I. A Digitális Technika alapjai

Bevezetés

Ebben a könyvben az információfeldolgozás alapjaival ismerkedünk, elsősorban a digitális technikával, és ennek matematikai alapjaival. A digitális technika számokkal dolgozik, a valóságot számokká (analógból digitálissá) alakítja, számokkal végez műveleteket, és a kapott szám-eredményeket csak a folyamat végén fordítja vissza analóggá (digitálisból analógra).
Mivel a digitális technika sok részből áll, melyek önmagukban is sok ismeretet jelentenek, könyvünket több részre, fejezetre bontjuk. A fejezeteket külön oldalszámozással láttuk el, fejlécükben pedig feltüntettük az éppen aktuális téma címét, hogy könnyen kereshető, áttekinthető legyen.
A fontos definíciókat ☞ jellel jelöltük, ezek megértése, megtanulása fontos. Az egyéb fontos tudnivalókat is jelöltük, keretezéssel, és szöveges figyelemfelhívással.
Az elektronika és a mérés fontos, hogy a digitális áramkörök áramköri tulajdonságait is megértsük, hogy a jövő szakembere el tudja dönteni, jó, vagy hibás az áramkör, megértse a katalógusok adatait, és ne tegye tönkre szakszerűtlenségével az általában nagy értékű eszközöket, áramköröket.
A fejezetek, olykor alfejezetek végén kérdések, elméleti- és gyakorlati feladatok segítenek a mélyebb elsajátításban. **A gyakorlati feladatok egy része mérési utasítás, amit a valóságban, a gyakorlati foglalkozásokon mérni is kell**, tehát könyvünk mind a hardver elmélethez, mind a hardver gyakorlathoz kell.
Ebben a könyvben csak az alapokkal foglalkozunk. Az alapok bővebben, a gyakorlati megvalósítások inkább példa, vagy bemutatójelleggel szerepelnek. Egyszerű, tanulmányozható áramköröket mutatunk be, dolgozunk ki, többnyire nem érjük el a gyakorlati megvalósítás, csak a megértés szintjét. Ennek oka, hogy a gyakorlatban már igen olcsók a nagybonyolultságú integrált áramkörök, melyekkel sokkal-sokkal olcsóbb felépíteni egy órát, számológépet, vagy általában bármely digitális eszközt, áramkört.
Nem az iparban, irodában használatos berendezéseket vizsgálunk, sem ilyeneket nem készítünk. Tanulmányaink során, csak kísérleti jelleggel összeállított áramkörökkel dolgozunk, laborkörülmények között. Tehát a kidolgozott, bemutatott, kimért áramkörök működnek ugyan, de inkább a megismerésre, és az ehhez kapcsolódó alapfogalmak elsajátítására szolgálnak. Azok az áramkörök, melyeket bemutatunk, többféleképpen is létrehozhatók, mi sokszor csak egy, legfeljebb néhány megvalósítást említünk, korántsem a teljesség igényével, hanem bemutató jelleggel. E könyv célja nem az átfogó ismeret átadása, hanem, hogy az olvasó a téma alapfogalmait megértse, és így önállóan tudjon tájékozódni a szakirodalomban, és katalógusokban, applikációkban.

Információ

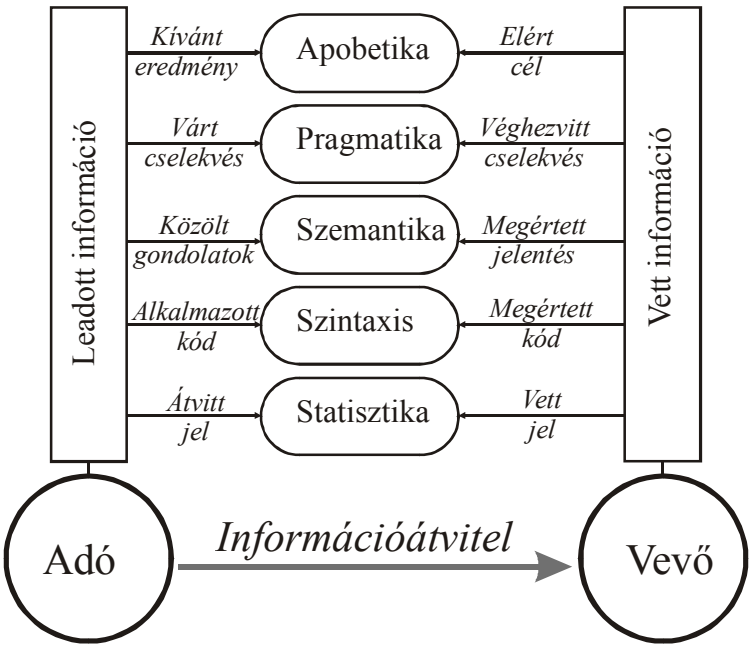
☞ Az információ hír, értesülés, mely ismereteinket megváltoztathatja, létrehozhatja, törölheti, stb. **Az információ mindig valamilyen változtató képességet jelent**.

Ha egy jelenség nem okozhat változást, nem információ. Pl. a Holdon levő szikladarab csak akkor válik információvá, ha megismerjük, meglátjuk, így változik a tudásunk, vagy tudatni szeretnénk másokkal. Így a legtöbb szikla a Holdon nem információ, csak az, amire figyelmünket irányítjuk (akarjuk megismerni), vagy amire mások irányítják a mi figyelmünket (akarják, hogy megismerjük).
Az akarat, értelem, érthetőség nélküli jelek sokszor inkább károsak, zavaróak. Sokszor zajnak, zavarnak is nevezzük őket, de természetesen a legtöbbet nem is észleljük.
Az információ tehát szabályoknak megfelelő, értelmes, és mindig valamilyen ismeretet, vagy cselekvést megváltoztató akarattal kapcsolatos dolog. Az információnak el kell jutni ahhoz, aki veszi, ezért az információ hordozója, közege miatt anyaghoz kötődő, de önmagában nem anyagi fogalom.

Az információ részletesebben

Szokás az információ öt szintjét megkülönböztetni (Werner Gitt nyomán):
Apobetika, mely tulajdonképpen maga a cél, a „mit” kérdésre adott válasz. (A görög *apobenion* = eredmény, siker, kimenetel). A cél az információt adó szempontjából a terv, a vevő oldalán célkitűzésként jelentkezik. A legjobban jellemző kérdés, melyre az apobetika választ ad, a **miért?** Könyvünkben az apobetikával többet nem foglalkozunk.
Pragmatika, mely a cél megvalósításának a **módját** jelenti. (Görögül *Pragmatike* = „A helyes cselekvés művészete”.) Az adó határozza meg, miként érje el a célját a vevőnél. A legjobban jellemző kérdés, amire a pragmatika választ ad a **hogyan, mi módon?**
Szemantika, mely az adott és vett jelsorozatok szabályait jelenti. (Görögül a *szemantikósz* = jelölő, jelentő, *szemémák* = jelek). Ez inkább a programozást érinti, könyvünkben ezzel sem foglalkozunk.
Szintaxis, (Görögül szüntaxis = elrendezés) az információ ábrázolás karakterkészletét (jelkészletét), és karakterkészletének összefüggéseit írja le. **Ezt a karakterkészletet és szabályait kódnak nevezik. Az egymáshoz ren-**

delt karaktereket pedig szavaknak. A szintaxis a jelcsoportok, szavak tulajdonképpeni jelentését írja le. Ez a nyelvtanra is érvényes, a beszélt nyelvre. A digitális technikában is használnak szavakat, lásd a *Több bites információ egységek* c. fejezetet *(I.4. old.)* Könyvünkben alaposan foglalkozunk a kódolással- és szabályaival.
Statisztika alatt a jelek, jel- és szósorozatok összességét értjük. A jelek, szavak sorozata még nem információ, ahhoz ismert, egyezményes kódnak megfelelő kell, hogy legyen, értelmes legyen, és változtatni képes (tudást, vagy cselekvést). A jelek-, jelsorozatok, az általuk kódolva tartalmazott hír terjedéséhez, útra van szüksége. Az út két végpontján tehát mindig áll legalább egy adó, meg egy vevő. Az információ hordozója egyben tárolhatja is az információt. Az információt át kell alakítani, hogy a hordozó ábrázolhassa. A jelek összességét, mint statisztikai mennyiséget nevezzük statisztikának. Ha a jelek mennyisége megnő, nem mindig nő az információ mennyisége, pl. egy megzavart jelsorozat információtartalma nem, hogy nő, de inkább csökkenhet.



Az információátvitel öt szintje Werner Gitt szerint

Könyvünkben csak a két alsó szinttel fogunk foglalkozni: **A jelekkel**, a rajtuk végzett műveletekkel, átalakításokkal, tárolásukkal, és **a kódokkal**, kódolással. Elsősorban számjegyes (digitális) jeleket tanulmányozunk, de fogunk venni egy kevés elektromos jelfeldolgozást is.

Az információ fogalmát nehéz pár mondatban meghatározni, inkább tulajdonságait szokták definiálni, és így írják le ezt a nehéz, sokféle értelmezés- és vita tárgyát képező fogalmat. A Werner Gitt féle információ fogalom elemeit egyre többen átveszik, megtalálni a Magyar nyelvtan tantárgyban, vagy az általános műveltség sok területén.
A téma után alaposabban érdeklődőknek ajánlom prof. Werner Gitt: *Kezdetben volt az információ* című könyvét, ennek 4. és 5. fejezete az információ fogalmáról szól, amihez vagy harminc tételt állított fel... Werner Gitt e könyvében [az 1. Függelékben] bemutat más információ-fogalmat is, pl. a Claude E. Shannon féle információelméletet, mely kiválóan alkalmas az információ-, mint statisztikai mennyiség meghatározására. Shannontól származik a bit fogalom is. Az információ fogalma azonban nem csak a statisztika szintjére értendő. Shannon információelmélete csak a statisztika (és a vele kapcsolatos mérnöki-műszaki szinten, azaz a jelek szintjén) szintjén elfogadott érvényű.
Nem vitatjuk senki információ-fogalmát, csak használjuk a Werner Gitt féle információátvitel öt szintjéből az alsó kettőt, a fizikailag létező (statisztikai mennyiségekkel kapcsolatos) jeleket, és a szintaxissal kapcsolatos kódolást-dekódolást. A két szintnél magasabbra mi könyvünkben nem jutunk.

Adat: előírt, kötött formájú információ

Az adat és az információ nagyon sokszor egymás szinonimájaként szerepel. Mivel az információfeldolgozás során csak megfelelő alakú és állapotú információt lehet a feldolgozó eszközöknek adagolni, **ad**ni, a megfelelő formájú, állapotú információt tekintjük adatnak.

- **Adatnak az előírt, feldolgozás számára megfelelő formájú információt nevezzük**

Kód, kódolás, dekódolás

- Az információ ábrázolás **karakterkészletét (egyezményes jeleit, jelrendszerét)**, és karakterkészletének **összefüggéseit, szabályait kódnak nevezik.**
- Az egymáshoz rendelt karaktereket (együtt szereplő több karaktert) szavaknak nevezik.

Erről lásd még a *Több bites információ egységek* c. fejezetet *(I.4. old.)*

- **Az egyezményes átalakítási szabályt kódolásnak nevezzük.**
  Akkor is kódolásnak nevezik az átalakítást, ha így kevesebb eszközt (pl. vezetéket, időt, tárolót, sávszélességet, stb.) igénylő ábrázolási módra jutnak.

Pl. kevés vezetéken is nagy mennyiségű információt lehet átjuttatni kódoltan. Mi is tanuljuk ennek néhány megvalósítási módját.

- **A megfejtést, és a nagyobb eszközigényűre alakítást dekódolásnak nevezik.**
  Az átalakítást akkor nevezik dekódolásnak, ha az átalakítás során kevesebb eszközigényű ábrázolásról nagyobb igényűre jutnak.

Ha pl. az egy vezetéken érkező kódolt információt szétosztják igen sok felhasználónak, akkor ez dekódolás.

- Bináris kódnak a bináris számjegyekkel történő ábrázolásra átalakító szabályt nevezzük.

Többféle bináris kódot tanulunk. Közülük a legegyszerűbbet, a bináris számrendszert nevezik sokszor pusztán bináris kódnak, míg a többit egyéb kiegészítő nevekkel látják el. Pl. binárisan kódolt decimális számok.
A terjedés során többször is átalakíthatják, átkódolhatják a jelet. A fentiek szerint hol kódolásnak, hol dekódolásnak nevezik az átkódolást, attól függően, mennyi eszközt igényel az ábrázolása az átalakítás előtt és után.

- Alfanumerikus kódnak a betűket, számokat és egyéb karaktereket átalakító szabály leírását nevezzük.

Alfanumerikus kóddal vannak ábrázolva a számítógépekben a fontok. Egybájtos számjeggyel vannak ábrázolva a CWI, ASCII (kiterjesztett), 852-es kódlap (codepage). A Windowsban több bájtos karakterkészletek is vannak, akár tízezernyi karaktert is képesek ábrázolni (unikód).

Az információ csak kódolt (dekódolt) formában kerülhet feldolgozásra és továbbításra, különben nem értenék sem a gépek, sem az ember. Ilyen kód pl. a hangrezgések változásának jelentése (beszédhangok), a jelek szintje változásának jelentése, stb.

Egy adott úton terjedő kódolt információ **jel** alakjában terjed, az adó jeleket ad, a vevő jeleket fog. A jelekkel, jelátvivő képességgel, és jelfeldolgozással az Elektronika, méréstechnika részben foglalkozunk.

Az információ a digitális technikában a leggyakrabban bináris kódban van ábrázolva.

> **Ezután a Digitális technikával foglalkozó fejezetekben csak a binárisan kódolt információval foglalkozunk csak.**

Ennek okai a *A legkedvezőbb alapszám* című fejezetben *(II.7. old.)* kerülnek kifejtésre.

Az információ útja, adathordozók, információtárolás

Az információ anyagban, közegben, hordozón, azaz valamilyen **úton kell, hogy terjedjen.** Ez az út az adathordozó (rádióhullám, optikai lemez, vagy magnószalag, hang, könyv, sajtóter-

mék, stb.) A tárolandó, vagy átszállítandó információnak korántsem mindegy, mekkora a mennyisége. Erről szól a következő fejezet.

Az információ mennyisége

- **az információ elemi egysége a bit**, **Bi**nary **D**igit. Jele **b** (kis b betű)
- **A bit egy darab kettes számrendszerbeli számjegyet jelen**t, **értéke 0 vagy 1 lehet,** egyszerre csakis az egyik.

A lehetséges legkisebb változás-, vagy nem-változás híre **kétféle lehetséges állapot**ról szóló hír. Pl. van, vagy nincs, igen, vagy nem, igaz, vagy hamis. Számjegyekkel 1, vagy 0. 1 bitnél kisebb mennyiségű információ nincs.

Eleminek azért nevezzük a lehetséges legkisebb információmennyiséget, mert csak egész számú többszöröse létezhet, úgy tekinthetjük, mint az információ elemeit, építőkészletét.

Az információegységeknek a fentiek alapján tökéletesen megfelelnek a kettes számrendszer számjegyei, a 0 és az 1. Látni fogjuk, hogy a számtani műveletekhez (aritmetika) és a logikai műveletekhez is kiválóan megfelel a kettes számrendszer. Alkalmazásával lehetségessé válik, hogy ugyanazok az áramköri elemek számolási és logikai műveleteket is végrehajthassanak. Így a digitális technikában kialakult, hogy amit csak lehet, a kettes számrendszer jegyeinek megfelelő kétállapotú jelekkel hajtanak végre, ill. dolgoznak fel, 0-kal és 1-ekkel. További előnye, hogy így egy vezetéknek csak kétféle állapotúnak kell lennie, nagyobb, vagy alacsonyabb feszültségűnek, stb. További példák: van áram, vagy nincs áram, van, vagy nincs, lyuk, sötét, vagy világos, így mágnesezem, vagy úgy, stb

Nem csak kettes számrendszerbeli, hanem más számrendszerrel, vagy betűkkel leírt információ-ábrázolás is létezhet. Mégis, **a digitális technika áramköreiben szinte kizárólag bitek, kétállapotú jelek találhatók**. Még az egyéb számok, betűk, karakterek is bitekkel, persze több bittel vannak ábrázolva..

Több bites információ egységek

- Bináris kódnak a kettes számrendszer számjegyeinek használatát nevezzük.

Ami bináris kódolású, azt bitekkel ábrázoljuk, hiszen a bit a kettes számrendsze számjegye.

Az eleminél több információt szükségszerűen több bit kell, hogy hordozza.

- **Szó** alatt az egynél több bites információt értjük. Szóhosszúságnak pedig a bitek számát. A szóban mindig kötött a bitek helye, mást kapunk, ha két, vagy több bitet felcserélünk (akár a számjegyeknél).

A technika története alatt 2-, 4-, 8-, 16 bites, 32 bites, 64, és több bites szóhosszúságú szavakat alkalmaztak a különböző információ-egységeket feldolgozó eszközökben.

Hogyan függ a bitek számától az, hányféle állapotú lehet egy szó?

- A bitekből álló szó lehetséges állapotainak száma 2^n, ahol n a bitek száma.
- X számú különböző állapot megkülönböztetésére $n = \log_2 X$ bitre van szükség

Minél részletesebben kívánjuk ábrázolni az információt, annál több bitre van szükségünk.

Természetesen nem csak bináris kódú információ létezik, de a másképp kódolt információ mennyiségét is lehet jellemezni bitekkel. Néhány példa információ mennyiségekre:

Egy átlagos nyelv egy betűjének információtartalma kb. négy bit (Shannon számításai nyomán.) Ehelyett legalább 8 bittel ábrázolják, mint karaktert. Ám így lehetőség nyílik a karakterkészletben nem csak betűket, hanem számokat és más írásjeleket is ábrázolni.

Egy nagylexikon összes információtartalma $1{,}28 \cdot 10^7$ bit (Meyer Nagy Univerzális Lexikona).

Az USA Kongresszusi Könyvtára (Washington, tízmillió könyv) 10^7 kötet, 10^6 bit/kötettel összesen 10^{13} bit.

Az emberiség könyvekben található össztudása 10^{18} bit (1997-es adat, 625 milliárd könyvvel számítva).

Egy ember szókincse, ha 100 000 szót feltételezünk $3{,}9 \cdot 10^6$ bit.

Az átlagos ember agyának tárolókapacitása tisztán materiális szempontból 10^{13}-10^{15} bit (McCullah szerint), és $2{,}65 \cdot 10^{20}$ bit, ha csak az idegsejtek adta lehetőségeket vizsgáljuk (sokkal-sokkal több, mint az egész emberiség könyvekben levő tudása).
A példákat Werner Gitt fentebb említett könyvéből vettem (159-161. oldal).

A nevezetes szavak

A kevés bitszámú szavaknak saját nevet adtak, ezért azokat nevükön, és nem szóként említjük. Tekintsük a kevés bitszámú információegységeket:

- **Triád** alatt egyszerre három bitet értünk. Egy triád 8-féle lehetséges állapotot vehet fel.

A triád 8 lehetséges állapotát az oktális (8-as) számrendszer számjegyeivel lehet úgy ábrázolni, hogy a 8 lehetséges állapot mindegyikéhez egy-egy oktális számjegyet rendelünk.

- **Tetrád** alatt egyszerre négy kötött sorrendű bitet értünk. Egy tetrád 16-féle lehetséges állapotot vehet fel.

A tetrád 16 lehetséges állapotát egy-egy számjeggyel lehet jellemezni, úgy, hogy mindegyikéhez egy-egy hexadecimális (16-os számrendszerbeli) számjegyet rendelünk.

- **A bájt (byte)** jelentése**: egyszerre nyolc bit** információ. Jele **B** (nagy B betű).

Egy bájt 256-féle lehetséges állapotot vehet fel (adhat róla információt). A 256 lehetséges állapot túl sok, hogy számjegyekkel lehetne ábrázolni, de lehet számokkal, betűkkel és egyéb írásjelekkel. Ilyenek az alfanumerikus kódok, pl. az ASCII. De ennek megjegyzése nehéz. Ám **a bájtot le tudja írni két hexadecimális számjegy**, és ez jóval rövidebb, mint ha 8 bittel írnánk le. Ezért a hexadecimális számrendszer elterjedt és használatos a programozásban.

Mára leggyakrabban a bájt használatos a technikában, sok esetben akkor is bájtot használnak (vagy egyszerre több bájtot), ha kevesebb bittel is megoldható lenne a feladat (vagy kevesebbel, mint 8 egész számú többszöröse).

Prefixumok (állandó szorzók)

A nagyobb decimális számok esetén prefixumokat (állandó szorzókat) használnak, mint a kilo (10^3), Mega (10^6), stb. A bináris számoknál is bevezették a prefixumokat, melyek hasonlítanak elnevezésben is, értékben is a decimálishoz, pl. a bináris kilo 1024, míg a decimális 1000.
Problémát jelentett a hasonló jelentés miatt, hogy olykor nem volt egyértelmű, mikor bináris, és mikor decimális prefixumot használnak. Ezért Nemzetközi Elektrotechnikai Bizottság (International Electrotechnical Commission, IEC) 1998-ban elkészítette az új prefixumok használatára vonatkozó ajánlását. Az ajánlás szerint, a bináris prefixumot ki kell egészíteni a „binary” szóval, ami a kettes számrendszer használatára utal. **A jelölésben ez egy kis „i” betű formájában látszik az eredeti prefixumjel mellett.** A kiejtésre is vonatkozik az ajánlás, helyesen az „i” betűt „bee”-nek kell ejtenünk (angol). Még egy változtatás történt, a kilót eredetileg „k” betűvel jelöltük. Ez a bináris rendszerben nagy K-ra változott, amivel a többi prefixum jelölését követi.
Az alábbi táblázatban foglaltuk össze a használható prefixumokat. Ilyen bináris prefixumok a következők:

- **10^3** neve **kilo**, jele **k**, értéke 1 000, kimondva ezer.
  Pl. kA kimondva kiloamper, jelentése 1000 A.
- **2^{10}** neve **kibi**, jele **K,** vagy **Ki**, értéke 1 024, kimondva **kilobinary**
  Példák: 1Kb kimondva kibibit, jelentése 1024 bit.
  64KiB kimondva 64 kibibájt, jelentése 65 536 bájt. Ezt sokszor még 64KB-nak írják.
- **10^6** neve **Mega**, jele **M**, értéke 1 000 000, kimondva egymillió
  Pl. MΩ kimondva megaohm, jelentése 1 000 000 Ω.
- **2^{20}** neve **Mebi,** jele **Mi**, értéke 1 048 576, kimondva **megabinary**,
  Pl. 256MiB jelentése 256 Mebibájt, pontosan 268 435 456 bájt.
- **10^9** neve **Giga**, jele **G,** értéke 1 000 000 000, kimondva milliárd (Amerikában billió)
  Pl. 2 GW kimondva két gigawatt, jelentése 2 000 000 000 watt (egy nagy erőmű teljesítménye lehet ennyi)

- 2^{30} neve **Gibi**, jele **Gi**, értéke 1 073 741 824, kimondva gigabinary.

  Pl. a Pentium processzorok legfeljebb 4 GiB memóriát képesek címezni. Ennek oka, hogy 32 bites címvezetékük van. 32 bittel (bináris számjeggyel) 2^{32} féle állapot írható le. 2^{32}B = 4GiB = 429 4967 296 B.
- 10^{12} prefixumként **Tera,** jele **T** értéke 1 000 000 000 000, kimondva billió (Amerikában milliárd)
- 2^{40} prefixumként **Tebi**, jele **Ti**, értéke 1 099 511 627 776, kimondva terabinary
- 10^{15} prefixumként **Peta,** jele **P**, értéke 1 000 000 000 000 000, kimondva ezerbillió
- 2^{50} prefixumként **Pebi**, jele **Pi**, értéke 1 125 899 906 842 624, kimondva petabinary
- 10^{18} prefixumként **Exa,** jele **E**, értéke 1 000 000 000 000 000 000, kimondva trillió
- 2^{60} prefixumként **Exbi**, jele **Ei**, értéke 1 152 921 504 606 846 976, kimondva exabinary

Megjegyzés: Egynél kisebb bináris prefixumok nincsenek.

Megjegyzés 2: Egyedül a nagy K betű is bináris prefixumot jelent, míg a többi nagybetű magában (i nélkül) decimálisat.

Példák prefixumok használatára:

3 KB mit jelent?
Három kibibájt információt jelent, azaz pontosan 3 072 bájtot.
Nem bitet, hanem bájtot, mert a nagy B betű azt jelenti.

Megjegyzés: Írhatunk Kib helyett KB-ot, mert a nagy K betű bináris kilót jelentett önmagában. Sok adathordozón is így van feltüntetve annak kapacitása.

0,5 Kb jelentése 512 bit. Nem bájt, hanem bit, mert kis betű!

0,5 kb jelentése 500 bit.

5 MiB pontosan $5 \cdot 2^{20}$ bájt információ, azaz 5 242 880 bájt. Ez több, mint 5 MB, azaz 5 000 KB, hiszen a bináris Mega is több ezer bináris kilonál (1 024 K több, mint 1 000 K).

64 KiB jelentése 65 536 B. Megkaphatjuk úgy, hogy $64 \cdot 2^{10}$-t kiszámítjuk. De úgy is, hogy mivel $64 = 2^6$, $K = 2^{10}$, így $64 \cdot 1024 = 2^6 \cdot 2^{10} = 2^{6+10} = 2^{16} = 65536$.

Példák számításokra:

1.) Hány gibibájt adatot lehet felírni egy 4,7 GB kapacitású DVD-R lemezre?

Megoldás: 4,7GB = X GiB

$$X = 4,7 \cdot \frac{GB}{GiB} = 4.7 \cdot \frac{10^9}{2^{30}} = 4,3772161$$

Tehát 4,377 2161 GiB-nyi adat fér a 4,7 GB kapacitású lemezre.

2.) 15,7 GiB adat hány GB helyet foglal a merevlemezen? (A merevlemezek kapacitását GB-ban szokás megadni.)

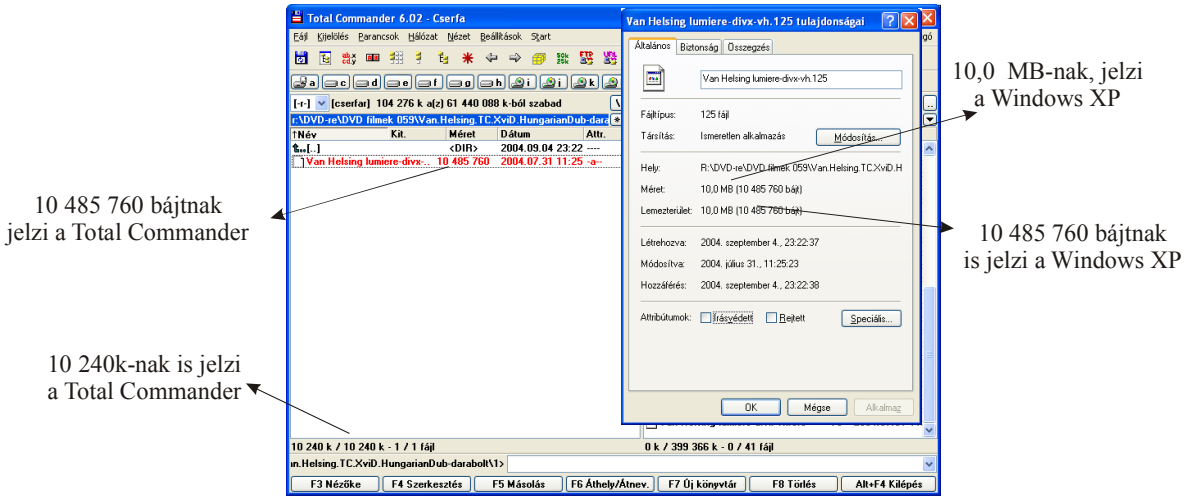
Megoldás: 15,7GiB = X GB

$$X = 15,7 \cdot \frac{GiB}{GB} = 15,7 \cdot \frac{2^{30}}{10^{9}} = 16,858$$

Tehát 16,858 GB helyet foglal a merevlemezen 15,7 GiB adat.

Természetesen 16,858 GB = 16 8578 KB, de 16,858 GB = 15,7GiB csupán.

Egy fájlnak háromféle méretét olvashatjuk az alábbi ábráról. Mekkora a mérete helyesen?



Megoldás: Pontosan valószínűleg a 10 485 760 bájt van megadva. Ezt 1024-gyel osztva 10 240-et kapunk, melyet tovább osztva 1024-gyel pontosan 10 a végeredmény.

Tehát a helyes fájlméret méret: 10 240KB, vagy 10MiB.

A Total Commander 6.02-es verziója nem mutatja rosszul a fájlméretet, csak éppen nem MiB-ban, sem MB-ban, hanem KB-ban olvasható a fájlméret. A Windows XP pedig a MiB-ban helyesen számított értéket MB-nak mutatja. Ember legyen a talpán, aki meg tudja jegyezni, melyik program melyik verziója miben érti a fájméretet!

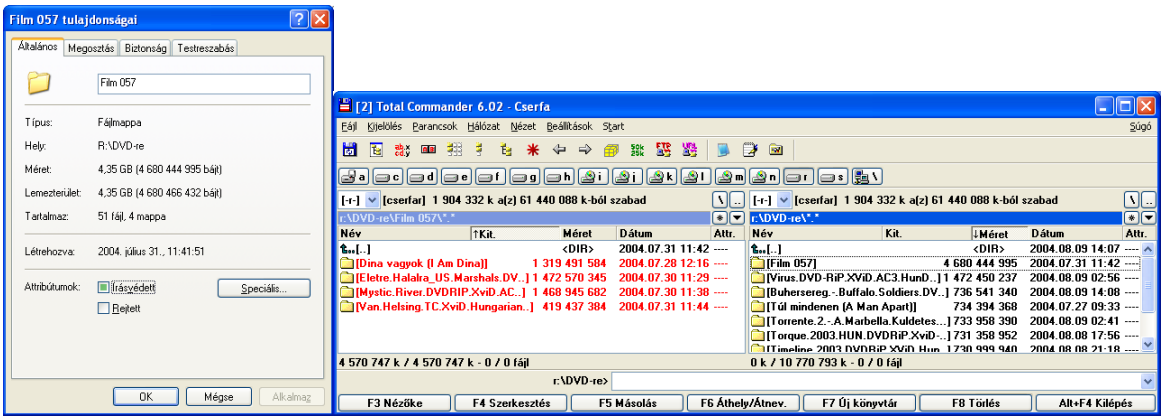
Ajánlott tanulmányozni még a következő rész 4. példáját!

Feladatok:

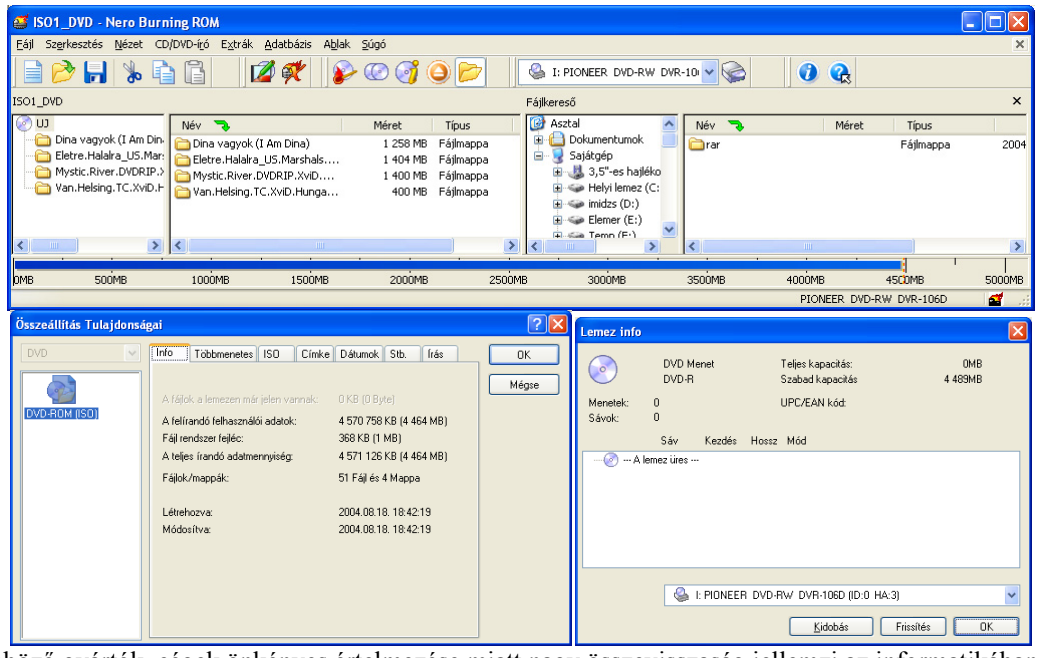
- Hány GiB 1200 MB?
- Fogalmazza meg saját szavaival, mit jelent, hogy egy floppylemez tulajdonképpen nem 1,44MB-os, hanem pontosan 1440 KB-os kapacitású! *(Erről lásd a következő rész 1. példáját!)*
- Hány 1,44 MB-os floppylemez tartalma írható fel egy 120MiB-os mágneslemezre? (Figyelem, az 1,44 MB-os floppylemez nem 1,44 MB-os, hanem 1440 KB-os!)
- Egy CD lemez 700 MiB-os kapacitású. Hány MB adat írható rá?
- Egy kétbájtos változó (a memóriában két bájttal van ábrázolva) hányféle értéket vehet fel? (2^{16})
- Hány bittel lehet ábrázolni az egymillió különböző lehetséges állapotot?
- Egy digitális aláírás kulcsa 2048 bájt hosszú. Hányféle variáció képzelhető el ilyen hosszúságú kulccsal? (Gondolja meg, nem véletlen, hogy hosszú ideig semmi esély nem lesz ilyen kulcsú hitelesítés meghamisítására.)

