Zalotay Péter

Digitális technika I

Elektronikus jegyzet

Kandó Kálmán Villamosmérnöki Kar

Tartalomjegyzék

Bevezetés

Az elektronikus jegyzet a ***BMF Kandó Kálmán Villamosmérnöki Kar*** érvényes tantervében szereplő ***Digitális technika I***, tantárgy oktatási anyagát tartalmazza. A jegyzet négy fő részben:

- ***A logikai alapismeretek,***
- ***Aritmetikai alapfogalmak,***
- ***Digitális integrált áramkörök, és***

- ***A digitális hálózatok***

fejezetekben tárgyalja a kötelező tananyagot. A tananyag elsajátítását segítik a tantermi foglalkozások során megoldott példák, és otthoni feladatok. A gyakorlati készség fejlesztését szolgálják laboratóriumi gyakorlatok. Mindezekhez bőséges oktatási segédlet áll a nappali, a levelező, és a távoktatásos hallgatók részére.

A ***digitális technika*** módszereivel az ***információ leképzés***, ***műveletvégzés*** és az eredmények továbbítása kétértékű elemi információk (bitek) sorozatával, digitális szavakkal történik. A különböző műveletvégzések egyszerű logikai döntések sorozatára vezethetők vissza. Ugyancsak logikai műveleteket kell végezni, pl. két - különböző mennyiség értékét hordozó - információ közötti viszony (kisebb, nagyobb, egyenlő) megállapításához.

Mielőtt a digitális technika alapjairól írnánk, röviden ismerkedjünk meg – a teljesség igénye nélkül – az e - technikát megalapozó legjelentősebb személyek munkásságával.

George Boole (1815-1864) angol matematikus foglakozott legelőször a formális logika algebrai szintű leírásával és alkotta meg a róla elnevezett algebrát, melyet 1847-ben a " The Mathematical Analiysis of Logic " című könyvében tett közzé.

C. Shannon mérnök-matematikus 1938 -ban megjelent 'Switching Theory' című könyvében adaptálta először G. Boole algebráját kétállapotú kapcsolóelemeket tartalmazó logikai rendszerek leírására. Az információelmélet megalapítása is nevéhez fűződik, az információ alapegységét is tiszteletére róla nevezték el

Azóta hihetetlen mértékű fejlődés következett be a technika és ezen belül is a logikai rendszerek fejlődésében és alkalmazásában. Ez a fejlődés mind az elmélet, a rendszertechnika mind pedig a technológia területén igen gyors volt és természetesen ma is még az. A technológia fejlődésén természetesen itt elsősorban az áramköri elemek és az ehhez kapcsolódó logikai illetve áramköri rendszerek szerelésének automatizálásra lehet gondolni. Érdekes megfigyelni - véleményem szerint a technika fejlődésében egyedülálló módon - hogy voltak időszakok amikor a technológia fejlődése - konkrétan a nagy bonyolultságú integrált áramkörök, a mikroprocesszorok megjelenése - készületlenül érte az elméletet, szinte lehagyva azt.

A következő felsorolás teljesen önkényes, de mindenképpen olyan tudománytörténeti neveket tartalmaz, akik igen nagymértékben elősegítették a logikai rendszerek elméletének kidolgozását, fejlődését,

Evarist Galois (1812-1832) francia matematikus a modern algebra egyik ágának megalapítója. Az általa létrehozott és róla elnevezett csoportelmélet adja a kódolás elmélet, a kriptográfia elméleti hátterét. Rövid élete alatt hozta létre ezt a nem éppen könnyen elsajátítható elméletet, még egyetemista korában, párbajban meghalt.

Wilkes angol matematikus, aki 1954 es években kifejlesztette a mikro-programozás elméletét, amelyet a technológia akkori szintjén még igen költséges lett volna alkalmazni. Ez az elmélet többek között a számítógépek központi vezérlőegységének tervezéshez adott univerzális megoldást. Első alkalmazásai között az igen népszerű IBM 360 -as számítógép is szerepelt.

1964-65 években ***Mealey*** és ***Moore*** mérnökök a logikai rendszerek tervezésének egy olyan zárt jól alkalmazható elméletét adták meg, mely a kor eszközbázisának megfelelő alkalmazását tette lehetővé.

Az 1971-es évre tehető az integrált áramköri gyártástechnológia olyan mértékű fejlődése, hogy lehetőséggé vált a számítógépek központi egységének megvalósítása egy vagy több tokban, vagyis megjelent a mikroprocesszor. Azóta a fejlődés még inkább felgyorsult és szinte nincs az iparnak, a szórakoztató-iparnak, a kereskedelemnek, a mezőgazdaságnak, a szolgáltatásoknak olyan területe, ahol a nagy integráltságú és olcsó digitális rendszerek ne terjedtek volna el. Kis túlzással azt mondhatnánk, hogy az utolsó egy-két évtized a digitális technika korszaka volt és talán még marad is. Az integrált áramkörök gyártástechnológiájának fejlődését igen jól mutatja az , hogy az 1972-es évek közkedvelt I8080 típusú mikroprocesszora még csak megközelítően 4700 tranzisztort tartalmazott, míg ma a kereskedelemben lehet kapni olyan Pentium alapú mikroprocesszort és egyéb rendszertechnikai elemeket tartalmazó chipet mely 150 millió tranzisztorból, épül fel

Természetesen nem csak mikroprocesszorokat fejlesztettek ki, de más univerzálisan, vagy nagy sorozatban használható áramköri készletek is kialakultak:

- ***memóriák***
- ***programozhat logikai elemek: FPGA, stb.***
- ***berendezés orientált integrált áramkörök***
- ***céláramkörök, pl. Quarz órák***

Az integráltság mértékének növekedésével egyre több funkció került egy tokba (chipbe), amely jelentősen megnövelte a kivezetések számát is. Ezeknek a nyomtatott áramköri lemezre, való beültetésére a hagyományos technológia nem volt alkalmas, ezért kifejlesztették a felületszerelési technológiákat (angolul Surface Mount Technology = SMT) és alkatrészeket (angolul Surface Mountage Devices) SMD.

Az egy chipben leintegrált logikai funkciók olyan bonyolultakká váltak, hogy tesztelésükre már a hagyományos módon nem volt lehetőség, ezért ki kellett fejleszteni új megoldásokat erre a feladatra, és ezek a ma oly közkedvelt szimulációs programok illetve hardware leíró nyelvek (VHDL).

Nagyon kevés műszaki szakterületet lehet találni, amelynek csak megközelítően is akkora irodalma volna, mint a digitális technikának illetve rendszereknek. Ugyanakkor és ez talán ellentmondásnak tűnik, hogy ritka az olyan szakterület is amelyben olyan rövid idő alatt lehet olyan tudásra szert tenni , mellyel már egész komoly logikai rendszerek építhetők fel. Az ellentmondást az oldja fel, hogy ma már nem elegendő, ha egy rendszer működik, ez csak egy alapkövetelmény, de annak számos esetben igen nagy megbízhatósággal, könnyű szervizelhetőséggel, versenyképes áron kell megvalósulnia. És az ilyen "hibatűrő" rendszerek tervezése és szervizelése nagy tudást igényel.

1. LOGIKAI ALAPISMERETEK

Mint ahogyan azt a bevezetőben is említettük, a ***digitális technika*** a ***műszaki***, ***technikai*** folyamatok ***megvalósítására*** alkalmas ***berendezések***, automaták tervezéséhez szükséges ***elmélettel***, ***módszerekkel***, és ***áramkörökkel*** foglalkozik.

A tervezendő készülékek, berendezések be-, és kimeneteinek jelei (logikai változói) csak ***két értéket*** vehetnek fel, és a ***döntések*** a formális ***logikában*** használt ***műveleteken*** alapulnak. A változók teljes halmazt alkotnak, amelyet eseménytérnek is nevezhetünk.

A következőkben először összefoglaljuk röviden a használt halmazelméleti alap-fogalmakat. Majd tárgyaljuk a logikai algebra rendszerét, valamint alkalmazási lehetőségeit, módszereit.

1.1. Halmazelméleti alapfogalmak

Halmazon valamilyen ***közös*** tulajdonsággal rendelkező dolgok ***összességét*** értjük. A halmazhoz tartozó "dolgok összességét" a halmaz ***elemeinek*** nevezik. Az adott tulaj-donságokkal nem rendelkező dolgok összessége alkotja a ***komplemens*** vagy kiegészítő halmazt.

A halmazok lehetnek ***végesek*** vagy ***végtelenek*** a halmazt alkotó elemek számától függően. Két speciális halmazt is definiálnak: üres halmaz melynek egyetlen eleme sincs, és a teljes vagy ***univerzális*** halmazt, amelyet valamely halmaz és ennek ***komplemens*** - e alkot.

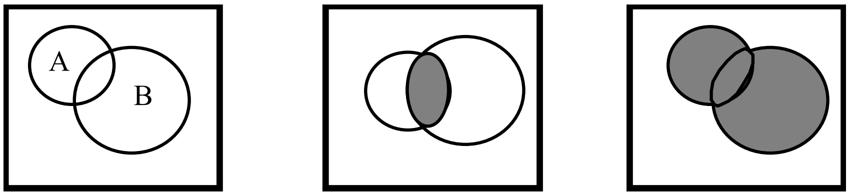
Egy halmaz általában további részekre úgy nevezett ***részhalmazokra*** is oszthatunk, mely úgy jön létre, hogy az adott halmazhoz még további szűkítő feltételt is rendelünk. Például vegyük egyszerűség kedvéért a természetes számok halmazát. A természetes számok részhalmazai lehetnek pl. a prímszámok, a 2-vel vagy a 3-al osztható számok stb.

Azon részhalmazt mely minden eleme része két vagy több halmaznak, azt a két halmaz ***közös részének*** (***metszet***) vagy latin kifejezéssel élve a két halmaz ***konjunkció*** - jának mondjuk.

A természetes számok közül tartalmazza az **A** halmaz a 2-vel, a **B** halmazt pedig a 3-mal osztható számokat. Azok a természetes számok melyek 2-vel és 3-mal is oszthatók a két halmaz ***közös*** részét más szóval ***metszetét*** képezik. Általánosan tehát az A halmaz elemei 2i ahol $i \subset [1, \infty]$, a B halmazé 3j ahol $j \subset [1, \infty]$, és így a közös rész halmazát a 6k ahol $k \subset [1, \infty]$ számok képezik. A közös rész jelölésére a halmazelméletben a **Λ**, vagy $\cap$ jelet használják.**(A Λ B,** vagy **A ∩ B)**

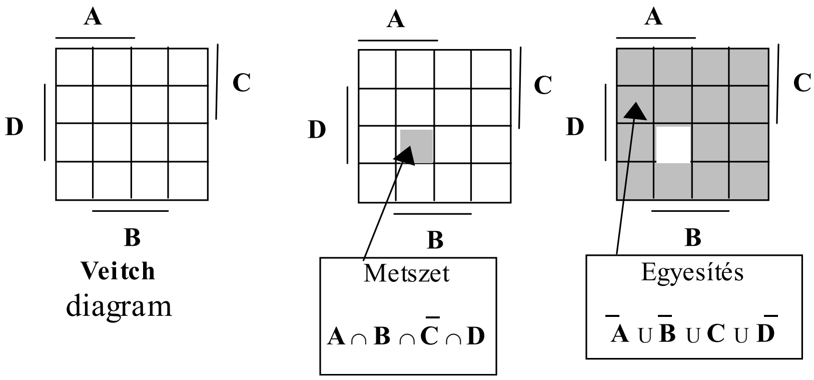
Azon elemekből felépülő halmazt mely tartalmazza mind az **A** mind pedig a **B** (vagy esetleg több halmaz) elemeit a két halmaz ***egyesített*** halmazának vagy ***uniójának*** nevezzük. Latin szóval ez a műveletet a diszjunkció. Előbbi példánknál maradva az egyesített halmaz elmei 6i, 6i-2, 6i-3, 6i-4 (i=1,2,3...). Az unió jelölésére az U, vagy a V jelöléseket használják. (**A U B** vagy **A V B**)

A halmazok és a rajtuk értelmezett műveletek jól szemléltethetők (a J.Venn és Veitch matematikusról elnevezett) ***diagramokkal*** is. A ***teljes halmazt*** egy ***négyszöggel***, míg a ***részhalmazokat*** egy ***zárt alakzattal*** célszerűen egy ***körrel*** – a ***Venn*** diagramban 1.ábra - vagy ugyancsak ***négyszöggel*** jelölik a 2.ábra szerinti ***Veitch*** diagramban.



Venn diagram | részhalmazok metszete | részhalmazok egyesítése

1. ábra



a. b. c.

2. ábra

A ***Veitch*** diagramban minden változó ***IGAZ*** értékéhez a teljes halmaz (esemény-tér) ***fele***, míg a ***másik*** térfél ugyanezen változó ***tagadott*** értékéhez tartozik. (Az algebrai leírásnál a változó fölé-húzásával jelöljük a tagadást). Az ábra négyváltozós halmazt ábrázol. A peremezésnél vonalak jelzik, hogy az egyes változók melyik térfélen IGAZ értékűek. A 2.b. ábrán a metszésnek (ÉS művelet) azt a változatát szemlélteti, amelyik mindegyik változó valamelyik értékének közös területe. Ez metszi ki a legkisebb elemi területet, ezért nevezik ezt ***minterm*** - nek. A 2.c ábrán az összes változó valamely értékeihez tartozó együttes terület. Az egyesített terület a legnagyobb részterület, amelyet ***maxterm*** -nek neveznek. Mind a két kitüntetett területből 2^n –en darab van, ahol n a változók száma.

1.2. A logikai algebra

A ***logikai*** algebra a ***Boole*** algebra alapjaira épül. Kiegészítésekkel a digitális rendszerek tervezésére, elemzésére alkalmas algebrává fejlődött.

A továbbiakban összefoglaljuk a logikai algebra alapjait. A logikai áramkörök később sorra kerülő ismertetésénél, valamint azok működésének megértéséhez az algebrai alapok biztos ismerete elengedhetetlen.

➢ ***Logikai változók, és értékük***

A ***logikai algebra*** csak ***kétértékű logikai változók*** halmazára értelmezett.

A ***logikai*** változók két csoportba oszthatók, úgymint

független-, és

függő változókra.

Mindkét csoport tagjait a latin ABC nagy betűivel (A, B, C . . . X, Y, Z) jelöljük. Általában az ABC első felébe eső betűkkel a független, az utolsó betűk valamelyikével, pedig a függő változókat jelöljük.

A változók két logikai értéke az ***IGAZ***, ill. a ***HAMIS*** érték. Ezeket **1**-el, ill. **0**-val is jelölhetjük (IGAZ: 1; HAMIS: 0).

1.3. A logikai algebra axiómái

Az ***axiómák*** olyan előre rögzített kikötések, ***alapállítások***, amelyek az algebrai rendszerben mindig érvényesek, viszont nem igazolhatók. Ezen állítások meghatározzák a halmaz ***elemeit***, a ***műveleteket***, azok ***tulajdonságait***. A ***tételek***, viszont az axiómák segítségével bizonyíthatók.

1. Az algebra ***kétértékű*** elemek halmazára értelmezett.

2. A halmaz minden elemének létezik a ***komplemens*** -e is, amely ugyancsak eleme a halmaznak, tehát ***teljes*** halmazt alkotnak.

3. Az elemek között végezhető ***műveletek***

- a ***konjunkció*** (logikai ÉS), illetve
- a ***diszjunkció*** (logikai VAGY).

4. A logikai műveletek ***tulajdonságai***:

- ***kommutatív*** –ak (a tényezők felcserélhetők),
- ***asszociatív*** – ak (a tényezők csoportosíthatók),
- ***disztributív*** – ak (a két művelet elvégzésének sorrendje felcserélhető).

5. A halmaz ***kitüntetett*** elemei az

- ***egység*** elem (értéke a halmazon belül mindig IGAZ), és a
- ***null*** elem (értéke a halmazon belül mindig HAMIS).

A ***logikai algebra*** a felsorolt axiómákra épül. A logikai feladatok technikai megvalósításához a halmaz egy elemének komplemenést képező művelet is szükséges. Ezért a műveletek között a logikai ***TAGADÁS*** (más szóhasználattal ***nem***, ***negáció***, ***invertálás***) is szerepel.

1.4. Logikai műveletek

A logikai algebra a következő logikai műveleteket alkalmazza. A változók logikai műveletekkel összekapcsolva alkotnak egy **logikai kifejezést**.

- ***ÉS*** (konjunkció**, AND)** - logikai szorzás;
- ***VAGY*** (diszjunkció**, OR)** - logikai összeadás;
- ***NEM*** (negáció**, invertálás, NOT)** - logikai tagadás.

A felsorolt műveletek közül az ***ÉS***, ill. a ***VAGY*** művelet ***két-***, vagy ***többváltozós***. Ez azt jelenti, hogy a változók legalább ***két eleme***, vagy ***csoportja*** között értelmezett logikai kapcsolatot határoz meg. A ***tagadás egy változós*** művelet, amely a ***változók***, vagy ***változócsoportok*** bármelyikére vonatkozhat.

A továbbiakban ismerkedjünk meg az egyes logikai műveletek definíciójával, és tulajdonságával.

- ***Az ÉS (AND) művelet***

A logikai változókkal végzett ***ÉS*** művelet ***eredménye akkor*** és ***csak akkor IGAZ,*** ha ***mindegyik*** változó értéke egyidejűleg ***IGAZ***. A logikai algebrában az ÉS kapcsolatot szorzással jelöljük (logikai szorzás).

(Megjegyzés: a logikai szorzás jelet - akár csak az Euklideszi algebrában - nem szokás kitenni, így a továbbiakban mi is eltekintünk ettől).

Az

$$\mathbf{AB = K}$$

logikai függvényben az **A** és a **B** a ***független változók***, a **K** pedig a ***függő változó***, vagy eredmény. Jelentése, pedig az, hogy a K akkor IGAZ, ha egyidejűleg az A és a B is IGAZ.

Fontos: a példában szereplő független változók vagy egyedi változók, vagy egy-egy másik logikai függvény megoldásának eredményei.

Vegyünk egy példát:

Ahhoz, hogy egy szobában a lámpa világítson, alapvetően két feltételnek kell teljesülni:

- legyen hálózati feszültség;

- a kapcsoló bekapcsolt állapotban legyen.

Szóban megfogalmazva: **ha** van hálózati feszültség **és** a kapcsoló bekapcsolt, ***akkor*** a lámpa világít. (Az egyéb követelmények teljesülését, hogy az áramkör elemei jók feltételezzük.) Ebben az egyszerű technikai példában a ***hálózati feszültség*** és a ***kapcsoló állapota*** a ***független-***, a lámpa működése, pedig a ***függő változó***. Mindhárom tényező kétértékű.

- ***A VAGY (OR) művelet***

A logikai ***változókkal*** végzett ***VAGY*** művelet eredménye akkor ***IGAZ***, ha a független változók közül ***legalább az egyik*** IGAZ.

Algebrai formában ezt a független változók összegeként írjuk le (logikai összeadás). Az

$$\mathbf{A + B = K}$$

alakú algebrai egyenlőségben a K eredmény akkor IGAZ, ha vagy az A, vagy a B, vagy mindkettő IGAZ.

Erre a logikai kapcsolatra ismert technikai példa egy gépkocsi ***irányjelzőjének*** működését ***ellenőrző lámpa***. A vezető előtt a műszerfalon levő lámpa ***világít***, ha a külső irányjelzők közül ***vagy*** a ***jobb*** oldali, vagy a ***bal*** oldali jelzőlámpacsoport világít. Azt az állítást, hogy jobb oldali jelzés van, jelölje **J** és azt, hogy bal oldali a jelzés, pedig **B**. Az eredményt, hogy a belső ellenőrző lámpa világít, jelöljük **L**-lel. A működést leíró logikai egyenlőség:

$$\mathbf{B + J = L}$$

alakú lesz.

- ***A TAGADÁS (INVERSIO) művelete***

A logikai tagadást ***egyetlen változón***, vagy ***csoporton*** végrehajtott műveletként értelmezzük. Jelentése, pedig az, hogy ***ha*** a ***változó IGAZ, akkor*** a ***tagadottja HAMIS*** és fordítva. Algebrai leírásban a tagadást a változó jele fölé húzott vonallal jelöljük. Ezek szerint a

$$\mathbf{K} = \overline{\mathbf{A}}$$

egyenlőség azt jelenti, hogy a **K** akkor IGAZ, ha az **A** HAMIS. (Szóban A nem - nek, A felülvonásnak vagy A tagadottnak mondjuk.)

Az

$$\overline{\mathbf{A} * \mathbf{B}} = \mathbf{K}$$

összefüggés azt írja le, hogy az eredmény (K) csak akkor igaz, ha az ***A*B*** logikai ***ÉS*** művelet eredménye ***HAMIS*** értéket ad.

A tagadás műveletének előzőek szerinti értelmezése alapján abban a példában, amelyet az ÉS művelet magyarázatára hoztunk az $\overline{\mathbf{A}}$ (A nem) azt jelenti, hogy ***nincs hálózati feszültség***, ill. a $\overline{\mathbf{B}}$ (B nem) jelenti azt, hogy a kapcsoló ***nincs bekapcsolva***. Az eredmény tagadása ($\overline{\mathbf{K}}$) azt fejezi ki, hogy a lámpa ***nem világít***. Az előzőek alapján a gépkocsi irányjelzését ellenőrző lámpa működését leíró összefüggésben is értelmezhetjük a $\overline{\mathbf{J}}$-t (jobb oldali jelzés nincs), a $\overline{\mathbf{B}}$-t (bal oldali jelzés nincs) és az $\overline{\mathbf{L}}$- t (ellenőrző lámpa nem világít) jelölések technikai tartalmát.

1.5. A logikai műveletek tulajdonságai

A következőkben a logikai **ÉS**, valamint logikai **VAGY** műveletek tulajdonságait elemezzük.

- ***Kommutativitás (tényezők felcserélhetősége)***

A leírt szemléltető példákat vegyük ismét elő. Azt állítottuk, hogy ha van hálózati feszültség, és a kapcsoló bekapcsolt, akkor a lámpa világít. Az eredmény változatlan, ha az állítások sorrendjét ***felcseréljük***, vagyis ha a kapcsoló be van kapcsolva és van hálózati feszültség, akkor világít a lámpa. Ez a látszólagos szójáték arra utal, - ami általánosan igaz - hogy az ***ÉS*** műveletekben a ***változók sorrendje felcserélhető***, amely algebrai formában az

$$\mathbf{AB} = \mathbf{BA}$$

azonossággal írható le.

Az előzőekhez hasonlóan meggyőződhetünk arról is, hogy a VAGY műveletekben is felcserélhető -ek az egyes állítások. Érvényes a

$$\mathbf{J} + \mathbf{B} = \mathbf{B} + \mathbf{J}$$

azonosság.

Tehát mindkét többváltozós logikai művelet ***kommutatív***.

- ***Asszociativitás (a tényezők csoportosíthatósága)***

A két logikai művelet további tulajdonsága a műveleti tényezők ***csoportosíthatósága*** is, vagyis az ***asszociativitás***. Algebrai alakban az

$$\mathbf{ABC} = \mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C} = \mathbf{B}(\mathbf{AC})$$

ill. az

$$\mathbf{A} + \mathbf{B} + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{B} + (\mathbf{A} + \mathbf{C})$$

azonosságok írják le az asszociatív tulajdonságot. A zárójel - a matematikai algebrához hasonlóan - a műveletvégzés sorrendjét írja elő. Eszerint a háromváltozós ÉS, ill. VAGY műveletet úgy is elvégezhetjük, hogy előbb csak két változóval képezzük az ÉS, ill. a VAGY kapcsolatot, majd annak eredménye és a harmadik változó között hajtjuk végre az előírt műveletet.

- ***Disztributivitás (a műveletek azonos értékűek)***

A harmadik jelentős tulajdonság, hogy a logikai ÉS, valamint a logikai VAGY ***azonos értékű*** művelet. Mindkettő disztributív a másikra nézve. Algebrai formában ez a következőképpen irható le:

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$$

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

Az első azonosság alakilag megegyezik a matematikai algebra műveletvégzés szabályával. A második azonosság csak a logikai algebrában érvényes. Kifejezi azt, hogy egy logikai szorzat (ÉS kapcsolat) és egy állítás VAGY kapcsolata úgy is képezhető, hogy ***először*** képezzük a ***VAGY műveletet a szorzat*** tényezőivel és az így kapott eredményekkel hajtjuk végre az ***ÉS*** műveletet.

A logikai műveletek megismert tulajdonságai segítségével a logikai kifejezések algebrai átalakítása hajtható végre, és így lehetőség van a legegyszerűbb alakú kifejezés megkeresésére. Ezt a későbbiekben még részletesebben fogjuk tárgyalni.

1.6. A logikai algebra tételei

A továbbiakban felsoroljuk a fontosabb **tételeket**, azok részletes bizonyítása nélkül.

- ***A kitüntetett elemekkel végzett műveletek:***

$$1*1 = 1 \qquad 0*0 = 0$$

$$1*A = A \qquad 0*A = 0$$

$$1+1 = 1 \qquad 0+0 = 0$$

$$1+A = 1 \qquad 0+A = A$$

- ***Az azonos változókkal végzett műveletek:***

$$A*A = A \qquad \mathbf{A} * \overline{\mathbf{A}} = 0$$

$$A+A = A \qquad \mathbf{A} + \overline{\mathbf{A}} = 1$$

Fontos: hogy az A-val jelzett logikai változó nem csak egy változó, hanem egy logikai műveletcsoport eredményét is jelentheti.

- ***A logikai tagadásra vonatkozó tételek:***

$$\overline{\overline{\mathbf{A}}} = \mathbf{A} \qquad \overline{\overline{\overline{\mathbf{A}}}} = \overline{\mathbf{A}}$$

Általánosan: a ***páros*** számú tagadás ***nem*** változtatja meg az értéket, míg a ***páratlan*** számú tagadás azt az ***ellenkezőjére*** változtatja.

- ***Logikai kifejezés tagadása:***

$$\overline{(\mathbf{A}+\mathbf{B})}=\overline{\mathbf{A}}*\overline{\mathbf{B}} \qquad \overline{\mathbf{A}*\mathbf{B}}=\overline{\mathbf{A}}+\overline{\mathbf{B}}$$

Az előző két tétel az un. ***De Morgan - tételek***, amelyek általánosan azt fogalmazzák meg, hogy egy logikai kifejezés tagadása úgy is elvégezhet, hogy az egyes változókat tagadjuk, és a logikai műveleteket felcseréljük (VAGY művelet helyett ÉS, ill. ÉS művelet helyett VAGY műveletet végzünk).

- ***Általános tételek:***

$$\mathbf{A}(\mathbf{A}+\mathbf{B})=\mathbf{A} \qquad \mathbf{A}+\mathbf{AB}=\mathbf{A}$$

E két tétel a műveletek disztributív tulajdonsága és a már felsorolt tételek segítségével a következőképpen bizonyítható:

$$\mathbf{A}(\mathbf{A}+\mathbf{B})=\mathbf{AA}+\mathbf{AB}=\mathbf{A}(\mathbf{1}+\mathbf{B})=\mathbf{A}$$

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

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

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

$$\mathbf{AB}+\overline{\mathbf{A}}\mathbf{B}=\mathbf{B}$$

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

$$\mathbf{AB}+\mathbf{BC}+\overline{\mathbf{A}}\mathbf{C}=\mathbf{AB}+\overline{\mathbf{A}}\mathbf{C}$$

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

A legutóbb felsorolt tételek is bizonyíthatók az alaptulajdonságok segítségével.

1.7. Algebrai kifejezések

A továbbiakban ismertetünk néhány módszert, amelyeket az algebrai kifejezések átala-kításánál gyakran használunk.

- ***Az algebrai kifejezés bővítése.***

Egy logikai szorzat értéke nem változik, ha a kifejezés és az **1**-el logikai szorzatát képezzük (ÉS).

$$\mathbf{AB}=\mathbf{AB}*\mathbf{1}$$

Az 1-et, pedig felírhatjuk, pl. $(\mathbf{C}+\overline{\mathbf{C}})$ alakban. Tehát:

$$\mathbf{AB}=\mathbf{AB}(\mathbf{C}+\overline{\mathbf{C}})=\mathbf{ABC}+\mathbf{AB}\overline{\mathbf{C}}$$

Egy logikai összeadás nem fog megváltozni, ha a kifejezés és a 0 logikai összegét képezzük (VAGY):

$$\mathbf{D}+\mathbf{E}=\mathbf{D}+\mathbf{E}+\mathbf{0}$$

A 0-t kifejezhetjük $\mathbf{F}*\overline{\mathbf{F}}$ alakban. A bővítést végrehajtva az

$$\mathbf{D}+\mathbf{E}=(\mathbf{D}+\mathbf{E})+\mathbf{F}*\overline{\mathbf{F}}=(\mathbf{D}+\mathbf{E}+\mathbf{F})(\mathbf{D}+\mathbf{E}+\overline{\mathbf{F}})$$

azonosságot kapjuk.

Ennél a bővítésnél felhasználtuk a disztributivitást leíró egyik algebrai összefüggést, mely szerint

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

Az előzőben ismertetett bővítési szabály megfordítva egyszerűsítésre is felhasználható.

1.8. Logikai függvények

A ***műszaki, technikai*** feladatok döntő hányada ***logikai döntések*** sorozatára épül. A logikai döntések elemei az állítások, amelyek értékei, és logikai kapcsolatuk határozza meg a döntések eredményét. A feladatokat megvalósító áramkörök, logikai hálózatok bemeneteire kapcsolt – az állításoknak megfelelő - kétértékű jelek a független logikai változók, míg a kimeneteken megjelenő – ugyancsak kétértékű – jelek a következtetések logikai értéke, és ezek a függő logikai változók. A ***függő-***, és a ***független*** változók közötti ***logikai kapcsolatot*** írják le a ***logikai függvények***. Minden függő változóra – kimeneti értékre – felírható egy-egy függvény.

A logikai függvény olyan ***egyenlőség***, amely ***változói kétértékűek***, és ezek között csak ***logikai műveleteket*** – ÉS, VAGY, TAGADÁS – végzünk.

A függvények megadása – leírása – történhet

- ***algebrai alakban,***
- ***táblázat segítségével,***
- ***matematikai jelölésekkel,***
- ***grafikus módon,***
- ***időfüggvény formájában.***

A felsorolt leírási módok teljesen egyenértékűek, és egymásba átírhatók!

A logikai kifejezések, függvények algebrai leírásának szabályait az 1.3. alfejezetben ismertettük. Az alábbiakban a további megadási formákat, és ezek kapcsolatát tárgyaljuk.

- ***Logikai feladatok leírása táblázattal***

A logikai formában megfogalmazható, műszaki, számítási és irányítási feladatokban mindig *véges* számú ***elemi*** állítás szerepel. Ezek mindig ***csak két*** értéket vehetnek fel, vagy ***IGAZ*** - ak, vagy ***HAMIS*** - ak. Ebből következik, hogy a független változók lehetséges ***érték-variációinak*** a száma is *véges*. Minden egyes variációhoz a függő változó meghatározott értéke tartozik.

A logikai kapcsolat leírásának táblázatos formája az ***igazságtáblázat***. A táblázat tartalmazza a független változók összes kombináció-ját (érték-variációját) és az azokhoz rendelt függőváltozó(k) értékét, amit ***függvényértéknek*** is nevezhetünk. Az igazság-táblázatban minden logikai változó IGAZ értékét 1-el, míg a HAMIS értéket 0-val jelöljük.

Összefoglalva: ***az igazságtáblázat oszlopainak száma az összes logikai változó számával (függő változók száma + független változók száma), sorainak száma pedig a független változók lehetséges kombinációinak számával egyezik meg.***

A lehetséges értékvariációk számát (V-t) általánosan a $V=2^n$ összefüggéssel határozhatjuk meg, ahol n az összes független logikai változó száma.

Megjegyezzük, hogy általában csak egy függő változót, tartalmazó igazságtáblázatot írunk fel. Azokban az esetekben, ha egy logikai kapcsolat-rendszerben több függő változó van, célszerűbb mindegyikre külön-külön felírni az igazságtáblázatot. Ezzel áttekinthetőbb képet kapunk.

A logikai alapműveletek igazságtáblázatait mutatja a 3. ábra.

$$\mathbf{K = A\,B}$$

B	A	K
0	0	0
0	1	0
1	0	0
1	1	1

$$\mathbf{K = A + B}$$

B	A	K
0	0	0
0	1	1
1	0	1
1	1	1

$$\mathbf{K = \overline{A}}$$

A	K
0	1
1	0

3. ábra

Írjuk fel a

$$\mathbf{Z = A\,\overline{B} + \overline{A}\,B}$$

logikai függvény igazságtáblázatát!

Első lépésként az igazságtáblázat oszlopainak és sorainak a számát határozzuk meg. Mivel két független-, (A,B) és egy függő változó (Z) van, az oszlopok száma 3. (4.a.ábra). A sorok száma a független változók számából (n=2) a $V = 2^n = 2^2 = 4$ összefüggésből számolható.

Második lépésként az értékvariációkat írjuk be (4.b.ábra).Célszerű ezt úgy végrehajtani, hogy az egyik oszlopban (pl. az A) soronként váltjuk a 0, és az 1 beírását. A következő oszlopban (B) párosával váltogatjuk az értékeket. (Nagyobb sorszámnál, a következő oszlopoknál négyesével, majd nyolcasával variálunk sit.).A beírásnak ez a rendszeressége biztosítja, hogy egyetlen variáció sem marad ki.

Harmadik lépés az egyes sorokba írandó Z érték meghatározása. Ezt úgy végezhetjük el, hogy a független változóknak értékeket adunk, s az adott függvényt kiszámítjuk.

B	A	Z

a.

B	A	Z
0	0	
0	1	
1	0	
1	1	

b.

B	A	Z
0	0	0
0	1	1
1	0	1
1	1	0

c.

4. ábra

1. sorban: $A = 0, B = 0$

Z = 0*1 + 1*0 = 0

2. sorban: ***A = 1, B = 0***

Z = 1*1 + 0*0 = 1

3. sorban: ***A = 0, B = 1***

Z = 0*0 + 1*1 = 1

4.sorban: ***A = 1, B = 1***

Z = 1*0 + 0*1 = 0

A példa szerinti logikai függvény igazságtáblázata a 4.c.ábrán látható.

Az előző példa egy sokszor használt függvény-kapcsolat, az un. ***KIZÁRÓ-VAGY*** (XOR) művelet. (Nevezik moduló összegnek is.) A művelet eredménye akkor 1, ha a két változó közül az egyik 1. Több változóval is végezhető ***moduló - összegzés***, és eredménye akkor 1, ha ***páratlan számú*** független változó értéke 1.

- ***Logikai függvény felírása az igazságtáblázatból***

Az előző pontban megismerkedtünk az igazság-táblázattal, amely a logikai kapcsolatrendszer leírásának egyik formája. Példa segítségével mutattuk be, hogy ismert logikai függvényből hogyan írható fel a táblázatos alak.

Ebben a részben azt tárgyaljuk, hogy ha ismert az igazságtáblázat, hogyan lehet abból felírni a logikai függvényt.

Az igazságtáblázat egy sora a független változók adott kombinációját, és az ehhez tartozó függvény értékét adja.

Az egy sorban levő értékeket az ÉS művelettel lehet összekapcsolni. A különböző sorok, pedig különböző esetnek megfelelő variációkat írnak le. Tehát egy adott időpillanatban vagy az egyik sor vagy egy másik sor variációja érvényes. A sorok logikai kapcsolata VAGY művelettel írható le.

Vegyük példaként az 5.ábrán látható igazságtáblázatot.

C	**B**	**A**	**K**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

5. ábra

A táblázatból kétféle alakú függvény írható fel a következő állítás alapján:

A függvényérték **IGAZ**

- ***azokban*** a sorokban, amelyekben a függő változó ***1***, illetve
- ***nem*** azokban a sorokban, ahol függő változó ***0***.

Az állítás első fele szerint felírjukni az 1 értékhez tartozó sorok változókombinációinak VAGY kapcsolatát.

A második rész szerint felírjuk a 0 értékű sorokhoz tartozó változókombinációik VAGY kapcsolatát, majd az egyenlőség mindkét oldalát tagadjuk.

Az igazságtáblázatból írjuk fel először a független változók 1 értékeihez tartozó függvény algebrai alakját.

Az igazságtáblázat tartalmát a következőképpen olvassuk ki. A K jelű függő változó értéke 1 (IGAZ),

ha C = 0 és B = 0 és A = 1 (2.sor),vagy

ha C = 0 és B = l és A = 0 (3.sor),vagy

ha C = 0 és B = 1 és A = 1 (4.sor), vagy

ha C = l és B = l és A = 0 (7.sor).

Az A,B,C és K változók közötti logikai kapcsolat az előbbiek szerint

$$\mathbf{K = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + AB\overline{C} + \overline{A}BC}$$

alakban írható fel.

A függvény rendezett ***ÉS-VAGY*** alakú. Az ÉS művelettel összekapcsolt részekben mindegyik változó szerepel egyenes (ponált) vagy tagadott (negált) alakban, vagyis a Veitch diagramnál definiált ***minterm***.

Az egyes minterm-ek között, pedig VAGY műveleteket kell végezni. Az ilyen függvényalakot idegen szóval ***diszjunktív kanonikus*** alaknak (teljes diszjunktív normál formának) nevezzük.

A felírás szabálya a következő:

1. azokat a sorokat kell figyelembe venni, amelyeknél a függő változó értéke 1;

2. az egy sorban levő független változók között ÉS műveletet kell végezni, ahol a független változó igaz (egyenes, más kifejezéssel ponált) alakban írandó, ha értéke 1 és tagadott (negált) alakban, ha értéke 0;

3. az egyes sorokat leíró ÉS műveletű rész-függvények VAGY művelettel kapcsolód-nak egymáshoz.

A kiinduló állítás második része szerint:

Azt nézzük meg, hogy mikor nem IGAZ (HAMIS) a következtetés.

A K értéke a következő kombinációknál (sorokban) **0**, (vagyis $\mathbf{\overline{K}}$)

ha C=0 és B=0 és A=0 (1.sor) vagy

ha C=1 és B=0 és A=0 (5.sor) vagy

ha C=1 és B=0 és A=1 (6.sor) vagy

ha C=1 és B=1 és A=1 (8.sor).

A logikai kapcsolatot a

$$\mathbf{\overline{K} = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}C + ABC}$$

függvénnyel írhatjuk le. Ebből a K értékét mindkét oldal tagadásával nyerjük.

$$\mathbf{\overline{\overline{K}} = \overline{\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}C + ABC}}$$

A baloldalon K-t kapunk. A jobb oldal átalakítását a de Morgan - tételek alkalmazásával végezhetjük el.

$$\mathbf{K = (\overline{\overline{A}\,\overline{B}\,\overline{C}})\,(\overline{\overline{A}\,\overline{B}C})\,(\overline{A\overline{B}C})\,(\overline{ABC}) =}$$

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

A kapott függvényről megállapíthatjuk, hogy ***VAGY-ÉS*** alakú. A zárójeles VAGY műveletek mindhárom független változót (A,B,C) tartalmazzák ***egyenes*** vagy ***tagadott*** alakban. Ezek ***maxterm*** -ek, melyeket a Veitch diagramnál definiáltunk. Az első maxterm az igazságtáblázat első sora szerinti állítás - vagyis, hogy az A=0 és B=0 és C=0 - tagadása. A további tagokat vizsgálva látjuk, hogy ezek is egy-egy olyan sornak a tagadásai, melyben K=0.

Az előzőek alapján most már megfogalmazhatjuk, hogy az igazságtáblázatból úgy is felírhatjuk a feladatot leíró logikai függvényt, hogy

1. azokat a sorokat vesszük figyelembe, melyekben a függő változó értéke 0;

2. az egy sorban levő független változók között VAGY kapcsolatot írunk elő;

3. a független változót egyenes alakban írjuk, ha értéke 0 és tagadott alakban, ha értéke 1;

4. az egyes sorokat leíró VAGY függvényeket ÉS művelettel kell összekapcsolni.

Azt a logikai függvényt, amely maxtermek logikai szorzata idegen szóval ***konjunktív kanonikus alakúnak***, rendezett VAGY-ÉS függvénynek (teljes konjunktív normál alakúnak) nevezzük.

- ***Logikai függvények matematikai, egyszerűsített felírási alakjai***

Mivel a logikai változónak két értéke – 0, illetve 1 – lehet, ezért ezt tekinthetjük egy ***bináris számjegy*** -nek is.

A függvény egy - egy ***maxterm*** – je, vagy ***minterm*** - je, oly módon is leírható, hogy az hányadik eleme a mintermek, illetve maxtermek ***rendezett sorának***.

A ***sorszám*** kiszámolásához első lépésként a változókhoz a bináris számrendszer egy-egy ***helyértékét*** kell hozzárendelnünk, vagyis ***súlyozunk***.

A súlyozás kiválasztása után az egyes kombinációkban a ponált változó helyére 1-t, míg a negált helyére 0-t írunk. Az így kapott szám lesz az adott maxterm, vagy minterm ***sorszám***-a (súlya). (A számolást bináris számrendszerben végzzük, de az indexet decimálisan fogjuk írni, mivel ez kevesebb helyet igényel.)

Súlyozzuk a következőképen egy háromváltozós függvény változóit:

$$\mathbf{C \div 2^2, B \div 2^1, A \div 2^0}$$

Ekkor a

$\mathbf{C\overline{B}A}$ minterm súlya: $\mathbf{1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 101_B = 5}$

a $\mathbf{\overline{C} + B + \overline{A}}$ maxterm súlya: $\mathbf{0 * 2^2 + 1 * 2^1 + 0 * 2^0 = 010_B = 2}$.

A mintermeket az $\mathbf{m_i^v}$ jelöléssel helyettesíthetjük, ahol az m jelzi, hogy a logikai egység minterm, a felső index v a változók számát, az alsó index i pedig a sorszámot jelenti.

Hasonlóan a maxterm -eket is helyettesíthetjük a $\mathbf{M_i^v}$ jelöléssel. Az indexek (v,i) jelentése ugyan az, míg az M jelzi, hogy a logikai kifejezés maxterm.

A leírtakat a példában szereplő kifejezésekre (ugyanazon változó súlyozásnál) a

$$\mathbf{C\overline{B}A \div m_5^3}$$

és a

$$\mathbf{\overline{C} + B + \overline{A} \div M_2^3}$$

helyettesítéseket alkalmazhatjuk.

- ***Függvények megadása matematikai alakban***

  Az ismertetett helyettesítésekkel a diszjunktív, valamint konjunktív kanonikus alakú függvények is rövidebben leírhatóak. Vegyük példának az előzőekben felírt függvények alaki helyettesítését az $\mathbf{A \div 2^2, B \div 2^1, C \div 2^0}$ változó súlyozás alkalmazásával:

$$\mathbf{K = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + AB\overline{C} + \overline{A}BC}$$

$$\mathbf{K = m_4^3 + m_2^3 + m_6^3 + m_3^3}$$

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

$$\mathbf{K = M_7^3 * M_6^3 * M_2^3 * M_0^3}$$

A függvények felírása tovább is egyszerűsíthető oly módon, hogy

- ***megadjuk a függvény – alak -ot***
- ***a változók számát, és***
- ***a függvényben szereplő term –ek sorszámait.***

A ***diszjunktív*** alaknál a logikai összegzést $\mathbf{\Sigma}$ –val jelöljük, és fölé írjuk a változók számát.

$$K = \sum^{n}(.......)$$

A ***konjunktív*** alaknál a logikai szorzást $\mathbf{\Pi}$–vell jelöljük, és fölé írjuk a változók számát:

$$K = \prod^{n}(........)$$

Mindkét alaknál a függvényben szereplő mintermek, vagy maxtermek sorszámát – a szimbólumot követő -zárójelben soroljuk fel.

A két mintafüggvény egyszerűsített felírása (ugyanazon változó-súlyozást alkalmazva):

$$\mathbf{K = \sum^{3}(2,3,4,6)}$$

$$\mathbf{K = \prod^{3}(7,6,2,0)}$$

- ***Kanonikus függvény-alakok közötti átalakítás***

  Az előzőekben megismertük, hogyan lehet a logikai feladat igazságtáblázatából felírni a logikai függvény két kanonikus alakját.

  Az egyik kanonikus alakú függvény egyszerűsített (indexelt) formája alapján nagyon egyszerűen felírható a másik rendezett alak egyszerűsített formája.

  Az átalakítás menete a következő: az ismert függvény alapján felírjuk az ***inverz függvényt*** (amely az alap függvény tagadottja), amely a hiányzó indexű term – ekből áll.

pl. ha ismert a diszjunktív alak:

$$K = \sum^{3}(2,3,4,6) \qquad \Rightarrow \qquad \overline{K} = \sum^{3}(0,1,5,7)$$

ismert a konjunktív alak:

$$K = \prod^{3}(7,6,2,0) \qquad \Rightarrow \qquad \overline{K} = \prod^{3}(5,4,3,1)$$

az inverz függvény tagadásával nyerjük a másik alakú rendezett függvényt. A tagadáskor a függvény - típusjele az ellenkezője lesz, és mindegyik index (i) B-1 –es kiegészítőjét ($\bar{i}$) kell vennünk a következő összefüggés alapján:

$$\bar{i} = (2^{v} - 1) - i$$

A tagadások elvégzése után

$$\overline{\overline{K}} = \overline{\sum^{3}(0,1,5,7)} \qquad \Rightarrow \qquad K = \prod^{3}(7,6,2,0)$$

$$\overline{\overline{K}} = \overline{\prod^{3}(5,4,3,1)} \Rightarrow \qquad K = \sum^{3}(2,3,4,6)$$

megkaptuk a keresett alakú függvényeket.

- ***A logikai függvények grafikus megadása***

  A logikai függvények gyakori ábrázolási módjai:

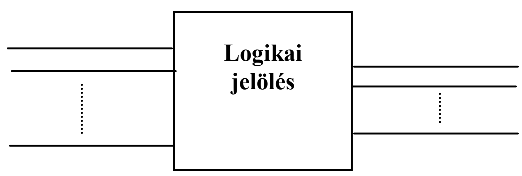
  - a logikai műveletek ***szimbólumaival*** megrajzolt ***logikai vázlat***,
  - síkban, vagy térben - a Veitch diagramból származtatott - ***minterm*** -, ***maxterm*** – diagram, illetve a ***Karnaugh*** – diagramok segítségével,
  - az idő függvényében rajzolt grafikon formájában.

