

Kombinációs hálózatok egyszerűsítése

© Benesóczky Zoltán 2004

A jegyzetet a szerzői jog védi. Azt a BME hallgatói használhatják, nyomtathatják tanulás céljából. Minden egyéb felhasználáshoz a szerző beleegyezése szükséges.

Cél: A specifikációval megadott KH *legolcsóbb megvalósítása*

Specifikáció: szöveges specifikáció, Boole algebrai kifejezés, normál alak, igazságtábla, de az egyszerűsítés előtt *igazságtáblává kell alakítani, ha nem abban van megadva.*

Mi a legolcsóbb?

Megvalósítási *környezettől függ* (alkatrész).

Pl. Ha önálló kapuk állnának rendelkezésre, a legkevesebb kapu és legkevesebb bemenet szám lehet a *célfüggvény*.

Diszkrét IC-kből (egy IC több kaput is tartalmazhat, hogy hányat, azt a benne levő kapuk bemenetszáma is befolyásolja), *a legkevesebb IC* felhasználása lehet a *célfüggvény*.

Programozható logika esetén pl. a *legkevesebb erőforrás* (logikai cella) felhasználása lehet a *célfüggvény*.

A példáinkban az egyszerű számíthatósága miatt a *legkevesebb kapu bemenet számra* törekszünk.

Hogyan számolható a bemenetek száma?

Pl. a megvalósított kapcsolási rajz alapján.

2 szintű megvalósítás esetén felrajzolás nélkül is egyszerűen számolható.

Legyen a megvalósítandó függvény:

$$F=A.B./C + /A.B.D + C./B$$

Minden termet egy annyi bemenetű kapu valósít meg, amennyi a változóinak száma (N_v). Annyi kapu szükséges, ahány term van a függvényben (N_t).

A fenti esetben 2 db 3 bemenetű kapu és 1 db 2 bemenetű kapuval valósítható meg a függvény, így az 1. szinten összesen 8 bemenet van.

$N_{v1}+N_{v2}+N_{v3}=8$, ahol N_{vi} az egyes termek változószáma.

A 2. szinten egy annyi bemenetű kapu szükséges, ahány termet tartalmaz a függvény (N_t).

A fenti esetben itt egy 3 bemenetű OR kapu szükséges. ($N_t=3$)

Így az összes bemenetek száma $SUM(N_{vi})+N_t=11$.

Kétszintű hálózattal megvalósított kombinációs hálózat egyszerűsítése

A következőkben először 2 szintű KH-ok egyszerűsítésével foglalkozunk.

Az egyszerűsítés elve, az 1 változóban különböző (1 Hamming távolságú) termék megkeresése, majd az eltérő változó elhagyása az

$A \cdot B + A \cdot \overline{B} = A \cdot (B + \overline{B}) = A$ (diszjunktív alak esetén) vagy az

$(A + B)(A + \overline{B}) = A + A \cdot \overline{B} + A \cdot B = A$ (konjunktív alak esetén) azonosság alapján

Az 1 Hamming távolságú termék megkeresésére kézi (ZH-ban jól használható) *grafikus módszerként* az ún. *Karnaugh táblát* használjuk.

A **Karnaugh tábla** egy speciálisan kialakított igazságtábla.