Elrettentő példák a helytelen prefixumok használatára:

1.) Egy floppylemezen bináris kilobájtokat, azaz az IEC ajánlás szerint kibibájtokat adnak meg. Ám azt írják egy lemezre, hogy kapacitása 1,44 MB, ami tulajdonképpen 1 440 KB kapacitást jelent (0,5KB·2 oldal·18 szektor ·80 track = 1 440KB). Tehát nem 1,44MB, hanem 1 440 KB, ami 1,406 25 MiB (binárisan), vagy 1,474 56 MB (decimálisan).Tehát egyetlen kifejezésben bináris és decimális számot is értenek, teljesen érthetetlenül. **Helyes lenne az 1,474 56 MB, vagy az 1,406 25 MiB, de hibás az 1,44 MB.**
2.) Egy CD-R lemez megadott kapacitása a gyártók szerint 650 MB, ami a valóságban azonban $650MiB = 650 \cdot K^2B$. Tehát összesen 681,5744MB adatot lehet erre a gyártók szerinti 650MB-os lemezre írni. Azaz 665 600 KB-ot.
3.) A merevlemezgyártók szinte mindegyike decimális Mega-, vagy Gigabájtban adja meg terméke kapacitását. Ugyanakkor minden fájlkezelő program KB-ban adja meg a méretet. **Ne csodálkozzunk, ha egy teljesen üres merevlemezen kevesebb a hely a fájlkezelő programok szerint, mint amit a gyártók megadnak.**
4.) Egy DVD-R lemez gyártók szerinti kapacitása 4,7 GB. Azonban erre csak 4,38GiB adatot lehet írni. Ezt a 4,38 GiB adatot a Nero CD író szoftver 4 489 MB-nak jelzi, amiben a 4 489 decimálisan-, míg az MB binárisan értendő (azaz MiB lenne).
Legyen egy fáj-összeállítás DVD-R írásra készen. Ennek a mappája a a Total Commander és a Windows XP fájlkezelése szerint 4 680 444 995 B, utóbbi szerint 4,35 GB is, majdnem eléri a maximális 4,38 GiB-ot.



Ez a fájlösszeállítás a Nero CD író szoftverben 4 571 126 KB-nak mutatkozik, amit a Nero már 4 464 MB-nak is mutat, így látható a lenti állapotlécen is. Ez az összeállítás majdnem teljesen kihasználja a DVD lemez Nero szerinti 4 489 MB-os, Windows szerinti 4,38 GB-os kapacitását, mint fentebb is láttuk.



A különböző gyártók, cégek önkényes értelmezése miatt nagy összevisszaság jellemzi az informatikában használt prefixumokat, méreteket. A hasonló jelölés is megtévesztő. Ez már a megával kezdődött, mert a decimális mega is nagybetűs M, itt nem lehetett élni azzal, amivel a Kilónál, hogy a binárist a decimálistól úgy különböztetjük meg, hogy nagybetűvel írjuk. A kis m betű meg millit jelent, így erre sem lehetett menni. Így vállalva a

kétértelműséget a decimális megát is, a bináris megát is nagy M betűvel jelölték, és az M betűből nem lehetett tudni, melyikre is gondolnak, csak abban reménykedtek, az informatikában bináris megára gondol mindenki. Igen ám, de jöttek a gyártók, és önkényesen decimális megával kezdték megadni gyártmányaik méretét, hogy nagyobbnak tüntethessék fel gyártmányaikat. Hasonlóan bántak a megánál nagyobb prefixumokkal is, így végül teljes összevisszaság alakult ki, összevissza keverik az azonos nevű bináris és decimális prefixumokat, gyakran egy adaton belül is (gondoljunk a floppyra, ahol a MB jelentése: decimális kilo szorozva bináris kilóval).

Mindig meg kell tehát vizsgálni, bináris, vagy decimális prefixumról van-e szó. **Mi pedig ragaszkodjunk az IEC ajánláshoz**, hogy elkerüljük a félreérthetőséget.

Elvárt a prefixumok helyes használata.

A bináris és decimális prefixumok használata, és helyes használata kötelező! Ne mondjunk olyan mennyiséget, hogy 4 680 444 995 (négymilliárd-hatszáznyolcvan- és félmillió) bájt, mondjuk 4,38 Gibibájtnak! Ne mondjunk 5000 Megabájtot, mondjuk öt Gigabájtnak!

A kötelezően ismert számok

Kötelező a következő kettes számrendszerbeli számok decimális értékeinek ismerete:

$2^4 = 16$
$2^8 = 256 = ¼$ K
$0{,}5$ K $= 2^9 = 512$
$2^{10}=1024 =$ K
$2^{16} = 65536$

Ezeket kérem megtanulni, fejből, gondolkodás nélkül tudni! Használatukat is tessék begyakorolni!

Ellenőrző kérdések az információ témaköréből:

- **Határozza meg:** Mi az információ? *Milyen főbb elemei vannak az információnak?* Melyik főbb elemeit vesszük részletesebben? Mi a szintaxis? Mi a statisztika? Mi a karakter? Mi az információ elemi egysége? *Mit tekintünk az adatnak?* Mit nevezünk kódnak? Mit értünk kódolás alatt? Mit értünk dekódolás alatt? Milyen kódokról volt szó eddig?
- **Válaszolja meg:** Miért a kettes számrendszert választották az információfeldolgozásban? Milyen több-bites információegységeket ismer? Milyen decimális prefixumokat ismer? Milyen bináris prefixumokat ismer? Mi az a bit? Mit értünk szó alatt? Mi a bájt? Mi a triád? Mi a tetrád? A zavar növeli, vagy csökkenti a jel információtartalmát? Nevezzen meg legalább egy alfanumerikus kódot, és mondja meg, hány bittel szoktak ábrázolni benne egy karaktert!
- **Gondolkodjon** (érdekességként.)**:** *Ki volt Shannon? Körülbelül hány bit emlékezete lehet az átlagos emberi agynak? Hány bit kell átlagosan (körülbelül) egy betű hordozta információhoz?*
- Számítsa ki, mondja meg pontosan: Pontosan hány bájt 1 KB? Pontosan hány MB 1MiB? 2 Gib pontosan hány bit? 2 MiB pontosan hány bájt? A CD-ken 1 másodpercnyi zene 150 KB-nyi adatnak felel meg. Hány MB adat írható egy 80 perc tárolóképességű írható CD-re? És hány MiB?
- 2 GiB adat hány GB helyet foglal a merevlemezen? (A merevlemezek kapacitását GB-ban szokás megadni.)

II. Kódok és aritmetika

Numerikus kódnak az információ számjegyekkel történő ábrázolására alakító szabályt nevezzük. A közéletben a legelterjedtebb a decimális számok használata, gondolkodásunkhoz is ez áll a legközelebb. A decimális kód egyszerűen a decimális számok használata.

A legkedvezőbb alapszám c. fejezetben *(II.7. oldal)* látni fogjuk, a kettes számrendszer használatának akkora előnye van, hogy részletesebben csak a bináris kódokat tanulmányozzuk.

Aritmetika alatt a számok tanát, a számábrázolást, és a számokon végzett matematikai műveleteket értjük. A számítógépeket az aritmetika miatt nevezik **számító**gépnek, noha sokkal több feladatot végeznek, mint a számológépek, ezért különböznek azoktól (ezért is számító-, és nem számológépek). A számítógépek processzorainak egyik fő egysége az ún. aritmetikai-logikai egység, mely működéséhez megértéséhez most fektetjük le az alapokat.

Számrendszerek

Helyiértékes számok

- Egy n számú egész számjegyet és h számú törtrész számjegyet tartalmazó R (radix) alapú számjegy jelentése a következő: $N_R = \pm \sum_{k=-h}^{n-1} A_k R^k$

Ez nagyon fontos definíciós képlet, ezzel könnyen ki tudjuk számítani egy R alapú szám decimális alakját.

A definíciós képlet szóval kimondva: szumma A index k szor R a k-adikon, ahol k -h-tól n-1-ig minden egész értéket felvesz. Ez lényegében egy n + h elemű tömb, ahol k az indexváltozó. Az összeg egy R alapú n + h jegyű helyiértékes számjegy, ahol az egész jegyek száma n, a törtrész jegyeinek száma h.

A definíciós képlet egész része: $N_E = A_{n-1}R^{n-1} + A_{n-2}R^{n-2} + ... + A_1R^1\ A_0R^0$

Szummás alakban az egész rész az egész rész: $N_E = \sum_{k=0}^{n-1} A_k R^k$

A definíciós képlet tört része: $N_T = N_T = A_{-1}R^{-1} + A_{-2}R^{-2} + ... + A_{-h+1}R^{+h+1} + A_{-h}R^{-h}$

Szummás alakban: $N_T = \sum_{k=-h}^{-1} A_k R^k$

Természetesen $N_E + N_T = \sum_{k=0}^{n-1} A_k R^k + \sum_{k=-h}^{-1} A_k R^k = \sum_{k=-h}^{n-1} A_k R^k = N_R$

Az egész rész és a törtrész közé egy vesszőt teszünk, a törtes vesszőt.
(Amerikában és sok országban törtespontot használnak). A törtesvesszőtől balra állnak az egész értéket-, jobbra pedig a törtet ábrázoló számjegyek.

Az N szám az R alapú számrendszerben tehát a következő alakban írható fel:

$\mathbf{N_R = A_{n-1}A_{n-2}...A_1A_0,A_{-1}A_{-2}...A_{-h+1}A_{-h}\ R}$

A kifejezésben alsó indexben szereplő R jelentése: R alapú helyiértékes szám.

Az egész rész $N_E = A_{n-1}A_{n-2}...A_1A_0\ _R$ A törtrész $N_T = A_{-1}A_{-2}...A_{-h}+1A_{-h\,R}$

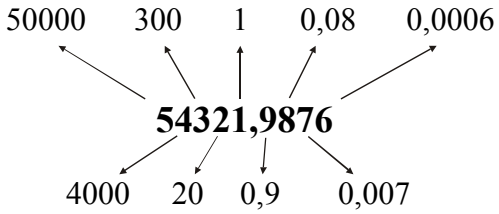
Pl. $4356{,}76_8$

A helyiértékes számoknál tehát nem kell jelölni az alap hatványait, azok a számjegyek előfordulási helyéből következik. Az n jegyű egész számok legnagyobb helyiértéke az n –1 szám, legkisebb helyiértéke a 0.
Például egy 12 jegyű egész szám legnagyobb helyiértéke 11, a legkisebb helyiértéke 0.

Példa:

$54321{,}9876_{10}$ jelentése:
decimális, tehát R = 10, öt egész jegyű, azaz n = 5 és négy törtjegyű, azaz k = 4, értéke a definíciós képlet szerint: $5\cdot10^4+4\cdot10^3+3\cdot10^2+2\cdot10^1+1\cdot10^0+9\cdot10^{-1}+8\cdot10^{-2}+7\cdot10^{-3}+6\cdot10^{-4}$

$54321{,}9876_{10}$ jelentése:



Pl. 1011_2 egy kettes számrendszerbeli számjegy, jelentése:
$1011_2=1\cdot2^3+0\cdot2^2+1\cdot2^1+1\cdot2^0$. Itt n = 4, azaz négyjegyű a szám.
A_3 = **1**, A_2 = **0**, A_1 = **1**, A_0 = **1**. Ezt összeírva kapjuk 1011-et, ahol az alapszám 2.
Röviden tehát ez a szám: $\mathbf{1011}_2$

A legtöbb esetben a decimális számokat semmivel nem jelöljük, a jelöletlen számok mindig decimálisak.

Pl. 1097 jelentése: $1\cdot10^3+0\cdot10^2+9\cdot10^1+7\cdot10^0$. Az alapszám 10.
A_3 = **1**, A_2 = **0**, A_1 = **9**, A_0 = **7**. Röviden (összeírva) tehát ez a szám **1097**.
Ez is négyjegyű szám, azaz n = 4, a legnagyobb helyiérték a 3.

Átszámítás decimális számrendszerbe

A helyiértékes számok definíciós képletével nagyon könnyű kiszámítani a tetszőleges számrendszerben adott számot, csak be kell helyettesíteni, és elvégezni a képlet műveleteit (hatványozás, szorzás és összeadás). Magyarázat helyett példákkal mutatjuk meg az átszámítást.

Példák decimálisra átszámításra

Számítsuk át $4356{,}76_8$-ot decimálisra

$4356{,}76_8$ oktális (8 alapú), 4 egész jegyű és 2 törtjegyű szám.

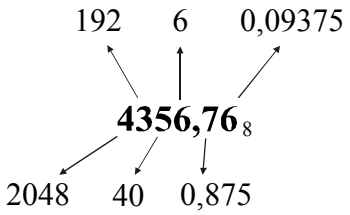
A definíció szerint értéke decimálisan: $4\cdot8^3+3\cdot8^2+5\cdot8^1+6\cdot8^0+7\cdot8^{-1}+6\cdot8^{-2}$

Decimálisra tagonként kiszámítva:

$4356{,}76_8=4\cdot512+3\cdot64+5\cdot40+6+7\cdot0{,}125+6\cdot0{,}015625=2286{,}96875$

Tehát $4356{,}76_8$ decimálisan: $\mathbf{4356{,}76_8=2286{,}96875_{10}}$

$4356{,}76_8$ átszámítása decimálisra:



Számítsuk át $10110{,}011_2$-et decimálisra

$10110{,}011_2$ bináris (2 alapú), 5 egész jegyű és 3 törtjegyű szám.

A definíció szerint értéke decimálisan: $1\cdot 2^4+1\cdot 2^2+1\cdot 2^1+1\cdot 2^{-2}+1\cdot 2^{-3}$.

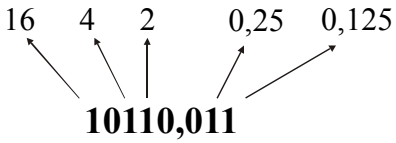
Figyeljük meg, a 0-val szorzott tagokat nem tüntettem fel, ezután sem fogom.

Decimálisra tagonként kiszámítva:

$10110{,}011_2 = 16+4+2+0{,}25+0{,}125=22{,}375$

Tehát $10110{,}011_2$ decimálisan: **$10110{,}011_2 = 22{,}375_{10}$**

$10110{,}011_2$ átszámítása decimálisra:



Számítsuk át $3F60{,}D6_{16}$-ot decimálisra

$3F60{,}D6_{16}$ hexadecimális (16 alapú), 3 egész jegyű és 2 törtjegyű szám.

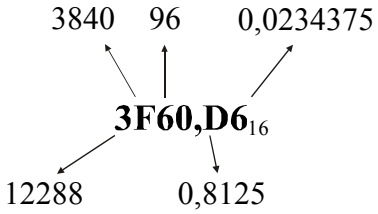
A definíció szerint értéke decimálisan: $3\cdot 16^3+15\cdot 16^2+6\cdot 16+13\cdot 16^{-1}+6\cdot 16^{-2}$

Decimálisra tagonként kiszámítva:

$3F60{,}D6_{16} = 3\cdot 4096+15\cdot 256+6\cdot 16+13\cdot 0{,}0625+6\cdot 0{,}00390625=16224{,}8359375$

Tehát $3F60{,}D6_{16}$ decimálisan: $3F60{,}D6_{16} = 16224{,}8359375_{10}$

$3F60{,}D6_{16}$ átszámítása decimálisra:



Hexadecimálisból binárisra- és visszaszámítás

Minden egyes hexadecimális számjegy helyettesíthető egy tetráddal. **Hexadecimálisból úgy kapunk bináris számot, hogy a minden egyes hexadecimális számjegynek megfelelő tetrádot formálisan egymás után leírjuk.**

Hasonlóan egyszerű a bináris számokat hexadecimálissá alakítani, ha tetrádokká tudjuk írni, csak leírjuk a tetrádoknak megfelelő hexadecimális számjegyeket. Itt azonban többnyire tetráddá kell kiegészíteni a bináris számokat, mégpedig az egész részt is, a törtrészt is. Ezt úgy végezzük, hogy négyes csoportokat jelölünk ki a kettedes ponttól, mégpedig az egész részen a kettedes ponttól balra-, a tört részen pedig jobbra haladva. Ha a végül az utolsó csoport nem négyes, azt a haladási irány szerint 0-kal egészítjük ki, úgy, négyes csoportra bővüljön. Ezután az így kapott tetrádok hexadecimális megfelelőit egyszerűen leírjuk.

Ezt példákkal mutatom be:

Példa hexadecimálisból binárisra- és visszaszámításra

Hex	Tetrád
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Alakítsuk át a fent már szerepelt $3F60,D6_{16}$-t binárisra!

Írjuk fel egymás után a 3, az F, a 6, a 0 tetrádját, a törtesvesszőt, majd a D és a 6 tetrádját formálisan, gondolkodás nélkül, és kész vagyunk.
Ezek az érthetőség kedvéért előbb kötőjellel elválasztva (a táblázat alapján): 0011-1111-0110-0000,-1101-0110. Ezt egybeírva kapjuk: 0011111101100000,11010110. Ebből elhagyva az első két, és az utolsó felesleges 0-t 11111101100000,1101011-t kapunk.
Tehát $3F60,D6_{16} = 11111101100000,1101011_2$

Feladat: A *Példák decimálisra átszámításra* alfejezetben kiszámítottuk, hogy $3F60,D6_{16}$ decimálisan: $3F60,D6_{16} = 16224,8359375_{10}$
Ellenőrizze, hogy 11111101100000,11010110 értéke decimálisan szintén 16224,8359375!

Példa binárisból hexadecimálisra alakításra

Számítsuk át 11101000000110000,01011101-t hexadecimálisra!
Előbb átalakítjuk négyes csoportokra, az egész részen is, a törtrészen is különkülön, a kettedes ponttól haladva: 1-1101-0000-0011-0000,1011-101. A kapott nem négyes csoportokat 0-kal kipótolva (a törtnél utána, az egésznél elé írva) kapjuk: 0001-1101-0000-0011-0000,1011-1010. Az így kapott a tetrádok a hexadecimális megfelelői összeírva: 1D030,BA.

Tehát $11101000000110000,1011101_2 = 1D030,BA_{16}$

Látható, csak a megfelelési szabály formális betartása eredményt hoz, különösebb gondolkodás nélkül.

Oktálisból binárisra- és visszaszámítás

Mivel az oktális számok számjegyei egy-egy triáddal helyettesíthetők, a fenti elveket itt is alkalmazhatjuk. Oktálisból binárisra alakításkor formálisan leírjuk az oktális számjegyeknek megfelelő triádokat, és készen vagyunk. Binárisból oktálisra a fentiekhez hasonlóan előbb triádokra csoportosítjuk a bináris számot, ha kell, ki is bővítjük triáddá a szélső csoportokat. Ezután formálisan leírjuk a kapott triádoknak megfelelő oktális karaktereket.

Mivel számunkra az oktális számok kevésbé fontosak, az átalakításokat nem mutatom be példákkal, és ilyen feladatot sem adok, de aki el kíván mélyedni e témában, könnyen elvégezheti az átalakításokat.

Átszámítás decimális számból más számrendszerbe

Ha nem decimális rendszerben adott számot kell átalakítani, előbb alakítsuk decimális számmá a definíciós képlet alapján, mivel nem tudunk gyakorlottan szorozni és osztani tetszőleges számrendszerben. Ezután a kapott decimális számot alakítjuk az új rendszerbe, az alábbiak szerint (minden számrendszerre értendő, nem csak decimálisra).

- **Az egész számok átszámítása** sorozatos osztással érhető el, úgy, hogy a maradékok adják az új számrendszer jegyeit a törtesvesszőtől haladva a nagyobb helyiértékek szerint.
- **Törtszámok átszámítása** sorozatos szorzással történik úgy, hogy az egészek adják az új tört jegyeit, szintén a törtesvesszőtől haladva az egyre kisebb helyiértékek szerint.

Megjegyzés: Ez nem vonatkozik a binárisból oktálisra, és viszont-, valamint hexadecimálisról binárisra és viszont alakításkor, mivel ezek különösen egyszerűek (lásd előző alfejezet). Ezeknél elég az oktális karakterek helyett a triádok, és a hexadecimális karakterek helyett a tetrádok formális felírása.

Példák a fenti szabályok alkalmazására:

Mennyi $312{,}375_{10}$ kettes számrendszerben?

312	: 2	0		2 · 0,375	= 0,75		0
156	: 2	0		2 · 0,75	= 1,5		1
78	: 2	0		2 · 0,5	= 1,0		1
39	: 2	1					
19	: 2	1					
9	: 2	1					
4	: 2	0					
2	: 2	0					
1	: 2	1					

$312_{10} = 100111000_2$ és $0{,}345_{10} = 0{,}011_2$

Tehát $312{,}375_{10} = 100111000{,}011_2$

Mennyi $93{,}835\,937\,5_{10}$ binárisan?

95	: 2	1		2 · 0,8359375	= 1,671875		1
47	: 2	1		2 · 0,671875	= 1,34375		1
23	: 2	1		2 · 0,34375	= 0,6875		0
11	: 2	1		2 · 0,6875	= 1,375		1
5	: 2	1		2 · 0,375	= 0,75		0
2	: 2	0		2 · 0,75	= 1,5		1
1	: 2	1		2 · 0,5	= 1		1

Tehát $93{,}835\,937\,5_{10} = 1011111{,}1101011_2$

Azon ne csodálkozzunk, ha nagyon sok, akár végtelen sok számjegyet kapunk a törtrész kiszámításakor, a fenti példák kifejezetten olyanok (az előző példák törtrészei), hogy véges jegyű eredményt kapjunk.

Mennyi 153,706 018 518 5_{10} hatos számrendszerben?

153	: 6	3	6 · 0,7060185185	=	4,2361111111		4
25	: 6	1	6 · 0,2361111111	=	1,4166666666		1
4	: 6	4	6 · 0,4166666666	=	2,5000000000		2
			6 · 0,500000000	=	3		3

Tehát $153{,}7060185185_{10} = 413{,}4123_6$

Ellenőrzés a helyiértékes törtszámok definíciójának felhasználásával:

$413{,}4123_6 = 4\cdot 6^2 + 1\cdot 6 + 3 + 4\cdot 6^{-1} + 1\cdot 6^{-2} + 2\cdot 6^{-3} + 3\cdot 6^{-4} =$

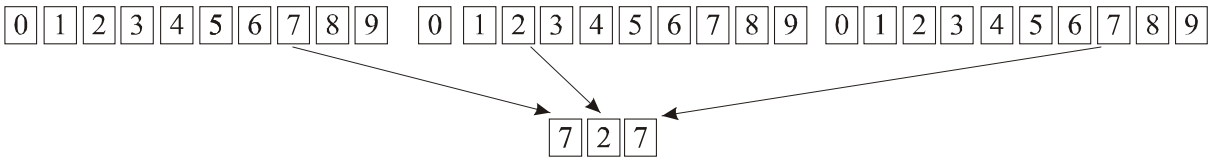
$=144+6+3+0,\dot{6}6\dot{6}+0,02\dot{7}7\dot{7}+0,009\dot{2}5\dot{9}+0,0023\dot{1}4\dot{8}=153,7060185185$

(A decimális számrendszerbeli számokat nem indexeltem, a felső pontok között végtelen szakasz van jelölve szokás szerint.).

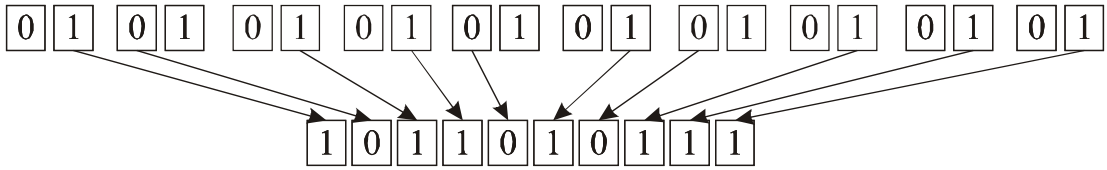
A minimális szerkezeti elemek száma

Keressük meg, legalább mennyi hardver elem kell egy adott R alapú, n jegyű szám ábrázolásához.

Példa decimális számra: Ha 0-999-ig terjedő számokat szeretnénk cserélhető táblácskákkal ábrázolni, helyiértékenként 10-10-10 db, táblácskát kell készítenünk. Minden helyiértékhez 10-et (0-9-ig), összesen 30 db-ot. A harminc db táblácskából így rakhatunk ki pl. 727-et:



Példa bináris számra: Kettes számrendszerben 2^{10}-féle szám ábrázolásához csak 20 db táblácskát kell készítenünk, 10 db 0-t, és 10 db 1-et, amivel 0-1 11111 1111-ig (decimális értékben 0-1023-ig) minden számot ábrázolni tudjuk. Rakjunk ki a húsz táblácskánkkal az előbbi decimális 727-et binárisan, azaz 1011010111_2-et (727_{10}):



A példa alapján mondhatjuk, a háromjegyű decimális számok minimális szerkezeti elemeinek száma számjegyenként 10, n jegyűhöz n·10. Hardverből tehát legalább ennyi fizikailag létező elem (pl. különböző táblácska, lámpa, huzal, vagy egyéb) kell a decimális számok tényleges ábrázolásához. Hasonlóan az n jegyű bináris számok minimális szerkezeti elemeinek száma n·20. Hardverből tehát legalább ennyi fizikailag létező elem (pl. különböző táblácska, lámpa, huzal, vagy egyéb) kell a bináris számok ábrázolásához.

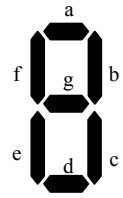
Megállapíthatjuk:

R alapú n jegyű szám minimálisan szükséges szerkezeti elemeinek száma R·n.

Ha az R alapú n jegyű szám első számjegye nem R, csak k-féle lehet, akkor k + R (n-1)

Erről lásd még a fél digit (számjegy) fogalmát a Hardver Gyakorlat könyvünk Műszerek tulajdonságai fejezetében

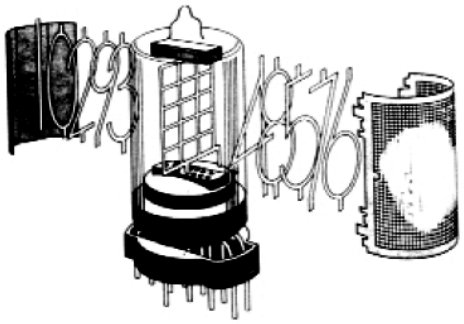
Érdekesség: Nem csak táblácskákkal, hanem szegmensekkel is lehet számjegyeket ábrázolni. A decimális számokat gyakran hétszegmenses kijelzővel ábrázolják. A hét szegmens tulajdonképpen hét kétállapotú szerkezeti elem (szegmens), így akár 2^7 féle ábrát ábrázolhatna. Amiből csak 10-et használnak ki, mert 10-féle decimális szám van. A hétszegmenses kijelzőkkel, és kódolásukkal találkozunk, és tervezni is fogjuk a dekóderüket (binárisból 7 szegmensesre kódoló áramkör) a *BCD kódból 7 szegmenses kijelzőre dekóder tervezése* c. fejezetben *(IV.21. old.)* A jobb oldali ábrán meg is jelöltem a hét szegmenset az abc betűivel:



Fogunk még vizsgálni más szerkezeti elemekkel megadott számok kijelzését is, pl. a dobókocka, vagy a dominó számait a *Kódoló és dekódoló áramkörök* c. fejezetben *(V.1. old.)*

Fizikailag mind a 10 számjegyet pl. az ún Nixie csövekben alakították ki. A Nixie csövekben a számjegyeknek csakugyan 10 szerkezeti eleme van. Ezek a számjegyek a csövekben meg is figyelhetők. Mindig csak az világít, amelyiket meg kell jeleníteniük, de a többi is meglátható, ha jobban megnézzük. Hasonló működési elvű kijelzőcsövek a mai napig működnek különféle készülékekben.

A gyakorlatban nagyon sokféle kijelző van, melyek szerkezeti elemeinek száma és sokféleségük miatt működésük felsorolhatatlan, mi csak a legegyszerűbbeket vesszük alaposabban figyelmünkbe.



Nixie cső "robbantott" rajza

A legkedvezőbb alapszám

Egyjegyű számok esetén annyiféle állapotot ábrázolhat a számjegy, amennyi az alapszáma. Egyjegyű szám esetén tehát R számú szerkezeti elem kell, ennyiféle állapotot írhat le.

Pl. egy decimális számjegy 10 féle állapotot írhat le, és 10 féle szerkezeti elem kell. Egy a 0-nak, egy az 1-nek, egy a 2-nek, ... végül egy a 9-nek.

Többjegyű számok esetén kedvezőbb a helyzet. **R alapú, n jegyű számok R^n féle állapotot írhatnak le. Szerkezeti elemeinek száma azonban csak R·n.** A többjegyű számok esetén az ábrázolható állapotok száma n-nel exponenciálisan növekedik, míg a szerkezeti elemeinek száma csak egyenes arányban.

Pl. kétjegyű decimális számok állapota 100 féle lehet, míg szerkezeti elemeinek száma csak 20. Hatjegyű decimális számjegy esetén már egymillió különböző számot kaphatunk 60 db szerkezeti elemmel.

Keressük meg, melyik a legkedvezőbb alapszám! Nyilvánvalóan az, amelyik alkalmazásakor egy adott számú lehetséges állapotszámhoz a legkevesebb szerkezeti elemszám kell. Ehhez nagyobb számot érdemes vizsgálni, hiszen kis számoknál kisebb az előnye a helyiértékes számoknak, egy jegy esetén el is tűnik.

Tekintsünk egy 10^3 körüli számot különböző számrendszerekben:

	Alapszám	Helyértékek száma	Ábrázolható különböző számjegyek (állapotok) száma	Szerkezeti elemek száma
Bináris	2	10	$2^{10} = 1\,024$	20
3-as	3	19	$2 \cdot 3^6 = 1\,458$	21
Oktális	8	10	$2 \cdot 8^3 = 1\,024$	27
Decimális	10	9	$10^3 = 1\,000$	30
Hexadec.	16	2	$4 \cdot 16^2 = 1\,024$	37

Tekintsünk egy 10^9 körüli számot különböző számrendszerekben:

	Alapszám	Helyértékek száma	Ábrázolható különböző számjegyek (állapotok) száma	Szerkezeti elemek száma
Bináris	2	30	$2^{30} = 1\ 073\ 741\ 824$	60
3-as	3	19	$3^{19} = 1\ 162\ 261\ 467$	57
Oktális	8	10	$8^{10} = 1\ 073\ 741\ 824$	80
Decimális	10	9	$10^9 = 1\ 000\ 000\ 000$	90
Hexadec.	16	7	$4 \cdot 16^7 = 1\ 073\ 741\ 824$	117

Figyeljük meg, a decimális számrendszerben másfélszer annyi szerkezeti elemre van szükség ugyanakkora mennyiség ábrázolásához, mint bináris számrendszerben. Látható, a legkevesebb szerkezeti elem a hármas számrendszerhez kell, de majdnem ilyen kedvező a kettes számrendszer.

A kettes számrendszernek azonban olyan előnye van, amely szinte kizárólagossá teszi használatát. Ez pedig a két egymástól távol eső, jól megkülönböztethető állapot, mely fizikai mennyiségekkel jól ábrázolható, és a zavaroktól, torzulásoktól éppen a könnyű felismerhetőség miatt a lehető leginkább független.

Aritmetikai műveletek bináris számokkal.

Mi csak az egész számokra értelmezett műveleteket vesszük, és azokban is csak az összeadást, megmutatva, hogy a kivonás, a szorzás és az osztás hogyan végezhető el átalakításokkal és összeadásra. Összeadással és átalakításokkal ugyanis minden matematikai műveletet el lehet végezni.

Bináris összeadás pozitív operandusoknál

Legyen két tetszőleges alapú számjegyünk, A és B. Legyen összegük S (szumma), és az átvitel C (carry).

Átvitel akkor jelentkezik, ha az adott helyiértéken túlcsordul a szám, ekkor az átvitel a következő, magasabb helyiértékhez adódik hozzá. Túlcsordulás akkor jelentkezik, ha a két szám összege nagyobb lenne, mint az R-1, azaz a maximálisan ábrázolható egyjegyű szám. Tízes számrendszerben, tehát ha 9-nél nagyobb, bináris számrendszerben, ha 1-nél nagyobb. Két szám összegekor minden számrendszerben legfeljebb 1 lehet az átvitel.

Az alábbi táblázatban nézzük a bináris A és B összegét.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Ez az igazságtáblája az ún. fél összeadónak. (*A fél összeadó* c. fejezet*, V.5.* old.*)*

Több bites számok összegzésekor bitenként képződhet átvitel, ami a magasabb helyiértéken levő bitekhez adódik hozzá. Így olykor bitenként akár három darab egyes összegét is kell képezni. Később,
A teljes összeadó c. fejezetben *(V.5. old.)* majd erre a műveletre készítjük az ún. teljes összeadót.

Példaképpen adjunk össze két számot decimálisan, és binárisan is:

		0	6	3				0	0	1	1	1	1	1	1
		0	4	5				0	0	1	0	1	1	0	1
C	=	1	0		C	=	0	1	1	1	1	1	1		
S	=	1	0	8	S	=	0	1	1	0	1	1	0	0	

Átvitel ott keletkezik, ahol egynél több 1-est kell összeadni (jelöltem)

Az összeg akkor 1, ha páratlan 1-est adunk össze.

