IRÁNYÍTÁSTECHNIKA I.

A projekt címe: „Egységesített Jármű- és mobilgépek képzés- és tananyagfejlesztés”

A megvalósítás érdekében létrehozott konzorcium résztvevői:

KECSKEMÉTI FŐISKOLA
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
AIPA ALFÖLDI IPARFEJLESZTÉSI NONPROFIT KÖZHASZNÚ KFT.

Fővállalkozó: TELVICE KFT.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Közlekedésmérnöki Kar

Írta:
TARNAI GÉZA
BOKOR JÓZSEF
SÁGHI BALÁZS
BARANYI EDIT
BÉCSI TAMÁS

Lektorálta:
GÁSPÁR PÉTER

IRÁNYÍTÁSTECHNIA I.

Egyetemi tananyag

TYPOTEX

2011

LEKTORÁLTA: Dr. Gáspár Péter

ISBN 978-963-279-602-4

KÉSZÜLT: a Typotex Kiadó gondozásában

FELELŐS VEZETŐ: Votisky Zsuzsa

TÁMOGATÁS:
Készült a TÁMOP-4.1.2/A/2-10/1-2010-0018 számú, „Egységesített Jármű- és mobilgépek képzés- és tananyagfejlesztés” című projekt keretében.

KULCSSZAVAK:

Irányítástechnika, digitális technika, logikai kapuk, logikai függvények, minimalizálás, kombinációs hálózatok, hazárdok, sorrendi hálózatok, szekvenciális hálózatok, szinkron hálózatok, aszinkron hálózatok.

ÖSSZEFOGLALÁS:

A jelen jegyzet a BME Közlekedésmérnöki és Járműmérnöki Karán oktatott Irányítástechnika I. c. tantárgyhoz készült. A jegyzet célja, hogy segítse a hallgatókat az előadási anyag elsajátításában és a gyakorlati feladatok megoldásában.
A jegyzet felépítésében az előadások anyagát követi. Elsőként megismerteti az olvasót a logikai függvényekhez kapcsolódó alapvető elméleti háttérrel, majd a kombinációs hálózatok tervezésének módszereivel foglalkozik. Ezt követően a sorrendi hálózatok tervezésének bemutatása, amelynek során először a sorrendi hálózatok általános működésével, majd elsőként a szinkron sorrendi hálózatok tervezésével, ezt követően pedig az aszinkron hálózatok tervezésével foglalkozunk.
A jegyzethez kiterjedt példatár kapcsolódik, amely kiegészíti az elméleti anyagot és megoldott gyakorlati tervezési feladatokkal segíti a hallgatókat az Irányítástechnika I tárgykörébe tartozó mérnöki ismeretek megszerzésében.

TARTALOMJEGYZÉK

1. BEVEZETÉS

A jelen jegyzet a BME Közlekedésmérnöki és Járműmérnöki Karán oktatott Irányítástechnika I. c. tantárgyhoz készült. A jegyzet célja, hogy segítse a hallgatókat az előadási anyag elsajátításában és a gyakorlati feladatok megoldásában.
A jegyzet felépítésében az előadások anyagát követi. Elsőként megismerteti az olvasót a logikai függvényekhez kapcsolódó alapvető elméleti háttérrel, majd a kombinációs hálózatok tervezésének módszereivel foglalkozik. Ezt követően a sorrendi hálózatok tervezésének bemutatása, amelynek során először a sorrendi hálózatok általános működésével, majd elsőként a szinkron sorrendi hálózatok tervezésével, ezt követően pedig az aszinkron hálózatok tervezésével foglalkozunk. A jegyzethez kiterjedt példatár kapcsolódik, amely kiegészíti az elméleti anyagot és megoldott gyakorlati tervezési feladatokkal segíti a hallgatókat az Irányítástechnika I tárgykörébe tartozó mérnöki ismeretek megszerzésében.

2. KOMBINÁCIÓS HÁLÓZATOK ÉS TERVEZÉSÜK

A mérnöki gyakorlatban számos olyan probléma, objektum létezik, amelynek két értéke van. Ilyenek a logikai állítások, ítéletek, de ilyenek a kapcsolók is, amelyeknek nyitott vagy zárt állapota lehet. Az Irányítástechnika I. tantárgy célja olyan elméleti háttér, módszerek és technikák megismertetése, amelyek segítségével az ilyen rendszerek kezelhetők, ilyen típusú rendszerek tervezhetők, megvalósíthatók.
A kétértékű rendszerek elméleti hátterét a Boole-algebra, illetve a logikai függvények adják. A jegyzet további részében ezt az elméleti hátteret nem matematikai módszerekkel, hanem elsősorban mérnöki megközelítéssel ismertetjük, elősegítve a gyakorlati alkalmazást.

2.1. Logikai függvények

A logikai függvények olyan matematikai leképezések, amelyek képhalmaza, vagy más néven értékkészlete logikai értékekből áll. A logikai értékeket a továbbiakban a 0 és 1 jelekkel írjuk le. A logikai függvények egy részhalmazát alkotják azok a függvények, amelyeknek értelmezési tartományát is a logikai értékek alkotják. A továbbiakban a logikai függvényeknek ezzel a részével foglalkozunk, logikai függvény alatt a továbbiakban olyan matematikai leképezéseket értünk, amelyek a 0 és 1 számokból álló véges sorozatokhoz rendelik a 0 vagy 1 számot. Egy logikai függvény eszerint olyan n változós függvény, amelynek független változói (értelmezési tartomány) a $\{0,1\}$ halmazból vehetnek fel értéket, a függő változók (függvényértékek vagy értékkészlet) pedig szintén a $\{0,1\}$ halmazból valók. Az 1 értékre gyakran mint az igaz, a 0 értékre mint a hamis hivatkoznak.
Formálisan, a $\{0,1\}^n$ Descartes-szorzat segítségével egy f függvény logikai, ha:

$$f : \{0,1\}^n \rightarrow \{0,1\}$$

A logikai függvény változóit a logikai függvény bemeneteinek, a függvény értékeket pedig a rendszer kimeneteinek is nevezzük. Egy adott időpillanatban fennálló független változó értékeket *bemeneti kombinációnak*, a függvényértékeket pedig *kimeneti kombinációnak* is nevezzük. Ilyen felfogásban a logikai függvények olyan rendszerek, amelyek egy adott időpillanatban fennálló bemeneti kombináció hatására egy meghatározott kimeneti kombinációt állítanak elő. A kimeneti kombináció meghatározása történhet

- kizárólag a bemeneti kombináció aktuális értékei alapján – ekkor *kombinációs hálózatról* beszélünk, vagy
- a pillanatnyi és a korábban fennállt bemeneti kombinációk alapján – ekkor *sorrendi vagy szekvenciális hálózatról* beszélhetünk.

A kimeneti kombináció és a bemeneti kombináció közötti összefüggést a kétértékű Boole-algebra segítségével írhatjuk fel. A Boole-algebrában három alapműveletet értelmezünk:

- negáció (jelölése felülvonás, pl. $\overline{A}$; a hosszú felülvonással jelölt negáció zárójelet is helyettesít, azaz $\overline{AB} \neq \overline{A}\,\overline{B}$),
- logikai ÉS művelet (jelölése: ·, amelyet betűvel jelölt változók esetén elhagyhatunk, pl. $A \cdot B$ vagy AB ugyanazt jelöli),
- logikai VAGY művelet (jelölése +, pl. $A + B$).

Megjegyezzük, hogy szokás a logikai ÉS műveletre mint logikai szorzásra, a logikai VAGY műveletre pedig mint logikai összeadásra hivatkozni.

A negáció egyoperandusos művelet, eredményeként az adott változó értéket vált, azaz

$$\overline{0}=1,\ \overline{1}=0.$$

A változó kétszeres negálása visszaadja a változó eredeti értékét:

$$\overline{\overline{0}}=0,\ \overline{\overline{1}}=1.$$

Két változó ÉS kapcsolata kétoperandusos művelet, a művelet eredménye akkor 1 (igaz), ha mindkét operandus értéke 1. Két változó VAGY kapcsolata szintén kétoperandusos művelet, a művelet eredménye akkor 1, ha az operandusok közül legalább az egyik 1 értékű. Formálisan:

$$0\cdot 0=0\cdot 1=1\cdot 0=0,\ 1\cdot 1=1,$$

$$0+1=1+0=1+1=1,\ 0+0=0,$$

Általánosabban a következőképpen írhatjuk fel a fenti azonosságokat:

$$A\cdot 0\equiv 0,\ A+0\equiv A$$

$$A\cdot 1\equiv A,\ A+1\equiv 1,$$

$$A\cdot\overline{A}\equiv 0,\ A+\overline{A}\equiv 1,$$

$$\overline{\overline{A}}\equiv A.$$

A Boole-algebrában a műveleti sorrend tekintetében a negáció a legmagasabb precedenciájú (hierarchiájú) művelet, amelyet a logikai ÉS, majd a logikai VAGY művelet követ. Az alapértelmezett műveleti sorrendet természetesen zárójelezéssel módosíthatjuk.
A műveletek, illetve általában logikai függvények megadására általánosan elterjedt módszer az igazságtáblázat. Az igazságtáblázatban soronként feltüntetjük az összes lehetséges bemeneti kombinációt, és mindegyikhez megadjuk az adott kombinációhoz tartozó kimeneteket. A negáció igazságtáblája az alábbiak szerint írható fel:

A	$\overline{A}$
0	1
1	0

Az ÉS és a VAGY műveletek igazságtáblája a következő:

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

A Boole-algebrában mind az ÉS, mind a VAGY művelet kommutatív, azaz felcserélhető:

$$A + B \equiv B + A\,,$$

$$A \cdot B \equiv B \cdot A\,,$$

és asszociatív, azaz csoportosítható:

$$A+(B+C) \equiv A+B+C\,,$$

$$A \cdot (B \cdot C) \equiv A \cdot B \cdot C\,.$$

Igazak továbbá a következő disztributív tulajdonságok is:

$$A \cdot (B+C) \equiv A \cdot B + A \cdot C$$

$$A + BC \equiv (A+B) \cdot (A+C)$$

Ez utóbbi bizonyítása:

$$(A+B) \cdot (A+C) = AA + AB + AC + BC = A(1+B+C) + BC = A + BC$$

A fentiekből következnek az alábbi ún. elnyelési tulajdonságok:

$A + AB + ABC + ... = A$, mert $A \cdot (1 + B + BC...) = A$, illetve

$A(A+B)(A+B+C) = A$, mert $A(A+B) = AA + AB = A(1+B) = A$.

Az előzőeken felül nagyon lényeges, az algebrai átalakítások során gyakran felhasznált azonosságok az ún. De Morgan-azonosságok, amelyek két változó esetén a következő alakban írhatók fel:

$$\overline{A+B} \equiv \overline{A} \cdot \overline{B} \text{ és } \overline{A \cdot B} \equiv \overline{A} + \overline{B}\,.$$

A De Morgan azonosságok több változóra is igazak, így:

$$\overline{A+B+C} \equiv \overline{A} \cdot \overline{B} \cdot \overline{C} \text{ és } \overline{A \cdot B \cdot C} \equiv \overline{A} + \overline{B} + \overline{C}\,.$$

2.2. Logikai függvények megadása

Az n változós logikai függvények száma 2^{2^n}, hiszen az n változó 2^n darab lehetséges értékének mindegyikéhez két értéket rendelhetünk. Ilyen módon például kétváltozós logikai függvényből összesen 16 darab létezik (ld. 1.1. táblázat).

A	B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

1.1. táblázat: Kétváltozós logikai függvények

A fenti függvények egy része triviálisan értelmezhető:

- az f_0 függvény azonosan 0, hasonlóképpen az f_{15} függvény azonosan 1,
- az f_1 függvény az A és B változók ÉS kapcsolata (AB), míg az f_7 függvény az A és B változók VAGY kapcsolata ($A+B$).

Figyeljük meg, hogy $f_3 = A$, $f_{12} = \overline{A}$, $f_5 = B$ és $f_{10} = \overline{B}$. Érdemes továbbá figyelmet szentelnünk az f_6 és az f_9 függvényeknek. Vegyük észre, hogy az f_6 függvény értéke akkor 1, ha a két változó értéke különböző, az f_9 függvény értéke pedig akkor 1, ha a két változó azonos értékű. Az f_6 függvényt szokás *antivalenciának*, az f_9 függvényt pedig *ekvivalenciának* nevezni. Ezt a két függvényt akár műveletként is értelmezhetjük, sőt szokásos jelölésük is van:

- antivalencia: $\oplus$, pl. $A \oplus B$,
- ekvivalencia: $\odot$, pl. $A \odot B$.

Az antivalencia műveletet szokás kizáró vagy műveletnek is nevezni, hiszen ennek eredménye csak akkor lesz logikai 1, ha egy és csak egy változó értéke 1. Innen származik az antivalencia angol rövidítése: XOR. Az antivalencia és az ekvivalencia összetett műveletek, felírhatók a három alapvető művelet kombinációjaként is:

$$A \oplus B = A\overline{B} + \overline{A}B,$$

$$A \odot B = \overline{A}\,\overline{B} + AB$$

A logikai függvények igazságtáblázattal történő megadásának módját már ismerjük. Egy másik lehetséges megoldás az algebrai alakban történő megadás. Példaként tekintsük az 1.1. táblázat egy általános oszlopát:

A	B	f_{11}
0	0	1
0	1	0
1	0	1
1	1	1

Ez a függvény általános algebrai alakban úgy írható fel, hogy logikai VAGY kapcsolattal felírjuk azokat a bemeneti kombinációkat, amelyek esetében a függvény értéke 1:

$$f_{11} = \overline{A}\,\overline{B} + A\overline{B} + AB.$$

Figyeljük meg, hogy ha egy bemeneti változó egy adott kombinációban 1 értékű, akkor a változó *ponáltja* szerepel a logikai alakban, míg ha a bemeneti változó értéke 0, akkor az illető változó *negáltja* szerepel. A fenti felírást tovább alakíthatjuk felhasználva a korábban megismert algebrai azonosságokat:

$$f_{11} = \overline{A}\,\overline{B} + A\overline{B} + AB = \left(\overline{A} + A\right)\overline{B} + AB = \overline{B} + AB\,.$$

Az előbbi eljárást felhasználva bármilyen igazságtáblázattal megadott logikai függvényt át tudunk írni algebrai alakba. Példaként vegyük az alábbi, háromváltozós függvényt:

A	B	C	F_1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

A fenti igazságtáblázat szerinti F_1 függvény algebrai alakja a következőképpen néz ki:

$$F_1(A,B,C) = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C}\,.$$

Felhasználva a logikai függvények azonosságait, tovább alakíthatjuk a fenti formát:

$$\overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C} = \overline{A}B(C + \overline{C}) + A\overline{C}(B + \overline{B}) = \overline{A}B + A\overline{C}\,.$$

2.3. Logikai függvények kanonikus alakjai

Mint láttuk, egy logikai függvény többféle algebrai alakban is megadható. Az egyértelműség érdekében célszerű olyan felírási módot alkalmazni, amely esetén egy adott függvény csak egyféleképpen írható le, továbbá ha két függvény különböző, akkor a leírt alakjuk is biztosan különbözik. Az ilyen leírási módokat a függvény kanonikus vagy normál alakjának nevezzük. Két ilyen kanonikus alakot tárgyalunk:

- diszjunktív kanonikus alak (minterm) és
- konjunktív kanonikus alak (maxterm).

2.3.1. Diszjunktív kanonikus alak (minterm)

A diszjunktív kanonikus vagy minterm alak tárgyalásához vegyük példaként az előző fejezetben tárgyalt F_1 függvényt. Egészítsük ki az igazságtáblázatot egy oszloppal.

	A (2^2)	B (2^1)	C (2^0)	F_1
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Az A, B és C változókat kettes számrendszerbeli helyiértékeknek tekintve (rendre 2^2, 2^1 és 2^0) az A, B és C értékekből kettes számrendszerbeli számot képzünk. Ennek a kettes számrendszerbeli számnak a tízes számrendszerbeli (decimális) értéke szerepel az első oszlopban. Például: $1|0|1 \rightarrow 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 5$.

Az így kapott decimális értékkel egyértelműen tudunk hivatkozni az igazságtáblázat egy-egy sorára. A háromváltozós F_1 függvény 5 decimális értékű sorára a következőképpen hivatkozunk: m_5^3, ahol a 3 azt mutatja, hogy függvény háromváltozós, az 5 pedig a decimális 5 értékre utal. Egy-egy ilyen sort szokás *term*nek vagy *minterm*nek is nevezni. E jelölésmód felhasználásával az F_1 függvény a következőképpen írható fel:

$$F_1 = m_2^3 + m_3^3 + m_4^3 + m_6^3 .$$

Tulajdonképpen felsoroljuk, pontosabban logikai VAGY kapcsolatba hozzuk azokat a termeket (bemeneti kombinációkat), amelyek esetén a függvény értéke 1. Létezik egy hasonlóan tömör leírási forma a fentiekre, amely az F_1 függvény esetén a következőképpen néz ki:

$$F_1 = \sum^3 (2,3,4,6).$$

A Σ jel utal az alak minterm voltára, a felette lévő 3 mutatja, hogy a függvény háromváltozós, a zárójelben lévő számok pedig azon termek decimális értékei, ahol a függvényérték 1.

A minterm alak általános jellemzői összefoglalva a következők:

- a minterm alak logikai szorzatok logikai összege,
- mindegyik szorzatban az összes független változó szerepel ponált vagy negált alakban,
- mindegyik szorzat olyan független-változó kombinációt képvisel, amelyhez tartozó függvényérték 1.

2.3.2. Konjunktív kanonikus alak (maxterm)

A konjunktív kanonikus vagy maxterm alak megismeréséhez szintén az F_1 függvényből induljunk ki. Írjuk fel az F_1 függvény negáltját, azaz azokat a termeket, amelyek esetében a függvényérték 0:

$$\overline{F_1(A,B,C)} = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + A\overline{B}C + ABC\ .$$

Negáljuk még egyszer a függvényt, majd a De Morgan-azonosság felhasználásával az alábbi alakra jutunk:

$$\overline{\overline{F_1(A,B,C)}} = \overline{\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + A\overline{B}C + ABC} =$$
$$= (A+B+C)\cdot(A+B+\overline{C})\cdot(\overline{A}+B+\overline{C})\cdot(\overline{A}+\overline{B}+\overline{C}).$$

Láthatjuk, hogy a fenti alak logikai összegek logikai szorzatából áll, mégpedig oly módon, hogy minden egyes tényezőben szerepel minden logikai változó, ponált vagy negált értékkel. Ha ezeket a ponált vagy negált értékeket kettes számrendszerbeli számjegyeknek tekintjük, és a minterm alakéhoz hasonlóan jelöljük, akkor a következőt kapjuk:

$$\overline{\overline{F_1(A,B,C)}} = M_7^3 M_6^3 M_2^3 M_0^3 = F_1(A,B,C).$$

Ezzel megkaptuk az F_1 függvény maxterm alakját.
A maxterm alak jellemzői a következők:

- a maxterm alak logikai összegek logikai szorzata,
- mindegyik összegben az összes független változó szerepel ponált vagy negált alakban,
- mindegyik összeg olyan független-változó kombinációt képvisel, amelyhez tartozó függvényérték 0.

A maxterm és a minterm alak közti áttérés másképpen is leírható:

$$F_1(A,B,C) = m_2^3 + m_3^3 + m_4^3 + m_6^3$$

$$\overline{F_1(A,B,C)} = m_0^3 + m_1^3 + m_5^3 + m_7^3$$

$$\overline{\overline{F_1(A,B,C)}} = \overline{m_0^3}\cdot\overline{m_1^3}\cdot\overline{m_5^3}\cdot\overline{m_7^3}\ .$$

Az $\overline{m_i^n} = M_{2^n-1-i}^n$ (n a változók száma) helyettesítést elvégezve a következőt kapjuk:

$$\overline{\overline{F_1(A,B,C)}} = M_7^3 M_6^3 M_2^3 M_0^3 = F_1(A,B,C)\ .$$

A minterm alakhoz hasonlóan a maxterm alaknak is van kompakt írásmódja, amely a fenti függvényre a következő:

$$F_1 = \prod^3(0,2,6,7)$$

A képlet értelmezése is hasonló: a Π utal a maxterm alakra, a 3 pedig a változók számát jelöli.

2.4. Nem teljesen határozott logikai függvények

Logikai feladatokban előfordul, hogy bizonyos bemeneti kombinációk fennállásával nem kell számolnunk működés közben, vagy nem fontos esetükben definiálni a kimeneti kombinációt. Ilyen esetben a kimenet értéke természetesen nem egy „harmadik" állapotba kerül, értéke ekkor is 0 vagy 1, csak nem lényeges a megoldandó feladat szempontjából.
Az ilyen nem határozott kimenetet szokás *közömbös* kimenetnek, angolul *don't care* kimenetnek is nevezni. A közömbös kimeneteket az igazságtáblában kihúzással (–) jelöljük, mint például az alábbi F_2 függvény igazságtáblájában:

	A	B	C	F_2
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	–
5	1	0	1	0
6	1	1	0	–
7	1	1	1	0

A függvény algebrai alakjában is jelölhetjük a közömbös mintermeket:

$$F_2 = \overline{A}B\overline{C} + \overline{A}BC + \left(A\overline{B}\overline{C}\right) + \left(AB\overline{C}\right),$$

illetve a minterm alakban is szokás ezt jelölni:

$$F_2 = m_2^3 + m_3^3 + \left(m_4^3\right) + \left(m_6^3\right), \text{ vagy}$$

$$F_2 = \sum^3 (2,3) + (4,6).$$

A fenti logikai feladat kielégítésére összesen négy lehetséges függvény létezik, amennyiben a két közömbös kimenetű bemeneti kombinációhoz összesen négyféleképpen rendelhetünk 1-est vagy 0-t.

A	B	C	F_{2a}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

A	B	C	F_{2b}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

A	B	C	F_{2c}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

A	B	C	F_{2d}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

A későbbiekben, a logikai függvények minimalizálása során látni fogjuk, hogy a legtöbb esetben a logikai függvények lehető legegyszerűbb megvalósítására törekszünk. A nem teljesen határozott logikai függvények lehetséges megoldásai közül ezért a legegyszerűbbet szoktuk választani. Ennek módjával a logikai függvények minimalizálása során fogunk megismerkedni.

2.5. Logikai függvények megvalósítása

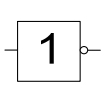
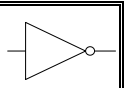
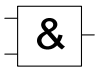
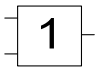
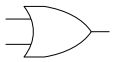
A logikai függvények által meghatározott mérnöki feladatok többféle technológiával is megoldhatók. A következő fejezetekben a logikai kapukkal (tipikusan integrált áramköri elemekkel) való megvalósítást tárgyaljuk részletesen, és e jegyzet további részében is elsősorban a logikai kapukkal történő megvalósításra koncentrálunk. Bemutatjuk továbbá a logikai függvények jelfogókkal történő megvalósításának alapjait is. E technológiákon kívül még további lehetőségek állnak rendelkezésre a pneumatikus megvalósítástól kezdve a programozható logikai vezérlőkkel, illetve a mikrokontrollerekkel és számítógéppel történő megvalósításig.

2.5.1. Logikai függvények megvalósítása logikai kapukkal

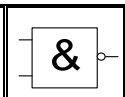
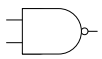
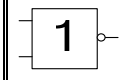
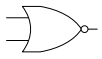
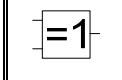
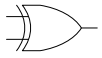
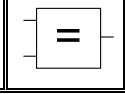
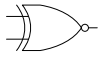
A logikai függvények logikai kapukkal, logikai kapcsolásként történő megvalósítását felfoghatjuk a logikai függvény egy speciális meghatározási módjának is. A logikai kapcsolásban ugyanolyan egyértelműen meghatározza a logikai függvényt, mint az igazságtáblázat, vagy az algebrai alak. A logikai kapcsolásban ún. logikai kapukat alkalmazunk; a logikai kapuk egy-egy logikai műveletnek felelnek meg. Ily módon beszélhetünk:

- negátor kapuról,
- ÉS-kapuról, valamint
- VAGY-kapuról.

Gyakran használjuk e kapuk angol megnevezését is, így gyakran emlegetjük AND- és OR-kapuként az ÉS-, illetve a VAGY-kaput. A kapuk rajzi megjelenítésére többféle konvenció létezik:

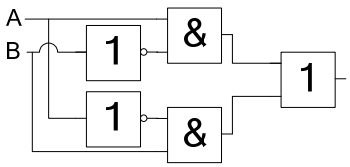
Negátor (NOT)		
ÉS (AND)		
VAGY (OR)		

A jegyzet további részeiben az első oszlopban lévő jelölésmódot fogjuk alkalmazni.
Az Boole-algebra alapműveleteihez tartozó logikai kapukon kívül az összetett műveleteknek vannak logikai kapui, saját jelöléssel. Ilyen gyakran használt logikai kapu a NEM-ÉS, angolul NAND, a NEM-VAGY, angolul NOR. Gyakran használjuk továbbá a korábban már tárgyalt antivalencia és ekvivalencia műveleteket logikai kapuként is. Általános jelölési konvenció, hogy a kapu kimenetéhez rajzolt karika negációt jelöl. Ebben különbözik az AND- és a NAND-kapu jelölése.

NAND		
NOR		
Antivalencia (XOR)		
Ekvivalencia		

A logikai kapcsolás előállításához az algebrai alakban lévő műveleteket képezzük le, természetesen a megfelelő műveleti sorrend betartásával. Példaként tekintsük az antivalencia függvényt, és vázoljuk fel az antivalencia logikai kapcsolását az elemi műveletek logikai kapuival.

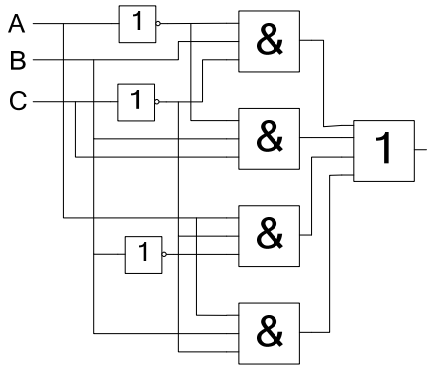
$$A \oplus B = A\overline{B} + \overline{A}B$$



Azt már tudjuk, hogy egy logikai függvénynek több különböző algebrai felírása is lehetséges. Ha egy F_1 függvényt

$$F_1 = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C}$$

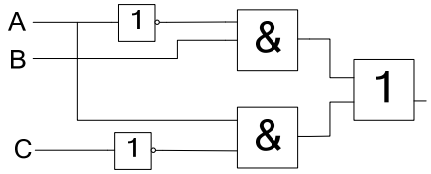
alakban írunk fel, akkor a logikai kapcsolás a következő lesz:



Amennyiben viszont azonos algebrai átalakításokkal egyszerűsítjük és a következő alakra hozzuk:

$$F_1 = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C} = \overline{A}B(C + \overline{C}) + A\overline{C}(B + \overline{B}) = \overline{A}B + A\overline{C}$$

akkor nemcsak az algebrai alak, hanem a logikai kapcsolás sokkal egyszerűbb lesz:



A logikai függvények ilyen értelmű egyszerűsítése, minimalizálása alapvető mérnöki feladat, ugyanis törekszünk a logikai függvény legolcsóbb, azaz legkevesebb logikai kaput tartalmazó megvalósítására. Ennek módszereivel foglalkozunk a 2.6. fejezetben.

2.5.2. Logikai függvények megvalósítása jelfogókkal

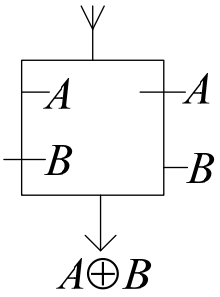
A jelfogó vagy relé elektromos áram mágneses hatására elektromos érintkezőket működtető kapcsolóelem. A jelfogók érintkezőinek megfelelő soros, illetve párhuzamos kapcsolásával szintén kialakíthatók logikai függvények. A jelfogóknak tipikusan két érintkező fajtája van (típustól függő számban):

- a nyugalmi érintkezők a jelfogó alaphelyzetében zárnak, a jelfogó működésekor (meghúzásakor) szakítanak, míg
- a munkaérintkezők a jelfogó alaphelyzetében szakítanak, húzott helyzetében pedig zárnak.

A jelfogós megvalósítás során egy-egy jelfogó egy-egy logikai változót reprezentál; a jelfogó ejtett helyzete a logikai változó 0 állapotának, a meghúzott helyzet a logikai változó 1 állapotának felel meg. A jelfogó munkaérintkezői a logikai változó ponáltjaként használhatók fel, a jelfogó nyugalmi érintkezői pedig a logikai változó negált állapotaként. A két alapműveletet a logikai szorzást és a logikai összeadást a jelfogók érintkezőinek soros, illetve párhuzamos kapcsolásával valósíthatjuk meg.

