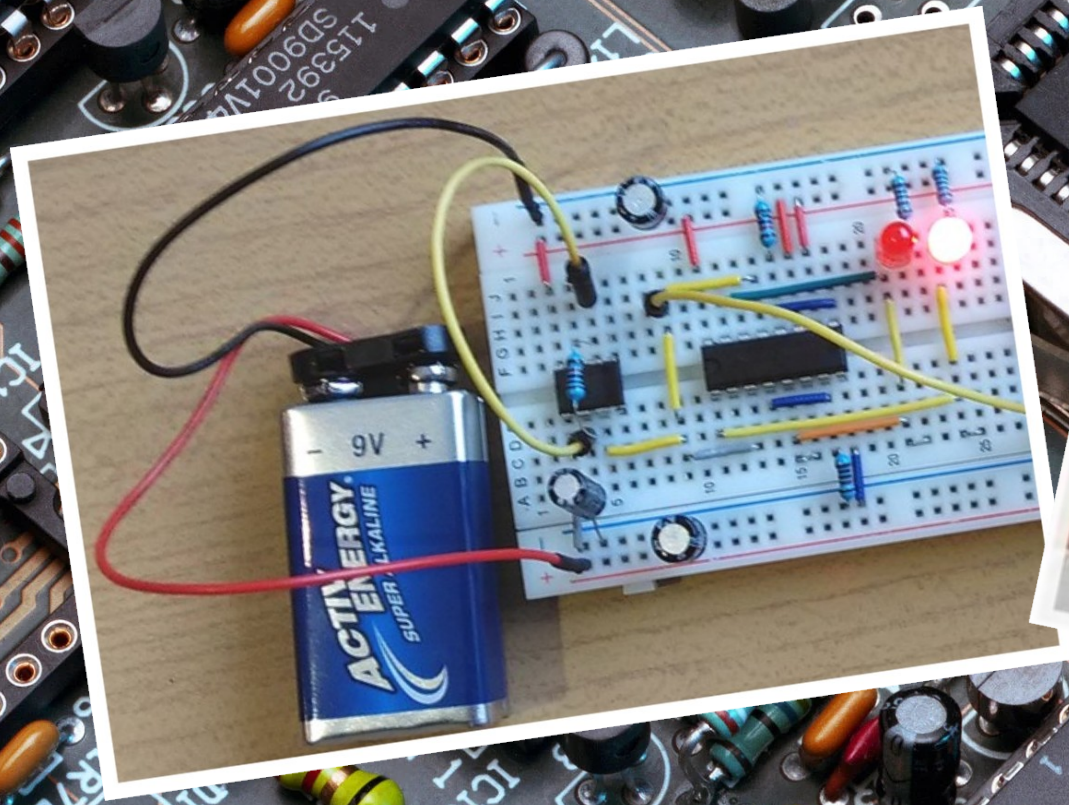
A digitális elektronika alapjai



6. Kombinációs logikai hálózatok – 3. rész

Felhasznált és ajánlott irodalom

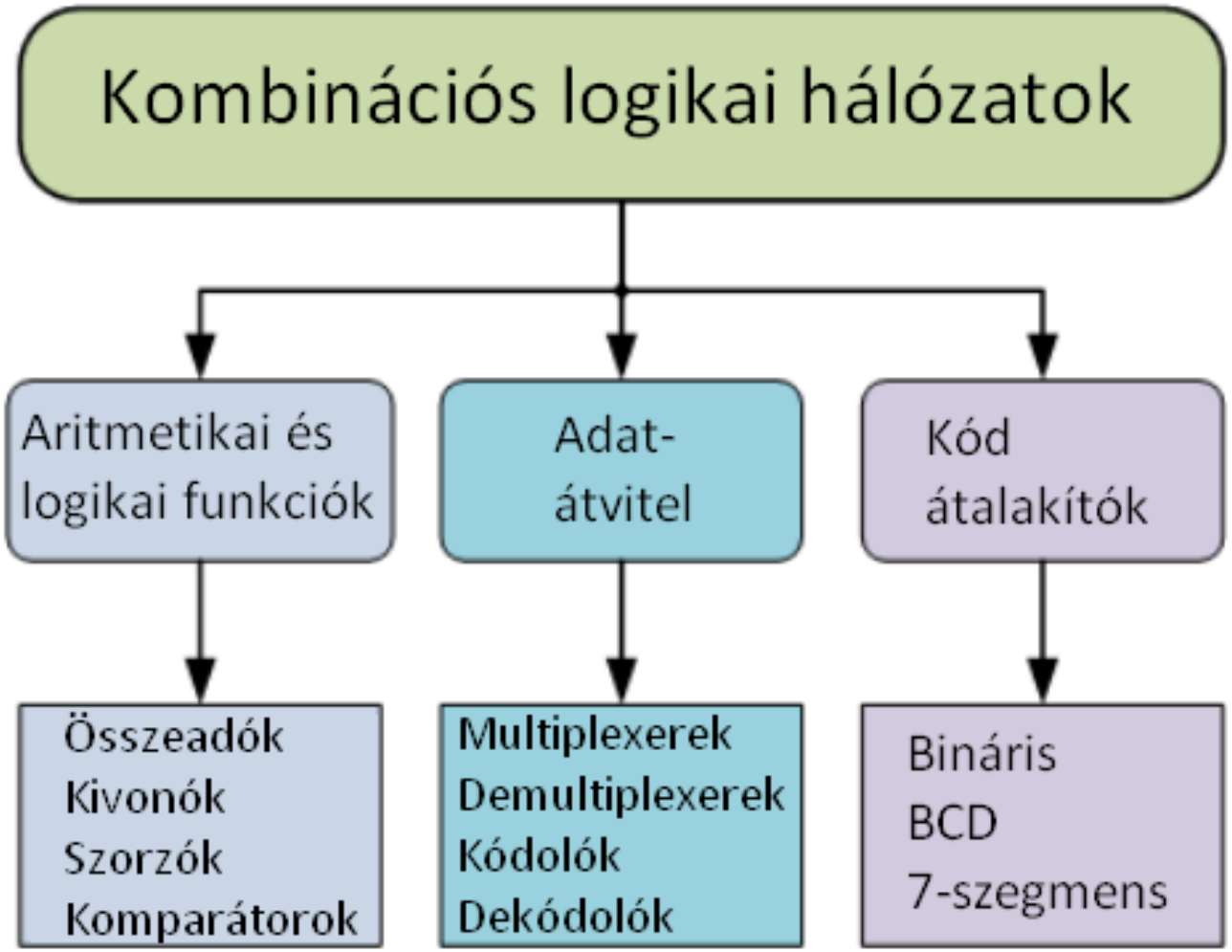
- Gulyás Dénes: Számítógép architektúrák *(interaktív jegyzet)*
- Mike Gábor: A digitális elektronika alapjai *(jegyzet és videók)*
- Zalotay Péter: Digitális technika
- Végh János: Ismerkedés a digitális elektronikával
- Mészáros Miklós: Logikai algebra alapjai, logikai függvények I.
- Mingesz Róbert: Digitális technikai tananyagok
- F-alpha.net: Digital Electronics
- Electronics Tutorials: Logic Gates
- M. Morris Mano and Michael D. Ciletti: Digital Design
- Simon Frazer University: CMPT-150: Introduction to Computer Design

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)

A kombinációs logikai hálózatok csoportosítása

- A kombinációs logikai hálózatok legelterjedtebben használt funkcionális moduljai az alábbiak szerint csoportosíthatók:



A mai és a következő előadásban érintett témakörök

- **Komplemensképzés**
- **Negatív számok ábrázolása**
- **4 bites összeadó/kivonó**
- **Magnitúdó komparátor**
- **Dekódolók**
- Multiplexerek és demultiplexerek
- Prioritás kódoló
- Veitch és Karnaugh diagramok
- Logikai függvények egyszerűsítése a Veitch/Karnaugh diagram alapján
- Logisim és leszármazottjai (Digital, Logisim-evolution)

Komplemensképzés

- A kétértékű elemek halmazán értelmezett **Boole-algebra** egyik posztulátuma (követelménye), hogy a halmaz minden elemének van **komplemense**
- Ha ezt kiterjesztjük a kettes számrendszerben felírt számok számjegyeire, akkor úgy képezhetjük a számok komplemensét, hogy az egyeseket nullára, a nullákat egyesekre írjuk át, például: $0100\ 1010_2$ komplemense $1011\ 0101_2$
- A fenti **komplemensképzés szabályát** másképp is írhatjuk:
  $0100\ 1010_2$ komplemense = $1111\ 1111_2$ - $0100\ 1010_2$ = $1011\ 0101_2$
  tehát a „csökkentett alapszám komplemensről” (*diminished radix complement*) van szó, amit esetünkben **1-es komplemensnek** hívnak
- Negatív számok ábrázolására azonban az 1-es komplemens nem jó, mert:

0000 0011 = +3
0000 0010 = +2
0000 0001 = +1
0000 0000 = +0
1111 1111 = −0
1111 1110 = −1
1111 1101 = −2
1111 1100 = −3

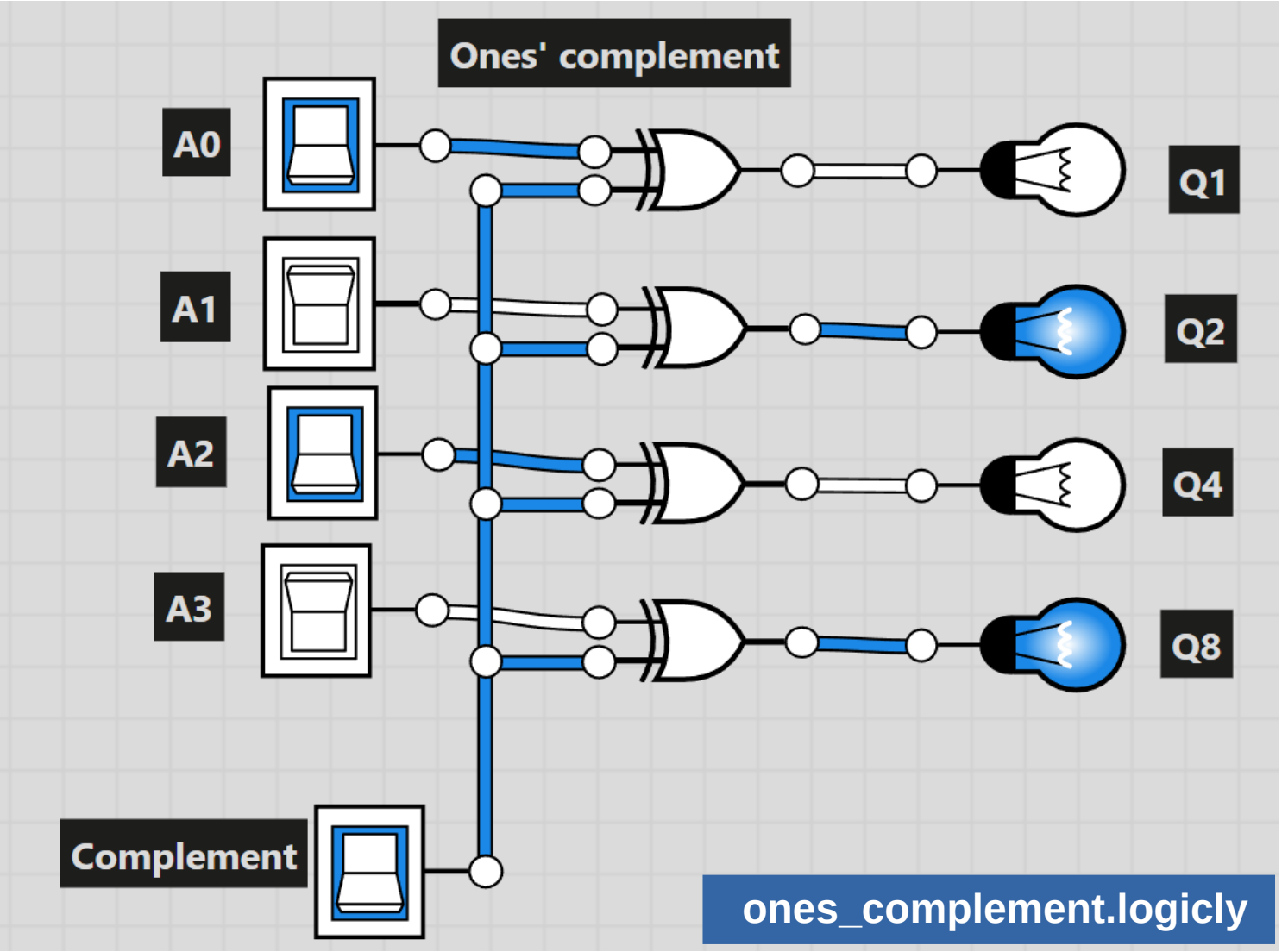
A probléma itt az, hogy két, különböző nullánk lett, emiatt a számegyenes középen „kettészakadt”.