A keletkező átvitelt a keletkezési helyénél eggyel magasabb helyiértékhez írjuk

Az egyes helyiértékeken levő szumma az azonos helyiértéken álló operandusok jegyeinek, és az előtte levő helyiértéken képződő átvitelnek az összege. $S_i = A_i + B_i + C_{i-1}$.
Az adott helyiértéken képződő átvitelek az eggyel magasabb helyiértékhez adódnak.

Bináris kivonás

A kivonás pozitív operandusok esetén is kivezethet a pozitív számok halmazából. Ezért szükséges a negatív előjel ábrázolása (kódolása) bináris számoknál. Ezenkívül jó lenne, ha a kivonást is összeadásra vezethetnénk vissza. Ehhez találtak is megoldást, a komplemens számábrázolást. Alább látni fogjuk, hogy a komplemensekkel a kivonás is összedással végezhető el.

Bináris kódok

A legkedvezőbb alapszám c. fejezetben láttuk, hogy a kettes számrendszer használatának akkora előnye van, hogy mi **ezután a Digitális technika fejezetein belül csak a bináris kódokat tanulmányozzuk**. Minden mást, még a decimális számokat is bináris kóddal ábrázolunk.

Bináris előjeles számok

A negatív számok szokásos jelölése a negatív előjel karakter. A bináris számoknál azonban nem lenne jó előjel karaktert is használni, mert le kéne mondanunk a csak kétféle (0 és 1) karakter alkalmazásáról, ekkor a bináris számábrázolás legnagyobb előnyét adnánk fel. Ezért bináris számoknál bináris számjegyet alkalmazunk az előjel ábrázolására is: a következőképpen: ha a szám első bitje 1, negatív számról van szó, ha 0, akkor pozitív számról. Így az előjeles bináris számok is csak 0-ás és 1-es karakterekből állnak.

Többféleképpen lehet így előjeles számokat kódolni, mi csak kettőt említünk:

- **1-es komplemens kód** (inverz kód): Úgy kapjuk, ha egy bináris számban az előjelbitet kivéve minden 1-est 0-ra, és minden 0-át 1-esre cserélünk.
- **2-es komplemens kód** (komplemens kód): Úgy kapjuk, ha a szám inverzéhez 1-et hozzáadunk.

Számábrázolás komplemens kóddal

- Pozitív számot 0-s előjelbittel, és abszolút értékével ábrázoljuk
- Negatív számot 1-es előjelbittel, és abszolút értékének komplemensével kell ábrázolni
- Az előjelbitekkel ugyanúgy kell műveletet végezni, mint az egyéb számjegyekkel

A 2-es komplemens kód alkalmazása gyakoribb, mert egyszerűbb vele matematikai műveletet végezni.

Összeadás és kivonás 1-es komplemens kódban

- Az operandusokat össze kell adni
- A keletkező átvitelt az összeghez kell adni
- Ha az eredmény előjelbitje 0, az eredmény pozitív, és abszolút értéke a tényleges eredmény
- Ha az eredmény előjelbitje 1, az eredmény negatív, és értéke a tényleges összeg abszolút értékének 1-es komplemense

Összeadás és kivonás 2-es komplemens kódolással

- Az operandusokat össze kell adni
- Az összeadásnál keletkező átvitel elhanyagolandó
- Ha az eredmény előjelbitje 0, az eredmény pozitív, és értéke a tényleges összeg abszolút értékét adja
- Ha az eredmény előjelbitje 1, az eredmény negatív, és értéke a tényleges összeg abszolút értékének 2-es komplemensét adja.

Egyszerű bináris kód

Ebben a kódban az ábrázolandó mennyiségeket egyszerűen bináris számokkal ábrázoljuk.

II-1. Egyszerű bináris kód

Két bites

Dec.	A	B
0	0	0
1	0	1
2	1	0
3	1	1

3 bites

Dec.	A	B	C
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

A legnagyobb helyiértékű bitet A-val jelöltük

4 bites

Dec.	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

4 bites magas szintű (1 = H)

Dec.	A	B	C	D
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

4 bites alacsony szintű (1 = L)

Dec.	A	B	C	D
0	H	H	H	H
1	H	H	H	L
2	H	H	L	H
3	H	H	L	L
4	H	L	H	H
5	H	L	H	L
6	H	L	L	H
7	H	L	L	L
8	L	H	H	H
9	L	H	H	L
10	L	H	L	H
11	L	H	L	L
12	L	L	H	H
13	L	L	H	L
14	L	L	L	H
15	L	L	L	L

Az áramkörökben nem számok, hanem vezetékek vannak. **A vezetékek feszültségei jellemzik logikai állapotukat.** A vezetékek feszültségszintje kétféle lehet:
Vagy H (high, magas) szintű, amikor egy meghatározott H_{min} feszültségnél csak magasabb-, vagy L (low, alacsony) szintű, amikor egy meghatározott L_{max} feszültségszintnél csak alacsonyabb lehet. A H és az L jelenti a 0-át és az egyet, vagy fordítva.

- **Magas szintű logikáról** akkor beszélünk, ha a H érték logikai 1, és az L érték a logikai 0.
- **Alacsonyszintű logikáról** beszélünk, amikor, a H 0-át jelent, és az L 1-et

Gray kód

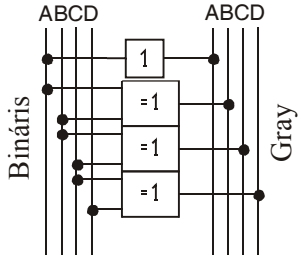
Dec.	A	B	C	D
0	0	0	0	0
1	0	0	0	1
3	0	0	1	1
2	0	0	1	0
6	0	1	1	0
7	0	1	1	1
5	0	1	0	1
4	0	1	0	0
12	1	1	0	0
13	1	1	0	1
15	1	1	1	1
14	1	1	1	0
10	1	0	1	0
11	1	0	1	1
9	1	0	0	1
8	1	0	0	0

A Gray kódot tükrözéssel lehet előállítani. A jobb felső saroktól kezdjük, felvesszük a 0-t és az 1-et. Ez alá vonalat húzunk (a táblázatban kettős vonal). A kettős vonal, mint tükör alá tükrözzük a felette levőket, majd a felső rész elé végig 0-át, az alsó elé 1-et írunk. Ezután az így kapott összes elem alá húzunk vonalat. Majd ezeket tükrözzük, majd a felső rész elé végig 0-át, az alsó elé 1-et írunk, s így tovább.

Gray kód képzése: a legnagyobb helyiértékű oszlop megegyezik a bináris kódéval
Az i-edik helyiérték oszlop elemei a bináris (i+1)-edik oszlop XOR i-edik oszlop függvénnyel állnak elő soronként. *(A XOR-t lásd a Kizáró VAGY (XOR, antivalencia) kapu c. fejezetet, III.11. old.)*

Így pl. a Gray kódú B helyiértékoszlop értékei a bináris kódú A XOR B függvénnyel állnak elő soronként. A Gray C helyiértékoszlop értékei pedig a bináris kód oszlopaiból a B XOR C függvénnyel állnak elő soronként. Ezért XOR kapukkal tudunk binárisból Grayba átkódoló kapcsolást készíteni.

Binárisból Gray kódba kódoló áramkör



A Gray kódnak az a sajátossága, hogy mindig csak egy számjegy változik meg benne, így elfordulások, elmozdulások jellemzésére különösen alkalmas. Gray kódban van peremezve a később tanult Karnaugh tábla is, így abban egymás mellé kerülnek azok a logikai állapotok, ahol csak egy bit különbözőség van.

Prioritás kód:

Az analógból digitálisba átalakítók egyik fajtája, az ún. flash A/D átalakító pl. prioritás-kódban adja az átalakítás eredményét, ezt kell kódolni binárissá. A prioritás kódnál csak az számít, melyik az első 1-es , ha a biteknél a legnagyobb prioritástól csökkenő prioritás felé haladunk. 2^{n-1} bitet $\log_2 n$ bites számmá tudunk kódolni, példánkban 7-ről 3-ra enkódoljuk:

K0	K1	K2	K3	K4	K5	K6	A2	A1	A0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	X	0	1	0
0	0	0	0	1	X	X	0	1	1
0	0	0	1	X	X	X	1	0	0
0	0	1	X	X	X	X	1	0	1
0	1	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	1	1	1

Prioritás enkóder (kódoló) 7-ről 3-ra igazságtáblázata

Az X-szel jelölt értékek mindegy, hogy 0-ák, vagy 1-ek, csak a legnagyobb helyiértékű bit számít

2n-1 bitet $\log_2 n$ bites számmá tudunk kódolni

Egyéb bináris kódok

Sokféle kódot ismerünk, részletes ismertetésükre nem vállalkozhatunk terjedelmi okokból.

Felsorolás jelleggel megemlítünk néhány kódot, melyekkel a gyakorlataink-, és feladataink során találkozunk:

- A hétszegmenses kijelző kódjára dekódoló kapcsolás, ezt tervezni is fogjuk
- A hétszegmenses kijelző, a dominó és a dobókocka lámpáinak kódolása, tervezése
- Nixie cső kódjára dekódoló kapcsolás tervezése
- Az 5x7-es mátrix kódolása, tervezése, stb.

BCD, Binárisan kódolt decimális számok

Jelentőségük miatt külön alfejezetet szentelünk a decimális számok bináris kódú ábrázolásának. A legegyszerűbb tetrád kód az egyszerű négybites bináris kód.

A tetrád kódok alkalmasak akár 16 különböző dolog kódolására, így decimális számjegyeket is lehet velük kódolni.

Egyszerű BCD kód

Decimális számjegyek bináris kódolását mutatja be az alábbi táblázat:

dec.	**BCD**			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
Pszeudotetrád	1	0	1	0
	1	0	1	1
	1	1	0	0
	1	1	0	1
	1	1	1	0
	1	1	1	1

Egyszerű BCD kóddal kódolt decimális számok,
vagy röviden BCD kódú decimális számok

A decimális számjegy nem lehet nagyobb 9-nél, így az utolsó hat tetrád kihasználatlan. Ezeket a kihasználatlan tetrádokat **pszeudotetrád**oknak nevezzük.
BCD kódolású számoknál ügyelni kell arra, hogy 1001 után 0000 jön, erre BCD számlálók és aritmetika esetén oda kell figyelni.

Tanulmányaink során terjedelmi okokból, és nagyobb szakértelmet igénylő (nehezebb) volta miatt nem foglalkozunk a BCD műveletekkel, sem a műveletekkel tetrád kódokban. A kiterjesztett tetrád kódokat sem említjük.

Az egyszerű, más szóval **közönséges BCD** kódon kívül a következő kódok terjedtek el decimális számokra:

II-2. A decimális számok bináris kódjai

Dec.	Alap BCD				Egyeseket minimáló BCD				AIKEN-kód				STIBBITZ-kód				GRAY-kód				Módosított GRAY-kód			
	8	4	2	1	7	4	2	1	2	4	2	1												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	1	1	0
2	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	1	0	0	1	1	0	1	1	1
3	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	0	0	0	1	0	0	1	0	1
4	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	1	0	0
5	0	1	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	1	1	1	1	1	0	0
6	0	1	1	0	0	1	1	0	1	1	0	0	1	0	0	1	0	1	0	1	1	1	0	1
7	0	1	1	1	1	0	0	0	1	1	0	1	1	0	1	0	0	1	0	0	1	1	1	1
8	1	0	0	0	1	0	0	1	1	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0
9	1	0	0	1	1	0	1	0	1	1	1	1	1	1	0	0	1	1	0	1	1	0	1	0

- Az **1-eseket minimáló BCD-kód**, mely helyiértékein a súlyozás 7421 (a szokásos 8421 helyett).
- **AIKEN-kód**, mely 2421 súlyozású komplemens kód
- **STIBBITZ-kód, vagy 3 többletes** komplemens kód. Megkapjuk, ha a BCD-kódhoz 3-at adunk.
- Mivel a **GRAY kód** nem komplemens, ezért mellé bevezették a többlet 3-assal **módosított GRAY kód**ot, ez is komplemens.

A komplemens képzés úgy történik, hogy a 0, 1, ..., 9 számjegyek tetrádjában a 0-kat 1-esekre és az 1-eseket 0-ákra cserélve éppen a 9, 8, ..., 0 számok tetrádjait kapjuk meg

Hibajelző és javító kódok

Az információátvitel akkor történik hibamentesen, h a vett adat megegyezik az adott üzenettel. A gyakorlatban ez nem mindig sikerül. Olykor egy, vagy több bit megváltozik, 0-ból 1-re, vagy 1-ből 0-ra változik. Ezeknek a hibáknak a felfedésére, vagy kiküszöbölésére csak akkor van lehetőség, ha a tiszta információt hordozó kódot kibővítjük olyan bittel, vagy bitekkel, melyek megnövelik a kódszavak hosszát úgy, hogy segítségükkel ellenőrizhetők, vagy javíthatók is a kódszavak, de információtartalmát nem változtatják. Ekkor azt mondjuk, a kód redundanciáját növeljük.

Egy redundanciát tartalmazó kódszó tehát a következő bitekből áll:

- Hasznos információt tartalmazó bitek
- Nem hasznos információt tartalmazó bitek, azaz redundancia bitek

A redundanciát tartalmazó kódrendszerek kétféle csoportba sorolhatók, hibafelfedő (ED, Error Detecting), és hibajavító (EC, Error Correcting) kódok.

Hibafelfedő kódok, paritás

Mi csak a paritáselemes kódokat tanuljuk. A paritáselemes kód elve, hogy egy adott kód kódszavát kiegészítjük úgy, hogy a kiegészített kódszóban az 1-esek száma páros, vagy páratlan legyen.

- A kiegészítő bitet paritásbitnek nevezzük.
- Páros paritáselemes kód (páros paritás) az a kiegészített kód, ahol a kiegészített kódszó 1-eseinek száma páros
- Páratlan paritáselemes kód (páratlan paritás) az a kiegészített kód, ahol a kiegészített kódszó 1-eseinek száma páratlan

A II-3. táblázatban tetrádokat és bájtokat látunk el paritásbittel.

II-3. táblázat

Páros					Páratlan					Páros								
A	B	C	D	P	A	B	C	D	P	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	P
1	0	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	1	0
0	0	1	0	1	0	0	1	0	0	1	1	0	1	1	0	0	1	1
1	0	0	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	1
0	1	1	1	1	0	1	1	1	0	0	1	0	1	1	1	1	0	1
0	1	0	0	1	0	1	0	0	0	1	1	0	0	1	0	1	0	0
1	1	0	0	0	1	1	0	0	1	1	1	0	1	1	1	1	0	0
1	0	1	0	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1

A paritásképzés előnye, hogy vele bármilyen kód felruházható hibaellenőrző képességgel.

A paritás képzés hátrányai:

– Nem tudjuk kijavítani a hibát, ha detektáljuk is
– Ha egyszerre több bit hibásodik meg, nem biztos, hogy a paritásellenőrzés felfedezi, mert lehet, hogy egyszerre két (vagy páros számú) bit is megváltoztatja értékét.

Hibajavító kódok

A többlet bitek nem csak hibajelzésre, de hibajavításra is alkalmasak. A hibák javítására nem elég kódszavanként 1 bit. Egyszerű példát mutatunk hibajavításra, a tömbátvitelt. Ekkor az adatokat csomagonként, tömbökben visszük át. Hibajavításra akkor leszünk képesek a tömb átvitelekor, ha az adatokat kereszt-, és hosszirányban is hibajelző kódokkal látjuk el. Így nem csak az derül ki, melyik szó hibás, hanem az is, benne melyik bit. Mivel egy bit hibája azt jelenti, értéke 0-ról 1-re (vagy fordítva) változott, csak vissza kell fordítani, így kijavítandó a hibás bit.

Lássunk erre egy példát egy 8 bájtból álló tömbnél, melyet mind kereszt-, mind hosszirányban elláttunk páros paritásbitekkel. Példaképp hibásodjon meg a tömb 4-ik bájtjának A_3 bitje. A II-1. Hibajavítás tömbátvitelkor c. ábrán látható a hiba jelentkezése, felfedésének és kijavításának módja:

II-1. Hibajavítás tömbátvitelkor

Küldött tömb

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	P
0	1	0	1	0	1	0	1	0
1	1	0	1	1	0	0	1	1
0	0	1	1	0	0	0	0	0
0	1	0	1	1	1	1	0	1
1	1	0	0	1	0	1	0	0
1	1	0	1	1	0	0	1	1
0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	0	1
1	0	1	0	1	1	1	1	1

Vett tömb egy hibás bittel

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	P
0	1	0	1	0	1	0	1	0
1	1	0	1	1	0	0	1	1
0	0	1	1	0	0	0	0	0
0	1	0	1	0	1	1	0	1
1	1	0	0	1	0	1	0	0
1	1	0	1	1	0	0	1	1
0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	0	1
1	0	1	0	1	1	1	1	1

A hiba kijavítása

→ 0 1 0 1 **1** 1 1 0

↑

Javított bit

Egyéb hibajavító kódok

A hibák javítására az ún. Hamminng kódot, vagy a Red Solomon kódot használják. Ezek ismertetésére nem térünk ki (2004/2005-ben).

További tanulmányainkban csak az egyszerű bináris és a közönséges BCD kódot használjuk. Kivétel a különböző példákban, feladatokban, ahol különböző kódolások, dekódolások is feladatul szerepelnek, de velük számolnunk, őket átalakítanunk nem kell, **ezt majd az általunk tervezett hardverre bízzuk**.

Kérdések és feladatok

Válaszolja meg: Mi az alfanumerikus kód? Mi a numerikus kód? Melyik a legkedvezőbb alapszám, és miért? Milyen bináris kódokat ismer? Mutassa meg az egyszerű BCD kódot! Mutassa meg a Gray kódot! Mi a pszeudotetrád? Mi a redundancia? Mi a paritásbit? Milyen lehetőséget ismer a hibák felfedésére bináris kód alkalmazása esetén? Milyen lehetőséget ismer hibajavító kódolásra?

Feladatok

- Határozza meg, legalább hány szerkezeti elem szükséges a 0-9999-ig terjedő decimális számok ábrázolásához: a.) Bináris számokkal; b.) Oktális számokkal; c.) Decimális számokkal; d.) Hexadecimális számokkal
- Számítsa át decimális számrendszerbe a következő számokat:
  3441_8; 11101_{16}; 1110_{18}; 1110_4; 11101_2; $999D_{16}$

- Számítsa át binárisba a következő számokat:
  3441_8; 11101_{16}; 1110_{16}; 1110_4; 11101_{10}; $999D_{16}$
- Alakítsa át binárisba a következő hexadeximális számokat számítás nélkül:
  11101_{16}; $999D_{16}$; ADD_{16}; $BABA_{16}$; FA_{16}; $1FED_{16}$;$110F_{16}$;
- Alakítsa hexadecimálisra az alábbi bináris számokat:
- 101; 11; 11010; 110011101111011011111011; 100110110011101011110001101010 1;
  (5; 3; 1A; 19DEEFB; 4D9D78D5)
- Végezze el az összeadást az alábbi bináris számokkal:
  1001000 + 10010 =
  110100111 + 1101101 =
  10110011011 + 11001111101 =
  1010001110100110111 + 100111110111010110 =

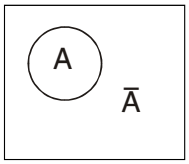
Egyéb tudnivalót, kérdéseket és feladatokat a Kódok és aritmetika című fejezetből 2004/2005-ben nem adok fel

III. Logika

Boole algebra

A halmazok logikája

A és $\bar{A}$ Venn diagrammja



- Halmazon a közös tulajdonságú dolgok összességét értjük.
- A halmaz elemei a halmazhoz tartozó dolgok
- Egy A halmaz **kiegészítő (komplemens) halmaza** alatt azt az $\bar{A}$ halmazt értjük, mely elemeinek nincs meg az A elemeinek tulajdonsága.

- Az üres halmaz olyan halmaz, melynek egyáltalán nincs eleme. Az üres halmazt nulla halmaznak, 0 halmaznak is nevezik. **Az üres halmaz jele a 0**.
- Részhalmaz egy halmaz elemeinek olyan csoportja, melyeket további tulajdonságokkal határozunk meg.
- Az üres halmaz komplemense az univerzális halmaz a Boole algebrában. **Az univerzális halmaz jele az 1.** Az univerzális halmaz komplemense a 0 halmaz.

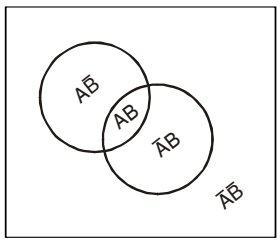
Példa: az egész számok halmazának részhalmaza a páros számok összessége. Ekkor a páratlan számok (a nem páros számok) a páros számok komplemense. A páros és a páratlan számok együtt alkotják az egész számok univerzális halmazát.

- Két halmaz (A és B) közös része, **logikai szorzata**, **metszete,** konjunkciója alatt azokat az elemeket értjük, melyek egyszerre elemei A-nak **és** B-nek is. A logikai szorzatot a Boole algebrában így jelöljük: $C = AB$. Természetesen $AB = BA$

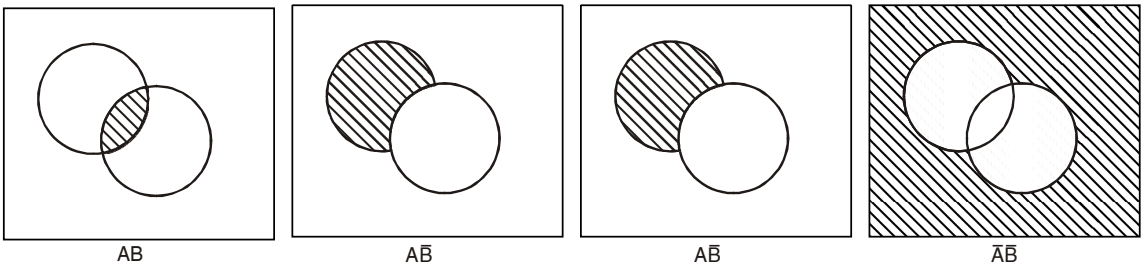
  A halmazelméletben szokás AB-t $A \cap B$ -vel jelölni.

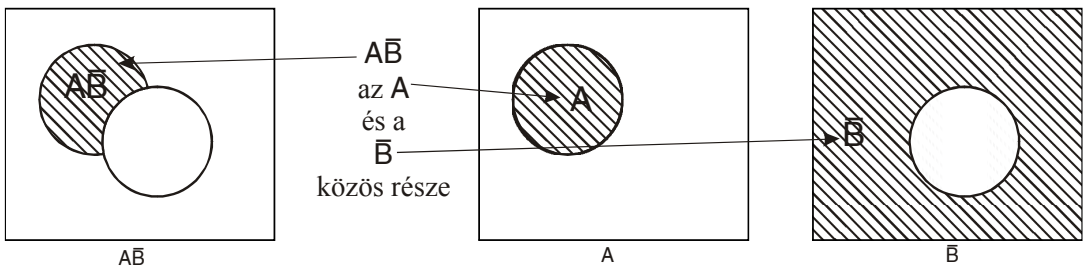
Az A és B halmaznak csak a következő részhalmazai képzelhetők el: AB, $A\bar{B}$, $\bar{A}B$ és $\bar{A}\bar{B}$

Finomabb (hogy kisebb részeket tartalmazó) felosztás nem képzelhető el, ezért az AB, $A\bar{B}$, $\bar{A}B$ és $\bar{A}\bar{B}$ részhalmazokat **mintermeknek** is nevezik. A mintermek tehát logikai szorzatok az összes szóba előforduló kombinációban.
A négy lehetséges részhalmaz külön-külön:



AB, $A\bar{B}$, $\bar{A}B$ és $\bar{A}\bar{B}$ Venn diagrammjai





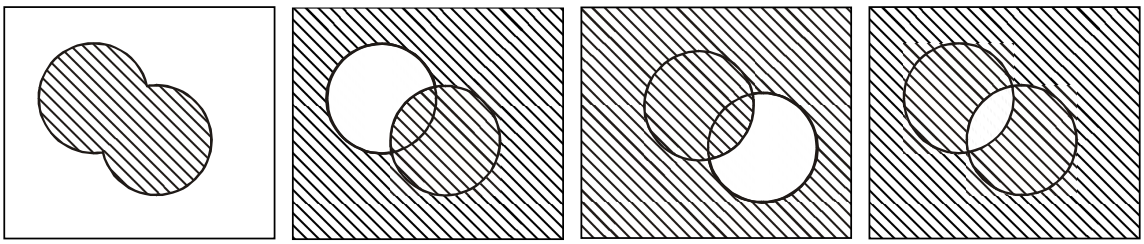
Magyarázat az $A\bar{B}$ Venn diagrammjának képzéséhez

- Azok az elemek, melyek **vagy** az A halmazhoz, **vagy** a B halmazhoz, **vagy** mindkettőhöz tartoznak, az **A éb B halmaz logikai összegét**, más néven egyesítését, únióját alkotják.

A logikai összeget (úniót) a Boole algebrában úgy jelöljük, mint a szokásos összeadást: $D = A + B$

Mivel A és B egyenként csak ponált, vagy negált lehet, így a logikai összeg képzés is csak négyféle esetre lehetséges, $A + B$, $A + \bar{B}$, $\bar{A} + B$ és $\bar{A} + \bar{B}$-re:

$A + B$, $A + \bar{B}$, $\bar{A} + B$ és $\bar{A} + \bar{B}$ Venn diagrammjai



$A + B$ $\bar{A} + B$ $A + \bar{B}$ $\bar{A} + \bar{B}$

Durvább (nagyobb részeket tartalmazó) felosztás nem képzelhető el, ezért az $A + B$, $A + \bar{B}$, $\bar{A} + B$ és $\bar{A} + \bar{B}$ részhalmazokat **maxtermeknek** is nevezik. A maxtermek tehát logikai összegek az összes szóba előforduló kombinációban.

Kétértékű logika és a bináris számok

Eddigi megállapításainkban láttuk, egy logikai halmaz vagy üres, vagy nem üres halmaz volt. Ez a tulajdonsága a Boole algebra által tárgyalt halmazoknak alkalmassá teszi a halmazok jellemzését kétféle értékű jelekkel, fogalmakkal. A két állapot jellemzésére használhatjuk a 0, és az 1 számokat. **0, ha üres a halmaz, és 1, ha nem üres.**

Látható, a bináris számrendszer számjegyei logikai halmazok jellemzésére is alkalmasak. Az univerzális halmazt, mely nem üres halmaz, eddig is 1-gyel jelöltük, míg a biztosan üres 0 halmazt 0-val.
A logikai egyenletek hasonlítanak a számok egyenleteihez. Pl. a logikai szorzatot szorzásnak jelöljük, a logikai összeget összegnek, stb. De látni fogjuk, hogy nem minden esetben van megfelelője a logikai tételeknek a számokon értelmezett műveleteknél (pl. abszorpció).

A továbbiakban mindig kétértékű logikát tanulmányozunk

További kétértékű jellemzők lehetnek felsorolás jelleggel: Van/nincs; fekete/fehér; jó/rossz, kicsi/nagy, ilyen-/olyan irányú, magas/alacsony, világos/sötét, **igaz/hamis**. Utóbbi pár külön alfejezetet érdemel:

Logikai függvények

A logikai függvények független logikai változókhoz rendelt függő logikai változó(k). Hasonlóan, mint a számoknál, csak itt az értékek logikai értékek. Tehát vizsgáljuk, hogyan függ egy (vagy több) logikai változó a független logikai változóktól. Pl. az $X = AB\bar{C}$ azt jelenti, X akkor igaz, ha A igaz és B is igaz és C hamis. Hasonlóan az $Y = \bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + ABCD$ egyenlet is egy logikai függvény, mely közli, Y mikor lesz igaz.

Ítéletek logikája

A kétértékű logika az olyan ítéletek meghozására alkalmas, mikor ítéletünk csak kétféle lehet. **Az ítéletet egy állításról hozzuk** a következő módon: **az állítás vagy igaz, vagy hamis**, csakis az egyik, de az mindenféleképpen.

Állításkalkulus

A logikai ítéletek meghozását állításkalkulusnak is nevezik.. Az állításkalkulus hasonlít a beszélt nyelvhez. Néhány példával mutatjuk meg ezt a hasonlóságot:

X igaz, hogy folyik a víz, ha **A** csap nyitott **ÉS B** csap is nyitott, **ÉS NEM** nyitott a **C** biztonsági szelep. Egyenlete: $X = AB\bar{C}$

Y igaz, hogy főzhetek, ha **A** van gáz, **ÉS B** van víz, **ÉS C** van élelmiszer. Egyenlete: $Y = ABC$

Z igaz, hogy bemehetek, ha **A** van kulcsom **VAGY NEM B** zárt az ajtó, **VAGY C** van elég erőm (tankom, kalapácsom-vésőm, bombám, stb.), **VAGY B** zárt az ajtó **ÉS (D** meghallják a csengőt, **VAGY E** elég nagyot tudok kiabálni, **VAGY F** elég nagyot dörömbölök**)**
Egyenlete: $Z = A + \bar{B} + C + B(D + E + F)$

Az **L** lámpa ég, ha az **A** keleti kapcsoló **1**-es **ÉS B** nyugati kapcsoló is **1**-es állásban van, **VAGY** az **A** keleti kapcsoló **NEM 1**-es **ÉS** a **B** nyugati kapcsoló is **NEM 1**-es állásban van. Ez egy úgynevezett alternatív kapcsolás, akkor világít **L**, ha a két kapcsoló azonos állásban van.
Egyenlete: $L = AB + \bar{A}\bar{B}$

Az **L** lámpa ég, ha az **A** északi kapcsoló **NEM 1**-es **ÉS B** déli kapcsoló **1**-es állásban van, **VAGY** az **A** északi kapcsoló **1**-es **ÉS** a **B** déli kapcsoló **NEM 1**-es állásban van. Ez is alternatív kapcsolás, akkor világít **L**, ha a két kapcsoló különböző állásban van.
Egyenlete: $L = AB + \bar{A}\bar{B}$

- A logikai negálást NEM-nek (nemzetközi NOT) mondjuk.

Pl. $\bar{A}$ kimondva NEM A. $\bar{A}$ jelentése: NEM A, NOT A, tagadott A, vagy negált A.

- Amit nem tagadunk, állítjuk. Az állított A-t ponált A-nak is mondják.
- A logikai szorzatot az ÉS-nek mondjuk.
- A logikai összeget az VAGY-nak mondjuk.

Példák:

AB kimondva: A ÉS B.
A + B kimondva: A VAGY B.
Az $X = A\bar{B}$ egyenlet kimondva: X (igaz), ha A ÉS NEM B (igaz).
Az $Y = \bar{D} + E$ egyenlet kimondva: Y (igaz), ha NEM D VAGY E (igaz).

Megjegyzés: Sokszor a zárójelben levőket nem mondják, nem kell kimondani, csak úgy érthetőbb.

Igazságtáblázat

Tanulmányaink során egyszerű, kétértékű ítéleteket hozunk, egyszerű, egyenként két lehetséges állapotú feltételekkel. Ezt táblázatban is ábrázolhatjuk. Ebben a táblázatban egyszerűen számba vesszük, az ítélet mikor, milyen feltételek esetén igaz, és mikor hamis. Az ilyen táblázatot nevezzük igazságtáblázatnak.

Logikai NEM igazságtáblázata:

$X = \bar{A}$

A	X
Hamis	1
Igaz	0

Logikai ÉS igazságtáblázata:

$Y = AB$

Beszélt			Jelölt		
A	B	Y	A	B	Y
Hamis	Hamis	Hamis	0	0	0
Hamis	Igaz	Hamis	0	1	0
Igaz	Hamis	Hamis	1	0	0
Igaz	Igaz	Igaz	1	1	1

Logikai VAGY igazságtáblázata:

$Z = A + B$

Beszélt			Jelölt		
A	B	Z	A	B	Z
Hamis	Hamis	Hamis	0	0	0
Hamis	Igaz	Igaz	0	1	1
Igaz	Hamis	Igaz	1	0	1
Igaz	Igaz	Igaz	1	1	1

A Boole algebra azonosságai és tételei

Alapvető azonosságok

$A + A = A$. Természetesen akárhányszor vesszük A önmagával való logikai összegét, A-t kapunk.

$A + \bar{A} = 1$. Ez következik a komplemens halmaz értelmezéséből, (lásd ott)

$AA = A$. Természetesen akárhányszor vesszük A önmagával való logikai szorzatát, A-t kapunk.

$A\bar{A} = 0$. Egy halmaznak és komplemens halmazának nincs közös része (metszete)

$\bar{\bar{A}} = A$ A kettős tagadás állításnak felel meg. **A tagadás ellentettje tehát állítás.** (A tagadás tagadása állítás.)

$A + 0 = A$ Bármilyen A halmazhoz hozzáadjuk a 0 halmazt, magát az A halmazt kapjuk

$A + 1 = 1$ Mivel A részhalmaza az univerzális halmaznak.

$A1 = 1A = A$

$A0 = 0A = 0$ Mivel A-nak és a 0 halmaznak nincs közös eleme

Kommutativitás (felcserélhetőség)

$A + B = B + A$

$AB = BA$

Asszociativitás (átzárójelezhetőség)

A logikai szorzat átzárójelezhető:

$ABC = A(BC) = (AB)C = B(AC)$, stb.,

A logikai összeg átzárójelezhető

$A + B + C = A + (B + C) = (A + B) + C = B + (A + C)$, stb.

Disztributivitás (átcsoportosíthatóság)

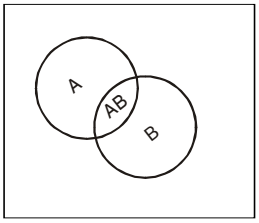
$A(B+C) = AB + AC$.