A jelfogós kapcsolások jelölésére többféle szimbolika létezik. A továbbiakban a vasúti biztosítóberendezések jelfogós kapcsolásainak is alkalmazott jelölésrendszert ismertetjük. Eszerint a jelfogó tekercsét vagy csévéjét (amely a mágneses hatást létrehozza), egy körrel jelöljük az áramkörben, a jelfogó érintkezőit pedig a vezeték merőleges áthúzásával (munkaérintkezők esetén), illetve érintkező merőleges vonallal (nyugalmi érintkezők esetén) jelöljük. Az *A* és *B*

logikai változók antivalencia kapcsolását ($A \oplus B = A\overline{B} + \overline{A}B$) jelfogók érintkezőkkel az alábbiak szerint ábrázolhatjuk:



Láthatjuk, hogy amennyiben az adott változó negáltja szerepel az algebrai alakban, úgy az adott változót reprezentáló jelfogó nyugalmi érintkezőjét kapcsoljuk, amennyiben változó ponált formája szerepel, akkor munkaérintkezőt használunk. Az is látszik, hogy a logikai ÉS műveletnek az érintkezők soros, a logikai VAGY műveletnek a párhuzamos kapcsolás felel meg.
Érdemes a jelfogós kapcsolásokkal kapcsolatban néhány megjegyzést tennünk. Mivel a jelfogó véges számú érintkezővel rendelkezik (még a nagy érintkezőszámú vasúti biztosítóberendezési jelfogók sem tartalmaznak 20 érintkezőnél többet), ezért a jelfogós kapcsolások egyszerűsítésekor arra törekszünk, hogy minél kevesebb *változó* szerepeljen a függvény algebrai alakjában. Szemben a logikai kapukkal történő megvalósítással, ahol alapvetően a lehető legkevesebb műveletre törekszünk. A logikai kapukkal történő megvalósítás esetében egy-egy változó „értékét” annyiszor használjuk fel, ahányszor akarjuk (természetesen az elektrotechnikai méretezések figyelembe vételével). Ezzel szemben a jelfogóknál szinte tetszőleges ÉS és VAGY művelet végezhetünk, hiszen azok megvalósítása „csak” vezetéket igényel.
Érdekes továbbá összevetni a két típusú megvalósítást olyan szempontból is, hogy amíg a logikai kapukkal történő megvalósítás esetén a kapcsolás csomópontjaiban a műveletek állnak (maguk a logikai kapuk), a kapcsolás struktúráját pedig az egyes változók vezetékezése adja, addig a jelfogó érintkezőkkel történő megvalósítás esetén éppen a fordítottja történik: a logikai műveletek határozzák meg a kapcsolás struktúráját, azaz a soros és párhuzamos ágakat.

2.6. Logikai függvények egyszerűsítése

Korábbi példák alapján már láttuk, hogy megfelelő algebrai átalakításokkal egy algebrai alakban megadott logikai függvény egyszerűbb alakra hozható. Minél egyszerűbb egy kombinációs hálózat logikai függvénye, azaz minél kevesebb a benne szereplő művelet és változó, annál kevesebb áramköri elemmel tudjuk azt megvalósítani. A célszerűen alkalmazandó átalakítások kiválasztása azonban nem egy szisztematikus tervezési eljárás, hatékonysága nagyban függ a tervezést végző gyakorlatától. Célszerű tehát a minimalizálásra valamilyen szisztematikus eljárást találni. A következőkben ezzel foglalkozunk.

2.6.1. Algebrai egyszerűsítés

Elsőként ismételjük át a minterm definícióját: a minterm olyan speciális elemi logikai szorzat (ÉS) függvény, amely valamennyi változót tartalmazza ponált vagy negált formában. Vezes-

sük a *szomszédos minterm* fogalmát: a szomszédos mintermek csak egy helyiértéken térnek el egymástól, azaz az egyik változó az egyik mintermben ponált, a másikban negált értékkel szerepel, a többi változó mindkettőben azonos módon. Az előbbiek értelmében egy *n* változós logikai függvény egy mintermjének *n* darab szomszédos mintermje lehet, hiszen *n* helyértéken különbözhetnek egy változóban. Példaként tekintsük a korábbi F_1 függvény minterm alakját:

$$F_1 = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C}$$

A fenti alakban az $\overline{A}B\overline{C}$ és az $\overline{A}BC$ mintermek szomszédosak, hiszen csak a *C* változó értékében különböznek. Hasonlóképpen szomszédosak az $A\overline{B}\overline{C}$ és $AB\overline{C}$ mintermek, hiszen csak a *B* változó értékében térnek el egymástól. A szomszédos mintermek az asszociatív tulajdonság, illetve az $X + \overline{X} \equiv 1$ azonosság felhasználásával mindig egyszerűsíthetők, így kaphatjuk meg az előző függvény egyszerűbb alakját is:

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

A szomszédos mintermek megtalálása azonban nem mindig kézenfekvő, továbbá az egyes mintermek többféleképpen rendezhetők párba a fenti egyszerűsítési lehetőség kihasználásához, tehát az egyszerűsítés során több megoldást is elemezni kellene.

2.6.2. A Karnaugh-tábla

A szomszédos mintermek felismeréséhez nagy könnyebbséget ad, ha a függvényt az ún. *Karnaugh-táblában* ábrázoljuk. A Karnaugh-tábla néhány egyszerű lépéssel származtatható az igazságtáblázatból. Egy háromváltozós függvény igazságtáblája a következőképpen írható fel általánosan:

A	*B*	*C*	*F*
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

Alakítsuk át ezt a táblázatot úgy, hogy oszloponként a *B* és *C* változók lehetséges kombinációt tartalmazzák (figyeljünk a kombinációk sorrendjére!), a két sor pedig az *A* változó két lehetséges értékét:

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

Figyeljük meg, hogy ezzel az elrendezéssel a szomszédos mintermek egymás mellé kerültek, feltételezve azt, hogy a táblázat jobb és bal oldala képzeletben szomszédosak (más Karnaugh-tábla formánál, vagy négyváltozós Karnaugh-táblánál a fenti és a lenti sorok is szomszédosak egymással.). Ha az m_3 mintermet vesszük például ($\overline{A}BC$), akkor annak három szomszédos mintermje az ABC, az $\overline{A}\overline{B}C$ és az $\overline{A}B\overline{C}$, azaz rendre az m_7, az m_1 és az m_2 jelölésű mintermek. A Karnaugh táblázat fejlécezését rendszerint el szoktuk hagyni, és csak „peremezni" szoktuk a táblát: a táblázat mellé húzott vonal azt jelöli, hogy a vonal alatt/mellett/fölött lévő mintermek esetében az adott változó értéke 1 (a többi helyen az adott változó értéke 0), például a következőképpen

A \ B			$\overline{C}$	
	m_0	m_1	m_3	m_2
A	m_4	m_5	m_7	m_6
			C	

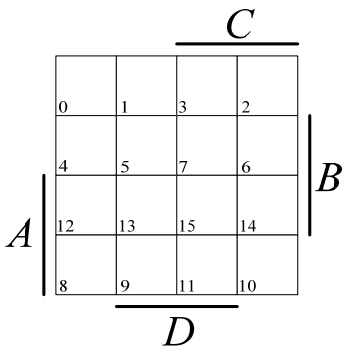
A Karnaugh-táblában a függvényt úgy jelöljük, hogy az adott minterm pozíciójába 1-et írunk, ha ott a függvény értéke 1, ha pedig a függvény értéke 0, akkor azt a cellát üresen hagyjuk, vagy 0-t írunk bele. (A cella üresen hagyása célszerűbb, mert az 1-esek jobban kitűnnek, és ennek később jelentősége lesz.) A fenti konvencióknak megfelelő jelöléssel az F_1 függvény Karnaugh-táblája a következő:

A \ B			$\overline{C}$	
			1	1
A	1			1
			C	

Amennyiben a függvény nem meghatározott (közömbös) az adott helyen, akkor azt kihúzással (–) jelöljük. A nem teljesen határozott F_2 függvény a Karnaugh-táblája a következő:

A \ B			$\overline{C}$	
			1	1
A	-			-
			C	

Korábban ugyan nem mutattunk példát négyváltozós függvényre, de természetesen léteznek négy-, sőt több változós logikai függvények is. A négyváltozós logikai függvények Karnaugh-táblája a következőképpen néz ki:

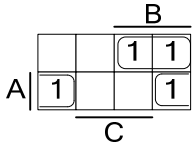


Mint látható, egy négyváltozós függvény esetében egy adott mintermhez négy szomszédos minterm tartozik (ne felejtsük el, hogy a Karnaugh-tábla „szélei” összeérnek), így az m_8 mintermnek szomszédja az m_{12}, az m_9, az m_{10} és az m_0 minterm. Az ábrán az is látszik, hogy a mintermek számát (azaz tulajdonképpen a bennük szereplő logikai változók által alkotott kettes számrendszerbeli szám decimális értékét) szokás az egyes cellák bal alsó sarkában is ábrázolni.

Itt jegyezzük meg, hogy a logikai változók jelölésére szokásos, de nem szigorúan rögzített konvenció, hogy a legnagyobb helyiértékű változót *A*-val, a következőt *B*-vel stb. jelöljük. E konvenciót követve természetesen az *A* jelű változó egy háromváltozós függvényben 2^2 helyiértéket képvisel, míg egy négyváltozós függvényben 2^3 helyiértéket. Szintén szokásos, de nem szigorúan rögzített konvenció, hogy a változók peremezését a bal oldalon kezdjük, majd rendre a jobbra, fent és lent folytatjuk (háromváltozós függvény esetében balra-fent-lent). Más sorrend is alkalmazható, ez azonban a mintermek számozásának változásával is jár, feltéve, hogy az *A* jelű változó még mindig a legnagyobb helyiértéket képviseli. A sok lehetséges és helyes Karnaugh-tábla elrendezésben közös, hogy a szomszédos mintermek egymás mellé kerülnek.

2.6.3. Logikai függvény egyszerűsítése Karnaugh-táblával

Nézzük meg, hogyan használhatjuk fel a Karnaugh-táblát az összevonások során. Vegyük példaként a már ismert F_1 függvényt. A Karnaugh-táblás elrendezésnél, mint már említettük, egymás mellé kerülnek a szomszédos mintermek, amelyekről tudjuk, hogy algebrai egyszerűsítés útján elhagyható belőlük egy-egy változó. Jelöljük összevonásokkal a szomszédos mintermeket, a következőképpen (emlékezzünk rá, hogy a Karnaugh-tábla szélei „összeérnek”).



Az algebrai alak felírásánál alkalmazzuk úgy az egyszerűsítési lehetőséget, hogy egy-egy összevonás helyett olyan algebrai logikai szorzatot írunk le, amelyben kizárólag azok a változók szerepelnek, amelyek az összevonás részét képező mintermekben közösek. Így a fenti függvény esetében írhatjuk a következőt is:

$$F_1 = \overline{A}B + A\overline{C} .$$

A fenti logikai összeg első tagját a következőképpen kapjuk meg:

$$\overline{A}BC + \overline{A}B\overline{C} = \overline{A}B(C + \overline{C}) = \overline{A}B\,.$$

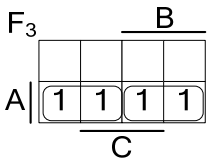
A második tag pedig a következőképpen adódik:

$$A\overline{B}\,\overline{C} + AB\overline{C} = A\overline{C}(B + \overline{B}) = A\overline{C}\,.$$

Az algebrai egyszerűsítést azonban nem kell elvégeznünk, elég, ha egy-egy összevonáshoz azoknak a változóknak az ÉS-kapcsolatát írjuk le, amelyek az összevonásban közösek, az összevonásban változó értékkel szereplő változókat egyszerűen elhagyjuk. Abban, hogy egy változót ponált vagy negált értékkel kell-e figyelembe vennünk, könnyen eldönthetjük a Karnaugh-tábla peremezéséből: a tábla mellé húzott vonal jelzi az adott változó ponált értékét.
A fenti egyszerű példát általánosítva és további lehetőségekkel kiegészítve a grafikus minimalizálás szisztematikus eljárását a következő lépésekkel határozhatjuk meg:

1. A szomszédos mintermek megkeresése, párba válogatása (Karnaugh-táblán grafikusan ábrázolva).
2. A lehetséges összevonások után a kiadódó termek közül szintén meg kell keresni a szomszédosakat.
3. Az eljárást addig kell folytatni, amíg a logikai függvény olyan szorzatok összege nem lesz, amelyekből már egyetlen változó sem hagyható el anélkül, hogy a logikai függvény meg nem változna. Az ilyen logikai összegekben szereplő logikai szorzatok a *prímimplikánsok*.

Nézzük meg az összevonások összevonásának lehetőségét egy másik példán. Vegyük az alábbi Karnaugh-táblával ábrázolt F_3 függvényt.



Az F_3 függvényre az összevonásokat első szinten alkalmazva a következő algebrai alakot kapjuk:

$$F_3 = A\overline{B} + AB\,.$$

Akár a fenti Karnaugh-táblát, akár az F_3 függvény algebrai alakját tekintjük, látható, hogy a két összevonás szomszédos, tehát akár az alsó sorban szereplő négy 1-est is összevonhatnánk. Az eredmény a következő lesz:

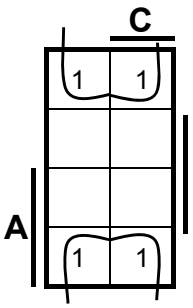
$$F_3 = A\overline{B} + AB = A(\overline{B} + B) = A\,.$$

A Karnaugh-tábla segítségével történő függvényegyszerűsítéshez a következő szabályokat kell betartanunk:

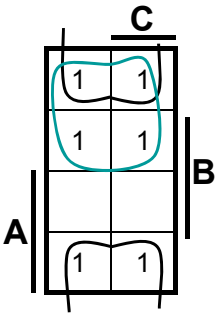
– Minden 1-est le kell fedni legalább egy huroknak, 0 nem kerülhet egyik hurokba sem.

- Mindig annyi 1-est lehet összevonni, amelyek száma megfelel 2 valamelyik egész hatványának (azaz kettőt, négyet, nyolcat stb.).
- Az összevonások alakja mindig téglalap kell legyen, ugyanis csak azok a mintermek szomszédosak egymással, de ahogy korábban is említettünk, az összevonás folytatódhat a tábla másik szélén.
- Minél több 1-est vonunk össze, annál több logikai változót hagyhatunk el a szorzatból (két 1-es összevonásakor 1 változót, négy 1-es összevonásakor 2 változót, nyolc 1-es összevonásakor 3 változót stb. hagyhatunk el.).
- Egyedülálló 1-es esetén egyszerűsítésre nincs mód, ekkor a teljes minterm felírásra kerül (egyes hurok) – egyetlen változót sem hagyhatunk el.
- Egy-egy Karnaugh-táblában szereplő 1-es akár több prímimplikánsban is szerepelhet, azaz a hurkok egymásba nyúlhatnak.
- Úgy kell minden 1-est lefedni, hogy ezt a lehető legkevesebb számú hurokkal tegyük, ezért a lehető legnagyobb hurkokat kell keresni.

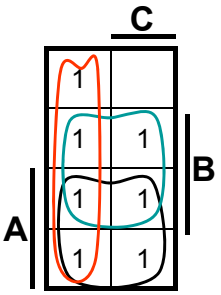
További magyarázatok helyett álljon itt néhány példa helyes összevonásokra. (A példatárban lévő feladatok további tanulságokkal szolgálnak.)



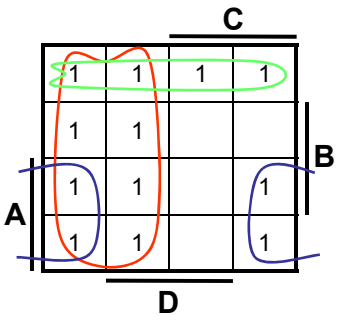
$F = \overline{B}$



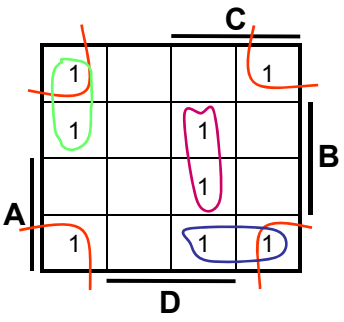
$F = \overline{A} + \overline{B}$



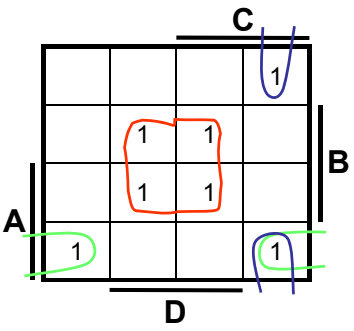
$F = A + B + \overline{C}$



$\overline{C} + \overline{A}\overline{B} + A\overline{D}$



$\overline{B}\overline{D} + \overline{A}C\overline{D} + BCD + A\overline{B}C$



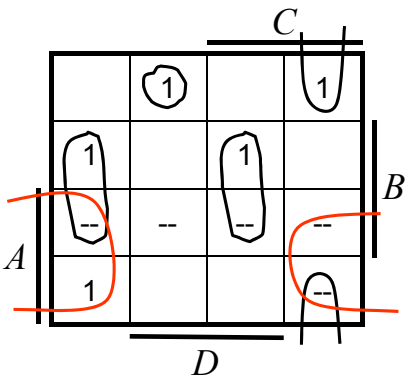
$BD + \overline{B}C\overline{D} + A\overline{B}\overline{D}$

2.6.4. Nem teljesen határozott függvény egyszerűsítése

A nem teljesen meghatározott kimeneteket is felhasználhatjuk a Karnaugh-táblás egyszerűsítés során. Ahogyan azt a 2.4. fejezetben bemutattuk, a nem teljesen határozott logikai függvény egyfajta tervezési szabadságot jelent a megvalósítás során. A 2.6.2. szakaszban

láttuk, hogy a közömbös kimeneteket is jelölhetjük a Karnaugh-táblán, mégpedig kihúzással (–). Ezek a helyek egyaránt viselkedhetnek 0-ként és 1-ként. A közömbös kimenetek figyelembevételével akkor kapjuk a legegyszerűbb megvalósítást, ha a közömbös kimeneteket a Karnaugh-táblában úgy használjuk fel, hogy a lehető legegyszerűbben fedjük le az 1-eseket. A közömbös bejegyzéseket nem kell lefedni, csupán arra használjuk őket, hogy a meghatározott 1-eseket a lehető legnagyobb összevonással fedjük le.

Példaként szolgáljon az alábbi Karnaugh-tábla:



Ennek a függvénynek a legegyszerűbb alakja a közömbös bejegyzések felhasználásával:

$$F = A\overline{D} + B\overline{C}\overline{D} + BCD + \overline{A}BCD\,.$$

2.7. Kombinációs hálózatok megvalósítási kérdései

2.7.1. Kétszintű és többszintű megvalósítás

A Karnaugh-táblás egyszerűsítés révén mindig eljuthatunk egy olyan alakhoz, amely szorzatok összegeként írható fel. Ha az ennek az alaknak megfelelő logikai kapcsolási rajzot előállítjuk, akkor azt látjuk, hogy a hálózat megvalósításában van egy sor ÉS-kapu, amelyek kimenetit egy VAGY-kapu kapcsolja össze (eltekintve az esetlegesen szükséges negátoroktól). Az ilyen fizikai kialakítást *kétszintű megvalósításnak* nevezzük. Az előzőekből következik, hogy *minden* kombinációs hálózat megvalósítható kétszintű logikai kapcsolással. Megjegyezzük, hogy maxterm megvalósítás esetén (összegek szorzata) szintén mindig realizálható kétszintű hálózat, csak egy sor VAGY-kapu eredményét kapcsolja össze egy ÉS-kapu.

Amennyiben más egyszerűsítési eljárást követünk, vagy az egyszerű szorzatok összege alakon további átalakítást (például kiemelést) végzünk, akkor az annak megfelelő megvalósítás *többszintű* lesz. Ez azzal is járhat, hogy a bemeneti jelek nem mindenirányban azonos számú kapun keresztül terjednek a kimenet felé – ennek a tranziens viselkedéseknél van szerepe.

2.7.2. Megvalósítás egyforma kaputípusokkal

Bármely kombinációs hálózat megvalósítható csak NOR vagy csak NAND kapuk felhasználásával is. Az ilyen megoldásoknak az az előnye, hogy az integrált áramkörök gyártóinak nem kell többféle kapu gyártástechnológiáját egyetlen chipen belül kombinálni. Az átalakítás a De-Morgan azonosságok alkalmazásával oldható meg. Felhasználjuk azt a tényt is, hogy negátort egy NOR vagy egy NAND kapu bemeneteinek összekötésével is meg lehet valósítani.
A csupa NOR kapus megvalósításhoz a legegyszerűbb szorzatok összege alakból induljunk ki. Az összeg minden tagját negáljuk kétszer, majd a „belső" negáció De Morgan-féle átalakításával változtassuk a szorzást összeadásra. Például:

$$AB + \overline{B}C = \overline{\overline{AB}} + \overline{\overline{\overline{B}C}} = \overline{\overline{A} + \overline{B}} + \overline{B + \overline{C}}\,.$$

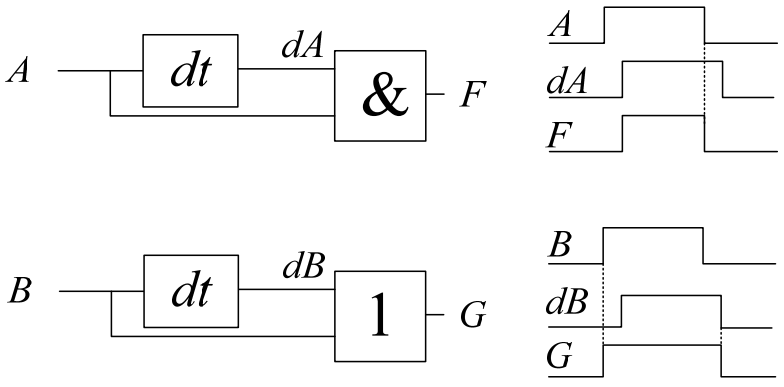
Csak NAND-kapukkal történő megvalósításkor ugyanabból az alakból célszerű kiindulni, majd a teljes függvényt negáljuk kétszer. Ezt követően a „belső" negáció De Morgan-féle átalakításával változtassuk a logikai összeadást szorzássá. Például:

$$AB + \overline{B}C = \overline{\overline{AB + \overline{B}C}} = \overline{\overline{AB} \cdot \overline{\overline{B}C}}\,.$$

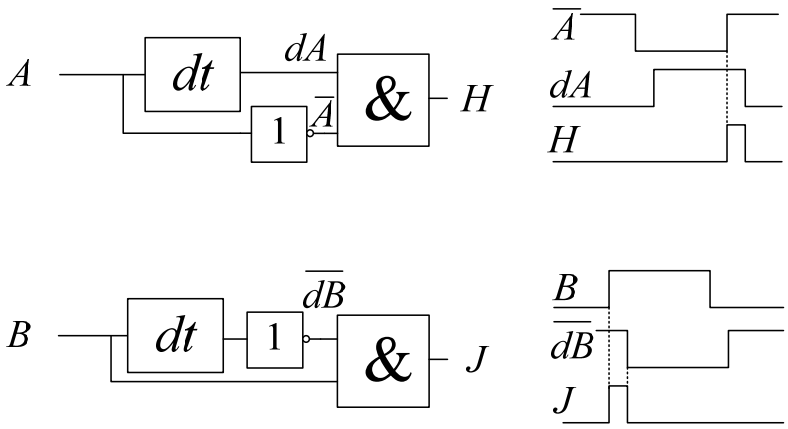
2.8. Hazárdjelenségek kombinációs hálózatokban

2.8.1. A jelterjedési idő

A valóságban a logikai kapuk nem ideálisan viselkednek. Az ideális működéstől való egyik lényeges eltérés, hogy a bemeneti jelek megváltozására nem azonnal reagálnak, hanem némi késleltetéssel. Ezt a késleltetést a kapu „megszólalási idejének", angolul pedig *propagation delay*-nek szokták nevezni. Mindez azt okozza, hogy a valóságban számolnunk kell a *jelterjedési idővel*. A késleltetés hatását a kapuk bemeneteinél és/vagy kimenetén modellezhetjük. Vizsgálatunkat kezdjük néhány egyszerű kapcsolással, amelyek mellett ábrázoltuk a jelek változását is az időben:



Nyilvánvaló, hogy az *F* kimenetnek azonosnak kellene lennie az *A* bemenettel, a *G* kimenetnek pedig a *B* bemenettel. Ehhez képest az *F* kimeneten egy kicsivel rövidebb, a *G* kimeneten egy kicsivel hosszabb impulzust kapunk a bemenethez képest.



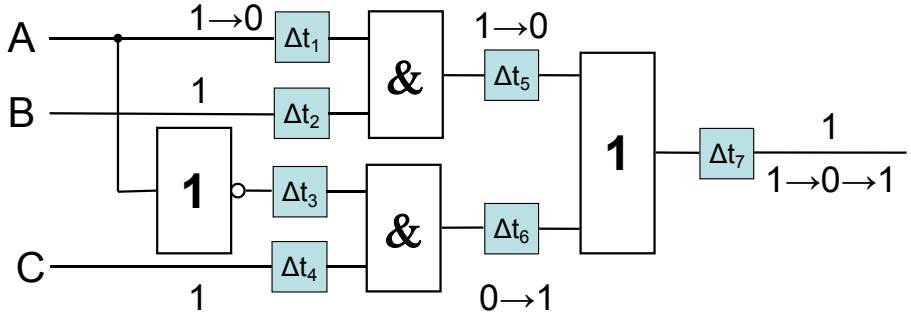
Ezekben az esetekben, ha nem lenne késleltetés, akkor az $X \cdot \overline{X} \equiv 0$ azonosság miatt a kimenet a bemenettől függetlenül 0 lenne. A *H* függvény esetében a jel hátsó élénél, a *J* függvény esetében pedig a jel első élénél egy-egy magas impulzus jelenik meg. Hasonló jelenséget tapasztalunk az $X + \overline{X}$ típusú kapcsolásoknál késleltetés esetén, annyi különbséggel, hogy ott alacsony impulzust tapasztalhatunk. Az impulzusokra természetesen nincs hatással az sem, ha az AND kapuk helyett NAND, az OR kapuk helyett pedig NOR kaput használunk, csupán az impulzusok alacsony vagy magas volta változik.

2.8.2. Statikus hazárd

Vizsgáljuk tovább a jelenséget és vegyünk egy összetettebb példát:

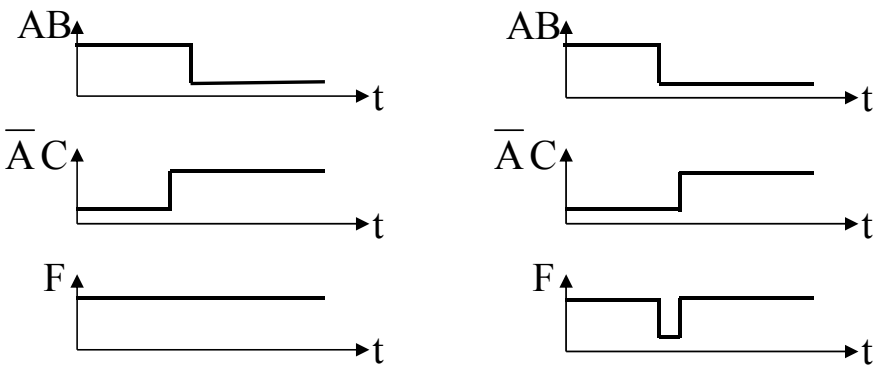
$$F = AB + \overline{A}C\,.$$

A függvény logikai kapukkal történő megvalósítása, a kapuk késleltetésének modellezésével a következőképpen ábrázolható:



Legyen az aktuális bemeneti kombináció *ABC*=111. Az ehhez a kombinációhoz tartozó kimenet 1. Változzon ezután a bemeneti kombináció a következőképpen *ABC*=011. A logikai függvénybe való behelyettesítéssel látszik, hogy az ehhez tartozó kimenet szintén 1. Ha azonban a jelterjedést vizsgáljuk, akkor azt láthatjuk, hogy ha az *A* és az $\overline{A}$ jel egymáshoz képest késik (konkrétan az $\overline{A}$ jel késik az *A*-hoz képest), akkor lesz egy rövid időszak, amikor a kimeneti VAGY-kapunak egyik bemenete sem 1 értékű (az *A* jel *már* nem 1 és az $\overline{A}$ jel *még* nem 1), aminek hatására a kapu kimenet 0-ra vált. Ez a 0 kimenet csak impulzusszerű: amint

az $\overline{A}$ jel „átjut” az alsó ÉS-kapun, a kimeneti VAGY-kapu az alsó bemenetén 1-et kap, aminek hatására a kimenet 1-re áll vissza. Ez a jelenség a különböző késleltetési idők miatt nem is feltétlenül következik be. Az alábbi ábrán a változások sorrendjétől függő kimeneti jelalakot látjuk.



A kombinációs hálózatok ilyen értelmű rendellenes működését *statikus hazárdnak* nevezzük. A statikus hazárd tehát definíciója szerint a kombinációs hálózat egy bemenetének változásakor jön létre,

$$x_1, x_2 \ldots x_i \ldots x_n \rightarrow x_1, x_2 \ldots \overline{x_i} \ldots x_n$$

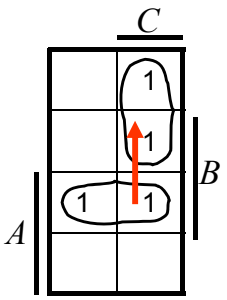
mégpedig úgy, hogy a függvény értéke a változás előtt és után ugyanaz:

$$f\left(x_1, x_2 \ldots x_i \ldots x_n\right) = f\left(x_1, x_2 \ldots \overline{x_i} \ldots x_n\right).$$

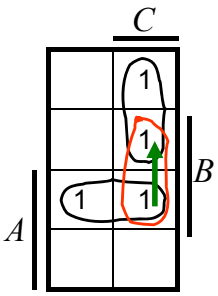
A hazárdjelenség hatására a kimeneten egy tranziens váltás történik:

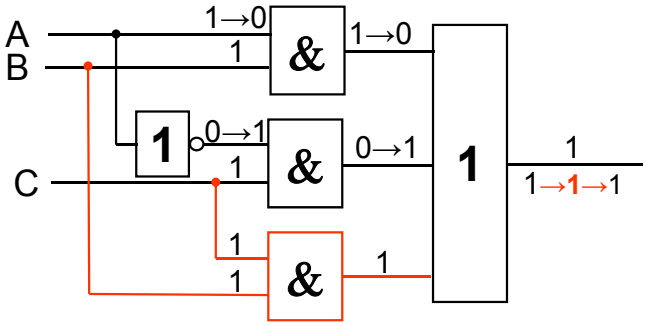
$$f\left(x_1, x_2 \ldots x_i \ldots x_n\right) \rightarrow \bar{f}\left(x_1, x_2 \ldots \overline{x_i} \ldots x_n\right) \rightarrow f\left(x_1, x_2 \ldots x_i \ldots x_n\right).$$

Amennyiben a hazárd zavaró hatású, a kombinációs hálózat helytelen működését okozza, akkor védekezni kell ellene.
A fenti példában is szereplő függvény Karnaugh-tábláját megvizsgálva látható, hogy a hazárd annál a bemeneti jel kombinációváltásnál következik be, amelyet nem fed le prímimplikáns.



