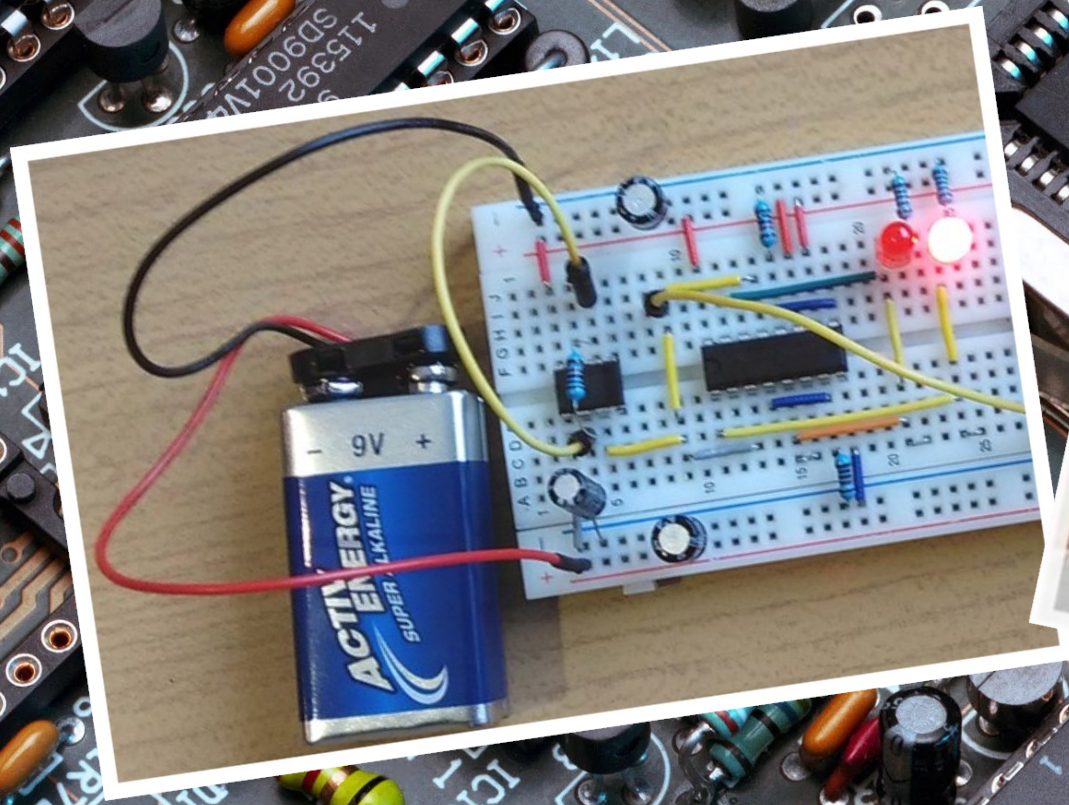
A digitális elektronika alapjai



4. Kombinációs logikai hálózatok – 1. 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)

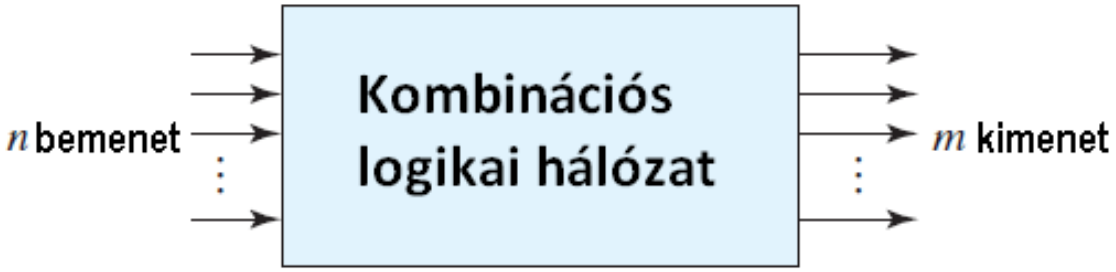
Kombinációs logikai hálózat

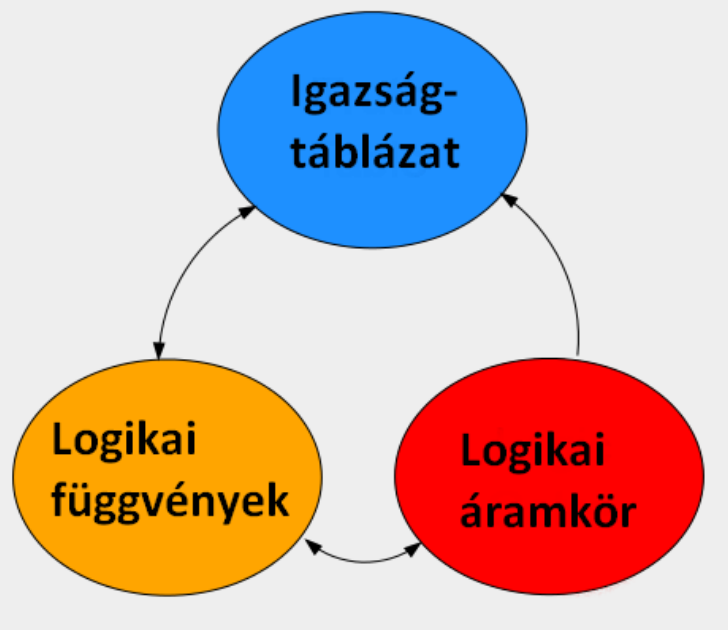
A logikai áramköröket két csoportba sorolhatjuk: **kombinációs** és **sorrendi** hálózatok.

- **A kombinációs logikai hálózat** viselkedése logikai függvényekkel leírható. Kimeneteinek állapota csupán a bemenetek pillanatnyi állapotának függvénye.
- **A sorrendi hálózatok** tárolóelemeket is tartalmaznak. Ezek kimeneteinek állapota emiatt nem csupán a bemenetek, hanem a tárolók állapotától is függ, vagyis a bemenetek korábbi állapotainak sorrendjétől.

A kombinációs logikai hálózat összekapcsolt logikai kapukból áll. Az n bemenetnek összesen 2^n állapota lehetséges, amelyek mindegyikéhez a kimenetek egy-egy állapota rendelhető. Így a kombinációs logikai hálózat egyértelműen megadható az **igazságtáblázatával,** amely leírja a „viselkedését”.

A kombinációs logikai hálózat kimenetei egy-egy n változós **logikai függvénnyel** is megadhatók.





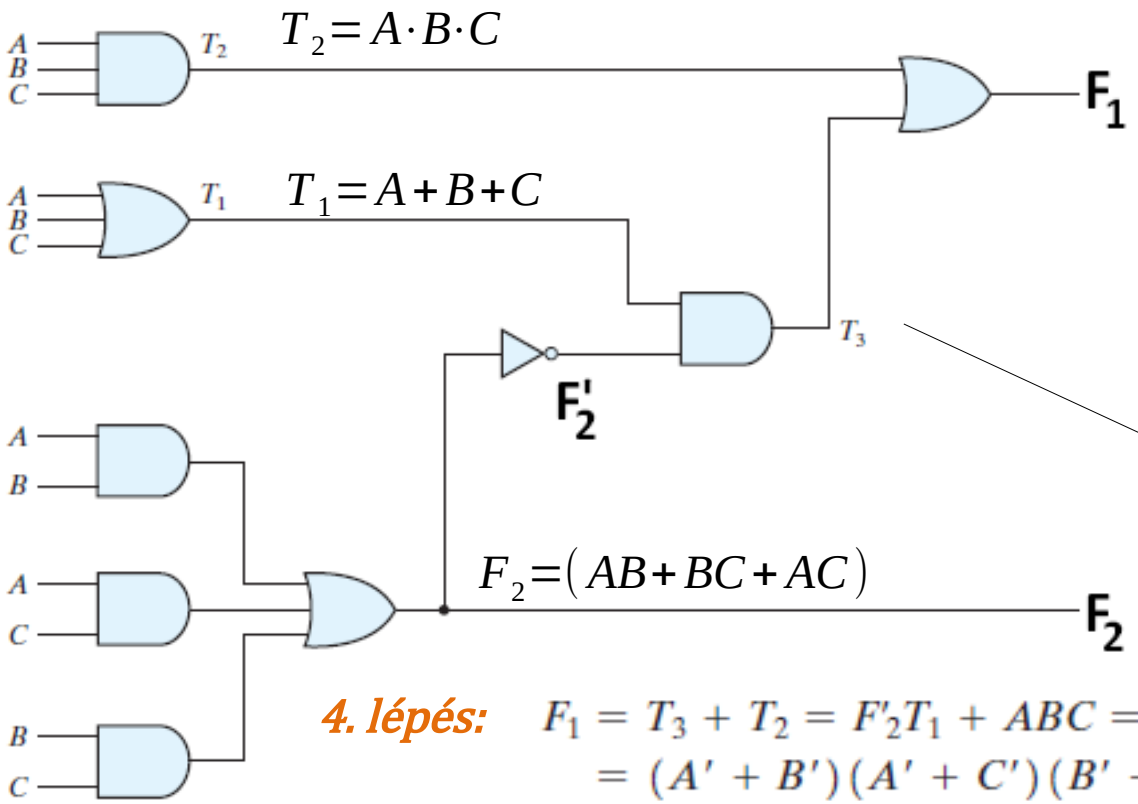
Logikai elemzés

- A kombinációs logikai áramkörök elemzésének az a célja, hogy meghatározzuk azokat a logikai összefüggéseket (függvényeket), amelyeket az adott áramkör megvalósít. Az elemzésnél **egy áramkörből indulunk ki,** s a végeredmény néhány függvény, egy igazságtáblázat, vagy az áramkör működésének leírása
- Ellenőriznünk kell. hogy az áramkör valóban **kombinációs hálózat,** nem pedig **sorrendi hálózat.** A kombinációs logikai hálózat nem tartalmaz visszacsatolást, sem tárolóelemeket.
- **A logikai áramkör rajzából a logikai függvényeket az alábbi lépésekben kaphatjuk meg:**

1) Címkézzünk fel minden kapu kimenetet, ami a bemeneti változók függvénye! Határozzuk meg a kapuk által megvalósított logikai függvényeket!
2) Címkézzünk fel azon kapuk kimenetét, amelyek a bemeneti változók és az előző pontban felcímkézett kapu kimenetek függvénye! Határozzuk meg az ezen kapuk által megvalósított logikai függvényeket is!
3) Ismételjük meg a 2. lépést mindaddig, amíg a kimenetek leírása elő nem áll!
4) Az előzőekben meghatározott függvényekbe ismételt behelyettesítésekkel állítsuk elő a kimeneteket a bemeneti változók logikai függvényeként!

Példa az elemzésre

Az előző oldalon leírt lépéseket az alábbi példával szemléltetjük: Az áramkör három bemenettel (A, B, C) és két kimenettel rendelkezik (F1 és F2).



1. lépés:

$$T_1 = A + B + C$$

$$T_2 = A \cdot B \cdot C$$

2. lépés:

$$T_3 = (A + B + C) \cdot \overline{F_2}$$

3. lépés:

$$F_2 = (AB + BC + AC)$$

4. lépés:

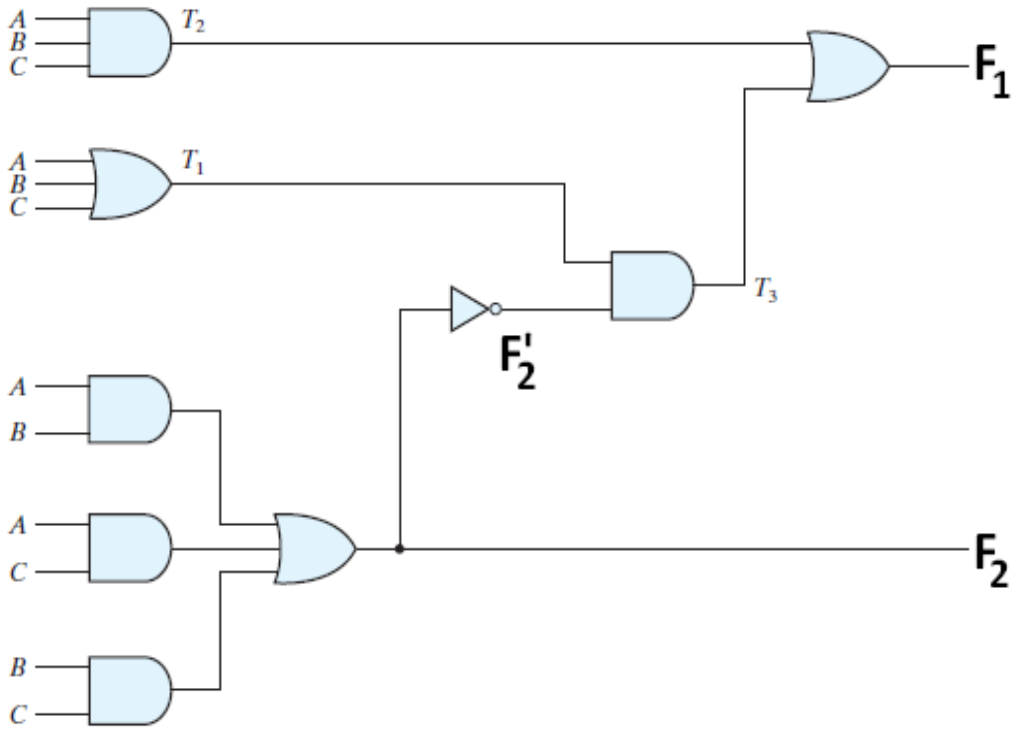
$$\begin{aligned} F_1 &= T_3 + T_2 = F_2' T_1 + ABC = (AB + AC + BC)'(A + B + C) + ABC \\ &= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC \\ &= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC \\ &= A'BC' + A'B'C + AB'C' + ABC = A \oplus B \oplus C \end{aligned}$$

A logikai függvények meghatározása önmagában nem elegendő az áramkör működésének megértéséhez. Később majd látni fogjuk, hogy ez az áramkör a teljes összeadót valósítja meg.