Abszorpciós törvény

$A + AB = A$

$A(A+B) = A$

Az abszorpció nem hasonlít a számok viselkedésére!



Itt a második alak a disztributív törvénnyel az első alakjára hozható:
$A(A + B) = AA + AB = A + B$. Ezért csak $A + AB = A$ -t kell igazolni. Ezt a Venn diagramból beláthatjuk. Később igazolni fogjuk igazságtáblával, és Karnaugh táblával is.

További abszorpciós törvények:

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

$A(\bar{A} + B) = AB$

De Morgan tételei

De Morgan tételei fontosak, gyakran fogjuk használni őket!

$\overline{A+B} = \bar{A}\bar{B}$, ill. akárhány tagra:

$\overline{A+B+\ldots+N} = \bar{A}\bar{B}\ldots\bar{N}$

$\overline{AB} = \bar{A}+\bar{B}$, ill. akárhány tagra:

$\overline{AB\ldots N} = \bar{A}+\bar{B}+\ldots+\bar{N}$

$\overline{A+B+\ldots+N} = \bar{A}\bar{B}\ldots\bar{N}$ belátható, ha meggondoljuk, mit is jelent az egyenlet bal és jobb oldala. A bal oldal akkor igaz, ha hamis az $A+B+\ldots+N$ állítás. Ez pedig akkor hamis, ha $A+B+\ldots+N$ mindegyik tagja hamis. Ekkor azonban ezek tagadottja, $\bar{A}+\bar{B}+\ldots+\bar{N}$ mind igaz. Utóbbi pedig pontosan a tétel egyenletének jobb oldala. Másképp fogalmazva: Ha az állítások logikai összege hamis, akkor az állítások mindegyike hamis. Természetesen ekkor minden tagadott állítás igaz.

Hasonlóképpen látható be $\overline{AB\ldots N} = \bar{A}+\bar{B}+\ldots+\bar{N}$ is. Itt a bal oldalon az $\overline{AB\ldots N}$ csak akkor igaz, ha $AB\ldots N$ hamis, azaz legalább egy eleme hamis. Ekkor az egyenlet jobb oldala is igaz lesz, mert legalább ez az egy elem negáltja igaz lesz, így állításunkat igazoltuk. Másképp fogalmazva: Ha az állítások logikai szorzata hamis, akkor az állítások legalább egyike hamis. Természetesen ekkor legalább egy tagadott állítás igaz.

Logikai állítások bizonyítása

Bizonyítás igazságtáblázattal

Ha egy állítás összes szóba jöhető esetét megvizsgáljuk egy táblázattal, az ún igazságtáblázattal, és igazolást nyer, amit állítottunk, állításunkat bizonyítottuk. Az igazságtáblázatról a *Logikai* egyenletek
alfejezetben *(IV.2. old.)* bővebben lesz szó

Nézzük meg ezt a módszert néhány példán:

Igazoljuk, az első abszorpciós tételt, hogy $A + AB = A$. Itt A és B egyenként két értékű lehet, vizsgáljuk meg hát az összes szóba jöhető esetet:

A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Látható, $A + AB$ oszlopában minden sorban megegyezik az érték A-val, függetlenül B értékétől.
Ezzel igazoltuk, $A + AB = A$.

Igazoljuk az $A + \bar{A}B = A + B$ abszorpciós tételt. Itt A és B egyenként kétértékű lehet, vizsgáljuk meg hát az összes szóba jöhető esetet:

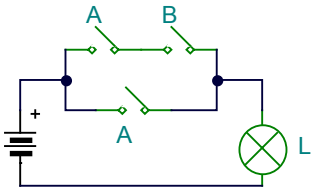
A	B	A + B	$\bar{A}$	$\bar{A}B$	$A + \bar{A}B$
0	0	0	1	0	0
0	1	1	1	1	1
1	0	1	0	0	1
1	1	1	0	0	1

Látható, $A + B$ oszlopa minden sorban megegyezik az érték $A + \bar{A}B$ -vel.
Ezzel igazoltuk, $A + \bar{A}B = A + B$.

Bizonyítás meggondolással (pl. érintkezőkkel)

Az érintkezőkről lásd az *Érintkezők, kapcsolók, nyomógombok logikája* fejezetet *(III.7 old.)*
Ha egy állítást megvalósítunk érintkezőkkel, egyszerűbb esetben ránézésre azonnal beláthatjuk, igaz-e, vagy sem. Nézzük meg ezt a módszert néhány példán:

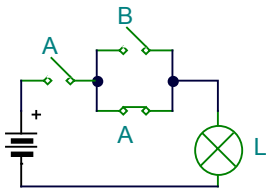
Igazoljuk, az első abszorpciós tételt, hogy $A + AB = A$. Valósítsuk meg érintkezőkkel –t, és vonjuk le a következtetést:



Meggondolás: Most $L = A + AB$.
Ha, $A = 1$, a lámpa mindig ég, ha $A = 0$, akkor pedig soha. Így a lámpa csak A-tól függ, $L = A.= A + AB$
Ezzel állításunkat igazoltuk

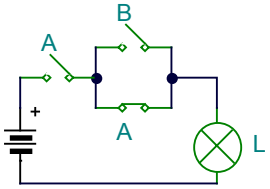
Ha $A = 1$, akkor mind a két A érintkező vezet. Ezt **együttmozgó érintkezőknek mondjuk**, mikor az egyik be (ki) kapcsolt, a másik is be (ki) kapcsol, azaz együtt mozognak.

Igazoljuk, az utolsó abszorpciós tételt tételt, hogy $A(\bar{A} + B) = AB$. Valósítsuk meg érintkezőkkel, és vonjuk le a következtetést:

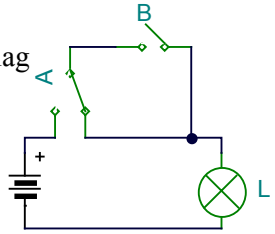


Meggondolás: Most $L = A(\bar{A} + B)$.
Ha $A = 0$, a lámpa sosem ég. Ha $A = 1$, a párhuzamos ág nem igaz, csak, ha $B = 1$, mert ekkor $\bar{A} = 0$.
Tehát a lámpa csak akkor ég, ha $A = 1$ ÉS $B = 1$.
Ezzel állításunkat igazoltuk

Ha $A = 1$, akkor az $\bar{A} = \bar{0}$ érintkező nem vezet. A két érintkező egyszerre mozog, mikor az egyik 0, a másik 1, és fordítva. Mikor az egyik be (ki) kapcsol, a másik ki (be) kapcsol, azaz együtt mozognak, de mindig ellenkező állásúak. Ilyen a váltóérintkező is, mikor egyik állásban a 0 felé, a másikban az 1 felé vezet. Sok esetben a két együttmozgó, de ellentétes állású érintkezőpárt át lehet alakítani váltóérintkezős megoldásúvá. Most is:



A két kapcsolás logikailag teljesen megegyezik.
$L = A(\bar{A} + B) = AB$



Bizonyítás algebrai módszerrel

Ez olyan átalakítást jelent, melyben a Boole algebra azonosságainak és tételeinek felhasználásával olyan alakra hozzuk az eredeti állítást, melyet igazolni szeretnénk. Ehhez bővíteni és csoportosítani kell, elég nagy leleményességet igényel, e módszer elég körülményes. Ha si-

kerrel járunk, a bizonyítandó állítást igazoltuk. Ha az állításnak ellentmondó alakra jutunk, azt bizonyítottuk, a bizonyítandó állítás hamis. Nézzünk egy példát:

Igazoljuk, az első abszorpciós tételt, hogy A + AB = A.

Az azonosságok és tételek alkalmazásával:

A + AB = AB + A Bővítsük A-t A1-re (1-gyel szorozva nem változik az értéke)

AB + A = AB + A1 A-t emeljük ki

AB + A1 = A(B + 1) A zárójeles kifejezés mindig 1, (B + 1) = 1, így

A(B + 1) = A. Amit bizonyítani akartunk.

Az algebrai módszerrel körülményes a bizonyítás, ezért nem alkalmazzuk.

Logikai áramkörök

Érintkezők, kapcsolók, nyomógombok logikája

Fontosságuk miatt külön alfejezetet érdemelnek a kapcsolók. Kapcsolónak számítanak a szelepek is, ha nyitottak, lehet áramlás, ha zártak, nem. Mi elektromos kapcsolókat tekintünk.

Érintkezők rajzjelei, megállapodások

A kapcsolót, érintkezőket mindig inaktív (nyugalmi) állapotában rajzoljuk. Ha a kapcsolót, érintkezőket aktivizáljuk (működésbe hozzuk), érintkezője helyzete megváltozik. Ha inaktív állapotban nyitott, aktív állapotában zárt.

Az inaktív (nyugalmi) állapotot 0-val jelöljük

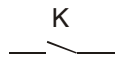
Az aktív (működtetett) állapotot 1-gyel.

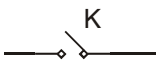
Mindig 0 állapotban rajzoljuk az érintkezőket (kapcsolókat, nyomógombokat).

Záróérintkező

Az aktív állapotban záró érintkezőt záróérintkezőnek mondjuk.

A záróérintkezőt nyitottnak (bontott állapotúnak) rajzoljuk. Akkor zár, ha működtetjük.

Szabványos jele: 

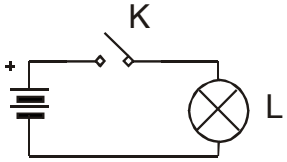
Tina áramkörszerkesztőben a jele: 

A továbbiakban a Tina áramkörszerkesztő program jeleit fogjuk használni.

Ilyen érintkezője van a Be nyomógombnak és a Be kapcsolónak.

Példa: Ha működtetjük a K kapcsolót, záródik érintkezője, és a lámpa világít.

Ha K IGAZ (hogy működtetett), L IGAZ (hogy világít)

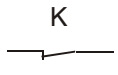


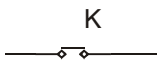
Ennek a kapcsolásnak a logikai egyenlete: L = K

Az L lámpa akkor világít, ha K = 1 Ekkor természetesen L = 1.

Bontóérintkező, nyitóérintkező

Az aktív állapotban bontó érintkezőt bontóérintkezőnek, vagy nyitóérintkezőnek mondjuk. A bontóérintkezőt zárt állapotúnak rajzoljuk (ami működtetéskor bont, kinyílik).

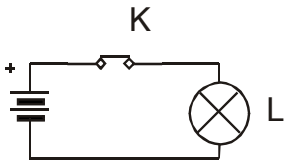
Szabványos jele: 

Tina áramkörszerkesztőben a jele: 

A továbbiakban a Tina áramkörszerkesztő program jeleit fogjuk használni.

Bontóérintkezője van a Ki nyomógombnak, Vész-gomboknak és a Ki kapcsolónak.

Példa: Ha nem működtetjük, érintkezője zárt, a lámpa világít. Ha működtetjük a K kapcsolót, bont érintkezője, és a lámpa nem világít. Ha K NEM működtetett, az L IGAZ (hogy világít)



Ennek a kapcsolásnak a logikai egyenlete: $L = \overline{K}$, vagy L = NOT K

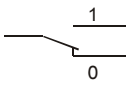
Az L lámpa akkor világít, ha K = 0 Ekkor természetesen L = 1.

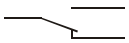
Figyelem! A bontó érintkező 0 állásban zár! A bontóérintkező is nyugalomban 0 állású, mégis zár. Ha működtetjük, bont, kikapcsol. Tehát **a 0 jelzésű állás nem kikapcsolt, hanem nyugalmi helyzetet jelent**.

Váltóérintkező, váltókapcsolók

A váltóérintkező hasonlít a vasúti váltóhoz: két állása közül a 0-ban az egyik, az 1-esben a másik irányba vezet.

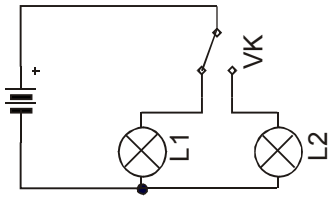
Szabványos rajzjele :



vagy 
Tinában 

A továbbiakban a Tina áramkörszerkesztő program jelét fogjuk használni.

Példa váltóérintkező alkalmazására (VK egy váltókapcsoló):

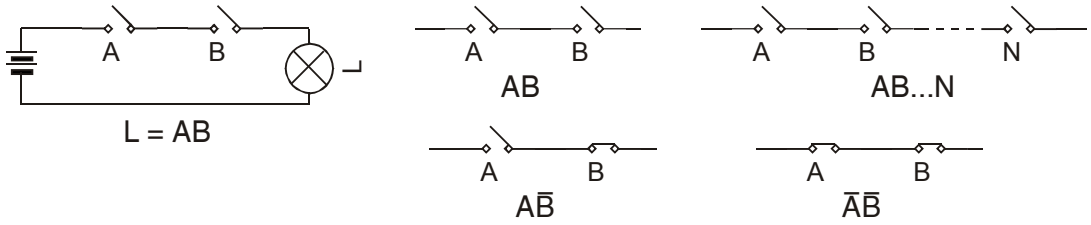


$L1 = \overline{VK}$ és L2 = VK . Tehát, ha VK = 0, L1 világít, ha VK = 1, L2 világít.

Egyéb, bonyolultabb érintkezőket az *Érintkezők* fejezetben mutatunk. *(VII.4 oldal)*

És kapcsolat érintkezőkkel:

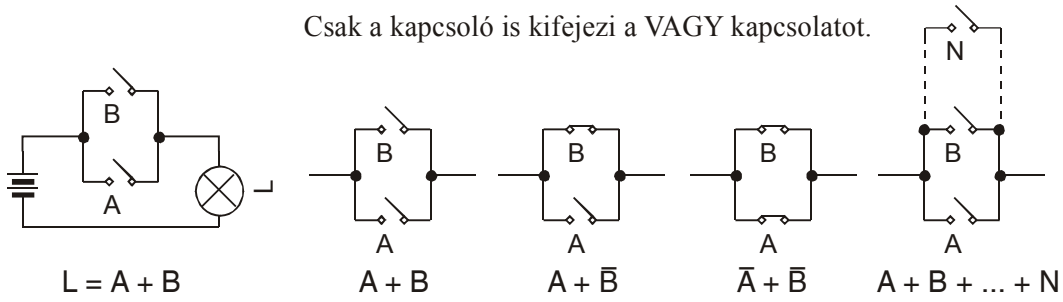
Csak a kapcsoló is kifejezi az ÉS kapcsolatot.



Amikor az ÉS igaz, az érintkezők vezetnek.

VAGY kapcsolat érintkezőkkel:

Csak a kapcsoló is kifejezi a VAGY kapcsolatot.



Amikor a VAGY igaz, az érintkezők vezetnek.

- Azt az elektromos kapcsolást, mely egy logikai állítást, egyenletet valósít meg, **logikai kapcsolásnak** nevezzük.

Az érintkezőkkel, kapcsolókkal, nyomógombokkal sokféle logikai állítás megvalósítható, modellezhető. A Példákban bemutatunk, a Feladatokban meg kell tervezni, és a Gyakorlaton mérni kell néhány érintkezőkkel, kapcsolókkal és nyomógombokkal megvalósított logikai kapcsolást.

Az elektronikában a legtöbb logikai kapcsolást félvezetős áramkörökkel valósítják meg. Erről később mi is részletesen tanulunk.

Kapuáramkörök

A digitális áramkörök alapvető elemei a logikai kapuáramkörök. Felépítésüket később tanuljuk, egyelőre dobozoknak fogjuk fel őket, logikai gépeknek, melyeknek bemenetei és kimenetei vannak. A bemenetek a független logikai változók, míg a kimenetek a bemenetek logikai állapotától függő logikai változók.

- A logikai kapuáramköröket sokszor egyszerűen kapuknak nevezik
- A kapuk minden ki- és bemenete kétféle feszültségű lehet, vagy egy magasabb feszültségű (jele **H**, High), vagy egy alacsonyabb feszültségű (jele **L**, Low).

A magas és az alacsony feszültség jól megkülönböztethető egymástól, ez a digitális technika egyik legfőbb jellegzetessége. Tehát nem fordulhat elő helyes működés esetén, hogy véletlenül a H olykor L lesz, vagy nem tudjuk eldönteni, hogy melyik.

- Ha a logikai 1-nek a H feszültségszint felel meg, pozitív logikának nevezzük. Ha a logikai 1-nek az L feszültségszint felel meg, negatív logikának nevezzük.

Mivel a digitális technikában a feszültség tulajdonképpeni értéke lényegtelen, rajzainkban, közvetlenül a logikai változók értékeit fogjuk jelölni. Tehát pl. egy magas értéket nem

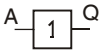
3,34V-tal, hanem H-val, vagy pozitív logika esetén 1-gyel. Hasonlóan egy alacsony értéket sem 0,37V-tal, hanem L-lel, vagy magas logika esetén 0-val.

A gyakran használt kapuk

Mi a leggyakrabban használt kapuáramkörökkel dolgozunk, melyek megegyeznek a Tina áramkörszerkesztő program kapuival. Tehát csak a legelterjedtebb, leggyakrabban használt kapukat említjük itt.

Erősítő kapu (buffer)

Buffer



A	Q
0	0
1	1

A buffer kimenete megegyezik a bemenetével, csak nagyobb áramú lehet

Ez a kapu csak erősíti az áramot, kimenete logikai szintje megegyezik a bemenet szintjével.

NEM (NOT) kapu

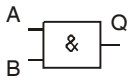
NEM kapu

A	Q
0	1
1	0

A NEM kapu kimenete a bemenet negáltja.
A tagadást kis körrel jelöljük!

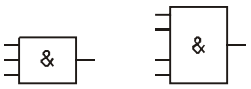
ÉS (AND) kapu

Kétbemenetű ÉS kapu



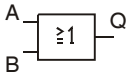
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Több bemenetű ÉS kapu



VAGY (OR) kapu

Kétbemenetű VAGY kapu

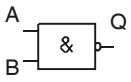


A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Több bemenetű VAGY kapu

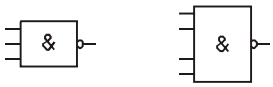
NEM ÉS (NAND) kapu

Kétbemenetű NEM ÉS kapu



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

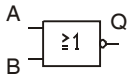
Több bemenetű NEM ÉS kapu



A tagadást kis körrel jelöljük

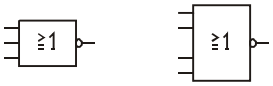
NEM VAGY (NOR) kapu

Kétbemenetű NEM VAGY kapu



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

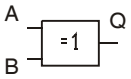
Több bemenetű NEM VAGY kapu



A tagadást kis körrel jelöljük

Kizáró VAGY (XOR, antivalencia) kapu

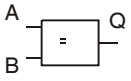
Kizáró VAGY kapu



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

Egyenlőség (ekvivalencia) kapu

Ekvivalencia kapu



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

Egyenlőség kapu a Tina áramkörszerkesztő programban nem található. Egy negált Kizáró VAGY kapuval helyettesíthetjük

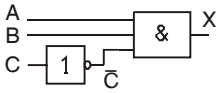
A Tina áramkörszerkesztő program többi kapuját, egyéb kapukat és egyéb elemi áramköröket később tanulunk.

Példák kapuk alkalmazására

Tekintsük meg, miképp lehet megvalósítani az *Állításkalkulus* alfejezet *(III.3. old)* példáit logikai kapukkal:

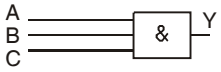
X igaz, hogy folyik a víz, ha **A** csap nyitott **ÉS B** csap is nyitott, **ÉS NEM** nyitott a **C** biztonsági szelep. Egyenlete: $X = AB\bar{C}$

Megoldás:



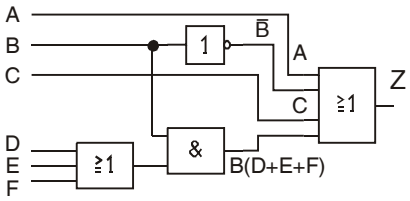
Y igaz, hogy főzhetek, ha **A** van gáz, **ÉS B** van víz, **ÉS C** van élelmiszer. Egyenlete: $Y = ABC$

Megoldás:



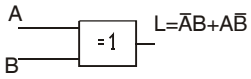
Z igaz, ha **A** igaz **VAGY NEM** igaz **B**, **VAGY** igaz **C**, **VAGY** igaz **B ÉS (**igaz **D**, **VAGY** igaz **E**, **VAGY** igaz **F).** Egyenlettel (függvénnyel): $Z = A + \bar{B} + C + B(D + E + F)$

Megoldás:



Az **L** lámpa ég, ha az **A** keleti kapcsoló **0**-ás **ÉS B** nyugati kapcsoló **1**-es állásban van, **VAGY** az **A** keleti kapcsoló **1**-es **ÉS** a **B** nyugati kapcsoló **0**-ás állásban van. $L = \bar{A}B + A\bar{B}$

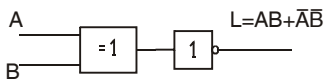
Megoldás:



Ez az egyik úgynevezett alternatív kapcsolás, a XOR logikai függvény megvalósítása. A lámpa akkor világít, ha a két kapcsoló nem egyenlő állásban van, azaz $A = \bar{B}$.

Az **L** lámpa ég, ha az **A** északi kapcsoló **1**-es **ÉS B** déli kapcsoló is **1**-es állásban van, **VAGY NEM** igaz, hogy az **A** északi kapcsoló **1**-es **ÉS NEM** igaz, hogy a **B** déli kapcsoló is **1**-es állásban van. Egyenlete: $L = AB + \bar{A}\bar{B}$

Megoldás:

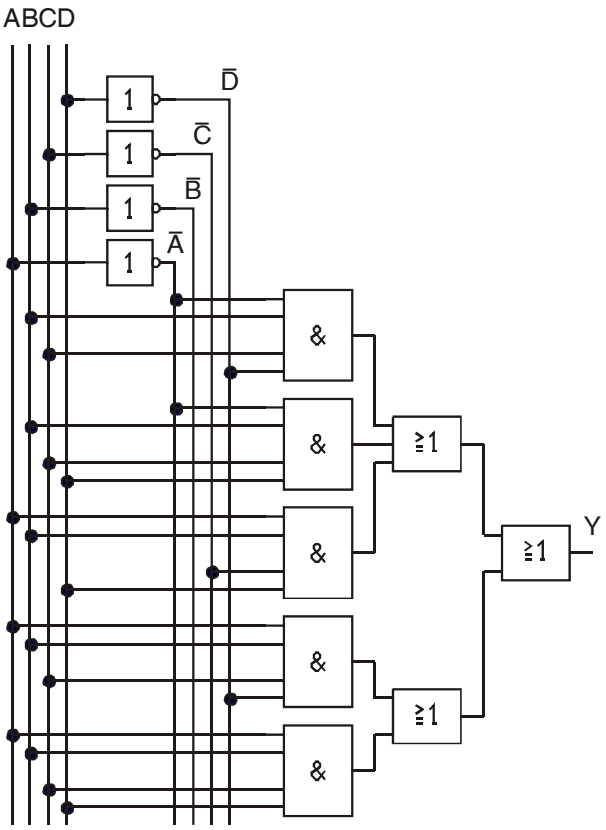


Ez a másik alternatív kapcsolás, az Egyelőség logikai függvény megvalósítása. A lámpa akkor világít, ha a két kapcsoló egyenlő állásban van, azaz $A = B$.

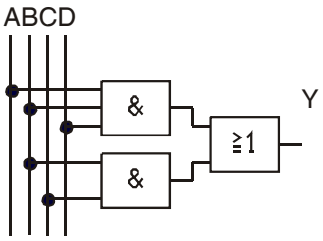
Példa bonyolultabb esetre, valósítsuk meg az alábbi logikai egyenletet (függvényt) kapukkal:

$Y = \bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + ABCD$. Ez egyszerűsítve: $Y = ABD + BC$

III-1. ábra



Egyszerűsítve sokkal kevesebb kapuval meg lehet oldani ugyanezt az egyenletet



$Y = \bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + ABCD$

$Y = ABD + BC$

$Y = \bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + ABCD$

Az egyszerűsítést érdemes elvégezni, mert sokkal olcsóbb és kisebb fogyasztású, meg gyorsabb áramkört kaphatunk. A fenti példa egyszerűsítését *Példa Karnaugh táblával való egyszerűsítésre* (Lásd *IV.19.* oldal) be is mutatjuk.

Hogyan érdemes az adott logikai egyenletet megvalósítani?

- Az előző példában adott volt A, B, C, és D.
- Ezekből készítettük NEM kapukkal az $\bar{A}$, $\bar{B}$, $\bar{C}$, és $\bar{D}$ logikai állapotú huzalokat.
- Ezután a logikai ÉS-eket huzaloztuk ki, mindegyik ÉS kaput a megfelelő huzalra kötve.
- Végül az ÉS kapuk kimeneteit VAGY kapuba kötöttük.

(A bal oldali ábrán előbb egy három, meg egy két bemenetű VAGY kapuba, és ezek kimeneteit is egy VAGY kapuba, mert a Tina áramkörszerkesztő program nem ismer négynél több bemenetű VAGY kaput. Így állítottuk elő tehát Y-t, kihasználva, hogy

$Y = \bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + ABCD = (\bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D) + (ABC\bar{D} + ABCD)$

Utóbbi egyenletben az első zárójeles kifejezés három négyváltozós ÉS-nek a VAGY kapcsolata, a második zárójeles kifejezés pedig két négyváltozós ÉS-nek a VAGY kapcsolata, és ez a két VAGY is VAGY kapcsolatban van egymással egy újabb VAGY kapuval. Pontosan ez látható a bal oldali rajzon.

Tekintsünk csak a III-1. ábra jobb oldalára! Mennyivel egyszerűbb! **Olykor sokkal egyszerűbb áramkörrel meg tudjuk oldani feladatunkat, ha előbb egyszerűsítünk**. Ezért a logikai egyenletek egyszerűsítése készségének elsajátítása nagyon indokolt.

IV. Logikai egyenletek

Logikai egyenletek megadása

A digitális technikában nagyon sokszor a feladatokat függvénytáblázatban adják meg, melyet igazságtáblázatnak nevezünk.

- Az igazságtáblázat egy rendszerezett megadási mód, ahol számba vesszük az összes lehetőséget, és megadjuk minden szóba jöhető lehetőségnél, hogy is függ a független változóktól az egy vagy több függő változó.

A rendszerezettség azt jelenti, az igazságtáblázatban **a független változók lehetséges állapotait bináris kód szerint soroljuk fel**, ezek szerint vizsgáljuk a függő változó(k) értékeit. Egy n számú független változójú logikai függvény 2^n féle állapotot vehet fel.

A felsorolás kétféle lehet: logikai szorzatokra, azaz mintermekre, vagy logikai összegekre, azaz maxtermekre vonatkozhat.

- **Minterm** alatt olyan logikai **ÉS**-t értünk, melyben **minden változó egyszer, és csakis egyszer fordul elő** tagadott, vagy nem tagadott formában.
- **Maxterm** alatt olyan logikai **VAGY**-ot értünk, melyben **minden változó egyszer, és csakis egyszer fordul elő** tagadott, vagy nem tagadott formában.

Egy n változós logikai függvénynek tehát 2^n féle mintermje és ugyanennyi maxtermje lehet.

Szabályos (normál) alak

Ezeket mintermes vagy maxtermes alaknak is nevezzük.

- Diszjunktív szabályos alak: olyan függvény, mely mintermek VAGY kapcsolatából áll. Szokás még egyszerűen **mintermes alak**nak, szorzatok összegének, röviden szorzatösszegnek, mintermek összegének is nevezni
- Konjuktív szabályos alak: olyan függvény, mely maxtermek ÉS kapcsolatából áll. Szokás még egyszerűen **maxtermes alak**nak, összegek szorzatának, röviden összegszorzatnak, maxtermek szorzatának is nevezni

IV-1. táblázat: Két változós mintermek és maxtermek

Minterm	Maxterm
$m_0^3 = \overline{A}\overline{B}$	$M_3^3 = A + B$
$m_1^3 = \overline{A}B$	$M_2^3 = A + \overline{B}$
$m_2^3 = A\overline{B}$	$M_1^3 = \overline{A} + B$
$m_3^3 = AB$	$M_0^3 = \overline{A} + \overline{B}$
A mintermek és a maxtermek egymás negáltjai e táblázatban	

Példák mintermről maxtermre alakításra De Morgan tételének felhasználásával $(\overline{X}\overline{Y} = \overline{X + Y})$

$$\overline{A}\overline{B} = m_0^3 = \overline{M_3^3} = \overline{A + B}$$

$$A\overline{B} = m_2^3 = \overline{M_1^3} = \overline{\overline{A} + B}$$

A 3 változós összetartozó mintermek és maxtermek alsó indexeinek összege 3.

IV-2. táblázat: Három változós mintermek és maxtermek

Minterm	Maxterm
$m_0^3 = \bar{A}\bar{B}\bar{C}$	$M_7^3 = A + B + C$
$m_1^3 = \bar{A}\bar{B}C$	$M_6^3 = A + B + \bar{C}$
$m_2^3 = \bar{A}B\bar{C}$	$M_5^3 = A + \bar{B} + C$
$m_3^3 = \bar{A}BC$	$M_4^3 = A + \bar{B} + \bar{C}$
$m_4^3 = A\bar{B}\bar{C}$	$M_3^3 = \bar{A} + B + C$
$m_5^3 = A\bar{B}C$	$M_2^3 = \bar{A} + B + \bar{C}$
$m_6^3 = AB\bar{C}$	$M_1^3 = \bar{A} + \bar{B} + C$
$m_7^3 = ABC$	$M_0^3 = \bar{A} + \bar{B} + \bar{C}$
A mintermek és a maxtermek egymás negáltjai e táblázatban	

Példák mintermről maxtermre alakításra
De Morgan tételének felhasználásával
$(\bar{X}\bar{Y}\bar{Z} = \overline{X + Y + Z})$

$$\bar{A}\bar{B}\bar{C} = m_0^3 = \overline{M_7^3} = \overline{A+B+C}$$

$$\bar{A}B\bar{C} = m_2^3 = \overline{M_5^3} = \overline{A+\bar{B}+C}$$

$$\bar{A}BC = m_3^3 = \overline{M_4^3} = \overline{A+\bar{B}+\bar{C}}$$

$$A\bar{B}C = m_5^3 = \overline{M_7^3} = \overline{\bar{A}+B+\bar{C}}$$

$$ABC = m_7^3 = \overline{M_0^3} = \overline{\bar{A}+\bar{B}+\bar{C}}$$

A 3 változós összetartozó mintermek és maxtermek alsó indexeinek összege 7.

Megállapítások:
Általában is az n változós összetartozó mintermek és maxtermek alsó indexe 2^n-1.
A mintermeket maxtermmé alakítani, és viszont a De Morgan azonosságok szerint lehet.

$$m_i^n = \overline{M_{2^n-1-i}^n} \text{ és } M_i^n = \overline{m_{2^n-1-i}^n}$$

Mi a továbbiakban csak a mintermes alakot használjuk, és a mintermes táblázatokat tárgyaljuk

Igazságtáblázat

Az ilyen táblázatban számba vesszük a független változók az összes lehetőségét, és megadjuk minden lehetőségnél, mi lesz az egy, vagy több függő változó értéke. A független változókat bitekkel (binary digit = kettes számrendszerbeli számjegy) jelöljük a következő módon: Értékük 0, ha a változó negált, különben 1. A változók helyett felvett bináris számjegyek értékei éppen azt a számot jelentik, hányadik sorban fordult elő az adott kombináció.
Az így előállított igazságtáblázat sorai mintermek.

Két változós igazságtáblázat független változóinak rendszerezése, mintermjei

A	B	Jelentés	Érték	Minterm
0	0	$\bar{A}\bar{B}$	0	m_0^2
0	1	$\bar{A}B$	1	m_1^2
1	0	$A\bar{B}$	2	m_2^2
1	1	AB	3	m_3^2

Három változós igazságtáblázat független változóinak rendszerezése, mintermjei

A	B	C	Jelentés	érték	Minterm
0	0	0	$\bar{A}\bar{B}\bar{C}$	0	m_0^3
0	0	1	$\bar{A}\bar{B}C$	1	m_1^3
0	1	0	$\bar{A}B\bar{C}$	2	m_2^3
0	1	1	$\bar{A}BC$	3	m_3^3
1	0	0	$A\bar{B}\bar{C}$	4	m_4^3
1	0	1	$A\bar{B}C$	5	m_5^3
1	1	0	$AB\bar{C}$	6	m_6^3
1	1	1	ABC	7	m_7^3

Ennek az igazságtáblázat sorai mintermek.

Ha a független változókat biteknek (binary digit = kettes számrendszerbeli számjegy) fogjuk fel, a velük képzett bináris számjegyek decimális értékei éppen azt adják, hányadik sorban fordult elő az adott kombináció.

Ugyanez a szám fordul elő a mintermek alsó indexeiben.

Példák igazságtáblázattal és mintermes alakkal való függvénymegadásra.

Ábrázoljuk az $Y = \bar{A}B + A\bar{B}$ antivalencia, azaz XOR függvényt!

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

ha $\bar{A}B = 1$, azaz $m_1^2 = 1$, akkor $Y = 1$

ha $A\bar{B} = 1$, azaz $m_2^2 = 1$, akkor $Y = 1$

Elég csak azt az esetet feltüntetni, ahol, ha igaz az abban a sorban levő minterm, akkor Y = 1

A többi sorban Y = 0.

Írhatjuk tehát egyszerűbben: $Y = m_1^2 + m_2^2 = \sum^2 1,2$. Ennek jelentése: Y akkor igaz, ha igaz a m_1^2 és a m_2^2 minterm, különben hamis. X tehát az igazságtáblázat 1. és 2. sorában igaz.

