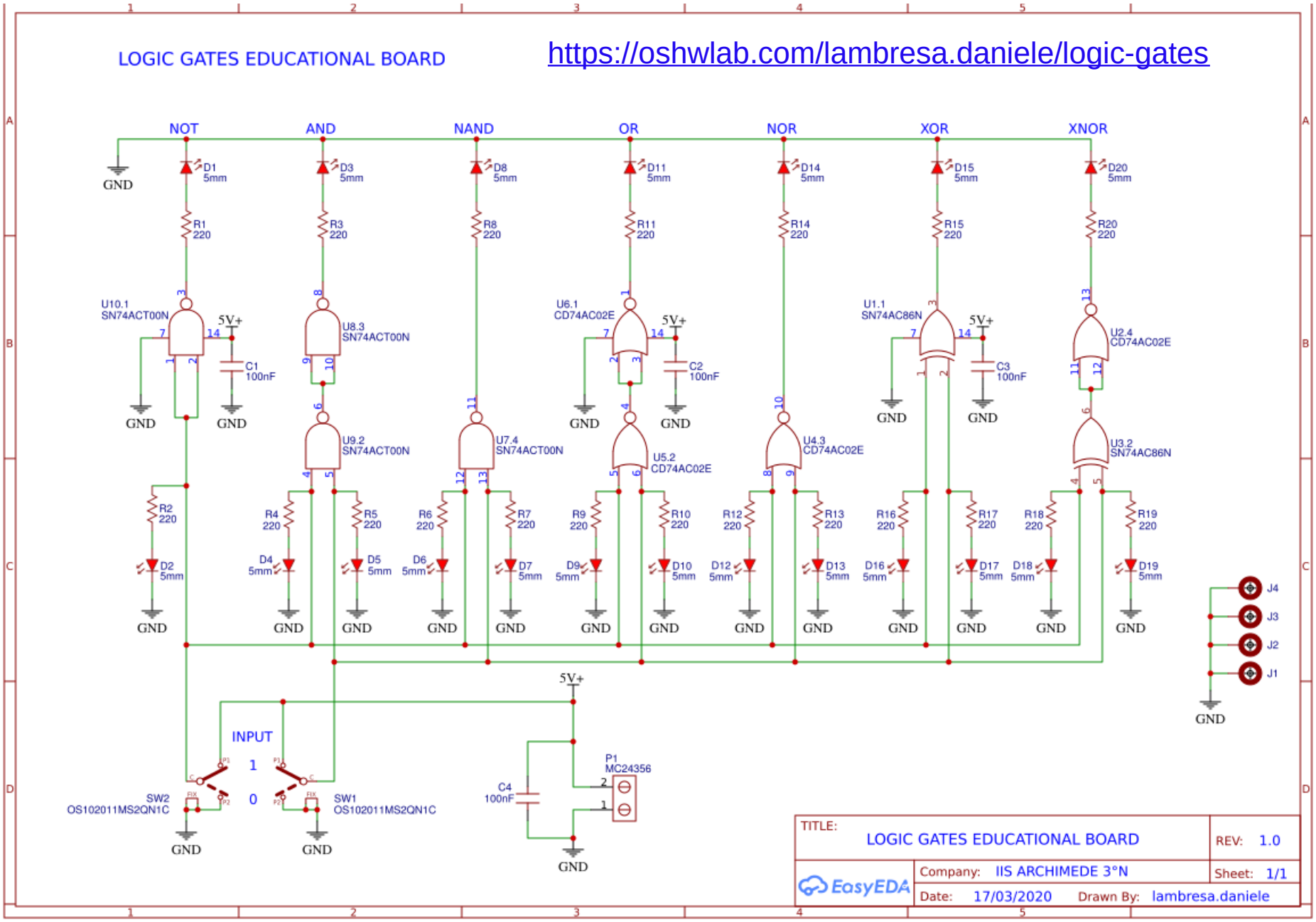


Tipp: Az ekvivalens lábak (8-9) felcserélésével a 18. oldalon bemutatott kapcsolási elrendezést is használhatjuk



Forrás: en.f-alpha.net/electronics/digital-electronics/boolean-logic/lets-go/experiment-10-logic-with-nor-gates-ii/