Komplemensképző áramkör

- Ahogy a 3. előadás 19. diáján láthattuk, **XOR** kapukkal tudunk invertálni, ha a másik bemenetet 1-re állítjuk, ezt terjesztjük ki:

Az **XOR** művelet igazságtáblázata:

Cpl	A	Q
0	0	0
0	1	1
1	0	1
1	1	0



Negatív számok ábrázolása: kettes komplemens

- Visszatérve a negatív számokra: az 1-es komplemens azért nem volt jó, mert két, különböző nullát eredményezett:

0000 0011 = +3
0000 0010 = +2
0000 0001 = +1
0000 0000 = +0
1111 1111 = −0
1111 1110 = −1
1111 1101 = −2
1111 1100 = −3

A probléma kiküszöbölésére a negatív tartományt eggyel feljebb kell tolni, azaz 1-et még hozzá kell adni

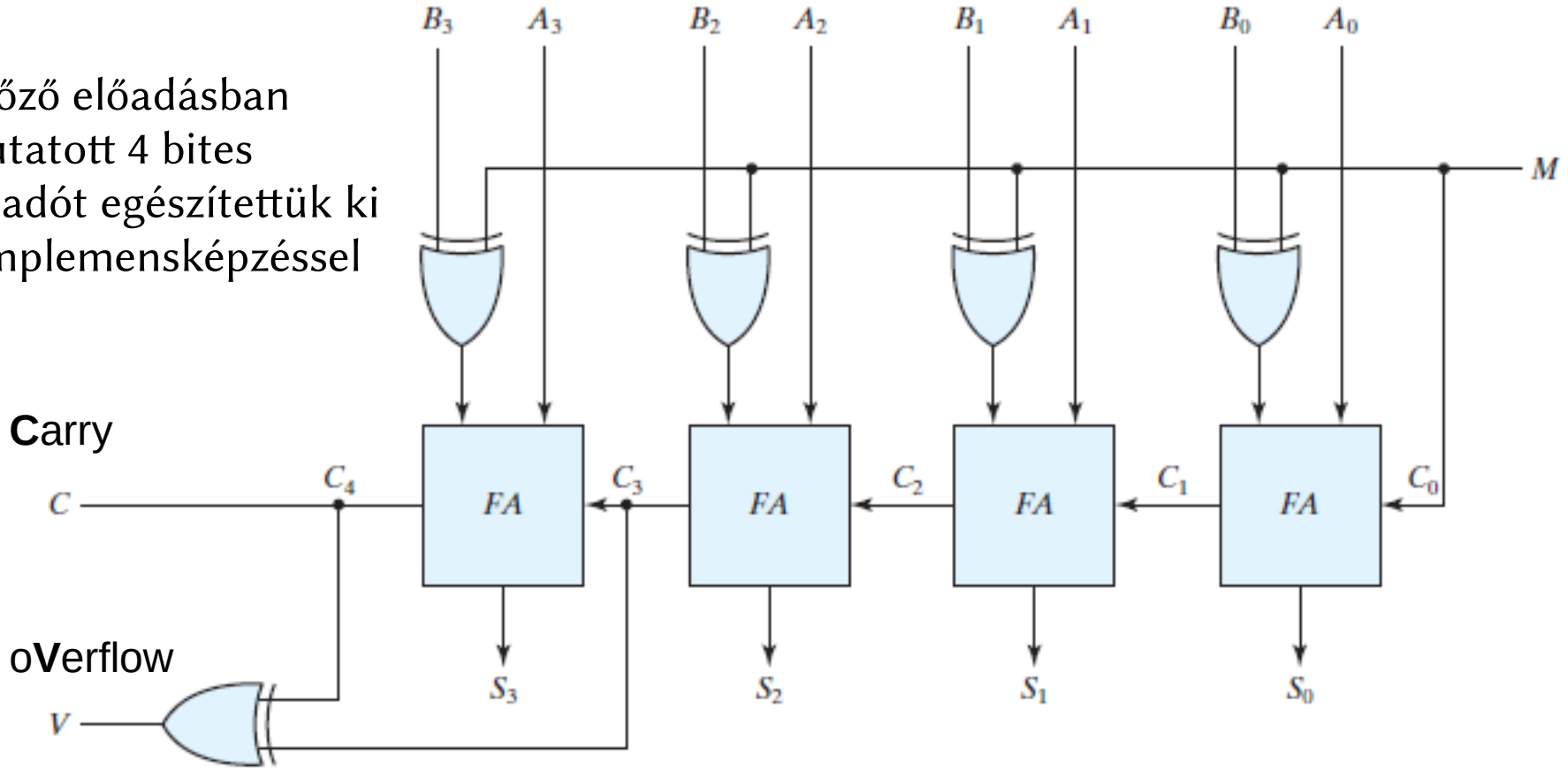
0111 1111 = **+127** (max)
. . .
0000 0011 = +3
0000 0010 = +2
0000 0001 = +1
0000 0000 = +0
1111 1111 = −1
1111 1110 = −2
1111 1101 = −3
1111 1100 = −4
. . .
1000 0000 = **−128** (min)

- A fenti korrekcióval a negatív számokra az ún. „alapszám komplemens” (*radix complement*) értékeket kapjuk, amit esetünkben kettes komplemensnek hívunk
- Kettes komplemens képzése kétféle módszerrel is lehetséges:
  - Egyes komplemens képzése után az eredményhez 1-et hozzáadunk
  - N szám kettes komplemense = 2^n − N (esetünkben 256 − N), kivéve, az N = 0 esetet, amikor -N = N = 0
- E számábrázolás előnye, hogy a kivonás negatív szám hozzáadásával végezhető, s a legmagasabb helyiértékű bit tükrözi, hogy negatív-e a szám

Bináris összeadó/kivonó

- Ha **M = 0,** akkor az alábbi áramkör összeadóként működik
  Ha **M = 1,** akkor a kapcsolás kivonó áramkörként működik
- Kivonásnál a **XOR** kapuk segítségével a **B** szám egyes komplemensét vesszük, s a C_0 áthozat bemenetet is ’1’-be állítjuk, így az **S** eredmény **S = A + B** egyes komplemense + 1 lesz, ami **S = A – B**-nek felel meg

Az előző előadásban bemutatott 4 bites összeadót egészítettük ki a komplemensképzéssel



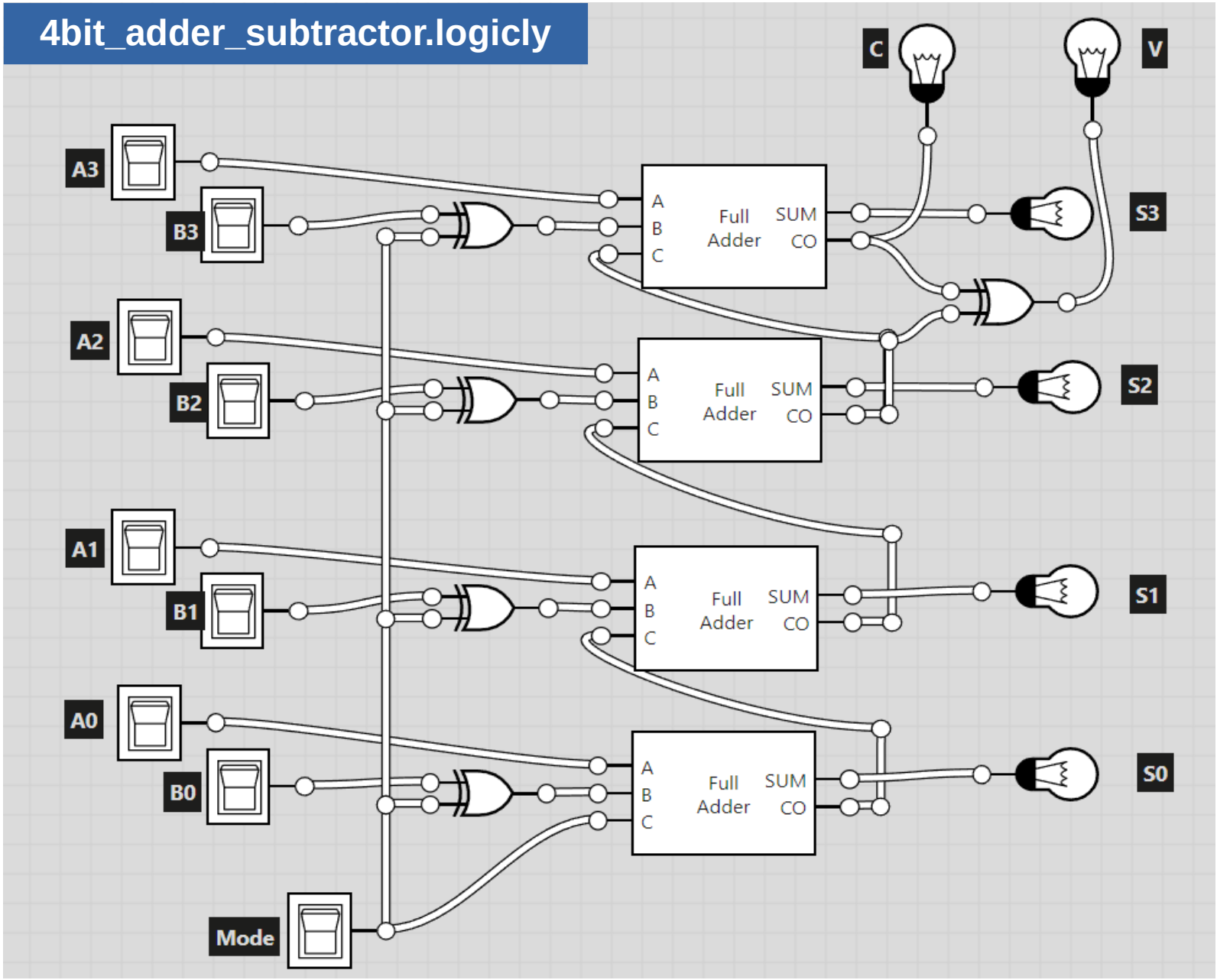
Túlcsordulás figyelése

- Előjel nélküli számok összeadásánál a **C** (Carry) bit jelzi, ha túlcsordulás történt, előjeles számoknál azonban használhatatlan
- Előjeles számok összeadásánál a $\mathbf{V = C_{MSB} \oplus C_{MSB-1}}$ bit jelzi a túlcsordulást
- A lenti példákban egy mikrovezérlő státuszbitjeinek viselkedését mutatjuk be

Összeadó	Előjel nélkül	Előjelesen	Összeadó	Előjel nélkül	Előjelesen
0x01 +0xFF 0x00	1 +255 0	1 + -1 0	0x80 +0xFF 0x7F	128 +255 127	-128 + -1 +127
C=1, Z=1, OV=0, N=0			C=1, Z=0, OV=1, N=0		

Összeadó	Előjel nélkül	Előjelesen	Összeadó	Előjel nélkül	Előjelesen
0x01 +0x7F 0x80	1 +127 128	1 +127 -128	0x80 +0x20 0xA0	128 + 32 160	-128 + +32 -96
C=0, Z=0, OV=1, N=1			C=0, Z=0, OV=0, N=1		

4-bites bináris összeadó/kivonó



Magnitúdó komparátorok

- Az aritmetikai/logikai funkcionális modulok közül már foglalkoztunk az **összeadó és kivonó** áramkörökkel, a **komparátorok** (összehasonlítók) közül pedig az **egyenlőség vizsgálattal**
- Most ismerkedjünk meg a kisebb/nagyobb/egyenlő relációkat figyelő összehasonlítóval, a **magnitúdó komparátorral!**
- Az igazságtáblázat alapján 1 bitre könnyen felírhatjuk a logikai függvényeket:

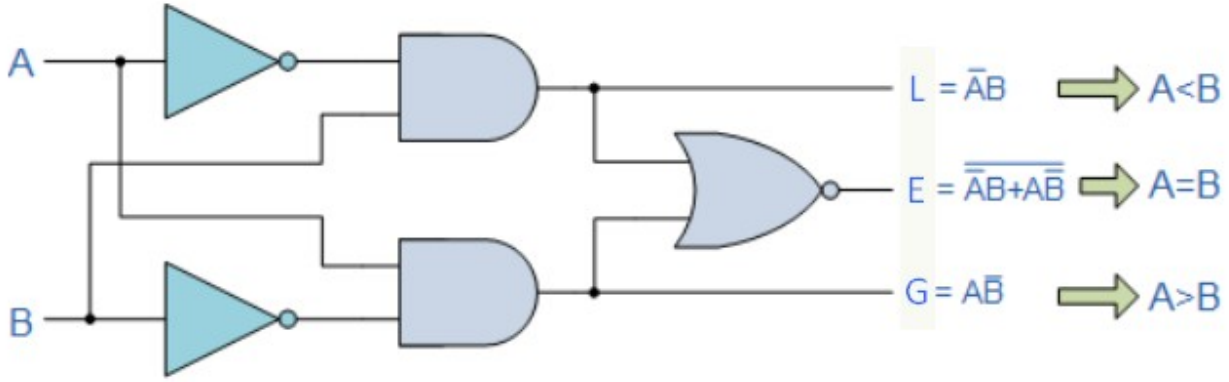
A	B	A<B	A>B	A=B
0	0	**0**	**0**	**1**
0	1	**1**	**0**	**0**
1	0	**0**	**1**	**0**
1	1	**0**	**0**	**1**