- ***Logikai vázlat***

  A szimbólumokkal történő ábrázolás az áramköri megvalósítást segítő megoldás, amelyet az elmúlt fél évszázadban, több változatban is szabványosítottak. Az érvényes európai, és hazai szabványok közös jellemzői:

  - ***a szimbólum kerete négyszög,***
  - ***a négyszögbe írt jelölés utal a logikai funkcióra,***
  - ***a független változókat jelző bemenetek a keret bal oldalához,***
  - ***míg a függő változókat jelző kimenetek a keret jobb oldalához csatlakoznak.***

bemenetek



kimenetek

6. ábra

A be-, és kimenetek jeleit általában a csatlakozó vezetékre kell írni. (Ettől eltérő felírással az összetett szimbólumoknál találkozunk.)

Nemzetközileg a szabványosítást az 1970 – es években kezdték el. Addig országonként, gyártó cégenként szabványosított szimbólumokat használtak. A módokról, és azok változásáról a mellékletben adunk áttekintést. A 6.ábrán csak a logikai alapműveleteket szemléltető szimbólumokat mutatjuk be.

Magyarországon 1950-60 | TEXAS jelölések 1967-től | 1975-től szabványos

ÉS (AND)

ÉS-NEM (NAND)

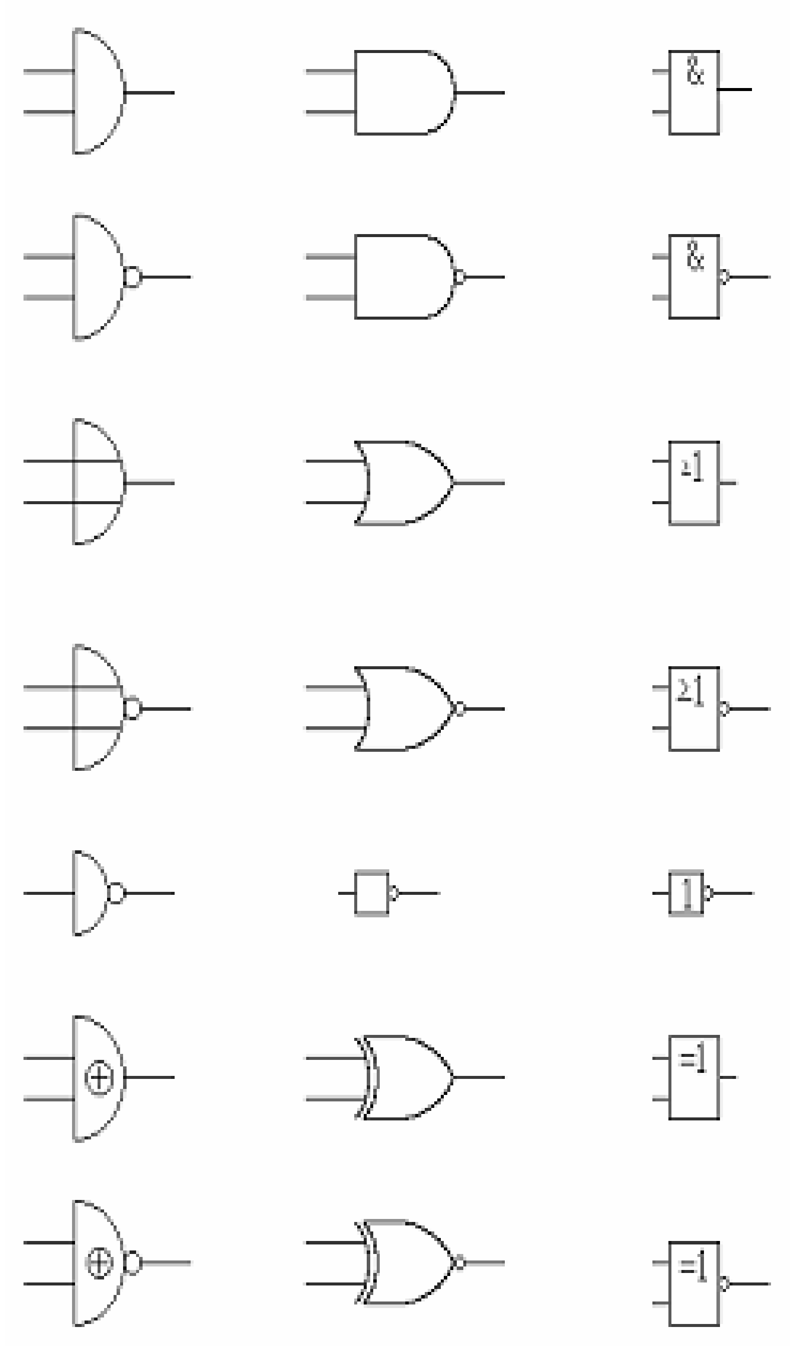
VAGY (OR)

VAGY-NEM (NOR)

NEM (INVERS)

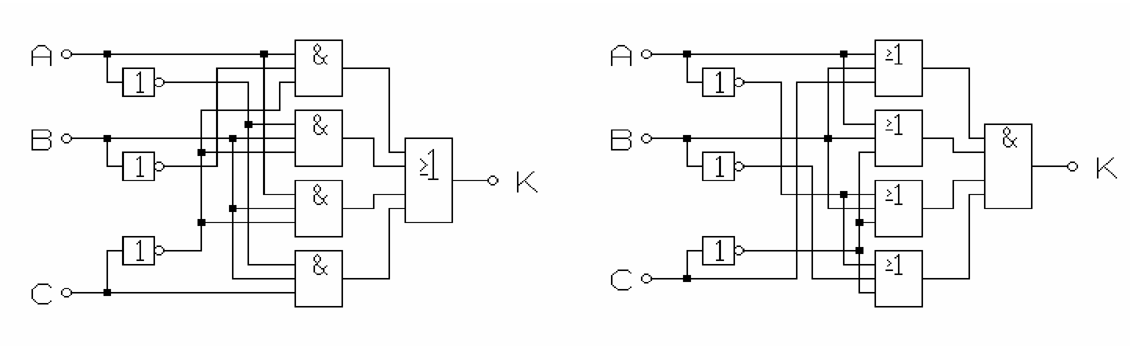
KIZÁRÓ-VAGY (XOR)

KIZÁRÓ-VAGY-NEM (NXOR) másképp **EGYENLŐ** (EQUALENCIA)



7. ábra

A fejezetben példaként felírt függvény (5.ábra) kétféle kanonikus alakjának logikai vázlatát mutatja a 8.a. és b. ábrák.



a. b.

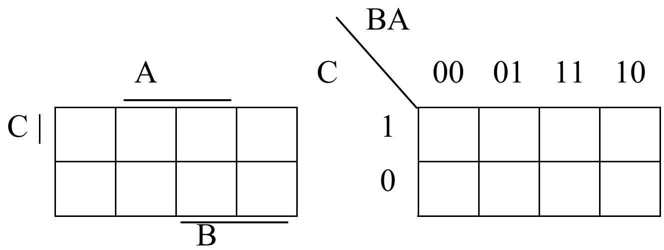
8. ábra

1.9. Grafikus ábrázolás

- ***Karnaugh diagram***

A grafikus ábrázolásainak egyik változata, hogy logikai sík-, vagy térbeli ***geometriai alakzatot*** rendelünk.

A függvényhez rendelt geometriai alakzat ***peremén*** adjuk meg a logikai ***változók jeleit***. Ezzel adjuk meg azt, hogy az alakzat melyik részén ***IGAZ*** értékű ez a változó. (Az alakzat másik részén – értelemszerűen – a változó HAMIS értékű.). Ezt a jelölés-rendszert ***peremezésnek*** nevezzük. A ***binárisan*** kódolt peremezésű változatot nevezzük ***Karnaugh*** táblázatnak. Használják még az ***oldal mellé húzott*** vonallal történő peremezést is. A tanulmányainkban a Karnaugh táblázatot fogjuk használni, mivel az igazságtáblázatból történő átírás egyszerűbb. Az 9.ábra háromváltozós (A B C) logikai függvény megadásához használható síkbeli elrendezés kétféle peremezését mutatja.



9. ábra

Mindkét változat formailag a Veitch diagramból származtatott. A különbségek a változók megadásának (a peremezésnek) módjában, valamint abban van, hogy egy elemi négyszög ***mintermet***, vagy ***maxtermet*** is jelképezhet. Egy n változós függvény $\mathbf{2^n}$ db elemi négyzetből álló táblázatban szemléltethető.

Az eljárás a 9.ábra alapján követhető. A halmazt egy négyszögben ábrázoljuk. Minden változó IGAZ értékéhez a teljes terület egyik felét, míg a HAMIS értékéhez, pedig a másik felét rendeljük. Az értékeket a négyszög szélére irt, ***vonallal*** (minterm / maxterm tábla vagy diagram), illetve ***kódolással*** (Karnaugh-diagram) adjuk meg. A továbbiakban a Karnaugh - diagramot használjuk. Több változó esetén a felezést úgy folytatjuk, hogy a változókhoz rendelt területeket jól meg lehessen különböztetni. A változók kódolását (kijelölését) úgy kell végezni, hogy az ***egymás melletti oszlopok***, ill. ***sorok*** mindig ***csak egy*** változóban ***térjenek*** el egymástól. A Hamming - távolság 1.

A háromváltozós Karnaugh - táblázat oszlopaihoz a BA változó-pár lehetséges érték-kombinációt rendeltük. A harmadik változó C értéke szerint két sora van a táblázatnak. Az egyikben C=0, a másikban, pedig C=1. Az egyes elemi négyszögekhez tehát a változók különböző értékvariáció tartoznak. A peremezés megváltoztatható, de csak úgy, hogy a szomszédos sorok, oszlopok egy változóban különbözhetnek. (A táblázat szélső oszlopai, illetve sorai mindig szomszédosak).

A 10.ábrán a négyváltozós Karnaugh diagram látható

DC \ BA	00	01	11	10
00				
01				
11				
10				

10. ábra

A 11.ábrán az 5, a 12.ábrán pedig a 6 változós táblázatot láthatjuk. (Az ábrázolási mód legfeljebb 6 változóig alkalmazható szemléletesen.)

Az öt-változós táblázatot célszerű két négy-változós táblázatból úgy kialakítani, hogy a két rész peremezése csak az egyik változóban - itt pl. a C – tér el egymástól.

ED \ CBA	000	001	011	010	100	101	111	110
00								
01								
11								
10								

11. ábra

A 6 változós táblázatnál függőlegesen duplázzuk meg a táblázat elemeit.

FED \ CBA	000	001	011	010	100	101	111	110
000								
001								
011								
010								
100								
101								
111								
110								

12. ábra

Így négy egyforma 4 változós egységeket kapunk. Az egyes rész-táblázatokban négy változót (ABED) azonosan variálunk. Az eltérés vízszintesen a C, míg függőlegesen az F változó.

Az eddigiekben csak az ábrázolás formai részével foglalkoztunk. Nézzük most meg a logikai tartalmat is. A két hozzárendelés szerint beszélünk ***Kp*** ill. ***Ks*** diagramról. A ***p*** index arra utal, hogy az elemi cellában logikai ***szorzat*** (produktum), míg az ***s*** a logikai ***összeget*** jelenti (summa). Tehát a ***Kp*** jelölés az ***ÉS-VAGY***, míg a ***Ks*** a ***VAGY-ÉS*** műveletes teljes függvényalakot adja meg.

A logikai függvényt ***diszjunkt*** alakját úgy kell a ***Kp*** diagramban ábrázolni, hogy a függvényben szereplő ***mintermeket*** reprezentáló cellákba ***1***-et írunk.

A ***konjunkt*** alakot ***Ks*** diagramban ábrázoljuk oly módon, hogy a megfelelő ***max-termeket*** jelentő cellákba írunk ***1***-t. (A 0-t egyik változatban sem szokták kiírni, a cella üres).

A fejezetben – az 5.ábrán adott igazságtáblázat - már leírt példa Karnaugh diagramjai láthatók a 13.a. és b. ábrákon.

K_p

C \ BA	00	01	11	10
0		1	1	1
1				1

a.

K_s

C \ BA	00	01	11	10
0	1		1	1
1			1	

b.

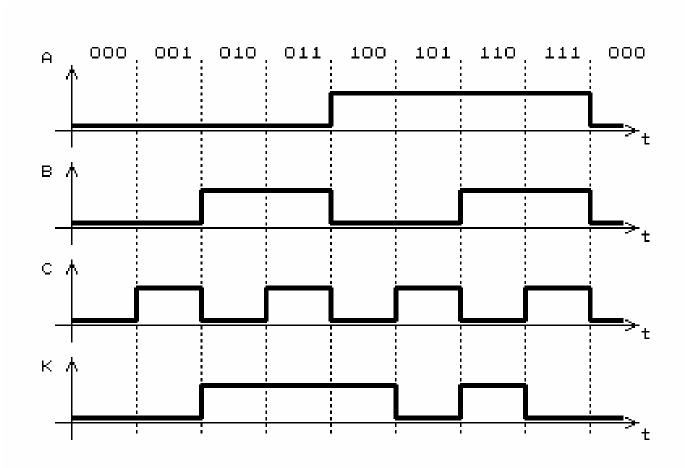
13. ábra

- ***Időfüggvény megrajzolása***

A függvény minden változójának ***időbeli lefolyását*** ábrázoljuk ***fázishelyesen*** egy-egy derékszögű ***koordináta*** rendszerben. A módszert elsődlegesen az egyes digitális áramkörök vizsgálatánál alkalmazzuk oly módon, hogy a bemeneteket (független változókat) ismert digitális jelekkel gerjesztjük. Az áramkör kimenetén – oszcilloszkóppal - mért jel a függvény értékének változását adja meg. A be-, és kimenetek jeleiből a vizsgált áramkör logikai függvényének bármelyik alakja meghatározható.

A fejezetben már ismert logikai függvény be-, és kimeneteinek időfüggvényét mutatja a 143. ábra. A bemeneteket bináris kód szerint változó kombinációsorozattal gerjesztjük

A szaggatott vonalak jelzik a gerjesztések változásának időpontjait. A matematikai leírásnál használt változó-súlyozással irtuk fel az egye kombináció bináris sorszámát. Ebből közvetlenűl kiovasható, hogy a K kimenet IGAZ értékű lesz, ha a bemeneteket a 2, 3, 4, és 6 sorszámú kombinációk valamelyike gerjeszti.



14. ábra

1.10. A logikai függvények egyszerűsítése

Az igazságtáblázat alapján felírt kanonikus alakú függvények a legtöbb esetben ***redundánsak***, tehát egyszerűsíthetőek. A redundancia azt jelenti, hogy a megadott információ több mint amennyi az egyértelmű függvényleíráshoz szükséges.

Az ***egyszerűsítés*** során a logikai algebra megismert tételeinek felhasználásával olyan alakot nyerhetünk, amelyben ***kevesebb művelet***, és vagy kevesebb ***változó*** szerepel. Az egyszerűsítésre azért van szükség, mert ezután a feladatot megvalósító logikai hálózat kevesebb áramkört, vagy programozott rendszer (mikrogép) programja kevesebb utasítást tartalmaz

Az ***algebrai*** módszer mellett kidolgoztak ***grafikus***, illetve ***matematikai*** egyszerűsítési eljárásokat is.

A felsorolt egyszerűsítési (minimalizálási) eljárásokat a fejezetben bemutatott igazságtáblázattal leírt logikai feladat segítségével ismertetjük. A 15. ábrán látható feladat igazságtáblázata:

C	**B**	**A**	**K**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

15. ábra

➢ *Algebrai egyszerűsítés*

A logikai algebra tárgyalásakor már bemutattunk néhány átalakítási eljárást. Itt egy újabb példa segítségével végezzük el a feladat legegyszerűbb alakjának megkeresését.

a. Egyszerűsítés a ***diszjunktív*** alakú függvényből

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

Először keressük meg, hogy vannak-e közös részeket tartalmazó ***mintermek***. Ezekből ”emeljük” ki a közös részeket!

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

A zárójelekben lévő mennyiségek értéke ***1***, ezért azok a logikai szorzatból elhagyhatók. A keresett, legegyszerűbb függvényalak a következő:

$$\mathbf{K = \overline{A}B + A\overline{C}}$$

b. Egyszerűsítés ***konjunktív*** alakú rendezett függvényből

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

Hasonlóan az előző egyszerűsítéshez itt is végezhetünk – a disztributív tulajdonság alapján - ”kiemeléseket” a ***maxterm*** - ekből.

$$\mathbf{K = ((A + B) + C\overline{C})\,((\overline{A} + \overline{C}) + B\overline{B})}$$

A $\mathbf{C\overline{C}}$ és $\mathbf{B\overline{B}}$ tényezők értéke ***0*** és ezért a logikai összegekből elhagyhatók. A keresett legegyszerűbb függvényalak tehát:

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

c. ***Igazoljuk*** a két alakból kapott függvények ***azonosságát***, vagyis hogy igaz az

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

egyenlőség.

Végezzük el a jobb oldalon a ”***beszorzást***”!

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

A kapott kifejezésben az első tényező 0. A negyedik tényezőt ”***szorozzuk***” 1-el.

$$\mathbf{0 + \overline{A}B + A\overline{C} + B\overline{C}(A + \overline{A}) = \overline{A}B + A\overline{C} + B\overline{C}A + B\overline{C}\overline{A}}$$

A közös részek ”***kiemelése***” után

$$\mathbf{\overline{A}B(1 + \overline{C}) + A\overline{C}(1 + B) = \overline{A}B + A\overline{C}}$$

a zárójeles kifejezések elhagyhatók, mivel értékük 1. A kapott eredménnyel igazoltuk az eredeti egyenlőség azonosságát.

Ezzel bizonyítottuk, hogy az igazságtáblázatból a két - ismertetett - módszer bármelyikével ugyanazt a függvényt kapjuk.

Összefoglalva: ***megállapíthatjuk, hogy az igazságtáblázatból rendezett ÉS-VAGY (diszjunktiv kanonikus) alakú vagy rendezett VAGY-ÉS (konjunktív kanonikus) alakú logikai függvényt írhatunk fel. A két alak azonos függvényt ír le.***

- ***Grafikus egyszerűsítés Karnaugh –táblázattal***

A leírt kikötések betartásával - az előző fejezetben megismert - mindkét logikai függvényalak (diszjunktiv, ill. konjunktív) ábrázolható, és egyszerűsíthető Karnaugh – diagram segítségével. A Karnaugh diagramok – mint ahogyan azt az előző fejezetben megismertük - az igazságtáblázatból közvetlenül felírhatók.

a. Kp diagram használata.

A Karnaugh diagram egyes celláiba kell beírni a független változók (A,B,C) megfelelő kombinációihoz tartozó függő változó (K) értéket (15.ábra).

Az A=0,B=0,C=0 kombinációnál a K értéke 0, tehát a BA=00 oszlop és C=0 sor által meghatározott cellába 0-t kell írni és így tovább.

C \ BA (K_p)	00	01	11	10
0		1	1	1
1				1

16. ábra

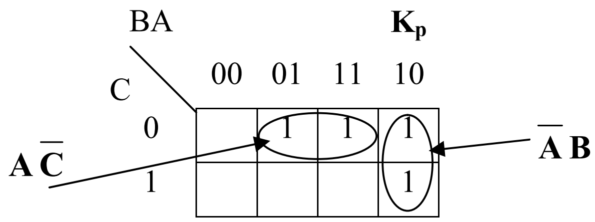
A 0 értékeket nem fontos beírni, ugyanis az egyszerűsítésnél csak az 1 értékű cellákat vesszük figyelembe.

Vizsgáljuk meg a diagram ***utolsó oszlopában*** lévő ***két*** cella tartalmát.

A ***felső*** cella tartalma az $\overline{A}B\overline{C}$, míg az ***alsó*** celláé $\overline{A}BC$ ***minterm***. Mivel mindkét cella értéke 1, azt jelenti, hogy mindkét minterm a függvény tagja, és közöttük VAGY kapcsolat van. A két minterm -ből álló függvényrész egyszerűsíthető.

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

A példa alapján is bizonyítottnak tekinthetjük, hogy ha ***két – élben érintkező*** – cellában 1 van, akkor ezek ***összevonhatók***, vagyis ***az a változó kiesik,*** amelyikben ***különböznek*** a cellák. Az összevonhatóságot ***lefedő hurokkal*** szokás jelölni (17.ábra):



17. ábra

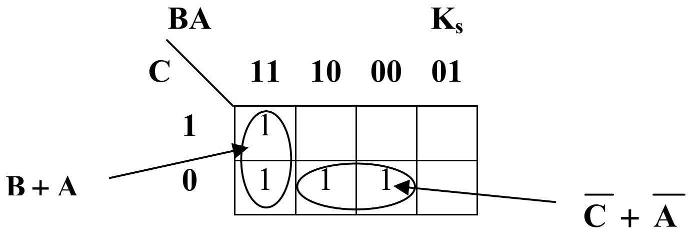
A lefedett (összevont) cellák VAGY kapcsolata adja az egyszerűsített függvényt:

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

b. Ks diagram használata.

Az egyszerűsített függvényalakoknál tárgyaltakhoz hasonlóan a ***Kp*** és a ***Ks*** diagramok is ***felrajzolhatók egymásból***.

Az átrajzolásnál a ***peremezés***, és a cella-értékek ***komplemens*** -ét kell írni, vagyis 0 helyett 1-e, és fordítva. A 18.ábrán látható a példa Ks diagramja:



18. ábra

A cellák most ***maxtermeket*** tartalmaznak, ezért az egyszerűsített függvény az összevonások (lefedések) közötti ÉS művelettel írható le:

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

c. *Több* cella összevonása.

A logikai függvények között vannak olyanok is, melyeknél ***többszörös*** algebrai ***összevonás*** is végezhető.

Keressük meg a következő négy (A,B,C,D) változós logikai függvény legegyszerűbb alakját!

A változókat súlyozzuk az $A \div 2^0, B \div 2^1, C \div 2^2, D \div 2^3,$ szerint. A függvény egyszerűsített alakja:

$$F = \sum^{4}(8,10,12,13,14,15)$$

Rajzoljuk meg a függvény Karnaugh táblázatát (19.ábra).

DC \ BA		(0) 00	(1) 01	(3) 11	(2) 10
(0)	00				
(4)	01				
(12)	11	1	1	1	1
(8)	10	1			1

19. ábra

A Karnaugh diagram – egyszerűsített alakú függvény alapján történő – felrajzolását könnyíti, ha az egyes ***sorok*** és ***oszlopok súlyát*** decimálisan (a zárójelben lévő számok) is jelöljük. Ezt tettük a zárójelbe írt számokkal.

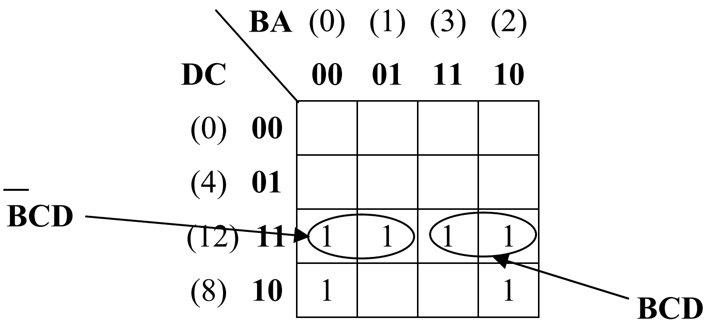
Először írjuk fel a harmadik sor rész-függvényét algebrai alakban, mivel mindegyik cellában 1 értékű a függvény.

$$\mathbf{\overline{A}\,\overline{B}CD + \overline{A}BCD + ABCD + A\overline{B}CD = \overline{B}CD(\overline{A} + A) + BCD(A + \overline{A}) =}$$

$$\mathbf{= \overline{B}CD + BCD = CD(\overline{B} + B) = CD}$$

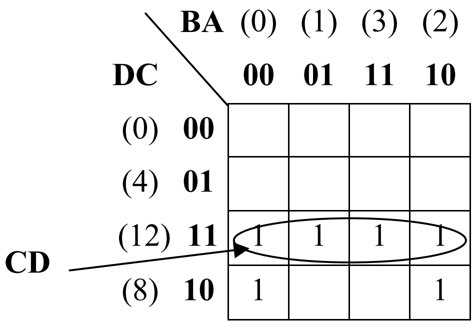
Az algebrai sorozatos kiemelések után két változó (A,B) kiesett.

Ugyanezt kövessük végig Karnaugh diagramon is. A 20.ábrán az első egyenlőségjel utáni két kettős összevonás látható.



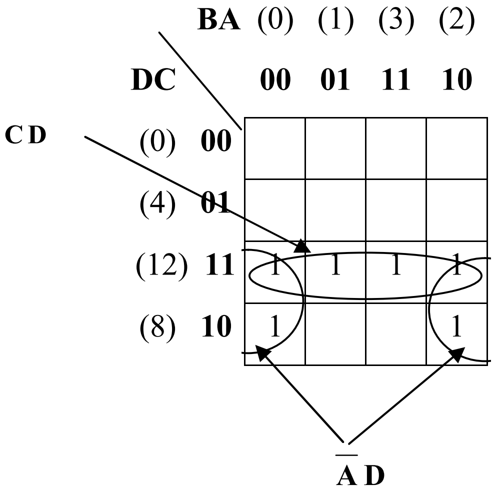
20. ábra

Mindkét lefedésnél kiesett az A változó. A két háromváltozós rész-függvényben közös a CD logikai szorzat, tehát összevonható. A grafikus módszernél ez egy közös lefedéssel jelölhető (21.ábra).



21. ábra

Hasonló négyes csoportot alkotnak a 8,10,12,14 sorszámú mintermek is, tehát összevonhatók (22.ábra).

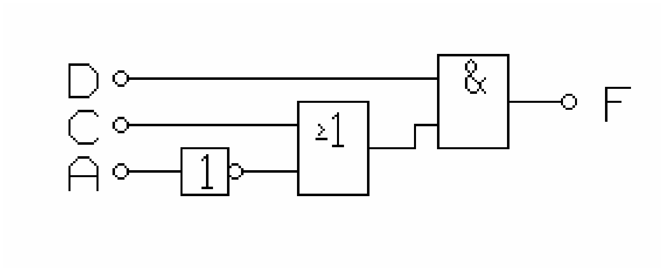


22. ábra

Az egyszerűsített függvény a két részfüggvény logikai összege, amely még algebrailag tovább egyszerűsíthető:

$$\mathbf{F = D\overline{A} + DC = D(\overline{A} + C)}$$

Az utolsó egyszerűsítés eredményeként kaptuk a legkevesebb művelettel megvalósítható alakot. A függvény logikai vázlata látható a 23.ábrán.



23. ábra

Összefoglalás:

A grafikus függvényegyszerűsítés szabályi:

- ***a lefedhető (összevonható) cellák száma 2^n (n pozitív egész szám), ha azok kölcsönösen szomszédosak,***
- ***a kölcsönösen szomszédos meghatározást úgy kell érteni, hogy a kiinduló cellától kezdve a élben érintkező szomszédos cellákon keresztül 2^n számú lépés után az kiindulóhoz jutunk vissza,***
- ***a lefedett cellákból a kitevőnek (n) megfelelő számú változó esik ki, amelyek a lefedés alatt változnak,***
- ***minden 1 -t tartalmazó cellát legalább egyszer le kell fedni.***

1.11. Példák

➢ ***Algebrai kifejezések átalakítása***

1. Igazoljuk a tételek között felsorolt $AB + BC + \overline{A}C = AB + \overline{A}C$ azonosságot!

- **Első** lépésként a baloldal mindhárom tagját kibővítjük úgy, hogy szerepeljen bennük mindegyik független változó (A,B,C).

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

Az így kapott hat szorzatot tartalmazó kifejezésben az egyforma aláhúzású mintermek azonosak, tehát egy elhagyható. Ezek közül egy - egy elhagyható.

- **Második** lépésként a bővítés fordítottját végezzük, vagyis ahol lehet az azonos tényezőket, kiemeljük.

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

A zárójelekben levő kifejezések 1 értékűek. ***Ezzel igazoltuk az eredeti azonosságot.***

2. Algebrai kifejezés tagadása (a De Morgan - tételek alkalmazása).

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

Az átalakításnál először a De Morgan - tételt használtuk (első és második sor). A következő lépésként az első két zárójeles kifejezés logikai szorzatát (ÉS művelet) képeztük (az eredmény aláhúzva).

Az aláhúzott részt célszerű tovább egyszerűsíteni az $A\overline{A} = 0$, és a $B\overline{B} = 0$ tényezők elhagyásával, illetve a $\overline{C}\,\overline{C} = \overline{C}$ helyettesítéssel. Majd tovább is egyszerűsíthető a $\overline{C}$ kiemelésével.

$$(\underline{AB + A\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{C} + B\overline{C} + \overline{C}}) = AB + \overline{A}\,\overline{B} + \overline{C}(A + \overline{B} + \overline{A} + B + 1) =$$
$$= AB + \overline{A}\,\overline{B} + \overline{C}$$

A zárójelben levő kifejezés azonosan 1, mert a logikai összeadás egyik tagja 1. Térjünk vissza az eredeti kifejezéshez, amelynél a zárójelbe tett kifejezések ”összeszorzása”, majd a lehetséges további átalakítás után (pl. az aláhúzott kifejezések értéke 0 stb.) kapjuk meg a végeredményt.

$$= (AB + \overline{A}\,\overline{B} + \overline{C})(A + B + \overline{C}) = ABA + \underline{\overline{A}\,\overline{B}A} + A\overline{C} + ABB + \underline{\overline{A}\,\overline{B}B} + \overline{C}B +$$
$$+ AB\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + \underline{\overline{C}\,\overline{C}} = AB + 0 + A\overline{C} + AB + 0 + \overline{C}B + AB\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + 0 =$$
$$= AB(1 + 1 + \overline{C}) + \overline{C}(A + B + \overline{A}\,\overline{B}) = AB + \overline{C}$$

3. Igazoljuk a $\overline{\overline{DF} + E\overline{F}} = F + D\overline{E}$ azonosságot!

Első megoldás:

$$\overline{\overline{D}\,\overline{F}+E\overline{F}} = \overline{(\overline{D}+E)\overline{F}} = \overline{(\overline{D}+E)}+F = D\overline{E}+F$$

Második megoldás**:**

$$\overline{\overline{D}\,\overline{F}+E\overline{F}} = (\overline{\overline{D}\,\overline{F}})(\overline{E\overline{F}}) = (D+F)(\overline{E}+F) = DF+FF+D\overline{E}+\overline{E}F =$$

$$DF+F+D\overline{E}+\overline{E}F = F(D+1+\overline{E})+D\overline{E} = F+D\overline{E}$$

- ***Logikai függvények egyszerűsítése***

4. **Egyszerűsítse** a **0,1,5,7,11,13,15** indexű MINTERM-ket tartalmazó 4 változós logikai függvényt! NAND kapuk alkalmazásával rajzolja meg a hálózat logikai vázlatát!

Kiinduló adatok:

- *Független változók, és súlyozásuk:* $\mathbf{A \div 2^0}$, $\mathbf{B \div 2^1}$, $\mathbf{C \div 2^2}$, $\mathbf{D \div 2^3}$
- *A megvalósítandó függvény:* $\mathbf{K = \sum^{4} (0,1,5,7,11,13,15)}$

Megoldás:

- *Karnaugh diagram felrajzolása:*

Kp (mintermek)

DC \ AB		(0) **00**	(1) **01**	(3) **11**	(2) **10**
(0)	**00**	1	1		
(4)	**01**		1	1	
(12)	**11**		1	1	
(8)	**10**			1	

- *Összevonási lehetőségek*

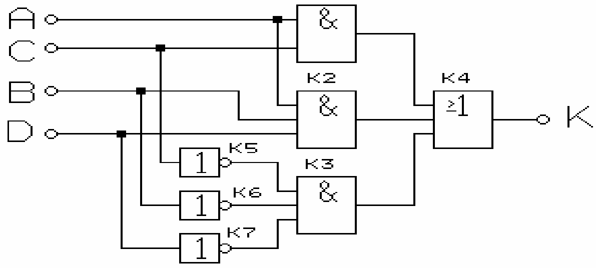
$\overline{B}\,\overline{C}\,\overline{D}$ $\quad$ **A C** $\quad$ **A B D**

- *Egyszerűsített függvény*

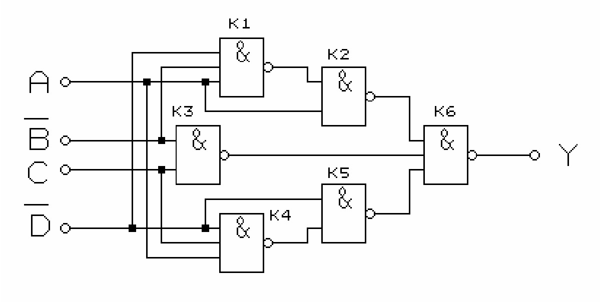
$$\mathbf{K = A\,C + A\,B\,D + \overline{B}\,\overline{C}\,\overline{D}}$$

- *Logikai vázlat*

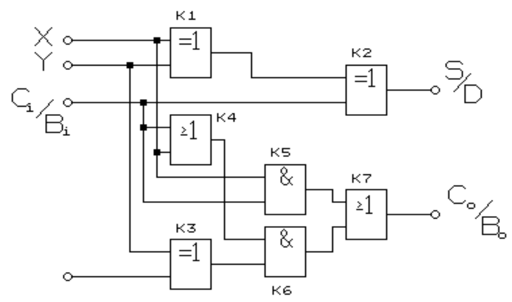
Az egyszerűsített függvény alapján megrajzolható kétszintű ÉS – VAGY hálózat



5. **Határozza** meg az ábra szerinti kombinációs hálózat logikai függvényét!



6. **Határozza** meg az ábra szerinti logikai hálózat kimenetének a függvényét!



7. **Egyszerűsítse** és két bemenetű NOR kapukkal, valósítsa meg azt a 4 változós függvényt, amely a 0,1,2,4,5,6,7,13,15 indexű MAXTERM - ket tartalmazza!

8. **Határozza** meg a 0,2,3,4,6,8,10,11 indexű MINTERMEK -et tartalmazó 4 változós függvény egyszerűsített konjunktív alakját! Rajzolja meg a megvalósítás NAND kapus logikai vázlatát!

1.12. Ellenörző kérdések

- Melyek a Boole algebra axiómái, alapműveletei, és alaptételei?
- Mi az igazságtáblázat, és mire használható?
- Milyen axiómák alapján lehet logikai kifejezéseket átalakítan?
- Mire vonatkoznak a de Morgan tételek?
- A logikai függvény melyik kanonikus alakja írható fel közvetlen az igazságtáblázatból?
- Hogyan írható át egy logikai függvény diszjunkt alakja konjunk alakúvá?
- Melyek az algebrai egyszerűsítés feltételei?
- A függvény egyszerűsítésének milyen grafikus módszerét ismeri?
- Milyen alakú függvényt ír le a K_s diagram?
- Melyek a Karnaugh diagram peremezésének feltételei?
- Mi a formai különbség a Kp és minterm tábla, illetve a Ks és maxterm tábla között?
- Hány szintű logikai függvényt kapunk a Karnaugh diagramos egyszerűsítésnél?

2. Aritmetikai alapfogalmak

A digitális berendezésekben – mérőegységek, számítóművek stb. – gyakori feladat ***aritmetikai műveletek*** végzése. Az eddig megismert logikai műveletek változói kétértékűek. A számok ***bináris*** – kettes – ***számrendszerben*** való ábrázolásánál is a ***0***, és az ***1*** számjegyeket használjuk. A későbbiekben igazoljuk, hogy az aritmetikai műveletek elvégzése logikai műveletekkel lehetséges. Itt most összefoglaljuk – az alábbi - alapvető ***aritmetikai fogalmakat:***

- ***szám, számjegy, számrendszer,***
- ***számábrázolási formák,***
- ***aritmetikai alapműveletek algoritmusai.***

➢ ***Szám, számjegy, számrendszer***

Röviden összefoglaljuk – a korábbi tanulásaikban már megismert – fogalmakat.

- A ***szám***:

A ***szám*** „valaminek" a számosságát, mennyiségét, értékét megadó ***jelcsoport***. A jelcsoportok mind a használt ***jelek***, mind a ***jelölési-rendszer*** felépítése szerint változtak az idő folyamán.

- A ***számjegy***:

A ***számjegyek*** a számként használt jelcsoport egyes ***jelei***, amelyekhez ***konkrét értéket*** rendeltek. Egy jelölési-rendszeren belül véges számú számjegy van.

- A ***számrendszer***:

A ***számrendszer*** határozza meg, hogy a használt ***jelekből*** milyen ***módon***, (algoritmus szerint) kell ***leírni*** (ábrázolni) egy ***számot***. A számrendszerek a korai időszakokban kultúránként különböztek. A tudományok, a technika fejlődésének eredményeként egységes számrendszerekről beszélhetünk.

- A ***RÓMAI*** számrendszer

A mai napig szélesebb körben is ismert számrendszert a rómaiak alkották meg. A ***római*** számokban a következő *7* ***számjegy*** (jel) létezik (A zárójelbe írjuk a jelhez rendelt értéket tízes számrendszer szerinti jelöléssel.)

Számjegyek és értékük

I (1)	**X** (10)	**C** (100)	**M** (1000)
V (5)	**L** (50)	**D** (500)	

A számjegyek megfelelő szabályok szerinti egymás utáni írásával fejezték ki a szám-értékeket. Tulajdonképpen a tízes váltószám, amely valószínűleg ujjaink számából ered megtalálható a számrendszer logikájában. A számalkotás szabályát itt nem részletezzük, csak egy példával illusztráljuk.

$$\mathbf{M\ C\ M\ L\ XX\ IV = 1974_{10}}$$

A romai számok segítségével értékeket - korlátozott terjedelemben – ki lehet fejezni. Számtani műveletek ezekkel nem végezhetők.

- A ***strukturált*** számrendszerek

Pontosan nem ismert, hogy a mai értelemben vett számrendszerek alapjait mikor és hol fektették le. Az európai kultúrában, és az abból építkezőkben használt számjegyek arab eredetűek.

A ma használt számrendszerek egy-egy ***alapszámra*** épülnek, és a számjegyek ***száma*** az alapszám ***értéke***. Felépítésük, pedig az alapszám egész számú hatványa - ***helyérték*** – szerint tagolódik. Általános leírása:

$$\mathbf{Z = (x_{n-1}\, B^{n-1} + x_{n-1}\, B^{n-1} + ... + x_1\, B^1 + x_0\, B^0) + (x_{-1}\, B^{-1} + x_{-2}\, B^{-2} + \dots + x_{-p}\, B^{-p})}$$

<u>egész rész</u> <u>tört rész</u>

ahol,	**B**	a számrendszer ***alapszáma,***
	$\mathbf{x_i}$	az i - ik helyérték ***számjegye*** ($0 \leq x \leq B-1$),
	n	az ***egész*** rész helyértékeinek száma,
	p	a ***tört*** rész helyértékeinek a száma.

- A ***leggyakrabban használt*** számrendszerek:

	alapszám	***számjegyek***
Tízes (decimális)	B = 10	0, 1, …8, 9
Kettes (bináris)	B = 2	0, 1
Nyolcas (oktális)	B = 8	0, 1,…6, 7
Tizenhatos (hexadecimális)	B = 16	0, 1, …. 9, A, B, C, D, E, F

Ismert módon a számok felírásánál csak az egyes helyértékekhez tartozó számjegyeket írjuk balról-jobbra, a legnagyobb helyértékű számjeggyel kezdve. A különböző alapszámok, valamint a részben azonos számjegyek miatt, a számoknál jelezni kell, hogy az milyen számrendszerben értendő. A jelzést lehet a szám ***előtt*** – ***prefix*** -, vagy a szám ***után*** – ***suffix*** – megadni. Legtöbb esetben a decimális számokat jelzés nélkül írják.

pl.

számrendszer	***jel nélkül***	***prefix***	***suffix***	
decimális:	1456	**0d** 1456	1456 **d**	1456_{10}
bináris	-	**0b** 100110	100110 **b**	100110_{2}
oktális	-	**0o** 273	273 **q**	273_{8}
hexadecimális	-	**0x** 1A2D	1A2D **h**	$1A2D_{16}$

Megjegyzés: a jelzőkben, illetve számjegyekként használt betűk kis-, és nagybetűk is lehetnek. A ***hexadecimális*** számoknál, ha azok betűvel kezdődnek, akkor egy ***0***-t kell írni a ***szám elé,*** pl. 0A4CF.

- A számok ***komplemens*** -e (***kiegészítő*** -je).

A szám ***komplemens*** -e (kiegészítője) - mint a neve is utal rá - az érték, amely a számot ***kiegészíti*** a számrendszer egy ***adott értékéhez***. A definíció szerint bármely értékhez számolhatnánk a kiegészítőt, de gyakorlati jelentősége csak az alábbi két változatnak van.

A kiegészítés történhet:

- a szám ***nagyságrendjébe*** tartozó ***legnagyobb*** értékéhez, vagyis a $(B^n - 1)$ -hez,
- a számnál ***egy nagyságrenddel*** nagyobb ***legkisebb értékéhez***, vagyis a B^n -hez,

ahol **B** az alapszám, és **n** a nagyságrendek száma. Könnyen belátható, hogy a $(B^n - 1)$ értéket - bármely számrendszerben - az ***n*** db. ***legnagyobb számjegyből*** álló szám adja, míg a B^n értékét – a legalacsonyabb helyértéktől kezdve – ***n*** db. ***legkisebb számjegyből***, és az ***n+1.*** helyen az ***eggyel*** nagyobb számjegyből álló szám adja.

Pl.	n=5 esetén:	
	$(B^n - 1)$	B^n
decimális számoknál:	99999_d	100000_d
bináris számoknál	11111_b	100000_b
hexadecimális számoknál:	$FFFFF_h$	100000_h

Az első meghatározás szerintit nevezik ***(B-1)***-es, míg a másodikat ***B***-s komplemens -nek. A B a számrendszer alapszáma (radix).

A ***Z*** szám ***(B-1)***-es komplemens –ét $\overline{\overline{Z}}$ - al, míg a ***B***-s komplemens –ét $\overline{Z}$ -al jelöljük. A kiegészítők – definíció szerinti - kiszámítása különbség-képzéssel történik. A számítás algoritmusa:

$$\overline{\overline{Z}} = (B^n - 1) - Z$$

$$\overline{Z} = B^n - Z$$

A választott számrendszer alapján beszélhetünk:

- a decimális számoknál ***kilences***-, illetve ***tízes-***, a
- a bináris számoknál ***egyes***-, és ***kettes***-,

komplemens –ről. (Más alapszám esetén az elnevezés hasonlóan adható meg.)

Az átszámítást – a kivonáson kívül – más eljárásokkal is elvégezhetjük. Előbb vezessük be a ***számjegy - komplemens*** fogalmát, amely az adott számjegy kiegészítő értéke a legnagyobb számjegyhez.

A Z szám (B-1)-es komplemensét megkapjuk, ha mindegyik helyértéken az adott számjegy kiegészítőjét írjuk:

$Z = 356_d$ $\qquad \overline{\overline{Z}} = 643_d$

$Z = 100110_b$ $\qquad \overline{\overline{Z}} = 011001_b$

$Z = 3A2Bh$ $\qquad \overline{\overline{Z}} = C5D4h$

A Z szám B-s komplemens –ét kétféle módon is megkaphatjuk, ha figyelembe vesszük, hogy a kétféle kiegészítő ***különbsége*** – bármilyen B értéknél – ***1***, mivel $\boldsymbol{B^n - (B^n - 1) = 1}$.

a. Képezzük a Z szám (B-1)-es komplemens –ét, és hozzáadunk 1-et.

$$\overline{Z} = \overline{\overline{Z}} + 1$$

$Z = 356_d$ $\quad \overline{Z} = \overline{\overline{Z}} + 1 = 643_d + 1 = 644_d$

$Z = 100110_b$ $\quad \overline{Z} = \overline{\overline{Z}} + 1 = 011001_b + 1 = 011010_b$

$Z = 3A2Bh$ $\quad \overline{Z} = \overline{\overline{Z}} + 1 = C5D4h + 1 = C5D5h$

b. A legkisebb helyértéktől kezdve a 0-kat leírjuk, az első „értékes” számjegy helyére a számjegy-kiegészítő + 1 értéket, míg a további számjegyek helyére, pedig azok kiegészítőjét írjuk.

$Z = 356_d$ $\quad \overline{Z} = 644_d$

$Z = 100110_b$ $\quad \overline{Z} = 011010_b$

$Z = 3A2Bh$ $\quad \overline{Z} = C5D5h$

A ***digitális*** számítógépek bináris számokkal végeznek aritmetikai műveleteket . A ***negatív*** előjelű számoknál a ***kettes - komplemens*** használata gyorsabb műveletvégzést tesz lehetővé.

- A különböző számrendszerek közötti ***átszámítás***

A műszaki gyakorlatban leggyakrabban a ***decimális***, ***bináris***, és a ***hexadecimális*** számrendszereket használják. A következőkben röviden áttekintjük az átszámítások algoritmusát. Az ***emberek*** számára legfontosabb a ***decimális*** forma, mivel minden közérdekű számleírás ebben a formában történik. A számok ***gépi*** tárolása, és az azokkal végzett műveletek szinte kizárólag ***bináris*** rendszerben történik.

Általánosan az egyes számrendszerek közötti váltást (átszámítást) az új rendszer ***alap-számával*** történő ***sorozatos osztással*** végezhetjük el.

Először a ***decimális*** – ***bináris*** átalakítást ismételjük át. Számítsuk ki a 107_d érték bináris megfelelőjét.

107	**1**	2^0
53	**1**	2^1
26	**0**	2^2
13	**1**	2^3
6	**0**	2^4
3	**1**	2^5
1	**1**	2^6
0		

Tehát $\mathbf{107_d = 1101011_b}$

Gyakran van szükség a ***bináris*** – ***hexadecimális*** átszámításra is, mivel a számítástechnikai megjelenítés legtöbbször – a ***kisebb helyfoglalás*** érdekében – a bináris helyett a hexadecimális alakot használja.