Példa a NAND és NOR logika használatára



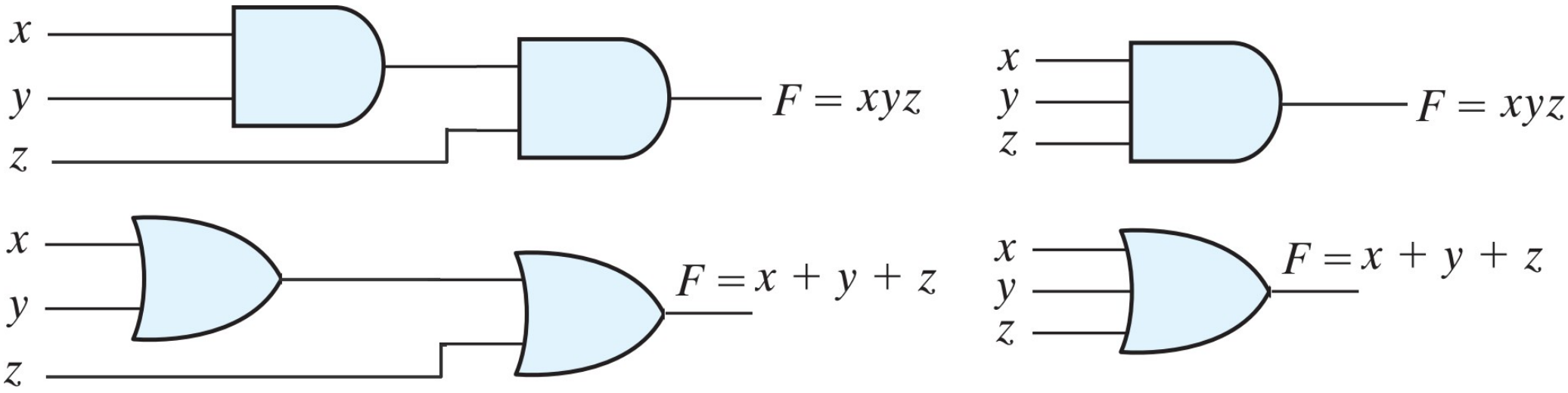
Több-bemenetű logikai kapuk

- Az **AND** (ÉS), illetve az **OR** (VAGY) műveletek asszociatív és kommutatív tulajdonságai miatt a kiterjesztés magától értetődő

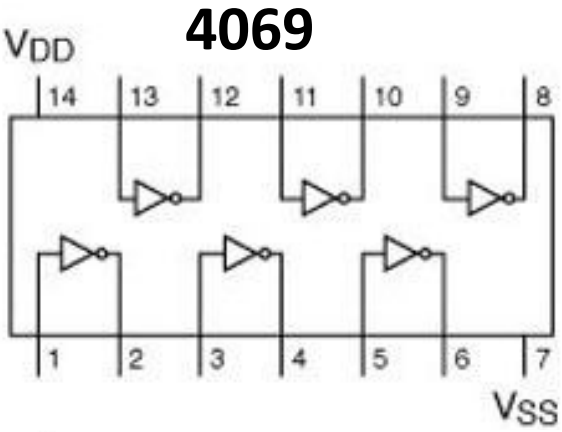
  Például az **OR** műveletre írhatjuk, hogy:

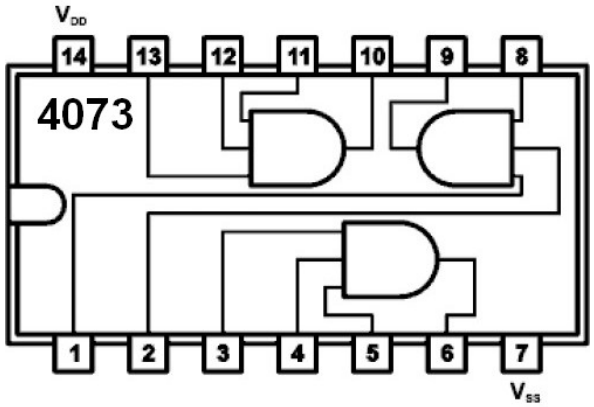
$$x + y = y + x \quad \text{(commutative)}$$

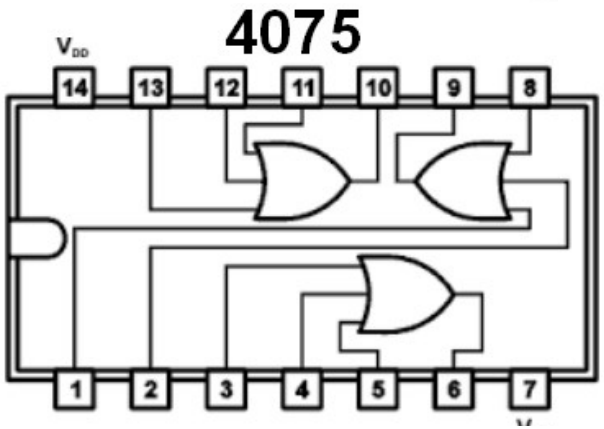
$$(x + y) + z = x + (y + z) = x + y + z \quad \text{(associative)}$$