Less Greather Equal

- **A<B:** $L = \overline{A} \cdot B$
- **A>B:** $G = A \cdot \overline{B}$
- **A=B:** $E = \overline{A} \cdot \overline{B} + A \cdot B$

vagy másképp:

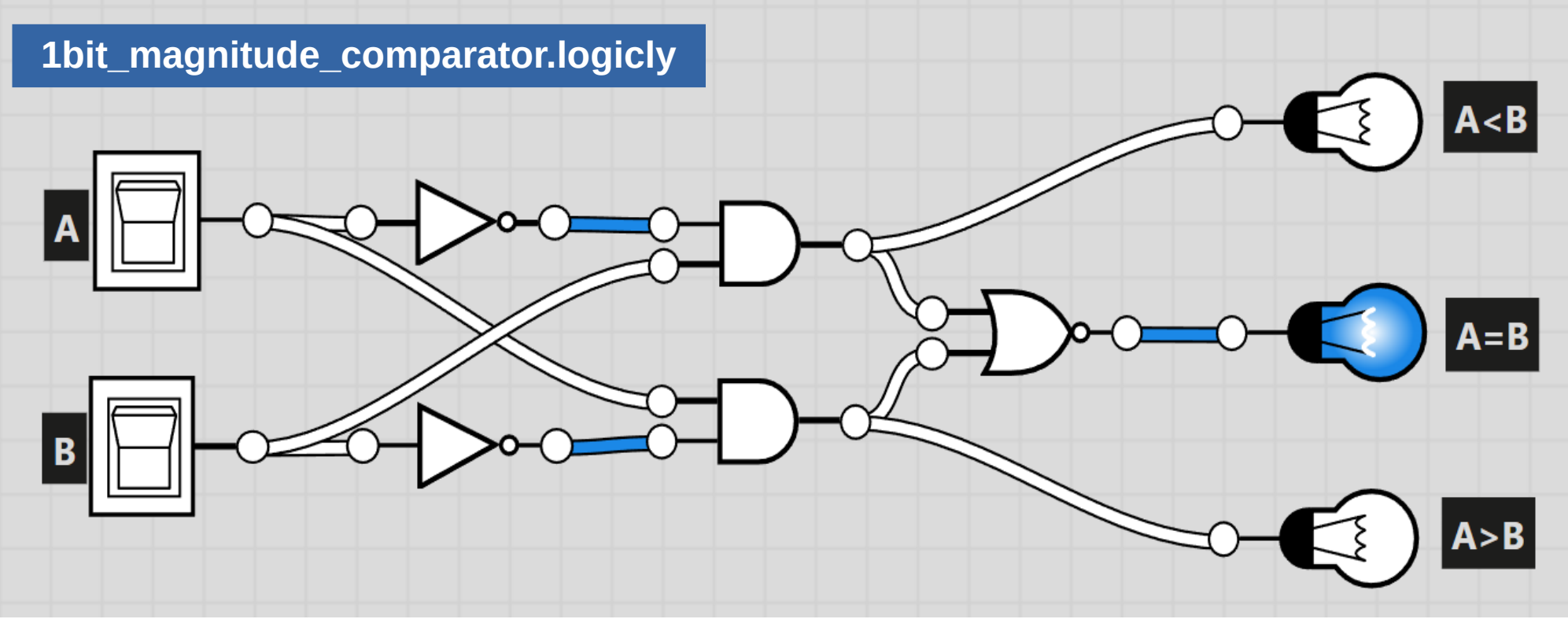
$$E = \overline{L} \cdot \overline{G} = \overline{L + G} = NOR(L, G)$$



1bites magnitúdó komparátor

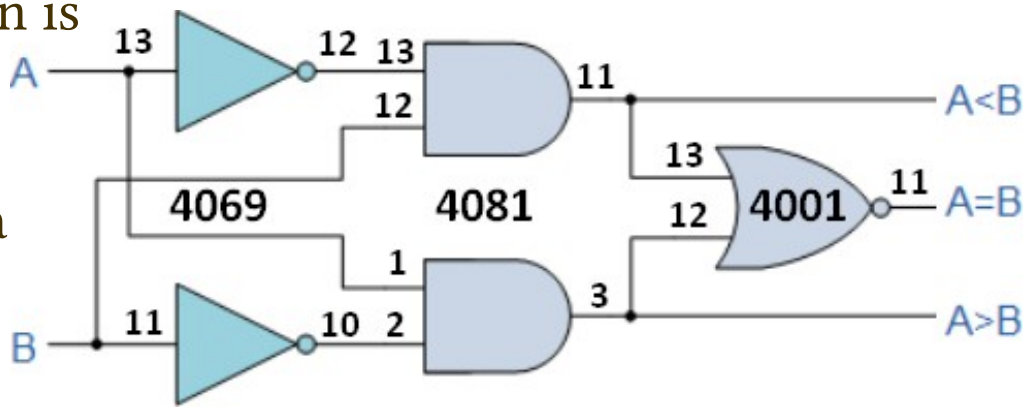
- Próbáljuk ki az előző oldalon bemutatott kapcsolást a **Logic.ly** programban és ellenőrizzük a működést és az igazságtáblázattal való egyezést!

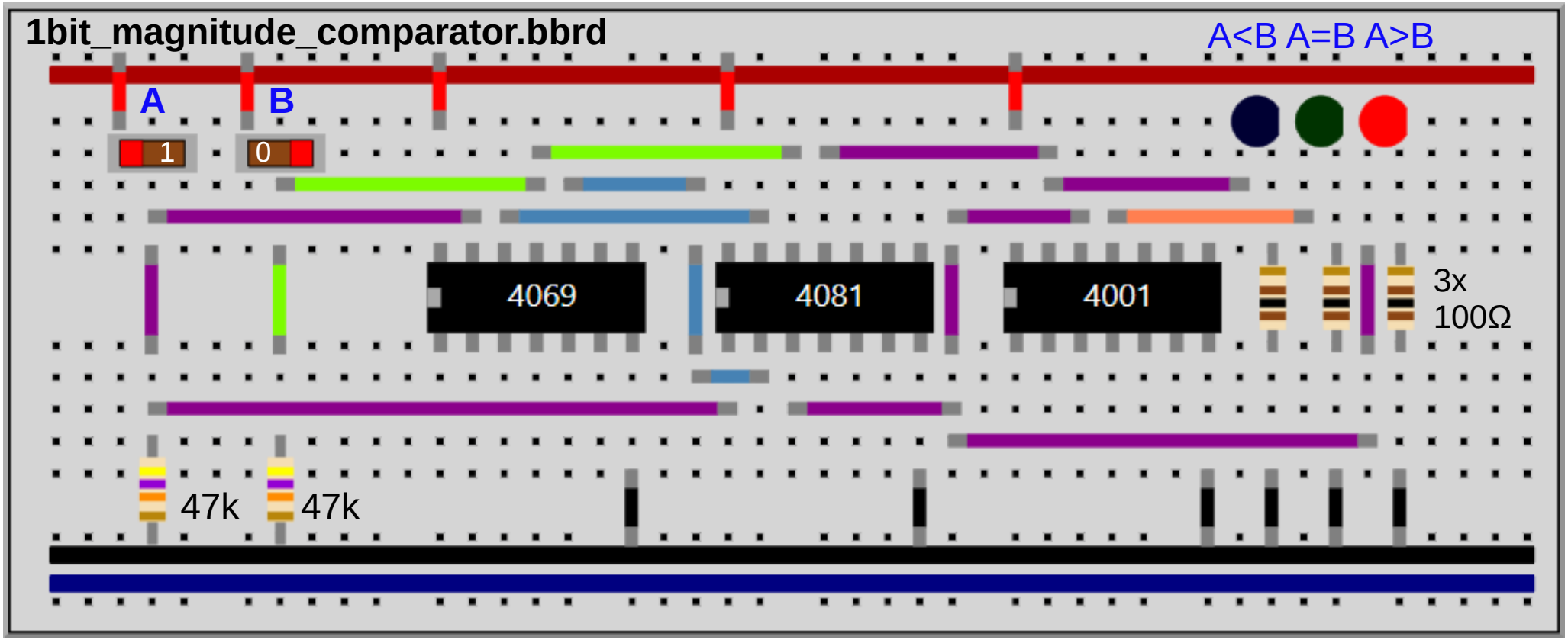
A	B	A<B	A>B	A=B
0	0	**0**	**0**	**1**
0	1	**1**	**0**	**0**
1	0	**0**	**1**	**0**
1	1	**0**	**0**	**1**



1bites magnitúdó komparátor

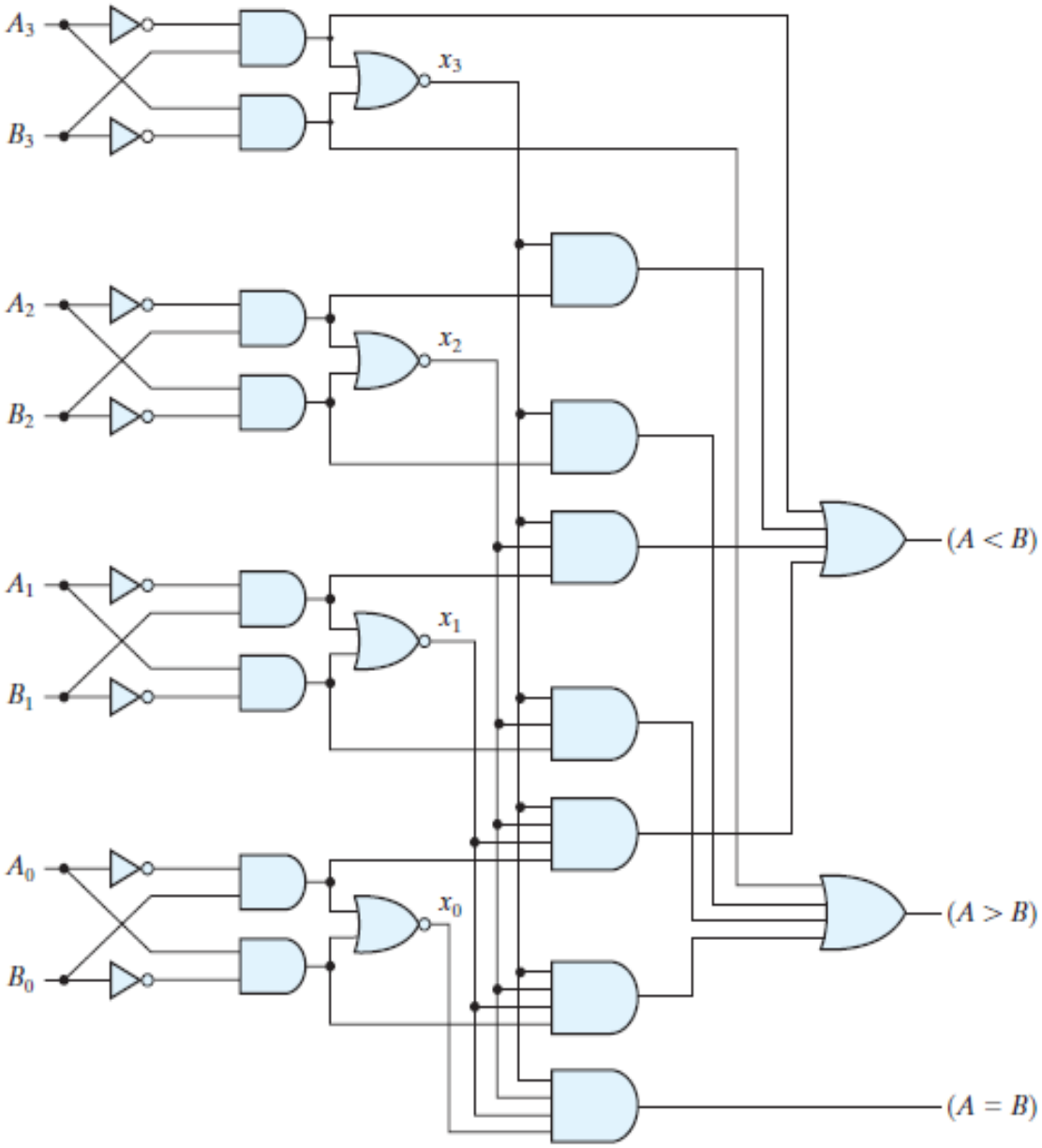
- Az áramkört a Breadboard Simulator-ban is kipróbálhatjuk, vagy megépíthetjük a 30 darabos CMOS IC készletünkből
- A **NEM** kapukat a 4069, az **ÉS** kapukat a 4081, a **NEM-VAGY** kaput pedig a 4001 típusú IC tartalmazza



