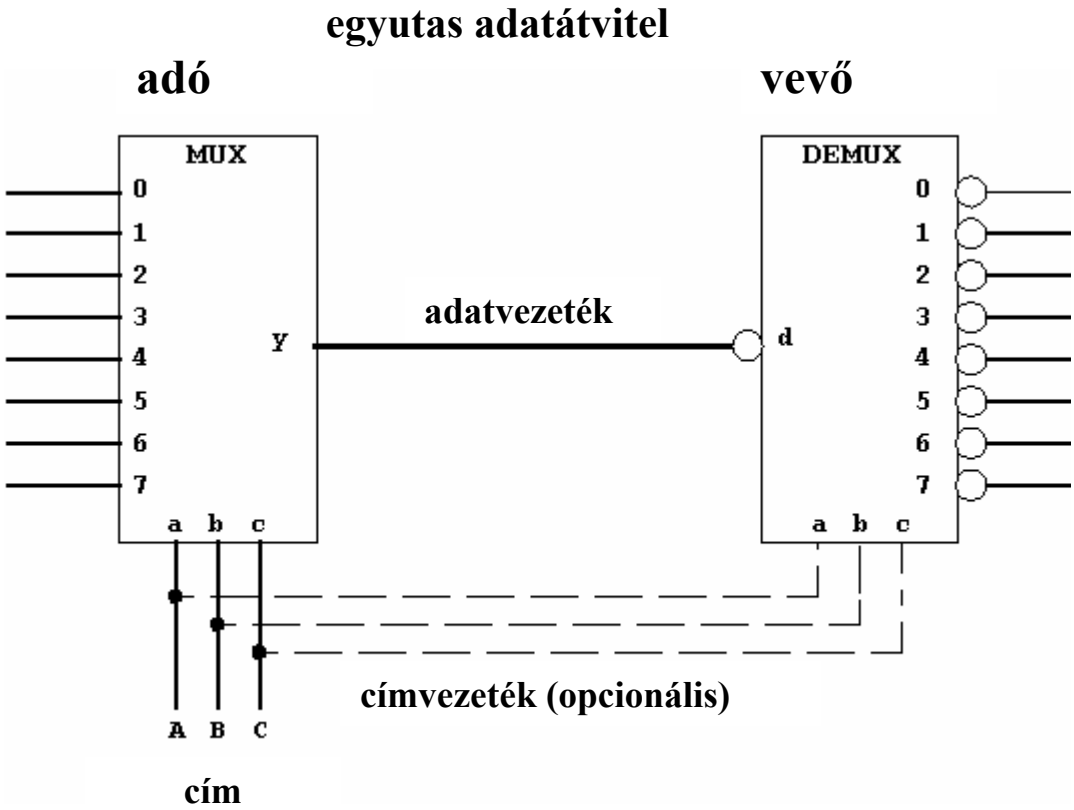
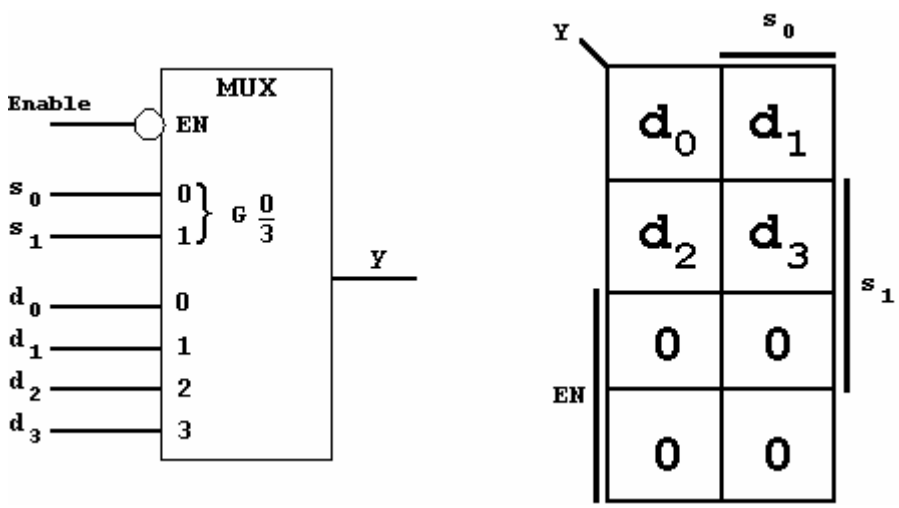
Kombinációs hálózatok
Adatszelektorok, multiplexer

Jellemző példa multiplexer és demultiplexer alkalmazására:



A multiplexer az adóoldali jelvezetékeken jelenlévő jelekből egyet kiválaszt és az egyetlen jeltovábbító vezetékre ad, amelyet a vevőoldalon a demultiplexer „visszaosztja” a vevőoldali megfelelő adatvezetékre.
A cím-információ átadására az adatvezetéket is fel lehet használni, viszont ekkor az adatok továbbítása előtt un. adatátviteli protokollt kell továbbítani.