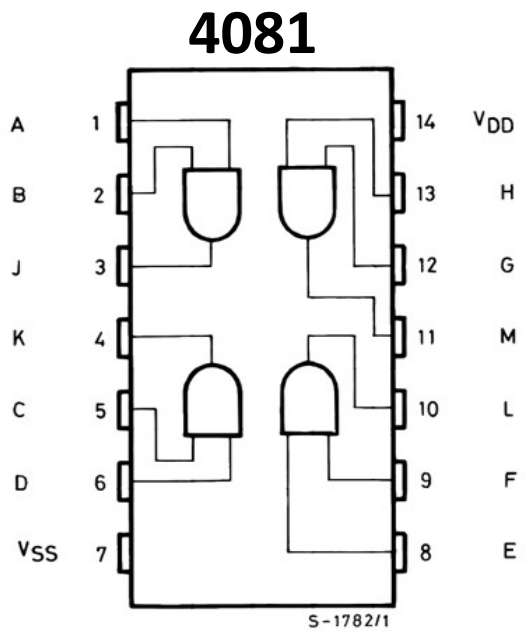


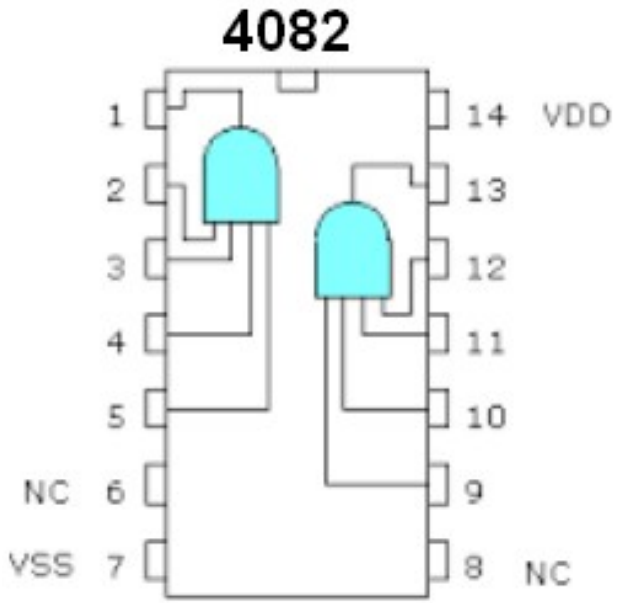
A 4000-es sorozat tipikus tagjai

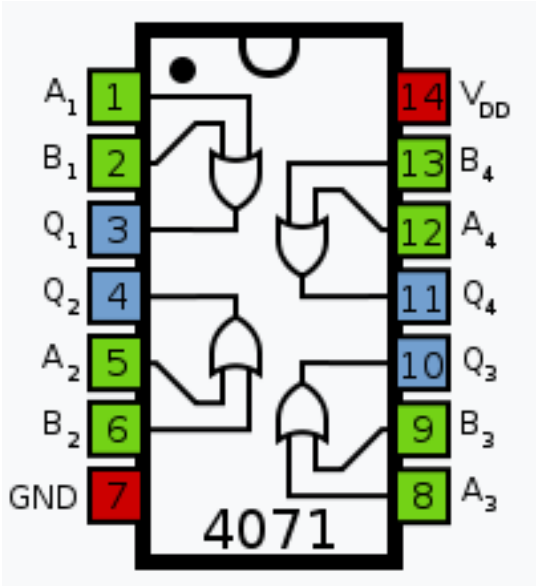


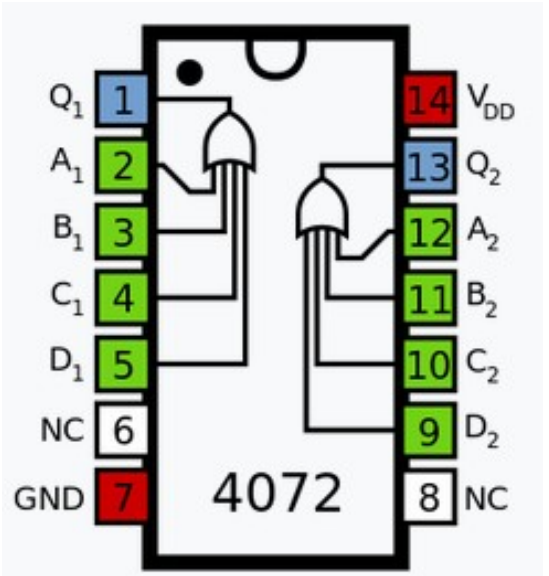










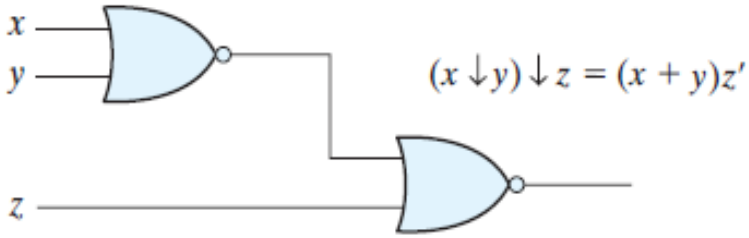


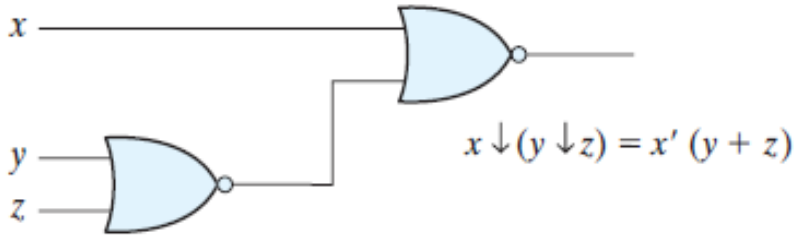
Több-bemenetű NAND és NOR kapuk

- A **NOR** és **NAND** műveleteknél, bár kommutatívak, gondot okoz, hogy nem asszociatívak. Például a NOR esetében:

$$(x \downarrow y) \downarrow z = [(x + y)' + z]' = (x + y)z' = xz' + yz'$$
$$x \downarrow (y \downarrow z) = [x + (y + z)']' = x'(y + z) = x'y + x'z$$

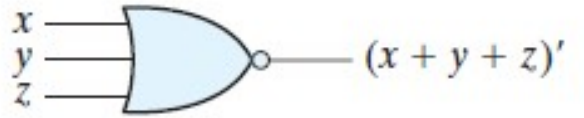
↓ NOR művelet
↑ NAND művelet



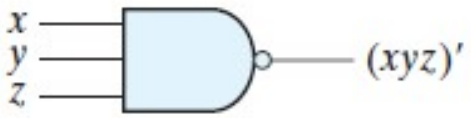


- Ezen a nehézségen úgy lehetünk úrrá, ha a több-bemenetű **NOR** (illetve **NAND**) kaput ***úgy definiáljuk,*** mint **komplementált kimenetű,** több-bemenetű **VAGY** (illetve **ÉS**) kapu:

$$x \downarrow y \downarrow z = (x + y + z)'$$
$$x \uparrow y \uparrow z = (xyz)'$$