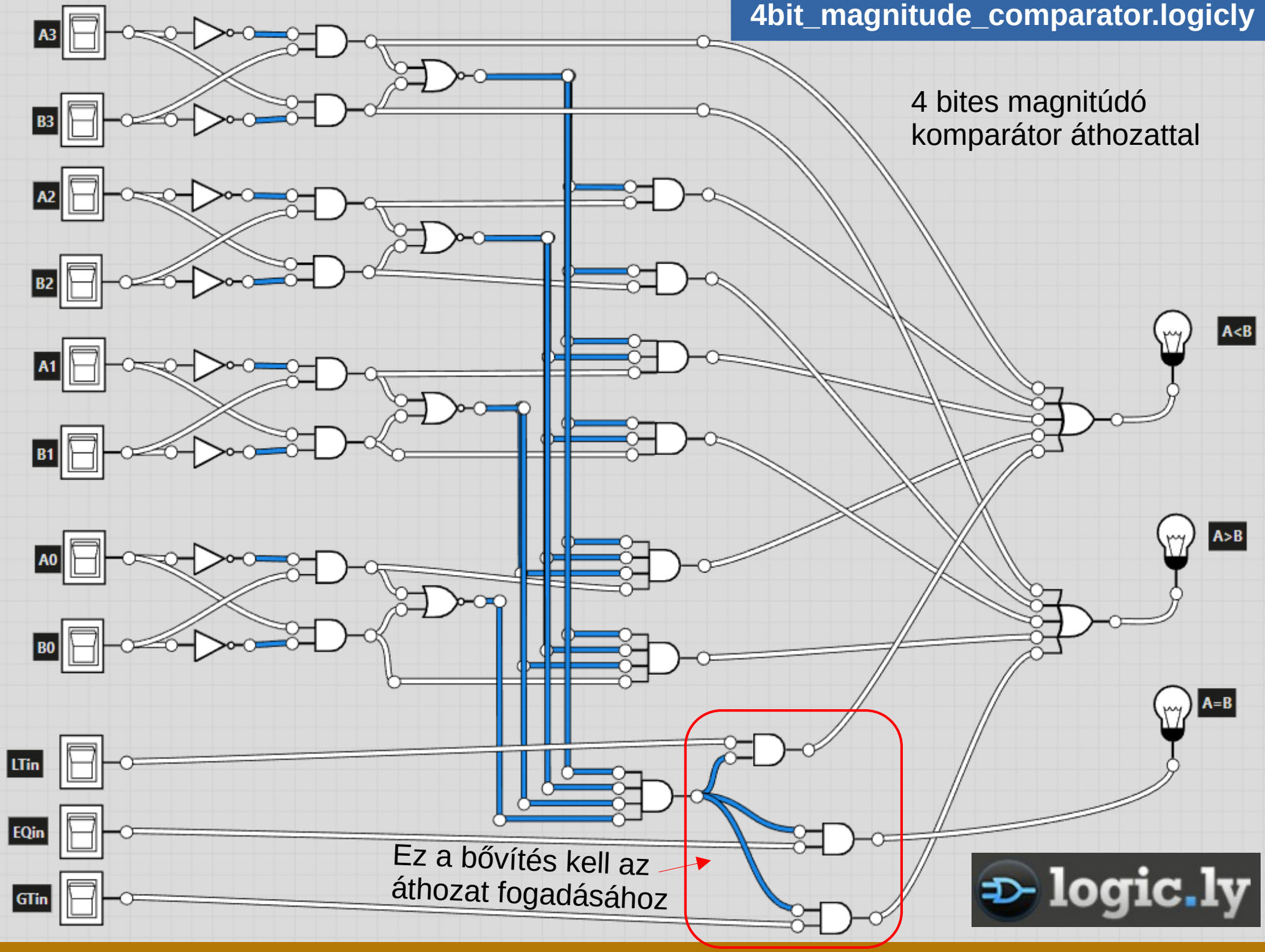


4-bites magnitúdó komparátor

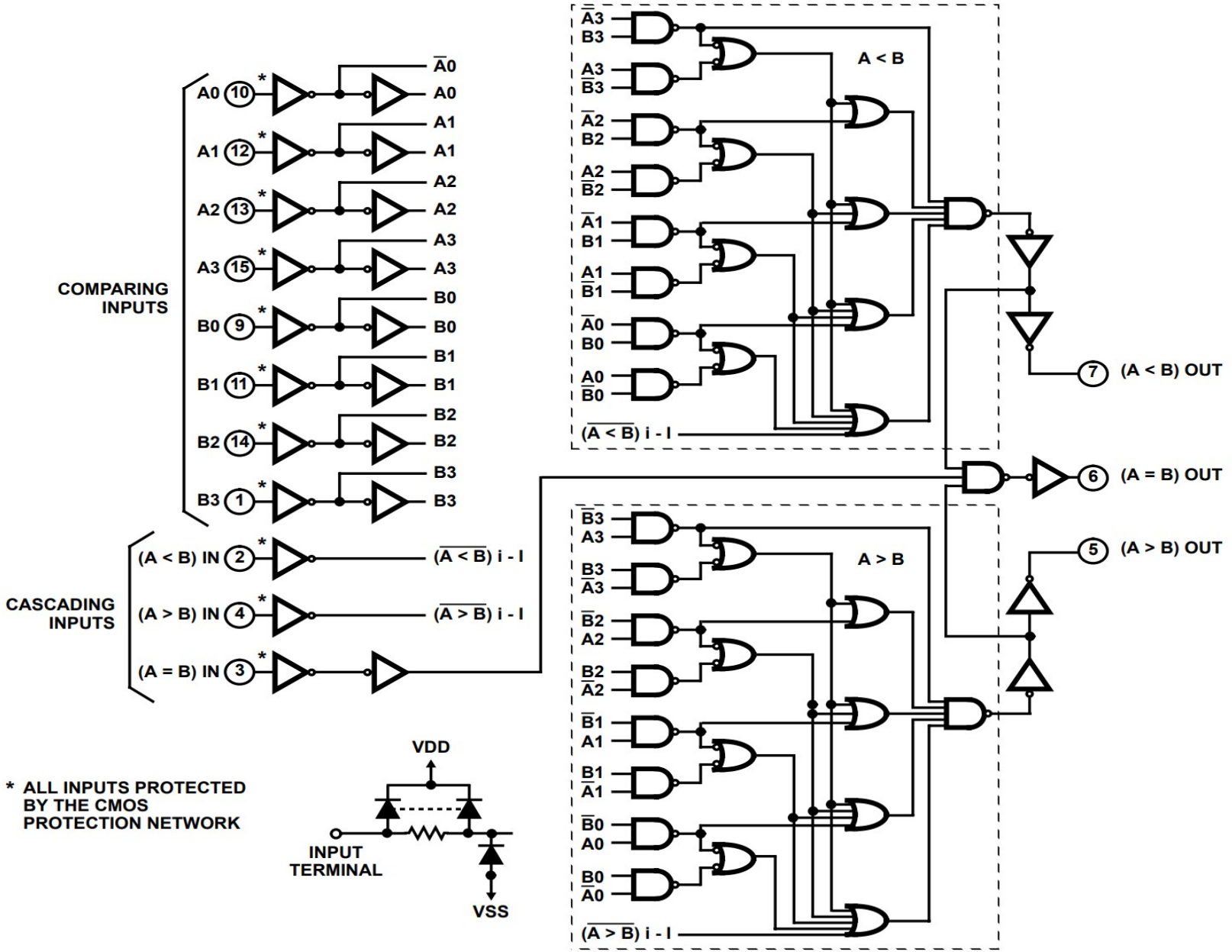
- Az **A > B** vagy **A<B** feltételek vizsgálatánál ha a legmagasabb helyiértéken egyezés van, akkor az eggyel alacsonyabb helyiértékű bit dönt. Ha ott is egyezés van, akkor a még alacsonyabb helyiértékű bit dönt, és így tovább…
- Kis kiegészítéssel az áthozatok kezelése is megoldható, s akkor az áramkör több bitesre bővíthető
- Ilyen, áthozatot is kezelő, 4-bites összehasonlítót valósít meg a **CD4063**-as CMOS IC is



4 bites magnitúdó komparátor áthozattal

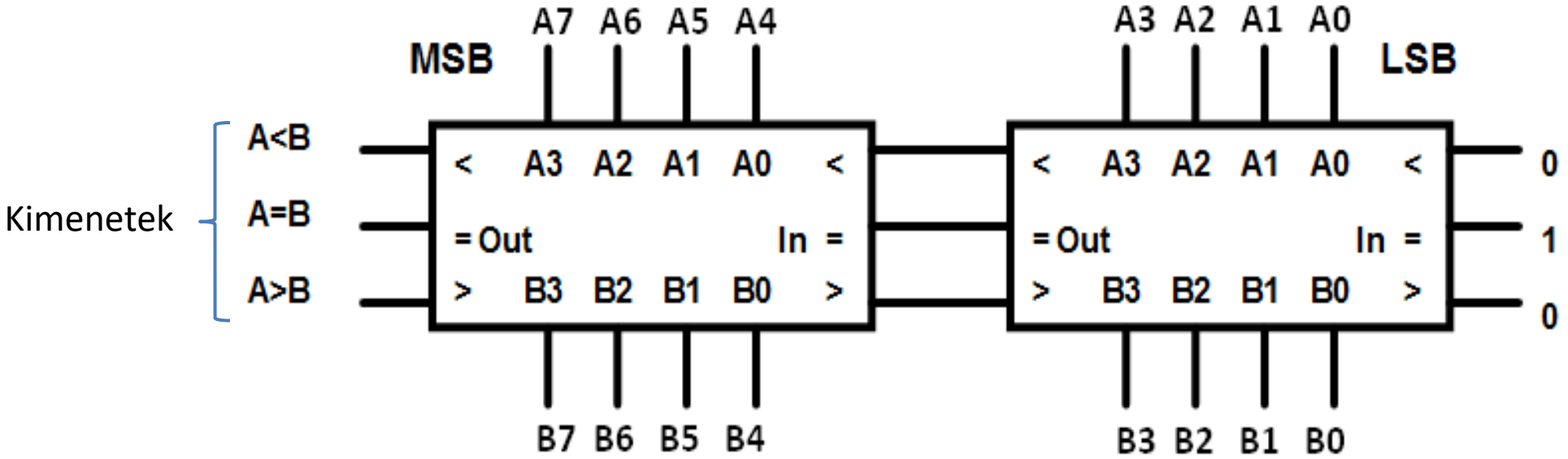


A CD4063 komparátor belső felépítése

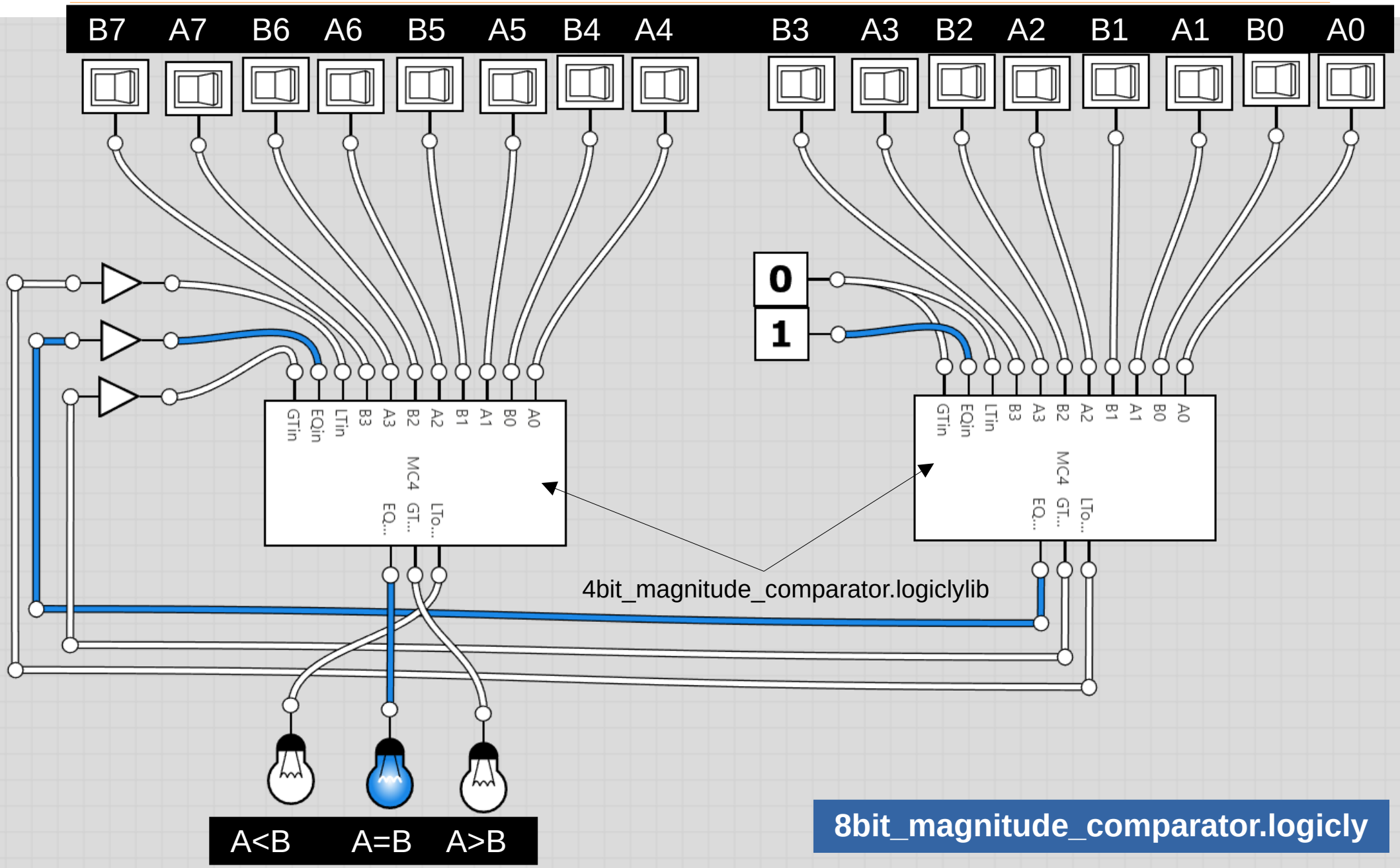


8-bites magnitúdó komparátor

- A **CD4063** logikai IC olyan 4-bites magnitúdó komparátort tartalmaz, ami „áthozat” bemeneteket is tartalmaz, tehát kaszkádba köthető.
- A legelső komparátor kaszkádosító bemeneteit úgy kell beállítani, mintha egy előző komparátor egyenlőséget jelezne.
- Két **CD4063** IC-vel az alábbi ábra szerint építhetünk 8-bites összehasonlítót. A megépítéshez az f-alpha.net oldalon találunk útmutatót.



