Írta:
MATIJEVICS ISTVÁN
Szegedi Tudományegyetem

DIGITÁLIS TECHNIKA INTERAKTÍV PÉLDATÁR

Egyetemi tananyag

TYPOTEX

2011

LEKTORÁLTA: Dr. Jeges Zoltán, Dunaújvárosi Főiskola Informatikai Intézet Számítógéprendszerek és Irányítástechnika Tanszék

TÁMOGATÁS:
Készült a TÁMOP-4.1.2-08/1/A-2009-0008 számú, „Tananyagfejlesztés mérnök informatikus, programtervező informatikus és gazdaságinformatikus képzésekhez” című projekt keretében.

ISBN 978-963-279-528-7

KÉSZÜLT: a Typotex Kiadó gondozásában

FELELŐS VEZETŐ: Votisky Zsuzsa

AZ ELEKTRONIKUS KIADÁST ELŐKÉSZÍTETTE: Faragó Andrea

KULCSSZAVAK:

digitális technika elemi kapuk, grafikus és numerikus egyszerűsítés, tárolók, számlálók, regiszterek, digitális rendszerek szimulálása.

ÖSSZEFOGLALÁS:

Az interaktív Digitális technika példatár segítségével a hallgatók, felhasználók az elméleti anyagot példákon keresztül, azok megoldásával, valamint a kapcsolások szimulálásával tehetik érthetővé.

A példatár a digitális technika következő fejezeteinek megértéséhez nyújt segítséget:

- Boole algebra, logikai függvények, algebrai alak, igazságtáblázat, Venn-diagram, kanonikus alak, minterm és maxterm. Logikai függvények minimalizálása.
- Elemi (alapkapuk) és összetett (kódolók, dekódolók, multiplexerek, demultiplexerek, összeadók, komparátorok stb.) kombinációs hálózatok.
- Szekvenciális hálózatok. Szinkron és aszinkron sorrendi hálózatok. Leírási módszerek (állapottábla, gráfok), állapotok, működési modellek.
- Tárolóelemek (S-R, J-K és T flip-flopok). Szinkron és aszinkron működésű flip-flopok.
- Összetettebb sorrendi hálózatok (számlálók, shift regiszterek, stb.)

TARTALOMJEGYZÉK

1. Támogató

Jelen interaktív Digitális technika példatár megírását a következő pályázat tette lehetővé:

Pályázat azonosító: **TÁMOP–4.1.2–08/1/A–2009–0008**

Pályázat megnevezése: **„Tananyagfejlesztés és tartalomfejlesztés különös tekintettel a matematikai, természettudományi, műszaki és informatikai (MTMI) képzésekre”** pályázati felhívásra benyújtott **„Tananyagfejlesztés mérnök informatikus, programtervező informatikus és gazdaságinformatikus képzésekhez”** című pályázati projekt.

2. Előszó

A Szegedi Tudományegyetem Természettudományi és Informatikai Karán az Informatikai Tanszékcsoport szervezi a mérnökinformatikus-képzést. A tantárgyak között szerepel a Digitális technika tárgy mind a nappali, mind a levelezős programban. Az előadások és laboratóriumi gyakorlatok mellett szükséges példák megoldása a tantervi tematikák szerint. Ebben nyújt segítséget ez az interaktív példatár, amelyik hagyományos nyomtatott és elektronikus formában is elérhető, kiegészítve egy interneten keresztül elérhető interaktív programcsomaggal.

A példatár mintapéldákon keresztül mutatja be a megoldásokat, részletesen elmagyarázva az egyes lépéseket, ugyanakkor a program biztosítja a hallgatónak a gyors megoldás lehetőségét, a megtervezett áramkör szimulációját.

A kombinációs és sorrendi hálózatok legfontosabb kérdéseit így a hallgató az alapoktól a bonyolultabb rendszerekig teljes egészében áttekintheti, a példatár képessé teszi a hallgatót a szöveges, matematikai stb. alakban megadott problémák teljes feldolgozására.

Digitális technika név alatt szokták tárgyalni a Boole-algebra törvényszerűségeit, kiegészítve gyakorlati–műszaki problémákkal, a berendezések tervezésével és üzemeltetésével. A műszaki berendezések analóg megoldásait mára már döntően kiszorították a digitális bináris (kétállapotú) elven működő főleg elektronikus berendezések (kétállapotú pneumatikus elemek is léteznek). A számítástechnika és informatika, amely szakterület a digitális technika eszközeit használja, igen erőteljes fejlődést mutat, a témakörökhöz kapcsolódó irodalom, tankönyvek, példatárak nagy számban jelennek meg, ezekkel szembeni követelmény a folyamatos karbantarthatóság, valamint a korszerűsítés lehetősége. Digitális technikával foglalkozó könyvekből igen széles a választék. A téma klasszikusnak számít, jó magyar nyelvű könyvek is vannak, valamint találhatók példatárak is. A számítástechnikával és informatikával kapcsolatba kerülő hallgatónak szükséges digitális technikai alapismeretekkel és ilyen rendszerek megértését és tervezését elősegítő ismeretekkel rendelkezni. Ezért van jelen az informatikai képzések tanterveiben megfelelő súllyal a tantárgy.

Mivel igen fontos a logikai áramkörök tervezésében a feladat, valamint a megoldást befolyásoló műszaki paraméterek megismerése és kezelése, az interaktív példatár azzal a céllal született, hogy a fenti követelményeknek eleget tegyen. Ennek megfelelően igyekeztünk áttekintést adni mind a kombinációs, mind a sorrendi hálózatok működéséről, tervezéséről, a különböző lehetséges megvalósítási módjairól.

Az interaktív példatár követi az előadássorozat menetét, számos helyen bővebb annál, illetve az előadásokon el nem hangzó témákat is tárgyal.

A feladatoknak általában több megoldása és megvalósítása van. Igyekeztünk a hallgatókkal a mérnöki konstruktivitást megismertetni. Egy–egy feladat elemzése kitér a lehetséges válaszokra és alternatív megoldásokra is. A mérnöki modellalkotó szemléletet erősíti a program, amelyik módot ad az igen gyors paramétermódosításra, a különböző részletek elemzésére, megváltoztatására, tárolására.

A tankönyv 10 fejezetből áll:

1. Fejezet: Támogató

2. Fejezet: Előszó

3. Fejezet: Az interaktív program. Ez a fejezet tartalmazza a program részletes leírását és a használati utasítást.

4. Fejezet: Számrendszerek. A fejezet a leggyakrabban, a bináris logikához, illetve a számítástechnikához kötődő számrendszereket és számok más számrendszerbe konvertálását tárgyalja.

5. Fejezet: Egyszerűsítési (minimalizálási) eljárások. A fejezet bemutatja az egyszerűsítés matematikai alapjait, valamint részletesen foglalkozik a Karnaugh-féle grafikus.

6. Fejezet: A fejezet bemutatja a közömbös kombinációk szerepét az egyszerűsítésnél.

7. Fejezet: Itt kerül ismertetésre a Quine-féle numerikus egyszerűsítés és a McCluskey-módszer alkalmazása a redundáns implikánsok kiszűrésére.

8. Fejezet: Több kimenetű kombinációs hálózatok. A fejezet tárgyalja az esetlegesen többször megjelenő részek megtalálását, kódátalakítókon és összeadókon keresztül mutatja be a tervezést.

9. Fejezet: Logikai kapcsolások átalakítása – a De Morgan-szabály használata. A kapott logikai egyenletek megvalósítása valós eszközökkel, figyelembe véve a korlátokat (kapuszám csippen belül). Logikai kapcsolásokban található különböző kapuk kiváltása egytípusú kapuval (NEM–ÉS, NEM–VAGY).

10. Fejezet: Sorrendi (szekvenciális) hálózatok. Regiszterek és számlálók tervezésének bemutatása történik a fejezetben.

A példatár elsősorban az SZTE mérnök informatikus hallgatóinak íródott, de más digitális technika iránt érdeklődők számára is ajánljuk, akik szeretnének megismerkedni a feladatok megoldási technikájával, gyors áttekinthető segédprogramok alkalmazásával.

A könyv szándékaink ellenére bizonyára számos hiányosságot és hibát tartalmaz. A felhasználók, amennyiben ilyennel találkoznak, jelezhetik a szerzőnek, aki ezeket orvosolja, hiszen mind a szöveges rész, mind a program nyitott struktúrát képez, lehetőséget adva a bővítésre, javításra.

A példatár megalkotásában nagyon fontos szerepet játszott Bodnár Péter végzős hallgató, aki elkészítette a szimulátort, az ábrákat, itt szeretném megköszönni lelkiismeretes munkáját.

Külön köszönetet mondok Dr. Jeges Zoltán főiskolai tanárnak, aki az anyag lektorálása során igen értékes szakmai tanácsokkal segítette a példatár megalkotását.

Szeged, 2010.11.12. Matijevics István

3. Interaktív program

3.1. Bevezetés

Az interneten nagyszámú digitális technika témakörben használható program található. Vannak közöttük áramkör-szimulátorok, egyszerűsítő programok. stb. Általában különböző programnyelveken íródtak, egyesek könnyen kezelhetők, mások nehézkesek.

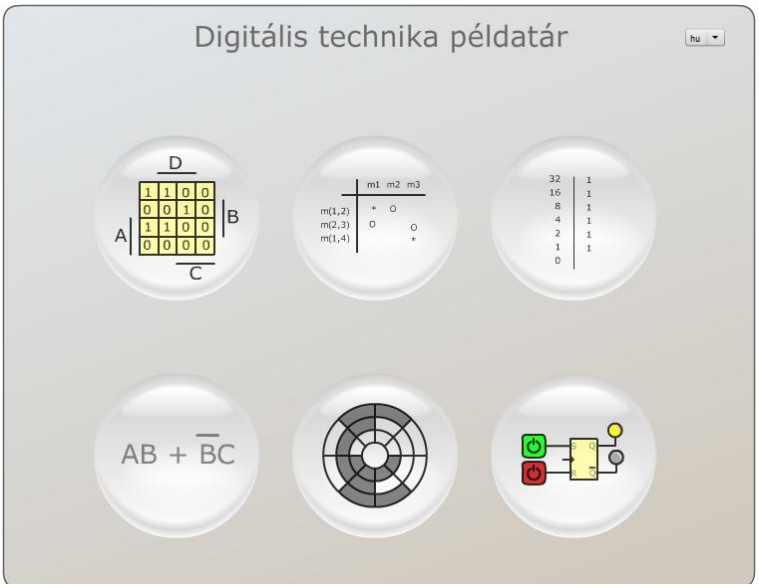
Ehhez a példatárhoz tartozó program Flash nyelven íródott, könnyen kezelhető látványos kapcsolatot teremt a felhasználó és a rendszer között, könnyen megtanulható, funkcionális.

A program leírása

A 3.1. ábrán látható a teljes programfelület, ami megjelenik a

http://www.inf.u–szeged.hu/projectdirs/digipeldatar/digitalis_peldatar.html

cím beírása után a böngészőben.



3.1. ábra: A program teljes képernyője

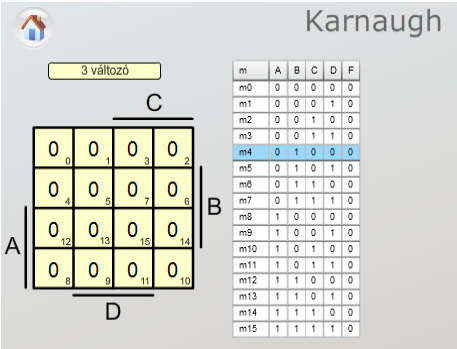
Az egyes buborékok fölé húzott egér hatására a buborék nagyobbá válik és megjelenik az adott rész teljes elnevezése. Így a következő részeket lehet elérni:

- Karnaugh-egyszerűsítés
- Quine-egyszerűsítés
- Számrendszerek
- Logikai függvények
- Gray-kód
- Szimulátor

A következő részben lehet megtalálni az egyes részekhez tartozó részletes leírást.

3.1.1. Egyszerűsítés Karnaugh módszerével

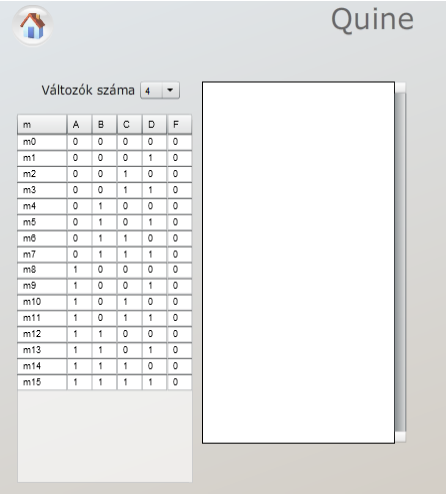
Az ablak tartalmazza a főprogramba való visszatérés ikont (bal felső sarok, kis házikó), választási lehetőséget a 3 és 4 változós egyszerűsítés között, valamint a Karnaugh- illetve az igazságtáblázatot (8 illetve 16 mintermmel).



3.2. ábra: A Karnaugh-féle egyszerűsítés képernyője

A mintermek bevitele történhet a Karnaugh-táblán, illetve az igazságtáblázaton keresztül, egyszerűen klikkelve az adott mezőre, ahol a 0 –→ 1 → x → 0 stb. átmenettel állíthatjuk be a megfelelő értéket (x nem meghatározott, közömbös érték). A hurkokat a Karnaugh-táblán berajzolja a program, körülvéve a megfelelő mintermeket. A program kiszűri a redundáns implikánsokat, kiírja a matematikai egyenletet és felrajzolja tetszőleges elemekkel a megoldást.

3.1.2. Egyszerűsítés Quine–McCluskey módszerével



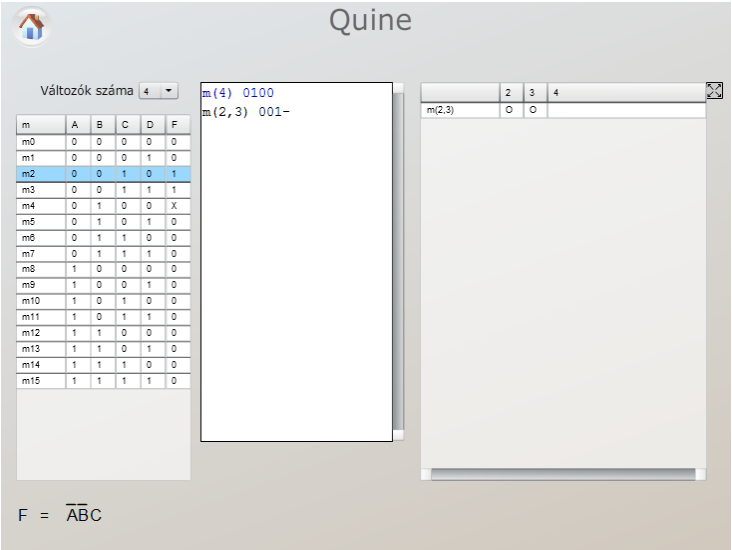
3.3. ábra: A Quine-egyszerűsítés képernyője

Az ablak tartalmazza a főprogramba való visszatérés ikont (bal felső sarok, kis házikó), választási lehetőséget a 3, 4 és 5 változós egyszerűsítés között, valamint az ablakokat az egyes lépések bemutatására.

Miután megtörténik minden implikáns megkeresése, utolsó lépésként megjelenik a McCluskey-tábla a redundáns implikánsok kiszűrésére.

Ennek az ablaknak a mérete a jobb felső sarokban levő 4-irányú nyíl segítségével nagyobbítható, görgethető a tartalom.

Ezzel egy időben a bal alsó sarokban az eredményt láthatjuk matematikai alakban. Ezt az ablakot egy példa segítségével mutatjuk meg.



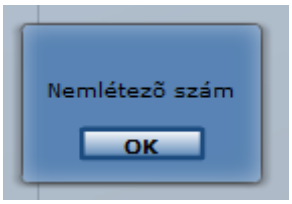
3.4. ábra: A redundáns implikánsok kiszűrése McCluskey módszerével

3.1.3. Számrendszerek

Itt lehet különböző számrendszerek között elvégezni az átalakítást Az ablak tartalmazza a főprogramba való visszatérés ikont (bal felső sarok, kis házikó). Meg kell adni az átalakítandó számhoz tartozó számrendszer alapját és meg kell adni a célszámrendszer alapját, amelyikbe át szeretnénk alakítani a számot.

3.5. ábra: A számok átalakítása egyik számrendszerből a másikba képernyő

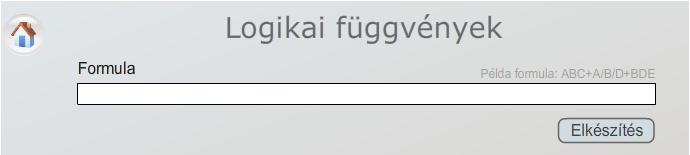
Amennyiben a számrendszerben nem szereplő számjegyet adunk meg, a rendszer hibajelzéssel válaszol.



3.6. ábra: Hibajelzés nem létező számjegy esetén

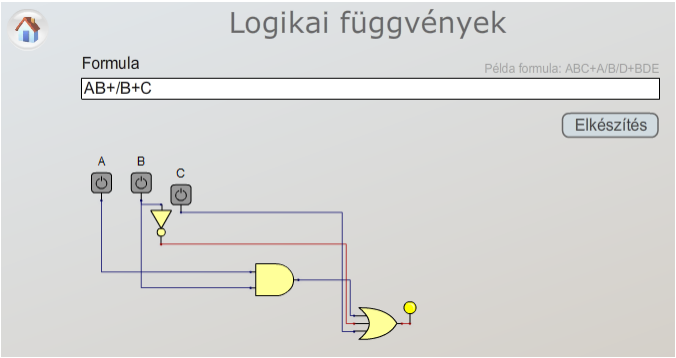
3.1.4. Logikai függvények

Itt lehet a matematikai alakban megadott logikai függvényekről logikai kapcsolási rajzot kapni. Az ablak tartalmazza a főprogramba való visszatérés ikont (bal felső sarok, kis házikó). Az ablakba be kell írni a logikai függvényt, kis és nagybetű szolgál a változók jelölésére. A negálást a változó előtti törtvonallal (/) kell jelölni, logikai VAGY kapcsolatot a plusz (+) jellel jelöljük, míg a logikai ÉS-t nem kell külön jelölni. A program nem fogadja el az F=AB alakú, csak az AB alakú jelölést is. A matematikai képlet beírása után az „ELKÉSZÍTÉS” gombbal indul a rajzolás.



3.7. ábra: A matematikai képlet beírására szolgáló ablak

Például az F = AB+/C+D függvény a következő alakú lesz:



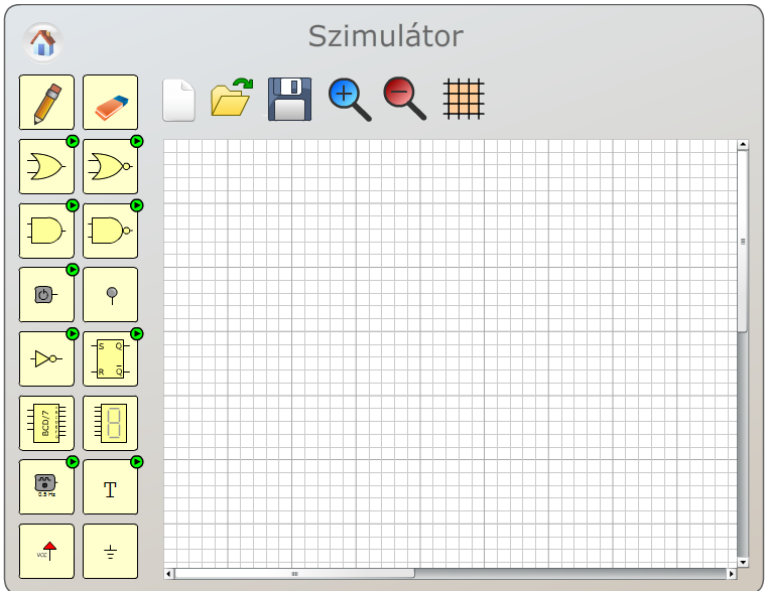
3.8. ábra: Az F = AB+/C+D függvény lerajzolva

3.1.5. Bináris Gray-átalakítás

A beírt bináris szám Gray kódú ekvivalense jelenik meg az ablakban, illetve kódtárcsán is ábrázolja a program az eredményt. Az ablak tartalmazza a főprogramba való visszatérés ikont (bal felső sarok, kis házikó).

3.1.6. A szimulátor

Ez a rendszer legösszetettebb része. A szimulátor ablaka a következőképpen néz ki:



3.9. ábra: A szimulátor teljes képernyője

 Az ablak tartalmazza a főprogramba való visszatérés ikont (bal felső sarok, kis házikó).

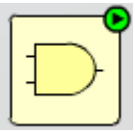
Baloldalt függőlegesen páronként találhatók meg a szimulátor elemei:

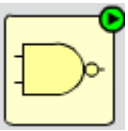
 – kimenet összekötése bemenettel. Egérrel az elem (kapu) kimenetére vitt „ceruza” és egérkattintás hatására egy nagyobb pont jelenik meg, az egérrel a másik elem bemenetére mozgatva a „ceruzát” és egérrel kattintva szintén egy pontnagyobbodást váltunk ki, aminek hatására összekötődik a két pont. Egy kimenet több bemenettel köthető össze, de a program nem enged kimenet kimenettel összekötni.

 – elemek törlése. Az egérrel a „radírt” a megfelelő törlendő elemre mozgatjuk és kattintunk, eltűnik az elem. Elem lehet kapu, összekötő vonal és szöveg.

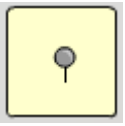
 – VAGY kapu. Jobb felső sarok zöld körben levő nyilára kattintva a kapuk lábainak számát válthatjuk 2, 3, 4, 8 bemenetűre, illetve itt található a KIZÁRÓ–VAGY kapu is. Amennyiben nem vezetünk jelet valamelyik bemenetre, a program 0-nak tekinti a jelet.

 – NEM–VAGY kapu. Jobb felső sarok zöld körben levő nyilára kattintva a kapuk lábainak számát válthatjuk 2, 3, 4, 8 bemenetűre. Amennyiben nem vezetünk jelet valamelyik bemenetre, a program 0–nak tekinti a jelet.

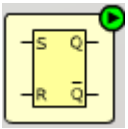
 – ÉS kapu. Jobb felső sarok zöld körben levő nyilára kattintva a kapuk lábainak számát válthatjuk 2, 3, 4, 8 bemenetűre. Amennyiben nem vezetünk jelet valamelyik bemenetre, a program 1-nek tekinti a jelet.

 – NEM – ÉS kapu. Jobb felső sarok zöld körben levő nyilára kattintva a kapuk lábainak számát válthatjuk 2, 3, 4, 8 bemenetűre. Amennyiben nem vezetünk jelet valamelyik bemenetre, a program 1-nek tekinti a jelet.

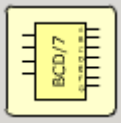
 – 0 vagy 1 előállítása. Szürke szín 0-át, zöld szín 1-est jelent. Jobb felső sarok zöld körben levő nyilára kattintva a jelforrás irányát változtathatjuk.

 – LED kimenet. A bemenetén levő logikai 0 esetén szürke, míg logikai 1 esetén sárga színű.

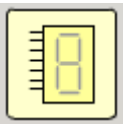
 – INVERTER. Jobb felső sarok zöld körben levő nyilára kattintva a kapu irányát határozhatjuk meg, jobbra illetve lefelé.



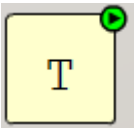
– tárolók. Jobb felső sarok zöld körben levő nyilára kattintva a tárolók típusát határozhatjuk meg, van aszinkron RS, szinkron RS, szinkron JK, szinkron T és szinkron D tároló.



– BCD – 7 szegmenses átalakító. 4 bites BCD számot alakít át 7 szegmenses kódra. A bemenet levegőben levő lába logikai 0.

 – 7 szegmenses kijelző. A nem bekötött láb logikai 0 értéken van.


– impulzusgenerátor. 50 %–os kitöltési tényezőjű négyszögjel (0 és 1) előállítására szolgál. Jobb felső sarok zöld körben levő nyilára kattintva a következő valós frekvenciák állíthatók elő: 0.5 Hz, 2 Hz, 5 Hz és 50 Hz.


– szövegbevitel. Tetszőleges szöveg írható egy szövegablakba. Jobb felső sarok zöld körben levő nyilára kattintva 4 féle méretű betű állítható elő.


– tápfeszültség. Logikai 1.


– földelés. Logikai 0.

Vízszintesen a következő parancsikonok vannak:


– új projekt. Rákattintva kérdezés nélkül törli a pillanatnyi ábrát és új projektet nyit.


– fájl olvasás. A háttértárolón megőrzött fájlt beolvassa a munkaterületre, hagyományos útkeresést biztosítva a számítógépen.


– fájl írás. A megrajzolt fájlt elmenti, hagyományos útkeresést biztosítva a számítógépen.


– kép nagyítása és kicsinyítése.


– háttérrács. A rajz mögött fehér felületet biztosít, vagy négyzetrácsot rajzol.

4. Számrendszerek

4.1. Példa

2010-et írjuk fel római számokkal.

Megoldás:

A római számok a nem helyiértékes, úgynevezett additív számábrázoláshoz tartoznak. A római szám felírásánál mindig a legnagyobb értéknél kezdünk, bal oldalon folytatjuk jobbra a kisebb és kisebb értékekkel, vagyis ezresek, százasok, tízesek és egyesek. A következő táblázatban találhatók a tízes számrendszer számai és a megfelelő betűk (római számok):

sorszám	római szám	tízes számrendszer
1.	I	1
2.	V	5
3.	X	10
4.	L	50
5.	C	100
6.	D	500
7.	M	1000

4.1. ábra: A római számok és a tízes számrendszer összefüggése

A felírásnak vannak szabályai, de sokszor eltérnek ettől a szabálytól, hogy egyszerűbb, rövidebb legyen a szám alakja. Fontos, hogy az I csak V és X előtt lehet.

2000 M és 10 = X, tehát 2000 = MMX.

4.2. Példa

1999-et írjuk fel római számokkal.

Megoldás:

Mivel megengedett a római számok felírásánál a kivonás is, ezért a példa megoldása a következő:

1000 + 900 + 90 + 9 = MCMXCIX.

Nem megengedett nagy számoknál a rövidítés, de ennek ellenére használják, így 1999 = MIM, illetve 1999 = IMM alak is előfordul.

4.3. Példa

Írjuk fel a 33-at, 34-et és 35-öt római számokkal.

Megoldás:

33 = XXXIII, 34 = XXXIV és 35 = XXXV.

4.4. Példa

Írjuk fel a 47-et, 48-at, 49-et és 50-et római számokkal.

Megoldás:

47 = XLVII, 48 = XLVIII, 49 = XLIX és L.

4.5. Példa

Írjuk fel a következő római számokat a tízes számrendszerben: DCCCXXXIX

Megoldás:

DCCCXXXIX = 500 + 100 + 100 + 100 + 10 + 10 + 10 + (–1) + 10 = 839.

4.6. Példa

Írjuk fel a következő római számokat a tízes számrendszerben: MCXI

Megoldás:

MCXI = 1000 + 100 + 10 + 1 = 1111.

4.7. Példa

Alakítsuk át 77–et bináris számmá.

Megoldás:

A helyiértékes számábrázolásnál a legkisebb helyiérték jobb-, a legnagyobb helyiérték baloldalt található, balra haladva duplázódnak az értékek. 77–et a következő módon írhatjuk fel:

$$1 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 64 + 8 + 4 + 1 = 77.$$

Egy lehetséges módja az átalakításnak a kettővel való egészszámú osztás, a maradék lesz a kettes alapú szám számjegye (0 vagy 1).

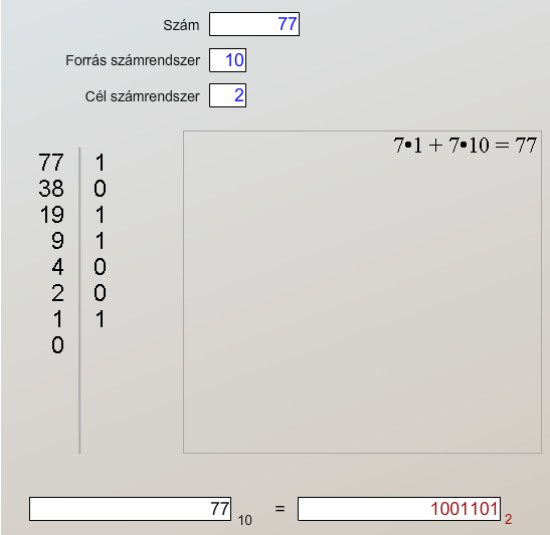
77 : 2 = 38	1	1
38 : 2 = 19	01	0
19 : 2 = 9	101	1
9 : 2 = 4	1101	1
4 : 2 = 2	01101	0
2 : 2 = 1	001101	0
1 : 2 = 0	1001101	1

4.2. ábra: 77 átalakítása kettes számrendszerbe

Az eredményt alulról – felfelé olvassuk, vagyis:

$$77_{10} = 1001101_2 .$$

A programmal lefuttatva:



4.3. ábra: 77 átalakítása bináris számmá a programmal

4.8. Példa

Alakítsuk át 0.738–at bináris számmá.

Megoldás:

Az eljárásnál a kettővel való szorzást kell alkalmazni, ha a szorzat egynél nagyobb, 1 az eredmény, ha kisebb 0. Amikor egynél nagyobb a kapott szorzat, akkor csak a törtrész szorzódik újból.

0.738 * 2 = 1.476	0.1	1
0.476 * 2 = 0.952	0.10	0
0.952 * 2 = 1.904	0.101	1
0.904 * 2 = 1.808	0.1011	1
0.808 * 2 = 1.616	0.10111	1
0.616 * 2 = 1.232	0.101111	1
0.232 * 2 = 0.464	0.1011110	0
0.464 * 2 = 0.928	0.10111100	0

4.4. ábra: 0.738 átalakítása kettes számrendszerbe

A kapott eredmény: $0.738_{10} = 0.10111100_2$.