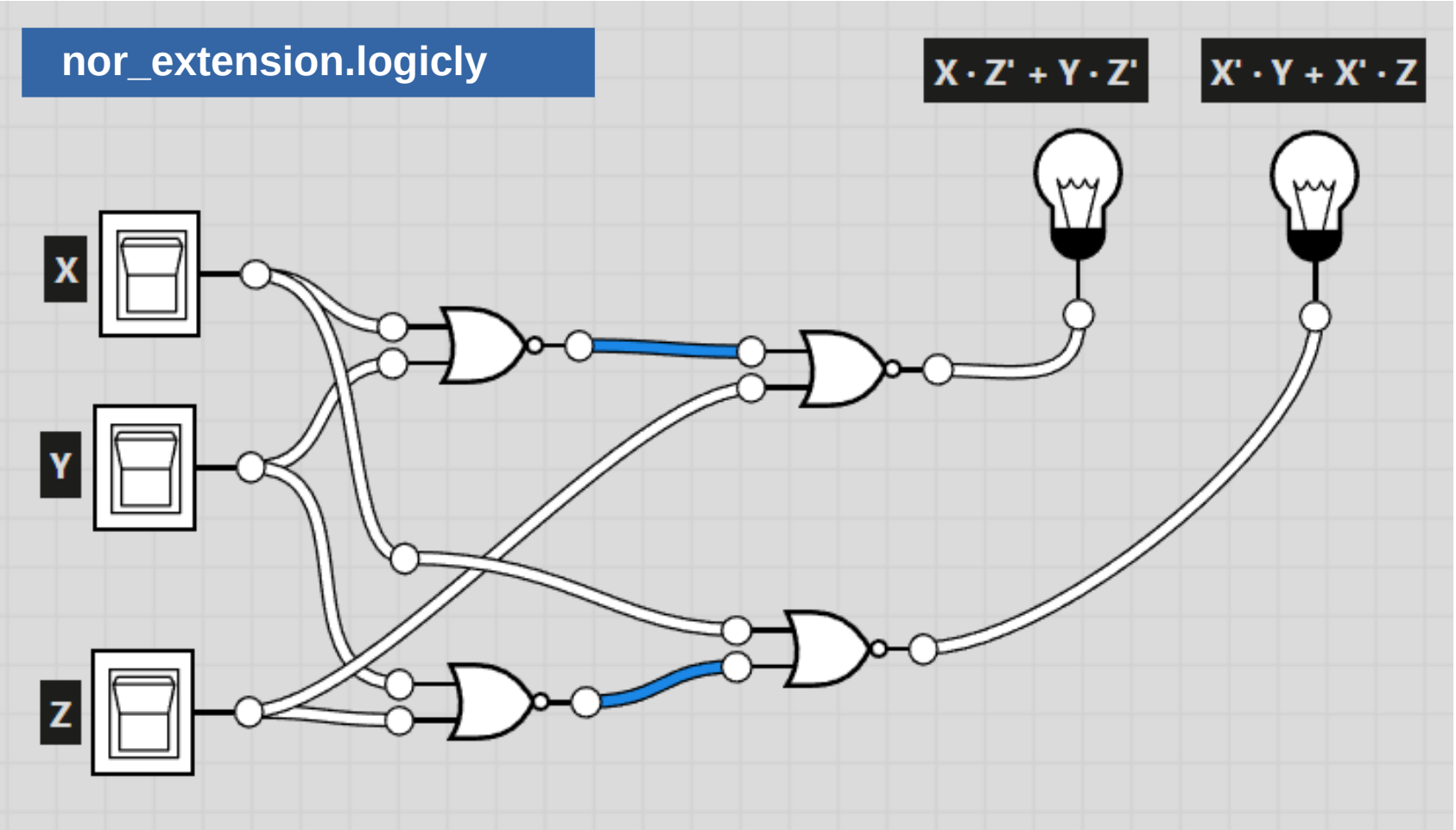
(a) 3-input NOR gate



(b) 3-input NAND gate

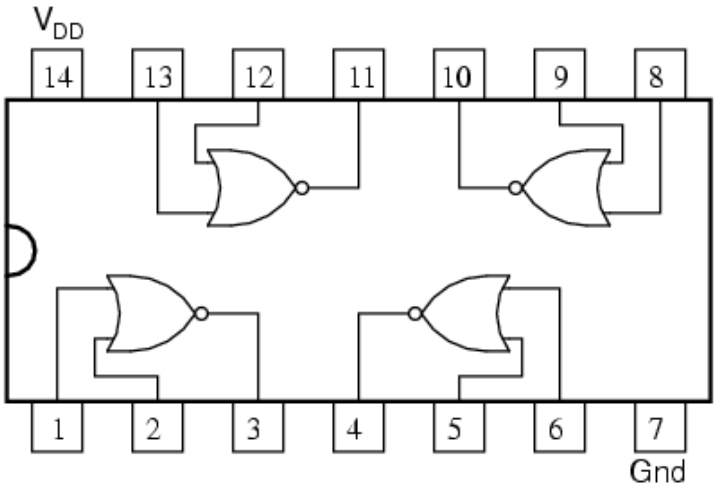
A NOR művelet nem asszociatív

- A **Logic.ly** szimulátor segítségével meggyőződhetünk róla, hogy az **((x NOR y) NOR z)** és az **(x NOR (y NOR z))** kifejezések különböző eredményre vezetnek