Az igazságtáblázat felírása

Az igazságtáblázatot az előző oldalon meghatározott logikai függvények segítségével is felírhatjuk, de magából az áramkörből is kiindulhatunk:

1. Állítsuk elő az *n* bemenet összes lehetséges (2^n darab) kombinációját!
2. Minden bemeneti kombinációhoz határozzuk meg azon kapuk kimeneti állapotát, amelyek csak a bemeneti változóktól függenek!
3. Minden bemeneti kombinációhoz határozzuk meg azon kapuk kimeneti állapotát is, amelyek a bemeneti változók mellett az előzőleg meghatározott kapu kimenetektől is függenek!

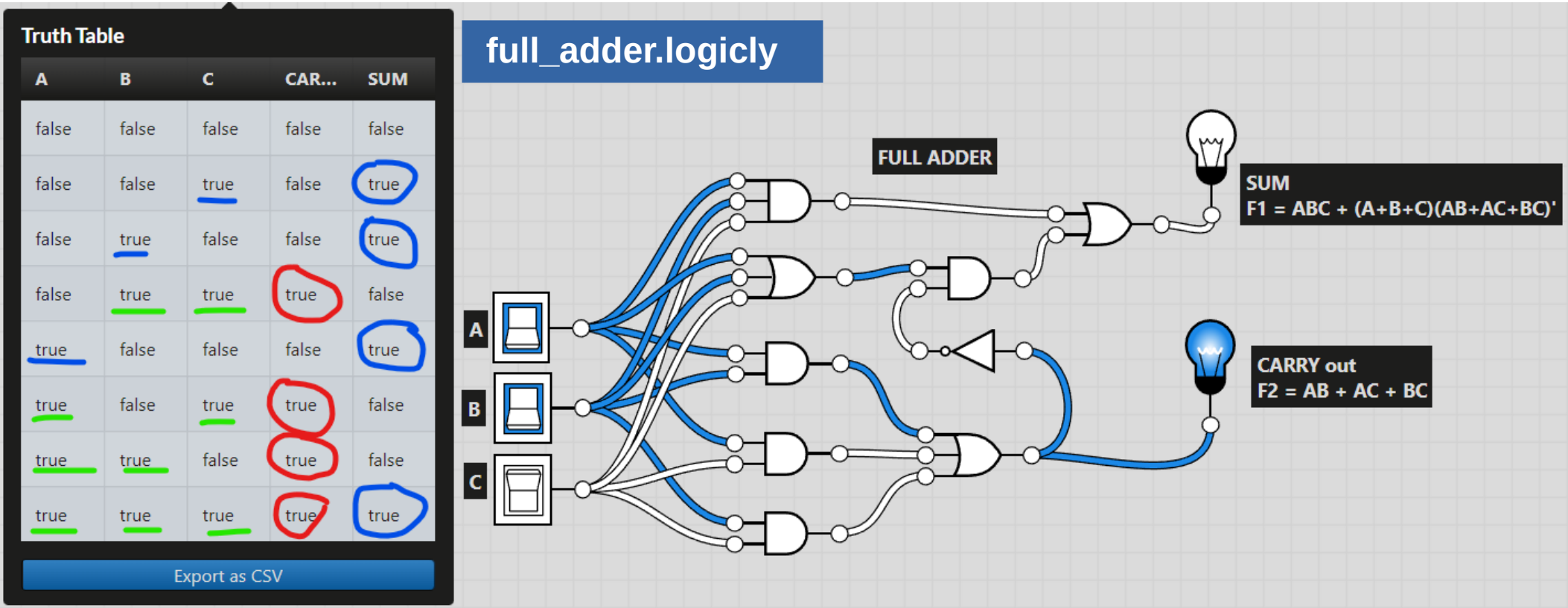


A	B	C	F_2	F'_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

Modellezzük az áramkört

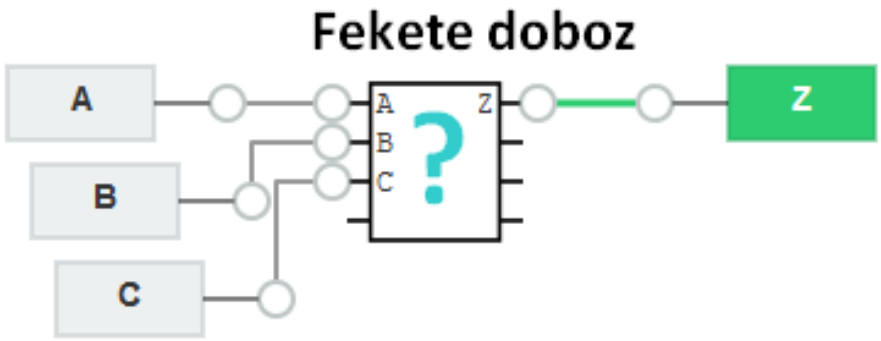
- A kapcsolást kipróbáltuk a **Logic.ly** programban is, és az az előző oldalon kiszámolt igazságtáblázatot adja vissza
- A kapcsolás hátránya, hogy ötféle kaput tartalmaz, így megvalósítása gazdaságtalan

A	B	C	F_2	F_1
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

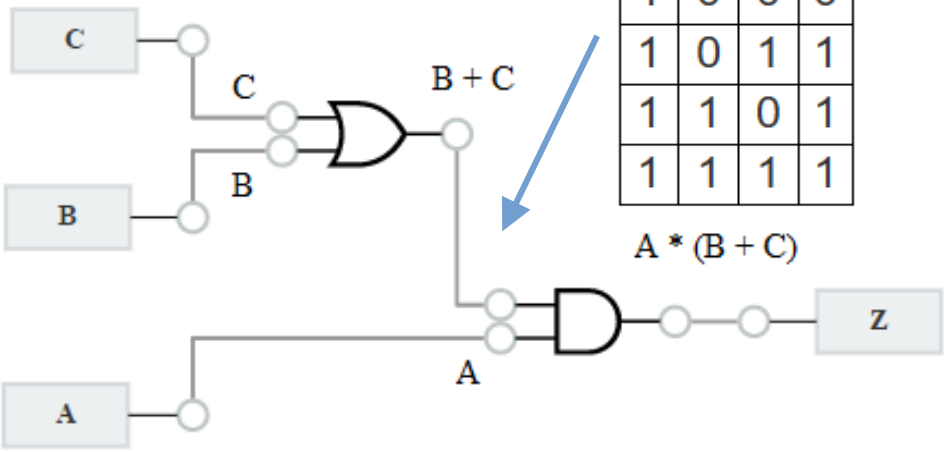


A tervezés folyamata

- **A tervezés folyamata** az előzőeknek az ellentettje:
  - A kiindulási alap általában **a feladat specifikációja**
  - A cél egy áramkör megtervezése, vagy a logikai függvények előállítása



A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

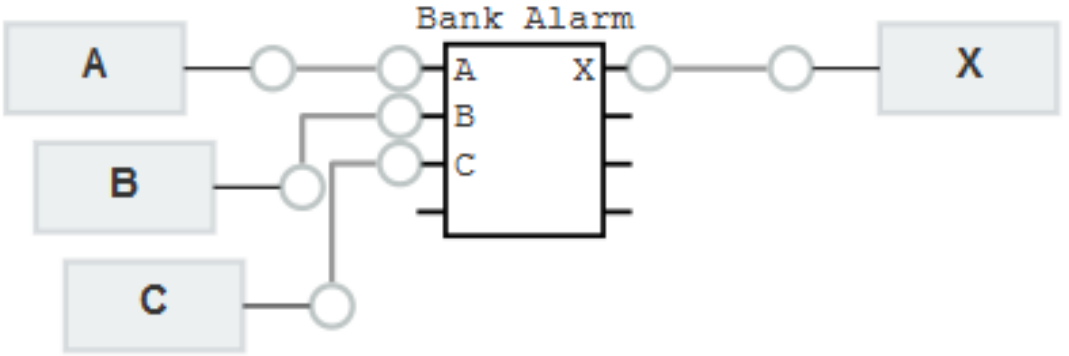


Forrás: learn.electronics-course.com/#6

- **A tervezés folyamatának lépései:**
  - A feladat specifikációjából határozzuk meg a szükséges bemenetek és kimenet számát, s nevezzük el azokat!
  - A bemeneti és a kimeneti állapotok viszonyából állítsuk elő az igazságtáblázatot!
  - Az igazságtáblázat alapján állítsuk elő a kimeneteket leíró logikai függvényeket és egyszerűsítsük azokat!
  - Rajzoljuk meg a logikai áramkört és ellenőrizzük a helyességét (kézzel vagy szimulációs programmal)!

Tervezési példa: banki riasztó

Tegyük fel, hogy egy bank riasztórendszert akar telepíteni, amely három mozgásérzékelőn (A, B, C) alapul. A hamis riasztások kiszűrésére (pl. egy pók éppen az egyik érzékelő előtt ereszkedik le…) a riasztás csak akkor induljon, ha a három érzékelő közül legalább kettő egyidejűleg jelez.
(Forrás: learn.electronics-course.com/#0)



C	B	A	X	
0	0	0	?	0
0	0	1	?	0
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	1	1
1	1	0	?	
1	1	1	?	1

Logikai függvény:

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

Hmm, kicsit komplikált!
Egyszerűbben, ha kérhetném!

Az igazságtáblázatból hogyan lett függvény?