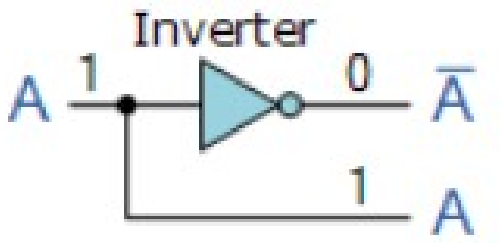
8 bites magnitúdó komparátor

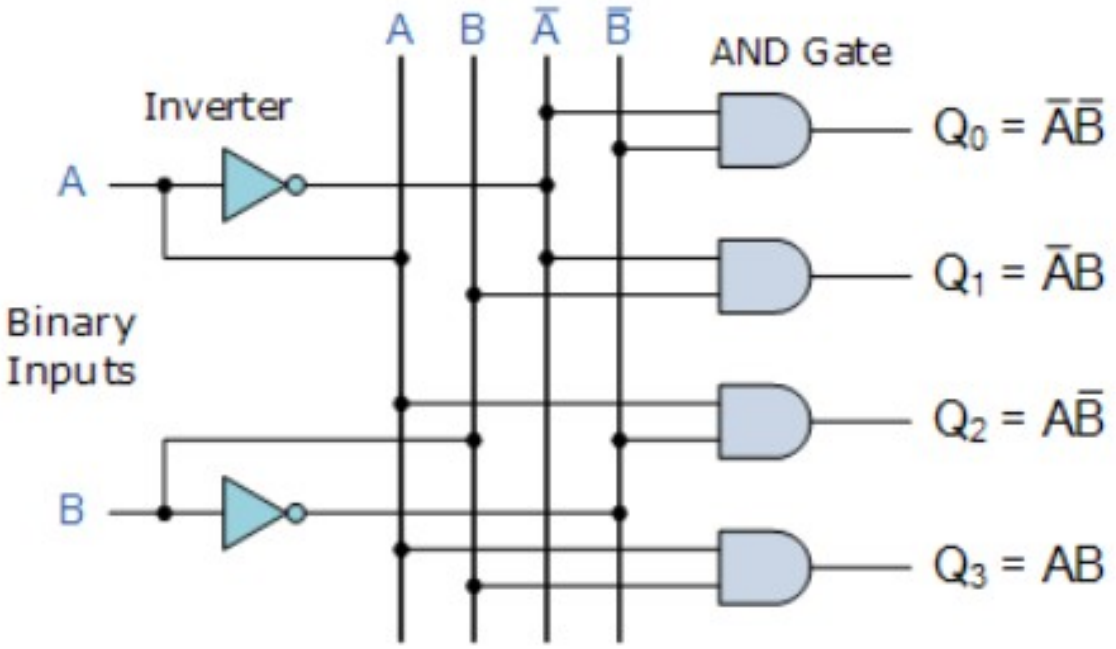


8bit_magnitude_comparator.logicly

Bináris dekódolók

- A **bináris dekódoló** olyan kombinációs logikai áramkör, ami az n db bemenetére érkező bináris információt $m \leq 2^n$ egyedi kimenő jellé alakítja. Tipikus alkalmazásai: cím- és utasításdekódolók, kijelző vezérlő dekódoló
- Az inverter (**NEM** kapu) például 1 bites bináris dekódolónak tekinthető.
- Általánosságban az n bitről m kimenetre dekódoló áramkör m darab **mintermet** valósít meg
- **Példa: 2 bitről 4 vonalra dekódoló áramkör**

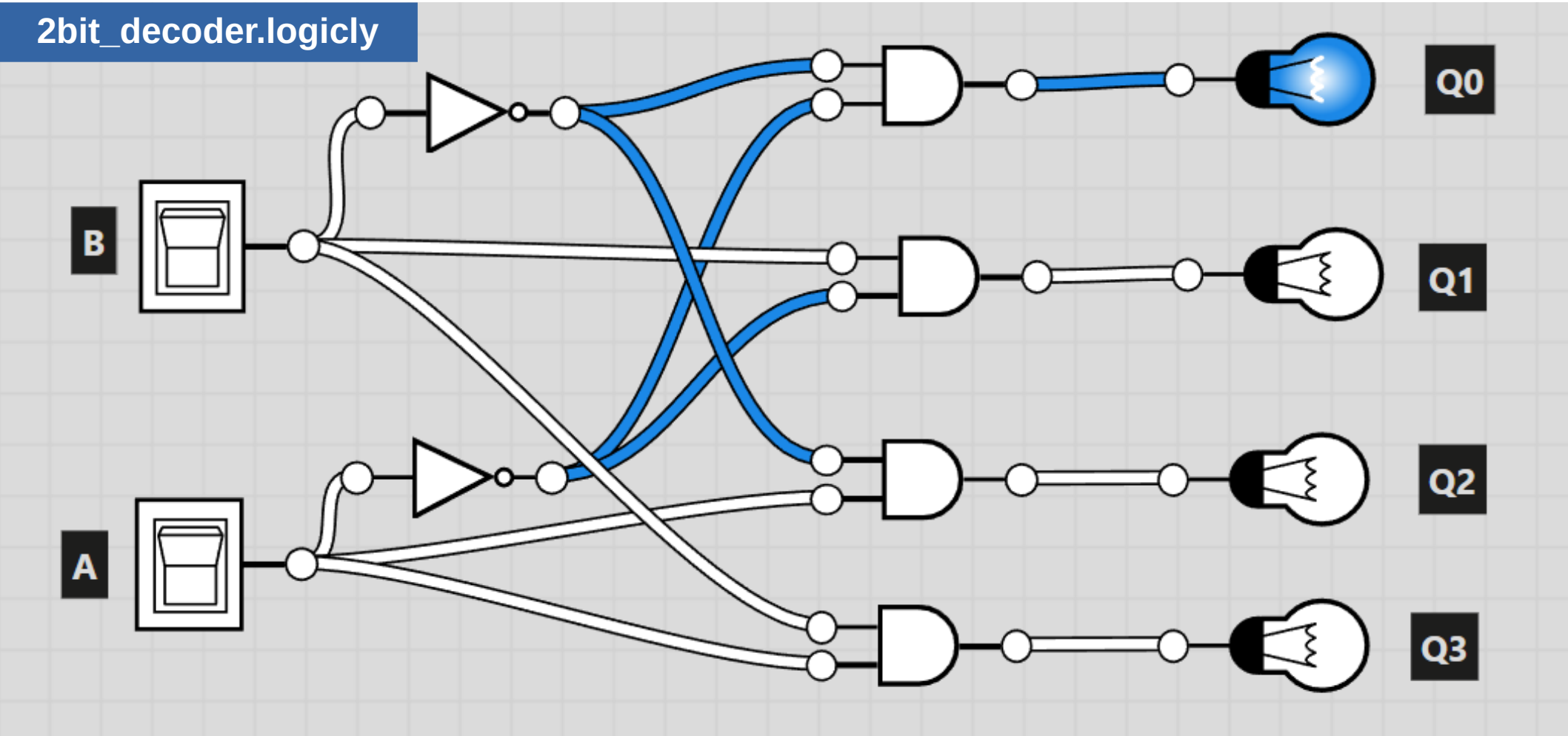




A	B	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

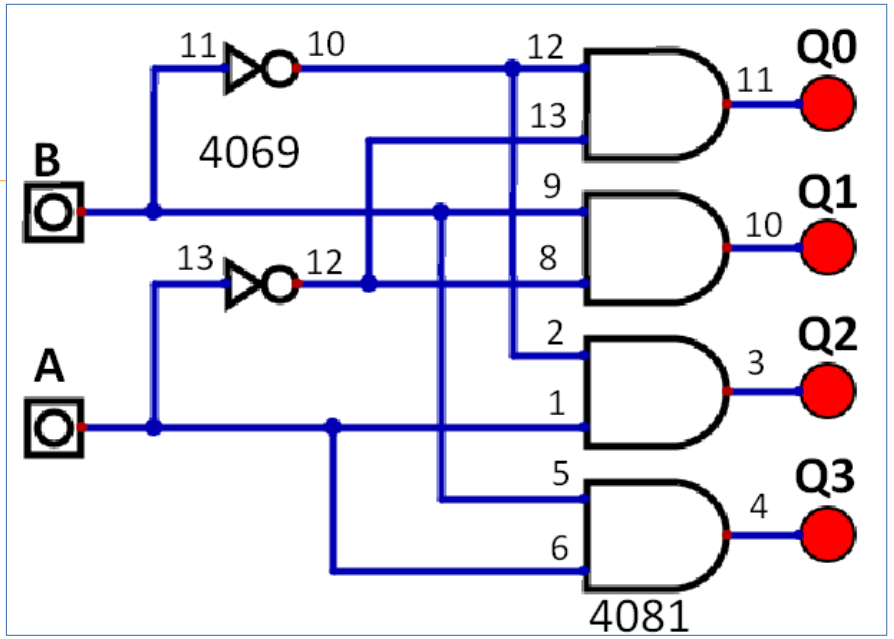
2bit_decoder.logicly

- A 2 bitről négy vonalra dekódóló áramkört először a **Logic.ly** szimulátorban próbáljuk ki
- **B** helyiértéke 1, az **A** helyiértéke pedig 2