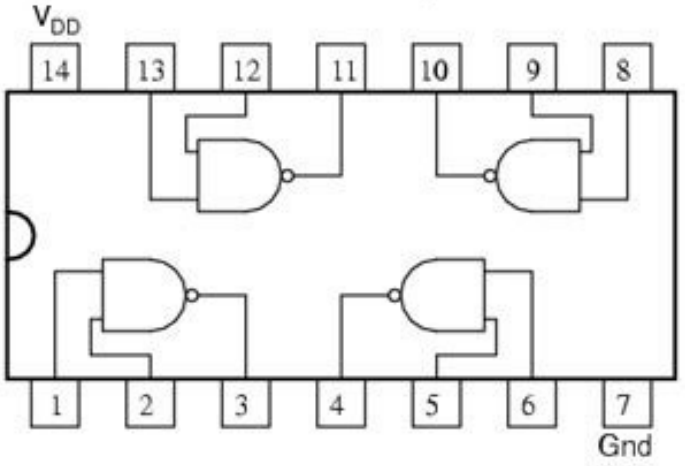


2 és 3 bemenetű NAND és NOR kapuk

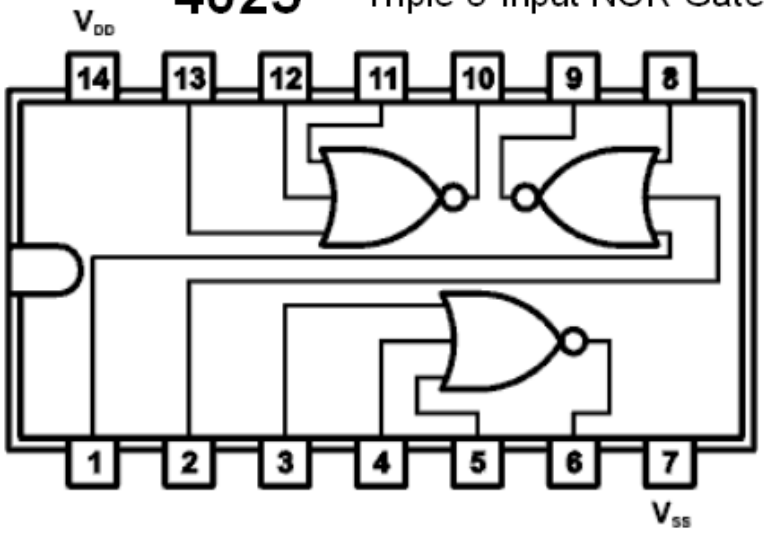
4001 Quad 2-Input NOR Gate



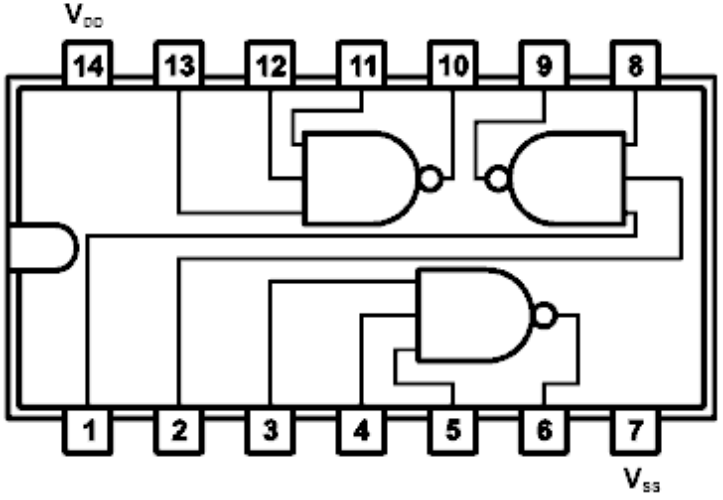
4011 Quad 2-input NAND



4025 Triple 3-Input NOR Gate

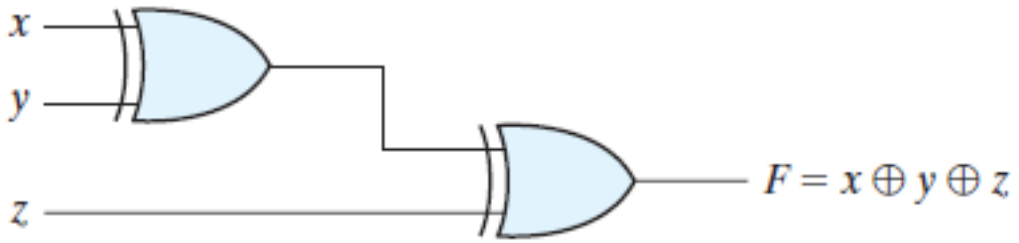


4023 Triple 3-Input NAND Gate

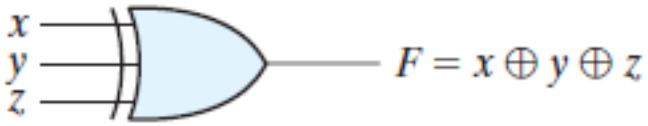


Több-bemenetű XOR kapu

- A **kizáró VAGY** (**XOR**) művelet kommutatív és asszociatív, így a több-bemenetű kiterjesztésének nincs elvi nehézsége. A gyakorlatban azonban nem terjedt el
- Azon ritka esetekben, amikor mégis szükség van rá, inkább több, kétbemenetű kapu összekapcsolásával valósítják meg
- Több változóra kiterjesztve az **XOR** művelet „páratlanszági műveletként” értelmezhető: a kimenet ’1’, ha páratlan számú bemenet van ’1’ állapotban