Az átalakításnál 16 –al történő sorozatos osztást oly módon végezhetjük, hogy a bináris szám – legkisebb helyértékétől kezdődő – ***négy-négy*** számjegye helyett írjuk be a megfelelő ***hexadecimális*** számjegyet. Számítsuk át az előző példa értékét hexadecimális alakra.

$$\mathbf{107_d = 0110|1011_b = 6B_h}$$

$$\mathbf{6 \quad\quad B}$$

Az utóbbi átszámítás – a leírtak szerint - könnyen elvégezhető fejben is. Csupán a hexadecimális számjegyek bináris megfelelőjét kell kiszámítani, vagy megjegyezni.

➢ ***Számábrázolási (számírási) formák***

Az előzőekben csak a szám leírásának változatairól adtunk – a teljesség igénye nélkül – ismétlő áttekintést.

A műszaki, és egyéb gyakorlatban is legtöbbször ***különböző előjelű*** mennyiségek mérőszámait kell felírni, és azokkal műveletet végezni. A következőkben tömören – a teljesség igénye nélkül – összefoglaljuk azokat az ***előjegyes számleírási*** (számábrázolási) formákat, amelyeket a számításainkban használunk.

- ***Előjeles abszolút-értékes*** ábrázolás

A számleírás ilyen formáját használjuk a hétköznapi gyakorlatban a nyomtatott, és egyéb dokumentumokban. A szám ***pozitív***, vagy ***negatív*** voltát nem számjeggyel, hanem a +, vagy a – ***írásjellel*** adjuk meg a szám előtt. A szám értékét mindkét esetben ***abszolút-értékével*** írjuk. (Ez megfelel a számegyenesen jobbra-balra történő ábrázolásnak.)

- ***Előjegyes*** számábrázolás

A digitális számítógépek mind a ***számjegyeket***, mind a különböző ***írásjeleket*** kétértékű bitekkel tárolják. Az írásjelek ***kódolt*** formája ***8 bitet*** foglal le. A helytakarékosság, valamint egyszerűbb műveletvégzési célból is, az ***előjelet*** is ***egy bittel*** – az ***előjegy*** –el – adják meg. A műszaki gyakorlatban ***0*** a ***pozitív***, az ***1*** pedig a ***negatív*** szám előjegy -e

Így beszélünk az ***előjegyes*** – számábrázolásról. A számrész megadási módja szerint megkülönböztetünk:

– előjegyes ***abszolút - értékes***, valamint

– előjegyes ***komplemens*** -es

formákat.

Az ***abszolút-értékes*** leírás tulajdonképpen az ***előjeles*** ábrázolás gépi változata. Ilyen alakú számokkal a műveletvégzés viszonylag összetett algoritmus szerint végezhető. A kijelölt, és az elvégzendő művelet (összeadás, vagy kivonás) a tényezők előjeleitől is függ. A tényleges műveletvégzés előtt döntés sorozatot kell végezni.

A ***komplemens*** –es ábrázolásoknál a ***pozitív*** számokat a számrész ***abszolút – értékével***, míg a ***negatív*** számokat, pedig a számrész valamelyik ***kiegészítőjével*** (komplemensével) adjuk meg.

pl. Írjuk fel a $+107_d$, illetve a -107_d számokat a különböző számábrázolási formában!

Előjeles decimális	**107**	**-107**
Előjegyes abszolutérték -es	**0 1101011**	**1 1101011**
1-es komplemens -ű	**0 1101011**	**1 0010100**
2-es komplemens -ű	**0 1101011**	**1 0010101**

- ***Számok normál alakja***

A ***műszaki*** gyakorlatban, főleg a számítógépek széleskörű elterjedése előtt a különböző számítási műveletek elvégzését könnyítette az a számok ***normál alakban*** történt megadása.

A normál alak ***két részben*** adja meg a számot, mégpedig a ***számrészben***, és az ***exponenciális*** részben.

A számrészben a ***törtvessző előtt*** csak ***egyetlen*** – a legnagyobb helyértékű – ***számjegyet*** írjuk, míg a többi számjegy ***törtrészként*** szerepel. Utána kell leírni az exponenciális részt, mint szorzó tényezőt, amely az ***alapszám*** (B) ***n*** -ik hatvány. Az ***n*** kitevő határozza meg, hogy a az ábrázolt szám milyen ***nagyságrendű***.

8.Példa. $\mathbf{3{,}1023 * 10^4 = 3\ 1023}$

$\mathbf{5{,}234 * 10^{-2} = 0{,}05234}$

$\mathbf{25\ 90{,}12 = 2{,}59012 * 10^3}$

Az alap és a normál alakok közötti átírás a példákból egyértelmű. Az n kitevő formailag azt a számot jelenti, amennyivel a tizedes-vesszőt ***jobbra***, vagy ***balra*** kell vinni. Az irányt a ***kitevő előjele*** adja.

- ***Bináris számok lebegőpontos (float) alakja***

A skalár számok ***lebegőpontos*** (float) ábrázolása, és tárolása - az IEEE-754 sz. szabványnak megfelelően - ***4 bájtban (32 bit)*** történik.

Az ábrázolási forma, a ***normál alakú*** számábrázolásnak a ***bináris*** számrendszerben történő alkalmazása.

A lebegőpontos szám két része az aktuális számot megadó un. ***mantissza***, és a nagyságrendet megadó kitevő, vagy másképp ***exponent***.

A ***kitevő*** 8 bites, amely 0 – 255 közötti érték adható meg. A ***kettes komplemens -***ű ábrázolás – az érték ***127***-el történő ***eltolása*** - lehetővé teszi, hogy negatív kitevőjű értéket is lehessen megadni, és ezzel a +128 és - 127 az értékkészlet szélső értékei.

A ***szám*** 24 biten fejezhető ki, de ebből ténylegesen csak ***23***-at, a tört-vesszőt követő részt tartalmazza a ***mantissza***. A normál alakú számábrázolásban csak egy, a 0-tól különböző számjegy lehet az egész részben. A bináris számoknál ez az 1, amit nem fontos megadni, mivel ez minden számnál azonos. Így lehet a 24 bites számot 23 biten megadni.

A négy bájtban – 32 biten - ábrázolt szám ***legnagyobb helyértékű*** bit a szám ***előjegy*** -e (signum).

A leírtak szerint tárolt lebegőpontos szám felépítése az alábbi:

SEEE EEEE EMMM MMMM MMMM MMMM MMMM MMMM

ahol:

S az előjegy bit, amely 0 értéke a pozitív, az 1, pedig a negatív számot jelzi,

E a 8 bites kitevő,

M a 23 bites mantissza.

Példa:

A **-12,5** értékű decimális szám lebegőpontos ábrázolásban **0xC1480000** lesz (a 32 bit helyett a rövidebb hexadecimális formát írtuk).

Értelmezzük az ábrázolási elv ismeretében az adott számot.

Forma	**SEEEEEEE EMMMMMMM MMMMMMMM MMMMMMMM**
Bináris	**11000001 01001000 00000000 00000000**
Hex.dec.	**C1 48 00 00**

A legnagyobb helyértékű bit (**S**) 1, tehát a szám negatív.

Az következő nyolc bit (E-k) **10000010** a kitevőt adja, ha ebből levonjuk az eltolást, a 127-t. A binárisan leírt exponens decimális értéke 130, amelyből levonva 127-t 3-at kapunk, amely a tényleges kitevő.

Az utolsó 23 bit a mantissa (M-ek):

10010000000000000000000

Mivel ez csak a törtvessző utáni rész, ezért még hozzá kell írnunk az egész-részt, vagyis 1-t. Az így kapott érték:

1.10010000000000000000000

amelyet szorozni kell $\mathbf{2^3}$ -al. Ekkor kapjuk meg a lebegőpontosan felírt szám ***abszolút-értékét***:

$$\mathbf{1100.10000000000000000000_b}$$

A szám egész része: 1100b binárisan, és átszámítva

$$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 12$$

decimális érték.

A törtvesszőt követő bináris rész: .100…b , átszámítva

$$(1 \times 2\text{-}1) + (0 \times 2\text{-}2) + (0 \times 2\text{-}3) + \ldots = 0.5$$

decimális érték.

A két rész összege, és az előjel adja a lebegőpontosan ábrázolt szám decimális értékét. Tehát igazoltuk, hogy, **0xC1480000** a **-12.5** szám lebegőpontos (float) formájú megadása.

- ***Kódolt decimális számok***

A tízes számrendszerbeli számok közvetlenleírása, tárolása kódolt változatban is történhet. Miután a számítógépekben csak kétértékű elemi információk (bit-ek) tárolhatók, ezért a tíz számjegy csak ***több bit***-ből álló ***kód***-al helyettesíthető (írható le).

A tízes számrendszer számjegyeinek megadásához legkevesebb ***4 bitből*** álló kód szükséges, mivel ***3*** bittel csak ***8 érték*** különböztethető meg, viszont tíz értéket kell megkülönböztetnünk. A ***4 bites*** bináris kód viszont ***16*** különböző ***információt*** hordozhat, ezért ***10 értékhez*** rendelik a ***decimális számjegyeket***, és ***hat*** értéket ***nem*** használnak.

A 4 bites összerendelés, vagy más néven kódolás – az alábbi táblázatban bemutatott - három változatát használják.

BCD

	8	4	2	1
0	***0***	***0***	***0***	***0***
1	***0***	***0***	***0***	***1***
2	***0***	***0***	***1***	***0***
3	***0***	***0***	***1***	***1***
4	***0***	***1***	***0***	***0***
5	***0***	***1***	***0***	***1***
6	***0***	***1***	***1***	***0***
7	***0***	***1***	***1***	***1***
8	***1***	***0***	***0***	***0***
9	***1***	***0***	***0***	***1***
Nem használt	1	0	1	0
	1	0	1	1
	1	1	0	0
	1	1	0	1
	1	1	1	0
	1	1	1	1

Aiken

	2	4	2	1
0	***0***	***0***	***0***	***0***
1	***0***	***0***	***0***	***1***
2	***0***	***0***	***1***	***0***
3	***0***	***0***	***1***	***1***
4	***0***	***1***	***0***	***0***
Nem használt	0	1	0	1
	0	1	1	0
	0	1	1	1
	1	0	0	0
	1	0	0	1
	1	0	1	0
5	***1***	***0***	***1***	***1***
6	***1***	***1***	***0***	***0***
7	***1***	***1***	***0***	***1***
8	***1***	***1***	***1***	***0***
9	***1***	***1***	***1***	***1***

3 többletes (Stibitz)

	(8 4 2 1)-3			
Nem	0	0	0	0
	0	0	0	1
	0	0	1	0
0	***0***	***0***	***1***	***1***
1	***0***	***1***	***0***	***0***
2	***0***	***1***	***0***	***1***
3	***0***	***1***	***1***	***0***
4	***0***	***1***	***1***	***1***
5	***1***	***0***	***0***	***0***
6	***1***	***0***	***0***	***1***
7	***1***	***0***	***1***	***0***
8	***1***	***0***	***1***	***1***
9	***1***	***1***	***0***	***0***
haszn	1	1	0	1
	1	1	1	0
	1	1	1	1

24. ábra

A bemutatott kódok közül a ***BCD***-kód (Binary Coded Decimal), és az ***Aiken***-kód súlyozottak, ami azt jelenti, hogy az egyes bitek értéke ***2 hatványaival*** kifejezhető. A ***Stibitz***-kód ***eltolt-súlyozású***, ami azt jelenti, hogy a kód ***bináris*** értéke ***3-al több*** mint a hozzá rendelt ***decimális*** érték. Az utóbbi két kód szimmetrikus felépítésű, ugyanis a szaggatott vonaltól, - mint szimmetria tengelytől – egyenlő távolságra lévő kódok egymás 1-es kiegészítői. Ez a tulajdonság felhasználható hibajelzésre.

A ***BCD-kódot*** a ***számítástechnikában*** használják tízes számrendszerben történő szám-ábrázoláshoz, illetve számoláshoz. Az egy számjegyet leíró 4 bit-et ***dekád***-nak nevezzük. A 8 bites ***bájt***-ban ***két dekád*** írható, vagyis ***0 – 99*** decimális értéket tárolhat.

Pl. $\mathbf{38_d = 0011\ 1000_{BCD}}$

A mikroprocesszorok többségének utasításkészlete lehetővé teszi a BCD számokkal való számolást is. Erre a 2. féléves tananyagban térünk vissza.

Több bites kódokkal is leírhatók a decimális számjegyek. Itt csak az ***öt-bites*** un. ***Johnsson*** - kódot mutatjuk be.

0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	1
3	0	0	1	1	1
4	0	1	1	1	1
5	1	1	1	1	1
6	1	1	1	1	0
7	1	1	1	0	0
8	1	1	0	0	0
9	1	0	0	0	0
0	0	0	0	0	0

Az öt bit ***32*** érték kifejezését tenné lehetővé. Ebben az esetben csak ***tízhez*** rendeltünk ***értékes információt***. A kód tehát ***redundáns***. A további 22 érték ***hibajelzésre***, esetleg ***hibajavításra*** is használható.

A ***hibajelző***, és ***javító*** kódokkal – az ***információ-elmélet*** egy külön ága - a ***kódolás-elmélet*** foglalkozik részletesen. Ide tartoznak a különböző ***tömörítési***, ***titkosítási*** és ***visszafejtési*** stb. eljárások kidolgozása, algoritmizálása.

- ***Aritmetikai műveletek algoritmusai***

A megismert számábrázolási formák közül a mikroprocesszoros rendszerekben (mikrogép - ekben) általában a ***bináris kettes komplemens*** - ű változatot használják. Miután az aritmetikai műveletek az összeadás, és a kivonás műveleteire vezethető vissza, ezért itt egy példán keresztül vizsgáljuk meg e két művelet elvégzésének szabályait.

Vegyük a 96, és a 43 abszolút-értékű számok közötti műveleteket. Mind az összeadásnál, mind pedig a kivonásnál négy-négy műveletet kell elvégeznünk.

Az adott számok bináris kettes komplemens –ü értékei:

96	**0 ⁝ 1 1 0 0 0 0 0**		**- 96**	**1 ⁝ 0 1 0 0 0 0 0**
43	**0 ⁝ 0 1 0 1 0 1 1**		**-43**	**1 ⁝ 1 0 1 0 1 0 1**

Szaggatott vonallal az előjegy – bitet határoltuk el.

Az elvégzendő műveleteket láthatók az alábbiakban. Mindkét műveletnél helyértékenként - a legkisebb helyérték kivételével – három számjegyet (bit -et) adunk össze, illetve vonunk ki. Ezek a két szám ***azonos helyértékű számjegyei*** (bit -jei), illetve az előző helyértéken keletkező ***átvitel*** (***Cy*** Carry), illetve ***áthozat*** (***Bw*** Borrow) bitek. Az utóbbi értékeket a negyedik sorba - egy kissé eltolva - írtuk, jelezve ezzel a helyérték-váltást.

A ***kettes komplemens***-ű ábrázolású számok esetében ***mindig*** a ***kijelölt*** műveletet – az összeadást, vagy a kivonást – kell végezni úgy, hogy az ***előjegy bitet*** is ***számbitként*** kezeljük. A példában félkövér számmal jelöltük az ***utolsó számjegynél***, és az ***előjegy bitnél*** keletkezett ***átvitel / áthozat*** biteket.

Összeadás

96	0	1	1	0	0	0	0	0
+ 43	0	0	1	0	1	0	1	1
139	1	0	0	0	1	0	1	1
0	***1***	1	0	0	0	0	0	

96	0	1	1	0	0	0	0	0
+ -43	1	1	0	1	0	1	0	1
53	0	0	1	1	0	1	0	1
1	***1***	0	0	0	0	0	0	

-96	1	0	1	0	0	0	0	0
+ 43	0	0	1	0	1	0	1	1
-53	1	1	0	0	1	0	1	1
0	***0***	1	0	0	0	0	0	

-96	1	0	1	0	0	0	0	0
+ -43	1	1	0	1	0	1	0	1
-139	0	1	1	1	0	1	0	1
1	***0***	0	0	0	0	0	0	

Kivonás

96	0	1	1	0	0	0	0	0
- 43	0	0	1	0	1	0	1	1
53	0	0	1	1	0	1	0	1
0	***0***	1	1	1	1	1	1	

96	0	1	1	0	0	0	0	0
- -43	1	1	0	1	0	1	0	1
139	1	0	0	0	1	0	1	1
1	***0***	0	1	1	1	1	1	

-96	1	0	1	0	0	0	0	0
- 43	0	0	1	0	1	0	1	1
-139	0	1	1	1	0	1	0	1
0	***1***	1	1	1	1	1	1	

-96	1	0	1	0	0	0	0	0
- -43	1	1	0	1	0	1	0	1
-53	1	1	0	0	1	0	1	1
1	***1***	0	1	1	1	1	1	

A példák elvégzése után megállapíthatjuk a szabályokat:

- Mindkét műveletnél az ***eredményt*** is ***kettes komplemens***-ű formában kapjuk. A ***pozitív*** szám ***előjegy*** -e ***0***, és a számrész ***abszolút*** értékű, míg a ***negatív*** szám ***előjegy*** -e ***1***, és a számrész a szám ***kettes komplemens*** -e.
- ***Hibátlan*** eredményt kapunk, ha a két utolsó ***átvitel/áthozat*** bit (az utolsó számjegynél, illetve az előjegynél) 00, vagy 11.
- ***Hibás*** az eredmény, ha csak az egyik helyen keletkezik ***átvitel/áthozat*** bit. Ilyenkor ***aritmetikai túlcsordulás*** van. Ez azt jelenti, hogy a keletkezett eredmény nem fér el a számrésznek fenntartott helyen, (kicsi a kapacitás) vagyis az eredmény nagyobb, mint a 7 bittel megadható legnagyobb érték. A hiba a tároló-hely ***kapacitásának növelésével*** küszöbölhető ki.
- A két túlcsordulás-bit ***XOR*** (moduló 2) műveletének eredménye az un. ***Overflow*** –bit (***OF***), amit ***aritmetikai túlcsordulás*** bitnek is neveznek. Minden mikroprocesszor un. ***Status***, vagy ***Flag*** bitjei között szerepel az OF bit.

A ***kettes komplemens*** –ű számábrázolás előnye tehát az, hogy ***gyorsabb*** a művelet-végzés. A negatív számok abszolút értékre való konvertálását, illetve fordítva csak az adat ***kiíratásánál***, illetve ***bevitelénél*** kell elvégezni. A műveletek a gépen belül gyorsabban hajthatók végre.

2.1. Példák

1. A felsorolt decimális számokat számítsa át bináris, oktális, illetve hexadecimális alakra!

126, 578, 792, 1514, 2810

2. A felsorolt hexadecimális számokat számítsa át bináris, oktális, illetve decimális alakra!

1A5H, 0E27H, 83FH, 0F32H, 0EF1AH

3. A felsorolt bináris számokat számítsa át hexadecimális, bináris, oktális, illetve decimális alakra!

100110B, 1101100B, 1011011B, 111000110B, 1010110011B

4. A felsorolt decimális számokat számítsa át lebegőpontos bináris, illetve hexadecimális alakra!

126, 578, 792, 1514, 2810

5. Számítsa át a megadott Z_A és Z_B decimális számokat kettes-komplemenses ábrázolású bináris számokká

Z_A : 36, -54, 156, -117

Z_B : -48, 182, -96, 281

6. Végezze el az 5. feladatban adott Z_A és Z_B számokkal – bináris kettes komplemenses alakban – a következő műveleteket:

$$Z_E = Z_A + Z_B$$

$$Z_E = Z_A - Z_B$$

$$Z_E = Z_B + Z_A$$

$$Z_E = Z_B + Z_A$$

2.2. Ellenörző kérdések

1. Milyen számábrázolási formákat használnak?
2. Hogyan kell átszámítani a decimális alakú számot bináris alakra?
3. Hogyan kell átszámítani a bináris alakú számot decimális alakra?
4. Hogyan kell átszámítani a decimális alakú számot hexadecimális alakra?
5. Hogyan kell átszámítani a bináris alakú számot hexadecimális alakra?
6. Hogyan kell átszámítani a decimális alakú számot oktális alakra?
7. Hogyan épül fel a lebegőpontosan ábrázolt szám?
8. Mit nevezünk komplemsű ábrázolásnak?
9. Hogyan kell összeadást, vagy kivonást végezni kettes-komplemensű ábrázolású számokkal?
10. Milyen BCD kodolásokat használnak?
11. Mi a Jhonson kód?

3. DIGITÁLIS INTEGRÁLT ÁRAMKÖRÖK

Az előző fejezetben tárgyaltuk meg az alapvető logikai ismereteket. Ezek alapján sajátíthatjuk el a digitális műveletvégzés és jeltovábbítás módszereit, ill. az automatikus irányítóberendezések működésének elvét. Ebben a fejezetben a logikai műveleteket megvalósító alapvető ***logikai áramköröket*** tárgyaljuk.

Részletesen tárgyaljuk az ***integrálási*** technológiával készült ***kapu-***, és az elemi ***tároló áramkörök*** (flip-flop) fizikai működését, logikai funkcióját és ezen elemi egységek egymáshoz csatlakoztatásának lehetőségeit, feltételeit. Bővebben foglalkozunk a ***TTL,*** és a ***CMOS rendszerű*** áramkörökkel. A ***funkcionális áramkör*** – kombinációs, és sorrendi – mindegyike kapuáramkörökből épül fel, ezért azoknak csak a legjellemzőbb ismérveit foglaljuk össze.

A fejezet második részében – a megismert - integrált áramkörök néhány jellemző ***alkalmazásával*** foglalkozunk.

3.1. Logikai áramkörök

A megismert logikai műveletek (ÉS, VAGY, NEM) technikai megvalósítása ma szinte kizárólag a ***félvezető*** alapú ***digitális áramkörökkel*** történik. Ezek részletesebb megismerése előtt célszerű a technikai fejlődést röviden összefoglalni.

Az elektronikus logikai áramköröket az alkalmazott áramköri elemek és az előállítási technológia alapján különböző generációkba soroljuk. Ez a besorolás egyúttal fejlődés-történeti csoportosítás is.

- Az ***első generációs*** áramkörök ***diszkrét passzív*** áramköri elemekből (ellenállások, kondenzátorok stb.), valamint ***elektroncsövekből*** épültek fel. Felhasználásukra elsősorban a negyvenes évek közepétől az ötvenes évek közepéig terjedő időszakban került sor.

- A ***második generációs*** áramkörök ugyancsak ***diszkrét passzív*** áramköri elemeket tartalmaznak, de aktív elemeik már a ***tranzisztorok***. Ezek az áramkörök a hatvanas évek közepéig voltak egyeduralkodók. Az áramkörök gyártástechnológiájára az alkatrészek nyomtatott áramköri lapokra szerelése a jellemző. Az egyszerű logikai funkciókat (ÉS, VAGY, NEM, TÁROLÁS) ellátó áramkörök egységes felépítésű - sorozatban gyártott - kártyákon (pl. EDS - kártyák) vagy térbeli elrendezésű, műgyantával kiöntött kockákban (pl. Terta kockák) kerültek forgalomba. Ezekből építették a különböző irányítóberendezéseket, mint pl. a forgalomirányító lámpák automatikus vezérléseit.

- A ***harmadik generációs*** áramkörök csoportját alkotják a kis és közepes bonyolultságú ***digitális*** (logikai) ***integrált áramkörök*** (IC-Integrated Circuit) (logikai kapuk, flip-flop -ok, regiszterek, számlálók stb.) alkalmazásával épített rendszerek. A rendszerépítés IC-kel is nyomtatott lapon történik. Ez a technika a hetvenes években vált egyeduralkodóvá, és napjainkban is alkalmazzuk.

- A ***negyedik generációs*** áramkörök közé a ***nagy bonyolultságú*** integrált áramkörök (a ***mikroprocesszor***, kiegészítő ***rendszerelemek***, ***memóriák*** stb.) tartoznak. A nagyfokú integrálás révén egyetlen tokban teljes rendszertechnikai egység (pl. központi egység) állítható elő. Néhány ilyen elem segítségével építhető „intelligens” berendezés (mikroszámítógép, irányítástechnikai berendezés stb.).

A logikai áramkörök és egységek működésének megértéséhez elengedhetetlenül szükséges a diszkrét elemes félvezetős (második generációs), valamint a kis és közepes bonyolultságú integrált áramkörök (harmadik generációs) ismerete.

A digitális hálózatokban az alapáramkörök végzik a logikai ÉS, VAGY, NEM (esetleg ezek kombinációjából álló) műveleteket, a tárolást, valamint a hálózat működését kisegítő, nem logikai funkciókat (időzítés, jelgenerálás, jelformálás stb.).

Ezek alapján a következő logikai alapáramköröket különbözetjük meg:

- ***kapu*** áramkörök,
- ***tároló*** áramkörök (flip-f1opok),
- ***jelgenerátorok,***
- ***késleltető*** áramkörök**,**
- ***jelformáló, illesztő*** áramkörök**.**

Az áramkörök elemzésénél használt gondolatmenet:

- az áramkör ***működésének,***
- ***logikai*** funkciójának**,**
- ***csatlakoztatási*** feltételeinek

ismertetése.

A legfontosabb fogalmak közül, mint a

- villamos ***jelhordozók,***
- ***terhelési*** viszony**,**
- ***jelterjedési*** idő

meghatározását előzetesen tárgyaljuk.

Külön kell még néhány mondatot szánni a ***passzív***, ill. ***aktív*** áramköri elem fogalmának.

- A ***passzív*** elemek - mint pl. az ellenállás, kondenzátor, dióda - csak villamos ***teljesítményt fogyasztanak***.
- Az ***aktív*** áramköri elemek - elektroncső, tranzisztor - villamos ***teljesítmény átalakítására*** is felhasználhatók. Önmaguk villamos energiát nem állítanak elő. A teljesítmény átalakításhoz (pl. erősítéshez) szükséges energiát a ***tápforrásból*** nyerik.

3.2. A logikai érték villamos jelhordozói

A különböző villamos áramkörökben az ***információt*** villamos jel, ***feszültség*** vagy ***áram*** hordozza. Amikor folytonosan változó információt - pl. hangerő - a villamos jel különböző jellemzője (pl. nagysága) jelenti meg, akkor ***analóg*** jelátvitelről beszélünk. A ***digitális technikában*** - mint ahogyan ezt már megismerték - az elemi információnak csak ***két értéke*** lehet (IGAZ, HAMIS).

Amikor a logikai információhordozó az ***áram***, akkor az egyik logikai értékhez rendeljük, hogy ***folyik*** áram, a másikhoz, pedig azt hogy ***nem folyik*** áram. Ez a jelhordozó-választás elsősorban az elektromechanikus relékkel megvalósított un. ***relé-logikai*** áramkörökben szokásos.

A félvezetős logikai áramkörökben (tananyagunk témája) a logikai értéket hordozó villamos jellemző leggyakrabban a ***villamos feszültség***. Mindkét logikai értékhez -

egymástól jól elválasztva - egy-egy ***feszültségtartományt*** rendelünk. A logikai értékhez rendelt feszültségértékeket logikai feszültségszinteknek vagy rövidebben ***logikai szinteknek*** nevezzük.

Az egyes logikai értékekhez rendelt szintek egy-egy ***feszültségsávot*** jelentenek. A sávon belüli bármely feszültségérték ugyanazon elemi információt (logikai értéket) jelenti. Ez biztosítja azt, hogy az áramköri elemek tényleges értékének különbözősége (szórása) és a különböző környezeti feltételek (hőmérséklet, terhelés stb.) változása az információtartalmat nem módosítja. Ezért is a digitális jelfeldolgozás a külső zavarójelekre kevésbé érzékeny, vagyis nagyobb ***zavarvédettségű*** az analóg módszernél.

A logikai **IGAZ** értékhez rendelt szintet **1**, vagy **IGEN** szintnek nevezik. A logikai **HAMIS** értékhez rendelt szint, pedig a **0** vagy **NEM** szint.

Az áramköri leírásokban a ***pozitívabb*** logikai feszültségszintet ***magas*** vagy **H** (High) szintnek, a ***negatívabb*** feszültségszintet, pedig ***alacsony*** vagy **L** (Low) szintnek is szokás nevezni.

A választott feszültségszintek egymáshoz viszonyított elhelyezkedése szerint kétféle ***logikai szintrendszerről*** beszélünk.

A szintek egymáshoz való viszonya szerint megkülönböztetünk:

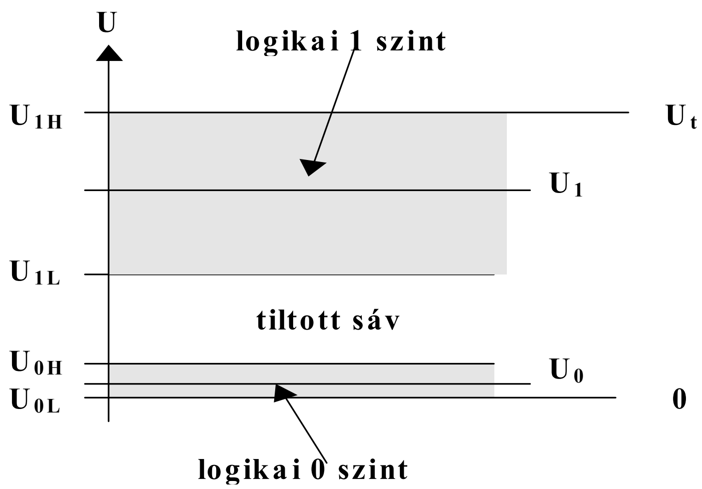
- ***pozitív*** és
- ***negatív***

logikai szintrendszert.

Pozitív logikai szintrendszerről akkor beszélünk, ha az **IGAZ** értékhez rendeljük a ***pozitívabb*** feszültségsávot. A **HAMIS** értéknek tehát a ***negatívabb*** feszültségsáv felel meg.

A ***negatív logikai*** szintrendszerben a ***negatívabb*** feszültségsávhoz (szinthez) tartozik az **IGAZ** érték és a ***pozitívabb*** szinthez, rendeljük a **HAMIS** értéket.

A 25.ábra szemlélteti a ***pozitív logikai szintrendszer*** egy lehetséges elhelyezését a függőleges feszültségtengely mentén.



25. ábra

A technikai gyakorlatban az egyik szint mindig az áramköri rendszer ***közös* 0** potenciálú értékét is magában foglaló ***feszültségsáv***.

A szintek tűrésének nagysága alapján megkülönböztetünk:

- ***szabad*** és
- ***kötött*** szintű

logikai áramköri rendszereket.

Szabad szintű a logikai áramköri rendszer, ha legalább az egyik feszültségszint széles határok között változhat. Általában ez a tűrés a tápfeszültség felével, egyharmadával egyező nagyságú.

Kötött szintű a logikai rendszer, ha mind az 1, mind pedig a 0 értékhez tartozó szint tűrése kicsi. Ennek értéke rendszerint a nyitott félvezető elemen (dióda, tranzisztor) eső feszültség két-háromszorosa.

A továbbiakban sorra kerülő áramköri elemzéseknél a logikai szintek és tűrések szélső értékeinek jelölésére a következőket fogjuk használni.

$\mathbf{U_1}$ - az **1** szint **névleges** értéke,

$\mathbf{U_{1H}}$ - az **1** szint **nagyobb** abszolút értékű szélső értéke,

$\mathbf{U_{1L}}$ - az **1** szint **kisebb** abszolút értékű szélső értéke,

$\mathbf{U_0}$ - a **0** szint **névleges** értéke,

$\mathbf{U_{0H}}$ - a **0** szint **nagyobb** abszolút értékű szélső értéke,

$\mathbf{U_{0L}}$ - a **0** szint **kisebb** abszolút értékű szélső értéke.

Az előző jelöléseket a 25.ábrán is feltüntettük.

3.3. Terhelési viszony

Összetett logikai hálózatokban egy áramkör - a logikai feladat függvényében - több áramkört is vezérelhet. Ezért ilyen esetekben azt is meg kell vizsgálni, hogy egy áramkör kimenetéhez hány további áramkör csatlakoztatható anélkül, hogy a megengedettnél nagyobb szinteltolódás vagy esetleg az áramköri elem tönkremenetele következne be.

Az egységesített áramkörrendszereknél a különböző funkciójú áramkörök legtöbb bemenete hasonló felépítésű, s így a bemeneti áram is azonos. Ezt szokták választani ***egységterhelés****nek* (terhelési egységnek). A ***terhelési viszonyban*** azt adják meg, hogy az egységterhelésnek hányszorosa az adott csatlakoztatásnál megengedett áram. Ez tehát egy ***relatív érték***, egy nevezetlen szám.

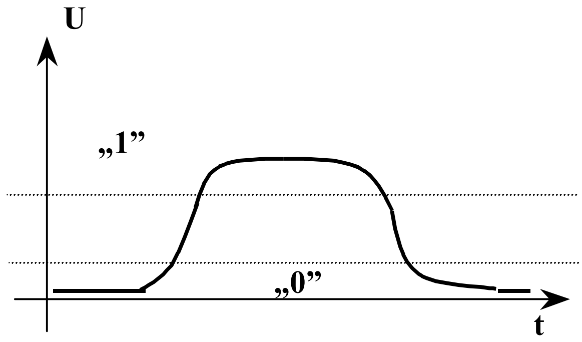
Bemeneti terhelési szám (fan-in) az áramkör bemeneti áramának és az egységterhelésnek a hányadosa.

Kimeneti terhelési szám (fan-out) az áramkör megengedett kimeneti áramának és az egységterhelésnek a hányadosa .A fan-out tehát megadja azt, hogy az áramkör hány áramkört tud vezérelni.

3.4. Jelterjedési idő

Bármely tetszőleges áramkör bemenetére jutó jelváltozóst a kimeneti jel változása mindig valamilyen késleltetéssel követi. A digitális áramkörökben a logikai információt hordozó villamos jel látszólag ugrásszerűen változik a 0-hoz és az 1-hez tartozó feszültségérték között. Valójában ez a szintváltás nem következhet be ***nulla idő*** alatt, mert ehhez ***végtelen*** nagy ***energia*** lenne szükséges.

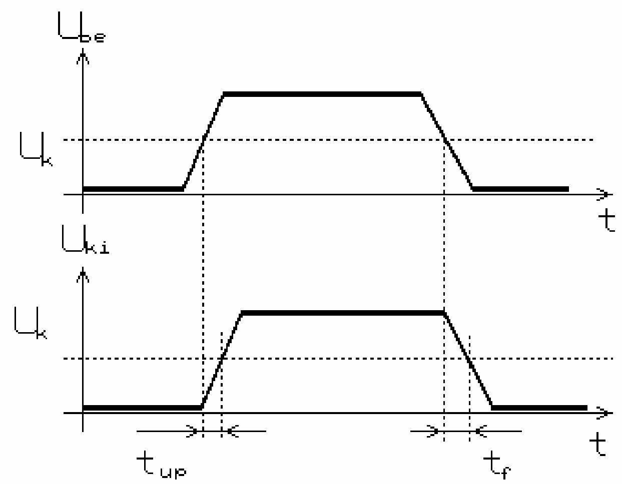
A tényleges változás az idő függvényében ***exponenciális,*** illetve ***logaritmikus*** jellegű. Ezt szemléletesen láthatjuk egy oszcilloszkópon is, ha a vízszintes eltérítés frekvenciáját kellően megnöveljük.



26. ábra ábra

A 26.ábra szemlélteti, hogy csak az „1”, illetve a „0” szinteken belül ***nem-lineárisan*** változik a jel, viszont ez nem jelent logikai értékváltozást. A változás a tiltott sávon belül viszont ***lineárisnak*** tekinthető. A négyszögjelet tehát egy ***trapézzal*** is helyettesíthetjük, és ekkor sem térünk el lényegesen a tényleges viszonyoktól-

A 27. ábra szemlélteti egy négyszöghullámú bemeneti jellel vezérelt digitális áramkör be-, és kimeneti jeleinek időfüggvényeit. Az ábrán $\mathbf{U_k}$ – val jelölt fezsültség, az un. ***komparálási*** (billenési) szint, amely általában a tiltott sáv közepére esik. Az elnevezés arra utal, hogy egy áramkör kimenetén csak akkor indul meg a jelváltozás, ha a bemeneti jel már túllépi az $\mathbf{U_k}$ szintet.



27. ábra

Egy tényleges áramkör mindig késleltetve válaszol, a bementi jelre. A késést két jel $\mathbf{U_k}$ komparálási (billenési) feszültségei között kell mérni. Rendszerint a két különböző irányú jelváltások ideje nem egyforma. A **0 -> 1** irányú változós késleltetését $\mathbf{t_u}$ – val (time-up ”emelkedési idő”), az **1 -> 0** váltás késleltetését, pedig t_f **–**el (time-fall ”esési idő”) jelöljük.

Az áramör ***átlagos jelkésleltetési*** idejét t_{pd} (propagation delay) a kétirányú változós késleltetésének számtani átlagaként számoljuk ki:

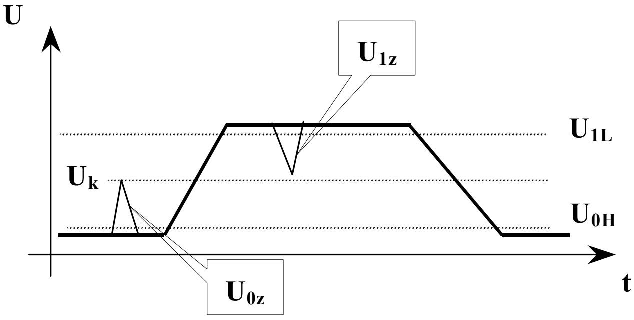
$$t_{pd} = \frac{t_u + t_f}{2}$$

3.5. Zavarvédettség

Zavar, vagy másképpen ***zavaró jel***, az áramkör jelvezetékein keletkező rendellenes ***feszültségimpulzus*** (U_z). Leggyakoribb az áramkör környezetében fellépő jelentősebb ***elektromágneses*** térerő-, ***áram***-változásból ***induktív*** csatolás révén kerül a jelvezetékekre. Zavaró feszültség – főleg ***nagyfrekvenciás*** – juthat ***kapacitív*** csatolás révén is az áramkörbe. A zavarok legnagyobb hányada a ***bemeneti*** vezetékeken jut be az áramkörbe. Az ipari környezetben működő hálózatokat különösen sok zavaró jel éri, amelyeket már, az áramkör tervezéskor figyelembe kell venni.

Egy áramkör ***zavarvédettségén*** – immunitásán - azt a (U_{zv}) ***feszültségértéket*** értjük, amely az áramkör bemeneti jelére ***szuperponálódva***, az áramkör ***kimenetén*** még ***nem okoz*** logikai ***szintváltást***.

Az elözőekben tárgyalt jelalakok alapján megállapíthatjuk a megengedhető legnagyobb zavarójel, vagy a ***zavarvédettség*** értékét is. A 28.ábrán feltüntettük a logikai szintek ***garantált*** szélső értékeit (U_{1L}, U_{0H}), valamint az U_k ***komparálási*** feszültséget.



28. ábra

Az ábrázolt jelnél mind a 0, mind pedig az 1 szint a megengedett szinttűrés határán van. Ugyanakkor mindkét szintre ráülő zavar-jelek (U_{0z}, U_{1z}) is láthatók. Bármelyik ***zavar*** csak akkor jelenik meg az áramkör ***kimeneti*** jelében is, ha a ***bemeneti*** jel ***túllépi*** a ***komparálási*** szintet. Az ábrázolt zavarjelek éppen a határhelyzetű értékek, vagyis ekkora zavaró feszültség ellen védett az áramkör.

A szemléltetett viszonyok alapján meghatározhatjuk az áramkör ***zavarvédettségét*** mindkét logikai szintre.

0 szintnél $U_{0zv} = U_k - U_{0H}$

1 szintnél $U_{1zv} = U_{1L} - U_k$

Az áramköröknél a komparálási érték függ a hőmérséklettől, ezért a zavarvédettség is változik a hőmérséklet fváltozás függvényében. A konkrét áramköri készleteknél ezeket a jellemzőket a katalógusok megadják.

3.6. Digitális integrált áramkörök

Az ***elektronikai ipar*** az elmúlt négy évtized alatt rendkívül gyors ütemben fejlődött. E fejlődés során az elektronikus berendezések és rendszerek bonyolultsága, és ezzel együtt mérete is rohamosan növekedni kezdett. A mind kisebb és mind megbízhatóbb elektronikus berendezések készítésére irányuló kutatásokat a hadiipar szükségletei indították el a második világháború idején. A háború befejezése utáni rövid visszaesést hamarosan megszüntette a tudományos és műszaki élet területén bekövetkezett fejlődés.

A ***miniatürizálást*** nagymértékben indokolta a világűrkutatás rohamos fejlődése. A berendezések bonyolultsága olyan mértékben nőtt, hogy a megbízhatóságot már nem is annyira az alkatrészek megbízhatósága, mint az összeköttetéseké határozta meg. A kialakult hálózatban nyilvánvalóvá vált, hogy a problémák megoldása (miniatürizálás, megbízhatóság, stb.) új ***technológiai módszereket*** kíván. Az új technológiai módszerek kidolgozása hozta létre az elektronika új ágát a ***mikroelektronikát*** és ezen belül az ***integrált áramkörök*** technikáját. Az integrált jelző arra utal, hogy az egy alaplemezen, azonos technológiai lépésekkel egyidejűleg létrehozott alkatrészekből álló áramkör nem bontható alkotóelemeire roncsolás nélkül.

A legkorszerűbb integrált áramkörök jelenleg az ún. ***monolit*** (félvezető alapú) integrált áramköri technikával készülnek. Ennek a lényege az, hogy a tranzisztorokat, diódákat, ellenállásokat, kondenzátorokat és az összekötő vezetékeket egyetlen szilicium kristályon alakítják ki, egymást követő technológiai lépések sorozatával.

A félvezető alapú integrált áramkörök bevezetésekor úgy tűnt, hogy a monolit technika főként ***digitális áramkörök*** realizálására alkalmas, elsősorban a nagy alkatrész-szórás miatt. A technológia finomításával és újszerű áramkör konstrukcióval azonban olyan tulajdonságokkal rendelkező ***analóg áramkörök is*** készíthetők, amelyek a diszkrét elemekből felépülő áramkörökhöz képest is kedvezőbbek.

Bár az integrált áramkörök fejlődését kezdetben főleg a hadiipar és az űrkutatás serkentette, a polgári életben is élvezhetőek az eredményei. A ma technikaja, a hétköznapi élet minden eszköze az integrált áramkörökre épül.

A napjainkban használt integrált áramkörök bonyolultságuk és alkatrészeik száma szerint a következő csoportokra oszthatók:

SSI (Small-Scale-Integration): ***alacsony*** fokú integrált áramkörök; egyszerűbb alapáramköröket tartalmaznak. Az egy tokban levő alkatrészek száma: **50...100**.

MSI (Medium-SI): ***közepes*** integráltságú áramkörök; bonyolultabb funkciókat elvégző egységeket tartalmaznak. Az egy tokban lévő alkatrészek száma: **500...1000**.

LSI (Large-SI): ***magas*** integráltságú áramkörök; tokonként egy-egy komplett rendszert alkotnak. Az alkatrészek száma: **1000...10 000**.

ELSI (Extra-LSI): az előbbinél több alkatrészt tartalmaznak, és bonyolultabb rendszereket valósítanak meg.

Az aktív logikai kapuk legkorszerűbb változatai a digitális integrált áramkörök választékaiban szerepelnek. Az egyetlen kristályban - integrálási technológiával - előállított áramkörök az **IC**-k (Integrated Circuit). A diszkrét elemes digitális áramkörökkel szemben sok előnnyel rendelkeznek. Jelentős a miniatűr méret, a sokkal nagyobb működési sebesség, kis disszipációs teljesítmény, valamint a nagy sorozatban való gazdaságos előállítás, tehát az alacsony ár.

A különböző integrált áramköri családok alapelemei az **ÉS-NEM** (**NAND**) vagy a **VAGY-NEM** (**NOR**) kapuk. Ezek mellett megtalálhatók a háromműveletes alaplapúk

(ÉS-VAGY-NEM), a tárolóelemek (***flip-flop*** -ok),valamint a bonyolultabb logikai feladatokra használható ***funkcionális áramkörök*** (dekódolók, multiplexerek, számlálók, regiszterek stb.). A különböző felépítésű integrált áramköri családok közül a **TTL** (Tranzisztor –Tranzisztor - Logika) és a **CMOS** (Complement Metal- Oxid Semiconductor) rendszerű integrált áramkörökkel foglalkozunk. Leginkább ezek terjedtek el.

A **TTL** rendszert a **TEXAS INSTRUMENTS** cég fejlesztette ki az **SN74...** jelű sorozatával. Ma már több országban is gyártják az eredeti sorozattal csereszabatos (kompatibilis) TTL alapáramköröket. A CMOS családokat is számos világcég (pl. RCA) gyártja ma már. Létezik olyan sorozat is a CMOS áramkörök között, amely a TTL áramkörökkel funkció és láb-kompatibilis. Ezek típus-jele: **SN74C...**,amelyben csak a C betű utal a technológiai kivitelre. A többi szám azonos a megfelelő TTL áramkörével.

- ***TTL rendszerű kapuk***

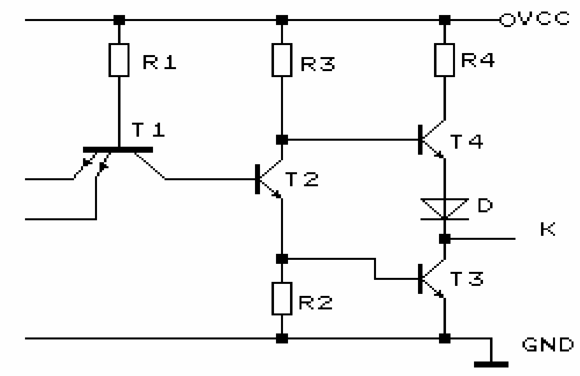
A TTL rendszerű integrált áramköri család pozitív logikai szinttel működik. A legfontosabb feszültség adatok a következők:

	Névleges	**Minimum**	**Maximum**
Tápfeszültség (Ucc)	+5 V	+4,75 V	+5,5 V
Bemeneti 1 szint (U_{iH}	+3,4 V	+2 V	+5,5 V
Bemeneti 0 szint (U_{iL})	+0,2 V	-1,5 V	+0,8 V
Kimeneti 1 szint (U_{OH})	+3,4 V	+2,4 V	+5,5 V
Kimeneti 0 szint (U_{OL})	+0,2 V	-0,8 V	+0,4 V

A ***normál*** TTL sorozat alap kapuja a NAND (NEM-ÉS) kapu. A sorozatban kettő, három, négy és nyolc bemenetű NAND kapukat készítenek. A kapuk mind különböző kialakítású - tokozásban kerülnek a kereskedelembe. A leggyakoribb változat az un. ***duál in line*** tokozás, amely műanyag burkolatú, két oldalt elhelyezkedő kivezetései (lábak) van. Egy ilyen tokban – legtöbbször - több azonos kapu van.

