KOVÁCS CSONGOR

DIGITÁLIS ELEKTRONIKA

Kovács Csongor DIGITÁLIS ELEKTRONIKA

Berúlé
Gábor
11/c

Kovács Csongor

Digitális elektronika

Kovács Csongor

Digitális elektronika

General Press

Kovács Csongor: Digitális elektronika

Lektorálta: Mészáros Miklós

Borítóterv: Drobek Ödön

Felelős szerkesztő: Füleki Beáta

ISBN 963 9076 34 1

Kiadja a General Press Kiadó
Felelős kiadó: Lantos Kálmánné ügyvezető
Irodalmi és művészeti vezető: Lantos Kálmán

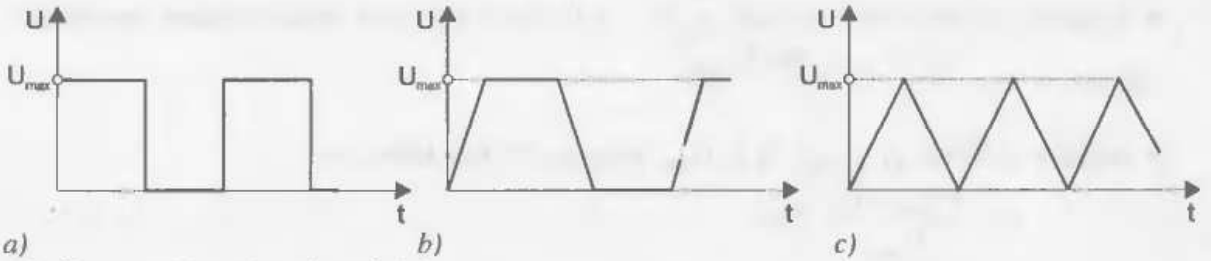
Tartalomjegyzék

1. Impulzustechnikai áramkörök

A szinuszos jelek áramköri technikája mellett nagy jelentőségük van a nem szinuszos jelalakokat előállító és feldolgozó ún. impulzustechnikai áramköröknek. Az ***impulzustechnika*** az elektronika egy olyan részterülete amely két nyugalmi állapot között ugrásszerűen változó mennyiségeket előállító, átalakító, valamint ezen mennyiségek mérésére alkalmas áramkörökkel és berendezésekkel foglalkozik.

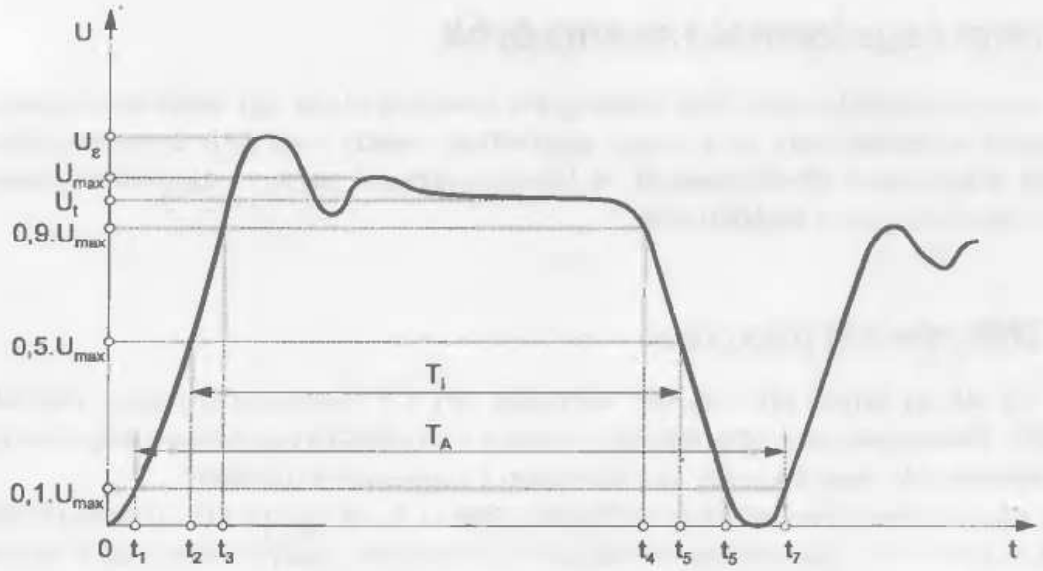
1.1. Az impulzusok jellemzői

Az impulzus olyan áram, vagy feszültség melynek értéke két nyugalmi állapot között ugrásszerűen változik. Az *1.1. ábra* különböző típusú, szabályos impulzusalakokat szemléltet.



1.1. ábra. Szabályos impulzusalakok
a) négyszögimpulzus; b) trapéz alaku impulzus; c) háromszögimpulzus

Az elektronikus áramkörök állapotának bármilyen megváltozása berezgési és lecsengési folyamatokkal jár együtt, amelyek hatása a jel időtartamához képest vagy elhanyagolható vagy nem hanyagolható el. Ennek következtében a valóságban csak bizonyos pontossági határok között léteznek szabályos impulzusalakok. Egy valóságos impulzusalakot a *1.2. ábra* szemléltet.



1.2. ábra: Általános impulzusalak

Egy általános impulzusalak jellemzésére a következő paramétereket használják:

- ***impulzus amplitúdó*** (jelölése U_{max});
- ***impulzus periódusidő*** (jelölése T_A): – a $0{,}1 \cdot U_{max}$ amplitúdóértékhez tartozó időtartam ($T_A = t_7 - t_1$);
- ***impulzus idő*** (jelölése T_i): – a $0{,}5 \cdot U_{max}$ amplitúdóértékhez tartozó idő ($T_i = t_5 - t_2$);
- ***felfutási idő*** (jelölése T_f): – azon időtartam amíg az impulzus amplitúdója $0{,}1 \cdot U_{max}$ értékről $0{,}9 \cdot U_{max}$ értékre változik ($T_f = t_3 - t_1$);
- ***lefutási idő*** (jelölése T_l): – azon időtartam amíg az impulzus amplitúdója $0{,}9 \cdot U_{max}$ értékről $0{,}1 \cdot U_{max}$ értékre csökken ($T_l = t_6 - t_4$);
- ***felfutási meredekség*** (jelölése v_f): – a felfutási idő alatt bekövetkezett amplitúdóváltozás: $v_f = \dfrac{0{,}9 \cdot U_{max} - 0{,}1 \cdot U_{max}}{T_f}$;
- ***lefutási meredekség*** (jelölése v_l): – a felfutási idő alatt bekövetkezett amplitúdóváltozás: $v_l = \dfrac{0{,}1 \cdot U_{max} - 0{,}9 \cdot U_{max}}{T_l}$;
- ***tetőesés*** (jelölése ε_2): – az U_t és U_{max} viszonya %-ban kifejezve:

$$\varepsilon_2 = \frac{U_{max} - U_t}{U_{max}} \cdot 100 ;$$

- ***túllövés*** (jelölése ε_1): – az U_ε és U_{max} viszonya %-ban kifejezve:

$$\varepsilon_1 = \frac{U_\varepsilon - U_{max}}{U_{max}} \cdot 100 ;$$

- ***kitöltési tényező*** (jelölése k): – az impulzusidő és a periódusidő viszonya:

$$k = \frac{T_i}{T_A} .$$

1.2. Impulzusformáló áramkörök

Az impulzusformáló áramkörök segítségével lehetőség nyílik egy adott impulzus-sorozat jelalakjának az átalakítására. Ez a feladat megoldható passzív vagy aktív áramköri elemekből felépített négypólusok alkalmazásával. A következőkben a passzív négypólusok impulzusformáló tulajdonságaival foglalkozunk.

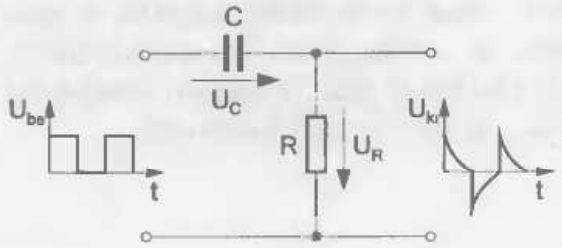
1.2.1. Differenciáló négypólus

Az *1.3. ábrán* látható differenciáló négypólus egy *CR* feszültségosztóként viselkedik. A négypólus viselkedését négyszögimpulzus hatására a következőképpen tanulmányozhatjuk.

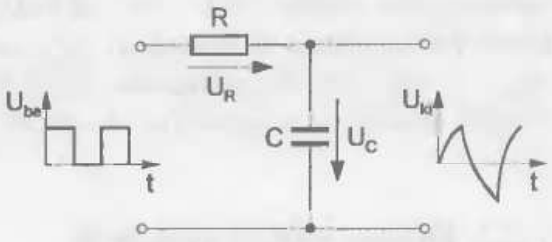
- Feltételezzük, hogy a kezdeti időpillanatban a kondenzátor töltetlen.
- A négyszögimpulzus értéke t=0 időpillanatban 0 V-ról egy pozitív U értékre változik (*1.4.a ábra*). A kondenzátor rövidzárként viselkedik (mivel nem tudja követni a feszültség gyors változását), így a teljes U feszültség megjelenik az ellenálláson.

A kondenzátor a töltőáram hatására elkezd töltődni az ellenálláson pedig csökken a feszültség. A kondenzátor töltési folyamata exponenciális görbével írható le (*1.4.b ábra*), aminek következtében az ellenálláson folyó áram exponenciális változású feszültséget hoz létre (*1.4.c ábra*). A kondenzátor töltődésének, valamint az ellenálláson eső feszültség csökkenésének sebességét az RC szorzat értéke határozza meg.
Ezt a szorzatot *időállandónak* nevezzük: $\tau = R \cdot C$.

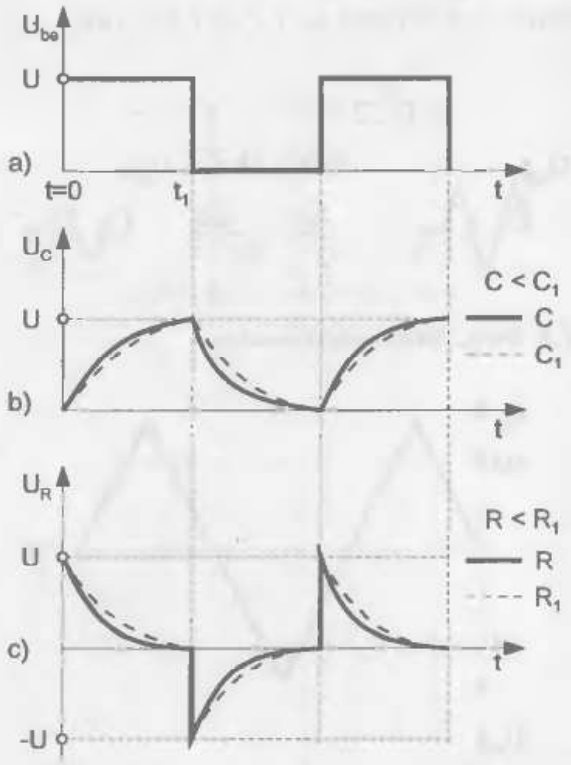
• A négyszögimpulzus értéke t_1=0 időpillanatban az U értékről 0 V-ra változik. A kondenzátor kisül, de mivel ez a folyamat nem olyan gyors mint az impulzus változási sebessége, az impulzus lefutó éle megjelenik az ellenálláson. A kondenzátor exponenciális kisülési folyamatának megfelelően az ellenálláson is változik a feszültség.



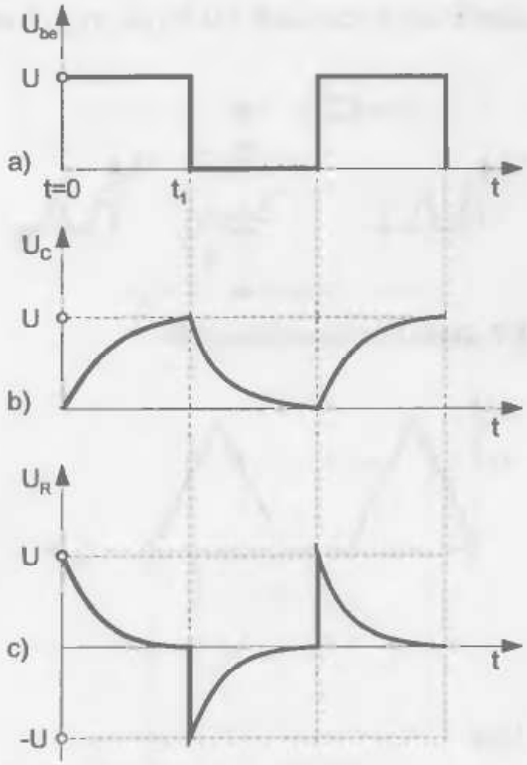
1.3. ábra. Differenciáló négypólus



1.5 ábra. Integráló négypólus



1.4. ábra. Differenciáló négypólus ki- és bemeneti jellemzöinek változása



1.6. ábra. Integráló négypólus ki- és bemeneti jellemzőinek változása

Az *1.4. ábra* szemlélteti, hogy a négypólus elemeinek a változása miként befolyásolja a feszültségváltozásokat. Megállapítható, hogy ha az impulzust jelentékenyebb torzulás nélkül kívánjuk egy soros kondenzátoron átvinni, akkor $R \cdot C >> \tau$-t kell választani. Ha pedig $R \cdot C > \tau$, a jelalak gyors komponensei jutnak csak át az áramkörön.

1.2.2. Integráló négypólus

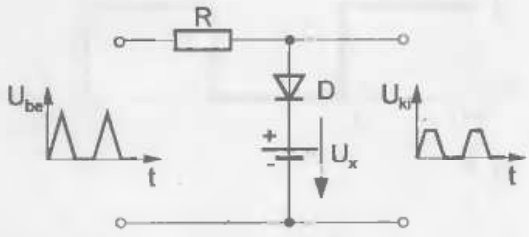
Az integráló négypólus kapcsolása az *1.5. ábrán* látható. A négypólus viselkedését négyszögimpulzus hatására a differenciáló négypólushoz hasonlóan tanulmányozhatjuk (*1.6. ábra*).

- Feltételezzük, hogy a kezdeti időpillanatban a kondenzátor töltetlen. A négyszögimpulzus felfutó élének megjelenésekor a kondenzátor rövidzárként viselkedik és kivezetésein 0 V feszültség alakul ki (az ellenálláson a feszültség maximális értéket vesz fel). A kondenzátor fokozatosan feltöltődik és rajta közel U feszültség lesz mérhető. A négyszögimpulzus lefutó élének megjelenésekor az előző folyamat fordítva játszódik le.

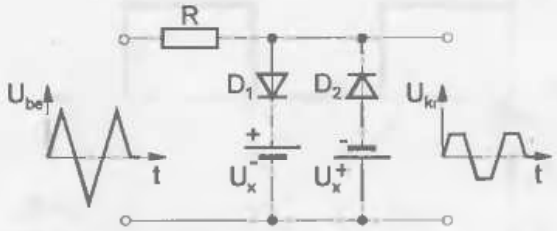
A kondenzátor töltődésének illetve az ellenálláson eső feszültség csökkenésének a sebességét itt is az RC szorzat (*az időállandó*) értéke határozza meg. A torzítatlan impulzusátvitel feltétele jelen esetben $R \cdot C << \tau$, ha pedig $R \cdot C > \tau$, akkor az előzőkhöz hasonló, de most kvázi-differenciálásnak nevezhető folyamat játszódik le. Az integrálás is csak hozzávetőleges: az impulzus megszűnte után a kondenzátor az időállandó által megszabott sebességgel elveszti töltését, nem marad azon az értéken, ahová az „integrálás" alatt feltöltődött.

1.2.3. Diódás vágóáramkörök

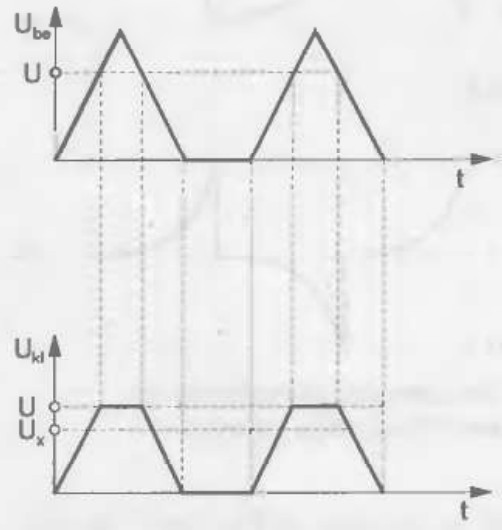
A diódás vágóáramkörök olyan impulzusformáló négypólusok, amelyek az impulzusok amplitúdó-határolását valósítják meg. Ilyen vágóáramkörök láthatók az *1.7* és *1.8 ábrán.*



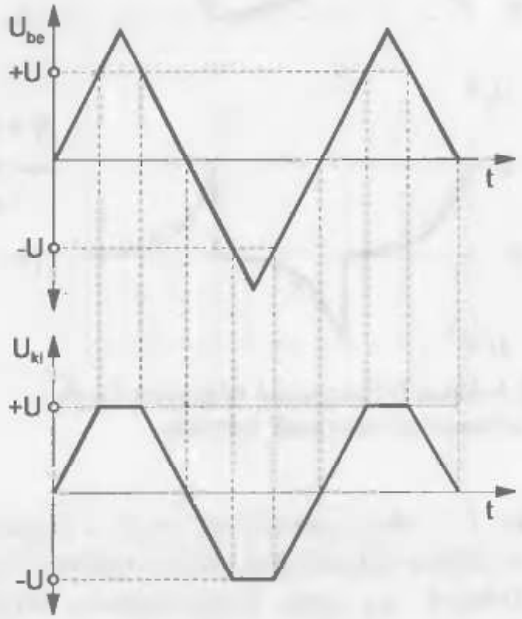
1.7. ábra. Diódás vágóáramkör



1.8. ábra. Diódás vágóáramkör



1.9. ábra. Kimeneti hullámformák



1.10. ábra. Kimeneti hullámformák

Ezekben az áramkörökben a dióda kapcsolóelemként működik. A nyitóirányban előfeszített félvezető dióda úgy viselkedik, mint egy kis értékű ellenállás (*rövidzárral helyettesíthető*), a záróirányban előfeszített dióda pedig úgy viselkedik mint egy nagy értékű ellenállás (*szakadással helyettesíthető*).
Az *1.7. ábrán* látható kapcsolás esetén az U_x feszültség a diódát záró irányban feszíti elő.

- Ha a bemeneti feszültség eléri az $U > U_x + U_D$ (Si-dióda esetén $U_D = 0{,}6$ V) értéket a dióda elkezd vezetni és a bemeneti jel többi részét levágja.
- Ha a bemeneti feszültség az $U < U_x + U_D$ érték alá csökken, a dióda zárt állapotba kerül.

A vágóáramkör be- és kimeneti jel alakját az *1.9. ábra* szemlélteti.

Az *1.8. ábrán* látható kapcsolás működése az előbbi vágóáramkörhöz hasonló, azzal a különbséggel, hogy képes vágni a bemeneti impulzus pozitív és negatív tartományában is.

- A D_1 jelzésű dióda az impulzus pozitív csúcsát, a D_2 jelzésű dióda az impulzus negatív csúcsát vágja le.

A vágóáramkör be- és kimeneti hullámformáit az *1.10. ábra* szemlélteti.

A diódás vágóáramkörök felhasználásával – lineáris szakaszokból összetéve – szinte tetszőleges kimeneti-bemeneti karakterisztikájú nemlineáris átvivő rendszereket konstruálhatunk.

A vágókapcsolások nagyon sokszor okoznak meglepetést a tervezőknek. Ennek oka az, hogy az áramkörökhöz csatlakozó szórt kapacitások jócskán eltorzíthatják a jelalakokat. Sok esetben jelent gondot a diódák véges kapcsolási ideje is.

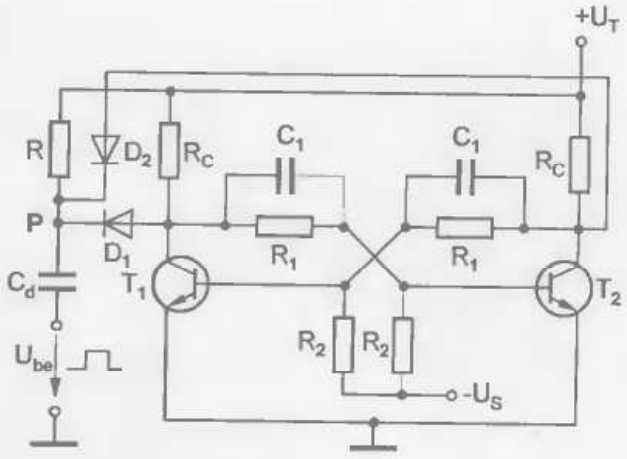
1.3. Impulzuselőállító áramkörök (billenőkapcsolások)

Az impulzuselőállító áramkörök, másnéven *billenőkapcsolások* pozitívan visszacsatolt digitális áramkörök. Abban különböznek a pozitívan visszacsatolt lineáris áramköröktől (*oszcillátoroktól*), hogy kimeneti feszültségük nem folyamatosan változik, hanem két meghatározott érték (egy magas – **H** – és egy alacsony – **L** – feszültségszint) között ugrál. A két állapot közötti átmenet nagyon gyorsan valósul meg.

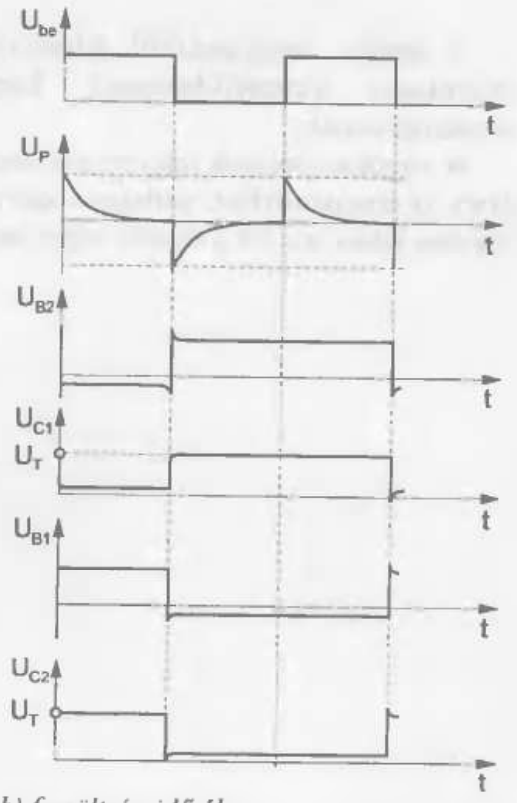
1.3.1. Bistabil billenőkapcsolás

A bistabil billenőkapcsolásnak (amit *bistabil multivibrátornak*, vagy *flipflopnak* is neveznek) ***két stabil állapota van***. A bistabil billenőköröknél a kimenet állapota csak akkor változik, ha az átbillenési folyamatot egy bemeneti jel kiváltja.