Hagyományos igazságtábla

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

"Peremezett" igazságtábla

F
0
1
1
1
0
1
0
1

C B A

Karnaugh tábla

F			
0	1	1	1
0	1	1	0

A

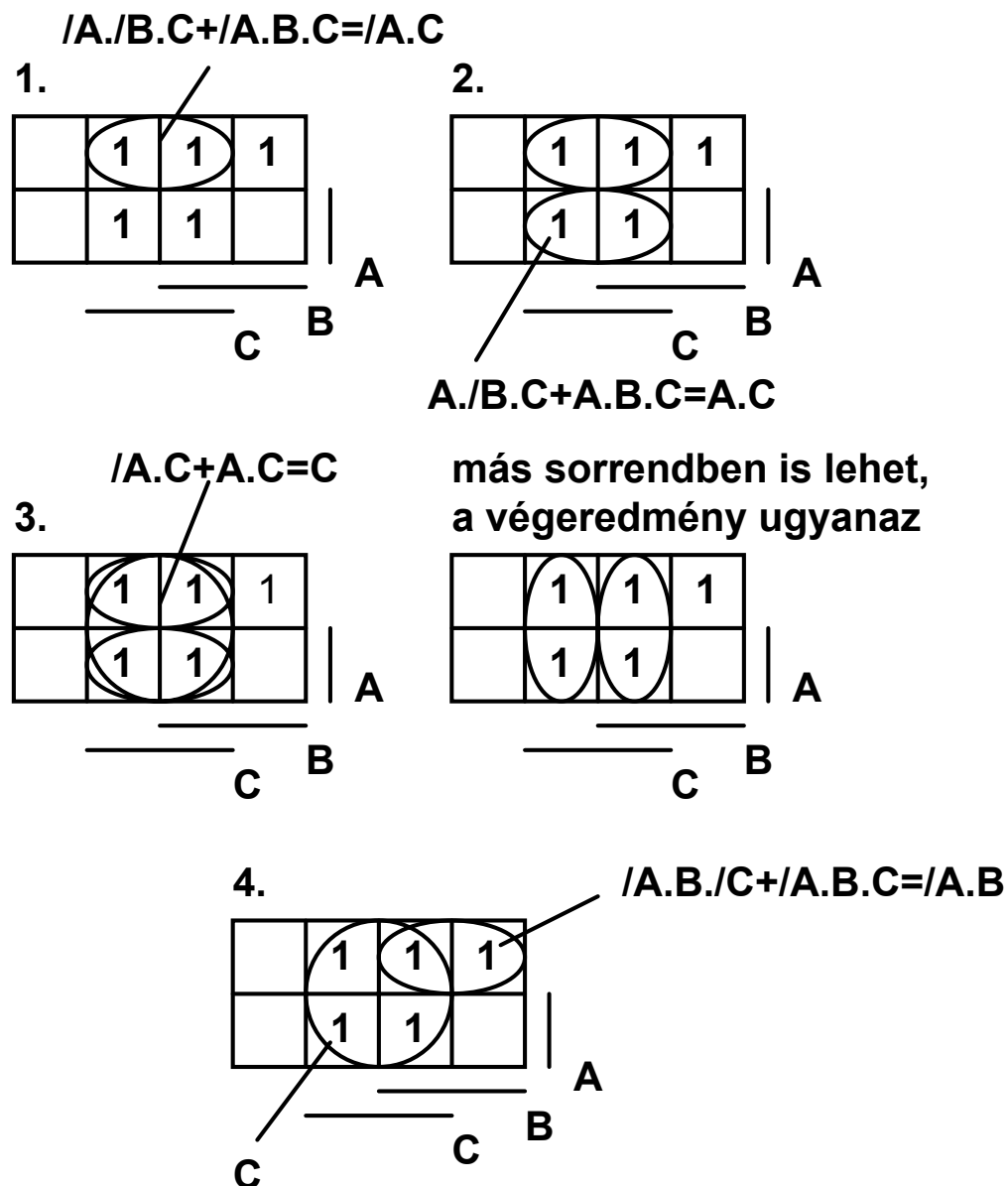
C B

A Karnaugh tábla peremezése olyan, hogy az egymás melletti sorok ill. egymás alatti oszlopok egy változóban térnek el, az un. Gray kód szerint.

A Gray kód szerint peremezett Karnaugh táblán a szomszédos rubrikák (a széleket is szomszédosnak tekintve) 1 változóban különböznek. Ezért a szomszédos termek megkeresése kis gyakorlás után “szemre” viszonylag egyszerű.

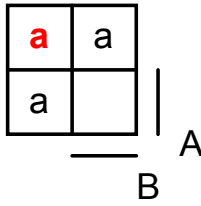
Az alábbi K-táblán az üresen hagyott rubrikákhoz 0 érték tartozik.

Az egyszerűsítés lépésenként elvégezve:

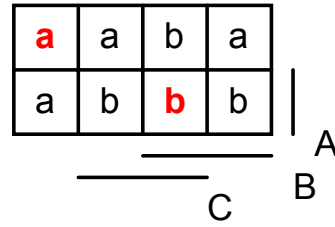


A 2-6 változós
Karnaugh
táblák
rubrikáinak
szomszé-
dossági
viszonyai

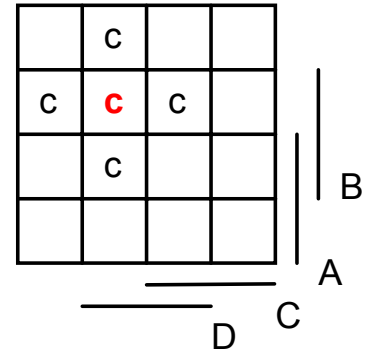
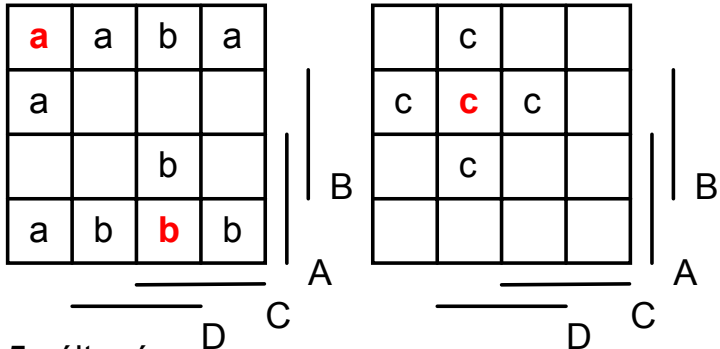
2 változós