Ha az eddig nem lefedett *BC* prímimplikánst is lefedjük, akkor az ezt megvalósító kapu az *A* bemenet értékétől függetlenül tartja az 1 bemenetet a VAGY-kapun, így annak kimenetén nem jön létre az 1→0→1 váltás.





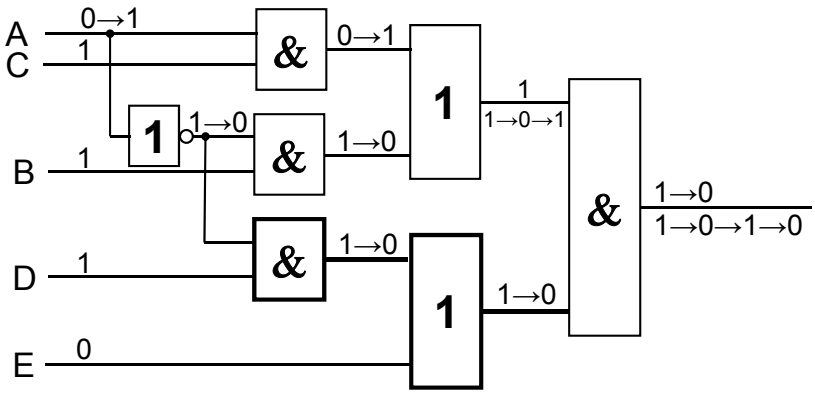
Összefoglalva tehát statikus hazárd legalább kétszintű hálózatban jön létre, kialakulásának feltétele, hogy a hazárdot okozó jel legalább két úton terjedjen. A statikus hazárdot a függvény Karnaugh-tábláján vehetjük észre: hazárddal terhelt átmenet ott van, ahol szomszédos mintermek nincsenek közös hurokkal (prímimplikánssal) lefedve. A statikus hazárd ellen úgy lehet védekezni, hogy az összes szomszédos mintermet le kell fedni közös hurokkal.

2.8.3. Dinamikus hazárd

Kettőnél többszintű hálózatok esetén a jelterjedési idő további rendellenes működést is okozhat. Amennyiben

- egy jel legalább három úton terjed a kimenetre, akkor
- olyan bemeneti jel változások esetén, amelynek során csak egyetlen bemenet változik, és
- a két bemeneti kombinációhoz tartozó függvényértékek különbözőek,

a kimeneten előfordulhat 1→0→1→0, vagy 0→1→0→1 változás. Ezt a jelenséget *dinamikus hazárdnak* nevezzük. A dinamikus hazárdra mutat példát az alábbi kapcsolás, amelyen a jelváltozások is megfigyelhetők.



Mivel a dinamikus hazárdot tulajdonképpen a hálózat kétszintű részhálózatain kialakuló statikus hazárdjelenségek okozzák, a dinamikus hazárd kivédése az egyes szinteken történő statikus hazármentesítéssel, vagy a hálózat kétszintű megvalósításával lehetséges.

2.8.4. Funkcionális hazárd

A statikus és a dinamikus hazárdokban közös volt, hogy olyan esetekben lépett fel, amikor két egymást követő bemeneti jelkombináció csak egyetlen helyiértéken tér el egymástól. Ha egy hálózat bemenetén egyszerre több jel változik, akkor ezt a változást a hálózat szinte biztosan nem egyidejűnek érzékeli. Ennek oka, hogy az egyes bemenetekre kapcsolódó kapuk késleltetése nem feltétlenül egyforma, de maguk a jelváltozások sem történnek egyidőben. Az ilyen bemeneti jel változás okozta helytelen működést *funkcionális hazárdnak* nevezzük. A funkcionális hazárd elleni védekezés kizárólag a bemeneti jelek megfelelő kapcsolásával oldható meg.

3. SORRENDI HÁLÓZATOK TERVEZÉSE

3.1. Bevezetés a sorrendi hálózatokba

Az előző fejezetben tárgyalt kombinációs hálózatok csak olyan logikai feladatok megoldására alkalmazhatók, amelyekben az egyes kimenetek kizárólag a mindenkori, aktuális éppen teljesülő feltételektől, azaz a pillanatnyi bemenetektől függenek. A kombinációs hálózat minden egyes bemeneti kombinációjához egyértelműen hozzárendelhetünk egy-egy kimeneti kombinációt:

$$Z = f(X),$$

ahol X a bemeneti kombinációk halmaza, Z a kimeneti kombinációk halmaza, f a hozzárendelést megvalósító leképezés, amely annyi logikai függvénnyel adható meg, ahány kimenetű a kombinációs hálózat.

3.1.1. A sorrendi hálózat működésmódja

Ha egy megoldandó probléma esetén a kimenet értékeit nem kizárólag a pillanatnyi bemeneti értékek alapján lehet meghatározni, hanem az a megelőzően fennálló bemeneti jelektől is függ, akkor erre a célra *sorrendi (szekvenciális)* logikai hálózatot kell terveznünk. A sorrendi hálózat ugyanis a kimeneti kombináció előállításához a pillanatnyi bemeneti kombináción felül a korábban fennállt bemeneti kombinációkat is, illetve azok sorrendjét is figyelembe veszi. Ilyen módon a sorrendi hálózatok esetében előfordulhat az is, hogy egy adott bemeneti kombinációhoz különböző kimeneti kombináció társuljon, a hálózat előéletétől függően. A hálózatot ért korábbi hatásoktól való függés megvalósítására a sorrendi hálózatnak minden egyes bemeneti kombináció fellépésének hatására elő kell állítania egy olyan ún. *szekunder kombinációt*, amely a hálózat előéletét hivatott képviselni, és a soron következő bemeneti kombinációval együtt egyrészt meghatározza a kimeneti kombinációt, másrészt pedig előállítja az új szekunder kombinációt, amely azután a soron következő bemeneti kombináció mellé reprezentálja a hálózat előéletét.
A szekunder kombinációkat a fenti szerepükből adódóan a sorrendi hálózat *állapotainak* nevezzük. A sorrendi hálózat állapotai az ún. *szekunder logikai változók* értékkombinációjaként jönnek létre. A szekunder logikai változókat *állapotváltozónak* is szokás nevezni.
Az állapotváltozók értékei függenek azok megelőző értékétől is, ezért tulajdonképpen az állapotváltozók *visszacsatolása* érvényesül a sorrendi hálózatban.
A sorrendi hálózat által megvalósítandó logikai feladattól függ, hogy az előírt működéshez hány állapot (más néven szekunder kombináció) szükséges.
A sorrendi hálózat működését a fentiek szerint az alábbi leképezéssel adhatjuk meg:

$$Z = f_Z(X, y),$$
$$Y = f_y(X, y),$$

ahol X a bemeneti kombinációk halmaza, Z a kimeneti kombinációk halmaza, y a bemenetre pillanatnyilag visszajutott szekunder kombinációk halmaza (azaz a pillanatnyi állapot), Y a bemeneti kombináció és a pillanatnyi állapot által meghatározott soron következő szekunder

kombinációk, azaz a következő állapot halmaza, f_Z a kimeneti kombinációt előállító leképezés *(kimeneti függvény)*, f_y a szekunder kombinációt előállító leképezés *(állapotfüggvény)*.
Mivel minden kialakuló szekunder kombináció visszajut a bemenetre, ezért a pillanatnyi és a következő állapotok halmaza tulajdonképpen ugyanaz a halmaz, amelyet az *állapotok halmazának* is nevezünk. A y és az Y jelölésbeli megkülönböztetésének csak az a szerepe, hogy a hálózat működésének fázisait, azaz az állapotváltozások menetét szemléltesse.
A kimeneti kombinációk előállítása szerint a sorrendi hálózatokat két csoportba oszthatjuk:

$$Z = f_Z(X, y)$$

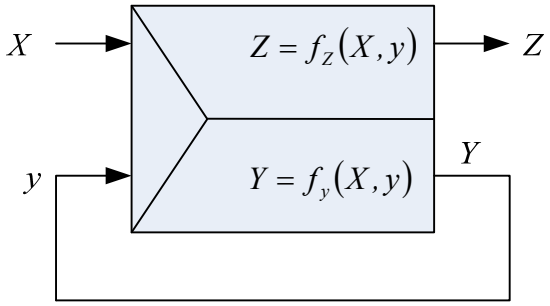
esetén *Mealy-modell* szerinti,

$$Z = f_Z(y)$$

esetén *Moore-modell* szerinti sorrendi hálózatról beszélünk. Ez utóbbi esetben a hálózat kimenete látszólag nem függ a bemenettől (X), valójában azonban a bemenet és a pillanatnyi állapot együttesen határozzák meg a kialakuló új állapotot, amely a visszacsatolás révén hatással lesz a kimenetre. Az előbbiekből az is következik, hogy Moore-modell szerint működő hálózatban egy adott állapothoz csak egyféle kimeneti kombináció rendelhető.

3.1.2. Az aszinkron sorrendi hálózatok működése

Vizsgáljuk meg a sorrendi hálózatok működését f_Z és f_y leképezések feltételezésével az alábbi ábra alapján.



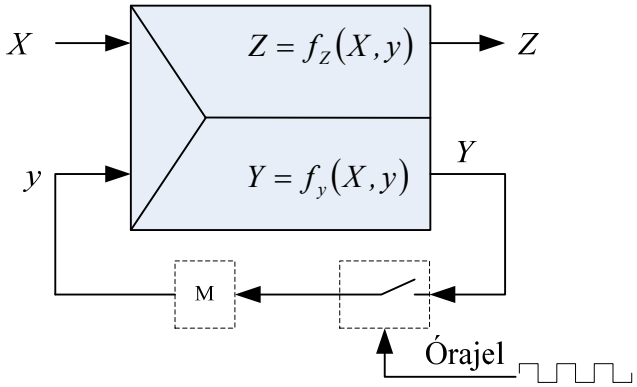
4.1. ábra: Aszinkron sorrendi hálózatok működése

Egy adott X bemeneti kombináció (amely tulajdonképpen x_1, x_2, … bemenetek pillanatnyi értékeinek egy kombinációja) és az éppen fennálló y kombináció (amely y_1, y_2, … szekunder változók vagy állapotváltozók értékeinek egy kombinációja) hatására az f_Z és f_y függvények szerint létrejön egy Z és Y kombináció. Még ha az X bemeneti kombináció változatlan marad, akkor sem biztos, hogy a hálózat azonnal nyugalomba kerül. Az Y kombináció ugyanis a visszacsatolás következtében y-ként visszajut a bemenetre, és az f_Z és f_y függvények révén újabb Z és Y értékeket hozhat létre. Az így kialakult Y új y kombinációt hoz létre a bemeneteken és így tovább. Nyugalmi állapot egy adott X bemeneti kombináció mellett csak akkor jöhet létre, ha egy kialakult Y kombináció a bemenetre y-ként visszajutva f_y alapján változatlan Y kombinációt hoz létre, vagyis Y=y. A hálózatnak ezt az állapotát a *stabil állapotnak* nevezzük. Egy adott bemeneti kombináció hatására tehát az állapotok addig változnak, amíg stabil állapot nem alakul ki. A változások alatti állapotok csak átmenetileg állnak fenn, és *instabil állapo-*

toknak nevezzük őket. Fennállási idejüket az határozza meg, hogy mennyi idő alatt jut vissza az új belső (Y) állapot a hálózat bemenetére (y).
Természetesen olyan esetek is előállhatnak, hogy nem minden bemeneti kombináció mellett jön létre stabil állapotot. Létezhetnek tehát olyan bemeneti kombinációk is, amelyek fennállása idején nem alakul ki stabil állapot, hanem az instabil állapotok valamilyen ciklus szerint ismétlődnek. Ennek következtében az Y és általában a Z kombinációk is ciklikusan változnak. A változás periódusideje természetesen csak egy adott kombináció értékére vonatkozhat, és nem jelenthet állandó ismétlődési időt, hiszen az instabil állapotok időtartamát meghatározó késleltetési hatások időben is változhatnak. Ha egy bemeneti kombináció mellett nem alakul ki stabil állapot, hanem hatására az említett módon az instabil állapotok állandóan váltják egymást, akkor azt mondjuk, hogy a sorrendi hálózat *oszcillál*.
Az eddig elmondottak szerint működő sorrendi hálózatokat *aszinkron* sorrendi hálózatoknak nevezik.

3.1.3. A szinkron sorrendi hálózatok működése

A sorrendi hálózatok másik típusának tárgyalásához vegyük a 4.2. ábrát.



4.2. ábra: Szinkron sorrendi hálózatok működése

Az aszinkron hálózatok blokkdiagramjához képest a visszacsatoló ágban látunk változást. A visszacsatoló ágban jelképesen olyan kapcsolót ábrázoltunk, amely periodikusan ismétlődő négyszögimpulzusok *(órajel)* hatására létrehozzák, illetve megszüntetik a visszacsatolást. A kapcsoló utáni M jelű elem feladata, hogy kimenetén azt az értéket (jelen esetben állapotváltozó kombinációt) jelenítse meg, amely a kapcsoló zárásának pillanatában bemenetére került, és ezt az értéket mindaddig fenntartsa, amíg újabb kapcsolózárás nem következik be. Az így felépített hálózat működése különbözik az aszinkron hálózatétól, ugyanis egy-egy állapot fennállásának időtartama jól meghatározható: a rendszer csak a visszacsatoló ágban lévő kapcsoló zárásának pillanatában vált állapotot, és ez az állapot az M jelű memóriaelemnek köszönhetően egészen a következő kapcsolózárásig nem változik.
A fentieken túl a bemenetek változására vonatkozóan is teszünk megkötést: a visszacsatoló ág ütemezése mellett megengedjük, hogy az X bement változzon, mégpedig úgy, hogy minden órajelperiódusban új X bemenet kerüljön a rendszerre. (Az órajel periódusának ismeretében egyértelműen meghatározható az X bemenet megváltozásának megfelelő időpontja.) Ekkor a bemeneti jelek *szinkronban* lesznek az órajellel.
A fentiek alapján nyilvánvaló, hogy a továbbiakban nem játszik szerepet az, hogy egy adott állapot stabil vagy instabil, hiszen minden y kombináció új X kombinációval találkozik (amely adott esetben természetesen lehet ugyanaz az X kombináció, mint az előző periódusban),új kimenetet és belső állapotot hozva létre.

3.1.4. Az aszinkron és a szinkron hálózatok összehasonlítása

Összefoglalva az eddigieket, hasonlítsuk össze a két hálózattípus legfontosabb jellemzőit.
Aszinkron hálózat

- Az aszinkron sorrendi hálózatok esetében az instabil állapotok miatt az állapotváltozók szükséges száma rendszerint nagyobb, mint szinkron esetben, ez megbonyolítja a logikai tervezés folyamatát.
- Viszont a bemeneti változások gyakoriságát, vagyis a működési sebességet csak az építőelemek működési sebessége és a jelterjedési késleltetések korlátozzák.
- A tervezés folyamán egyszerűséget jelent, hogy nem kell biztosítani a szinkronizációs feltételeket.

Szinkron hálózat

- A szinkron hálózatban nem értelmezünk külön instabil és stabil állapotot.
- A működés sebességet az órajel frekvenciája határozza meg.
- A bemeneti változásokra és a kimeneti kombináció értelmezésére szinkronizációs feltételeknek kell teljesülniük.

3.2. Sorrendi hálózatok működésének leírása

Ahhoz, hogy egy sorrendi hálózat működését megadjuk, le kell írnunk az f_Z kimeneti függvényt és az f_y állapotfüggvényt. Más szavakkal, le kell írnunk a rendszer állapotait, a lehetséges állapotátmeneteket és a rendszer kimenetének viselkedését az egyes állapotokban, különböző bemeneti kombinációk hatására. Ehhez többféle formalizmus áll rendelkezésre. A továbbiakban bemutatjuk az *állapottábla* segítségével történő leírást, majd az *állapotgráf* alkalmazásának a lehetőségeit.

3.2.1. Állapottábla

Az állapottábla a sorrendi hálózatok esetében ugyanúgy leír minden lehetséges esetet a hálózat működésében, mint ahogyan az igazságtáblázat teszi ugyanezt a kombinációs hálózatok esetében. Természetesen az összetettebb működésmód miatt a táblázat is bonyolultabb. Először vizsgáljuk meg azt, hogyan ábrázolja az állapottábla az egyes állapotok közötti átmenetet, illetve azt, hogy az állapotátmenetek milyen bemeneti kombinációk hatására jönnek létre (mindez tulajdonképpen az f_y állapotfüggvény leírása).
A táblázat egyes soraiban a lehetséges állapotok vannak ábrázolva, a táblázat oszlopaiban pedig a lehetséges bemeneti kombinációk. A táblázat egyes celláiban pedig az látszik, hogy ha az adott sor által reprezentált állapotban az adott oszlop által reprezentált bemeneti kombináció fellép, akkor milyen új belső állapotot vesz fel a rendszer.

y \ X	X^1	X^2	X^3	X^4
y^1	Y^1	Y^2	Y^3	Y^1
y^2		Y^2	Y^3	
y^3	Y^1		Y^3	Y^1

A fenti állapottábla ismeretében az általa ábrázolt rendszerről és annak működéséről a következőket tudhatjuk meg:

- A rendszernek összesen három állapota lehetséges: y^1, y^2 és y^3.
- A rendszerben összesen négy lehetséges bemeneti kombináció fordulhat elő: X^1, X^2, X^3 és X^4. (Ez esetben tipikusan két bemenetről beszélünk [x_1 és x_2], együtt összesen négy lehetséges kombinációt alkothatnak: 00, 01, 10 és 11 – ezek a lehetséges bemeneti kombinációk.)
- Ha rendszer az y^1 állapotban van (első sor) és X^1 bemeneti kombináció kapcsolódik a bemenetére, akkor az előálló új belső állapot az Y^1, amely a hálózat bemenetére visszacsatolva ismét az y^1 állapot hozza létre. Aszinkron hálózat esetében azt mondanánk, hogy az X^1 bemeneti kombináció *stabilizálja* az y^1 állapotot, így az y^1 állapot ilyenkor *stabil*. Aszinkron hálózatok esetében ezt a tényt jelölni is szoktuk az új belső állapot jelének bekarikázásával. (Szinkron hálózatok esetén nem értelmezünk stabil és instabil állapotokat, így jelölni sem lehet őket.)

y \ X	X^1	X^2	X^3	X^4
y^1	Y^1	Y^2	Y^3	Y^1
y^2		Y^2	Y^3	
y^3	Y^1		Y^3	Y^1

- Az X^1 bemeneti kombinációhoz hasonlóan az X^4 bemeneti kombináció is stabilizálja az y^1 állapotot, továbbá ugyanígy viselkedik az X^2 bemeneti kombináció az y^2 és az X^3 bemeneti kombináció az y^3 állapot vonatkozásában.
- Amennyiben stabilan az y^1 állapotban vagyunk és a bemeneti kombináció X^1-ről X^2-re változik, úgy a hálózat az Y^2 új állapotot veszi fel, amely a hálózat bemenetére visszacsatolódva létrehozza az y^2 állapotot. Ehhez hasonlóan tudjuk értelmezni a táblázatban feltüntetett valamennyi állapotátmenetet.
- Az is látszik a táblázatból, hogy nem minden cellában találunk bejegyzést: ezekben az esetekben a hálózat működése nem meghatározott, hasonlóan a kombinációs hálózatok közömbös bejegyzéseihez.

Az állapottábla tehát valamennyi állapot esetén megadja, hogy a lehetséges bemeneti kombinációk esetén milyen új állapotba kerül a rendszer, aszinkron hálózatok esetén pedig a stabil állapotokat is.
Az eddigi leírás nem adja meg a hálózat kimenetét a különböző esetekben. A kimenet jelölése a két típusú kimeneti modell (Mealy vagy Moore) esetén különböző. Már megtárgyaltuk, hogy Moore típusú hálózat esetében a kimenetet kizárólag a belső állapot határozza meg, tehát egy adott belső állapothoz egyféle kimeneti kombináció tartozhat. Ebben az esetben az állapottáblában soronként csak egy kimeneti kombinációt kell feltüntetnünk, például a következőképpen:

y \ X	X^1	X^2	X^3	X^4	Z
y^1	Y^1	Y^2	Y^3	Y^1	Z^1
y^2		Y^2	Y^3		Z^2
y^3	Y^1		Y^3	Y^1	Z^2

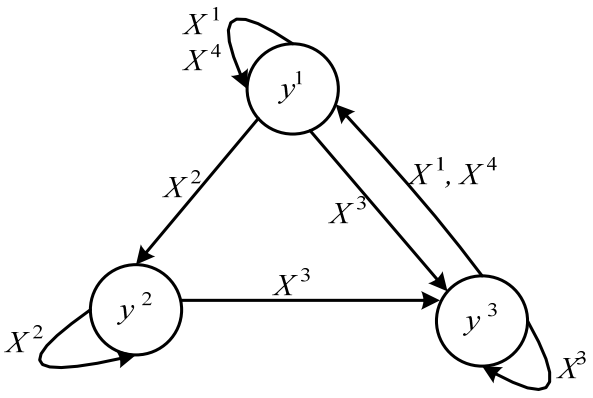
A fenti tábla utolsó oszlopa azt mutatja, hogy az y^1 állapotban mindig Z^1, az y^2 és y^3 állapotokban mindig Z^2 a hálózat kimenete, függetlenül attól, hogy mi a hálózat bemenete.
Mealy-modell esetén a hálózat kimenetének értékét a fennálló állapot és a pillanatnyi bemeneti kombináció együttesen határozza meg, ezért a kimenet értékét az egyes cellákba írjuk, például a következőképpen:

y \ X	X^1	X^2	X^3	X^4
y^1	Y^1/Z^1	Y^2/-	Y^3/Z^3	Y^1/Z^2
y^2		Y^2/Z^2	Y^3/Z^2	
y^3	Y^1/Z^2		Y^3/Z^3	Y^1/Z^2

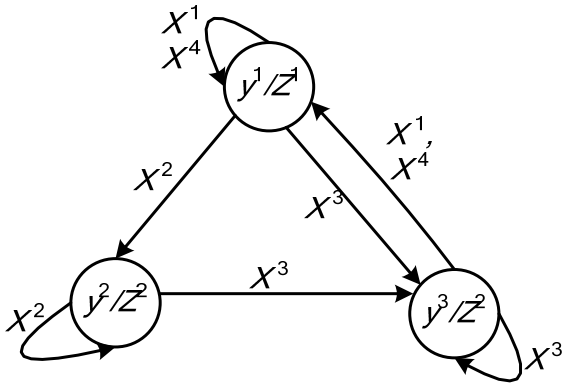
A fenti táblából látható például, hogy az y^1 állapotban Z^1 a kimenet, ha a bemeneti kombináció X^1, de ugyanebben az állapotban Z^2 a kimenet, ha a bemeneti kombináció X^4. Azt is vegyük észre, hogy nem minden állapotátmenethez szükséges megadni a kimeneti kombinációt (pl. y^1 állapotban, X^2 esetén), ezeket az eseteket a kombinációs hálózatok közömbös kimeneteihez hasonlóan kihúzással jelöljük.

3.2.2. Állapotgráf

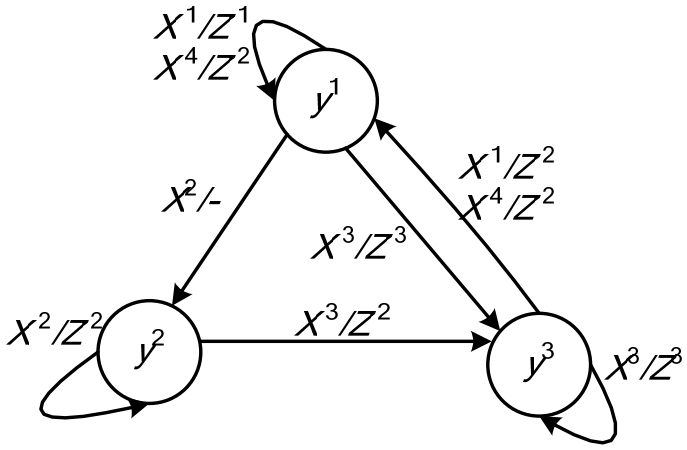
Az állapotgráf segítségével grafikusan lehet megadni a sorrendi hálózatok működését. Az állapotgráf és az állapottábla egyértelműen alakítható át egymásba.
Az állapotgráf ábrázolásakor a gráf csomópontjait körökkel jelöljük, amelyek a sorrendi hálózat állapotait reprezentálják. Az egyes állapotok azonosítóját a körökbe szoktuk írni. Az egyes állapotok közötti átmeneteket a gráf irányított élei ábrázolják, mégpedig úgy, hogy az élre írt címke mutatja azt a bemeneti kombinációt, amelynek hatására az állapotátmenet végbemegy.
Az előző szakaszban állapottáblával bemutatott sorrendi hálózat gráfja a fenti jelölések figyelembevételével a következőképpen alakul:



Ami a kimenetek ábrázolását illeti, az állapotgráf esetében ugyanúgy eltér a Mealy- és a Moore-modell megjelenése. Mivel Moore-modell esetében a kimenet a belső állapottól függ, ezért az egyes állapotokhoz tartozó kimenet értékét az állapotot reprezentáló körbe írjuk.



Mealy-modell esetén a kimeneteket a címkézett, irányított élekre írjuk, mégpedig az adott állapotátmenetet kiváltó bemeneti kombináció mellé, a következőképpen:



3.3. Elemi sorrendi hálózatok (tárolók)

A sorrendi hálózatok megvalósításához szükségünk van egy olyan elemkészletre, amelynek segítségével a sorrendi hálózatok felépíthetők, hasonlóan ahhoz, ahogyan a kombinációs hálózatok megvalósításához rendelkezésre álltak a logikai kapuk, amelyek tulajdonképpen elemi kombinációs hálózatok. A sorrendi hálózatok esetében ezek az elemi alkotóelemek, vagy elemi sorrendi hálózatok a *tárolók*, amelyek segítségével, a logikai kapukat továbbra is felhasználva meg tudjuk valósítani a sorrendi hálózatokat.
Megjegyezzük, hogy 3.1.2. szakaszban leírtak miatt az aszinkron sorrendi hálózatok megvalósíthatók *visszacsatolt kombinációs hálózatként*. Ilyenkor nincs szükség tárolók alkalmazására. Ennek módszerét a 3.5. fejezet ismerteti. A következőkben ismertetésre kerülő tároló típusoknak van néhány közös tulajdonsága:

- Ezek a hálózatok mind a Moore-modell szerint működnek, vagyis kimenetüket kizárólag a belső állapotuk határozza meg,
- mégpedig a lehető legegyszerűbb függvény szerint: $Z=y$, azaz a tároló kimenete mindig azonos a belső állapottal.

- Egyetlen szekunder változóval *(y)* csak két állapotot tudunk megkülönböztetni, így a tárolóknak két belső állapota lehetséges, ezért szokás ezeket *kétállapotú, billenő elemeknek* vagy *flip-flopoknak* nevezni.

Az egyes tárolók abban térnek el egymástól, hogy a két állapotuk közötti változást milyen bemeneti kombinációval lehet előidézni, illetve hogy alkalmasak-e aszinkron működésre is, vagy csak szinkron sorrendi hálózatokban alkalmazhatók.

3.3.1. SR-tároló

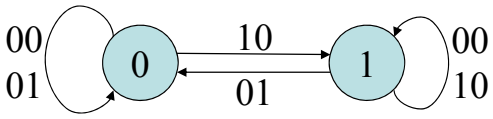
Az SR-tároló elnevezése a Set (beállítás) és a Reset (törlés) szavak rövidítéséből származik. Definiált működése szerint az *S* bemenetre jutó 1 érték a tároló állapotát 1 értékűre állítja be (beír), míg az *R* bemenetre jutó 1 érték a tároló állapotát 0-ra állítja (töröl). Ha mindkét bemenet 0, akkor a tároló állapota nem változik ($Y=y$). Az $S=1$ és $R=1$ bemenetre a tároló működése nincs definiálva (úgy is szoktuk mondani, hogy ez egy tiltott bemeneti kombináció SR-tároló esetén). Természetesen az SR-tároló fizikai megvalósítása során *valami* történik $SR=11$ bemenet esetén is: a fizikai kialakítástól függően a hálózat vagy írási vagy törlési elsőbbségűként működik, és valamelyik parancs érvényre jut.
A tároló állapottáblája a következő (a kimeneti kombináció értékét nem tüntetjük fel külön, hiszen az azonos a belső állapottal):

y \ *SR*	00	01	11	10
0	0	0	--	1
1	1	0	--	1

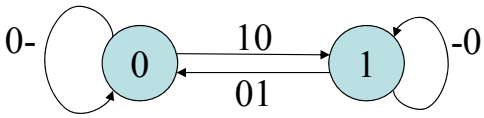
Az állapottáblán az $SR=1$ rovatokban azért került közömbös bejegyzés, mert a definiálatlan (tiltott) működés miatt feltételezhetjük, hogy a tároló nem kap ilyen vezérlést. Az állapottábla alapján megállapíthatjuk, hogy a működés mind szinkron, mind aszinkron módban értelmezhető, hiszen egyetlen specifikált oszlopban sem történik oszcilláció, sőt az is látszik, hogy szinkron és aszinkron esetben ugyanazt a működést kapjuk, azaz minden bemeneti kombinációsorozatra ugyanazt a kimeneti kombinációsorozatot (vagy állapotsorozatot) kapjuk szinkron és aszinkron esetben. Ebből következik, hogy az SR-tároló alapján tervezhető aszinkron sorrendi hálózat is. Aszinkron SR-tároló esetében jelölhetjük a stabil állapotot is:

y \ *SR*	00	01	11	10
0	(0)	(0)	--	1
1	(1)	0	--	(1)

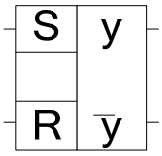
Az SR-tároló állapotgráfja a következő:



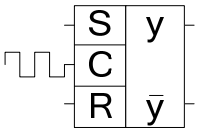
Látható, hogy az SR-tároló a 0 állapotát mind *SR*=00, mind pedig *SR*=01 esetén megtartja. Ez azt is jelenti, hogy ha a 0 állapotban az *S* bemenet értéke 0, akkor az *R* bemenet értékétől függetlenül a 0 állapotban marad a tároló. Azt is mondhatjuk tehát, hogy ilyen esetben az *R* bemenet értéke közömbös. Ugyanilyen egyszerűsítést hajthatunk végre az 1 állapot megtartásánál. Ekkor a következő állapotgráfot kapjuk:



Az SR-tároló szokásos áramköri rajza aszinkron esetben következő:



Szinkron SR-tároló esetén a szinkronizációs feltételek megteremése érdekében fel szoktuk tüntetni a tároló órajel bemenetét is (*C*-vel jelölve):



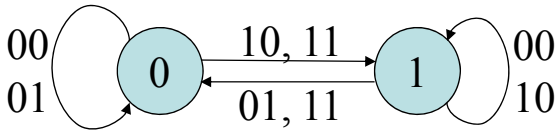
Már most megjegyezzük, hogy valamennyi ismertetett tároló képes szinkron módon működni, így alkalmas szinkron sorrendi hálózatok megvalósítására, de csak az SR- és a DG-tároló alkalmas arra, hogy aszinkron hálózatot valósítsunk meg segítségükkel.