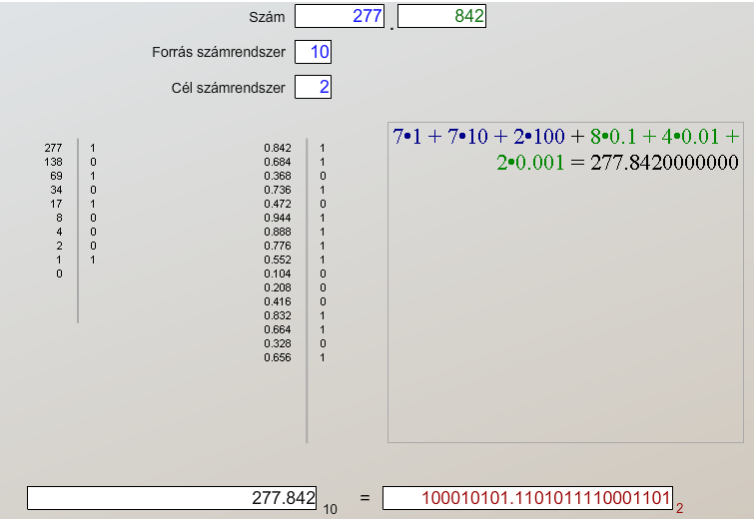
Megjegyzés: felvetődik a kérdés, hány tizedesig kell folytatni az eljárást, ugyanis minden újabb bináris érték pontosabbá teszi az eredményt. A feladat megadásakor kell megadni a pontosságot.

4.9. Példa

Alakítsuk át 277.842–t bináris számmá és írjuk be az egész részt egy 8 bites, a tört részt szintén egy 8 bites regiszterbe.

Megoldás:

A programot használva a következő eredményt kapjuk:



4.5. ábra: 277.842 átalakítása bináris számmá

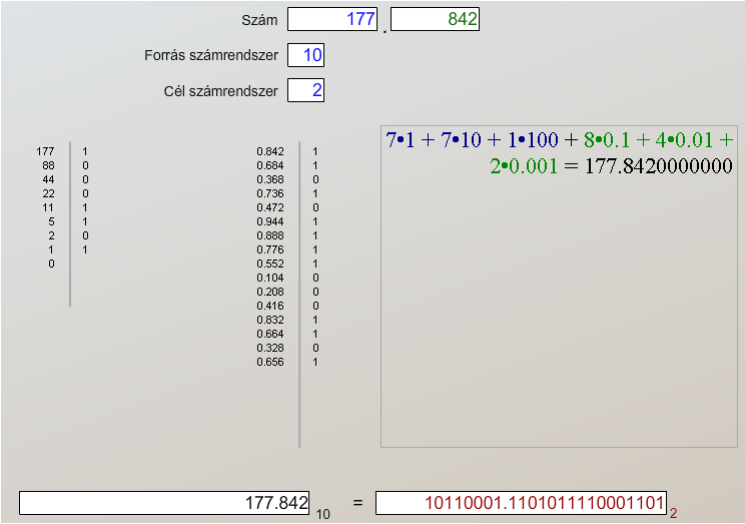
A kettes számrendszerben kapott eredmény egész része 9 bitet tesz ki, így a feladat ezen része nem oldható meg.

4.10. Példa

Alakítsuk át 177.842–t bináris számmá és írjuk be az egész részt egy 8 bites, a tört részt szintén egy 8 bites regiszterbe. Alakítsuk vissza a kapott eredményt tízes számrendszerbe és vizsgáljuk meg mekkora az esetleges eltérés az eredeti és a visszaállított érték között.

Megoldás:

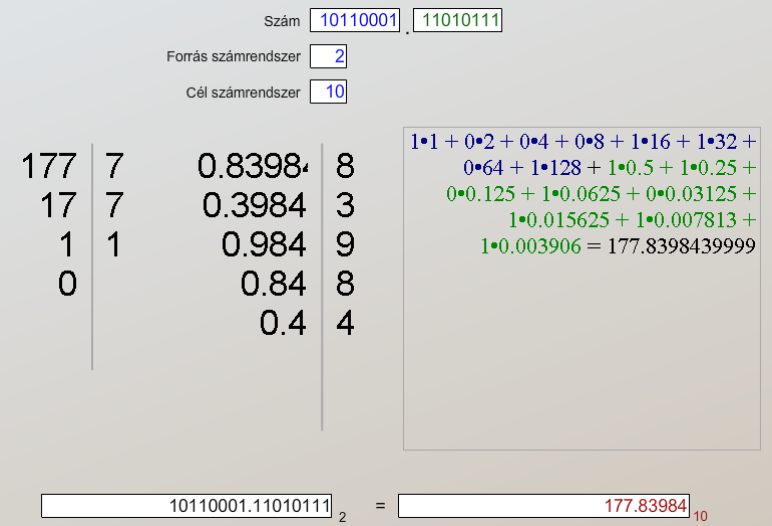
A programot használva a következő eredményt kapjuk:



4.6. ábra: 177.842 átalakítása bináris számmá

Az eredmény: $177.842_{10} = 10110001.11010111_2$

A programmal visszaalakítjuk a bináris számot tízes számrendszerbe:



4.7. ábra: 177.842 bináris ekvivalense visszaalakítva tízes számrendszerbe

Látható, hogy a szám egész részét pontosan visszakaptuk, a tizedes résznél azonban kisebb az érték, 0.842 – 0.83984 = 0.00216, azaz az eredeti érték csak 99,74 %.

Amennyiben a törtrésznél több bitet használtunk volna az ábrázolás pontosabb lett volna.

4.11. Példa

Egy 8 bites regiszter felosztunk 4 bit egész- és 4 bit törtrészre. Mekkora a legkisebb és mekkora a legnagyobb tízes számrendszerbeli szám amelyet beírhatunk a regiszterbe?

Megoldás:

A legkisebb érték 0.0, a legnagyobb értéket pedig úgy kapjuk, hogy az összes bitet a regiszterben 1-re állítjuk, ekkor az eredmény 19.8775.

4.12. Példa

Egy nyolcbites regiszterbe írjuk be a –53 számot. A számítógépek által támogatott kettes komplemensű számábrázolást alkalmazzuk a negatív számoknál.

Megoldás:

Az átalakítás lényege, hogy először a szám abszolút értékét átalakítjuk kettes számrendszerbe, utána minden egyes bitet negálunk, majd ehhez az értékhez binárisan 1–et adunk. Ez lesz a szám negatív értéke kettes komplemensű számábrázolással.

$$53_{10} = 00110101_2$$

53 kettes számrendszerben	0	0	1	1	0	1	0	1
1-es komplemens	1	1	0	0	1	0	1	0
+ 1	0	0	0	0	0	0	0	1
Kettes komplemens, –53	1	1	0	0	1	0	1	1

4.8. ábra: –53 átalakítása bináris kettes komplemenssé

4.13. Példa

Egy nyolcbites regiszterbe írjuk be a –4.72 számot. A számítógépek által támogatott kettes komplemensű számábrázolást alkalmazzuk a negatív számoknál. Egész rész 4 bit, tizedes 4 bit.

Megoldás:

A tizedes pont egy elképzelt hely, ugyanúgy kell a számot átalakítani, mintha egész szám lenne.

4.72 kettes számrendszerben	0	1	0	0	1	0	1	1
1-es komplemens	1	0	1	1	0	1	0	0
+ 1	0	0	0	0	0	0	0	1
Kettes komplemens, – 4.72	1	0	1	1	0	1	0	1

4.9. ábra: –4.72 átalakítása bináris kettes komplemenssé

Megjegyzés: Gyakran jelentkezik az a hiba, hogy a +1-et az egész résznél adják hozzá, ez helytelen, mindig a szám legkisebb helyiértékénél kell hozzáadni.

4.14. Példa

Mekkora a legkisebb és a legnagyobb beírható szám egy nyolcbites regiszterbe, ha a számok negatívak és pozitívak lehetnek?

Megoldás:

Mivel a pozitív számoknál a legnagyobb helyiértéken mindig 0 van (beleértve a 0–át is), ezért a legnagyobb szám 0 és csupa egyes, illetve a negatív számoknál (kettes komplemens) mindig 1–es van a legnagyobb helyiértéknél, ezért a legkisebb szám (legnagyobb negatív) csupa 1–es.

Legnagyobb beírható szám: $01111111_2 = +127$.

Legkisebb beírható szám: $11111111_2 = -128$.

Ezek szerint összesen 256 különböző számot írhatunk be, ugyanannyit, mintha csak egész pozitív számokat írnánk.

4.15. Példa

Alakítsuk át 38-at hexadecimális számmá.

Megoldás:

A hexadecimális szó jelentése 16, tehát egy olyan számrendszerről van szó, amelyiknek alapja 16. Mivel a mindennapi életben a tízes számrendszer terjedt el, ennek jelölésére 10 különböző számjegyet használunk, a 16-os számrendszer 16 számjegyéből csak az első 10-re van jelölésünk, a maradék 6 számjegy jelölésére az ABC első 6 betűjét használjuk.

tízes	bináris	hexadecimális
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

4.10. ábra: A tízes, bináris és hexadecimális számok összefüggése

Átalakítható a tízes számrendszerbeli szám közvetlenül 16-os számrendszerbe, de kihasználva a $2^4 = 16$ összefüggést tízes → kettes → tizenhatos átalakítást alkalmazunk, vagyis 4 bitenként csoportosítva végezzük az átalakítást.

$$38_{10} = 100110_2 = 00100110_2 = 26_{16}\,.$$

Látható, hogy a kettes számrendszerben csak 6 számjegyünk van, mivel egy–egy 16-os számrendszerbeli számjegy négy bitet kell, hogy tartalmazzon, ezért két vezető nullával bővítettük a bináris értéket nyolc bitre, jobbról kezdve 4 bitenként alakítjuk át 16-os számrendszerre.

4.16. Példa

Alakítsuk át 77.55–öt hexadecimális számmá, két egész és 2 tizedes helyen.

Megoldás:

$$77.55_{10} = 1001101.10001100_2 = 01001101.10001100_2 = 4D.8C_{16}.$$

5. Egyszerűsítési (minimalizálási) eljárások

Műszaki berendezések tervezésekor alapvető szempont az előírt specifikáció leggazdaságosabb megvalósítása. A kapuáramkörös kombinációs hálózatok esetében ez azt jelenti, hogy a függvényrendszert a lehető legkisebb költséggel kell megvalósítani. Ennek az eljárásnak a neve logikai hálózat egyszerűsítése (minimalizálása). A kevesebb kapuáramkört tartalmazó megoldás kisebb, kevesebb energiát fogyaszt, olcsóbb, kevésbé melegszik (egyszerűbb a hűtése) valamint nagyobb a megbízhatósága. Továbbá könnyebb javítani, kevesebb féle alkatrészt cserélünk.

A megvalósításhoz használt elemek típusának kiválasztására nincs szisztematikus eljárás, figyelembe kell venni az elemek működési sebességét, disszipációját, zajérzékenységét, méretét, más rendszerekhez való illesztését stb.

Mivel a logikai elemek tervezésekor is a leggazdaságosabb megoldást keressük, célszerű lenne egy olyan költségfüggvényt találni, amelyik figyelembe veszi a felhasznált áramköri rendszer tulajdonságait. Ahhoz, hogy egy ilyen eljárás minél pontosabb legyen, lehetőleg sok tényező együttes hatását kell figyelembe venni, valamint ezek egymással való kapcsolatát is. Ez viszont növeli a költségeket.

Gyakorlatban a kombinációs hálózatok tervezésénél egyszerű, jó eredményt adó költségfüggvénnyel dolgoznak, ez pedig az, hogy a megvalósítási költségeket a megvalósításhoz szükséges kapuáramkörök bemeneteinek számával veszik arányosan. Logikusan következik az előzőekből, hogy a kapuáramkörök bemeneteinek számát csökkentve csökkentjük a költségeket is.

A függvények általában redundánsak, vagyis több információ áll rendelkezésre, mint amennyi szükséges annak leírására. Egyszerűsítéskor két dolog történhet, csökken a műveletek száma és/vagy csökken a változók száma.

5.1. Kétszintű kombinációs hálózatok tervezési eljárásai

Ennél az eljárásnál ÉS és VAGY kapukat használunk, ami azt jelenti, hogy a függvény logikai konjunkciót és diszjunkciót tartalmaz.

5.2. Az egyszerűsítés matematikai alapjai

Tulajdonképpen két azonossággal lehet kifejezni az egyszerűsítést:

$$A \cdot B + \bar{A} \cdot B = (A + \bar{A}) \cdot B = B \tag{5.1}$$

és

$$(A + B) \cdot (\bar{A} + B) = A \cdot B + \bar{A} \cdot B + B = B. \tag{5.2}$$

A fenti azonosságok az algebrai, grafikus és numerikus eljárások alapjait képezik.

5.3. Grafikus minimalizálás (Karnaugh-táblázatok)

Az eljárás tulajdonképpen az algebrai egyszerűsítés grafikus megjelenítése. Mivel az ember számára a szimbólumok, ábrák könnyebben értelmezhetők és megjegyezhetők, mint a szöveges, vagy más leírás, ezért ez az eljárás gyorsabb, hatásosabb, áttekinthetőbb mint más technikák. Hátránya az eljárásnak, hogy 2–4 változós függvényeknél könnyen használható, míg négynél több változó esetében már körülményes az eljárás.

5.1. Példa

Vegyünk egy példát, legyen egy négyváltozós függvény a következő alakban megadva:

$$F(A,B,C,D) = \sum^{4}(3,5,7,13,15)$$

Töltsük ki az igazságtáblázatot, ha a függvény független változói **A**, **B**, **C** és **D**, ahol az **A** változóhoz a legnagyobb súlyt rendeljük (5.1. ábra). A **B** változó súlya fele az **A** változóénak és így tovább. Az igazságtáblázatban a sorrendet az egyes kombinációk között a súlyuk határozza meg, oly módon, hogy az 1-es változóérték hozzáadja a decimálisan keletkező értékhez a súlyt, a 0-ás nem. Figyeljük meg, hogy minden változó ugyanannyi 0 értéket, mint 1 értéket tartalmaz, csak az eloszlás különbözik. Így az **A** változónál a példában 8 0 után 8 1-es következik, a **B** változónál 4 0, 4 1, 4 0 és 4 1-es van, stb.

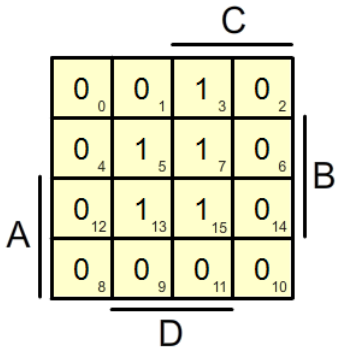
8	4	2	1
2^3	2^2	2^1	2^0

m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	0
m3	0	0	1	1	1
m4	0	1	0	0	0
m5	0	1	0	1	1
m6	0	1	1	0	0
m7	0	1	1	1	1
m8	1	0	0	0	0
m9	1	0	0	1	0
m10	1	0	1	0	0
m11	1	0	1	1	0
m12	1	1	0	0	0
m13	1	1	0	1	1
m14	1	1	1	0	0
m15	1	1	1	1	1

5.1. ábra: A kitöltött igazságtáblázat.

A Karnaugh-tábla az igazságtáblázat egy kissé átalakított alakja, ahol a mintermeket egymáshoz képest úgy rendezzük el, hogy minden minterm a szomszédjához képest csak egy változóban térjen el (akármelyik mintermben egy adott változó igaz, a mellette levő

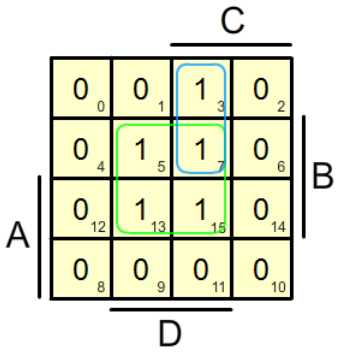
mintermben hamis, vagyis a közöttük levő Hamming-távolság 1). A mintermek csak függőlegesen és vízszintesen szomszédosak, átlósan nem, de a peremek mintermjei a túloldal peremein levő mintermekkel is „szomszédok”. Minden minterbe írjuk még be a decimális sorszámot is, feltüntetve a változókat a táblán kívül (a peremeken). Az így kitöltött táblát az 5.2. ábrán láthatjuk.



5.2. ábra: A kitöltött Karnaugh-tábla.

Az így kapott táblázatban hurkokat keresünk (összevonás, lefedés), amely hurkok azokat a mintermeket ölelik fel, amelyek egymással egyszerűsíthetők. Cél a lehető legnagyobb hurkok megtalálása (többszörös összevonás, lefedés), ez adja a legegyszerűbb alakot, hiszen a nagyobb hurokból több változó esik ki. A hurkokba foglalt 1-esek száma csak a bináris szabályt követhetik, tehát 1, 2, 4, 8, 16 stb. számú 1-est foghatnak össze. Az egyes hurkok között lehet átfedés, vagyis egy minterm több hurokban is megtalálható. Ezen kívül még figyeljünk arra is, hogy egy–egy hurokban legyen legalább egy olyan 1-es, amelyik csak az adott hurokhoz tartozzon.

Mivel a fenti példában a 3–7, 5–7, 13–15, 5–13 és 7–15 mintermkapcsolat létezik (szomszédos mintermek), ezért itt lehet egyszerűsíteni, mégpedig kialakítva a 3–7 és 5–7–13–15 hurkokat (5.3. ábra).

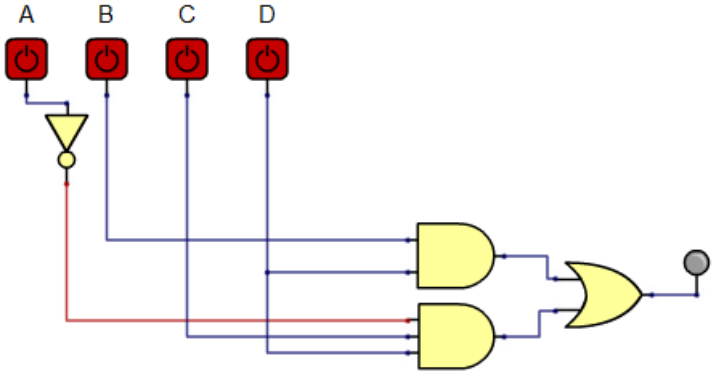


5.3. ábra: A kitöltött Karnaugh-tábla a hurkokkal.

Következő lépés az egyszerűsített függvény kiírása. A négy 1–est tartalmazó hurok A változót nem fogja tartalmazni, mivel A „metszi” a hurkot, B belekerül az egyszerűsített függvénybe, hiszen a hurkon B 1–es, C kiesik és D marad, a 2 1–est tartalmazó huroknál A negált lesz, B kiesik, C marad és D is marad, vagyis az egyszerűsített függvény:

$$F = B \cdot D + \bar{A} \cdot C \cdot D$$

Az egyszerűsített függvény rajza az 5.4. ábrán látható.



5.4. ábra: Az egyszerűsített függvény kapcsolási rajza.

Összefoglalás: A grafikus függvényegyszerűsítés szabályai:

- a lefedhető (összevonható) cellák száma 2^n (n pozitív egész szám), ha azok kölcsönösen szomszédosak,
- a lefedett cellákból a kitevőnek (n) megfelelő számú változó esik ki, amelyek a lefedés alatt változnak,
- minden 1-et tartalmazó cellát legalább egyszer le kell fedni,
- minden lefedés legalább 1 csak ehhez a lefedéshez tartozó 1-et tartalmaz.

A feladatot megoldhatjuk a példatárhoz tartozó interaktív szoftverrel is, ahol a „Karnaugh-minimalizálás" menüpontra kattintva megkapjuk a szükséges ablakokat a megfelelő utasítással. Itt a kapcsolás különböző bemeneti kombinációkkal kivizsgálható, tanulmányozható.

5.2. Példa

Tervezzünk meg egy olyan áramkört, amelyik a B logikai jelet átengedi, ha A vezérlőjel A = 0, illetve B jelet negálja, ha A = 1.

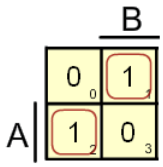
Megoldás:

Töltsük ki az igazságtáblázatot:

m	A	B	F
m0	0	0	0
m1	0	1	1
m2	1	0	1
m3	1	1	0

5.5. ábra: Az igazságtáblázat

Töltsük ki a kétváltozós Karnaugh-táblát az 5.5. ábrán levő igazságtáblázat alapján:

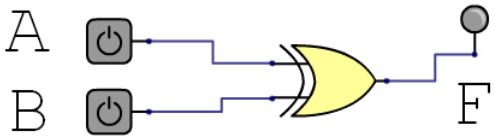


5.6. ábra: A Karnaugh-tábla

Írjuk ki a táblából a függvényt:

$$F = A\,\bar{B} + \bar{A}\,B = A \oplus B$$

Tehát a feladat megoldható egy KIZÁRÓ–VAGY kapuval is:



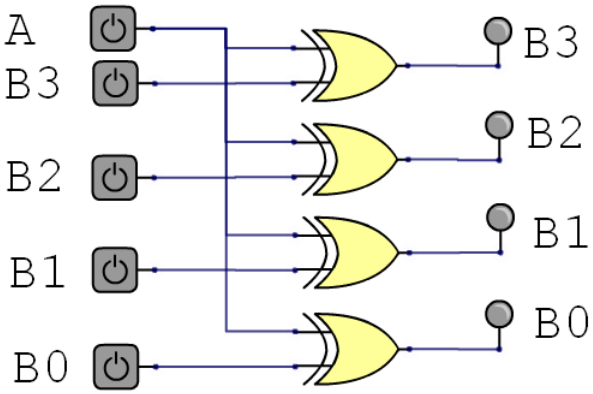
5.7. ábra: A megoldás logikai kapcsolása

5.3. Példa

Tervezzünk meg egy olyan áramkört, amelyik a 4 bites B párhuzamos adatot átengedi, ha A vezérlőjel A = 0, illetve a 4 bites B párhuzamos adatot negálja, ha A = 1.

Megoldás:

A négybites B adat minden egyes jele független a többitől. Ha az előző példa szerint megoldjuk egy bit átengedését illetve negálását, akkor ugyanezt kell tenni minden bittel egyenként, így a feladat megoldása:



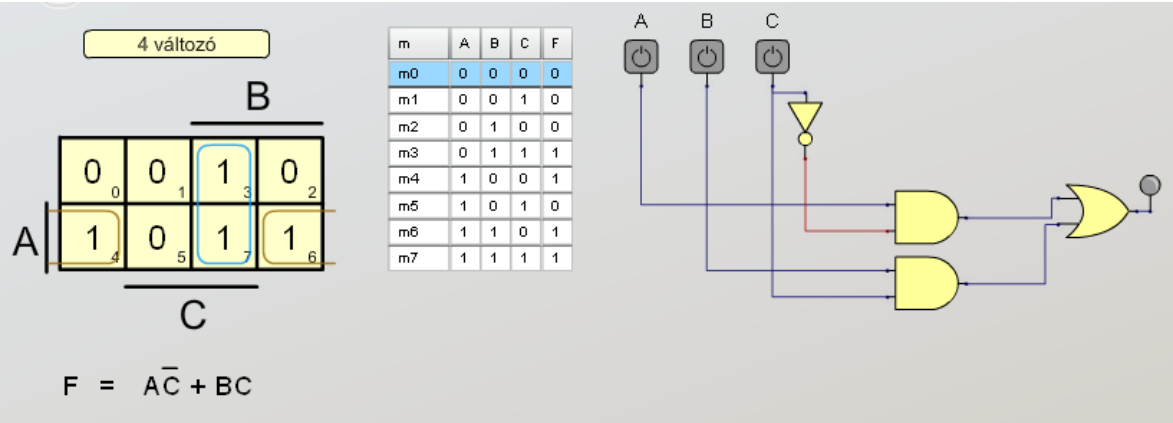
5.8. ábra: A megoldás logikai kapcsolása

5.4. Példa

Tervezzünk meg egy olyan áramkört, amelyikre két logikai jel van kapcsolva, A és B, valamint C vezérlőjel. A kapcsolás F kimenetén A jel jelenik meg, ha C = 0, illetve B jel, ha C = 1.

Megoldás:

Az áramkör neve multiplexer. Töltsük ki az igazságtáblázatot, végezzük el Karnaugh grafikus módszerével az egyszerűsítést és rajzoljuk le a logikai kapcsolást:



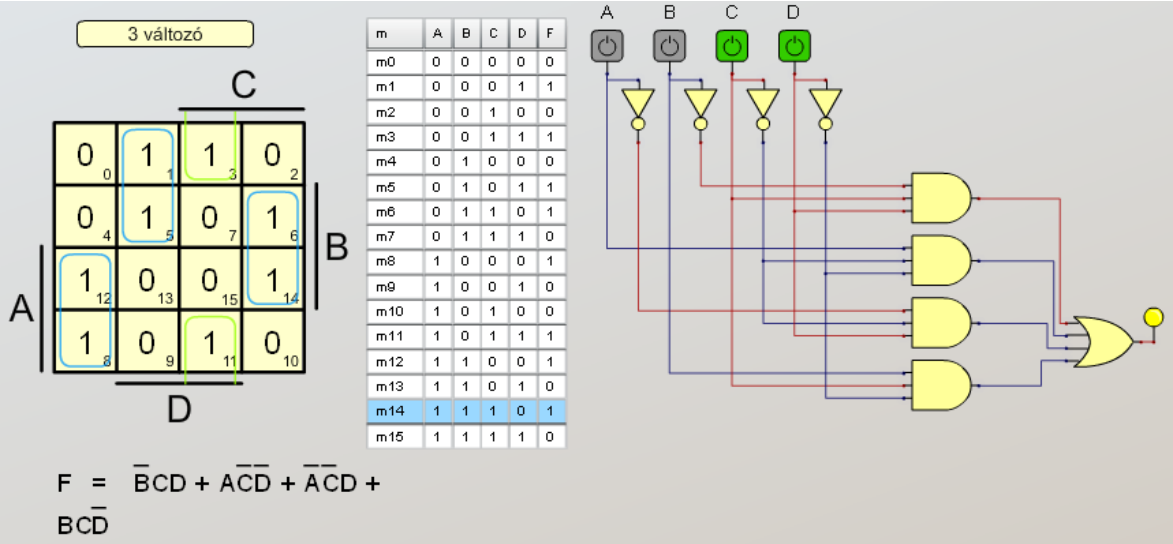
5.9. ábra: Az igazságtáblázat, Karnaugh-tábla és a kapcsolás

5.5. Példa

Tervezzünk meg egy olyan áramkört, amelyikre két logikai jel van kapcsolva, A és B, valamint C és D vezérlőjel. A kapcsolás F kimenetén A jel jelenik meg, ha C = 0, illetve B jel, ha C = 1, D = 0 esetén az eredeti jel, míg D = 1 esetén az invertált.

Megoldás:

Az előző példa multiplexerét kell kibővíteni még egy vezérlőjellel, az igazságtáblázat, a Karnaugh-tábla és a kapcsolás a következő:



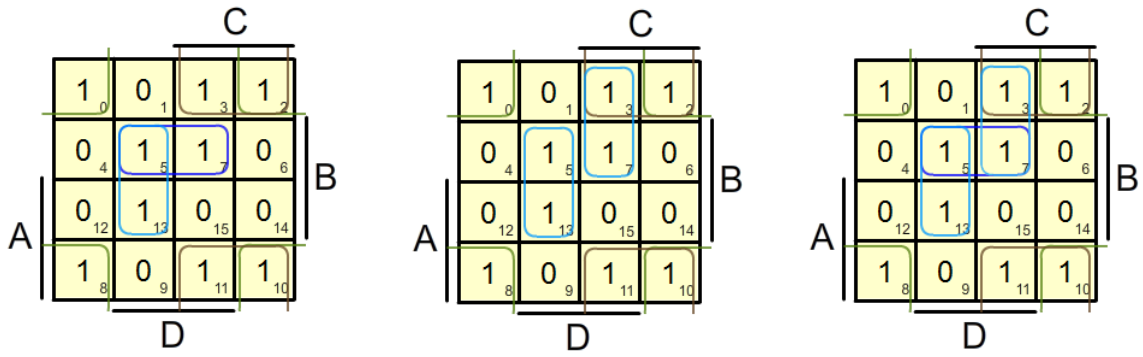
5.10. ábra: Az igazságtáblázat, Karnaugh-tábla és a kapcsolás

5.6. Példa

Minimalizáljuk a következő négyváltozós függvényt:

$$F(A,B,C,D) = \sum^{4} (0,2,3,5,7,8,10,11,13)$$

Töltsük ki a Karnaugh-táblát (5.11a, 5.11b és 5.11c ábra)



a, b, c,

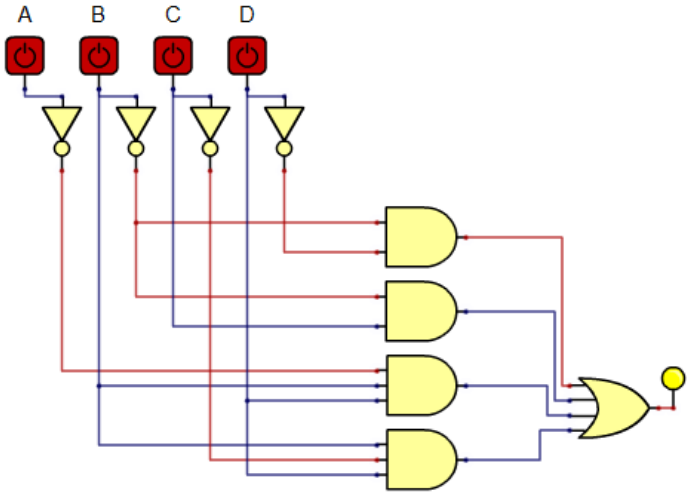
5.11. ábra: A Karnaugh-táblák

Négy dologra figyeljünk fel a megoldásnál. Első, hogy a négy sarok is egy négyes hurkot alkot, a második, hogy 3–2–11–10 mintermek is szomszédosak, így egy négyest alkotnak. A harmadik, hogy a feladatnak két megoldása van, mint ahogyan az az 5.11a és 5.11b ábrán látszik. A negyedik észrevétel, hogy az 5.11c ábra hibás, hiszen az 5–7 minterm-kapcsolat (hurok) ebben az esetben nem tartalmaz olyan mintermet (1–est), amelyik csak ehhez a hurokhoz tartozik.