- Az igazságtáblázatból többféle módon lehet logikai függvényeket származtatni, most csak a legegyszerűbb módszert mutatjuk be:
- Az igazságtáblázatban minden olyan sorhoz, amelyben a kimenet értéke 1, tartozik egy szorzat, amelyben az összes bemeneti változó szerepel. Ezeket „összeadjuk”, azaz **VAGY** kapcsolatba hozzuk, így áll elő a függvény, amelyet a továbbiakban még egyszerűsíthetünk

A	B	C	X	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}\cdot B\cdot C$
1	0	0	0	
1	0	1	1	$A\cdot\overline{B}\cdot C$
1	1	0	1	$A\cdot B\cdot\overline{C}$
1	1	1	1	$A\cdot B\cdot C$

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

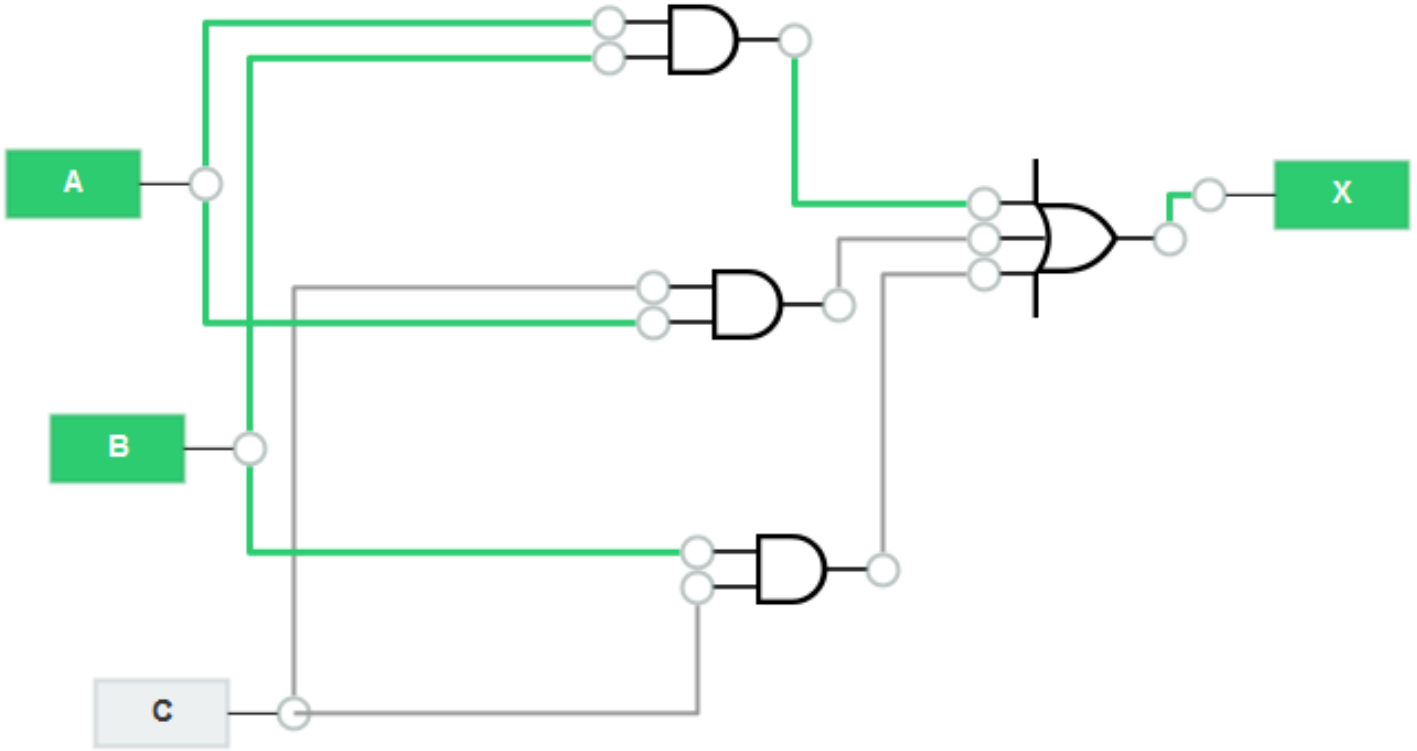
Egyszerűsítés Karnaugh-tábla alapján

A **Karnaugh-tábla** az igazságtáblázat egy másik ábrázolási formája. Az adatok úgy vannak csoportosítva, hogy az 1-bites változások szomszédos cellákba essenek, így az **F = ABC' + ABC = AB** típusú egyszerűsítések könnyen felismerhetők benne.

		C B			
		00	01	11	10
A	0	0	0	1	0
	1	0	1	1	1

$$X = \overline{C}\bullet B\bullet A + C\bullet\overline{B}\bullet A + C\bullet B\bullet\overline{A} + C\bullet B\bullet A \quad \Rightarrow \quad X = B\bullet A + C\bullet A + C\bullet B$$

C	B	A	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



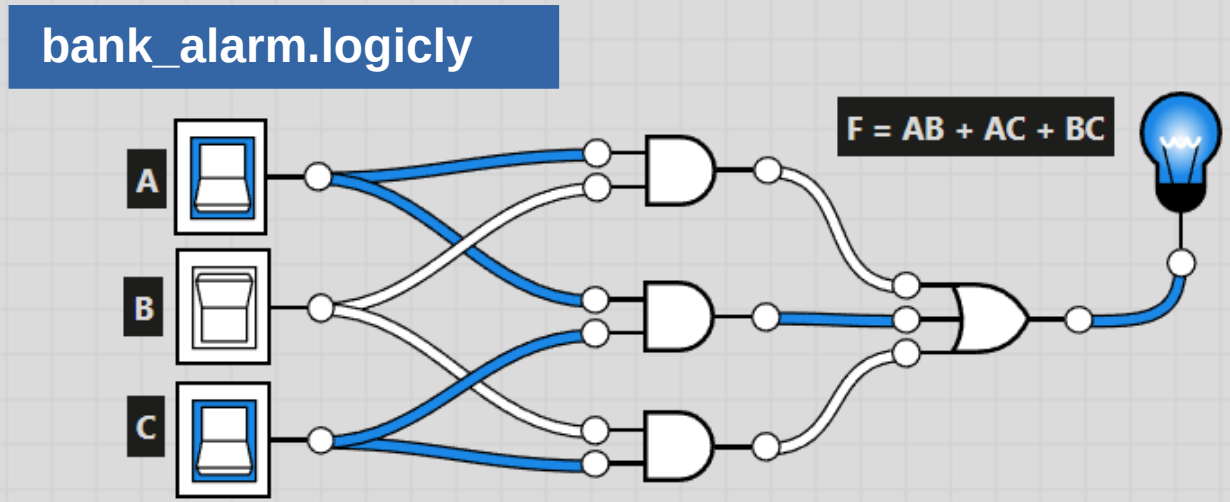
Modellezzük a kapcsolást!

- A „banki riasztórendszert”, ami egy egyszerű többségi szavazó-áramkör, kipróbáltuk a **Logic.ly** programban, ellenőriztük a működését és az igazságtáblázatát

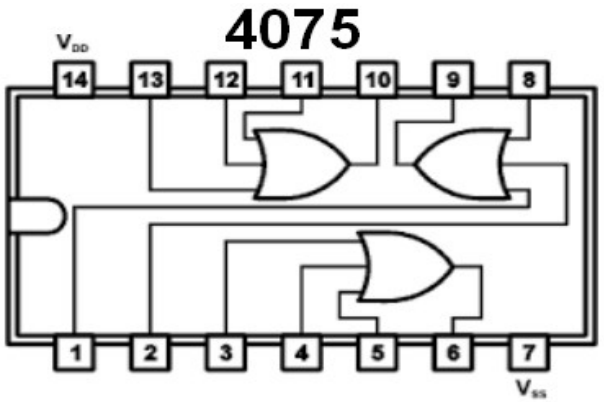
Truth Table

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

Export as CSV

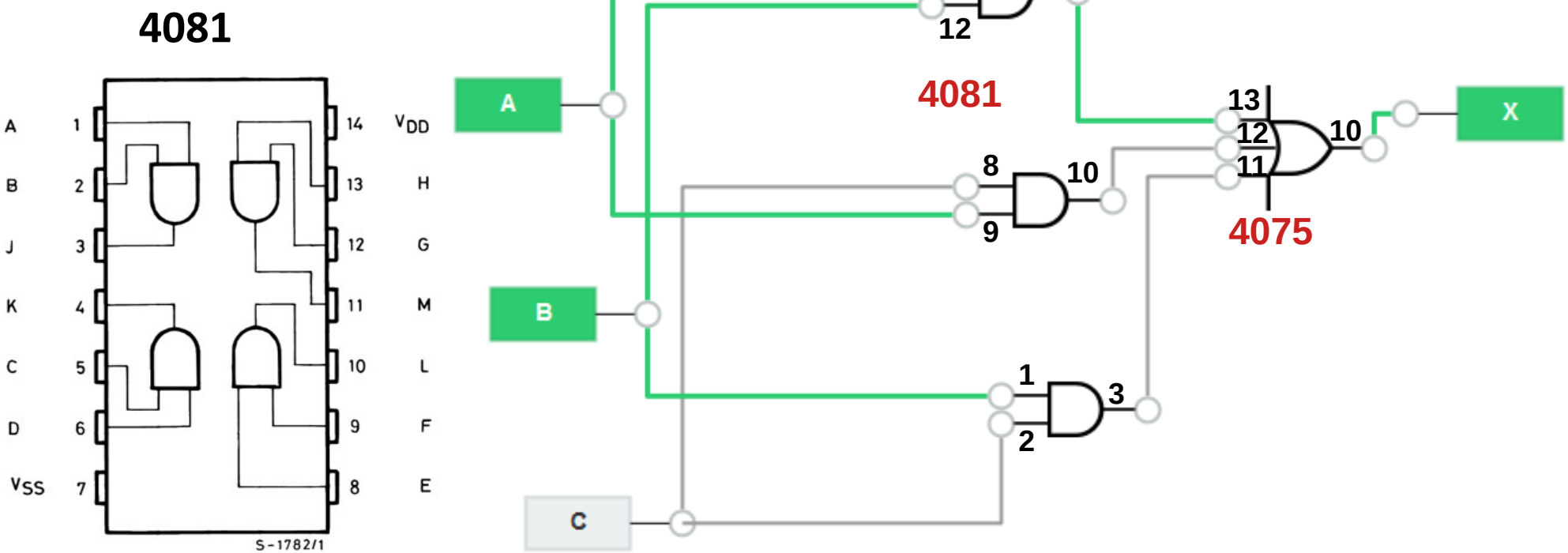


Építsük meg a kapcsolást!

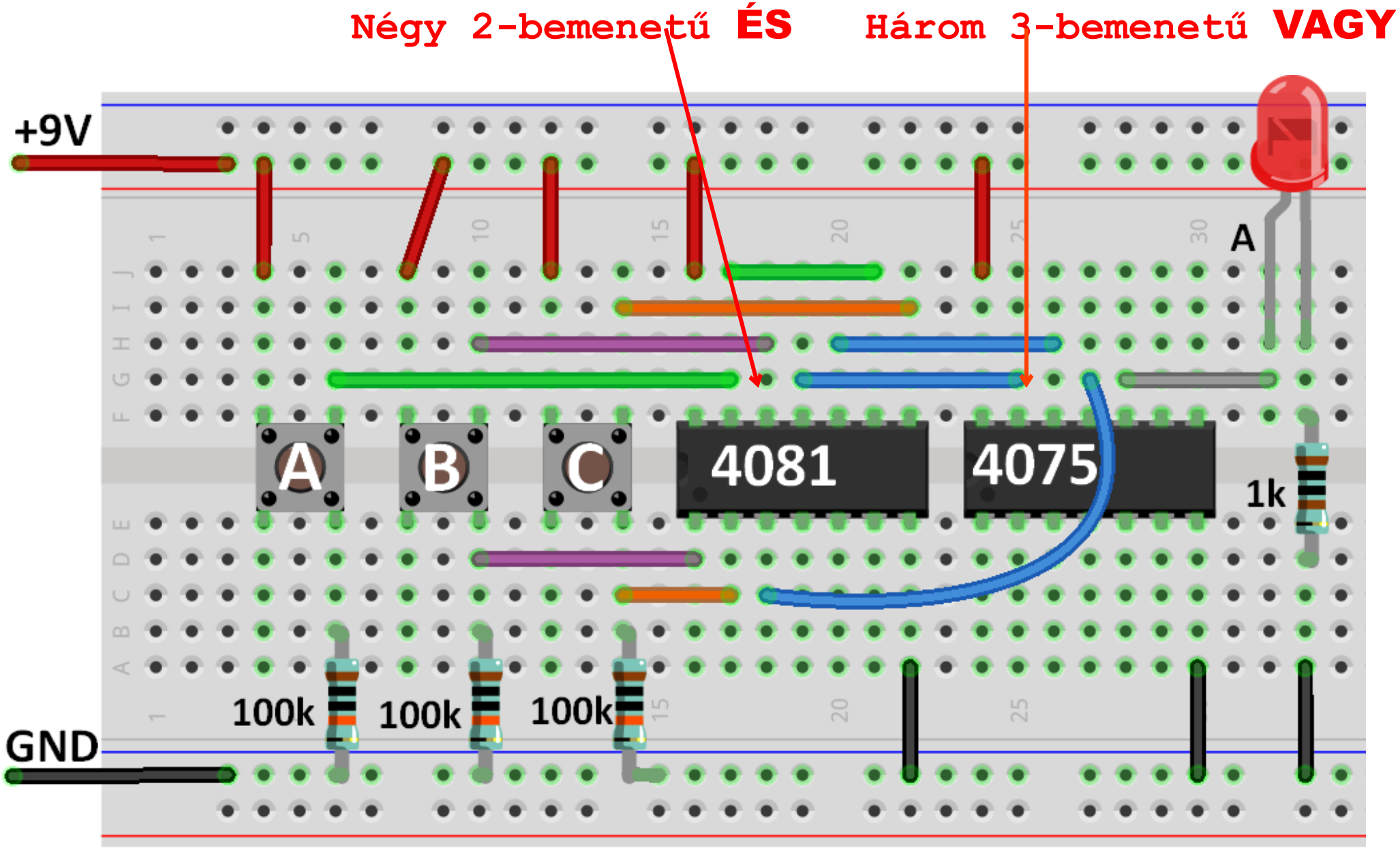


Az ábrán bejelöltük a javasolt lábkiosztást

$$X = \overline{C} \cdot B \cdot A + C \cdot \overline{B} \cdot A + C \cdot B \cdot \overline{A} + C \cdot B \cdot A \quad \Rightarrow \quad X = B \cdot A + C \cdot A + C \cdot B$$