Ábrázoljuk igazságtáblázattal az $Y = \bar{A}\bar{B} + AB$ ekvivalencia függvényt!

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

$$Y = m_0^2 + m_3^2 = \sum^2 0,3$$

Az igazságtáblázat 0-ik és 3-ik sorában igaz Y

Ábrázoljuk igazságtáblázattal az $Y = \overline{A}\overline{B}$ NAND függvényt!

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

$$Y = m_0^2 + m_1^2 + m_2^2 = \sum^2 0,1,2$$

Y az igazságtáblázat 0., 1. és 2. sorában igaz.

Ábrázoljuk az $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C}$ függvényt!

A	B	C	X
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	0

Ha $\overline{A}\overline{B}C = 1$, azaz $m_1^3 = 1$, akkor $Y = 1$

Ha $\overline{A}B\overline{C} = 1$, azaz $m_2^3 = 1$, akkor $Y = 1$

Ha $A\overline{B}\overline{C} = 1$, azaz $m_4^3 = 1$, akkor $Y = 1$

Itt is elég csak azt az esetet feltüntetni, ahol, ha igaz az abban a sorban levő minterm, akkor Y = 1

A többi sorban Y = 0.

Írhatjuk tehát egyszerűbben: $X = m_1^3 + m_2^3 + m_4^3 = \sum^3 1,2,4$

X az igazságtáblázat 1., 2. és 4. sorában igaz.

Látható, hogy sokkal rövidebb a mintermes alak, mint az igazságtáblázat. Több változó esetén ez még inkább így van, mert n változós igazságtáblázatnak 2^n számú sora van.

Egyszerűbben is áttérhetünk mintermes alakra. A független változókat bitekkel jelöljük a következő módon: Értékük 0, ha a változó negált, különben 1. Ha a legnagyobb helyiértékű számjegynek az A-t vesszük, és ügyelünk a változók sorrendjére, akkor a változók helyett felvett bitek, mint bináris számok értékei éppen a szóba jövő mintermet adják.

Pl. az $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C}$ függvénynél az átalakítással kapjuk:

$$X = \underset{0\,0\,1}{\overset{1}{\overline{A}\overline{B}C}} + \underset{0\,1\,0}{\overset{2}{\overline{A}B\overline{C}}} + \underset{1\,0\,0}{\overset{4}{A\overline{B}\overline{C}}} = m_1^3 + m_2^3 + m_4^3$$

tehát $X = m_1^3 + m_2^3 + m_4^3 = \sum^3 1,2,4$

További példák:

Alakítsuk mintermes alakra az $Y = \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}CD$ függvényt.

$$Y = \underset{0\,0\,1\,1}{\overset{3}{\bar{A}\bar{B}CD}} + \underset{0\,1\,0\,0}{\overset{4}{\bar{A}B\bar{C}\bar{D}}} + \underset{1\,0\,0\,1}{\overset{9}{A\bar{B}\bar{C}D}} + \underset{1\,0\,1\,1}{\overset{11}{A\bar{B}CD}} = m_3^4 + m_4^4 + m_9^4 + m_{11}^4$$

$$Y = m_3^4 + m_4^4 + m_9^4 + m_{11}^4 = \sum^4 3,4,9,11$$

Alakítsuk mintermes alakra az $Y = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + A\bar{B}CD + AB\bar{C}\bar{D} + ABC\bar{D}$ függvényt.

$$Y = \underset{0\,0\,0\,1}{\overset{1}{\bar{A}\bar{B}\bar{C}D}} + \underset{0\,1\,0\,1}{\overset{5}{\bar{A}B\bar{C}D}} + \underset{1\,0\,1\,1}{\overset{11}{A\bar{B}CD}} + \underset{1\,1\,0\,0}{\overset{12}{AB\bar{C}\bar{D}}} + \underset{1\,1\,1\,0}{\overset{14}{ABC\bar{D}}}$$

$$Y = \sum^4 1,5,11,12,14$$

Karnaugh tábla

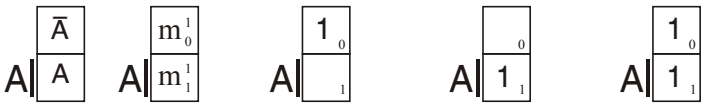
Az ember gondolkodása közelebb áll a grafikus ábrázoláshoz, mint az algebrai egyenletekhez. A kevés, legfeljebb négy változós logikai függvények szabályos ábrázolására alkalmazzák a Karnaugh táblákat.

Az függvényt négyzetekből, cellákból álló táblán ábrázoljuk. Minden cella egy-egy szabályos termet (minterm, vagy maxterm) képvisel. Mi csak a mintermes alakot tanulmányozzuk.

A cellákat úgy helyezik el egymás mellett, hogy a szomszédos cellák termjei csak egyetlen egy változó logikai értékében különbözzenek egymástól. Ez a szomszédosság függőlegesen és vízszintesen is értendő.

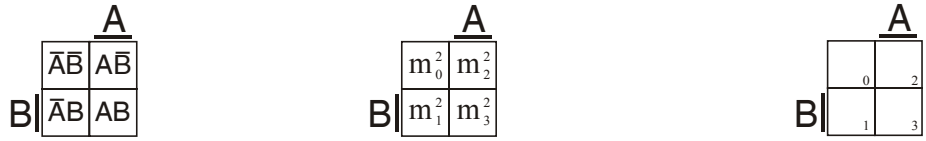
A logikai függvényt úgy írjuk a Karnaugh táblába, hogy amelyik termje 1, az annak a termnek megfelelő cellába 1-et írunk, a többi cellát üresen hagyjuk, azaz a 0-kat nem írjuk be

Egyváltozós Karnaugh tábla



Az egyváltozós Karnaugh táblának nem sok jelentősége van, csak a megértéshez, az alapok lefektetéséhez mutattuk be.

Kétváltozós Karnaugh tábla



Cellák jelentése | Mintermek a cellákban | A mintermek számait kis számokkal jelöljük a cellákban

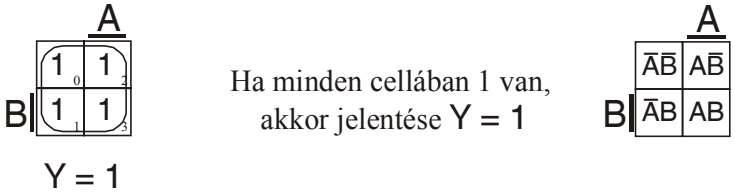
A cellák mintermek szerinti számozása (a jobb oldali táblán) nem kötelező, de segíti a megértést. Ezért e könyvben mindig feltüntetjük őket. Természetesen a rendes munkához, dolgozathoz nem szükséges a feltüntetésük.

Nézzük a kétváltozós Karnaugh táblában az alapokat:

A függőlegesen egymás mellett levő cellákat összevonhatjuk

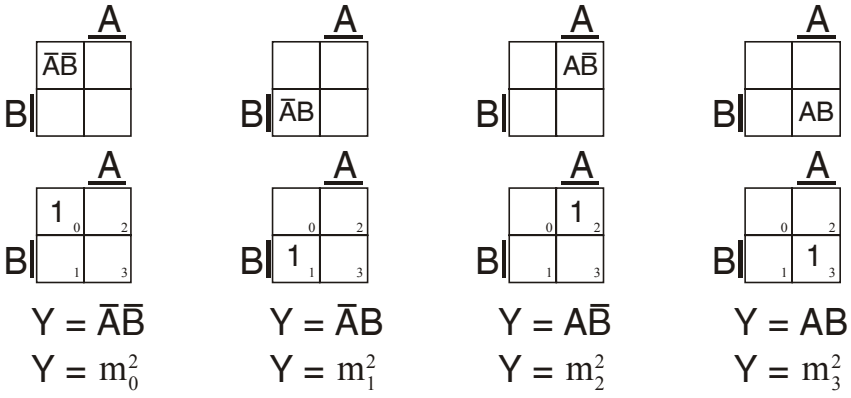
A vízszintesen egymás mellett levő cellákat összevonhatjuk

Az összevonhatóságnak nagy jelentősége van. Az összevonás egyszerűsítést is jelent: amelyik változó mindkét (0 és 1) értékét felveszi az összevonásban, az egyszerűsített alakból kiesik. Ez jól meg is figyelhető a fenti táblákon.
Látni fogjuk, nem csak 2, de 4 cella is összevonható vízszintesen is, függőlegesen is. Lássuk, mit jelent négy egymással vízszintesen és függőlegesen szomszédos cella összevonva:

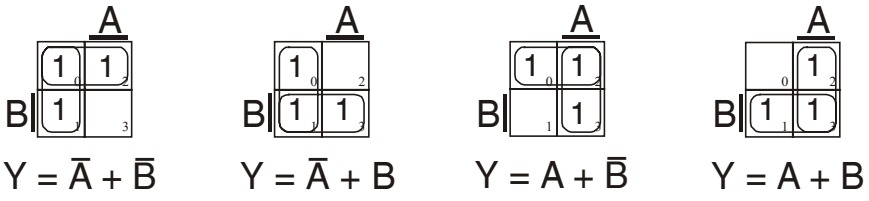


Itt tehát mindkét (most, kétváltozósnál az összes) változó kiesett, mert minden lehetséges változókombinációban a függvény értéke 1, azaz nem függ a változóktól.
Megállapíthatjuk, ha két cellát vonunk össze, egy változó esik ki. Ha pedig négy cellát vonunk össze, két változónk esik ki. Ez a három-, és négyváltozós Karnaugh táblában is így van.

Nézzük meg a többi kétváltozós logikai függvényt a Karnaugh táblába írva:

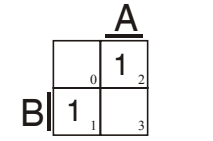


Logikai szorzatok a Karnaugh táblában.

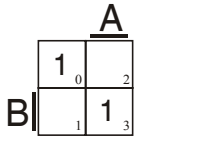


Logikai összegek a Karnaugh táblában.

Antivalencia (XOR)



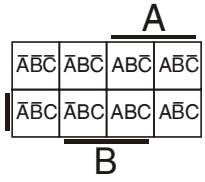
Ekvivalencia



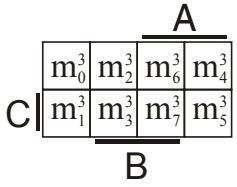
Háromváltozós Karnaugh tábla

A három változó már $2^3 = 8$ lehetséges állapotot vehet fel, így 8 cellás Karnaugh tábla kell. Ez lehet 2-szer 4, vagy 4-szer 2 cellás tábla.

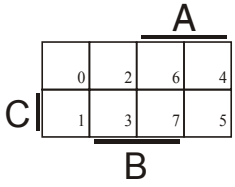
Két soros 4 oszlopos Karnaugh tábla



Cellák jelentése

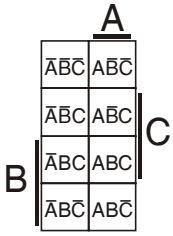


Mintermek elhelyezkedése a cellákban

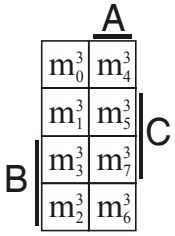


A mintermek számaival jelöljük a cellákat

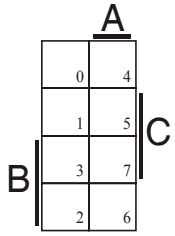
4 soros 2 oszlopos Karnaugh tábla



Cellák jelentése



Mintermek elhelyezkedése a cellákban



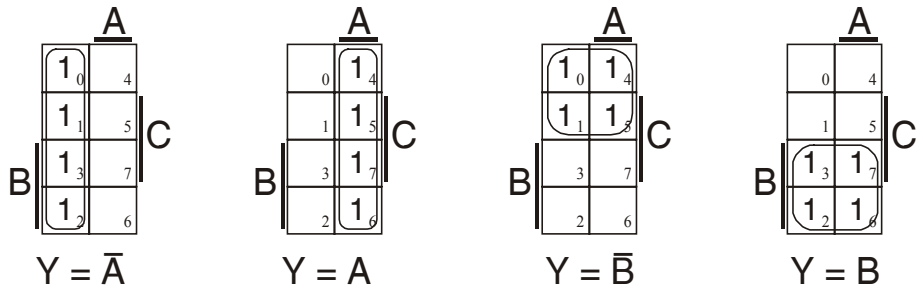
A mintermek számaival jelöljük a cellákat

A különböző peremezésű, és álló, vagy fekvő Karnaugh táblák egymásba alakíthatók tükrözéssel, a betűk cseréjével, forgatással, mely műveletek a logikai jelentést nem módosítják.

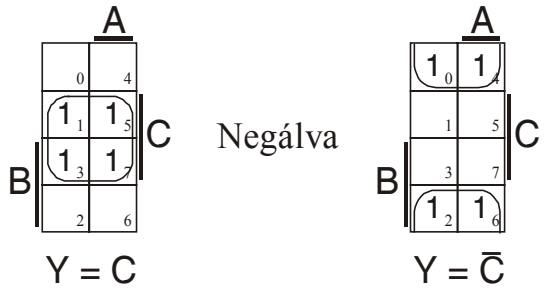
Mi a továbbiakban a 4 soros 2 oszlopos háromváltozós Karnaugh táblákkal dolgozunk, mert ezek celláinak számozása sorrendje hasonlít iskolánk Karnaugh programjainak cella számozásához. Ez a számozás megegyezik az általunk használt négyváltozós Karnaugh táblák első két oszlopának cellasorszámaival.

Ha alaposabban megfigyeltük az egy-, és kétváltozós Karnaugh tábla sajátosságait, ezeket kiterjeszthetjük a háromváltozósra is. Azokat az egymás melletti cellákat, ahol a mintermek

csak egy változóban különböznek, az előzőek alapján összevontuk. Itt már egy irányban négy cellát is össze tudunk vonni.

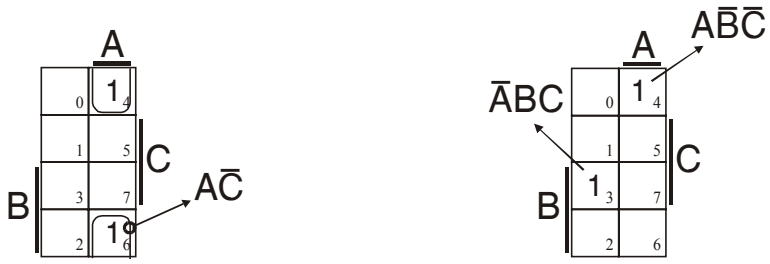


Ezek megegyeznek a kétváltozós Karnaugh táblán látottakkal. Azonban a $\bar{C}$ változó már egy eddig szokatlan összevonás eredménye:

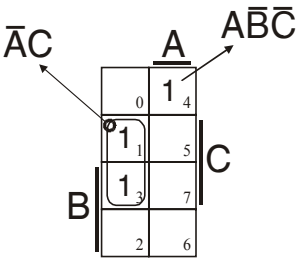


A a $\bar{C}$ változót a C negálásából kapjuk. Amint az ábrán látható, a $\bar{C}$ változót tartalmazó mintermek cellái már a tábla alsó és felső szélein vannak. Ezért ezeket is össze tudtuk vonni. Láthatjuk, **a Karnaugh tábla szélei is szomszédosak egymással**. A szélén (alján és tetején) levő mezőket is összevonhatjuk.

Természetesen nem csak egy, hanem mind a három változót tartalmazhatja a háromváltozós Karnaugh tábla. Ezután ahol lehet, az összevonható cellákat össze is vonjuk. Lássunk további példákat háromváltozós Karnaugh táblára. Azt érthetőség kedvért a táblákkal ábrázolt logikai függvények egyenlete alá a mintermes alakot feltüntettük.

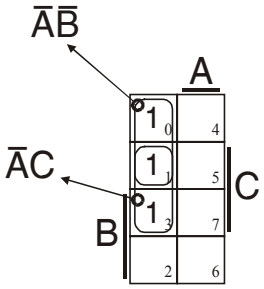


$Y = m_4^3 + m_6^3$ $Y = m_3^3 + m_4^3$



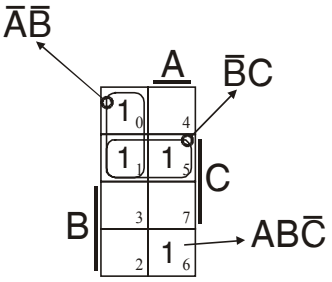
$Y = \bar{A}C + A\bar{B}\bar{C}$

$Y = m_1^3 + m_3^3 + m_4^3$



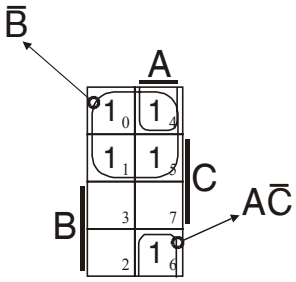
$Y = \bar{A}\bar{B} + \bar{A}C$

$Y = m_0^3 + m_1^3 + m_3^3$



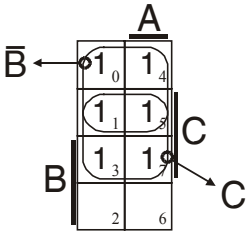
$Y = \bar{A}\bar{B} + \bar{B}C + AB\bar{C}$

$Y = m_0^3 + m_1^3 + m_5^3 + m_6^3$

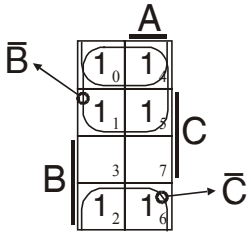


$Y = A\bar{C} + \bar{B}$

$Y = m_0^3 + m_1^3 + m_4^3 + m_5^3 + m_6^3$



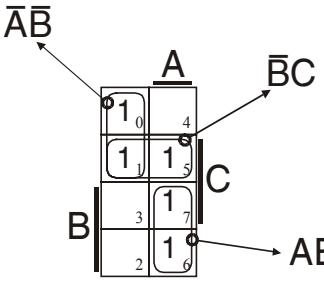
$Y = \bar{B}C$



$Y = \bar{C}$

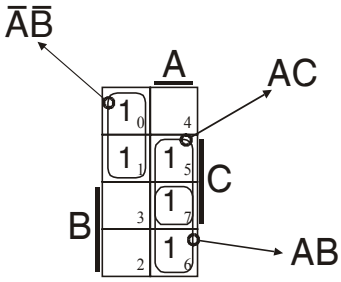
Hat cellát nem lehet összevonni!

Olyan is előfordulhat, hogy ugyanazt a táblát többféleképpen olvashatjuk ki jól. Tekintsük az alábbi példát:



$Y = \bar{A}\bar{B} + AB + \bar{B}C$

$Y = m_0^3 + m_1^3 + m_5^3 + m_6^3 + m_7^3$



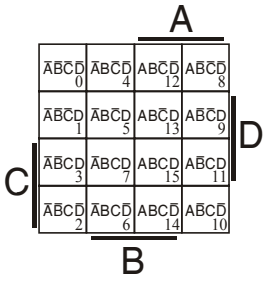
$Y = \bar{A}\bar{B} + AB + AC$

$Y = m_0^3 + m_1^3 + m_5^3 + m_6^3 + m_7^3$

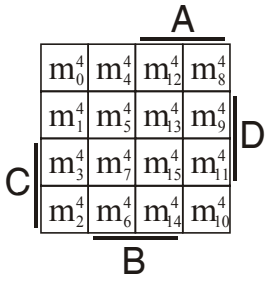
Mindkét egyszerűsített egyenlet helyes.

Négyváltozós Karnaugh tábla

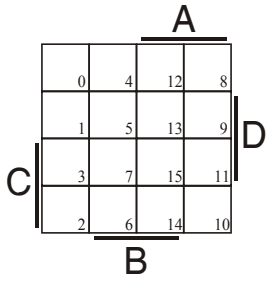
Itt már vízszintesen is, függőlegesen is négy-négy cella van, így



Cellák jelentése

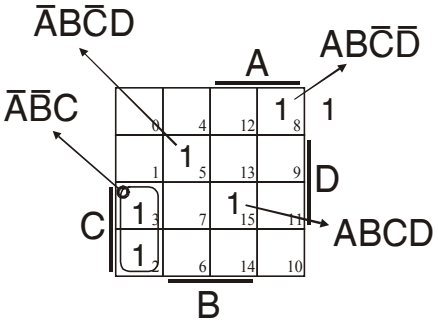


Mintermek elhelyezkedése a cellákban

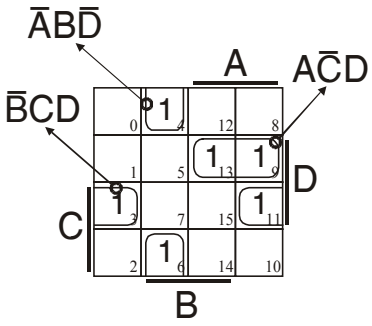


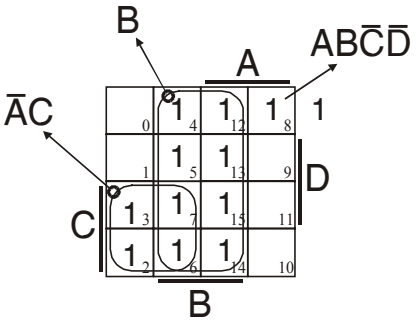
A mintermek számaival jelöljük a cellákat

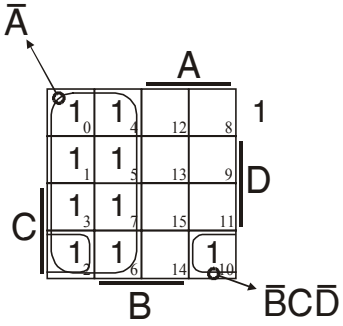
Ha alaposabban megfigyeltük az egy-, két-, és háromváltozós Karnaugh tábla sajátosságait, ezeket kiterjeszthetjük a négyváltozósra is. Azokat az egymás melletti cellákat, ahol a mintermek csak egy változóban különböznek, az előzőek alapján összevonhatjuk egy irányba kettőt, vagy négyet. Példákon mutatjuk meg a négyváltozós Karnaugh tábla használatát és a lehetőségeket, figyelembevéve, hogy a miket figyelhettünk meg egy-, két-, és háromváltozós táblánál:



$Y = \bar{A}BC + AB\bar{C}$







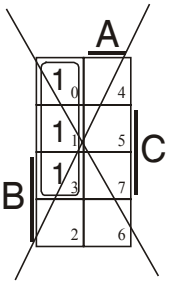
A mintermeket már nem tüntettük fel, az eddigiek alapján egyszerűen kiolvashatók a táblából (ahol 1-es áll, annak a cellának megfelelő számú minterm szerepel).

A logikai egyenletet sem tüntettük fel, egyszerűen össze kell olvasni a nyilakkal jelzett összevonások és szomszéd nélküli 1-esek jelentését.

Súlyos hibák, rossz összevonások a Karnaugh táblákban

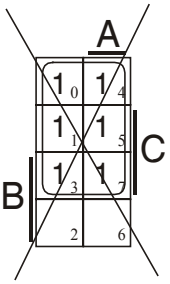
Ezekre azért hívjuk el a figyelmet, mert felületesebb diákok gyakran elkövetik őket. Ez ilyen rosszul felállított egyenlet nyomán olykor rosszul működő áramkörök születnek, ami felesleges fáradság, sőt, kár.

Hiba!



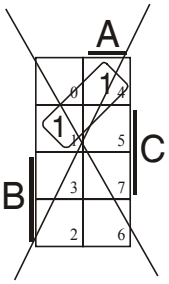
Három cellát nem lehet összevonni!

Hiba!



Hat cellát nem lehet összevonni!

Hiba!

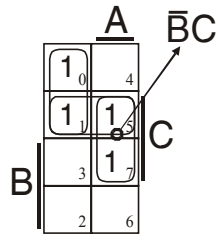


Átlósan nem lehet összevonni!

Kevésbé súlyos hibák a Karnaugh táblában

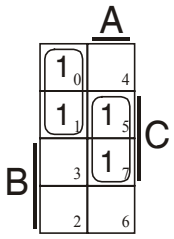
Az ilyen hibák nem okoznak hibás működést, csak drágábbá teszik a megvalósítást, mert több logikai kapu kell, ami többe kerül, nagyobb áramkört jelent, több helyet igényel, és több villamos teljesítményt fogyaszt.

Hibás tábla, $\bar{B}C$ felesleges



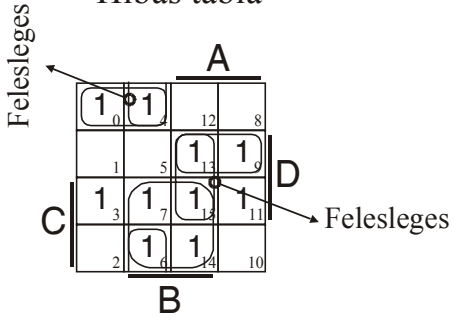
$Y = \bar{A}\bar{B} + AC + \bar{B}C$

Helyesen



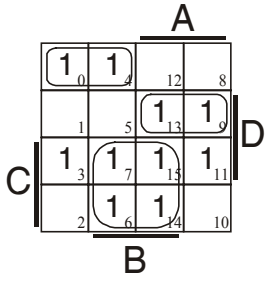
$Y = \bar{A}\bar{B} + AC$

Hibás tábla



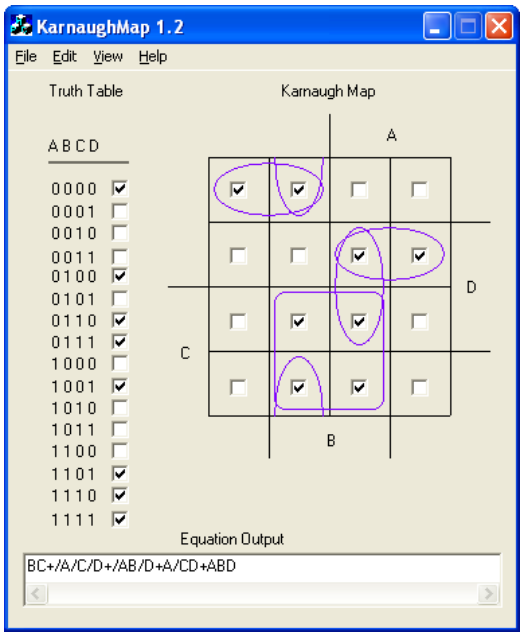
$Y = \bar{A}C\bar{D} + \bar{A}B\bar{D} + A\bar{C}D + \bar{A}\bar{B}D + BC$

Helyesen

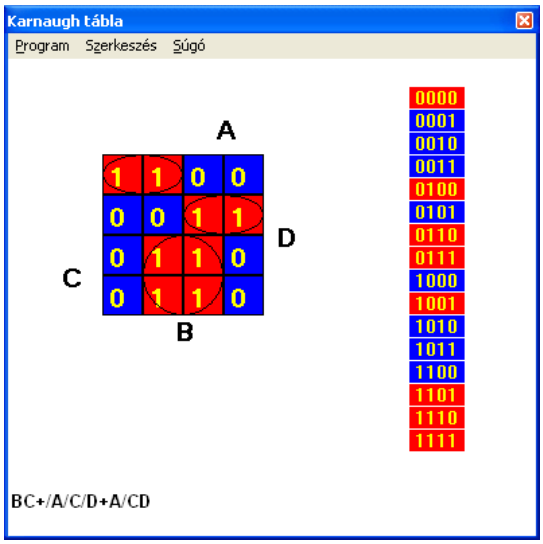


$Y = \bar{A}C\bar{D} + A\bar{C}D + BC$

Egyes Karnaugh tábla programok shareware verziói is szándékosan el vannak rontva, hogy az ingyenes verziók helyesen, de drágábban működjenek (ne lehessen velük pénzt keresni). Pl. felesleges összevonásokat készít a Karnaughmap12 nevű program (kmap12.exe fájlnévvel), melynek pénzért lehet megvenni a teljesen jól működő verzióját:



Iskolánk Nagy Attila nevű diákja által passzióból készített Karnaugh tábla nevű programja nem készít felesleges karikát
(Nagy Attila 2003. febr.–márc., 1.0 beta verzió,)



A Karnaugh táblából kiolvasható egyszerűbb alakú logikai függvényeket sokszor tovább lehet egyszerűsíteni, erre látunk módszert és példákat a *További egyszerűsítések* című fejezetben *(IV.23 oldal)*. Az egyszerűsítés legfontosabb eszköze legfeljebb négy változóig a Karnaugh tábla.

Látni fogjuk, a Karnaugh tábla kiváló segédeszköz a logikai függvények szabályos alakra hozásánál (lásd *Szabályos alakra hozás* című fejezet *IV.14.* oldal).

Négy változó felett nem alkalmazzák a Karnaugh táblát, mert 5 és hat változónál háromdimenziós, még nagyobb változószámnál többdimenziós, nehezen elképzelhető térbeli alakzat lenne, és éppen a legfontosabb tulajdonságát, az áttekinthetőséget veszítenénk el.

Nagy változószámra szisztematikus eljárásokat alkalmaznak, mint pl. a Quine – McCluskey-módszer. Mi ezeket nem tanulmányozzuk, a szakirodalomban, Interneten fellelhetők, ha valakinek nem elég e könyv által adott információ.

Logikai függvények átalakítása

Szabályos alakra hozás

A logikai függvények szabályos alakjának nagy jelentősége van, mert a logikai hálózatok tervezésénél használt szisztematikus eljárásokat csak szabályos alakban adott függvényekkel lehet elvégezni (mint általában mindent a gépi feldolgozás során). Csak szabályos alakban megadott logikai függvényt lehet programozható logikai áramkörökbe (PLA, ULA, ROM) programozni, így a szabályos alakra hozás készsége fontos.

Nem szabályos alakú logikai függvényeket a Boole algebra tételeivel és azonosságaival lehet szabályos alakra hozni. Ehhez ajánlott áttekinteni az *A Boole algebra azonosságai és tételei (III.4. oldal)* alfejezetet.

A legfontosabb két tételt emlékeztetőül ide is felírjuk:

De Morgan tételei:

$\overline{A+B} = \overline{A}\,\overline{B}$, ill. akárhány tagra: $\overline{A+B+...+N} = \overline{A}\,\overline{B}...\overline{N}$

$\overline{AB} = \overline{A} + \overline{B}$, ill. akárhány tagra: $\overline{AB...N} = \overline{A} + \overline{B} + ... + \overline{N}$

$(\overline{X} + X) = 1$, és 1-gyel bármit lehet szorozni. Ezzel a módszerrel a hiányzó változókkal kiegészíthetjük azokat a tagokat, melyekből hiányzik valamelyik változó ponált vagy negált formában.

A Karnaugh táblát is használhatjuk a szabályos alakra hozásnál
legfeljebb négy változóig természetesen.

Példák a szabályos alakra hozásra:

Négy változónál többet nem tanulmányozunk, így elég lenne a Karnaugh táblával való feladatmegoldás. Azonban a gyakorlatban előforduló logikai feladatoknál olykor négynél sokkal több változó van, így az algebrai módszerrel való szabályos alakra hozást fontos gyakorolnunk. Egyes feladatokat a Karnaugh táblával is megoldunk. Lássunk néhány feladatot:

$Y = \overline{A}BC + \overline{B}\overline{C}$ nem szabályos, hozzuk szabályos alakra

$Y = \overline{A}BC + \overline{B}\overline{C} = \overline{A}BC + (\overline{A} + A)\overline{B}\overline{C}$ a második tagot szoroztuk $1 = (\overline{A} + A)$-val

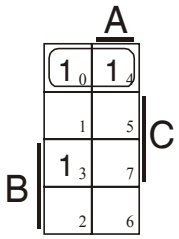
$Y = \overline{A}BC + \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C}$ Ezt rendezve kapjuk

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

$Y = \sum^{3} 0,3,4$

Megoldás Karnaugh táblával:

Írjuk be a táblába $Y = \bar{A}BC + B\bar{C}$ egyenletet, olvassuk ki a cellákhoz tartozó mintermeket, és készen is vagyunk



A Karnaugh táblából egyszerűen kiolvashatjuk a három mintermet.

$$Y = \sum^{3} 0,3,4$$

$$Y = \bar{A}BC + B\bar{C}$$

$X = \bar{A}BC + \bar{B}\bar{C}D + A\bar{B}CD$ nem szabályos, hozzuk szabályos alakra!

$$X = \bar{A}BC(D + \bar{D}) + (\bar{A} + A)\bar{B}\bar{C}D + A\bar{B}CD$$

$X = \bar{A}BCD + \bar{A}BC\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}\bar{C}D + A\bar{B}CD = \sum^{4} 7,6,1,9,11$ Ezt rendezve kapjuk

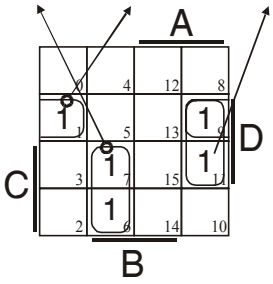
$$X = \bar{A}\bar{B}\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}D + A\bar{B}CD$$

$$X = \sum^{4} 1,6,7,9,11$$

Megoldás Karnaugh táblával:

Írjuk be a táblába az $X = \bar{A}BC + \bar{B}\bar{C}D + A\bar{B}CD$ egyenletet, olvassuk ki a cellákhoz tartozó mintermeket, és készen is vagyunk.

$$X = \bar{A}BC + \bar{B}\bar{C}D + A\bar{B}CD$$



A Karnaugh táblából egyszerűen kiolvashatjuk a mintermeket.

$$X = \sum^{4} 1,6,7,9,11$$

$Z = \bar{A} + \bar{B}\bar{C}$ nem szabályos, hozzuk szabályos alakra.