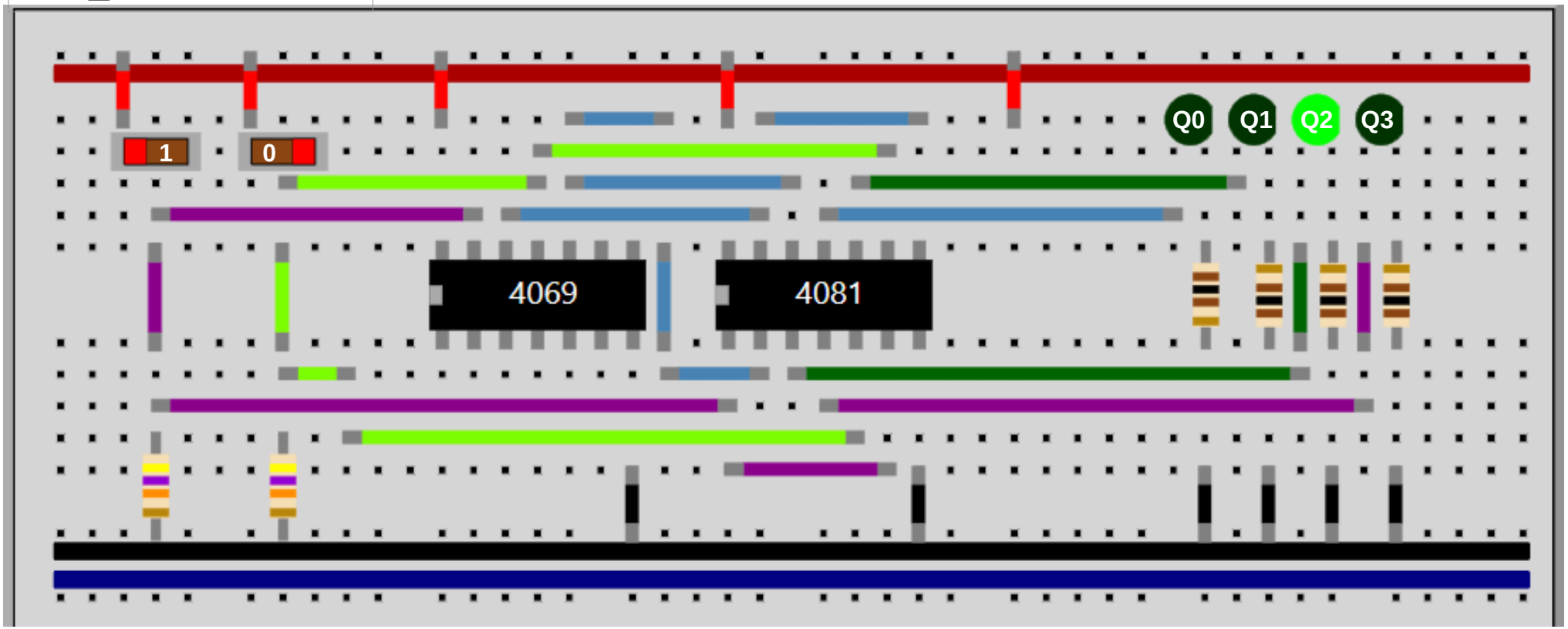


2bit_decoder.bbrd

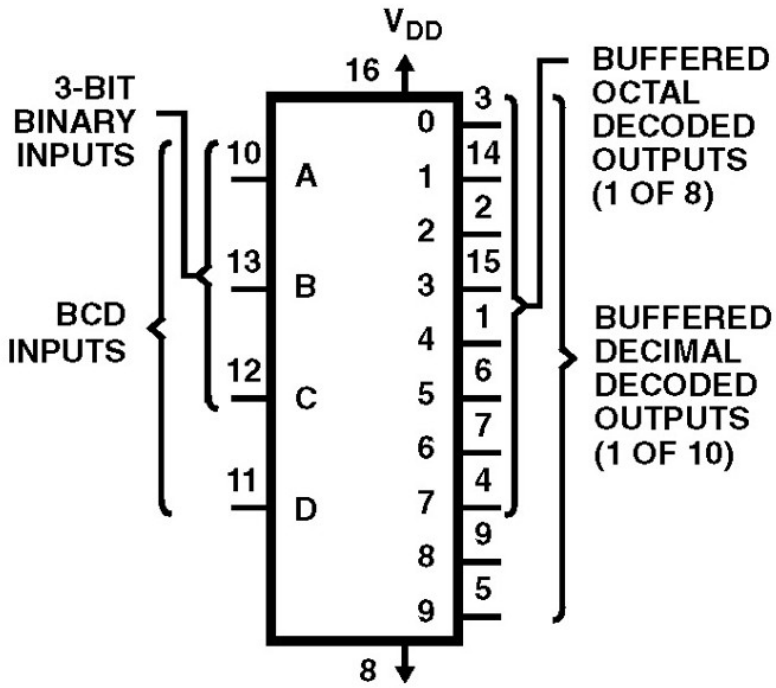
- Az áramkört a Breadboard Simulator-ban is kipróbálhatjuk, vagy megépíthetjük a 30 darabos CMOS IC készletünkből
- A **NEM** kapukat a 4069, az **ÉS** kapukat a 4081 típusú IC tartalmazza

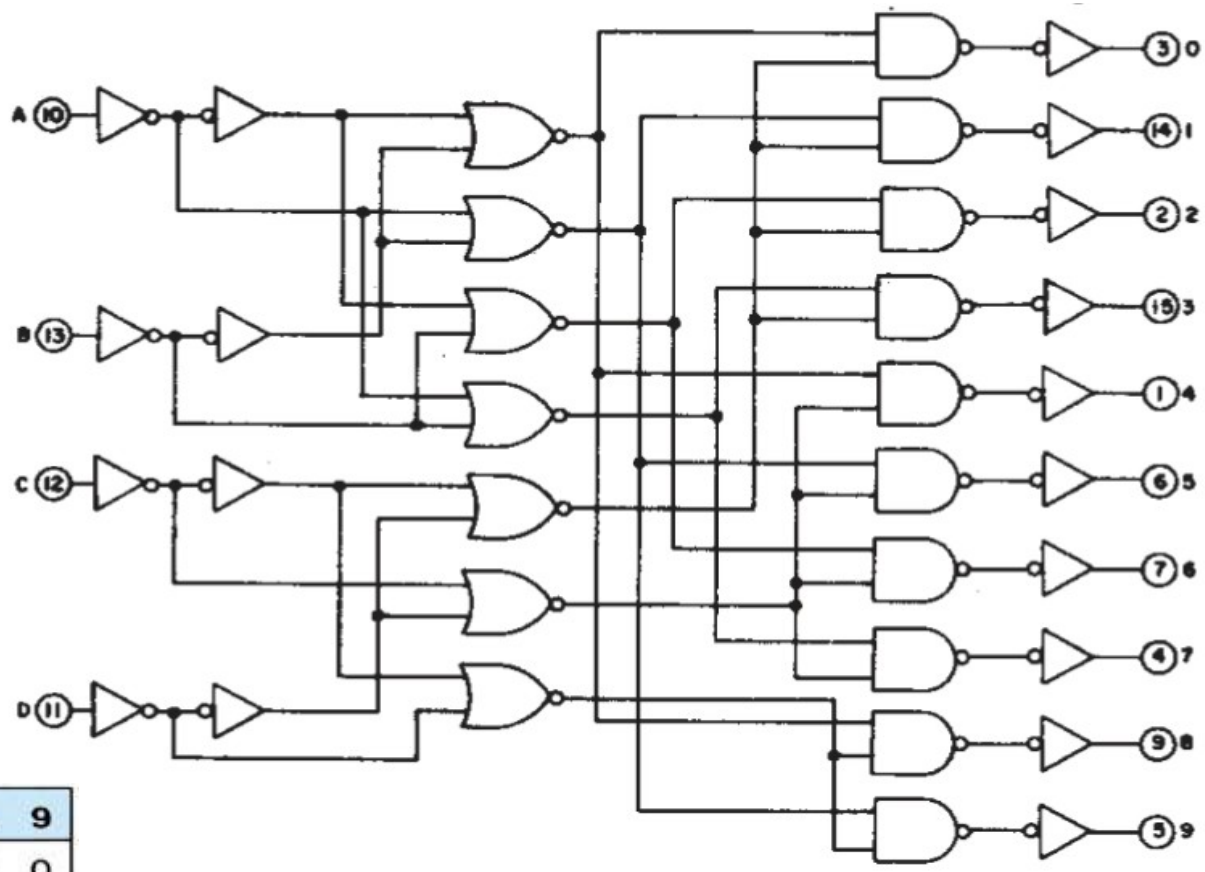


2bit_decoder.bbrd



CD4028: BCD – decimális dekódoló





D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

1 = High Level 0 = Low Level

BCD = Binary Coded Decimal, azaz decimális számjegyenként binárisan kódolt szám (0-9)

Ha csak az A,B,C bemeneteket használjuk, akkor az IC bináris → oktális dekódolóként működik (0-7)

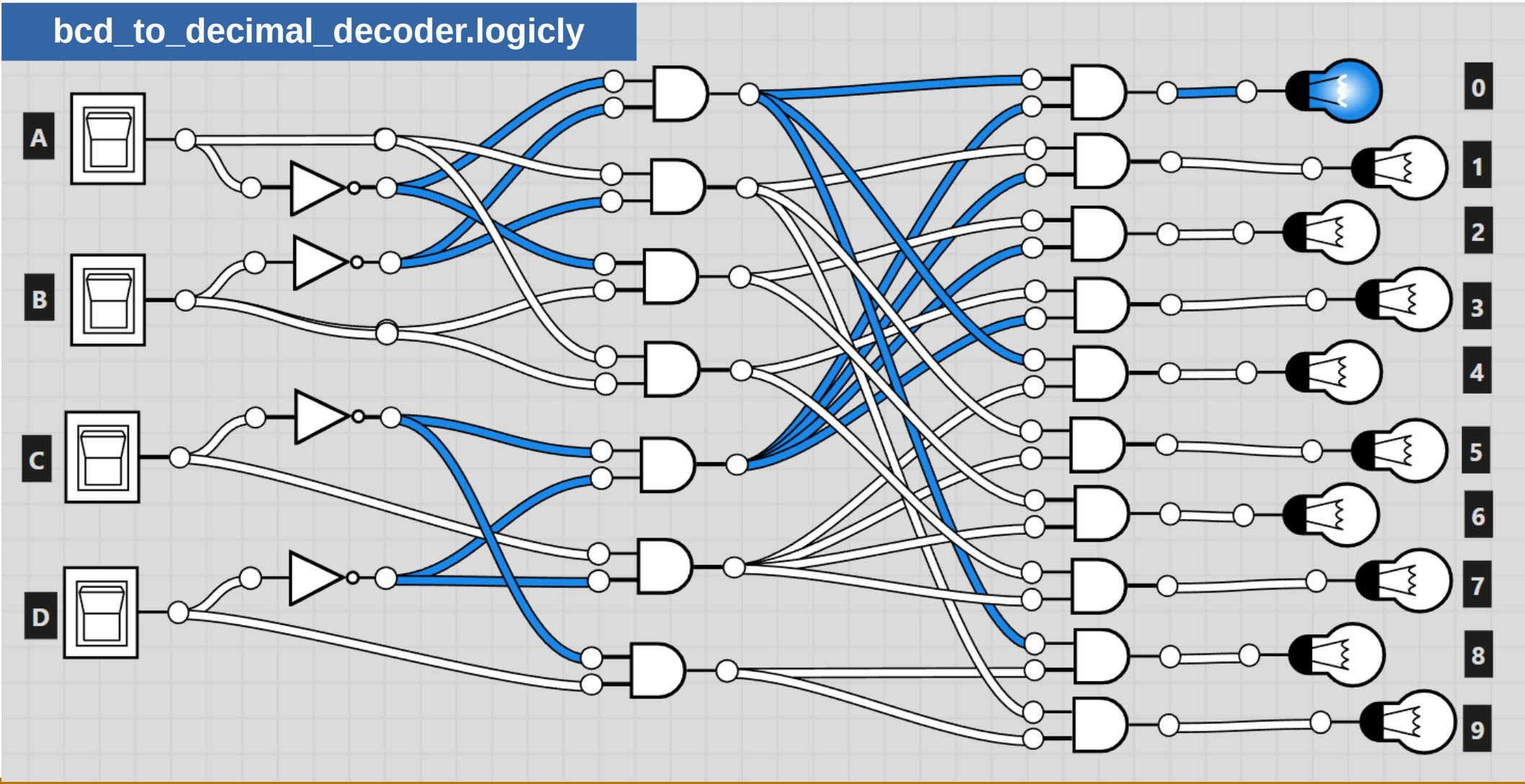
A BCD – decimális dekódoló működése

- A BCD ⟶ decimális dekódoló megvalósításához az ábrán látható **mintermeket** kell képezni
- Egyszerűsítést jelent, ha a négytényezős szorzatokat két, kéttényezős szorzatra bonjuk, mert – az ismétlődések figyelembevételével - ennek megvalósításához csak 4 + 3 db kétbemenetű ÉS kapu kell (a DC szorzatra nincs szükség, mert az csak a 11-nél nagyobb számjegyekhez kellene…)
- Végeredményben a dekóder megépítése csak 4 db invertert, és 4 + 3 + 10 db kétbemenetű ÉS kaput igényel (a 10 db ÉS kapuval a kéttényezős szorzatok szorzatát képezzük)

$$d0 = \overline{D}\cdot\overline{C}\cdot\overline{B}\cdot\overline{A}$$
$$d1 = \overline{D}\cdot\overline{C}\cdot\overline{B}\cdot A$$
$$d2 = \overline{D}\cdot\overline{C}\cdot B\cdot\overline{A}$$
$$d3 = \overline{D}\cdot\overline{C}\cdot B\cdot A$$
$$d4 = \overline{D}\cdot C\cdot\overline{B}\cdot\overline{A}$$
$$d5 = \overline{D}\cdot C\cdot\overline{B}\cdot A$$
$$d6 = \overline{D}\cdot C\cdot B\cdot\overline{A}$$
$$d7 = \overline{D}\cdot C\cdot B\cdot A$$
$$d8 = D\cdot\overline{C}\cdot\overline{B}\cdot\overline{A}$$
$$d9 = D\cdot\overline{C}\cdot\overline{B}\cdot A$$

BCD – decimális dekódoló

- Az előző oldalon vázolt koncepció megvalósítása látható az alábbi ábrán, amelyet a **Logic.ly** programban is kipróbálhatunk



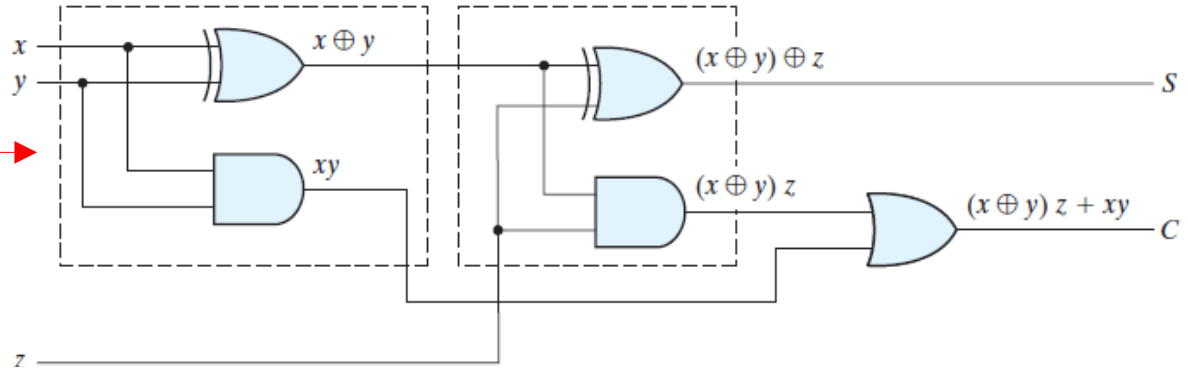
Kombinációs logikai áramkörök - dekódolókkal