A „banki riasztó” megépítése



Bináris összeadás

A bináris összeadás szabályai

0	1	0	1
+ 0	+ 0	+ 1	+ 1
0 0	0 1	0 1	10

x

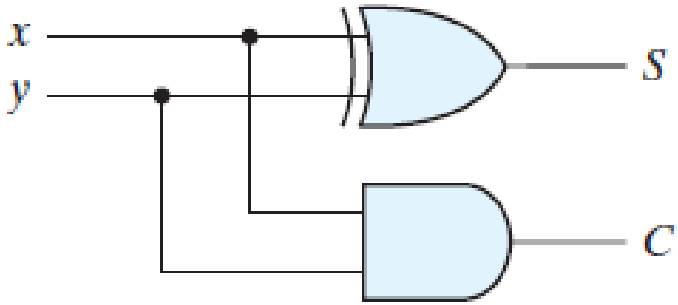
y

C átvitel

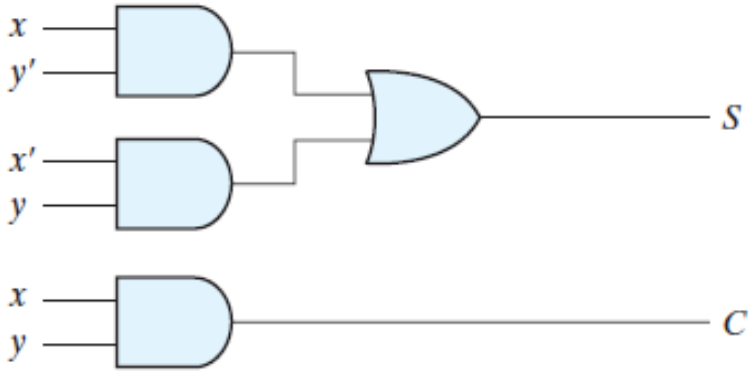
S összeg

10-féle ember van:
- az egyik érti a kettes számrendszert,
- a másik nem...

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



(b) $S = x \oplus y$
$C = xy$



(a) $S = xy' + x'y$
$C = xy$

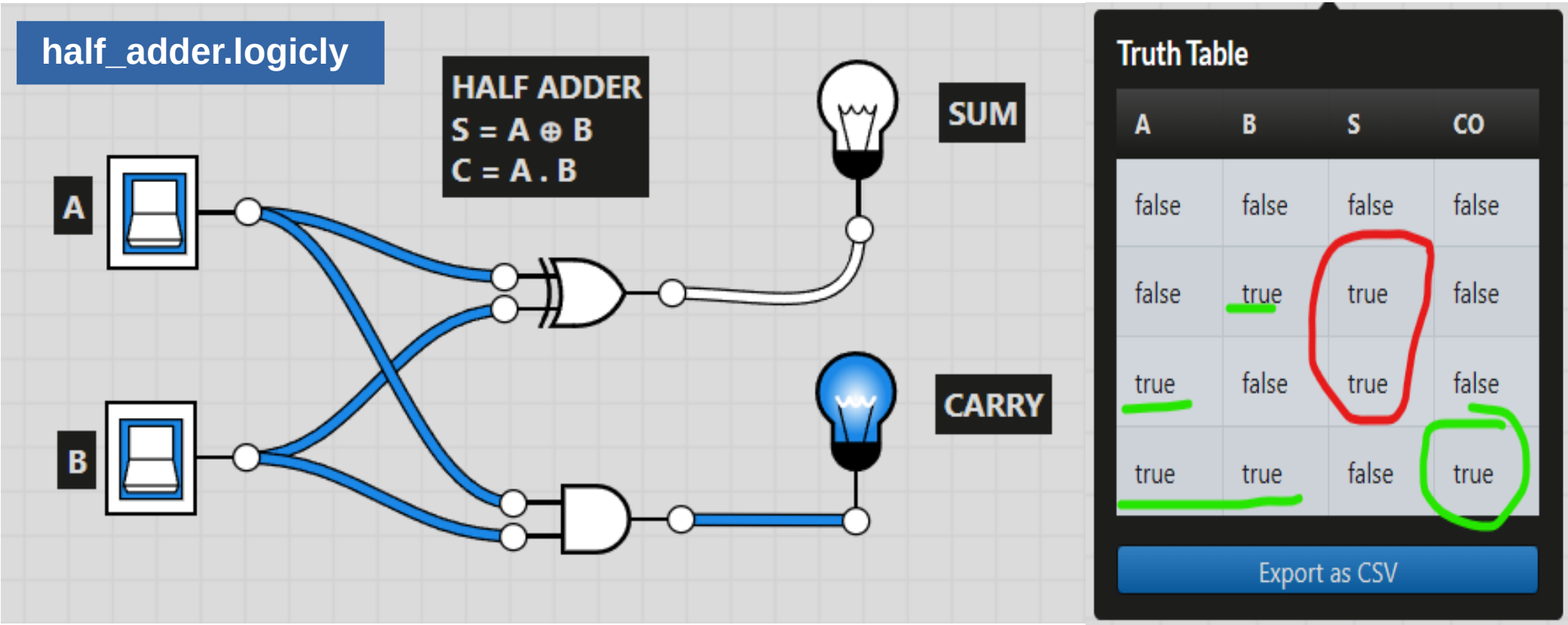
Az **XOR** kapu alkalmazása egyszerűbb megépítést tesz lehetővé.

Ezt a kapcsolást **fél-összeadónak** (**HA**) hívják.
A teljes összeadáshoz az áthozatot is kezelni kell...

A fél-összeadó modellezése

- Az egyszerűsített (**XOR** és **AND** kapuval megvalósított) kapcsolást kipróbáltuk a **Logic.ly** programban
- Az elvárt igazságtáblázatot adja vissza

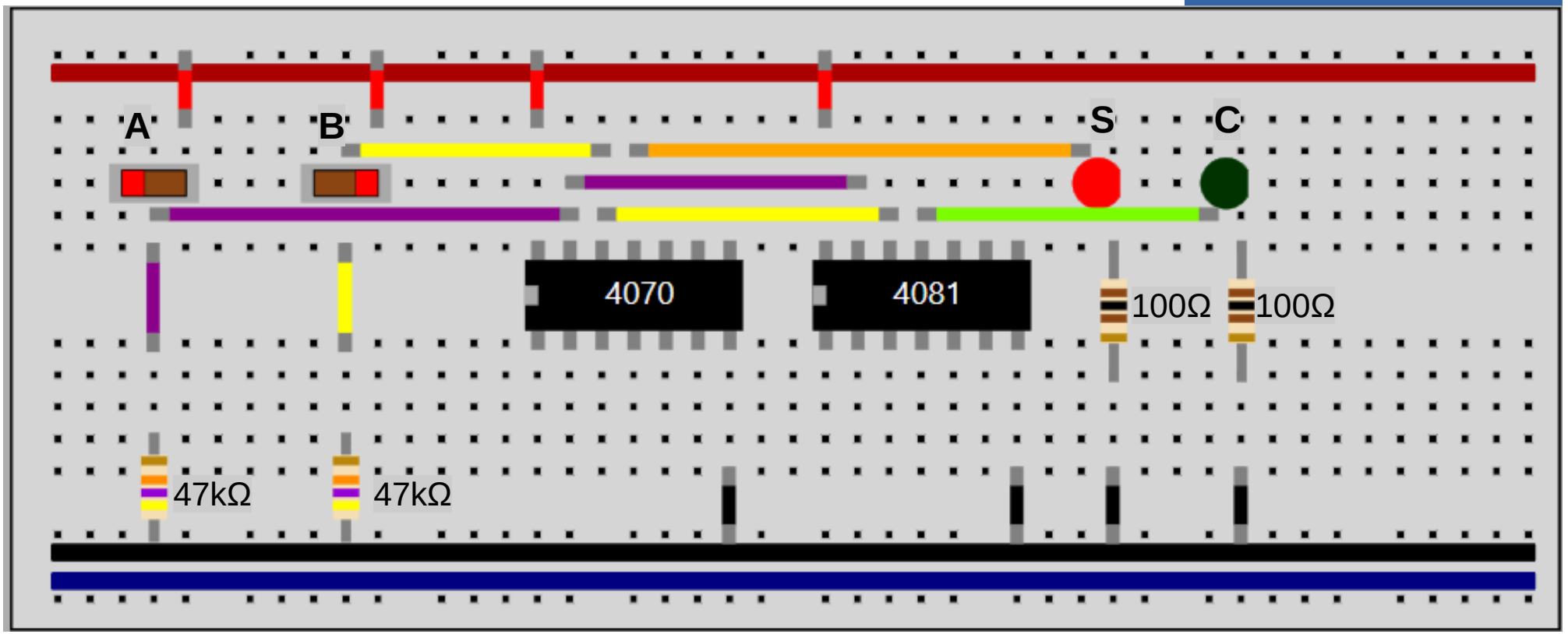
x	*y*	*C*	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



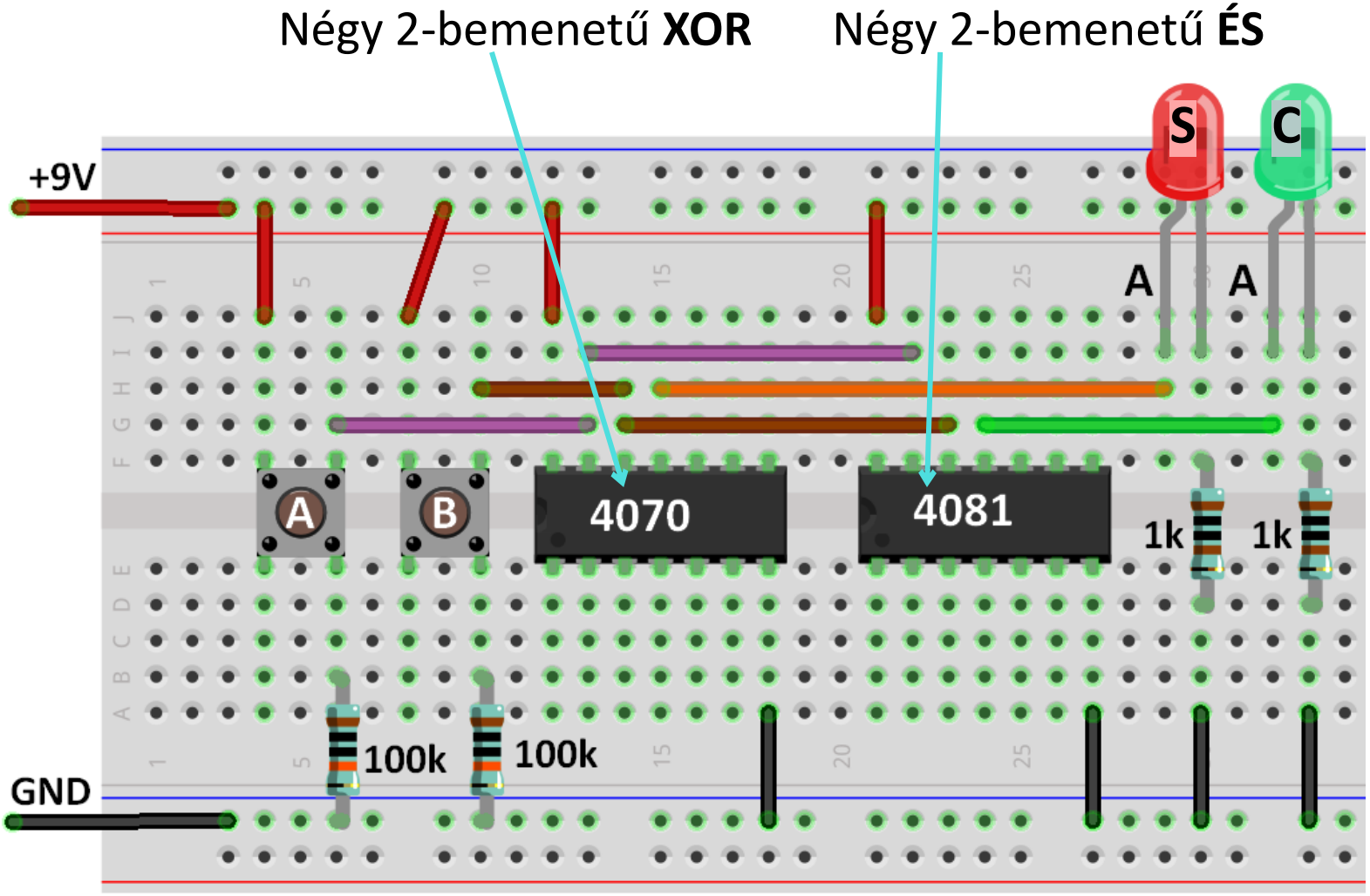
A fél-összeadó modellezése

- Modellezzük az áramkört a **Breadboard Simulator**ban! A bemeneteket itt két tolókapcsolóval vezéreljük (jobbra tolva lesz a bemenet magas szinten, azaz „1” állapotban)
- Az **S** (összeg) kimenet állapotát a piros LED a **C** (átvitel) kimenet állapotát pedig a zöld LED jelzi ki