4:1-multiplexer DIN rajzjele és a működését leíró KV-diagram:



4:1-multiplexer működését leíró logikai függvény:

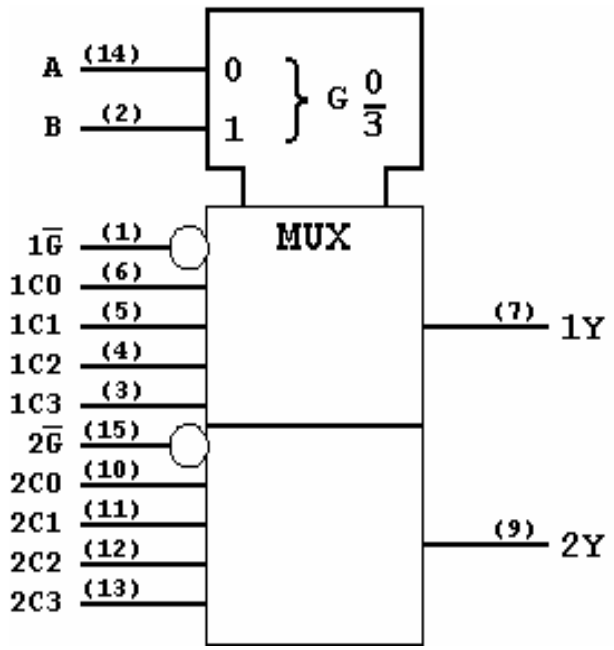
$$Y = \overline{EN} \cdot (\bar{s}_0 \bar{s}_1 d_0 + s_0 \bar{s}_1 d_1 + \bar{s}_0 s_1 d_2 + s_0 s_1 d_3)$$

Az *enable* bemenet több multiplexer összekapcsolását segíti elő. („*active low*”)

Példák multiplexerekre:

74LS153 kétszeres 4:1 multiplexer rajzjele (DIN, IEC, …)

(*dual 4-line to 1-line data selectors*)



74LS604 8 bites adat-átváltó-kapcsoló buszrendszerekhez „tristate”-kimenettel (puffer)

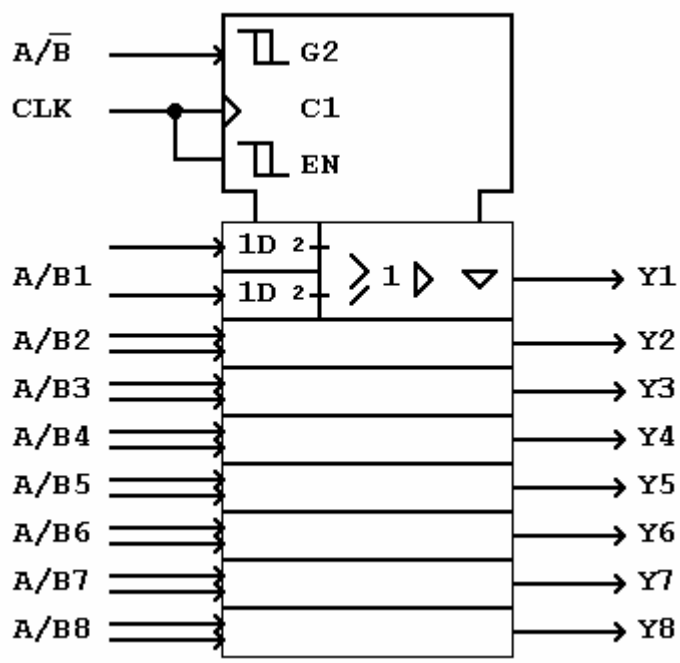
(*octal 2-input multiplexed latches*)

Adatok átvétele az ütemjel CLK („clock”) felfutó élével. Adatok tárolása és lekérdezhetősége a kimeneten CLK=„1” ideje alatt.


„Schmitt-trigger”-bemenet


élvezérelt


„tristate”-kimenet



Kombinációs hálózatok
Adatszelektorok, multiplexer

TTL-Multiplexer típusok:

Egy IC-be integrált multiplexerek száma

Kód	Típus	#	Kimenetek
74150	$16 \rightarrow 1$	1	Totem Pole
74151	$8 \rightarrow 1$	1	Totem Pole
74157/258	$2 \rightarrow 1$	4	Tristate
74251	$8 \rightarrow 1$	1	Tristate
74253	$4 \rightarrow 1$	2	Tristate
74257/258	$2 \rightarrow 1$	4	Tristate

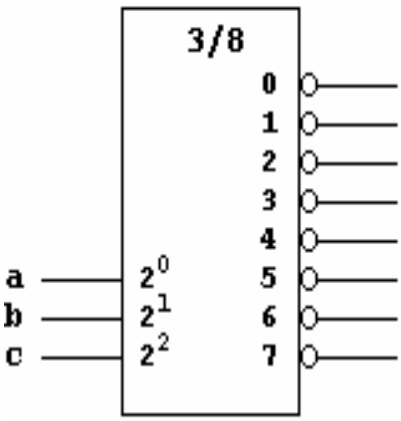
Kombinációs hálózatok
Adatszelektorok, dekódoló/demultiplexer

A demultiplexer a multiplexer feladatával ellentétes, inverz funkciót lát el. Rokon a dekódolóval, mely felépítésében egyszerűbb.