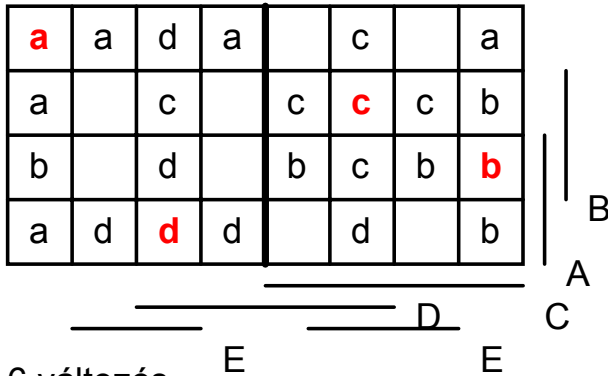
3 változós



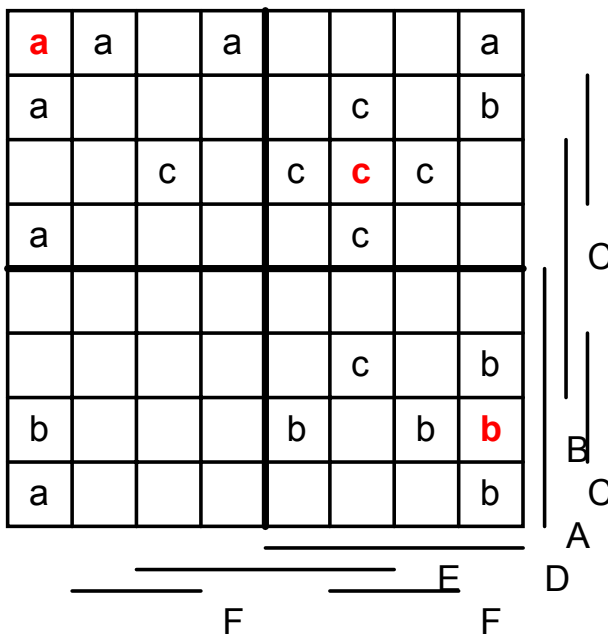
4 változós



5 változós



6 változós



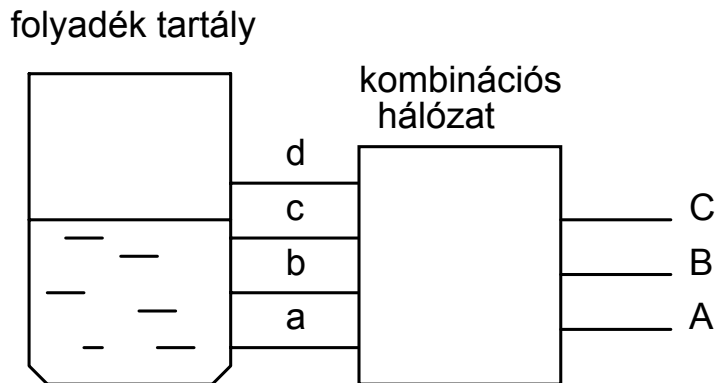
Az igazságtáblában szereplehetnek *közömbös bejegyzések* is! Jelölése: x vagy -

- A bemeneti kombináció, amelynél a közömbös bejegyzés szerepel, *nem fordul elő a bemeneten*
- A bemeneti kombináció előfordulhat a bemeneten, de *a kimenetre csatlakozó logika nem veszi figyelembe*

A közömbös bejegyzések egyszerűsítést tesznek lehetővé, mivel a közömbös bejegyzéseket az egyszerűsítéshez legmegfelelőbb logikai értékkel vehetjük figyelembe.

Pl. Folyadékszint mérő.

A készülék az a-d bemenetein érzékel és a CBA kimenetén jelzi a folyadék állását.



A folyadékszint mérő igazságtáblája:

a	b	c	d	C	B	A
0	0	0	0	0	0	0
0	0	0	1	-	-	-
0	0	1	0	-	-	-
0	0	1	1	-	-	-
0	1	0	0	-	-	-
0	1	0	1	-	-	-
0	1	1	0	-	-	-
0	1	1	1	-	-	-
1	0	0	0	0	0	1
1	0	0	1	-	-	-
1	0	1	0	-	-	-
1	0	1	1	-	-	-
1	1	0	0	0	1	0
1	1	0	1	-	-	-
1	1	1	0	0	1	1
1	1	1	1	1	0	0

A folyadékszint mérő Karnaugh táblái:

C
0 x x x
x x x x
0 x 1 0
0 x x x

B
0 x x x
x x x x
1 x 0 1
0 x x x

A
0 x x x
x x x x
0 x 0 1
1 x x x

A diszjunktív alakban keresett megoldás szerint egyszerűsített Karnaugh táblák:

C
0 x x x
x x x x
0 x 1 0
0 x x x

B
0 x x x
x x x x
1 x 0 1
0 x x x

A
0 x x x
x x x x
0 x 0 1
1 x x x

$C = d$

1 bemenet

$B = b./d$

2 bemenet

$A = a./b + c./d$

6 bemenet

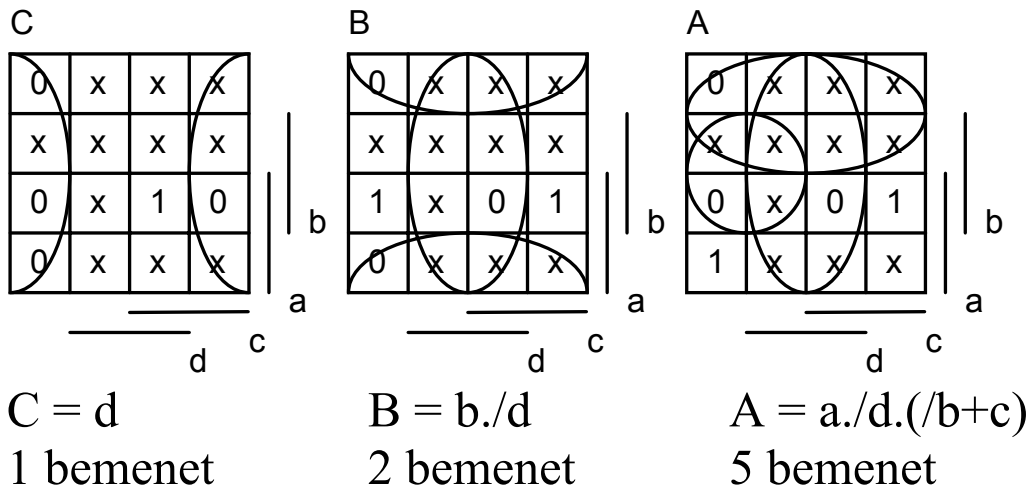
Az egyszerűsített függvény igazságtáblája:

C
0 1 1 0
0 1 1 0
0 1 1 0
0 1 1 0

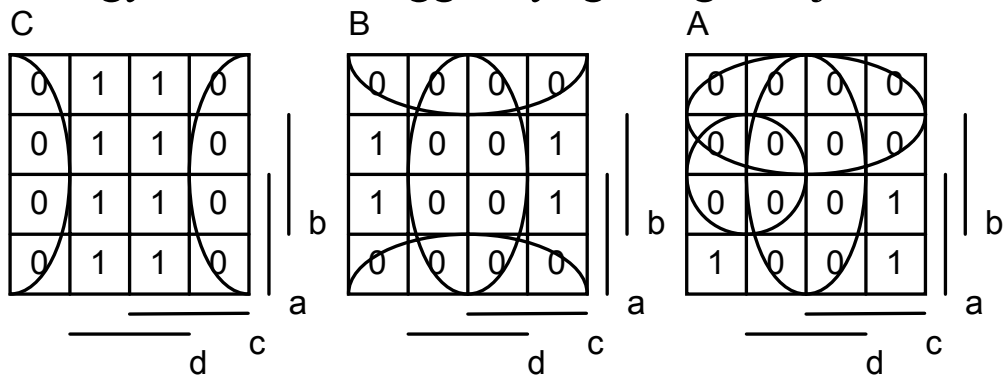
B
0 0 0 0
1 0 0 1
1 0 0 1
0 0 0 0

A
0 0 0 1
0 0 0 1
0 0 0 1
1 1 1 1

A konjunktív alakban keresett megoldás szerint egyszerűsített Karnaugh táblák:



Az egyszerűsített függvény igazságtáblája:



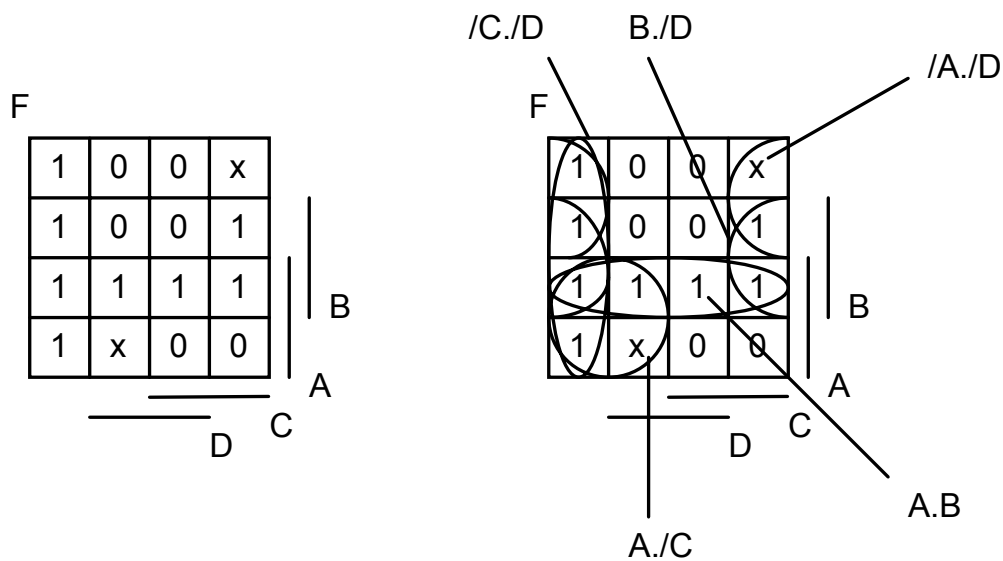
A konjunktív alakú megoldás az egyszerűbb.

A minimalizálás teljes algoritmusa:

1. Keressük meg az összes prímisszorzót

Prímisszorzó: olyan term, amelyből nem hagyható el több változó (nem egyszerűsíthető tovább).

Példa:



A prímisszorzók:

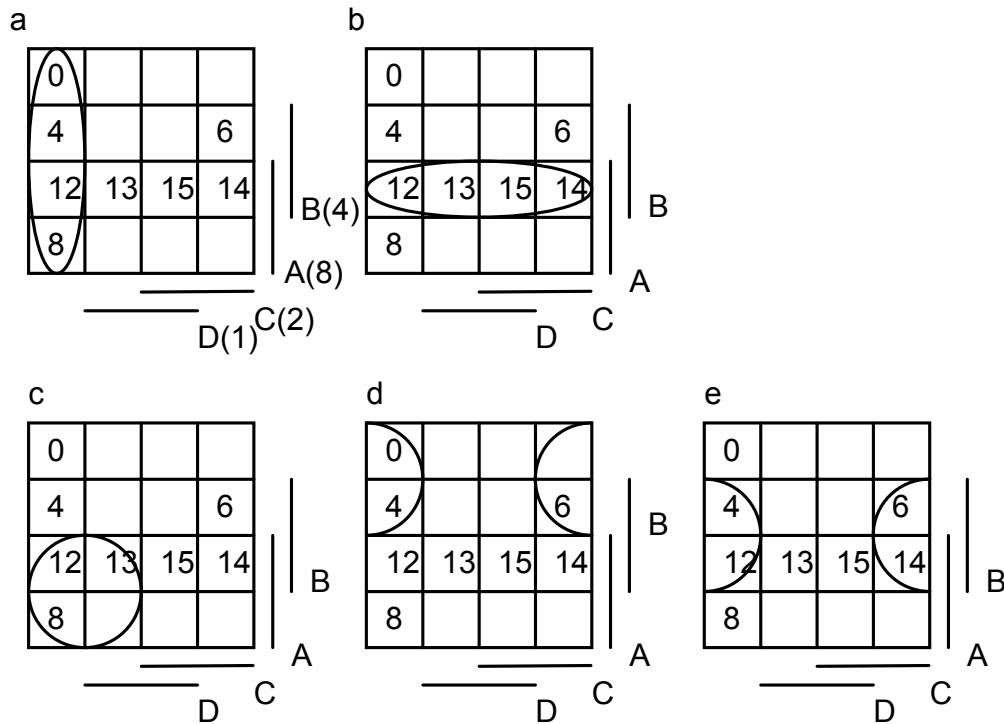
- a: /C./D b: A.B c: A./C d: /A./D
 e: B./D

2. *Válasszuk ki a prímisszorzók közül a legkevesebbet, amely lefedi az igazságtábla összes mintermjét (maxtermjét konjunktív megvalósítás esetén).*

A fedés megvalósításához az ún. *prímisszorzó tábla* ad segítséget. Egyszerűbb esetekben ennélkül “szemre” is elvégezhető az egyszerűsítés.