Az *1.11.a ábrán* egy bistabil billenőkapcsolás áramköri rajza, az *1.11.b ábrán* a kialakuló jelalakok láthatók. A bemeneti négyszögimpulzust a C_d jelű kondenzátor differenciálja, míg a D_1 és D_2 diódák biztosítják, hogy a lezáró impulzus mindig a nyitott tranzisztor bázisára kerüljön.



a) kapcsolási rajz



b) feszültség-idő ábra

1.11. ábra. Bistabil billenőfokozat

Az áramkör működése a fenti ábrák alapján tárgyalható.

- Az áramkört tápfeszültségre kapcsolva valamelyik tranzisztor vezetésbe kerül (tételezzük fel, hogy ez a T_1 tranzisztor), amely a pozitív visszacsatolás révén lezárja a másik tranzisztort (T_2-őt).

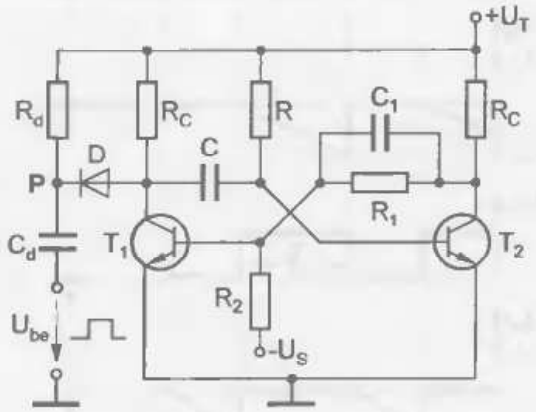
Ebben az esetben a T_1 kollektorán közel 0 V feszültség, bázisán az R_1 és R_2 ellenállások által leosztott pozitív feszültség mérhető. A D_1 dióda ilyenkor záróirányban van előfeszítve. A lezárt T_2 tranzisztor kollektorán közel tápfeszültség, bázisán a $-U_S$ segédfeszültség jön létre. A negatív segédfeszültség a tranzisztorok stabil lezárását biztosítja. A D_2 dióda anódja és katódja közelítően azonos potenciálon van.

• A bemeneti négyszögjel differenciált lefutó éle a T_1 bázisára jut, amelynek hatására a T_1 tranzisztor lezár és kollektorpotenciálja közel tápfeszültségre kerül. Ez az R_1 és R_2 feszültségosztón a T_2 bázisára jut és nyitja azt. Megtörténik az átbillenés és ez az állapot mindaddig megmarad, amíg a következő negatív impulzus a T_1 tranzisztort ki nem nyitja. E tulajdonsága miatt a bistabil billenőkapcsolást bináris információtárolóként is használják.

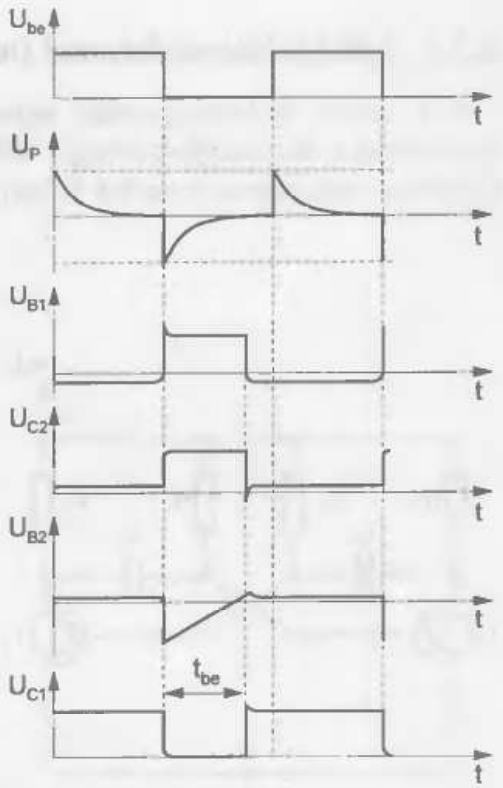
1.3.2. Monostabil billenőfokozat (monostabil multivibrátor)

A monostabil billenőkörnek ***egyetlen stabil állapota van***, azaz bemeneti vezérlő-impulzus nélkül a kimeneti feszültség egy rögzített értéken marad. Ha egy külső vezérlőjellel a másik állapotába billentjük, ezt az állapotát csak meghatározott ideig tartja meg, majd visszabillen stabil állapotába.

Az *1.12.a ábrán* egy monostabil billenőkapcsolás áramköri rajza, az *1.12.b ábrán* a kialakuló jelalakok láthatók.



a) kapcsolási rajz



b) feszültség-idő ábra

1.12. ábra. Monostabil billenőfokozat

A monostabil billenőfokozat működése az előbbi ábrák alapján magyarázható.

- Az áramkör stabil állapotában a T_2 tranzisztor vezet, a T_1 pedig zárva van. Mivel a T_1 kollektora közel tápfeszültségen, bázisa pedig közel födpotenciálon van a C kondenzátor ennek megfelelő polaritású feszültséggel töltődik fel.
- A bemenetre adott négyszögimpulzust a C_d kondenzátor differenciálja, a D dióda pedig a jelnek csak a negatív félperiódusát engedi át. A negatív vezérlőimpulzus a T_2 tranzisztor bázisára jutva lezárja azt, kollektorpotenciálja közel tápfeszültségre kerül, ami a T_1 tranzisztort vezető állapotba hozza. A T_1 nyitásával a C kondenzátor az R ellenálláson keresztül elkezd kisülni, majd ellentétes polaritással feltöltődni. Ha a kondenzátor feszültsége a töltődés során eléri a T_2 nyitásához szükséges szintet, a T_2 kinyit és lezárja a T_1 tranzisztort. Az áramkör ismét stabil állapotba kerül.

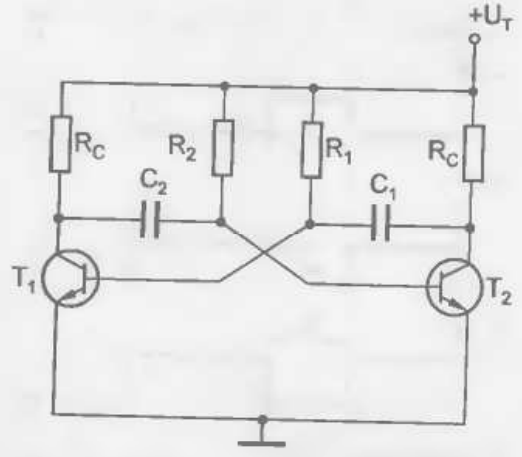
A T_2 tranzisztor bázisára jutó impulzus időtartamát a C és R elemek értéke határozza meg:

$$t_{be} \cong 0{,}7 \cdot R \cdot C$$

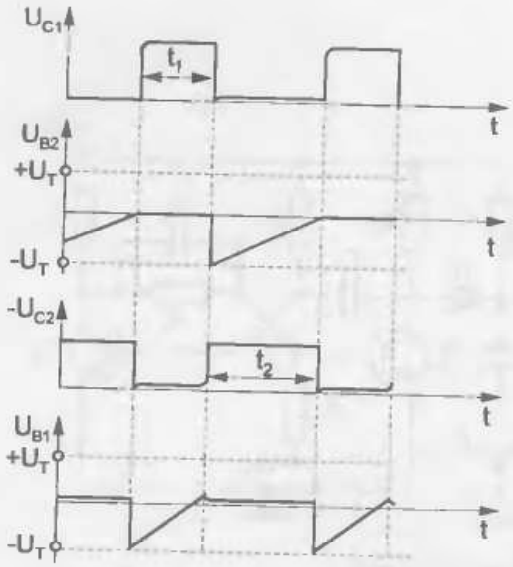
A kimenet akkor is visszabillen a kiszámított kapcsolási idő elteltével, ha a bemeneti impulzus hosszabb mint t_{be}. Ebben az esetben a T_1 tranzisztor a bemeneti impulzus megszűnéséig nyitva marad, és a pozitív visszacsatolás hatástalan. A kapcsolási folyamat végén kell a C kondenzátort az R ellenálláson keresztül feltölteni. Ha a kondenzátor a következő impulzusig nem töltődik fel teljesen, akkor a következő bekapcsolási idő lerövidül. Ahhoz, hogy ez a jelenség 1 %-nál kisebb hibát okozzon, T_1-et legalább $5 \cdot R_C \cdot C$ feléledési időre zárva kell tartani.

1.3.3. Astabil billenőfokozat (astabil multivibrátor)

Az astabil billenőkapcsolás egyetlen stabil állapottal sem rendelkezik. Az astabil multivibrátor négyszögfeszültséget állít elő. Azért kapta a ***multivibrátor*** elnevezést, mert a négyszögfeszültségnek igen sok felharmonikusa van.



a) kapcsolási rajz



b) feszültség-idő ábra

1.13. ábra. Astabil billenőfokozat

Az *1.13.a ábrán* egy monostabil billenőkapcsolás áramköri rajza, az *1.13.b ábrán* a kialakuló jelalakok láthatók. A pozitív visszacsatolás mindkét tranzisztor esetén kapacitív jellegű.
Vizsgáljuk meg az áramkör működését!

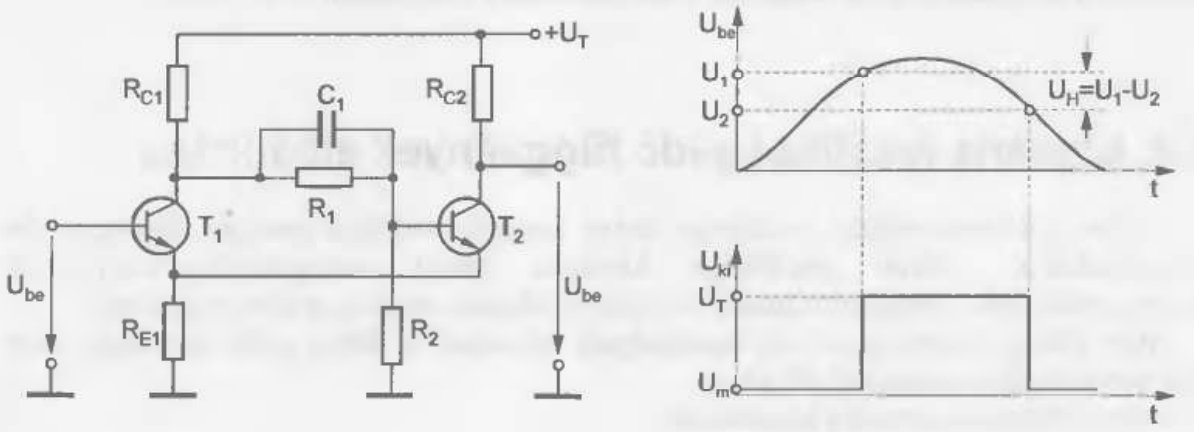
- Feltételezzük, hogy a T_2 tranzisztor vezet, a T_2 zárva van. A C_2 kondenzátor az R_2 ellenálláson keresztül töltődik. Amikor a kondenzátor feszültsége eléri a T_2 tranzisztor nyitófeszültségét, a T_2 kinyit és lezárja a T_1 tranzisztort. Vagyis az áramkör átbillen a másik állapotába.
- A T_2 tranzisztor csak addig vezet, amíg a C_1 kondenzátor az R_1 ellenálláson keresztül nem töltődik fel annyira, hogy a T_1 tranzisztort kinyissa. Ha a T_1 tranzisztor kinyit – a pozitív visszacsatolás miatt – lezárja a T_2-t és az áramkör újra átbillen.

A kapcsolás tehát folyamatosan a két állapot között billeg. Az időtartamok az időzítő elemek értékétől függnek:

$$t_1 \cong 0{,}7 \cdot R_1 \cdot C_1$$
$$t_2 \cong 0{,}7 \cdot R_2 \cdot C_2$$

1.3.4. Schmitt-trigger

A Schmitt-trigger egy olyan bistabil billenőkör, melynek kimeneti jele a bemeneti jel amplitódójának a nagyságától függ. Ez az áramkör tulajdonképpen egy küszöbérték-kapcsoló. Kapcsolási rajza az *1.14.a ábrán* látható.



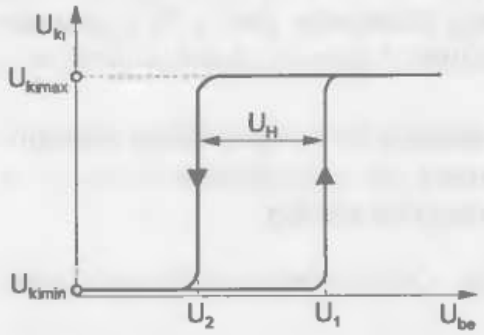
a) kapcsolási rajz *b) be- és kimeneti feszültség*

1.14. ábra: Schmitt-trigger

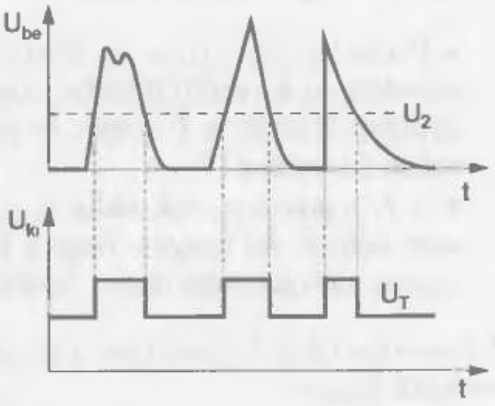
Az áramkör működése a a fenti ábrák alapján magyarázható.

- Alapállapotban a T_1 tranzisztor zárva van, T_2 pozitív nyitófeszültséget kap, így kollektorán közel földpotenciál mérhető. Ha a bemeneti feszültség elér egy adott pozitív U_1 értéket, a T_1 kollektorárama lezárja a T_2 tranzisztort, amelynek kollektorpotenciálja közelítően a tápfeszültséggel lesz egyenlő.

• A bemeneti feszültséget az elért pozitív értékről folyamatosan csökkentve, egy U_2 feszültségértéknél ($U_2 < U_1$) a T_1 tranzisztor lezár és T_2 kinyit.



1.15. ábra. Schmitt-trigger átviteli karakterisztikája



1.16. ábra. Jelek amplitúdó uniformizálása

Megfigyelhető, hogy a billenések nem ugyanazon a feszültségszinten következnek be. Ezt a jelenséget az áramkör ***hiszterézisének*** nevezzük. A triggerkapcsolás hiszterézisén azt a feszültségkülönbséget értjük, amely az U_1 bekapcsolási és az U_2 kikapcsolási küszöbérték között fennáll: $U_H = U_1 - U_2$.

A Schmitt-trigger átviteli karakterisztikáját az *5.15. ábra* szemlélteti. A felhasználástól függően a hiszterézis lehet kívánt, vagy nem kívánt. A gyakorlatban az U_H feszültségkülönbséget általában előre beállítják.

A Schmitt-trigger nagyon fontos, igen sokszor alkalmazott áramkör, többek között különböző amplitúdójú jelek amplitúdó uniformizálására használható (*1.16. ábra*).

1.4. Lineáris feszültség-idő függvények előállítása

Időben lineárisan változó feszültség- illetve áramjel előállítása gyakran szükséges. Az oszcilloszkópok eltérítő feszültsége, bizonyos típusú analóg-digitál konverterek összehasonlító jele , impulzustechnikai időzítési problémák mind ilyen jelet igényelnek.

Ilyen időben lineárisan változó feszültségnek tekinthető a fűrész alakú feszültség, vagy más néven fűrészfeszültség (*1.17. ábra*).

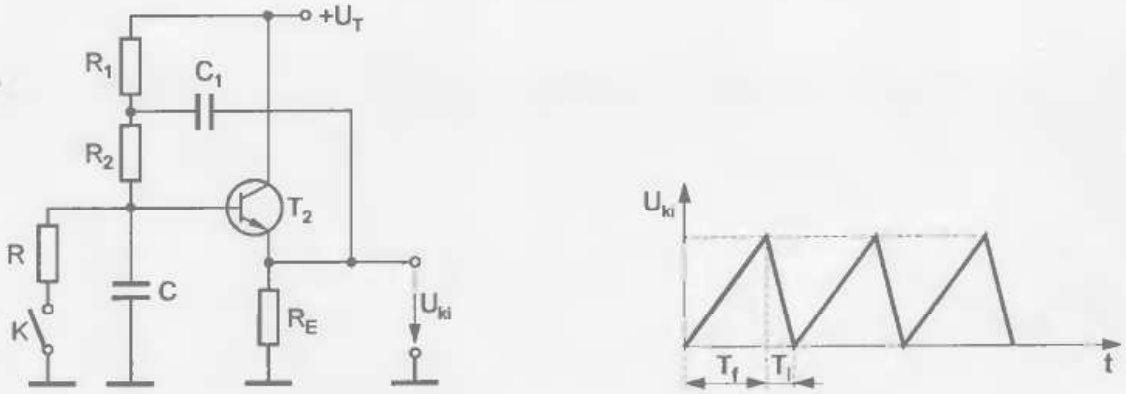
A fűrészfeszültség jellemzői a következők:

- felfutási idő: T_f
- lefutási idő: T_l
- felfutási sebesség: $v_f = \frac{U}{T_f}$
- lefutási sebesség: $v_l = \frac{U}{T_l}$

a) ideális jelalak *b) valós jelalak*

1.17. ábra. Fűrészfeszültség

Lineárisan növekvő feszültséget elvben egy kondenzátornak állandó árammal való töltésével lehet előállítani. Az áramkörök részletes vizsgálata azonban azt mutatja, hogy ez az ideális eset a gyakorlatban sohasem valósítható meg, mindig egy exponenciális jelalak alakul ki. Ennek kezdeti szakasza jó közelítéssel lineárisnak tekinthető. Ezen az elven működik az *1.8. ábrán* látható fűrészjel keltő kapcsolás.



a) kapcsolási rajz *b) kimeneti jelalak*

1.18. ábra. Fűrészfeszültség előállító áramkör

Egyszerű áramkörök esetén (egy telep, ellenállás és kondenzátor) a kondenzátorban kialakuló feszültség nemlineárisan növekszik, mert a kondenzátor feltöltődésével a töltőáram csökken. Az *1.18. ábrán* látható áramkörben a feltöltött C_1 kondenzátor által szolgáltatott segédfeszültség minden időpillanatban akkora értékű, hogy a C kondenzátort töltő áramkörben folyó áram állandó értéken tudjon maradni. A fenti elvek alapján működő áramkört *utánhúzó* (angolul: boot-strap) kapcsolásnak nevezik. Az áramkör működése a K kapcsoló nyitására indul, kimenetén egy lineárisan növekvő feszültséget kapunk. Ha a K kapcsolót zárjuk a C kondenzátor kisül.

Összefoglaló kérdések:

1. Mit nevezünk impulzusnak?
2. Milyen impulzusjellemzőket ismer?
3. Milyen elemek alkotják és hogyan működik egy differenciáló négypólus?
4. Milyen elemek alkotják és hogyan működik egy integráló négypólus?
5. Mi a feltétele az impulzus torzítatlan átvitelének differenciáló- és integráló négypólus esetén?
6. Milyen tulajdonságokkal rendelkezik egy diódás vágóáramkör?
7. Milyen jellemzőkkel rendelkezik egy bistabil multivibrátor?
8. Milyen jellemzőkkel rendelkezik egy monostabil multivibrátor?
9. Milyen jellemzőkkel rendelkezik egy astabil multivibrátor?
10. Mi a hiszterézis a Schmitt-trigger esetén?
11. Milyen alkalmazási lehetőségei vannak a Schmitt-triggernek?
12. Mit nevezünk fűrészfeszültségnek és mik a jellemzői?

2. Logikai áramkörök információelméleti alapjai

Az impulzustechnika fejlődése megalapozta egy új tudományág, a digitális technika kialakulását. A ***digitális technika*** az információ digitális feldolgozásával, előállításával és továbbításával foglalkozik. A félvezető alapú technológiák gyors fejlesztése és ezen belül az integrált áramkörök megjelenése nagy lendületet adott a digitális technika fejlődésének.

Az integrált áramkörben a különböző rendeltetésű aktív és passzív áramköri építőelemeket, valamint a hozzájuk tartozó összekötéseket egyetlen gyártási folyamatban közös félvezető alapon állítják elő. Az integrált áramkört 1960-ban fejlesztették ki a Fairchild és a Texas Instruments cégek szakemberei: R. Noice, valamint J. St. Clair Kilby. G. Moore 1965-ben megállapítja, hogy évenként megduplázódik az egy félvezető (szilícium) lapkán levő integrált tranzisztorok száma. Ez a tendencia bizonyos mértékben napjainkban is tart. Az integrált áramkörök olcsósága, nagy megbízhatósága és kis mérete széleskörű elterjedést eredményezett.

A digitális technikán belül – egymással szoros kölcsönhatásban – fejlődött ki a digitális rendszerek és a digitális áramkörök tervezése. A digitális technika tette lehetővé a számítástechnika dinamikus fejlődését is

2.1. Analóg és digitális mennyiségek

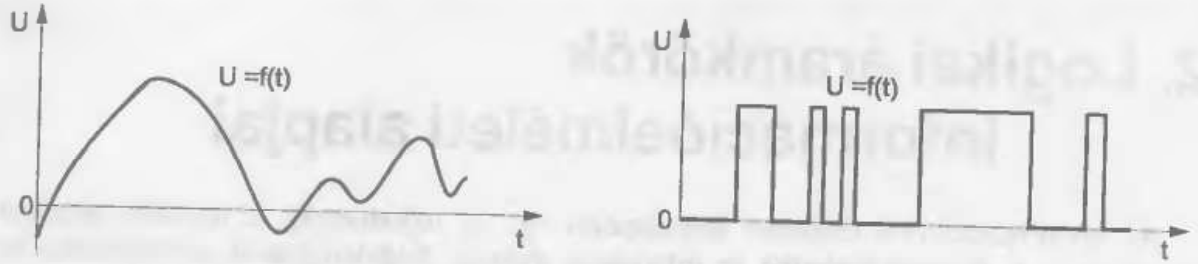
Egy fizikai mennyiség (pl. feszültség vagy áram) másik fizikai mennyiséggel való leképezése két módon történhet.

1. ***Analóg leképezés:***
   - a leképzendő és a leképző mennyiség időbeni lefolyása függvénykapcsolattal megadható;
   - a leképző mennyiség változása folyamatos (bármilyen értéket felvehet).
2. ***Digitális* leképezés:***
   - a leképzendő és a leképző mennyiség időbeni lefolyása függvénykapcsolattal megadható;
   - a leképző mennyiség változása nem folyamatos, hanem ugrásszerű (csak diszkrét értéket vehet fel).