Példa-dekódoló működését bemutató igazságtáblázat:
(2^n db kimeneti jelek egyikét aktivizálja n bit szélességű bemeneti kód alapján itt n=3)

Bináris kód			**"8-ból 1" kód**							
c	**b**	**a**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Gyakoribb dekódoló-megoldás az „*active low*” kimenet:



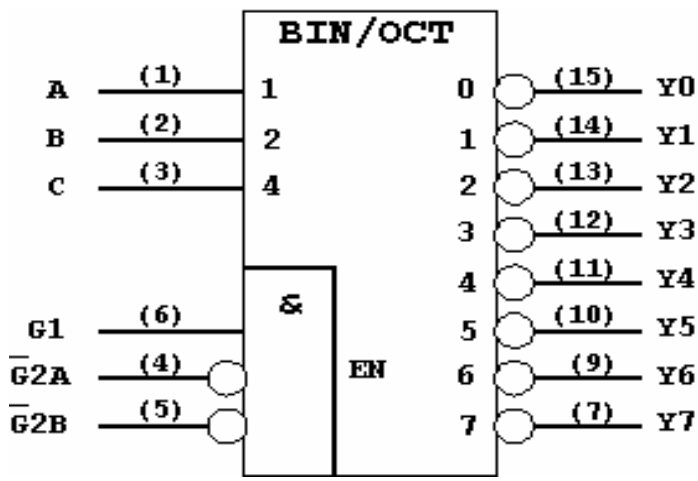
Kapcsolási jel

Bináris kód			"8-ból 1" kód							
c	b	a	0	1	2	3	4	5	6	7
0	0	0	**0**	1	1	1	1	1	1	1
0	0	1	1	**0**	1	1	1	1	1	1
0	1	0	1	1	**0**	1	1	1	1	1
0	1	1	1	1	1	**0**	1	1	1	1
1	0	0	1	1	1	1	**0**	1	1	1
1	0	1	1	1	1	1	1	**0**	1	1
1	1	0	1	1	1	1	1	1	**0**	1
1	1	1	1	1	1	1	1	1	1	**0**

Példák dekódolóra:

Dekódolókat használnak különböző eszközök kiválasztására, ezért többnyire bonyolult engedélyező (*enable*) bemeneti kombinációkkal rendelkeznek.

74LS138



Engedélyezve a dekódoló, ha

$G1 = \text{"1"}$ és

$\overline{G2A} = \overline{G2B} = \text{"0"}$

Ha a fenti feltétel nem tejesül, az áramkör passzív állapotba megy át, azaz a dekódoló-funkció nem működik és emiatt az összes kimenet „1”-értéket vesz fel.

A 74 LS138 dekódoló igazságtáblázata:

Enable-bemenet			**Cím**			**Kimenetek**							
G1	$\overline{G2A}$	$\overline{G2B}$	**c**	**b**	**a**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
0	x	x	x	x	x	1	1	1	1	1	1	1	1
1	**0**	**0**	**0**	**0**	**0**	**0**	1	1	1	1	1	1	1
1	**0**	**0**	**0**	**0**	**1**	1	**0**	1	1	1	1	1	1
1	**0**	**0**	**0**	**1**	**0**	1	1	**0**	1	1	1	1	1
1	**0**	**0**	**0**	**1**	**1**	1	1	1	**0**	1	1	1	1
1	**0**	**0**	**1**	**0**	**0**	1	1	1	1	**0**	1	1	1
1	**0**	**0**	**1**	**0**	**1**	1	1	1	1	1	**0**	1	1
1	**0**	**0**	**1**	**1**	**0**	1	1	1	1	1	1	**0**	1
1	**0**	**0**	**1**	**1**	**1**	1	1	1	1	1	1	1	**0**

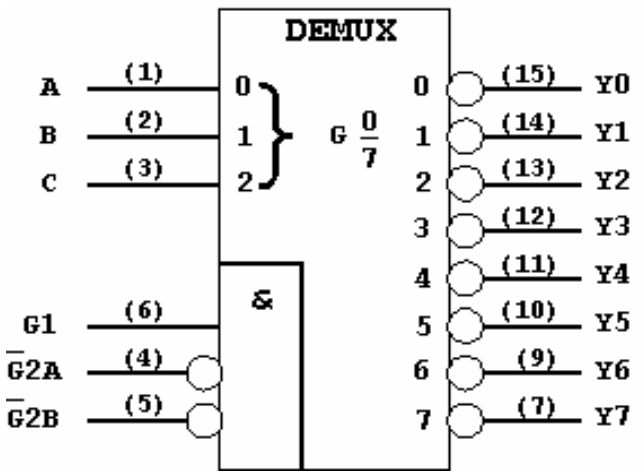
Kombinációs hálózatok
Adatszelektorok, demultiplexer

A demultiplexer feladata az egy jelvezetéken beérkező információ „szétosztása” a max. 2^n számú kimeneti jelvezetékek egyikére.

- A dekódoló egyben demultiplexer is, amennyiben a legnagyobb helyiértékű címbemenet-bitet kinevezzük adat-bemenetnek.

d	**cím**		**kimenetek**				kihasználatlan kimenetek			
c	**b**	**a**	**0**	**1**	**2**	**3**	4	5	6	7
0	**0**	**0**	***0***	**1**	**1**	**1**	1	1	1	1
0	**0**	**1**	**1**	***0***	**1**	**1**	1	1	1	1
0	**1**	**0**	**1**	**1**	***0***	**1**	1	1	1	1
0	**1**	**1**	**1**	**1**	**1**	***0***	1	1	1	1
1	**0**	**0**	***1***	**1**	**1**	**1**	0	1	1	1
1	**0**	**1**	**1**	***1***	**1**	**1**	1	0	1	1
1	**1**	**0**	**1**	**1**	***1***	**1**	1	1	0	1
1	**1**	**1**	**1**	**1**	**1**	***1***	1	1	1	0