A tábla oszlopait a minterm (maxterm) sorszámokhoz rendeljük. Itt a közömbös bejegyzésekhez tartozó sorszámok nem szerepelnek, mivel azokat nem kell lefedni.

Az alábbi Karnaugh táblákban külön-külön feltüntettük, az egyes prímisszimplikánsok által lefedett mintermeket.



Prímisszimplikáns tábla

	0	4	6	8	12	13	14	15
a	+	+		+	+			
b					+	+	+	+
c				+	+	+		
d	+	+	+					
e		+	+		+		+	

Lehetnek olyan mintermek, amelyeket csak egy prímisszimplikáns fed le. Az ilyen a mintermeket *megkülönböztetett mintermeknek* nevezik.

Azon príimplikánsokat, melyek legalább egy megkülönböztetett mintermet tartalmaznak, *lényeges príimplikánsok*nak hívjuk.

Példánkban a 15-ös minterm megkülönböztetett minterm, a b príimplikáns pedig lényeges príimplikáns.

A lényeges príimplikánsok feltétlenül szükségesek a fedéshez, mivel más nem tudja helyettesíteni őket.

A továbbiakban azt kell kideríteni, hogy a lényeges príimplikánsokon kívül még melyek szükségesek feltétlenül a fedéshez.

Prímimplikáns tábla

	0	4	6	8	12	13	14	15
a	+	+		+	+			
b					+	+	+	+
c				+	+	+		
d	+	+	+					
e		+	+		+		+	
	a+d	a+d+e	d+e	a+c	+	+	+	+

A prímimplikáns tábla utolsó sorában feltüntettük, hogy egy-egy oszlophoz tartozó mintermet mely prímimplikánsok képesek lefedni, ill. hogy a lényeges prímimplikáns(ok) miket fed(nek) le. Pl. a 0-ás lefedéséhez az a-ra vagy a d-re van szükség.

Mivel az összes lényeges prímimplikáns által le nem fedett prímimplikánst le kell fedni, ezt a *fedési feladatot* az alábbi Boole kifejezéssel fogalmazhatjuk meg:

$$S=(a+d)(a+d+e)(d+e)(a+c)$$

Ezt egy csupa ponált változóból álló kifejezés, melyet ezért könnyű egyszerűsíteni, az elnyelési szabály alkalmazásával.

$$(a+d)(a+d+e) = a + d$$

$$(d+e)(a+c) = ad + cd + ae + ce$$

$$(a + d)(ad + cd + ae + ce) = \underline{ad + ae + cd}$$

A fedési feladatnak a lényeges prímmimplikánst (b -vel jelölt) is figyelembe véve 3 megoldása létezik:

$$abd + abe + bcd$$

Szóban: az összes mintermet lefedéséhez kell: a ÉS b ÉS d , vagy kell a ÉS b ÉS e , vagy kell b ÉS c ÉS d .

$$\begin{aligned} a: & /C./D & b: & A.B & c: & A./C & d: & /A./D \\ e: & B./D \end{aligned}$$

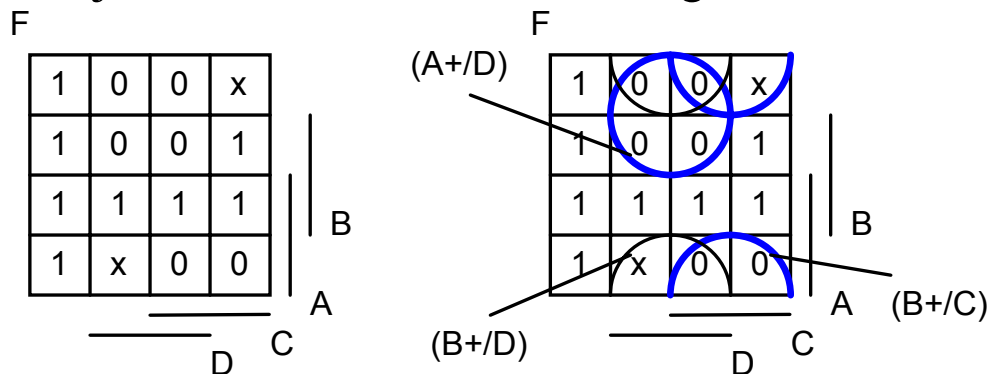
Az ezekhez tartozó függvények, mivel itt a , b , c , d diszjunktív alakú termeket jelöl (mert diszjunktív alakú megoldást keresünk), a megfelelő termek ÉS kapcsolataként adódnak :

$$\begin{aligned} abd: & F = /C./D + A.B + /A./D \\ abe: & F = /C./D + A.B + B./D \\ bcd: & F = A.B + A./C + /A./D \end{aligned}$$

Ezek itt egyformán minimálisak, a kapu bemenetek száma 9.

Általában a különböző megoldásokhoz különböző kapu bemenetszám tartozhat, ekkor a megoldások közül ki kell választani egy legegyszerűbbet.