A két-bemenetű NAND kapuból négy db, a három-bemenetűből három db, a négy-bemenetűből kettő db, és a nyolcbemenetűből pedig egy db van a tokban. Mindezek a kapuk csak a bemenetszámban térnek el. Ezért a továbbiakban csak a két-bemenetű NAND kapu működését elemezzük.

A 29.ábrán látható a két-bemenetű **TTL NAND** kapu kapcsolási vázlata.



29. ábra

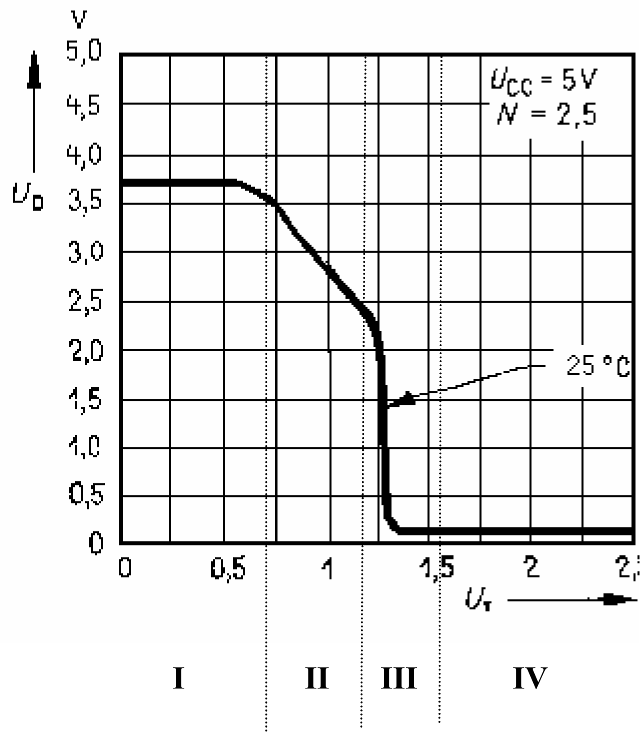
Az áramkör három fő egységre tagolható. Ezek:

- ***több emitter*** -es (múlti emitter) ***bemenet*** (T1 tranzisztor),
- ***vezérlő*** fokozat (T2 tranzisztor);
- ***teljesítményillesztő*** kimenet (T3,T4 tranzisztorok, totem-pole).

A T1 múlti emitter -es tranzisztor az **ÉS** kapu, míg a vezérlő és az ellenütemű (totem-pole) kimeneti fokozat feladata együtt az invertálás, valamint a szint-, és teljesítményillesztés.

Az áramkör elemzéséhez bemutatjuk a működést szemléltető, un. ***átviteli*** (transzfer) ***karakterisztikát*** is. Ez a karakterisztika koordinátarendszerben ábrázolja a K kimenet feszültsége (U_{ki}) és a kimeneti szintet meghatározó U_{be} vezérlőfeszültség közötti kapcsolatot. A NAND kapunál mindig a legalacsonyabb szintű bemenő feszültség szabja meg a kimeneti szintet.

A 104.ábrán látható a ***normál TTL*** rendszerű NAND kapu ***transzfer*** - átviteli – ***karakterisztikája***.



30. ábra

A vízszintes tengely mentén négy jellemző tartományt különböztethetünk meg. Ezeket római számokkal jelöltük.

Az **I.** szakaszban az áramkör legalább egyik, vagy mindkét bemenetén az U_{be} feszültség a **$0<U_{be}<0{,}7V$** feszültségtartományba esik. Ekkor a **T1** tranzisztor ***normál telitett*** üzemmódban van, mivel bázisa az **R1** ellenálláson keresztül az **Ucc** tápfeszültségre kapcsolódik. A tranzisztor kollektor-feszültsége a maradék feszültséggel (0,1 . . 0,2 V) pozitívabb az emitter feszültségénél. Ez az alacsony szint még ***zárva*** tartja a **T2** tranzisztort. A **T3** tranzisztor is zárt, mivel nem kap nyitóirányú bázisáramot. A **T4** tranzisztor az **R3** ellenálláson folyó bázisáram hatására ***vezet***. A ***kimeneti*** feszültség (U_{ki}) az **R4** ellenálláson, a nyitott **T4** tranzisztoron és a **D** szinttoló diódán keresztül ***magas*** pozitív feszültségű lesz, amely a logikai **1 szint**. Jellemző értéke terheletlenül **+3,6 V**.

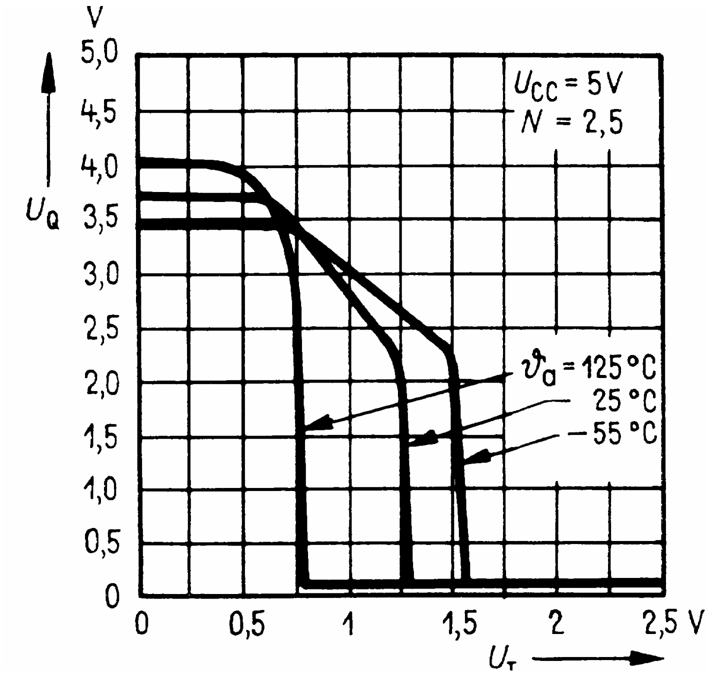
II. szakasz, amikor az ***alacsonyabb*** szintű bemeneti feszültség, a **0,7 < Ube < 1,2 V** tartományba kerül, akkor már a **T2** tranzisztor a lineáris tartományba kerül. A tranzisztor kollektorfeszültsége csökken, ezt a **T4** tranzisztor emitter-, és igy az áramkör kimeneti feszültsége is követi. A **T3** tranzisztor még ***zárt***, mivel a bemeneti feszültség nem elégséges két **pn** átmenet (T2 és T3 bázis - emitter dióda) nyitó irányú előfeszítéséhez.

A **III. szakasz** az un. ***billenési tartomány***. Amikor az alacsonyabb szintű bemenő feszültség eléri a **~ 1,4 V** -os értéket, a **T3** tranzisztor is ***lineáris*** tartományban kerül. Az áramkör ekkor ellenütemű erősítőként üzemel. Miután a feszültség-erősítése nagy (A_u>10) ezért kis bemenő-feszültség változás mellett nagy a kimenőfeszültség változása. A karakterisztika itt meredek.

A IV. szakasz, amikor **U_{be} > 1,5 V**. Ebben a szakaszban a **T2** és **T3** tranzisztor is telítésbe kerül. A kimenő-feszültség logkai 0 szintű, és értéke a telitett **T3** tranzisztor maradékfeszültsége (0,1... 0,2 V). Amikor a T2 válik ***telitetté***, akkor kollektorán kb. 0,8 ... 0,9 V a feszültség, ami egyúttal a T4 tranzisztor bázisfeszültsége is. Ez az érték az U_{ki}-nél csak ~ 0,7 V-al pozitívabb, ami nem elég a T4 tranzisztor és a D dióda nyitva tartásához, tehát a **T4 *lezár***. Az előzőekből lesz érthető a D ***szinttoló*** dióda szerepe. Megnövelte a T4 nyitásához szükséges bázisfeszültséget. Ez teszi biztonságossá annak lezárását is.

Ebben a működési szakaszban a T1 múlti-emitteres tranzisztor kollektor-feszültségét a két nyitott pn átmenet (T2,T3) 1,4 V értéknél megfogja. A tranzisztor bázisfeszültsége sem emelkedik 2,1 V fölé. Ezért a bemeneti feszültségek további növelésekor a bázis-emitter diódák lezárnak s a tranzisztor ***inverz telitett*** üzemmódba kerül. Az inverz üzemmódban az emitter és kollektor szerepe felcserélődik. Ilyenkor a bemeneteken nagyon kis áram fog folyni.

Az áramkörök jellemzői a hőmérséklet függvényében változnak, amely az átviteli karakterisztika alapján követhető. A 31.ábrán látható karakterisztikák különböző hőmérséklethez tartoznak.



31. ábra

➢ *Bemeneti áramok*

Az áramkörök bemenő árama (I_{be}) különböző szintű vezérlésnél eltérö. A **0** szintnél a tipikus áramérték I_{be0} **= 1 mA**, de a legkedvezőtlenebb esetben is legfeljebb **1,6 mA**. Az **1** szintű vezérlésnél - az inverz üzemmódban működő tranzisztor emitter-árama - I_{be1} = **5 µA** (határérték 40 µA).

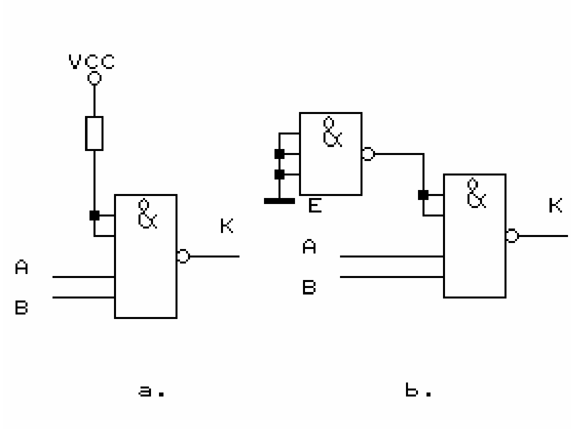
Ezeket az áramértékeket tekintjük az áramkörkészlet terhelési egységének, amelyek alapján számolhatók a terhelési számok.

A kapuk terhelhetőségét a terhelési egységre vonatkoztatott terhelési szám, a fan-out adja meg. A tipikus fan-out érték 10. Ez abszolút terhelésben - 0 szintű kimenetnél - 16 mA, 1 szintű kimenetnél 400 µA határterhelést ad. Az áramkörcsalád újabb típusainál, 800 µA a határérték, amely 20 egységterhelésnek felel meg.

A NAND kapuk vezérlésekor a kimeneti feszültség 0 -> 1, ill. 1 -> 0 irányú szintváltozósa különböző idejű késleltetéssel következik be. A lefutási késés $\mathbf{t_f}$ = 7-8 ns, a felfutási késés, pedig $\mathbf{t_u}$ = 11 -13 ns. Az átlagos jelterjedési idő $\mathbf{t_{pd}}$ **= 10 ns**. Az átkapcsolási idők függenek a terhelés nagyságától, jellegétől, a tápfeszültségtől, valamint a hőmérséklettől. A tápfeszültség és a hőmérsékletfüggés általában elhanyagolható. A terhelésváltozós késleltető hatását - az áramkörök felhasználásakor - már figyelembe kell venni. A terhelés hatását a katalógusokban adják meg.

A késleltetéseket még növeli az is, ha a bemenetek közül egyet vagy többet nem kötünk sehova (ez a működést logikailag nem változtatja meg). A bemeneti T1 jelű múlti emitteres tranzisztor árammentes bemeneteinek kapacitása 0,5 ... 1,5 pF értékű, ami üresen hagyott bemenetenként 1 ns - al növeli a késleltetési időt.

A járulékos késleltetés megszűnik, ha a fel nem használt bemeneteket egy vezérelt bemenettel kötjük össze. Ez a megoldás 1 szintű vezérlésnél növeli a bemenő áramot s így csak a meghajtó áramkör terhelhetőségi határáig használható. Ezért előnyösek az 1 szintnél N = 20 terhelhetőségű kapuk. Ha a terhelési viszonyok nem engedik meg a bemenetek összekötését, akkor a 32. ábra szerint kell a fel nem használt bemeneteket R = 1 ... 5 kohm értékű ellenállással a tápfeszültségre (a.ábra) vagy egy szabad NAND kapu (inverter) 1 szintű kimenetéhez csatlakoztatni (b.ábra).

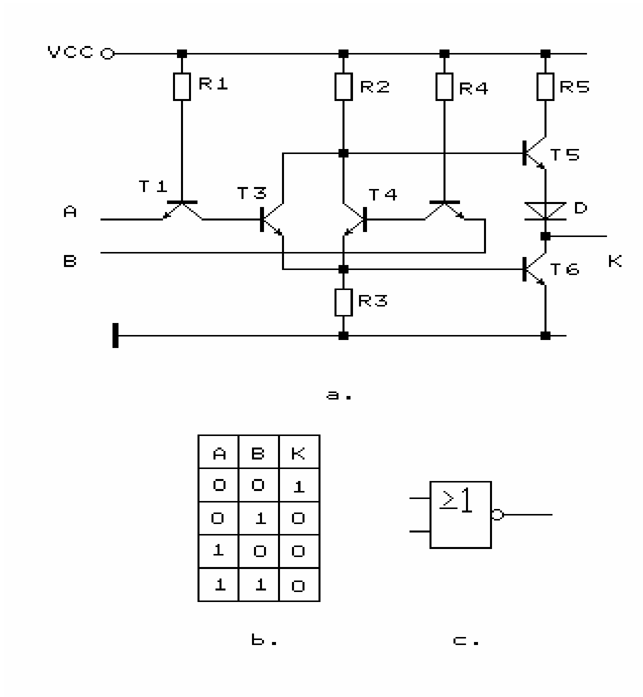


32. ábra

Késleltetés-növekedés e megoldásoknál is van, de értéke bemenetenként csak 0,5 ns.

A logkai kapuk tápáram felvétele (Icc) is változik a különböző vezérlési állapotokban. Kimeneti **0** szintnél a kapu áramfelvétele ~**3 mA**, az **1** szintnél pedig ~**1 mA**. (Ezek az értékek terhelet1enül érvényesek.) A 0 -> 1 átkapcsolások során az áramfelvétel átmenetileg megnövekszik, mert ilyenkor az ellenütemű kimenet mindkét tranzisztora (T3 és T4) rövid ideig együtt vezet.

Az SN sorozatban - az eddigiekben tárgyalt NAND kapuk mellett - NEM-VAGY (NOR) kapu csak két-bemenetű változatban van. A kapu kapcsolási vázlatát mutatja az 33.ábra.



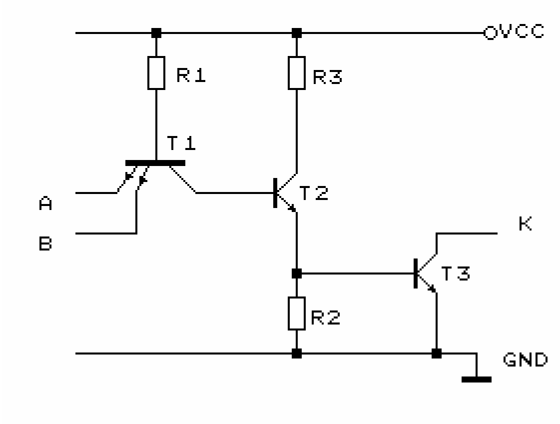
A	B	K
0	0	1
0	1	0
1	0	0
1	1	0

33. ábra

Az áramkör működése a következő. A kimenet logikai 1 szintű, ha a kimenő (totem-pole) fokozatot meghajtó **T3**, és**T4** tranzisztorok zártak. Ekkor a **T5** tranzisztor az **R2** ellenálláson keresztül telítésbe kerül, s ugyanakkor a **T6** tranzisztor lezár. A T3 és T4 tranzisztorok akkor zárnak, ha mind az **A**, mind a **B** bemeneten logikai **0 szint** van. Ha a bemenetek valamelyike, vagy mindkettő **1 szintű** vezérlést kap, akkor a bemeneti tranzisztor (T1 vagy T2, vagy mindkettő) inverz üzemmódban működik és a meghajtó tranzisztorok (T3,T4) közül az egyik vagy mindkettő nyit. A három kombináció mindegyikében a kimenet **T6** tranzisztora nyit s így kollektorán - a K kimeneten - logkai 0 szint lesz. A fenti működést írja le a b.ábra szerinti igazságtáblázat, amely a NOR függvénykapcsolatot adja. A kapu szimbolikus jele a c.ábra szerinti.

Az SN áramkör családban csak inverterek -et tartalmazó tokok is készülnek (6 db inverter 1 tokban). Ezek tulajdonképpen egy-bemenetű NAND kapunak tekinthetőek. Az inverterek működése a már leírték alapján elemezhető.

Az áramkörcsalád speciális kapui a **nyitott kollektoros** (open-collector) változatok. Az ezekben levő kimenő fokozat egyetlen tranzisztor, amelynek szabadon hagyott kollektora van kivezetve. Ilyen kimenettel két-bemenetű NAND kapuk és inverterek készülnek. A két-bemenetű NAND kapcsolási vázlatát mutatja a 34.ábra. A T3 tranzisztor munka-ellenállását kívülről kell bekötni.



34. ábra

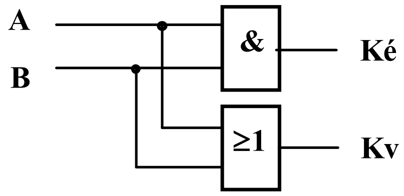
A nyitott kollektoros NAND kapukkal több szintű logikai függvény is megvalósítható. Az un. **huzalozott ÉS** kapcsolatot kapjuk, ha két vagy több nyitott kollektoros NAND kapukimeneteit közös R_T munkaellenállásra kapcsoljuk. Négy bemeneti változóra az áramköri kapcsolást az 14.a.ábra mutatja. A kimeneten csak akkor lehet logkai 1 szint, ha mindkét NAND kapu kimenete 1 szintű, vagyis a kimeneti tranzisztorok zártak. Ez az egyes kapuk által megvalósított függvények ÉS kapcsolatát jelenti. A kapcsolás szimbolikus jelölését a b. ábrán láthatjuk.

- ***A késleltetésekből adódó átmeneti jelenségek (hazárdok)***

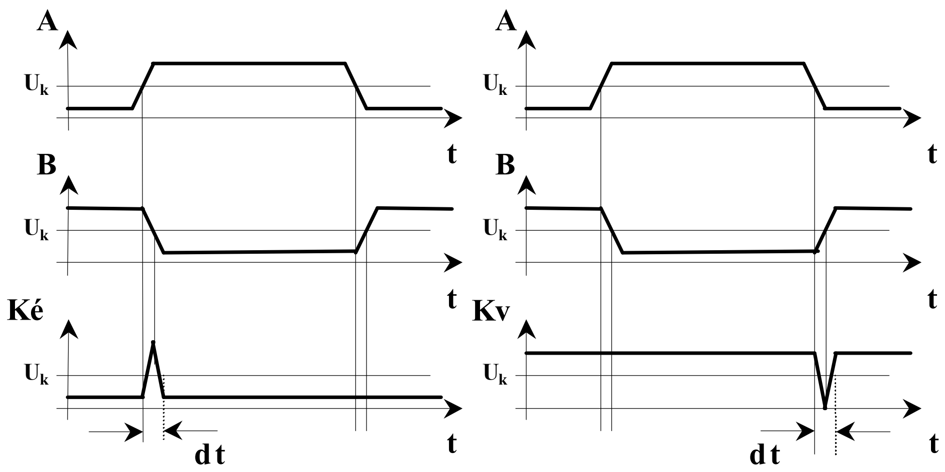
  - ***Hazárd, keletkezésének okai, és fajtái***

Hazárd olyan "***rövid idejű impulzus***" (átmeneti jelváltozás), amely eltér a logikai függvénye által ***meghatározott*** értéktől. Hazárd csak bemeneti ***jelváltozásakor*** keletkezik. Az impulzus ***szélessége*** (időtartama) rövidebb, mint a hálózat ***saját késleltetése***, és ***nagysága*** tullépi a kapu ***komparálási*** szintjét. Az ilyen jel további hibás működést okozhat, tehát ***zaj***. Miután egy kombinációs hálózat kimenete ***logikai kapu*** kimenet, ezért először vizsgáljuk meg, hogy mi a feltétele a hazárd keletkezésének a logikai kapuknál.

Egy logikai kapu ***kimenetén*** akkor ***keletkezhet hazárd***, ha két bemenetén ***ellenkező irányú*** jelváltás van, a jelváltás nem azonos időpontban történik, és a késés ***kisebb***, mint a ***hálózat*** teljes ***késleltetése***. A 35.a. ábrán látható áramkörben az ***ÉS***, illetve ***VAGY*** kapuk bemeneteire azonos jelek eérkeznek. A ***B*** jel **dt** értkkel késik az ***A*** jelhez képest.A két jel közötti **dt** késleltetést legtöbbször az ***áramkörön belüli*** hosszabb jelút - nagyobb ***késleltetés*** - eredményezi.A b, és c ábrákon követhetjük végig a két kapu kimenetén megjelenő jelalakot. Megállapíthatjuk, hogy ***ÉS*** kapu kimenetén a ***késleltetett jel 1-0*** átmenetekor jelenik meg hazárd, mégpedig az állandósult ***0 szintben***. A ***VAGY*** kapu kimenetén az állandósult ***1 szintből*** a 0 irányába mutató hazárd a ***késleltetett jel 0-1*** átmenetekor jelenhet meg.



a.



b. c.

35. ábra

A kombinációs hálózat a bemeneti jelek kombinációváltásának jellegétől függően ***három*** változatát különböztetjük meg a keletkező hazárdoknak Ezek

- ***a statikus-,***
- ***a funkcionális-, és***
- ***a dinamikus hazárd.***
  - ***Statikus***-nak nevezzük a hazárdot, ha a hálózat bemenetin csak ***egy jel*** változik, de ehhez - a függvény szerint - nem tartozik kimeneti jelszint váltás.
  - ***Funkcionális***-nak nevezzük az olyan hazárdot, mely ***két***, vagy ***több*** bemeneti jel változik a hálózat késleltetésen belüli időtartam alatt.

- *Dinamikus* hazárdnál a kimenet jel váltása ***duplázódik***, és azt ***egyetlen*** bemeneti jel változása eredményezi. Ilyen jellegű jelváltás többszintű hálózatoknál keletkezhet, ha a hálózat egyik részében ***statikus*** hazárd ***van***.

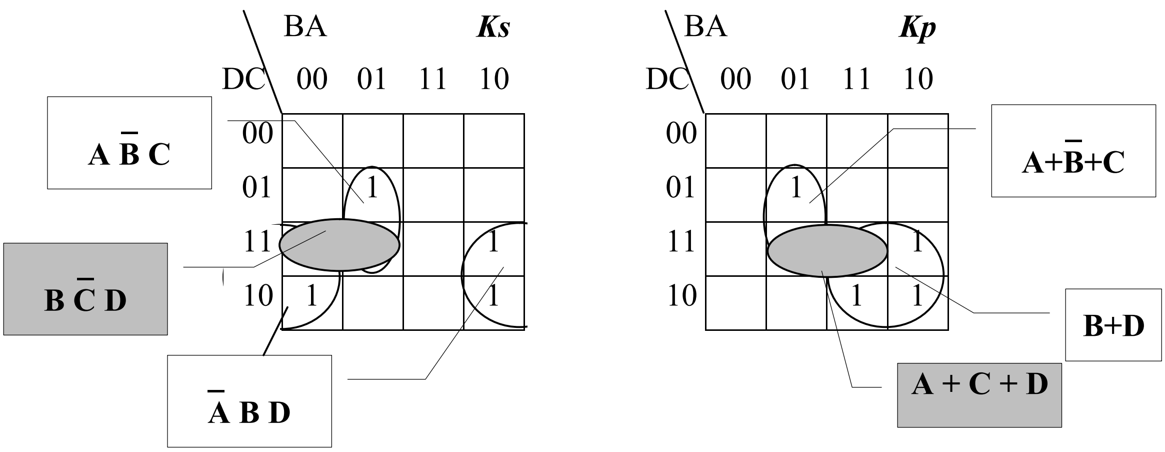
- Hazárd ***keletkezésének*** meghatározása

A kombinációs hálózat kanonikus logikai ***függvényeiből*** állapítható meg legkönnyebben, hogy ***statikus*** hazárd keletkezhet-e. Ha függvényben van két olyan ***logikai „szorzat”***, vagy ***logikai „összeg”***, amelyekben ugyanaz a változó az egyikben ***ponált***, a másikban pedig ***negált*** aklakú, akkor létrejöhet hazárd. Pl.

$$\ldots\ \mathbf{A\overline{B}C + \overline{A}D}\ \ldots\text{ill.}\ \ldots\ \mathbf{(A+\overline{B}+C)(B+D)}\ \ldots$$

Az első páldában az ***A*** jel ***váltása*** okozhat hazárdot, amikor a ***C=D=1***, és ***B=0***. A másodikban a ***B*** változó ***jelváltásánál*** keletkezhet hazárd, ha ***A=C=D=0***.

A függvény ***Karnaugh*** diagramjából is meghatározható, hogy a megvalósított hálózatban keletkezhet-e ***statikus hazárd***. A 36.ábrán megrajzoltuk – az algebrai alakban leírt – példák Ks, illetve Kp diagramjait. Ezen mutatjuk be, hogyan határozható meg a statikus hazárd keletkezése.



36. ábra

- ***Hazárdmentesítés***

A ***statikus*** hazárd kiküszöbölhető bővítő kapu beiktatásával. Olyan kombinációval kell bőviteni a hálózatot, amely a hazárdot eredményező változót nem tartalmazza, de az adott ***kombinációkban*** a kimenet logikai értékét nem változtatja meg. Az elözö példáknál:

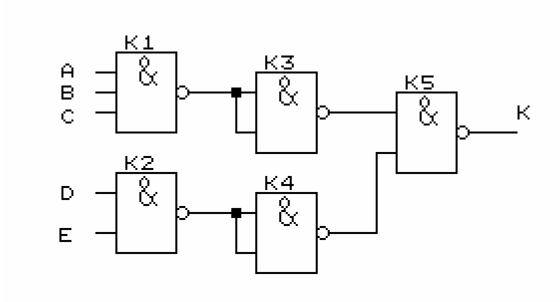
$$\ldots\ \mathbf{A\overline{B}C + \overline{A}D + \underline{\overline{B}CD}}\ \ldots\ \text{ill.}\ \ldots\ \mathbf{(A+\overline{B}+C)(B+D)\underline{(A+C+D)}}\ \ldots$$

az aláhúzott minterm (maxterm) kiegészítésnél a függvényérték nem változik, de a hazárdot okozó változó ezeben nem hat a kimenetre.

A 36.ábrán ***sraffozással*** jelöltük a hazárdmentesítő hurkokat. Látható, hogy ezek olyan egységeket fognak össze, amelyeket már más hurkok is lefednek. A megoldással ***nem a legegyszerübb*** megoldást kapju, viszont a statikus ***hazárdot megszüntetjük***.

- ***A TTL kapuk alkalmazása***

A megismert NAND kapuk felhasználásánál előfordulhat olyan eset is, hogy pl. nagyobb bemenetszámot kell megvalósítanunk, mint amilyen tokok rendelkezésünkre állnak. Erre példa a 37.ábra szerinti kapcsolás.



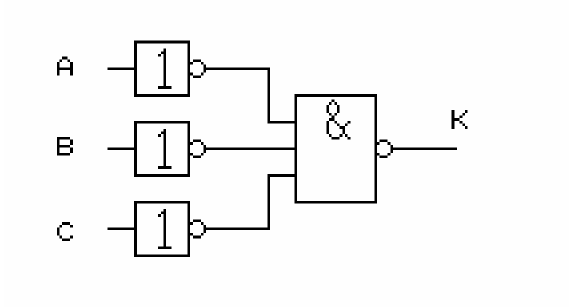
37. ábra

Itt öt bemenetű NAND kapcsolatot valósítottunk meg két és három bemenetű kapukkal. A logkai vázlat alapján fel írható a függvény-kapcsolat.

$$\mathbf{K = (\overline{\overline{ABC}})(\overline{\overline{DE}}) = \overline{ABCDE}}$$

A K1 jelű három-bemenetű kapu az első zárójeles mennyiség első tagadását, míg a második tagadást a K3 jelű kapu végzi. (A NAND kapu két bemenetét összekötve invertert kapunk). A második zárójeles mennyiséget - az előzőekhez hasonlóan - a K2 és K4 jelű kapuk képezik. E két mennyiség közötti ÉS -NEM műveletet hozza létre a K5 jelű kapu. A megoldáshoz 1 tok kellett a két-bemenetű változatból (K2,K3,K4,K5) és egy a három-bemenetű kapukat tartalmazó tokból (K1).

Csak NAND kapuk segítségével ***ÉS - VAGY*** típusú logikai hálózat is megvalósítható. Ennek megértéséhez először nézzük meg, hogyan hozhatunk létre NAND kapuval VAGY műveletet. A 38.ábra szerinti logikai vázlatnak megfelelően a NAND kapu bemeneteire az A,B,C változók tagadottjai jutnak .



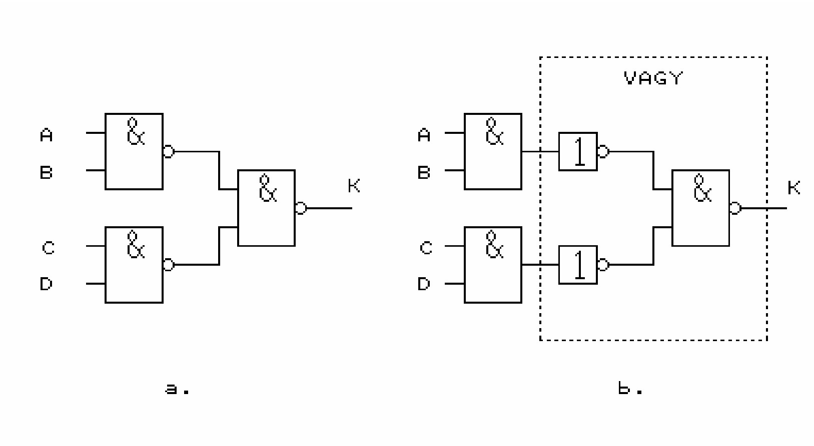
38. ábra

Felírva a logikai egyenletet a

$$\mathbf{K = \overline{\overline{A}\,\overline{B}\,\overline{C}} = A + B + C}$$

összefüggést kapjuk. Összefoglalva mondhatjuk, hogy a NAND kapu a bemeneteire jutó változók tagadottjainak VAGY kapcsolatát képezi.

A 39.a.ábra szerinti logkai vázlatot felrajzolhatjuk a b ábra szerint is, ha külön tekintjük a kapu invertereit. A szaggatott vonallal körülhatárolt részlet bemenetei között VAGY műveletet végez. Ezen két bemenet, pedig **AB**, valamint **CD** értékű.



39. ábra

Ezek alapján a megvalósított függvénykapcsolatunk

$$\mathbf{K = AB + CD}$$

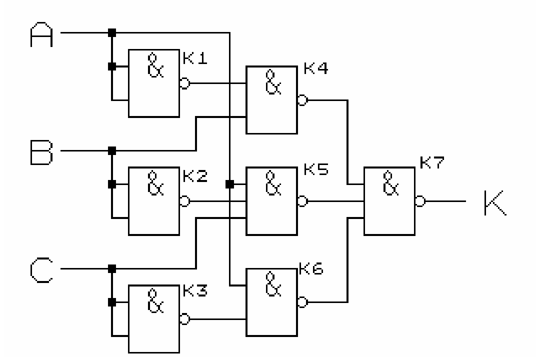
alakú függvénnyel adható meg.

A feladatot fordítva fogalmazva: egy ÉS-VAGY alakú logikai függvény csak NAND kapukkal is megépíthető.

Példaként rajzoljuk meg a

$$\mathbf{Z = \overline{A}B + A\overline{B}C + A\overline{C}}$$

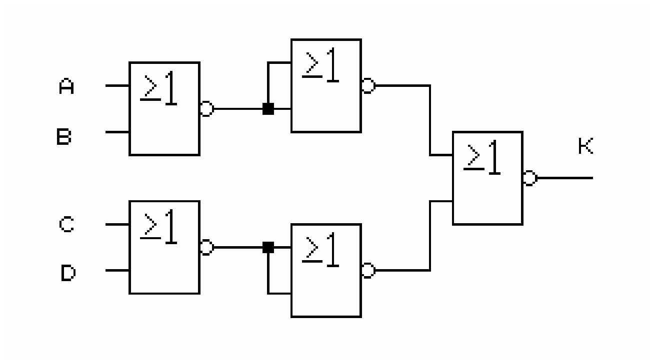
logikai függvénykapcsolatot létrehozó hálózat logikai vázlatát! A tagadásokat is NAND kapukkal állítsuk elő. A megoldást mutatja a 40.ábra.



40. ábra

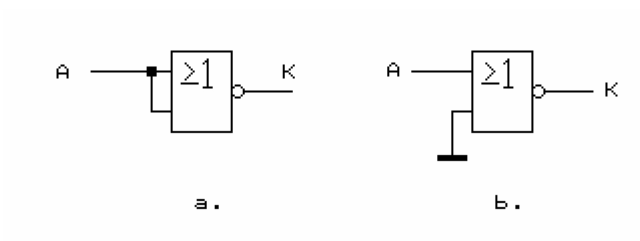
A logikai hálózat két tokkal építhető meg, úi. négy két-bemenetű kaput (K1,K2, K4,K6) és 3 három-bemenetűt (K3,K5,K7) használtunk. Az SN 7400 (négy két-bemenetű NAND kapu) és az SN7410 (három darab három-bemenetű NAND kapu) típusú IC tokokat használtuk.

Több logikai változó NEM-VAGY kapcsolatát - több két-bemenetű kapuból - a 41.ábra szerinti kapcsolásban lehet megvalósítani.



41. ábra

Az **inverter** áramkör - amely a logikai tagadás műveletét valósítja meg - tulajdonképpen egy-bemenetű kapu. Több bemenetű kapukból a bemenetek összekötésével, vagy csak egy bemenet használatával alakítható ki. Erre már a NAND kapu elemzésénél kitértünk. NOR kapuból a 42.ábra szerinti kapcsol sokkal alakítható ki inverter. A nem használt bemenetet - a logikai feltételekből adódóan - **0** szintre kell kötni.



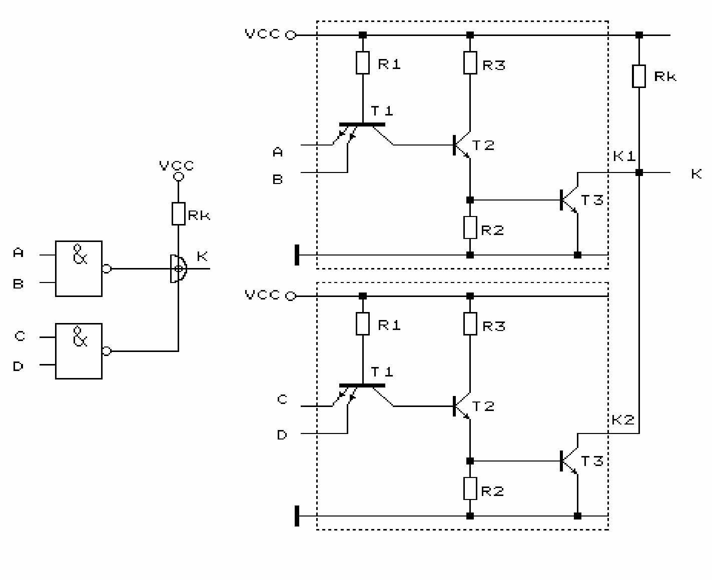
42. ábra

- ***Nyitott (open) kollektoros kapuk használata***

Több nyitott kollektoros kapu összekapcsolásával un. huzalozott logikai műveletet valósíthatunk meg. A 43.a. ábra szerinti logikai vázlaton két nyitott kollektoros ***NAND*** kapu kimenete közös $\boldsymbol{R_k}$ munkaellenálláson keresztül csatlakozik az $\boldsymbol{U_{cc}}$ tápfeszültségre. A b. ábrán az áramköri kapcsolási rajzot láthatjuk, amely segítségével határozhatjuk meg a ***K=f (A,B,C,D)*** logikai függvényt.

A K kimeneten csak akkor mérhetünk magas szintet, ha mindkét kapu kimeneti tranzisztora zárt, vagyis ***K= K1*K2*** logikai állítás igaz. Az egyes kapuk kimeneti tranzisztorai akkor zártak, ha a bemeneti jelek szintjei közül legalább az egy 0 értékű. A logikai függvények tehát:

$$\mathbf{K1 = \overline{A * B} \qquad K2 = \overline{C * D} \qquad K = K1 * K2 = \overline{A * B} * \overline{C * D}}$$

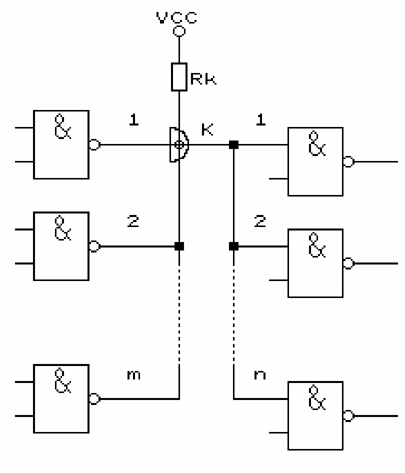


a. b.

43. ábra

A huzalozott kapcsolásokban alkalmazott külső munkaellenállás értékének megválasztásánál különböző feltételeknek kell teljesülnie.

Tételezzük fel, hogy m db nyitott kollektoros kapu kimenete van összekötve közös R_K. munkaellenálláshoz. A K kimenet, pedig n db további kapubemenetet vezérel az 44.ábra szerint



44. ábra

Az R_K meghatározása a következők szerint végezhető:

1. A kimenet 0 szintű értékénél a ***legkritikusabb*** eset az, amikor ***egyetlen*** kimeneti tranzisztor vezet. Az áram nem haladhatja meg a tranzisztor ***határáramát*** I_{cmax} - ot. Ezen a tranzisztoron folyik keresztül munkaellenállás árama, valamint a kimenet által vezérelt ***n*** db ***kapu*** bemeneti árama (I_{be0}). Ezek alapján teljesülnie kell a következő egyenlőtlenségnek.

$$\frac{\mathbf{Ucc - U_{01}}}{\mathbf{R_{KMIN}}} + \mathbf{n(-I_{be0}) \leq I_{c\,max}}$$

Az I_{be0} értékénél a legkedvezőtlenebb érték - az 1,6 mA - veendő figyelembe.

2. ***Lezárt*** kimeneti tranzisztoroknál, vagyis ***1 szintű*** kimenetnél az R_T ellenálláson folyik keresztül az ***m*** számú összekötött ***bemenet kollektor visszárama*** (I_{C0}) és az n számú vezérelt bemenet 1 szintjéhez tartozó árama (I_{be1}). Az összáram hatására sem csökkenhet a logikai 1 szint a megengedett alsó érték (U_{12}) alá. Ezt leíró egyenlőtlenség:

$$\mathbf{Ucc - R_{K\,max}(mI_{C0} + nI_{be0}) \geq U_{12}}$$

Az I_{C0} és az I_{bel} értékeknél az alkalmazott áramkör paramétereinek legkedvezőtlenebb szélsőértékeit kell figyelembe venni. (A tápfeszültség Ucc értékét állandónak tekinthetjük.) Az előző egyenlőtlenségekből számolható ki az R_T ellenállás névleges értéke és megengedett tűrése.

A ***nyitott kollektoros*** áramkörök külön csoportját alkotják az ***SN 7406***, és az ***SN7407*** típusú un. ***meghajtók***. A 7406 egy tokjában 6 darab ***invertáló***, míg a 7407 tokjában ugyancsak 6 db, de ***nem invertáló*** áramkör van Az áramkörök kimeneti tranzisztora 15 ... 30 V-os záró-feszültségű, ill. 40 mA áramterhelhetőségű. Ezek az inverterek, ill. csak kapcsoló erősítők meghajtó áramkörökként, vagy magasabb logikai szintű és TTL rendszer illesztésére használhatók.

- ***CMOS rendszerű kapuk***

A digitális integrált áramkörök technológiai és áramköri fejlesztésében a 80-as évtizedben terjedt el a térvezérelt tranzisztorok (**FET**) szélesebb körű alkalmazása. A digitális áramkörcsaládok kialakításban szigetelt vezérlőelektródájú **MOS-FET** (Metal Oxide Semiconductor - Field Effect Transistor), vagy röviden **MOS** tranzisztorokat, használnak. Ezekben az áramkörökben nagy elemürüség érhető el, mert egy MOS tranzisztor helyigénye lényegesen kisebb, mint a bipoláris tranzisztoré.

A MOS integrált áramkör bemeneti ellenállása közel végtelen, ezért nagy egyenáramú (dc) fan-out érhető el. Gyakorlatilag a fan-out értékét csak a működési sebesség korlátozza .A működési sebesség általában alacsonyabb, mint a bipoláris tranzisztorokból kialakított IC-ké. (Mai áramkörök már elérik a TTL sebességét). Ez alapvetően abból adódik, hogy a MOS - elemek nagy impedanciája mellett a szórt és terhelő kapacitások hatása számottevőbb.

A MOS integrált áramkörök két nagy csoportba sorolhatók:

MOS LSI és a

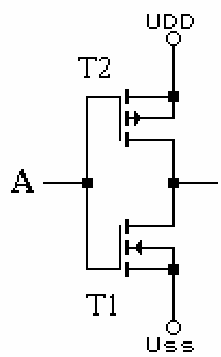
CMOS áramkörökre.

Az azonos típusú MOS tranzisztorokkal az alacsony integráltságú (SSI) digitális áramkörök (kapuk, flip-flopok stb.), illetve a közepes integráltságú (MSI) funkcionális egységek (számlálók, regiszterek stb.) gyártása gazdaságtalan. Ezért elsősorban a nagy integráltságú (LSI) áramkörök (mikroprocesszorok, memóriák stb.) készülnek ilyen megoldásban.

A ***komplementer*** - **p** és **n** ***csatornás*** - MOS tranzisztorokat együttesen alkalmazva, készülnek a ***CMOS*** vagy más néven ***COS-MOS*** integrált áramkörök. A CMOS kialakításban kiváló tulajdonságú ***SSI*** és ***MSI*** digitális áramkörök kerültek forgalomba. (Kisebb mennyiségben mikroprocesszorok és memóriák is készülnek CMOS technológiával.) A fejezetben a CMOS kapuk alapvető felépítésével, jellemzőivel foglakozunk.

➢ *CMOS kapuk*

A CMOS digitális áramkörök legegyszerűbb eleme a két komplementer tranzisztorból álló ***inverter*** (45. ábra). A sorba kötött **T1** (n csatornás) és **T2** (p csatornás) ***növekményes*** típusú tranzisztor közösített vezérlőelektródája – ***GATE*** **-** az áramkör bemenete (A). A kimenet (K) az összekötött ”kollektorokhoz”- ***DRAIN*** **-** (nyelő) csatlakozik. A tranzisztorok ”emitterei” – ***SOURCE*** - (forrás) a tápfeszültség két pontjához csatlakoznak.



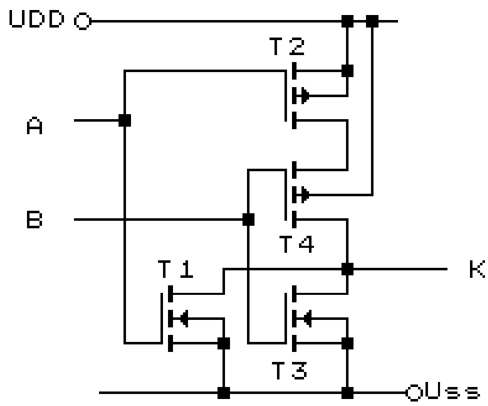
45. ábra

Az együttesen vezérelt komplementer tranzisztorok közül minden vezérlési állapotban (H agy L szintnél) csak az egyik vezet. Az alacsony szintű (***U_{SS}***) bemenő jelnél az n csatornás (T1) tranzisztor zár, mert a tranzisztor vezérlőfeszültsége (***U_{GS11})*** negatívabb a küszöbfeszültségnél. Ugyanakkor a T2 tranzisztor nyit, mivel a vezérlő bemenetére adott feszültség (***U_{GS12}***) - abszolutértéke- meghaladja a küszöbfeszültséget. A K kimenet - a vezető T2 tranzisztor kis csatorna-ellenállásán keresztül - az **U_{DD}** tápfeszültség pontra kapcsolódik, s ezért a feszültsége (U_K) közel azonos lesz azzal. Az **U_{DD}** szintű vezérlésnél a tranzisztorok állapota felcserélődik, s ezért a kimeneti feszültségszint jó közelítéssel az **U_{SS}** értékével fog megegyezni. A logikai szintek névleges értékének az **U_{SS}** -t ill. az **U_{DD}** -t választva, az áramkör a logikai tagadást valósítja meg. Jelentős előny, hogy mind ***pozitív***, mind ***negatív logikai*** rendszerben alkalmazható ugyanez az áramkör inverterként.

Az áramkör mindössze két aktív áramköri elemből áll. Mindkét logikai szintnél azonos a kimeneti ellenállás, és ezért a zavarvédettség is egyforma. A vezető tranzisztorok csatorna-ellenállása kisebb ***1 kΩ*** -nál. Jellemző - megengedett - kimeneti áram ***0,5 mA***.

A bemenet feszültség-vezérelt, s csupán az átkapcsolásoknál - az elektróda kapacitások átpolarizálásához - kell **nA** nagyságú áramot szolgáltatnia a meghajtó áramkörnek. Ez az előnyös tulajdonság viszont néhány hátránnyal is jár. A vezérlőelektródák kapacitásai csökkentik a kapcsolási sebességet. A késleltetés miatt a két tranzisztor átkapcsolása között átfedés jöhet létre. Ennek következtében - amikor mindkét tranzisztor vezet - átmenetileg megnő a tápáram felvétel. Ennek mértéke a tápfeszültség növelésével arányosan növekszik. (A táp-feszültség **U_{DD} - U_{SS}** 3 és 15 V, néhány típusnál 30 V közötti tetszőleges érték lehet.). Nagyon jelentős hátrány, hogy a szabadon hagyott bemenet kapacitása statikusan olyan mértékben feltöltődhet, hogy tönkremehet az áramkör. Ez viszont csak a korábbi típusoknál volt így. Ma már az áramkörökön belüli Zener diódás védőkapcsolásokkal gyártják az áramköröket.