Az 5.11a ábrához tartozó megoldás: $F = \bar{B} \cdot \bar{D} + \bar{B} \cdot C + B \cdot \bar{C} \cdot D + \bar{A} \cdot B \cdot D$

Az 5.11b ábrához tartozó megoldás: $F = \bar{B} \cdot \bar{D} + \bar{B} \cdot C + B \cdot \bar{C} \cdot D + \bar{A} \cdot C \cdot D$

Ezek után az 5.12 ábrán látható az 5.11a ábrán található Karnaugh-módszerrel történt egyszerűsítés logikai rajza.



5.12. ábra: A példa logikai kapcsolása egyszerűsítés után

5.7. Példa

Egy háromtagú zsűri egyidejű szavazással dönt. A kijelző akkor gyulladjon fel, ha legalább ketten támogatták a javaslatot.

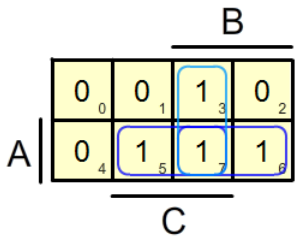
Megoldás:

Töltsük ki az igazságtáblázatot az 5.13. ábra szerint.

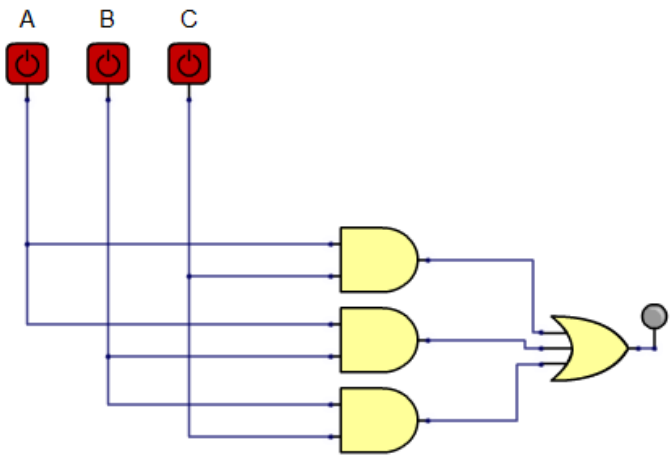
A változókhoz még egy súlyt rendelünk hozzá, mivel a szavazatok egyenértékűek, ezért mindegyik változó 1 súlyt kap. A három zsűritag 8 különböző módon alakítja ki a kombinációkat, 0 összsúlytól egészen 3-ig, belátható, hogy a szavazás akkor ad igen választ, ha a felénél több az érték, ami a mi esetünkben 2-t jelent. Így a táblázatba a 3., 5., 6. és 7. kombinációnál lesz 1-es logikai érték a kimeneten. Ezek után most már csak ki kell tölteni a Karnaugh-táblát (5.14. ábra), kiírni a függvényt és lerajzolni a logikai kapcsolást (5.15. ábra).

m	A	B	C	F
m0	0	0	0	0
m1	0	0	1	0
m2	0	1	0	0
m3	0	1	1	1
m4	1	0	0	0
m5	1	0	1	1
m6	1	1	0	1
m7	1	1	1	1

5.13. ábra: A példa igazságtáblázata



5.14. ábra: A példa Karnaugh-táblája



5.15. ábra: A feladat logikai kapcsolása

5.8. Példa

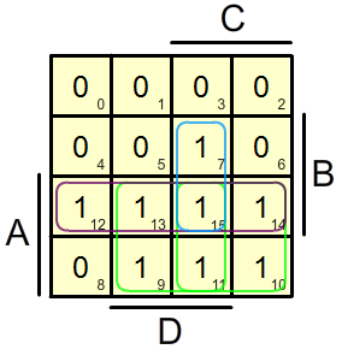
Egy négytagú zsűri egyidejű szavazással dönt. A zsűri elnöke 3, elnökhelyettese 2, a tagok pedig 1–1 szavazattal rendelkeznek. A kijelző akkor gyulladjon fel, ha legalább négy pontot érően szavaznak. Az 5.16. ábrán látható igazságtáblázatot elemezve kiderül, hogy az elnök egyedül nem tudja átvinni az akaratát, az elnökhelyettes még két zsűritagot kell, hogy megnyerjen stb.

m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	0
m3	0	0	1	1	0
m4	0	1	0	0	0
m5	0	1	0	1	0
m6	0	1	1	0	0
m7	0	1	1	1	1
m8	1	0	0	0	0
m9	1	0	0	1	1
m10	1	0	1	0	1
m11	1	0	1	1	1
m12	1	1	0	0	1
m13	1	1	0	1	1
m14	1	1	1	0	1
m15	1	1	1	1	1

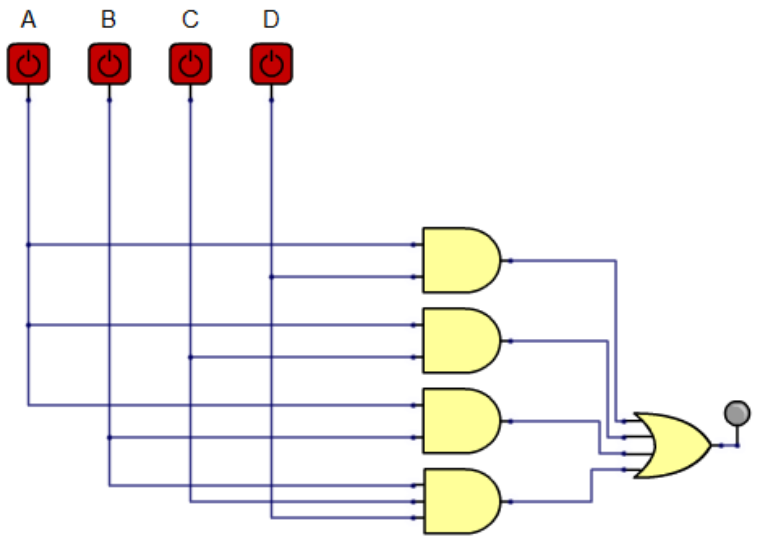
5.16. ábra: A példa igazságtáblázata

A változókhoz még egy súlyt rendelünk hozzá, az előbb említett feltételek alapján (elnök 3 pont, helyettes 2 és a tagok 1–1 pont). A négy zsűritag 8 különböző módon alakítja ki a kombinációkat, 0 összsúlytól egészen 7-ig, belátható, hogy a szavazás akkor ad igen választ, ha a felénél több az érték, ami a mi esetünkben 4–t jelent. Így a táblázatba a 7., 9., 10., 11., 12., 13., 14. és 15. kombinációnál lesz 1–es logikai érték a kimeneten. Ezek után most már csak ki kell tölteni a Karnaugh-táblát (5.17. ábra), kiírni a függvényt és lerajzolni a logikai kapcsolást (5.18. ábra).

Az igazságtáblázat alapján töltsük ki a négyváltozós Karnaugh-táblát (5.17. ábra).



5.17. ábra: A példa Karnaugh-táblája



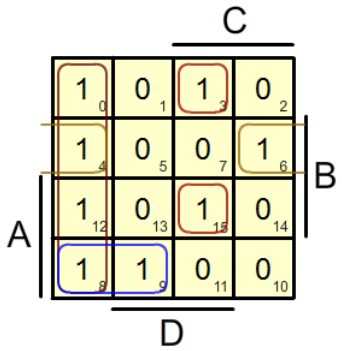
5.18. ábra: A feladat logikai kapcsolása

5.9. Példa

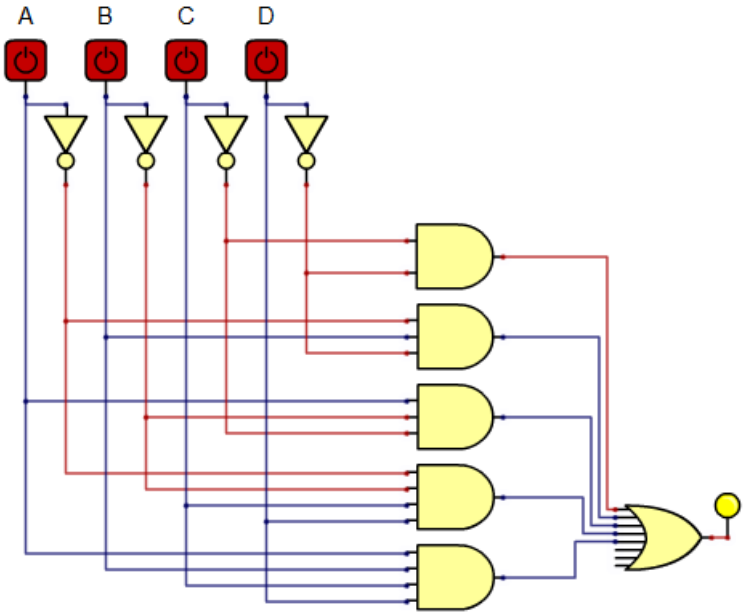
Tervezzük meg azt a négy bemenetű és egy kimenetű logikai áramkört, amelyik az F kimeneten akkor ad 1-et, ha a bemenetre maradék nélkül 3-mal és/vagy 4-gyel osztható bináris kombináció érkezik. A bemenet A, B, C és D, legmagasabb helyérték A.

Megoldás:

Mivel a bináris szabály szerint a számokat 0 és 15 között kódoljuk, ezért a fenti feladatnak a 0, 3, 6, 9 12 és 15 minterm felel meg a 3-mal való osztásnál, illetve 0, 4, 8, 12 a 4-gyel való osztásnál. Átfedés van a 0 és 12 mintermnél a két osztásnál. A Karnaugh-táblában a fenti mintermekhez rendeljünk egyeseket (5.19. ábra).



5.19. ábra: A feladat Karnaugh-táblája



5.20. ábra: A feladat logikai kapcsolása

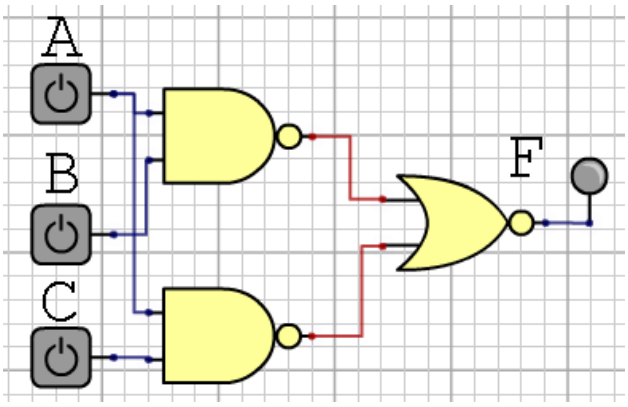
5.10. Példa

Adott a következő logikai függvény: $F(A,B,C) = \overline{\overline{A\,B} + \overline{A\,C}}$

a) Rajzolja le a logikai kapcsolást és
b) Rajzolja le az egyszerűsített logikai kapcsolást.

Megoldás:

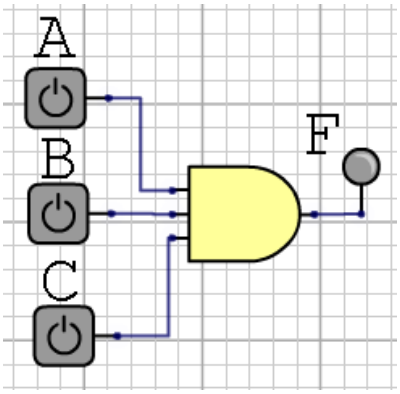
a) A függvény logikai elemekkel:



5.21. ábra: Logikai kapcsolás

b) Az egyszerűsítést elvégezhetjük matematikai módszerekkel, a De Morgan-szabály alkalmazásával:

$F(A,B,C) = \overline{\overline{A\,B} + \overline{A\,C}} = \overline{\overline{A\,B}}\,\overline{\overline{A\,C}} = A\,B\,A\,C = A\,B\,C$, vagyis a rajz az egyszerűsítés után:



5.22. ábra: A logikai kapcsolás az egyszerűsítés után

Próbáljuk meg Karnaugh módszerével is a megoldást megtalálni. Ehhez ismerni kell a mintermeket, de nekünk csak az eredeti függvény áll rendelkezésre. Kitölthetjük a függvény alapján az igazságtáblázatot, onnan a Karnaugh-táblát, vagy közvetlenül a Karnaugh-táblát. Egyszerűbb az igazságtábla kitöltése. Ha ránézünk az eredeti függvényre, akkor láthatjuk, hogy az F logikai kimenet 0 értéket vesz fel ha bármelyik változó 0 (a tag ilyenkor 1, hiszen negálva van). Csak minden változó 1 értéke esetén (7. minterm) lesz a kimenet 1.

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

5.23. ábra: A függvény igazságtáblázata

Innen pedig a függvény F = A B C.

6. Közömbös kombinációk figyelembevétele az egyszerűsítésnél

Megtörténhet, hogy a logika működése közben a bemeneten az összes kombináció közül némelyik nem fordulhat elő. Ezeket a mintermeket az igazságtáblázatban, illetve a Karnaugh-táblában x-szel jelöljük. A közömbös bejegyzések egyszerűsítést tesznek lehetővé, mivel a közömbös bejegyzéseket az egyszerűsítéshez legmegfelelőbb logikai értékkel (0 vagy 1) vehetjük figyelembe. Ezzel a módszerrel lényegében a hurkok nagyságát növeljük, aminek a következménye a változók számának csökkenése az implikánsokban.

6.1. Példa

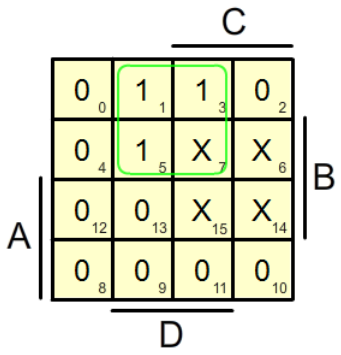
Egyszerűsítsük a következő alakban adott feladatot Karnaugh módszerével:

$$F(A,B,C,D) = \sum^4 (1,3,5) + \sum^4 (6,7,14,15)$$

A feladat igazságtábláját a 6.1. ábrán, míg a Karnaugh-táblát a 6.2. ábrán láthatjuk.

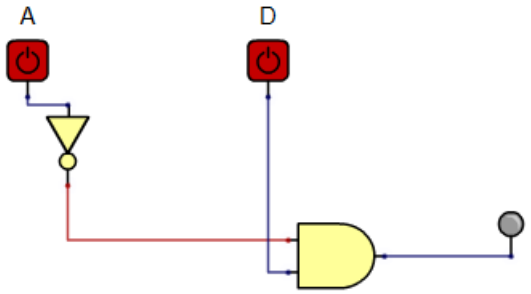
m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	1
m2	0	0	1	0	0
m3	0	0	1	1	1
m4	0	1	0	0	0
m5	0	1	0	1	1
m6	0	1	1	0	X
m7	0	1	1	1	X
m8	1	0	0	0	0
m9	1	0	0	1	0
m10	1	0	1	0	0
m11	1	0	1	1	0
m12	1	1	0	0	0
m13	1	1	0	1	0
m14	1	1	1	0	X
m15	1	1	1	1	X

6.1. ábra: A feladat igazságtáblája



6.2. ábra: A feladat Karnaugh-táblája

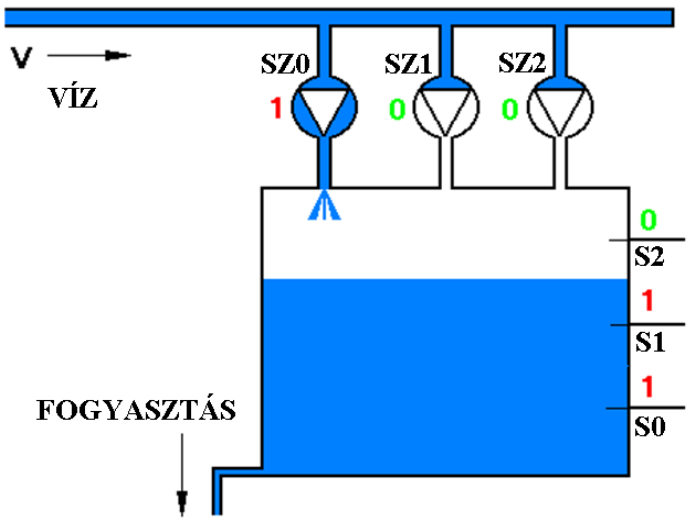
A hurkok kialakításánál láthatjuk, hogy az 1,3 és 5 minterm mellett a 7 közömbös kombinációt is figyelembe vettük, mint 1-et, így csökkentve a lefedésből adódó változók számát, ugyanakkor a 6,7,14 és 15-nél kialakítható négy mintermet tartalmazó hurok azért nem kerül kiválasztásra, mert ezek a mintermek 1 érték esetén bonyolultabb, több változót és kaput tartalmazó kapcsolást adnának, 0 esetén pedig nem szerepelnek a kifejezésben. Nem is kell, hogy szerepeljenek, hiszen elő sem fordulhatnak ezek a kombinációk. A kapcsolás az 6.3. ábrán látható.



6.3. ábra: A feladat megoldásának egyszerűsített logikai kapcsolása

6.2. Példa

Egy víztartályból 0–100 l/mp kapacitással történik a víz fogyasztása. (6.4. ábra). A tartályban 3 helyen vízszintérzékelő van, mindegyik 0 logikai értéket jelez a kimenetén, ha a víz alatta van, 1-et, ha felette van. Az elfogyasztott vizet 3 db egyforma kapacitású vízszivattyúval pótoljuk attól függően, milyen magasan áll éppen a vízszint a tartályban. Minden szivattyú kapacitása egyenként 30 l/mp. Tervezzük meg azt a logikai vezérlőt, amelyik biztosítja azt, hogy „nagyon üres" tartály esetén (vízszint S_0 alatt) a lehető legnagyobb sebességgel tölti a vizet a tartályba, „üres" tartály esetén (vízszint S_0 és S_1 között) már kisebb kapacitással mint az imént és akkor ha van elegendő víz a tartályban, de még éppen nincs tele (vízszint S_1 és S_2 között) a legkisebb kapacitással. Természetesen S_2 szint elérésekor nincs töltés. A víz pótlásakor vegyük ideálisnak a rendszert, vagyis ne törődjünk azzal, hogy a vízfelület a töltéskor mozoghat, hullámzik, így az érzékelő alatt és felett a kimenő jelet 0 és 1 érték között változtatja.



6.4. ábra: Víztartály szintérzékelőkkel és szivattyúkkal

Megoldás:

Először ellenőrizzük le azt, hogy pótolható-e az elfogyasztott víz mennyisége az adott paraméterekkel. Mivel a fogyasztás sztochasztikus, napszak, évszak stb. függő, ezért bármikor előfordulhat a 0 l/mp és 100 l/mp közötti érték, ebből a legnagyobb a 100 l/ mp, ami alatta van a 3 szivattyú összkapacitásának, vagyis

vízpótlás = 3 x szivattyúkapacitás = 3 x 30 l/mp = 90 l/mp
fogyasztás = 100 l/mp, tehát
vízpótlás < fogyasztás.

A következtetés, a feladat nem oldható meg az adott értékekkel.

6.3. Példa

Egy víztartályból 0–100 l/mp kapacitással történik a víz fogyasztása. (6.4. ábra). A tartályban 3 helyen vízszintérzékelő van, mindegyik 0 logikai értéket jelez a kimenetén, ha a víz alatta van, 1-et ha felette van. Az elfogyasztott vizet 3 db egyforma kapacitású vízszivattyúval pótoljuk attól függően, milyen magasan áll éppen a vízszint a tartályban. Minden szivattyú kapacitása egyenként 40 l/mp. Tervezzük meg azt a logikai vezérlőt, amelyik biztosítja azt, hogy „nagyon üres" tartály esetén (vízszint S_0 alatt) a lehető legnagyobb sebességgel tölti a vizet a tartályba, „üres" tartály esetén (vízszint S_0 és S_1 között) már kisebb kapacitással mint az imént és akkor ha van elegendő víz a tartályban, de még éppen nincs tele (vízszint S_1 és S_2 között) a legkisebb kapacitással. Természetesen S_2 szint elérésekor nincs töltés. A víz pótlásakor vegyük ideálisnak a rendszert, vagyis ne törődjünk azzal, hogy a vízfelület a töltéskor mozoghat, hullámzik, így az érzékelő alatt és felett a kimenő jelet 0 és 1 érték között változtatja.

Megoldás:

Először ellenőrizzük le azt, hogy pótolható–e az elfogyasztott víz mennyisége az adott paraméterekkel. Mivel a fogyasztás sztochasztikus, napszak, évszak stb. függő, ezért bármikor előfordulhat a 0 l/mp és 100 l/mp közötti érték, ebből a legnagyobb a 100 l/ mp, ami felette van a 3 szivattyú kapacitásának, vagyis

vízpótlás = 3 x szivattyúkapacitás = 3 x 40 l/mp = 120 l/mp
fogyasztás = 100 l/mp , tehát
vízpótlás > fogyasztás.

A következtetés, a feladat megoldható az adott értékekkel. Állítsunk fel néhány feltételt, majd ezekből a feltételekből kíséreljünk meg az igazságtáblázatot felállítani:

ha 0 ≥ vízszint > S_0 akkor SZ_0 + SZ_1 + SZ_2,
ha S_0 ≥ vízszint > S_1 akkor SZ_0 + SZ_1,
ha S_1 ≥ vízszint > S_2 akkor SZ_0 és
ha vízszint > S_2 akkor nincs szivattyú bekapcsolva.

Mivel a szivattyúk kapacitása azonos, más kombinációk is elképzelhetők a megoldásnál (pl. a 2. sorban SZ_1 + SZ_2, vagy SZ_0 + SZ_2), stb.

A fenti egyenlőtlenségek ugyan leírják a rendszer működését, de még nem alkalmasak a logikai vezérlő megtervezésére, ezért próbáljuk meg kitölteni az igazságtáblázatot (6.5. ábra).

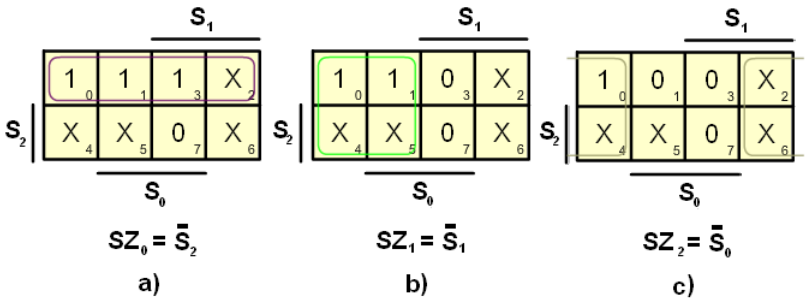
				40 l/mp	40 l/mp	40 l/mp
s.sz.	S_2	S_1	S_0	SZ_2	SZ_1	SZ_0
0.	0	0	0	1	1	1
1.	0	0	1	0	1	1
2.	0	1	0	x	x	x
3.	0	1	1	0	0	1
4.	1	0	0	x	x	x
5.	1	0	1	x	x	x
6.	1	1	0	x	x	x
7.	1	1	1	0	0	0

6.5. ábra: A feladat igazságtáblázata

Vegyük észre, hogy a 2–es sorszámú kombináció nem fordulhat elő, hiszen ha a vízszint meghaladja S_1 szintet, S_0 biztosan 1–es (természetesen feltételezzük, hogy az elemek nem hibásodhatnak meg). Ugyanez vonatkozik a 4., 5. és 6. sorra is. Ezért ezekben a sorokban x (közömbös kombinációt) írhatunk, egyszerűsítésnél a számunkra kedvező értékkel számolva.

Azt is vegyük észre, hogy valójában 3 kimenetünk van, ezt tekinthetjük 3 rendszernek, vagy 3 bemenetű és 3 kimenetű rendszernek is. Ebben a példában 3 különálló rendszerként kezeljük a kapcsolást.

Mivel a szivattyúkat kell ki–/be kapcsolgatni, ezért mindhárom szivattyúra kitöltjük a Karnaugh-táblázatokat (6.6. a, b és c ábra).



6.6. ábra: A 3 szivattyú vezérlése

6.4. Példa

Egy víztartályból 0–100 l/mp kapacitással történik a víz fogyasztása. (6.4. ábra). A tartályban 3 helyen vízszintérzékelő van, mindegyik 0 logikai értéket jelez a kimenetén, ha a víz alatta van, 1-et, ha felette van. Az elfogyasztott vizet 3 db különböző kapacitású vízszivattyúval pótoljuk attól függően, milyen magasan áll éppen a vízszint a tartályban.

A szivattyúk kapacitása a következő:

$$SZ_0 = 20 \text{ l/mp}, SZ_1 = 50 \text{ l/mp}, SZ_2 = 80 \text{ l/mp}.$$

Tervezzük meg azt a logikai vezérlőt, amelyik biztosítja azt, hogy „nagyon üres” tartály esetén (vízszint S_0 alatt) a töltés több mint 100 liter/másodperc, S_0 és S_1 között legalább 90 l/mp de legfeljebb 110 l/mp legyen, és akkor ha van elegendő víz a tartályban, de még éppen nincs tele (vízszint S_1 és S_2 között) a töltés legalább 55 l/mp de legfeljebb 75 l/mp legyen. Természetesen S_2 szint elérésekor nincs töltés. A víz pótlásakor vegyük ideálisnak a rendszert, vagyis ne törődjünk azzal, hogy a vízfelület a töltéskor mozoghat, hullámzik, így az érzékelő alatt és felett a kimenő jelet 0 és 1 érték között változtatja.

Megoldás:

Először ellenőrizzük le azt, hogy pótolható-e az elfogyasztott víz mennyisége az adott paraméterekkel. Mivel a fogyasztás sztochasztikus, napszak, évszak stb. függő, ezért bármikor előfordulhat a 0 l/mp és 100 l/mp közötti érték, ebből a legnagyobb a 100 l/ mp, ami alatta van a 3 szivattyú kapacitásának, vagyis

vízpótlás = $SZ_1 + SZ_2 + SZ_3$ = 20 l/mp + 50 l/mp + 80 l/mp = 150 l/mp
fogyasztás = 100 l/mp, tehát
vízpótlás > fogyasztás.

A szivattyúk egyenkénti és kombinált bekapcsolásával a következő l/mp értékek érhetők el:

20, 50, 70, 80, 100, 130 és 150 l/mp.

Most vizsgáljuk meg sorban az érzékelők közötti töltési értékeket:

- Üres tartály esetén több mint 100 l/mp, ehhez be kell kapcsolni mindhárom szivattyút, SZ_0 és SZ_2 együtt pont 100 l/mp, ami még nem több.
- Az S_0 és S_1 között legalább 90 l/mp de legfeljebb 110 l/mp feltétel $SZ_0 + SZ_2 = 20\frac{l}{mp} + 80\frac{l}{mp}$ = 100 l/mp értékkel elégíthető ki.
- Ha van elegendő víz a tartályban, de még éppen nincs tele (vízszint S_1 és S_2 között) a töltés legalább 55 l/mp de legfeljebb 75 l/mp legyen feltétel nem elégíthető ki az adott szivattyúk különböző kapcsolásaival.

Végkövetkeztetés: A feladat nem oldható meg.

6.5. Példa

Egy víztartályból 0–100 l/mp kapacitással történik a víz fogyasztása. (6.4. ábra). A tartályban 3 helyen vízszintérzékelő van, mindegyik 0 logikai értéket jelez a kimenetén, ha a víz alatta van, 1-et ha felette van. Az elfogyasztott vizet 3 db különböző kapacitású vízszivattyúval pótoljuk attól függően, milyen magasan áll éppen a vízszint a tartályban.
A szivattyúk kapacitása a következő:

SZ_0= 20 l/mp, SZ_1 = 50 l/mp, SZ_2 = 80 l/mp.

Tervezzük meg azt a logikai vezérlőt, amelyik biztosítja azt, hogy „nagyon üres” tartály esetén (vízszint S_0 alatt) a töltés több mint 110 liter/másodperc, S_0 és S_1 között legalább 90 l/mp de legfeljebb 110 l/mp legyen, és akkor ha van elegendő víz a tartályban, de még éppen nincs tele (vízszint S_1 és S_2 között) a töltés legalább 65 l/mp de legfeljebb 85 l/mp legyen. Természetesen S_2 szint elérésekor nincs töltés. A víz pótlásakor vegyük ideálisnak a rendszert, vagyis ne törődjünk azzal, hogy a vízfelület a töltéskor mozoghat, hullámzik, így az érzékelő alatt és felett a kimenő jelet 0 és 1 érték között változtatja.

Megoldás:

Először ellenőrizzük le azt, hogy pótolható-e az elfogyasztott víz mennyisége az adott paraméterekkel. Mivel a fogyasztás sztochasztikus, napszak, évszak stb. függő, ezért bármikor előfordulhat a 0 l/mp és 100 l/mp közötti érték, ebből a legnagyobb a 100 l/ mp, ami felette van a 3 szivattyú kapacitásának, vagyis

vízpótlás = $SZ_1 + SZ_2 + SZ_3$ = 20 l/mp + 50 l/mp + 80 l/mp = 150 l/mp
fogyasztás = 100 l/mp , tehát
vízpótlás > fogyasztás.