Konjunktív alakban keresve a megoldást:



A prímisszorzatok:

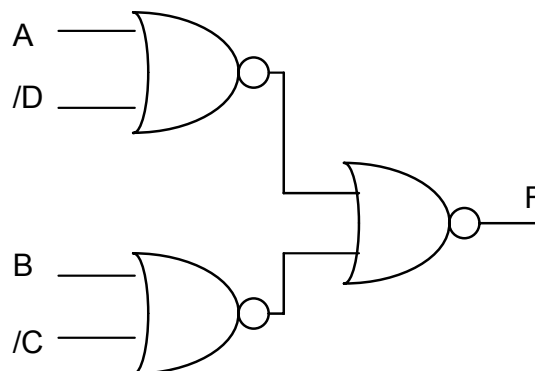
- a. $(A+/D)$ b. $(B+/C)$ d. $(B+/D)$

Lefedési tábla nélkül, “szemre” is látható, hogy az a és b prímisszorzatok lefedik az összes maxtermet. Mivel a és b konjunktív alakú termek (mivel konjunktív alakú megoldást keresünk), a függvényt ezek ÉS kapcsolata adja.

$$F = (A + /D)(B + /C)$$

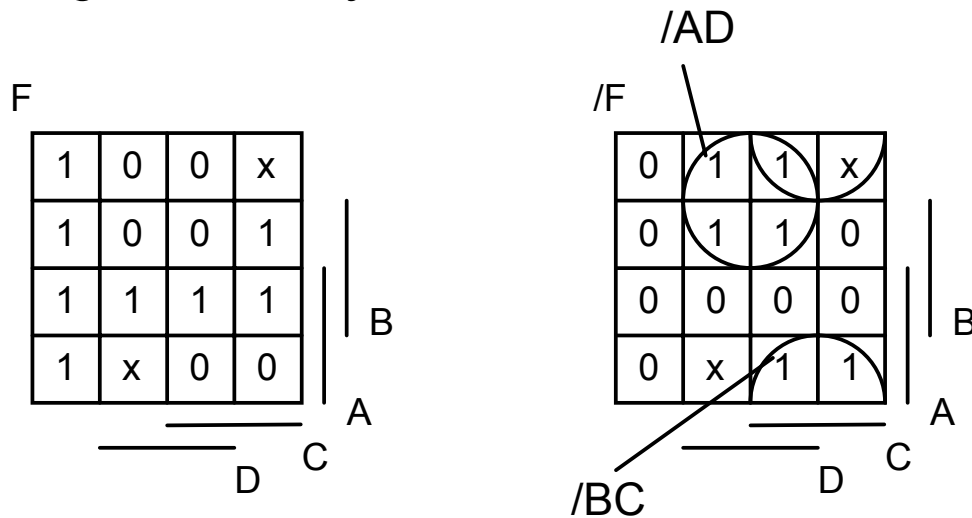
Ehhez 6 kapu bemenet szükséges, így egyszerűbb mint a diszjunktív alakú megoldás.

A kapcsolási rajz homogén NOR megvalósítása:

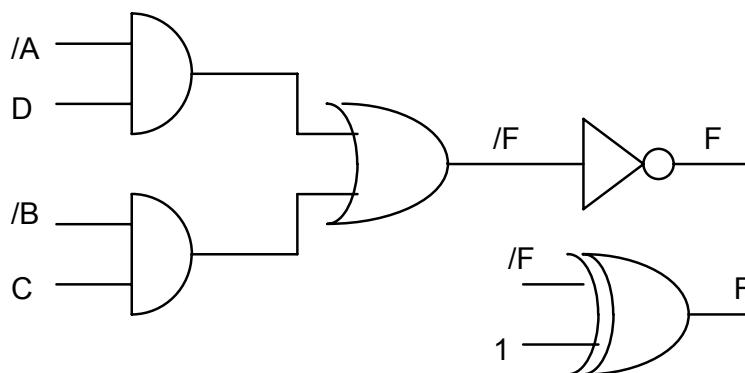


Ha meg van kötve, hogy a megoldást milyen alakban kell keresni és a másik alakban lenne az egyszerűbb, akkor a függvény negáltját lehet megvalósítani, majd negálni. (Például a PLD-ben diszjunktív megvalósítás lehetséges, és a függvényt egy EXOR kapuval meg lehet invertálni.)

Az előbbi példa esetén $\neg F$ -at diszjunktív alakban megvalósítva majd invertálva:



$$F = \neg(\neg A.D + \neg B.C)$$



Számjegyes minimalizálás (Quine-McCluskey módszer)

A logikai függvények minterm (maxterm) indexét úgy képezzük, hogy mintermben ABC sorrendben szereplő változókhoz ponált esetben 1-et, a negált esetben 0-át rendelünk, majd az így kapott bináris szám decimálissá (10-es számrendszerűvé) alakítjuk.

Pl. $\overline{A}/\overline{B}/C/D$

0 0 0 1

A **prímimplikánsok** meghatározásakor a szomszédos termeket kell megkeresni és összevonni. Pl. $\overline{A}/\overline{B}/C/D + \overline{A}/\overline{B}/C/\overline{D} = \overline{A}/\overline{B}/C$

A számítógépes módszer a mintermeket a bináris indexükkel reprezentálja. A szomszédos (egy változóban ellentétes előjelű) termék indexének bináris formája 1 bitben tér el. Az a változó esik ki, ahol az eltérés van.

Pl. $\overline{A}/\overline{B}/C/D + \overline{A}/\overline{B}/C/\overline{D} = \overline{A}/\overline{B}/C$

0000

0001

000-

Tehát az egy bitben eltérő számokat kell megkeresni, és az eltérő bit helyhez rendelt változót elhagyni (itt ezt “-,-szal jelöltük)

A szomszédos termeket gyorsabban meg lehet találni, ha először a bináris minterm indexeket a bennük levő 1-esek száma (*bináris súlyuk*)

alapján sorbarendezzük. Ekkor már csak a súlyuk szerint egymás melletti indexeket kell egymáshoz hasonlítani.