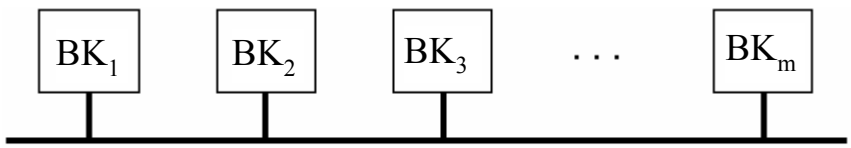
- Dekódoló multiplexerként történő felhasználására másik lehetőség, amikor az engedélyező bemeneteket használjuk ki:
  Például, ha a fenti példában a $\overline{G}2A$ engedélyező bemenetet adatbemenetként használjuk, a kiválasztott Y_i kimeneten a $\overline{G}2A$ aktuális értéke jelenik meg.



Kombinációs hálózatok
Adatszelektorok, demultiplexer

Sok alkalmazásnál (pl. mikroprocesszoros rendszereknél) szükséges az egyes egységeket „címzéssel” ellátni.

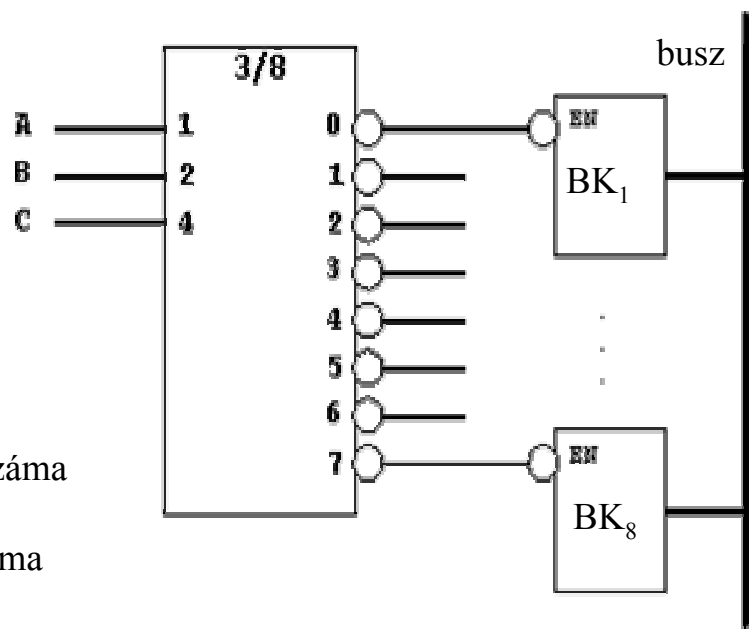
Demultiplexerek alkalmazására tipikus példa buszrendszerre csatlakozó eszközök (tárolóegység, periféria-egység, stb.)kiválasztása, pl. adatátvitel céljából.
A buszrendszerre csatlakozás feltétele, hogy a rácsatlakozó eszközök közül adott időpontban mindig csak az egyik érvényesüljön és a többi ne tudja befolyásolni a buszvezeték állapotát. Ehhez például az összes eszköz „tristate”-kimenettel kell, hogy rendelkezzen.



Közös kommunikációs csatorna (busz)

BK: Bemeneti-/Kimeneti perifériaegység

Dekódoló/demultiplexer alkalmazása:

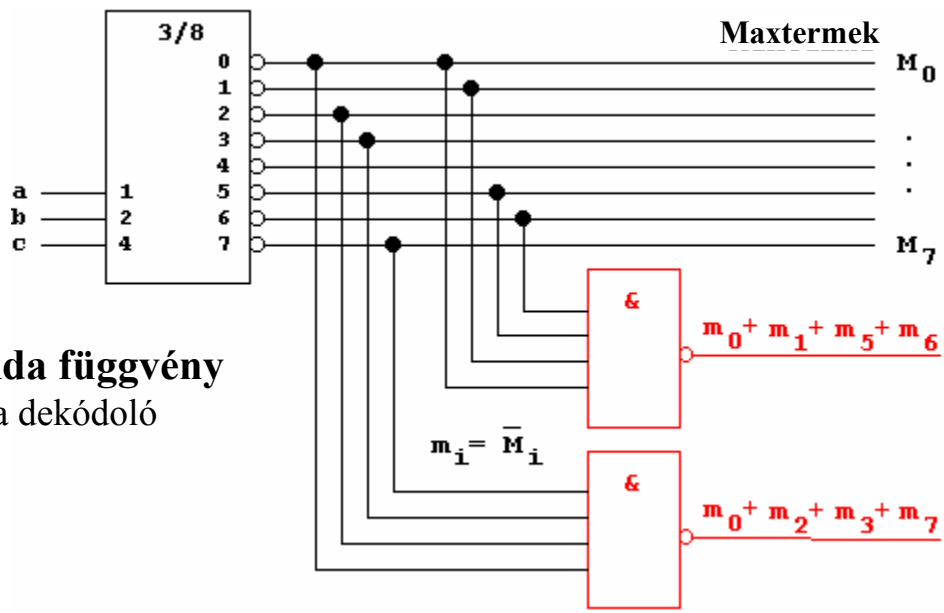


$m \leq 2^n$

m: a BK-egységek száma
n: a dekódoló cím-bemeneteinek száma

Dekódoló alkalmas univerzális logikai áramkörként is.
A 3/8-as dekódoló igazságtáblázatából látható, hogy mindegyik kimenet egy-egy maxterm-függvényt (M_0-M_7) valósít meg. (Neminvertáló kimenetek esetében mintermeket: m_0-m_7)

c	b	a	0	1	2	3	4	5	6	7
0	0	0	**0**	1	1	1	1	1	1	1
0	0	1	1	**0**	1	1	1	1	1	1
0	1	0	1	1	**0**	1	1	1	1	1
0	1	1	1	1	1	**0**	1	1	1	1
1	0	0	1	1	1	1	**0**	1	1	1
1	0	1	1	1	1	1	1	**0**	1	1
1	1	0	1	1	1	1	1	1	**0**	1
1	1	1	1	1	1	1	1	1	1	**0**