A szivattyúk egyenkénti és kombinált bekapcsolásával a következő l/mp értékek érhetők el:

20, 50, 70, 80, 100, 130 és 150 l/mp.

Most vizsgáljuk meg sorban az érzékelők közötti töltési értékeket:

- Üres tartály esetén több mint 110 l/mp, két megoldás van, $SZ_1 + SZ_2 = 130\ l/mp$, illetve $SZ_0 + SZ_1 + SZ_2 = 150\ l/mp$.
- Az S_0 és S_1 között legalább 90 l/mp de legfeljebb 110 l/mp feltétel $SZ_0 + SZ_2 = 20\frac{l}{mp} + 80\frac{l}{mp}$ = 100 l/mp értékkel elégíthető ki.
- Ha van elegendő víz a tartályban, de még éppen nincs tele (vízszint S_1 és S_2 között) a töltés legalább 65 l/mp de legfeljebb 85 l/mp legyen, két megoldás kínálkozik, $SZ_0 + SZ_1 = 70\ l/mp$ és a SZ_2 bekapcsolása.

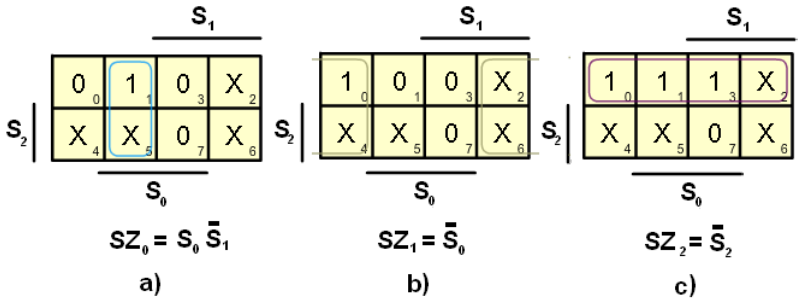
Mivel a különböző megoldások mind kielégítik az eredeti feltételeket, ezért egyéb műszaki feltételeket is figyelembe veszünk a továbbiakban, az első, berendezések ki-be kapcsolgatása jobban igénybe veszi a rendszert mint az állandósult üzemmód, illetve a második, kisebb teljesítményű berendezéseket könnyebb újból indítani, így

$SZ_1 + SZ_2$
$SZ_0 + SZ_2$ és
SZ_2 lesz a megoldás.

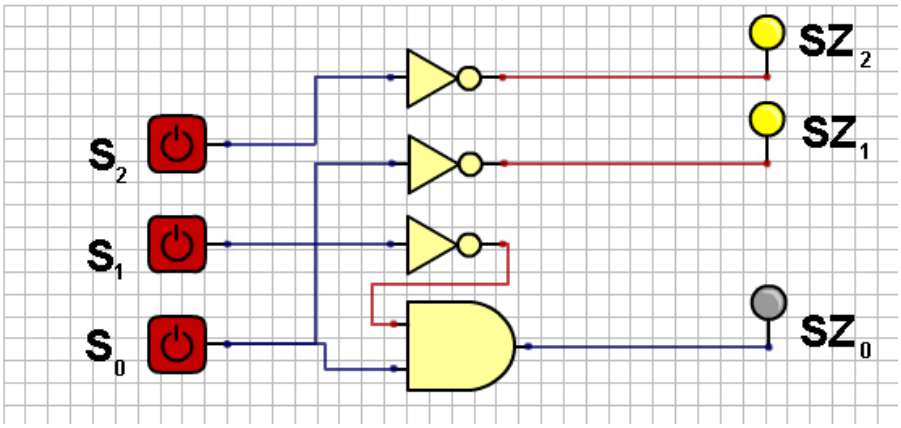
				20 l/mp	50 l/mp	80 l/mp
s.sz.	S_2	S_1	S_0	SZ_2	SZ_1	SZ_0
0.	0	0	0	1	1	0
1.	0	0	1	1	0	1
2.	0	1	0	x	x	x
3.	0	1	1	1	0	0
4.	1	0	0	x	x	x
5.	1	0	1	x	x	x
6.	1	1	0	x	x	x
7.	1	1	1	0	0	0

6.7. ábra: A feladat igazságtáblázata

A Karnaugh-táblák:



6.8. ábra: A logikai kapcsolás



6.9. ábra: A vezérlés logikai kapcsolása

7. Egyszerűsítés Quine–Mccluskey-módszerrel

7.1. A Quine-féle egyszerűsítés

Ennek a módszernek is ugyanaz a matematikai alapja mint az előző grafikus technikának. Ennél az eljárásnál az összevonási lehetőségek könnyebben megtalálhatók mint az előző módszernél, továbbra is megmarad a táblázatos ábrázolás, de most a táblázat sorait hasonlítjuk össze a szomszédsági feltétel szerint. A módszer nagy előnye, hogy gyakorlatilag tetszőleges számú változó esetén is könnyedén használható, valamint nagyon könnyű átültetni az algoritmust számítógépes programba.

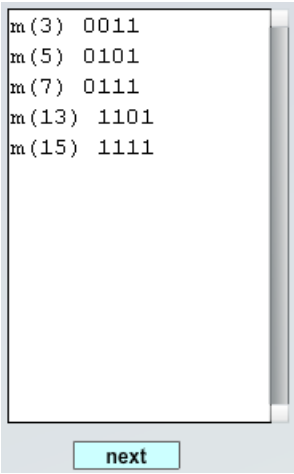
Ugyanazokat a példákat oldjuk meg, mint a Karnaugh-módszernél (5.3. fejezet). Ahol egyeznek táblázatok illetve rajzok, nem rajzoljuk újból, ezeket meg lehet találni az 5.1. fejezetben.

7.1. Példa

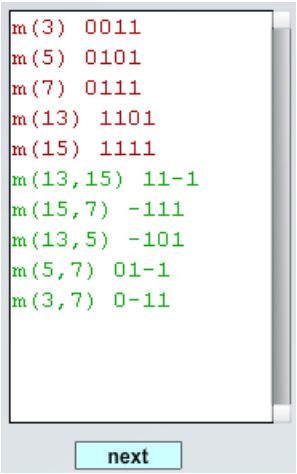
Vegyünk egy példát, legyen egy négyváltozós függvény a következő alakban megadva (egyszerűsítés Karnaugh módszerével, 5.1. példa):

$$F(A,B,C,D) = \sum^{4}(3,5,7,13,15)$$

A szimulátor Quine opcióját választva töltsük ki az ott található igazságtáblázatot, ez ugyanaz mint a 5.1. példa 5.1. ábrája. Ahogy jelöljük a mintermeket, a program kiválasztja és egy új táblázatba helyezi a mintermeket (7.1. ábra). Értelemszerűen ABCD a változók sorrendje, ahol A indexsúlya a legnagyobb, vagyis 8, míg D a legkisebb, 1.



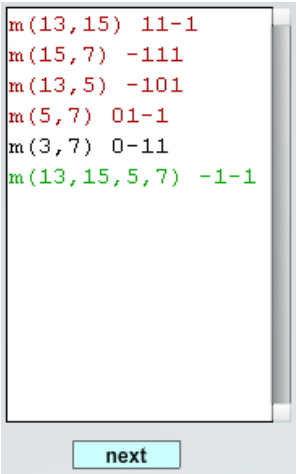
7.1. ábra: Az 1-es értékű mintermeket tartalmazó igazságtáblázat



7.2. ábra: Implikánsok egy változó egyszerűsítése után

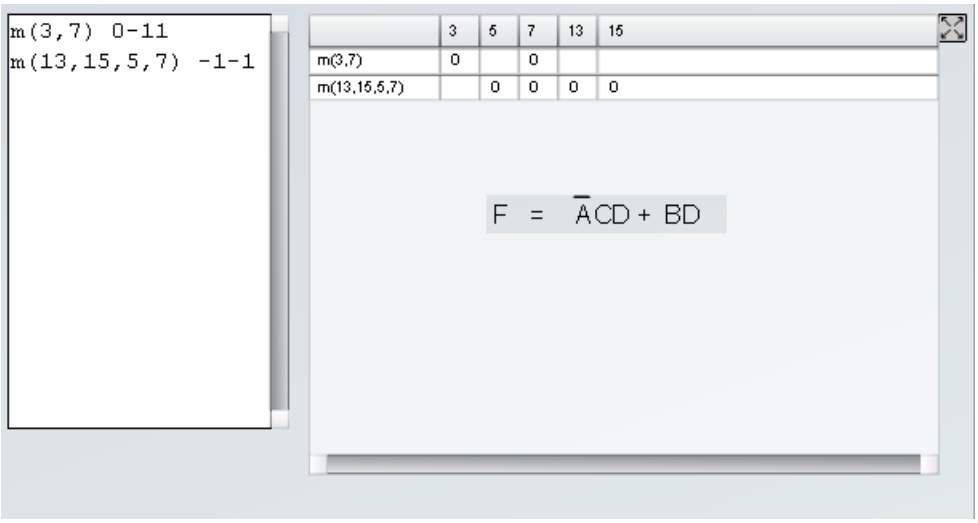
A NEXT gombra kattintva a program elvégzi az egyes mintermek összehasonlítását (minden mintermet minden mintermmel összehasonlít), vagyis a szomszédsági elv alapján minden minterm adott változójánál a ponált–négált értéket egy „–” gondolatjellel helyettesíti (7.2. ábra).

Itt leolvasható, hogy melyik két minterm vonható össze, ezt a zárójelben levő két index jelöli, illetve mi lesz a keletkezett új implikáns, 0 és 1 értékekkel felírva, a kieső változót „–” jellel helyettesítve. Ismételten a NEXT gombbal továbblépve a program összehasonlítja az előző lépésben kapott implikánsokat, szintén keresve a szomszédokat. Ennek a lépésnek az eredménye a következő:



7.3. ábra: Implikánsok két változó egyszerűsítése után

Látható, hogy a négy implikánsból (barna színnel jelezve) előállított egy olyan implikánst, amelyik az előző négyből keletkezett (zöld színnel jelölve) és nem tudta bevonni az egyszerűsítésbe a feketével jelzett implikánst. Továbblépve a NEXT gombbal az ablakba kiírja a prímimplikánsokat, vagyis nem tud tovább egyszerűsíteni, a McCluskey táblázatban pedig feltünteti az implikánsokat és bejelöli a prímimplikánsokat, ami az eredmény lesz (7.4. ábra).



7.4. ábra: A prímiplikánsok és az eredmény

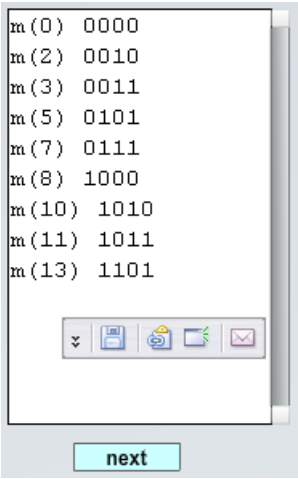
Ugyanazt az eredményt kaptuk, mint a Karnaugh-egyszerűsítésnél.

7.2. Példa

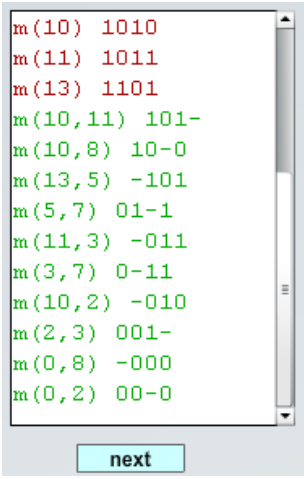
Minimalizáljuk a következő négyváltozós függvényt (5.6. példa):

$$F(A,B,C,D) = \sum^4 (0,2,3,5,7,8,10,11,13).$$

Az igazságtáblázat kitöltése után a következő értékeket kapjuk:



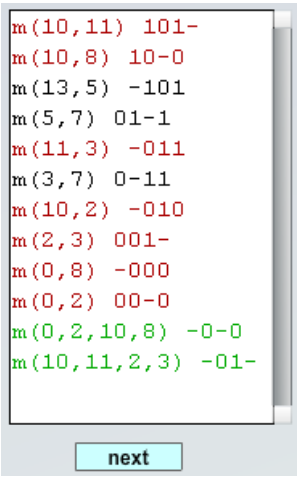
7.5. ábra: A mintermek, melyek értéke 1



7.6. ábra: Implikánsok egy változóval egyszerűsítve

A NEXT gombbal továbblépve megkapjuk azokat az implikánsokat, melyekből egy változó esett ki (az ablak mérete miatt a csuszkát húzzuk lefelé).

A NEXT gombbal továbblépve megkapjuk azokat az implikánsokat, melyekből két változó esett ki (barnával jelölve azok az implikánsok, amelyek egyszerűsíthetők voltak, feketével, amelyek nem és zölddel az új eredmény):

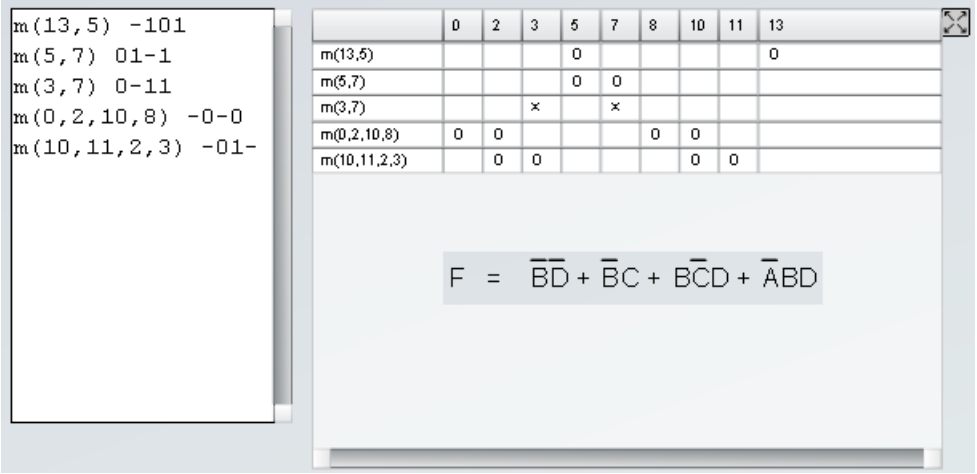


7.7. ábra: Implikánsok két változóval egyszerűsítve

Továbblépve a NEXT gombbal az ablakba kiírja a prímimplikánsokat, vagyis nem tud tovább egyszerűsíteni, a McCluskey táblázatban pedig feltünteti az implikánsokat és bejelöli a prímimplikánsokat, ami az eredmény lesz (7.8. ábra).

A McCluskey táblázatban jól látható, hogy a program a lehetséges implikánsok közül kiválasztotta a prímiplikánsokat, ezek mellett „o” jel található, míg a redundáns implikánst „x” jellel jelölte, ez nem került bele az eredménybe. Itt is látható, hogy az implikáns kevesebb változót tartalmaz, ha több mintermet lehetett a szomszédsági szabállyal egyszerűsíte-

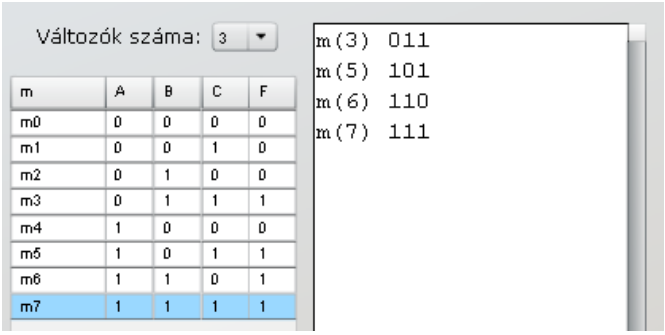
ni (nagyobb hurok a Karnaugh-egyszerűsítésnél), illetve az is látszik, hogy az összevonható mintermek száma követi a bináris szabályt (1, 2, 4, 8, 16 stb.) mint a Karnaugh-táblánál. Az eredmény megegyezik a Karnaugh-módszerrel történt egyszerűsítés eredményével, az első megoldással.



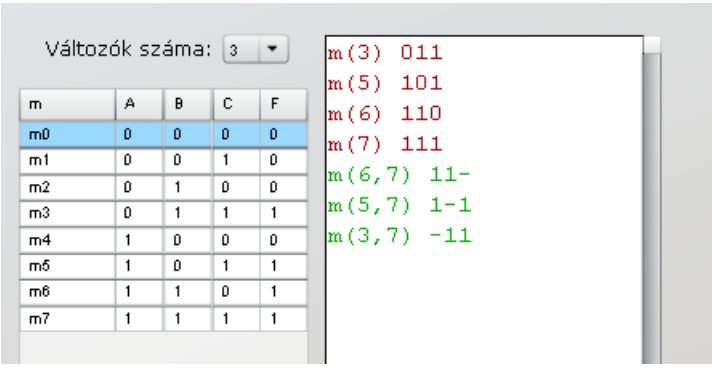
7.8. ábra: Az eredmény

7.3. Példa

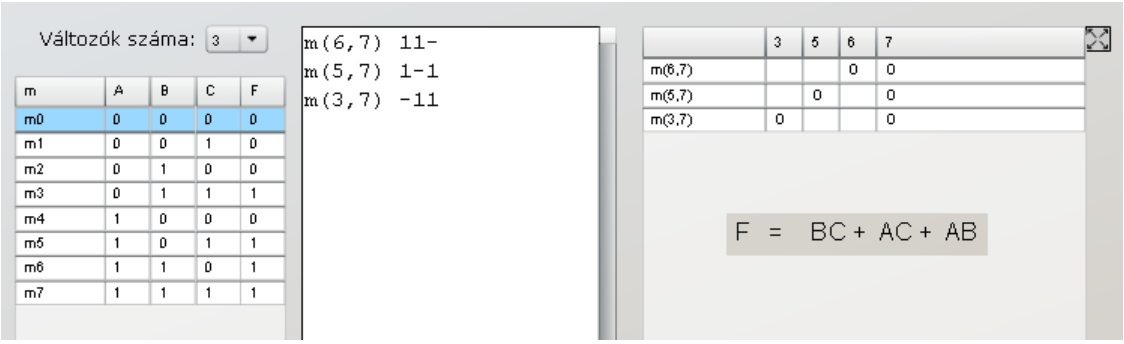
Egy háromtagú zsűri egyidejű szavazással dönt. A kijelző akkor gyulladjon fel, ha legalább ketten támogatták a javaslatot (5.7. példa). Az igazságtábla és a mintermek:



7.9. ábra: A kitöltött igazságtáblázat és a mintermek



7.10. ábra: Implikánsok egy változóval egyszerűsítve

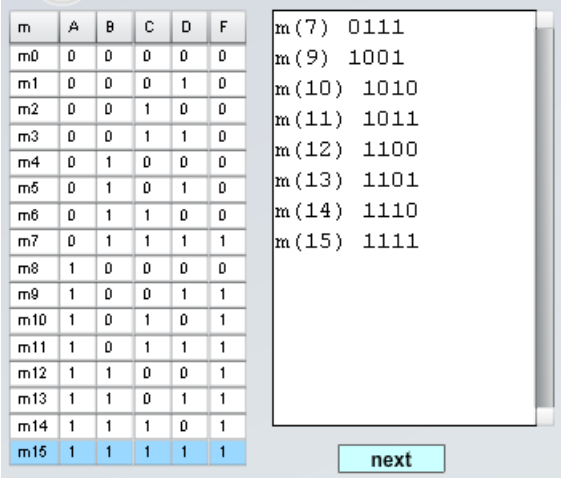


7.11. ábra: Az eredmény

Látható, hogy ugyanazt az eredményt kaptuk, mint a Karnaugh-egyszerűsítésnél.

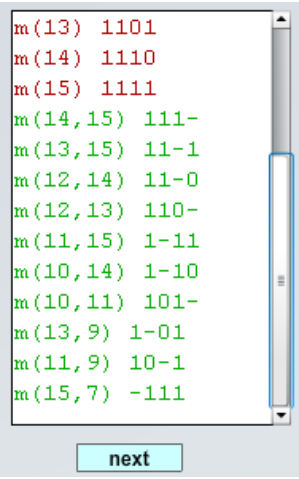
7.4. Példa

Egy négytagú zsűri egyidejű szavazással dönt. A zsűri elnöke 3, elnökhelyettese 2, a tagok pedig 1–1 szavazattal rendelkeznek. A kijelző akkor gyulladjon fel, ha legalább négy pontot érően szavaznak (5.8. példa). Az igazságtáblázat kitöltése után:

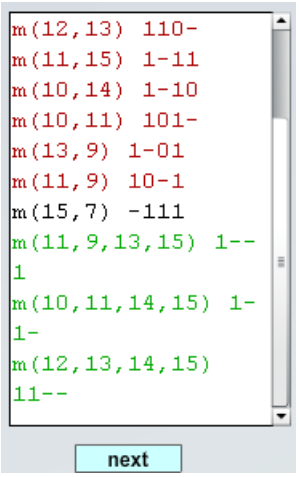


7.12. ábra: 1. lépés

Továbblépkedve a NEXT gombbal:

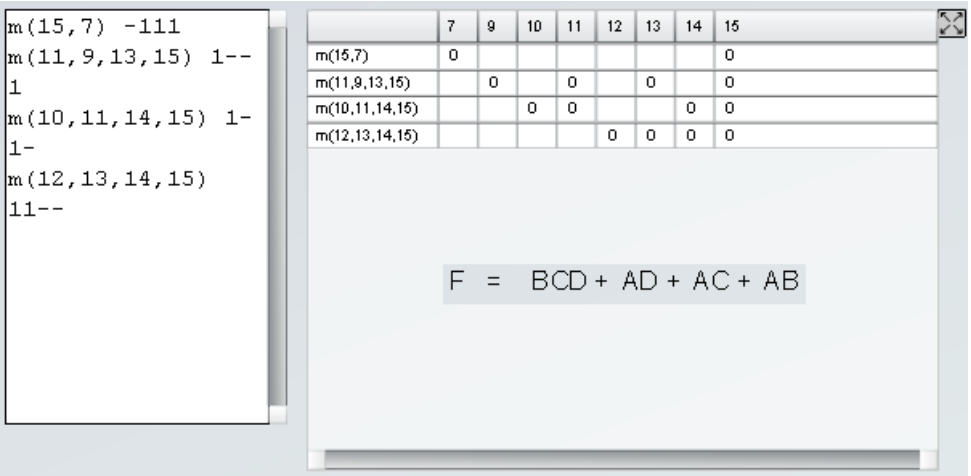


7.13. ábra: Implikánsok egy változóval egyszerűsítve



7.14. ábra: Implikánsok két változóval egyszerűsítve

Továbblépve a NEXT gombbal:



7.15. ábra: Prímimplikánsok McCluskey-táblázatban és eredmény

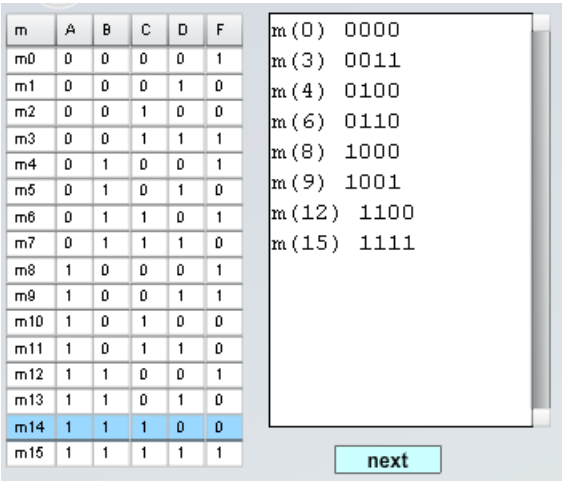
Tovább nem lehet egyszerűsíteni, ugyanazt az eredményt kaptuk, mint a Karnaugh-módszerrel.

7.5. Példa

Tervezzük meg azt a négy bemenetű és egy kimenetű logikai áramkört, amelyik az F kimeneten akkor ad 1-et, ha a bemenetre maradék nélkül 3-mal és/vagy 4-gyel osztható bináris kombináció érkezik. A bemenet A, B, C és D, legmagasabb helyérték A.

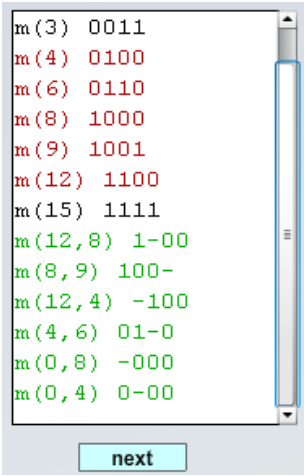
Megoldás:

Mivel a bináris szabály szerint a számokat 0 és 15 között kódoljuk, ezért a fenti feladatnak a 0, 3, 6, 9 12 és 15 minterm felel meg a 3-al való osztásnál, illetve 0, 4, 8, 12 a 4-el való osztásnál. Átfedés van a 0 és 12 mintermnél a két osztásnál. Az igazságtáblázatban a fenti mintermekhez rendeljünk egyeseket 7.16. ábra).

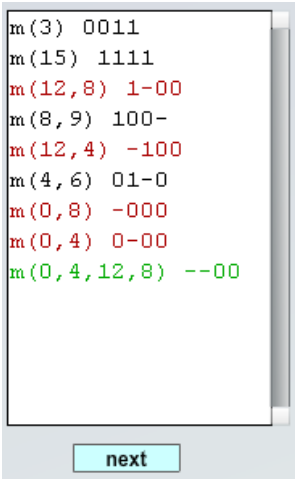


7.16. ábra: A feladat igazságtáblázata

Továbblépkedve a NEXT gombbal:

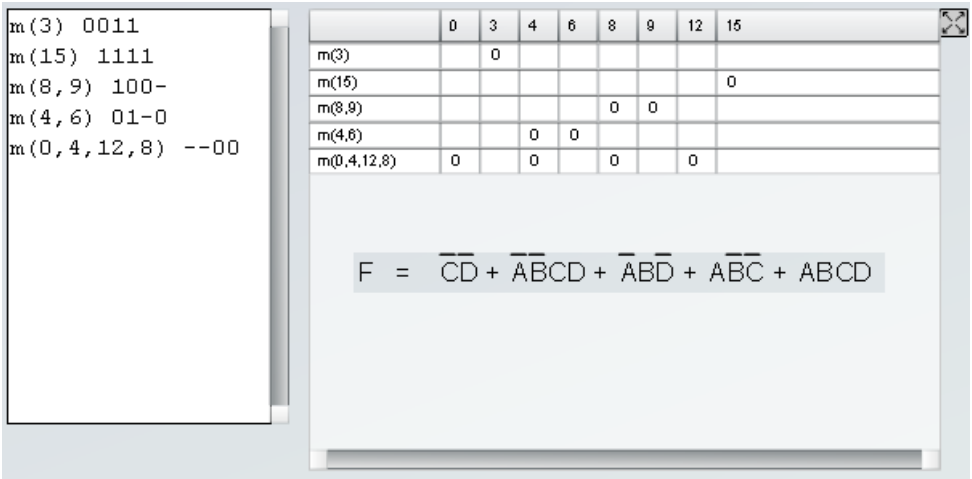


7.17. ábra: Implikánsok egy változóval egyszerűsítve (első lépés)



7.18. ábra: Implikánsok egy változóval egyszerűsítve (második lépés)

Továbblépve a NEXT gombbal:



7.19. ábra: Prímimplikánsok a McCluskey-táblázatban és az eredmény

Tovább nem lehet egyszerűsíteni.

7.2. Egyszerűsítés numerikus módszerrel

A numerikus módszer az előző fejezetben ismertetett Quine–McCluskey-módszer változata. Ha a logikai változók kombinációit bináris 8–4–2–1 súlyozású számoknak tekintjük, akkor ezek az értékek tulajdonképpen a logikai változók 0–1 értékeinek tízes számrendszerbeli megfelelői. Ezeken a számokon végzett aritmetikai műveletek segítségével egyszerűsíthetünk.

7.6. példa

Az $F(A,B,C,D,E) = \sum^5 (0,1,2,3,4,8,9,10,18,19,20,30,31) + \sum_x^5 (5,6,7,22,23,24)$

ötváltozós függvényt egyszerűsítse a Quine–McCluskey-féle numerikus módszer segítségével és rajzolja le a kapcsolást tetszőleges logikai kapukkal.

A Quine-féle numerikus egyszerűsítésnél az x-szel jelölt közömbös (határozatlan) kombinációkat is be kell vonni, töltsük ki a mintermeknek megfelelő bináris táblázatot:

	A	B	C	D	E
0.	0	0	0	0	0
1.	0	0	0	0	1
2.	0	0	0	1	0
3.	0	0	0	1	1
4.	0	0	1	0	0
5.	0	0	1	0	1
6.	0	0	1	1	0
7.	0	0	1	1	1
8.	0	1	0	0	0
9.	0	1	0	0	1
10.	0	1	0	1	0
18.	1	0	0	1	0
19.	1	0	0	1	1
20.	1	0	1	0	0
22.	1	0	1	1	0
23.	1	0	1	1	1
24.	1	1	0	0	0
30.	1	1	1	1	0
31.	1	1	1	1	1

7.20. ábra: Az összes 1-es és közömbös (x) mintermet tartalmazó igazságtáblázat

Mivel egyszerűsíteni mindig csak a szomszédos mintermek között lehet, ezért rendezzük át a táblázatot úgy, hogy egy–egy csoportban ugyanannyi számú egyest tartalmazó minterm legyen:

	A	B	C	D	E	
0.	0	0	0	0	0	√
1.	0	0	0	0	1	√
2.	0	0	0	1	0	√
4.	0	0	1	0	0	√
8.	0	1	0	0	0	√
3.	0	0	0	1	1	√
5.	0	0	1	0	1	√
6.	0	0	1	1	0	√
9.	0	1	0	0	1	√
10.	0	1	0	1	0	√
18.	1	0	0	1	0	√
20.	1	0	1	0	0	√
24.	1	1	0	0	0	√
7.	0	0	1	1	1	√
19.	1	0	0	1	1	√
22.	1	0	1	1	0	√
23.	1	0	1	1	1	√
30.	1	1	1	1	0	√
31.	1	1	1	1	1	√

7.21. ábra: A mintermek csoportosítása a bennük található 1-esek száma szerint

Erre az átcsoportosításra azért van szükség, mert a szomszédsági szabály szerint csak az 1 Hamming-távolságra levő mintermek egyszerűsíthetők, vagyis nincs értelme összehasonlítani eleve olyan kombinációkat egymással, amelyek úgysem egyszerűsíthetők. A példában látható, hogy 0 számú 1-es a 0-ás mintermben, 1 szamú minterm az 1, 2, 4 és 8-as mintermekben található, stb. Így összehasonlítást csak a szomszédos csoportok elemei között kell végezni, csökkentve ezzel az összehasonlítások számát. Mindig a következő csoport (tehát nagyobb számú 1-est tartalmazó) elemének indexéből vonjuk ki az előző csoport tagjának indexét. Eredményül kaphatunk bármilyen, tehát negatív illetve pozitív számot, de csak azokat a különbségeket vesszük figyelembe, amelyek értéke megfelel a bináris szabálynak, vagyis 1, 2, 4, 8 16 stb. lehet. Egyúttal azokat a mintermeket, amelyek belekerültek egy egyszerűbb implikánsba, megjelöljük, ezzel kizárjuk a további műveletekből. Látható, hogy az első lépés után mindegyik minterm összevonható volt valamelyik másik mintermmel.