(a) Using 2-input gates



(b) 3-input gate

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

(c) Truth table

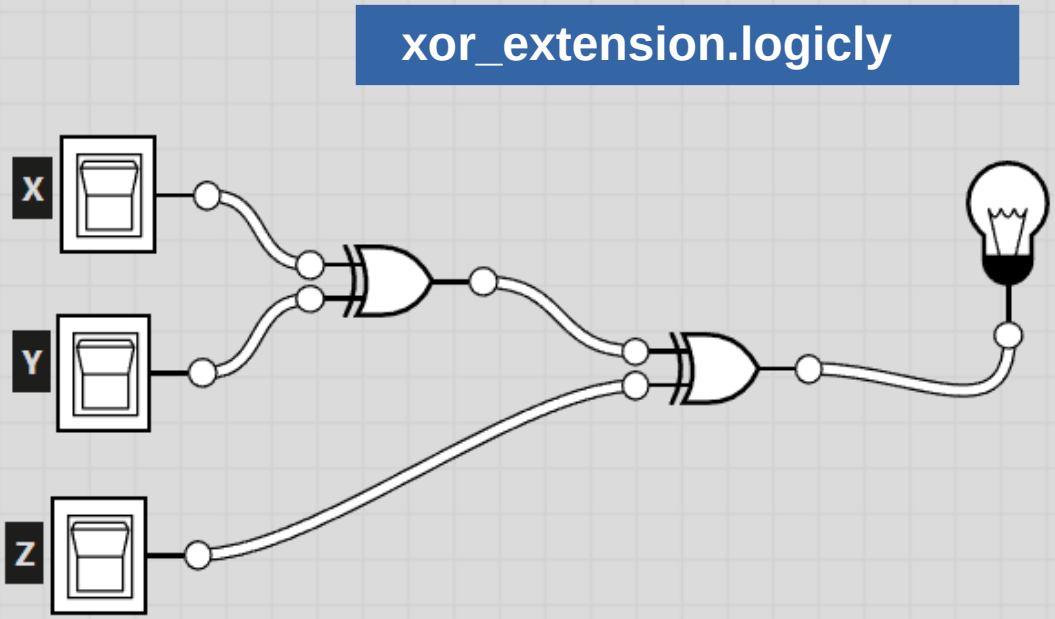
Az XOR művelet kiterjesztése

- Az **XOR** művelet könnyen kiterjeszthető több változóra, s ahogy az igazságtáblázatban látható, „páratlansági műveletként” értelmezhető

Truth Table

I1	I2	I3	O1
false	false	false	false
false	false	true	true
false	true	false	true
false	true	true	false
true	false	false	true
true	false	true	false
true	true	false	false
true	true	true	true