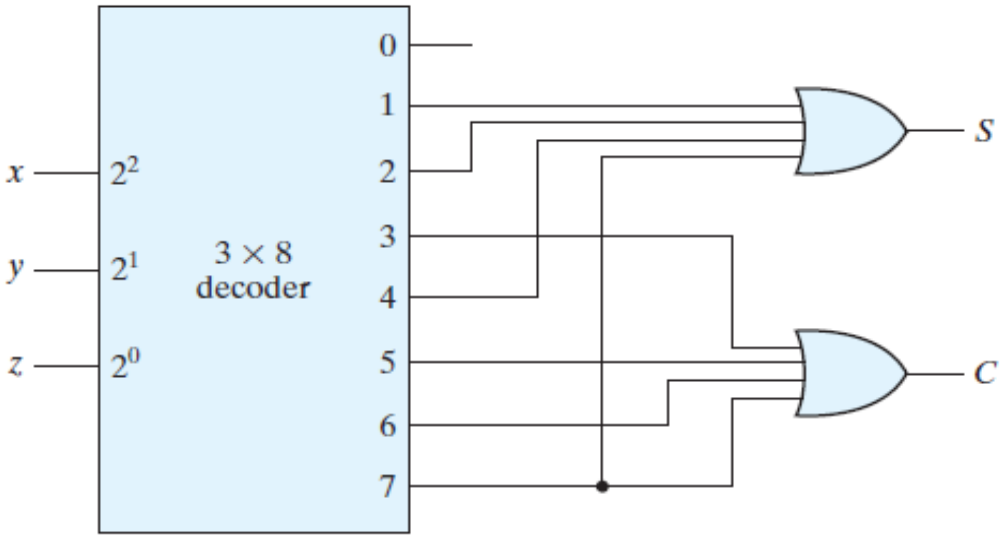
- Minden kombinációs logikai hálózat felírható a bemenetek *ponált* vagy *negált* állapotai szorzatának összegeként (kanonikus alak), a bináris dekódoló pedig éppen ezen szorzatokat állítja elő, így egy **VAGY** kapu hozzáadásával tetszés szerinti kombinációs logikai áramkört építhetünk

- **Példa: teljes összeadó**

Az eredeti kapcsolás

Az új megvalósítás

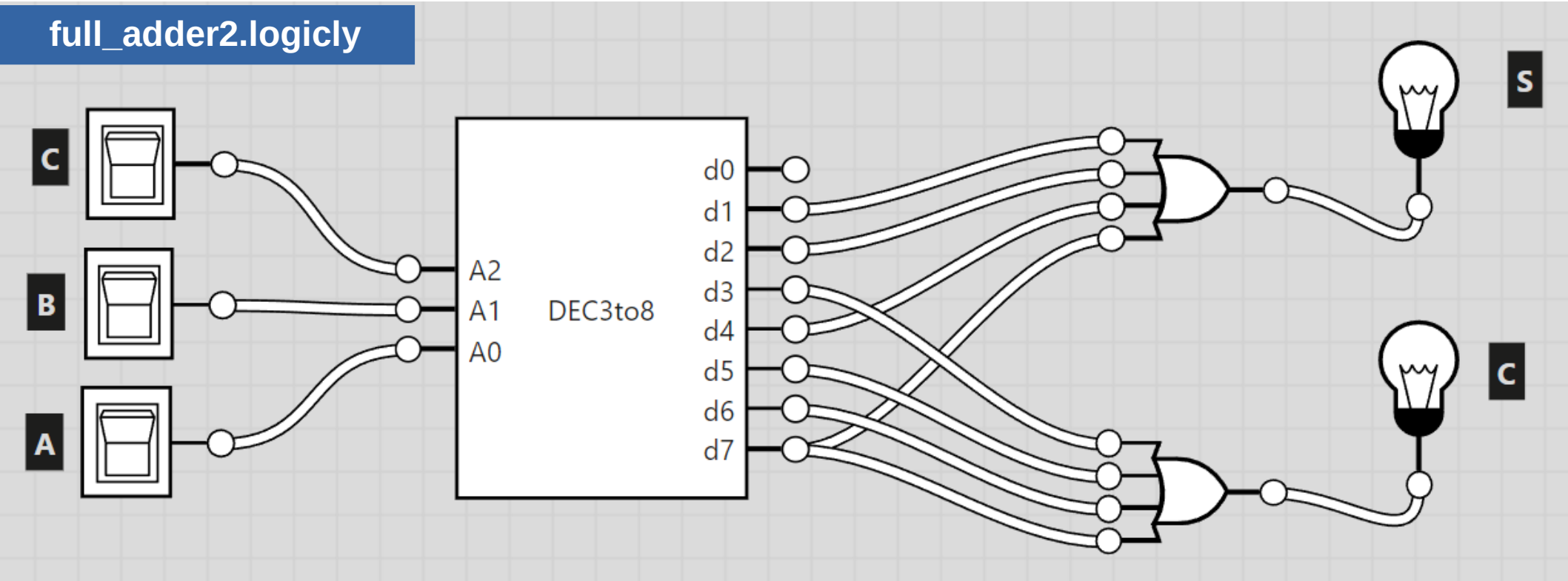




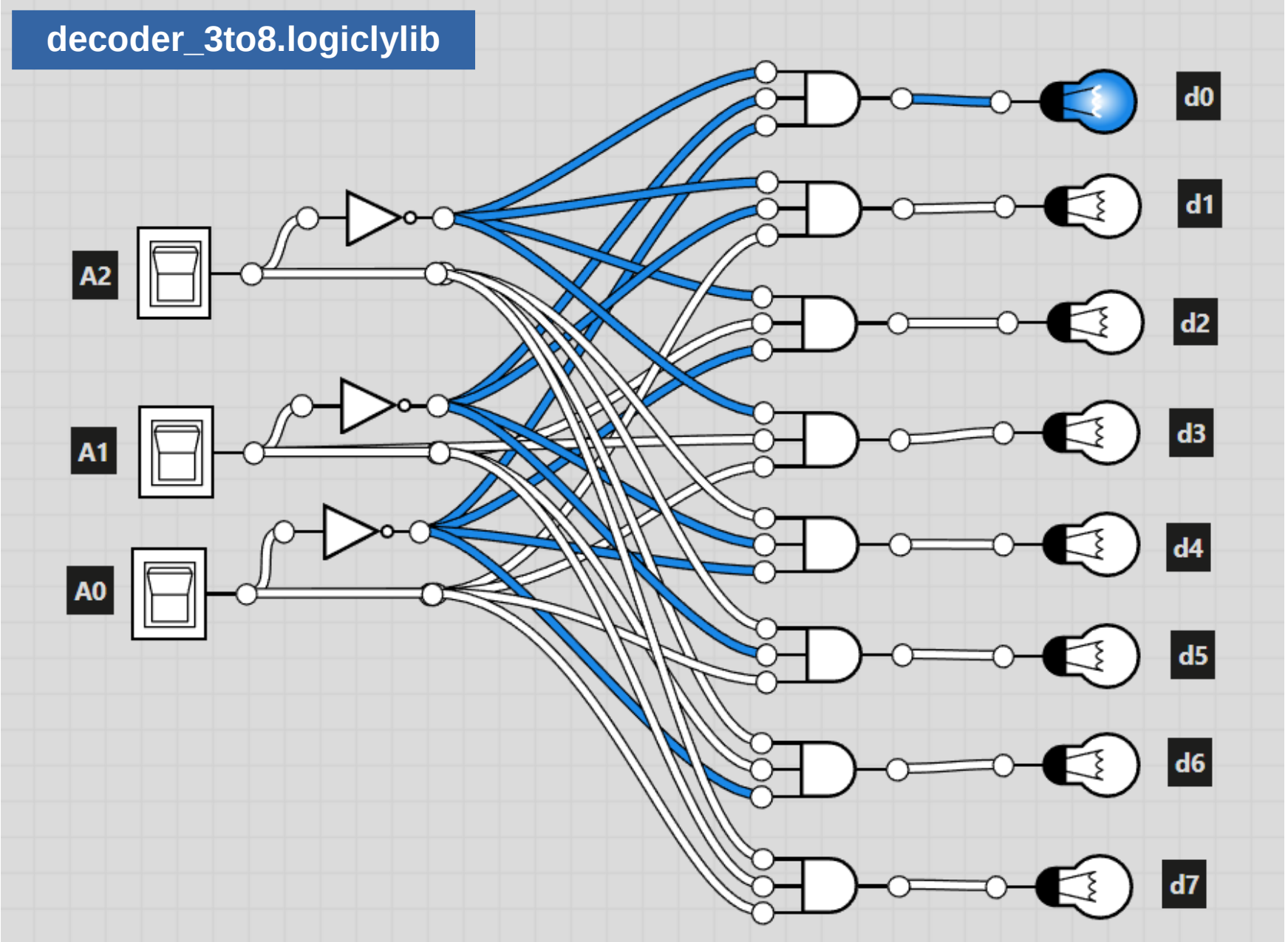
x	y	z	C	S	Sorsz.
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	2
0	1	1	1	0	3
1	0	0	0	1	4
1	0	1	1	0	5
1	1	0	1	0	6
1	1	1	1	1	7

Teljes összeadó, 3-ból 8 dekódolóval

- **A teljes összeadót** a 3 bitről 8 vonalra dekódolóval megvalósító áramkört először a **Logic.ly** szimulátorban próbáljuk ki
- A bemeneteknél **A** és **B** a két összeadandó bit, **C** pedig az áthozat
- A kimeneteknél **S** az összeg (Sum), **C** pedig az átvitel (Carry)
- A dekódoló megvalósítása a következő oldalon látható, de az előző BCD⟶decimális dekódoló is használható, a D = 0 bekötéssel

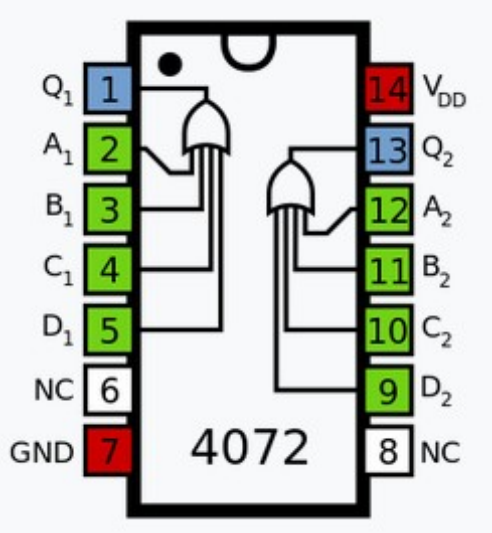


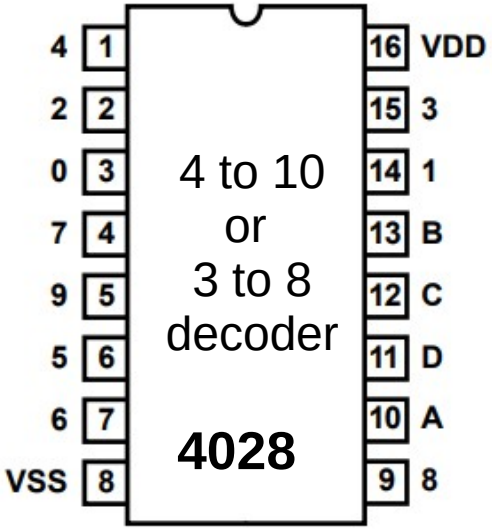
A 3 bitről 8 vonalra dekódoló

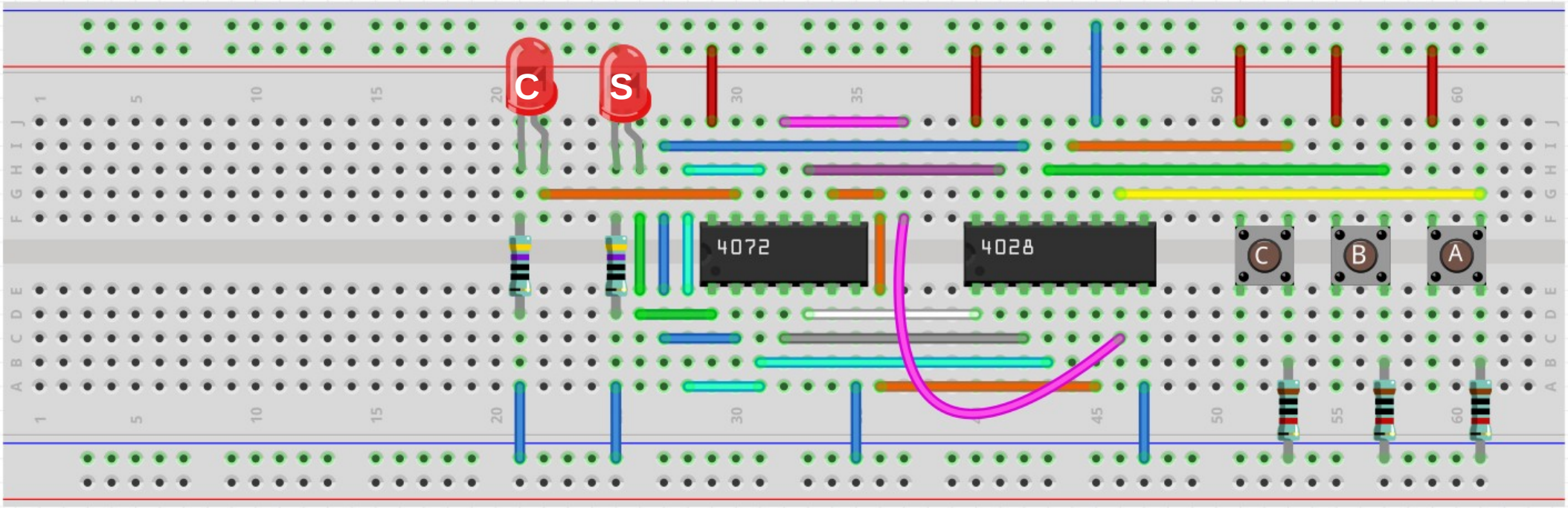


Teljes összeadó, 3-ból 8 dekódolóval

- Az áramkört a 30 darabos CMOS IC készletünkből is megépíthetjük
- A 3-ból 8 vonalra dekódolót a 4028, a **VAGY** kapukat a 4072 típusú IC tartalmazza