Az analóg leképző mennyiség a megadott határok között tetszőleges értéket vehet fel, míg a digitális leképző mennyiség csak meghatározott értékeket. Ennek megfelelően az információ megjelenési formája is kétféle lehet:

- ***analóg mennyiség*** (*1.1.a ábra*);
- ***digitális mennyiség*** (*1.1.b ábra*).

****Digitális***: – az angol ***digit*** (*számjegy*) szóból származtatható



a) *b)*

1.1. ábra. Analóg jel (a) és digitális jel (b) ábrázolása

2.2. Számrendszerek

2.2.1. A decimális számrendszer

A mindennapi életben a tízes vagy decimális számrendszert használjuk. Ezt már az ókori egyiptomiak is használták, de a mai értelemben csak akkor vált teljessé; amikor a hinduk i.sz. 400 körül bevezették a nullát, és számjegyként alkalmazták. A hinduktól arab közvetítéssel jutott Európába. Itt először 1202-ben Fibonacci Liber Abaci című munkájában ismertette.

Egy N számot matematikailag az r az alábbi fogyó hatványai szerint rendezett több tagú kifejezés ad meg:

$$N = a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \ldots + a_1 \cdot r^1 + a_0 \cdot r^0 + a_{-1} \cdot r^{-1} + \ldots + a_{-(m-1)} \cdot r^{-(m-1)} + a_{-m} \cdot r^{-m} \tag{2.1}$$

ahol:

r – a *számrendszer alapja* vagy *alapszáma* (egy természetes szám),
a_k – 0 és r-1 közötti értékeket felvevő számok,
$k = n, n\text{-}1, \ldots 1, 0, -1; \ldots -(m-1), -m$ egész valós számok.

Az (2.1) kifejezés által meghatározott számot szimbolikusan a következőképpen szokás írni:

$$N = a_n a_{n-1} \ldots a_1 a_0, a_{-1} \ldots a_{-(m-1)} a_{-m} \tag{2.2}$$

Itt a vessző a szám egész és törtrészét választja el. A szám törtrésze, amely a vesszőtől jobbra található, sohasem nagyobb 1-nél. A szám értékét úgy is lehet tekinteni, mint számjegyeinek súlyozott összegét. A ***súlyozást*** az alapszám hatványa adja: a k helyértékű számjegy súlyozása r^k. A legnagyobb helyi értékű szám baloldalt van, a legkisebb helyértékű meg jobboldalt.

A tízes számrendszer alapja, amint az elnevezése is mutatja, $r = 10$. A számábrázoláshoz a 0-tól 9-ig terjedő tíz számjegyet használjuk.

Példaként:

Az $N = 1421,4721$ szám szimbolikus ábrázolása:

$$N = 1 \cdot 10^3 + 4 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0 + 4 \cdot 10^{-1} + 7 \cdot 10^{-2} + 2 \cdot 10^{-3} + 1 \cdot 10^{-4}.$$

2.2.2. Bináris számrendszerek

A kettes vagy bináris számrendszer alapja $r = 2$, és a legkevesebb, vagyis csak két elemet használ a számok ábrázolásához. A kettes számrendszer számjegyei vagy más szóval bit-jei **0** és **1**. A bit elnevezés a bináris számjegy angol megfelelőjének – **binary digit** – rövidítése. Ez a számrendszer tökéletesen összeegyeztethető a kétállapotú áramkörökkel felépített elektronikus számítógépek működésével.

Egy adott bináris szám értékét a (2.1) kifejezés adja meg, ahol $r = 2$ és $a_k = 0$ vagy **1**. Szimbolikus jelölése a (2.2) kifejezés alapján történik. Például:

$$N = 10111,101$$

szimbolikusan felírt bináris szám értéke:

$$N = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}.$$

A (2.1.) kifejezés alapján ki lehet számítani egy megadott bináris szám decimális megfelelőjét. A fenti példa esetén: $1011{,}101_2 = 11{,}625_{10}$ (az indexbe írt tízes számrendszerbeli szám a számrendszer alapját jelzi). Fordítva, a decimális számrendszerből a binárisba való átalakítás (konverzió) valamivel bonyolultabb. A decimális szám kettes alapszámmal való sorozatos osztása után keletkező maradékok adják az adott szám bináris megfelelőjét. Az eljárást az alábbi példa szemlélteti: – legyen az átalakítandó decimális szám **25**.

Osztás 2-vel		*Eredmény*	*Maradék*	
$\frac{25}{2}$	=	12	1	Legkisebb helyértékű bit (LSB–Last Significant Bit)
$\frac{12}{2}$	=	6	0	
$\frac{6}{2}$	=	3	0	
$\frac{3}{2}$	=	1	1	
$\frac{1}{2}$	=	0	1	Legnagyobb helyértékű bit (MSB–Most Significant Bit)

A bináris számot a nyíl irányában olvasott maradékok képezik. Tehát $25_{10} = 11001_2$. Ezzel a szabállyal kizárólag csak az egész decimális számokat lehet binárissá átalakítani. A tizedes törtek bináris törtekké való átalakítása a kettes alapszámmal történő sorozatos szorzáson alapul. Az eredmény átvivendő mennyisége (az egész rész) képezi a keresett bináris törtet.

Legyen például az átalakítandó tizedestört **0,6875**

Szorzás 2-vel		*Átvitel*		*Eredmény*
$0{,}6875 \times 2$	=	1	+	0,3750
$0{,}3750 \times 2$	=	0	+	0,7500
$0{,}7500 \times 2$	=	1	+	0,5000
$0{,}5000 \times 2$	=	1	+	0,0000

A bináris törtet az átvivendő mennyiségek nyíl irányában való felírása fejezi ki, vagyis $0{,}6875_{10} = 0{,}1011_2$.

2.2.2.1. Pozitív és negatív bináris számok

Egy pozitív bináris szám éppen úgy mint egy decimális szám, lehet pozitív vagy negatív. A számítógépekben az előjel (+ plusz, – mínusz) ábrázolása is csak **0** és **1** szimbólumokkal valósulhat meg, úgy, hogy a plusznak **0**, a mínusznak **1** felel meg. Ez az ún. ***előjelbit***, amely után következik a szám abszolút értéke. A *2.1. táblázat* szemlélteti a 4-bites, előjel és abszolút érték ábrázolású bináris számokat, valamint azok decimális megfelelőjét.

Egy másik fontos ábrázolása a pozitív és negatív bináris számoknak az **1**-es ***kiegészítős számábrázolás*** vagy más nevén az ***1-es komplemens***. Ebben a pozitív számok ábrázolása megegyezik az előjeles abszolút érték számábrázolással. A negatív számok ábrázolása viszont eltérő. Egy negatív szám az azonos abszolút értékű pozitív szám komplemense (**1**-es kiegészítője). Ha egy n-bites pozitív szám (az egyszerűség kedvéért egész szám) szimbolikus jelölése:

$$N_P = 0\ a_{n-2}\ a_{n-3} \ldots a_1\ a_0, \tag{2.3}$$

akkor az azonos abszolút értékű negatív számé:

$$N_Q = 1\ \overline{a}_{n-2}\ \overline{a}_{n-3} \ldots \overline{a}_1\ \overline{a}_0. \tag{2.4}$$

Az **1**-es komplemensű bináris szám valós értéke az előjelbit $-(2^{n-1}-1)$ súlyozásával kapható meg. A számítógép működése leegyszerűsödik, ha az előjelbit súlyozása megfelel az előjel nélküli n-bites bináris szám legnagyobb helyértékű bitjének kijáró 2^{n-1} súlyozásával.

Az N_P pozitív szám, valamint az N_Q negatív szám értéke ezzel a súlyozással legyen $N_P^{(1)}$, illetve $N_Q^{(1)}$. A pozitív szám esetén, ha összevetjük a (2.2) és (2.3) szimbolikus alakokat valamint a (2.1) kifejezést, akkor:

$$N_P^{(1)} = N_P \tag{2.5}$$

Decimális szám	Bináris számábrázolások		
	Előjel és abszolút érték	Egyes komplemens	Kettes komplemens
+7	0 1 1 1	0 1 1 1	0 1 1 1
+6	0 1 1 0	0 1 1 0	0 1 1 0
+5	0 1 0 1	0 1 0 1	0 1 0 1
+4	0 1 0 0	0 1 0 0	0 1 0 0
+3	0 0 1 1	0 0 1 1	0 0 1 1
+2	0 0 1 0	0 0 1 0	0 0 1 0
+1	0 0 0 1	0 0 0 1	0 0 0 1
+0	0 0 0 0 1 0 0 0	0 0 0 0 1 1 1 1	0 0 0 0
–1	1 0 0 1	1 1 1 0	1 1 1 1
–2	1 0 1 0	1 1 0 1	1 1 1 0
–3	1 0 1 1	1 1 0 0	1 1 0 1
–4	1 1 0 0	1 0 1 1	1 1 0 0
–5	1 1 0 1	1 0 1 0	1 0 1 1
–6	1 1 1 0	1 0 0 1	1 0 1 0
–7	1 1 1 1	1 0 0 0	1 0 0 1
–8	–	–	1 0 0 0

2.1. táblázat. Pozitív és negatív négybites bináris számok ábrázolása

A negatív szám esetén először is kiszámítjuk az $N_P^{(1)} + N_Q^{(1)}$ összeget, amely az

$$a_k + \bar{a}_k = 1 \qquad (k = n-1, n-2, n-3, \ldots 1, 0)$$

egyenlőség figyelembevételével:

$$N_P^{(1)} + N_Q^{(1)} = \underbrace{111\ldots11}_{n} = \underbrace{100\ldots00}_{n} - 1 = 2^n - 1$$

Végül is a fenti és a (2.5) egyenlőség alapján:

$$N_Q^{(1)} = 2^n - 1 - N_P \,. \tag{2.6}$$

A pozitív és negatív bináris számok legelterjedtebb ábrázolása a ***2-es kiegészítős számábrázolás*** vagy más nevén a ***2-es komplemens***. A pozitív számok ábrázolása azonos a két előbbi számábrázolással. A negatív számok ábrázolása **1**-es komplemensből származik **1** hozzáadásával. Egy n-bites pozitív szám (az egyszerűség kedvéért egész szám) szimbolikus jelölése:

$$M_P = 0\ a_{n-2}\, a_{n-3} \ldots a_1\, a_0 \,, \tag{2.7}$$

egy azonos abszolút értékű negatív számé pedig a következő összeg eredménye:

$$M_Q = 1\ \bar{a}_{n-2}\, \bar{a}_{n-3} \ldots \bar{a}_1\, \bar{a}_0 + 1 \tag{2.8}$$

A 2-es komplemensű szám valós értéke az előjelbit $-(2^{n-1})$ súlyozásával kapható meg. Ha a számítógép az előjelbit 2^{n-1} súlyozását alkalmazza, akkor legyen az M_P és M_Q számok értéke $M_P^{(2)}$, illetve $M_Q^{(2)}$. A pozitív szám esetén a (2.7) és (2.3) szimbolikus alakok, valamint a (2.5) egyenlőség alapján:

$$M_P^{(2)} = M_P, \tag{2.9}$$

negatív szám esetén a (2.8) és (2.6) kifejezésekből kapjuk:

$$M_Q^{(2)} = 2^n - M_P, \tag{2.10}$$

A *2.1. táblázat* szemlélteti a 4-bites pozitív és negatív bináris számok ábrázolását az előbbiekben tárgyalt három legelterjedtebb számábrázolási módszer segítségével.

2.2.3. A hexadecimális számrendszer

A digitális technikában és a mikroszámítógépeknél elterjedt a ***hexadecimális*** (*tizenhatos*) számábrázolás használata is. Ennek a számrendszernek, mint ahogy az elnevezése is mutatja, az alapszáma 16. A számábrázoláshoz szükséges 16 számjegy közül az első 10 a decimális számrendszer 0-tól 9-ig terjedő számjegye. A következő 6 számjegyet (10-től 15-ig) ***A***, ***B***, ***C***, ***D***, ***E*** és ***F*** betűszimbólumok jelölik. Ezért a hexadecimális számábrázolást ***alfanumerikusnak*** is nevezik.

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

2.2. táblázat. A hexadecimális számjegyek és bináris megfelelőik

A hexadecimális-decimális és a fordított átalakítás (*konverzió*) hasonló az előbbiekben bemutatott bináris-decimális, illetve a fordított átalakításhoz. A mikroszámítógépek alkalmazásánál nagyon fontos a hexadecimális-bináris és a bináris-hexadecimális átalakítás is.

A hexadecimális-bináris konverzió esetében, a (2.1) kifejezés alapján és az $r^k = 16^k = 2^{4k}$ egyenlőség figyelembevételével, a hexadecimális szám külön minden számjegyét át lehet alakítani 4-bites bináris számmá (*2.2. táblázat*). Példaként a **B5A** hexadecimális számot így lehet kifejezni:

$$B5A = (1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0)\cdot 2^8 + (0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0)\cdot 2^4 + (1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0)\cdot 2^0$$

vagyis:

$$B5A = 1\cdot 2^{11} + 0\cdot 2^{10} + 1\cdot 2^9 + 1\cdot 2^8 + 0\cdot 2^7 + 1\cdot 2^6 + 0\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0$$

amely végül is szimbolikusan így írható:

$$B5A_{16} = 101101011010_2$$

Tehát egy hexadecimális szám bináris megfelelőjét nagyon egyszerűen kaphatjuk meg:

- a hexadecimális szám minden külön számjegyét átalakítjuk négybites bináris számmá, amelyeket azután a megfelelő sorrendben egymás mellé írunk.

A bináris-hexadecimális átalakítás az előbbi eljárásnak a fordítottja. A bináris számot jobbról balra 4-bites csoportokba, ún. ***tetrádokba*** (angolul: *nibble*) osztjuk (a legnagyobb helyértékű csoport kevesebb számjegyből is állhat). Végül a tetrádok hexadecimális megfelelőit az adott sorrendben egymás mellé írjuk.
Például a **110110011** bináris szám átalakítása:

$$1\ 1011\ 0011_2 = 1B3_{16}$$

A digitális technikában (pl. a mikroszámítógépekben) az információ bitszáma rendszerint 4 többszöröse (4, 8, 12, 16, 24, 32, 40, 48). A bináris-hexadecimális és a fordított átalakítási eljárás egyszerűsége miatt is célszerű a hosszú bináris számokat a négyszerte rövidebb hexadecimális megfelelőjükkel ábrázolni. Csak két hexadecimális számjegy szükséges annak az információnak a kifejezésére, amely 8 bitet foglal magába, és amelyet ***bájt***-nak (***byte***) is neveznek.

2.3. Az információ kódolása

Az információ és az adat szavakat gyakran szinonimaként használják. Sokszor ez nem okoz problémát, de a pontos számítástechnikai terminológiában célszerű különbséget tenni köztük. Az ***információ*** valamely jelenségre vonatkozó értelmes közlést jelent, melynek általában az új (legalábbis akkor számára új) ismereteket szolgáltató része fontos a felhasználó részére. Általános megfogalmazásban az *információ* bizonyos fokú tájékozatlanságot szüntet meg. Az ***adat*** az információnak a digitális rendszerekben (számítógépes rendszerekben) való konkrét megjelenési formája.

Az információ szimbólumok sokaságából áll. Ezek a szimbólumok az emberi beszéd esetén hangok, az írás esetében betűk, míg a digitális áramkörök vagy számítógép esetében számjegyek. Egy szimbólumhalmaz meghatározott rendszere alkotja a ***kódot***, amelyet ***kódszavak*** alkotnak (a szimbólumhalmaz elemeiből alkotott szimbólumsorozat). Két szimbólumhalmaz egymáshoz rendelését ***kódolásnak*** nevezzük. A kódszavak lehetnek fix- és változó szóhosszúságúak. A kódszavak képzésében többféle szimbólum fordulhat elő. Ennek megfelelően egy kód karakterkészlete szerint két fő csoportot különböztethetünk meg:

- ***numerikus kódokat***: – a karakterkészlet csak számokat tartalmaz;
- ***alfanumerikus kódokat***: a karakterkészlet számjegyeket és betűket, valamint írásjeleket tartalmaz.

A következőkben a digitális technikában legelterjedtebb – két szimbólum (**0** és **1**) által felépíthető – ***bináris kódokkal*** foglalkozunk.

2.3.1 Bináris kódolású számrendszerek

A tízes számrendszerben kifejezett adatokat a számítógépbe való bevitelnél bináris számrendszerbe kell átalakítani. A számítógép bináris számrendszerben adja meg az eredményeket, melyeket azután át kell alakítani decimális számrendszerbe. Kis számítógépek esetén az adatok átalakítása rendszerint több időt vesz igénybe, mint az adatfeldolgozás. Az átalakítás megkönnyítése végett nem az egész decimális számot alakítják át bináris számmá, hanem helyiértékenként alakítják át binárissá. Ez az ún. ***binárisan kódolt decimális*** (BCD – Binary Coded Decimal) kód. Minden decimális számjegyet egy négyjegyű bináris szám ábrázol. A legelterjedtebb a ***természetes BCD-kód*** (2.3. táblázat). Ebben a kódban a decimális számokat a megfelelő négybites bináris szám ábrázolja. A számjegyek súlyozása $2^0, 2^1, 2^2, 2^3$ vagyis 1, 2, 4, 8. Ezért ezt 1248 súlyozású BCD-kódnak is nevezik.
Például 47_{10} ebben a BCD-kódban így írható: $47_{10} = 0100\ 0111_{BCD}$
Fordítva, ebben a BCD-kódban kifejezett szám decimális megfelelője:

$$1001\ 0101\ 0011_{BCD} = 953_{10}$$

Egy másik BCD-kód 1242 súlyozású. Ezt ***Aiken-kódnak*** is nevezik. A 0-tól 4-ig terjedő decimális számokat a megfelelő bináris számok, és az 5-től 9-ig terjedő decimális számokat a 11-től 15-nek megfelelő tetrádok ábrázolják (*2.3. táblázat*). Az *Aiken-kód* előnye a kilences komplemens képzése egyszerű bitenkénti komplementálással. A 9-es komplemens hasonló az 1-es komplemenshez. Egy decimális szám 9-es komplemense, ha a szám pozitív, akkor azonos az adott számmal, és ha a szám negatív, akkor az abszolút értékének összege a 9-es komplemenssel 9-et eredményez. A 9-es komplemens jelentősége a BCD-kódolású számok kivonásánál nyilvánul meg.

A ***háromtöbbletes kód*** vagy az ún. ***Stibitz-kód*** (***Excess 3-kód***) is hasonló előnyös tulajdonsággal rendelkezik a 9-es komplemens képzésénél, mint az *Aiken-kód*. A *háromtöbbletes kód* tetrádjai az 1248 súlyozású BCD-kód tetrádjainál hárommal nagyobbak.

A BCD kódolású számok összeadása valamelyest eltér a tiszta bináris számok összeadásától. Abban az esetben, ha egy decimális helyiérték összeadásának eredménye 9-nél nagyobb, vagy átvitel keletkezik, akkor ahhoz, hogy helyes eredményt kapjunk, a helyiértékhez 6-ot kell hozzáadni. Ellenkező esetben az eredmény helyes. Például számítsuk ki BCD-kódban a következő összeadások eredményeit:

$12_{10} + 36_{10} \cdot 15_{10} + 36_{10}$ és $17_{10} + 39_{10}$

$0001\ 0010_{BCD}$	12_{10}				
$0011\ 0110_{BCD}$	$+36_{10}$				
$0100\ 1000_{BCD}$	48_{10}				
$0001\ 0101_{BCD}$	15_{10}		$0001\ 0111_{BCD}$	17_{10}	
$0011\ 0110_{BCD}$	$+36_{10}$		$0011\ 1001_{BCD}$	$+39_{10}$	
$0100\ 1011_{BCD}$			$0101\ 0000_{BCD}$		
0110_{BCD}	$+6_{10}$		0110_{BCD}	$+6_{10}$	
$0101\ 0001_{BCD}$	51_{10}		$0101\ 0110_{BCD}$	56_{10}	

Decimális szám	Bináris kódok					
	BCD kód súlyozás: 8421	Aiken kód súlyozás: 2421	Excess-3 kód	Gray-kód	Johnson-kód	Hamming-kód
0	0000	0000	0011	0000	00000	0000000
1	0001	0001	0100	0001	00001	1101001
2	0010	0010	0101	0011	00011	0101010
3	0011	0011	0110	0010	00111	1000011
4	0100	0100	0111	0110	01111	1001100
5	0101	1011	1000	0111	11111	0100101
6	0110	1100	1001	0101	11110	1100110
7	0111	1101	1010	0100	11100	0001111
8	1000	1110	1011	1100	11000	1110000
9	1001	1111	1100	1101	10000	0011001

2.3. táblázat. A decimális számok bináris kódjai

Lyukszalag-lyukasztók és –leolvasók vagy térbeli kódoló átalakítók esetén nem ajánlatos az előbbi kódok alkalmazása. A kódolás vagy az olvasás eredményének egyértelműsége az egyik karakterről a másikra való átmenetnél nem biztosított. Példaként a decimális 7-ből (**0111**) a 8-ba (**1000**) való átmenetet nézve, 4 bit változik meg. Ha valamelyik átmenet a tűrések miatt előbb változik meg, mint a többi, akkor a bináris szám szekvenciája nem egyezik a valóságos decimális szám szekvenciájával. A ***ciklikusan permutált kódok***, amelyeknél csak egy bit változik meg egy-egy egységnyi decimális szám megváltozására, ezt a hibát küszöbölik ki. A legismertebb ciklikusan permutált kód a ***Gray-kód*** (*2.4. táblázat*). A Gray-kód a legkisebb helyértékű számjegyek szimmetriatengelyek körüli sorozatos tükrözéséből származik, mindig **1**-gyel növelve a tükrözött helyértékeket.

Decimális szám	Gray-kód				
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	1	Szimmetriatengely
3	0	0	1	0	
4	0	1	1	0	Szimmetriatengely
5	0	1	1	1	
6	0	1	0	1	
7	0	1	0	0	
8	1	1	0	0	Szimmetriatengely
9	1	1	0	1	
10	1	1	1	1	
11	1	1	1	0	
12	1	0	1	0	
13	1	0	1	1	
14	1	0	0	1	
15	1	0	0	0	

2.4. táblázat. Négybites Gray-kód

A ***Johnson-kód*** egy ötbites kód amelynek jellemzője, hogy mindig csak egy elemben tér el két szomszédos kódszó, így a Gray-kódhoz hasonló előnyökkel rendelkezik (*2.3. táblázat*).

A ***Hamming-kód*** egy különleges bináris kód amelynek jellemzője, hogy bináris adatátvitel során hibajavításra alkalmas (*2.3. táblázat*).