3.3.2. JK-tároló

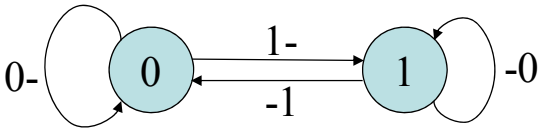
A JK-tárolónak szintén két bemenete van, amelyek jelölése *J* és *K*. Működése hasonlít az SR-tárolóéhoz, amennyiben a *J* bemenet megfelel az *S* bemenetnek, a *K* bemenet pedig az *R* bemenetnek. A különbség a két tároló között a *JK*=11 bemeneti kombináció esetében van. Erre a bemenetre az SR-tároló működése nincs definiálva, a JK-tároló esetében ez a működés is definiált: hatására a tároló állapotot vált, azaz ha eddig a 0 állapotban volt, akkor 1-be kerül, ha eddig az 1 állapotban volt, akkor a 0-ba kerül. Mindez az állapottáblán a következőképpen ábrázolható:

JK / *y*	00	01	11	10
0	0	0	1	1
1	1	0	0	1

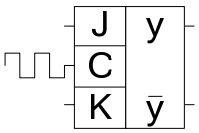
Az állapottáblát megvizsgálva látható, hogy az 11 bemeneti kombináció oszlopában nem alakul ki stabil állapot: a hálózat a két állapot között oszcillál. Ezért a JK-tároló nem alkalmas aszinkron működésre, csak szinkron hálózatok tervezése során használható fel. A tároló állapotgráfja következő ábrákon látható:



Az SR-tárolóhoz hasonló bemeneti összevonások után a JK-tároló állapotgráfja a következőképpen is felírható:



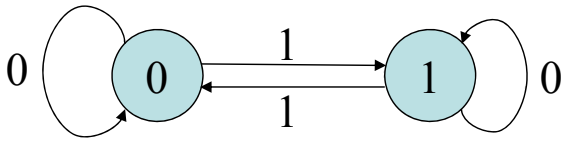
A JK-tároló szimbolikus áramköri jelölése:

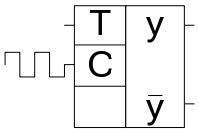


3.3.3. T-tároló

A T-tárolót a JK-tárolóból származtathatjuk úgy, hogy a *J* és *K* bemeneteket összekötjük. Ezáltal olyan működést kapunk, mintha egy JK-tárolót kizárólag 00 és 11 bemeneti kombinációkkal vezérelnénk. A JK-tároló működésmódjának ismeretében már megállapíthatjuk, hogy a *T* bemenetre érkező 0 (*JK*=00) esetén a T-tároló állapota nem változik, míg a *T*-re érkező 1 (*JK*=11) esetén a tároló állapota az ellenkezőjére változik. Természetesen a T-tároló sem képes aszinkron módon működni, ugyanazon okból, mint a JK-tároló. A tároló állapottáblája, állapotgráfja és szimbolikus jelölése a következő:

y \ T	0	1
0	0	1
1	1	0

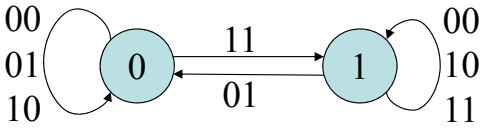




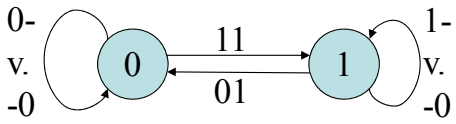
3.3.4. DG-tároló

A DG-tároló kétbemenetű flip-flop, bemeneteit *D*-vel és *G*-vel jelöljük a Data (adat) és a Gate (kapu) szavak rövidítéseként. A DG-tároló által megoldott logikai feladat úgy fogalmazható meg, hogy *G*=1 időtartama alatt a tároló kimenete (állapota) követi a *D* bemenetre jutó jelváltozásokat (azaz *Y*=*D*). Ha viszont *G*=0, akkor egy újabb *G*=1 jelig a flip-flop a *D* bemenet értékétől függetlenül megtartja a *G*=0 bekövetkezésekor éppen jelenlévő kimeneti értékét *(Y=y)*. Az állapottáblát megvizsgálva megállapítható, hogy egyik bemeneti kombináció esetén sem alakul ki oszcilláció, így a DG-tároló aszinkron hálózatok megvalósításához is felhasználható. A tároló állapottáblája, állapotgráfja következő:

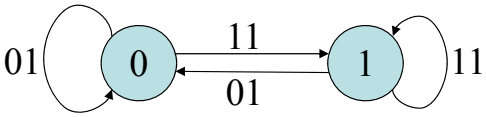
y \ DG	00	01	11	10
0	0	0	1	0
1	1	0	1	1



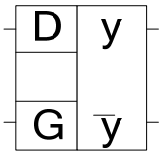
Egyszerűsítések után:



A fenti jelölés azt jelenti, hogy a tároló megőrzi 0 állapotát akár 0–, akár –0 bemeneti kombináció esetén, azaz ugyanazt a működést többféleképpen is kiválthatjuk a DG-tárolóban. Amennyiben az állapotváltozások előidézésére csak a következő kombinációkat használjuk:



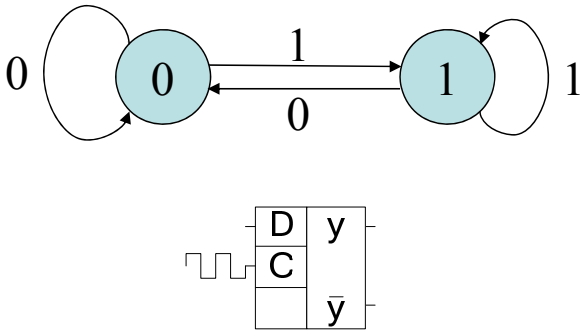
akkor azt láthatjuk, hogy a *G* bemenet mindig 1 lesz, a *D* bemenet pedig mindig a kívánt állapot. Más szavakkal a hálózat kimenetén az jelenik meg, ami a *D* bemeneten van. Nyilvánvaló, hogy ennek a működésnek a megvalósításához nincs szükség tárolóra, hiszen egy vezeték éppen így viselkedik. A DG-tároló alkalmazásának akkor látjuk igazán hasznát, ha a sokféle vezérlési lehetőséget ki tudjuk használni a sorrendi hálózat egyszerűbb megvalósítása érdekében.
A DG tároló és szimbolikus jelölése a következő:



3.3.5. D-tároló

A D-tároló egybemenetű flip-flop. A kimenet (állapot) minden egyes órajelimpulzus hatására azt az értéket veszi fel, amely a bemeneten az órajelimpulzus fellépésekor fennáll. A D-tároló ezt az értéket (állapotot) a bemeneti érték változásaitól függetlenül megtartja egy újabb órejelimpulzus megjelenéséig.
Látható, hogy a D flip-flop nem más, mint a szinkron sorrendi hálózatok visszacsatoló ágában (pontosabban ágaiban) feltételezett elemek tulajdonságait megvalósító hálózat (ld. 3.1.3. szakasz). A D-tároló állapottáblája formailag aszinkron módon is értelmezhető, azaz nem alakul ki oszcilláció, de természetesen így nem oldaná meg az előírt logikai feladatot, sőt ez a működés nem is sorrendi, hiszen *Y* független *y*-tól, azaz nincs visszacsatolás. Ez egyébként abból is látszik, hogy az állapottábla két sora azonos, aminek következtében a két állapot megkülönböztetése is felesleges.

y \ *D*	0	1
0	0	1
1	0	1



3.4. Szinkron sorrendi hálózatok tervezése

A sorrendi hálózatok tervezési eljárásainak ismertetését a szinkron sorrendi hálózatok tervezésével kezdjük. Ennek oka az, hogy a szinkronizált működésmód miatt a szinkron sorrendi hálózatok tervezésénél nem jelentenek gondot a hazárdokhoz hasonló tranziens jelenségek (ezeket a sorrendi hálózatok esetében *versenyhelyzetnek* nevezzük), ezért a tervezés eljárás némiképp egyszerűbb. Természetesen a tervezési eljárás egyszerűsödése nincs ingyen: az árat az alkalmazott elemek komplikáltabb kialakításánál (szinkron tárolók, órajel generátorok stb.) fizetjük meg.
A tervezési eljárásokat, a tervezés lépéseit a jelen fejezetekben röviden, áttekintő jelleggel ismertetjük, a példatárban számos példán keresztül lehet a gyakorlati ismereteket részletesebben elsajátítani.

3.4.1. A logikai feladat meghatározása (specifikáció)

Akár szinkron, akár aszinkron hálózatról van szó, a tervezés első lépése a logikai feladat megfogalmazása. Ez történhet szövegesen, ekkor a tervezőnek kell a szöveg értelmezése alapján állapotgráfot, vagy állapottáblát készíteni. Hogy melyiket célszerű, az a feladat jellegétől függ.

3.4.2. Az előzetes állapottábla összeállítása

A tervezés további lépéseihez szükség van a hálózat működését leíró állapottábla előállítására. A szöveges megfogalmazásból, de még az állapotgráfos leírásból sem mindig derül ki egyértelműen, hogy a hálózatnak minimálisan hány állapottal kell rendelkeznie. Ezért a szöveges megfogalmazás alapján rendszerint több állapotot különböztetünk meg, mint ahány állapotra a feladat megoldásához végül szükség lesz. Az előzetes állapottábla ezeket az *előzetesen* megállapított állapotokat tartalmazza.
Az előzetes állapottáblában az állapotokat szokásosan az ábécé kisbetűivel jelöljük. Az állapottáblában az állapotátmeneteken kívül fel kell tüntetnünk a hálózat kimenetét is, mégpedig olyan formában, amely megfelel a hálózat kimeneti modelljének (Mealy vagy Moore, lásd 3.2. fejezet). Ez azt is jelenti, hogy a kimeneti modellt ebben a lépésben kell meghatároznunk.

3.4.3. Az összevont állapottábla

Az előzetes állapottábla felvételét követően célunk, hogy megtaláljuk azokat az állapotokat, amelyeket a feladat értelmezése során feleslegesen különböztettünk meg. Az összevonási,

egyszerűsítési eljárás célja, hogy a lehető legkevesebb állapottal oldjuk meg a logikai feladatot. Az állapotok összevonásának az eredménye lesz az *összevont állapottábla*.
Általánosan fogalmazva: két állapotot akkor vonhatunk össze, ha a két állapotban a rendszer azonosan viselkedik. Részletesebben ez annyit jelent, hogy a két állapotban az egyes bemeneti kombinációk esetén az előálló új belső állapotok megegyeznek, és az adott bemeneti kombinációhoz tartozó kimeneteik is megegyeznek. Az összevonások során nagy szerephez jutnak az előzetes állapottáblában nem meghatározott állapotátmenetek, illetve kimenetek, mivel ezek bármilyen más specifikált állapottal vagy kimenettel összevonhatók.
Moore-modell szerinti hálózat esetében célszerű az egyszerűsítés során a kimenetekből kiindulni: csak azok az állapotok vonhatók össze, amelyek esetében a kimeneti kombinációk megegyeznek. Természetesen ezenfelül az egyes bemeneti kombinációkhoz tartozó új belső állapotoknak *(Y)* is meg kell egyezniük az összevonandó állapotokban.
Az összevont állapottábla állapotait az ábécé nagybetűivel szoktuk jelölni.
Néhány megjegyzés az állapotösszevonásokhoz:

- Az állapotok összevonása nem feltétlenül lehetséges.
- Az összevonási szabályokat egy állapotpárra fogalmaztuk meg a fentiekben, de amennyiben a feltételek három vagy annál több állapotra is igazak, akkor ezek is összevonhatók (a Karnaugh-táblától eltérően itt nem kell ragaszkodni a kettő hatványai szerinti összevonáshoz.)
- Az összevonható állapotok ilyen módon való felismerése nem közvetlen, szisztematikus eljárás, részben a tervező gyakorlatán múlik, hogy felismeri-e az összevonható állapotokat – mindez hasonló a Karnaugh-táblán kiválasztandó prímimplikánsokhoz. Megjegyezzük azonban, hogy léteznek szisztematikus állapotminimalizálási eljárások is; ezekre a nagy állapotszámú hálózatok esetén feltétlenül szükség van, mivel azok nem tekinthetők át olyan könnyen, mint a 3-4-5 állapotú rendszerek.

3.4.4. Állapotkódolás

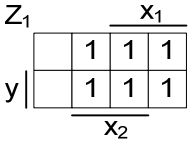
Miután rendelkezésünkre áll az összevont állapottábla, az egyes, még betűkkel jelölt állapotokhoz egy-egy állapotkódot (szekunder változó kombinációt) kell rendelni. Az összevont állapottábla sorainak számától függ, hogy ehhez hány szekunder változóra, más néven állapotváltozóra van szükség. Az egyes állapotváltozók lehetséges értékkombinációinak legalább annyinak kell lenni, mint ahány állapot szerepel az állapottáblában. Ha például két állapotra sikerült az összevont állapottáblában redukálni az állapotok számát (pl. *A* és *B*), akkor egyetlen állapotváltozó elegendő, amelynek 0 értéke az egyik (pl. *A*), 1 értéke a másik (pl. *B*) állapotot jelöli. Ha három állapot van az állapottáblában, akkor két állapotváltozóra (y_1, y_2) van szükség, amelynek négy lehetséges kombinációjából (00, 01, 10 és 11) kell hármat az egyes állapotokhoz rendelni. Négy állapot esetén szintén két állapotváltozó szükséges, és ekkor mind a négy lehetséges kombinációt felhasználjuk az állapotkódolásra. Három állapotváltozóval már egészen 8 állapotig tudjuk biztosítani az állapotkódot, hiszen $2^3=8$.
Az egyes kódok állapotokhoz való rendelése tetszőleges, de a későbbi megvalósításra van hatása a választott kódolásnak. A kódolás elvégzése után előállíthatjuk a *kódolt állapottáblát*, amelyben az egyes, korábban betűkkel jelölt állapotkódokat a bináris állapotkódokkal helyettesítjük.

3.4.5. Kimeneti függvény meghatározása

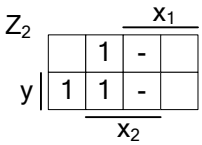
A kódolt állapottábla alapján felírhatjuk a $Z=f_Z(X,y)$ függvényt, illetve megadhatjuk annak algebrai alakját, hiszen a kódolt állapottábla tartalmazza ezt a belső állapotváltozóktól és a bemeneti jelektől függő kimeneti függvényt vagy függvényeket. Sőt, a kódolt állapottábla megfeleltethető egy Karnaugh-táblának, például a következő esetben:

y \ x_1x_2	00	01	11	10
0	0/00	0/11	-/--	1/10
1	1/01	0/11	-/--	1/10

Az ábrázolt függvénynek két bemenete van (x_1 és x_2), és mivel két állapota van, ezért egyetlen állapotváltozóval (y) meg lehetett oldani a kódolást. Az állapottáblából az is látszik, hogy a függvénynek két kimenete van (Z_1 és Z_2), ugyanis az állapotkódok utáni / jelet követően két értéket látunk. A fenti esetben a Z_1 és Z_2 kimenetekhez is egy-egy Karnaugh-táblát rendelhetünk, amelynek változói az y, az x_1 és az x_2. A peremezés az állapottábla fejlécezését helyettesíti:



$$Z_1 = x_1 + x_2$$



$$Z_2 = y\overline{x}_1 + x_2$$

Természetesen annyi Karnaugh-táblát kell alkalmazni, ahány kimenete van a hálózatnak (a fenti esetben kettő). A Karnaugh-táblák mérete szintén a feladattól függ. Azt mondhatjuk, hogy annyi oszlopa van a Karnaugh-táblának, ahány lehetséges bemeneti kombináció (a fenti példában 4) és annyi sora van a Karnaugh-táblának, ahány állapotváltozó (szekunder változó) kombináció van a hálózatban. (Ez utóbbi többnyire megegyezik az állapotok számával; eltérés akkor van, ha például 3 állapota van a hálózatnak: a Karnaugh-táblának ilyenkor is 4 sorosnak kell lennie, a nem használt állapotkódok esetén a kimenet közömbös.)
A fent ismertetett eljárás Mealy-modellek esetében igaz, de alkalmazható Moore-modell esetében is. Moore-modellek esetében a kimeneti függvény felírása jóval egyszerűbb, mivel azok csak a belső állapotváltozóktól függenek, a bemenetektől nem.

3.4.6. A vezérlési tábla összeállítása

A tervezés következő fázisában minden egyes állapotváltozóhoz (szekunder változóhoz) egy-egy tárolót rendelünk. Ez a tároló fogja reprezentálni az adott állapotváltozó értékét a hálózat működése során. Amennyiben az adott állapotváltozóhoz rendelt tároló által tárolt érték egy adott pillanatban 0, akkor az az adott állapotváltozó 0 értékét reprezentálja, ha 1-et tárol, akkor az adott állapotváltozó értéke 1. Az egyes állapotváltozók értékkombinációi együttesen határozzák meg a rendszer állapotát. Azaz egy négyállapotú rendszerben a két állapotváltozót egy-egy tárolóban tároljuk; az egyes tárolók 0-t vagy 1-t tárolhatnak, így alakul ki a rendszer négy lehetséges állapota (00, 01, 10 és 11).
A vezérlési tábla, illetve a vezérlési függvények előállításának az a célja, hogy a kódolt állapottábla által leírt, a hálózattól elvárt működést (megfelelő állapotváltozást) segítségével a tárolókban létre lehessen hozni. Úgy is fogalmazhatunk, hogy a vezérlési tábla, illetve a vezérlési függvény fordítja le az adott tároló nyelvére a kódolt állapottáblát.
A vezérlési tábla előállításához meg kell vizsgálnunk az egyes, a kódolt állapottábla által leírt állapotváltozásokat, és meg kell adnunk azt, hogy az adott állapotváltozást az adott típusú tároló esetében milyen bemeneti kombinációval érhetjük el. Az általános magyarázatot legegyszerűbb egy példán keresztül megérteni. Vegyük az előző kódolt állapottáblát:

y \ x_1x_2	00	01	11	10
0	0/00	0/11	-/--	1/10
1	1/01	0/11	-/--	1/10

Az állapottábla sora mutatja, hogy jelenleg melyik állapotban van a rendszer, az oszlop pedig azt jelzik, hogy az oszlop fejlécében fellépő bemeneti kombináció esetén milyen új belső állapotba kell kerülnie a rendszernek. Példaként tekintsük a bekarikázott állapotváltozást. Az adott hely azt jelenti, hogy ha a hálózat a 0 állapotban van (a sor elején lévő y érték 0)és a bemenetére 10 bemeneti kombináció kerül, akkor az 1-jelű állapotba kell kerülni, azaz y:0→1, (és eközben a kimenete 10 legyen). Legyen a választott tárolónk a JK-tároló. A JK-tároló állapotgráfját megvizsgálva látható, hogy az y:0→1 váltást úgy lehet előidézni, ha a tároló J bemenetére 1-et, a K bemenetére pedig 1-et vagy 0-t kapcsolunk (az állapotváltozás 11 és 10 hatására is végbemegy), azaz a J bemenetére 1-et, a K bemenetre bármit kapcsolhatunk (JK=1–). A vezérlési tábla megfelelő cellájába ezért ezt az értéket írjuk (ld. a lenti táblázat bekarikázott részét). Ugyanezt az eljárást követve tölthetjük ki a teljes vezérlési táblát, a következőképpen:

y \ x_1x_2	00	01	11	10
0	0-	0-	--	1-
1	-0	-1	--	-0

A vezérlési tábla mérete ugyanúgy a bemeneti és a belső állapotváltozók függvénye, az egy-egy cellába írandó jelek száma pedig egyrészt a tárolók számától függ, másrészt attól, hogy az adott tárolótípus egy- vagy kétbemenetű.
A vezérlési tábla ismeretében már megvalósíthatjuk a tárolók bemenetét vezérlő kombinációs hálózatot, amely már megfelelően fogja vezérelni a tárolók bemeneteit ahhoz, hogy az eredetileg szükséges $Y=f_y(X,y)$ leképezés megvalósuljon. A vezérlési tábla tulajdonképpen tartalmazza a tárolók bemenetét vezérlő függvényeket (hasonlóan a kimenetet vezérlő függvényekhez).
A megvalósításhoz felhasznált tároló megválasztását legtöbbször az befolyásolja, hogy melyik típus áll rendelkezésre az adott időben, az adott feladathoz. A megfelelő vezérléshez alkalmazandó vezérlőfüggvények bonyolultsága azonban jelentősen függhet a választott tároló típusától. Ha módunkban áll, célszerű megvizsgálni különböző tárolók választásának hatását, de ezt általában csak próbálgatással tudjuk megtenni. Több állapotváltozó esetén természetesen nem szükségszerű, hogy mindegyikhez azonos típusú flip-flopot válasszunk.

3.4.7. Realizáció

Amennyiben mind a kimenetet vezérlő függvények (akár Mealy-, akár Moore-modellről van szó), mind a tárolók bemeneteit vezérlő függvények rendelkezésre állnak, megvalósítható a kapcsolás, felrajzolható a logikai vázlat.

3.5. Aszinkron sorrendi hálózatok tervezése

Az aszinkron hálózatok tervezési folyamata többé-kevésbé megegyezik a szinkron hálózatokéval. A következő ismertetésben ezért csak az eltérésekkel foglalkozunk, feltételezzünk, hogy az olvasó a szinkron hálózatok tervezésének folyamatával tisztában van.
A tervezési eljárás első lépése, a logikai feladat meghatározása alapvetően nem különbözik a szinkron hálózatokétól. Az előzetes és az összevont állapottábla felépítését vizsgálva azonban már találunk különbségeket.

3.5.1. Előzetes és összevont állapottábla

A szinkron és az aszinkron hálózatok közötti lényeges különbség, hogy az aszinkron hálózatokban a jelterjedés nincs ütemezve, ezért ott megkülönböztetünk stabil és instabil állapotokat. Az aszinkron állapottáblázatban az állapotok stabil voltát jelölni szoktuk, ahogyan azt a 3.2.1. szakaszban bemutattuk.
Egy aszinkron hálózat akkor valósítható meg, ha

- minden specifikált bemeneti kombinációhoz tartozik legalább egy stabil állapot, amelyben az adott hálózat az adott bemenet esetén stabilizálódik, továbbá
- ha minden egyes belső állapothoz tartozik legalább egy olyan bemeneti kombináció, amely esetén az adott belső állapot stabilizálódik.

E feltételek meglétének ellenőrzésében segít a stabil állapotok jelének bekarikázása az állapottáblában – akár az előzetes, akár az összevont állapottábláról beszélünk. Az első feltétel meglétét úgy ellenőrizhetjük, ha megvizsgáljuk, hogy az állapottábla minden oszlopában van-e legalább egy bekarikázott (stabil) állapot, a második feltételt pedig a soronkénti legalább egy bekarikázott (stabil) állapot meglétével ellenőrizhetjük. Amennyiben valamelyik feltétel nem teljesül, akkor az adott hálózat nem valósítható meg aszinkron módon, mert oszcilláció léphet fel.
Az állapotok összevonásának szabályai nem térnek el szinkron és aszinkron hálózatok esetén, a szinkron hálózatok tervezése kapcsán elmondottak az aszinkron hálózatra is igazak.

3.5.2. Állapotkódolás, versenyhelyzetek

Az állapotok kódolásának eljárása sem különbözik a szinkron és az aszinkron hálózatokban. A kódolt állapottábla alapján azonban szükséges bizonyos ellenőrzések elvégzése, ugyanis a nem megfelelő állapotkódolás nemcsak a kialakuló hálózat egyszerűségét befolyásolja, hanem a hálózat helyes vagy hibás működését és meghatározhatja. Ennek tárgyalását vegyük az alábbi kódolt állapottábla részletet mint példát (az üresen hagyott résszel nem foglalkozunk, és nem foglalkozunk a kimenetekkel sem):

y_1y_2 \ x_1x_2	00	01	11	10
00	(00)	11	00	11
01		11	–	(01)
11		(11)	00	(11)
10		11	–	11

Az elemzéshez tételezzük fel, hogy a rendszer a 00 állapotban van (y_1y_2=00), a bemeneti kombináció pedig 00 (x_1x_2=00): ekkor a hálózat stabil állapotban van. Változtassuk a bemeneti kombinációt x_1x_2=01-re. Ennek hatására a hálózat új állapotváltozóinak az y_1y_2=11 értéket kell felvenniük. Ugyanez történik akkor is, ha a bemeneti kombinációt x_1x_2=10-ra változtatjuk, továbbá ugyanez a helyzet az 11 állapot 01 bemeneti kombinációjával stabilizált állapotát követő 11 bemeneti jelre való váltást követően, amikor is a 00 állapotba kell a hálózatnak kerülnie. A hálózat aszinkron működéséből következik, hogy a két állapotváltozó aktuális értékét tartalmazó tároló sem működik egymással szinkronizálva, így semmi nem garantálja, hogy a két állapotváltozó váltása egyidejűleg történik. Amennyiben a két változó nem egyszerre változik, akkor egy közbenső állapot fellépésével is számolni kell: a 00→11 helyett a

00→10 vagy a 00→01 átmenet (vagy az 11→00 átmenet helyett az 11→10 vagy az 11→01 átmenet) történik meg, attól függően, hogy melyik állapotváltozót tartalmazó tároló vált előbb értéket. Ezt a jelenséget *versenyhelyzetnek* nevezzük. Az 10 és a 01 kódok valódi állapotokat kódolnak: a rendszer tehát az 11 jelű állapot helyett a 01 vagy az 10 jelű állapotba kerül.
Vizsgáljuk először az x_1x_2=10 bemeneti kombináció hatására létrejövő állapotváltozást: ha a hálózat az 10 állapotba kerül, akkor az ottani bejegyzésnek megfelelően „továbbmegy" az 11 állapotba, azaz tulajdonképpen eléri az eredetileg is megcélzott állapotot, igaz egy közbenső állapoton keresztül. Amennyiben azonban a 01 állapotba kerül a rendszer, akkor az ottani 01 bejegyzés stabilizálja a hálózatot, így a kívánt 11 stabil állapot helyett a szintén stabil 01 állapotba kerül a rendszer. Ez nyilvánvalóan nemkívánatos működést, a jelenséget ilyenkor *kritikus versenyhelyzetnek* nevezzük.
Ha az x_1x_2=01 bemenet hatására létrejövő állapotátmenet vizsgáljuk, akkor azt látjuk, hogy akár a 01, akár az 10 állapotba kerül a hálózat, onnan végül eléri a kívánt 11 állapotot, igaz, közbenső, nem kívánt állapotokon keresztül. A jelenséget ilyenkor *nem-kritikus versenyhelyzetnek nevezzük*. A versenyhelyzetnek ez az „enyhébb" formája szintén nem előnyös, mivel a kimeneten könnyen okozhat szándékolatlan impulzusokat, ugyanis a közbenső instabil állapotokhoz tartozó kimenetek átmenetileg felléphetnek.
A kritikus versenyhelyzeteket azonban meg kell szüntetni a hálózatokban. Ehhez elsősorban fel kell ismerni a versenyhelyzeteket. A lehetséges versenyhelyzeteket úgy ismerhetjük fel, hogy megvizsgáljuk a kódolt állapottáblát, hogy az aktuális állapotból történik-e olyan állapotba való átmenet valamely bemeneti kombináció hatására, amely két helyiértéken különbözik a fennálló állapottól. (Azaz a 00 állapot sorában van-e 11-be való átmenet, a 01 sorában van-e 10-ba való átmenet stb.).
A versenyhelyzet megszüntetésének a legegyszerűbb módját a az 11 állapotból a 00 állapotba való, 11 bemeneti kombináció hatására történő váltás esetében tanulmányozhatjuk. Mivel a szándékolatlanul bekövetkező 01 és 10 állapotok esetén az állapotváltozás nem definiált, ezért a nem-definiált átmenetet átírhatjuk oly módon, hogy a kritikus versenyhelyzetet nem-kritikus versenyhelyzetté alakítsuk, a következőképpen:

y_1y_2 \ x_1x_2	00	01	11	10
00	(00)	11	00	11
01		11	**11**	(01)
11		(11)	00	(11)
10		11	**11**	11

Az 10 bemeneti kombináció esetén létrejövő versenyhelyzetet ilyen módon nem tudjuk megoldani: a specifikált működést ugyanis nem írhatjuk át, mert azzal megváltoztatnánk a megoldandó logikai feladatot. Az egyik lehetséges megoldást ilyenkor a kódolás megváltoztatása jelenti. Mint azt már korábban említettük a kódoknak az egyes állapotokhoz való rendelése tetszőleges. Ha az eredetileg 11 kombinációval kódolt állapotot 10-val kódolnánk, akkor a 00 és az 10 állapot egymástól csak egy helyiértéken különbözne, ezért a versenyhelyzet sem lépne fel. Megfelelő állapotkódolás megválasztásával ezért gyakran kiküszöbölhető a kritikus versenyhelyzet. Természetesen az állapotok átkódolása esetén ismét ellenőriznünk kell a versenyhelyzeteket, mert a kódolás megváltoztatásának hatására lehet, hogy olyan helyen fordul elő versenyhelyzet, ahol korábban ez nem jelentkezett
Előfordulhat azonban olyan eset is, hogy akármilyen kódolást is választunk, nem tudjuk elkerülni a kritikus versenyhelyzetet. Ilyenkor egy többlet állapotváltozó (bemeneti változó) felvé-

telével tudjuk megoldani a problémát. Három állapotváltozó ugyanis minden állapotkódnak három olyan szomszédos kódja van, amelyek csak 1 helyiértéken térnek el egymástól (pl. 000, 001, 010 és 100).
Mérnöki szemszögből vizsgálva természetesen megoldást jelent a hálózat szinkronizálása is, ugyanis szinkron hálózat esetén egyáltalán nem kell számolnunk versenyhelyzetekkel.