half_adder.bbrd



A fél-összeadó megépítése



A teljes összeadó

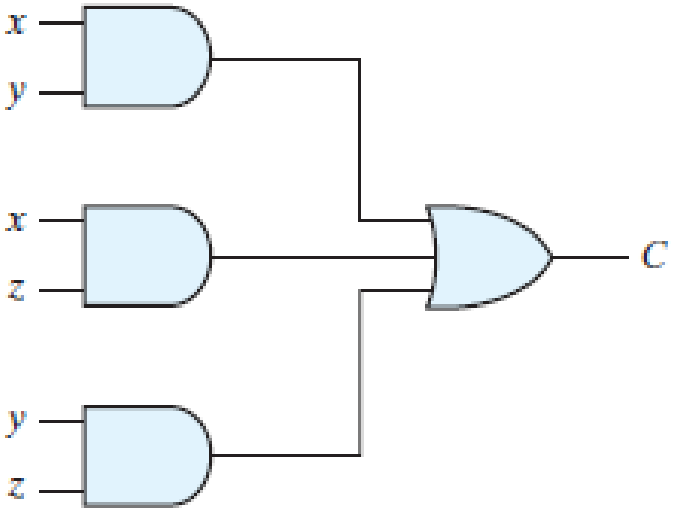
- **A teljes összeadó** a két összeadandón (x, y) kívül az **áthozatot** is fogadja. Az igazságtáblázatot az összeadás szabályai szerint könnyen felírhatjuk. Az ebből kapott függvények:

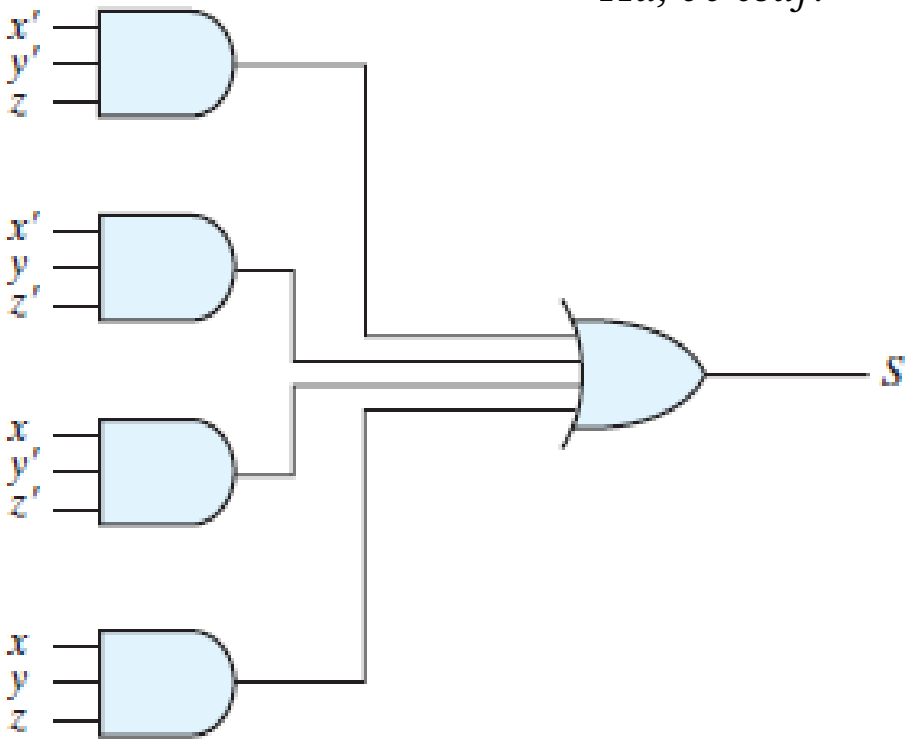
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$C = xy + xz + yz$$

$$S = x'y'z + x'yz' + xy'z' + xyz$$

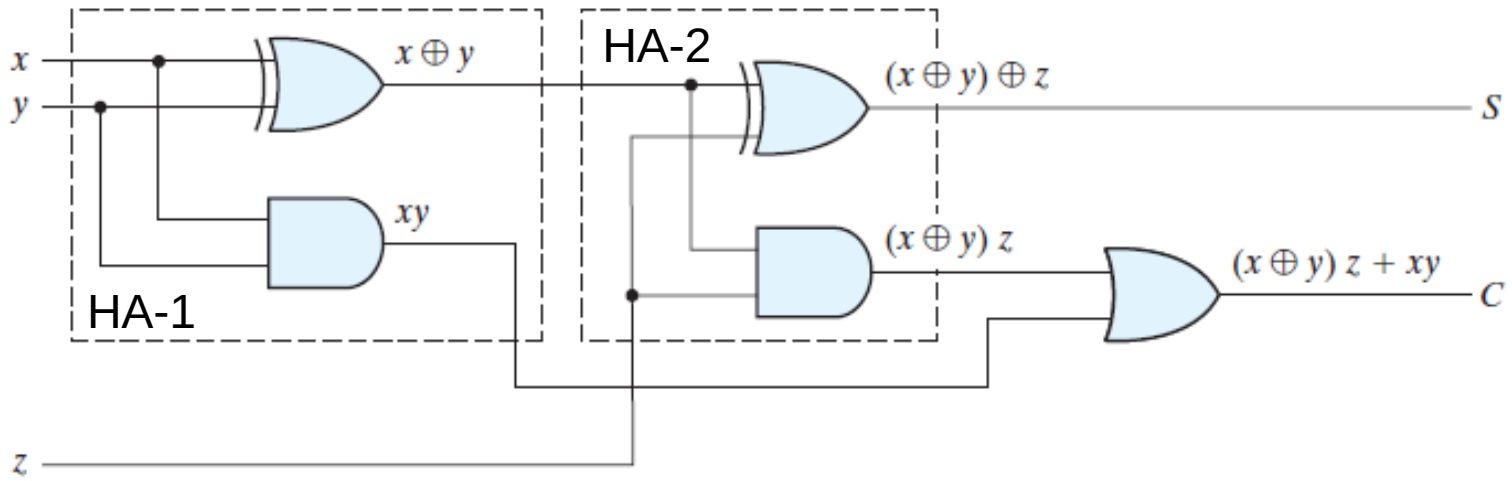
Hű, be csúf!





A teljes összeadó - másképp

- Két **fél-összeadó** összekapcsolásával és egy **VAGY** kapuval is megépíthetjük a teljes összeadót



- Ellenőrizzük a kimeneteket előállító logikai függvényeket!

$$
\begin{aligned}
S &= z \oplus (x \oplus y) \\
&= z'(xy' + x'y) + z(xy' + x'y)' \\
&= z'(xy' + x'y) + z(xy + x'y') \\
&= xy'z' + x'yz' + xyz + x'y'z
\end{aligned}
$$

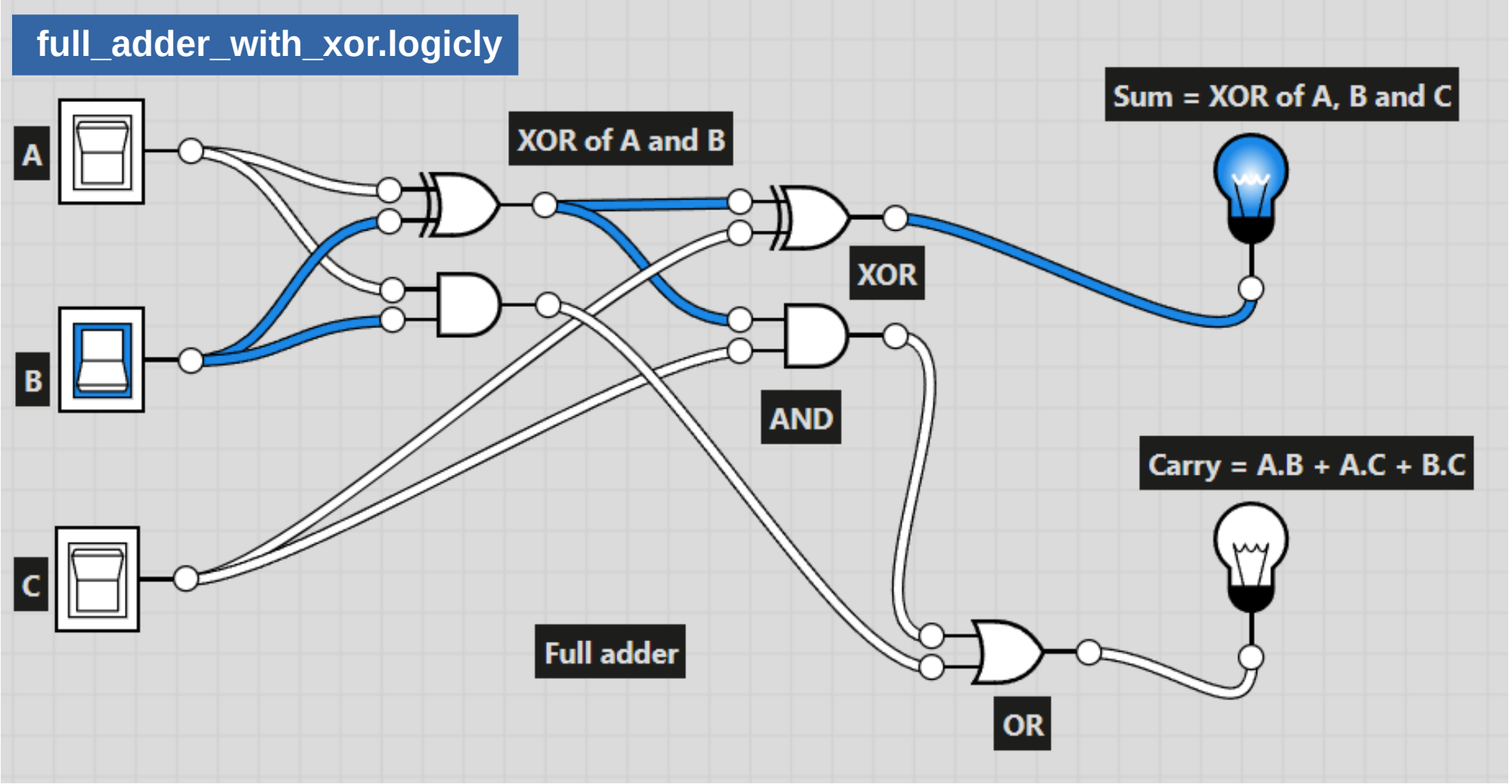
Hasonlítsuk össze ezeket az 5. oldalon kapott függvényekkel!