Logikai példa függvény megvalósítása dekódoló segítségével:

Az egyes maxterm-függvények NAND-összekapcsolásával előálló függvény algebrai alakja (kanonikus diszjunktív normálalak):

$$m_0 + m_1 + m_5 + m_6 = \overline{M_0 \cdot M_1 \cdot M_5 \cdot M_6}$$

Fentieknek megfelelően a neminvertáló kimenetekkel rendelkező dekódoló (*active high*) előállítja az összes mintermet, m_i-t, melyek OR művelettel történő összekapcsolása a függvény kanonikus konjunktív normálalakját adja.

Logikai függvények megvalósítási módszerei:

- Műszaki feladatok kapcsolástechnikai megoldására egyik módszer a probléma algebrai megfogalmazása és ennek logikai kapuáramkörökkel történő megvalósítása.
- Könnyebb utólagos módosításokat, továbbfejlesztéseket, teszteket elvégezni a másik módszernél: a bemenő adatokon végzett logikai műveletek sorrendjét algoritmizáljuk és megfelelő mikroprocesszoros rendszerre írt **programmal** valósítjuk meg a kívánt függvényt. (program→ROM, köztes eredmények→RAM) ⇒ *programmable logic device (PLD)*
- Harmadik módszer, amikor az algebrailag, vagy az ezzel ekvivalens igazság-táblázattal megadott függvényt egyszerűen a táblázat memóriában történő elhelyezésével valósítjuk meg.

Példa: BCD-kód → hétszegmenses kijelző-kód

Bemenetek				Kimenetek						
Címek				Memória-tartalom						
d	c	b	a	A	B	C	D	E	F	G
0	*0*	*0*	*0*	1	1	1	1	*1*	1	0
0	0	0	1	0	1	1	0	*0*	0	0
0	*0*	*1*	*0*	1	1	0	1	*1*	0	1
0	0	1	1	1	1	1	1	*0*	0	1
0	1	0	0	0	1	1	0	*0*	1	1
0	1	0	1	1	0	1	1	*0*	1	1
0	*1*	*1*	*0*	1	0	1	1	*1*	1	1
0	1	1	1	1	1	1	0	*0*	0	0
1	*0*	*0*	*0*	1	1	1	1	*1*	1	1
1	0	0	1	1	1	1	1	*0*	1	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

BCD-kód nem teljes kód → 10-15 tartom. „*don't care*", itt „0"-k

Kombinációs hálózatok
Adatszelektorok, táblázat → (P)ROM

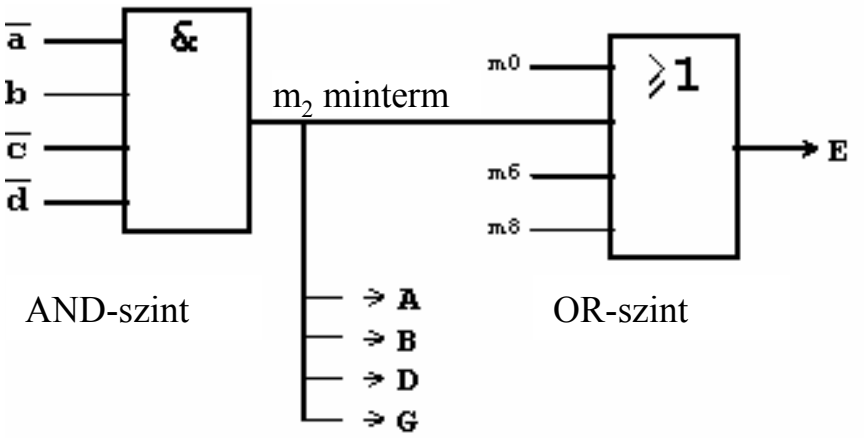
A BCD-kód → hétszegmenses-kód átalakító memóriás megvalósításához 16*7 bit tárolókapacitás szükséges (teljes táblázat).

A hét Boole-függvény [A(a,b,c,d); B(a,b,c,d);….;G(a,b,c,d)]megvalósítása történhet a megfelelő mintermek OR-összekapcsolásával, melyeket a bemeneti változók AND kapus dekódolásával kapunk meg.