$Z = \bar{A} + \bar{B}\bar{C}$ $\bar{A}$-t $(\bar{B} + B)(\bar{C} + C)$-vel kell szorozni (bővíteni), míg $\bar{B}\bar{C}$-t $(\bar{A} + A)$-val

$Z = \bar{A}(\bar{B} + B)(\bar{C} + C) + (\bar{A} + A)\bar{B}\bar{C}$

$Z = \bar{A}\bar{B}(\bar{C} + C) + \bar{A}B(\bar{C} + C) + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$

$Z = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$

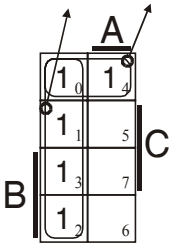
$Z = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C}$

$$Z = \sum^{3} 0,1,2,3,4$$

Megoldás Karnaugh táblával:

Írjuk be a táblába az $X = \bar{A}BC + \bar{B}\bar{C}D + A\bar{B}CD$ egyenletet, olvassuk ki a cellákhoz tartozó mintermeket, és készen is vagyunk.

$Z = \bar{A} + \bar{B}\bar{C}$



A Karnaugh táblából egyszerűen kiolvashatjuk a mintermeket.

$$Z = \sum^{3} 0,1,2,3,4$$

$M = \overline{AB} + \bar{B}\bar{C}$ nem szabályos, hozzuk szabályos alakra!

Itt előbb a NAND tagot alakítjuk összeggé a De Morgan tétel alkalmazásával:

$M = \overline{AB} + \bar{B}\bar{C}$

$M = \bar{A} + \bar{B} + \bar{B}\bar{C}$ Ezeket tagonként három változósra kibővítve kapjuk:

$M = \bar{A}(\bar{B} + B)(\bar{C} + C) + (\bar{A} + A)\bar{B}(\bar{C} + C) + (\bar{A} + A)\bar{B}\bar{C}$

$M = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C$

$$M = \sum^{3} 0,1,2,3,4,5$$

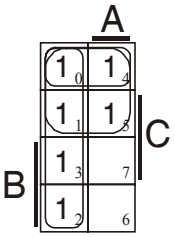
Megoldás Karnaugh táblával:

Most nem tudjuk beírni a táblába az $M = \overline{A\bar{B}} + \bar{B}\bar{C}$ egyenletet. Ezért előbb a NAND tagot alakítjuk összeggé a De Morgan tétel alkalmazásával:

$M = \overline{A\bar{B}} + \bar{B}\bar{C} = \bar{A} + \bar{B} + \bar{B}\bar{C}$.

Ezt már be tudjuk írni a Karnaugh táblába, kiolvassuk a cellákhoz tartozó mintermeket, és készen is vagyunk.

$M = \bar{A} + \bar{B} + \bar{B}\bar{C}$



A Karnaugh táblából egyszerűen kiolvashatjuk a mintermeket.

$M = \sum^{3} 0,1,2,3,4,5$

$\bar{B}\bar{C}$ elhagyható, mert $\bar{B}$ tartalmazza

$U = A\bar{B}C + \overline{B + \bar{C}}$ nem szabályos, hozzuk szabályos alakra!

Itt előbb a NOR tagot alakítjuk szorzattá a De Morgan tétel alkalmazásával:

$U = A\bar{B}C + \bar{B}\bar{\bar{C}}$

$U = A\bar{B}C + \bar{B}C = \bar{B}C$ (abszorpció). $\bar{B}C$-t $(\bar{A} + A)$-val szorozva

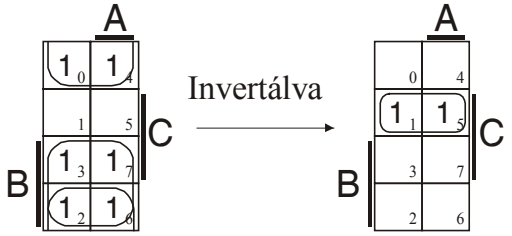
$U = (\bar{A} + A)\bar{B}C = A\bar{B}C + \bar{A}\bar{B}C$

$U = \sum^{3} 1,5$

Megoldás Karnaugh táblával:

Most nem tudjuk beírni a táblába az $U = A\bar{B}C + \overline{B + \bar{C}}$ egyenletet.
Gondoljuk meg, a NOR az OR tagadása. Írjuk tehát előbb a NOR tag változóit, azaz a $B + \bar{C}$ függvényt a Karnaugh táblába, majd tagadjuk meg, invertáljuk a tábla celláit. Így jutunk el a NOR taghoz.
Ehhez még beírhatnánk az $A\bar{B}C$ tagot, ha $\bar{B}C$ nem tartalmazná. Ezután már csak kiolvassuk a cellákhoz tartozó mintermeket, és készen is vagyunk.

$B + \bar{C}$ $\overline{B + \bar{C}} = \bar{B}C$



A Karnaugh táblából egyszerűen kiolvashatjuk a mintermeket.

$U = \sum^{3} 1,5$

$V = \overline{AB\bar{C} + \bar{A}B\bar{C}} + AB\bar{C}$ nem szabályos, hozzuk szabályos alakra!

Itt előbb a NOR tagot alakítjuk szorzattá a De Morgan tétel alkalmazásával:

$V = \overline{AB\bar{C}} \cdot \overline{\bar{A}B\bar{C}} + AB\bar{C}$ A NAND tagokat tovább alakítjuk összegekké

$V = (\bar{A} + \bar{B} + C)(A + B + \bar{C}) + AB\bar{C}$ Végezzük el a szorzást

$V = \bar{A}A + A\bar{B} + AC + \bar{A}B + \bar{B}B + BC + \bar{A}\bar{C} + \bar{B}\bar{C} + C\bar{C} + AB\bar{C}$ Rendezve:

$V = A\bar{B} + AC + \bar{A}B + BC + \bar{B}\bar{C} + AB\bar{C}$ Bővítsük a tagokat háromváltozósra:

$V = (A\bar{B}\bar{C} + AB\bar{C}) + (A\bar{B}C + ABC) + (\bar{A}B\bar{C} + \bar{A}BC) + (\bar{A}BC + ABC) + (\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}) +$

$+ AB\bar{C}$ Zárójellel jelöltem a tagonkénti bővítést az érthetőség kedvéért. Tehát

$V = A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}C + ABC + \bar{A}B\bar{C} + \bar{A}BC + \bar{A}BC + ABC + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + AB\bar{C}$

$V = \sum^{3} 4,5,5,7,2,3,3,7,0,4,5$ Így könnyebben átlátható. Ez rendezve:

$V = \sum^{3} 0,2,3,4,5,7$ vagy egyenlettel:

$V = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C} + ABC$

Megoldás Karnaugh táblával:

Most nem tudjuk beírni a táblába az $V = \overline{AB\bar{C} + \bar{A}B\bar{C}} + AB\bar{C}$ egyenletet.

Az előző példa alapján azonban beírhatjuk a táblába a NOR függvényt egyből. Ezt úgy érjük el, hogy ahol $AB\bar{C} + \bar{A}B\bar{C}$ IGAZ, oda 0-át írunk és ahol az $AB\bar{C} + \bar{A}B\bar{C}$ NEM IGAZ, oda írunk 1-est. (Vagy beírjuk az $AB\bar{C} + \bar{A}B\bar{C}$-t és invertáljuk). Ehhez még oda-írnánk a $AB\bar{C}$ –t, ha már nem tartalmazná $\overline{AB\bar{C} + \bar{A}B\bar{C}}$, és megvagyunk a Karnaugh táblába írással. Csak kiolvassuk a mintermeket és készen vagyunk.

$\overline{AB\bar{C} + \bar{A}B\bar{C}}$

B \ A		
	1_0	1_4
C	0_1	1_5
	1_3	1_7
	1_2	0_6

A Karnaugh táblából egyszerűen kiolvashatjuk a mintermeket.

$V = \sum^{3} 0,2,3,4,5,7$

$\overline{AB\bar{C} + \bar{A}B\bar{C}}$ tartalmazza $AB\bar{C}$-t is

Logikai függvények egyszerűsítése

E fejezetben az a célunk, hogy nem szabályos, hanem a lehető legegyszerűbb alakot kapjuk. Ennek a legegyszerűbb alakú logikai függvénynek a kapuáramkörökkel való megvalósítása a legolcsóbb, legkevesebb kapuból álló áramkört jelenti, melynek a helyigénye és villamos energiafogyasztása is a legkisebb.

Az egyszerűsítésnek olyan nagy a jelentősége, hogy önálló fejezetet szentelünk e témának.

Egyszerűsítés Karnaugh táblával

A Karnaugh tábla tulajdonságait eddig is használtuk, de nem egyszerűsítésre, hanem a mintermek kiolvasására. Azonban láthattuk, sokszor egyszerűbb alakot kaptunk, mint amilyen alakban az egyenletet eredetileg kaptuk.

Hogy is végezzük el az egyszerűsítést a Karnaugh tábla segítségével?

- A logikai egyenletet beírjuk a Karnaugh táblába
  Ha nem tudjuk egyből beírni, algebrai módszerrel olyan alakra hozzuk, hogy már be tudjuk írni. Lásd a *Szabályos alakra hozás* c. fejezetet.
- A lehetséges összevonásokat elvégezzük úgy, hogy a lehető legnagyobb csoportokat kapjuk.

A vízszintesen, vagy függőlegesen szomszédos cellákat össze lehet vonni. 1x2-es, 1x4-es, 2x2-es, 2x4-es, és 4x4-es csoportot is képezhetünk vízszintesen, vagy függőlegesen. **Azaz 2, vagy 4 egymás melletti mezőt vonhatunk össze egy csoporttá**.

Ha két mezőt vonunk össze egy csoporttá, belőlük egy változó esik ki, melyek a csoportban mindkét lehetséges értéküket felveszik, így a csoporton belül ők mindig igazak, azaz 1-gyel helyettesíthetők. Az elv: $1 = (\bar{X} + X)$.
Ha négy mezőt vonunk össze egy csoporttá, belőlük két változó esik ki (melyek mind a négy lehetséges állapotukat felveszik). Az elv: $1 = (\bar{X}\bar{Y} + \bar{X}Y + X\bar{Y} + XY)$.
Ha nyolc mezőt vonunk össze egy csoporttá, belőlük már három változó, míg ha 16 cellát, akkor négy változó esik ki.

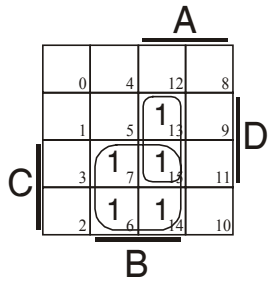
Példa Karnaugh táblával való egyszerűsítésre

A *Példák kapuk alkalmazására* c. fejezetben már láttuk a következő feladatot, melyet ott kapuáramkörökkel is megvalósítottunk. Ígértük, e fejezetben megmutatjuk, hogy egyszerűsítettük le sokkal olcsóbb megoldású áramkörre, mint amit úgy kapunk, ha a feladott egyenletet valósítjuk meg egyből, egyszerűsítés nélkül.

Valósítsuk meg az alábbi logikai egyenletet (függvényt) kapukkal:

$Y = \bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + ABCD.$

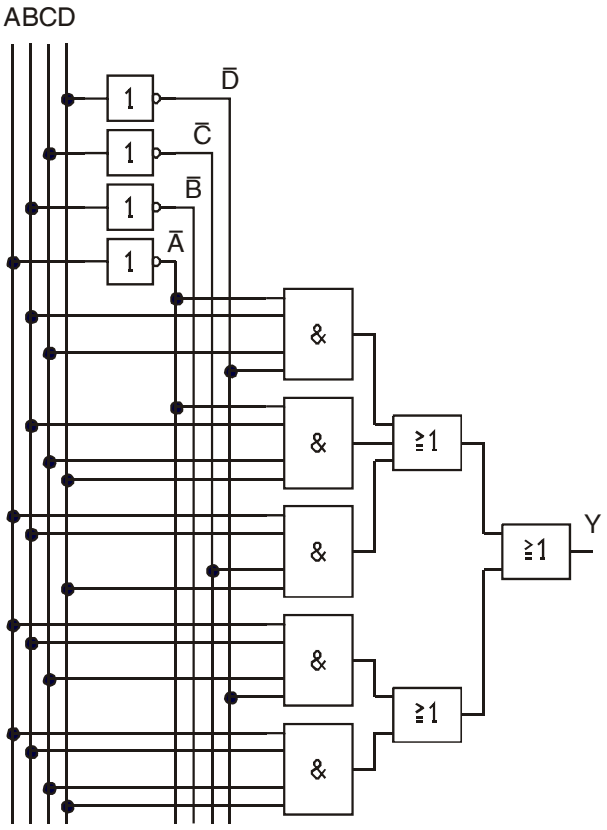
Ezt írjuk Karnaugh táblába:



$$V = \sum^{3} 6,7,13,14,15$$

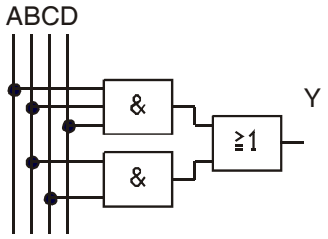
Tehát $Y = \bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + ABCD$ egyszerűsítve: $Y = ABD + BC$

Érdemes megfigyelni, mennyivel egyszerűbb áramkört kaptunk!



$Y = \bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + ABCD$

Egyszerűsítve sokkal kevesebb kapuval meg lehet oldani ugyanezt az egyenletet



$Y = \bar{A}BC\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + ABCD$

$Y = ABD + BC$

Az egyszerűsítést érdemes elvégezni, mert sokkal olcsóbb és kisebb fogyasztású, meg gyorsabb áramkört kaphatunk.

A meg nem engedett állapotok kihasználása egyszerűsítésre

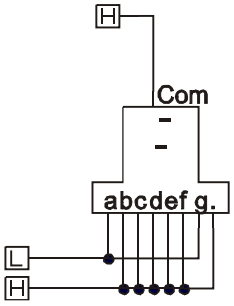
Meg nem engedett állapotokkal eddig is találkoztunk. Pl. a BCD kódnál a pszeudotetrádok is ilyenek voltak. Ezek nem fordulhatnak elő, ezért nem szoktuk a logikai függvény megadásánál definiálni őket. Eddig nem írtuk elő, ezeknél az állapotoknál a függvény értéke 1, vagy 0 leegyen, nem foglalkoztunk velük. Olykor azonban hasznos lehet, ha ezt is előírjuk, így egyszerűbb áramköröket tervezhetünk. Mivel kétértékű logikáról beszélünk, ezek az elő nem forduló állapotok esetén a függvényünk értéke vagy 0, vagy 1 **lenne**. Ha tehát egyszerűsítés céljából **0-t, vagy 1-et határozunk meg nekik**, nem hamisítjuk meg a feladatot, de valamelyik értéket úgyis felvennék (gondolatban persze), és különben úgysem fordulhatnak elő a valóságban. Így további egyszerűsítéseket hajthatunk végre.

Két példát mutatunk a meg nem engedett állapotok kihasználására egyszerűsítés céljából. Az egyik a *BCD kódból 7 szegmenses kijelzőre dekóder tervezése*, *(IV.21 oldal)*, a másik a *Dominó pöttyeinek megjelenítése (IV.26. oldal)*. Ezek a példák természetesen más egyszerűsítéseket is tartalmaznak, mellettük alkalmazzuk a meg nem engedett állapotokra tetszésünk szerint felvett értékeket.

BCD kódból 7 szegmenses kijelzőre dekóder tervezése

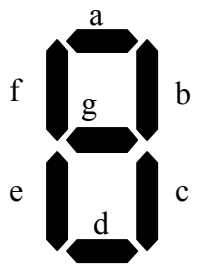
Ehhez bemutatjuk a hétszegmenses kijelzőt:
Az ilyen kijelzőnek hét szegmense van, ezek láthatóvá válnak, ha aktívak, különben nem láthatók (nem vesszük őket figyelembe). Így a hétszegmenses kijelző számokat képes megjeleníteni. A kijelző szegmenseit egyezményesen betűkkel jelölik, az ABC első hét betűjével.

A Tina áramkörszerkesztő program 7 szegmenses kijelzője



Alacsony, azaz L szintre válnak aktívvá a szegmensek.

Az ábrán L szintű csak az “a” és a “g” szegmens, tehát e két szegmens aktív, a többi nem.



A szegmensek betűjelei

A hét szegmens működésének igazságtáblája

Írjuk be a táblázatba, hogy a kijelző hét szegmense a különböző számjegyek megjelenítése esetén milyen logikai értéket kap. A megjelenített számjegyeket a táblázatban és a táblázat mellett nagyobb ábrákkal is felvettük. Az a, b, ... g szegmenseket L_a, L_b, ...L_g jelekkel (lámpákkal, LED-ekkel, stb.) jelöljük a táblázatban.

Dec.	Jel	A	B	C	D	L_a	L_b	L_c	L_d	L_e	L_f	L_g
0	0	0	0	0	0	L	L	L	L	L	L	H
1	1	0	0	0	1	H	L	L	H	H	H	H
2	2	0	0	1	0	L	L	H	L	L	H	L
3	3	0	0	1	1	L	L	L	L	H	H	L
4	4	0	1	0	0	H	L	L	H	H	L	L
5	5	0	1	0	1	L	H	L	L	H	L	L
6	6	0	1	1	0	L	H	L	L	L	L	L
7	7	0	1	1	1	L	L	L	H	H	H	H
8	8	1	0	0	0	L	L	L	L	L	L	L
9	9	1	0	0	1	L	L	L	L	H	L	L
10	X	X	X	X	X	*	*	*	*	*	*	*
11	X	X	X	X	X	*	*	*	*	*	*	*
12	X	X	X	X	X	*	*	*	*	*	*	*
13	X	X	X	X	X	*	*	*	*	*	*	*
14	X	X	X	X	X	*	*	*	*	*	*	*
15	X	X	X	X	X	*	*	*	*	*	*	*

Az X-szel jelölt állapotok nem fordulhatnak elő, ezek a pszeudotetrádok
A *-gal (csillaggal) jelölt állapotok tetszésünk szerint lehetnek H, vagy L

Ha L-logikai szintnek a 0-át, H logikai szintnek az 1-et vesszük, akkor a következő egyenleteket kapjuk:

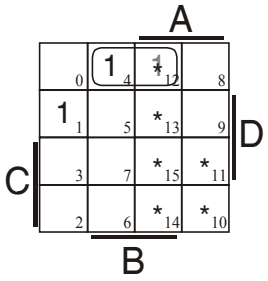
$$L_a = \sum^4 1,4 \;;\quad L_b = \sum^4 5,6 \;;\quad L_c = m_2^4 \;;\quad L_d = \sum^4 1,4,7 = L_a + m_7^4 \;;$$

$$L_e = \sum^4 1,3,4,5,7,9 = L_d + \sum^4 3,5,9 \qquad L_f = \sum^4 1,2,3,7 \;;\quad L_g = \sum^4 0,1,7$$

Ezeket egyszerűsítsük Karnaugh táblával. A Karnaugh táblában is csillaggal jelöltem a pszeudotetrádokat, melyek értéke tehát tetszésünk szerint lehet 0, vagy 1. Az ilyen 1-est szürkével jelöltük

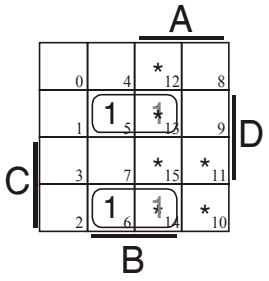
$L_g = \bar{A}\bar{B}\bar{C} + BCD$

L_a



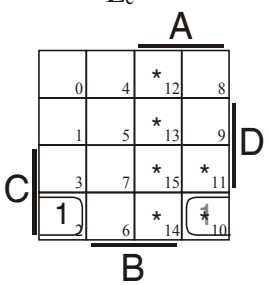
$L_a = \bar{A}\bar{B}\bar{C}D + B\bar{C}\bar{D}$

L_b



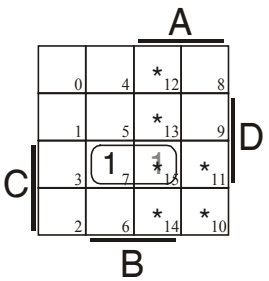
$L_b = BC\bar{D} + B\bar{C}D$

L_c



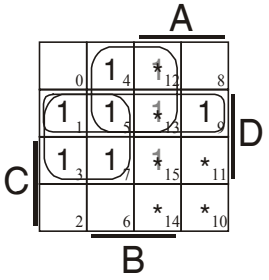
$L_c = \bar{B}C\bar{D}$

m_7^4



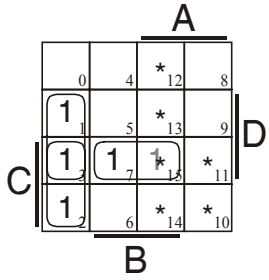
$m_7^4 = BCD$

L_e



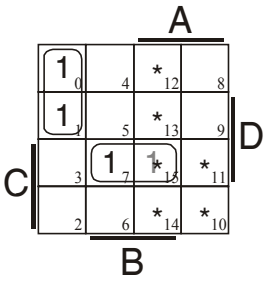
$L_e = \bar{A}D + B\bar{C} + \bar{C}D$

L_f



$L_f = \bar{A}\bar{B}C + \bar{A}\bar{B}D + BCD$

L_g

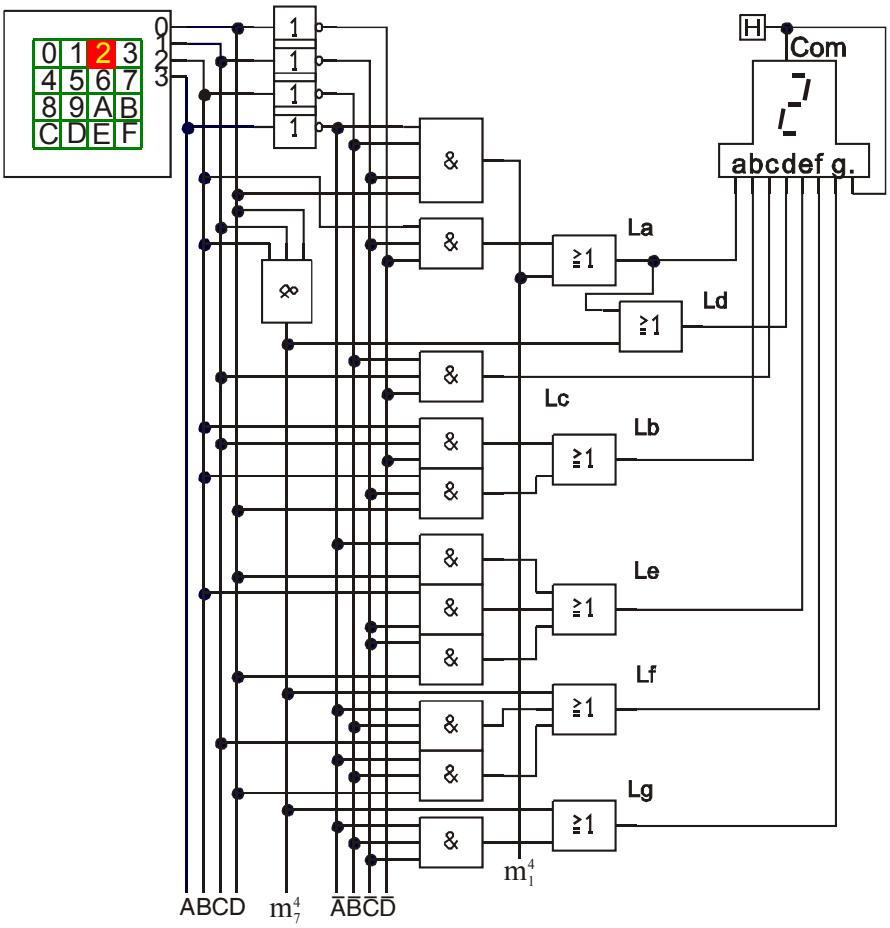


$L_g = \bar{A}\bar{B}\bar{C} + BCD$

Érdemes külön megvalósítani a m_7^4 mintermet, és így több helyen felhasználni.
Tehát az L_d-nél, mert $L_d = L_a + m_7^4$, az L_f-nél és az L_g-nél

A következő oldalon látható ezek megvalósítása.

BCD kódról 7 szegmensesre dekódoló áramkör elemi kapukkal



Ehhez hasonló az áramköröket (hétszegmenses dekóderek) integrált áramköri lapkákon már régóta gyártanak, nem mi találtuk fel a spanyolviaszt. Azonban a megértéhez, a kész áramkörök javításához, ismeretlen, ritkán, vagy eddig még nem alkalmazott kódolások-dekódolások megvalósításához ismeretekre, tapasztalatokra tehetünk szert. Így képessé válhatunk eddig meg nem valósított feladatok megoldására is olcsón, hatásosan, elegánsan.

Természetesen más elemi kapukkal is meg lehet valósítani a fenti feladatot.

További egyszerűsítések

(nem csak NOT, AND és OR kapuk használata)

A Karnaugh tábla NOT, AND és OR kapukkal való megoldást eredményez. Mi eddigi feladataink során csak ezeket használtuk, de ha jól megértjük az egyszerűsítéseket, az áramkörök adta lehetőségeket, a megvalósítás során építkezhetünk a valóban létező elemekkel, nem feltétlenül muszáj csak NOT, OR és AND kapukkal dolgoznunk.

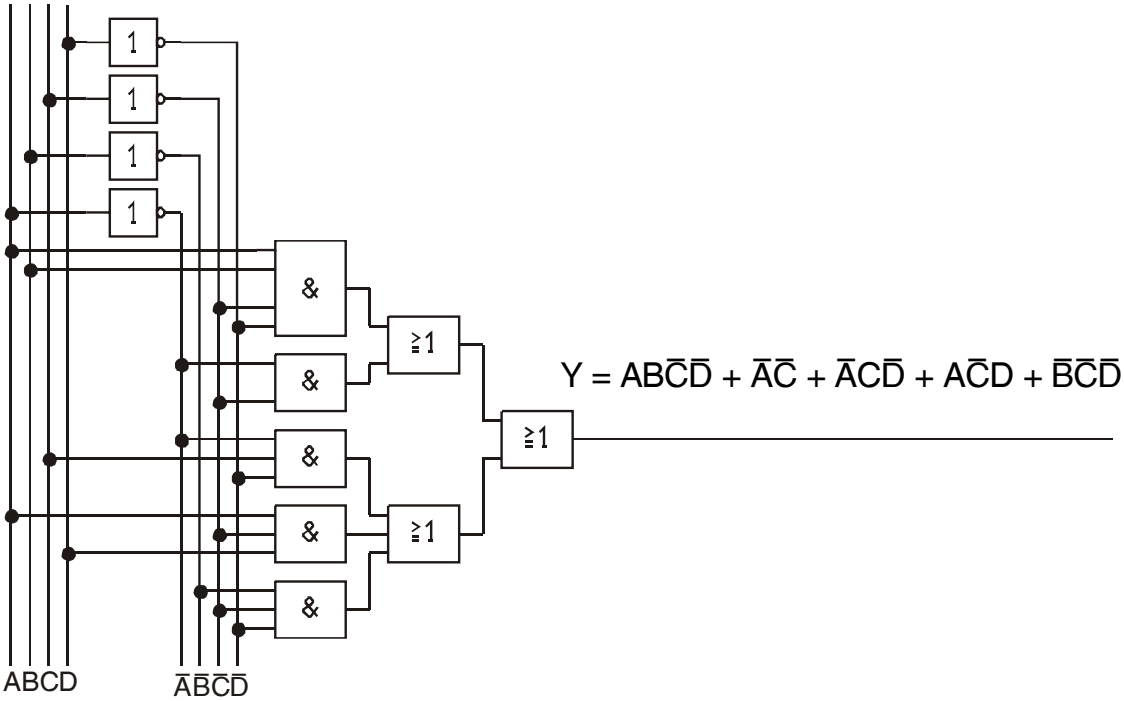
A felhasznált áramkörök típusai azonban nem csak ilyen kapuk lehetnek, hanem elsősorban attól függnek, mi is van raktáron. Ha választásunk van, akkor dolgozhatunk meg az olcsóbb, gyorsabb, kisebb helyigényű és fogyasztású kapukkal. Erre példa a következő feladatok megoldása, vagy a *Logikai függvények megvalósítása Funkcionálisan* teljes logikai rendszerek című fejezet *(IV.29. oldal)*, ahol megmutatjuk, hogy csak NAND, vagy csak NOR kapukkal is minden logikai feladatot meg tudunk valósítani. Ehhez hozzátesszük, a valóságban olcsóbb (kevesebb tranzisztort tartalmaz), gyorsabb, kisebb fogyasztású a NAND kapu az AND, vagy az OR kapunál, így ne csodálkozzunk, ha a gyakorlati életben sokkal több NAND kaput találunk az alkalmazások között, mint AND és OR kaput.

Sokszor a Karnaugh tábla összevonásai után is lehet tovább egyszerűsíteni, még olcsóbban megvalósítható alakra hozni a logikai egyenletet. Mutatunk erre is néhány példát:

Legyen a megvalósítandó áramkör $Y = AB\bar{C}\bar{D} + \bar{A}\bar{C} + \bar{A}C\bar{D} + A\bar{C}D + \bar{B}\bar{C}\bar{D}$

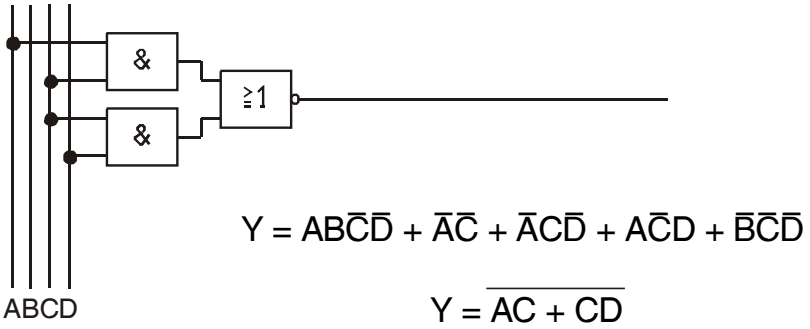
Ha valaki egyszerűsítés nélkül belevág a megvalósításba, célt érhet ugyan, de elég drága kapcsolást fog építeni, pl. a következőt:

A feladat kapuáramkörökkal való megvalósítása egyszerűsítés nélkül



Ha egyszerűsítjük a feladat logikai egyenletét, és így valósítjuk meg, a következőt kapjuk:

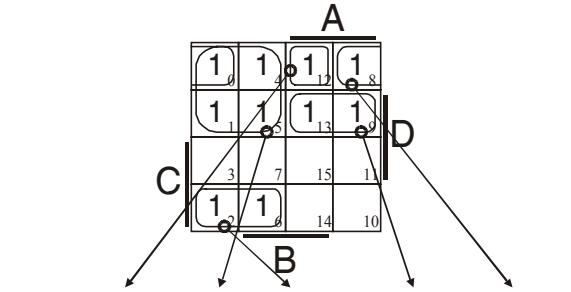
A feladat kapuáramkörökkal való megvalósítása egyszerűsítés után



Láthatjuk, érdemes egyszerűsíteni. Hogy is kaptuk ezt az egyszerű alakot? Vegyük észre, hogy nem csak Karnaugh táblával! Hiszen NOR kaput is alkalmaztunk, amit a Karnaugh táblából nem tudunk kiolvasni. Nézzük hát az egyszerűsítés módját:

Első egyszerűsítésre, áttekintésre alkalmas a Karnaugh tábla, melybe írva, és egyszerűsítve az $Y = AB\bar{C}\bar{D} + \bar{A}\bar{C} + \bar{A}C\bar{D} + A\bar{C}D + \bar{B}\bar{C}\bar{D}$ egyenletet kapjuk:

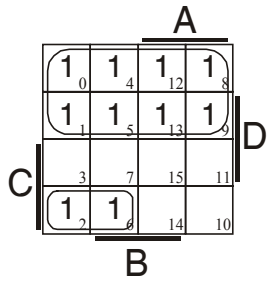
A feladat szövege szerinti tábla



$Y = AB\bar{C}\bar{D} + \bar{A}\bar{C} + \bar{A}C\bar{D} + A\bar{C}D + \bar{B}\bar{C}\bar{D}$

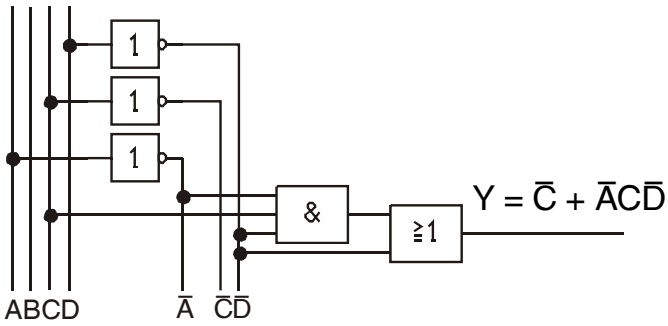
Ebben elvégezve az összevonásokat és megrajzolva a kapcsolást elemi kapuáramkörökkel:

Egyszerűsített tábla



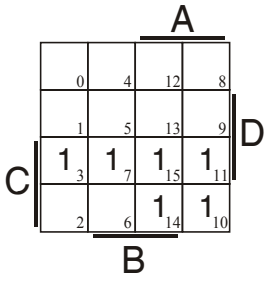
$Y = \bar{C} + \bar{A}C\bar{D}$

Egyszerűsített táblának megfelelő kapcsolás



Azonban ennél egyszerűbb kapcsolást is építhetünk!
Vegyük észre, hogy a nem 1-essel jelölt cellák is összevonhatóak lennének. Így, ha az eredeti függvény negáltját egyszerűsítjük, és végül megtagadjuk, egyszerűbb alakot kaphatunk. Most a Karnaugh táblába azokba a cellákba írjunk 1-eseket, melyekbe az eredeti egyenlet szerint nem azok voltak, majd végezzük el az összevonást, és végül az egészet tagadjuk meg! Így szintén a fenti feladat megoldásához jutunk, azonban még egyszerűbb, olcsóbb, előnyösebb áramköri kapcsolást kapva:

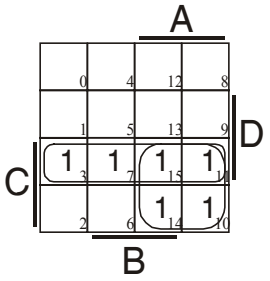
Az előző tábla invertálva



$\overline{Y = \bar{C} + \bar{A}C\bar{D}}$

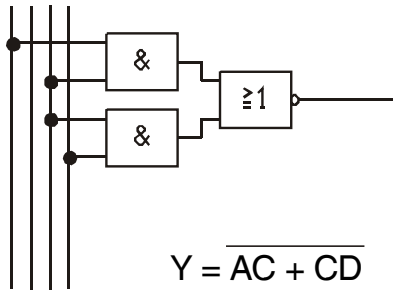
Ebben elvégezve az összevonásokat és megrajzolva a kapcsolást elemi kapuáramkörökkel a feladat még egyszerűbb, olcsóbb, logikusabb megoldását kapjuk:

Az előző tábla invertálva és egyszerűsítve



$\overline{Y} = \overline{\overline{C} + \overline{A}C\overline{D}} = AC + CD$

Ha az inverz egyszerűsített táblát megint invertáljuk (NOR az OR helyett), az eredeti táblának megfelelő működésű kapcsolást kapjuk



$Y = \overline{AC + CD}$

Figyeljük meg, jóval olcsóbb megvalósítást találtunk, mint a csak a Karnaugh tábla szerint valósítjuk meg logikai kapuáramkörökkel feladatunkat. Példánkban 5 db kapuáramkör helyett 3 db is elég.