$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$ = xy + xz + yz

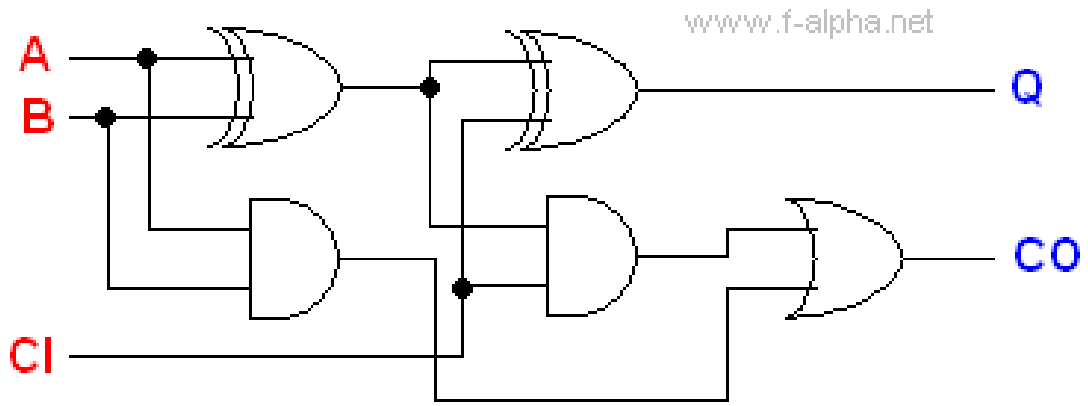
Lásd pl. Boolean Algebraic Solver

A teljes összeadó modellezése

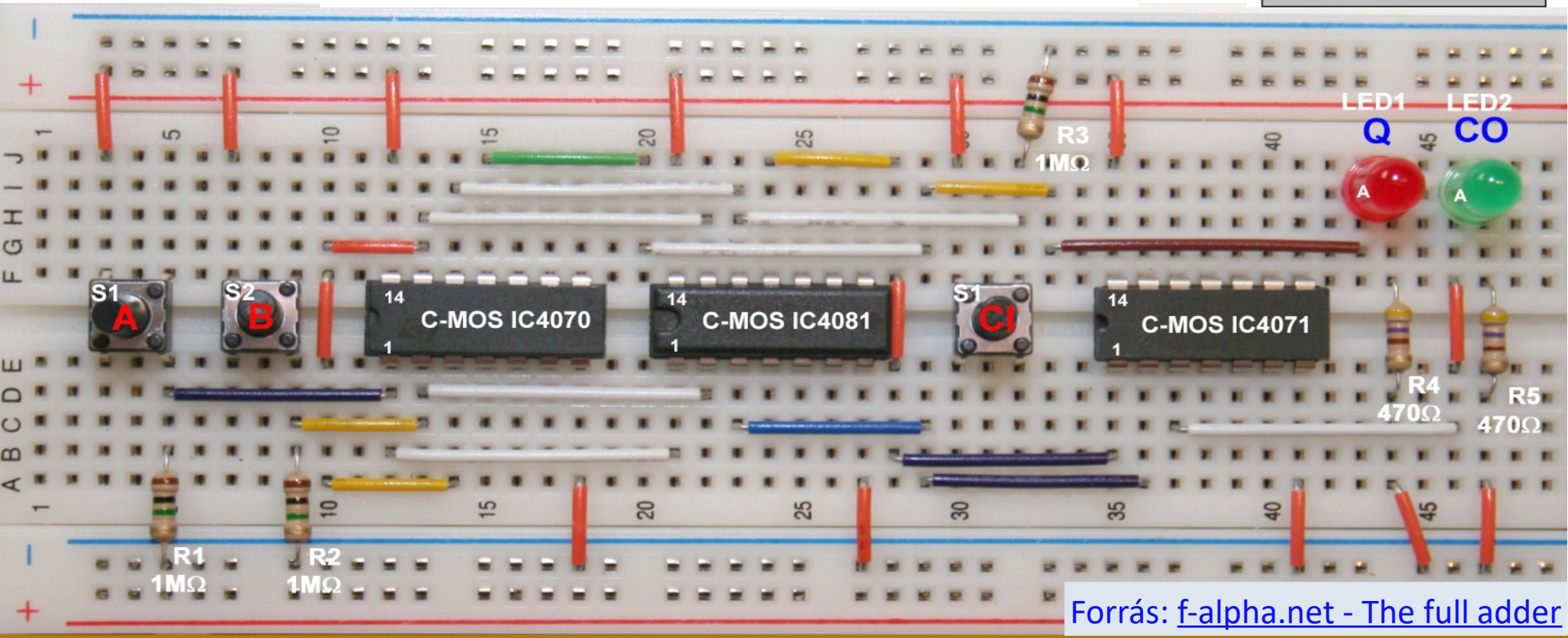
- A teljes összeadó kapcsolást kipróbáltuk a **Logic.ly** programban, s az elvárásoknak megdelelően működött



A teljes összeadó megépítése

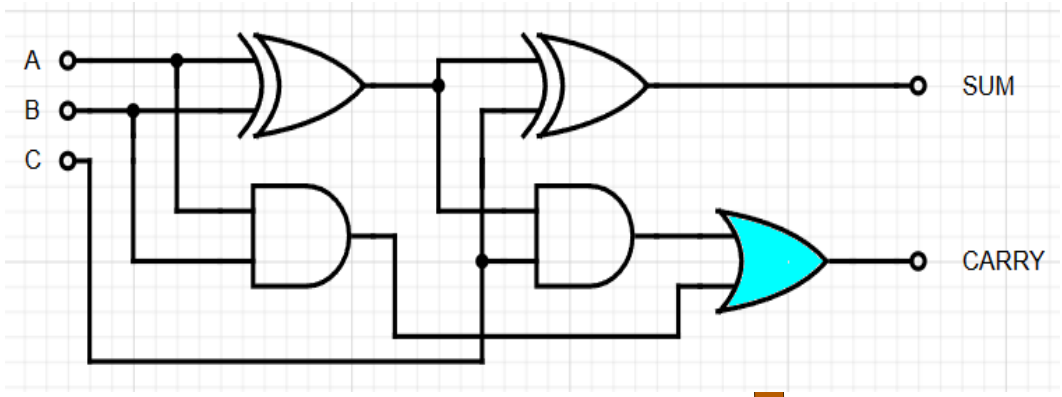


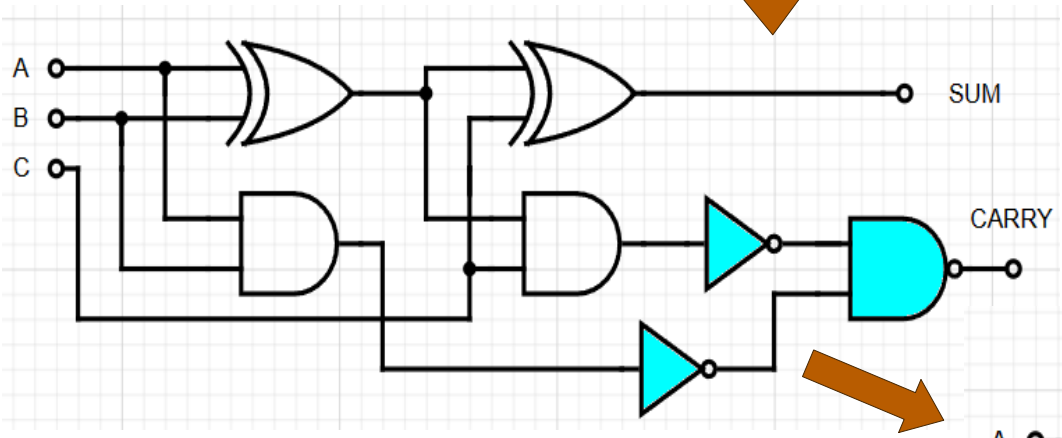
CI	A	B	Q	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Forrás: f-alpha.net - The full adder

A teljes összeadó - még egyszerűbben

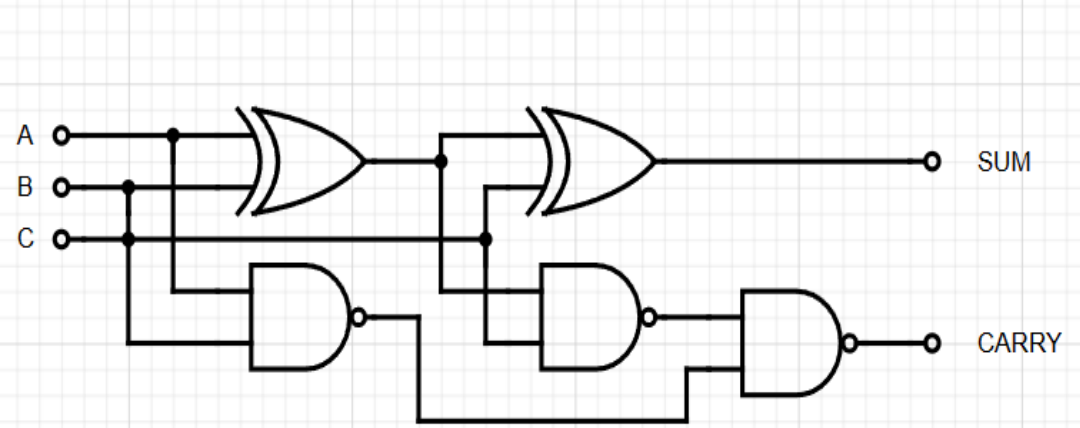




- A teljes összeadóban szereplő **VAGY** kaput helyettesíthetjük két **NEM** és egy **NEM-ÉS** kapuval a **De Morgan** szabály alapján:

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

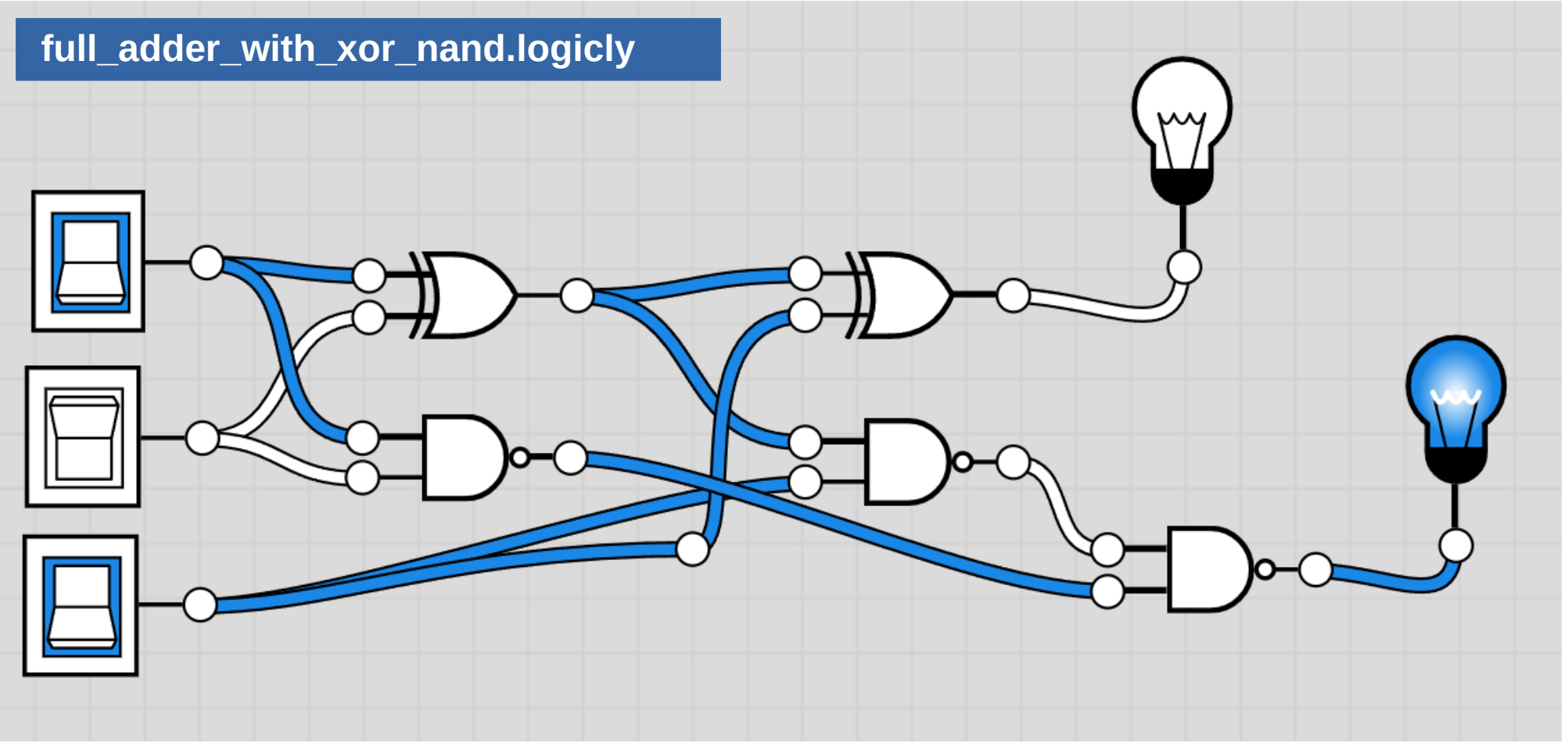
- A két **ÉS** kaput viszont összevonhatjuk az őket követő **NEM** kapuval



- Az átalakításnak köszönhetően a kapcsolás mindössze kétfajta IC-t igényel: **4070** (XOR kapuk) és **4011** (NAND kapuk)