Példa: minterm-dekódolás módszere

m_2 minterm és az E(a,b,c,d) függvény megvalósítása AND és OR kapukkal

4/16 dekódoló AND kapukból



Logikai függvények tárolós megvalósításainak előnyei:

- a kapcsolások gyorsan megvalósíthatóak (prototípusoknál fontos)
- olcsó

hátrányai:

- a tárolt adatokhoz a hozzáférés (és ezzel a függvényérték előállításának) ideje hosszú
- mivel a tárolókapacitás előre meghatározott, gyakran a nem igényelt információkat is definiálni kell - ***„don't care"*-állapotok nem lehetségesek**
- ebben a táblázatos megoldásban az azonos tartalmú szomszédos tárolóelemek (cellák) nem vonhatók össze- **implikánsokat nem lehet képezni**

A **„programozható logika”** a tárolós megvalósítással ellentétben a minterm-dekódolás hátrányait kiküszöböli:

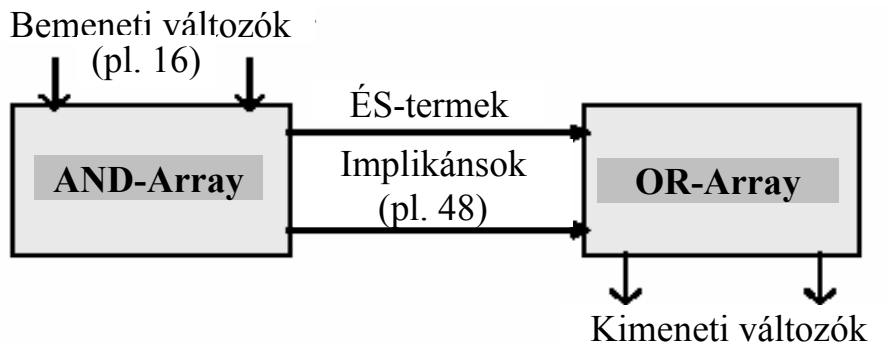
implikánsok képzése és a „*don’t care*” esetek figyelembevétele is lehetséges.

A PLD (programmable logic devices) összefoglaló néven emlegetett áramkör-típusok három csoportra oszthatók:

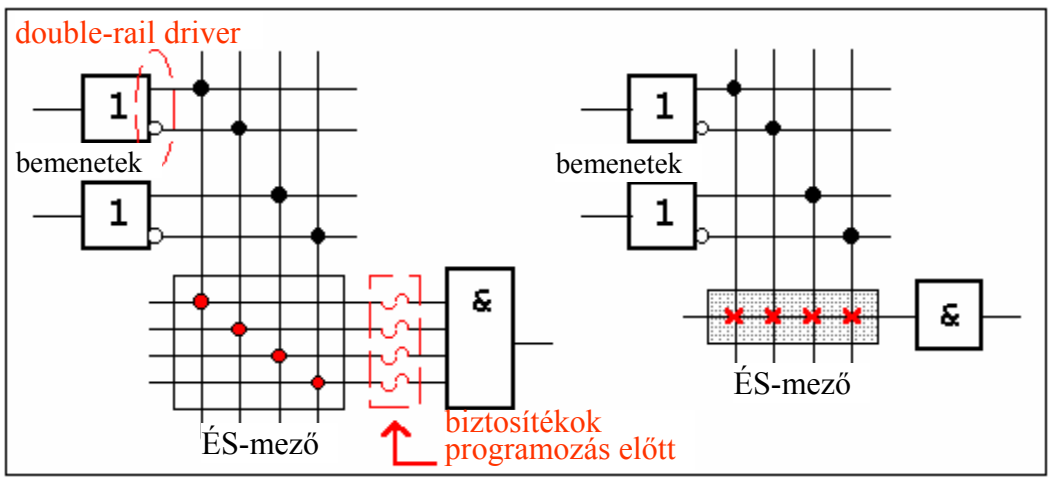
- PLA: *Programmable Logic Array* (FPLA: Fieldprogrammable Array Logic)
- PAL: *Programmable Array Logic*
- PROM: *Programmable Read Only Memory* (PLE: Programmable Logic Element)

Programmable Logic Array

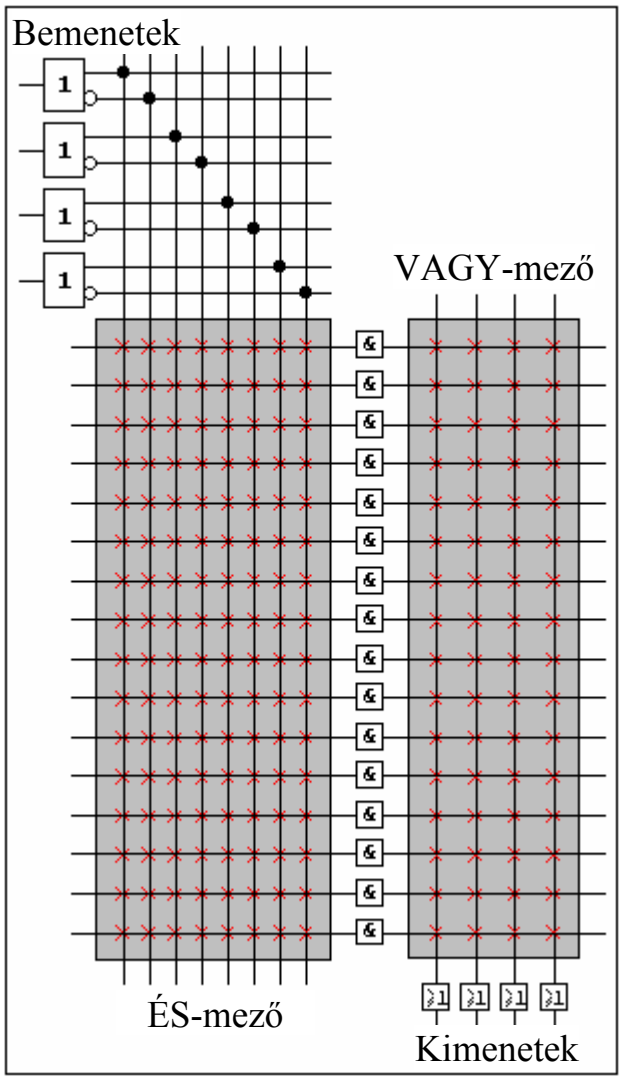
A már ismert AND-OR felépítést annyiban egészíti ki, hogy az előre megadott AND-Array-ban nem csak mintermek, hanem összevont cella-tartományok képzése is lehetséges