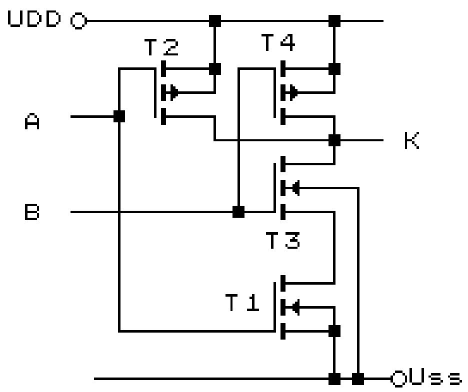
Komplementer MOS tranzisztorok vegyes kapcsolásával ***VAGY-NEM*** (***NOR***), ***ÉS-NEM*** (***NAND***), valamint összetett logikai műveleteket, megvalósító kapukat is készítenek. A 46.ábra szerinti kapcsolású áramkör működése a következő. Amikor a bemenetek közül (A, B) legalább az egyik $\mathbf{U_{DD}}$ vezérlést kap, akkor az ide kapcsolódó n - csatornás tranzisztorok (T1,T3) közül az egyik, vagy mindkettő ***vezet***. A p - csatornás tranzisztorok (T2,T4) közül az egyik, vagy mindkettő ***zárt***.



46. ábra

A K kimenet - a vezető tranzisztoron keresztül - az $\mathbf{U_{SS}}$ pontra kapcsolódik és feszültsége közel ezzel az értékkel lesz egyenlő. A kimeneti feszültség (U_K) csak akkor lesz $\mathbf{U_{DD}}$ értékű, ha mindkét bemenet $\mathbf{U_{SS}}$ szintű vezérlést kap. Ha **pozitív logikai** szintet veszünk alapul, akkor az **1**-szint az $\mathbf{U_{DD}}$ és a **0**-szint pedig az $\mathbf{U_{SS}}$. Az áramkör ilyenkor VAGY-NEM (NOR) kapu. Negatív logikai rendszerben - az értelemszerű fordított szintválasztás eredményeként - az áramkör ÉS-NEM (NAND) kapu.

A 47. ábra szerinti áramkör is az előzőekhez hasonlóan elemezhető



47. ábra

Az áramkör pozitív logikai rendszerben NAND, negatív logikai rendszerben pedig NOR kapu.

A CMOS áramkörökben kialakított növekményes MOS tranzisztorok küszöbfeszültsége Us = 2V. A vezérlő-elektródára megengedett feszültség (U_{GS}) maximuma 15-20 V. Az áramkör ezért használható széles tápfeszültség tartományban. Ez az áramkörcsaládok legtöbbjénél 3-15 V lehet.

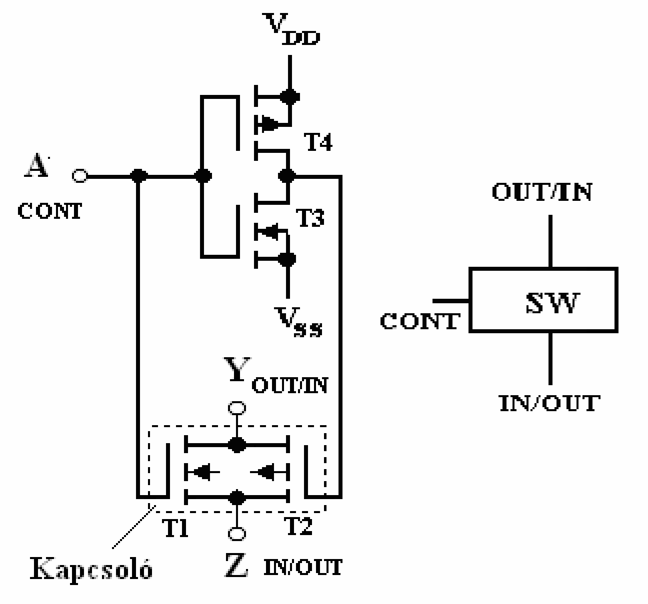
Az áramkörök nyugalmi tápáram-felvétele nagyon kicsi, és a disszipáció is 10 nW nagyságrendű. A működési frekvencia növekedésével a disszipáció hatványozottan emelkedik.

A CMOS áramkörök korábbi változataiban az átlagos jelterjedési idő t_{pd}=50 ns. A legújabb fejlesztések eredményeként már léteznek a normál TTL sorozat késleltetési idejét megközelítő CMOS áramkörök is.

- ***CMOS kapcsoló***

Ellenpárhuzamosan kapcsolt ***komplementer*** tranzisztor-párból, digitális jellel vezérelt kétirányú jelátvitelre alkalmas (**bilaterális**) ***elektronikus kapcsoló*** (48. ábra) alakítható ki.

A kapcsoló a **T1 n** csatornás és **T2 p** csatornás tranzisztor, amelyek a T3-T4 tranzisztorokból álló inverter beiktatásával ellenütemben kapnak vezérlést. Ha a **Z** pontot (közösített drain) tekintjük a bemenetnek, és az **Y** (közösített source) a kimenet, akkor a működés a következő. (Az $\mathbf{U_{be}}$ bemenő feszültség $\mathbf{U_{SS}}$ - $\mathbf{U_{DD}}$ érték közötti lehet.) Az A bemenet alacsony szintű vezérlésénél mindkét tranzisztor zárt, mivel az n - csatornás T1 tranzisztor vezérlőfeszültsége a küszöbfeszültségnél negatívabb, ill. a T2 p - csatornás tranzisztornál, pedig pozitívabb. Ezért a Z és Y pont között nagy impedancia mérhető. A magas szintű vezérlésnél - az U_{ZY} értékétől függően - legalább az egyik tranzisztor vezet, és így a Z és Y között kis impedanciájú a kapcsolat .A MOS tranzisztorok szimmetrikusak, ezért a source és drain felcserélhető. Ez az adott kapcsolásban a be-; és a kimenet (Z, Y) felcserélését is lehetővé teszi. Az integrált technológiával kialakított önálló bilaterális kapcsoló-elem az átvivő tranzisztorok mellett az invertert is tartalmazza.



48. ábra

Az ellenpárhuzamosan kapcsolt tranzisztorok közül az n csatornás substrátja az Uss, míg a p csatornásé az U_{DD} tápfeszültség ponthoz kapcsolódik.

3.7. Példák

3.8. Ellenörző kérdések

4. Digitális hálózatok

A fejezetben a ***műszaki feladatok*** megoldására tervezett, és megvalósított ***digitális hálózatok***

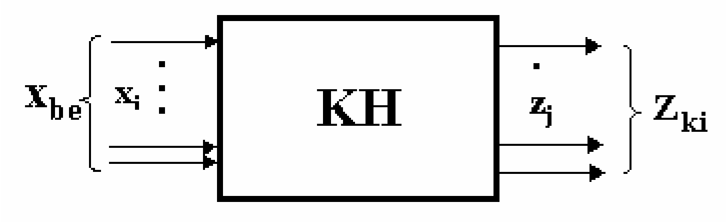
- alapvető tulajdonságait,
- a tervezés módszereit

tárgyaljuk.

Első lépésben megvizsgáljuk, hogy a logikai feladatok milyen ***rendszertechnikai*** megoldásokkal valósíthatók meg. Másodsorban megismerkedünk a ***logikai tervezés*** bevált módszereivel. Befejezésként áttekintjük azokat a ***funkcionális*** egységeket, amelyek felhasználásával egyszerűen építhetők össze a nagyobb digitális hálózatok.

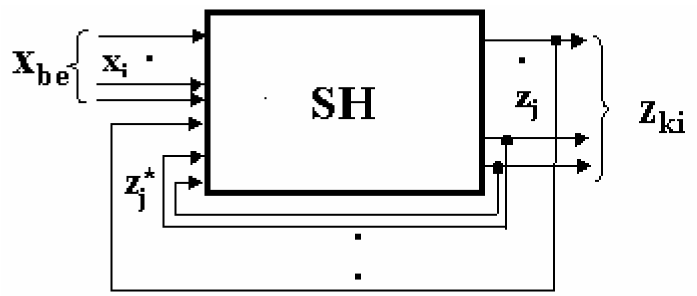
A műszaki feladatok megoldására alkalmazott digitális hálózatok ***alapvetően két nagy csoportba*** sorolható. A besorolást a be-, és a kimeneti digitális jelek közötti ***időbeli kapcsolat*** alapján tehetjük meg.

A feladatok egyik nagy csoportjánál a ***kimenet***(ek) – ***függő változó***(k) - logikai ***érték-kombinációját csak*** a ***bemeneti*** jel(ek) – ***független változó***(k) – vizsgált ***időpontbeli értékkombinációjától*** függ. Az ilyen feladatokat megvalósító digitális áramköröket ***kombinációs hálózatnak*** nevezzük. Blokkvázlata a 49.ábrán látható.



49. ábra

A másik csoportba az olyan az olyan logikai feladatok tartoznak, amelyeknél a ***kimenet***(ek) – függő változó(k) - logikai ***értékét*** a ***bemeneti*** jel(ek) – független változók – , és ***kimeneti*** jel(ek) – függő változó(k) - vizsgált ***időpontbeli értékkombinációja, együtt határozzák*** meg. Blokkvázlata a 50.ábra szerinti. A leírt tulajdonságú logikai feladatokat megvalósító digitális áramköröket nevezzük ***sorrendi hálózatoknak***. Még használják a ***szekvenciális***-, illetve ***emlékező***-hálózat elnevezéseket is.



50. ábra

4.1. Kombinációs hálózatok

A műszaki feladatok egy jelentős csoportjában a ***kimenetek jeleit csak*** a ***bemenetekre*** jutó ***jelek*** aktuális értéke ***szabja meg***. Az ilyen feladatok – a már definiált – kombinációs logikai hálózatokkal megvalósíthatók. Röviden áttekintjük a feladat ***logikai leírásának*** változatait. Utána megismerkedünk a feladatot megvalósító áramköri ***tervezés*** alapvető ***módszereivel***. Összefoglaljuk a leggyakrabban alkalmazott, un. ***funkcionális*** kombinációs egységek feladatait, működésüket.

A kombinációs hálózatok meghatározásából következik, hogy a ***bemenetek*** (független változók), valamint a ***kimenetek*** jelei (függő változók) között ***egyértelmű logikai*** függvénykapcsolat van. Ez megadható az

$$\mathbf{Z_i = f_l\,(X_i\,)}$$

általános függvényleírással, amelyben $\mathbf{Z}_i$ a hálózat kimeneti kombinációja az **i**-ik időpillanatban, az $\mathbf{X_i}$ ugyanekkor érvényes bemeneti kombináció, és az $\mathbf{f_l}$ írja le a ***logikai*** függvénykapcsolatot. Logikai függvények ténylegesen nem az értékkombinációkra, hanem egy-egy valós kimenetre írhatók fel.

A tananyag első fejezetében – az elméleti alapokban – már részletesen tárgyaltuk a logikai függvények megadásának (leírásának) használt változatait. Itt most ismétlésként ismételjük meg azok összefoglalását.

A függvények megadása – leírása – történhet

algebrai alakban,

táblázat segítségével***,***

matematikai jelölésekkel,

grafikus módon,

időfüggvény formájában.

A felsorolt leírási módok teljesen egyenértékűek, és egymásba átírhatók!

A kombinációs logikai feladatokat megvalósító hálózatok tervezésénél a megismert függvénymegadási formákat használjuk.

- ***Kombinációs hálózatok logikai tervezése***

Egy ***hálózat tervezése*** több részből áll. Először a függő változók – kimenetek – ***logikai függvényeit*** kell meghatározni. Ezt nevezzük ***logikai*** tervezésnek. Az eddigi munka eredményéből lehet az ***áramkört*** megtervezni. Mindezek után következhet a ***huzalozási***, illetve ***nyomtatási*** terv elkészítése. A következő szakasz már a ***gyártás***, és az ***ellenőrzés***.

A ***logikai tervezés*** bármelyik formája a feladat ***egyértelmű leírására*** épül. A ***gyakorlatban*** a megadás első változata a feladat ***szöveges leírása***. Miután – bármely nyelvben – a szöveges definíció félreértésekre is adhat okot, valamint a ***szisztematikus*** tervezést sem támogatja, ezért azt egy közbenső – egyértelműen értelmezhető – formára kell átfordítani.

A logikai tervezés a következő lépésekre bontható:

- a feladat ***változóinak*** egyértelmű meghatározása,
- a függvény ***igazságtáblázatának*** felírása,
- a függvények ***kanonikus*** alakjainak – valamelyik módszerrel (algebrai, indexelt, grafikus) történő felírása,
- az ***egyszerűsítések*** – minimalizálás – végrehajtása,
- ***logikai vázlat*** megrajzolása.

A legegyszerűbb, egyszerűsített logikai függvény meghatározása történhet:

- ***Algebrai*** úton
- A ***kanonikus*** (diszjunkt, vagy konjunkt) ***alakokból.***
- ***Grafikus*** módszerekkel
- ***Karnaugh táblázatok*** (minterm Kp, maxterm Ks) felhasználásával.
- ***Numerikus*** módszerrel (Quin,- Mc Closkey eljárás)

A példa segítségével tekintsük át a kombinációs hálózat tervezésének menetét.

10.Példa

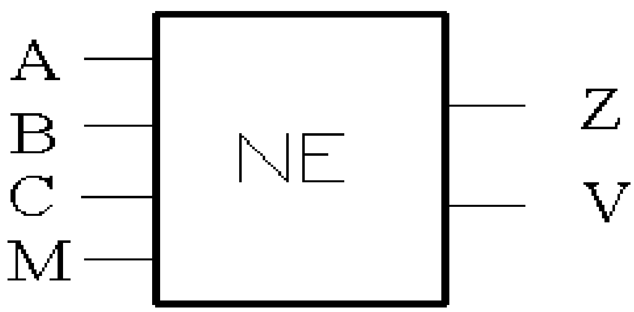
Feladat olyan hálózat tervezése, amelynek ***négy bemenete***, és ***két kimenete*** van.

A bemenetek közül három ***adat-***, egy pedig ***parancs*** jelet fogad. Az ***egyik kimenetén*** akkor kapunk logikai 1 értéket, ha a ***három*** adatbemenete közül – a parancs értékétől függetlenül - ***kettőn*** van logikai ***1*** érték. A másik kimeneten 1 szintű legyen, ha a ***parancs 0*** és az adatbemeneteken ***több*** az ***1*** érték, mint a ***0***, amikor, pedig a ***parancs 1*** értékű a ***több 0*** értéknél legyen ***1*** a kimenet.

Jelöljük az ***adatbemeneteket A***, ***B***, ***C***, míg a ***parancsot M*** betűkkel.

A ***kimeneteket,*** pedig jelöljék a ***Z*** (a parancstól független), és ***V*** betűk.

A7 51. ábrán rajzoltuk meg megvalósítandó hálózat elvi ***blokkvázlatát***



51. ábra

A hálózat ***igazságtáblázatát*** láthatjuk az 52.ábrán

M	*C*	*B*	*A*	*Z*	*V*
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	0

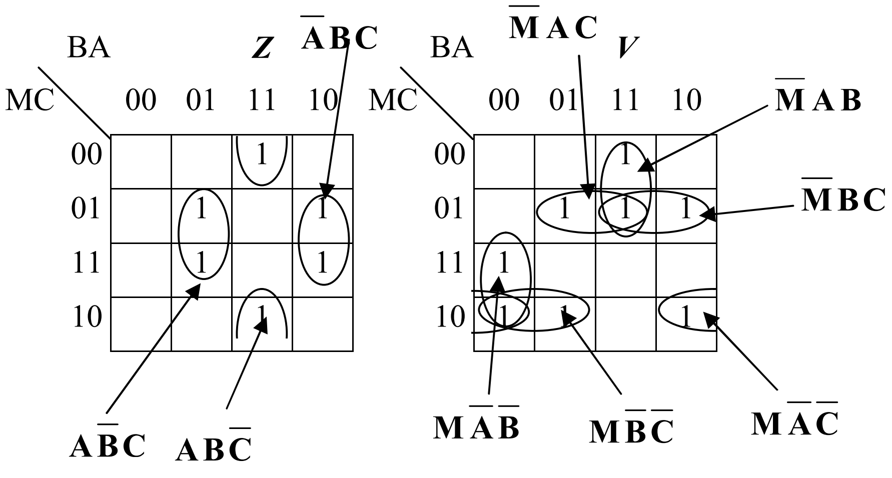
52. ábra

a kimeneti ***diszjunktív kanonikus*** függvények algebrai alakjai, és az összevonások.

$$\begin{aligned}
\mathbf{Z} &= \mathbf{A\cdot B\cdot \overline{C}\cdot \overline{M} + A\cdot \overline{B}\cdot C\cdot \overline{M} + \overline{A}\cdot B\cdot C\cdot \overline{M}} + \\
&\mathbf{+ A\cdot B\cdot \overline{C}\cdot M + A\cdot \overline{B}\cdot C\cdot M + \overline{A}\cdot B\cdot C\cdot M} = \\
&= \mathbf{(A\cdot B\cdot \overline{C} + A\cdot \overline{B}\cdot C + \overline{A}\cdot B\cdot C)\cdot \overline{M}} + \\
&+ \mathbf{(A\cdot B\cdot \overline{C} + A\cdot \overline{B}\cdot C + \overline{A}\cdot B\cdot C)\cdot M} = \\
&= \mathbf{A\cdot B\cdot \overline{C} + A\cdot \overline{B}\cdot C + \overline{A}\cdot B\cdot C = A(B\cdot \overline{C} + \overline{B}\cdot C) + \overline{A}\cdot B\cdot C}
\end{aligned}$$

$$\begin{aligned}
\mathbf{V} &= \mathbf{A\cdot B\cdot \overline{C}\cdot \overline{M} + A\cdot \overline{B}\cdot C\cdot \overline{M} + \overline{A}\cdot B\cdot C\cdot \overline{M} + A\cdot B\cdot C\cdot \overline{M}} + \\
&\mathbf{+ \overline{A}\cdot \overline{B}\cdot \overline{C}\cdot M + A\cdot \overline{B}\cdot \overline{C}\cdot M + \overline{A}\cdot B\cdot \overline{C}\cdot M + \overline{A}\cdot \overline{B}\cdot C\cdot M} = \\
&= \mathbf{A\cdot B\cdot \overline{M} + A\cdot C\cdot \overline{M} + B\cdot C\cdot \overline{M} + \overline{A}\cdot \overline{B}\cdot M + \overline{A}\cdot \overline{C}\cdot M + \overline{B}\cdot \overline{C}\cdot M} = \\
&= \mathbf{\overline{M}\cdot(A\cdot B + A\cdot C + B\cdot C) + M\cdot(\overline{A}\cdot \overline{B} + \overline{A}\cdot \overline{C} + \overline{B}\cdot \overline{C})}
\end{aligned}$$

Egyszerűsítés Karnaugh (Kp) diagram segítségével (53.ábra):



53. ábra

Az ***összevonások*** után kapott függvények

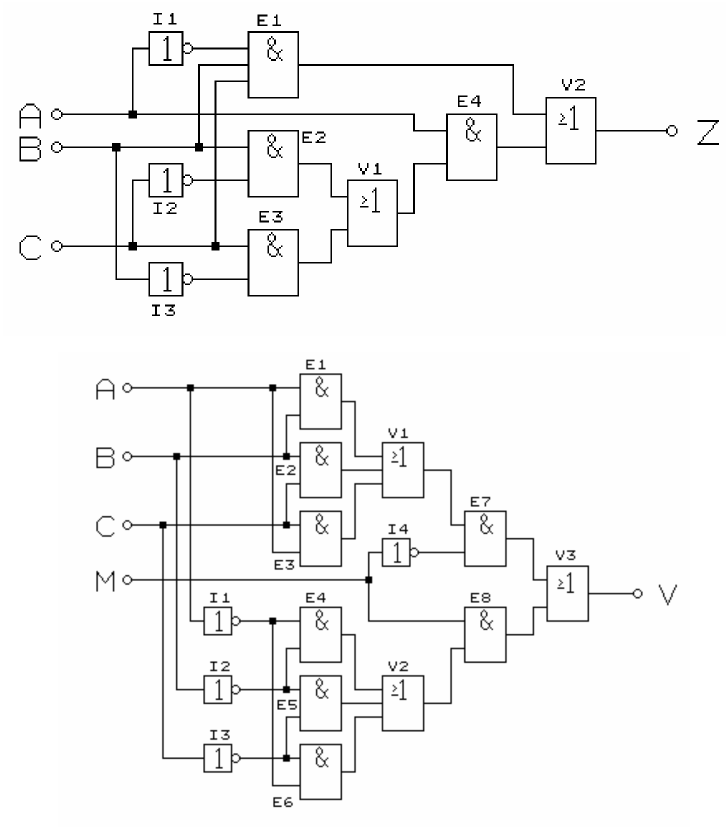
$$\mathbf{Z= A\cdot B\cdot \overline{C}+A\cdot \overline{B}\cdot C+\overline{A}\cdot B\cdot C=A(B\cdot \overline{C}+\overline{B}\cdot C)+\overline{A}\cdot B\cdot C}$$

$$\mathbf{V=A\cdot B\cdot \overline{M}+A\cdot C\cdot \overline{M}+B\cdot C\cdot \overline{M}+\overline{A}\cdot \overline{B}\cdot M+\overline{A}\cdot \overline{C}\cdot M+\overline{B}\cdot \overline{C}\cdot M=}$$
$$\mathbf{=\overline{M}\cdot(A\cdot B+A\cdot C+B\cdot C)+M\cdot(\overline{A}\cdot \overline{B}+\overline{A}\cdot \overline{C}+\overline{B}\cdot \overline{C})}$$

A függvények megegyeznek az algebrai egyszerűsítés eredményeivel.

A ***logikai vázlat*** megrajzolása

A logikai kapuk ***szimbólumaival*** rajzolható meg - a megvalósítandó - kombinációs hálózat, 54. ábra szerinti ***logikai vázlata***.



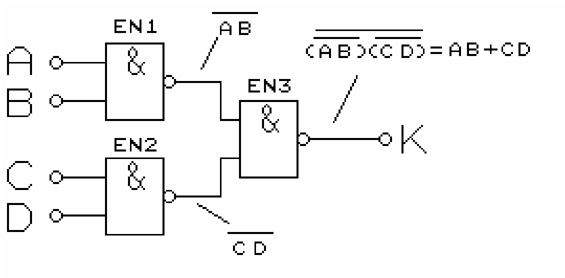
54. ábra

- ***Összetett műveletek használata***

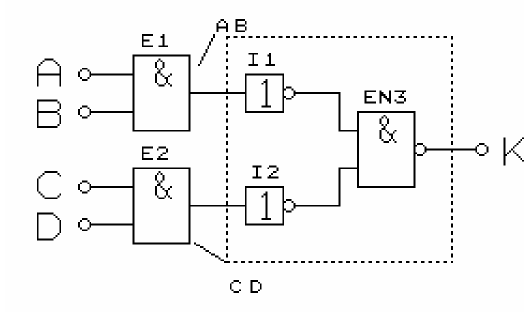
Az eddigiekben az ***ÉS***, ***VAGY***, valamint a ***TAGADÁS*** műveletét alkalmaztuk. Továbbiakban foglalkozunk a ***NAND***, a ***NOR***, az ***XOR*** (kizáró vagy), valamint az ***XORN*** (az XOR tagadottja, equivalencia) műveletek alkalmazásával a kombinációs hálózatok megvalósításánál. Az előző kettőt nevezik ***univerzális műveletnek*** is, és elsősorban az általános kombinációs hálózatok megvalósításához használjuk. Az utóbbi két műveletet ***moduló - műveleteknek*** is nevezik, és elsődlegesen az aritmetikai feladatok (összeadás – kivonás, összehasonlítás) végrehajtásához alkalmazzák.

- **Az *univerzális* műveletek alkalmazása**
  - ***NAND kapuk alkalmazása***

Az 55.a.ábrán látható hálózat csak NAND kapukból áll. Írjuk fel az egyes kapuk kimenetein érvényes logikai függvényeket, és végül a teljes hálózat K kimenetének függvényét.



a.



b.

55. ábra

A kapuk kimenetein felírható függvények alapján tehát kimeneti jel értéke a

$$K = A\,B + C\,D$$

függvény szerinti.

Az 55.b.ábrán az EN1, és az EN2 jelű NAND kapukat szétválasztottuk ÉS kapukra, (E1, E2)valamint inverterekre (I1, I2). A ***szaggatott*** vonallal körülhatároltuk részlet ***VAGY*** műveletet valósít meg, mivel

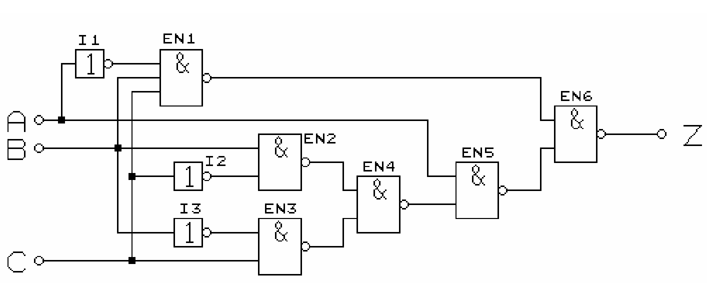
$$\overline{\overline{(A\,B)}\,\overline{(C\,D)}} = A\,B + C\,D$$

A példa azt szemlélteti, hogy NAND kapukkal megépített ***két-szintű*** kombinációs hálózat ***kimenet felőli*** szintje ***VAGY*** műveletet, azt ***megelőző szint***, pedig ***ÉS*** műveletet hajt végre. Az előző megállapítás több szintű hálózatra is kiterjeszthető, ha azt vesszük, hogy a szintek párosával mindig csoportosíthatók. A többszintű hálózatot a kimenet felől – jelutak szerint - osztjuk ***páratlan*** – ***páros*** szintekre. Általánosan tehát igaz a következő:

- ***a NAND kapu páratlan szinten VAGY, míg páros szinten ÉS műveletet valósít meg,***
- ***a páros szinten bevezetett jel változatlan értékkel, míg a páratlan szinten bevezetett jel az eredeti tagadottjaként szerepel a kimenet függvényében.***

A megismert törvényszerűségek alapján, az 56. ábrán az előző példa Z kimenetét létrehozó hálózat NAND kapuk alkalmazásával megrajzolt logikai vázlata látható.

$$Z = A\cdot B\cdot\overline{C} + A\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot C = [A(B\cdot\overline{C} + \overline{B}\cdot C)] + [\overline{A}\cdot B\cdot C]$$

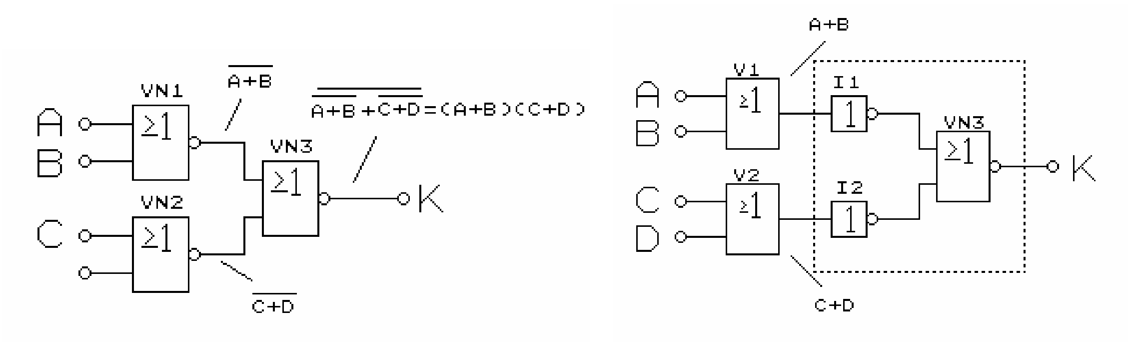


56. ábra

A függvény algebrai alakját – a szögletes zárójelek segítségével – csoportosítottuk. A Z kimenet értékét tehát két mennyiség VAGY kapcsolata adja, amit az EN6 jelű NAND kapu valósít meg. Az EN1 jelű kapu a jobboldali, míg az EN5 jelű kapu a baloldali szögletes zárójeles ÉS műveletet állítja elő. Az EN4 kapu ismét VAGY (páratlan szint), míg az EN2, EN3 jelű kapuk, pedig ÉS műveletet hoznak létre. A B, és C változók. És tagadottjaik is páros szinten jutnak a rendszerbe, ezért a függvényben is így szerepelnek. Az A változó két különböző műveletben is szerepel, és páros szinten vezetjük be. Mivel az egyik műveletben negált alakban kell, szerepeljen, ezért kellett az I1 jelű inverter.

- ***NOR kapuk alkalmazása***

Az 57.a.ábrán látható hálózat csak NOR kapukból áll. Láthatók az egyes kapuk kimenetein érvényes logikai függvények, és végül a teljes hálózat K kimenetének függvénye.



a. b.

57. ábra

A kapuk kimenetein felírható függvények alapján tehát kimeneti jel értéke a

$$K = (A + B)(C + D)$$

logikai függvény szerint függ a bemenetek logikai értékeitől.

Az 57.b.ábrán az VN1, és az VN2 jelű NOR kapukat szétválasztottuk VAGY kapukra, (V1, V2)valamint inverterekre (I1, I2) . A szaggatott vonallal körülhatárolt részlet ÉS műveletet valósít meg, mivel

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

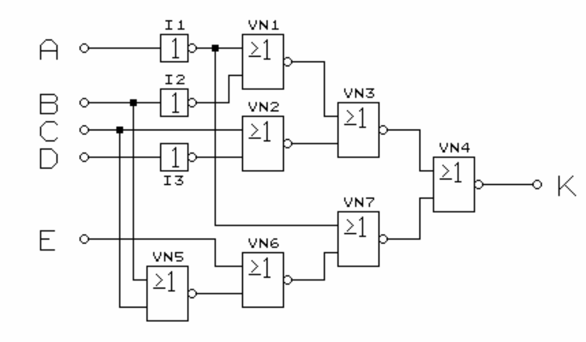
A példa azt szemlélteti, hogy a NOR kapukkal megépített ***két-szintű*** kombinációs hálózat ***kimenet felőli*** szintje ***ÉS*** műveletet, azt ***megelőző szint***, pedig ***VAGY*** műveletet hajt végre. A NAND kapus hálózatokhoz hasonlóan, törvényszerűség több szintre is megállapítható. Általánosan tehát igaz a következő:

- ***a NOR kapu páratlan szinten ÉS, míg páros szinten VAGY műveletet valósít meg,***
- ***a páros szinten bevezetett jel változatlan értékkel, míg a páratlan szinten bevezetett jel az eredeti tagadottjaként szerepel a kimenet függvényében.***

A leírtak alapján rajzoljuk meg a

$$K = (A\,B + \overline{C}\,D)(\overline{A} + \overline{E}\,(B + C))$$

függvény NOR kapukkal történő megvalósításának logikai vázlatát. A kimenet (K) a ***két*** – szögletes zárójelbe tett - ***mennyiség ÉS*** kapcsolata, amit a ***páratlan*** szinten álló két-bemenetű - VN4 jelű - ***NOR*** valósít meg. A kapu bemeneteihez csatlakozó – VN2, és VN7 jelű kapuk – ***páros szinten*** vannak, és ezért ***VAGY*** műveletet realizálnak sit.. A további kapuk szintje, s az általuk megvalósított logikai műveletek 58. ábrán követhetők végig.



58. ábra

Összefoglalva: megállapíthatjuk, hogy bármely kombinációs feladat megvalósítható csak NAND, vagy csak NOR kapuk alkalmazásával. Vegyesen nagyon ritkán használják a különböző ***univerzális*** kapukat. Az integrált áramköri digitális áramkörök tárgyalásakor, (3. fejezet) tárgyaltuk azt is, hogy az univerzális kapuk inverterként is használhatók.

- **A *kizáró*-vagy kapuk alkalmazása**

Az 1. fejezetben megismertük a ***KIZÁRÓ-VAGY*** (XOR) logikai műveletet, amelyet ***moduló - összegzésnek*** is neveznek. Minden olyan alkalmazásban, amelyben a függő változó csak akkor IGAZ, ha a független változók érték-variációban csak páratlan, vagy csak páros számú az IGAZ, akkor használhatók az XOR, vagy az NXOR műveletek. (Mivel a technikai, műszaki feladatokban gyakori a hasonló feltétel, ezért az integrált áramköri kapuk között ezek megtalálhatók.)

A függvények algebrai, illetve Karnaugh diagramon történő egyszerűsítésekor felismerhető az XOR kapu alkalmazhatósága. A ***kétváltozós*** moduló - összeg algebrai alakja: $\mathbf{A\overline{B}+\overline{A}B}$, Kp diagramja, pedig az 59. ábrán látható. A 60.a. és b. ábrákon a ***három***-, illetve ***négyváltozós*** XOR művelet Kp diagramja látható.

B \ A	0	1
0		1
1	1	

59. ábra

C \ BA	00	01	11	10
0		1		1
1	1		1	

a.

DC \ BA	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

b.

60. ábra

A 60. ábra mindkét táblázatában megfigyelhetjük, hogy a ***szaggatott*** vonalak mentén „***összehajtva***" a táblát, akkor az 1-ek a másik rész 0 értékére esnek. A megfigyelés lehetővé teszi az XOR művelet lehetséges használatát.

A fejezetben megoldott példa Z, és V kimeneti változók Karnaugh diagramjai láthatók a 61.ábrán.

Z

MC \ BA	00	01	11	10
00			1	
01		1		1
11		1		1
10			1	

a.

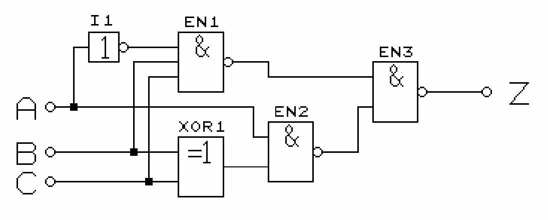
V

MC \ BA	00	01	11	10
00			1	
01		1	1	1
11	1			
10	1	1		1

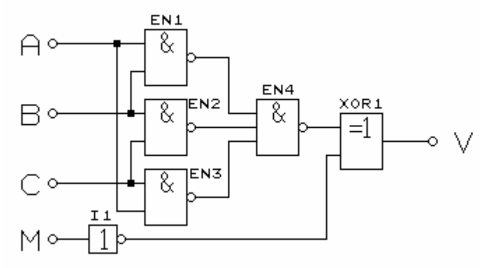
b.

61. ábra

A 62.ábrán az XOR kapuk alkalmazásával kialakított logikai vázlatok láthatók.



a.



b.

62. ábra

A kétbemenetű XOR kapu vezérelt inverter -nek is használható.

➢ *Funkcionális kombinációs feladatok*

A logikai kapukka1 elvileg minden logikai feladat megvalósítható. Ugyanakkor a legkülönbözőbb rendeltetésű hálózatokban megtalálhatók olyan nagyobb ***funkciókat*** ellátó egységek is, amelyek egyedi tervezéssel, kapukból is megépíthetők. Mivel gyakran használják digitális hálózatok elemeiként, ezért ***önálló áramkörként*** gyártják. Ezeket általában **rendszertechnikai** (funkcionális) áramköröknek nevezzük.

Ilyen funkcionális egységek a következők:

- kódolók, dekódolók;

- adatelosztók;

- adatkiválasztók;

- aritmetikai műveletvégzők.

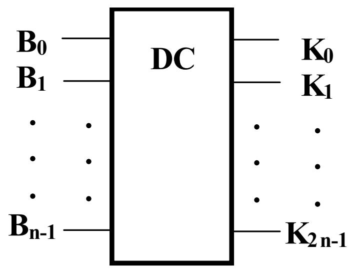
E fejezetben csak ezen funkcionális egységek logikai felépítésével, és működésével foglalkozunk.

- ***Dekódolók***

A dekódoló egy ***kódátalakító*** kombinációs feladatot valósít meg. Az egység ***bemenetire*** adott ***bináris*** vagy ***BCD*** kódból az ***n*** db ***kimeneten*** un. **1 az n - ből** kódot állit elő. Ez a kód **n** bitből áll, és ezek közül mindig csak ***egy lehet aktív*** logikai értékű. Azt pedig, hogy melyik kimeneten lesz ***aktív*** érték, azt a bemeneteken lévő ***aktuális*** kód határozza meg. Ha az aktív logikai érték 1, akkor a többi bit 0, illetve fordítva.

Az ismertetett logikai hálózat működése lényegében ***kiválaszt egy kimenetet***, ugyanis a bemenetire adott kód alapján ***egyetlen*** kimenetet tesz ***aktívvá***. Joggal vetődik fel a kérdés, hogy miért is nevezzük dekoder -nek? A számítástechnika „hőskorában” a decimális számjegyek kijelzésére használt ***Nixie*** csövek (gáztöltésű kijelző csövek) ***megfelelő*** katódját kellet kiválasztani – 0 feszültséggel – ahhoz, hogy a kijelezni kívánt ***számjegy*** látsszon. Innen ered, hogy a bináris, vagy BCD kódolású számot „***dekódolta***” decimális formájú karakterre.

A dekódoló elvi blokkvázlatát a 37.ábra szemlélteti. A $\mathbf{B_0 \,.\,.\,.\, B_{p-1}}$ jelű bemenetekhez csatlakozik az **p** bites átalakítandó kód (BCD kódnál p = 4). A $\mathbf{K_0 \,.\,.\,.\, K_{2p-1}}$ jelű kimeneteken kapjuk az **1 az n-ből** kódot. (**p** bites ***bináris*** kódnál, a kimeneti ***bit*** -ek száma (n) maximálisan $\mathbf{2^p}$ lehet.).



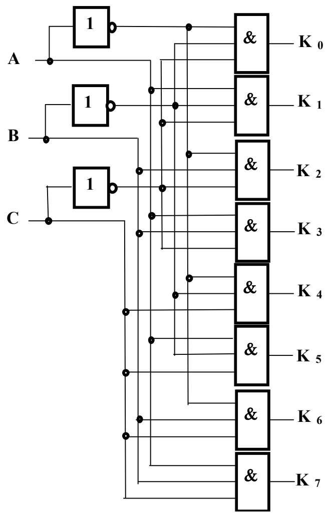
63. ábra

Bináris dekódoló

A 3 bites bináris kód (3-ról 8-ra) dekódolását végző kombinációs hálózat igazságtáblázata - logikai 1 szintű aktív kimenetet választva - a.38.a.ábrán a logikai vázlata, pedig a b ábrán látható.

C	B	A	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

a.



b.

1.

A bináris kódban általánosan az A,B,C,D betűkkel jelöljük az egyes helyi értékeket, az $\mathbf{A \div 2^0, B \div 2^1, C \div 2^2, D \div 2^3, \ldots}$ stb. súlyozás választásával. Az igazságtáblázatból felírhatjuk a következő logikai függvényeket:

$$\mathbf{K_0 = \overline{A}\,\overline{B}\,\overline{C}} \qquad \mathbf{K_1 = A\overline{B}\,\overline{C}}$$

$$\mathbf{K_2 = \overline{A}B\overline{C}} \qquad \mathbf{K_3 = AB\overline{C}}$$

$$\mathbf{K_4 = \overline{A}\,\overline{B}C} \qquad \mathbf{K_5 = A\overline{B}C}$$

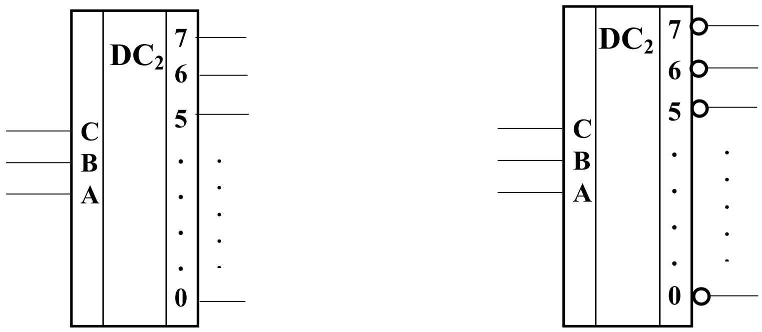
$$\mathbf{K_6 = \overline{A}BC} \qquad \mathbf{K_7 = ABC}$$

Az áramkör a mintermeket megvalósító **ÉS** kapukból és a változók tagadott értékeit előállító **i n v e r t e r** -ekből áll.

A 0 értékű aktív kimeneti logikai szintnél az előző összefüggések tagadásával kapjuk a logikai egyenleteket. Az áramkör, pedig **NAND** kapukkal épül fel.

A dekódolandó bináris kód bitjeinek növelésével - az előbbiekben elemzett mindkét változatnál - a felhasznált kapuk és azok bemeneteinek száma növekszik.

A dekódolók szimbolikus jelölése látható a 39. ábrán. A 0 -val aktív kimenetet a karika (tagadás) jelzi a b. ábrán.



a. b.

64. ábra

BCD dekódoló

A digitális áramköri készletek többségében van BCD decimális dekódoló is. A négy bites **BCD** kód (8 4 2 1 súlyozású), és a tíz decimális számértéket a bináris kód első tíz (K0 . . . K9) kombinációjához rendeli.

A BCD dekódoló áramköri kialakításánál egyszerűsítésre felhasználhatók a kódban elő nem forduló (K10 . . . K15) kombinációk is. A legegyszerűbb felépítésű BCD dekóder logikai függvényei a következők:

$$\mathbf{K_0 = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \quad K_1 = A\overline{B}\,\overline{C}\,\overline{D}}$$

$$\mathbf{K_2 = \overline{A}B\overline{C} \quad K_3 = AB\overline{C}}$$

$$\mathbf{K_4 = \overline{A}\,\overline{B}C \quad K_5 = A\overline{B}C}$$

$$\mathbf{K_6 = \overline{A}BC \quad K_7 = ABC}$$

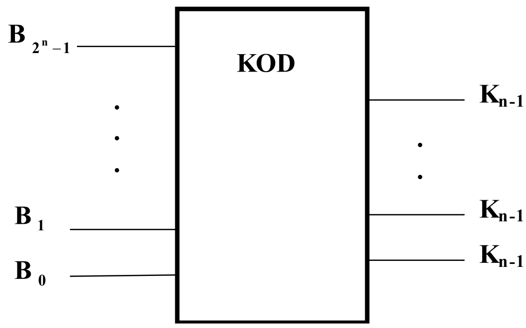
$$\mathbf{K_6 = \overline{A}D \quad K_9 = AD}$$

Az ilyen megoldás ***nem teljesen dekódolt***, mivel a bemenetre adott tiltott kombinációk is aktiválhatnak kimenetet, (esetleg kimeneteket). Pl.: a DCBA kombináció hatására a K7 kimenet lesz akti - 1 szintű.

- **Kódoló *áramkörök***

A kódolás során **1 az N** - ből kódot (pl. 1 a 10-ből a decimális kód) kívánunk átalakítani **bináris, BCD** vagy **egyéb** kóddá. Ezt a feladatot megvalósító kombinációs hálózat a **kódoló**.

A kódolás tulajdonképpen a dekódolás duálja. A kódoló blokksémája a 40. ábrán látható. A maximálisan $\mathbf{2^n}$ számú bemenet ($\mathbf{B_0 \cdots B_{2^n-1}}$) egyikére jut csak aktív logikai szint (1 vagy 0), s ennek alapján állítja elő az **n** db kimeneten ($\mathbf{K_0 \cdots K_{n-1}}$) a megfelelő **n** bites bináris kódot.



65. ábra

Vizsgáljuk meg az egyik leggyakrabban használt kódoló áramkör, a decimális - BCD átalakító logikai függvényét és megvalósításának lehetőségeit. A 41.ábrán látható a kódolási feladat igazságtáblázata. A kimenetek jelölésére a szabványos A,B,C,D betűket használtuk.

A táblázat alapján felírhatók az egyes kimeneteket megvalósító logikai függvények.

$$\mathbf{A = B_1 + B_3 + B_5 + B_7 + B_9}$$
$$\mathbf{B = B_2 + B_3 + B_6 + B_7}$$
$$\mathbf{C = B_4 + B_5 + B_6 + B_7}$$
$$\mathbf{D = B_8 + B_9}$$

B_9	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	D	C	B	A
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	1	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

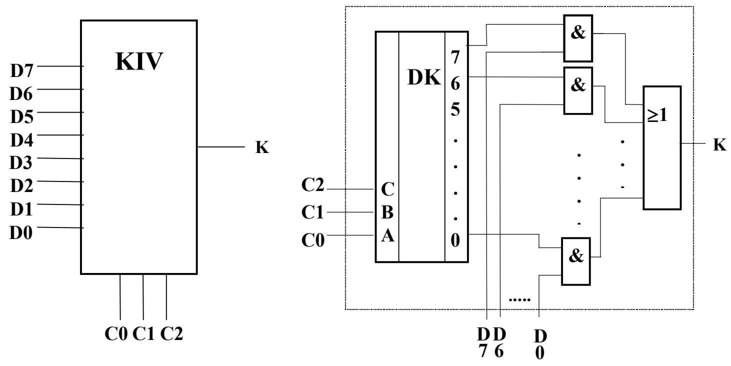
66. ábra

A logikai függvények ismeretében megtervezhető a kódolást megvalósító kombinációs hálózat.

A kódolók kitüntetett alkalmazási területe az adatbevitelre szolgáló billentyűzet (klaviatúra) és a digitális berendezés illesztése. Viszonylag egyedi felhasználásúk miatt integrált áramköri kialakításban nem készítenek ilyen kódolót. Diszkrét elemekből. IC kapukból könnyen megépíthetőek.

- ***Kiválasztó* áramkörök (multiplexerek)**

A **kiválasztó** áramkör (adatszelektor) az **adat-**bemenetek (**D_1 . . D_p**) egyikének információját kapcsolja a **Q** kimenetre. A kiválasztást az **n** darab**címző** (kiválasztó) bemeneten (**C_0 ... C_{n-1}**) érvényes bináris kód határozza meg. (Az **n** bittel címezhető adatbemenet maximális száma **2^n** .) Elvi blokkvázlata a 42.a.ábra szerinti.



a.

b.