3.5.3. Megvalósítás

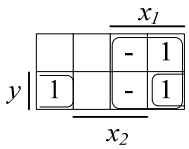
A kimeneti függvények megvalósítása aszinkron hálózatok esetén ugyanazzal az eljárással történik, mint szinkron hálózatok esetén. Arra azonban felhívjuk a figyelmet, hogy aszinkron hálózatok esetében mindig hazárdmentes megvalósítást kell keresnünk, mert a hazárdjelenségek további, a versenyhelyzethez hasonló tranziens jelenségeket okozhat az aszinkron sorrendi hálózatokban.
Az aszinkron sorrendi hálózatok megvalósítása két módon lehetséges:

- aszinkron tárolók felhasználásával (amint azt a 3.3. pontban tárgyaltuk, aszinkron működésre az SR és a DG tároló alkalmas), vagy
- visszacsatolt kombinációs hálózatként.

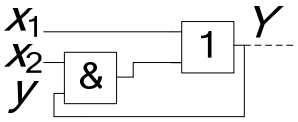
Az aszinkron tárolókkal történő megvalósítás nem különbözik a szinkron hálózatok esetében megtárgyalt eljárástól: ugyanúgy vezérlési táblát kell készíteni, majd abból Karnaugh-táblák segítségével meg lehet határozni a vezérlő függvényeket. Ezen függvények esetében is ügyelni kell arra, hogy a megvalósítás hazárdmentes legyen.
Visszacsatolt kombinációs hálózatként történő megvalósítás esetén tulajdonképpen fizikailag is a 3.1.2. pontban leírt és ábrázolt működést valósítjuk meg: az egyes állapotváltozók fizikailag is visszacsatolódnak a hálózat bemenetére, így nincs szükség külön tároló elemek alkalmazására. A tervezés kiindulópontja ebben az esetben a kódolt állapottábla, amely már nem tartalmaz versenyhelyzetet. A kódolt állapottáblából közvetlenül felírhatjuk az $Y=f_y(X,y)$ függvényt. Példaként vegyük az alábbi kódolt állapottáblát (a kimenetekkel most nem foglalkozunk):

y \ x_1x_2	00	01	11	10
0	(0)/00	(0)/11	-/--	1/10
1	(1)/01	0/11	-/--	(1)/10

Az állapotkódok közvetlenül átírhatók egy Karnaugh-táblába, amelynek peremezését az y, az x_1 és az x_2 adja:



Ebből felírható a függvény: $Y = x_1 + y\bar{x}_2$. Az y állapotváltozót úgy kapjuk meg, ha az Y új belső állapotváltozót visszacsatoljuk a hálózat bemenetére, fenti példában a következőképpen:



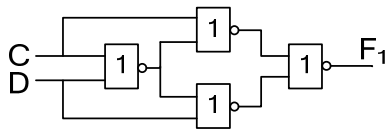
Amennyiben kettőnél több állapotú rendszer, akkor természetesen minden állapotváltozónak meg kell határozni a függvényét és azokat egyenként kell visszacsatolni a hálózat megfelelő bemenetére. Itt is felhívjuk a figyelmet arra, hogy az állapotfüggvények esetében is hazárdmentes kapcsolásokat kell alkotni az aszinkron hálózat esetleges tranziens jelenségeinek elkerülése érdekében.

4. PÉLDATÁR

4.1. Kombinációs hálózatok

4.1.1. Példa

Milyen logikai kapcsolatot valósít meg az F_1 kimenetű hálózat és az F_2 függvény? Írja fel az F_3 függvény legegyszerűbb alakját az F_1 és az F_2 függvények változóival, valamint rajzolja fel a függvényt megvalósító kapcsolást legfeljebb 3 darab, bármilyen típusú két-bemenetű kapuval. Adja meg a függvények MINTERM Karnaugh-tábláját is. Mi a kapcsolat a három függvény között?



$$F_2 = (A + \overline{B})(\overline{A} + B)$$

$$F_3 = \prod^{4}(1,2,4,7,8,11,13,14)$$

Megoldás:
Az F_1 hálózat függvényét a DeMorgan azonosság segítségével bonthatjuk ki:

$$F_1 = \overline{\overline{C + \overline{C + D} + D + \overline{C + D}}} = (C + \overline{C}\,\overline{D})(D + \overline{C}\,\overline{D}) = (C + \overline{D})(D + \overline{C}) =$$
$$CD + \overline{C}\,\overline{D} = C \bullet D$$

$$F_2 = (A + \overline{B})(\overline{A} + B) = AB + \overline{A}\,\overline{B} = A \bullet B$$

Ahhoz, hogy a másik két függvénnyel azonos alakban legyen, az F_3 függvényt minterm alakra hozzuk:

$$F_3 = \prod^{4}(1,2,4,7,8,11,13,14)$$

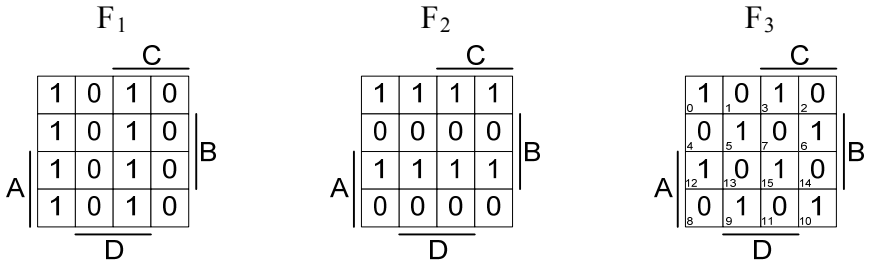
$$\overline{F_3} = \prod^{4}(0,3,5,6,9,10,12,15)$$

$$F_3 = \sum^{4}(0,3,6,9,10,12,15)$$

A minterm alakot felírjuk algebrai alakban, majd egyszerűsítjük, amelyből megkapjuk a három függvény összefüggését:

$$F_3 = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + AB\overline{C}\,\overline{D} + ABCD + A\overline{B}CD + A\overline{B}C\overline{D} =$$
$$= AB(\overline{C}\,\overline{D} + CD) + \overline{A}\,\overline{B}(\overline{C}\,\overline{D} + CD) + \overline{A}B(\overline{C}D + C\overline{D}) + A\overline{B}(\overline{C}D + C\overline{D}) =$$
$$= (A \bullet B)(C \bullet D) + (A \oplus B)(C \oplus D) = (A \bullet B) \bullet (C \bullet D) = F_1 \bullet F_2$$

Tehát, az F_3 felírható az F_1 és az F_2 függvények ekvivalenciájaként. Ugyanezt a megállapítást tehetjük meg, amennyiben felrajzoljuk a három függvény minterm Karnaugh tábláját:



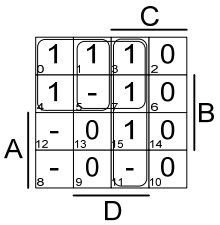
A Karnaugh táblákból látható, hogy az F_3 azokon a mintermeken vesz fel 1 kimenetet, ahol a másik két függvény kimenete megegyezik, ami megfelel az ekvivalencia definíciójának.

4.1.2. Példa

Valósítsa meg az alábbi, MINTERM alakjával megadott függvényt!

$$F = \sum^{4}(0,1,3,4,7,15) + (5,8,11,12)$$

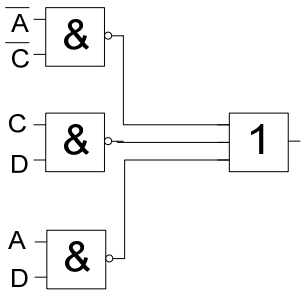
Megoldás:
Vegyük fel a függvény Karnaugh tábláját! A közömbös kimenetek megfelelő felhasználásával a következő legegyszerűbb lefedést választhatjuk:



Így az F függvény az alábbi alakban írható fel:

$$F = \bar{A}\bar{C} + DC + \bar{A}D$$

A függvényhez tartozó egyszerű, készintes logikai hálózat a következő:



4.1.3. Példa

Valósítsa meg az F_1 és F_2 függvények ekvivalenciájaként előállított függvényt, kizárólag NAND kapuk felhasználásával.

$$F_1 = \sum^4(1,2,4,5,7,9) \qquad F_2 = \prod^4(0,1,8,9,10,11)$$

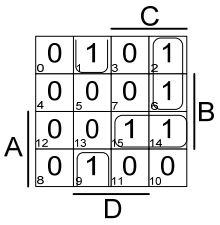
Az F_2 függvényt hozzuk minterm alakra:

$$\overline{F}_2 = \prod^4(2,3,4,5,6,7,12,13,14,15),\ F_2 = \sum^4(0,1,2,3,8,9,10,11,12,13)$$

A két függvény ekvivalenciája azon termek összessége, amelyek vagy mindkét függvényben, vagy egyikben sem szerepelnek:

$$F_1 \bullet F_2 = \sum^4(1,2,6,9,14,15)$$

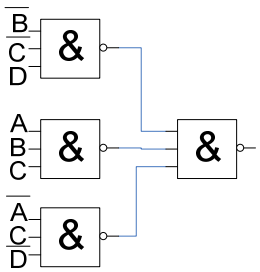
Innen már egyszerűen felrajzolható az eredmény Karnaugh táblája, jelen esetben nem hazárdmentes összevonásokkal, illetve annak algebrai alakja:



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

A NAND kapukkal való megvalósításhoz a három tagra alkalmazott DeMorgan azonossággal juthatunk el:

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

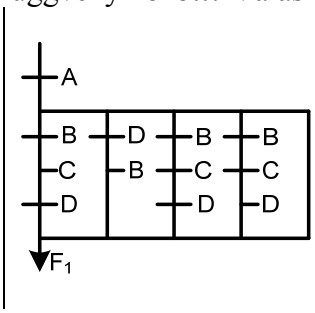


4.1.4. Példa

Mi a kapcsolat az alábbi három logikai függvény között? Válaszát indokolja!

$$F_2 = \prod^{4}(1,3,4,5,6,7,9,11,12,14)$$

$$F_3 = ABD$$



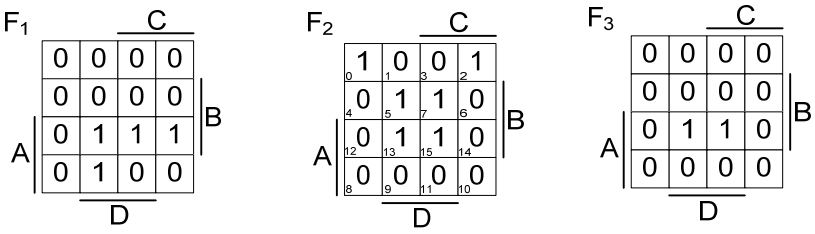
Megoldás

A függvényeket közös minterm alakra hozzuk:

$$F_1 = A\left(B\overline{C}D + D\overline{B} + BCD + BC\overline{D}\right) = A\left(BC\left(\overline{D} + D\right) + B\overline{C}D + D\overline{B}\right) =$$
$$= A\left(BC + B\overline{C}D + D\overline{B}\right) = A\left(B\left(C + \overline{C}D\right) + D\overline{B}\right) = A\left(BC + BD + \overline{C}D\right) =$$
$$= ABC + ABD + A\overline{C}D$$

$$\overline{F_2} = \prod^{4}(0,2,8,10,13,15),\ F_2 = \sum^{4}(0,2,5,7,13,15)$$

Majd ábrázoljuk őket Karnaugh táblán:



Látható, hogy a három függvény kapcsolata:

$$F_3 = F_1 F_2$$

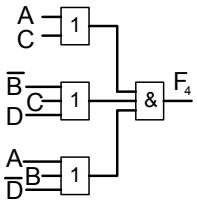
4.1.5. Példa

Mi a kapcsolat az alábbi függvények között? Válaszát indokolja!

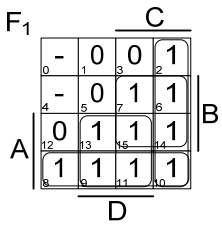
$$F_1 = \sum^{4}(2,6,7,8,9,10,11,13,14,15) + (0,4)$$

$$F_2 = \prod^{4}(0,1,2,4,5,6,7,8,9,13)$$

$$F_3 = C(B + \overline{D}) + A(\overline{B} + D)$$



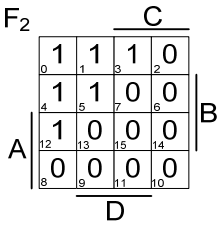
Megoldás



$$F_1 = A\overline{B} + C\overline{D} + BC + AD$$

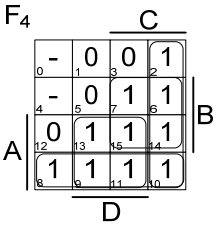
$$\overline{F_2} = \prod^{4}(3,10,11,12,14,15)$$

$$F_2 \sum^{4}(0,1,3,4,5,12)$$



$$F_3 = C(B + \overline{D}) + A(\overline{B} + D) = CB + C\overline{D} + A\overline{B} + AD$$

$$F_4 = (A + C)(\overline{B} + C + D)(A + B + \overline{D}) =$$
$$= (A\overline{B} + C\overline{B} + AC + C + AD + DC)(A + B + \overline{D}) =$$
$$= (C(\overline{B} + A + 1 + D) + A\overline{B} + AD)(A + B + \overline{D}) =$$
$$= AC + A\overline{B} + AD + BC + ADB + \overline{D}C + A\overline{B}\overline{D} =$$
$$= AD(1 + B) + A\overline{B}(1 + \overline{D}) + AC + BC + \overline{D}C$$



A függvények közötti kapcsolat:

$$F_1 = \overline{F_2} = F_3 = F_4$$

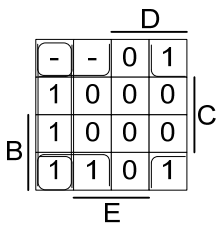
4.1.6. Példa

Valósítsa meg az alábbi függvényt két bemenetű AND és OR kapukkal, illetve két bemenetű NAND kapukkal!

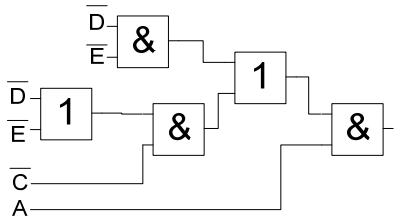
$$F = \sum^{5}(18,20,24,25,26,28) + (16,17)$$

Megoldás

a)



$$F = A\left(\overline{E}\,\overline{D} + \overline{C}\,\overline{D} + \overline{C}\,\overline{E}\right)$$

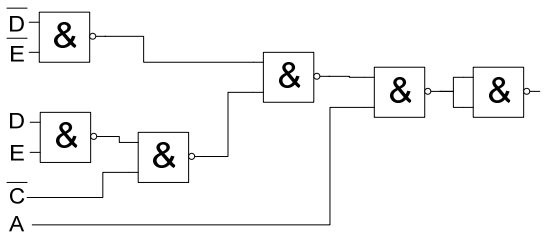


b)

$$F = A\left(\overline{C}\left(\overline{\overline{D} + \overline{E}}\right) + \overline{D} + \overline{E}\right) =$$

$$A\left(\overline{\overline{C}\,\overline{D}\,\overline{E} + \overline{D}\,\overline{E}}\right) =$$

$$A\left(\overline{\overline{C}\,\overline{D}\,\overline{E}\,\overline{D}\,\overline{E}}\right)$$



4.1.7. Példa

Írja fel az F_1 és F_2 függvények és kapcsolataként előálló logikai függvényt. Valósítsa meg a logikai függvényt tetszőleges kapuk felhasználásával!

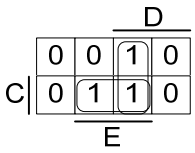
$$F_1 = \sum^{5}(0,1,3,5,7,16,21,22)$$

$$F_2 = \prod^{5}(0,2,3,4,5,6,7,8,9,10,12,14,15,16,17,18,19,20,21,22,23,25,27,30,31)$$

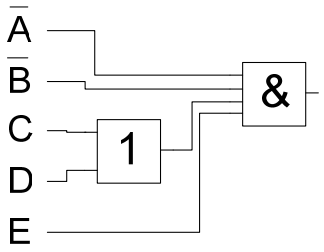
Megoldás

$$\overline{F_2} = \prod^{5}(1,11,13,24,26,28,29),\ F_2 = \sum^{5}(2,3,5,7,18,20,30)$$

$$F_1F_2 = \sum^{5}(3,5,7)$$



$$F_1F_2 = \overline{A}\overline{B}(CE + DE) = \overline{A}\overline{B}E(C + D)$$



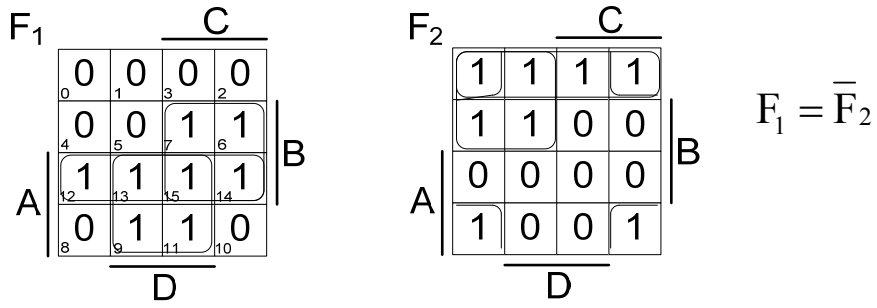
4.1.8. Példa

Adott a mellékelt két logikai függvény. Valósítsa meg a függvényeket NAND kapuk felhasználásával. Van-e összefüggés a két függvény között?

$$F_1 = \sum^{4}(6,7,9,11,12,13,14,15)$$
$$F_2 = \overline{A}\overline{B} + \overline{A}\overline{C} + \overline{B}\overline{D}$$
$$A = 2^3; B = 2^2; C = 2^1; D = 2^0;$$

Megoldás

A két függvény Karnaugh táblán történő ábrázolása után látható, hogy a két függvény egymás negáltja:

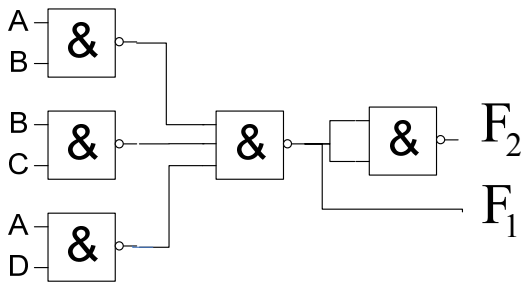


$$F_1 = \overline{F}_2$$

A NAND kapus megvalósításhoz a DeMorgan azonosságot alkalmazzuk F_1-re:

$$F_1 = AB + BC + AD = \overline{\overline{AB}\,\overline{BC}\,\overline{AD}}$$

Így egy hálózatban felrajzolható mindkét függvény:

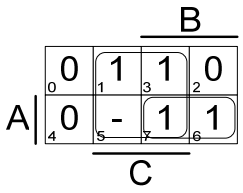


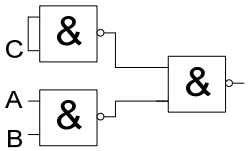
4.1.9. Példa

Valósítsa meg az alábbi függvényt hazárdmentesen, kizárólag NAND-kapuk felhasználásával.

$$F_1 = \sum^{3}(1,3,6,7)+(5)$$

Megoldás

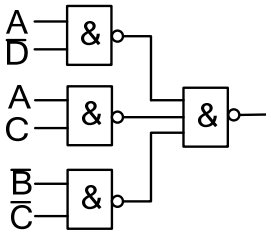




$$F_1 = C + AB = \overline{\overline{C}\,\overline{AB}}$$

4.1.10. Példa

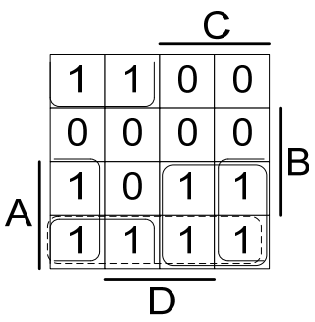
Vizsgálja meg az ábrázolt logikai hálózatot! Keresse meg azt a bemeneti kombináció váltást, amelynél statikus hazárd lép fel. Hazárdmentesítse a hálózatot.



Megoldás

$$F = \overline{\overline{A\overline{D}}\,\overline{AC}\,\overline{B\overline{C}}} = A\overline{D} + AC + \overline{B}\overline{C} + (A\overline{B})$$

A Karnaugh táblán az eredeti függvény folyamatos, a hazárdmentesítés szaggatott vonallal van jelölve:



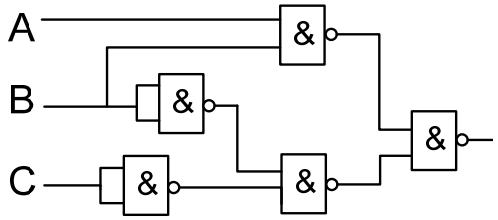
A kiküszöbölt kombinációváltás:

$ABCD = 1001$

$ABDC = 1011$

4.1.11. Példa

Vizsgálja meg az ábrázolt logikai hálózatot! Keresse meg azt a bemeneti kombináció váltást, amelynél statikus hazárd lép fel. Rajzolja fel a bemeneti és a kimeneti jelek időbeli alakulását egy hazárdot okozó váltásnál. Hazárd-mentesítse a hálózatot.



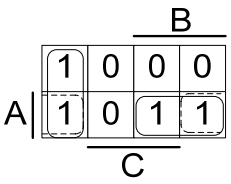
Megoldás

$$F = \overline{\overline{AB}\,\overline{BC}} = AB + \overline{B}\,\overline{C} + (A\overline{C})$$

Kombinációváltás:

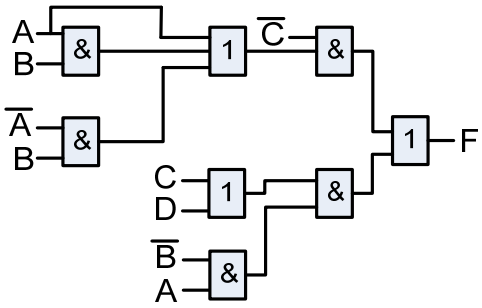
$ABC = 110$

$ABC = 100$



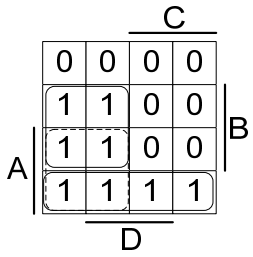
4.1.12. Példa

Tervezzen 2-bemenetű NAND-kapuk felhasználásával olyan hazárdmentes áramkört, amely ugyanazt a függvényt valósítja meg, mint mellékelt kapcsolás.



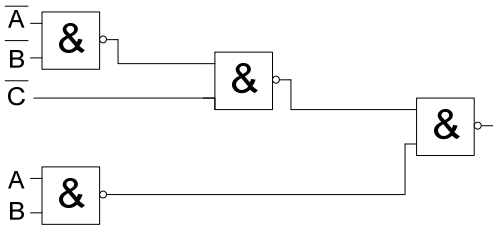
Megoldás

$$F = \left(AB + \overline{A}B + A\right)\overline{C} + A\overline{B}\left(C + D\right) = \overline{C}AB + \overline{C}\,\overline{A}B + A\overline{C} + A\overline{B}C + A\overline{B}D =$$
$$= A\overline{C}\left(B + 1\right) + \overline{C}\,\overline{A}B + A\overline{B}C + A\overline{B}D$$



$$F = A\overline{B} + B\overline{C} + A\overline{C} =$$
$$\overline{\overline{C}\left(\overline{A + B}\right)} + AB =$$
$$\overline{\overline{C}AB}\,\overline{AB}$$

$$F = A\overline{B} + B\overline{C} + A\overline{C} = \overline{\overline{C}\left(\overline{A + B}\right)} + AB = \overline{\overline{C}AB}\,\overline{AB}$$



4.1.13. Példa

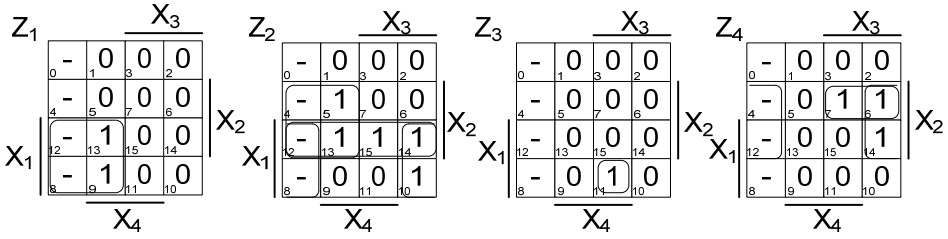
Tervezzen osztó áramkört! Az áramkörnek négy bemenete: x_1, x_2, x_3, x_4 (rendre 20, 21, 22, 23) és négy kimenete: Z_1, Z_2, Z_3, Z_4 van. Az áramkör feladata, hogy az x_1 és x_2 bemeneteken binárisan kódolt számot elossza az x_3 és x_4 bemeneteken binárisan kódolt számmal. Az osztás egész részét az áramkör a Z_1 és Z_2 kimeneteken binárisan kódolva adja ki, az osztás maradékát pedig a Z_3 és Z_4 kimeneteken binárisan kódolva adja ki. Az értelmezhetetlen osztások esetén minden kimenet közömbös. Valósítsa meg az áramkört a lehető legegyszerűbben AND és OR kapuk segítségével! Rajzolja fel a kapcsolást!

Megoldás

Igazságtábla

	X_1	X_2	X_3	X_4	A	B	A/B	MAR	Z_1	Z_2	Z_3	Z_4
0	*0*	*0*	*0*	*0*	*0*	*0*	-	-	-	-	-	-
1	*0*	*0*	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*
2	*0*	*0*	*1*	*0*	*0*	*2*	*0*	*0*	*0*	*0*	*0*	*0*
3	*0*	*0*	*1*	*1*	*0*	*3*	*0*	*0*	*0*	*0*	*0*	*0*
4	*0*	*1*	*0*	*0*	*1*	*0*	-	-	-	-	-	-
5	*0*	*1*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*1*	*0*	*0*
6	*0*	*1*	*1*	*0*	*1*	*2*	*0*	*1*	*0*	*0*	*0*	*1*
7	*0*	*1*	*1*	*1*	*1*	*3*	*0*	*1*	*0*	*0*	*0*	*1*
8	*1*	*0*	*0*	*0*	*2*	*0*	-	-	-	-	-	-
9	*1*	*0*	*0*	*1*	*2*	*1*	*2*	*0*	*1*	*0*	*0*	*0*
10	*1*	*0*	*1*	*0*	*2*	*2*	*1*	*0*	*0*	*1*	*0*	*0*
11	*1*	*0*	*1*	*1*	*2*	*3*	*0*	*2*	*0*	*0*	*1*	*0*
12	*1*	*1*	*0*	*0*	*3*	*0*	-	-	-	-	-	-
13	*1*	*1*	*0*	*1*	*3*	*1*	*3*	*0*	*1*	*1*	*0*	*0*
14	*1*	*1*	*1*	*0*	*3*	*2*	*1*	*1*	*0*	*1*	*0*	*1*
15	*1*	*1*	*1*	*1*	*3*	*3*	*1*	*0*	*0*	*1*	*0*	*0*

Az egyes kimenetek Karnaugh táblái:

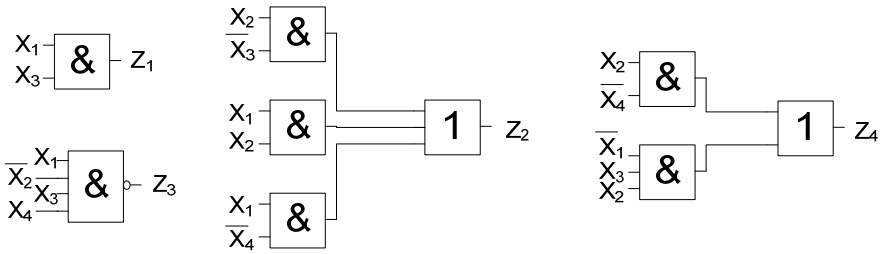


A függvények algebrai alakban:

$$Z_1 = X_1\overline{X}_3 \qquad Z_2 = X_2\overline{X}_3 + X_1X_2 + X_1\overline{X}_4$$

$$Z_3 = X_1\overline{X}_2X_3X_4 \qquad Z_4 = X_2\overline{X}_4 + \overline{X}_1X_2X_3$$

Hálózat:



4.1.14. Példa

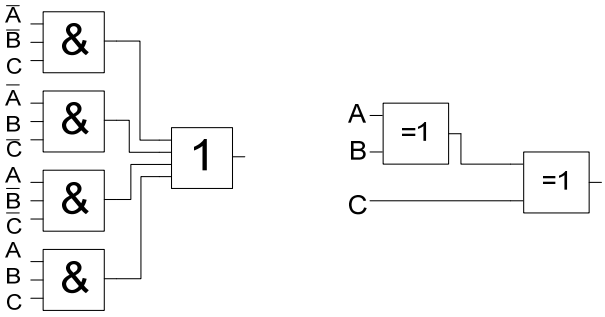
Tervezzen paritásbit generátort, amely egy 3 bites kódszó “1”-eit páros számú “1”-re egészíti ki! (A 000 kombináció párosnak tekintendő, paritásbitje “0”.)Valósítsa meg a generátort VAGY kapukkal és ÉS kapukkal!
Igazolja algebrailag, hogy a generátor megépíthető mindössze 2 darab 2 bemenetű antivalencia kapu felhasználásával is!Rajzolja fel ezt a megoldást is!