PLD „programozása”:



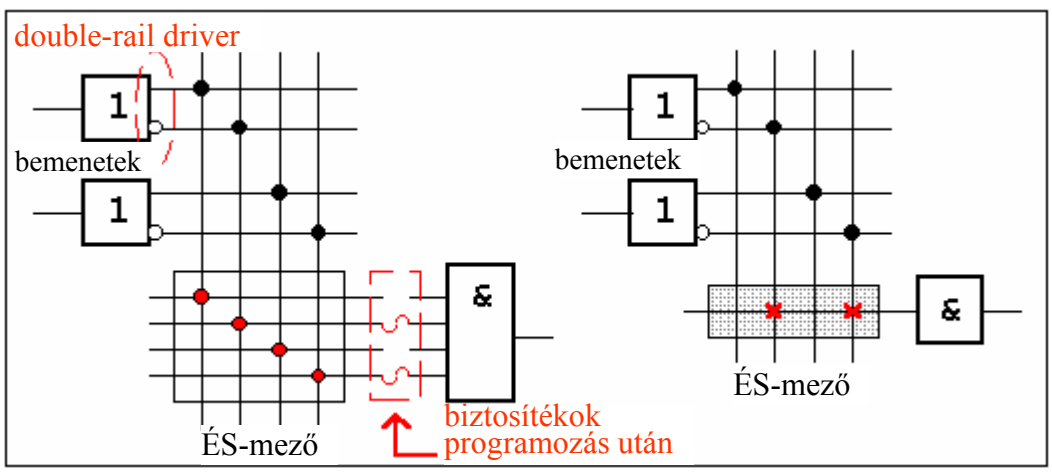
Programmable Logic Array



Adott a bemeneti AND-struktúra és a kimeneti OR-struktúra, az összeköttetések száma és miként je **mindkét szinten teljes mértékben** programozható (PLA, FPLA).

PLD „programozása”:

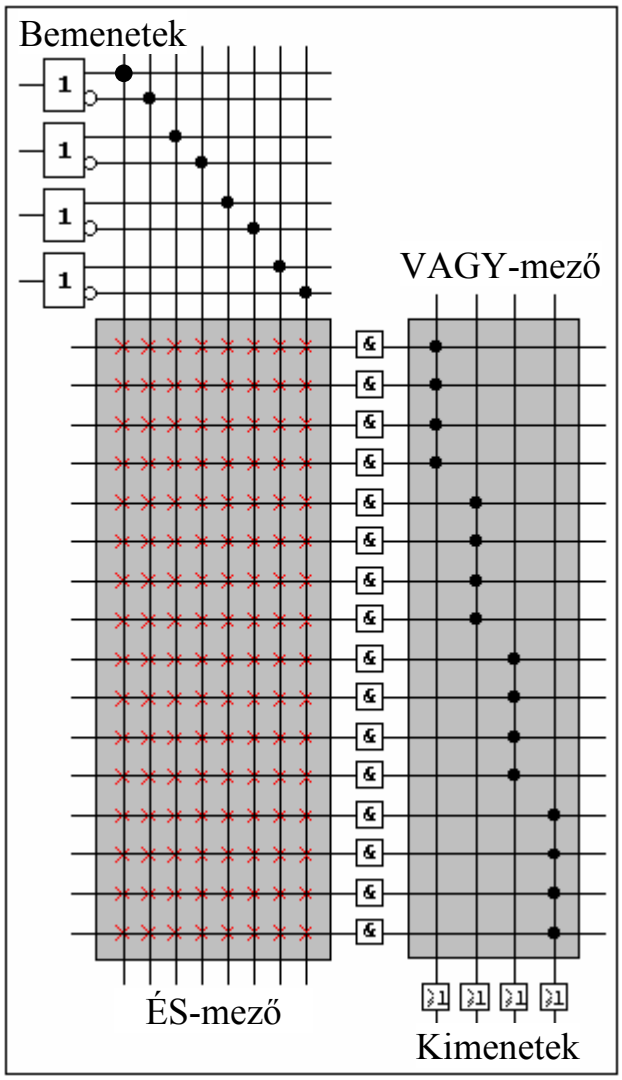
összekötések: *cross point switches*
oldható „biztosítékok”: *fusable links*