> Megállapíthatjuk, sokszor a szabályos alaknál van egyszerűbb, tömörebb alak, mely olcsóbban megvalósítható. Érdemes ezt megkeresni!

Általános szabályt nem mondhatunk a legegyszerűbb kapcsolás megkereséséhez, de mindenképp érdemes elgondolkodni, nincs-e az eddig találtnál egyszerűbb, olcsóbb megoldás

Megjegyzés: Figyeljük meg, a fenti művelet fordítottját végeztük a *Szabályos alakra hozás* című fejezetben *(IV.14. oldal).* Akkor szabályos alakra (mintermes) hoztuk a logikai függvényt, most pedig az ott szerzett tapasztalatainkat hasznosítottuk a szabályosnál tömörebb, egyszerűbb alak megkeresésére.
Nem kell mindenáron a szabályos alakhoz ragaszkodni!

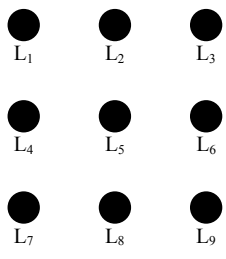
Nézzünk meg most egy eseti, nem megszokott, a gyakorlatban (iparban, számítástechnikában, irodatechnikában) nem előforduló feladatot:

Dominó pöttyeinek megjelenítése

Egy elektronikus dominójáték kijelzőjét szeretnénk elkészíteni lámpákkal. A feladat a lámpákat vezérlő logikai áramkör elkészítése. A lámpák feszültségével, áramfelvételével nem törődünk, csak a csak a logikai feladat megoldása a célunk.

Megoldás:

- Hogy beszélhessünk a lámpákról, beszámoztuk őket a számozás az ábrán látható:
- Definiáljuk az egyértelműség miatt, milyen állapotokat vehet fel a dominó. Összesen 10 félét, az ábrákat beírtuk a táblázatba.
- Mivel a dominó összesen 10 féle alakot jelezhet, így **négy bites logikai függvénnyel tudjuk leírni** ezt a 10 különböző állapotot.
- Rendeljük a dominó által jelzett számokat az első tíz négyváltozós mintermhez, melyek változóit A, B, C, és D-vel jelöljük.
- Készítsük úgy az áramköri kapcsolást, hogy a logikai magas szintre világítsanak a lámpák
- A fentieknek megfelelő állításokat táblázatban rögzítjük

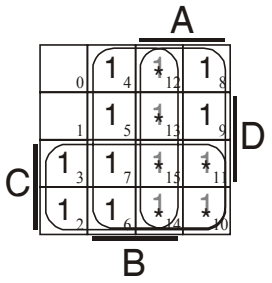


IV-3. táblázat

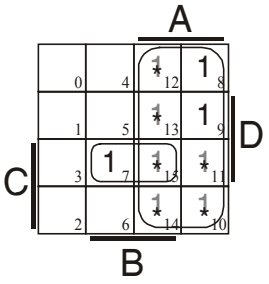
A dominó pöttyeinek megjelenítése

	Alak	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	A	B	C	D
0		0	0	0	0	0	0	0	0	0	0	0	0	0
1	•	0	0	0	0	1	0	0	0	0	0	0	0	1
2	• •	1	0	0	0	0	0	0	0	1	0	0	1	0
3	• • •	1	0	0	0	1	0	0	0	1	0	0	1	1
4	• • • •	1	0	1	0	0	0	1	0	1	0	1	0	0
5	• • • • •	1	0	1	0	1	0	1	0	1	0	1	0	1
6	• • • • • •	1	0	1	1	0	1	1	0	1	0	1	1	0
7	• • • • • • •	1	1	1	1	0	1	1	0	1	0	1	1	1
8	• • • • • • • •	1	1	1	1	0	1	1	1	1	1	0	0	0
9	• • • • • • • • •	1	1	1	1	1	1	1	1	1	1	0	0	1
10	X	*	*	*	*	*	*	*	*	X	X	X	X	X
11	X	*	*	*	*	*	*	*	*	X	X	X	X	X
12	X	*	*	*	*	*	*	*	*	X	X	X	X	X
13	X	*	*	*	*	*	*	*	*	X	X	X	X	X
13	X	*	*	*	*	*	*	*	*	X	X	X	X	X
14	X	*	*	*	*	*	*	*	*	X	X	X	X	X
15	X	*	*	*	*	*	*	*	*	X	X	X	X	X

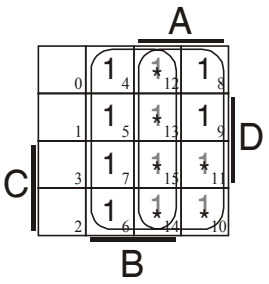
Itt is az utolsó hat bináris kombináció nem fordulhat elő, akárcsak a BCD kódban, így ezt a hat elő nem fordulható állapotot szabadon felhasználhatjuk egyszerűsítéseinkhez. Az L_1-L_9 lámpákról az A, B, C és D függvényében a következőket olvashatjuk ki a táblázatból, egyből elkészítve a Karnaugh táblákat is:



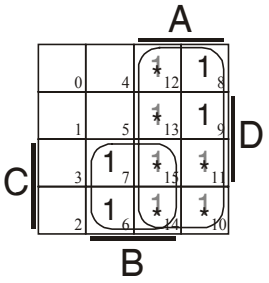
$$L_1 = L_9 = \sum_4 2,3,4,5,6,7,8,9,*,*,*,*,*,*, = A + B + C$$



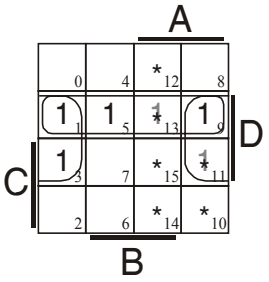
$$L_2 = \sum_4 7,8,9,*,*,*,*,*,*,* = A + BCD$$



$$L_3 = L_7 = \sum_4 4,5,6,7,8,9,*,*,*,*,*,*,* = A + B$$

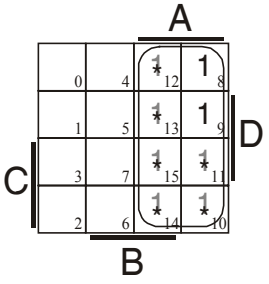


$$L_4 = L_6 = \sum_4 6,7,8,9,*,*,*,*,*,*,* = A + BC$$



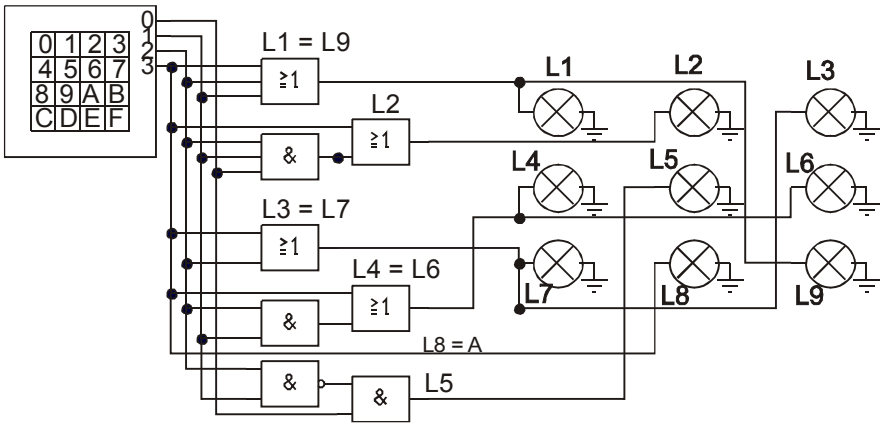
$$L_5 = \sum_4 1,3,5,9,*,*,*,*,*,*,* = \bar{B}D + \bar{C}D = D(\bar{B} + \bar{C}) = D(\overline{BC})$$

$L_5 = D(\overline{BC})$ alakja kedvezőbb, ez az alak ugyanis csak két kapuval megvalósítható, szemben az $L_5 = \bar{C}D + \bar{B}D$ alakkal, mely öt kaput (2 NEM, 2 ÉS és 1 VAGY) tartalmaz. NEM kapu máshova sem kell, így a $D(\overline{BC})$ alakot érdemes megvalósítani.



$$L_8 = \sum_4 8,9,*,*,*,*,*,*,* = A$$

A kapott logikai egyenleteknek megfelelő áramköri kapcsolás elemi alapáramkörökkel:



Logikai függvények megvalósítása

Funkcionálisan teljes logikai rendszerek

- Funkcionálisan teljes értékűnek mondjuk azokat a logikai kapukat, melyekből tetszőleges funkciójú logikai hálózat (kapcsolás) kiépíthető

Három ilyen teljes értékű logikai rendszert ismertetünk:

NÉV rendszer (NEM, ÉS, VAGY rendszer)

Három kapu alkotja ezt a teljes értékű rendszert, a NOT, az AND és az OR kapu. Ez a rendszer az, amit eddig is sokat alkalmaztunk, a szabályos alakokkal és az igazságtáblázattal megadható logikai függvényeket e háromféle kapuval mindig megvalósíthatjuk. Tehát **a NOT, az AND és az OR kapuk alkalmazásával tetszőleges funkciójú logikai hálózat kiépíthető.** Ez az állítás evidens, nem bizonyítjuk.

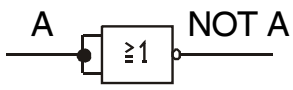
NOR rendszer

Ez olyan rendszer, melyben csak NOR kapuk vannak. **Csak NOR kapuk alkalmazásával tehát tetszőleges funkciójú logikai hálózat kiépíthető.**

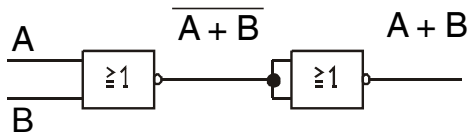
Bizonyítás:

Mivel eddig meggyőződhettünk arról, hogy a NÉV rendszer három kapuja teljes értékű logikai rendszer, **elég azt bebizonyítanunk, hogy e három kaput csak NOR kapu alkalmazásával előállíthatjuk**, hiszen azokkal minden logikai függvény megvalósítható.

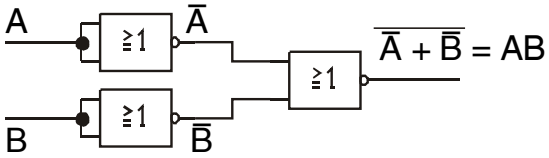
NOT kapu NOR kapuval készítve



OR kapu NOR kapuval készítve



AND kapu NOR kapuval készítve



Itt felhasználtuk az $\overline{AB} = \bar{A} + \bar{B}$ alakú De Morgan tételt, ennek mindkét oldalát negálva éppen az $\overline{\bar{A} + \bar{B}} = AB$ függvényt kapjuk.

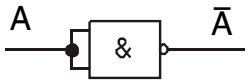
NAND rendszer

Ez olyan rendszer, melyben csak NAND kapuk vannak. **Csak NAND kapuk alkalmazásával tetszőleges funkciójú logikai hálózat kiépíthető.**

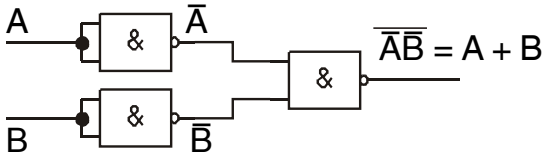
Bizonyítás:

Mivel a NÉV rendszer három kapuja teljes értékű logikai rendszer, **elég azt bebizonyítanunk, hogy e három kaput csak NAND kapu alkalmazásával előállíthatjuk**, azokkal már minden logikai függvény megvalósítható.

NOT kapu NAND kapuval készítve

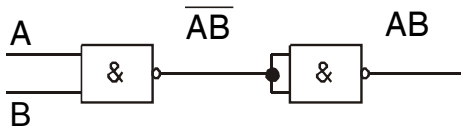


OR kapu NAND kapuval készítve



Itt felhasználtuk az $\overline{A+B} = \bar{A}\bar{B}$ alakú De Morgan tételt, ennek mindkét oldalát negálva éppen az $\overline{\bar{A}\bar{B}} = A + B$ függvényt kapjuk.

AND kapu NAND kapuval készítve



Láthatjuk, a NAND rendszerrel is, és a NOR rendszerrel is megvalósíthatjuk a NÉV rendszer három kapuját, így tetszés szerinti logikai függvényt csak NOR, ill. csak NAND kapuk alkalmazásával megvalósíthatunk. A digitális elektronika áramkörei című fejezetben láthatjuk, NOR, de különösen a NAND kapuk olcsóbbak, mint a NÉV rendszer kapui, ezért a funkcionálisan teljes értékű logikai rendszereknek különösen nagy jelentősége van.

Kérdések a Logikai egyenletek című fejezethez:

- Mi az igazságtáblázat? Mi az a minterm? *Mi a maxterm?* Hányféle állapotot vehet fel egy n változójú logikai függvény? Mit értünk szabályos alakban megadott logikai függvényen? Milyen szabályos alakú logikai függvényeket ismer? Egy n változós logikai függvénynek hány mintermje lehet? *És hány maxtermje?* Mit értünk mintermes alakú logikai függvényen?
- Mit értünk a logikai függvény egyszerűsítésén? Milyen előnyei lehetnek az egyszerűsítésnek? Mondjon két-három példát arra, milyen esetet ismer, ahol meg nem engedett állapotok fordulnak elő! Hogy lehet kihasználni ezeket a meg nem engedett állapotokat egyszerűsítésre? Mik azok a pszeudotetrádok?
- Mit tud a Karnaugh tábláról? Hány változós Karnaugh táblákat ismer? Miért nem használunk négynél több változós logikai függvényekre Karnaugh táblát? Mire jó a Karnaugh tábla?

Feladatok a Logikai egyenletek című fejezethez:

IV. 1. Adja meg a háromváltozós logikai ÉS függvény, és a logikai VAGY függvény igazságtáblázatát! Adja meg a kétváltozós antivalencia függvény igazságtáblázatát! Adja meg az ekvivalencia függvényt (kétváltozós) mintermes alakban!

IV. 2. Igazolja a De Morgan azonosságokat három változóra Karnaugh táblával!

IV. 3. Valósítsa meg (készítsen rajzot) az ekvivalencia függvényt logikai kapuáramkörökkel!

IV. 4. Állítsa ki az alábbi a logikai függvények igazságtábláját

IV. 5. $K = (\overline{A} + B)(A + C)$ $L = \overline{AC} + \overline{A}\overline{C}$

IV. 6. Hozza szabályos alakra az alábbi logikai függvényeket:

$K = (\overline{A} + B)(A + C)$ $L = \overline{AC} + \overline{A}\overline{C}$

$M = A\overline{B}\overline{C} + \overline{A + C}$ $N = A\overline{B}C + \overline{B + C}$

$P = \overline{A\overline{B}\overline{C} + \overline{A}BC}$ $Q = \overline{AB\overline{C} + A\overline{B}C} + \overline{A}\overline{B}C$

IV. 7. Egyszerűsítse az alábbi logikai függvényeket!

$R = \overline{A}BC + \overline{B}\overline{C}D + A\overline{B}CD$ $N = A\overline{B}C + \overline{B + C}$

$S = \overline{A\overline{B}\overline{C} + \overline{B + \overline{A}BC}}$ $Q = \overline{AB\overline{C} + A\overline{B}C} + \overline{A}\overline{B}C$

$F = \sum^{2} 0,2,3$ $G = \sum^{3} 2,3,5,7$ $H = \sum^{3} 0,2,3,4,5,7$

$J = \sum^{4} 0,2,3,4,5,7$ $X = \sum^{4} 0,1,2,3$ $E = \sum^{4} 0,2,4,6$

$U = \sum^{4} 0,2,4,6,8,10,12,14$ $V = \sum^{4} 1,3,5,7,9,11,13,15$ $W = \sum^{4} 5,6,7,13,14,15$

IV. 8. Rajzolja le az F, H, és a W logikai függvényt megvalósító áramköröket logikai kapukkal!

IV. 9. Valósítsa meg a K, L, F és U logikai függvényeket kapcsolókkal!

IV. 10. Oldja meg a *Dominó pöttyeinek megjelenítése című* alfejezet *(IV.26 old.)* példáját, de úgy, hogy most a lámpák a logikai L (alacsony) szintre világítsanak!

IV. 11. Rajzolja le az F, H, K, X, U és a W logikai függvényt megvalósító áramköröket NOR rendszerrel! (Csak NOR kapukkal)

IV. 12. Rajzolja le az M, P, J, E és a V logikai függvényt megvalósító áramköröket NAND rendszerrel (csak NAND kapukkal)!

IV. 13. Készítsen egy fél összeadó kapcsolást. A fél összeadót a *Bináris összeadás pozitív operandusoknál* című fejezetben már vettük *(II.8. oldal)*

A és B összege fél összeadóval

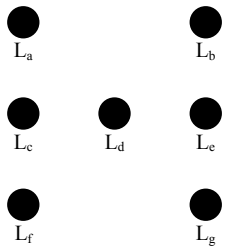
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

IV. 14. Készítsen egy dobókocka lámpáit működtető kapcsolást, hasonlóan a *Dominó pöttyeinek megjelenítése* című alfejezet *(IV.26 old.)* példájához. Ehhez tegye meg a következőket:

- Készítsen ábrát a dobókocka lámpáinak elhelyezkedéséről!
- Jelölje (pl. számozza) meg a lámpákat, hogy beszélhessünk a lámpákról!
- Vegyen fel a dominó által jelzett számokhoz logikai állapotokat, és definiálja az egyértelműség miatt, milyen állapotokat vehet fel a dobókocka. Készítsen táblázatot, melyben a különböző állapotnak megfelelően világító lámpákat, és az állapotoknak megfelelő értékű logikai változókat feltünteti!
- Állapítsa meg e táblázatból a különböző lámpákat vezérlő logikai függvényeket úgy, hogy a logikai H (magas) szintre világítsanak a lámpák!
- Egyszerűsítse ezeket a logikai függvényeket!
- Készítse el Tina áramkörszerkesztő programmal az áramköri kapcsolást!

Javasolt táblázat és rajz a dobókocka pöttyeinek megjelenítéséhez

	alak	L_a	L_b	L_c	L_d	L_e	L_f	L_g	A	B	C
0									0	0	0
1									0	0	1
2									0	1	0
3									0	1	1
4									1	0	0
5									1	0	1



V. Kombinációs logikai hálózatok

Bevezetés

- Hálózatnak olyan berendezést, kapcsolást, áramkört nevezünk, melynek bemenetei és kimenetei vannak, és olyan feladatot old meg, ahol a kimenetek a bemenetek és egyéb mennyiségek, események (pl. idő, előzmények, stb.) függvényében vesznek fel különböző állapotokat.

Mi kizárólag elektromos áramköri elemekből felépített hálózatokat tanulnunk, de a gyakorlatban természetesen más hálózatok is vannak (pneumatikus, stb.) Ezért **szinonimaként használjuk az áramkör, hálózat, kapcsolás fogalmakat e fejezetben**, noha ez máshol nem egészen pontos.

- Logikai hálózatnak (áramkörnek) az olyan hálózatot (áramkört) nevezzük, melynek bemenetei is és kimenetei is logikai állapotokkal jellemezhetők.

Nem csak logikai, hanem analóg és vegyes hálózatok is vannak. Az analóg és vegyesen analóg és digitális áramköröket tartalmazó hálózatokat később tanuljuk.

- **Kombinációs logikai hálózat:** olyan logikai hálózat, mely kimenetei csak a bemenetek állapotaitól, kombinációitól függnek, semmi mástól.
- **Sorrendi,** nemzetközi szóval **szekvenciális**: az olyan logikai hálózat melynek kimenetei nem csak a bemenetek kombinációitól, hanem az előzményektől, a különböző kombinációk sorrendjétől is függenek.

Eddig mi tulajdonképpen kombinációs logikai áramköröket tanulmányoztunk a IV fejezetben, és ilyen jellegűek voltak eddigi példáink is. A szekvenciális hálózatok bonyolultabbak, ilyen áramköröket nem tanultunk, a következő fejezetben tanulmányozzuk őket.

Kombinációs hálózatok

Kódoló és dekódoló áramkörök

Emlékeztetőül leírjuk, mikor beszélünk kódolásról, és mikor dekódolásról:

- Kódolásnak nevezik az átalakítást, ha így kevesebb eszközt (pl. vezetéket, időt, tárolót, sávszélességet, stb.) igénylő ábrázolási módra jutnak.
- Az átalakítást akkor nevezik dekódolásnak, ha az átalakítás során kevesebb eszközigényű ábrázolásról nagyobb igényűre jutnak.

Pl. a kevesebb (akár egy) vezetéken érkező kódolt információt úgy alakítják át, hogy több vezetéken halad tovább, akkor ez dekódolás.

Tekintsük át, milyen kódoló és dekódoló áramkörökkel foglalkoztunk eddig:
A dominó-, és a dobókocka kijelzőinek működtetése, hétszegmenses kijelző működtetése, stb.
A hétszegmenses kijelző az elektronikában igen elterjedt, ezért BCD-ből hétszegmensre dekóder áramkört integráltan is gyártanak (Integrált áramkör, rövidítve IC), sőt, bonyolultabb feladatokat megoldó IC-k része.

Gyakori feladat a binárisan kódolt információ dekódolása. E fejezet végén, a *Dekóder-demultiplexer* alfejezetben meg fogjuk ismerni a binárisból dekódoló áramköröket, melyeket azért ott tanulmányozzuk, mert ezek egyben demultiplexerek is *(lásd ott, a V.12 oldalon*).

Sok egyéb, számunkra kevésbé fontos, legalábbis nem tanulmányozott kódolást, dekódolást lehetne említeni, felsorolunk néhányat, korántsem a teljesség igényével:

Prioritásból BCD-be, vagy bináris kódba enkóder

Különböző egyéb kódokból átkódoló áramkörök (pl. binárisból oktálisba-, vagy BCD-be, stb.)

Hibajelző kódok előállítása, paritásképzés-, és ellenőrzés

BCD-ről 10-re dekódoló áramkör (pl. Nixie csövekhez)

Stb.

Sokféle különböző IC-t gyártanak a különböző kódolási és dekódolási feladatokra. Ilyen áramköröket nem érdemes kapukból építeni, olcsóbb készen megvenni őket. Mi a jobb megértésért tanulmányozzuk őket, ahogy a többi kombinációs hálózatot is.

- Feladat: Nézze meg a Tina áramkörszerkesztő program által ismert kódoló (encoder) és dekódoló áramköröket! Jegyezzen fel legalább 5 különböző funkciójú kódoló (encoder), és 5 dekóder áramkörtípus nevét!

Digitális komparátorok

Komparálás alatt összehasonlítást értünk. A digitális komparátor digitálisan ábrázolt számokat hasonlít össze. Két szám között mindig reláció állítható fel, egyik szám vagy nagyobb a másiknál, vagy egyenlő vele, vagy kisebb nála. A három közül egy, és csakis egy igaz (ezt a csakis egyet használni fogjuk).

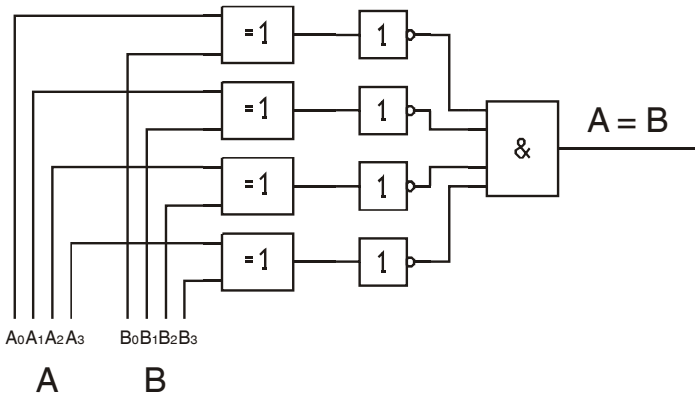
Mi csak pozitív egész, és azonos számú bittel ábrázolt azonos elv szerint binárisan kódolt számok összehasonlítását tanulmányozzuk komparátor áramkörrel.

Egyenlőség összehasonlítása

Két pozitív egész, azonos számú bittel ábrázolt azonos elv szerint binárisan kódolt szám egyenlő, ha minden megfelelő bitjük egyenlő. Az egyenlőséget bitenként végezzük ekvivalencia (Tinában negált XOR) kapukkal, majd ÉS kapuval vizsgáljuk, minden ekvivalencia kapu igaz-e. Logikai egyenlettel az Y egyenlőség akkor teljesül (pl. 4 bites számokra), ha

$Y = (A_3 \text{ EQV } B_3)(A_2 \text{ EQV } B_2)(A_1 \text{ EQV } B_1)(A_0 \text{ EQV } B_0)$

Egyenlőség komparálása Tina áramkörszerkesztő programmal



Az egyenlőség vizsgálata egyszerű, ezért sok bonyolultabb feladatnál alkalmazzák feltétel vizsgálatára (mint a programozásban a ciklus szervezés) stb. Pl. addig ismételnek, addig alakítanak egy folyamatot, amíg az egyenlőség nem teljesül.

Egyenlőtlenség összehasonlítása

Ez már az egyenlőség vizsgálatánál összetettebb feladat. Vizsgáljuk meg, mikor nagyobb egy (A) szám a másik (B) számnál, ha mindkét szám pozitív egész, azonos számú bittel binárisan kódolt ábrázolású.

1. feltétel: A > B, ha az A szám legnagyobb helyiértékű bitje 1, míg a B számé 0.

   Ha n bites számokat vizsgálunk $Y = A_{n-1}\overline{B}_{n-1}$

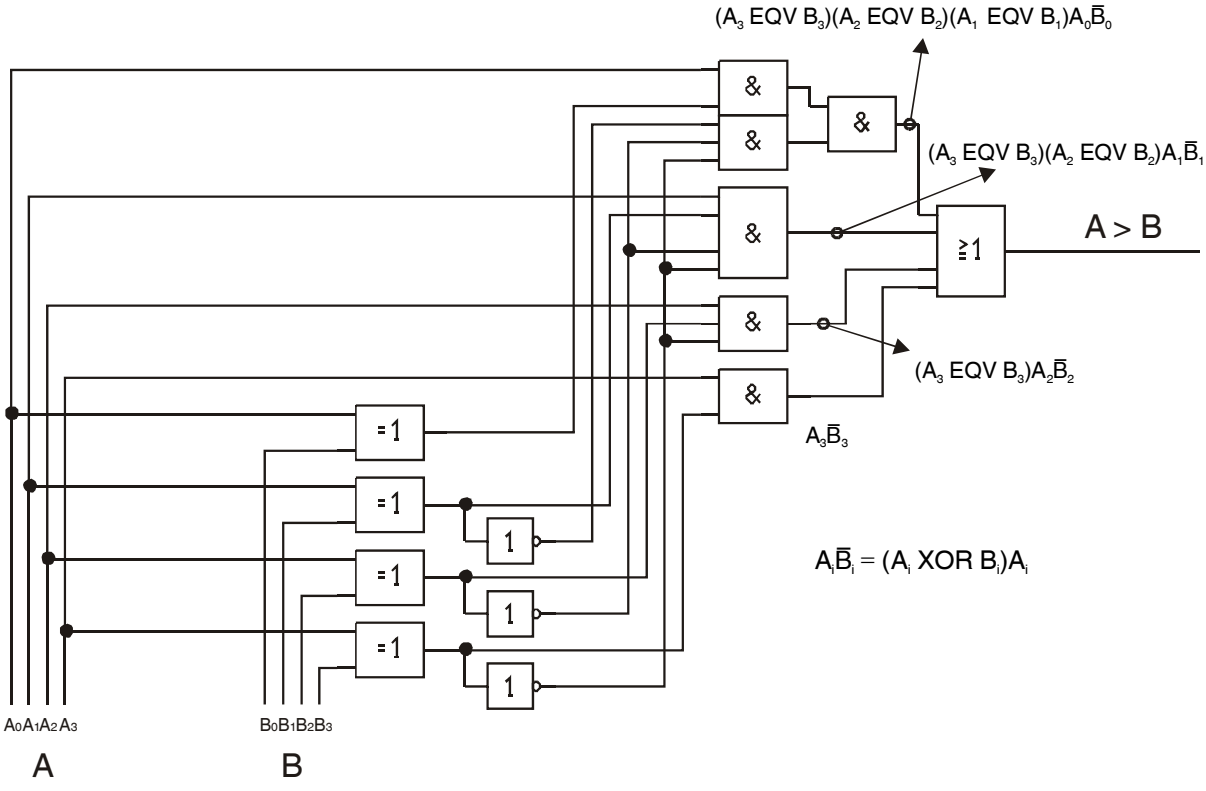
   (n jegyű számnál a legnagyobb helyiérték az n-1-edik helyiérték)

2. feltétel: akkor lép érvényre, ha találunk olyan i-edik bitet, melynél Ai = 1, és Bi = 0, míg az i előtt levő összes helyiértéken az A és a B szám bitenként megegyezik. Ekkor az i-nél kisebb helyiértéken levő egyenlőségek, vagy egyenlőtlenségek nem játszanak szerepet.

Az $A > B$ egyenlőtlenség logikai függvénye 4 bites számokra az 1. és 2. feltétel alapján:

$$X= A_3\overline{B}_3 + (A_3 \text{ EQV } B_3)A_2\overline{B}_2 + (A_3 \text{ EQV } B_3)(A_2 \text{ EQV } B_2)A_1\overline{B}_1 +$$
$$(A_3 \text{ EQV } B_3)(A_2 \text{ EQV } B_2)(A_1 \text{ EQV } B_1)A_0\overline{B}_0$$

Egyenlőtlenség komparálása Tina áramkörszerkesztő programmal

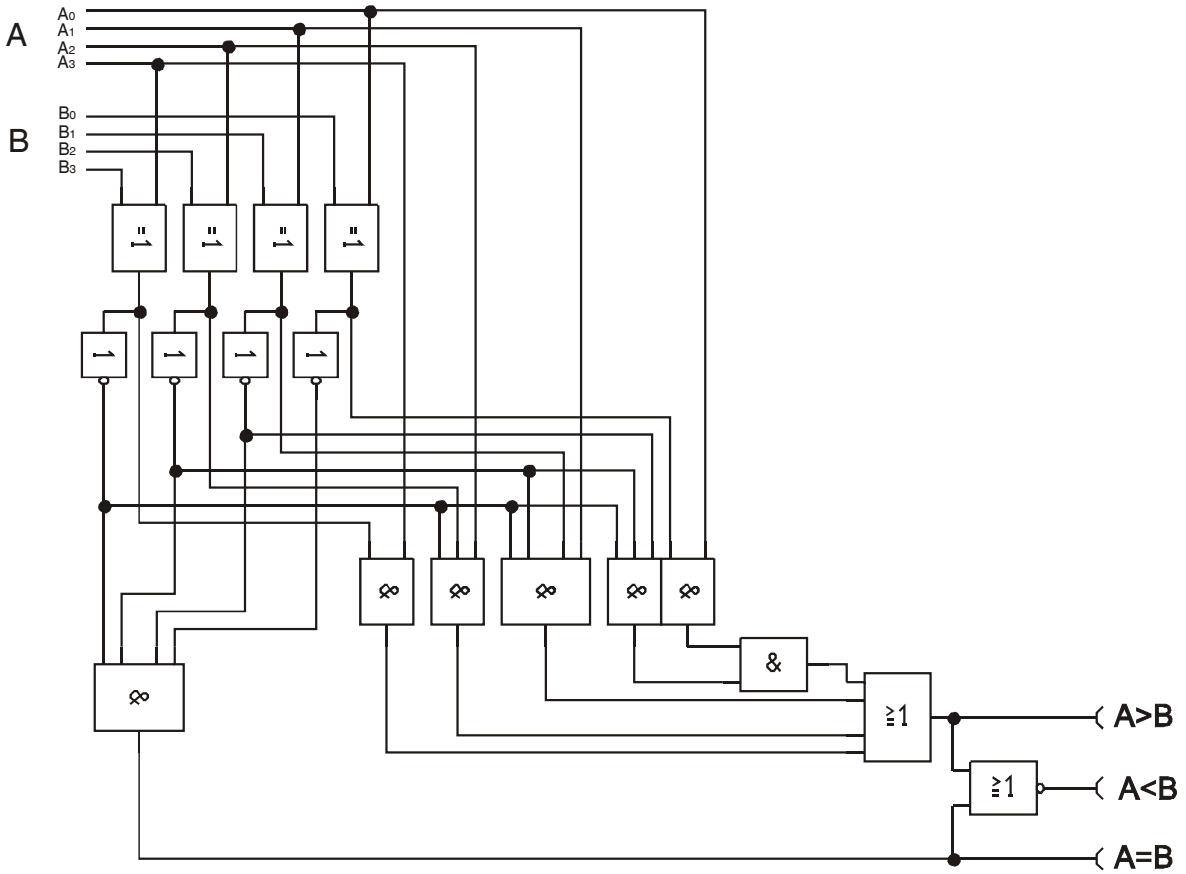


Az ábrán egy egyszerűsítést alkalmaztunk: Vegyük észre, megtehetjük, hogy az i-edik bitre $A_i\overline{B}_i$ helyett $(A_i \text{ XOR } B_i)A_i$ -t vizsgálunk, hiszen a kizáró vagy függvény definíciója szerint ha $A_i \text{ XOR } B_i$ igaz, és A igaz, akkor B csak 0 lehet. Azaz $A_i\overline{B}_i = (A_i \text{ XOR } B_i)A_i$. Így egy NOT kaput megspórolunk (B tagadásához). Az $(A_i \text{ XOR } B_i)$ érték pedig amúgy is rendelkezésre áll, mivel a Tina áramkörszerkesztő program nem ismeri az EQV kaput, így a 2. feltétel szerinti bitenkénti egyenlőség vizsgálatát negált XOR kapukkal amúgy is ki kell építenünk.