Megoldás

Igazságtábla:

	X_1	X_2	X_3	F
0	*0*	*0*	*0*	*0*
1	*0*	*0*	*1*	*1*
2	*0*	*1*	*0*	*1*
3	*0*	*1*	*1*	*0*
4	*1*	*0*	*0*	*1*
5	*1*	*0*	*1*	*0*
6	*1*	*1*	*0*	*0*
7	*1*	*1*	*1*	*1*

$$F = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC = C\left(\overline{A}\,\overline{B} + AB\right) + \overline{C}\left(A\overline{B} + \overline{A}B\right) = C \oplus \left(A \oplus B\right)$$



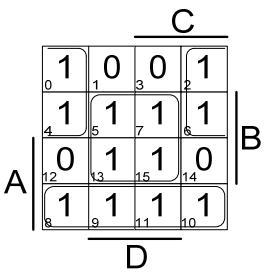
4.1.15. Példa

Adott négy kapcsoló, amelyek mindegyikéhez egy-egy értéket rendelünk: A=7, B=6, C=5, D=4. Írja fel azt a logikai függvényt, amelynek értéke akkor és csak akkor logikai „1”, ha a lenyomott kapcsolókhoz tartozó számok összege maradék nélkül, vagy legfeljebb 2 maradékkal osztható 5-tel. (A Karnaugh táblához: $A=2^3$, $B=2^2$, $C=2^1$, $D=2^0$)!

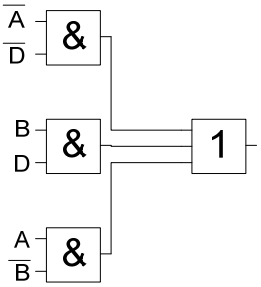
Megoldás

Igazságtábla:

	A	B	C	D	S	Mar	F
0	*0*	*0*	*0*	*0*	*0*	*0*	*1*
1	*0*	*0*	*0*	*1*	*4*	*4*	*0*
2	*0*	*0*	*1*	*0*	*5*	*0*	*1*
3	*0*	*0*	*1*	*1*	*9*	*4*	*0*
4	*0*	*1*	*0*	*0*	*6*	*1*	*1*
5	*0*	*1*	*0*	*1*	*10*	*0*	*1*
6	*0*	*1*	*1*	*0*	*11*	*1*	*1*
7	*0*	*1*	*1*	*1*	*15*	*0*	*1*
8	*1*	*0*	*0*	*0*	*7*	*2*	*1*
9	*1*	*0*	*0*	*1*	*11*	*1*	*1*
10	*1*	*0*	*1*	*0*	*12*	*2*	*1*
11	*1*	*0*	*1*	*1*	*16*	*1*	*1*
12	*1*	*1*	*0*	*0*	*13*	*3*	*0*
13	*1*	*1*	*0*	*1*	*17*	*2*	*1*
14	*1*	*1*	*1*	*0*	*18*	*3*	*0*
15	*1*	*1*	*1*	*1*	*22*	*2*	*1*

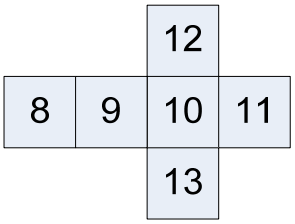


$$F = \overline{A}\,\overline{D} + BD + A\overline{B}$$

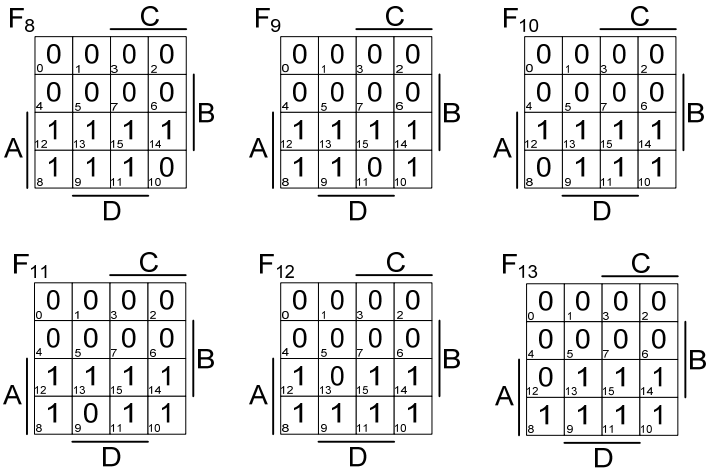


4.1.16. Példa

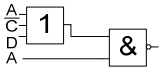
Az ábra egy kocka kiterített 6 oldalát ábrázolja. A kocka oldalain látható szám egy-egy termet jelent. A kocka minden oldalához egy-egy logikai függvényt rendelünk (F8, F9, F10, F11, F12, F13) a következő szabály szerint: minden függvény tartalmazza a (14, 15) termeket, valamint a kocka adott oldalához és a vele szomszédos oldalakhoz rendelt termeket. Írja fel az így kapott logikai függvények teljes, szabályos MINTERM alakját. Realizálja a függvényeket maximum 6 db kétbemenetű és 6 db hárombemenetű, bármilyen típusú kapu felhasználásával.



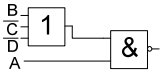
Megoldás



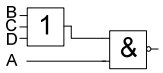
$$F_8 = A\overline{A\overline{B}C\overline{D}} = A\left(B + \overline{C} + D\right)$$



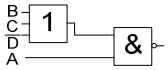
$$F_9 = A\overline{A\overline{B}CD} = A\left(B + \overline{C} + \overline{D}\right)$$



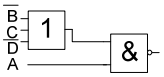
$$F_{10} = A\overline{A\overline{B}\overline{C}\overline{D}} = A\left(B + C + D\right)$$



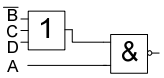
$$F_{11} = A\overline{A\overline{B}\overline{C}D} = A\left(B + C + \overline{D}\right)$$



$$F_{12} = A\overline{AB\overline{C}\overline{D}} = A\left(\overline{B} + C + \overline{D}\right)$$



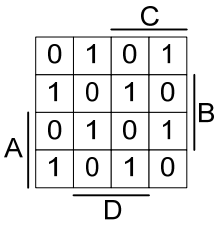
$$F_{13} = A\overline{AB\overline{C}\overline{D}} = A\left(\overline{B} + C + D\right)$$



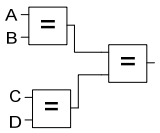
4.1.17. Példa

Aladár és Béla, Cecil és Dóra ellen játszik egy speciális játékkal. A játék négy nyomógombot tartalmaz, és abban az esetben ad pontot Aladárnak és Bélának (Z=0), ha a négyük által megnyomott nyomógombok száma páros, ellenkező esetben Cecil és Dóra kapják a pontokat. Tervezzen kombinációs hálózatot, amely a játék kimenetét valósítja meg maximum 3 bármilyen típusú kapuáramkör felhasználásával.

Megoldás



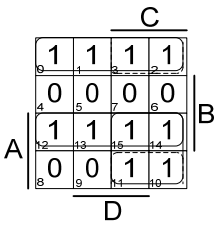
$$F = \overline{A}\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + AB\overline{C}D + ABC\overline{D} + A\overline{B}C\overline{D} + A\overline{B}CD =$$
$$= \overline{A}\overline{B}(C \oplus D) + AB(C \oplus D) + \overline{A}B(C \bullet D) + A\overline{B}(C \bullet D) =$$
$$= (A \oplus B) \oplus (C \oplus D)$$



4.1.18. Példa

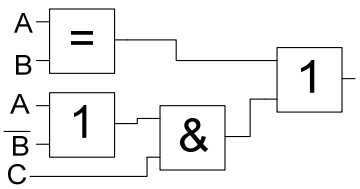
Adott egy négybites kódszó. Tervezze meg azt a hazárdmentes kombinációs hálózatot, amely akkor és csak akkor ad logikai egy értéket a kimenetén, ha a kódszó értelmezhető, mint a négy legkisebb BCD szám, valamint ha a kódszó BDC számként nem értelmezhető érték. Írja fel a függvényt A, B, C, D logikai változókkal. Valósítsa meg a kapcsolást hazárdmentesen, maximum 4 db bármilyen típusú csak két bemenetű kapuval, majd realizálja kétszintű hálózatként csak NAND kapuk felhasználásával. ($A=2^3$, $B=2^2$, $C=2^1$, $D=2^0$)

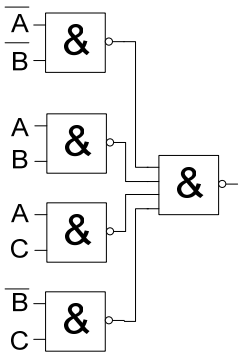
Megoldás



$$F = \overline{A}\,\overline{B} + AB + \overline{B}C + AC = A \bullet B + C\left(A + \overline{B}\right)$$

$$F = \overline{\overline{A}\,\overline{B} + AB + \overline{B}C + AC} = \overline{\overline{A}\,\overline{B}}\;\overline{AB}\;\overline{AC}\;\overline{\overline{B}C}$$





4.1.19. Példa

Tervezzen kódváltót megvalósító négy bemenetű, négy kimenetű (F_1, F_2, F_3, F_4) kombinációs hálózatot, amely a következő feltételek szerint működik ($X_1=2^3$, $X_2=2^2$, $X_3=2^1$, $X_4=2^0$):
F_1=1, (X_1=0 és X_2=1) vagy (X_1=1 és X_2=0) F_2=1, (X_3=0 és X_4=0) vagy (X_3=1 és X_4=1)
F_3=1, (X_1=0 és X_2=0) vagy (X_1=1 és X_2=1) F_4=1, (X_3=0 és X_4=1) vagy (X_3=1 és X_4=0).
Valósítsa meg a feladatot
a,/ 2 bemenetű AND és OR kapuk segítségével,
b,/ maximum 4 db 2 bemenetű bármilyen kapuval,
c,/ jelfogós hálótattal!

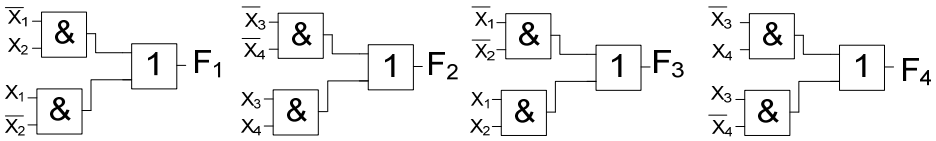
Megoldás

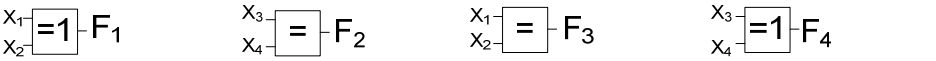
$$F_1 = \overline{X_1}X_2 + X_1\overline{X_2} = X_1 \oplus X_2$$

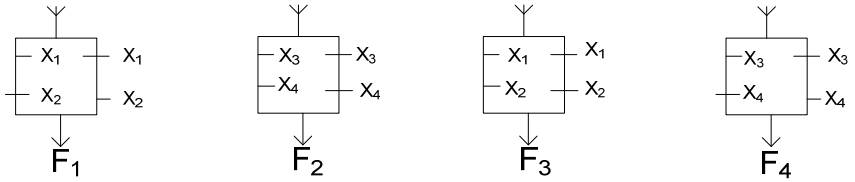
$$F_2 = \overline{X_3}\,\overline{X_4} + X_3X_4 = X_3 \bullet X_4$$

$$F_3 = \overline{X_1}\,\overline{X_2} + X_1X_2 = X_1 \bullet X_2$$

$$F_4 = \overline{X_3}X_4 + X_3\overline{X_4} = X_3 \oplus X_4$$







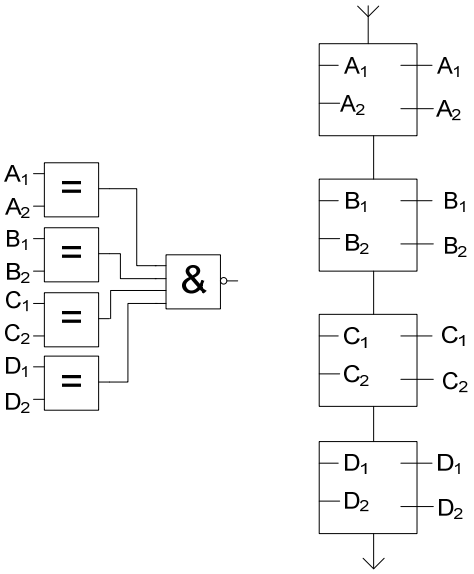
4.1.20. Példa

Adott két 4 bites kódszó: $A_1B_1C_1D_1$ és $A_2B_2C_2D_2$. Írja fel azt a logikai függvényt, amely akkor ad logikai 1 értéket a kimenetén, ha a két kódszó bitenként egymás ekvivalense! Valósítsa meg a kapcsolást:

- maximum 5 kapuáramkör felhasználásával
- jelfogókkal.

Megoldás

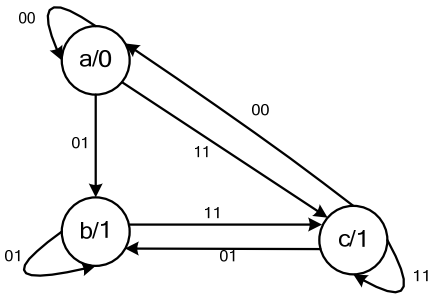
$$F = (A_1 \bullet A_2)(B_1 \bullet B_2)(C_1 \bullet C_2)(D_1 \bullet D_2)$$



4.2. Szinkron sorrendi hálózatok

4.2.1. Példa

Tervezzen szinkron sorrendi hálózatot, amely a megadott állapotgráf szerint működik. Valósítsa meg a kapcsolást minden típusú tároló felhasználásával.



Megoldás

Előzetes állapottábla

y_1y_2 \ x_1x_2	00	01	11	10	Z	
a	a	b	c	-	**0**	A
b	-	b	c	-	**1**	B
c	a	b	c	-	**1**	

Összevont állapottábla

y \ x_1x_2	00	01	11	10	Z
A	A	B	B	-	**0**
B	A	B	B	-	**1**

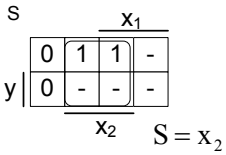
Kódolt állapottábla

y \ x_1x_2	00	01	11	10	Z
0	0	1	1	-	**0**
1	0	1	1	-	**1**

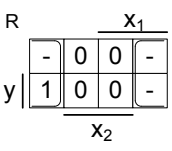
$Z = y$

Megvalósítás SR tárolóval:

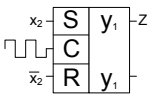
y \ x_1x_2	00	01	11	10
0	0-	10	10	--
1	01	-0	-0	--



$S = x_2$

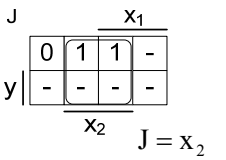


$R = \overline{x}_2$

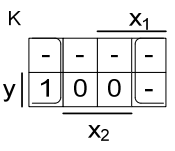


Megvalósítás JK tárolóval:

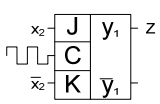
y \ x_1x_2	00	01	11	10
0	0-	1-	1-	--
1	-1	-0	-0	--



$J = x_2$

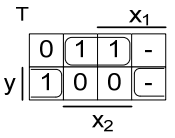


$K = \overline{x}_2$

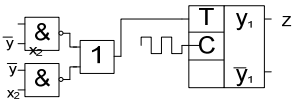


Megvalósítás T tárolóval:

y \ x_1x_2	00	01	11	10
0	0	1	1	-
1	1	0	0	-

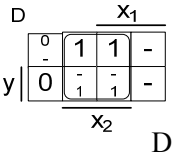


$$T = x_2\bar{y} + y\overline{x_2}$$

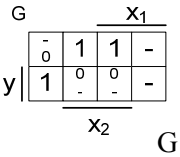


Megvalósítás DG tárolóval:

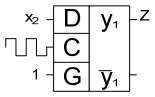
y \ x_1x_2	00	01	11	10
0	0- -0	11	11	--
1	01	-0 1-	-0 1-	--



$$D = x_2$$

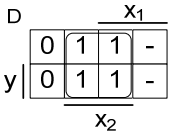


$$G = 1$$

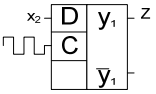


Megvalósítás D tárolóval:

y \ x_1x_2	00	01	11	10
0	0	1	1	-
1	0	1	1	-

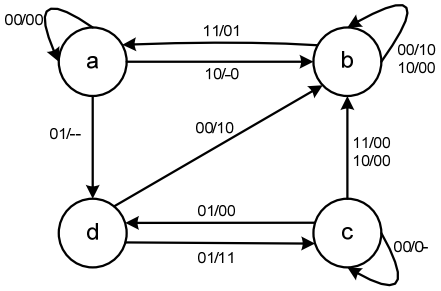


$$D = x_2$$



4.2.2. Példa

Valósítsa meg az ábrán látható állapotgráffal megadott szinkron sorrendi hálózatot JK tárolóelemek segítségével.

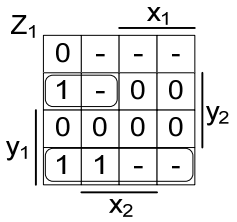


Megoldás

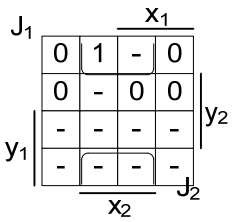
y_1y_2 \ x_1x_2	00	01	11	10
a	a/00	d/--	-/--	b/-0
b	b/10	-/--	a/01	b/00
c	c/0-	d/00	b/00	b/00
d	b/10	c/11	-/--	-/--

y_1y_2 \ x_1x_2	00	01	11	10
00	00/00	10/--	--/--	01/-0
01	01/10	-/--	00/01	01/00
11	11/0-	10/00	01/00	01/00
10	01/10	11/11	--/--	--/--

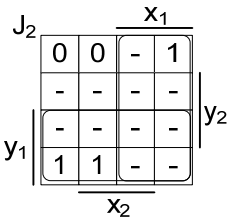
y_1y_2 \ x_1x_2	00	01	11	10
00	0- 0-	1- 0-	-- --	0- 1-
01	0- -0	-- --	0- -1	0--0
11	-0 -0	-0 -1	-1 -0	-1 -0
10	-1 1-	-0 1-	-- --	-- --



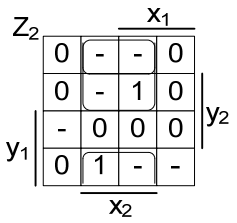
$$Z_1 = \bar{y}_1 y_2 \bar{x}_1 + y_1 y_2$$



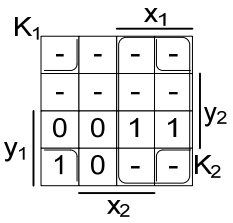
$$J_1 = x_2 \bar{y}_2$$



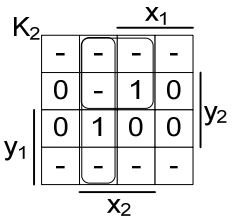
$$J_2 = y_1 + x_1$$



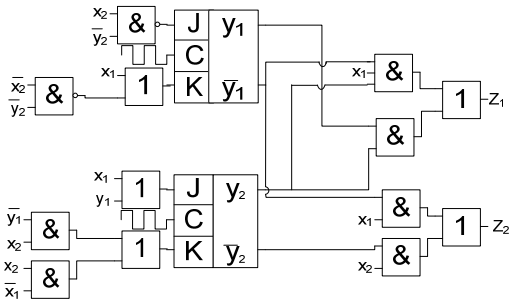
$$Z_2 = x_2 \bar{y}_1 + \bar{y}_2 x_2$$



$$K_1 = x_1 + \bar{y}_2 \bar{x}_2$$

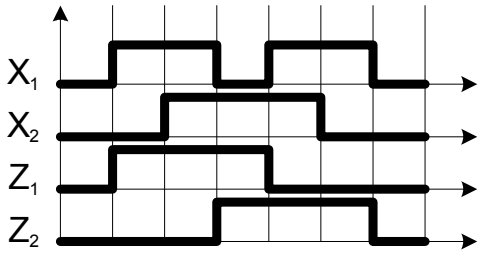


$$K_2 = \bar{y}_1 x_2 + x_2 \bar{x}_1$$


x2
ȳ2
&
J
C
K
y1
ȳ1
x1
1
x̄2
ȳ2
&
x1
&
1
z1
&
x1
1
y1
1
J
C
K
y2
ȳ2
ȳ1
x2
&
1
x1
&
x2
x̄1
&
x2
&
1
z2
&

4.2.3. Példa

Tervezzen szinkron sorrendi hálózatot, amely a megadott idődiagram szerinti ciklusban működik! Valósítsa meg a kapcsolást JK tárolóval! A Z1 függvényt Mealy, a Z2 függvényt pedig Moore modell szerint valósítsa meg!



Megoldás

y \ x1x2	00	01	11	10	Z2	
a	a/0	-/-	-/-	b/-	0	
b	-/-	-/-	c/1	b/1	0	A
c	-/-	d/1	c/1	-/-	0	
d	-/-	d/1	e/-	-/-	1	
e	-/-	-/-	e/0	f/0	1	B
f	a/0	-/-	-/-	f/0	1	

y \ x1x2	00	01	11	10	Z2
0	A/0	B/1	A/1	A/1	0
1	A/0	B/1	B/0	B/0	1

y \ x1x2	00	01	11	10	Z2
0	0/0	1/1	0/1	0/1	0
1	0/0	1/1	1/0	1/0	1

y \ x1x2	00	01	11	10
0	0-	1-	0-	0-
1	-1	-0	-0	-0

Z1

y \ x1	1	0
1	1	0
0	1	0

J

y \ x1	1	0
1	-	-
0	1	0

K

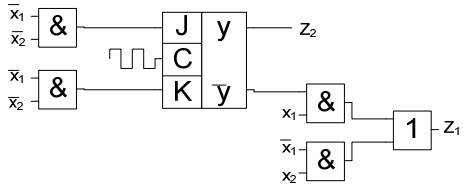
y \ x1	1	0
1	-	-
0	-	-

$$Z_1 = \bar{x}_1 x_2 + \bar{y} x_1$$

$$J = \bar{x}_1 x_2$$

$$K = \bar{x}_1 \bar{x}_2$$

$$Z_2 = y$$

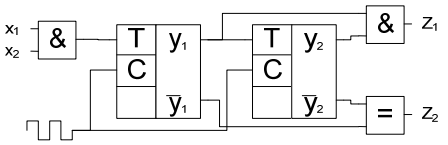


4.2.4. Példa

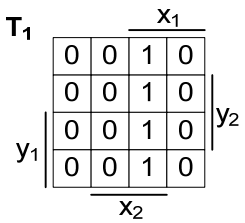
Adott egy T tárolókkal megvalósított szinkron sorrendi hálózat a hálózat működését leíró függvényekkel.
Rajzolja fel a hálózat kapcsolási rajzát.
Milyen modell szerint működik a hálózat?
Adja meg a kapcsolás vezérlési tábláját és kódolt állapottábláját.
Rajzolja fel a kódolt állapotgráfot.

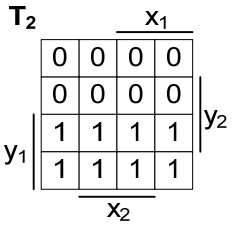
$$Z_1 = y_1 y_2; Z_2 = \overline{y}_1 \cdot \overline{y}_2; T_1 = x_1 x_2; T_2 = y_1$$

Megoldás



A hálózat Moore modell szerint működik.



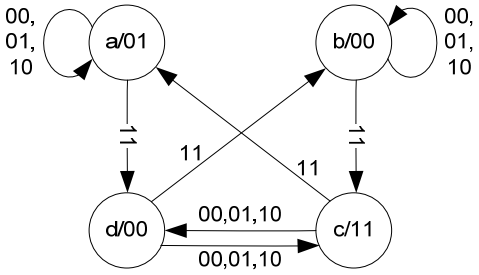


Vezérlési tábla

y_1y_2 \ x_1x_2	00	01	11	10
00	00	00	10	00
01	00	00	10	00
11	01	01	11	01
10	01	01	11	01

Állapottábla

y_1y_2 \ x_1x_2	00	01	11	10
00	00	00	10	00
01	01	01	11	01
11	10	10	00	10
10	11	11	01	11



4.2.5. Példa

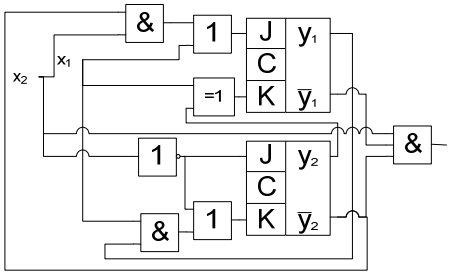
Adott egy JK tárolókkal megvalósított szinkron sorrendi hálózat. A tárolókat vezérlő függvények, illetve a kimenet függvényei a következők.

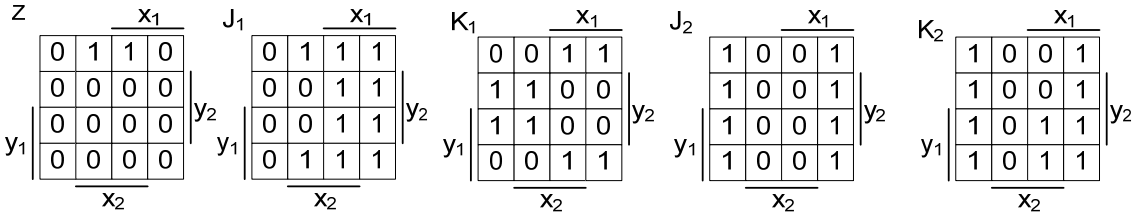
- Milyen modell szerint működik a hálózat?
- Rajzolja fel a hálózat kapcsolási rajzát?
- Adja meg a hálózat kódolt állapottábláját és állapotgráfját!
- Írjon fel egy olyan bemeneti kombináció szekvenciát, amelynek hatására a kimenet 0→1→0→1 szekvenciában változik meg!

$$J_1 = x_1 + x_2\bar{y}_2;\ K_1 = x_1 \oplus y_2;\ J_2 = \bar{x}_2;\qquad K_2 = \bar{x}_2 + x_1y_1;\qquad Z_1 = x_2\bar{y}_1\bar{y}_2$$

Megoldás

A hálózat Mealy modell szerint működik.



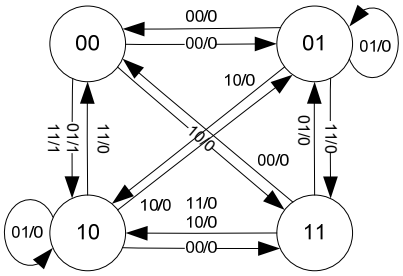


Vezérlési tábla

x_1x_2 / y_1y_2	00	01	11	10
00	00 11	10 00	11 00	11 11
01	01 11	01 00	10 00	10 11
11	01 11	01 00	10 01	10 11
10	00 11	10 00	11 01	11 11
	J_1K_1 J_2K_2	J_1K_1 J_2K_2	J_1K_1 J_2K_2	J_1K_1 J_2K_2

Állapottábla

x_1x_2 / y_1y_2	00	01	11	10
00	01/0	10/1	10/1	11/0
01	00/0	01/0	11/0	10/0
11	00/0	01/0	10/0	10/0
10	11/0	10/0	00/0	01/0



4.2.6. Példa

Adott egy JK tárolókkal megvalósított szinkron sorrendi hálózat. A tárolókat vezérlő függvények, illetve a kimenet függvényei a következők:

$J_1 = xy_2;$ $K_1 = xy_2;$ $J_2 = x;$ $K_2 = x;$ $Z_1 = y_1;$ $Z_2 = y_2$

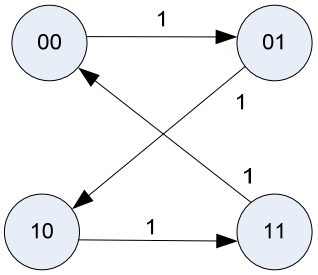
Milyen modell szerint működik a hálózat?
Adja meg a hálózat kódolt állapottábláját és állapotgráfját!
Milyen feladatot valósít meg a kapcsolás x=1 esetén?

Megoldás

A hálózat Moore modell szerint működik.

y_1y_2 \ x	0	1	Z_1Z_2
00	00 00	00 11	00
01	00 00	11 11	01
11	00 00	11 11	11
10	00 00	00 11	10

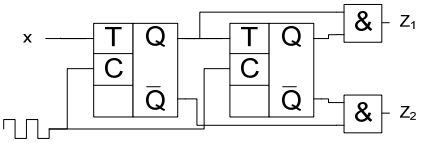
y_1y_2 \ x	0	1	Z_1Z_2
00	00	01	00
01	01	10	01
11	11	00	11
10	10	11	10



A hálózat x=1 esetén négyes számlálóként működik.