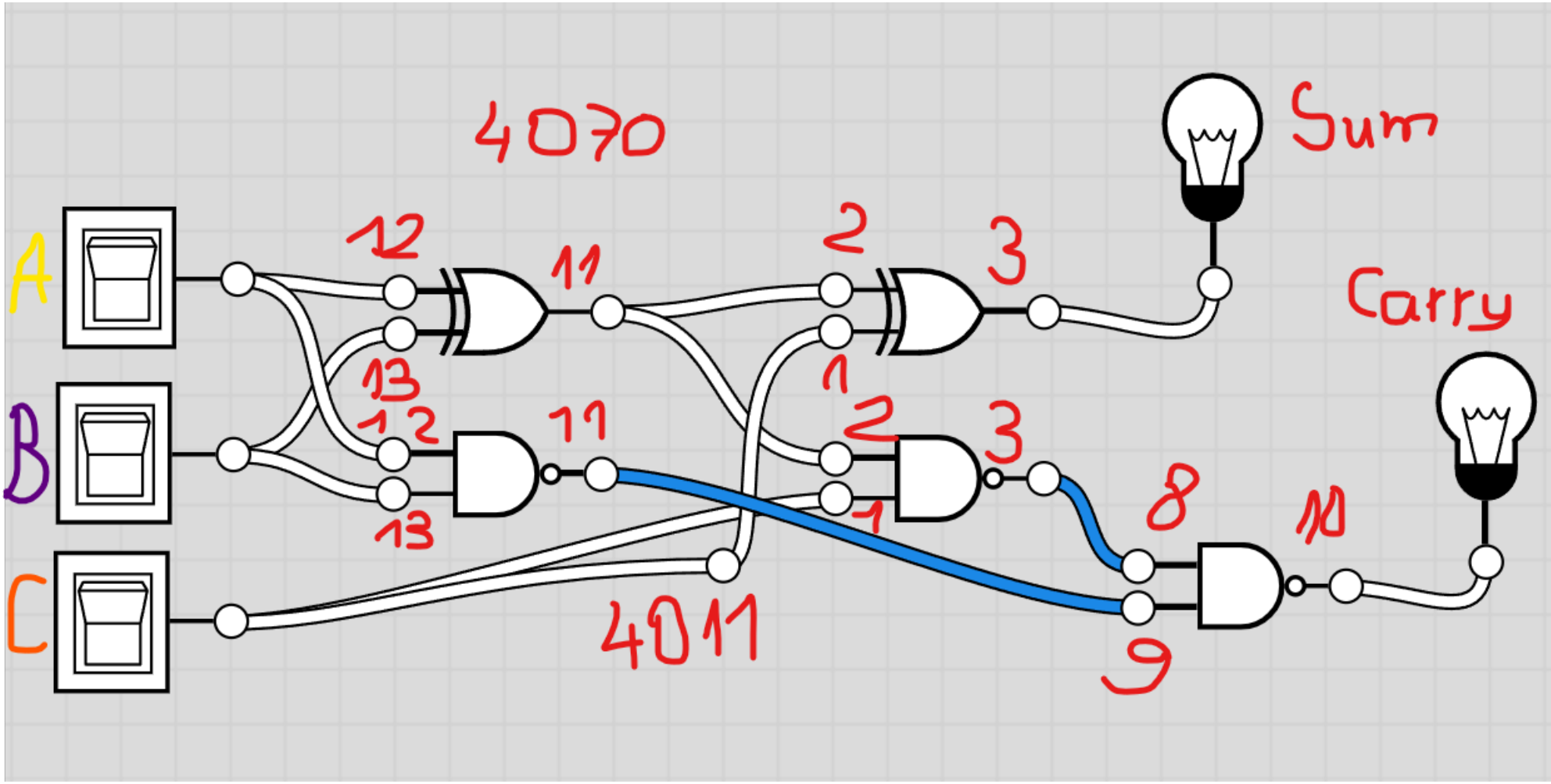
A teljes összeadó - még egyszerűbben

- A módosított teljes összeadó kapcsolást is kipróbáltuk a **Logic.ly** programban, s az az elvárásoknak megfelelően működött



A teljes összeadó - még egyszerűbben

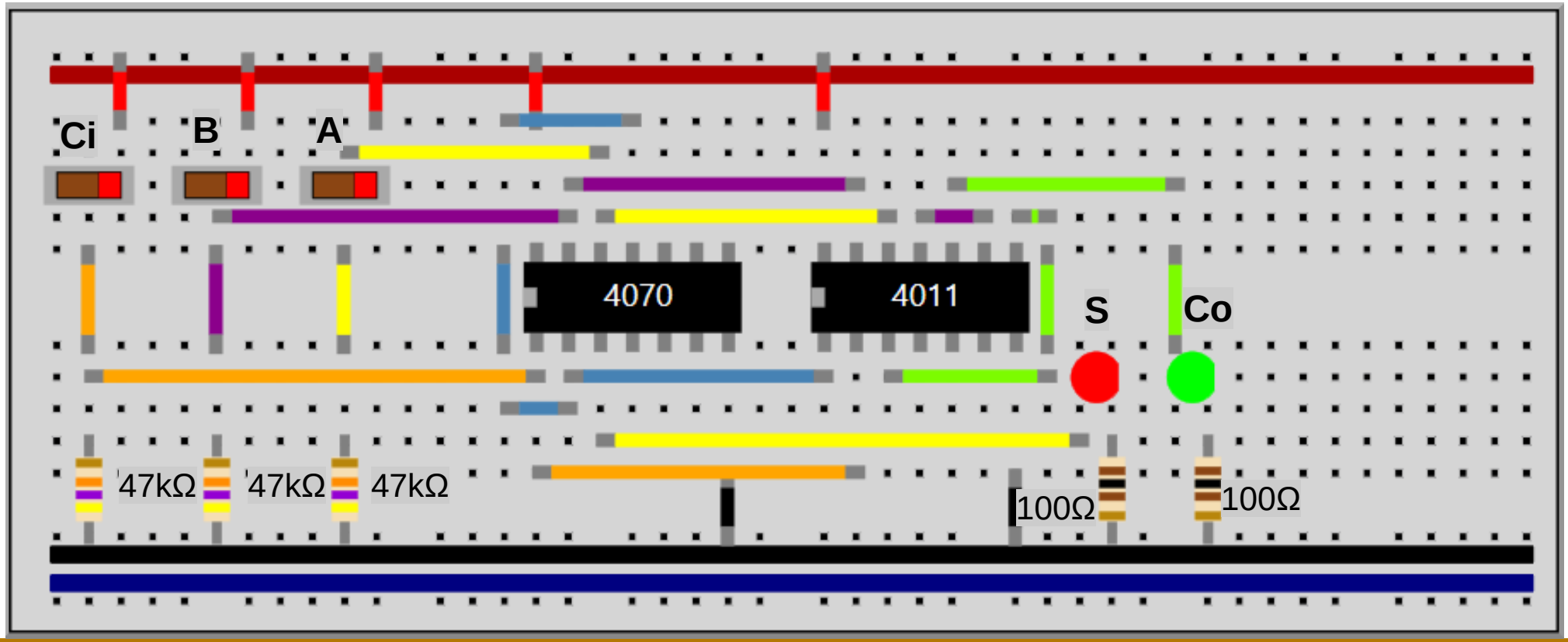
- A megépítéshez tehát egy **4070** (4 db **XOR** kapu) és egy **4011** (4 db **NAND** kapu) típusú IC-re lesz szükség. Az ábrán a javasolt lábkiosztást is feltüntettük



A teljes összeadó modellezése

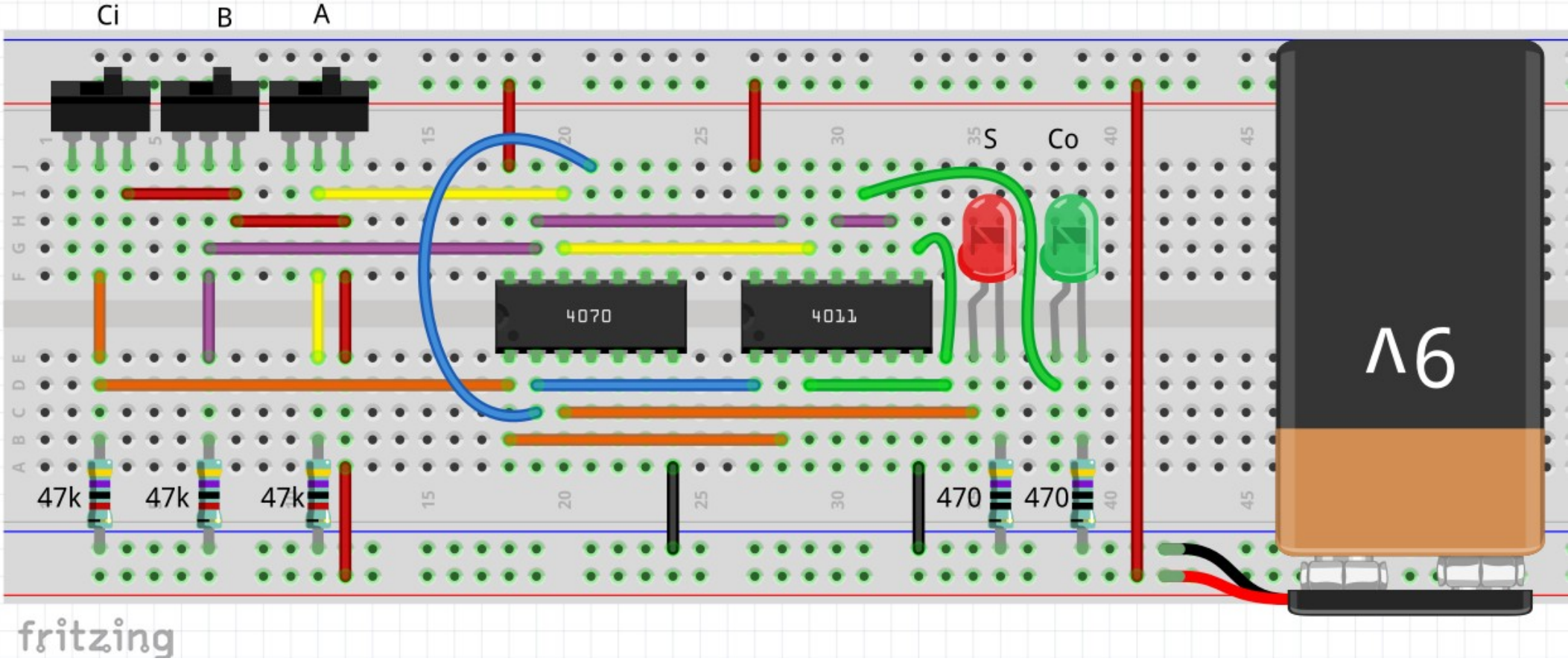
- Modellezzük az áramkört a **Breadboard Simulator**ban! A bemeneteket három tolókapcsolóval vezéreljük (jobbra tolva lesz a bemenet magas szinten, azaz „1” állapotban)
- Az **S** (összeg) kimenet állapotát a piros LED a **C** (átvitel) kimenet állapotát pedig a zöld LED jelzi ki

full_adder.bbrd



A teljes összeadó megépítése

- Az egyszerűsített kapcsolást a **Breadboard Simulator**ral megegyező elrendezésben is megépíthetjük, itt csupán a jobb áttekinthetőség kedvéért változtattunk az elrendezésen egy kicsit
- A **Ci** bemenet az áthozatot, a **Co** kimenet pedig az átvitelt jelzi, az **S** kimenet pedig az összeg (esetünkben az egyesek száma)