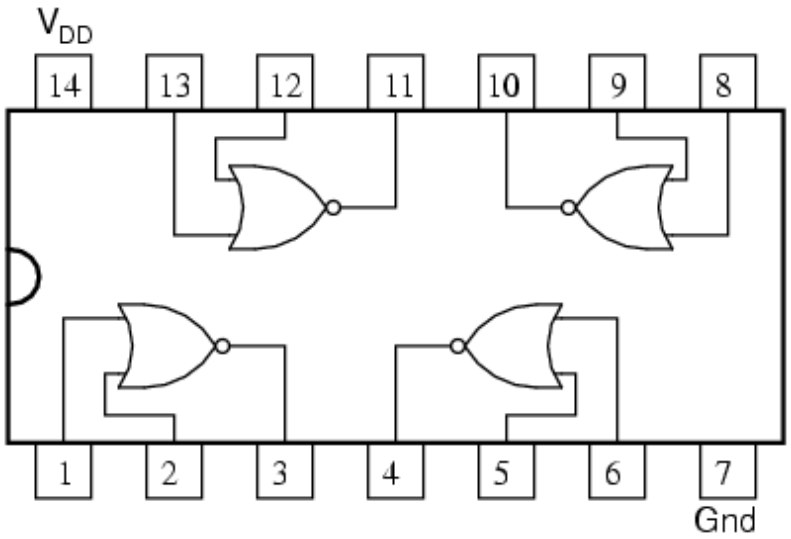


A 4000-es sorozat tipikus tagjai

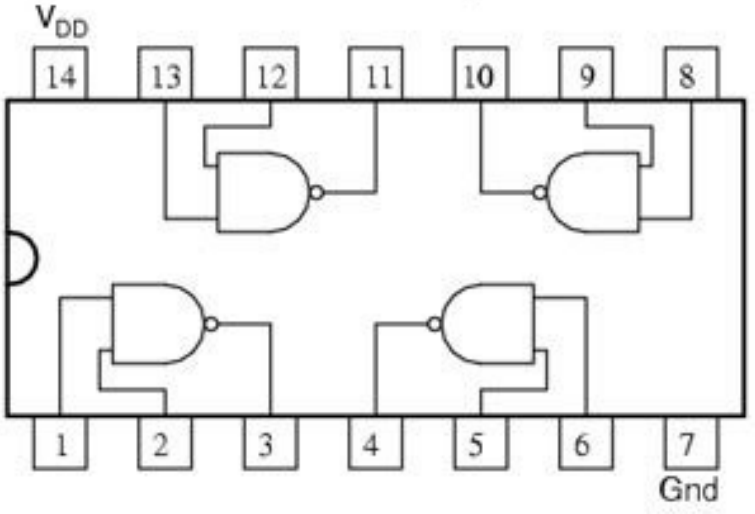
4001 CMOS Quad 2-Input NOR Gate
4011 CMOS Quad 2-Input NAND Gate
4013 CMOS Dual D-Type Flip Flop
4017 CMOS Decade Counter with 10 Decoded Outputs
4021 CMOS 8-Stage Static Shift Register
4022 CMOS Octal Counter with 8 Decoded Outputs
4023 CMOS Triple 3-Input NAND Gate
4025 CMOS Triple 3-Input NOR Gate
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

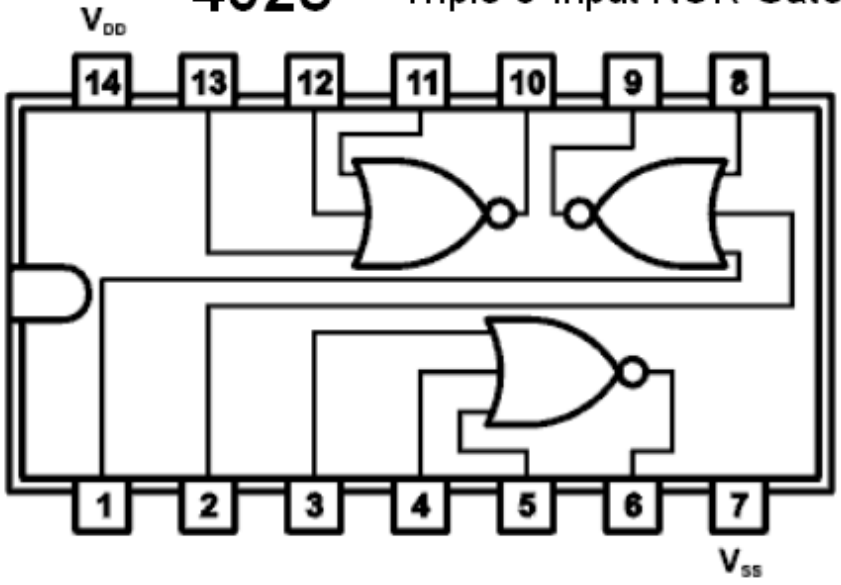
4001 Quad 2-Input NOR Gate



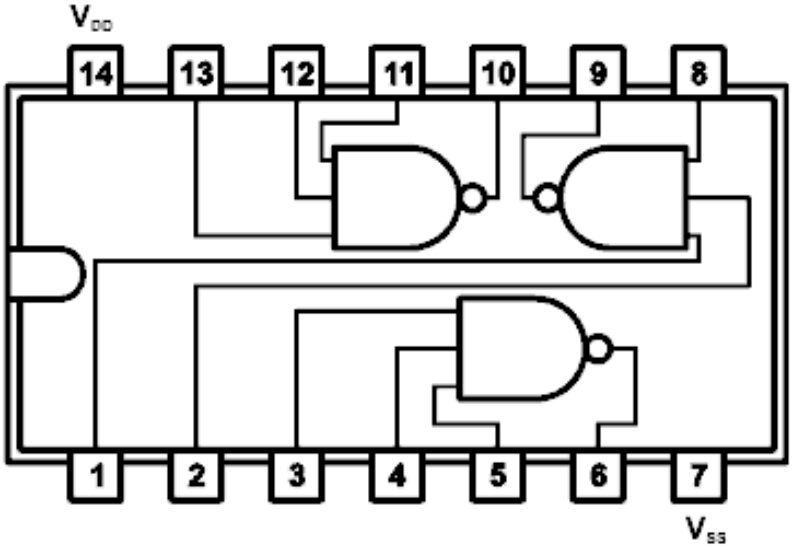
4011 Quad 2-input NAND



4025 Triple 3-Input NOR Gate

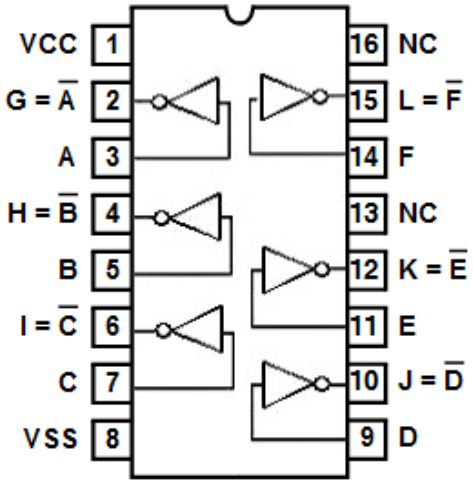


4023 Triple 3-Input NAND Gate

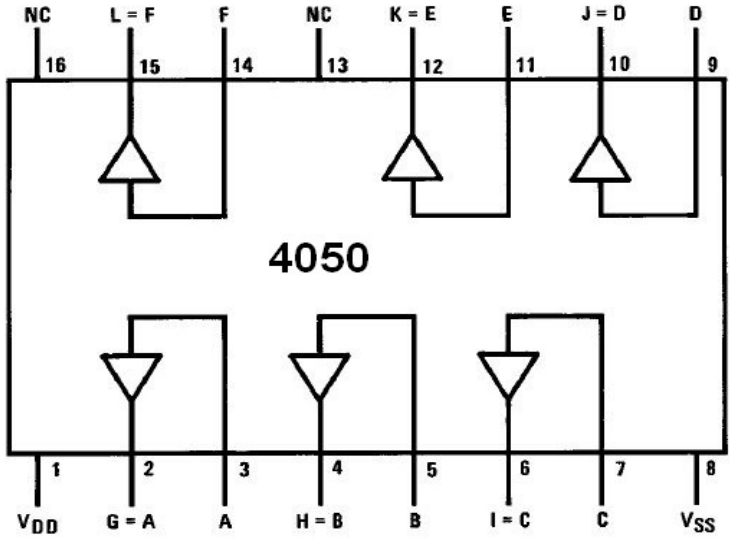


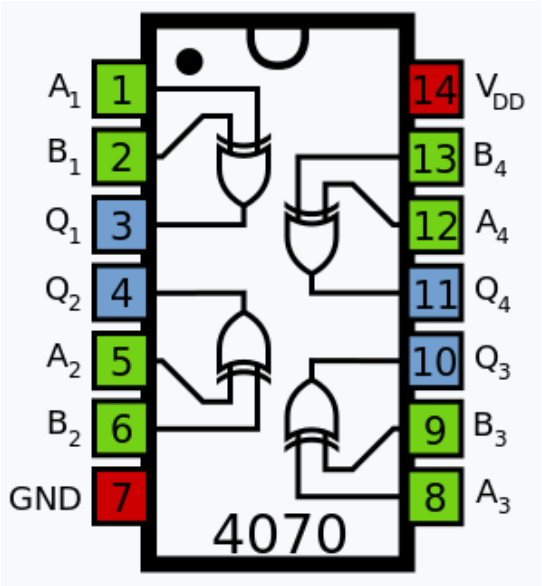
A 4000-es sorozat tipikus tagjai

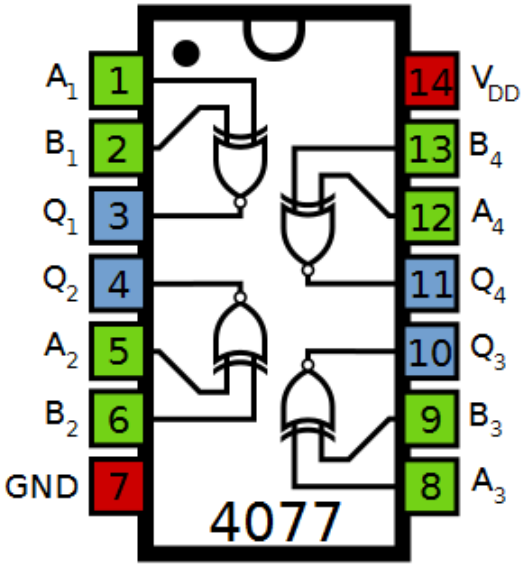
4049
Hex Inverting Buffer/Converter

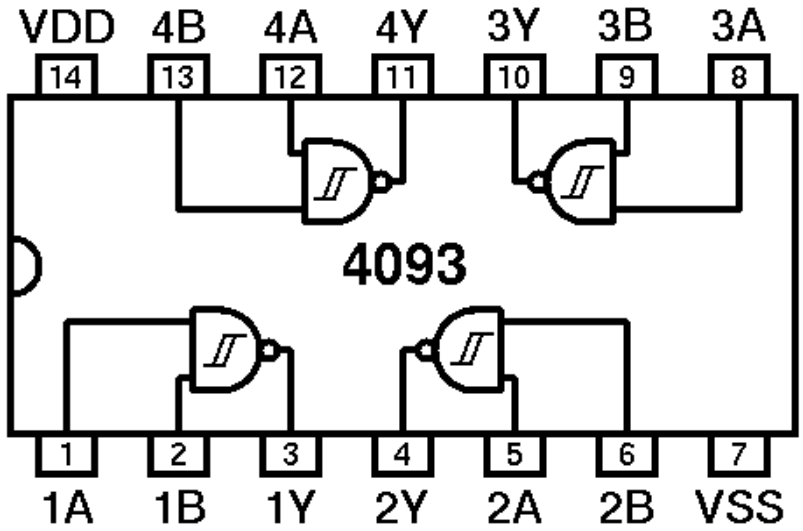


4050 Hex Non-Inverting Buffer/Converter









A 4000-es sorozat tipikus tagjai