4.2.7. Példa

Vizsgálja meg a következő szinkron sorrendi hálózat működését!
Írja fel a kimeneti függvényeket, valamint a tárolókat vezérlő függvényeket!
Adja meg a kapcsolás vezérlési és kódolt állapottábláját!
Milyen modell szerint működik a kapcsolás? Rajzolja fel a rendszer állapotgráfját!
Rajzolja fel a működését ütemdiagrammját az első négy ütemre, x=1 esetére!

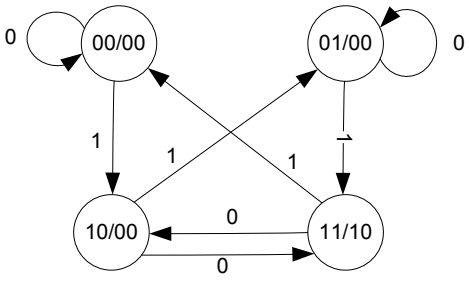


Megoldás

$$Z_1 = y_1 y_2; Z_2 = \overline{y}_1 \overline{y}_2; T_1 = x; T_2 = y_1$$

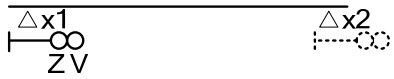
y_1y_2 \ x	0	1	Z_1Z_2
00	00	10	01
01	00	10	00
11	01	11	10
10	01	11	00

y_1y_2 \ x	0	1	Z_1Z_2
00	00	10	01
01	01	11	00
11	10	00	10
10	11	01	00

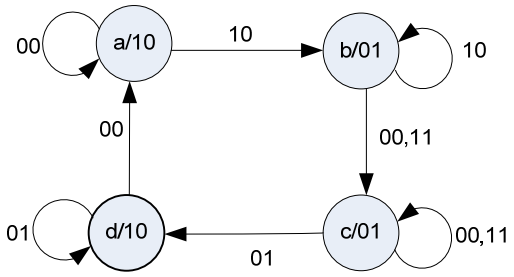


4.2.8. Példa

Tervezzen olyan szinkron sorrendi hálózatot Moore-modell alapján, amely megfelelően vezérli a mellékelt elrendezésben a vasúti jelző Zöld és Vörös fényét. Alaphelyzetben a jelző zöld fénye van bekapcsolva, a vörös pedig kikapcsolva. Ha vonat érkezik az x_1 érzékelőhöz, a jelzőt zöldről át kell kapcsolni és mindaddig vörös állásban kell maradnia, amíg a vonat el nem hagyta az x_2 érzékelőt. Az x_1 és x_2 kerékérzékelők a hálózat bemenetei, akkor adnak 1 jelet, ha vonat van felettük. Figyeljen arra is, hogy a vonat „elférhet” a két érzékelő között, de végig is érhet rajtuk. Valósítsa meg a kapcsolást JK-tárolóelemek segítségével.



Megoldás

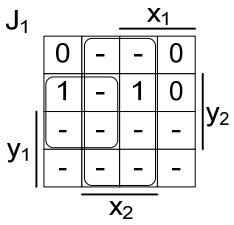


y_1y_2 \ x_1x_2	00	01	11	10	ZV
a	a	-	-	b	10
b	c	-	c	b	01
c	c	d	c	-	01
d	a	d	-	-	01

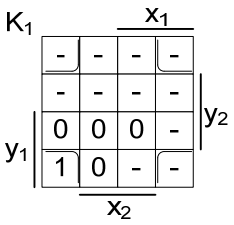
$$Z = \overline{y}_1\overline{y}_2$$

y_1y_2 \ x_1x_2	00	01	11	10	ZV
00	00	--	--	01	10
01	11	--	11	01	01
11	11	10	11	--	01
10	00	10	--	--	01

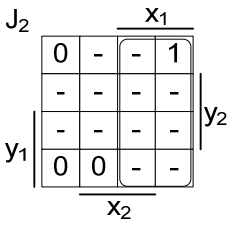
$$V = y_1 + y_2$$



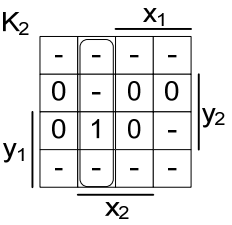
$$J_1 = x_2 + y_2\overline{x}_1$$



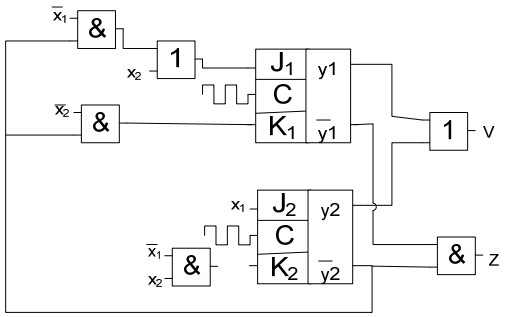
$$K_1 = \overline{x}_2\overline{y}_2$$



$$J_2 = x_1$$

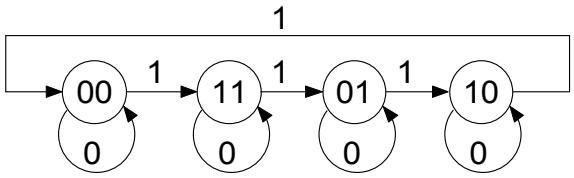


$$K_2 = \overline{x}_1x_2$$



4.2.9. Példa

Tervezzen összevissza számlálót, amely a megadott állapotgráf szerint működik, valósítsa meg JK, és T tárolókkal.

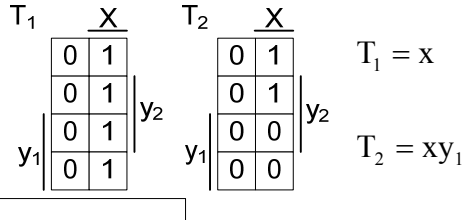


Megoldás

y_1y_2 \ x	**0**	**1**
00	00	11
01	01	10
11	11	01
10	10	00

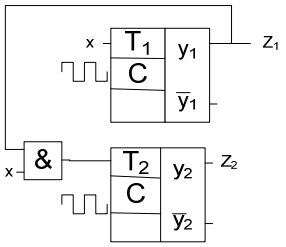
Megoldás T tárolóval

y_1y_2 \ x	**0**	**1**
00	0 0	1 1
01	0 0	1 1
11	0 0	1 0
10	0 0	1 0



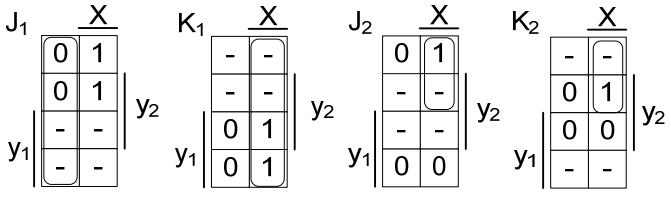
$T_1 = x$

$T_2 = xy_1$

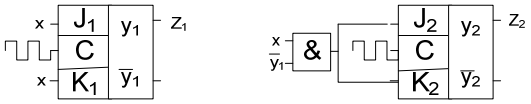


Megoldás JK tárolóval

y_1y_2 \ x	**0**	**1**
00	0- 0-	1- 1-
01	0- -0	1- -1
11	-0 -0	-1 -0
10	-0 0-	-1 0-



$J_1 = x$ $K_1 = x$ $J_2 = x\bar{y}_1$ $K_2 = x\bar{y}_1$



4.2.10. Példa

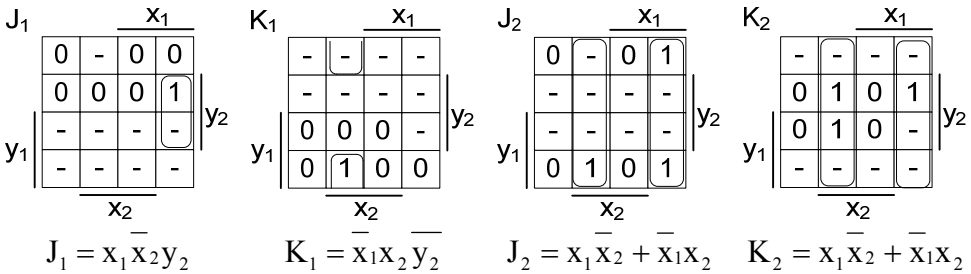
Egy gyártócella két végén két érzékelő vizsgálja az áthaladó munkadarabokat.: A gyártócellában egyszerre maximum 3 db munkadarab engedhető be. Tervezzen Moore modell szerint működő szinkron sorrendi hálózatot JK tárolók felhasználásával, amely Z_1, Z_2 kimeneteken kódolva megadja a cellában lévő munkadarabok számát.

- X_1: beérkezik egy munkadarab;
- X_2 távozik egy munkadarab;

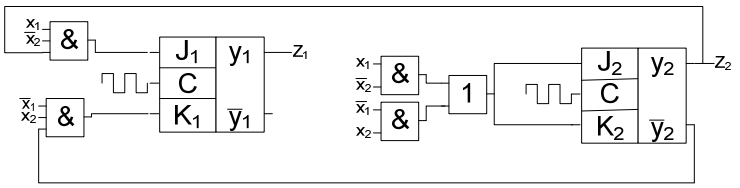
Megoldás

y_1y_2 \ x_1x_2	00	01	11	10
00	00	--	00	01
01	01	00	01	10
11	11	10	11	--
10	10	01	10	11

y_1y_2 \ x_1x_2	00	01	11	10
00	0- 0-	-- --	0- 0-	0- 1-
01	0- -0	0- -1	0- -0	1- -1
11	-0 -0	-0 -1	-0 -0	-- --
10	-0 0-	-1 1-	-0 0-	-0 1-



$$J_1 = x_1\bar{x}_2y_2 \qquad K_1 = \bar{x}_1x_2\bar{y}_2 \qquad J_2 = x_1\bar{x}_2 + \bar{x}_1x_2 \qquad K_2 = x_1\bar{x}_2 + \bar{x}_1x_2$$

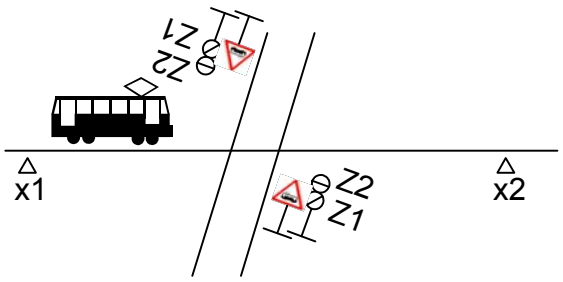


4.2.11. Példa

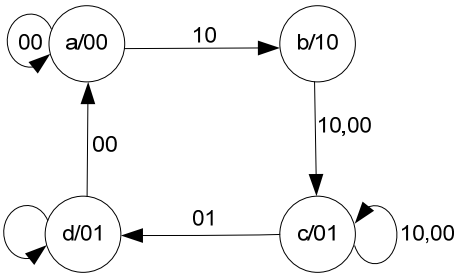
Tervezzen villamos fedezőjelzőt szinkron sorrendi hálózatként, Moore modellel! Bemenetek: x_1, x_2 járműérzékelők: értékük 1, ha jármű van felettük, egyébként 0. Kimenetek: Z_1: a fedezőjelző sárga lámpája (1= bekapcsolva), Z_2: a fedezőjelző piros lámpája (1= bekapcsolva). A fedezőjelző a következőképpen működik:

- Alaphelyzetben nincs villamos, a fedezőjelző sötét.
- A villamos érkezésekor (x1=1) a fedezőjelző sárgára vált.
- A fedezőjelző egy órajel ütem után pirosra vált.
- A villamos eléri az x2 kikapcsolópontot (x2=1) (a jelző továbbra is piros). Amikor a villamos elhagyja a kikapcsoló elemet (x2: 1→0), a fedezőjelző ismét sötét lesz.

Egyidejűleg csak az egyik érzékelő lehet foglalt (azaz a villamos elfér a két érzékelő között). Csak egyirányú közlekedésre kell felkészülni.
Valósítsa meg a kapcsolást JK tároló felhasználásával.



Megoldás

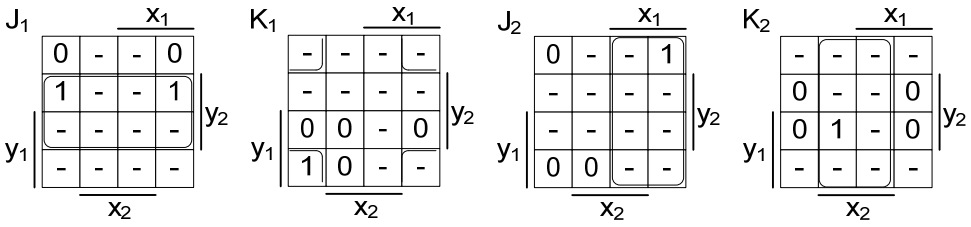


y \ x_1x_2	00	01	11	10
a	a	-	-	b
b	c	-	-	c
c	c	d	-	c
d	a	d	-	-

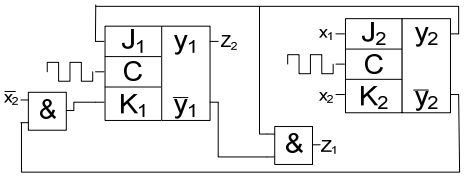
y_1y_2 \ x_1x_2	00	01	11	10	Z1Z2
00	00	--	--	01	00
01	11	--	--	11	10
11	11	10	--	11	01
10	00	10	--	--	01

y_1y_2 \ x_1x_2	00	01	11	10
00	0- 0-	-- --	-- --	0- 1-
01	1- -0	-- --	-- --	1- -0
11	-0 -0	-0 -1	-- --	-0 -0
10	-1 0-	-0 0-	-- --	-- --

$$Z_2 = y_1 \qquad Z_1 = \bar{y}_1 y_2$$



$J_1 = y_2$ $K_1 = \overline{x}_2 \overline{y}_2$ $J_2 = x_1$ $K_2 = x_2$



4.2.12. Példa

Tervezzen olyan jelzőlámpa-vezérlő szinkron sorrendi hálózatot JK tárolók felhasználásával, amely az alábbiak szerint működik: alaphelyzetben az 1-es lámpa zöldet, a 2-es pirosat mutat. Az x bemenet (pl. nyomógomb) hatására először mindkét lámpa pirosra vált, majd az 1-es lámpa pirosat, a 2-es zöldet mutat. Ezután ismét mindkét lámpa pirosra vált, végül visszaáll az alaphelyzet. A fenti szekvencia x hatására, de annak további értékétől függetlenül, csak az órajel hatására megy végbe. A hálózathoz két kimenetet rendeljen a következőképpen:

		piros	**zöld**
1-es lámpa	Z_1	*0*	*1*
2-es lámpa	Z_2	*0*	*1*

Megoldás

y \ x	0	1
a	a/10	b/00
b	c/01	c/01
c	d/00	d/00
d	a/10	a/10

y_1y_2 \ x	0	1
00	00/10	01/00
01	11/01	11/01
11	10/00	10/00
10	00/10	00/10

y_1y_2 \ x	0	1
00	0- 0-	0- 1-
01	1- -0	1- -0
11	-0 -1	-0 -1
10	-1 0-	-1 0-

J_1	X=0	X=1
y_1 y_2=0	0	0
y_1 y_2=1	1	1
$\bar{y}_1$ y_2=0	-	-
$\bar{y}_1$ y_2=1	-	1

K_1	X=0	X=1
y_1 y_2=0	-	0
y_1 y_2=1	-	-
$\bar{y}_1$ y_2=0	0	0
$\bar{y}_1$ y_2=1	1	1

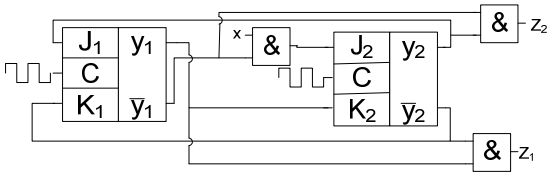
J_2	X=0	X=1
y_1 y_2=0	0	1
y_1 y_2=1	-	-
$\bar{y}_1$ y_2=0	-	-
$\bar{y}_1$ y_2=1	0	0

K_2	X=0	X=1
y_1 y_2=0	-	-
y_1 y_2=1	0	0
$\bar{y}_1$ y_2=0	1	1
$\bar{y}_1$ y_2=1	0	0

Z_1	X=0	X=1
y_1 y_2=0	1	0
y_1 y_2=1	0	0
$\bar{y}_1$ y_2=0	0	0
$\bar{y}_1$ y_2=1	1	1

Z_2	X=0	X=1
y_1 y_2=0	0	0
y_1 y_2=1	1	1
$\bar{y}_1$ y_2=0	0	0
$\bar{y}_1$ y_2=1	0	0

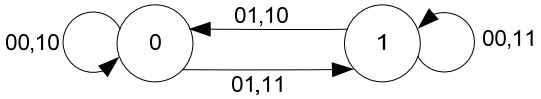
$$J_1 = y_2 \qquad K_1 = \bar{y}_2 \qquad J_2 = x\bar{y}_1 \qquad K_2 = y_1 \qquad Z_1 = y_1\bar{y}_2 \qquad Z_2 = \bar{y}_1 y_2$$



4.2.13. Példa

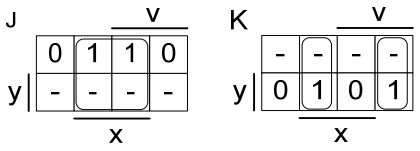
Tervezzen olyan szinkron flip-flopot, amely két bemenettel (V,X) és egy kimenettel (Q) rendelkezik. A flip-flop a V bemenet 1 értéke esetén az X bemenetre nézve D tárolóként, V bemenet 0 értéke esetén pedig T tárolóként működjön! Valósítsa meg a hálózatot JK tárolóval!

Megoldás

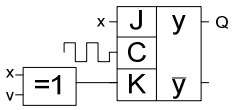


y \ vx	00	01	11	10
0	0	1	1	0
1	1	0	1	0

y \ vx	00	01	11	10
0	0-	1-	1-	0-
1	-0	-1	-0	-1



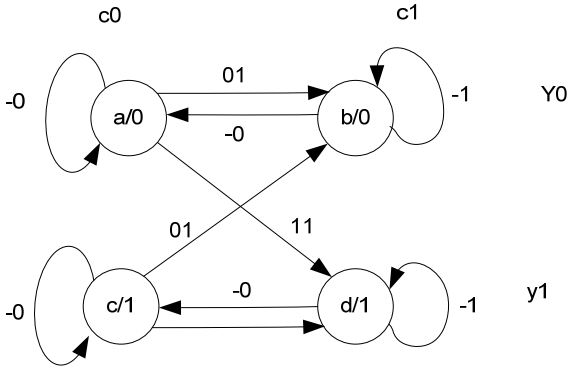
$$J = x \qquad K = x\bar{v} + \bar{x}v$$



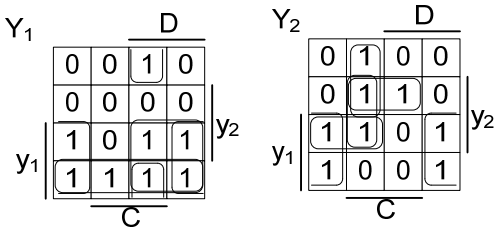
4.2.14. Példa

Tervezzen D tárolót Moore modellel, aszinkron visszacsatolt kombinációs hálózattal! A hálózatnak két bemenete van (D-vezérlés, C-órajel)és egy kimenete (Q). A D tároló mindaddig tartsa a kimenet értékét, ameddig az órajel bemenetén 0 – 1 váltás nem történik. Ekkor a kimenet vegye fel a D vezérlőbemenet értékét.

Megoldás



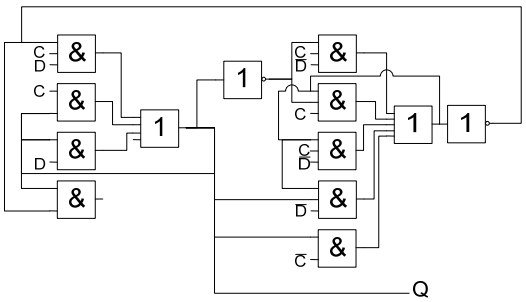
y_1y_2 \ DC	00	01	11	10	Q
00	00	01	10	00	0
01	00	01	01	00	0
11	11	01	10	11	1
10	11	10	10	11	1



$$Q = y_1$$

$$Y_1 = \overline{y}_2CD + y_1\overline{C} + y_1D + y_1\overline{y}_2$$

$$Y_2 = \overline{y}_1C\overline{D} + \overline{y}_1y_2C + y_2C\overline{D} + y_1y_2\overline{D} + y_1\overline{C}$$



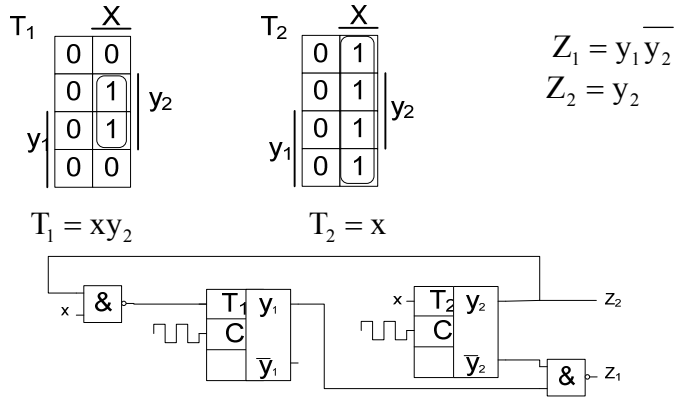
4.2.15. Példa

Tervezzen egy bemenetű (x) és két kimenetű (z1,z2) szinkron négyes számlálót, amely 0 bemenő jel esetén nem számol, 1 bemenő jel esetén a (00-01-10-11-00-..)végtelen szekvenciát valósítja meg. A hálózatot Moore modellel, T tárolók felhasználásával valósítsa meg!

Megoldás

y_1y_2 \ x	0	1	Z_1Z_2
00	00	01	00
01	01	10	01
11	11	00	01
10	10	11	10

y_1y_2 \ x	0	1
00	0 0	0 1
01	0 0	1 1
11	0 0	1 1
10	0 0	0 1



4.2.16. Példa

Készítsen ablaktörlő és –mosó berendezést vezérlő automatikát. A rendszer bemenete a berendezés kapcsolója (x), kimenetei pedig az ablakmosó víz szivattyújának be/kikapcsolása (Z_1), illetve az ablaktörlő motorjának be/kikapcsolása (Z_2). A működés 4 ütemben történik.

- Alaphelyzetben minden kikapcsolt állapotban van.
- A kapcsoló bekapcsolásakor elindul a vízszivattyú, de az ablaktörlő lapátok még nem indulnak el.
- A következő ütemben a szivattyú tovább működik, és bekapcsol az ablaktörlő lapátok motorja.
- A 4. ütemre a szivattyú kikapcsolódik, a motor pedig tovább működik az ütem végéig.

Ezután visszaáll az alaphelyzet. A 3. és 4. ütem alatt az x bemenet tetszőleges lehet.
Készítse el a vezérlés állapotgráfját! Valósítsa meg a vezérlést JK tárolóval, a Z_2 kimenetet Mealy-, a Z_1-et Moore-modell alapján.

Megoldás

y_1y_2 \ x	0	1
a	a/00	b/10
b	c/11	c/11
c	d/01	d/01
d	a/00	a/00

Z_1
0
1
1
0

Z_2

y_1y_2 \ x	0	1
0	0	1
1	1	1
0	0	0
0	0	0

y_1y_2 \ x	0	1
00	0- 0-	0- 1-
01	1- -0	1- -0
11	-0 -1	-0 -1
10	-1 0-	-1 0-

$Z_1 = y_2$ $\quad Z_2 = y_2\overline{y_1} + x\overline{y_1}$

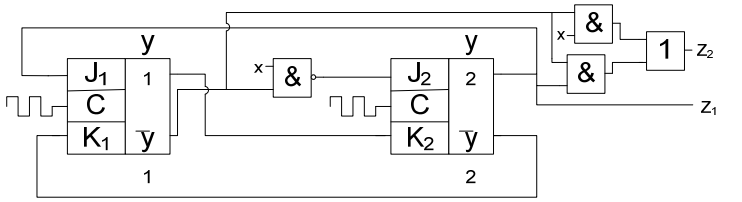
J_1	X	
y_1 \ y_2	0	0
	1	-
	-	-
	-	0

K_1	X	
y_1 \ y_2	-	-
	-	-
	0	0
	1	1

J_2	X	
y_1 \ y_2	0	1
	-	-
	-	-
	0	0

K_2	X	
y_1 \ y_2	-	-
	0	0
	1	1
	-	-

$J_1 = y_2$ $\quad K_1 = \overline{y_2}$ $\quad J_2 = x\overline{y_1}$ $\quad K_2 = y_1$



4.2.17. Példa

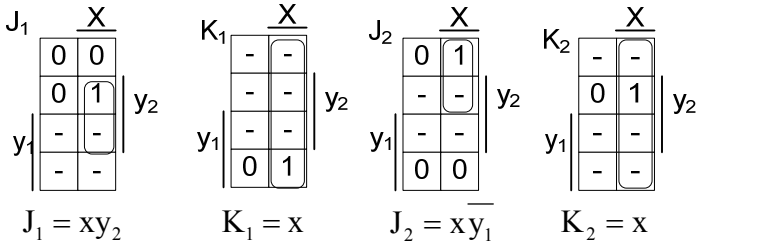
Tervezzen bináris számlálót, mely 3-as ciklusban működik! Valósítsa meg JK-tárolók felhasználásával.

Megoldás

x / y_1y_2	0	1	Z_1Z_2
00	00	01	00
01	01	10	01
10	10	00	10
11	--	--	--

x / y_1y_2	0	1	Z_1Z_2
00	0- 0-	0- 1-	00
01	0- -0	1- -1	01
10	-0 0-	-1 0-	10
11	-- --	-- --	--

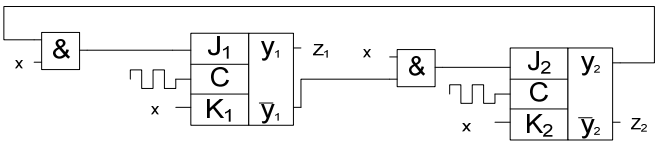
x / y_1y_2	0	1	Z_1Z_2
00	0- 0-	0- 1-	00
01	0- -0	1- -1	01
11	-- --	-- --	--
10	-0 0-	-1 0-	10



$J_1 = xy_2$ $K_1 = x$ $J_2 = x\overline{y_1}$ $K_2 = x$

$Z_1 = y_1$

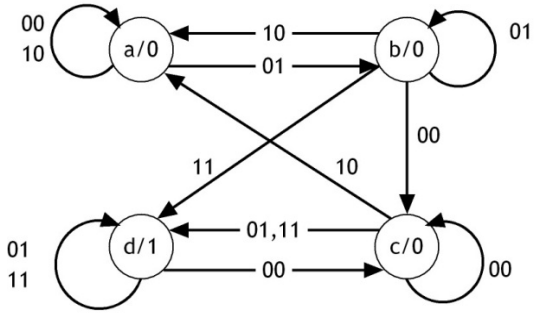
$Z_2 = y_2$



4.3. Aszinkron sorrendi hálózatok

4.3.1. Példa

Valósítsa meg az ábrán látható állapotgráf szerint működő aszinkron sorrendi hálózatot visszacsatolt kombinációs hálózattal! Vizsgálja meg, hogy az adott hálózat tartalmaz-e versenyhelyzetet, és amennyiben igen, mentesítse tőle a hálózatot.



Megoldás

x_1x_2 / y_1y_2	00	01	11	10	Z
a	a	b	-	a	0
b	c	b	d	a	0
c	c	d	d	a	0
d	c	d	d	-	1

x_1x_2 / y_1y_2	00	01	11	10	Z
00	(00)	01	-	(00)	0
01	11	(01)	10	00	0
11	(11)	10	10	00	0
10	11	(10)	(10)	-	1

A nem kritikus versenyhelyzeteket állapotátvezetéssel küszöböljük ki:

x_1x_2 / y_1y_2	00	01	11	10	Z
00	(00)	01	**10**	(00)	0
01	11	(01)	10	00	0
11	(11)	10	10	00	0
10	11	(10)	(10)	**00**	1

$$Z = y_1 y_2$$

Y_1

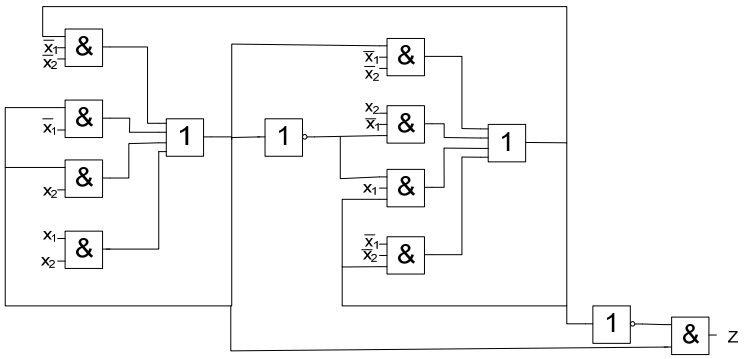
y_1 \ x_1x_2	00	01	11	10
y_2	0	0	1	0
	1	0	1	0
$\bar{y}_2$	1	1	1	0
	1	1	1	0

Y_2

y_1 \ x_1x_2	00	01	11	10
y_2	0	1	0	0
	1	1	0	0
$\bar{y}_2$	1	0	0	0
	1	0	0	0

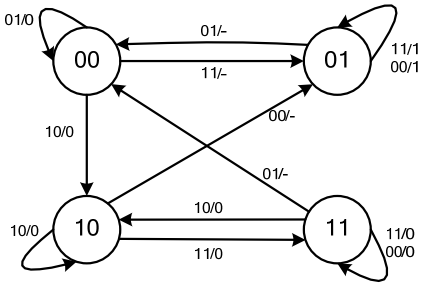
$$Y_1 = y_2\bar{x}_1\bar{x}_2 + y_1\bar{x}_1 + y_1x_2 + x_1x_2$$

$$Y_2 = y_1\bar{x}_1\bar{x}_2 + y_2\bar{x}_1\bar{x}_2 + y_2\bar{x}_1\bar{y}_1 + \bar{x}_1x_2\bar{y}_1$$


&
1
z

4.3.2. Példa

A mellékelt állapotgráf egy aszinkron sorrendi hálózat állapotgráfja. Vizsgálja meg, hogy tartalmaz-e a rendszer kritikus versenyhelyzetet, és ha igen, megfelelő módszerrel szüntesse meg a versenyhelyzetet. Valósítsa meg ezután a kapcsolást is visszacsatolt kombinációs hálózatként.