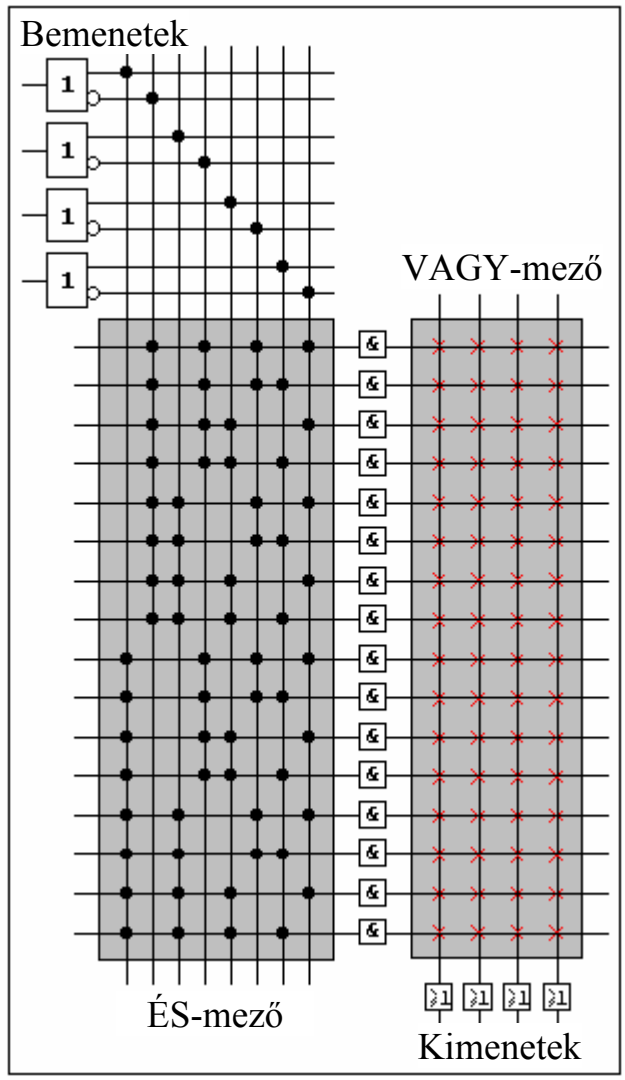


PAL és PROM (PLE)

PAL: csak az AND-mező programozható

PROM: csak az OR-mező programozható



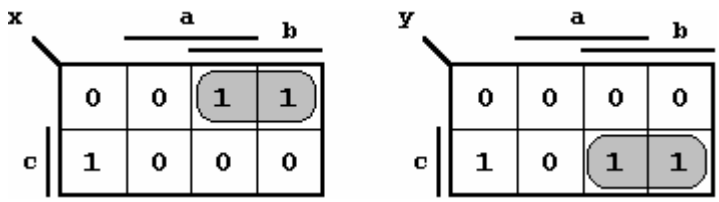


PLD-típus	ÉS-mező	VAGY-mező
PLA (FPLA)	**programozható**	**programozható**
PROM (PLE)	meghatározott	**programozható**
PAL	**programozható**	meghatározott

x(a,b,c) és y(a,b,c) példafüggvények

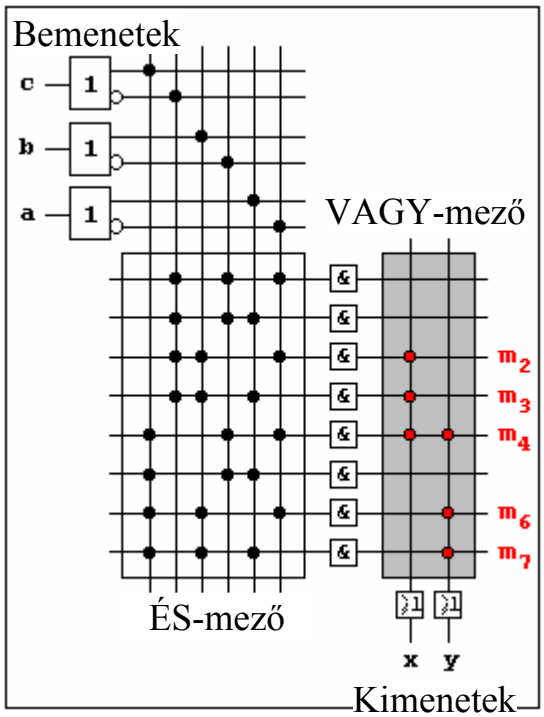
megvalósítása PLA és PROM segítségével

KV-diagram és igazságtáblázat:

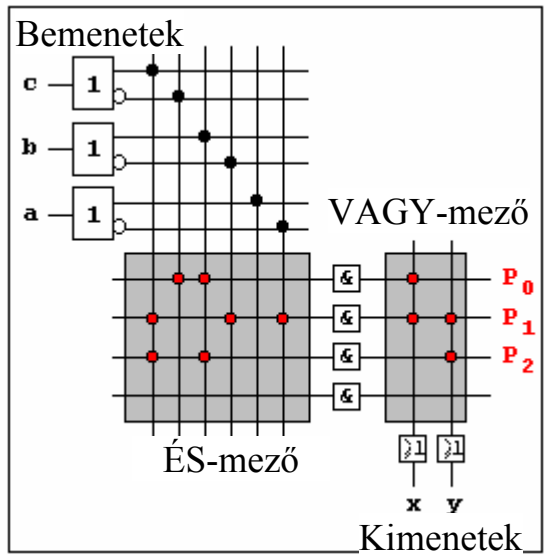


c	b	a	x	y
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	0	1

PLA:



PROM:



PLD-típus	bemeneti jelek száma	ÉS-termek száma
PROM (PLE)	5 - 12	32 - 4096
PLA	10 - 20	2 - 16