Export as CSV

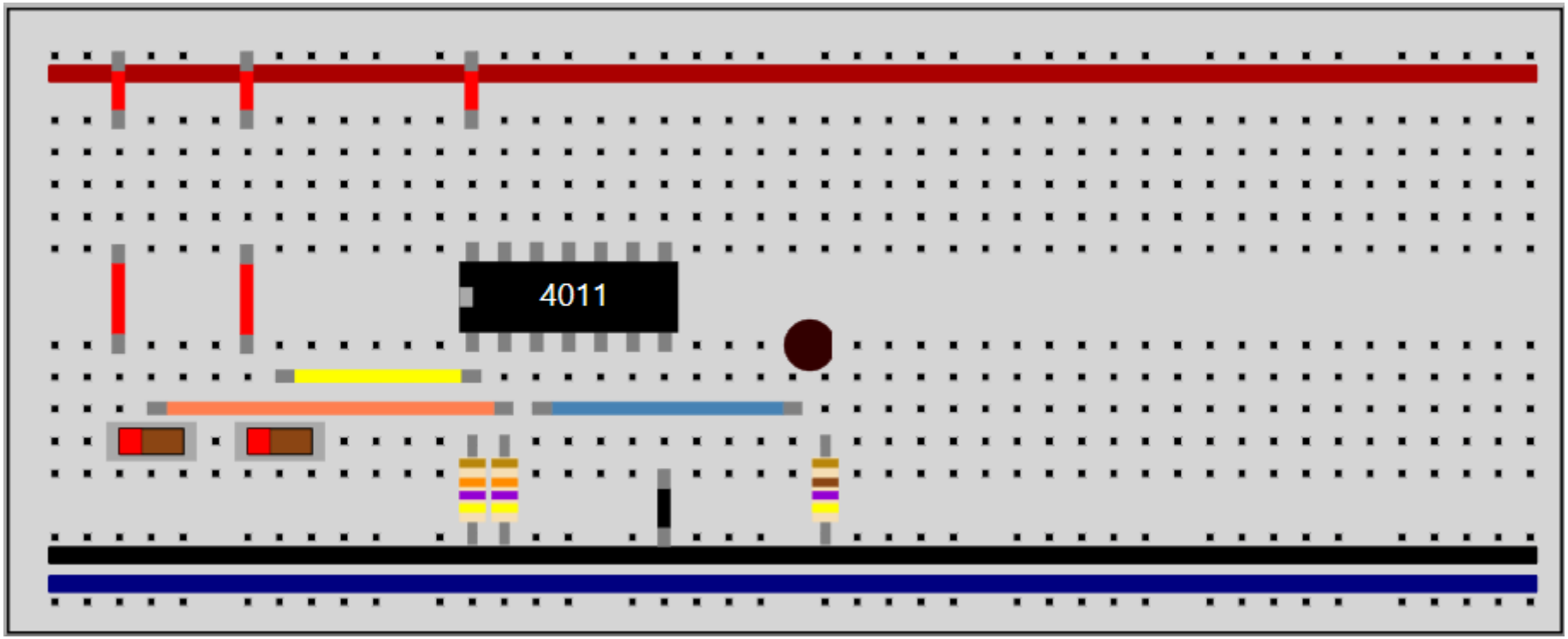


Új játék: Breadboard Simulator

- A https://ds0.me/csim/ címen található ingyenes és nyílforrású program egy áramkör szimulátor, ami Windows alatt futtatható (a .NET futtató keretrendszer 4.5 verzióját igényli)
- 1 – 6 db. dugaszolós próbapanelon az alábbi alkatrészeket szimulálja:
  - Passzív alkatrészek: ellenállások, kondenzátorok
  - Interaktív bemenetek: kapcsolók, potenciométerek és LDR-ek
  - Diszkrét félvezetők: diódák, bipoláris és MOSFET tranzisztorok
  - Analóg IC-k: műveleti erősítők, 555 időzítő
  - Digitális IC-k: CMOS logikai IC-k, billenőkörök, számláló, 7-szegmens dekóder
  - Kijelzők: színes LED-ek, 7-szegmens LED kijelző, oszcilloszkóp tapogató
- A programot nem kell telepíteni, csak kibontjuk és futtatjuk az .exe-t!
- A kijelzőkön kívül az interaktivitást kezelő **csavarhúzó ikont** az IC lábak fölé húzva kiírja az adott láb pillanatnyi feszültségét
- A program számos mintapéldát is tartalmaz, mi most a mai előadás mintaáramköreit próbáljuk ki (**NAND** és **NOR** logika mintapéldák)

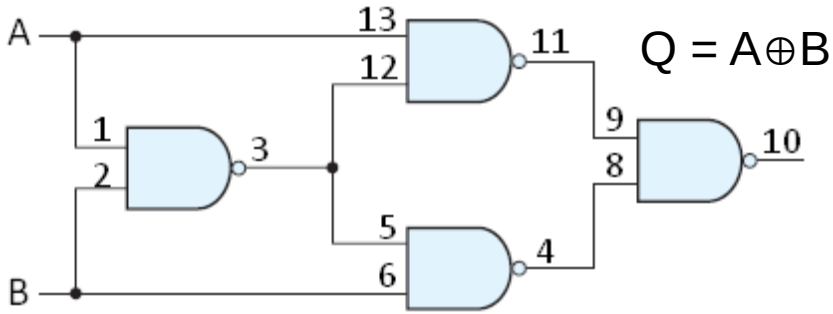
nand_gate.bbrd

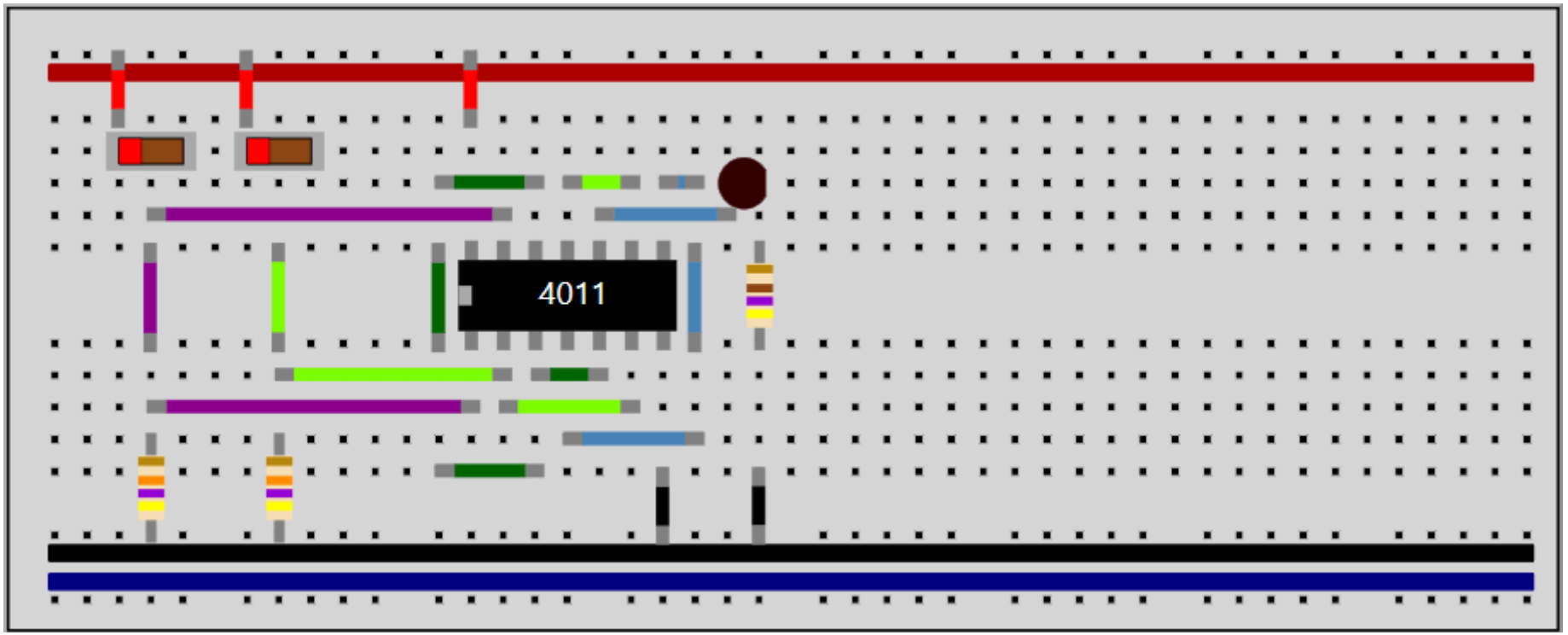
- Az előadásvázlathoz mellékelt segédanyagok között található **nand_gate.bbrd** állományt betöltve a 4 db. kétbemenetű **NAND** kaput tartalmazó 4011 IC egyik kapujának igazságtábláját vizsgálhatjuk.
  A „lejátszás” ikon indítja a szimulációt, a tolókapcsolókat a „csavarhúzó” ikonra kattintás után az egérrel kattintva válthatjuk át másik állásba