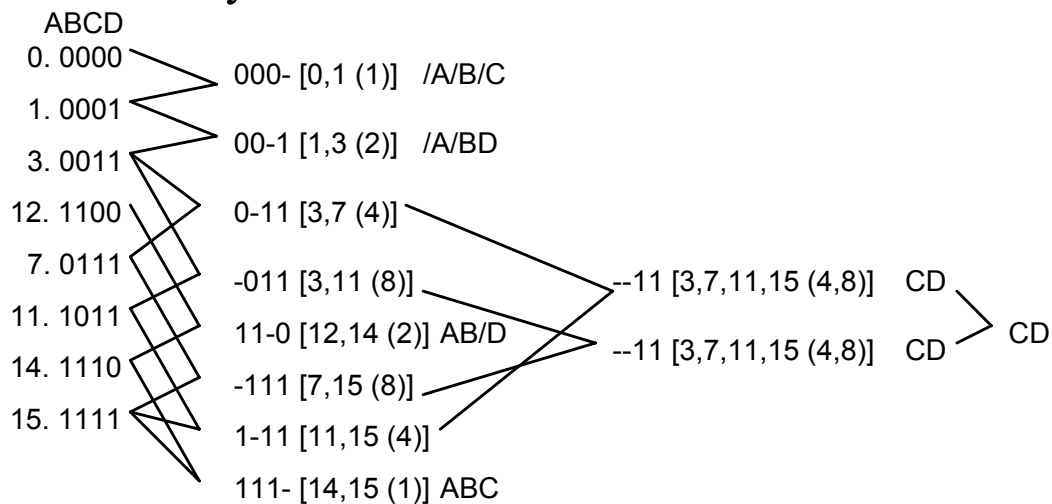
Az algoritmust egy mintapéldán mutatjuk be.

$$f = \overline{A}B\overline{C}D + \overline{A}B\overline{C}D + \overline{A}BCD + AB\overline{C}D + \overline{A}BCD + ABC\overline{D} + ABCD =$$

$$m_{0000} + m_{0001} + m_{0011} + m_{1100} + m_{0111} + m_{1011} + m_{1110} + m_{1111}$$

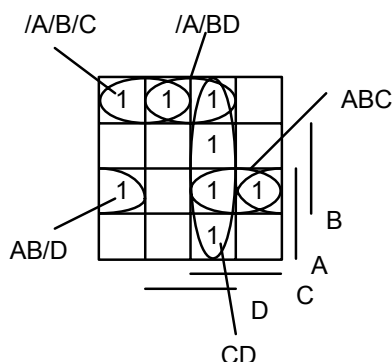
A továbbiakban csak az indexet írjuk le.

Bináris súly szerint rendezve:



Az “[]”-jelek közé téve az Arató könyvben használt jelöléseket is feltüntettük.

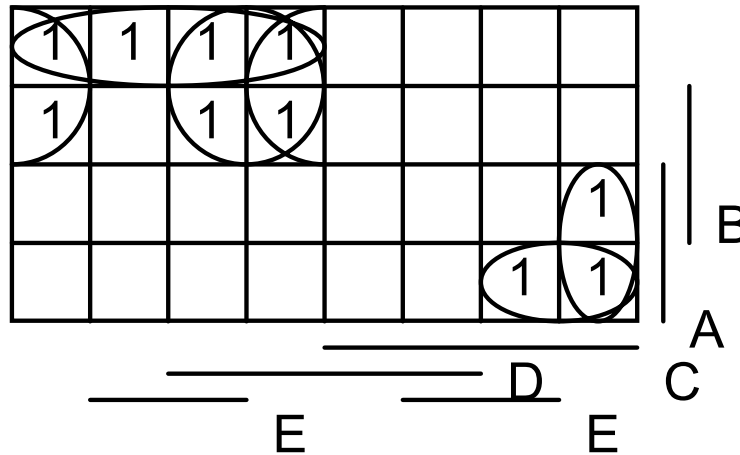
Karnaugh táblával ugyanezt az eredményt kapjuk:



$$f = CD + ABC + \overline{A}BD + \overline{A}B\overline{C} + AB\overline{D}$$

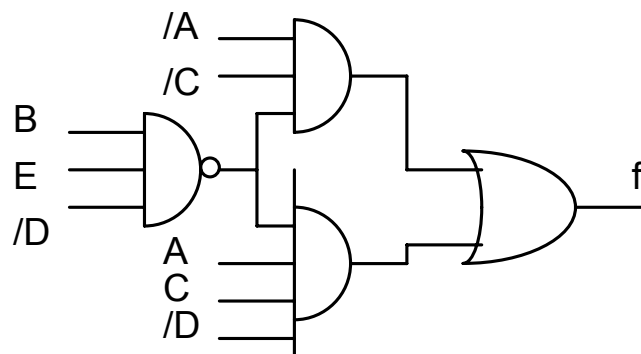
Logikai függvények realizálása többszintű hálózattal

1. *Algebrai átalakításal hozzuk többszintű hálózattal realizálható alakba a függvényt*



A kétszintű minimális diszjunktív alak 22 bemenettel valósítható meg.