67. ábra

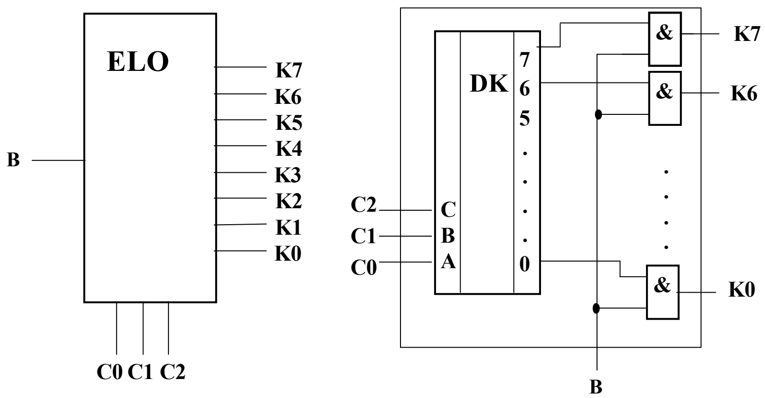
Egy n = 3 címző bemenetű multiplexer 8 adatból (**2^3**) választ ki egyet. Ennek logikai függvénye a következő:

A függvény minden egyes logikai szorzatában szerepel valamelyik adat (**D_0 ... D_7**) és a címző bitek (**C_0,C_1,C_2**) egyik kombinációja, mintermek (a zárójelbe tett mennyiségek), amelyek kimenetei közül egyidejűleg csak egyik lehet logikai 1 értékű. Ezért a **Q** kimenetre az **adat-bit** (**D_i**) jut, amelyhez tartozó címző variáció értéke1.

A függvénykapcsolat megvalósítható a címző **C_0,C_1,C_2** kódot **dekódoló** áramkörből és egy **ÉS-VAGY** hálózatból. Ennek logikai vázlat t mutatja a 42.b. ábra.

- ***Elosztó* áramkör (demultiplexer)**

Az adatelosztásra alkalmazható demultiplexer egyetlen adatbemenetről osztja szét az információt **2^n** számú kimenetre, ahol **n** a címző (elosztó) bemenetek száma.Az áramkör elvi blokkvázlata a 43.a.ábrán látható.



a.

b.

68. ábra

A függvényeket megvalósító hálózat felépíthető a címző bemeneteket dekódoló áramkörből, és ennek kimeneteit a D adattal kell kapuzni. A megvalósítás logikai vázlata látható a 43.b.ábrán.

Az elosztási feladat logikai függvényei n=3 esetén a következők:

$$\mathbf{K_0 = D(\overline{C}_2\overline{C}_1\overline{C}_0)} \qquad \mathbf{K_4 = D(C_2\overline{C}_1\overline{C}_0)}$$

$$\mathbf{K_1 = D(\overline{C}_2\overline{C}_1C_0)} \qquad \mathbf{K_5 = D(C_2\overline{C}_1C_0)}$$

$$\mathbf{K_2 = D(\overline{C}_2C_1\overline{C}_0)} \qquad \mathbf{K_6 = D(C_2C_1\overline{C}_0)}$$

$$\mathbf{K_3 = D(\overline{C}_2C_1C_0)} \qquad \mathbf{K_7 = D(C_2C_1C_0)}$$

(A zárójelekbe tett kifejezések a dekódoló kimeneteinek a függvényei).

A demultiplexer kapuzott dekódolóként is alkalmazható, mivel a D bemenet 0 értékénél – a címző bemenetek vezérlésétől függetlenül – mindegyik kimenet 0 szintű lesz.

- ***Aritmetikai műveletek megvalósítása***

A digitális számítógépekben, műszerekbe, vezérlő egységekben stb. végzendő ***számítási*** műveletek ***bináris számrendszerben*** történik. A kettes számrendszer ***alapműveletei***, az ***összeadás***, ***kivonás***, ***összehasonlítás*** logikai műveletekkel elvégezhetőek. Az ***aritmetikai*** műveletvégző egységek ***kombinációs*** logikai ***hálózatokkal*** megvalósíthatóak. Itt ismertetjük a ***teljes összeadó-kivonó*** (TAK), és a két-bites ***nagyság-komparátor*** logikai felépítését, működését.

- **Egy helyértékű összeadó-kivonó egység**

Az összeadásnál, és a kivonásnál - bármelyik számrendszerben – helyértékenként kell a műveletet elvégezni. Az teljes ***összeadást***, vagy a ***kivonást*** az ***adott helyértékű*** két számjegye és az ***előző helyértéken*** keletkezett átvitel, illetve áthozat értékével, kell elvégezni. (teljes jelző utal arra, hogy az előző helyértéken keletkező túlcsordulással – átvitellel, áthozattal - is végzünk műveletet). Az ***összeadás*** eredményei az ***összeg*** (**S** - summa) és az ***átvitel*** (**C** - Carry), míg a ***kivonásnál*** a ***különbség*** (**D** - different) és az ***áthozat*** (**B** - Borrow).

A ***bináris számrendszerben*** mind a tényezők helyértékein a számjegyek mind, pedig a műveletek eredménye ***0***, vagy ***1*** lehet. Írjuk fel mindkét művelet értéktáblázatát (44. ábra). Az egyes tényezőket – mindkét műveletnél – jelöljük **X**, illetve **Y**, míg az előző helyérték átvitelét **C***, áthozatás **B*** betűvel.

Összeadás (S=X+Y)

X	Y	C*	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1

Kivonás (D=X-Y)

X	Y	B*	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0

1	1	1	1	1

1	1	1	1	1

69. ábra

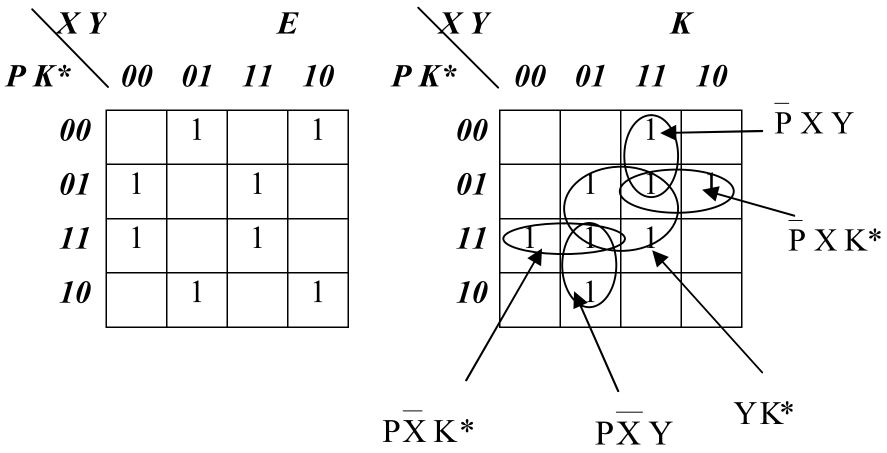
Az értéktáblázatok megegyeznek egy kombinációs hálózat igazságtáblázatával, amiből következik, hogy ezeket az ***aritmetikai*** műveleteket ***kombinációs hálózattal*** meg is lehet valósítani. A két táblázatot összehasonlítva azt látjuk, hogy a két műveletnél az összeg, illetve a különbség azonos értékeket ad, és csak az átvitel, illetve áthozat különbözik. A táblázatok megvalósítása adja a ***teljes összeadó*** (**TA**), illetve ***teljes kivonó*** (**TK**) áramköröket. Ezek logikai vázlatát most nem rajzoljuk meg.

A két művelet egyetlen hálózattal, az un. ***teljes összeadó-kivonó*** (**TAK**) áramkörrel is megvalósítható. Az áramkörnek ***négy*** bemenete, és ***két*** kimenete van. Bemenetek az **X, Y** jelzésű – azonos helyértékű - bitek, amelyekkel a kijelölt műveletet kell elvégezni, az előző helyértéknél keletkezett átvitelt - áthozatot adó **K*** jelű ***kiegészítő*** - bit, valamint a P műveleti parancs. A ***P=0*** értéknél ***összeadást***, míg a ***P=1*** értéknél ***kivonást*** végez az áramkör. A parancs értékétől függően az **E** jelű - ***eredményt*** adó - kimeneten az összeg (S), vagy a különbség (D), az **K** jelű – ***kiegészítő*** – kimeneten, pedig az átvitel(C), illetve az áthozat (B) értékét kapjuk. Az áramkör igazságtáblázata látható 45. ábrán.

	P	**X**	**Y**	**K***	**E**	**K**
összeadás	0	0	0	0	0	0
	0	0	0	1	1	0
	0	0	1	0	1	0
	0	0	1	1	0	1
	0	1	0	0	1	0
	0	1	0	1	0	1
	0	1	1	0	0	1
	0	1	1	1	1	1
kivonás	1	0	0	0	0	0
	1	0	0	1	1	1
	1	0	1	0	1	1
	1	0	1	1	0	1
	1	1	0	0	1	0
	1	1	0	1	0	0
	1	1	1	0	0	0
	1	1	1	1	1	1

70. ábra

Végezzük el a két kimenetre – **E**, illetve **K** – a lehetséges egyszerűsítést Kp diagram használatával (46. ábra).



71. ábra

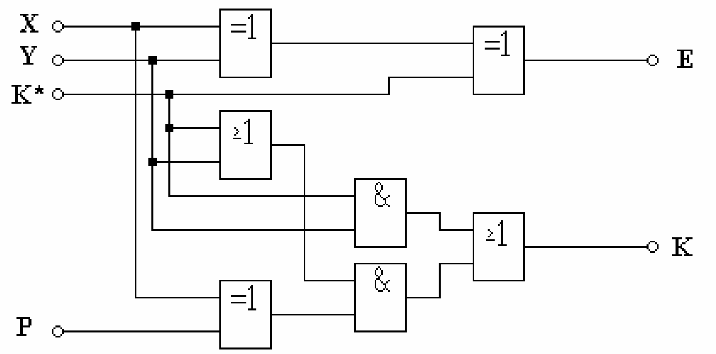
Az ***S/D*** (összeg - különbség) kimenetre felírt K-táblázatból is eldönthető, hogy az a P ***parancstól független*** és a három aritmetikai változó ***moduló összege*** adja az eredményt. A kimenet logikai függvénye:

$$\mathbf{S/D = X \oplus Y \oplus C/B_{-1}}$$

A ***C/B*** (átvitel – áthozat) kimenet logikai függvénye a ***Kp diagramból*** felírva az alábbi:

$$
\begin{aligned}
C/B &= \overline{P}XY + \overline{P}XK^* + YK^* + P\overline{X}Y + P\overline{X}K^* = \\
&= \overline{P}X(Y+K^*) + P\overline{X}(Y+K^*) + YK^* = \\
&= (\overline{P}X + P\overline{X})(Y+K^*) + YK^* = \\
&= (P \oplus X)(Y+K^*) + YK^*
\end{aligned}
$$

A teljes összeadó-kivonó áramkör logikai vázlatát mutatja 47. ábra.



72. ábra

- ***Kétbites* nagyság-*komparátor***

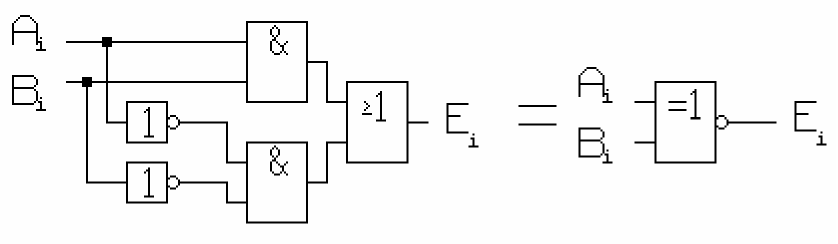
Nagyság-komparátornak nevezzük azt az áramkört, amely két bináris számot hasonlít össze, és kimenetein jelzi a számok közötti ***relációkat*** (***egyenlő***, ***kisebb***, ***nagyobb***).

Az összehasonlítás az összetartozó - azonos nagyságrend – bit-párok relációjának megállapításán alapul.

Két bit (A és B) ***egyenlőségét*** az

$$\mathbf{E_i = A_i B_i + \overline{A_i}\,\overline{B_i}}$$

(equivalencia) írja le. Az áramkör logikai vázlata a 48. ábrán látható, amely a kizáró-vagy tagadása logikai függvény



73. ábra

Több ***bites szám*** akkor ***egyenlő***, ha az ***azonos helyértékű*** bitek egyenlők. Legyen a a két szám:

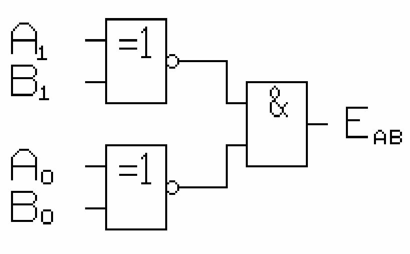
$$\mathbf{Z_A = A_1 2^1 + A_0 2^0}$$

$$\mathbf{Z_B = B_1 2^1 + B_0 2^0}$$

Az egyenlőséget leíró logikai függvény:

$$\mathbf{E_{AB} = (A_1 B_1 + \overline{A_1}\,\overline{B_1})(A_0 B_0 + \overline{A_0}\,\overline{B_0})}$$

A függvényt megvalósító áramkör logikai vázlata a 49.ábra szerinti.



74. ábra

További bővítés az előzőek ismétlésével történik.

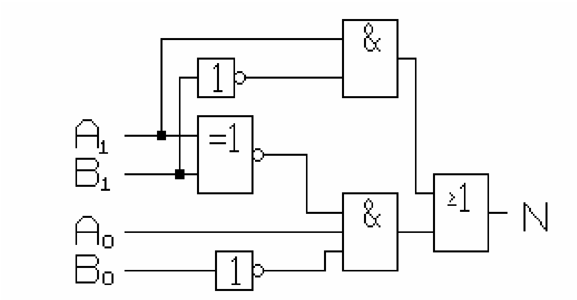
Két szám összehasonlításánál gyakran feladat - az egyenlőség jelzése mellett - a ***kisebb***, ill. ***nagyobb*** viszony kijelzése is.

A következőkben vizsgáljuk meg - két-bites számok összehasonlításánál - az egyenlőtlenségi relációkat jelző áramkörök működési feltételeit és határozzuk meg a logikai függvényeket.

A $\mathbf{Z_A > Z_B}$ akkor igaz, ha $\mathbf{A_1 > B_1}$, ill. ha $\mathbf{A_1 = B_1}$ és $\mathbf{A_0 > B_0}$. A leírt feltétel teljesülését az **N** logikai változó jelölje. Logikai függvényben ez a következőképpen fogalmazható meg:

$$\mathbf{N = A_1\overline{B_1} + (A_1B_1 + \overline{A_1}\,\overline{B_1})A_0\overline{B_0}}$$

A függvény első logikai ÉS kapcsolata fejezi ki az $\mathbf{A_1>B_1}$ feltételt, ugyanis csak az $\mathbf{A_1=1}$ és $\mathbf{B_1=0}$ esetén ad 1 értéket. A zárójeles rész az $\mathbf{A_1=B_1}$ feltételt teljesíti, míg az $\mathbf{A_0\overline{B_0}}$ tag az $\mathbf{A_0 > B_0}$ relációt adja. A függvényt megvalósító áramkör logikai vázlata látható az 50. ábrán.



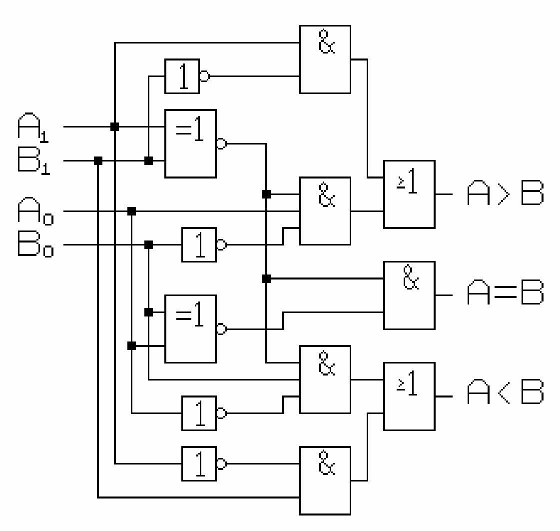
75. ábra

A $\mathbf{Z_A < Z_B}$ reláció logikai függvénye következik az előzőből, ha értelemszerűen felcseréljük a megfelelő biteknél a tagadást. Ezt a

$$\mathbf{K = \overline{A_1}B_1 + (A_1B_1 + \overline{A_1}\,\overline{B_1})\overline{A_0}B_0}$$

logikai függvény fejezi ki.

A kétbites számok teljes összehasonlítását végző komparátor logikai vázlata a 51.ábrán látható.



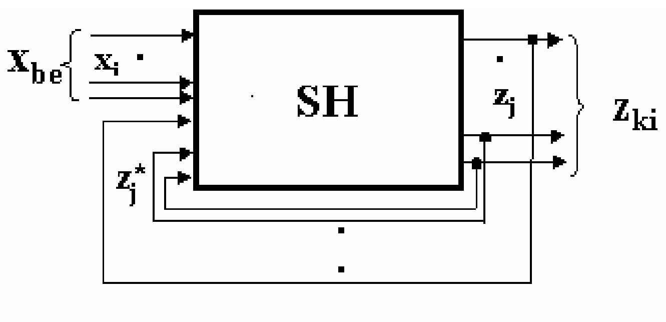
76. ábra

4.2. A sorrendi hálózatok

A logikai feladatok jelentős hányadánál – mint ahogyan ezt már az előzőekben leírtuk - a ***következtetések*** értéke - az éppen teljesülő ***állítások*** (feltételek) mellett – a következtetések ***megelőző*** értékétől is függ. Miután egy feladatban ***állítások sorozata*** követheti egymást, ezért a ***következtetések*** is jól meghatározható ***sorozatot*** alkotnak. Ezek a **sorrendi** vagy **szekvenciális** logikai feladatok. Természetesen csak az olyan feladatok valósíthatók meg egyértelműen, amelyeknél egy új következtetést mindig egy megváltozott állítás „indít".

Sorrendi feladatokat megvalósító logikai hálózat alapvetően két különböző módon építhető fel.

Az 1.ábra szerinti blokkvázlat szerinti felépítés az ***alapdefiníciónak*** felel meg, mely szerint a sorrendi logikai hálózat (**SH**) az új ***következtetéseket*** megadó ***kimeneti jelek*** ($\mathbf{Z_{ki}}$) értékkombinációját az éppen ***érvényes*** állításokból adódó ***bemeneti jelek*** ($\mathbf{X_{be}}$), valamint ***kimeneti jelek*** ($\mathbf{Z_{ki}}^*$) értékkombinációjából határozzák meg. (A kimeneti jeleknél a *-al azt jelezzük, hogy állapotváltozáskor az előző állapothoz tartozó kimeneti jel. A $\mathbf{Z_{ki}}$ és a $\mathbf{Z_{ki}}^*$ jelkombinációk a változáskor különböző értékek, viszont stabil - állapotban azonos értékek.).



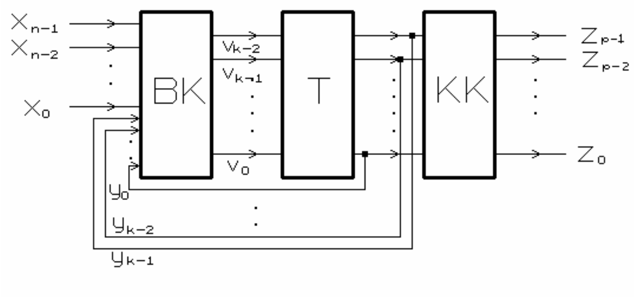
77. ábra

A kimeneti jelek ***visszavezetése*** következtében az új állapotkor létrejövő kimeneti változók kombinációi függnek a bemeneti-, és az előző állapot kimeneti kombinációitól. A meghatározást az alábbi összefüggés írja le:

$$\mathbf{Z_{ki} = f_z\,(X_{be}, Z_{ki}^{\,*})}$$

ahol $\mathbf{f_z}$ a kimenetek, és a bemeneti-, valamint előző állapot közötti logikai kapcsolatot (logikai függést) adja meg. Gondolati kísérlettel belátható, hogy a 1.ábra szerinti hálózat csak akkor stabil, ha $\mathbf{Z_{ki}}^* = \mathbf{Z_{ki}}$ és a bemeneti változók is **állandósultak**. Ez csak - egy bemeneti kombinációváltást követően - késleltetve, legkevesebb a hálózat ($\mathbf{t_H}$) késleltetési ideje múlva következhet be.

A feladatok megvalósíthatóak úgy is, hogy egy ***tároló*** egység „emlékszik" az érvényes állapotra. A ***kimenetek*** jeleit - egy kombinációs hálózaton keresztül (***KK***) - a tároló kimeneti jelei – y_i ***állapotjelek*** határozzák meg. A bemeneti értékek változása és a tárolt állapotjelek csak együtt – ugyancsak egy kombinációs hálózaton keresztül (***BK***) – állítják elő a vezérlő v_j vezérlőjeleket, amelyek meg változtathatják a tárolók állapotát, és ezáltal hoznak létre a kimeneteken új jeleket. A leírt megoldás blokkvázlata látható a 2. ábrán.



78. ábra

Az állapotjeleket **y** -al, míg a tárolók vezérlőjeleit **v** -vel jelöltük.

Az alapdefiníciónak megfelelő függvénykapcsolat ekkor is érvényesül, mivel

$$\mathbf{Y_i = f_b\,(X_i, Y_i^*)}$$

$$\mathbf{Z_{ki} = f_z\,(Y_i)}$$

ahol $\mathbf{Y_i}$ a tárolók kimeneti jeleinek (y_i) aktuális kombinációja. Kiolvasható, hogy a kimeneti jelek ***aktuális kombinációja*** ($\mathbf{Z_{ki}}$) az $\mathbf{Y_i}$ –től függ. Ennek értéke viszont ez az $\mathbf{X_i}$ ***bemeneti jelkombináció***, és az ***előző állapotjelek*** ($\mathbf{Y_i^*}$), végeredményben, pedig az ***előző kimeneti kombinációk*** ($\mathbf{Z_{ki}^*}$) függvénye.

Az állapottárolókkal való megoldást olyan feladatok esetében célszerű alkalmazni, amelyeknél több kimenet van, mint ahány állapottároló-elemet (flip-flop -ot) kell felhasználni.

- **Aszinkron, és szinkronműködés**

Két **stabil** (állandósult) állapot között a kimeneti jelek – ***átmenetileg***, a belső késleltetésektől függően – ***több állapotkombinációt*** is felvehetnek, mielőtt állandósulna a ***kívánt új jelkombináció***. Az ***állapotváltozást*** vagy azok sorozatát a bemeneti jelek változása ***elindítja***, de a tranziensváltozásokat a visszacsatolás jeleinek (állapotjel) változása eredményezi. Az ilyen működésű hálózatot ***aszinkron sorrendi*** hálózatnak nevezzük. A következőkben az aszinkron magoldásnál csak számlálókat tárgyaljuk röviden.

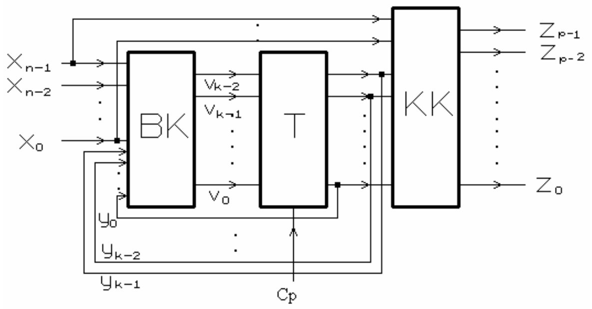
Sorrendi hálózat kialakítható olyan működéssel is, amelynél az egymást követő ***állapotváltozásokat*** az **x** bemeneti jelek megváltozása csak előkészíti, és egy ütemező jel, az un. ***szinkronozó*** („órajel”) jel hajtja végre.

A két szinkronjel közötti időben, ***tárolni*** kell az állapotra jellemző információt. Ezek az ***állapotjelek*** (szekunder változók). Az aktuális ***bemeneti*** jelek, és a előző jel hatására tárolt ***állapotjelek*** előkészítik a kívánt új állapot vezérlőjeleit, és a következő szinkronjel fogja ezt az tároló egységbe beírni. ***Lényeges, hogy a szinkronozó jel aktív ideje alatt a bemeneti-, és az állapotjelek értéke ne változzon!***

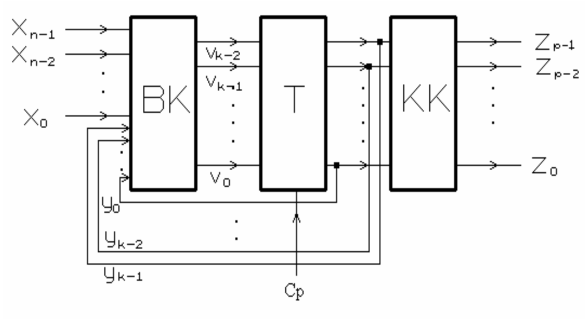
- ***Szinkron sorrendi hálózat rendszertechnikai felépítése.***

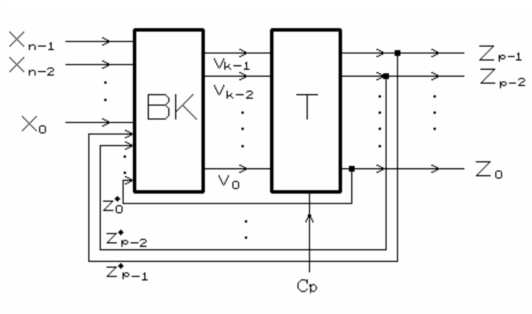
A leírtak szerint működő ***szinkron sorrendi hálózat***, különböző felépítés szerint célszerű megvalósítani (2. ábra). Mindhárom változatban a **T** ***állapottárolók*** új vezérlőjeleit (v_j) a ***bemeneti kombinációs*** hálózat (**BK**) állítja elő az x_n ***bemeneti-***, és az y_k ***állapotjelekből***. Az állapottárolókat az ütemező jel (**Cp**) billenti a v_j által meghatározott ***új állapotba***.

Az a. ábra szerinti megvalósításban a kimenetek jeleit (z_p) egy kombinációs hálózat (**KK**) az ***állapotjelekből*** és közvetlenül a ***bemeneti*** jelekből, ***vagy*** azok egy ***részéből*** állítja elő. Ez a megoldás az un. ***Mealy- modell***.



a.





b.
c.

79. ábra

A b. ábra szerinti változatban a ***kimenetek jeleire*** csak a tárolt ***állapotjeleken*** keresztül hatnak a ***bemeneti jelek***. A kimeneti kombinációs hálózat (**KK**) csak az állapotjelekből (y_k) állítja elő a hálózat kimeneti jeleit, z_p -ket. E változat az un. **M o o r e - modell**. Az utóbbi megoldásnál esetleg több tárolóra van szükség, de egyszerűbb a felépítés. A korszerű integrált áramkörök alkalmazásával már elhanyagolandó szempont lett a tárolók száma (különösen a programozott rendszerekben), s ezért a Moore modell szerinti felépítés mind hardverben, mind pedig a szoftveres megoldásban nagyobb teret kap.

A Moore - modell egy változatát mutatja a c. ábra. Itt nem használunk külön kimeneti kombinációs hálózatot, hanem az y_k állapotváltozókat állítjuk elő oly módon, hogy azok, vagy egy részük egyúttal a hálózat kívánt kimeneti változói is. Elsősorban a számlálóknál találkozunk ezzel a változattal.

Mindhárom felépítésű megoldásban a hálózat állapota, s így a kimenő jelek is az ***szinkronozó-jel*** (Cp) ütemezésében váltanak értéket.

Tételezzük fel, hogy a vizsgált $\mathbf{t_i}$ időpillanatban - amely a két ***szinkronozó-jel*** közötti időpont - a hálózati tranziensek lejátszódtak, a hálózat állapotát és a kimeneti értékeket a $\mathbf{Z_i}$ kimeneti jelkombináció írja le. Ugyanebben az ***előkészítési fázisban*** az új bemeneti jelkombináció állandósult értéke $\mathbf{X_i}$. Ekkor a **BK** állítja elő a **T** tárolók $\mathbf{v_{ji}}$ vezérlőjeleit bemeneti kombinációs hálózat bemenetén lévő $\mathbf{X_i}$ és $\mathbf{Y_i = Z_i}$ bemeneti értékekből. A $\mathbf{t_{i+1}}$-edik időpontban érkező szinkronozó-jel fogja - a $\mathbf{v_{ji}}$ által meghatározott állapotba - billenteni a tárolókat. Ennek eredményeként alakul ki az új (Z_{i+1}) kimeneti jelkombináció, ami egyúttal a következő mintavételezéshez tartozó állapotjel is. Az előzőek alapján felírhatjuk a

$$\mathbf{Z_{i+1} = fz(v_{ji})}$$

$$\mathbf{v_{j_i} = fv(X_i, Z_i)}$$

függvénykapcsolatokat. Az **fz** kimeneti függvény az előállítani kívánt kimeneti (Z_{i+1}) és a tárolókat vezérlő $\mathbf{v_{ji}}$ jelek - billentés előtti - értékei közötti logikai kapcsolatot adja meg.

- ***Sorrendi feladatok logikai leírása***

A sorrendi logikai feladatokat megvalósító sorrendi hálózatok tervezéséhez szükségünk van a kívánt működést egyértelműen megadó, az ismert hálózattervezési módszerek alkalmazását elősegítő leírásra. A leírás egyértelműen kell meghatározza (jelölje)

az állandósult állapotokat,

az állapotátmeneteket indító bemeneti jelkombinációkat,

az állapotátmenetek irányát,

a kimeneti jelek állandósult értékeit.

A továbbiakban röviden ismertetünk egy-egy, ***a logikai kapcsolatokat***

állapotgráf -al jelölt grafikus,

az ***állapottáblázat*** -ba foglalt táblázatos,

a kimenetek ***állapotfüggvényét*** megadó algebrai, és

a ki-, valamint bemeneti jelek időbeli változását mutató ***ütemdiagram*** (állapot-diagram)grafikus

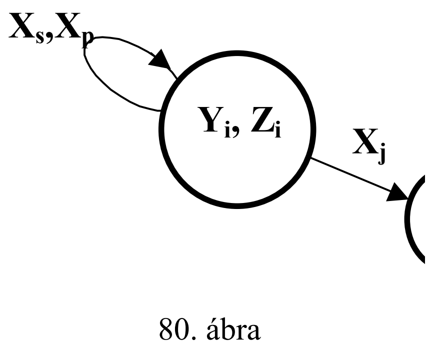
feladat-leírási módszert.

- ***Állapotgráf***

A sorrendi logikai feladatokhoz gyakran használják az **állapot-gráfnak** nevezett szemléltető leírást, amelynek elemeit mutatja az 55. ábra.

A hálózat minden ***állandósult állapotát*** egy **körrel -** gráf-csomópont – jelöljük. A körökbe az adott állapothoz tartozó ***állapot-*** (***Yi***), és a ***kimeneti*** (***Zi***) változók kombinációját írjuk.

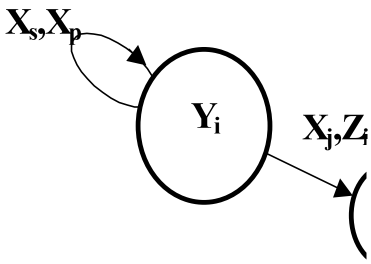
A körökből **nyilak** indulnak ki, amelyek vagy egy ***másik***-, vagy az ***induló*** körnél (állapotnál) végződnek. Ezek jelzik a lehetséges állapotátmeneteket, és irányukat. A nyilakra ráírjuk az ***állapotváltozást*** elindító bemeneti kombinációt ($\boldsymbol{X_j}$).



80. ábra

Minden körtől annyi nyíl indul, amennyi az ***állapotváltozást okozó bemeneti kombinációk*** száma. Ha több kombináció eredményez azonos állapotátmenetet, akkor azokat egyazon nyílra írjuk. Az ábrán pl. az $\mathbf{X_s}$**,** $\mathbf{X_p}$ jelű bemeneti kombinációk nem indítanak állapotváltozást, míg az $\mathbf{X_j}$ egy másik állapotot jelző körig tart . A nyíl irányítása adja meg az állapotváltozás irányát.

A bemutatott állapotgráf részlet olyan megoldásra utal, amelynél a ***kimenetek*** kombinációját csak az ***állapotjelek*** határozzák meg (Moore modell), és ezért a körbe írjuk a kimeneti kombináció (***Z_i***) jelét. Amikor a kimeneti kombinációt az állapotváltozást jelző nyílra írjuk, azt jelezzük, hogy a kimeneti ***értékváltozást*** már a ***bemeneti jelváltozás*** indítja (Mealy modell) .(56.ábra).



81. ábra

Egy sorrendi feladatot leíró állapot-gráf felrajzolásához ismernünk kell:

az állandósult ***állapotok számát***, vagyis a ***modulust*** (m),

a szükséges ***kimeneti*** jelek ***kombinációját***,

az állapotváltozásokat ***eredményező bemeneti*** jelkombinációkat.

A leírt kiinduló adatok ismeretében az alábbi lépésekben kell megrajzolni az állapotgráfot:

megrajzoljuk az ***m*** darab ***állapotcsomópontot*** (kört),

beírjuk a körökbe az ***állapotjellemző*** (Y) jelét, és indexét, valamint a ***kimeneti*** jelkombináció (Z) jelét és indexét

körökre rárajzoljuk a ***visszatérő nyilat***, és arra felírjuk azokat a ***bemeneti*** jelkombinációk (X) jeleit, és indexeit, amelyek az adott állapotot ***nem változtatják*** meg,

csompontonként felrajzoljuk az ***állapotváltozást*** jelentő nyilakat, és azokra felírjuk a ***kiváltó*** bemeneti jelkombináció jelét, és indexét.

- ***Állapottáblázat***

Egy sorrendi hálózat állapotai, valamint a bemeneti-, és kimeneti változói közötti kapcsolatrendszert táblázattal, az un ***állapottáblázattal*** is megadhatjuk. A táblázat minden egyes sora egy állandósult állapotot jelent, és ezt az $\mathbf{Y_i^*}$ ***állandósult állapotváltozóval*** jelöljük. Az oszlopok a lehetséges állapotváltozásokat okozó bemeneti jelkombinációkat $\mathbf{X_k}$ jelentik. A táblázat celláiba kell beírni, hogy az adott állapotból (sor) milyen ***új állapotba*** $\mathbf{Y_j}$ viszi a hálózatot az oszlop által jelölt bemeneti jelkombináció. Ugyancsak a cellába kell jelölni, hogy milyen kimeneti kombináció $\mathbf{Z_s}$ érvényes az adott állapotban.

Áll. jell.	Bemeneti jelkombinációk		
	X_0		X_p
Y_0^*	Y_1, Z_0		Y_0, Z_0

Y_j^*	Y_0, Z_s		Y_j, Z_s

82. ábra

Az 57. ábra szerinti táblázat bal felső cellájában az $\mathbf{Y_1}$ bejegyzés azt jelenti, hogy ha a hálózat $\mathbf{Y_0}$ ***állapotban van*** és a ***bemeneti*** kombináció $\mathbf{X_0}$ ra vált, akkor a ***hálózat új állapota*** $\mathbf{Y_1}$ lesz. Amikor az állapotsor indexe megegyezik a cellába írt index-el, akkor – az oszlop szerinti bemeneti kombináció – ***nem okoz*** állapotváltozást (pl. az első sor utolsó cellája).A példa táblázatban a kimeneti kombinációk $\mathbf{Z_s}$ egy sorban azonosak, és azt jelzi ez, hogy értékét csak az aktuális állapot határozza meg (Moore modell). A Mealy modell szerinti megoldás állapottáblázatának egy sorában különböző kimeneti kombinációk is lehetnek.

A leírtak szerint az állapotgráf, és az állapottáblázat ugyanazt írja le. A táblázat ***sora*** felel meg a ***gráf-csomópontnak***, és az ***oszlopok*** jelentik a ***nyilakat***.

- ***Állapotfüggvény***

A sorrendi hálózatok minden kimenetére felírható egy algebrai alakú logikai függvény. Ezek a függvények abban különböznek a kombinációs feladatoknál megismert logikai függvényektől, hogy független változói között szerepelnek az állandósult kimeneti értékek is. Általánosan tehát a $\mathbf{Z_j}$ kimeneti kombináció függvénye az állandósult $\mathbf{Z_k}$ kimeneti-, és az $\mathbf{X_i}$ bemeneti kombinációknak.

$$\mathbf{Z_j = f(X_i, Z_k^*)}$$

Az adott összefüggést úgy is értelmezhetjük, hogy a $\mathbf{Z_j}$ kimeneti kombináció akkor következik be, ha a hálózat kimenetén $\mathbf{Z_k}$ kombináció van, és a bemenetekre az $\mathbf{X_i}$ jelkombinációra vált.

- ***Ütem- (állapot-) diagram***

Az első fejezetben már megismertük a logikai függvények idő-diagramban történő ábrázolását. Tulajdonképpen a sorrendi hálózatok be-, és kimeneti jelei is ugyanúgy ábrázolhatók az idő függvényében. Ránézésre nem állapítható meg azonnal, hogy kombinációs-, illetve sorrendi-hálózat jeleit látjuk-e. A lényeges eltérés, hogy egy kimeneti jel változását a bemeneti-, és a kimeneti jelek előző értékei együtt határozzák meg.

- ***A sorrendi hálózat áramköri megvalósítása***

Az előzőekben megismert feladat-leírási módszerek közül

- az ***állapotgráf***

  szemléletes, de csupán a feladat értelmezését segíti,

- az ***állapottáblázat***

  alapján az áramköri tervezést – a táblázat kódolása, és felbontása után - elvégezhetjük,

- az ***állapotfüggvény***

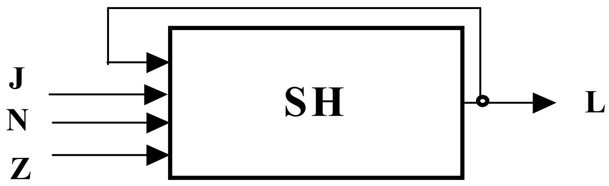
  segítségével, az esetleges algebrai egyszerűsítés után tervezhetjük meg az áramkört,

- az ***ütemdiagram*** -ot

  a PLC- megjelenése előtt főleg a relés vezérlések tervezésénél használták

A ***leírtak magyarázataként*** tervezzük meg egy autóbusz ajtajának nyitását kérő jelzőlámpa vezérlését. Az ***L*** jelű ***lámpa*** kezdjen világítani, ha az ajtó zárva van, és megnyomjuk a ***J*** jelű ***nyomógombot***. A világítás szűnjön meg, ha kinyílt az ajtó. Az ajtó zárt állapotát a ***Z*** jelű, míg nyitott állapotát az ***N*** jelű ***érintkezők*** zárása jelzi.

A feladat egy három bemenetű, és egy kimenetű sorrendi (emlékező) hálózattal valósítható meg, amelynek blokkvázlatát szemlélteti az 84.ábra.



83. ábra

Állapotok száma: ***m = 2***

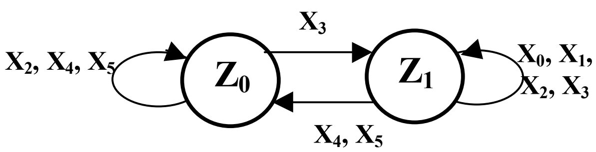
Kimenetek szám **1** (L), tehát két kimeneti kombináció van $\mathbf{Z_0}$ (L=0), **Z1** (L=1).

Bemenetek száma **3** (J,N,Z), tehát az alábbi nyolc bemeneti kombináció lehetséges.

	N	Z	J	Jelentés
$\mathbf{X_0}$	0	0	0	Ajtó közbenső helyzetben, jelzés nincs
$\mathbf{X_1}$	0	0	1	Ajtó közbenső helyzetben, jelzés van
$\mathbf{X_2}$	0	1	0	Ajtó zárt helyzetben, jelzés nincs
$\mathbf{X_3}$	0	1	1	Ajtó zárt helyzetben, jelzés van
$\mathbf{X_4}$	1	0	0	Ajtó nyitott helyzetben, jelzés nincs
$\mathbf{X_5}$	1	0	1	Ajtó nyitott helyzetben, jelzés van
$\mathbf{X_6}$	1	1	0	Nem fordulhat elő
$\mathbf{X_7}$	1	1	1	Nem fordulhat elő

Az $\mathbf{X_6}$ és $\mathbf{X_7}$ kombinációk azt jelentik, hogy mindkét érintkező zárt, ami viszont sohasem fordulhat elő.

Állapotgráf:



Állapottáblázat:

Z^*	Bemeneti kombinációk							
	$\mathbf{X_0}$	$\mathbf{X_1}$	$\mathbf{X_2}$	$\mathbf{X_3}$	$\mathbf{X_4}$	$\mathbf{X_5}$	$\mathbf{X_6}$	$\mathbf{X_7}$

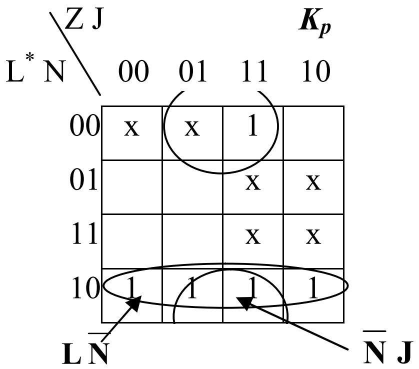
Z^*_0	x	x	Z_0	Z_1	Z_0	Z_0	x	x
Z^*_1	Z_1	Z_1	Z_1	Z_1	Z_0	Z_0	x	x

x – el jelöltük azokat a bemeneti kombinációkat, amelyek az adott állapotban nem fordulhatnak elő, csak hiba esetén. Pl. az Z^*_0 állapotban X_0 azért nem fordulhat elő, mert ebben a kombinációban az N, és Z érzékelők közül egyik sem zárt, amely csak az ajtó nyitása közben fordulhat elő. Ha nem világít a lámpa, akkor nincs ajtónyitás. Az X_6 és X_7 kombinációkról már írtunk.

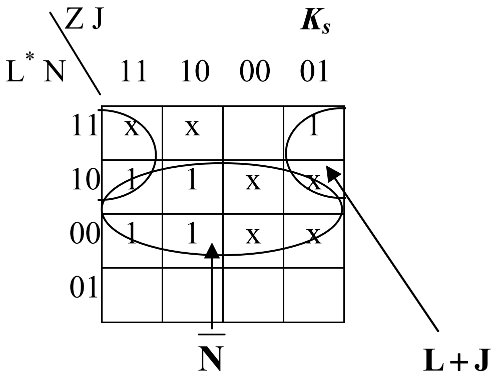
Kódolt állapottáblázat:

L^*	Bemeneti kombinációk							
	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
	N Z J	N Z J	N Z J	N Z J	N Z J	N Z J	N Z J	N Z J
	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0	x	x	0	1	0	0	x	x
1	1	1	1	1	0	0	x	x

Az L kimenetre érvényes Kp diagram:



Az L kimenetre érvényes Ks diagram:



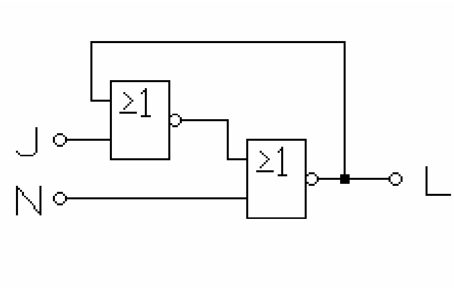
A kimenet állapotfüggvényei:

$$\mathbf{L = L\overline{N} + \overline{N}J = \overline{N}(L+J)}$$ (a Kp diagram alapján)

$$\mathbf{L = \overline{N}(L + J)}$$ (a Ks diagram alapján)

A két megoldás ugyanazt az eredményt adta.

Az áramkör logikai vázlata:



84. ábra

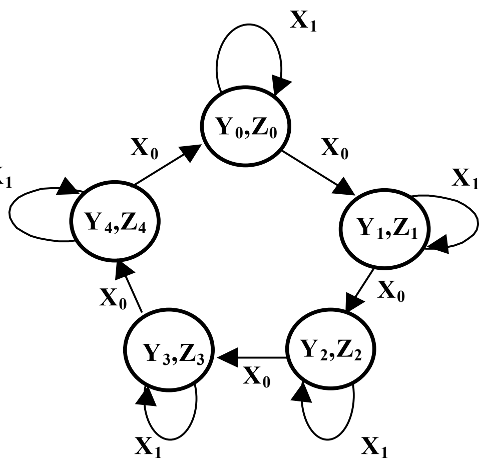
A példa megoldása a tároló alapáramkör az un. RS flip-flop. (A tárolók tárgyalásánál térünk vissza e megoldásra).

- ***Sorrendi hálózatok főbb típusai***

A sorrendi hálózatok egyik csoportosítása az ***állapot-sorozatok*** száma alapján is történhet. Beszélhetünk ***egy-***, és ***több-szekvenciájú*** sorrendi hálózatokról.

- Az **egy-*szekvenciájú*** hálózatban

az állapotok ***mindig ugyanabban a sorrendben*** követik egymást. A 59.ábrán egy öt állapotú – egy-szekvenciájú – sorrendi hálózat állapotráfja látható.



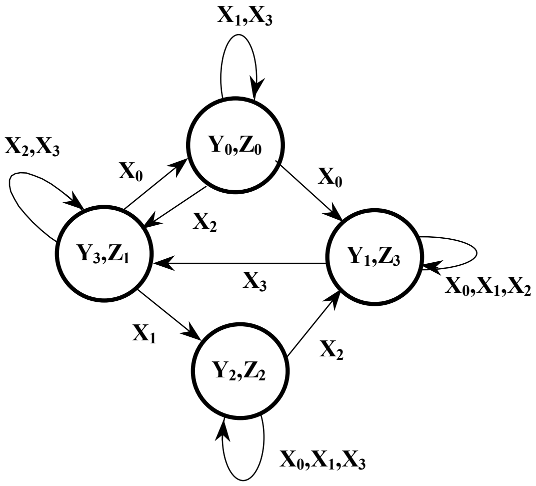
85. ábra

Itt az **....Y_0 – Y_1 – Y_2 – Y_3 – Y_4 – Y_0 - ......**állapotsor ismétlődik. Az **X_0** bementi jelkombináció indít minden állapotváltást.