	A	B	C	D	E	
0,1	0	0	0	0	-	√
0,2	0	0	0	-	0	√
0,4	0	0	-	0	0	√
0,8	0	-	0	0	0	√
1,3	0	0	0	-	1	√
1,5	0	0	-	0	1	√
1,9	0	-	0	0	1	√
2,3	0	0	0	1	-	√
2,6	0	0	-	1	0	√
2,10	0	-	0	1	0	√
2,18	-	0	0	1	0	√
4,5	0	0	1	0	-	√
4,6	0	0	1	-	0	√
4,20	-	0	1	0	0	√
8,9	0	1	0	0	-	√
8,10	0	1	0	-	0	√
8,24	-	1	0	0	0	←
3,7	0	0	-	1	1	√
3,19	-	0	0	1	1	√
5,7	0	0	1	-	1	√
6,7	0	0	1	1	-	√
6,22	-	0	1	1	0	√
18,19	1	0	0	1	-	√
18,22	1	0	-	1	0	√
20,22	1	0	1	-	0	√
7,23	-	0	1	1	1	√
17,23	1	0	1	1	1	√
19,23	1	0	1	1	1	√
22,23	1	0	1	1	0	√
22,30	1	-	1	1	0	√
23,31	1	-	1	1	1	√
30,30	1	1	1	1	-	√

7.22. ábra: Implikáns-táblázat az első lépés után

Az első egyszerűsítési lépés után a fenti implikáns-táblázatot kapjuk. A táblázat első oszlopa azt a két számot tartalmazza, amelyek jelölik az összevonható mintermeket. A következő A, B, C, D és E oszlopok a változók értékeit tartalmazzák, 1-et vagy 0-át, illetve

ha valamelyik változó kiesik a „–” jelet. Látható, hogy kiinduláskor 6 csoportba soroltuk a mintermeket, az első lépés után 5 csoport keletkezett.

Tovább egyszerűsíthetünk a szomszédos csoportok között, kivonjuk a következő csoport indexeit az előző csoport indexeiből, de csak ott, ahol azonos helyen (változónál) „–” jel van és az eredmény a bináris szabálynak felel meg (1, 2, 4, 8 stb.). Például 1,3 – 0,1 nem vonható össze, mert nem azonos helyen van a „–” de 0–1 és 2–3 összevonható, azonos helyen „–” és a kivonás eredménye bináris szabálynak megfelel (2). Mindig megjelöljük az egyszerűsíthető implikánsokat (esetünkben a √ jellel), ugyanakkor néhány implikáns nem volt összevonható, itt ezek megjelölésére a „←” (nyilat) használjuk.

	A	B	C	D	E	
0,1,2,3	0	0	0	-	-	√
0,1,4,5	0	0	-	0	-	√
0,1,8,9	0	-	0	0	-	←
0,2,4,6	0	0	-	-	0	√
0,2,8,10	0	-	0	-	0	←
1,3,5,7	0	0	-	-	1	√
2,3,18,19	-	0	0	1	-	√
2,3,6,7	0	0	-	1	-	√
2,6,18,22	-	0	-	1	0	√
4,5,6,7	0	0	1	-	-	√
4,6,20.22	-	0	1	-	0	←
3,7,19,23	-	0	-	1	1	√
6,7,22,23	-	0	1	1	-	√
18,19,22,23	1	0	-	1	-	√
22,23,30,31	1	-	1	1	-	←

7.23. ábra: Implikáns-táblázat a második lépés után

A következő lépés:

	A	B	C	D	E	
0,1,2,3,4,5,6,7	0	0	-	-	-	←
2,3,6,7,18,19,22,23	-	0	-	1	-	←

7.24. ábra: Implikáns-táblázat a második lépés után

Miután itt már nincs lehetőség összevonni implikánsokat, az előző összes táblázat összes „←” jellel (nyíllal) jelölt implikánsait és az eredeti mintermeket bevisszük a McCluskey-táblázatba:

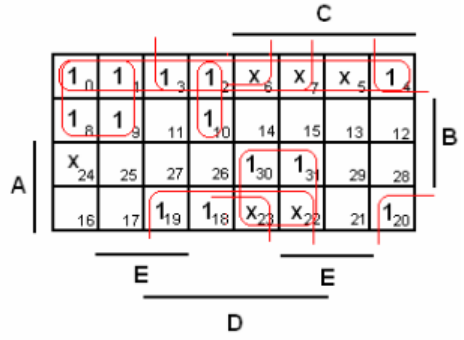
						x	x	x							x	x	x		
	0	1	2	3	4	5	6	7	8	9	10	18	19	20	22	23	24	30	31
$B\cdot\overline{C}\cdot\overline{D}\cdot\overline{E}$									□								□		
$\overline{A}\cdot\overline{C}\cdot\overline{D}$	■	■							■	■									
$\overline{B}\cdot C\cdot\overline{E}$					■		■							■	■				
$A\cdot C\cdot D$															■	■		■	■
$\overline{A}\cdot\overline{B}$	□	□	□	□	□	□	□	□											
$\overline{B}\cdot D$			■	■			■	■				■	■		■	■			

7.25. ábra: A McCluskey-tábla

A táblázatban szereplő implikánsok között lehet, hogy van redundáns, ezekez meg kell találni és kiiktatni az egyszerűsített alakból. Mivel itt közömbös kombinációk is vannak, ezért az eljárás a következő:

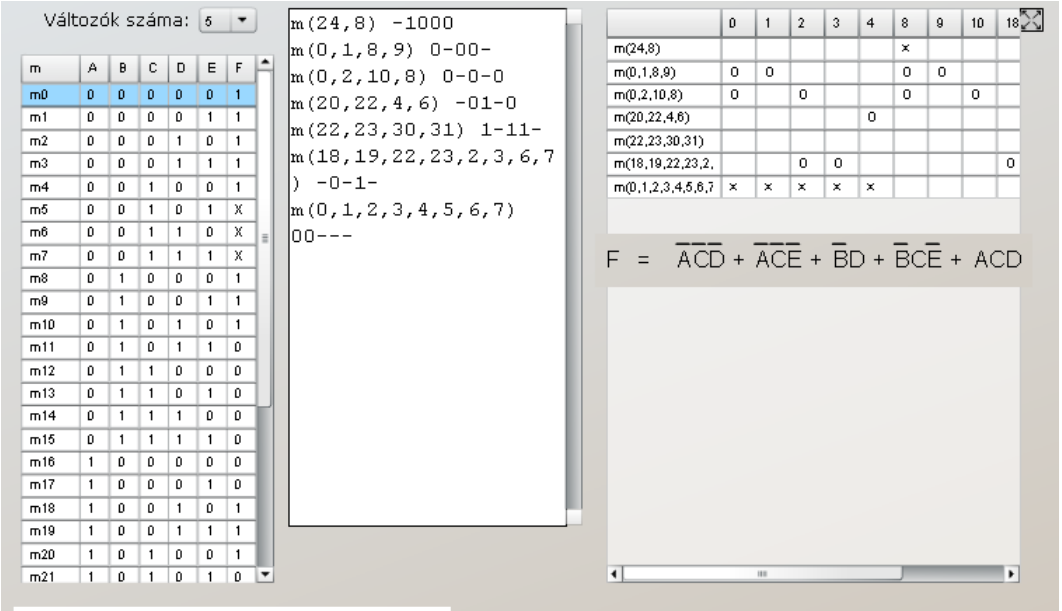
- összekapcsoljuk egy jellel (esetünkben egy üres négyzet) az implikánst azokkal a mintermekkel, amelyek egyszerűsíthetők voltak,
- megkeressük azokat az oszlopokat (mintermeket) ahol csak egy négyzet van, ezeket ha 1-es mintermből származnak jelöljük (teli négyzet), ha közömbös minterm, nem választjuk ki,
- az előző pontban jelölt mintermekkkel azonos sorban levő négyzeteket is jelöljük,
- ahol több üres négyzet van egymás fölött, azt a négyzetet választjuk, amelyik vízszintesen több négyzetet tartalmaz (2, 4, 8 stb)
- cél minden oszlopot lefedni.

A mi esetünkben a 8–24 implikáns redundáns, tehát nem kerül bele a legegyszerűbb alakba, mert a 8-as mintermet más implikáns tartalmazza, a 24-es minterm pedig közömbös. Az egyszerűsített függvény:
$F = \overline{A}\cdot\overline{C}\cdot D\cdot\overline{E} + B\cdot\overline{C}\cdot\overline{D}\cdot\overline{E} + \overline{A}\cdot\overline{C}\cdot\overline{D} + \overline{B}\cdot C\cdot\overline{E} + A\cdot C\cdot D + \overline{A}\cdot\overline{B} + \overline{B}\cdot D$, ahonnan a $B\cdot\overline{C}\cdot\overline{D}\cdot\overline{E}$ kiesik. Ellenőrizzük le a példát a Karnaugh-módszerrel:



7.26. ábra: A feladat Karnaugh-táblája

Látható, hogy ugyanazt az eredményt kaptuk.
Ellenőrizzük le a példát a Quine–McCluskey-módszerrel:



7.27. ábra: A feladat Quine–McCluskey-módszerrel

Látható, hogy ugyanazt az eredményt kaptuk.

7.7. példa

Egyszerűsítse a következő logikai függvényt Quine-módszerrel, használja a McLusky-módszert a felesleges implikánsok kiszűrésére:

$$F(A,B,C,D) = \sum(4,5,6,7,10,14) + \sum_x(2,3,13,15).$$

Határozzuk meg a mintermekben található 1-esek számát, majd csoportosítsuk a mintermeket az 1-esek száma szerint:

	A	B	C	D	'1'
2	0	0	1	0	1
3	0	0	1	1	2
4	0	1	0	0	1
5	0	1	0	1	2
6	0	1	1	0	2
7	0	1	1	1	3
10	1	0	1	0	2
13	1	1	0	1	3
14	1	1	1	0	4
15	1	1	1	1	5

7.28. ábra: A mintermekben található 1-esek száma

felhasználva	minterm	'1'
√	2	1
√	4	
√	3	2
√	5	
√	6	
√	10	
√	7	3
√	13	
√	14	
√	15	4

7.29. ábra: A mintermek csoportjai a bennük található 1-esek száma alapján

Végezzük el az egyszerűsítés első lépését, vagyis keressük meg az összevonható mintermeket:

felhasználva	összevonás	különbség
√	2,3	(1)
√	2,6	(4)
√	2,10	(8)
√	4,5	(1)
√	4,6	(2)
√	3,7	(4)
√	5,7	(2)
√	5,13	(8)
√	6,7	(1)
√	6,14	(8)
√	10,14	(4)
√	7,15	(8)
√	13,15	(2)
√	14,15	(1)

7.30. ábra: Implikánsok táblázata az első lépés után

Vonjuk össze az egyszerűsíthető implikánsokat, az azonos értékeket iktassuk ki a táblázatból, második lépés:

felhasználva	összevonás	különbség
	2,3,6,7	(1,4)
	~~2,6,3,7~~	~~(4,1)~~
	2,6,10,14	(4,8)
	~~2,10,6,14~~	~~(8,4)~~
	4,5,6,7	(1,2)
	~~4,6,5,7~~	~~(2,1)~~
	5,7,13,15	(2,8)
	~~5,13,7,15~~	~~(8,2)~~
	6,7,14,15	(1,8)
	~~6,14,7,15~~	~~(8,1)~~

7.31. ábra: Implikánsok táblázata a második lépés után

Látható, hogy, tovább nem egyszerűsíthető a függvény.
A McCluskey-tábla a redundáns implikánsok kiszűrésére:

	2_x	3_x	4	5	6	7	10	13_x	14	15_x
2,3,6,7	x	x			x	x				
2,6,10,14	⊗				⊗		⊗		⊗	
4,5,6,7			⊗	⊗	⊗	⊗				
5,7,13,15				x		x		x		x
6,7,14,15					x	x			x	x

7.32. ábra: A McCluskey-tábla a redundáns implikánsok kiszűrésére

Itt x-szel jelöltük a kapcsolatot a minterm és az implikáns között, a megmaradó implikánsok jelölésére pedig bekarikáztuk az x-et. Mivel itt az implikánsok jelölése decimális számmal történt, ezért ebből meg kell találni az implikáns változókkal leírt alakját. Ezt végezhetjük a következő táblázattal, ahol eltávolítjuk azokat a változókat, amelyek értéke változott:

	~~A~~	~~B~~	C	D
2	0	0	1	0
6	0	1	1	0
10	1	0	1	0
14	1	1	1	0

7.33. ábra: Az első implikáns felírása változókkal.

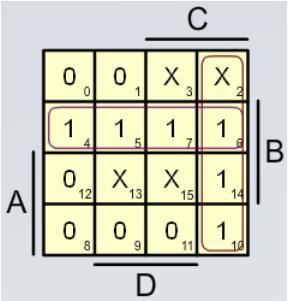
	A	B	~~C~~	~~D~~
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1

7.34. ábra: A második implikáns felírása változókkal.

Az egyszerűsített függvény:

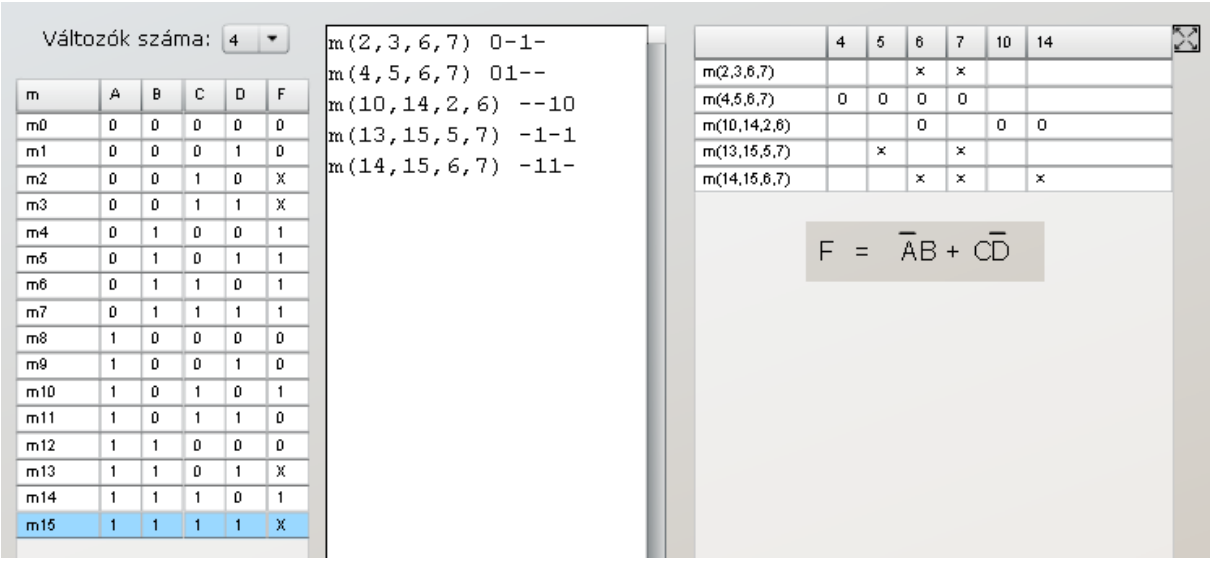
$$F = C \cdot \overline{D} + \overline{A} \cdot B$$

Ellenőrzés Karnaugh-táblázattal:



7.35. ábra: A feladat Karnaugh-táblája

Látható, hogy ugyanazt az eredményt kaptuk.
Ellenőrzés Quine–McCluskey-módszerrel:



7.36. ábra: A feladat Quine–McCluskey-módszerrel

Látható, hogy ugyanazt az eredményt kaptuk.

8. Többkimenetű kombinációs hálózatok

8.1. példa

Tervezzük meg a 4221 BCD kódból 8421 BCD kódba történő kódátalakítás áramkörét.

Megoldás:

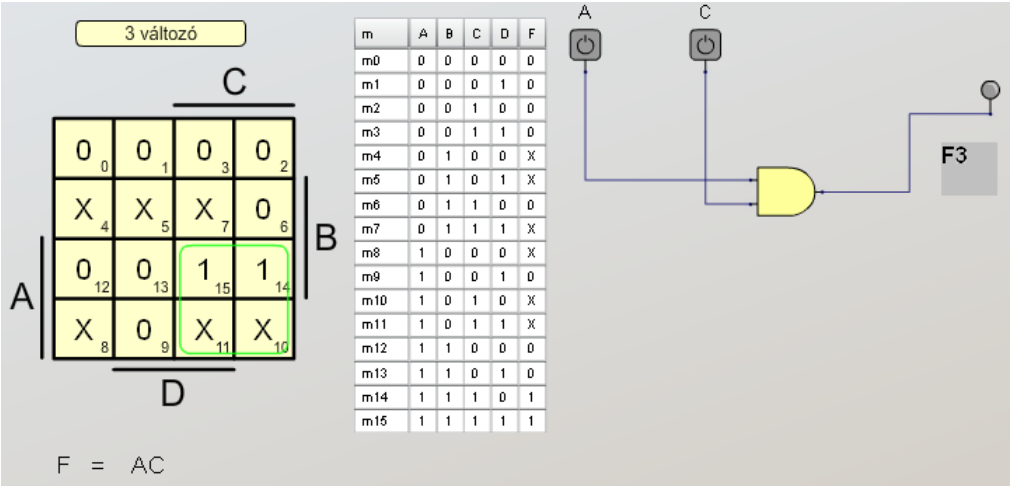
Első lépésként meg kell adni 0 – 9 számok BCD kódját mind 4221 mind 8421 súlyozással:

	BEMENET				KIMENET			
súly	4	2	2	1	8	4	2	1
változó	A	B	C	D	F3	F2	F1	F0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	1	0	0	1	0	0
5	1	0	0	1	0	1	0	1
6	1	1	0	0	0	1	1	0
7	1	1	0	1	0	1	1	1
8	1	1	1	0	1	0	0	0
9	1	1	1	1	1	0	0	1

8.1. ábra: 0 – 9 számok BCD kódja 4221 és 8421 súlyozással

Következő lépés a kimenet (F3, F2, F1 és F0) összekapcsolása a bemenetekkel (A, B, C és D). Mivel mindkét BCD kód négybites, ezért 16 kombinációnk lesz, de ebből a tízes számrendszerben csak 10 különböző kombinációra (számjegyre) van szükség, ezért a nem létező 6 kombinációt közömbös értékekkel (x) vesszük figyelembe. A Karnaugh-módszer a példa egyszerűsége miatt könnyen használható, így négy táblánk lesz (F3, F2, F1 és F0 kimenetekre). Vigyázzunk arra, hogy a BCD 4221 táblázat mintermindexei adják a Karnaugh-tábla megfelelő mintermjeit.

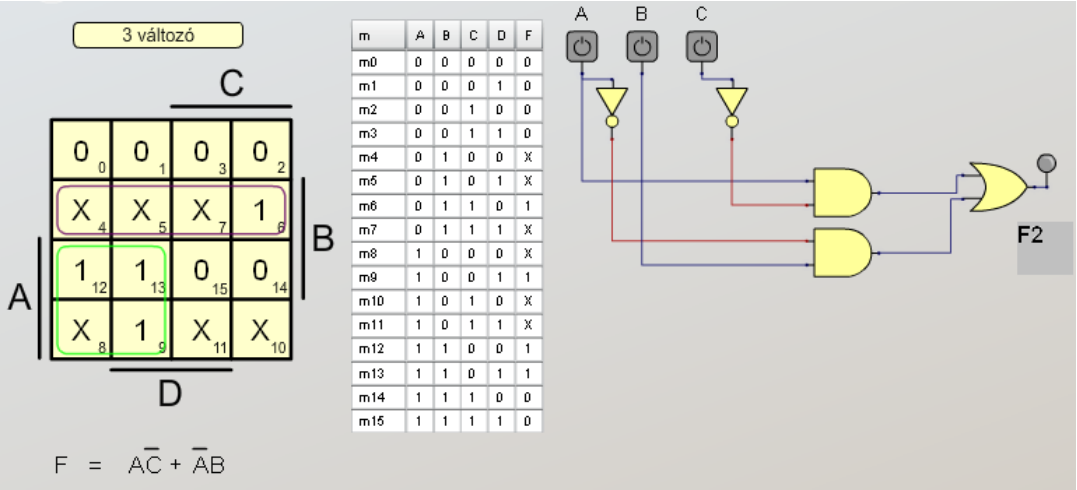
Ezek után F3:



m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	0
m3	0	0	1	1	0
m4	0	1	0	0	X
m5	0	1	0	1	X
m6	0	1	1	0	0
m7	0	1	1	1	X
m8	1	0	0	0	X
m9	1	0	0	1	0
m10	1	0	1	0	X
m11	1	0	1	1	X
m12	1	1	0	0	0
m13	1	1	0	1	0
m14	1	1	1	0	1
m15	1	1	1	1	1

8.2. ábra: F3 meghatározása Karnaugh-táblával

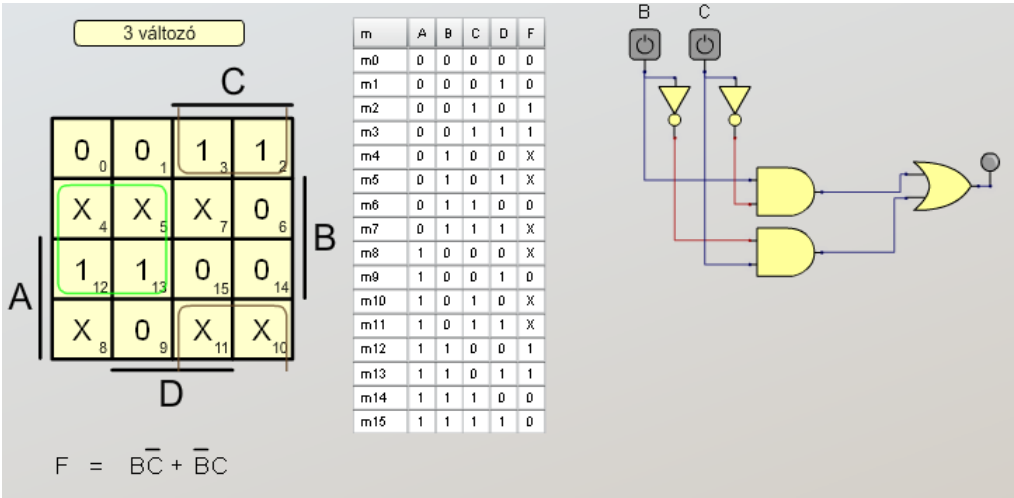
F2 meghatározása:



m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	0
m3	0	0	1	1	0
m4	0	1	0	0	X
m5	0	1	0	1	X
m6	0	1	1	0	1
m7	0	1	1	1	X
m8	1	0	0	0	X
m9	1	0	0	1	1
m10	1	0	1	0	X
m11	1	0	1	1	X
m12	1	1	0	0	1
m13	1	1	0	1	1
m14	1	1	1	0	0
m15	1	1	1	1	0

8.3. ábra: F2 meghatározása Karnaugh-táblával

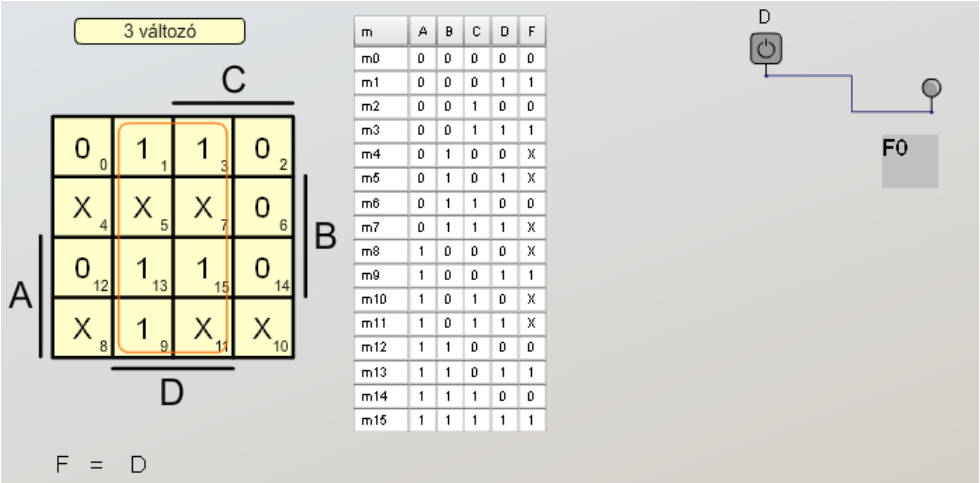
F1 meghatározása:



m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	1
m3	0	0	1	1	1
m4	0	1	0	0	X
m5	0	1	0	1	X
m6	0	1	1	0	0
m7	0	1	1	1	X
m8	1	0	0	0	X
m9	1	0	0	1	0
m10	1	0	1	0	X
m11	1	0	1	1	X
m12	1	1	0	0	1
m13	1	1	0	1	1
m14	1	1	1	0	0
m15	1	1	1	1	0

8.4. ábra: F1 meghatározása Karnaugh-táblával

F0 meghatározása:



m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	1
m2	0	0	1	0	0
m3	0	0	1	1	1
m4	0	1	0	0	X
m5	0	1	0	1	X
m6	0	1	1	0	0
m7	0	1	1	1	X
m8	1	0	0	0	X
m9	1	0	0	1	1
m10	1	0	1	0	X
m11	1	0	1	1	X
m12	1	1	0	0	0
m13	1	1	0	1	1
m14	1	1	1	0	0
m15	1	1	1	1	1

8.5. ábra: F0 meghatározása Karnaugh-táblával

8.2. példa

Tervezzük meg a 8421 BCD kódból 4221 BCD kódba történő kódátalakítás áramkörét.