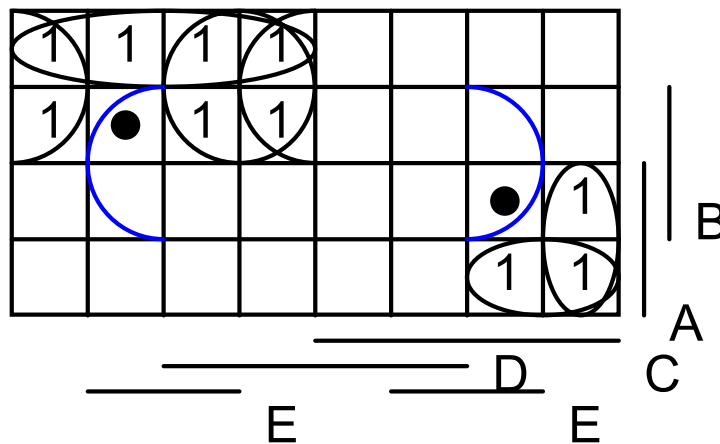
$$\begin{aligned}
 f &= \overline{A}/B/C + \overline{A}/C/E + \overline{A}/CD + AC/D/E + A/BC/D = \\
 &= \overline{A}/C(\overline{B} + \overline{E} + D) + AC/D(\overline{B} + \overline{E}) = \\
 &= \overline{A}/C(\overline{B} + \overline{E} + D) + AC/D(\overline{B} + \overline{E} + D) = \\
 &= \overline{A} \cdot \overline{C} \cdot \overline{(BE/D)} + AC/D \cdot \overline{(BE/D)}
 \end{aligned}$$



A 3 szintű megvalósításhoz 12 kapu bemenet szükséges.

A többszintű realizálás *sokszor egyszerűbb áramkört eredményez*, viszont *hosszabb lesz a hálózat jelterjedési ideje*.

2. Kitiltás módszere többszintű hálózat létrehozására



a. Ha a ponttal jelölt helyeken 1 lenne, sokkal egyszerűbb megoldás adódna. Egyszerűsítsük a függvényt, mintha ez lenne az eredeti függvény.
 $f^* = /A./C + A.C./D$

b. Megfelelő (lehetőleg egyszerű) K kitiltó függvénnyel megszorozva f^* -t biztosítsuk, hogy a függvény újra az eredeti igazságtábla valósuljon meg.

$$f = f^* \cdot K \quad K = /(B/DE)$$

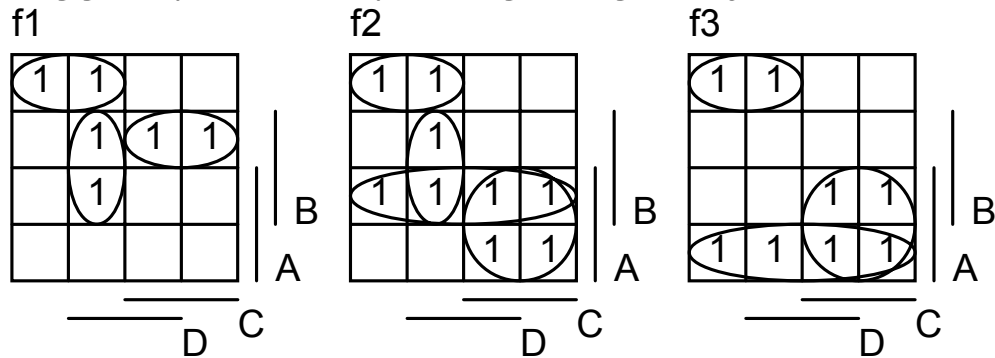
$$f = /(B/DE) (/A/C + AC/D) =$$

$$= /A./C./(B/DE) + AC/D/(B/DE)$$

Ez most ugyanarra az eredményre vezetett, mint a Boole algebrai átalakítás.

Többkimenetű kombinációs hálózat minimalizálásának elve

Legyen egy 3 kimenetű 4 változós logikai függvényünk, melynek igazságtáblája az alábbi.



$$f1 = \overline{A}.\overline{B}.\overline{C} + \overline{A}.B.C + B.\overline{C}.D$$

$$f2 = \overline{A}.\overline{B}.\overline{C} + A.B + B.\overline{C}.D + A.C$$

$$f3 = \overline{A}.\overline{B}.\overline{C} + A.\overline{B} + A.C$$

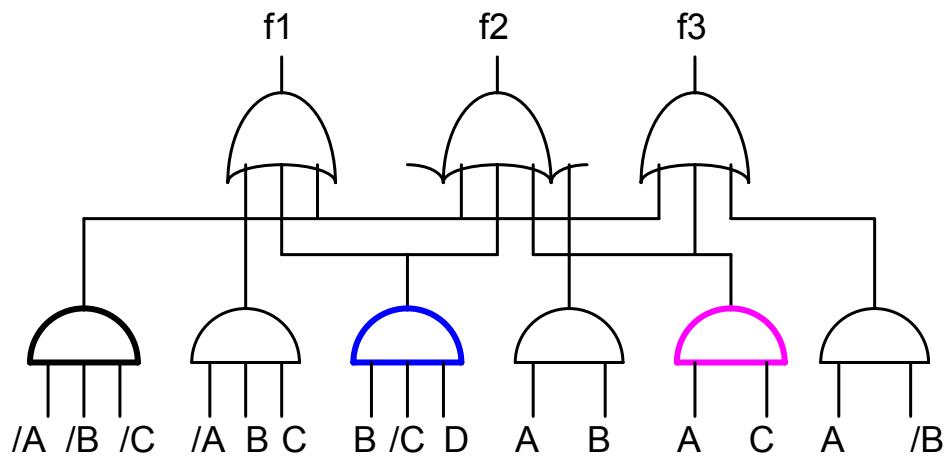
Elv: a több függvényben is előforduló (azonos) prímisszorzatok csak egyszer valósítjuk meg.

$$f1, f2, f3: \overline{A}.\overline{B}.\overline{C}$$

$$f1, f2: B.\overline{C}.D$$

$$f1, f3: \text{-----}$$

$$f2, f3: A.C$$



Az eredeti 36 kapubemenet helyett csak 25 kell.