xor_from_nand_gates.bbrd

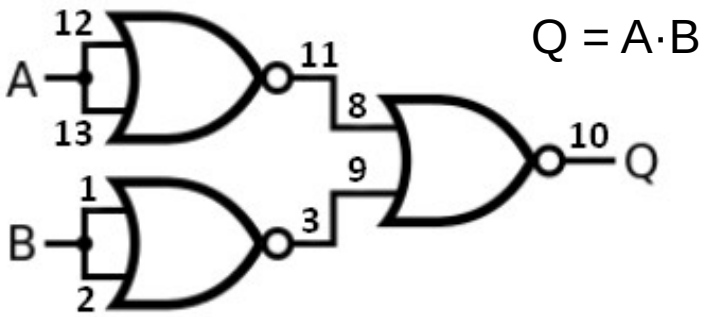
- Az előadásvázlathoz mellékelt segédanyagok között található **xor_from_nand_gates.bbrd** állományt betöltve a 18. oldalon bemutatott kapcsolást szimulálhatjuk
- A **NAND logikát** használó kapcsolás egy **XOR** kaput valósít meg négy **NAND** kapu felhasználásával

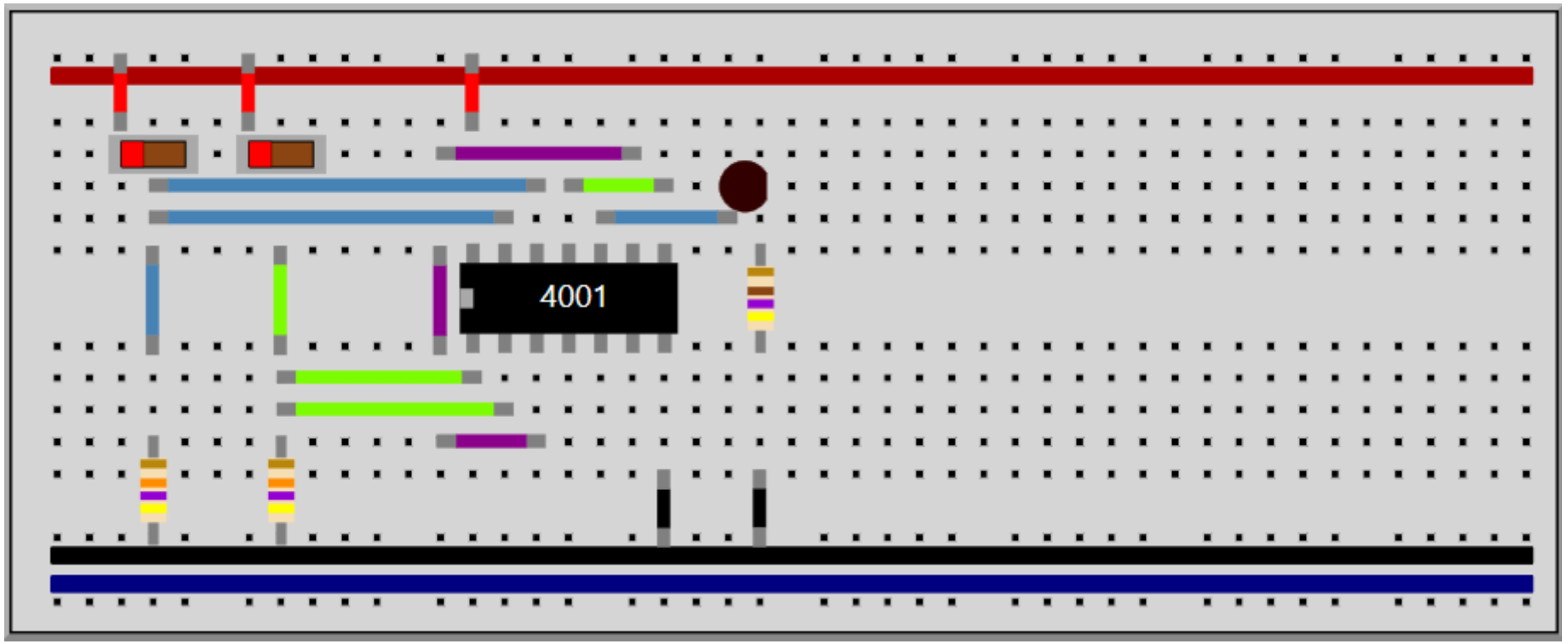




and_from_nor_gates.bbrd

- Az előadásvázlathoz mellékelt segédanyagok között található **and_from_nor_gates.bbrd** állományt betöltve a 22. oldalon bemutatott kapcsolást szimulálhatjuk
- A **NOR logikát** használó kapcsolás egy **AND** kaput valósít meg, három **NOR** kapu felhasználásával





xnor_from_nor_gates.bbrd

- Az előadásvázlathoz mellékelt segédanyagok között található **xnor_from_nor_gates.bbrd** állományt betöltve a 23. oldalon bemutatott kapcsolást szimulálhatjuk (a 8-9 lábak felcserélésével!)
- A **NOR logikát** használó kapcsolás egy **XNOR** kaputvalósít meg négy **NOR** kapu felhasználásával