A sorrendi hálózatok ilyen változatát ***lefutó típusú*** -nak is szokták nevezni.

- A **több-*szekvenciájú*** hálózat

  Olyan változat, amelyben ***több***, egymástól ***eltérő állapotsorozat*** is felléphet a különböző bemeneti jelkombináció-sorozat hatására. Egy négy állapotú ***általános*** – több szekvenciájú - sorrendi hálózat állapot-gráfja látható a 60.ábrán.



86. ábra

A példa szerint működő hálózatban lehetséges szekvenciák közül néhányat írtunk fel a következő sorokban.

....$Y_0 – Y_1 –Y_3 –Y_0$ - ......

....$Y_0 – Y_1 – Y_3 – Y_2 – Y_1 – Y_0$ - ......

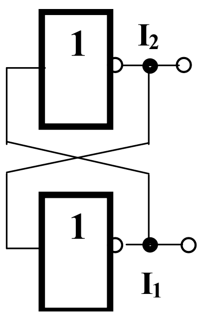
....$Y_0 – Y_3 – Y_2 – Y_1 – Y_3 – Y_0$ - ......

4.3. Sorrendi hálózatok alapelemei

A **sorrendi**, vagy más néven **szekvenciális** feladatok megvalósításához – az eddig megismert kapukon kívül - olyan elemekre is szükség van, amelyek az ***elemi információt – bit*** - et – ***tárolják.*** A következőkben ismertetjük a leggyakrabban alkalmazott tároló-elemek (**flip-flop**) felépítését, és működését.

➢ ***Tároló alapáramkörök***

A tároló alapáramkörök - flip-flop -ok - **két stabil** állapotú áramköri kapcsolások. A két stabil állapot **1 bit** információ **tárolására** teszi alkalmassá a flip-flop -ot.
A kétállapotú elem két keresztbecsatolt inverter -ből alakítható ki. (62. ábra)



87. ábra

A két inverter **keresztbe csatolása** biztosítja a felvett állapot tartását. Az ábra szerint elrendezésben a tápfeszültség bekapcsolása után – a két inverter kapcsolási sebességének különbözősége miatt - véletlenszerűen alakul ki a stabil helyzet. Ha az $\mathbf{I_1}$ jelű inverter kimenetén **1** szint lesz, az $\mathbf{I_2}$ bemenetére jutva biztosítja ennek a kimenetén a **0** szintet. A keresztbecsatolás fent tartja az $\mathbf{I_1}$ kimenetén az **1** szintet.

A fentiekben röviden elemzett áramkörnek nincs állapotváltozást **vezérlő** bemenete. A kívánt állapotváltozást a **vezérlő**-bemenetek és **billentési módok** különböző változataival lehet megoldani. A megoldási módozatok alapján csoportosítjuk a flip-flop -kat.

- ***Flip-flop típusok***

A flip-flop -ok két nagy csoportba sorolhatók annak alapján, hogy az **információ-** közlést és a **billentés**-t (az állapot beállítását) ugyanaz, vagy különböző jelek látják-e el. Ennek megfelelően:

közvetlen, **és**

kapuzott vezérlésű

tárolókat különböztünk meg.

A tárolandó értéket (információt) közlő bemenetek alapján leggyakrabban alkalmazott típusok az

RS,

JK,

T és

D típusú flip-flop -ok.

Az egyes flip-flop -ok **billentési módja** szerint megkülönböztetünk:

statikus és

dinamikus billentési megoldásokat.

A kapuzott vezérlésű tároló elemek között - elsődlegesen az integrált áramköri kialakításban - további két nagy csoport létezik, a

együtemű, és

kétütemű vezérlésű

áramköri változat. A kétütemű vezérlést ***közbenső tároló*** alkalmazásával valósítják meg.

Az ***aszinkron***, illetve a ***szinkron***-működési módot, már a flip-flop -ok esetében is értelmezhetjük. ***Aszinkron*** működésűnek nevezhetjük azokat a flip-flop - kat, amelyeknél a beírandó értéket (információt), valamint ennek beírását a tároló elembe ugyanazon jel végzi. Ilyenek a ***közvetlen*** vezérlésű tárolók. Értelemszerűen a ***szinkron***-működésű tárolóknál a két vezérlési funkciót – információ, és tárolást (billentést) – vezérlő jelek különbözőek. A ***kapuzott*** vezérlésű tárolok, alkotják ezt a csoportot.

A további tárgyalásoknál többször is beszélünk valamelyik bemenet ***aktív vezérlési*** szintjéről. A fogalom azt jelenti, hogy melyik az a logikai érték, amely a jelölt funkciót (beírás, törlés, billentés stb.) vezérli.

Az általános csoportosítás után először az információs (tárolandó értéket közlő) bemenetek alapján megkülönböztetett típusok elvi működését elemezzük.

Az **RS flip-flop** olyan tároló, amelynek két információt közlő bemenete van, amelyek közül az **S** jelű (set) a beíró és az **R** jelű (reset) a törlő bemenet. Ezek szerint a tárolt információ **1** lesz, ha a ***beíró*** bemenetre (S) érkezik ***aktív*** logikai szintű vezérlő jel, és **0** érték lesz, ha a ***törlő*** (R) bemenet kap ilyen vezérlést. A helyes működés feltétele, hogy a két vezérlőbemenet ***együttesen*** nem kaphat aktív vezérlést.

A **JK flip-flop** ugyancsak két információt közlő bemenete van. A **J** jelű bemenet ***beíró***, míg a **K** jelű a ***törlő*** feladatokra szolgál. A fentiekben ismertetett RS flip-flop -tól az különbözteti meg, hogy ***engedélyezett*** a J és K ***együttes*** aktív vezérlése is. Ebben az esetben a flip-flop a tárolt állapot ***ellenkezőjére*** (komplemens -ére) vált át.

A **T flip-flop** egyetlen vezérlőbemenettel rendelkező tároló elem. A **T** bemenetre jutó aktív vezérlés a tároló állapotát ***ellenkezőjére*** változtatja.

A **D flip-flop** –nak ugyancsak egyetlen vezérlőbemenete van. A tároló mindenkor a D bemenet logikai értékét tárolja, vagyis **D=0** esetén a tároló ***törlődik***, míg **D=1** értéknél ***beíródik***. A leírt működés alapján a tárolót ***adat*** flip-flop -nak is nevezik.

- ***Statikus billentésű flip-flop -ok***

Statikusnak nevezzük azt a billentési módot, melynél a vezérlőjel logikai szintje a hatásos. A flip-flop mindaddig vezérelt állapotban van, míg a bemeneten az aktív logikai szint érvényes. Aktív lehet a logikai 1 és a logikai 0 szint is. A kívánt aktív vezérlés kiválasztása után a megvalósítandó flip-flop ***működési***, vagy ***állapot táblázatából*** felírhatók működést leíró ***állapot-egyenletek***. Ezek alapján a szükséges vezérlési megoldás áramköri változata kialakítható.

A ***statikus billentésű*** - logikai 1 szinttel vezérelt - **RS** flip-flop állapottáblázata a 63. ábrán látható táblázat szerinti.

S_n	R_n	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	*
1	1	1	*

88. ábra

A táblázat oszlopai között a $\mathbf{Q_n}$ mint bemenő változó szerepel (a változás előtti állapot). A vezérlés utáni ***új állapotot*** ($\mathbf{Q_{n+1}}$) a ***vezérlés*** (R_n, S_n) mellett az ***előző állapot*** (Q_n) is befolyásolja. A *-al jelölt vezérlési kombinációk ***tiltottak***. A táblázatból felírható ***állapotfüggvények*** az alábbiak:

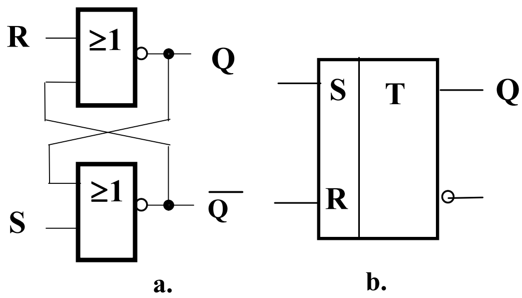
$$\mathbf{Q_{n+1} = S_n + \overline{R}_n Q_n}$$

$$\mathbf{S_n R_n = 0}$$

(Az összefüggésben és a továbbiakban is az **n** index a vezérlés időpontjára utal.)

Ezek a logikai függvények az **RS** flip-flop ***működését*** írják le. Az összefüggés szerint az új állapot (Q_{n+1}) 1 szintű lesz - az előző állapottól függetlenül - ha a beíró (S) bemenet 1 szintű. Ugyancsak 1 szintű lesz a kimenet, ha már a vezérlés előtti állapotban is Q_n = 1 és az R bemeneten 0 szint van. A második összefüggés a ***tiltott vezérlést*** adja meg. Eszerint a két vezérlő bemeneten, együttesen nem lehet 1 szint.

A fentiek szerint működő RS flip-flop két NOR kapuból alakítható ki a 64.a. ábra szerinti kapcsolásban. Az 1 szinttel vezérelhető RS flip-flop ***szimbolikus*** jele a b. ábra szerinti.



89. ábra

A kapcsolás működésének elemzése alapján könnyen belátható, hogy azért kell tiltani az együttes aktív vezérlést, mert ekkor mindkét kapu kimenete 0 szintű lesz. Az új állapot, pedig a vezérlőjelek megszűnésének sorrendjétől függ, ezért – legtöbbször - előre nem határozható meg.

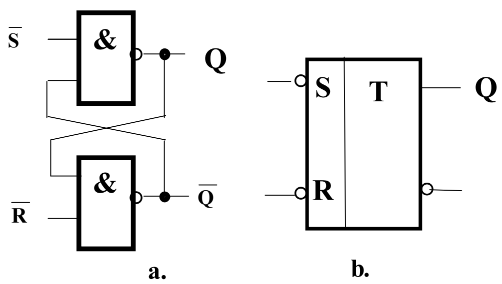
A **0** ***aktív vezérlési*** szintre billenő flip-flop működését az

$$\mathbf{Q_{n+1} = (\overline{S}_n + Q_n)R_n}$$
$$\mathbf{S_n + R_n = 1}$$

állapot egyenletek írják le .

Az összefüggés szerint az új állapot **1** lesz, ha a törlő bemenet (R) **1** (nem aktív), és a beíró bemenet (S) **0** (aktív) szintű, vagy vezérlés előtt is Q_n = 1 állapot volt.

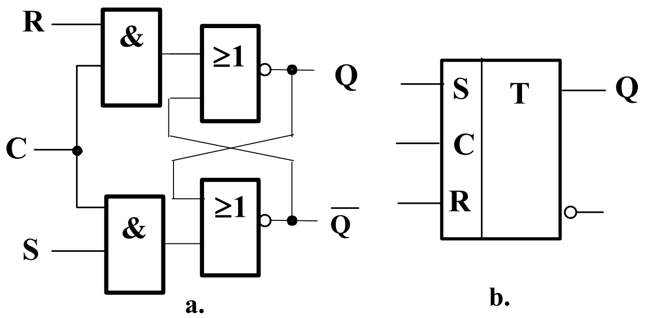
A második összefüggés írja le a tiltást, miszerint a két bemenet legalább egyikén 1 szintnek kell lenni. Áramkörileg NAND kapukkal valósítható meg statikus billentésű - 0 szinttel vezérelhető - RS flip-flop (63. ábra)



90. ábra

Az ismertetett két flip-flop ***közvetlen*** vezérlésű. A tárolandó információt hordozó ***beíró*** vagy **törlő** jel egyúttal a ***billentést*** is vezérli. A beírandó adatot hordozó-, és a billentő jelet kapuzással lehet fizikailag szétválasztani.

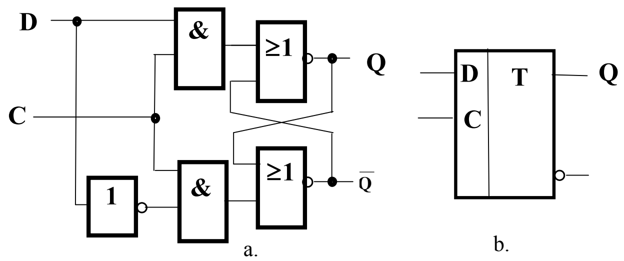
Statikus 1 szinttel vezérelt RS flip-flop vezérlőbemeneteit **C** ***billentő jellel kapuzva*** (66. ábra) kapjuk a kapuzott (szinkronozott) vezérlést. Ennél a flip-flop típusnál az R és S együttes aktív vezérlése csak a billentő (C) jel 1 szintjénél tiltott. Az **S** és **R** információs bemeneteken az ***előkészítés*** és a **C** jel hatására a tényleges beírás vagy törlés, vagyis az ***adat*** (információ) ***bevitel*** következik be.



91. ábra

A statikus billentésű 0 szinttel vezérelt RS flip-flop kapuzása VAGY kapukkal oldható meg, miután az aktív szint 0 mind a billentő, mind pedig az információs bemeneteknél.

A kapuzott RS flip-flop -ból alakítható ki a **D** flip-flop. Az inverter biztosítja a beíró és törlő bemenetek ellentétes szintű vezérlését (67. ábra). A **D = 1** szintnél az **S** előkészítő bemeneten **1**, míg az **R** bemeneten **0** szint lesz. A **C** (szinkronozó) bemenetre érkező 1 szint a flip-flop –ot **1-be** ***billenti***, vagyis ettől kezdődően **1-et** ***fog tárolni***.



92. ábra

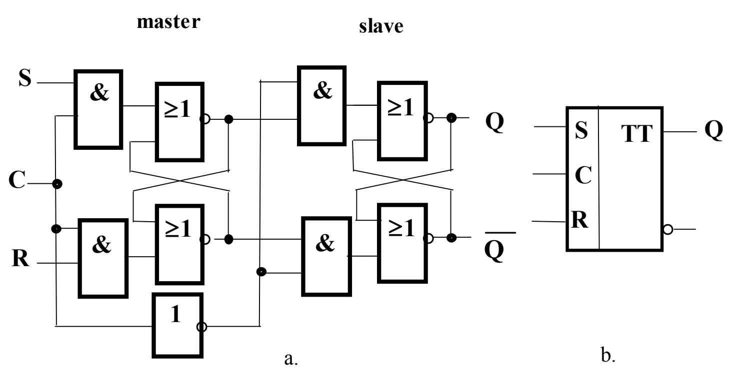
D = 0 esetében a törlés előkészítése, és a C jel hatására a **0** beírása következik. A D flip-flop két szinkronozó jel közötti időtartamra tárolja az információt. A D tároló egyik legfontosabb felhasználási területe az ***információ-bevitel szinkronozása*** a C jel által meghatározott ütemezésben.

Az eddigiekben elemzett két flip-flop típus (RS és D) fő jellemzője, hogy billentő jel hatására az új információ - a billenési idő elteltével - azonnal megjelenik a kimeneten. Az ilyen működésű flip-flop -ot nevezzük egy-ütemű billentésű tárolónak. A ***szimbolikus jelbe*** beírt **egy T** betű jelenti az ***együtemű*** működést.

A digitális módon megvalósított jelfeldolgozásokban jelentős helyet foglalnak el azok a feladatok, melyekben az alkalmazott flip-flop -ok vezérlőbemenetére kimenetük értékét is ***vissza*** kell ***vezetni***. Ilyen esetekben csak olyan flip-flop -ok alkalmazhatók, melyek kimenetén csak akkor jelenik meg az új állapot értéke, amikor a bemeneti vezérlés már hatástalan. Ez az igény ***közbenső-tárolással*** vagy ***élvezérelt*** billentéssel oldható meg.

- ***Közbenső tárolós (ms) flip-flop***

A közbenső tárolós **ms** (master-slave) flip-flop legegyszerűbb elvi változata a 66. ábra szerinti két kapuzott RS flip-flop -ból áll. Amíg a **C** billentő jel szintje **0**, addig a külső (RS) bemenetek szintjétől függetlenül az első flip-flop (master) $\mathbf{R_1}$ és $\mathbf{S_1}$ bemenetein is **0** szint van. A két flip-flop -ot elválasztó kapukra jutó C = 1 szintű jel hatására a master állapota átíródik a második (slave) flip-flop –ba.



93. ábra

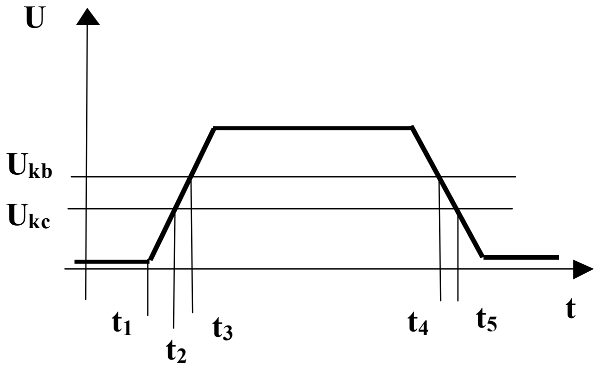
Amikor a **C** jel logikai **1** szintű, akkor a bemeneti vezérlés (S és R értéke) határozzák meg az első flip-flop állapotát, és ***tiltott*** a két tároló közötti csatolás. A ***második flip-flop*** változatlanul tárolja az ***előző információt***, és ezért a kimenet logikai értéke is változatlan. Az ***új információ*** a kimeneten csak **C=0** szintnél jelenik meg, amikor már a bemeneti információközlő vezérlés az első tárolóra hatástalan.

Az ***ms flip-flop szimbólumában*** a kétütemű billentést a **TT** (kettős T) jelöli.

Az elemzett megoldás csak elvileg ad helyes működést. Ha az ellenütemű vezérlést biztosító ***inverter késleltetése nagyobb***, mint a ***flip-flop billenési ideje***, akkor a master még a slave vezérlésének tiltása előtt felveheti az új állapotot, és az át is íródhat a slave -be. Ekkor a billentés közvetlen lesz, vagyis egy-ütemű. Ez hibás működést eredményezhet. A tényleges áramköri megoldásoknál ezért a ***két flip-flop közötti csatolás*** letiltása ***hamarabb*** kell bekövetkezzen, mint a ***bemeneti*** kapuzás ***engedélyezése***. Egyik megoldást a két komparálási szintű kapuzás biztosítja. A megoldás lényege, hogy a billentő (kapuzó) más értékénél – $\mathbf{U_{kb}}$ - nyitnak a bemeneti kapuk, és más értéknél – $\mathbf{U_{kc}}$ - a két flip-flop közötti csatoló kapuk

A kettős komparálás fogalmát a billentő-jel időbeli változása alapján elemezzük. A 69.ábra a Cp bemenetre jutó billentő impulzus időbeli változását mutatja.

A $\mathbf{t_1}$ időpontban kezdődik a billentő-jel felfutó éle, és amikor a $\mathbf{t_2}$ időpontban eléri az $\mathbf{U_{kc}}$ értéket, akkor lezár a master és a slave flip-flop -ok közötti csatolás, de még zárt a bemeneti csatolás is. Egyik tároló tartalma sem változik. A billentő jel további növekedésekor – a $\mathbf{t_3}$ időpontban - eléri a bemeneti kapuk komparálási szintjét – $\mathbf{U_{kb}}$ –t -, és ezután az R és S bemenetekre jutó jel értékétől függő információ íródik a master tárolóba. A slave tároló még tartja az előző értéket, tehát – a bemenetekre jutó információtól függetlenül – a ***„régi"*** érték van a kimeneten is. A billentő jel csökkenésekor a $\mathbf{t_4}$ időpontban lezárnak a bemeneti kapuk, és a $\mathbf{t_5}$ időpontban a flip-flop -ok közötti csatolás nyit ki. Ekkor kerül a kimenetre az ***„új"*** érték. A leírt működés biztosítja azt, hogy a tároló kimeneti értékét vissza lehessen vezetni a bemenetre is.



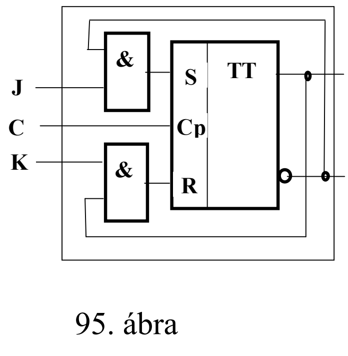
94. ábra

- ***Közbenső* tárolós JK flip-flop**

A **JK** típusú flip-flop - amelynek elvi működési feltételét a korábbiakban már elemeztük - csak ***kétütemű***, vagy ***élvezérelt*** billentéssel alakítható ki. A flip-flop állapot egyenlete az alábbi:

$$\mathbf{Q_{n+1} = J_n\overline{Q}_n + \overline{K}_nQ_n}$$

A JK flip-flop - közbenső tárolós RS flip-flop -ból a 8. ábra szerint épül fel. A ***bemeneti*** vezérlő ***jelek kapuzása*** a ***kimeneti jelekkel*** biztosítja a kívánt működést. Amikor a flip-flop 1-t tárol (Q = 1) akkor csak a K bemenetre jutó vezérlés eredményez állapotváltozást, ill. 0 tárolását követően (Q = 0) a J bemenetre jutó vezérlés a hatásos. Ez a kapuzás (70. ábra) egyúttal engedélyezi a **J és K *bemenetek együttes vezérlését*** is. Ekkor ugyanis a flip-flop előző állapota határozza meg a billentő-jel hatására bekövetkező állapotváltozást.

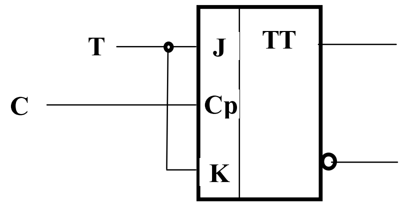


95. ábra

Az előzőekben elemzett JK flip-flop –ból **T *típusú tároló*** olyan módon alakítható ki, hogy a két vezérlő bemenetet (J és K) összekötjük s ez lesz a T vezérlő bemenet. Az állapotegyenlet - 1 szintű aktív vezérlésnél - a következő:

$$\mathbf{Q_{n+1} = T_n\overline{Q}_n + \overline{T}_nQ_n}$$

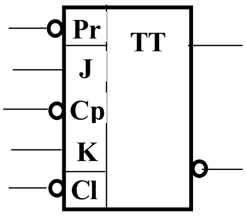
A tárolóba információt csak a T ***vezérlő bemenet 1 szintjénél*** lehet beírni. Az állapot-változást a T = 0 vezérlés letiltja. A T flip-flop -ot elsősorban számláló áramkörök kialakítására használják. Integrált áramköri kialakításban ezt a flip-flop változatot önállóan nem gyártják, miután a JK típusból külső kötéssel kialakítható. A 69. ábra a T flip-flop logikai felépítését és szimbolikus jelét ábrázolja.



96. ábra

- ***Közbenső* tárolós flip-flop -ok aszinkron billentése**

Az integrált áramköri közbenső tárolós - master-slave - flip-flop -oknak ***aszinkron törlő*** és ***beíró*** bemenetei is vannak. Az aszinkron statikus vezérlés együtemű. Ez azt jelenti, hogy a vezérlőjel - a flip-flop mindkét tárolóját - egyidejűleg billenti a kívánt állapotba. A 72. ábra közbenső tárolós **JK** ***preset flip-flop*** szimbolikus jelét mutatja. Az aszinkron vezérlő bemenetek, a **Cl** (Clear) **törlő** és a **Pr** (Preset) ***beíró*** bemenet.



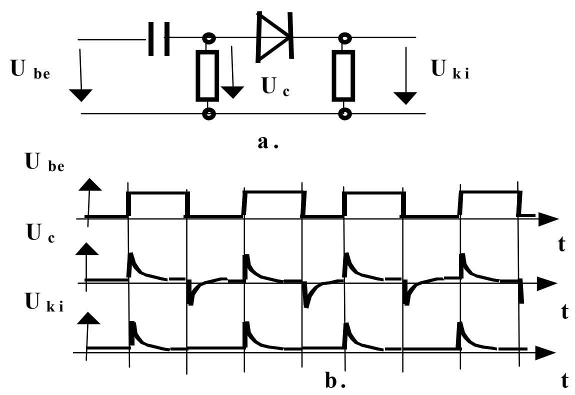
97. ábra

A jelölésben a bemeneti mező középső részéhez csatlakoznak a kétütemű vezérlésű ms flip-flop bemeneti jelei. A **Cp** billentő bemeneten lévő invertáló jel (karika) azt jelzi, hogy az új érték a kimeneteken a ***billentő jel*** **0** szintjénél jelenik meg. A **Pr** aszinkron beíró, illetve Cl törlő bemenetek ***aktív*** szintje **0**. A két utóbbi bemenet szerint a tároló **RS** ***típusú*** együtemű flip-flop.

- **Dinamikus billentésű flip-flop -ok**

Az eddigiekben elemzett flip-flop -ok közös jellemzője a statikus billentés. A tárolók másik nagy csoportját alkotják a ***dinamikus*** billentésű (***élvezérelt***) áramköri megoldások. A továbbiakban külön elemezzük a dinamikus billentés diszkrét ill. integrált áramköri megoldásait.

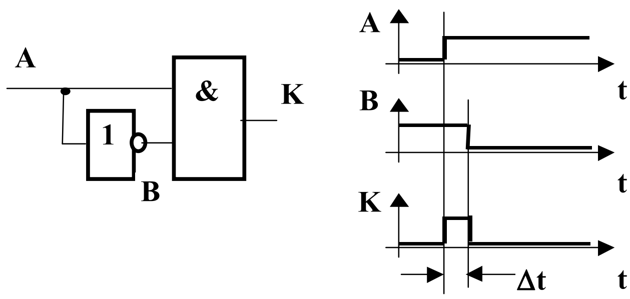
Az ***élvezérlés*** diszkrét elemekkel un ***trigger - áramkörrel*** alakítható ki. A trigger áramkör kimenetén csak akkor jelenik meg jel, ha bemenetén logikai szintváltás van. A trigger áramkör vagy más néven dinamikus csatolókapu egyik legegyszerűbb változata a 73. ábra szerinti.



98. ábra

Az a. ábrán a kapcsolási vázlat, míg a b. ábrán a jellegzetes feszültségalakokat szemlélteti. Ha a kapu bemenetére kapcsolt $\mathbf{U_{be}}$ feszültség négyszöghullám, akkor a belső ponton csak a bemeneti ***szintváltáskor*** mérhető feszültségugrás, mégpedig a szintváltás irányának megfelelő polaritású. A diódán csak a ***pozitív*** feszültség-változás hajt át áramot, ezért a kimeneti ponton a felfutó élekkor lesz jel. A diódát fordítva kötve, a negatív éleknél lesz a kimeneten jelváltás.

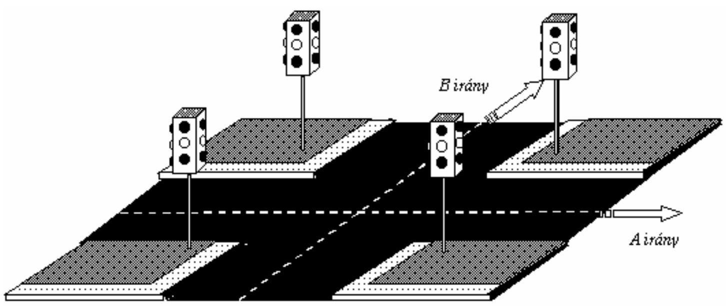
Az élvezérlés egyik megoldásánál a billentő impulzus felfutó élénél mesterségesen létrehozott ***hazárd*** vezérli az állapotváltozást. Ennek elve, hogy ha a logikai ÉS kapu két bemenetén a jelek ellenkező értelemben változóak és az 1 - 0 átmenet késleltetett, akkor a kimeneten a késleltetéssel megegyező idejű – tű-impulzus (hazárd) jön létre. A kapcsolást és az időviszonyokat a 74. ábra szemlélteti. A **Δt** jelű elem késleltetési ideje határozza meg a tű-impulzus szélességét.



99. ábra

12.Példa

Kétirányú útkereszteződés forgalomirányító lámpáinak – 75. ábra -vezérlése



100. ábra

A feladat: olyan vezérlőáramkőr kialakítása, amelynek a bemenetére jutó jel ***váltja*** az ***állapotokat.*** A hat logikai – irányonként 3-3 – ***kimenet vezérli*** a megfelelő ***lámpák*** teljesítményillesztő egységét. A lámpák vzérlésének sorrendje a KRESZ szabályainak feleljen meg.

A tervezés lépései:

- ***Be-, kimenetek deklarálása:***

C	bemenet	a jel ***1 – 0*** átmenete váltja az állapotokat, ezt jelöljük az $\mathbf{X_0}$ kombinációval, míg a nem hatásosat $\mathbf{X_1}$ – el.
P_B	kimenet	***B*** irány ***piros***
S_B	kimenet	***B*** irány ***sárga***
Z_B	kimenet	***B*** irány ***zöld***
P_A	kimenet	***A*** irány ***piros***
S_A	kimenet	***A*** irány ***sárga***
Z_A	kimenet	***A*** irány ***zöld***

- ***A szükséges kimeneti variációk meghatározása***

A ***hat*** kimeneti jelek lehetséges ***64 variációja*** (Zi) lehetséges, viszont a vezérléshez ***csak*** a következő ***négy*** szükséges:

Zi	P_B	S_B	Z_B	P_A	S_A	Z_A	Funkció
$\mathbf{Z_{12}}$	0	0	1	1	0	0	**B** irány ***zöld***, **A** irány ***piros***
$\mathbf{Z_{24}}$	0	1	0	1	1	0	**B** irány ***sárga***, **A** irány ***piros-sárga***
$\mathbf{Z_{33}}$	1	0	0	0	0	1	**B** irány ***piros***, **A** irány ***zöld***
$\mathbf{Z_{50}}$	1	1	0	0	1	0	**B** irány ***piros-sárga***, **A** irány ***sárga***

Megjegyzés: A kimeneti variációk indexét a változók balról –jobbra történő bináris súlyozása alapján számítottuk ki.

- ***A kimeneti variáció szekvenciái:***

Az állapotváltások csak egy kötött sorrendben követhetik egymást, tehát az előállítandó szekvencia:

$$\mathbf{\ldots\ldots\ Z_{12} - Z_{24} - Z_{33} - Z_{50} - Z_{12}\ \ldots..}$$

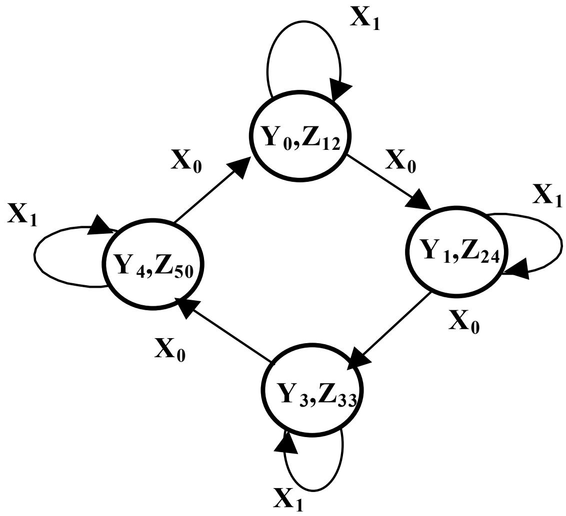
- ***A szükséges állapotok száma***

Az állapotok számát – ***m*** – a szükséges kimeneti kombinációk száma határozza meg, amely a jelen feladatban ***négy***, melyeket jelöljük $\boldsymbol{Y_0}$, $\boldsymbol{Y_1}$, $\boldsymbol{Y_2}$, $\boldsymbol{Y_3}$ –al.

- ***Az alkalmazandó rendszertechnikai felépítés***

A feladatban ***hat*** kimeneten kell jeleket kiadni, viszont a belső állapotok száma csak ***négy***. Az állapotok tárolásához elégséges ***két*** flip-flop, amelyek maximálisan csak négy kimenetet adhatnak. Szükséges tehát ***kimeneti kombinációs*** hálózat alkalmazása. Célszerű a ***Moore-modell*** szerint ***szinkron sorrendi*** hálózattal megvalósítani a feladatot.

- ***Állapotgráf felrajzolása***



- ***Állapottáblázat felrajzolása***

Állapot	Bemeneti komb. X_0	Bemeneti komb. X_1
Y_0^*	Y_1, Z_{12}	Y_0, Z_{12}
Y_1^*	Y_2, Z_{24}	Y_1, Z_{24}
Y_2^*	Y_3, Z_{33}	Y_2, Z_{33}
Y_3^*	Y_0, Z_{50}	Y_3, Z_{50}

Megjegyzés: A * indexet az előző állapot jelzésére használtuk.

- ***A feladat megoldásához alkalmazott flip-flop kiválasztása***

A sorrendi hálózatokban csak elvezérelt, vagy kétütemű vezérlésű tárolók használhatók. Alkalmazzunk 1 szinttel engedélyezhető ***T*** típusú ***master-slave*** (ms) flip-flop -ot.

- ***A kódolt vezérlési táblázat megrajzolása, és a vezérlési függvények meghatározása***

Az állapotok tárolásához szükséges két flip-flop Qi kimenetein megjelenő jelkombinációkhoz rendeljük az állapotokat, és a Ti bemeneteken tiltjuk, vagy engedélyezzük a tároló billenését, amelyet a C jel 0-1 átmenete vezérel.

	Y_i^*		X_0		X_1	
	Q_1	Q_0	T_1	T_0	T_1	T_0
Y_0^*	0	0	0	1	x	x
Y_1^*	0	1	1	1	x	x
Y_2^*	1	0	0	1	x	x
Y_3^*	1	1	1	1	x	x

Karnaugh diagramok

Kp T_1

X \ Q_1Q_0	00	01	11	10
0		1	1	
1	x	x	x	x

$T_1 = Q_0$

Kp T_0

X \ Q_1Q_0	00	01	11	10
0	1	1	1	1
1	x	x	x	x

$T_0 = 1$

- ***A kódolt kimeneti táblázat megrajzolása, és a kimenetek függvényeinek meghatározása***

	Q_1	Q_0	P_B	S_B	Z_B	P_A	S_A	Z_A
Y_0^*	0	0	0	0	1	1	0	0
Y_1^*	0	1	0	1	0	1	1	0
Y_2^*	1	0	1	0	0	0	0	1
Y_3^*	1	1	1	1	0	0	1	0

A kimenetek Kp diagramjai:

P_B

Q_1 \ Q_0	0	1
0		
1	1	1

S_B

Q_1 \ Q_0	0	1
0		1
1		1

Z_B

Q_1 \ Q_0	0	1
0	1	
1		

$P_B = Q_1$ $S_B = Q_0$ $Z_B = \overline{Q_1}\,\overline{Q_0}$

P_A

Q_1 \ Q_0	0	1
0	1	1
1		

S_A

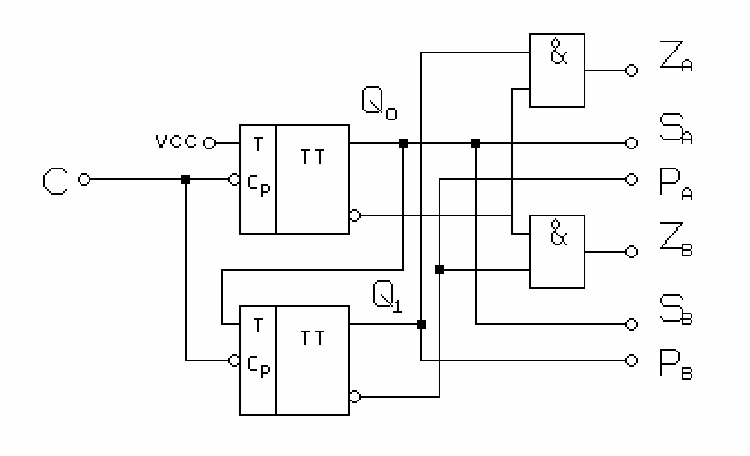
Q_1 \ Q_0	0	1
0		1
1		1

Z_A

Q_1 \ Q_0	0	1
0		
1	1	

$P_A = \overline{Q_1}$ $S_A = Q_0$ $Z_A = Q_1\,\overline{Q_0}$

- *Logikai vázlat*



101. ábra

4.4. Funkcionális sorrendi hálózatok.

A sorrendi logikai feladatok nagytöbbségében szükséges valamilyen változó (esemény) bekövetkeztének ***számosságát*** meghatározni. Ugyancsak gyakori az a feladat, hogy egy ***adatvonalon*** – szabályos időközönként – érkező ***n*** darab logikai érték ***tárolása***.

A vázolt feladatok gyakorisága indokolja, hogy funkcionális egységenként állítsák elő – gyártsák le – ezeket a hálózatokat. Két leggyakoribb hálózat a:

számláló, és a

léptetőregiszter.

- ***Számlálók***

A különböző irányítási, adatfeldolgozási és mérési feladatokban gyakran szereplő részfeladat a legkülönfélébb ***jelek***, tágabb értelemben ***események számlálása***. A funkciót ellátó áramköröket nevezzük ***számlálóknak***.

A számláláskor alapvetően két műveletet kell végezni, úgymint **tárolni** az eddig már bekövetkezett események ***számát***, majd az újabb esemény hatására - a kiválasztott számlálási iránynak megfelelően - az eddigi értéket ***növelni*** (inkrementálás), ill. ***csökkenteni*** 1-gyel (dekrementálás).

A hozzáadás, ill. a levonás - mint ahogy ezt már megismertük - kombinációs logikai feladatként is kezelhető. Az előbbiek alapján tehát a számlálás sorrendi hálózattal megvalósítható.

Az egyirányú számlálók olyan speciális sorrendi hálózatok, amelyeknek állapotai csak egy meghatározott sorrendben követik egymást, s ez az állapotsorozat ciklikusan ismétlődik. A számlálandó jel fogadására a számlálónak egyetlen bemenete van. A kimenetek és a szükséges ***állapottárolók*** számát a ***kapacitás*** szabja meg. A ***kapacitás*** azt a legnagyobb számot jelenti, amellyel egy ciklus befejeződik. Ezt a továbbiakban **k-**val jelöljük. A számsorozat így **0, 1, 2. . . k** értékekből áll. A számlálónak tehát

$$\mathbf{m = k + 1}$$

különböző értéket kell megkülönböztetnie. Az **m** - et nevezzük a számláló ***modulusának***.

- ***A számlálók csoportosítása***

A számlálókat többféle szempont alapján csoportosíthatjuk, úgymint:

működési ***mód***,

számlálási ***irány***,

az információ-tárolás ***kódja***,

áramköri megvalósítás

szerint végezhetjük el a felosztást.

A ***működési mód*** szerint

aszinkron és

szinkron

számlálókat különböztetünk meg. Az ***aszinkron*** működés lényege, hogy a számlálandó jel csak elindítja a szükséges állapotváltozási sorozatot. A továbbiakban az egyes ***flip - flop*** - ok ***billentik*** egymást.

A ***szinkron*** működés alapja, hogy két számlálandó jel között történik a következő állapotba billentés előkészítése, s a flip-flop - okat **a *számlálandó jel billenti***.

A számlálás **iránya** szerint

előre (UP - fel),

hátra (DOWN - le),

előre-hátra (reverzibilis - UP/DOWN)

működések lehetnek. A ***reverzibilis*** számlálóknál a számlálási irányt külső vezérlőjel változtatja meg.

A számtartalmat (információt) ***tárolhat****juk*

- ***bináris***
- ***BCD és***
- ***egyéb***

kódokban. Ezt a számláló megnevezésében jelöljük, pl. szinkron bináris előre számláló.

Az áramköri megvalósításnál

diszkrét elemes és az

integrált áramköri

számláló megkülönböztetés elsődlegesen formai és nem a működés lényegére utal.

A számlálókat - elsődlegesen az integrált áramköri kivitelben - ki szokták még egészíteni járulékos funkciókkal. Ilyen kiegészítés, hogy az egyes flip-flop - ok - a számlálási funkciótól függetlenül is - külső jellel beállíthatók 0 vagy 1 állapotba. Ezek az elő-beírású vagy PRESET számlálók. A másik gyakori megoldás, hogy a számlálás végszámát jelző áramkör is a számláló tartozéka (egyazon tokban van).

- ***Bináris számlálók***

  Leggyakrabban használjuk a 2-es számrendszerben számláló bináris számlálók különböző változatait. Ebben a pontban részletesen foglalkozunk a számlálók e csoportjának működésével, és logikai tervezésével.

  - **Bináris** ***számlálók logikai tervezése***

    Egy **k** kapacitású ***bináris számláló*** logikai tervezésének lépései:

a tárolók ***számának*** meghatározása,

az ***állapottáblázat*** felvétele,

az alkalmazott ***flip-flop típus*** kiválasztása,

a kódolt állapottáblázat felírása,

a ***vezérlő függvények*** meghatározása,

a ***logikai vázlat*** megrajzolása.

A logikai tervezést egy **k = 7** kapacitású ***szinkron*** **bináris** számláló példáján ismertetjük.

A számlálónak **m = k + 1 = 8** állapotot kell megkülönböztetnie. A szükséges állapottárolók száma tehát ***három*** (2^3=8). A számláló kimenetein - az egyes ütemekben - a 0 - 7 értékek bináris kódját kell kapjuk. Ezek az értékek három bináris helyértékkel kifejezhetők, tehát az állapottárolók és a hálózat kimenetei ugyanazok is lehetnek.

A számláló állapottáblázata (77. ábra) három oszlopot, és nyolc sort tartalmaz. Miután egyetlen bemeneti jel van (C), két bemeneti kombináció lehetséges**,** $\mathbf{X_0}$**,** $\mathbf{X_1}$. Válasszuk $\mathbf{X_0}$ -hoz, amikor nincs számlálandó jel (***C = 0***), az $\mathbf{X_1}$ –hez, pedig, amikor van (***C = 0***). A nyolc kimeneti kombináció ($\mathbf{Z_0}$ – $\mathbf{Z_7}$) pedig a bináris számértékeknek megfelelő állapotok. A megállapodás szerint * felső index a ***jelenlegi***, míg anélkül a ***következő*** állapotot jelenti.

Ki-mene-tek	***Bemenetek***	
Z^*	X_0	X_1
Z_0^*	Z_0	Z_1
Z_1^*	Z_1	Z_2
Z_2^*	Z_2	Z_3
Z_3^*	Z_3	Z_4
Z_4^*	Z_4	Z_5
Z_5^*	Z_5	Z_6
Z_6^*	Z_6	Z_7
Z_7^*	Z_7	Z_0

102. ábra

Következő lépésként a kódolt állapottáblázatot kell felírni (78.ábra).

A ***bemeneti*** kódolást már meghatároztuk, amely szerint

X_0-nál **C = 0,** **X_1**-nél **C= 1.**

A ***kimeneti*** kombináció-sorozat, pedig a $\mathbf{z_0 = 2^0}$, $\mathbf{z_1 = 2^1}$, $\mathbf{z_2 = 2^2}$ kimeneteken megjelenő ***bináris számsor*** (a helyértékek a jelölés szerintiek).

Az ***állapotvezérlő*** jelek kombinációi (**V_0 – V_7**) billentik a flip-flop -okat a soron következő állapotkombinációba. Változás csak a C=1 értéknél van, és ekkor értékük egyértelműen meghatározza a következő állapotot. Az egyes vezérlőjelek (**v_i**) flip-flop – okat billentik. A táblázatban a következő jelölést használtuk: jel = 1, ha kell változtatni a tároló értékét, és 0 ha nem.

Megjegyzés: a különböző tárolóknál más, és más lehet a vezérlőjel értéke.

Kimeneti kombinációk			***Bemeneti-*(C)*, és vezérlőjel* (V_i) *kombinációk***					
			0			***1***		
z_2	z_1	z_0	v_2	v_1	v_0	v_2	v_1	v_0
0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	0	0	0	0	0	0	1
0	1	1	0	0	0	1	1	1
1	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	1	1
1	1	0	0	0	0	0	0	1
1	1	1	0	0	0	1	1	1

103. ábra

A táblázatból megállapítható, hogy a számlálók vezérlőtáblázatából elhagyhatók a C=0 – állapotváltozás nincs – oszlopok, és ezzel egyszerűbb táblázatot kapunk. A következő példáknál ezt követjük.

- ***Szinkron bináris számlálók***

Számlálók ***él-vezérelt*** vagy ***közbenső tárolós*** (master-slave) flip-flop –ból építenek, mivel ezeknél lehet a billentés feltételébe a kimenetek jeleit visszacsatolni. Ugyanakkor a számlálandó jel mindegyik tároló billentő bemenetére vezethető, vagyis kettéválasztottuk az ***előkészítő***, és a ***billentő*** jeleket. Ez a számlálás ***szinkron*** üzemű megoldása.

A számlálót alakítsuk ki **T-**típusú **ms** flip-flop –al. Ekkor az egyes tárolók T bemeneteire kell csatlakoztatni az állapotvezérlő jeleket. Ekkor az állapotváltozók kódolását abból a feltételből írjuk fel, hogy **1** szint ***engedélyezi*** a flip-flop billentését, **0** szint, pedig ***nem***. Az előbbi megállapodások szerinti kódolt állapottáblázat látható az

78.ábrán. (A táblázatban az állapotváltozókat a ***T0, T1, T2*** –al, a kimeneteket, pedig ***Q0, Q1, Q2*** –al - a szakirodalomban legtöbbször használt betűkkel - jelöltük.)

Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	0	0	1
1	1	1	1	1	1

104. ábra

105. ábra

A ***kódolt állapottáblázatból*** az egyes engedélyező bemenetekre külön-külön felírhatunk egy-egy ***Karnaugh - diagramot***. Ezek segítségével aztán a legegyszerűbb logikai függvények meghatározhatók. Ezzel tulajdonképpen a bemeneti kombinációs hálózat logikai felépítését is meghatározzuk.