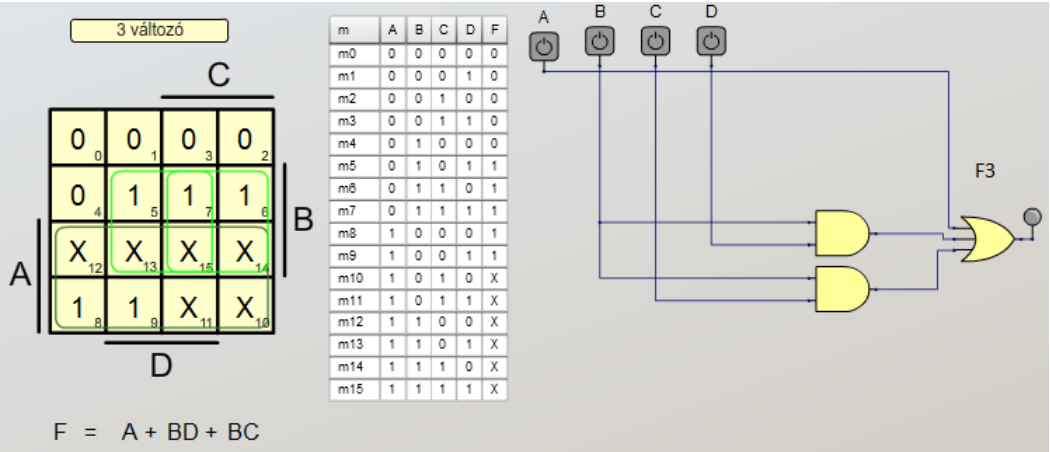
Megoldás:

Első lépésként meg kell adni 0 – 9 számok BCD kódját mind 8421 mind 4221 súlyozással:

	BEMENET				KIMENET			
súly	8	4	2	1	4	2	2	1
változó	A	B	C	D	F3	F2	F1	F0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	1	0
5	0	1	0	1	1	0	0	1
6	0	1	1	0	1	1	0	0
7	0	1	1	1	1	1	0	1
8	1	0	0	0	1	1	1	0
9	1	0	0	1	1	1	1	1

8.6. ábra: 0 – 9 számok BCD kódja 8421 és 4221 súlyozással

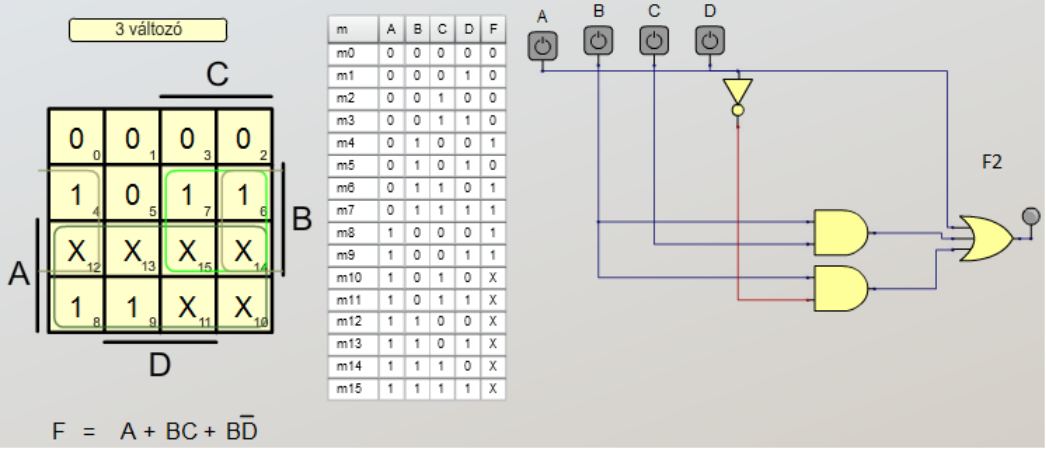
Következő lépés a kimenet (F3, F2, F1 és F0) összekapcsolása a bemenetekkel (A, B, C és D). Mivel mindkét BCD kód négybites, ezért 16 kombinációnk lesz, de ebből a tízes számrendszerben csak 10 különböző kombinációra (számjegyre) van szükség, ezért a nem létező 6 kombinációt közömbös értékekkel (x) vesszük figyelembe. A Karnaugh-módszer a példa egyszerűsége miatt könnyen használható, így négy táblánk lesz (F3, F2, F1 és F0 kimenetekre). Ezek után F3:



m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	0
m3	0	0	1	1	0
m4	0	1	0	0	0
m5	0	1	0	1	1
m6	0	1	1	0	1
m7	0	1	1	1	1
m8	1	0	0	0	1
m9	1	0	0	1	1
m10	1	0	1	0	X
m11	1	0	1	1	X
m12	1	1	0	0	X
m13	1	1	0	1	X
m14	1	1	1	0	X
m15	1	1	1	1	X

8.7. ábra: F3 meghatározása Karnaugh-táblával

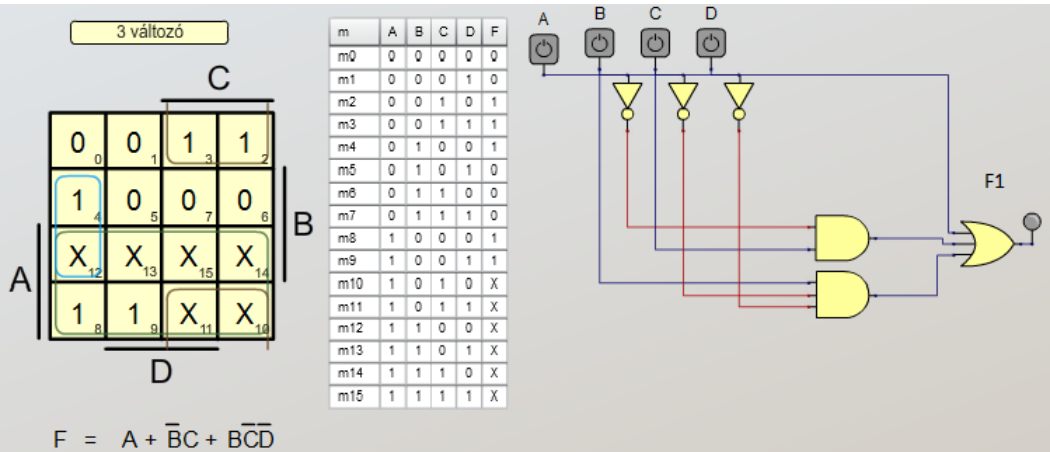
F2 meghatározása:



m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	0
m3	0	0	1	1	0
m4	0	1	0	0	1
m5	0	1	0	1	0
m6	0	1	1	0	1
m7	0	1	1	1	1
m8	1	0	0	0	1
m9	1	0	0	1	1
m10	1	0	1	0	X
m11	1	0	1	1	X
m12	1	1	0	0	X
m13	1	1	0	1	X
m14	1	1	1	0	X
m15	1	1	1	1	X

8.8. ábra: F2 meghatározása Karnaugh-táblával

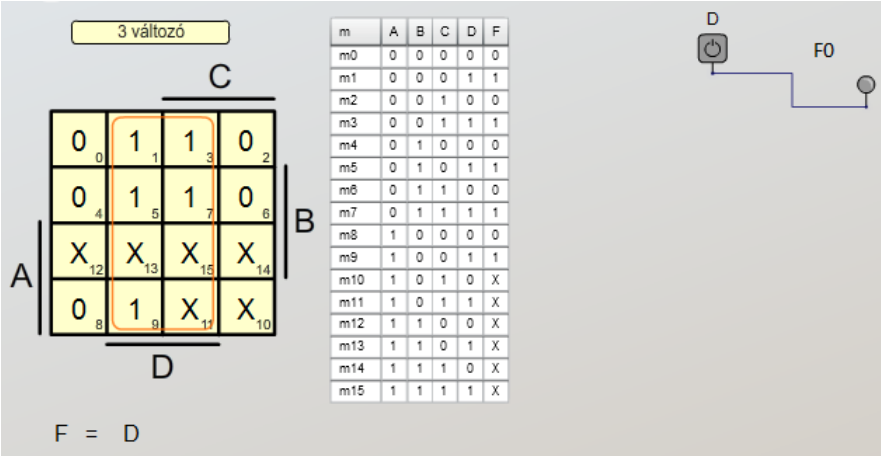
F1 meghatározása:



m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	1
m3	0	0	1	1	1
m4	0	1	0	0	1
m5	0	1	0	1	0
m6	0	1	1	0	0
m7	0	1	1	1	0
m8	1	0	0	0	1
m9	1	0	0	1	1
m10	1	0	1	0	X
m11	1	0	1	1	X
m12	1	1	0	0	X
m13	1	1	0	1	X
m14	1	1	1	0	X
m15	1	1	1	1	X

8.9. ábra: F1 meghatározása Karnaugh-táblával

F0 meghatározása:



m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	1
m2	0	0	1	0	0
m3	0	0	1	1	1
m4	0	1	0	0	0
m5	0	1	0	1	1
m6	0	1	1	0	0
m7	0	1	1	1	1
m8	1	0	0	0	0
m9	1	0	0	1	1
m10	1	0	1	0	X
m11	1	0	1	1	X
m12	1	1	0	0	X
m13	1	1	0	1	X
m14	1	1	1	0	X
m15	1	1	1	1	X

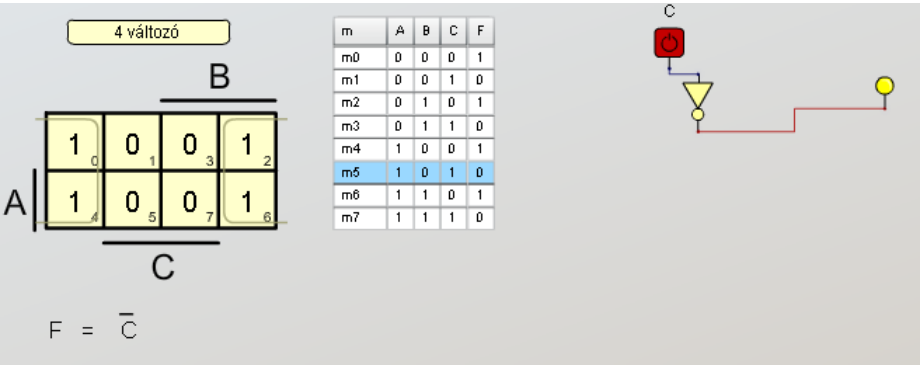
8.10. ábra: F0 meghatározása Karnaugh-táblával

8.3. Példa

Tervezzen meg egy olyan logikai kapcsolást, amelyik a bemenetre érkező 0–7 közötti számot a kimenetén eggyel megnövelve adja ki, 0 → 1, 1 → 2 stb. és ha a bejövő érték 7, akkor a kimenet 0 legyen.

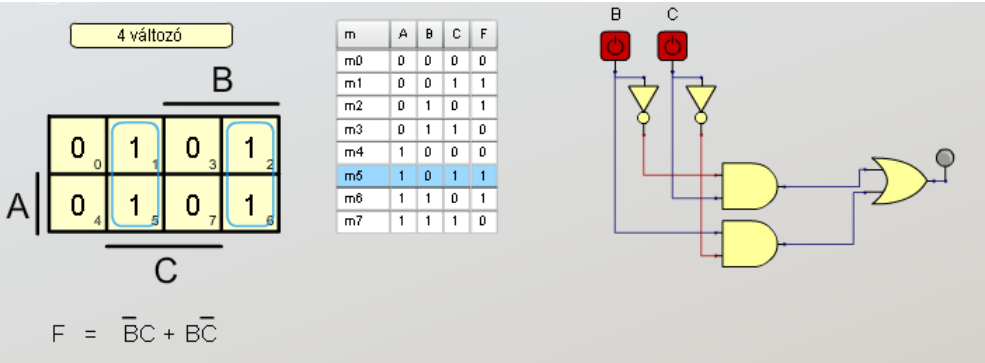
Megoldás:

Mivel a 0 – 7 számok 3 biten kódolhatóak, ezért egy háromváltozós igazságtáblázatot kell kitölteni, de látható, hogy most egy 3 bemenetű és 3 kimenetű áramkört kapunk:



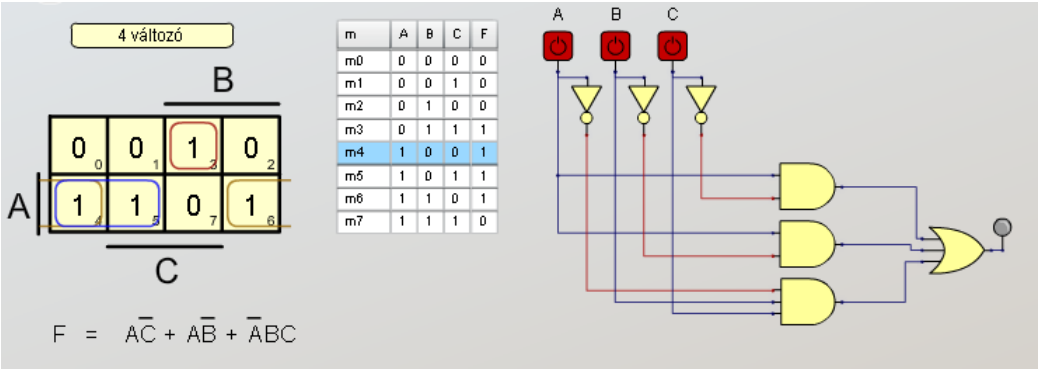
m	A	B	C	F
m0	0	0	0	1
m1	0	0	1	0
m2	0	1	0	1
m3	0	1	1	0
m4	1	0	0	1
m5	1	0	1	0
m6	1	1	0	1
m7	1	1	1	0

8.11. ábra: F_0, a legkisebb helyiérték



m	A	B	C	F
m0	0	0	0	0
m1	0	0	1	1
m2	0	1	0	1
m3	0	1	1	0
m4	1	0	0	0
m5	1	0	1	1
m6	1	1	0	1
m7	1	1	1	0

8.12. ábra: F_1 , helyiérték



8.13. ábra: F_2 , a legnagyobb helyiérték

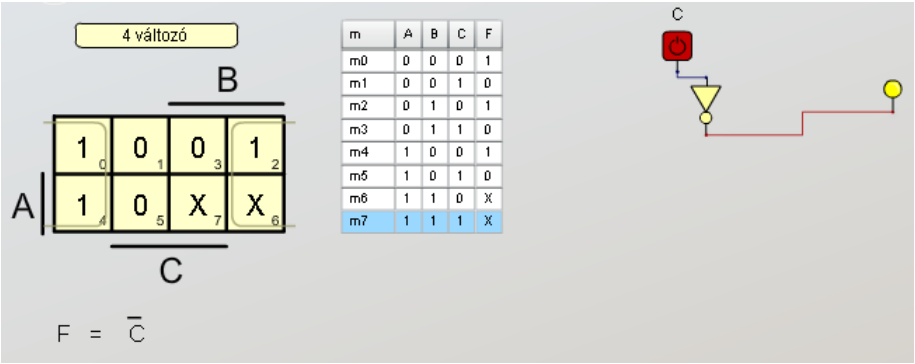
Mivel a B és C változóknál használt invertereken kívül nincs közös rész, ezért a kapcsolási rajzon nem tudunk semmit sem összevonni.

8.4. Példa

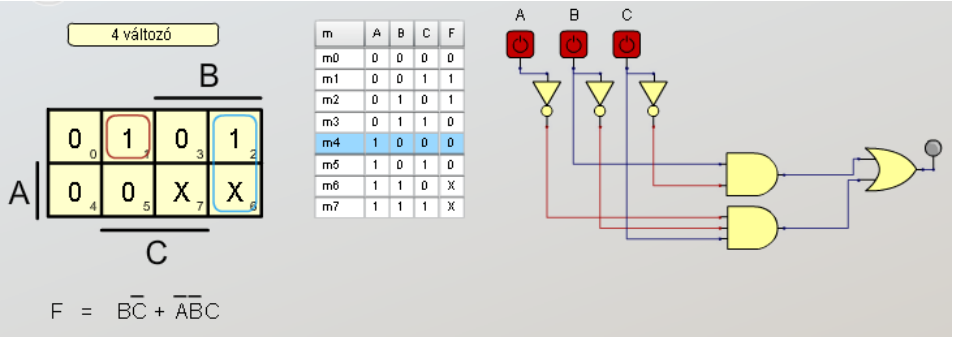
Tervezzen meg egy olyan logikai kapcsolást, amelyik a bemenetre érkező 0 – 5 közötti számot a kimenetén eggyel megnövelve adja ki, 0 → 1, 1 → 2 stb. és ha a bejövő érték 5, akkor a kimenet 0 legyen.

Megoldás:

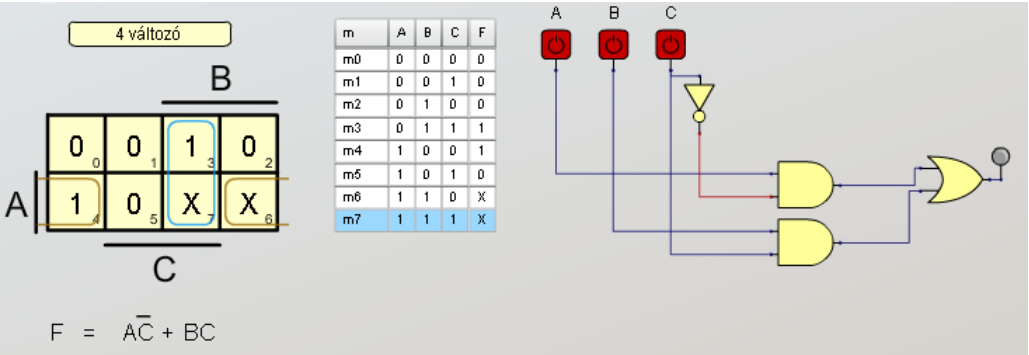
Mivel a 0 – 5 számok 3 biten kódolhatóak, ezért egy háromváltozós igazságtáblázatot kell kitölteni, de látható, hogy most egy 3 bemenetű és 3 kimenetű áramkört kapunk, ugyanakkor a 6 és 7 kombináció nem fordulhat elő:



8.14. ábra: F_0 , a legkisebb helyérték



8.15. ábra: F_1 , helyérték



8.16. ábra: F_2 , a legnagyobb helyérték

8.5. Példa

Az áramkör bemenetére BCD szám érkezik, 0–9 tartományban. A kimeneten egy hétszegmenses kijelzőn jelenik meg a szám. Tervezzük meg a BCD–hétszegmenses dekódolót.

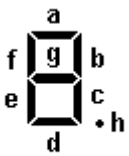
Megoldás:

A tízes számrendszerben összesen 10 karakterrel tudjuk kódolni a számokat, 0–9 tartományban. Ezeket a számokat kettes számrendszerben legalább 4 biten lehet kódolni, mert $2^4 = 16 > 10$. Így 6 közömbös kombinációt találunk a táblázatban:

Decimális	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	x	x	x	x
11	x	x	x	x
12	x	x	x	x
13	x	x	x	x
14	x	x	x	x
15	x	x	x	x

8.17. ábra: A számok BCD kódolása

A hétszegmenses kijelző 7 + 1 bemenettel rendelkezik, a + 1 bemenet a tizedes pontot vezérli. A lábkiosztás a következő:



8.18. ábra: A hétszegmenses kijelző lábkiosztása

Miután ismerjük a számok BCD kódolását és a hétszegmenses kijelző lábkiosztatás felállítjuk a dekódoló igazságtáblázatát:

decimális szám	BCD szám	Hétszegmenses kijelző
	ABCD	a bcdef g
0	0 0 0 0	1111110
1	0 0 0 1	0110000
2	0 0 1 0	1101101
3	0 0 1 1	1111001
4	0 1 0 0	0110011
5	0 1 0 1	1011011
6	0 1 1 0	1011111
7	0 1 1 1	1110000
8	1 0 0 0	1111111
9	1 0 0 1	1111011
10	1 0 1 0	xxxxxxx
11	1 0 1 1	xxxxxxx
12	1 1 0 0	xxxxxxx
13	1 1 0 1	xxxxxxx
14	1 1 1 0	xxxxxxx
14	1 1 1 1	xxxxxxx

8.19. ábra: A BCD–hétszegmenses dekódoló igazságtáblázata

A következő lépésben mind a hét pálcikára kitöltjük a Karnaugh-táblát és kiírjuk az egyszerűsített függvényeket:

$$a = C + A + B\,D + \bar{B}\,\bar{D}$$
$$b = \bar{B} + \bar{C}\,D + C\,D$$
$$c = \bar{C} + D + B$$
$$d = A + \bar{B}\,\bar{D} + \bar{B}\,C + C\,\bar{D} + B\,\bar{C}\,D$$
$$e = \bar{B}\,\bar{D} + C\,\bar{D}$$
$$f = A + B\,\bar{C} + B\,\bar{D} + \bar{C} + \bar{D}$$
$$g = A + B\,\bar{C} + B\,\bar{D} + \bar{B}\,C$$

8.6. Példa

Az áramkör bemenetére BCD szám érkezik, 0–9 tartományban. A kimeneten egy hétszegmenses kijelzőn jelenik meg a szám. Tervezzük meg a BCD–hétszegmenses dekódolót, úgy, hogy a BCD szám mellett egy T (tesztelés) bemeneten megjelenő logikai egyes az összes pálcikát gyújtsa meg, valamint a bemenetre érkező nem–BCD szám esetén egy E (error – hiba) kimeneti LED világítson.

Megoldás:

A BCD számok kiírását az előző példa szerint oldhatjuk meg. A T bemeneten érkező logikai egyes jelet egyszerűen minden pálcika vezérlését biztosító VAGY kapu plusz egy bemenetével oldhatjuk meg:

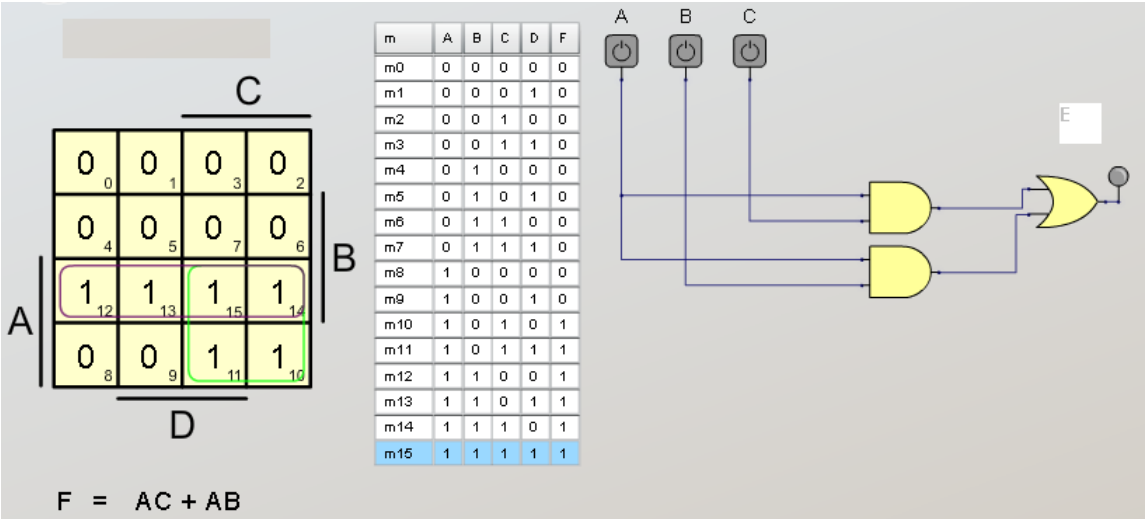
$$a = C + A + B\,D + \bar{B}\,\bar{D} + T$$
$$b = \bar{B} + \bar{C}\,D + C\,D + T$$
$$c = \bar{C} + D + B + T$$

$d = A + \bar{B}\,\bar{D} + \bar{B}\,C + C\,\bar{D} + B\,\bar{C}\,D + T$
$e = \bar{B}\,\bar{D} + C\,\bar{D} + T$
$f = A + B\,\bar{C} + B\,\bar{D} + \bar{C} + \bar{D} + T$
$g = A + B\,\bar{C} + B\,\bar{D} + \bar{B}\,C + T$

A nem–BCD számok kijelzését a következő módon oldhatjuk meg:

$$E(A,B,C,D) = \sum^{4} (10,11,12,13,14,15).$$

A Karnaugh-tábla:



m	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	0
m3	0	0	1	1	0
m4	0	1	0	0	0
m5	0	1	0	1	0
m6	0	1	1	0	0
m7	0	1	1	1	0
m8	1	0	0	0	0
m9	1	0	0	1	0
m10	1	0	1	0	1
m11	1	0	1	1	1
m12	1	1	0	0	1
m13	1	1	0	1	1
m14	1	1	1	0	1
m15	1	1	1	1	1

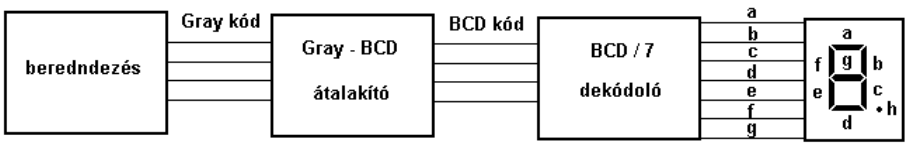
8.20. ábra: A példa Karnaugh-táblája és a kapcsolás

8.7. Példa

Egy berendezés Gray-kódban küld mérési eredményeket (0 és 9 között). A Gray-kód használatát az indokolja, hogy a tízes számok közötti átmenet binárisan úgy történik, hogy egyszerre csak egy bit vált, ellentétben a természetes bináris kódolással, ahol például az 1 → 2 átmenetnél egyidejűleg két váltás is van, A_0 helyen 1 → 0 és A_1 helyen 0 → 1, mivel műszakilag lehetetlen azonos időpontban ezt a váltást létrehozni, egy rövid ideig hamis adat jelenhet meg (0 illetve 3). Hétszegmenses kijelzőn jelenítsük meg az eredményt.

Megoldás:

A hétszegmenses kijelző számára egy BCD-hétszegmenses dekódoló biztosítja a megfelelő kódot, mivel a berendezésből Gray-kód alakban jön a szám egy Gray–BCD dekódolót kell tervezni a berendezés és a BCD-hétszegmenses dekódoló közé.



8.21. ábra: A megoldás blokksémája

A Gray – BCD értékek közötti kapcsolat:

	a számok felcserélése	A kód neve							
		Gray kód				Bináris kód			
		A	B	C	D	X	Y	Z	W
0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	1
2	3	0	0	1	1	0	0	1	0
3	2	0	0	1	0	0	0	1	1
4	6	0	1	1	0	0	1	0	0
5	7	0	1	1	1	0	1	0	1
6	5	0	1	0	1	0	1	1	0
7	4	0	1	0	0	0	1	1	1
8	12	1	1	0	0	1	0	0	0
9	13	1	1	0	1	1	0	0	1
10	15	1	1	1	1	x	x	x	x
11	14	1	1	1	0	x	x	x	x
12	10	1	0	1	0	x	x	x	x
13	11	1	0	1	1	x	x	x	x
14	9	1	0	0	1	x	x	x	x
15	8	1	0	0	0	x	x	x	x

8.22. ábra: A Gray – BCD kód összefüggése

A következő lépésben mind a négy BCD bemenetre (X, Y, Z és U) kitöltjük a Karnaugh-táblát és kiírjuk az egyszerűsített függvényeket. Itt azonban van egy probléma, általában a mintermeket indexeik szerinti sorrendbe rendezzük, a Gray–kódnál viszont a sorrend nem egyezik meg a természetes bináris számok kódolásával. Ezt a problémát úgy hidaljuk át, hogy „megcseréljük” a mintermeket, a Gray-kódnak megfelelően, a táblázatban ez a 2. oszlopban, a „számok felcserélése” helyen található. Ezek után már az új indexek szerint minimalizálunk. A Gray–BCD dekódoló függvényei:

$$X = A$$
$$Y = \bar{A}\,B$$
$$Z = B\,\bar{C} + \bar{A}\,B\,\bar{C}$$
$$U = A\,D + \bar{B}\,\bar{C}\,D + \bar{B}\,C\,\bar{D} + B\,C\,D + \bar{A}\,B\,\bar{C}\,\bar{D}$$

8.8. Példa

Egy logikai áramkörre két 1 bites szám érkezik, A és B. Tervezzen meg egy olyan áramkört, amelyik összeadja a két számot.

Megoldás:

A feladat szerint itt csak két bemeneti változó van (A és B), kimeneten két jel keletkezik, az összeg S (sum) és az átvitel C (carry). Ezt az áramkört félösszeadónak hívjuk és a következő igazságtábla alapján kell megtervezni:

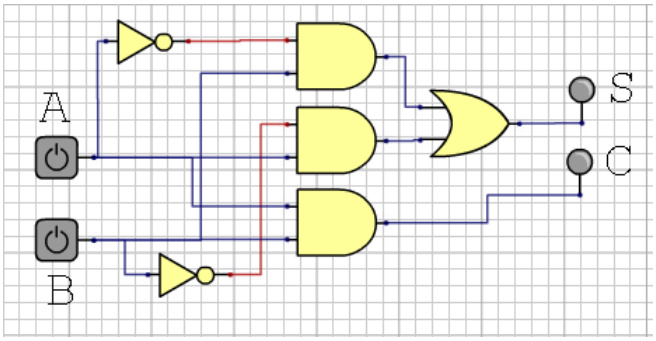
BEMENET		KIMENET	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

8.23. ábra: A félösszeadó igazságtáblája

Az igazságtáblában ránézésre láthatjuk, hogy az átvitel egy ÉS függvény, az összeg pedig egy KIZÁRÓ–VAGY függvény. A KIZÁRÓ–VAGY függvény felírható a következő alakban:

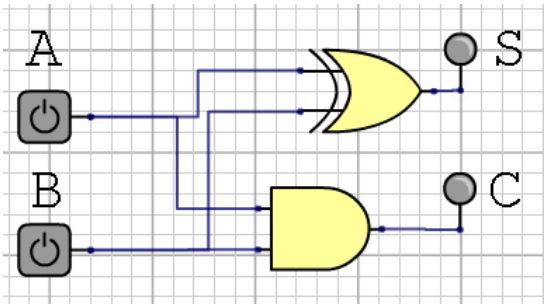
$$F = A \oplus B = A\,\bar{B} + \bar{A}\,B$$

A félösszeadó ezek után ÉS, VAGY kapukkal és INVERTER–rel:



8.24. ábra: Félösszeadó kapcsolási rajza ÉS, VAGY kapukkal és INVERTER–rel

A félösszeadó ÉS és KIZÁRÓ–VAGY kapuval:



8.25. ábra: Félösszeadó kapcsolási rajza ÉS és KIZÁRÓ–VAGY kapuval

8.9. Példa

Egy logikai áramkörre két 1 bites szám (A_i és B_i), valamint az eggyel kisebb helyiérték átvitele C_{i-1} érkezik. Tervezzen meg egy olyan áramkört, amelyik összeadja a két számot.