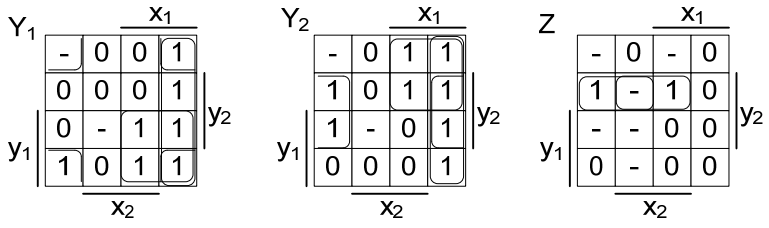
Megoldás

y_1y_2 \ x_1x_2	00	01	11	10
00	--/-	(00)/0	01/-	10/0
01	(01)/1	00/-	(01)/1	--/-
11	(11)/0	00/-	(11)/0	10/0
10	01/-	--/-	11/0	(10)/0

Az állapottábla 00 bemenet esetén kritikus versenyhelyzetet tartalmaz, amelyet az 11-10 állapotátkódolással küszöbölhetünk ki:

y_1y_2 \ x_1x_2	00	01	11	10
00	--/-	(00)/0	01/-	11/0
01	(01)/1	00/-	(01)/1	**11/0**
11	01/-	--/-	10/0	(11)/0
10	(10)/0	00/-	(10)/0	11/0

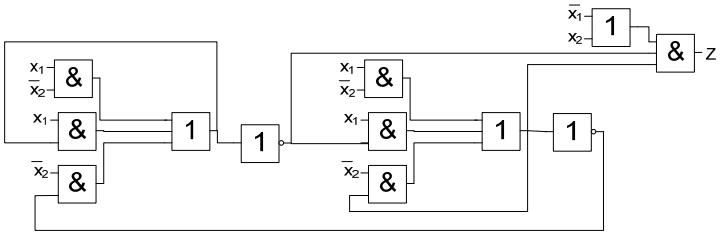
Ekkor az 10 bemeneten megjelenik egy nem kritikus versenyhelyzet, ezt azonban állapotátvezetéssel kiküszöbölhetjük.



$$Y_1 = x_1\bar{x}_2 + y_1x_1 + \bar{y}_2\bar{x}_2$$

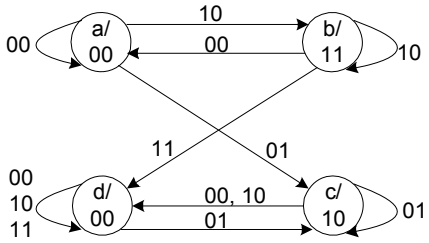
$$Y_2 = x_1\bar{x}_2 + \bar{y}_1x_1 + y_2\bar{x}_2$$

$$Z = \bar{y}_1y_2\overline{x_1} + \bar{y}_1y_2x_2 = \bar{y}_1y_2(\overline{x_1} + x_2)$$


x1
x2
&
1
z

4.3.3. Példa

Tervezzen aszinkron sorrendi hálózatot, amely az alábbi állapotgráf szerint működik! Ellenőrizze a versenyhelyzeteket, és ha szükséges küszöbölje ki azokat! Valósítsa meg a kapcsolást SR tárolók alkalmazásával!



Megoldás

x_1x_2 / y	00	01	11	10	Z_1Z_2
a	(a)	c	-	b	00
b	a	-	d	(b)	11
c	d	(c)	-	d	10
d	(d)	c	(d)	(d)	00

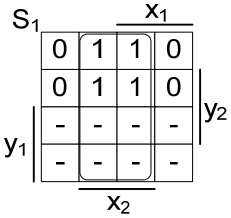
x_1x_2 / y_1y_2	00	01	11	10	Z_1Z_2
00	(00)	11	--	01	00
01	00	--	10	(01)	11
11	10	(11)	--	10	10
10	(10)	11	(10)	(10)	00

Az állapottábla csak nem kritikus versenyhelyzeteket tartalmaz, ezeket állapotátvezetéssel küszöböljük ki.

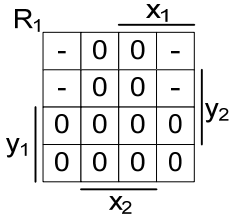
x_1x_2 / y_1y_2	00	01	11	10	Z_1Z_2
00	(00)	11	***10***	01	00
01	00	**11**	10	(01)	11
11	10	(11)	***10***	10	10
10	(10)	11	(10)	(10)	00

x_1x_2 / y_1y_2	00	01	11	10	Z_1Z_2
00	0- 0-	10 10	10 0-	0- 10	00
01	0- 01	10 -0	10 01	0- -0	11
11	-0 01	-0 -0	-0 01	-0 01	10
10	-0 0-	-0 10	-0 0-	-0 0-	00

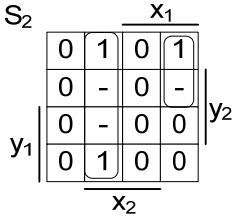
$Z_1 = y_2 \quad Z_2 = \overline{y}_1 y_2$



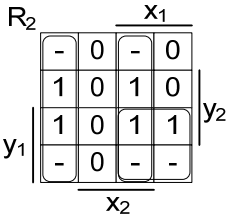
$S_1 = x_2$



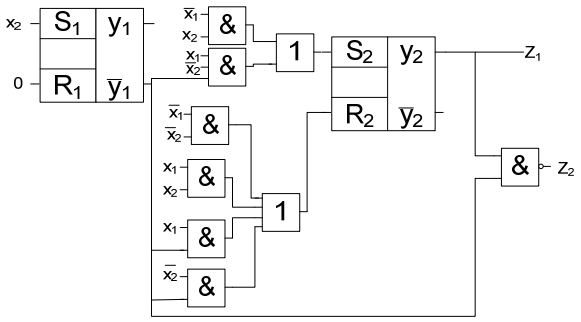
$R_1 = 0$



$S_2 = \overline{x}_1 x_2 + x_1 \overline{y}_1 \overline{x}_2$

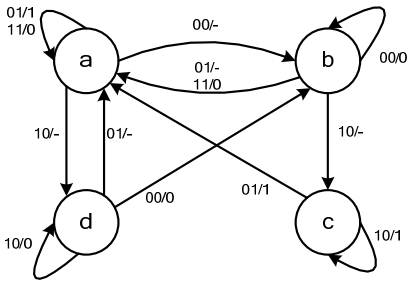


$R_2 = \overline{x}_1 \overline{x}_2 + x_1 x_2 + x_1 y_1 + y_1 \overline{x}_2$


x2
S1
y1
0
R1
ȳ1
x̄1
x2
x1
x̄2
&
1
S2
y2
R2
ȳ2
z1
z2

4.3.4. Példa

Az ábrán egy két bemenetű (x_1, x_2) és egy kimenetű (Z) aszinkron sorrendi hálózat állapotgráfja látható. Írja fel a hálózat előzetes állapottábláját. Vizsgálja meg az összevonási lehetőségeket. Valósítsa meg az aszinkron sorrendi hálózatot versenyhelyzet mentesen, SR tárolók felhasználásával.



Megoldás

y \ x_1x_2	00	01	11	10
a	b/-	(a/1)	(a/0)	d/-
b	(b/0)	a/-	a/0	c/-
c	-/-	a/1	-/-	(c/1)
d	b/0	a/-	-/-	(d/0)

B A

y \ x_1x_2	00	01	11	10
A	B/0	(A/1)	(A/0)	(A/0)
B	(B/0)	A/1	A/0	(B/1)

y \ x_1x_2	00	01	11	10
0	1/0	(0/1)	(0/0)	(0/0)
1	(1/0)	0/1	0/0	(1/1)

y \ x_1x_2	00	01	11	10
0	10	0-	0-	0-
1	-0	01	01	-0

S

y \ x_1	1	0	0	0
y	-	0	0	-

x_2

$S = \bar{x}_1\bar{x}_2$

R

y \ x_1	0	-	-	-
y	0	1	1	0

x_2

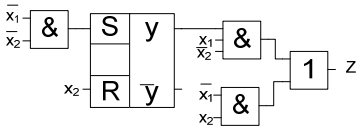
$R = x_2$

Z

y \ x_1	0	1	0	0
y	0	1	0	1

x_2

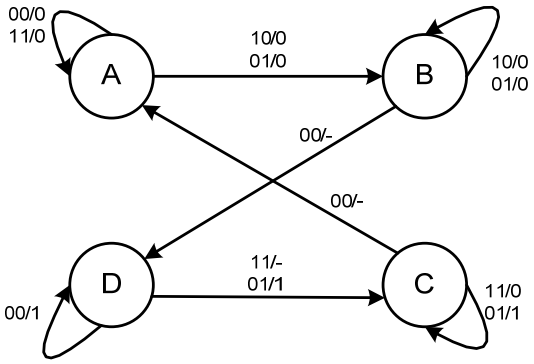
$Z = \bar{x}_1x_2 + yx_1\bar{x}_2$



4.3.5. Példa

Az ábrán egy két bemenetű (x1, x2) és egy kimenetű (Z) aszinkron sorrendi hálózat állapotgráfja látható. Ellenőrizze, hogy az a), illetve a b) szerinti állapotkódolás közül melyik tartalmaz kritikus vagy nem kritikus versenyhelyzetet. Adja meg ezeket az eseteket.
Valósítsa meg a megadott állapotgráf szerint működő visszacsatolt kombinációs hálózatot a helyes állapotkódolással.

	a)	b)
A:	00	00
B:	01	01
C:	10	11
D:	11	10



Megoldás

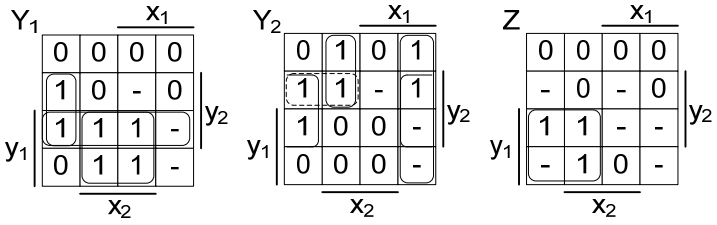
a)

x_1x_2 / y_1y_2	00	01	11	10
00	(00)/0	01/0	(00)/0	01/0
01	10/-	(01)/0	--/-	(01)/0
11	00/-	(11)/1	(11)/0	--/-
10	(10)/1	11/1	11/-	--/-

b)

x_1x_2 / y_1y_2	00	01	11	10
00	(00)/0	01/0	(00)/0	01/0
01	11/-	(01)/0	--/-	(01)/0
11	(11)/1	10/1	10/-	--/-
10	00/-	(10)/1	(10)/0	--/-

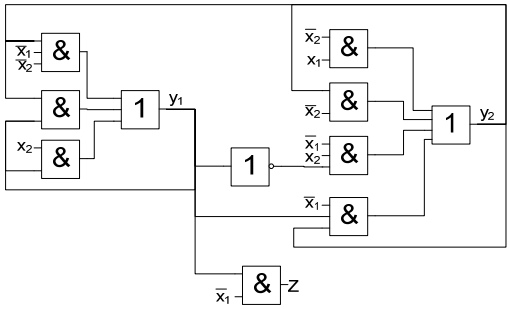
Az a) állapotkódolás 00 bemenet esetén kritikus versenyhelyzetet tartalmaz, ezért a b) kódolás szerint történik a hálózat megvalósítása.



$$Y_1 = y_2\overline{x}_1\overline{x}_2 + y_1y_2 + x_2y_1$$

$$Y_2 = \overline{x}_2x_1 + y_2\overline{x}_2 + x_2\overline{x}_1\overline{y}_1 + y_1y_2\overline{x}_1$$

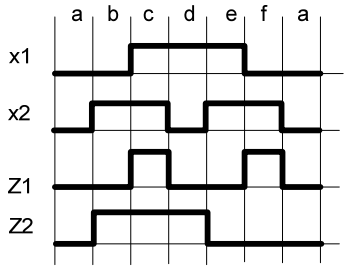
$$Z = y_1\overline{x}_1$$



4.3.6. Példa

Tervezzen aszinkron sorrendi hálózatot, amely a megadott ütemdiagram szerint, Mealy modell alapján működik

- visszacsatolt kombinációs hálózattal, csak NAND kapukkal, valamint
- SR tárolóval

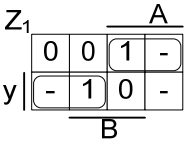


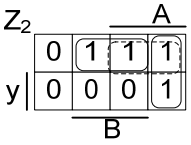
Megoldás

y \ x_1x_2	00	01	11	10	
a	a/00	b/0-	-/--	-/--	
b	-/--	b/01	c/-1	-/--	A
c	-/--	-/--	c/11	d/-1	A
d	-/--	-/--	e/0-	d/01	
e	-/--	f/-0	e/00	-/--	B
f	a/-0	f/10	-/--	-/--	B

y \ x_1x_2	00	01	11	10
A	A/00	A/01	A/11	B/-1
B	A/-0	B/10	B/00	B/01

y \ x_1x_2	00	01	11	10
0	0/00	0/01	0/11	1/-1
1	0/-0	1/10	1/00	1/01



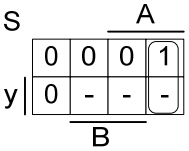


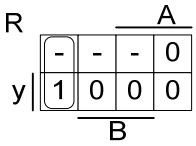
$$Z_1 = y\overline{A} + \overline{y}A \qquad Z_2 = A\overline{B} + \overline{y}A + \overline{y}B$$

$$Z_1 = \overline{y\overline{A}\,\overline{y}A} \qquad Z_2 = \overline{A\overline{B}\,\overline{y}A\,\overline{y}B}$$

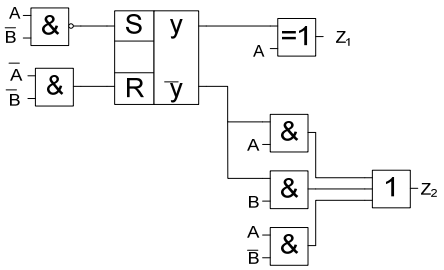
Megoldás SR tárolókkal:

y \ x_1x_2	00	01	11	10
0	0-	0-	0-	10
1	01	-0	-0	-0

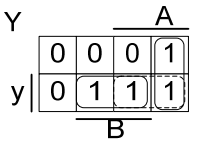




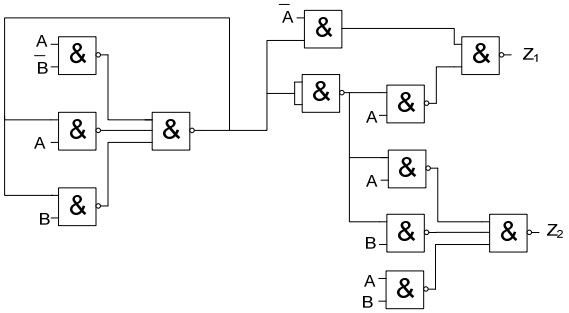
$$S = A\overline{B} \qquad R = \overline{A}\overline{B}$$



Megoldás visszacsatolt kombinációs hálózattal, NAND kapuk felhasználásával tárolókkal:

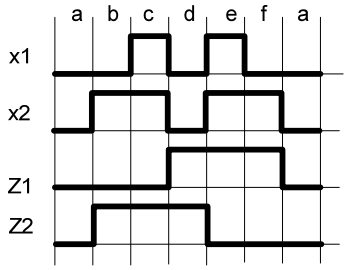


$$Y = A\overline{B} + yA + yB$$



4.3.7. Példa

Tervezzen Mealy-modell szerint működő aszinkron sorrendi hálózatot, amely az alábbi ütemdiagram szerint működik. Valósítsa meg a kapcsolást visszacsatolt kombinációs hálózatként.

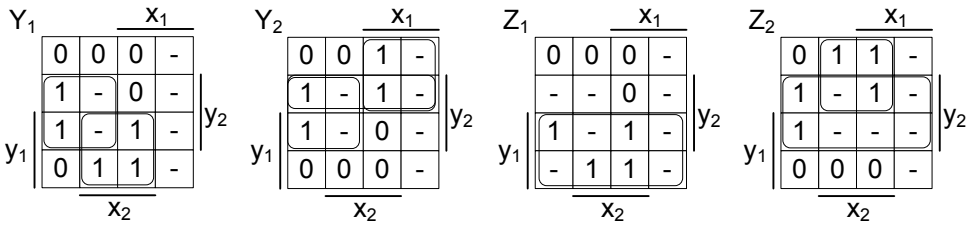


Megoldás

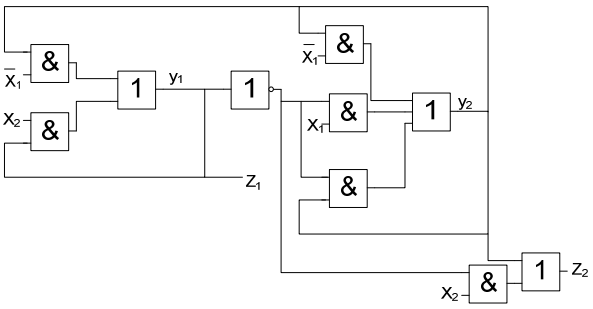
y \ x_1x_2	00	01	11	10	
a	(a)/00	b/0-	-/--	-/--	A
b	-/--	(b)/01	c/01	-/--	
c	d/-1	-/--	(c)/01	-/--	B
d	(d)/11	-/--	e/1-	-/--	C
e	-/--	f/10	(e)/10	-/--	
f	a/-0	(f)/10	-/--	-/--	D

y_1y_2 \ x_1x_2	00	01	11	10
A	(A)/00	(A)/01	B/01	-/--
B	C/-1	-/--	(B)/01	-/--
C	(C)/11	-/--	D/1-	-/--
D	A/-0	(D)/10	(D)/10	-/--

y_1y_2 \ x_1x_2	00	01	11	10
00	(00)/00	(00)/01	01/01	--/--
01	11/11	--/--	(01)/01	--/--
11	(11)/11	--/--	10/1-	--/--
10	00/-0	(10)/10	(10)/10	--/--

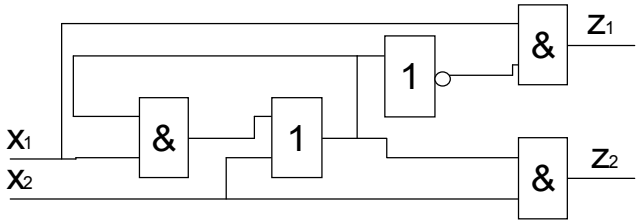


$$Y_1 = y_2\overline{x_1} + x_2y_1 \qquad Y_2 = y_2\overline{x_1} + \overline{y_1}x_1 + \overline{y_1}y_2 \qquad Z_1 = y_1 \qquad Z_2 = y_2 + \overline{y_1}x_2$$



4.3.8. Példa

Tervezze át a mellékelt visszacsatolt kombinációs hálózatot SR tárolóval működő hálózattá! Adja meg a hálózat működését leíró függvényeket, adja meg a vezérlési táblát, és a tárolókat vezérlő függvényeket és rajzolja fel a kapcsolást!



Megoldás

$$Z_1 = \bar{y}x_1; Z_2 = yx_2; Y = x_2 + yx_1$$

Z_1	x_1			
	0	0	1	1
y	0	0	0	0
			x_2	

Z_2	x_1			
	0	0	0	0
y	0	1	1	0
			x_2	

Y	x_1			
	0	1	1	0
y	0	1	1	1
			x_2	

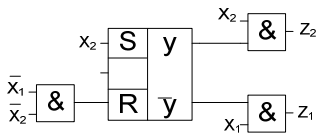
y \ x_1x_2	00	01	11	10
0	0/00	1/00	1/10	0/10
1	0/00	1/01	1/01	1/00

y \ x_1x_2	00	01	11	10
0	0-	10	10	0-
1	01	-0	-0	-0

S	x_1			
	0	1	1	0
y	0	-	-	-
			x_2	

R	x_1			
	-	0	0	-
y	1	0	0	0
			x_2	

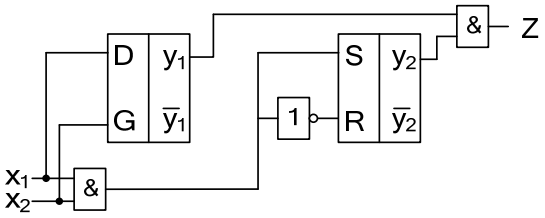
$$S = x_2 \qquad R = \bar{x}_1\bar{x}_2$$



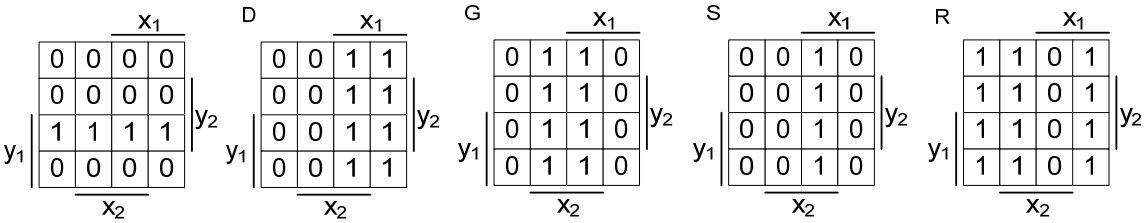
4.3.9. Példa

Tervezze át az ábrán látható kapcsolást a következő lépések szerint:

- Írja fel a kimeneti függvényt és a tárolók vezérlő függvényeit.
- Írja fel a hálózat vezérlési és állapottábláját.
- A kapott állapottábla alapján valósítsa meg a hálózatot visszacsatolt kombinációs hálózatként.
-



Megoldás



Vezérlési tábla

y_1y_2 \ x_1x_2	00	01	11	10	Z
00	00 01	01 01	11 10	10 01	0
01	00 01	01 01	11 10	10 01	0
11	00 01	01 01	11 10	10 01	1
10	00 01	01 01	11 10	10 01	0

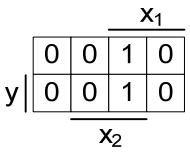
Állapottábla

y_1y_2 \ x_1x_2	00	01	11	10	Z
00	00	00	11	00	0
01	00	00	11	00	0
11	10	00	11	10	1
10	10	00	11	10	0

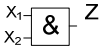
0
1

Összevont állapottábla

y \ x_1x_2	00	01	11	10	Z
0	0	0	1	0	0
1	0	0	1	0	1

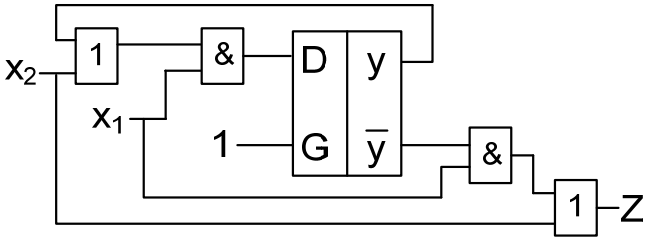


$$Y = x_1x_2; Z = y$$

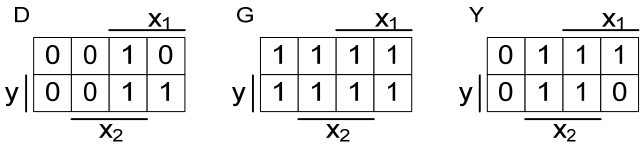


4.3.10. Példa

Alakítsa át az ábrán látható aszinkron sorrendi hálózatot visszacsatolt kombinációs hálózattá.

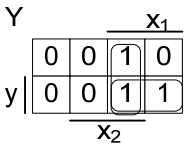


Megoldás

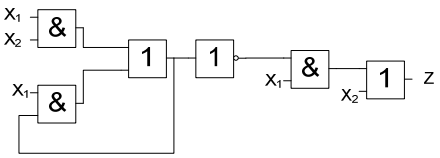


y \ x_1x_2	00	01	11	10
0	01	01	11	01
1	01	01	11	11

y \ x_1x_2	00	01	11	10
0	0/0	0/1	1/1	0/1
1	0/0	0/1	1/1	1/0

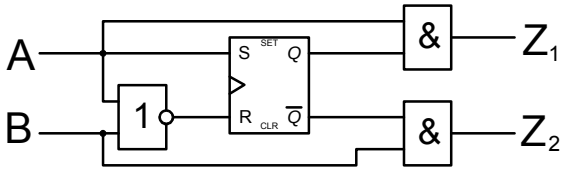


$$Y = x_1x_2 + yx_1$$

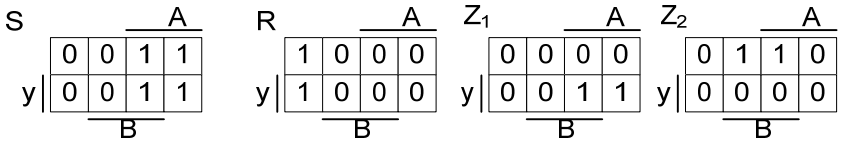


4.3.11. Példa

Az ábrán egy aszinkron sorrendi hálózat látható. Írja fel a hálózat kódolt állapottábláját és állapotgráfját! Alakítsa át a kapcsolást visszacsatolt kombinációs hálózattá!

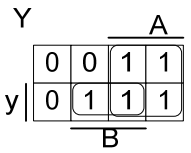


Megoldás

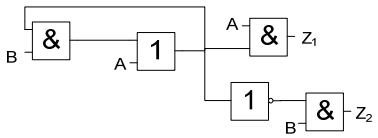


y \ AB	00	01	11	10
0	01	00	10	10
1	01	00	10	10

y \ AB	00	01	11	10
0	0/00	0/01	1/01	1/00
1	0/00	1/00	1/10	1/10



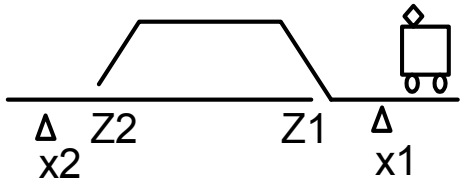
$$Y = A + yB$$



4.3.12. Példa

Az ábrán egy egyszerű vasútállomás látható két váltóval (Z1 és Z2), valamint két vonatérzékelővel (x1 és x2). A váltókat a Z1 és Z2 kimenetekkel lehet vezérelni (Zx=0: egyenes; Zx=1: kitérő). A Z2 váltó alaphelyzetben egyenesben áll, a Z1 váltó alaphelyzete kitérő. Tervezzen olyan aszinkron sorrendi hálózatot, amely a vonat haladásának megfelelően vezérli az állomáson lévő váltókat, a következőképpen: amikor a hálózat érzékeli, hogy a jobbról érkező vonat elhagyja az x1 pontot, állítsa a Z2 váltót kitérőbe; majd amikor a vonat elhagyja az x2 pontot, állítsa vissza az alaphelyzetet! Tervezze meg a hálózatot

- Moore modell szerint visszacsatolt kombinációs hálózattal!
- Mealy modell szerint SR tárolók felhasználásával!



Megoldás

Moore modell, visszacsatolt kombinációs hálózat:

y_1y_2 \ x_1x_2	00	01	11	10	Z_1Z_2
a	a	-	-	b	10
b	c	-	-	b	10
c	c	d	-	-	11
d	a	d	-	-	11

y_1y_2 \ x_1x_2	00	01	11	10	Z_1Z_2
00	00	-	-	01	10
01	11	-	-	01	10
11	11	10	-	-	11
10	00	10	-	-	11

$$Z_1 = 1 \qquad Z_2 = y_1$$

Y_1

y_1 \ x_1	0	1	x_2
0	-	-	0
1	-	-	0
1	1	-	-
0	1	-	-

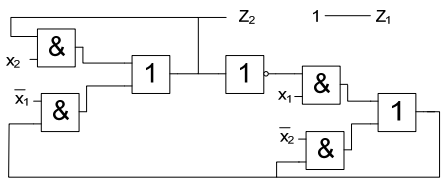
y_2

Y_2

y_1 \ x_1	0	1	x_2
0	-	-	1
1	-	-	1
1	0	-	-
0	0	-	-

y_2

$$Y_1 = x_2y_1 + \overline{x_1}y_2 \qquad Y_2 = x_1\overline{y_1} + \overline{x_2}y_2$$



Mealy modell, SR tárolókkal:

y_1y_2 \ x_1x_2	00	01	11	10
a	a/10	-/--	-/--	b/10
b	c/1-	-/--	-/--	b/10
c	c/11	d/11	-/--	-/--
d	a/1-	d/11	-/--	-/--

y \ x_1x_2	00	01	11	10
a	a/10	a/11	-/--	a/10
b	b/11	a/11	-/--	b/10

y \ x_1x_2	00	01	11	10
0	0/10	0/11	-/--	1/10
1	1/11	0/11	-/--	1/10

y \ x_1x_2	00	01	11	10
0	0-	0-	--	10
1	-0	01	--	-0

Z_1

y \ x_1	1	1
1	1	1
1	1	1

(x_2)

Z_2

y \ x_1	0	-
1	1	0
1	1	0

(x_2)

$S = x_1 \qquad R = x_2$

$Z_1 = 1$

$Z_2 = yx_1 + x_2$