2.4. A digitális adatok ellenőrzése és javítása

Az adatok digitális rendszereken való átvitele és tárolása zavarokkal járhat, amelyek az átviteli vezetékekre és a tárak területére hatnak. A bináris helyértékeknek ezáltal okozott változásai az információ előállításában hibát okoznak. Ahhoz, hogy ilyen hibákat felismerjünk és adott esetben javíthassunk, a hasznos információt ellenőrző információval kell kiegészíteni (információ redundáns előállítása). (*Az információ ismételgetése is lehetne egyfajta eljárás, de kimutatható, hogy igen kevéssé hatékony*). Egy ilyen redundáns információ (kódszó) tehát két részből áll:

- a hasznos információt hordozó bitekből,
- nem hasznos információt hordozó bitekből (redundancia bitekből).

Az így kialakított kódrendszerek két csoportba sorolhatók:

- ***hibaellenőrző*** kódrendszerek (angolul kifejezve: – *error detecting code*),
- ***hiba-javító*** kódrendszerek (angolul kifejezve: – *error correcting code*).

Valamely kód-szóban felismerhető illetve korrigálható hibák számának mértéke a redundáns kód ***Hamming***-féle ***h*** távolsága. A Hamming-távolságot a kódszavakra és a kódra egyaránt értelmezhetjük.

Két kódszó Hamming-távolsága: – az egyik kódszó hány elemét kell megváltoztatni, ahhoz, hogy a másik kódszóval megegyezzék.

☞ Példa: Határozzuk meg a következő két kód-szó *Hamming-távolságát*!

	7	6	5	4	3	2	1	0
X_1	1	0	0	0	1	1	0	0
X_2	1	0	1	0	1	0	0	0

Megfigyelhető, hogy az X_1 és X_2 kód-szó két elemét kell megváltoztatni ahhoz, hogy $X_1 \equiv X_2$ legyen. Tehát a *Hamming-távolság* ebben az esetben $h = 2$.

Kód Hamming-távolsága: – a kódot alkotó kódszavak közötti legkisebb *Hamming-távolság*.

A következőkben megvizsgáljuk hogyan jellemzi a *Hamming-távolság* egy kód hibaellenőrző és a hibajavító képességét.

☞ Példa: Határozzuk meg egy háromelemes kód (*2.5. táblázat*) *Hamming-távolságát*!

	2	1	0
X_1	0	0	0
X_2	0	0	1
X_3	0	1	0
X_4	0	1	1
X_5	1	0	0
X_6	1	0	1
X_7	1	1	0
X_8	1	1	1

2.5. táblázat.

1. Eset: – ha minden kódszót felhasználunk adat-továbbítására, akkor a *Hamming-távolság* $h = 1$. Ebben az esetben egy téves adattovábbítást nem tudunk észrevenni. Egy bináris számot hiába vizsgálunk akármeddig, nem jövünk rá, hogy hibás-e vagy sem, hiszen bármilyen kombinációt látunk is, az mind megengedett kódszó.

2. Eset: – ha a kód X_2, X_3, X_5 és X_8 jelű kódszavait használjuk fel adat-továbbításra, akkor a *Hamming-távolság* $h = 2$. Ebben az esetben ha a beérkezett kódszó egy eleme megváltozik, a vevő észleli a hibás kódszót.

3. Eset: – ha a kód X_1 és X_8 jelű kódszavait használjuk fel adat-továbbításra, akkor a *Hamming-távolságra* $h = 3$ adódik. Ebben az esetben nemcsak hibaellenőrzést tudunk megvalósítani, hanem lehetőség van egy bit hibája esetén – megfelelő áramkör segítségével – hibajavításra is; (*az áramkör a X_2, X_3 és X_5 jelű kód-szót X_1 jelűvé a X_4, X_6 és X_7 jelű kód-szót X_8 jelűvé kell alakítania*).

A példa kapcsán a következő következtetések vonhatók le:

- a redundancia bitek *növelik* a kód *Hamming-távolságát*,
- a hibafelfedés feltétele: – $h \geq 2$,
- a hibajavítás feltétele: – $h \geq 3$.

2.4.1. Hibaellenőrző és hibajavító kódok

A redundancia legegyszerűbb gyakorlati formája az ún. ***paritás ellenőrző bit*** (*paritás-bit*) használata. Egy n bitből álló „információcsomaghoz" egy $n+1$-ik bitet is hozzáadnak. A paritásbit az információ érdemi részében található logikai 1-ek páros vagy páratlan számáról tudósít. A paritás-bit értékét úgy határozhatjuk meg, hogy a redundáns kód szóban a bitek számértékét összeadjuk és az vagy ***páros*** („*even parity*") vagy ***páratlan*** („*odd parity*"). A *2.6. táblázat* bináris kódok paritás-bittel való kiegészítését szemlélteti. A paritásbit itt az egyesek számát párosra egészíti ki. Megfigyelhető, hogy a kód Hamming-távolsága h = 1-ről h = 2-re változott. Ezzel egy 1-bites hibát fel lehet ismerni, de nem lehet korrigálni mert nem lokalizálható.

	Eredeti kód				*Paritás-bit*	*Új kód*				
X_1	0	0	0	0	0	0	0	0	0	0
X_2	0	0	0	1	1	0	0	0	1	1
X_3	0	0	1	0	1	0	0	1	0	1
X_4	0	0	1	1	0	0	0	1	1	0
X_5	0	1	0	0	1	0	1	0	0	1
X_6	1	0	0	0	1	1	0	0	0	1

2.6. táblázat. Hibaellenőrzés paritás-bittel

Ez az egyszerűen képezhető és ellenőrizhető paritás-bit rendszer a gyakorlatban igen hasznos. Alkalmas arra, hogy minden – az információtartalomban bekövetkező – páratlan számú meghibásodást detektálhatóvá tegyen. Maga az eljárás igen egyszerű: az információ elküldésénél, elrakásánál képezzük a paritásbitet, amit természetesen szintén elküldünk vagy elrakunk. A felhasználásnál az információs bitek alapján újraképezzük a paritásbitet és egybevetjük azzal, ami eredetileg volt. Ha ezek megegyeznek, akkor tudjuk, hogy vagy nem következett be hiba, vagy pedig páros számú bithiba fordult elő.

A hibajavító képességű kódrendszerek illusztrálására a *Hamming-féle* kód egy egyszerű változatát említjük meg, mely **4** információs bithez **3** speciálisan képzett paritásbitet ad, és ennek következtében egy hiba helye megállapítható. A kódolás a következőképpen történik:

hibás	hiba helye
P_5	X_5
P_6	X_6
P_7	X_7
P_5P_6	X_2
P_5P_7	X_3
P_6P_7	X_4
$P_5P_6P_7$	X_1

2.7. táblázat. A hiba helyének megállapítása

- az X_1, X_2, X_3, X_4 információs bitekből X_5, X_6, X_7 paritásbitet képezünk a következő szabály szerint:

$$X_1 + X_2 + X_3 + X_5 \rightarrow \textit{páros}$$
$$X_1 + X_2 + X_4 + X_6 \rightarrow \textit{páros}$$
$$X_1 + X_3 + X_4 + X_7 \rightarrow \textit{páros}$$

legyen. Ennek megfelelően az **1010** információ paritásbitekkel kibővítve: **1010010** lesz.

Az információ vételekor tehát az X_1, X_2, X_3, X_4 információs bitekből a fenti szabály szerint P_5, P_6, P_7 értékekkel összehasonlítjuk. A hiba helyének megállapítását a *2.6. táblázat* mutatja. Természetesen, ha a hibás bitek száma egynél több, a rendszer helytelenül fog javítani.

Megjegyzés: Ha n információs bithez k paritásbitet adunk, és a kódrendszer egy hibát tud javítani, akkor a következő egyenlőtlenségnek kell fennállnia: $2^k \geq (n+k)+1$.

A paritásbittel rá kell tudni mutatni a hibás bitre, amelyik vagy az információs bitek, vagy a paritásbitek között foglal helyet. Ezen túlmenően a hibamentes állapotot is jeleznie kell.

Összefoglaló kérdések és feladatok:

1. Mi a különbség az analóg és a digitális mennyiségek között?
2. Ábrázolja szimbolikusan a következő decimális számokat:
$$N_1 = 154{,}14\,;\ N_2 = -3414{,}254\,;\ N_3 = 4876{,}4721\,.$$
3. Alakítsa át a következő decimális számokat bináris számmá és írja fel szimbolikus formában őket: $N_1 = 226\,;\ N_2 = -1115{,}68\,;\ N_3 = 894{,}456\,.$
4. Fejezze ki decimális és bináris formában a következő hexadecimális számokat:
$$N_1 = C3F\,;\ N_2 = B7A\,;\ N_3 = FF2\,.$$
5. Miért van szükség az információ kódolására?
6. Mi a különbség és a hasonlóság a *BCD*-kód és a *Gray*-kód között?
7. Miért van szükség a digitális adatok ellenőrzésére és javítására?
8. Mi a Hamming-távolság?
9. Mi a feltétele egy kódszó hibafelfedésének és mi a hibajavítás feltétele?

3. Logikai algebra

3.1. Alapfogalmak

A szaktudományok a valóság egy-egy területét kívánják megismerni, a logika viszont a megismerés feltételeit vizsgálja. A logika – melynek tárgya a *gondolkodás* – segíti az egyes tudományágakat a valóság megismerésében, a megismert tényekből való helyes következtetések elérésében. A logika feladata a helyes gondolkodásformák és a helyes következtetési szabályok kidolgozása. A következtetések folyamán alapfeltételekből indulunk ki és ítéletek alkotásán keresztül jutunk el a következtetésig. Ebből következik, hogy a logika a gondolkodás tárgyát képező konkrét problémáktól, tartalmi információktól függetleníti magát és a gondolkodásban részt vevő elemek vagy megállapítások közös, a következtetések szempontjából lényeges tartalmát használja fel. Ez a közös tartalom vagy közös jellemző az állítások igazságértéke, ami alatt a klasszikus kétértékű logikában azt a tényt értjük, hogy egy kijelentés ***igaz*** vagy nem igaz (***hamis***). Ezzel a közelítéssel kíván a logika a gondolkodás, a valóság hű tükrözésének eszközeként működni.

George Boole (1815-1864) angol matematikus volt az, aki a logikai törvényszerűségeket a matematika nyelvén fogalmazta meg, lehetővé téve a logikai következtetések mechanikus levezetését. Emiatt a logikai algebrát ***Boole-algebrának*** is nevezik, bár a teljes Boole-algebrának ez csak egy kis részét képezi. A kétértékű logika gyakorlati alkalmazásához igen alkalmas a kettes számrendszer, mivel két számjegyet használ fel az adatok reprezentálására. Igen egyszerűen megvalósítható az *igaz* és a *hamis* kijelentés hozzárendelése a két számjegyhez. Ennek megfelelően az igaz kijelentés logikai értéke „**1**", a hamis kijelentés logikai értéke pedig „**0**".

A logikai algebra a hagyományos algebrához hasonlóan változókat és állandókat használ. A logikai algebra csak kétféle állandót ismer: a **0**-t és az **1**-et. A logikai algebra változói olyan mennyiségek, amelyek a **0** és az **1** állapotokat vagy értékeket vehetik fel. A változók között, illetve a változók és az állandók között összefüggéseket adhatunk meg.

3.2. Logikai függvények

A logikai feltételek és az események közötti kapcsolatok matematikailag ***logikai függvényekkel*** írhatók le (*3.1. ábra*). A matematikai függvényekhez hasonlóan itt is megkülönböztetünk független és függő változókat. A logikai feltételeket ***független változóknak***, a hatásukra bekövetkező eseményeket ***függő változóknak*** nevezzük.

3.1. ábra. A logikai függvény alapsémája

A változókat általában az *ABC* nagybetűivel jelöljük. A változók igaz voltát jelölésmentes nagybetűvel (pl. A, B) a hamis voltát a nagybetű felé húzott vonallal jelöljük (pl. $\overline{A}, \overline{B}$).
A függvénykapcsolatokat logikai szimbólumokkal jelöljük:

„·" az ***ÉS*** kapcsolat (*használják a* $\wedge$ *jelet is*)
„+" a **VAGY** kapcsolat (*használják a* $\vee$ *jelet is*), stb.

A logikai függvények esetében a független változók száma meghatározza a függvénykapcsolatok számát. Mivel a független változók két értéket vehetnek fel (**0** és **1**), ***n*** darab független változó esetén 2^n kombinációt különböztethetünk meg. A függvénykapcsolatok száma:

$$N = 2^{2^n}.$$

A logikai függvényeket csoportosíthatjuk:
- a logikai változók *időbeni függése* szerint;
- a logikai változók *száma* szerint.

A változók időbeni változása szerint lehetnek:

1. ***időfüggetlen logikai függvények:*** – a függő változó értéke csak a független változó értékétől függ; általános alakban:

$$F = f(X_1, X_2, ..., X_n);$$

Ezeket a függvényeket valósítják meg a *kombinációs logikai hálózatok.*

2. ***időfüggő logikai függvények:*** – a függő változó aktuális értékét nemcsak a független változók adott időpontban felvett értéke, hanem más időpillanatban felvett értékei is meghatározzák; általános formában:

$$F = f(X_{1t}, X_{2t}, ..., F_{t-1}, X_{1(t-1)}, ...).$$

Ezeket a függvényeket megvalósító hálózatokat nevezzük *sorrendi* vagy *szekvenciális hálózatoknak.*

A független változók száma szerint lehetnek:
- ♦ egyváltozós logikai függvények: –csak elméleti jelentőségük van, a gyakorlatban ritkán fordulnak elő;
- ♦ kétváltozós logikai függvények: – két független változóval rendelkeznek;
- ♦ többváltozós logikai függvények: – „*n*" számú független változót tartalmaznak, a gyakorlatban ezekkel találkozunk a leggyakrabban.

3.2.1. Logikai függvények leírásmódjai

A logikai függvények többféle módon adhatók meg. Ezek a következők:

a) ***szöveges megadási mód:*** – a független változók kombinációit, a logikai kapcsolatot, valamint a függő változó értékét szavakkal fogalmazzuk meg;

b) ***táblázatos leírásmód:*** – a független változók értékvariációit és a függvénykapcsolat hatására létrejövő függő változók értékeit egy sorba írjuk egy függőleges vonallal elválasztva (*3.2.a ábra*). Ez egy olyan értéktáblázat, amely tartalmazza a függvény értékét minden lehetséges interpretáció esetén. Mivel a táblázat a feltételek és események közötti logikai igazságokat rögzíti, ezért ***igazságtáblázatnak*** nevezzük:

c) ***logikai vázlat:*** – a függvénykapcsolatot az őt megvalósító áramköri szimbólumokkal ábrázoljuk (lásd a *3.2.b ábrát*);

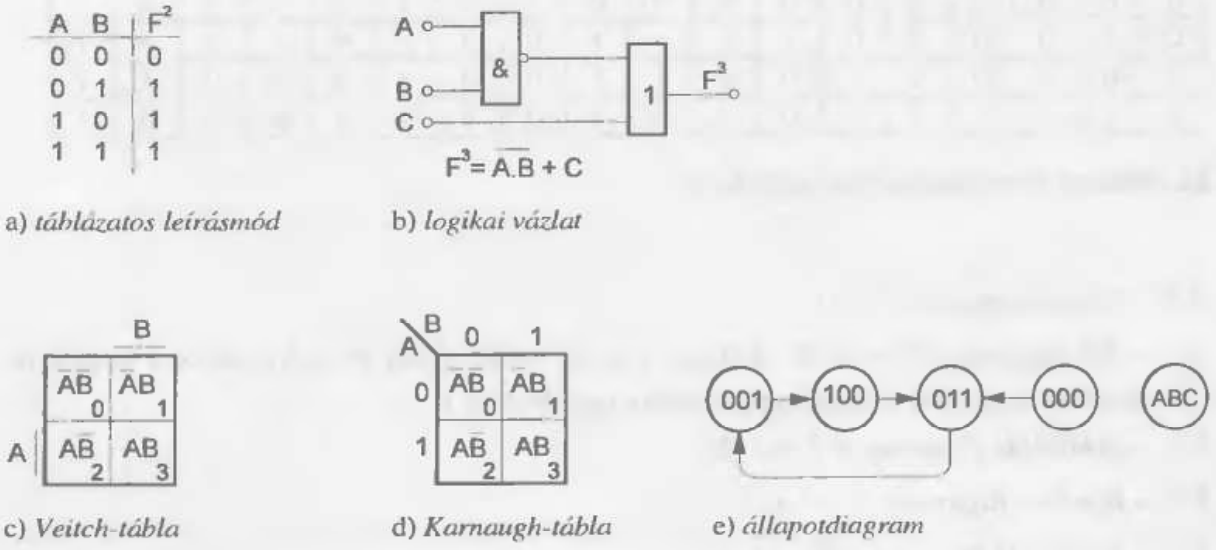
d) ***algebrai alak:*** – a független változókat a függvénykapcsolatra jellemző műveleti szimbólumokkal kapcsoljuk össze;

$$F^3 = (A \cdot B \cdot \overline{C} + \overline{B} \cdot C) \cdot \overline{A} \cdot B$$

Megjegyzés: – a fenti példában a "·" ÉS, a " + " VAGY logikai kapcsolatot jelent.

e) ***grafikus megadási mód:*** – többféle ábrázolásmód közül csak hármat ismertetünk.

1. Veitch-tábla: – a függő változók értékeit egy cellákból álló diagramban ábrázoljuk; a független változókat a diagram kerete mentén jelöljük. Azokban a sorokban és oszlopokban, ahol jelölés van a független változó *igaz* értékű. A *3.2.c ábrán* egy kétváltozós tábla látható, melybe szemléltetésül a celláknak megfelelő változók állapotait is jelöltük.
2. Karnaugh-tábla: – a Veitch-táblától annyiban különbözik, hogy a független változók értékvariációit tüntetjük fel a tábla kontúrjai mentén. Erre láthatunk példát a *3.2.d ábrán*, melybe szemléltetésül a celláknak megfelelő változók állapotait is jelöltük.
3. Állapotdiagram: – az időfüggő logikai függvények leírására alkalmas. A változók aktuális értékeit körökben jelezzük. A köröket összekötő irányított vonalak a változás irányát jelölik. A jelölésmódot bemutató példát a *3.2.e ábra* szemlélteti.



a) *táblázatos leírásmód* b) *logikai vázlat*

c) *Veitch-tábla* d) *Karnaugh-tábla* e) *állapotdiagram*

3.2. ábra. Logikai függvények leírásmódjai

3.2.2. Egyváltozós logikai függvények

Egy független változó esetén négy logikai függvénykapcsolat határozható meg. Ezek a következők (*lásd a 3.1. táblázatot*):

F_0^1 – ***soha függvény***: $F_0^1 = 0$; a függő változó **0** értékű, függetlenül a független változó értékétől (az F alsó indexe a függvénykapcsolat sorszámát jelöli);

F_1^1 – ***negáció (tagadás) függvény***: a függő változó értéke mindig a független változó ellentétes értékét (negáltját) veszi fel (pl. $F_1^1 = \overline{A}$ alakban írható fel);

F_2^1 – ***ismétlő függvény***: a függő változó mindig a független változó értékét veszi fel (pl. $F_2^1 = A$ alakban írható fel);

F_3^1 – ***mindig függvény***: a függő változó értéke nem függ a független változó értékétől (pl. $F_3^1 = 1$ alakban írható fel).

A	F_0^1	F_1^1	F_2^1	F_3^1
0	0	1	0	1
1	0	0	1	1

3.1. táblázat. Egyváltozós logikai függvények

3.2.3. Kétváltozós logikai függvények

Két független változó esetén a logikai függvénykapcsolatok száma $N = 2^{2^n} = 2^{2^2} = 16$ db. A függvénykapcsolatok igazságtáblázatát a *3.2. táblázat* tartalmazza.

A	B	F_0^2	F_1^2	F_2^2	F_3^2	F_4^2	F_5^2	F_6^2	F_7^2	F_8^2	F_9^2	F_{10}^2	F_{11}^2	F_{12}^2	F_{13}^2	F_{14}^2	F_{15}^2
0	***0***	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	***1***	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	***0***	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	***1***	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

3.2. táblázat. Kétváltozós logikai függvények

F_0^2: – ***Soha függvény*** $F_0^2 = 0$.

F_1^2: – ***ÉS függvény*** $F_1^2 = A \cdot B$. A függő változó értéke akkor és csakis akkor **1** értékű, ha mindkét független változó logikai értéke egyidejűleg **1**.

F_2^2: – ***Inhibitáló függvény*** $F_2^2 = A \cdot \overline{B}$.

F_3^2: – ***Ismétlés függvény*** $F_3^2 = A$.

F_4^2: – ***Inhibitáló függvény*** $F_4^2 = \overline{A} \cdot B$.

F_5^2 : – ***Ismétlő függvény*** $F_5^2 = B$.

F_6^2 : – ***Antivalencia (KIZÁRÓ VAGY) függvény*** $F_6^2 = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$.

F_7^2 : – ***VAGY függvény*** $F_7^2 = A + B$. A független változó **1** értékű, ha bármelyik független változó egyenként vagy együttesen is **1** értékű. A kapcsolat műveleti szimbóluma "+".

F_8^2 : – ***VAGY NEM (NOR) függvény*** $F_8^2 = \overline{A + B}$. A logikai függvény értéke akkor és csakis akkor **1**, ha mindkét független változó logikai értéke **0**.

F_9^2 : – ***Ekvivalencia függvény*** $F_9^2 = A \cdot B + \overline{A} \cdot \overline{B} = A \Theta B$. A függő változó értéke **1**, ha a független változók logikai értéke megegyezik. A kapcsolat műveleti szimbóluma "Θ".

F_{10}^2 : – ***Negáció függvény*** $F_{10}^2 = \overline{B}$. A függő változó értéke mindig a független változó ellentétes értékét veszi fel. A kapcsolat műveleti szimbóluma " ¯ " (*felülvonás*).

F_{11}^2 : – ***Implikáció függvény*** $F_{11}^2 = A + \overline{B}$.

F_{12}^2 : – ***Negáció függvény*** $F_{12}^2 = \overline{A}$.

F_{13}^2 : – ***Implikáció függvény*** $F_{13}^2 = \overline{A} + B$.

F_{14}^2 : – ***ÉS NEM (NAND) függvény*** $F_{14}^2 = \overline{A \cdot B}$. A függvény logikai értéke akkor és csakis akkor **0**, ha mindkét független változó értéke **1**.

F_{15}^2 : – ***Mindig függvény*** $F_{15}^2 = 1$. A függvény logikai értéke **1**, függetlenül a független változók értékétől.

A *3.2. táblázatot* figyelmesen tanulmányozva megállapíthatjuk, hogy ha az ekvivalencia (F_7^2) és az antivalencia (F_8^2) függvény közé egy képzeletbeli szimmetriavonalat húzunk, a vonaltól azonos távolságra levő függvények egymás negáltjai.

Duál tétel: – ha a logikai ÉS műveletet VAGY művelettel, valamint a **0**-t **1**-gyel (vagy **1**-et **0**-val) helyettesítjük, az eredeti függvény ***duál függvényét*** kapjuk meg.