T_2

Q_2 \ Q_1Q_0	00	01	11	10
0			1	
1			1	

T_1

Q_2 \ Q_1Q_0	00	01	11	10
0		1	1	
1		1	1	

T_0

Q_2 \ Q_1Q_0	00	01	11	10
0	1	1	1	1
1	1	1	1	1

106. ábra

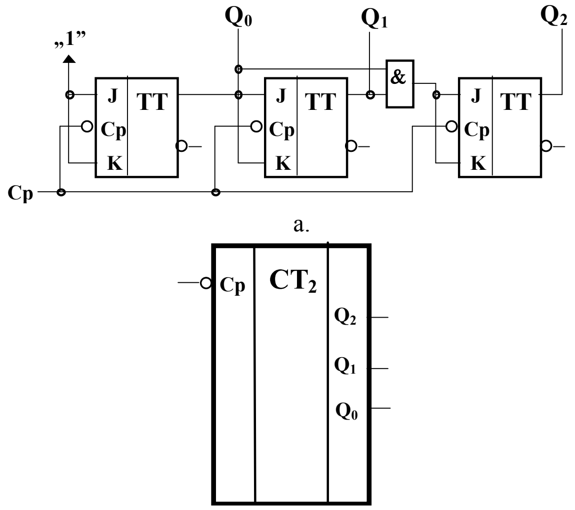
A 80.ábrán láthatók a T_0, a T_1 és a T_2 változókra felirt K diagramok, amelyek alapján az egyes vezérlőfüggvények:

$$\mathbf{T_0 = 1,}$$

$$\mathbf{T_1 = Q_0,}$$

$$\mathbf{T_2 = Q_0Q_1}$$

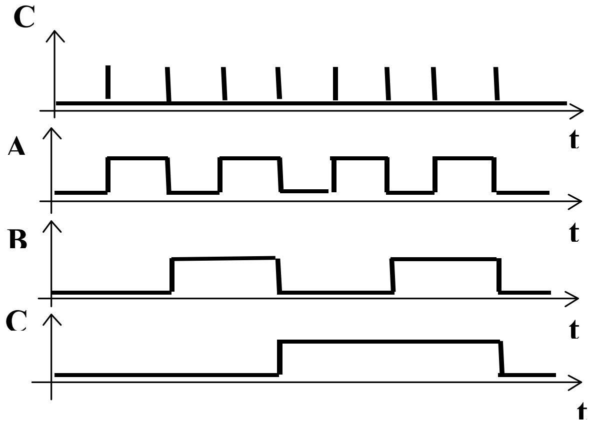
A számláló logikai vázlata az 81.a.ábrán, a számláló szimbolikus jele, pedig a b. ábrán láthatók. A CT (COUNTER) jelölés melletti index a számrendszerre utal, pl.CT_2 bináris számláló.



b.

107. ábra

A kimenetek, és a számlálandó jel időfüggvényeit mutatja a 82. ábra. Az időfüggvények felett feltüntettük az egyes ütemek kimeneti állapot - kombinációit. Ezek sorozata - a kitűzött célnak megfelelően - a növekvő bináris számsort adják.



108. ábra

A számláló kapacitását további flip-flop -okkal növelni lehet. Ezek vezérlőfüggvényeit az előzőekhez hasonlóan határozhatjuk meg. Ezt most mellőzve, az időfüggvényekből is következtethetünk a törvényszerűségre. A soron következő flip-flop mindig olyankor vált állapotot, amikor minden előző flip-flop nál 1 - 0 állapotátmenet van. Ennek alapján a az **i.** flip-flop vezérlőfüggvényének általános alakja:

$$T_i = Q_0 Q_1\, Q_2\, .\,.\,.\;\; Q_{i-1}$$

A függvény alapján megállapíthatjuk, hogy a kapacitásbővítéshez - az újabb flip-flop mellett - mindig 1-gyel több bemenetű ÉS kapu kell. Ezt a megoldást nevezzük **párhuzamos átvitelűnek**. A megnevezés arra utal, hogy minden egyes előkészítő bemenetre egyidejűleg (párhuzamos csatornákon) jutnak a megfelelő kimenetek értékei. Ezáltal egy kapunyi jelkésleltetés múlva az újabb billentés előkészítése befejeződik.

Az összefüggések átalakíthatók a következők szerint:

$$T_0 = 1,$$

$T_1 = Q_0 = \mathbf{T_0\ Q_0}$

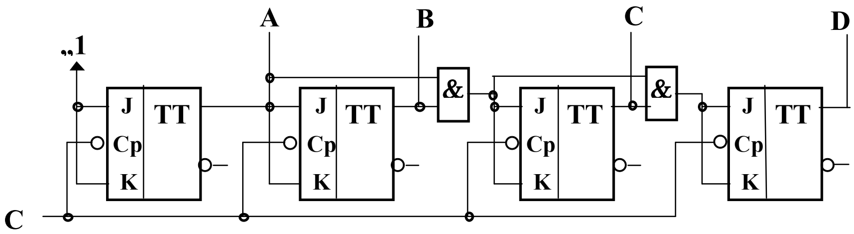
$T_2 = Q_0Q_1 = \mathbf{T_1\ Q_1}$

.

.

$T_i = Q_0Q_1\ Q_2\ .\ .\ .\ \ Q_{i-1} = \mathbf{T_{i-1}\ Q_{i-1}}$

Az átalakított vezérlőfüggvények szerint kialakított m = 16 modulusú bináris számláló logikai vázlatát mutatja a 83. ábra. A kimenetek elnevezésénél – a számlálóknál használt – **A,B,C,D** jelölést rajzoltuk.



109. ábra

Ebben a megoldásban egységesen két bemenetű ÉS kapuk állítják elő a vezérlőjeleket. Az áramköri egyszerűsítés ára, hogy a számláló határfrekvenciája csökken, mert a legutolsó flip-flop előkészítő bemenetére a jel két sorba kötött kapun keresztül jut. Ezt az áramköri megoldást nevezzük ***soros átvitelűnek***. További kapacitásbővítés újabb kapukat, s így késleltetéseket iktat be.

Hárombites bináris ***hátraszámláló*** kódolt állapottáblázatát láthatjuk az 84.ábrán. Ezt is T típusú flip-flop -okból alakítjuk ki.

Q_2	Q_1	Q_0	T_2	T_1	T_0
1	1	1	0	0	1
1	1	0	0	1	1
1	0	1	0	0	1
1	0	0	1	1	1
0	1	1	0	0	1
0	1	0	0	1	1
0	0	1	0	0	1
0	0	0	1	1	1

110. ábra

T_2

Q_2 \ Q_1Q_0	11	10	00	01
1			1	

T_1

Q_2 \ Q_1Q_0	11	10	00	01
1		1	1	

T_0

Q_2 \ Q_1Q_0	11	10	00	01
1	1	1	1	1

0			1		0		1	1		0	1	1	1	1

111. ábra

Az előkészítő bemenetek vezérlőfüggvényei a Kp diagramok alapján (85.ábra) a következők:

$$\mathbf{T_0 = 1}$$

$$\mathbf{T_1 = \overline{Q}_0}$$

$$\mathbf{T_2 = \overline{Q}_0 \overline{Q}_1}$$

A vezérlőfüggvény általános alakja:

$$\mathbf{T_i = \overline{Q}_0 \overline{Q}_1 \cdots \overline{Q}_{i-1}}$$

lesz. E függvények közvetlen megvalósításával párhuzamos átvitelű ***bináris hátraszámlálót*** kapunk. Logikai átalakítások után – az előreszámlálóhoz hasonlóan - a soros átvitelű bináris hátraszámláló is kialakítható.

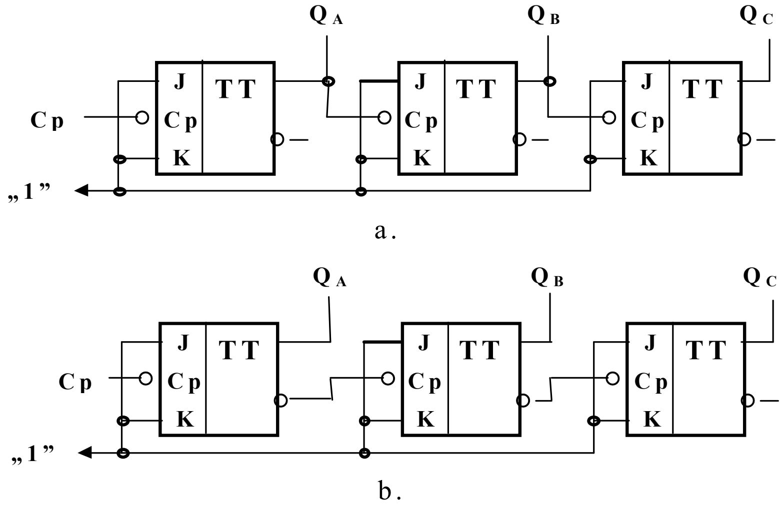
Az előre-, ill. hátraszámláló vezérlőfüggvényeinek ismeretében felírhatjuk a ***reverzibilis*** bináris számláló függvényeit is. A számlálási irányt a **P** külső parancs vezérli (P = 0 előreszámlálás, P = 1 hátraszámlálás

- **Aszinkron bináris számlálók**

Az aszinkron működésű számlálóknál a számlálandó jel csak ***elindítja*** a soron következő ***állapotváltozást***, de az egyes flip-flop -ok ***egymást*** billentik. A bináris előreszámláló kimeneteinek időfüggvényénél (82. ábra) láttuk, hogy mindegyik flip-flop a megelőző tároló 1 - 0 átmeneténél kell, billenjen. Így **1 - 0** átmenetre billenő **T** flip-flop -ok 85.a.ábra szerinti kapcsolásával bináris előreszámlálót kapunk.

A bináris hátraszámlálónál az előző flip-flop -ok a megelőző tároló 0 - 1 állapotváltozásánál kell billenjenek. Ugyancsak 1 - 0 átmenetre billenő T flip-flop -okból kialakított hátraszámláló logikai vázlata a 86 b. ábrán látható.

Az aszinkron számlálók nagyon egyszerű felépítése mellett hátránya a ***kisebb határfrekvencia***. Ez abból adódik, hogy az új stabil állapot csak az egymást követő billenések, befejezése után áll be. Ugyancsak hátrány, hogy az átmeneti időszakban nem ***kívánt kombinációk*** is előfordulnak a kimeneteken. Ez a csatlakozó hálózatnál zavart okozhat.



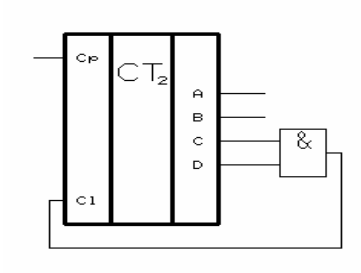
112. ábra

➤ *BCD kódolású számlálók*

A különböző digitális mérő- és egyéb adatfel-dolgozó berendezésekben az adatok kijelzése, ill. bevitele rendszerint 10-es számrendszerben történik. Ezért a belső adat-forgalomnál, így a számlálásnál is esetenként célszerű a használata. Ezt teszik lehetővé a különböző ***BCD*** kódolású számlálók.

A tananyagban csupán a ***BCD 8421*** súlyozású számlálókkal foglalkozunk. A megismert tervezési módszer alapján azonban a további BCD kódolású számlálók is megtervezhetők.

A ***8 4 2 1*** súlyozású BCD számláló működése a 4 bites bináris számlálótól abban tér el, hogy a tizedik impulzus hatására a kezdő ***0 0 0 0*** állapotba tér vissza a számláló.



113. ábra

Ennek a ***modulus csökkentésnek*** egy lehetséges megoldása, hogy egy 4 bites bináris számlálót - amelynek aszinkron törlő bemenete is van - olyan logikai hálózattal egészítünk ki, ami az **1 0 1 0** (decimális 10) állapot megjelenésekor minden flip-flop -ot töröl és ezzel **0 0 0 0** állapot áll be. Ezt a megoldást szemlélteti a 87. ábra.

E megoldás hátránya, hogy a 11. állapot egy rövid ideig - a kapu késleltetés és a billenési idő összegéig - bekövetkezik. Ez járulékos hibát okoz, különösen frekvenciaosztóként való alkalmazáskor.

A logikai tervezés során, a bináris számlálónál megismert induló fázisokat - az általános állapot táblázat és vezérlőfüggvényeinek felírását - elhagyjuk. A tervezést a kiválasztott flip-flop típusra érvényes kódolt állapottáblázat felírásával kezdjük.

- **Szinkron BCD számlálók**

A szinkron BCD számlálók felépítését és működését JK típusú ms flip-flop -ok alkalmazásával ismertetjük.

Először röviden összefoglaljuk a JK flip-flop vezérlésének feltételeit. Ezt mutatja a 88. ábrán levő táblázat. A $\mathbf{Q_i}$ a billentés előtti, $\mathbf{Q_{i+1}}$ pedig a billentő impulzus hatására bekövetkező új állapotot jelzi. Az **x** közömbös értéket jelent, vagyis 0 vagy 1 is lehet.

$\mathbf{Q_i}$	$\mathbf{Q_{i+1}}$	**J**	**K**
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

114. ábra

A BCD 8 4 2 1 súlyozású előreszámláló kódolt állapottáblázata látható a 89. ábrán. A kimeneteket - a nemzetközileg egységesen használt - A, B, C, D betűkkel jelöltük, ahol az $\mathbf{A=2^0}$ helyérték.

D	C	B	A	J_D	K_D	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	x	0	x	0	x	1	x
0	0	0	1	0	x	0	x	1	x	x	1
0	0	1	0	0	x	0	x	x	0	1	x
0	0	1	1	0	x	1	x	x	1	x	1
0	1	0	0	0	x	x	0	0	x	1	x
0	1	0	1	0	x	x	0	1	x	x	1
0	1	1	0	0	x	x	0	x	0	1	x
0	1	1	1	1	x	X	1	x	1	x	1
1	0	0	0	x	0	0	x	0	x	1	x
1	0	0	1	x	1	0	x	0	x	x	1

115. ábra

Az egyes flip-flop -ok J és K bemeneteinek vezérlési feltételeit 90. ábrán levő Kp diagramok segítségével határozzuk meg.

J_A

BA \ DC	00	01	11	10
00	1	1	x	1
01	x	x	x	x

K_A

BA \ DC	00	01	11	10
00	x	x	x	x
01	1	1	x	1

J_B

BA \ DC	00	01	11	10
00	0	0	x	0
01	1	1	x	0

K_B

BA \ DC	00	01	11	10
00	x	x	x	x
01	x	x	x	x

11	x	x	x	x
10	1	1	x	x

11	1	1	x	x
10	x	x	x	x

11	x	x	x	x
10	x	x	x	x

11	1	1	x	x
10	0	0	x	x

J_C

BA \ DC	00	01	11	10
00	0	x	x	0
01	0	x	x	0
11	1	x	x	x
10	0	x	x	x

K_C

BA \ DC	00	01	11	10
00	x	0	x	x
01	x	0	x	x
11	x	1	x	x
10	x	0	x	x

J_D

BA \ DC	00	01	11	10
00	0	0	x	x
01	0	0	x	x
11	0	1	x	x
10	0	0	x	x

K_D

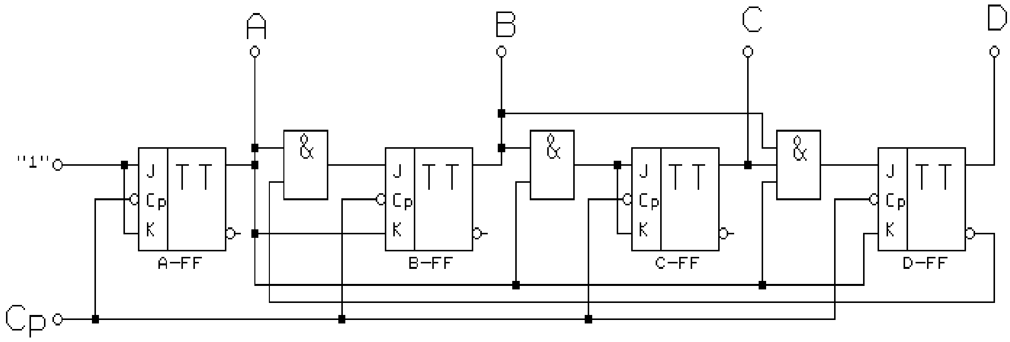
BA \ DC	00	01	11	10
00	x	x	x	0
01	x	x	x	1
11	x	x	x	x
10	x	x	x	x

116. ábra

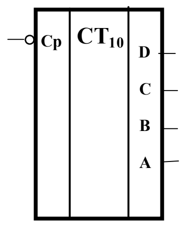
A BCD kódban nem szereplő kombinációkat is **x** -el jelölhetjük, s így a függvény egyszerűsítéseknél felhasználhatjuk. A logikai függvények:

$$\mathbf{J_A = 1} \qquad \mathbf{K_A = 1}$$
$$\mathbf{J_B = A\overline{D}} \qquad \mathbf{K_B = A}$$
$$\mathbf{J_C = AB} \qquad \mathbf{K_C = AB}$$
$$\mathbf{J_D = ABC} \qquad \mathbf{K_D = A}$$

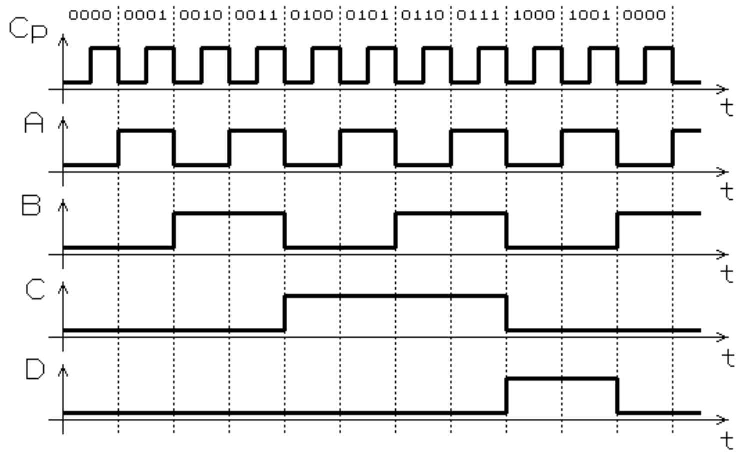
A szinkron BCD előreszámláló logikai vázlatát, szimbolikus jelölését, és a kimenetek időbeli változását a 91. a, b, c. ábrák mutatják.



a.



b.



c.

117. ábra

- **Aszinkron BCD számlálók**

A legegyszerűbb **aszinkron** üzemű **BCD** előreszámlálót egyetlen billentő bemenettel rendelkező flip-flop -okból építhetünk. Az egyes tárolók billentési feltételeit a 90. c. ábra jelalakjai alapján is felírhatjuk. 1 - 0 átmenetre billenő flip-flop -nál az egyes billentési feltételek:

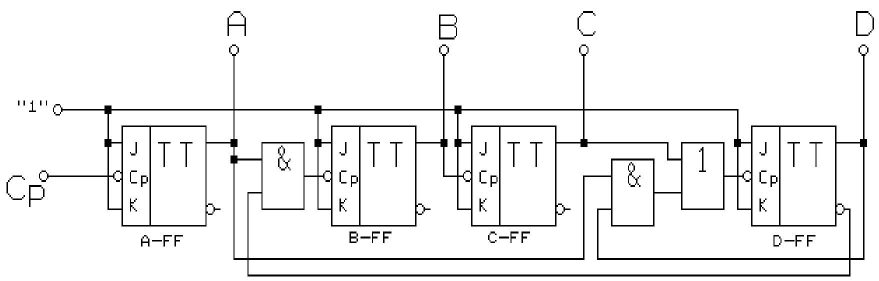
$$\mathbf{C_{pA} = C_s}$$

$$\mathbf{C_{pB} = A\overline{D}}$$

$$\mathbf{C_{pC} = B}$$

$$\mathbf{C_{pD} = C + AD}$$

Az élvezérelt flip-flop -okból kialakított aszinkron BCD előreszámláló logikai vázlata a 92. ábrán látható.



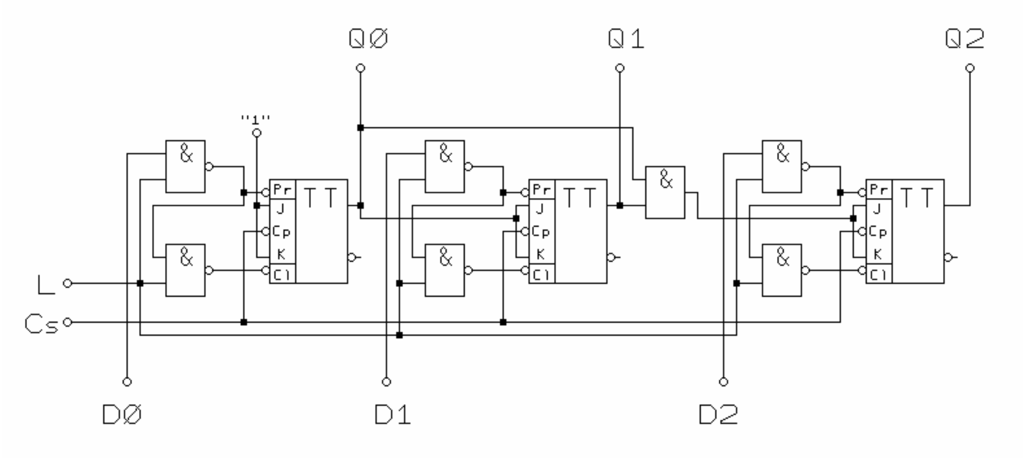
118. ábra

Az aszinkron hátra- és reverzibilis BCD számlálók kialakítása a megismert logikai tervezési eljárás segítségével lehetséges.

- **Preset számlálók**

Számlálók alkalmazásakor szükség lehet arra, hogy a számlálást esetenként ne a 0-tól, vagy a kapacitás végértékétől kezdjük, hanem egy közbenső számtól. Ehhez szükséges, hogy külön külső parancs hatására, a számláló flip-flop -jait tetszőleges állapot-kombinációba lehessen billenteni. Ezeket nevezzük preset (elő beírású) számlálóknak.

A számláló tartalmának - egyidejű párhuzamos - változtatása, programozása is történhet aszinkron, ill. szinkron módon.

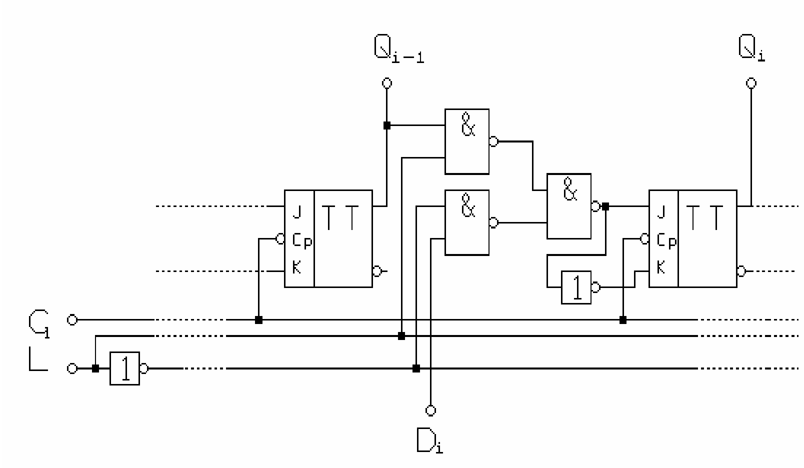


119. ábra

Az ***aszinkron*** programozás - az adatbeírás - a számlálandó jeltől ***függetlenül*** történik. A 93. ábrán látható 3 bites szinkron bináris előre számláló flip-flop -jai aszinkron üzemű beíró (**Pr**) és törlő (**Cl**) bemeneteit használjuk fel a párhuzamos adatbeírásra. Az adatbemenetek $\mathbf{D_0}$, $\mathbf{D_1}$, $\mathbf{D_2}$ és a beírást vezérlő jel az **L** (Load). Ha az L-re 0 szintet adunk, akkor a flip-flop -okba az adatbemeneteken érvényes információ íródik.

Amíg az L-en aktív jel van, addig az áramkör számlálóként nem működik. Az aszinkron bemenetek (Pr, Cl) hatása erősebb a Cp billentő jelnél.

A szinkronprogramozású megoldásnál a beírandó adatot mindig a soron következő Cp jel írja be a flip-flop -okba. Ennek egy áramköri megvalósítására mutat példát a 94. ábra.



120. ábra

Mindegyik flip-flop előtt azonos felépítésű kiválasztó áramkör van. Az L beíró jel 1 szintjénél a $\mathbf{D_i}$ adatút tiltott és a számlálási feltételek kerülnek a flip-flop -ok előkészítő bemeneteire.

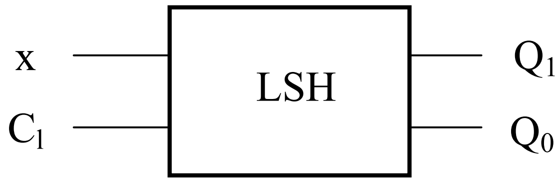
L = 0 vezérlésnél az adatok értéke jut a JK be-menetekre. A tényleges beírás ekkor is a Cp jel 1 - 0 átmenetekor következik be. Ameddig az L aktív, addig a párhuzamos beírás érvényesül.

- ***Léptetőregiszterek***

A léptető regiszterek (shift - regiszter) kettős feladatot ellátó ***funkcionális*** áramkörök. Egyrészt egy n bites **digitális szó tárolására**, másrészt egy-egy léptető jel hatására, a bemenetre érkező jel, valamint a már tárolt információ **léptetésére** használhatók. A hálózat annyi tárolóból (flip-flop -ból) áll, ahány bites információt kell tárolni, illetve léptetni.

A következőben határozzuk meg a hálózat felépítését. Az általánosságon nem esik csorba, ha két bitre végezzük el a feladatot.

A megvalósítandó hálózatnak két kimenete ($\mathbf{Q_0, Q_1}$), egy információs (**x**), és egy léptető ($\mathbf{C_l}$) bemenete kell legyen. A blokkvázlat látható a 95. ábrán.



121. ábra

Írjuk fel a feladat állapottáblázatát (96. ábra). Négy lehetséges bemeneti (Xi), kimeneti kombináció (Z_i) lehet. A lehetséges állapotok száma is négy.

A bemeneti kombinációk az x, és a C_l értékvariációiból adódnak.

	x	C_l
X_0	0	0
X_1	0	1
X_2	1	0
X_3	1	1

A kimeneti kombinációkat a Q_0, és a Q_1 variációi adják,

	Q_1	Q_0
Z_0	0	0
Z_1	0	1
Z_2	1	0
Z_3	1	1

Az állapotok megegyeznek a kimeneti kombinációkkal.

Ki-mene-tek	***Bemenetek***			
Z^*	**X_0**	**X_1**	**X_2**	**X_3**
Z_0^*	Z_0	Z_0	Z_0	Z_1
Z_1^*	Z_1	Z_2	Z_1	Z_3
Z_2^*	Z_2	Z_0	Z_2	Z_1
Z_3^*	Z_3	Z_2	Z_3	Z_3

122. ábra

A 123. ábrán látható kódolt vezérlési táblázatot – a számlálóéhoz hasonlóan (lásd. 78. ábra) – írtuk fel.

Kimeneti kombiná-ciók		***Bemeneti-*** (X_i)***, és vezérlőjel*** (V_i) ***kombinációk***							
		X_0		***X_1***		***X_2***		***X_3***	
		x C_l		***x C_l***		***x C_l***		***x C_l***	
		0 0		***0 1***		***1 0***		***1 1***	
Q_1	**Q_0**	**v_1**	**v_0**	**v_1**	**v_0**	**v_1**	**v_0**	**v_1**	**v_0**
0	0	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1	0
1	0	0	0	1	0	0	0	1	1
1	1	0	0	0	1	0	0	0	0

123. ábra

A táblázatból elhagyhatók azok a bemeneti kombinációk oszlopai, amelyeknél (X_0, X_2) léptetőjel nincs (C_l=0). A táblázat tovább egyszerűsíthető azáltal, hogy a léptetőjel minden flip-flop billentő bemenetére kapcsolódik. Az így kapott táblázatban (98.ábra) már csak az előkészítő bemenetek (v_i) maradnak.

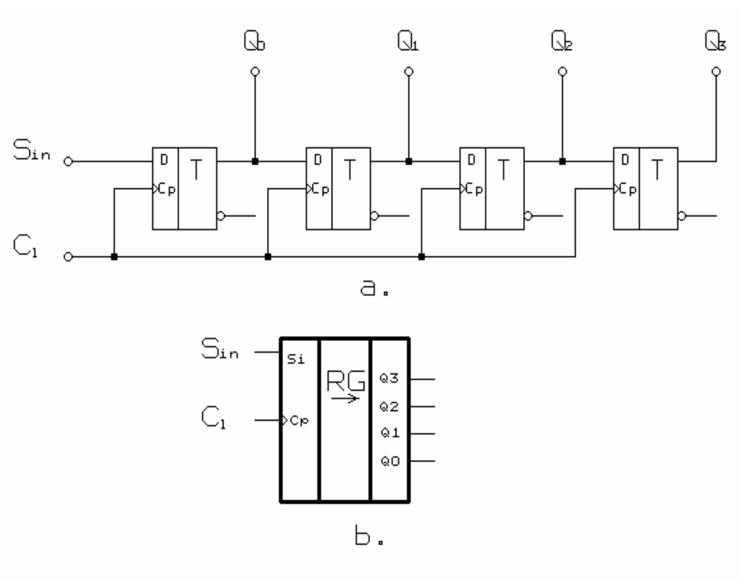
Kimeneti kombiná-ciók		***Bemenet*** (x)***, és vezérlőjelek*** (v_i)			
		0		***1***	
Q_1	**Q_0**	**v_1**	**v_0**	**v_1**	**v_0**
0	0	0	0	0	1

0	1	1	1	1	0
1	0	1	0	1	1
1	1	0	1	0	0

124. ábra

A megfelelő flip-flop kiválasztása után írható fel a tényleges vezérlési táblázat

A 95. ábrán látható egy 4 bites - élvezérelt D típusú flip-flop -okból kialakított – léptető regiszter logikai vázlata. Léptető regiszternél - az információ átmeneti tárolása mellett - a szomszédos flip-flop -ok között olyan csatolást kell megvalósítani, amely biztosítja, hogy közöttük egy külső léptető jel hatására információ átadás történjen.



t	Q_0	Q_1	Q_2	Q_3
t_1	1	x	x	x
t_2	0	1	x	x
t_3	1	0	1	x
t_4	0	1	0	1

c.

125. ábra

A $\mathbf{C_l}$ léptető jel 0 - 1 átmeneténél mindegyik tároló elembe a D bemenetén érvényes logikai érték íródik. Azáltal, hogy az egyes $\mathbf{D_i}$ bemeneteket az előző flip-flop $\mathbf{Q_{i-1}}$ kimenetével kötöttük össze, a tárolt digitális szó léptetése történik. A legelső flip-flop -ba pedig az $\mathbf{S_i}$ jelű bemenet aktuális értéke íródik. A 37.b. ábrán a léptető regiszter **szimbolikus jele** látható. A 94.c. ábrán táblázatban szemléltetjük a működési ütemeket, ha a soros bemenetre ($\mathbf{S_i}$) **1 0 1 0** jelsorozat érkezik a léptető-jel ütemezésében (az x a léptetés előtti ismeretlen tartalmat jelzi). Az egyes sorok az egymás után érkező léptető impulzusok hatására bekövetkező állapotokat tartalmazzák.

➢ ***A léptető regiszterek fajtái***

A léptető regisztereket csoportosíthatjuk

az információ beírása és

a kiolvasás

módja szerint, valamint a ***léptetés*** iránya alapján.

A ***beírás*** és a ***kiolvasás*** szerint megkülönböztetünk

párhuzamos és

soros

beírású, ill. kiolvasású léptető regisztereket.

A léptetés ***iránya*** szerint

jobbra (az alacsonyabb nagyságrend irányába),

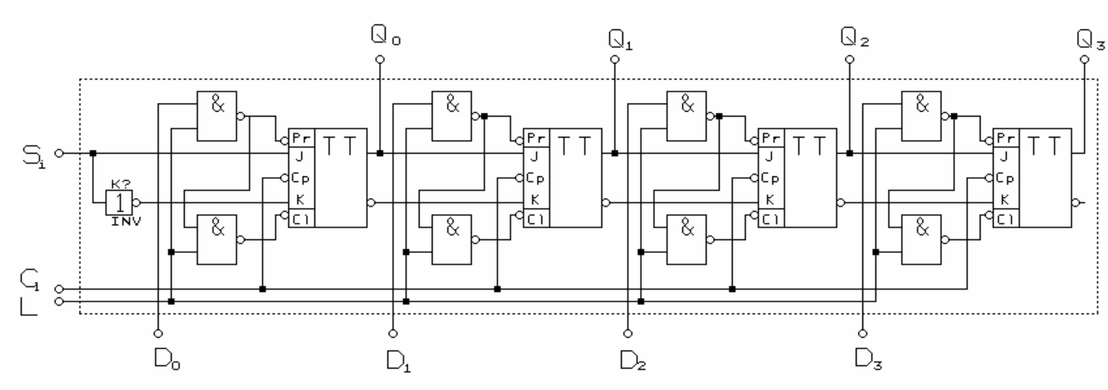
balra (a magasabb nagyságrend irányába), és

kétirányú léptetésű

regiszterek vannak.

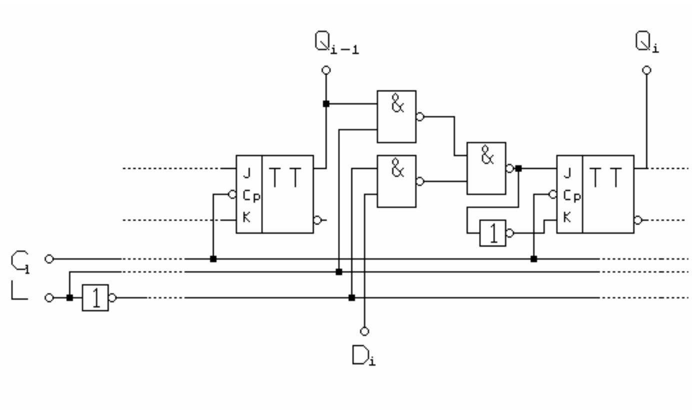
A 99. ábra szerinti léptető regiszter **soros beírású** jobbra léptető regiszter. Amennyiben a kimenetek (**Q_0, Q_1, Q_2, Q_3**) mindegyike kivezetett, akkor a kiolvasás **párhuzamos**. Amennyiben csak a **Q_3** kimenethez csatlakozhatunk, akkor a **kiolvasás** módja **soros**. Ez az utóbbi megoldás elsődlegesen az integrált áramköri kialakításoknál használt a szükséges lábszám csökkentéséhez.

A párhuzamos információ-beírás - a számlálóknál már megismertekhez hasonlóan - **aszinkron** és **szinkron** módon történhet. Az 100. ábrán egy aszinkron párhuzamos beírású, jobbra léptető regiszter logikai vázlata látható. A párhuzamos szó beírása az **L**=1 értéknél történik az egyes flip-flop -ok **Pr** és **Cl** bemenetein keresztül, tehát aszinkron módon. Ekkor a **C_l** léptető jel hatása nem érvényesül. Az **L**=0 értéknél soros beírású (**S_i** = serial input), jobbra léptető regiszterként működtethető az áramkör.



126. ábra

A **szinkron** üzemű **párhuzamos beírás** egy áramköri megoldását mutatja a 101. ábra, amely egy léptető regiszter egy részletét mutatja. Az **L** beíró jel **1** értéke a léptetési üzemmódot választja. Ekkor az **i** - ik flip-flop -ba - a **C_l** 1 - 0 jelváltásakor - az **i-1** - ik flip-flop értéke íródik. ábra



127. ábra

Az **L=0** vezérlésnél a **Cl** jel a **Di** információ érvényes értékét írja az **i** - ik flip-flop -ba, vagyis párhuzamos beírás történik.

➢ ***A léptetőregiszterek alkalmazása***

A léptetőregisztereket, mint átmeneti tárolókat (tartó áramkörök) szinte minden digitális berendezésben alkalmazzák. Ezzel itt részletesebben nem foglalkozunk. Viszont tárgyaljuk a léptetőregiszterek

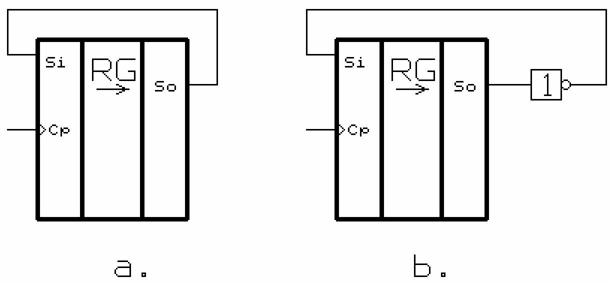
- **gyűrűs** számlálóként, ill.
- **soros**-párhuzamos és
- **párhuzamos**-soros

kódátalakítókban való felhasználását.

- **Gyűrűs számlálók**

Amennyiben egy léptető regiszter soros kimenetét (**So**) a soros bemenettel (**Si**) összekötjük, akkor olyan áramkört kapunk, amelyben az információ kering (a kilépő bit beíródik az első tárolóba). Ezt a megoldást nevezzük ***gyűrűs számlálónak***.

Az adat visszavezetése történhet egyenes és tagadott alakban is (102. ábra). Az a. ábra szerinti visszavezetési megoldással ***n - modulusú***, míg a b. ábra szerint ***2n - modulusú*** gyűrűs számlálót kapunk, ahol **n** a regiszter tárolóinak száma.



128. ábra

Az ***n - modulusú*** gyűrűs számlálónál az eredeti információ az n. lépés után kerül vissza a regiszter megfelelő helyértékeire. Erre mutat példát a 103.ábra szerinti működési táblázat, amelyen egy 4 bites n modulusú gyűrűs számláló egyes ütemeinek állapota látható az **1 0 0 0** kezdő feltételből indulva.

t	Q_0	Q_1	Q_2	Q_3
t_1	1	0	0	0
t_2	0	1	0	0
t_3	0	0	1	0
t_4	0	0	0	1
t_5	1	0	0	0

129. ábra

Az áramkört felhasználhatjuk, pl. soros működésű aritmetikai egység átmeneti tárolójaként, ha az egyik tényezőt - műveletvégzés után - változatlanul kívánjuk megtartani.

Számlálóként is használhatjuk a gyűrűs számlálót. Ha a regiszterben egy darab 1-et léptetünk, akkor minden állapotban egyetlen kimenet értéke lehet 1 szintű. Ha az n kimenet mindegyikéhez egy N alapszámú számrendszer egy számjegyét rendeljük, akkor ***1 az N-ből*** kódolású számlálót kapunk.

A ***2n modulusú*** gyűrűs számlálóban **2n** számú léptetés után kapjuk vissza az eredeti állapotot. A 104.a. ábrán levő táblázat mutatja egy 4 bites 2n modulusú gyűrűs számláló állapotsorozatát, ha a **0000** állapotból indulunk ki. Ugyanezen gyűrűs számlálóban a b. ábra táblázata szerinti állapotsorozat is kialakulhat. Mindkét sorozat 8-8 állapotból (2n) - két teljes ciklusból - áll. A kettő együtt tartalmazza a lehetséges 16 kombinációt.

Ütem	Q_3	Q_2	Q_1	Q_0
1	0	0	0	0
2	0	0	0	1
3	0	0	1	1
4	0	1	1	1
5	1	1	1	1
6	1	1	1	0
7	1	1	0	0
8	1	0	0	0
9	0	0	0	0

a.

Ütem	Q_3	Q_2	Q_1	Q_0
1	1	0	0	1
2	0	0	1	0
3	0	1	0	1
4	1	0	1	1
5	0	1	1	0
6	1	1	0	1
7	1	0	1	0
8	0	1	0	0
9	1	0	0	1

b.

130. ábra

Általánosan a következő törvényszerűség fogalmazható meg: egy **n** bites léptető regiszterből kialakított ***2n*** modulusú gyűrűs számláló **k** féle ***teljes*** ciklusban működtethető, ahol

$$\mathbf{k = \frac{2^n}{2n}}$$

hányados egész része. Amennyiben az osztás eredménye nem egész szám, akkor csonka ciklus is van. ***Csonka ciklusnak*** nevezzük az olyan sorozatot, amely 2n lépésnél hamarabb veszi fel a kezdő kombinációt. A csonka ciklus állapotainak száma az osztásnál kapott maradékkal egyezik meg.

Példa:

n=3 esetén **egy** 6 állapotú (2n = 6) ***teljes*** ciklus és **egy** kétállapotú ***csonka*** ciklus lehetséges. **n=5** bites gyűrűs számlálónál **három** 10 állapotú ***teljes*** ciklus és **egy** kétállapotú ***csonka*** ciklus létezik. A kezdőszám fogja meghatározni, hogy melyik ciklusban üzemel a számláló. Amennyiben több teljes ciklus is lehetséges, ezek közül azt tekintjük ***alap-ciklusnak,*** amely tartalmazza az összes bit 0 kombinációt.

Az **öt**bites **2n** modulusú gyűrűs számlálót decimális számlálóként is használjuk. A lehetséges három teljes ciklusból a 00000 állapotot is tartalmazó sorozatot (alap-ciklus) nevezzük ***Johnson - kódnak***. Ahhoz, hogy a gyűrűs számláló mindig az alap- ciklusban üzemeljen, biztosítani kell, hogy az esetleges ciklustévesztés után (pl. külső zavar) automatikusan kerüljön vissza az alap-ciklusba. Egyik megoldás lehet, ha egy élvezérelt D flip-flop a soros kimenet 1 - 0 átmenetekor bebillen és törli a számláló flip-flop -jait. Ez a törlés a helyes működést nem zavarja, mivel az alap-ciklusban egyébként is ez az állapot kell következzen. A következő órajel 1 szintje aszinkron módon törli a D flip-flop –ot. Ha valamilyen okból hibás állapot áll be, ezt - néhány ütem után - automatikusan törölni fogja a D tároló.

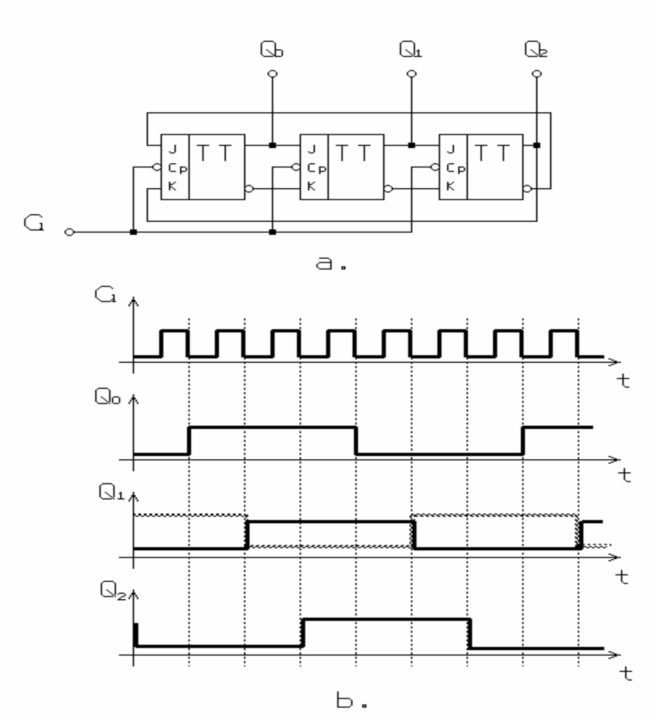
Példa: Vegyük azt, hogy valamilyen zavar eredményeként az **10010** hibás állapotot lép fel. A következő ütem az **11001**, majd **<u>01100</u>** lenne, de az utóbbi beálltakor a D flip-flop is bebillen s ez a számláló **00000** állapotát állítja be. Ennek eredményeként csak egyetlen hibás ciklus lesz.
Röviden említést teszünk a **2n** modulusú gyűrűs számlálók egy speciális vezérlés-technikai felhasználásáról. Amennyiben **n=k*3**, vagyis a három egész számú több-szöröse, akkor a számláló kimenő jeleiből mindig előállítható ***3 fázisú szimmetrikus jelrendszer***.

A 136. a. ábrán **3** bites 2n modulusú gyűrűs számláló logikai vázlata látható. A b. ábra szemlélteti az órajel és a kimeneti jelek idő-függvényeit.

Mindhárom kimenet jele szimmetrikus négyszögjel, és frekvenciája

$$\mathbf{f_{ki} = \frac{f_q}{2n}}$$

ahol $\mathbf{f_q}$ a léptető jel frekvenciája, és n a regiszter bitjeinek száma. A b. ábrán látható, hogy az egyes kimenetek jelei egy léptető-jel periódus idejével késnek egymáshoz képest.



131. ábra

Mindegyik jel periódus-ideje 6 ütem, amit tekinthetek 360 villamos foknak. Ebből következik, hogy az egyes jelek közötti fázistolás:

$$\mathbf{\Phi = \frac{2\Pi}{n} = \frac{360^\circ}{6} = 60^\circ}$$

Amennyiben a $\mathbf{Q_0, \overline{Q}_1, Q_2}$ jelsorozatot tekintjük, ezek - bármilyen órajel frekvenciánál - pozitív sorrendű szimmetrikus háromfázisú rendszert alkotnak. Ezért háromfázisú rendszerek - pl. aszinkron motorok fordulatszám változtatásánál stb.- vezérlő jeleként felhasználhatók.

- **Párhuzamos-soros kódátalakítás**

A fejezetben röviden ismertetjük a léptetőregiszterek alkalmazásával megvalósítható **párhuzamos-soros** kódátalakítást.

Az átalakítás elve, hogy az átalakítandó, párhuzamos kódolású információt a léptetőregiszterbe - a **párhuzamo**s adatbemeneteken keresztül - **írjuk be**. Ezt követően - az órajel ütemében - léptetve a regiszter tartalmát, annak **soros kimenetén** (So) időben egymás után - egyetlen csatornán - kapjuk az információ egyes bitjeit. Ezzel soros kódolásban áll rendelkezésünkre az eredeti információ. A kódátalakító áramkörnek biztosítania kell, hogy minden párhuzamos beírást - a szóhossznak megfelelő - **n** számú léptetés kövessen. Ezután ismét a párhuzamos beírás, vagyis az új információ fogadása következik.

A párhuzamos-soros kódátalakítókat leggyakrabban a nagyobb távolságú adatátviteli rendszereknél használják. Soros kódban való információátvitelhez egyetlen adatcsatorna szükséges.

- **Soros-párhuzamos kódátalakítás**

A soros-párhuzamos kódátalakítás elve, hogy az átalakítandó **n** bites ***információt*** CL órajel lépteti be a ***regiszterbe***. Majd az **n+1**-edik ütemben (”szó szünet”) kerül a párhuzamosan kódolt információ a kimeneti csatornákra.

4.5. Példák

4.6. Ellenörző kérdések