Teljes komparátor

Ez az egyenlőtlenség és az egyenlőség összehasonlításának egyszerre való vizsgálata. A vizsgálatnak három eredménye lehet, A > B, A = B, vagy A<B. E háromból egy, csakis egy igaz, így elég megvizsgálni, kettő esetet, ha ezek egyike sem igaz, akkor a harmadik igaz. Ha tehát megvizsgáljuk, A > B, és A = B, és ezek egyike sem igaz, akkor A < B.

A teljes komparátor megvalósítása Tina áramkörszerkesztő programmal:



A teljes komparátor és az egyenlőtlenségi komparátor alkalmazása diszkrét áramkörként ritkább az egyenlőség komparálásánál. A teljes komparátor viszont minden aritmetikai-logikai egységben szerepel. (ALU=Arithmetic Logical Unit)

ALU minden processzorban van, de önálló, diszkrét áramkörként is gyártják. Az aritmetikai egységek nem csak összehasonlítást, hanem egyéb matematikai műveletet is képesek végezni, köztük a legfontosabbat, az összeadást is. Tekintsük meg, miképp lehet digitális számokat összeadni logikai áramkörökkel, erről szól a következő alfejezet.

Bináris összadók

A fél összeadó (Half Adder)

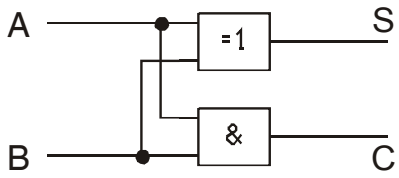
A fél összeadó olyan logikai hálózat, melynek két bemenete és két kimenete van. A bemeneteket jelöljük A-val és B-vel. A fél összeadó ezeket, mint 1 bites számokat adja össze. Összegük S (szumma), és az átvitel C (carry).

Az alábbi táblázatban nézzük a bináris A és B összegét.

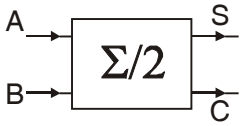
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Látható, hogy S = A XOR B, és C = AB. Tehát két kapuval megvalósítható a félösszeadó.

Félösszeadó Tina áramkörszerkesztő programmal:



A félösszeadó blokkrajza



A teljes összeadó (Full Adder)

Tekintsünk két bináris n jegyű pozitív egész számot. Ha ezeket kell összeadni, ezt **bitenként** a teljes összeadóval végezhetjük el. A két összeadandó szám (operandus) i-edik bitjén levő teljes összeadó nem csak a két operandus i-edik bitjét adja össze, hanem az i-1 helyiértékről esetlegesen keletkezett átvitelt is figyelembe veszi.

A teljes összeadó olyan logikai hálózat, melynek három bemenete és két kimenete van. A bemeneteknél a két operandus i-edik bitjét jelöljük A_i-vel és B_i-vel, az előző (i-1) helyiértéken keletkezett átvitelt C_{i-1}-gyel. A teljes összeadó ezeket, mint 1 bites számokat adja össze. Az összegük S_i (szumma), és az átvitel C_i (carry) legyen.

A teljes összeadó igazságtáblázata:

A_i	B_i	C_{i-1}	S_i	C_i
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

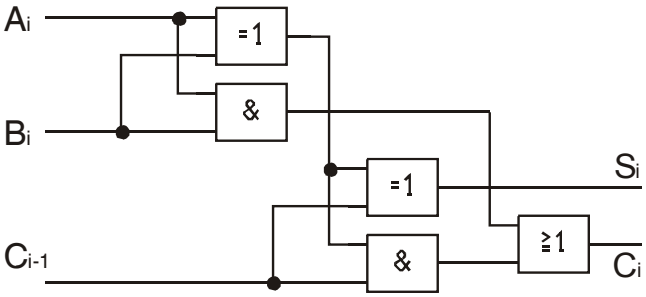
Könnyű megérteni – megjegyezni, ha meggondoljuk:

Si = 1, ha páratlan számú (1 vagy 3) 1-est kell összeadni

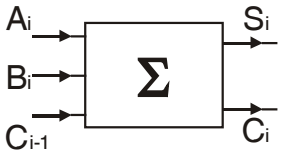
Ci = 1, ha legalább két 1-est (2 vagy 3) kell összeadni.

A teljes összeadó nagyon fontos áramkör. A teljes összeadót az igazságtáblázat alapján könnyen el lehet készíteni. De még könnyebben, ha két félösszeadót használunk építéséhez.

Teljes összeadó két félösszeadóból Tina áramkörszerkesztő programmal:



A teljes összeadó blokkvázlata

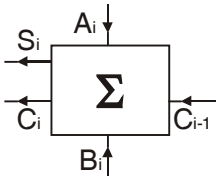


Több bites összeadók

A több bites számokat teljes összeadókból, mint elemekből építhetjük meg. Minden egyes helyiértéken bitenként adjuk össze az operandusokat, és az előző helyiértéken képződő átviteleket.

A több bites teljes összeadó is képes arra, hogy az előző helyen keletkezett átvitelt fogadja, hozzáadja az operandusokhoz. A több bites fél összeadónak nem sok értelme van, ilyen áramköröket nem gyártanak. A gyártók a több bites összeadókat mégis teljes összeadónak nevezik (nemzetközi szóval Full Adder), átvéve az egybites teljes összeadótól az elnevezést.

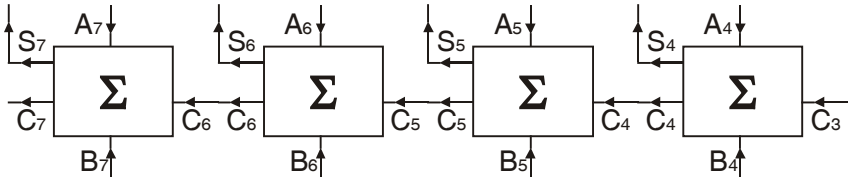
Tekintsük meg az egybites teljes összeadó blokkrajzát kicsit átrajzolva:



Ezzel az egybites teljes összeadóval, mint építőelemmel építhetünk több bites teljes összeadót

Nézzük meg, hogy lehet több bites teljes összeadót építeni egybites teljes összeadókból. Pl. készítsünk olyan összeadót, mely a 4. bittől a 7-edikig végzi el a teljes összeadást.

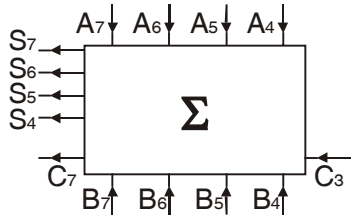
Több bites összeadó kapcsolási vázlata (blokkvázlata) egybites teljes összeadókból:



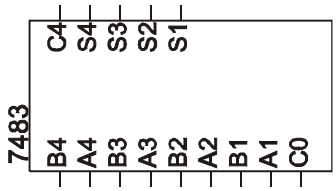
Természetesen ezt a rajzot is blokként foghatjuk fel, immár 4 bites teljes összeadó áramköri blokként, és így építhetünk ilyen blokkokkal akár k-szor 4 bites teljes összeadót.

Tekintsük meg, hogy néz ki a 4 bites teljes összeadó blokkrajza az előző példa egy blokkba rajzolásával, ill. nézzük meg, milyen rajzjele van egy valóban gyártott 4 bites összeadónak:

Az előző ábra blokkba foglalva



Az SN7484 típusú full adder logikai IC kapcsolási rajza (Tina áramkörszerkesztő program)



Több bites teljes összeadókat sokan gyártottak, azonban a korszerű elektronikában egyre inkább csak a nagybonyolultságú integrált áramkörök (chipek) része, önálló, diszkrét áramkörként inkább csak a meglevő elavult áramkörök javításához kellenek. Az összeadók működésének megértéséhez, és a blokkok kialakításának szemléltetéséhez azonban hasznos a teljes összeadókról ennyi ismeret.

Az összeadók hátrányai, problémák

Láthatjuk, igen egyszerű egybites teljes összeadókból sokbites teljes összeadót készíteni. **A gyakorlatban azonban nem így készítenek összeadókat, mert jelentős hátránya van az eddig tárgyalt építési módnak, lassú.** A logikai áramkörök nem végtelen gyorsak, feladatuk elvégzéséhez egy kis idő kell. Természetesen nagyon kicsi idő a késés. 10 ns körüli az egyszerű áramköröknél, a bonyolultabbaknál is csak 100ns körüli volt már az 1970-es években is. *(Itt ns nanosecundumot jelent, a másodpec egymilliomod részét. A ns tipikus számítástechnikai alapidő, ns-ban adják meg a legtöbb gyors folyamat idejét.)* Az összeadók a számítást végző áramkörökben, leginkább a processzorokban (a bennük levő aritmetikai-logikai egységben) szerepelnek. A processzorok lehető leggyorsabb működése nagyon fontos, sokkal fontosabb, mint amit meg lehet takarítani az egyszerűbb és olcsóbb, de sokkal lassabb összeadóegységek kisebb költségén.

Nézzük meg, miért lassúak az eddig tárgyalt összeadók:

Legyen mindegyik egybites összeadó késése 10ns. Ha megfigyeljük a működésüket, láthatjuk, hogy az operandusok jelentkezése után az első biten 10ns múlva lesz helyes eredmény, és helyes átvitel is. A következő biten azonban éppen az átvitel miatt csak 10 ns-mal később, azaz összesen 20 ns múlva lesz helyes eredmény és helyes átvitel. Láthatjuk az átvitel minden egyes egybites összeadón 10 ns-mal késik, így **n bites öszeadón n-szer 10 ns teljes késés lesz. Közben a helyes összeg helyett mindenféle közben kialakult hamis eredmény jelenik meg**, amihez ajánlott letiltani a kimenetet a késés miatt előállt bizonytalansági időre, nehogy rossz eredményt olvasson ki az összeadó kimenetére kötött áramkör.

Hogyan lehet gyorsítani az összeadók így keletkező késését? Gyors összedás, átvitel képzés

Az összeadóknál ki lehet logikázni (pl. mi is elvégezhetjük igazságtáblázat alapján), mikor keletkezik átvitel, bonyolultabb áramkör révén olyan áramkört lehet készíteni, melyben nem bitről bitre terjed az átvitel (szintén pl. négybites igazságtáblából). Az ilyen áramkörök már sokkal bonyolultabbak, semhogy könyvünkben kirajzoljuk őket (egy 8 bites összeadónak csak az igazságtáblája 256 soros lenne, egy mai korszerű processzor 64 bites, 64 bites igazságtáblát lehetetlen kitölteni, de elképzelni is, 2^{64} sora lenne). Ilyen áramköröket készítenünk sem érdemes, hiszen gyártják őket, ill. chipek részeiként szerepelnek.
A négy-, ill. nyolcbites teljes összeadók átvitelének gyorsítására készítették az ún gyors átvitelképző (look ahead carry generator) integrált áramköröket. A Tina áramkörszerkesztő program is ismer ilyeneket, pl. a négybites SN74182-es logikai IC, mellyel pl. a szintén négybites SN74181 típusjelű aritmetikai-logikai egység átvitele gyorsítható fel.
Mi ezzel többet nem foglalkozunk, a különálló összeadóegységek is ritkák, elavultak. Mi csak a digitális technika alapjait tanuljuk, csak a korszerű processzorok igen nagy áramköri elemszáma miatt, a problémák jobb megértéséhez említettük az összeadók kését, és a gyorsabb megoldásokat. A mai processzorokba, chipekbe már a mi ismereteinknél sokkal több, bonyolultabb és gyorsabb aritmetikai áramköröket építenek.

Multiplexelés, demultiplexelés

(Adatgyűjtés, adatelosztás)

A multiplexer

Az elektronikában gyakran igen nagymennyiségű jelet kell továbbítani, ki, vagy bevezetni berendezésekbe. Ezt egyszerre nem lehet az igen nagy mennyiség miatt megtenni, ezért a legtöbb esetben azt a megoldást választják, hogy a jeleket időben egymás után továbbítják. Ehhez a jeleket össze kell gyűjteni, és a megfelelő sorrendben, vagy időben továbbítani őket egyesével, egymás után. Ekkor kevés számú adathordozó (vezeték, rádióhullám, stb.) igénybevételével is igen nagy számú jelet lehet továbbítani.

- Multiplexelésnek, adatszelekciónak, adatválasztásnak az adatok összegyűjtését, kiválogatását nevezzük.
- Multiplexernek, adatszelektornak a kiválogatást végző berendezést nevezzük.

Eddigi értelmezésünk szerint a multiplexelés kódolás, nagyobb mennyiségű hordozón (vezetéken, stb.) érkező információt kevesebb eszközigényű (kevesebb számú vezetéken) berendezésen, hordozón továbbítjuk.

- Adatelosztásnak, demultiplexelésnek a multiplexelés fordított műveletét nevezzük, az egymás után érkező adatokat elosztjuk, mindegyiket az ő rendeltetési helyére, vagy útvonalára.
- Adatelosztónak, demultiplexernek az adatelosztást végző eszközt nevezzük.

Az adatelosztás eddigi értelmezésünk szerint dekódolás, a kisebb eszközigényű ábrázolásról nagyobb eszközigényűre térünk át.

Az adatok nem csak digitális, hanem analóg, általános jelek is lehetnek. Az utóbbit analóg multiplexelésnek nevezik. Általában mintát vesznek a különböző jelekből, és ezeket a mintákat küldik tovább a vezetéken, adathordozón. Sokszor e mintákat digitalizálják, ekkor már digitális jeleket továbbítanak tovább, az analóg multiplexelés zavarérzékeny, pontatlan, korszerűtlen (elavult).
Mi könyvünkben csak digitális jelekre vizsgáljuk a multiplexelést.

A digitális adatok lehetnek bitek, vagy szavak, a legtöbbször bájtok. Ha bájtszervezésű adatok vannak, egyszerre egy bájt, ha bitszervezésű adatok vannak, akkor egyszerre csak egy bitnyi adat kerül továbbításra. A bájtszervezés elve ugyanaz, mint a címszervezésűé, kivéve, hogy nem egy, hanem egyszerre nyolc adat van rendelve ugyanahhoz a címhez.
Mi e fejezetben csak a bitszervezésű multiplexelést tanulmányozzuk, később mutatjuk meg, hogyan lehet bájtszervezésre kiterjeszteni.

Többféle protokoll (szabály) szerint történhet az adatok kiválogatása
Történhet idő szerint, bizonyos időben előbb az egyik, majd a másik, majd így tovább, egy bizonyos megállapodás, vagy sorrend szerint egymás után küldjük az adatokat. Ez az időmultiplexelés.
Történhet úgy is, hogy megjelöljük az adatokat, és minden adatot saját jelével együtt küldünk ugyanazon az úton (adathordozón, vezetéken). Sokféle ilyen jel képzelhető el, ahogy sokféle szabály (protokoll) is ismert.
Mi egyet, a legelterjedtebbet, és a leggyakrabban alkalmazott módszert tanulmányozunk, a **címszerinti multiplexelés**t, ahol az adatokat címmel látjuk el, megcímezve továbbítjuk.

Címzés

A címszerinti adatgyűjtés-elosztás hasonló a posta működéséhez. A posta összegyűjti a továbbítandó adatokat (leveleket, küldeményeket), melyek egyesével meg vannak címezve. Az összegyűjtött adatokat (melyeket bizonyos elvek szerint csoportosít) kevés úton elviszi nagy távolságra (pl. a magyarországi keddi New York-i küldeményeket New Yorkba egyetlen egy útvonalon (repülőgépen), egyszerre az összes küldeményt), és ott szétosztja, minden egyes küldeményt az ő címére juttatva.

Az elektronikában is így teszünk. Az adatokat egységenként, pl. bájtonként, vagy bitenként címmel látjuk el, és így továbbítjuk, nem csak az adatot, hanem az adat címét is. Ehhez sokkal

kevesebb adathordozó (pl.) vezeték kell, mintha minden adatnak külön vezetéke, adathordozója lenne. Az adatokkal továbbított címek segítségével osztjuk szét rendeltetési helyükön az adatokat.

Multiplexer áramkörök

Előbb készítsünk egy 4-ről 1-re multiplexelő áramkört, majd egy 8-ról 1-re multiplexelőt. Vizsgáljuk meg, milyen következtetéseket vonhatunk le.

4-ről 1-re multiplexelő áramkör

Meggondolások:

4 féle adathoz négy különböző cím kell

4 különböző címet 2 bittel tudunk előállítani, így két címbitünk lesz

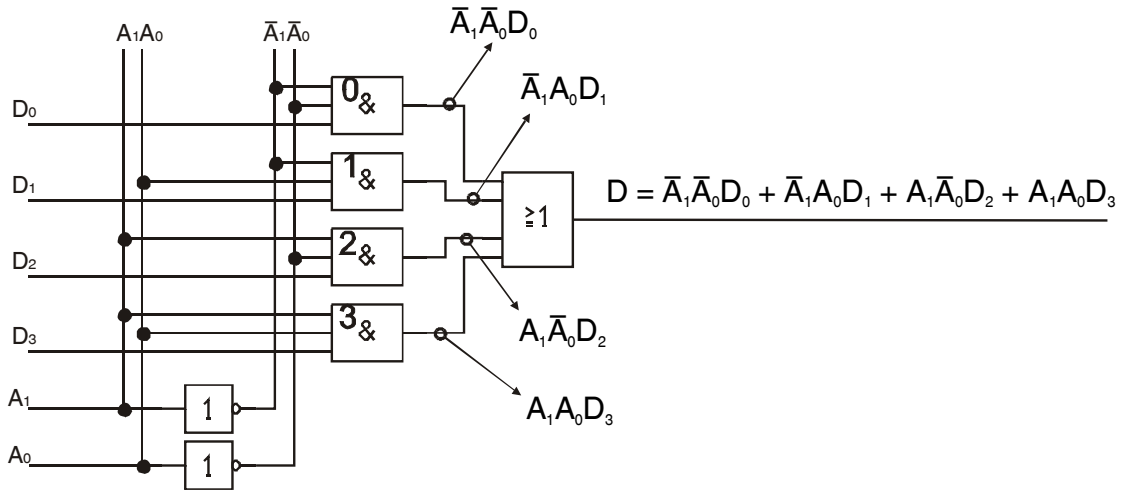
Adjunk az adatoknak D_0, D_1, D_2, és D_3 jelölést. Jelöljük a címbiteket A_0-val és A_1-gyel.

Minden adathoz az ő címét rendeljük a következőképpen, hozzuk a megfelelő számú adatot ÉS kapcsolatba a számának megfelelő kombinációjú (mintermű) címmel: $\overline{A}_1\overline{A}_0D_0$, $\overline{A}_1A_0D_1$, $A_1\overline{A}_0D_2$, $A_1A_0D_3$

Ekkor a továbbítandó adatokból egy cím esetén csak egy adat lehet az ő címével ÉS kapcsolatban igaz egyszerre, így egyetlen egy vezetéken tudjuk az összes adatot továbbítani, mindig egyszerre csak egy adatot, amelyik a cím szerint ki van választva. Ennek logikai egyenlete a következő: $D = \overline{A}_1\overline{A}_0D_0 + \overline{A}_1A_0D_1 + A_1\overline{A}_0D_2 + A_1A_0D_3$

Tekintsük meg ennek a logikai egyenletnek a rajzát:

Adatválasztó (multiplexer) 4-ről 1-re



Az ábrán bejelöltük, hogy kapuzzuk ki a különböző címek szerint a továbbítandó adatokat.

Ennek a kapcsolásnak nem sok előnye van, a négy adatvezeték helyett egy adatvezeték, de még két címvezeték is kell, ráadásul egyszerre csak egy bit információt tudunk átvinni, azaz négyszer lassabb lesz az adatátvitel sebessége, mintha minden adatbit saját vezetéket kap.

Azonban ha csak eggyel növeljük a címbitek (címvezetékek) számát, már kétszer annyi adatot tudunk átvinni. Ha megint eggyel növeljük a címbitek számát, ismét kétszer annyit, azaz a címbitek számának növelésével rohamosan nő a kevés számú vezetéken átvihető adatmennyiség.

8-ról 1-re multiplexelő áramkört rajzolunk fel.

Meggondolások:

8 féle adathoz nyolc különböző cím kell

8 féle címet három bittel tudunk előállítani, így három címbitünk lesz

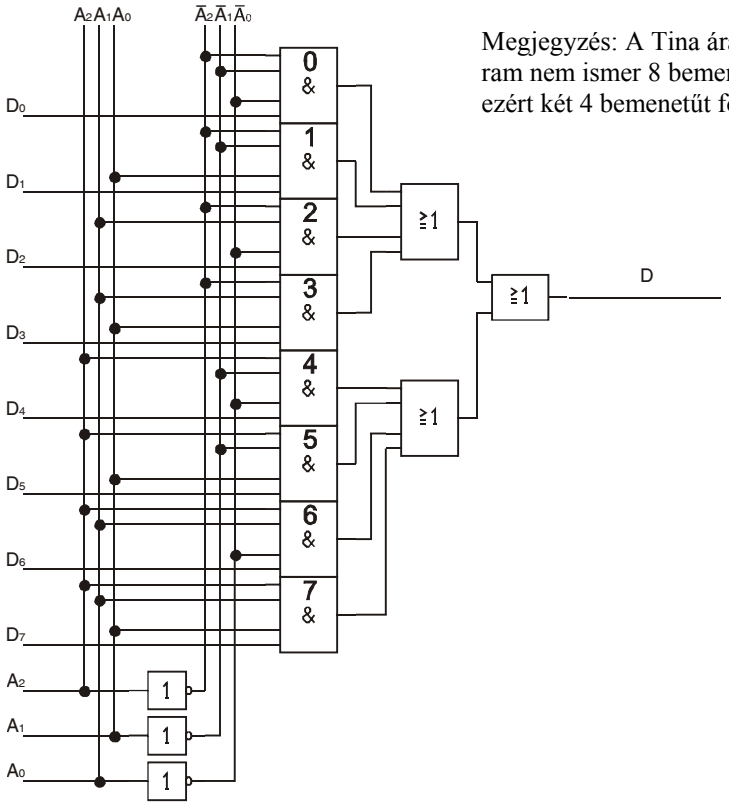
Jelöljük az adatokat D_0, D_1, ... D_7-tel. Jelöljük a címbiteket A_0, A_1, és A_2-vel.

Minden adathoz az ő címét rendeljük a következőképpen, hozzuk a megfelelő számú adatot ÉS kapcsolatba a számának megfelelő kombinációjú (mintermű) címmel: $\overline{A}_2\overline{A}_1\overline{A}_0D_0$, $\overline{A}_2\overline{A}_1A_0D_1$, $\overline{A}_2A_1\overline{A}_0D_2$, $\overline{A}_2A_1A_0D_3$, $A_2\overline{A}_1\overline{A}_0D_4$, $A_2\overline{A}_1A_0D_5$, $\overline{A}_2A_1A_0D_6$, $A_2A_1A_0D_7$

Ekkor a továbbítandó adatokból egy cím esetén csak egy adat lehet az ő címével ÉS kapcsolatban igaz egyszerre, így egyetlen egy vezetéken tudjuk az összes adatot továbbítani, mindig egyszerre csak egy adatot, amelyik a cím szerint ki van választva. Ennek logikai egyenlete a következő: $D = \overline{A}_2\overline{A}_1\overline{A}_0D_0 + \overline{A}_2\overline{A}_1A_0D_1 + \overline{A}_2A_1\overline{A}_0D_2 + \overline{A}_2A_1A_0D_3 + A_2\overline{A}_1\overline{A}_0D_4 + A_2\overline{A}_1A_0D_5 +$
$+ \overline{A}_2A_1A_0D_6 + A_2A_1A_0D_7$

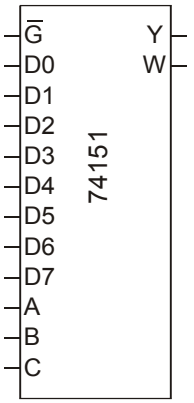
Tekintsük meg ennek a logikai egyenletnek a rajzát:

Adatválasztó (multiplexer) 8-ról 1-re Tina áramkörszerkesztő programmal



Megjegyzés: A Tina áramkörszerkesztő program nem ismer 8 bemenetű VAGY kaput, ezért két 4 bemenetűt fogtunk össze.

Az SN74151 típusú 8line-to1 multiplexer IC



Demultiplexer áramkörök

Adatelosztó, demultiplexer 1-ről 4-re

Meggondolások:

4 féle adathoz négy különböző cím kell

4 féle címet 2 bittel tudunk előállítani, így két címbitünk lesz

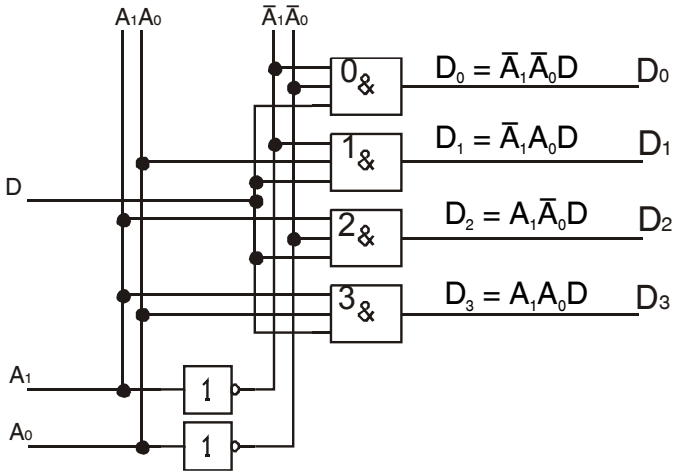
Adjunk az adatoknak D_0, D_1, D_2, és D_3 jelölést. Jelöljük a címbiteket A_0-val és A_1-gyel.

Minden adatot az ő címe szerint válogatjuk ki a következőképpen, hozzuk az adatot ÉS kapcsolatba a számának megfelelő kombinációjú (mintermű) címmel: $D_0 = \overline{A}_1\overline{A}_0D$, $D_1 = \overline{A}_1A_0D$, $D_2 = A_1\overline{A}_0D$ és végül $D_3 = A_1A_0D$

Ekkor az érkezett, elosztandó adatokból egy cím esetén csak egy adat lehet az ő címével ÉS kapcsolatban igaz egyszerre, így az egyetlen egy vezetéken érkezett összes adatot el tudjuk osztani, de mindig egyszerre csak egy adatot, amelyik az ő címe szerint ki van választva.

Tekintsük meg a fentieknek megfelelő kiválogatás a logikai áramköri a rajzát:

Adatelosztó, demultiplexer 1-ről 4-re Tina áramkörszerkesztő programmal



Ennek az áramkörnek sem sok jelentősége van, de a címbitek számával rohamosan (exponenciálisan, kettő mértani sora szerint) nő a szétoszthtató adatok mennyisége.

Az eddigiek alapján építsünk kétszer több adatot fogadni képes áramkört. Tekintsük meg az 1-ről 8-ra demultiplexer áramkört.

Adatelosztó, demultiplexer 1-ről 8-ra

Meggondolások:

8 féle adathoz nyolc különböző cím kell

8 féle címet 3 bittel tudunk előállítani, így három címbitünk lesz

Adjunk az adatoknak D_0, D_1, ... és D_7 jelölést. Jelöljük a címbiteket A_0, A_1 és A_2-vel.

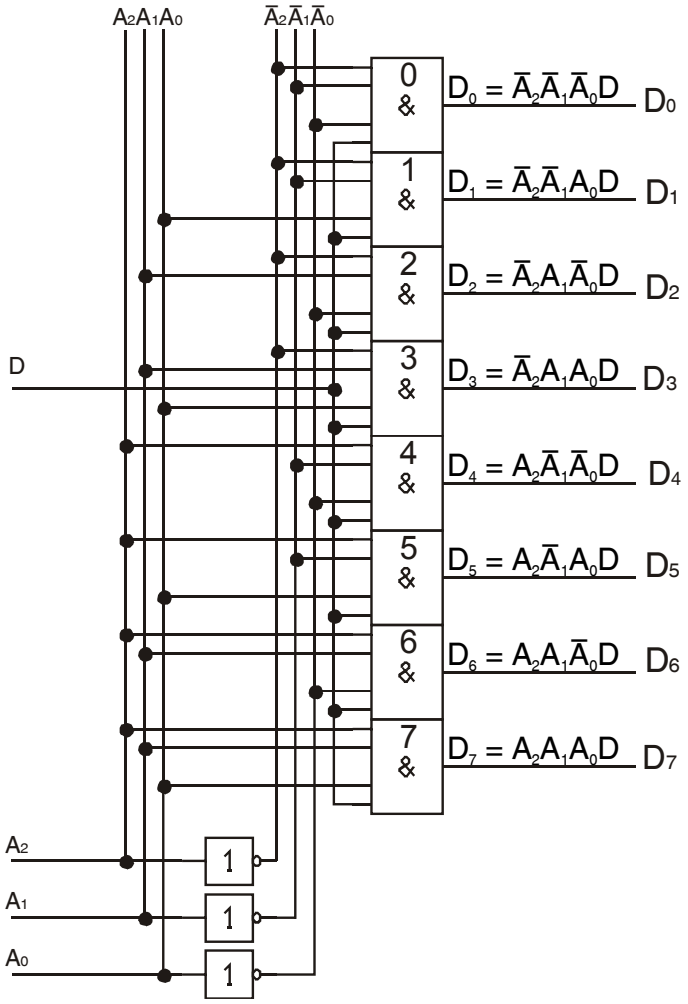
Minden adatot az ő címe szerint válogatjuk ki a következőképpen, hozzuk az adatot ÉS kapcsolatba a számának megfelelő kombinációjú (mintermű) címmel:

$D_0 = \overline{A}_2\overline{A}_1\overline{A}_0D$, $D_1 = \overline{A}_2\overline{A}_1A_0D$, $D_2 = \overline{A}_2A_1\overline{A}_0D$, $D_3 = \overline{A}_2A_1A_0D$, $D_4 = A_2\overline{A}_1\overline{A}_0D$, $D_5 = A_2\overline{A}_1A_0D$, $D_6 = A_2A_1\overline{A}_0D$, $D_7 = A_2A_1A_0D$

Ekkor az érkezett, elosztandó adatokból egy cím esetén csak egy adat lehet az ő címével ÉS kapcsolatban igaz egyszerre, így az egyetlen egy vezetéken érkezett összes adatot el tudjuk osztani, de mindig egyszerre csak egy adatot, amelyik az ő címe szerint ki van választva.

Tekintsük meg a fentieknek megfelelő kiválogatás a logikai áramköri a rajzát:

Adatelosztó, demultiplexer 1-ről 8-ra Tina áramkörszerkesztő programmal



Dekóder-demultiplexer

Ha megnézzük az áramkörök katalógusaiban vagy a Tina áramkörszerkesztő programban, általában a demultiplexereket dekódernek is nevezik. A demultiplexerek felépítésüknél fogva dekódolják a bináris kódot. Ha ugyanis a D adat mindig 1, akkor a demultiplexer áramkör kimenetei közül egyszerre egy és csakis egy mindig igaz lesz, a többi pedig mind hamis. Pontosan ez **a bináris kódból való dekódolás: a bemenetre érkező n bites bináris kódból fizikailag létező 2^n huzalra dekódolja a jelet**.

Ha tehát a címbemeneteket a bináris kód szerint használjuk, fizikailag létező (huzalozott) kombinációt kapunk a logikaiból, a bemenetekre binárisan kódolt n-edik kombinációt adjuk, a kimenetek közül éppen az n-edik huzal lesz igaz.

A demultiplexerhez készített adat-kivezetést kapunak (Gate, G) nevezzük, ha G hamis, le van tiltva a dekóder, ha G igaz, engedélyezve van.