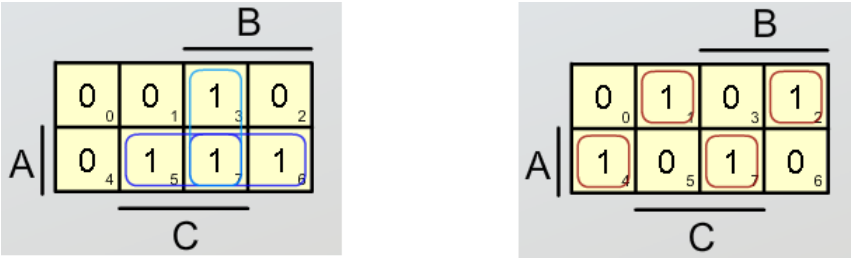
Megoldás:

A feladat szerint itt már három bemeneti változó van (A_i, B_i és C_{i-1}), a kimeneten két jel keletkezik, az összeg S_i (sum) és az átvitel C_i (carry). Az i index az épp aktuális helyiértéket, míg az $i-1$ az eggyel kisebb helyiértéket jelöli. Ezt az áramkört teljes összeadónak hívjuk és a következő igazságtábla alapján kell megtervezni:

BEMENET			KIMENET	
A_i	B_i	C_{i-1}	C_i	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

8.26. ábra: A teljes összeadó igazságtáblája

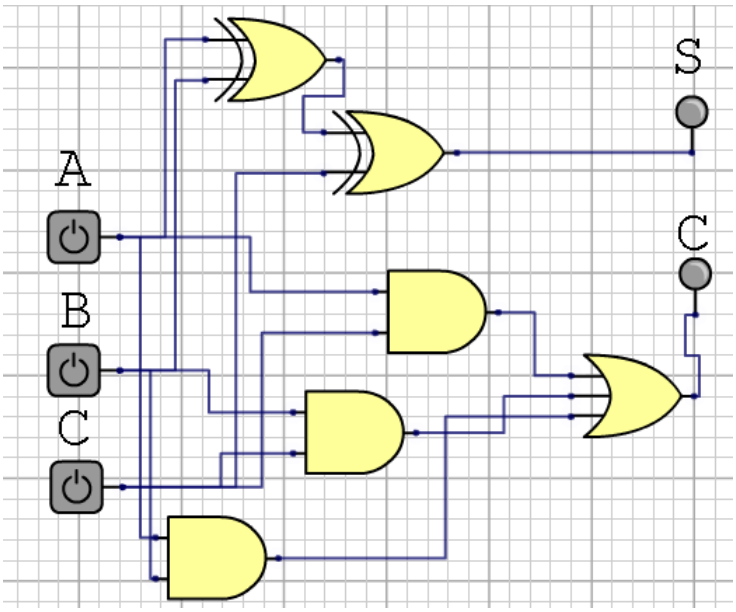
A Karnaugh-egyszerűsítéssel határozzuk meg a teljes összeadó összegét és az átvitelt (S_i , C_i):



8.27. ábra: Az összeg és átvitel meghatározása

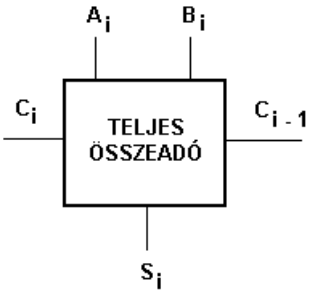
$$C_i = A_i\, C_{i-1} + A_i\, B_i + B_i\, C_{i-1}$$
$$S_i = \overline{A_i}\,\overline{B_i}\, C_{i-1} + \overline{A_i}\, B_i \overline{C_{i-1}} + A_i\, \overline{B_i}\, \overline{C_{i-1}} + A_i\, B_i\, C_{i-1} =$$
$$= C_{i-1}(\overline{A_i}\,\overline{B_i} + A_i\, B_i) + \overline{C_{i-1}}(\overline{A_i}\, B_i + A_i\, \overline{B_i}) =$$
$$= \overline{(A_i \oplus B_i)}\, C_{i-1} + (A_i \oplus B_i)\, \overline{C_{i-1}} = A_i \oplus B_i \oplus C_{i-1}$$

A kapcsolási rajz:



8.28. ábra: A teljes összeadó kapcsolási rajza

A fenti kapcsolást mint funkcionális elemet teljes összeadóként használhatjuk, ennek sémája:



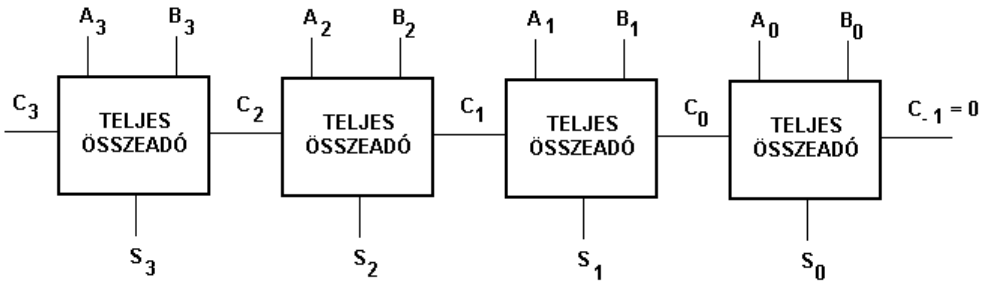
8.29. ábra: Teljes összeadó szimbolikus rajza

8.10. Példa

Tervezzünk meg egy 4 bites össszeadót, ha a szám A illetve B.

Megoldás:

A legkisebb helyiértéknél a C bemenetre logikai 0–át kell kapcsolni, hiszen nincs kisebb helyi érték, majd mind a négy teljes összeadónál a kisebb helyiérték átvitelét a következő helyiérték bemenetére kell kapcsolni.



8.30. ábra: 4 bites összeadó 4 teljes összeadóból felépítve

Megjegyzés: mivel első lépésben a legkisebb helyiértéken kerül meghatározásra az átvitel, ami az elemek késleltetése miatt egy bizonyos ideig eltart, a következő biten ugyanez a helyzet, az áramkör $n\ x\ t_{késl}$ idő alatt oldja meg a feladatot. Ezek szerint a bitek számával egyenesen arányos a kapcsolás késleltetése.

8.11. Példa

Tervezzünk meg egy olyan áramkört, amelyik két egybites számot hasonlít össze, ha a számok azonosak logikai 1, ha eltérőek logikai 0 jelenik meg a kimeneten.

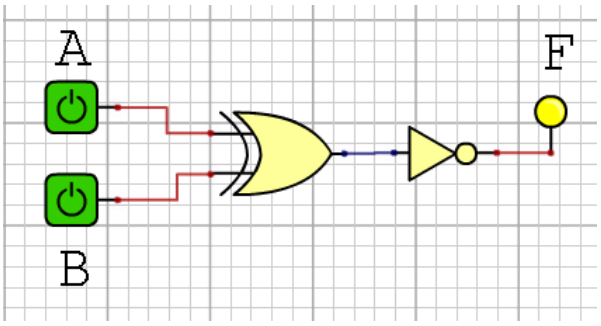
Megoldás:

Töltsük ki az igazságtáblázatot:

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

8.31. ábra: Két bináris szám összehasonlítása, az igazságtáblázat

Látható, hogy az F kimenet negáltja egy KIZÁRÓ–VAGY kapunak, így a kapcsolás:



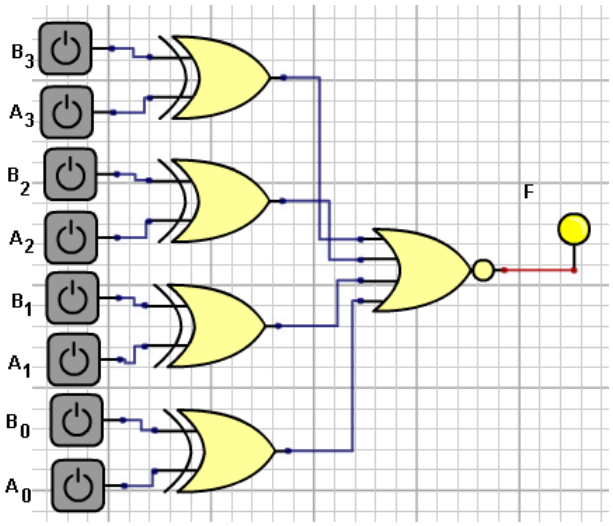
8.32 ábra: Két szám összehasonlítása, a megoldás KIZÁRÓ–VAGY kapuval

8.12. Példa

Tervezzünk meg egy olyan áramkört, amelyik két négybites számot hasonlít össze, ha a számok azonosak logikai 1, ha eltérőek logikai 0 jelenik meg a kimeneten.

Megoldás:

Mivel a két szám azonos helyiértékű bitjeit kell összehasonlítani egymással, ezért négy, az előző példában megtervezett áramkör kimenetét kell egy ÉS kapura vezetni. De Morgan szabályával kiiktathatjuk az invertereket, ha az ÉS kapu helyett NEM–VAGY kaput alkalmazunk.



8.33. ábra: Két négybites szám összehasonlítása, logikai kapcsolás

9. Logikai kapcsolások átalakítása – a De Morgan-szabály használata

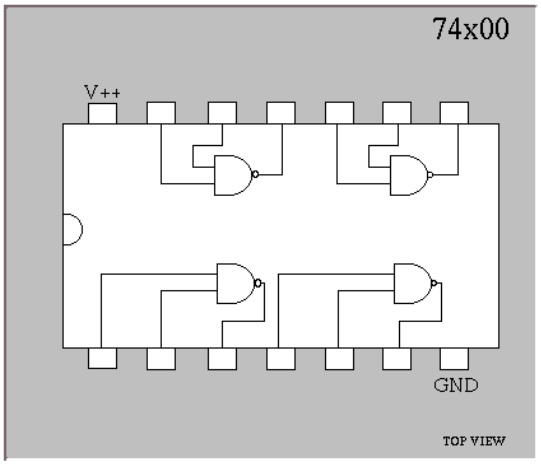
Az egyszerűsítési eljárások feladata csökkenteni a kapcsolás elemeinek számát. Miután megkaptuk a logikai kapcsolást, érdemes elemezni azt abból a szempontból, hogy milyen valós elemekkel (kapukkal) lehet azt megvalósítani. Logikai kapukat különböző technológiákkal állítanak elő, például TTL, CMOS stb. Ezeknek a kapuáramköröknek a létrehozásánál méret, lábkiosztás, fogyasztás stb. feltételeket vettek figyelembe.

A TTL sorozat néhány tagját elemezve látjuk, hogy 14 vagy 16 kivezetéssel rendelkezik a család legtöbb tagja, ebből következik, hogy korlátozott számban tartalmazhat kapukat. 14 láb esetén 2 tápvezeték és 12 láb áll rendelkezésre, egy két bemenettel (és egy kimenettel) rendelkező logikai kapu 3 ponton csatlakozik a külvilághoz, így 12 / 3 = 4 kapu helyezhető el egy csipen belül. Ugyenezzel a logikával a három bemenetű (és egy kimenetű) kapuból 12 / 4 = 3 kapu helyezhető el egy csipen belül.

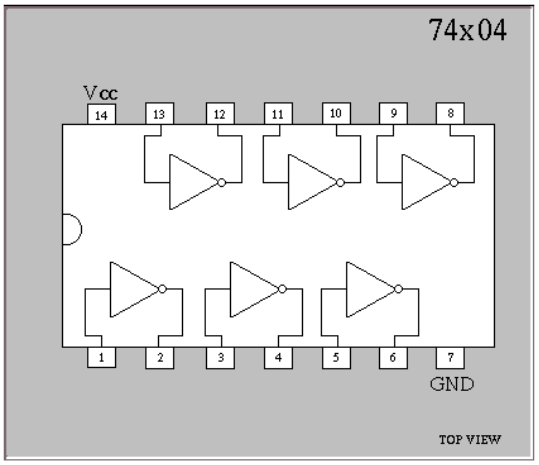
Nemcsak az számít, hogy a csipek száma csökkenjen (fogyasztás- és méret csökkenés, megbízhatóság növekedés), hanem az is fontos, hogy ha ugyanolyan típusú kapukat használunk, nem kell beszerezni sokféle elemet, karbantartásnál is elég egy típust raktáron tartani.

Nézzünk meg néhány csipet a TTL családból (forrás:

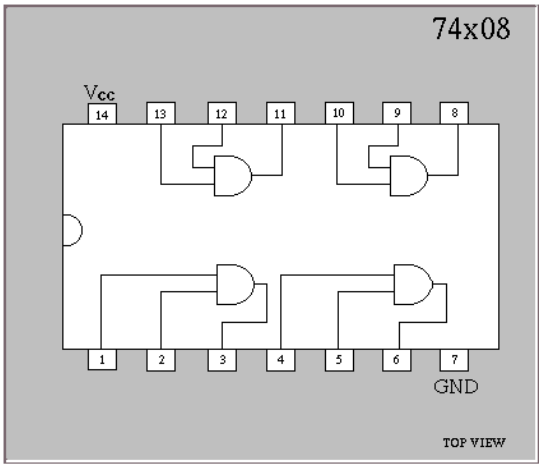
http://www.qsl.net/yo5ofh/data_sheets/data_sheets_page.htm):



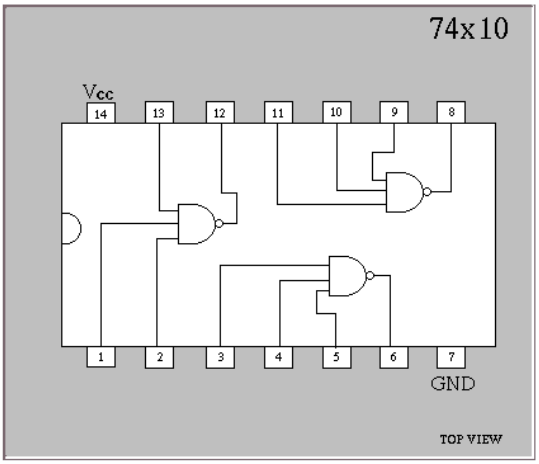
9.1. ábra: TTL 7400 csip, négy darab két bemenetű NEM–ÉS kapu



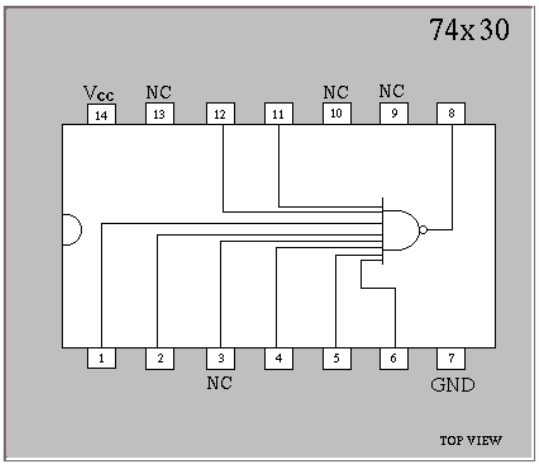
9.2. ábra: TTL 7404 csip, hat darab inverter



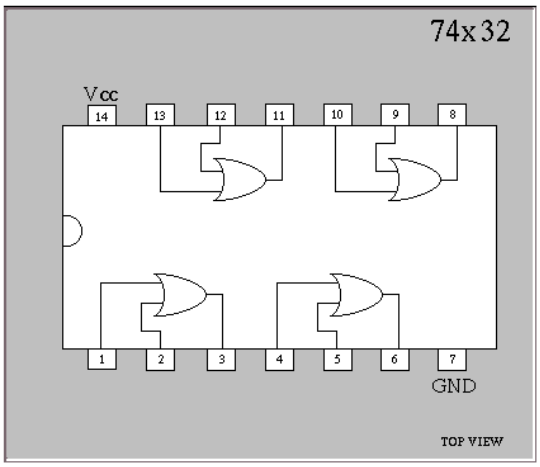
9.3. ábra: TTL 7408 csip,
négy darab két bemenetű ÉS kapu



9.4. ábra: TTL 7410 csip,
három darab három bemenetű ÉS kapu



9.5. ábra: TTL 7430 csip,
egy darab nyolc bemenetű NEM–ÉS kapu



9.6. ábra: TTL 7433 csip,
négy darab két bemenetű NEM–ÉS kapu

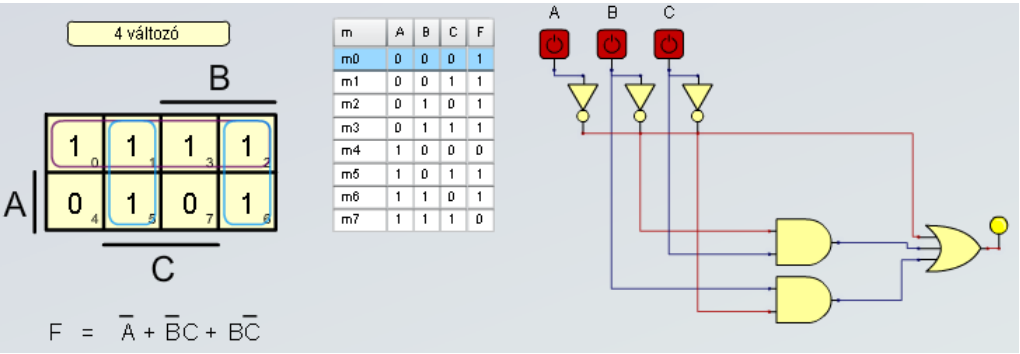
9.1. Példa

Adott a következő függvény: $F(A,B,C) = \sum^3 (0,1,2,3,5,6)$

a) egyszerűsítsük a függvényt Karnaugh módszerével,
b) egyszerűsítsük a függvényt A Quine–McCluskey módszerével,
c) rajzoljuk le a kapcsolást tetszőleges kapukkal,
d) rajzoljuk le a kapcsolást tetszőleges kapukkal, de legfeljebb 2 bemenetű kapukat használjunk,
e) rajzoljuk le a kapcsolást kizárólag NEM–ÉS kapukkal,
f) rajzoljuk le a kapcsolást kizárólag NEM–VAGY kapukkal,
g) rajzoljuk le a kapcsolást kizárólag két bemenetűNEM–ÉS kapukkal,
h) rajzoljuk le a kapcsolást kizárólag két bemenetű NEM–VAGY kapukkal,
i) rajzoljuk le a kapcsolást kizárólag három bemenetű NEM–ÉS kapukkal,
j) rajzoljuk le a kapcsolást kizárólag három bemenetű NEM–VAGY kapukkal,
k) elemezzük ki, melyik megoldásnál kell a legkevesebb csipet használni, valamint azt, hogy a megoldásoknál a csipekben hány szabad kapu marad.

Megoldás:

a) és c):

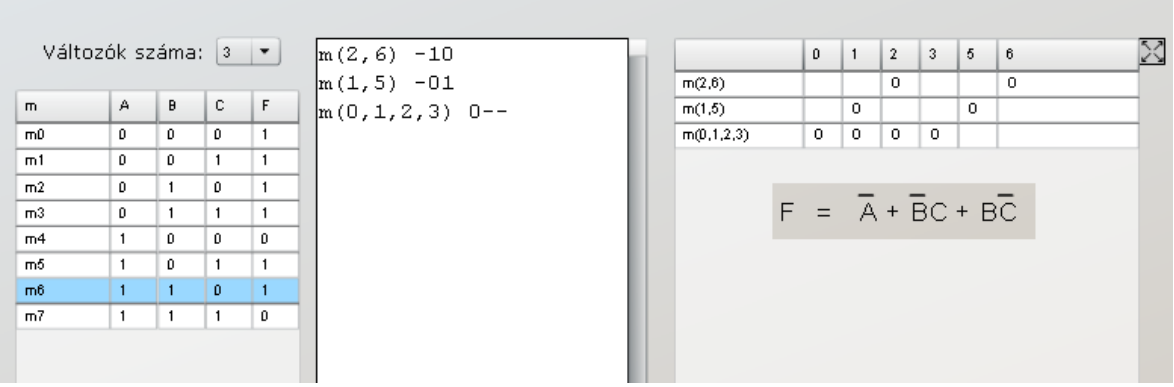


9.7. ábra: Az igazságtáblázat, a Karnaugh-tábla és a logikai kapcsolási rajz

A megvalósításhoz kell:

3 INVERTER, 1 csip, szabadon maradó kapuk száma = 3,
2 két bemenetű ÉS kapu, 1 csip, szabadon maradó kapuk száma = 2,
1 három bemenetű VAGY kapu, 1 csip, szabadon maradó kapuk száma = 2,
3 csip.

b)

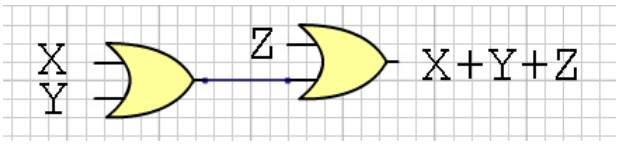


9.8. ábra: Az igazságtáblázat, megoldás Quine–McCluskey módszerével

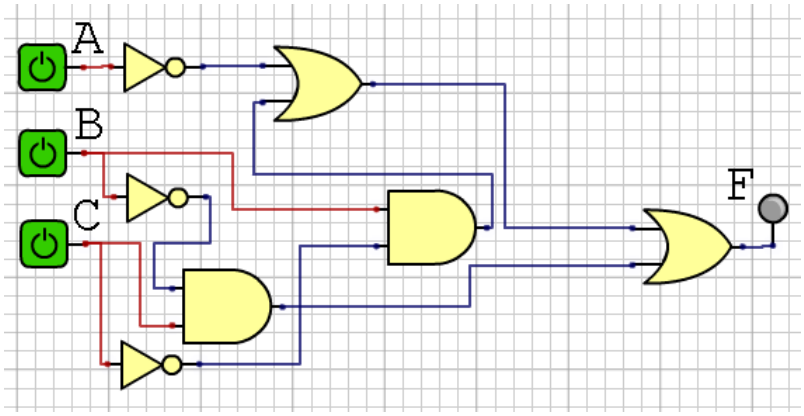
Látható, hogy ugyanazt a megoldást kaptuk mint a Karnaugh-módszerrel.

d)

A kapcsolás 3 bemenetű VAGY kapuja helyettesíthető 2 darab két bemenetű VAGY kapuval:



9.9. ábra: 3 bemenetű VAGY helyettesítése 2 darab két bemenetű kapuval



9.10. ábra: Kapcsolás legfeljebb 2 bemenetű kapukkal

A megvalósításhoz kell:

3 INVERTER, 1 csip, szabadon maradó kapuk száma = 3,
2 két bemenetű ÉS kapu, 1 csip, szabadon maradó kapuk száma = 2,
2 két bemenetű VAGY kapu, 1 csip, szabadon maradó kapuk száma = 2,
3 csip.

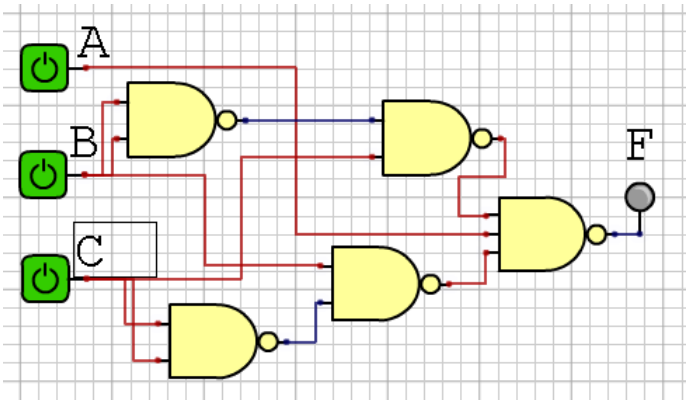
e)

Az a) pontban kapott egyenletet át kell alakítani De Morgan szabállyal, amelyik a következő:

$A + B = \overline{\bar{A}\,\bar{B}}$, illetve $A\,B = \overline{\bar{A} + \bar{B}}$, vagyis mindig átalakítható a VAGY kapu ÉS (NEM–ÉS) kapuvá, illetve az ÉS kapu VAGY (NEM–VAGY) kapuvá.

$$F = \bar{A} + \bar{B}C + B\bar{C} = \overline{A\,\overline{\bar{B}C}\,\overline{B\bar{C}}}$$

Mivel a feladatban szerepel a „kizárólag NEM–ÉS kapukkal” feltétel, az inverterek helyett is NEM–ÉS kaput használunk. Ilyenkor vagy összekötjük a NEM–ÉS kapu két (vagy több) lábát, ekkor jobban leterheljük az előző fokozat kimenetét, vagy állandó „1”-es jelet kötünk a szabad bemenetekre. Mi összekötöttük a két bemenetet.



9.11. ábra: A kapcsolás kizárólag NEM–ÉS kapukkal

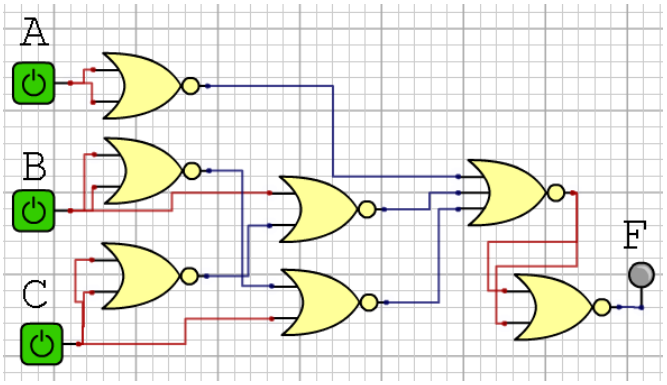
A megvalósításhoz kell:

4 két bemenetű NEM–ÉS kapu, 1 csip, szabadon maradó kapuk száma = 0,
1 három bemenetű NEM–ÉS kapu, 1 csip, szabadon maradó kapuk száma = 2,
2 csip.

f)

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

Mivel a feladatban szerepel a „kizárólag NEM–VAGY kapukkal” feltétel, az inverterek helyett is NEM–VAGY kaput használunk. Ilyenkor vagy összekötjük a NEM–VAGY kapu két (vagy több) lábát, ekkor jobban leterheljük az előző fokozat kimenetét, vagy állandó „0”–ás jelet kötünk a szabad bemenetekre. Mi összekötöttük a két bemenetet.



9.12. ábra: A kapcsolás kizárólag NEM–VAGY kapukkal

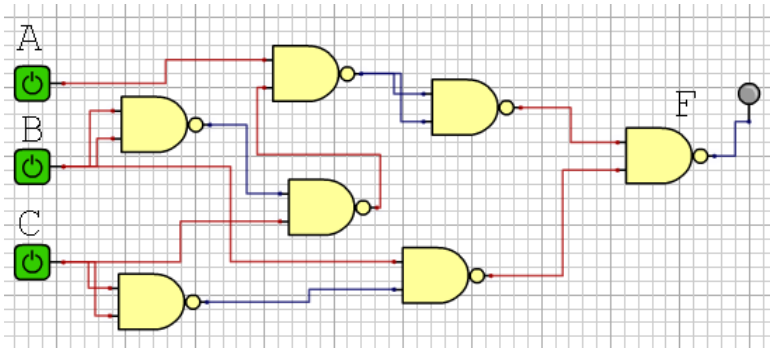
A megvalósításhoz kell:

6 két bemenetű NEM–VAGY kapu, 2 csip, szabadon maradó kapuk száma = 2,
1 három bemenetű NEM–VAGY kapu, 1 csip, szabadon maradó kapuk száma = 2,
3 csip.

g)

Az e) pont képletéből indulunk ki, a három bemenetű NEM–ÉS kaput átalakítjuk két bemenetűvé:

$$F = \bar{A} + \bar{B}C + B\bar{C} = \overline{A\,\overline{\bar{B}C}\;\overline{B\bar{C}}} = \overline{(A\,\overline{\bar{B}C})\;\overline{B\bar{C}}} = \overline{\overline{\overline{(A\,\overline{\bar{B}C})}\;\overline{B\bar{C}}}}$$



9.13. ábra: A kapcsolás kizárólag két bemenetű NEM–ÉS kapukkal

A megvalósításhoz kell:

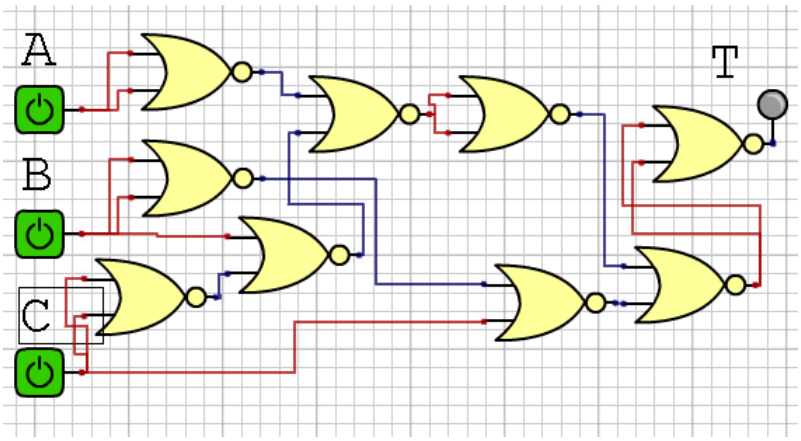
7 két bemenetű NEM–ÉS kapu, 2 csip, szabadon maradó kapuk száma = 1,
2 csip.

h)

az f) pontban kapott képletből indulunk ki:

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

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



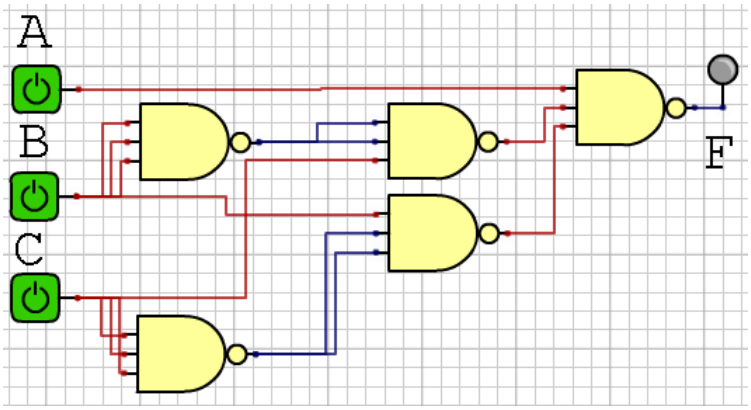
9.14. ábra: A kapcsolás kizárólag két bemenetű NEM–VAGY kapukkal

A megvalósításhoz kell:

9 két bemenetű NEM–VAGY kapu, 3 csip, szabadon maradó kapuk száma = 3,
3 csip.

i)

Az e) pont xx.11. ábráját alakítjuk át, úgy hogy a két bemenetű kapukat három bemenetű-vel helyettesítjük:



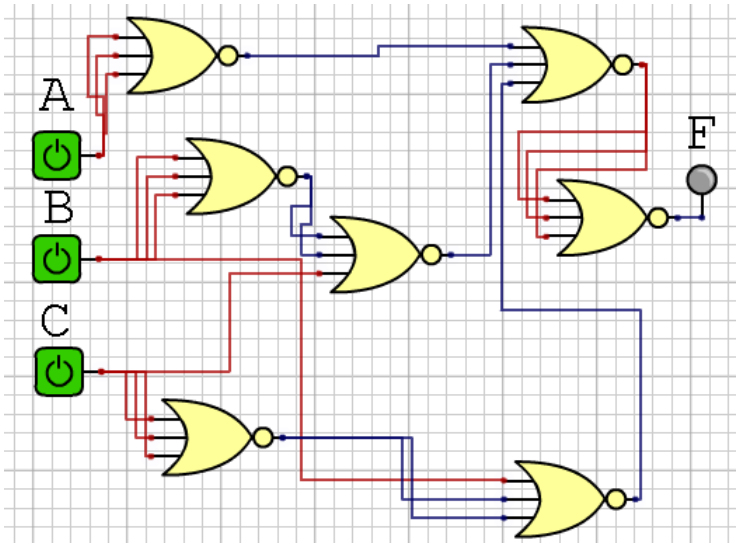
9.15. ábra: A kapcsolás kizárólag három bemenetű NEM–ÉS kapukkal

A megvalósításhoz kell:

4 három bemenetű NEM–ÉS kapu, 2 csip, szabadon maradó kapuk száma = 1,
2 csip.

j)

Az e) pont xx.10. ábráját alakítjuk át, úgy hogy a két bemenetű kapukat három bemenetűvel helyettesítjük:



9.16. ábra: A kapcsolás kizárólag három bemenetű NEM–VAGY kapukkal

A megvalósításhoz kell:

7 három bemenetű NEM–VAGY kapu, 3 csip, szabadon maradó kapuk száma = 2,
3 csip.

k)

Ábra	A csipek száma	Szabadon maradó kapuk száma
9.7.	3	7
9.10.	3	5
9.11.	2	2
9.12.	3	4
9.13.	2	1
9.14.	3	3
9.15.	2	1
9.16.	3	2

Ha a cél az, hogy a lehető legkevesebb csippel oldjuk meg a feladatot, akkor 3 ilyen megoldást is találtunk (9.11., 9.13. és 9.15. ábra).