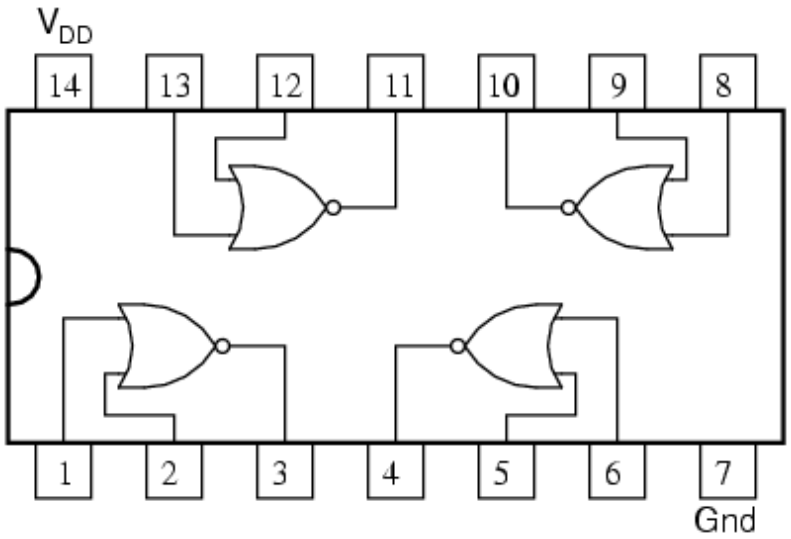


A 4000-es sorozat tipikus tagjai

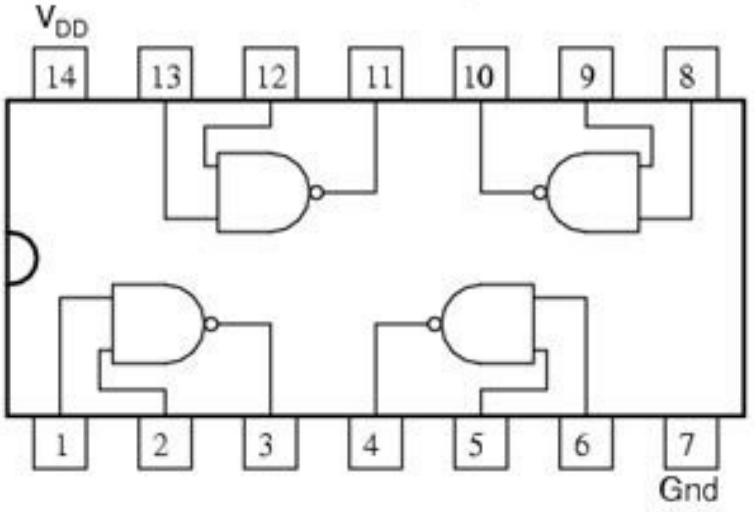
4001 CMOS Quad 2-Input NOR Gate
4011 CMOS Quad 2-Input NAND Gate
4013 CMOS Dual D-Type Flip Flop
4017 CMOS Decade Counter with 10 Decoded Outputs
4021 CMOS 8-Stage Static Shift Register
4022 CMOS Octal Counter with 8 Decoded Outputs
4023 CMOS Triple 3-Input NAND Gate
4025 CMOS Triple 3-Input NOR Gate
4026 CMOS Decade Counter/Divider with Decoded 7-Segment Display Outputs and Display Enable
4027 CMOS Dual J-K Master-Slave Flip-Flop
4028 CMOS BCD-to-Decimal or Binary-to-Octal Decoders/Drivers
4043 CMOS Quad NOR R/S Latch with 3-State Outputs
4046 CMOS Micropower Phase-Locked Loop
4049 CMOS Hex Inverting Buffer/Converter
4050 CMOS Hex Non-Inverting Buffer/Converter
4051 CMOS Single 8-Channel Analog Multiplexer/Demultiplexer with Logic-Level Conversion
4052 CMOS Differential 4-Channel Analog Multiplexer/Demultiplexer with Logic-Level Conversion
4053 CMOS Triple 2-Channel Analog Multiplexer/Demultiplexer with Logic-Level Conversion
4060 CMOS 14-Stage Ripple-Carry Binary Counter/Divider and Oscillator
4066 CMOS Quad Bilateral Switch
4069 CMOS Hex Inverter
4070 CMOS Quad Exclusive-OR Gate
4071 CMOS Quad 2-Input OR Gate
4072 CMOS Dual 4-Input OR Gate
4073 CMOS Triple 3-Input AND Gate
4075 CMOS Triple 3-Input OR Gate
4081 CMOS Quad 2-Input AND Gate
4082 CMOS Dual 4-Input AND Gate
4093 CMOS Quad 2-Input NAND Schmitt Triggers
4094 CMOS 8-Stage Shift-and-Store Bus Register

A 4000-es sorozat tipikus tagjai

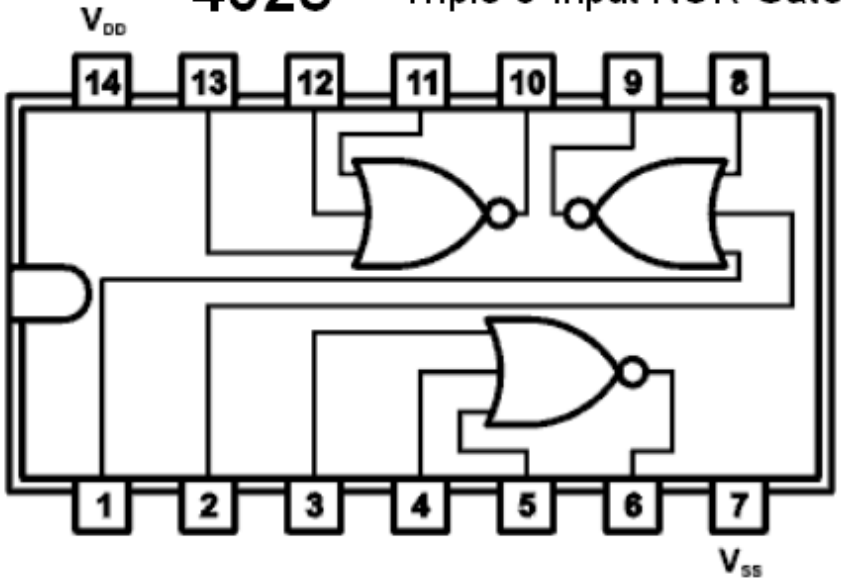
4001 Quad 2-Input NOR Gate



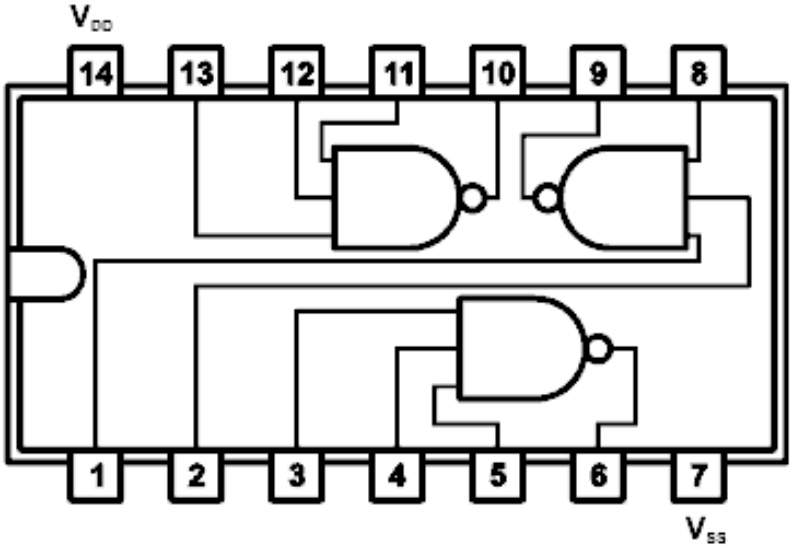
4011 Quad 2-input NAND



4025 Triple 3-Input NOR Gate

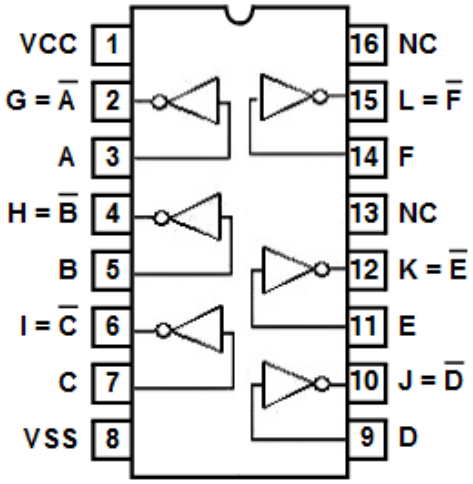


4023 Triple 3-Input NAND Gate

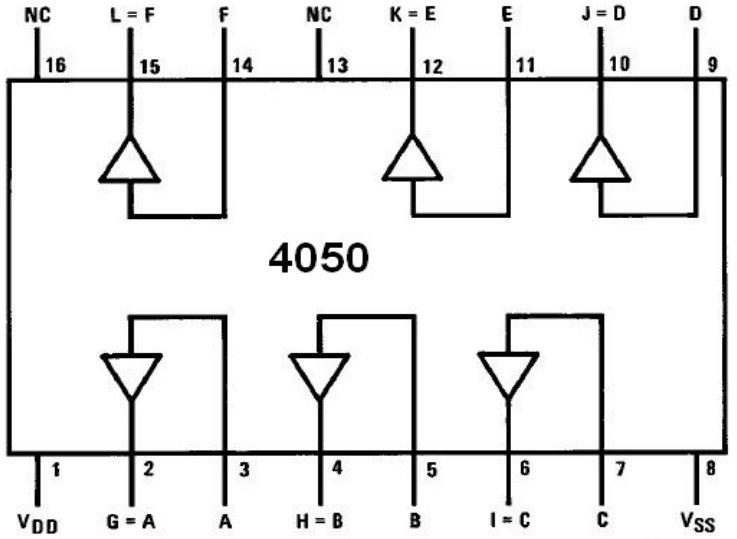


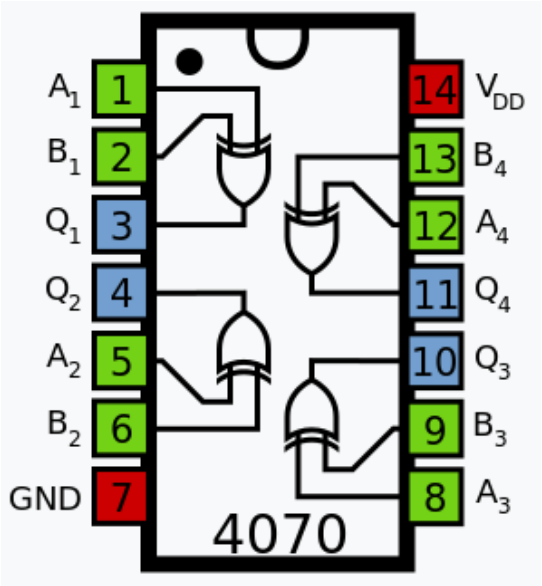
A 4000-es sorozat tipikus tagjai

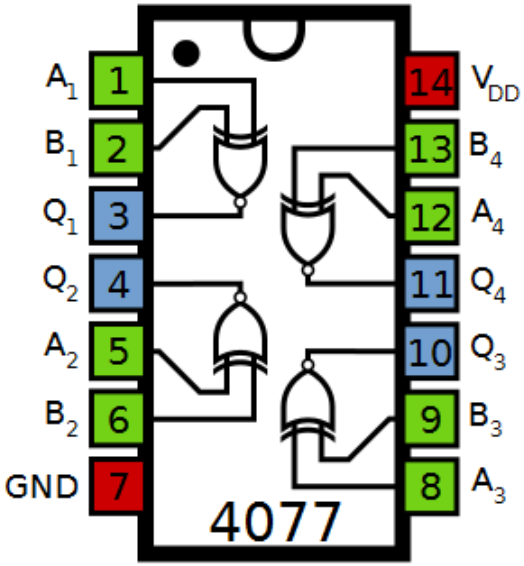
4049
Hex Inverting Buffer/Converter

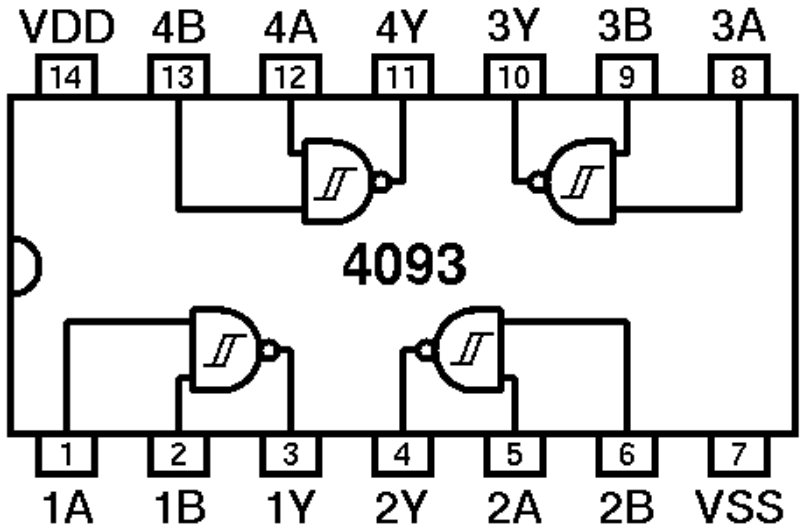


4050 Hex Non-Inverting Buffer/Converter









A 4000-es sorozat tipikus tagjai