3.2.4. Többváltozós logikai függvények

A gyakorlati problémák megoldása során a legtöbbször többváltozós logikai függvényekkel találkozhatunk. Kettőnél több független változó esetén, az előzőekben már megismert függvénykapcsolatokon kívül még számos függvénykapcsolat jön létre, amelyekkel terjedelmi okokból itt nem foglalkozunk.

3.3. A logikai algebra szabályai és alkalmazásuk

Egy adott gyakorlati problémát ha közvetlenül algebrai alakban megadott logikai függvény formájában írunk le, szinte elkerülhetetlen a redundancia (*túlhatározottság*).

Az algebrai formában megadott logikai függvények esetén a logikai algebra (Boole-algebra) olyan azonosságokat illetve szabályokat fogalmazott meg, melyekkel ezek a függvények egyszerűbb alakra hozhatók.

3.3.1. A logikai algebra szabályai

1. ***Kommutatív szabály:*** azonos logikai kapcsolatban levő változók sorrendje tetszőleges.

$$A+B=B+A$$
$$A\cdot B=B\cdot A \qquad (3.1)$$

2. ***Asszociatív szabály:*** a művelet eredménye nem függ a műveletvégzés sorrendjétől.

$$(A+B)+C=(B+C)+A=(A+C)+B$$
$$(A\cdot B)\cdot C=(B\cdot C)\cdot A=(A\cdot C)\cdot B \qquad (3.2)$$

1. ***Disztributív szabály:***

$$A+(B\cdot C)=(A+B)\cdot(A+C)$$
$$A\cdot(B+C)=(A\cdot B)+(A\cdot C) \qquad (3.3)$$

3.3.2. A logikai algebra alaptételei

A logikai algebra alaptételei a következők:

$$A\cdot 0=0$$
$$A+0=A \qquad (3.4)$$
$$A\cdot 1=A$$
$$A+1=1 \qquad (3.5)$$
$$A\cdot A=A$$
$$A+A=A \qquad (3.6)$$
$$A\cdot\overline{A}=0$$
$$A+\overline{A}=1 \qquad (3.7)$$
$$\overline{\overline{A}}=A \qquad (3.8)$$
$$A\cdot(B+A)=A$$
$$A\cdot B+A=A \qquad (3.9)$$
$$\overline{A\cdot B}=\overline{A}+\overline{B}$$
$$\overline{A+B}=\overline{A}\cdot\overline{B} \text{ (De Morgan-tétel)} \qquad (3.10)$$

A 3.4 – 3.8 alaptételek az igazságtáblázatból (*3.2. táblázat*) egyszerűen beláthatók.
A 3.9 alaptétel ($A\cdot(B+A)=A$) bizonyítása a következő:

- $A\cdot(B+A)=A\cdot B+A\cdot A$ (a disztributív szabály szerint).
- $A\cdot(B+A)=A\cdot B+A\cdot A=A\cdot B+A$ (a 3.6 alaptétel alapján).
- Felhasználva a 3.3 és 3.5 összefüggéseket: $A\cdot B+A=A\cdot(B+1)=A\cdot 1=A$.
- Következésképpen: $A\cdot(B+A)=A$.

A 3.10. alaptételt ***De Morgan-tételnek*** nevezzük. Bizonyítása a következő:

• A bizonyítást igazságtábla segítségével végezzük. Felírjuk az egyenletek jobb és bal oldalán található összefüggések igazságértékét (*3.3. táblázat*).

A	B	$\overline{A+B}$	$\overline{A}+\overline{B}$	$\overline{A\cdot B}$	$\overline{A}\cdot\overline{B}$
0	0	1	1	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	0	0	0	0

3.3. táblázat: A De Morgan-tétel bizonyítása

• A táblázatból látható, hogy $\overline{A\cdot B}=\overline{A}+\overline{B}$ és $\overline{A+B}=\overline{A}\cdot\overline{B}$.

A tétel tetszőleges számú független változóra igaz. Általánosan felírva:

$$\overline{A\cdot B\cdot C\cdot .....}=\overline{A}+\overline{B}+\overline{C}+.....$$
$$\overline{A+B+C+.....}=\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot.... \qquad (3.11)$$

3.3.3. A logikai algebra alkalmazása

A következő megoldott feladatokban a logikai algebra szabályait és tételeit alkalmazzuk.

1. **Feladat:** Egyszerűsítsük az alábbi logikai függvényt!

$$F^3=A\cdot(B+\overline{A})+(A+C)\cdot C+A$$

Megoldás:

• A disztributív szabályt alkalmazva: $F^3=A\cdot B+A\cdot\overline{A}+A\cdot C+C\cdot C+B$.

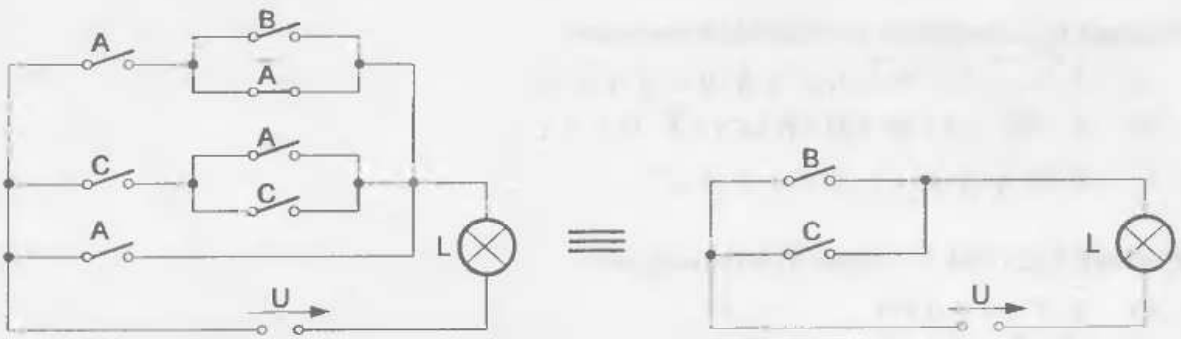
• Mivel $A\cdot\overline{A}=0$, $C\cdot C=C$, $\Rightarrow$ $F^3=A\cdot B+A\cdot C+C+B$.

• Átrendezve az egyenletet: $F^3=B\cdot(A+1)+C\cdot(A+1)$.

• Mivel $A+1=1$, kapjuk: $F^3=B\cdot 1+C\cdot 1$.

• Mivel $B\cdot 1=B$ és $C\cdot 1=C$, kapjuk: $F^3=B+C$.

A *3.3. ábra* az eredeti és az egyszerűsített logikai függvényt megvalósító áramkört szemlélteti.



3.3. ábra. Logikai függvényt megvalósító áramkör egyszerűsítése

2. **Feladat:** Igazoljuk a következő azonosságot!

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

Megoldás:

- A disztributív szabályt alkalmazva:
$$(A+\overline{B}\cdot\overline{C})\cdot(\overline{A}+C)+\overline{A}\cdot B\cdot\overline{C}=A\cdot\overline{A}+\overline{A}\cdot\overline{B}\cdot\overline{C}+A\cdot C+\overline{B}\cdot\overline{C}\cdot C+\overline{A}\cdot B\cdot\overline{C}.$$
- Mivel, $A\cdot\overline{A}=0$ és $C\cdot\overline{C}=0$ következik:
$$A\cdot\overline{A}+\overline{A}\cdot\overline{B}\cdot\overline{C}+A\cdot C+\overline{B}\cdot\overline{C}\cdot C+\overline{A}\cdot B\cdot\overline{C}=\overline{A}\cdot\overline{B}\cdot\overline{C}+A\cdot C+\overline{B}\cdot 0+\overline{A}\cdot B\cdot\overline{C}.$$
- Felhasználva, hogy $\overline{B}\cdot 0=0$ és átrendezve a kifejezést kapjuk:
$$\overline{A}\cdot\overline{B}\cdot\overline{C}+A\cdot C+\overline{B}\cdot 0+\overline{A}\cdot B\cdot\overline{C}=A\cdot C+\overline{A}\cdot\overline{C}\cdot(\overline{B}+B).$$
- Mivel, $\overline{B}+B=1$ és $\overline{A}\cdot\overline{C}\cdot 1=\overline{A}\cdot\overline{C}$ így:
$$A\cdot C+\overline{A}\cdot\overline{C}\cdot(\overline{B}+B)=A\cdot C+\overline{A}\cdot\overline{C}.$$
- Tehát $(A+B\cdot\overline{C})\cdot(\overline{A}+C)+B\cdot\overline{C}=A\cdot C+\overline{A}\cdot\overline{C}$.

3. Feladat: Írjuk fel a *3.4. táblázatban* megadott függvényt és egyszerűsítsük!

A	B	C	F^3
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

3.4. táblázat.

Megoldás:

$$\begin{aligned}F^3 &= \overline{A}\cdot\overline{B}\cdot C+\overline{A}\cdot B\cdot\overline{C}+\overline{A}\cdot B\cdot C+A\cdot B\cdot\overline{C}=\\ &=\overline{A}\cdot C\cdot(B+\overline{B})+B\cdot\overline{C}\cdot(A+\overline{A})=\\ &=\overline{A}\cdot C\cdot 1+B\cdot\overline{C}\cdot 1=\\ &=\overline{A}\cdot C+B\cdot\overline{C}\end{aligned}$$

Az egyszerűsített logikai függvény:

$$F^3=\overline{A}\cdot C+B\cdot\overline{C}$$

☞ Összefoglaló feladatok:

1 Feladat: Egyszerűsítsük az alábbi kifejezéseket!

a) $F^3=\overline{A}\cdot B\cdot C+\overline{A}\cdot B\cdot\overline{C}+A\cdot B\cdot C+A\cdot\overline{B}\cdot C$

b) $F^4=C\cdot(A+B)+D\cdot(B+C)+\overline{B}\cdot D+D\cdot C$

c) $F^3=\overline{A}\cdot\overline{B}\cdot C+\overline{A}\cdot B\cdot C+A\cdot B\cdot C$

2. Feladat: Igazoljuk a következő azonosságokat!

a) $\overline{A}\cdot B+A=A+B$

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

3.4. A logikai függvények szabályos alakjai

Az algebrai alakban megadott logikai függvények hátránya, hogy egy függvényt több egymással ekvivalens módon is felírhatunk. Ezt a hátrányt a logikai függvények szabályos (kanonikus) alakjának az alkalmazása küszöböli ki, amelyek segítségével egy függvény csak egyetlen algebrai alakban írható fel. A logikai függvények szabályos alakjának ismeretéhez néhány alapvető fogalmat kell tudni.

Term: – a független változók azon csoportja, amelyeket azonos logikai kapcsolatra jellemző szimbólummal kötünk össze.

Minterm: – a független változók logikai *ÉS* kapcsolata, amelyben minden változó (igaz vagy tagadott formában) egyszer és csakis egyszer szerepel.

Jelölése: m_i^n, ahol • m – *minterm*

• n – *a független változók száma*

• i – *a minterm sorszáma (indexszáma)*

A minterm sorszámát bináris kód alapján a term változóiból képezzük. A változókat jobbról balra növekvő sorrendű bináris helyértéknek tekintjük, majd az igaz változókat **1**-nek, a tagadott változókat **0**–nak tekintve a keletkezett bináris számot decimálissá alakítjuk.

Példa átalakításra:

$$\overline{A}\cdot B\cdot\overline{C}\cdot D \qquad 0\times2^3+1\times2^2+0\times2^1+1\times2^0=4+1=5 \quad\Rightarrow\quad m_5^4$$

Maxterm: – a független változók logikai *VAGY* kapcsolata, amelyben minden változó (igaz vagy tagadott formában) egyszer és csakis egyszer szerepel.

Jelölése: M_j^n, ahol • M – *maxterm*

• n – *a független változók száma*

• j – *a maxterm sorszáma (indexszáma)*

A maxterm sorszámát a mintermekéhez hasonlóan számítjuk ki.

Példa átalakításra:

$$\overline{A}+B+C+\overline{D} \qquad 0\times2^3+1\times2^2+1\times2^1+0\times2^0=4+2=6 \quad\Rightarrow\quad M_6^4$$

A szabályos alakú logikai függvényeket két jellemző csoportra oszthatjuk.

Diszjunktív szabályos alak: – olyan logikai függvény, mely mintermek *VAGY* kapcsolatából áll.

Például: • $F^4=A\cdot\overline{B}\cdot C\cdot\overline{D}+\overline{A}\cdot\overline{B}\cdot C\cdot\overline{D}+A\cdot B\cdot C\cdot\overline{D}+\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}$

• $F^4=m_{10}^4+m_2^4+m_{13}^4+m_0^4$

Konjunktív szabályos alak: – olyan logikai függvény, mely maxtermek *ÉS* kapcsolatából áll.

Például: • $F^4=(A+\overline{B}+C+\overline{D})\cdot(A+\overline{B}+\overline{C}+\overline{D})\cdot(\overline{A}+\overline{B}+C+\overline{D})$

• $F^4=M_{10}^4+M_8^4+M_2^4$ (átalakítva)

Az egyszerűbb írásmód érdekében egy logikai függvényt megadhatunk a termek sorszámával is.

Diszjunktív szabályos alak esetén:

- $F^4 = m_{10}^4 + m_2^4 + m_{13}^4 + m_0^4$
- $F^4 = \sum^4(0, 2, 10, 13)$

– ahol a $\sum$ jel a logikai *VAGY* kapcsolatot, kitevője a független változók számát, a (0, 2, 10, 13) a mintermek sorszámát jelöli.

Konjunktív szabályos alak esetén:

- $F^4 = M_{10}^4 + M_8^4 + M_2^4$ (átalakítva)
- $F^4 = \prod^4(2, 8, 10)$

– ahol a $\prod$ jel a logikai *ÉS* kapcsolatot, kitevője a független változók számát, a (2, 8, 13) a mintermek sorszámát jelöli.

A sorszámokkal megadott logikai függvények nagy előnye a könnyű kezelhetőség és sok változó esetén kicsi a felíráskor elkövethető tévesztés lehetősége. Az általánosított De Morgan-azonosságok lehetővé teszik a mintermek és maxtermek közötti átalakítást. Az átalakítás szabályai általános formában a következők:

$$m_i^n = \overline{M}_{2^n-1-i}^n \tag{3.12}$$

$$M_i^n = \overline{m}_{2^n-1-i}^n \tag{3.13}$$

Az átalakításokra láthatunk példákat a *3.5. és 3.6. táblázatokban.*

m_i^n	i	j	$\overline{M}_j^n$
$A \cdot B \cdot C$	7	0	$\overline{\overline{A}+\overline{B}+\overline{C}}$
$\overline{A} \cdot B \cdot \overline{C}$	2	5	$\overline{A+\overline{B}+C}$
$A \cdot \overline{B} \cdot C$	5	2	$\overline{\overline{A}+B+\overline{C}}$

3.5. táblázat. Mintermek maxtermekké alakítása

$j = 2^n - 1 - i$

M_i^n	i	j	$\overline{m}_j^n$
$A+B+C$	7	0	$\overline{\overline{A} \cdot \overline{B} \cdot \overline{C}}$
$\overline{A}+\overline{B}+\overline{C}$	0	7	$\overline{A \cdot B \cdot C}$
$A+\overline{B}+\overline{C}$	4	3	$\overline{\overline{A} \cdot B \cdot C}$

3.6. táblázat. Maxtermek mintermekké alakítása

$j = 2^n - 1 - i$

A minterm-maxterm átalakításhoz hasonlóan történik a diszjunktív-konjunktív függvényátalakítás is. A diszjunktív konjunktív alak logikai kapcsolatai egymásnak negáltjai, így kézenfekvő, hogy az átírást kétszeres negálással oldhatjuk meg. Az átírás módját kövessük végig egy példán.

Feladat: Írjuk át az alábbi diszjunktív alakban megadott függvényt, konjunktív formára!

$$F^3 = \sum^3(2, 3, 5)$$

Megoldás:

- Felírjuk a mintermeket, amelyek a megadott függvényben nem *szerepelnek (első negálás)*.

$$\overline{F}^3 = \sum^3(0,1,4,6,7).$$

- A kapott mintermeket átírjuk maxtermmé.

$$m_0^3 = \overline{M}^3_{8-1-0} = \overline{M}^3_7$$

$$m_1^3 = \overline{M}^3_{8-1-1} = \overline{M}^3_6$$

$$m_4^3 = \overline{M}^3_{8-1-4} = \overline{M}^3_3$$

$$m_7^3 = \overline{M}^3_{8-1-6} = \overline{M}^3_1$$

$$m_7^3 = \overline{M}^3_{8-1-7} = \overline{M}^3_0$$

- Ezek alapján felírható a következő azonosság:

$$\overline{F}^3 = \sum^3(0,1,4,6,7) = \sum^3\left(\overline{M}^3_0, \overline{M}^3_1, \overline{M}^3_3, \overline{M}^3_6, \overline{M}^3_7\right).$$

- A kapott függvényre alkalmazzuk a De Morgan-azonosságot (*második negálás*).

$$\overline{\overline{F}}^3 = F^3 = \overline{\sum^3\left(\overline{M}^3_0, \overline{M}^3_1, \overline{M}^3_3, \overline{M}^3_6, \overline{M}^3_7\right)} = \prod^3(0,1,3,6,7).$$

- Az eredmény tehát:

$$F^3 = \overline{\sum^3(2,3,5)} = \prod^3(0,1,3,6,7).$$

A logikai függvények szabályos alakjának igen nagy jelentősége van, mivel a logikai hálózatok tervezésénél használt szisztematikus (módszeres) függvényegyszerűsítő eljárásokat csak szabályos alak esetén lehet alkalmazni. Nem szabályos alakú függvényeket vagy Boole-algebrai azonosságokkal lehet egyszerűsíteni, vagy a függvényt szabályossá kell tenni.

3.5. Logikai függvények egyszerűsítése

A műszaki gyakorlatban a logikai hálózatok gazdaságos megvalósításának alapfeltétele, hogy az áramkört lehetőleg a legkevesebb áramköri elem segítségével építsük fel. A hálózat működését leíró logikai függvények általában nem a legegyszerűbb formában állnak rendelkezésre. Ezért szükséges a logikai függvényeket egyszerűsítő eljárások ismerete.

3.5.1. Boole-algebrai egyszerűsítés

A logikai függvények egyszerűsítése Boole-algebrai azonosságok felhasználásával elvileg a legegyszerűbb egyszerűsítési mód. Hiányossága ennek az eljárásnak, hogy bonyolultabb függvények esetén az egyszerűsítés igen körülményes és egyáltalán nem biztos, hogy a legegyszerűbb megoldáshoz jutunk. Ezt a problémát oldják meg a szisztematikus egyszerűsítési eljárások.

3.5.2. Szisztematikus (módszeres) egyszerűsítési eljárások

A szisztematikus (*módszeres*) egyszerűsítési eljárások nagy előnye, hogy a logikai függvények egyszerűsítését mechanikussá teszik. Alkalmazásukkal lehetővé válik, hogy biztosan a logikai függvény legegyszerűbb formáját kapjuk meg.

A szisztematikus egyszerűsítési eljárások elve a logikai algebra két azonosságán alapszik:

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

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

Azaz ha két szabályos term csak egyetlen változó logikai értékében tér el egymástól, akkor a két term összevonható és a képződött új termben csak azon változók szerepelnek, melyek mindkét termben azonos logikai értékűek voltak. Az azonosságok sorozatos ismétlésével jutunk el az egyszerűsített függvényhez.

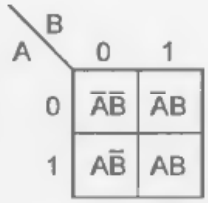
3.5.2.1. Logikai függvények egyszerűsítése grafikus módszerrel

Grafikus egyszerűsitő módszereket ***Veitch*** és ***Karnaugh*** dolgozott ki. Alkalmazásuk maximum négy független változóig tekinthető egyszerűnek, négy változó felett nem célszerű alkalmazni.

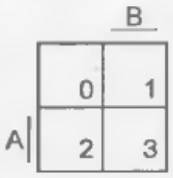
Az egyszerűsítendő logikai függvényt négyzetekből (*cellákból*) álló diagramban ábrázoljuk. A diagramot alkotó cellák számát n változó esetén az $N = 2^n$ összefüggés segítségével határozzuk meg. Minden egyes cella egy szabályos termet képvisel. Az egymás mellet levő cellákban olyan termek vannak, amelyek ***egy és csakis egy változó logikai értékében különböznek egymástól***.

A ***Karnaugh-táblában*** a cellák és termek kapcsolatát, a diagram oszlopainak és sorainak bináris kódolásával adjuk meg (*3.4. ábra*).

A ***Veitch-táblában*** a cellák és termek kapcsolatát a termekből képzett bináris szám decimális értékével és az egyes változók igaz értékéhez tartozó oszlopok és sorok megjelölésével adjuk meg (*3.5. ábra*).



3.4. ábra. Karnaugh-tábla



3.5. ábra. Veitch-tábla

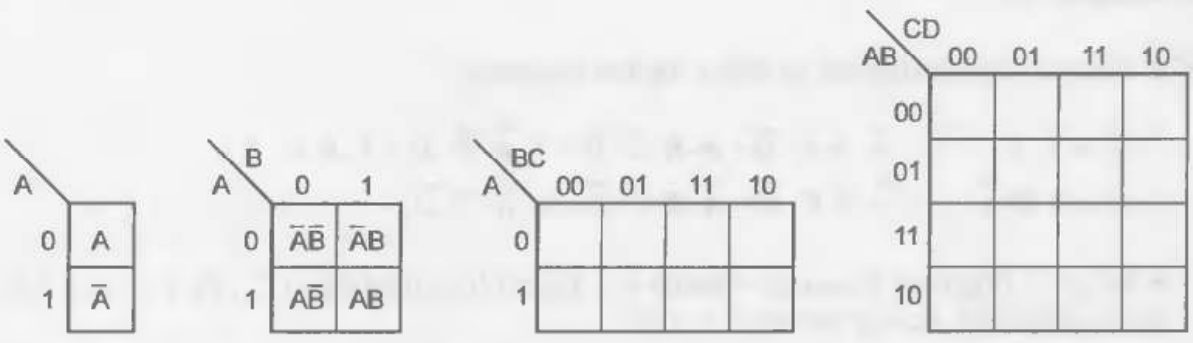
Karnaugh-tábla

Egyváltozós tábla: – egy független változónak (pl. ***A***) két lehetséges állapota lehet, tehát ebben az esetben a tábla két darab cellát tartalmaz (*3.6.a ábra*). A cella kontúrjai mellett a független változó logikai értékét, a cella sarkán a változó betűjelét tüntetjük fel. Az érthetőség kedvéért a cellákba be vannak írva az általuk képviselt termek.