10. Sorrendi (szekvenciális) hálózatok

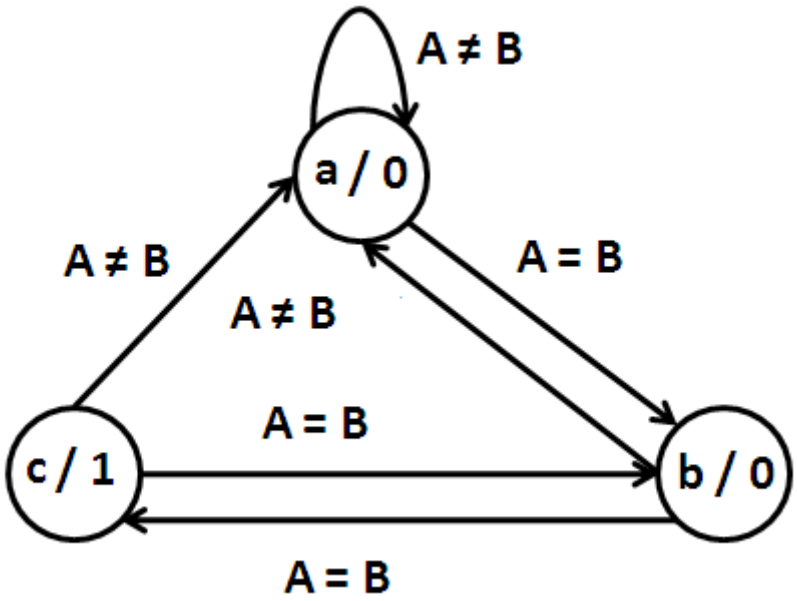
A sorrendi hálózatoknál a kimenet értékeit a bemenet értékei mellett a tárolt adatok is befolyásolják. Leggyakrabban RS, JK, T és D tárolókat használunk a logikai értékek tárolására. A feladatoknak általában több megoldása van, a sorrendi hálózatoknál is ez a helyzet, a tárolók tulajdonságai, így felhasználási területük is eltérő. Bizonyos típusú feladatoknál egyik típusú tároló a másikkal előnyben van, könnyebben oldható meg a feladat.

10.1. Példa

Tervezzünk meg egy szinkron hálózatot, amelyiknek két bemenete van, A és B, valamint egy kimenete, F. A és B logikai jel az órajellel szinkronban érkezik a bemenetre. Ha különböznek a jelek értékei (A≠B), akkor az F kimenet 0 értéket adjon az áramkör, ha A és B bemeneten azonos érték van (A=B=0 vagy A=B=1), akkor az utána következő azonos érték esetében F=1 legyen, egyébként kezdje újra az értékek figyelését a logika.

Megoldás:

A feladat megoldásához az előző szöveges leírásból ki kell hámozni az összefüggéseket és azokat valahogy olyan alakban ábrázolni ahol egyértelművé válnak a kapcsolatok. Ennek egyik lehetősége az átmeneti tábla, egy másik hatásos módszer pedig az állapotgráf, vagy más néven az állapotdiagram. Használjuk ki az állapotgráf előnyeit és rajzoljuk le a gráfot. A körök az állapotok, a nyilak az átmenetek, amelyeket az órajel lefutó éle vált ki.



10.1. ábra: A feladat állapotgráfja

„a” állapotban van a rendszer, amikor a két bemeneten (A és B) különböző értékek jelennek meg (A = 0 és B = 1, illetve A = 1 és B = 0), az F kimeneten pedig 0 érték van. Ebből az állapotból csak a bemeneti értékek azonossága esetén mozdul ki, átugrik „b” állapotba, de ekkor még 0 a kimenet. Ezt természetesen az órajel lefutó éle okozza. A következő óra-

jel hatására vagy újból „a” állapotba kerül a rendszer (A ≠ B), vagy „c” állapotba, ekkor a kimenet F = 1 értéket vesz fel. A következő órajel esetén, amikor A = B, akkor „b” állapotba kerül vissza a rendszer, hiszen ha ismét A = B bemenet van újból „c” állapotba kerül, de „c”–ből közvetlenül „a”–ba kerül A ≠ B esetén.

Ahhoz, hogy megtervezzük a kapcsolást ki kell tölteni a rendszer állapottábláját, ezt az állapotgráfból egyszerűen megtehetjük:

Jelenlegi állapot	Következő állapot / kimenet	
	A ≠ B	A = B
a	a / 0	b / 0
b	a / 0	c / 1
c	a / 0	b / 0

10.2. ábra: A rendszer állapottáblája

Három állapot kódolása csak 2 tárolóval lehetséges ($2^2 = 4 > 3$). Válasszuk a JK szinkron tárolót, amelynek állapottáblájából megkaphatjuk a vezérlési táblát.

Q_t	Q_{t+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

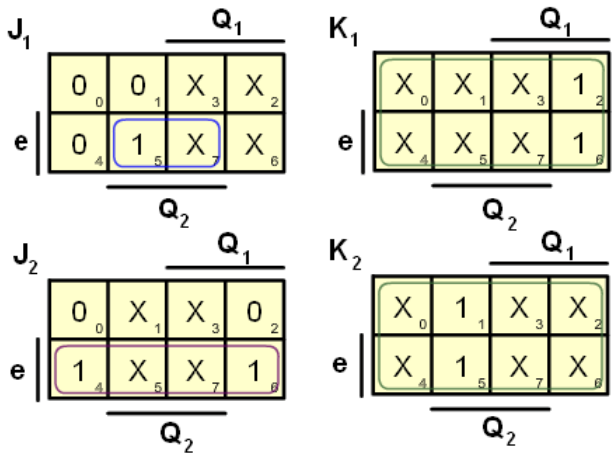
10.3. ábra: A JK tároló vezérlési táblája

Miután meghatároztuk a tároló típusát kitölthetünk egy kombinált állapottáblát, amelyik tartalmazza a rendszer szimbolikus állapottábláját, a kódolt állapottáblát és a vezérlési táblát.

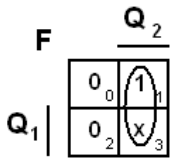
szimbolikus állapottábla				kódolt állapottábla		vezérlési tábla			
aktuális állapot/ kimenet	következő aktuális állapot	következő állapot		kódolt következő állapot		$\bar{e}$		e	
	$Q_1\ Q_2$	$\bar{e}$	e	$\bar{e}$ $Q_1\ Q_2$	e $Q_1\ Q_2$	$J_1\ K_1$	$J_2\ K_2$	$J_1\ K_1$	$J_2\ K_2$
a / 0	0 0	a	b	0 0	0 1	0 x	0 x	0 x	1 x
b / 0	0 1	a	c	0 0	1 0	0 x	x 1	1 x	x 1
c / 1	1 0	a	b	0 0	0 1	x 1	0 x	x 1	1 x
	1 1	x	x	x x	x x	x x	x x	x x	x x

10.4. ábra: A szimbolikus állapottábla, a kódolt állapottábla és a vezérlési tábla

A vezérlési tábla adataiból Karnaugh módszerével megkaphatjuk az egyszerűsített vezérlési egyenleteket a két tároló számára, valamint meghatározhatjuk a kimenetet is.



10.5. ábra: A két JK tároló vezérlési egyenleteinek meghatározása Karnaugh-egyszerűsítéssel

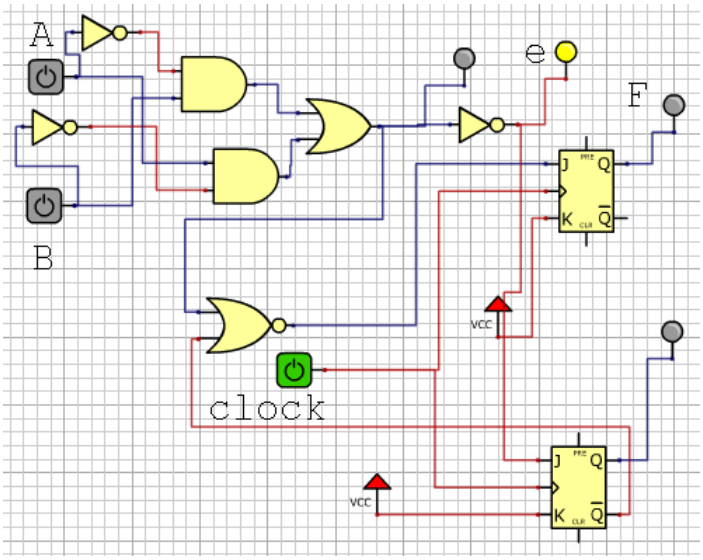


10.6. ábra: Az F kimenet meghatározása Karnaugh-egyszerűsítéssel

A kapott vezérlési és kimeneti egyenletek:

$$J_1 = Q_2\, e \qquad K_1 = 1 \qquad J_2 = e \qquad K_2 = 1 \quad \text{és} \quad F = Q_1$$

Ezeket a képleteket átalakítva kapjuk a következő kapcsolási rajzot:



10.7. ábra: A megoldás kapcsolási rajza

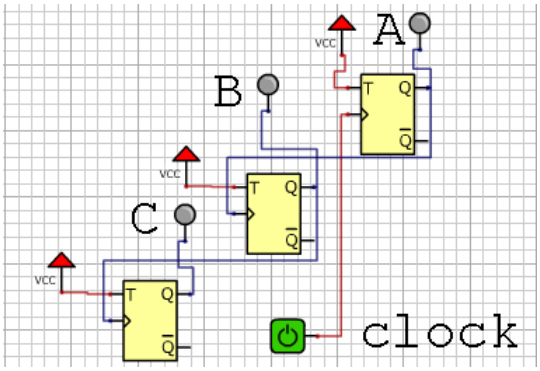
10.2. Példa

Tervezzünk meg egy olyan számlálót, amelyik minden bejövő impulzus után a kimenetén eggyel növeli binárisan az értéket, a számláló 0, 1, 2 stb. 7 után újból 0 értékre váltson.

Megoldás:

A T tároló minden órajel lefutó élére a kimeneten ellenkező logikai értéket ad ki, vagyis 0 → 1 és 1 → 0 változás történik. Ez azt jelenti hogy két bemenő impulzus egy kimeneti impulzust ad, másképpen a bemenő frekvencia fele jelenik meg a kimeneten. Így az A kimeneten f / 2, B kimeneten f / 4 és C kimenet f / 8 frekvencia jelenik meg, ha a bemeneten f frekvenciájú impulzussorozat van.

Mivel a T tárolónál 2 bejövő impulzus egy impulzust ad a kimenetén, ezért felhasználható számlálónak. A 0 – 7 szám előállítása ezek után 3 biten megoldható a következő ábra szerint:



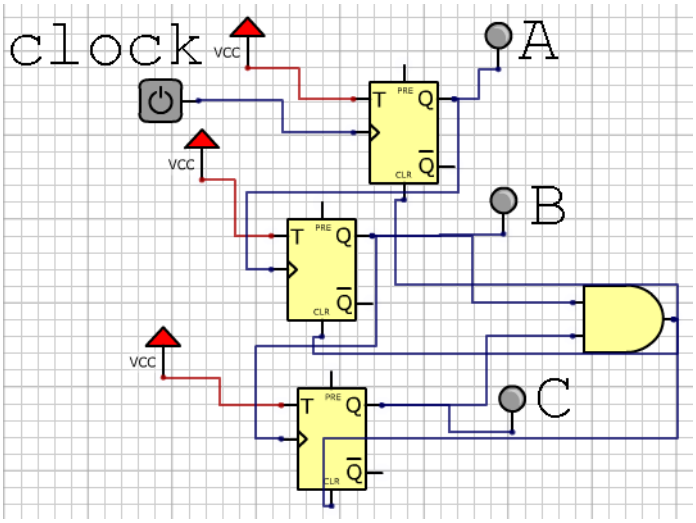
10.8. ábra: A megoldás logikai rajza

10.3. Példa

Tervezzünk meg egy olyan számlálót, amelyik minden bejövő impulzus után a kimenetén eggyel növeli binárisan az értéket, a számláló 0, 1, 2 stb. 5 után újból 0 értékre váltson.

a) Megoldás:

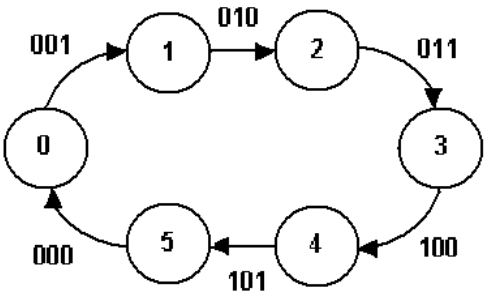
Hasonlóan kell gondolkodni a megoldásnál mint az előző példánál, de most a számláló 5 után visszabillen 0 értékre. Ezt úgy tudjuk megoldani, hogy a 6 szám elérésekor töröljük a T tárolókat a Clear bemeneten keresztül. Ezt egy ÉS kapu hozzáadásával érhetjük el.



10.9. ábra: A 0 – 5 számláló JK tárolókkal

b) Megoldás:

Állítsuk fel a feladat állapotgráfját:



10.10. ábra: A feladat állapotgráfja

Az állapotgráf felállítása során feltételeztük azt, hogy a rendszer bekapcsoláskor kezdőállapotból, vagyis 0 (000) alapállapotból indul, ezt a tárolók Preset ill. Clear bemeneteinek meghatározásával (esetünkben a Clear) érhetjük el. Az állapotgráfban az állapotokat decimális értékekkel számoztuk, 0 –5 között, míg az átmeneteknél a nyílakon a kimenetek értékei találhatók, CBA sorrendben, A a legkisebb helyi érték, míg C a legnagyobb. Minden bejövő órajel lefutó élére a számláló a következő állapotba jut, a kapcsolás CBA kimenetén megjelenítve az aktuális számot bináris alakban. Az állapottábla tartalmazza a jelenlegi állapotot (Q_t) decimálisan és binárisan, a következő állapotot (Q_{t+1}) szintén decimálisan és binárisan, valamint a vezérlési táblát. Mindhárom JK tároló szerepel a vezérlési táblában, mégpedig J és K bemeneteik vezérlését meghatározó értékekkel. Ezt a részét a táblának a JK tároló vezérlési táblájával (10.3. ábra) és ebben a példában az aktuális átmenetek kombinációjával töltjük ki. Vegyük a 0 → 1, vagyis a 000 →001 átmenetet, itt érvényes CBA sorrend, a vezérlési táblában a

C JK tároló értékei a 0 → 0 $Q_t \rightarrow Q_{t+1}$ átmenetnél J = 0 és K = x,
B JK tároló értékei a 0 → 0 $Q_t \rightarrow Q_{t+1}$ átmenetnél J = 0 és K = x és
A JK tároló értékei a 0 → 1 $Q_t \rightarrow Q_{t+1}$ átmenetnél J = 1 és K = x.

Ezt az eljárást végigvisszük a táblázat utolsó soráig, 5 → 0, vagyis a 101 →000 átmenetet ahol a következő az eljárás:

C JK tároló értékei a 1 → 0 $Q_t \rightarrow Q_{t+1}$ átmenetnél J = x és K = 1,
B JK tároló értékei a 0 → 0 $Q_t \rightarrow Q_{t+1}$ átmenetnél J = 0 és K = x és
A JK tároló értékei a 1 → 0 $Q_t \rightarrow Q_{t+1}$ átmenetnél J = x és K = 1.

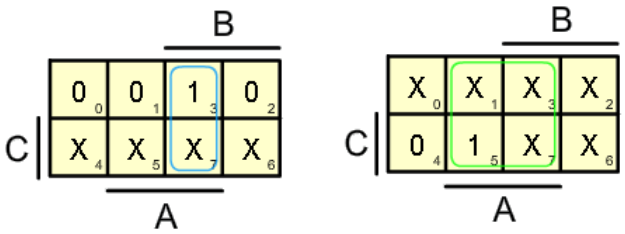
Az állapotgráf alapján felállítjuk az állapottáblát:

Jelenlegi állapot		Következő állapot		Vezérlési tábla					
				C		B		A	
Q_t		Q_{t+1}		J	K	J	K	J	K
0	000	1	001	0	x	0	x	1	x
1	001	2	010	0	x	1	x	x	1
2	010	3	011	0	x	x	0	1	x
3	011	4	100	1	x	x	1	1	x
4	100	5	101	x	0	0	x	1	x
5	101	0	000	x	1	0	x	x	1

10.11. ábra: A feladat állapottáblája

Mivel JK tárolót használunk, ezért a JK tároló vezérlési táblájából kell kitölteni a három kimenet átmenetei alapján a vezérlést. Három változó esetén 8 kombináció van, itt ebből hat fordul elő, a 6 és 7 szám mint közömbös kombináció kerül be a tervezésbe, hiszen ezek sohasem fordulnak elő.

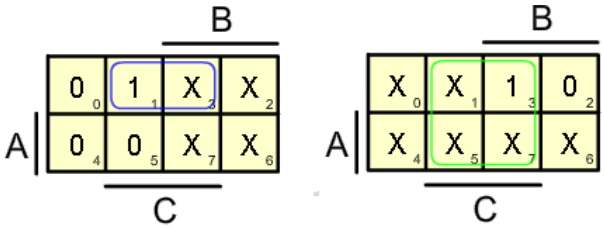
A következő lépésben a vezérlési táblából meghatározzuk mindhárom tároló J és K bemenetének függvényét. Ha Karnaugh-féle egyszerűsítési eljárást választjuk, akkor a C JK tároló J és K bemenetének táblája a következő:



10.12. ábra: A C JK tároló vezérlésének egyszerűsítése

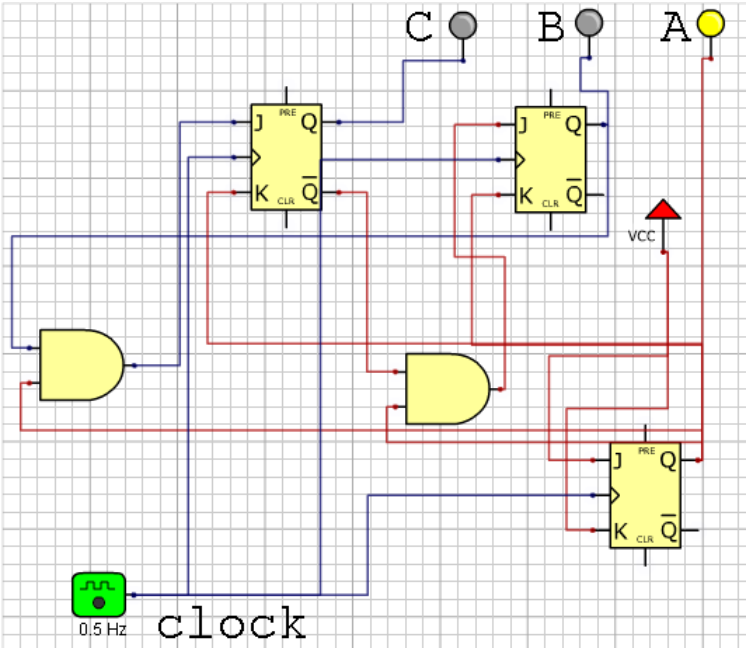
A vezérlés egyenletei $J_C = A\,B$ és $K_C = A$. Figyeljünk fel rá, hogy a Karnaugh-táblánál a C változó a legnagyobb helyiértékkel rendelkezik, míg A a legkisebb.

A B JK tároló J és K bemenetének táblája a következő:



10.13. ábra: A B JK tároló vezérlésének egyszerűsítése

A vezérlés egyenletei $J_B = A\,\bar{C}$ és $K_B = A$.



10.14. ábra: A számláló logikai kapcsolási rajza.

Az A JK tároló J és K bemenetének vezérlését ránézésre is meghatározhatjuk, mindkét bemenetnél 1–esek és x–ek vannak, ebből következik, hogy mind a nyolc minterm egy hurokkal fogható össze, így egyik változónak sincs hatása a J és K bemenetekre, vagyis $J_A = 1$ és $K_A = 1$.

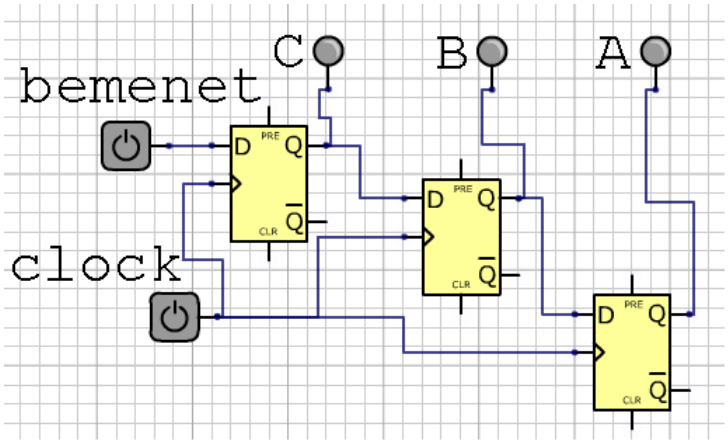
Ezek után a fenti egyenleteket felhasználva és bekötve egy impulzusgenerátort a fenti kapcsolást kapjuk.

10.4. Példa

Tervezzünk meg egy 3 bites jobbra léptető regisztert, ha C bemenetre érkező 0 vagy 1 órajel hatására lép be.

Megoldás:

A szinkron D tároló lefutó él hatására a bemenetet átmásolja a kimenetre, így ez az elem alkalmas a feladat megoldására. Mivel három bites a regiszter, három D tárolóra van szükség.



10.15. ábra: A feladat megoldása D tárolókkal.

10.5. Példa

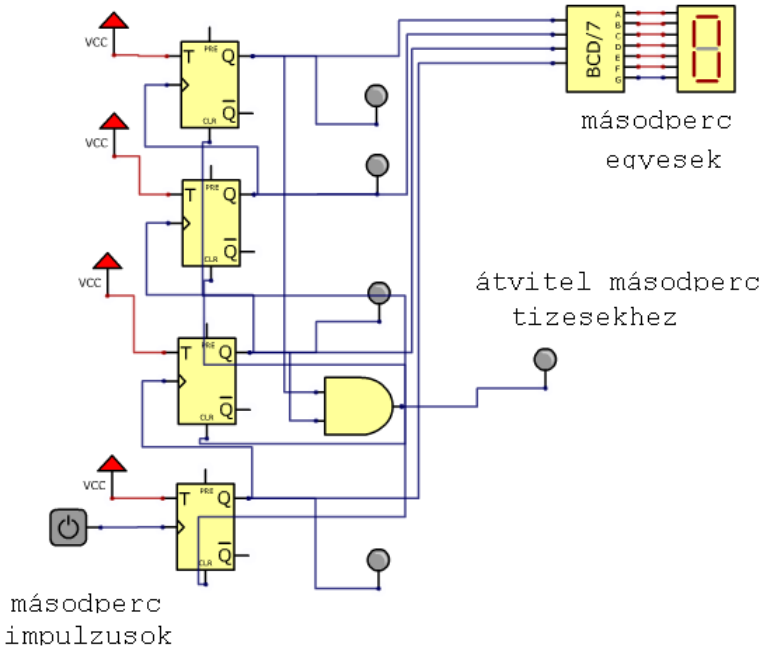
Készítsünk el egy órát, amelyik egy ütemadótól 1 másodpercenként kap órajelet. Az óra jelezze ki a másodperceket, perceket és az órát 0-tól 24-ig.

Megoldás:

A megoldásban kell, hogy szerepeljen olyan számláló, amelyik periódusa 60 illetve 24 (lehet 12 és kijelezni a délelőttöt–délutánt). A tízes számrendszer általános használata miatt sokszor szükséges a 10-es periódusú számláló is. Ezeket a számlálókat összeállíthatjuk a 2, 3 és 5 periódusú számlálók sorbakötésével. A 2-vel osztó számláló maga a JK tároló, 3 és 5 periódusú számláló előállítható néhány kiegészítő kapuval. A 6-os periódusú számláló például a 2, 3 és 1.

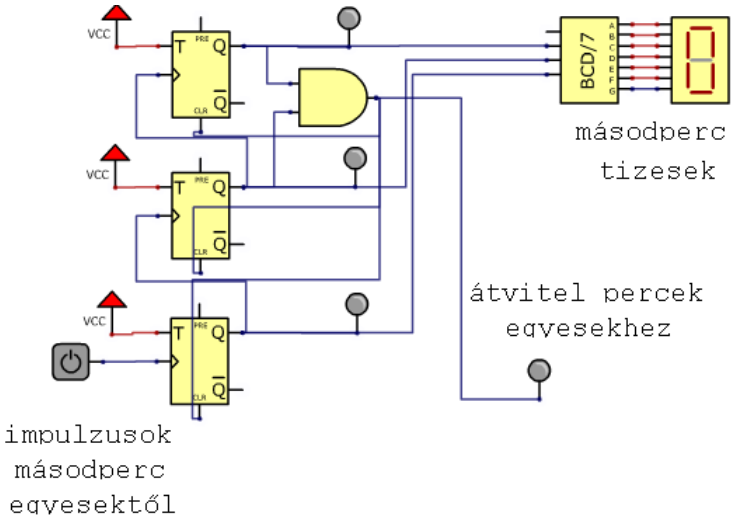
Mivel az értékeket decimálisan is ki kell jelezni, ezért érdemes 10-es, 6-os és 2-es számlálókat használni.

Először készítsük el a másodpercek számlálását végző rész másodperceket számláló 10 periódusú számlálóját 4 darab T tárolóval, visszacsatolással 10-nél. A visszacsatolás jel egyidejűleg bemenet a következő 6-os számlálóba, amely rész a másodpercek tízes helyértékét mutatja.



10.16. ábra: Másodperc egyesek

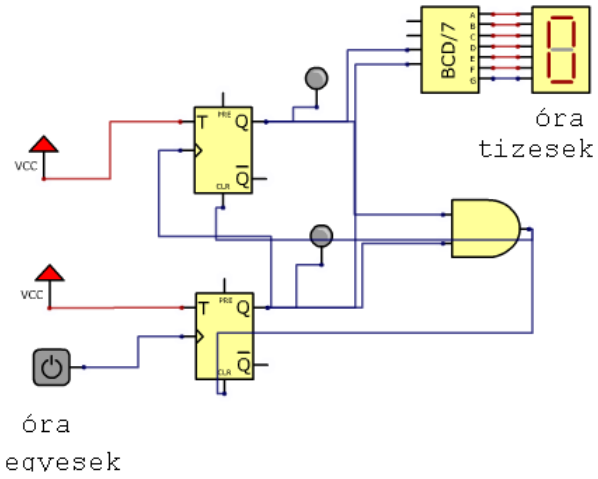
A másodpercek tízesek számlálását 3 T tárolóval lehet megoldani, 0-tól 5-ig kell számolni, visszacsatolás 6 értéknél, amely kimenet egyúttal bemenet a percek egyesek kapcsoláshoz.



10.17. ábra: Másodperc tízesek

A perceket ugyanolyan 10 és 6 periódusú számláló számlálja a rendszer mint a másodperceket, bemenet a másodperc tízesek kimenettől, kimenet a perc tízes.

Az órákat 0-tól 23-ig mérjük, az óra egyeseket egy 10 periódusú számlálóval, a tízeseket egy 3 periódusúval:



10.18. ábra: Óra tízesek

Irodalom

Matijevics István: Digitális technika, előadás – segédanyag, SZTE

Szittya Ottó: Digitális és analóg technika informatikusoknak 1 és 2, LSI, 2000.

Zsom Gyula: Digitális technika I, Műszaki Könyvkiadó, Budapest, 2000, (KVK 49-273/I).

Rőmer Mária: Digitális rendszerek áramkörei, Műszaki Könyvkiadó, Budapest, 1989, (KVK 49-223).

Arató Péter: Logikai rendszerek tervezése, Tankönyvkiadó, Budapest, 1990, Műegyetemi Kiadó 2004, 55013 (műegyetemi jegyzet).

Rőmer Mária: Digitális technika példatár, KKMF 1105, Budapest 1999.

Arató Péter: Logikai rendszerek. Tankönyvkiadó, Bp. 1985.

J.F.Wakerley: Digital Design. Principles and Practices; Prentice Hall 1990.

Selényi - Benesóczky: Digitális technika példatár. BME jegyzet, 1991.

U. Tietze, Ch. Schenk: Analóg és digitális áramkörök, Műszaki Könyvkiadó, Budapest, 1993

Janovich Sándor, Tóth Mihály: A logikai tervezés módszerei, Műszaki Könyvkiadó, Budapest, 1973.

Ábrajegyzék