Kétváltozós tábla: – két független változónak (pl. ***A, B***) négy lehetséges állapota lehet, tehát ebben az esetben a tábla négy darab cellát tartalmaz (*3.6.b ábra*). Az érthetőség kedvéért a cellákba be vannak írva az általuk képviselt termek.

Háromváltozós tábla: – három független változónak (pl. ***A, B, C***) nyolc lehetséges állapota lehet, tehát ebben az esetben a tábla nyolc darab cellát *tartalmaz (3.6.c ábra*).

Négyváltozós tábla: – négy független változónak (pl. ***A, B, C, D***) 16 lehetséges állapota lehet, tehát ebben az esetben a tábla 16 darab cellát tartalmaz (*3.6.d ábra*).



a) egyváltozós b) kétváltozós c) háromváltozós d) négyváltozós

3.6. ábra. Karnaugh-táblák

Egy szabályos logikai függvény a Karnaugh-táblában úgy ábrázolható, hogy azon cellákba írunk 1-et, melyek a függvényben levő termeket képviselik. A Karnaugh-tábla univerzálisnak tekinthető, mivel mind a mintermek mind a maxtermek ábrázolására ugyanaz a tábla használható.

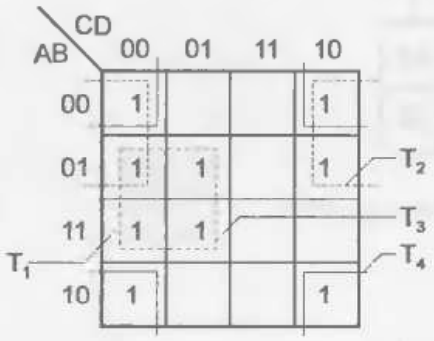
☞ ***1. Feladat:*** Ábrázoljuk Karnaugh-táblában a következő logikai függvényt!

$$F^3 = A \cdot B \cdot C + \overline{A} \cdot B \cdot C + A \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot \overline{C}$$

• A logikai függvény Karnaugh-tábláját a *3.7. ábra* szemlélteti.

A \ BC	00	01	11	10
0	1		1	
1	1		1	1

3.7. ábra.



3.8. ábra.

A Karnaugh-tábla segítségével történő függvény-egyszerűsítés (*tömbösítés*) azon alapul, hogy az egy változó logikai értékében különböző szabályos termek párosával összevonhatók. Azokat a cellákat kell megkeresni és összekapcsolni, amelyek az összevonható termeket jelképezik. Ezek alapján párosával összekapcsolhatók egymással:

- az egymás melletti **1**-et tartalmazó cellák;
- a sorok és oszlopok szélein levő **1**-et tartalmazó cellák.

Az összekapcsolás (tömbösítés) során maximális számú **1**-et tartalmazó tömbök kialakítására kell törekedni. A kialakított tömbben levő **1**-esek száma *kettő egész számú többszöröse* kell legyen (pl. 1, 2, 4, 8, 16, ...). Egy már tömbösített cellát új tömb képzésénél akkor lehet felhasználni, ha az új tömb még nem tömbösített termet is tartalmaz.

Kövessük végig egy logikai függvény egyszerűsítését az előbb ismertetett szabályok alkalmazásával.

☞ ***2. Feladat:*** Egyszerűsítsük az alábbi logikai függvényt!

$$F^4 = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot D +$$
$$+ A \cdot B \cdot \overline{C} \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot \overline{D} + A \cdot \overline{B} \cdot C \cdot \overline{D}$$

• A logikai függvény Karnaugh-tábláját és a kialakítható tömböket (T_1, T_2, T_3, T_4) a *3.8. ábra* szemlélteti. Az egyszerűsítés menete:

$$F^4 = T_1 + T_2 + T_3 + T_4, \qquad \text{ahol:} \qquad T_1 = \overline{C} \cdot \overline{D}$$
$$T_2 = \overline{A} \cdot \overline{D}$$
$$T_3 = B \cdot \overline{C}$$
$$T_4 = \overline{B} \cdot \overline{D}$$

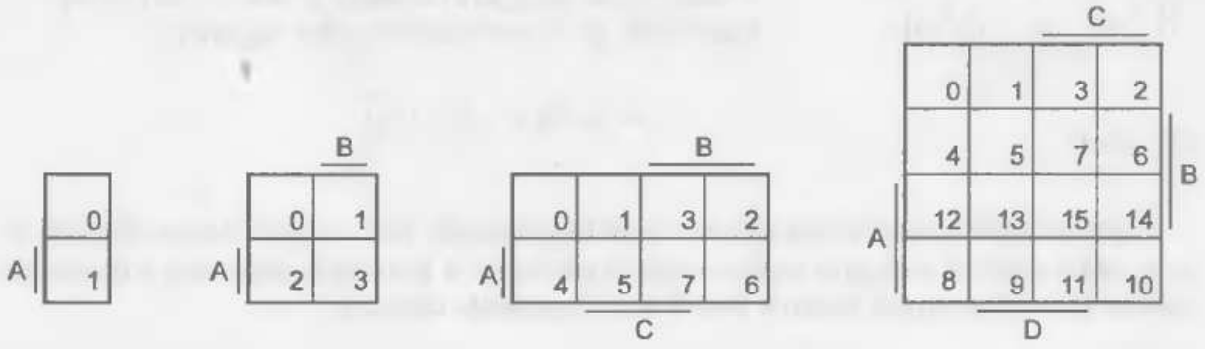
• Az egyszerűsített logikai függvény: $F^4 = \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{D} + B \cdot \overline{C} + \overline{B} \cdot \overline{D}$

Veitch-tábla

A logikai függvények két szabályos alakjának megfelelően Veitch kétféle táblát vezetett be. A ***minterm táblát*** a diszjunktív szabályos függvények számára és a ***maxterm táblát*** a konjunktív szabályos függvények számára. A független vátozók logikai értékeit a tábla kontúrja mentén húzott vonallal tüntetjük fel, és a cellákba beírjuk az ábrázolt term sorszámát.

Minterm táblák:

Egy, két-, három- és négyváltozós minterm-tábla látható a *3.9.a), b), c), és d) ábrán.*

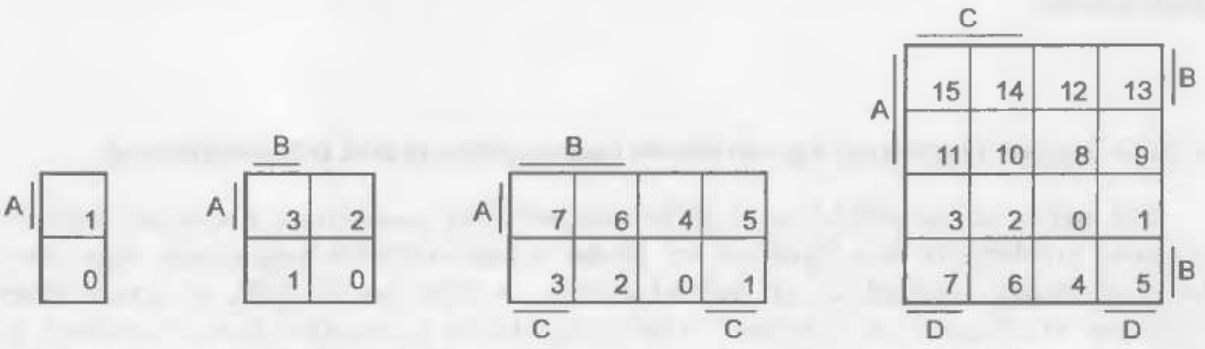


a) egyváltozós b) kétváltozós c) háromváltozós d) négyváltozós

3.9. ábra. Minterm-táblák

Maxterm táblák:

A maxterm tábla felrajzolásához legegyszerübben úgy juthatunk, ha a mintermből maxtermbe való átírás szabályait követjük. Azaz képezzük a változók negáltját és a cellák sorszámait kiegészítjük (átsorszámozzuk a cellákat az $i_M = 2^n - 1 - i_m$ összefüggés alapján). Egy, két-, három- és négyváltozós maxterm-tábla látható a *3.10.a), b), c), és d) ábrán.*

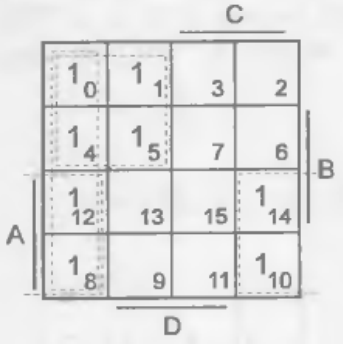


a) egyváltozós b) kétváltozós c) háromváltozós d) négyváltozós

3.10. ábra. Maxterm-táblák

A logikai függvények egyszerüsítéshez szükséges csoportok (tömbök) képzése a Veitch-táblában ugyanúgy történik, mint a Karnaugh-tábla esetében.

☞ *Feladat:*
Ábrázoljuk és egyszerűsítsük az alábbi logikai függvényt Veitch tábla felhasználásával!



$$F^4 = \sum^4(0,1,4,5,8,10,12,14)$$

• A függvény Veitch-táblában való ábrázolását a *3.11. ábrán* láthatjuk (*minterm tábla*).

• Ha az ábrán szaggatott vonallal jelzett csoportosítást használjuk, az egyszerűsített logikai függvény:

$$F^4 = \overline{A} \cdot \overline{C} + \overline{C} \cdot \overline{D} + A \cdot \overline{D}$$

3.11. ábra:

A két grafikus egyszerűsítési eljárás – mint tapasztalható volt – csak jelölésmódban tér el. Az algebrai alakban megadott logikai függvények esetén a Karnaugh-tábla, míg a sorszámos alakban adott függvények esetén a Veitch tábla használata célszerű.

3.5.2.2. Logikai függvények egyszerűsítése numerikus módszerrel

A grafikus egyszerűsítési eljárások maximálisan négy független változó esetén nyújtanak egyszerű megoldást a logikai függvények egyszerűsítésére. Négy változó felett már csak numerikusan célszerű az egyszerűsítést elvégezni, amelyet Quine-McCluskey eljárásnak is neveznek. A numerikus egyszerűsítés nagy előnye a többi módszerhez viszonyítva, hogy algoritmizálható és így számítógépre implementálható.

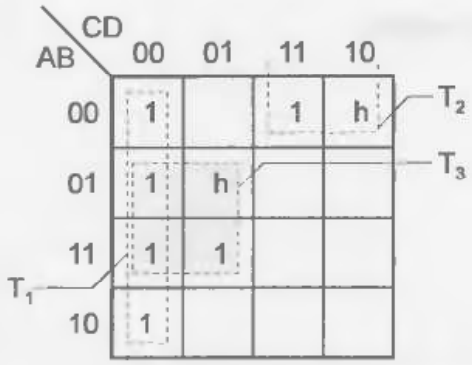
Az egyszerűsítés elve azonos a grafikus eljárásoknál használttal, de itt az összevonást a termek sorszámainak kivonásával szimbolizáljuk. Az eljárás részletes ismertetésével nem foglalkozunk.

3.5.2.3. Logikai függvények egyszerűsítése határozatlan termek felhasználásával

Egy logikai hálózat működésénél előfordulhatnak olyan esetek, hogy a bemeneti változók bizonyos kombinációi nem léphetnek fel. Ezeket a kombinációkat határozatlan vagy tiltott kombinációknak nevezzük és „h" betűvel jelöljük. A tiltott kombinációk az egyszerűsítés folyamán tetszőleges logikai értékkel vehetők figyelembe (*számunkra kedvező értékkel*) és így a kapott logikai függvény egyszerűbb alakú lehet. A könnyebb érthetőség kedvéért vizsgáljuk meg ezt egy konkrét feladat esetén.

Egyszerűsítsük a következő logikai függvényt:

$$F^4 = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{C} \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot D +$$
$$\textit{határozatlan}: \quad \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D}, \quad \overline{A} \cdot B \cdot \overline{C} \cdot D$$



3.12. ábra. Egyszerűsítés Karnaug-táblában

- A függvény ábrázolását és egyszerűsítését Karnaugh-táblában a *3.12. ábra* szemlélteti.
- Az egyszerűsített függvény:

$$F^4 = \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C + B \cdot \overline{C}$$

ha a két határozatlan termet **1** értékkel vesszük figyelembe.

- Ha a határozatlan termeket nem vettük volna figyelembe, a függvény:

$$F^4 = \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot D + A \cdot B \cdot \overline{C}$$

alakban lett volna megadható.

Közvetlenül belátható tehát, hogy a határozatlan termek felhasználása egyszerűbb és természetesen gazdaságosabb logikai hálózatot eredményez.

Összefoglaló feladatok:

1. Határozza meg a következő logikai függvények szabályos alakját:

   a) $F^3 = B + \overline{A} \cdot C$

   b) $F^3 = A \cdot B + \overline{A} \cdot \overline{C}$

   c) $F^3 = A \cdot \overline{B} + \overline{A} \cdot C$

2. Határozza meg a következő termek sorszámait:

   a) $\overline{A} \cdot B \cdot C$

   b) $\overline{A} \cdot \overline{B} \cdot C$

   c) $\overline{A} \cdot B \cdot \overline{C}$

   d) $\overline{A} \cdot \overline{B} \cdot C \cdot D$

   e) $A \cdot \overline{B} \cdot C \cdot \overline{D}$

   f) $A \cdot B \cdot C \cdot D$

3. Írja át konjunktív alakba a következő logikai függvényeket:

   a) $F^3 = \sum^3 (0, 2, 5, 6)$

   b) $F^3 = \sum^3 (1, 3, 4, 6)$

   c) $F^4 = \sum^4 (0, 4, 8, 11, 13)$

4. Írja át diszjunktív alakba a következő logikai függvényeket:

a) $F^3 = \prod^3 (0,1,3,5)$

b) $F^3 = \prod^3 (1,4,6)$

c) $F^4 = \prod^4 (3,5,7,9)$

5. Ábrázolja Veitch-táblában a következő logikai függvényeket:

a) $F^3 = \sum^3 (1,2,5,6)$

b) $F^3 = \prod^3 (0,1,4,6)$

c) $F^4 = \sum^4 (0,1,4,5,11,15)$

d) $F^4 = \sum^4 (0,1,4,5,11,15)$

e) $F^4 = \prod^4 (0,1,4,6,8,10,12,14)$

f) $F^4 = \prod^4 (2,3,4,7,8,9,11,15)$

6. Határozza meg a következő logikai függvények legegyszerűbb alakját:

a) $F^3 = \prod^3 (2,3,4,6)$

b) $F^3 = \prod^3 (0,2,4,6)$

c) $F^3 = \sum^3 (1,3,5,6)$

d) $F^4 = A \cdot B \cdot C + \overline{A} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot C \cdot \overline{D}$

e) $F^4 = A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + B \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{D} + \overline{B} \cdot C + \overline{B} \cdot C \cdot \overline{D}$

f) $F^4 = \sum^4 (0,2,3,4,5,11,15)$

g) $F^4 = \sum^4 (0,1,3,13,14,15)$

h) $F^4 = \prod^4 (2,4,6,7,8,9,13,14)$

i) $F^4 = \prod^4 (1,3,4,7,8,9,11,15)$

j) $F^4 = \prod^4 (1,3,4,7,8,9,$
hat : $2,11,15)$

4. Logikai hálózatok

A logikai algebrával foglalkozó fejezetben már láttuk, hogy a logikai tervezés során a megoldandó feladat által felvetett összefüggéseket logikai függvénnyé alakítjuk át. Ezután a logikai függvényt egy megfelelő eljárással egyszerűsítjük. A következő igen fontos lépés az egyszerűsített logikai függvények műszaki megvalósítása.

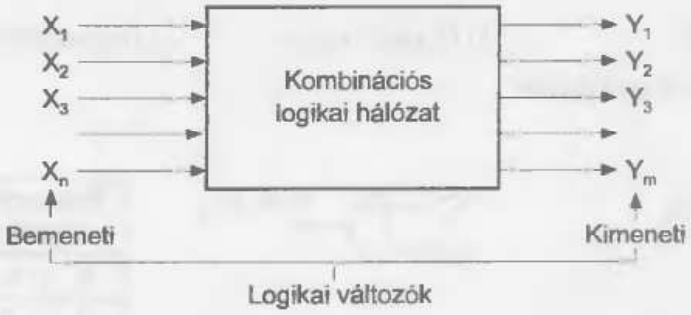
Ez és a következő fejezet a logikai függvényeket megvalósító logikai hálózatokkal foglalkozik. A logikai függvények az időtől való függés szerint lehetnek ***időfüggetlen***, és ***időfüggő*** logikai függvények. Ennek megfelelően az őket megvalósító logikai hálózatok is két csoportra oszthatók:

1. ***kombinációs hálózatokra:*** – időfüggetlen logikai függvényeket valósítanak meg;
2. ***sorrendi (szekvenciális) hálózatokra:*** – időfüggő logikai függvényeket valósítanak meg.

A logikai rendszerek megvalósítása az építőelem-elv alapján történik. Ez lehetővé teszi különféle célokat szolgáló logikai áramkörök gyors és gazdaságos tervezését és kivitelezését. A kombinációs logikai hálózatoknak két alapvető jellegzetessége van:

– memória nélküli logikai áramkörök (kimeneti logikai jel csak a bemeneti logikai jelek jelenléte esetén van).

– a kimeneti logikai változókat (Y_k) az adott időpontban megjelenő bemeneti logikai változók határozzák meg $Y_k = f_k(X_1, X_2, ..., X_n)$ (lásd a *4.1. ábrát*).

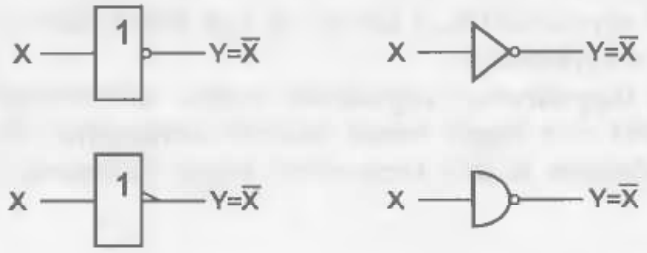


4.1. ábra. Egy kombinációs logikai hálózat tömbvázlata

4.1. Kombinációs logikai hálózatok

Műszakilag a logikai függvények megvalósítása a felhasználás jellegétől függ. Egy adott logikai függvény megvalósítható diszkrét elemek (jelfogó, ellenállás, dióda, tranzisztor) felhasználásával is. Azonban a jelenlegi technikai színvonal integrált áramkörök (*rövidítve **IC**; **I**ntegrated **C**ircuit=Integrált áramkör, angol kifejezés*) felhasználását követeli meg. A logikai alapfüggvényeket megvalósító áramköri elemeket ***logikai kapuknak*** (*logic gates*) nevezzük. Egy digitális integrált áramkörben – az áramkör bonyolultságától függően – egy vagy több logikai kapu található.

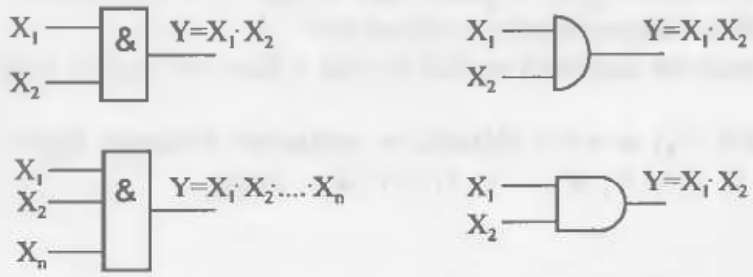
A *NEGÁCIÓ*, *ÉS*, valamint a *VAGY* alapműveleteknek megfelelően a *4.2., 4.3. és 4.4. ábra* szemlélteti az *inverter*, *ÉS* (*AND*), *VAGY* (*OR*) kapu logikai rajzjelét és igazságtáblázatát. A kapuk logikai rajzjeleként két elterjedt szabvány által előírt rajzjelek használatosak: az egyik az amerikai IEEE szabványa, a másik a nemzetközi IEC szabványa (négyszögletes típusú rajzjel, amely megegyezik a magyar szabvánnyal MSZ 9200-53).



Bemeneti változó	Megvalósított logikai függvény
X	$Y=\overline{X}$
0	1
1	0

a) Szabványos rajzjel *b) Használt rajzjel* *c) Igazságtáblázat*

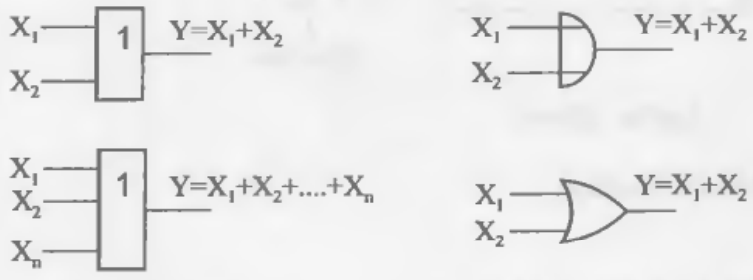
4.2. ábra. NEM (*Inverter*) kapu rajzjele



Bemeneti változók		Megvalósított logikai függvény
X_1	X_2	$Y=X_1 \cdot X_2$
0	0	0
0	1	0
1	0	0
1	1	1

a) Szabványos rajzjel *b) Használt rajzjel* *c) Igazságtáblázat*

4.3. ábra. ÉS (*AND*) kapu rajzjele

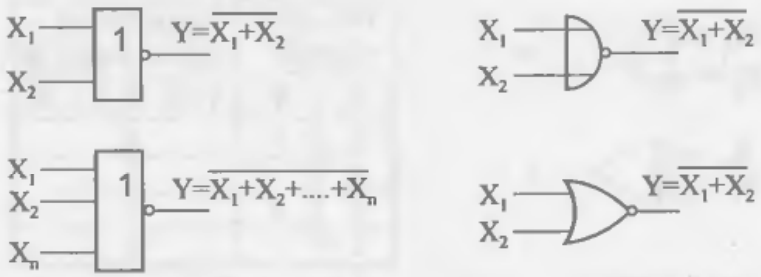


Bemeneti változók		Megvalósított logikai függvény
X_1	X_2	$Y=X_1+X_2$
0	0	0
0	1	1
1	0	1
1	1	1

a) Szabványos rajzjel *b) Használt rajzjel* *c) Igazságtáblázat*

4.4. ábra. VAGY (*OR*) kapu rajzjele

Áramköri szempontból sokkal egyszerűbb az *ÉS* és a *VAGY*-kapu helyett a *NEM-ÉS* (*NAND*), illetve a *NEM-VAGY* (*NOR*) kapu megvalósítása. A *NEM-ÉS* művelet $Y = \overline{A \cdot B}$ és a *NEM-VAGY* $Y = \overline{A+B}$ művelet kifejezéseiből látható, hogy elvileg ez a két kapu úgy tekinthető mint egy *ÉS*, illetve *VAGY* kapu, amelyet egy inverter követ (*4.5. és 4.6. ábra*).

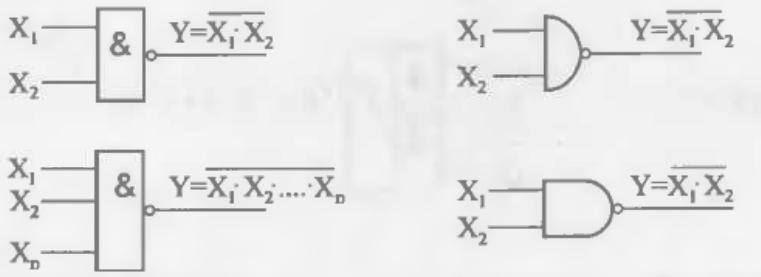


a) *Szabványos rajzjel* b) *Használt rajzjel*

Bemeneti változók		Megvalósított logikai függvény
X_1	X_2	$Y=\overline{X_1+X_2}$
0	0	1
0	1	0
1	0	0
1	1	0

c) *Igazságtáblázat*

4.5. ábra. VAGY-NEM (*NOR*) kapu rajzjele



a) *Szabványos rajzjel* b) *Használt rajzjel*

Bemeneti változók		Megvalósított logikai függvény
X_1	X_2	$Y=\overline{X_1\cdot X_2}$
0	0	1
0	1	1
1	0	1
1	1	0

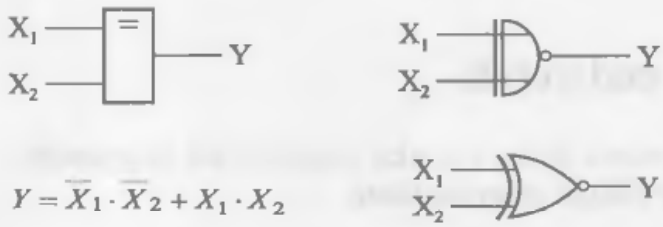
c) *Igazságtáblázat*

4.6. ábra. ÉS-NEM (*NAND*) kapu rajzjele

A logikai hálózatok egyszerűbb felépítése érdekében nemcsak 2 bemenetű kapukat fejlesztettek ki, hanem 3, 4 és 8 bemenetűeket is. Ugyancsak a logikai hálózatok leegyszerűsítését szolgálja az *EKVIVALENCIA* függvény, a *KIZÁRÓ-VAGY* (*EXCLUSIVE-OR*) kapu és az ÉS-VAGY-NEM (AND-OR-NOT) kapu.

Az *EKVIVALENCIA* függvény (*4.7. ábra*) által megvalósított logikai művelet jelölése:

$$Y = A\Theta B = \overline{A}\cdot B + A\cdot\overline{B}$$



a) *Szabványos rajzjel* b) *Használt rajzjel*

Bemeneti változók		Megvalósított logikai függvény
X_1	X_2	$Y=\overline{X}_1\cdot\overline{X}_2+X_1\cdot X_2$
0	0	1
0	1	0
1	0	0
1	1	1

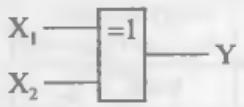
c) *Igazságtáblázat*

4.7. ábra. EKVIVALENCIA-függvény rajzjele

A *KIZÁRÓ-VAGY* kapu (*4.7. ábra*) által végrehajtott logikai művelet (az ekvivalencia-függvény negáltja) jelölése:

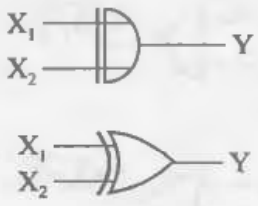
$$Y = A\oplus B = \overline{A}\cdot B + A\cdot\overline{B}$$

Ezt a függvényt antivalenciának is nevezik, mert értéke csak akkor **1**, ha a bemeneti változók ellentétes értékűek (**0** és **1**).



$Y = X_1 \cdot X_2 + X_1 \cdot X_2$

a) Szabványos rajzjel

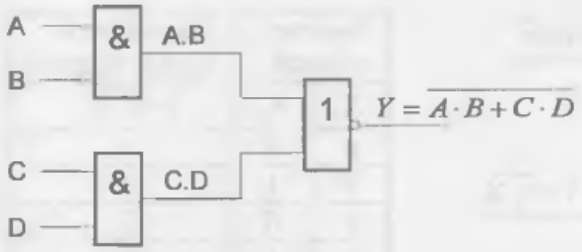


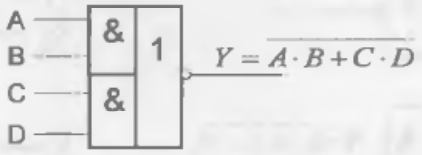
b) Használt rajzjel

Bemeneti változók		Megvalósított logikai függvény
X_1	X_2	$Y=\overline{X}_1 X_2 + X_1 \overline{X}_2$
0	0	0
0	1	1
1	0	1
1	1	0

c) Igazságtáblázat

4.8. ábra. ANTIVALENCIA (KIZÁRÓ-VAGY) - függvény rajzjele





4.9. ábra. ÉS-VAGY-NEM – kapu logikai rajzjele

A *4.9. ábra* egy 2×2 bemenetű *ÉS-VAGY-NEM* (*AND-OR-NOT*) kaput szemléltet, amelyet a következő logikai függvény ír le:

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

Ezen kívül elterjedt a 4×2 bemenetű *ÉS-VAGY-NEM* kapu is.

4.1.1. Funkcionálisan teljes rendszerek

Funkcionálisan teljes rendszernek nevezzük azokat a logikai függvényeket megvalósító kapukat, amelyekből bármilyen tetszőleges hálózat megvalósítható.

1. NEM-ÉS-VAGY rendszer. Tetszőleges logikai függvény kifejezhető az ÉS, a VAGY és a NEGÁCIÓ műveletet megvalósító logikai kapuk megfelelő kombinációjával.

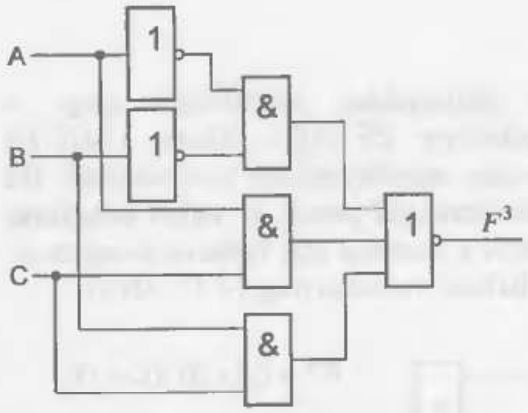
☞ ***Példa:*** –valósítsuk meg logikai hálózattal a következő logikai függvényeket:

$$F^3 = \overline{A \cdot B} + A \cdot C + \overline{A} \cdot B$$

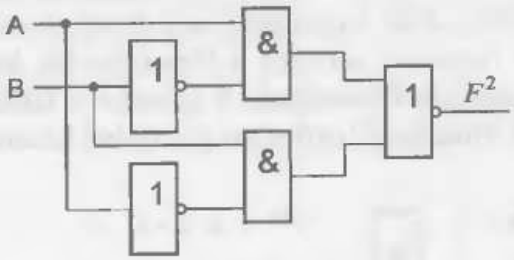
$$F^3 = A \cdot \overline{B} + \overline{A} \cdot B$$

A függvényeket megvalósító hálózatok a *4.10. és 4.11 ábrán* láthatók

A gyakorlati megvalósítás szempontjából a NEM-ÉS-VAGY rendszer nem terjedt el, mivel áramköri szempontból sokkal egyszerűbb az *ÉS* és a *VAGY*-kapu helyett a *NEM-ÉS* (*NAND*), illetve a *NEM-VAGY* (*NOR*) kapu megvalósítása.



4.10. ábra. $F^3 = \overline{A \cdot B} + A \cdot C + \overline{A} \cdot B$



4.11. ábra. $F^3 = A \cdot \overline{B} + \overline{A} \cdot B$

2. NAND-rendszer. A De Morgan azonosságokból következik, hogy az *ÉS*, *NEM* és *VAGY* függvények átírhatók *NAND* függvénnyé. A szükséges átalakítások a következők:

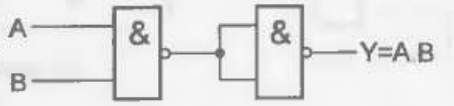
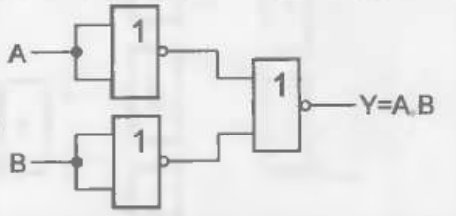
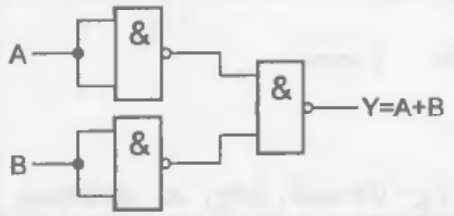
- *ÉS* kapcsolat: $F^2 = \overline{\overline{A \cdot B}} = A \cdot B$
- *VAGY* kapcsolat: $F^2 = \overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A}} + \overline{\overline{B}}$
- *NEM* kapcsolat: $F^2 = \overline{A \cdot A} = \overline{A} + \overline{A} = \overline{A}$

A felírt műveleteket megvalósító kapukat a *4.12. ábrán* láthatjuk.

3. NOR-rendszer. NOR-rendszer esetén a szükséges átalakítások a következők:

- *ÉS* kapcsolat: $F^2 = \overline{\overline{A} + \overline{B}} = A \cdot B$
- *VAGY* kapcsolat: $F^2 = \overline{\overline{A + B}} = A + B$
- *NEM* kapcsolat: $F^2 = \overline{A + A} = \overline{A} \cdot \overline{A} = \overline{A}$

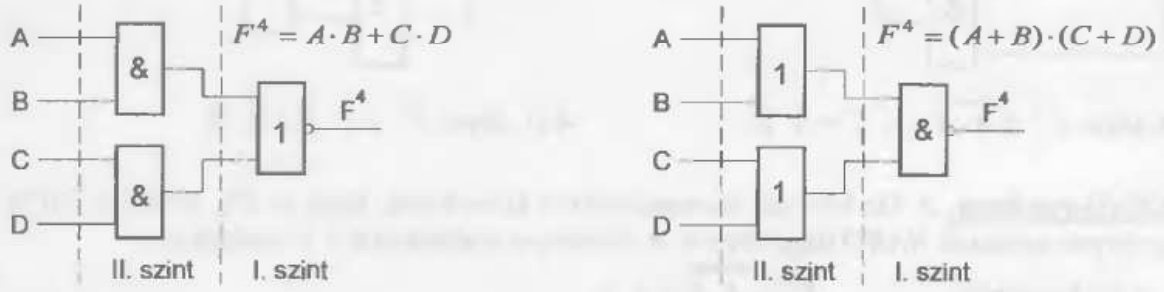
A felírt műveleteket megvalósító kapukat a *4.12. ábrán* láthatjuk.

Logikai függvény	*A logikai kapu típusa*	
	NEM-ÉS (NAND)	NEM-VAGY (NOR)
NEM		
ÉS		
VAGY		

4.12. ábra. Alapfüggvények megvalósítása NOR és NAND kapukkal

4.1.2. Két- és többszintű hálózatok

A logikai függvények műszakilag logikai hálózatokkal valósíthatók meg. A szisztematikus függvényegyszerűsítő eljárások eredménye *ÉS-VAGY*, illetve *VAGY-ÉS* hálózatként valósítható meg. Ezek a hálózatok *két logikai műveleti szinttel* rendelkeznek. Ha a bemeneti változók a kimenetre két kapuáramkörön keresztül jutnak el, akkor ***kétszintes hálózatról*** beszélünk. A szinteket a hálózat kimenetétől a bemenet felé haladva számozzuk. A következő logikai függvényeket kétszintes logikai hálózat valósítja meg (*4.13. ábra*).

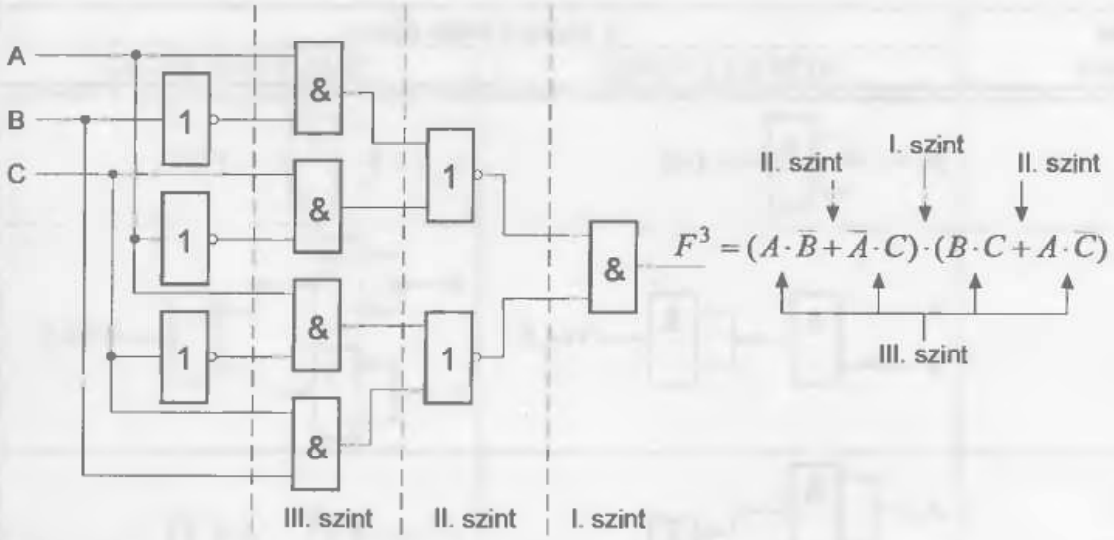


4.13. ábra. Kétszintű logikai hálózatok

Ha egy diszjunktív vagy konjuktív alakban megadott logikai függvényt újabb logikai művelettel, egy vagy több másik függvénnyel bővítjük, az új logikai függvény már nem valósítható meg kétszintű hálózattal.

☞ 1. Példa: Vizsgáljuk meg a következő logikai függvényt megvalósító logikai hálózatot:

$$F^3 = (A \cdot \overline{B} + \overline{A} \cdot C) \cdot (B \cdot C + A \cdot \overline{C})$$



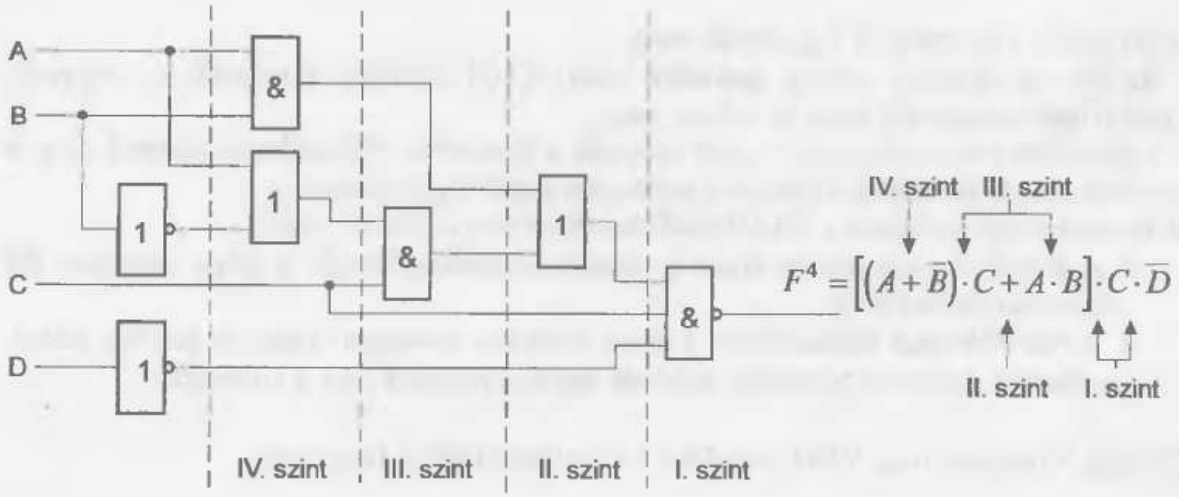
4.14. ábra. Háromszintű logikai hálózat

A logikai szintek megállapítása úgy történik, hogy az utolsónak elvégzendő logikai művelettől kezdve a zárójeleken befelé haladva megszámoljuk az egymás után elvégzendő műveletek számát.

A logikai hálózat háromszintű. A harmadik szinten az $A\cdot\overline{B}, \overline{A}\cdot C, B\cdot C, A\cdot\overline{C}$ ***ÉS*** kapcsolatokat, a második szinten az $(A\cdot\overline{B}+\overline{A}\cdot C)$ *és* $(B\cdot C+A\cdot\overline{C})$ ***VAGY*** kapcsolatot, az első szinten a teljes logikai függvényt alkotó kifejezést valósítjuk meg.

☞ 2. Példa: Vizsgáljuk meg a következő logikai függvényt megvalósító logikai hálózatot:

$$F^4=\left[(A+\overline{B})\cdot C+A\cdot B\right]\cdot C\cdot\overline{D}$$



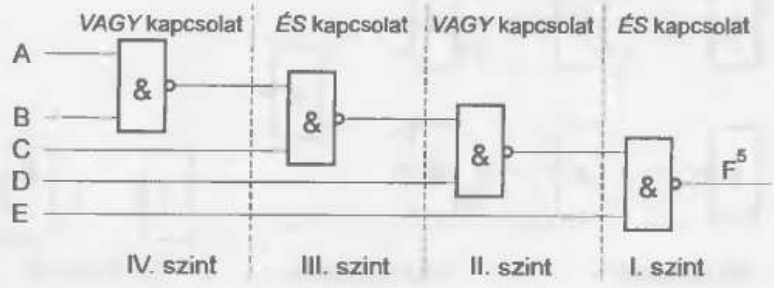
4.15. ábra. Négyszintű logikai hálózat

A logikai függvényt megvalósító logikai hálózat négyszintű. A hálózat és a logikai szinteknek megfelelő műveletek a *4.15. ábrán* láthatók.

A példákból látható, hogy a logikai szintek száma a függvény algebrai alakjából is megállapítható, ha az utolsónak elvégzendő logikai művelettől kezdve a zárójeleken befelé haladva megszámoljuk az egymás után elvégzendő (*azonos logikai szinteket jelentő*) logikai műveleteket.

NAND kapukkal felépített logikai hálózatok analízise

A *NAND* kapu-rendszer *funkcionálisan teljes rendszer*, vagyis tetszőleges logikai függvény megvalósítható NAND kapuk felhasználásával. A következőkben megvizsgáljuk, hogy többszintes hálózatok hogyan valósulnak meg a *NAND*-rendszerben. Vizsgáljuk meg a *4.16. ábrán* látható *NAND* kapukból felépített többszintű logikai hálózatot.



4.16. ábra. *NAND* kapukkal felépített logikai hálózat

A hálózat logikai függvénye a következő:

$$F^5 = \overline{\overline{\overline{\overline{\overline{A \cdot B \cdot C \cdot D \cdot E}}}}}$$

A függvényt a De Morgan-azonosságok felhasználásával átalakítva, kapjuk:

$$F^5 = \overline{\overline{\overline{\overline{A \cdot B \cdot C \cdot D}}} + \overline{E}} = \overline{\overline{A \cdot B \cdot C} \cdot D} + \overline{E}$$

$$F^5 = \left(\overline{\overline{A \cdot B} + \overline{C}}\right) \cdot D + \overline{E} = \left(A \cdot B + \overline{C}\right) \cdot D + \overline{E}$$

A *4.16. ábrán* a következők figyelhetők meg:

– az első és harmadik szinten (***páratlan szint***) *VAGY* művelet, a második és negyedik szinten (***páros szint***) *ÉS* művelet valósul meg.

– a hálózatba páros szinten bevezetett változók a kimeneten változatlanul jelennek meg; a páratlan szinten bevezetett változók a kimeneten negálva jelentkeznek.

Általánosan megfogalmazva a *NAND* rendszerekre érvényes szabályokat:

- A *NAND* logikai hálózatokban a páratlan szinteken ***VAGY***, a páros szinteken ***ÉS*** kapcsolat valósul meg.
- A *NAND* logikai hálózatokban a páros szinteken bevezetett változók negálás nélkül, a páratlan szinteken bevezetett változók negálva jelennek meg a kimeneten.

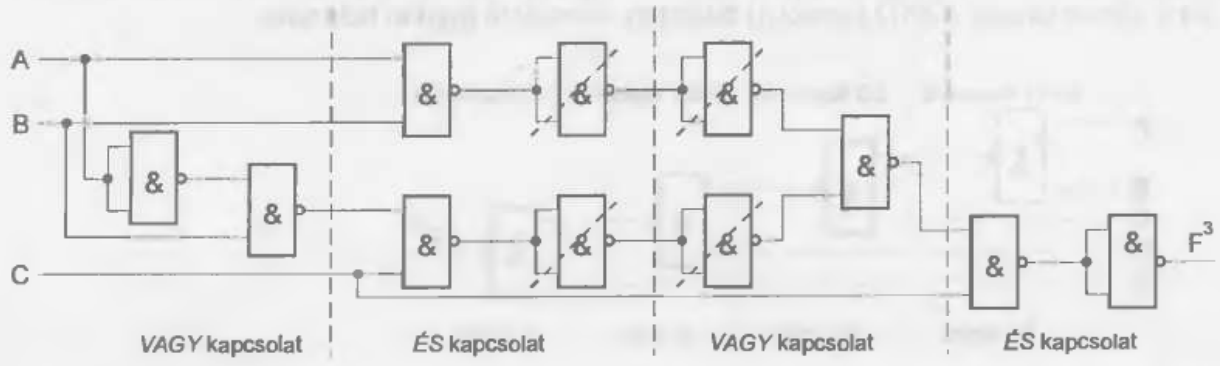
☞ Példa: Valósítsuk meg *NAND* kapukkal a következő logikai függvényt!

$$F^3 = \left[(A + \overline{B}) \cdot C + A \cdot B\right] \cdot C$$

Először a műveleti szinteket kell megállapítani:

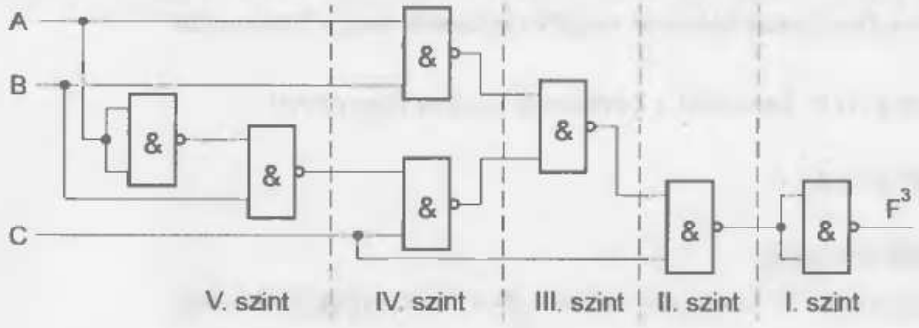
- I. szint: – *ÉS* kapcsolat a C változó és az $(A + \overline{B}) \cdot C + A \cdot B$ kifejezés között.
- II. szint: – *VAGY* kapcsolat az $(A + \overline{B}) \cdot C$ és $A \cdot B$ term között.
- III. szint: – *ÉS* kapcsolat az A és B változó, valamint az $(A + \overline{B})$ és C kifejezés között.
- IV szint: – *VAGY* kapcsolat az A és $\overline{B}$ változók között.

A felírásból látható, hogy a logikai kapcsolatok nem felelnek meg a NAND kapukkal megvalósítandó szinteknek (*4.17 ábra*). Ha átalakítjuk a logikai függvényt az $X + X = X$ azonosság felhasználásával: $F^3 = \left[(A + \overline{B}) \cdot C + A \cdot B\right] \cdot C + \left[(A + \overline{B}) \cdot C + A \cdot B\right] \cdot C$.



4.17. ábra. Többszintű logikai hálózat NAND kapukkal

A *4.17. ábrán* látható logikai hálózatban az áthúzott inverterekre nincs szükség, mivel kétszeres tagadást valósítanak meg ($\overline{\overline{Y}} = Y$). Ezeket a kapukat elhagyva, a megvalósított hálózat a *4.18. ábrán* látható.



4.18. ábra. Egyszerűsített többszintű logikai hálózat NAND kapukkal

NOR kapukkal felépített logikai hálózatok analízise

A *NOR* kapu-rendszer a *NAND*-rendszerhez hasonlóan *funkcionálisan teljes rendszer*, vagyis tetszőleges logikai függvény megvalósítható *NOR* kapuk felhasználásával. A következőkben megvizsgáljuk, hogy többszintes hálózatok hogyan valósulnak meg a *NOR*-rendszerben.

4.19. ábra. Többszintű logikai hálózat NOR kapukkal

A hálózat logikai függvénye a következő:

$$F^5 = \overline{\overline{\overline{\overline{A+B+C+D+E}}}}$$

A függvényt a De Morgan-azonosságok felhasználásával átalakítva, kapjuk:

$$F^5 = \overline{\overline{\overline{\overline{A+B+C+D}}}} \cdot \overline{E} = \left(\overline{\overline{\overline{A+B+C+D}}}\right) \cdot \overline{E}$$

$$F^5 = \left[\left(\overline{\overline{A+B}}\right) \cdot \overline{C} + D\right] \cdot \overline{\overline{E}} = \left[(A+B) \cdot \overline{C} + D\right] \cdot \overline{E}$$

A *4.19. ábrán* a következők figyelhetők meg:

– az első és harmadik szinten (***páratlan szint***) *ÉS* művelet, a második és negyedik szinten (***páros szint***) *VAGY* művelet valósul meg.

– a hálózatba páros szinten bevezetett változók a kimeneten változatlanul jelennek meg; a páratlan szinten bevezetett változók a kimeneten negálva jelentkeznek.

Általánosan megfogalmazva a *NOR* rendszerekre érvényes szabályokat:

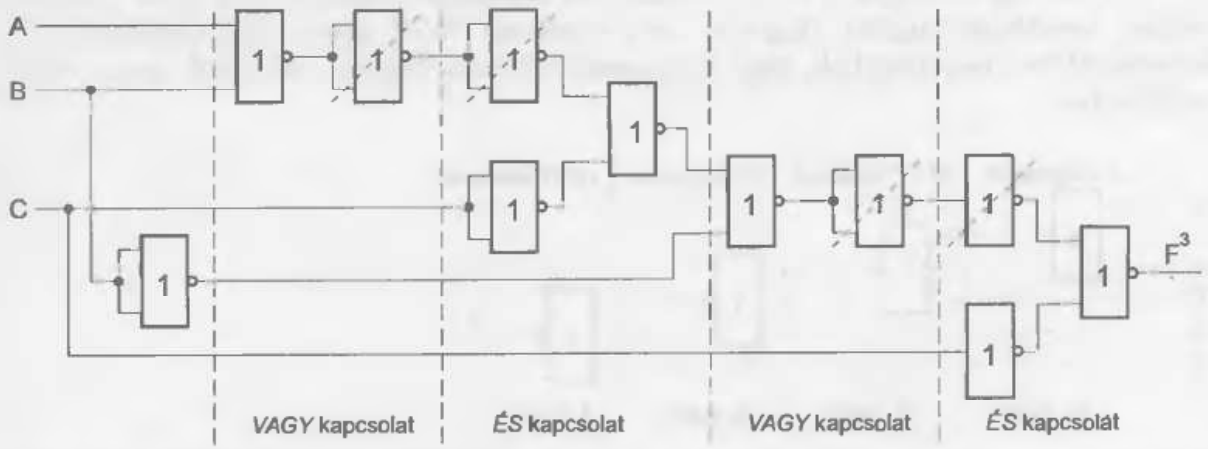
- a *NOR* logikai hálózatokban a páratlan szinteken ***ÉS***, a páros szinteken ***VAGY*** kapcsolat valósul meg,
- a *NOR* logikai hálózatokban a páros szinteken bevezetett változók negálás nélkül,a páratlan szinteken bevezetett változók negálva jelennek meg a kimeneten.

☞ Példa: Valósítsuk meg *NOR* kapukkal a következő logikai függvényt!

$$F^3 = \left|(A+B)\cdot C+\overline{B}\right|\cdot C$$

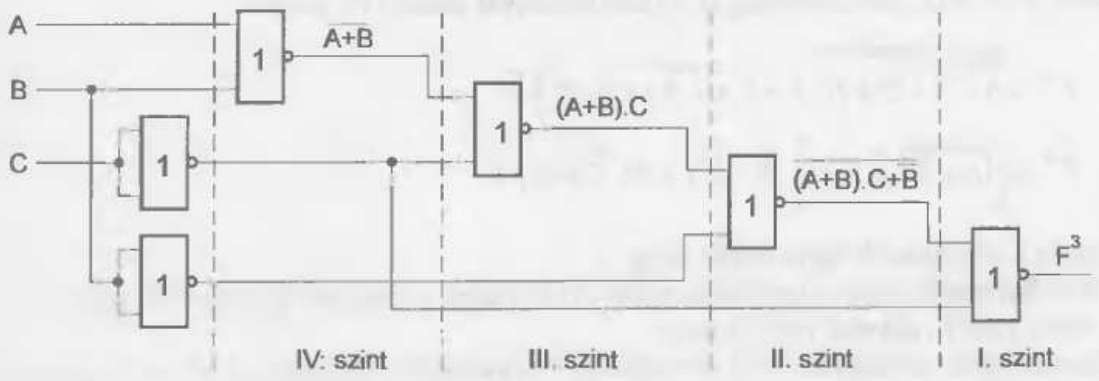
Megállapítjuk a műveleti szinteket:

- I. szint: – *ÉS* kapcsolat a C változó és az $(A+B)\cdot C+\overline{B}$ kifejezés között.
- II. szint: – *VAGY* kapcsolat az $(A+B)\cdot C$ és $\overline{B}$ kifejezés között.
- III. szint: – *ÉS* kapcsolat az $(A+B)$ és C kifejezés között.
- IV szint: – *VAGY* kapcsolat az A és B változók között.



4.20. ábra. Többszintű logikai hálózat NOR kapukkal

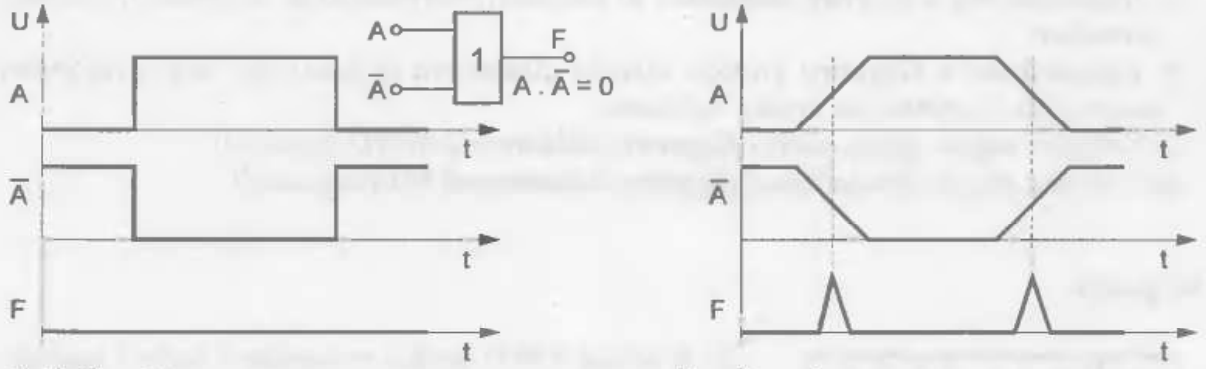
A *4.17. ábrán* látható logikai hálózatban az áthúzott inverterekre nincs szükség mivel kétszeres tagadást valósítanak meg ($\overline{\overline{Y}} = Y$). Ezeket a kapukat elhagyva, a megvalósított hálózat a *4.18 ábrán* látható.



4.21. ábra. Egyszerűsített többszintű logikai hálózat NOR kapukkal

4.1.3. Nem megfelelő jelek a kimeneten, hazárdok

Az eddigiek során mindig feltételeztük, hogy a jelek binárisan vagy jelen vannak a bemeneteken vagy nem – átmeneti állapotaik nincsenek. Ha azonban a logikai áramkörök bemenetére gyors impulzussorozatot adunk, akkor számolnunk kell ezen jelek véges felfutási és lefutási idejével is.



a) ideális eset *b) valós eset*

4.22. ábra. Áramköri hazárdok keletkezése

Ha egy *ÉS* áramkör bemeneteire egy változót és annak negáltját visszük, a kimeneti jelnek – legalább is a Boole-algebra alapszabályai szerint – folyamatosan **0** logikai szinten kellene lennie (*4.22.a ábra*). A valóságban azonban a fel és lefutó élek időtartama alatt impulzusok keletkez(het)nek, melyek a további áramkörökben zavarokat idézhetnek elő (*4.22.b ábra*). Magyarázatként megjegyezhető, hogy az *ÉS* áramkör kimeneti feszültsége mindig a kisebb bemeneti jellel egyezik meg.

Az ***áramköri hazárdok*** megszüntetése speciális eljárást igényel. Elsősorban persze fel kell deríteni őket, és ez nem is olyan egyszerű. Az

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

logikai függvény teljesen „ártalmatlan"-nak tűnik, míg nem vesszük észre, hogy B és C logikai **1** értéke mellett az előző eset áll elő. Természetesen vannak módszerek a kellemetlen kimeneti jelek távoltartására. Így megoldást jelent, ha a függvényt kibővítjük egy új tag hozzáadásával:

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

Ezzel a függvényt nem módosítottuk, amiről annak igazságtáblázatából könnyen meggyőződhetünk. Ez a bővítés azonban mégiscsak figyelmet érdemel, mert előzőleg elfogadott tervezési szempontunknak – minimalizálás a lehető legvégsőkig – ellentmond. Az áramköri hazárdok megszüntetésére több tervezési eljárás ismeretes, amelyekkel itt nem foglakozunk.

Ismételten megemlítjük, hogy a logikai áramkörök kimenete helytelen értékre kerülhet – időlegesen – a logikai változók különböző késleltetési ideje miatt is. Ha egy kimeneti kapura egy jel közvetlenül, a másik jel csak pl. öt más kapun keresztül kerülhet, akkor a jel terjedési ideje alatt – hisz minden kapu mindig több-kevesebb kapacitással van terhelve, így legalább ezért a kimeneti jel késik a bemeneti jelhez képest – a logikai áramkör kimenete helytelen logikai értékre kerülhet.

4.1.4. Példák kombinációs hálózatok megvalósítására

1. feladat:

Tervezzünk kombinációs logikai hálózatot, melynek kimenete **0**, ha ***C*** jelen van, illetve ***A*** jelen van ***B***-vel, vagy ***D***-vel egyidejűleg (*A, B, C, D* logikai változók).

a) Határozza meg a függvény diszjunktív és konjunktív normálalakját decimális (rövidített) formában!
b) Egyszerűsítse a függvényt grafikus eljárást alkalmazva és határozza meg a függvényt megvalósító kombinációs logikai hálózatot!
c) Valósítsa meg az egyszerűsített függvényt kétbemenetű NAND-kapukkal!
d) Valósítsa meg az egyszerűsített függvényt kétbemenetű NOR-kapukkal!

Megoldás:

A	B	C	D	F^4
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

4.2. táblázat. Igazságtáblázat

a) A feladat *ABCD* logikai változókhoz tartozó igazságtábláját a *4.2 táblázat* tartalmazza.
A keresett logikai függvény diszjunktív normálformája:

$$F^4 = \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\overline{D} + \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D + \overline{A}\cdot B\cdot\overline{C}\cdot D + \overline{A}\cdot B\cdot\overline{C}\cdot D + A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}$$

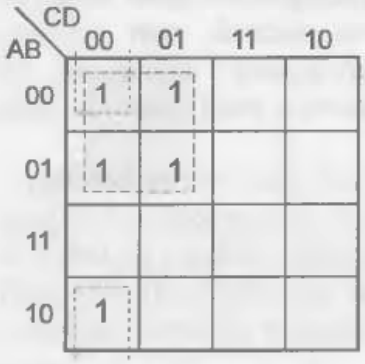
$$F^4 = m_0^4 + m_1^4 + m_4^4 + m_5^4 + m_8^4 = \sum^4(0,1,4,5,8)$$

A keresett logikai függvény, konjunktív normálformája:

$$F^4 = (A+B+\overline{C}+D)\cdot(A+B+\overline{C}+\overline{D})\cdot(A+\overline{B}+\overline{C}+D)\cdot(A+\overline{B}+\overline{C}+\overline{D})\cdot(\overline{A}+B+C+\overline{D})\cdot(\overline{A}+B+\overline{C}+D)\cdot(\overline{A}+B+\overline{C}+\overline{D})\cdot(\overline{A}+\overline{B}+C+D)\cdot(\overline{A}+\overline{B}+C+\overline{D})\cdot(\overline{A}+\overline{B}+\overline{C}+D)\cdot(\overline{A}+\overline{B}+\overline{C}+\overline{D})$$

$$F^4 = M_0^4 + M_1^4 + M_2^4 + M_3^4 + M_4^4 + M_5^4 + M_6^4 + M_8^4 + M_9^4 + M_{12}^4 + M_{13}^4$$

$$F^4 = \prod^4(0,1,2,3,4,5,6,8,9,12,13)$$



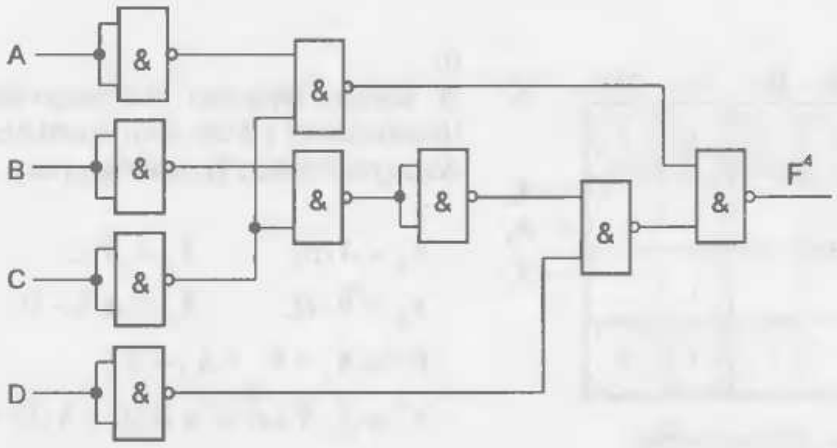
4.23. ábra. Karnaugh-tábla

b) A függvény Karnaugh-táblája a megfelelő tömbösítéssel a *4.23. ábrán* látható.

Az egyszerűsített logikai függvény, a következő:

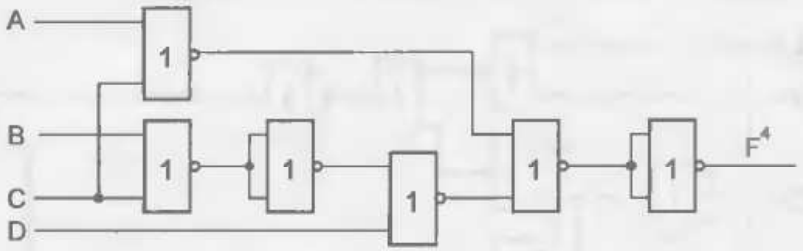
$$F^4 = \overline{A}\cdot\overline{C} + \overline{B}\cdot\overline{C}\cdot\overline{D}$$

c) Az egyszerűsített logikai függvény megvalósítása kétbemenetű *NAND*-kapukkal:



4.24. ábra.

d) Az egyszerűsített logikai függvény megvalósítása kétbemenetű *NOR*-kapukkal:



4.25. ábra.

2. feladat:

Egy kombinációs logikai hálózatot a következő logikai függvény jellemez:

$$F^4 = \sum^4 (0,1,2,3,9,10,11,15)$$

a) *Határozza meg a függvény diszjunktív és konjunktív normálalakját!*
b) *Határozza meg a logikai függvény legegyszerűbb alakját!*
c) *Tervezze meg az egyszerűsített függvényt megvalósító kombinációs logikai hálózatot, ha csak kétbemenetű NOR kapuk állnak a rendelkezésünkre!*

Megoldás:

a) A keresett logikai függvény diszjunktív normálformája:

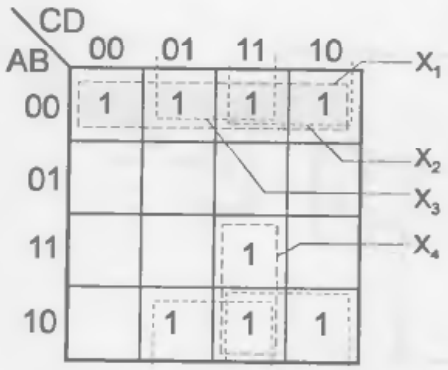
$$F^4 = \sum^4 (0,1,2,3,9,10,11,15) = m_0^4 + m_1^4 + m_2^4 + m_3^4 + m_9^4 + m_{10}^4 + m_{11}^4 + m_{15}^4$$

$$F^4 = \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\overline{D} + \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D + \overline{A}\cdot\overline{B}\cdot C\cdot\overline{D} + \overline{A}\cdot\overline{B}\cdot C\cdot D +$$
$$A\cdot\overline{B}\cdot\overline{C}\cdot D + A\cdot\overline{B}\cdot C\cdot\overline{D} + A\cdot\overline{B}\cdot C\cdot D + A\cdot B\cdot C\cdot D$$

A logikai függvény, konjunktív normálformája:

$$F^4 = \left(A+\overline{B}+C+D\right)\cdot\left(A+\overline{B}+C+\overline{D}\right)\cdot\left(A+\overline{B}+\overline{C}+D\right)\cdot\left(A+\overline{B}+\overline{C}+\overline{D}\right)\cdot\left(\overline{A}+B+C+D\right)\cdot$$
$$\left(\overline{A}+\overline{B}+C+D\right)\cdot\left(\overline{A}+\overline{B}+C+\overline{D}\right)\cdot\left(\overline{A}+\overline{B}+\overline{C}+D\right)$$

$$F^4 = M_1^4 + M_2^4 + M_3^4 + M_7^4 + M_8^4 + M_9^4 + M_{10}^4 + M_{11}^4 = \prod^4(1,2,3,7,8,9,10,11)$$



4.26. ábra. Karnaug-tábla

b)
A logikai függvény Karnaugh-tábláját a megfelelő tömbösítéssel a *4.26. ábra* szemlélteti.
Az egyszerűsített logikai függvény, a következő:

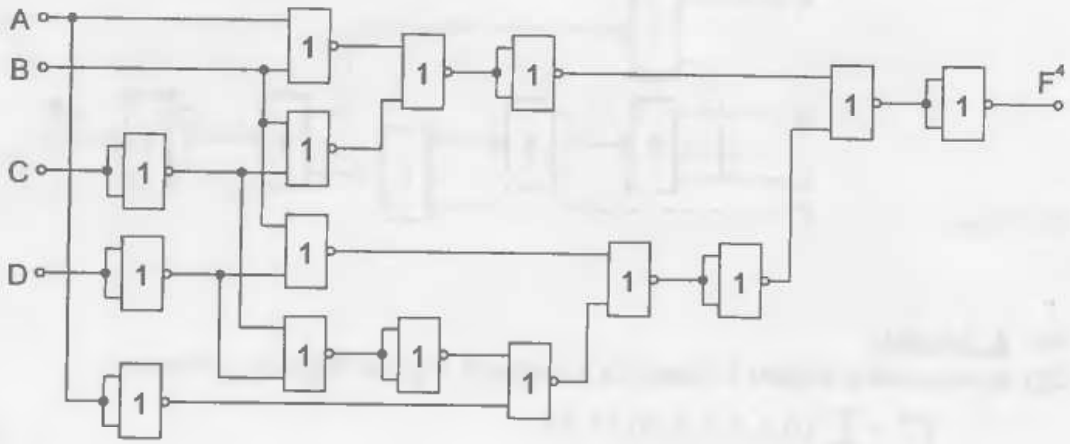
$$X_1 = \overline{A} \cdot \overline{B}, \qquad X_2 = \overline{B} \cdot C$$

$$X_3 = \overline{B} \cdot D, \qquad X_4 = A \cdot C \cdot D$$

$$F^4 = X_1 + X_2 + X_3 + X_4$$

$$F^4 = \overline{A} \cdot \overline{B} + \overline{B} \cdot C + \overline{B} \cdot D + A \cdot C \cdot D$$

c) A logikai függvény megvalósítása kétbemenetű *NOR*-kapukkal:



4.27. ábra.

3. feladat:

Tervezzünk egy olyan kódoló áramkört, amely a 8421 súlyozású *BCD*-kódot *Johnson*-kóddá alakítja! A *Johnson*-kód decimális megfeleltetését az *4.3. táblázat* tartalmazza.

Decimális szám	*Johnson-kód*				
	F_1	F_2	F_3	F_4	F_5
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

4.3. táblázat. A Johnson-kód decimális megfeleltetése

a) Készítse el a kódoló áramkör igazságtábláját!
b) Határozza meg a kódoló kimeneteinek logikai függvényeit a legegyszerűbb formában, szisztematikus egyszerűsítő eljárást alkalmazva!
c) Valósítsa meg a kódoló áramkört megvalósító logikai hálózatot NAND kapukból!
d) Valósítsa meg az F_2 függvényt, csak kétbemenetű NOR kapukat felhasználva!

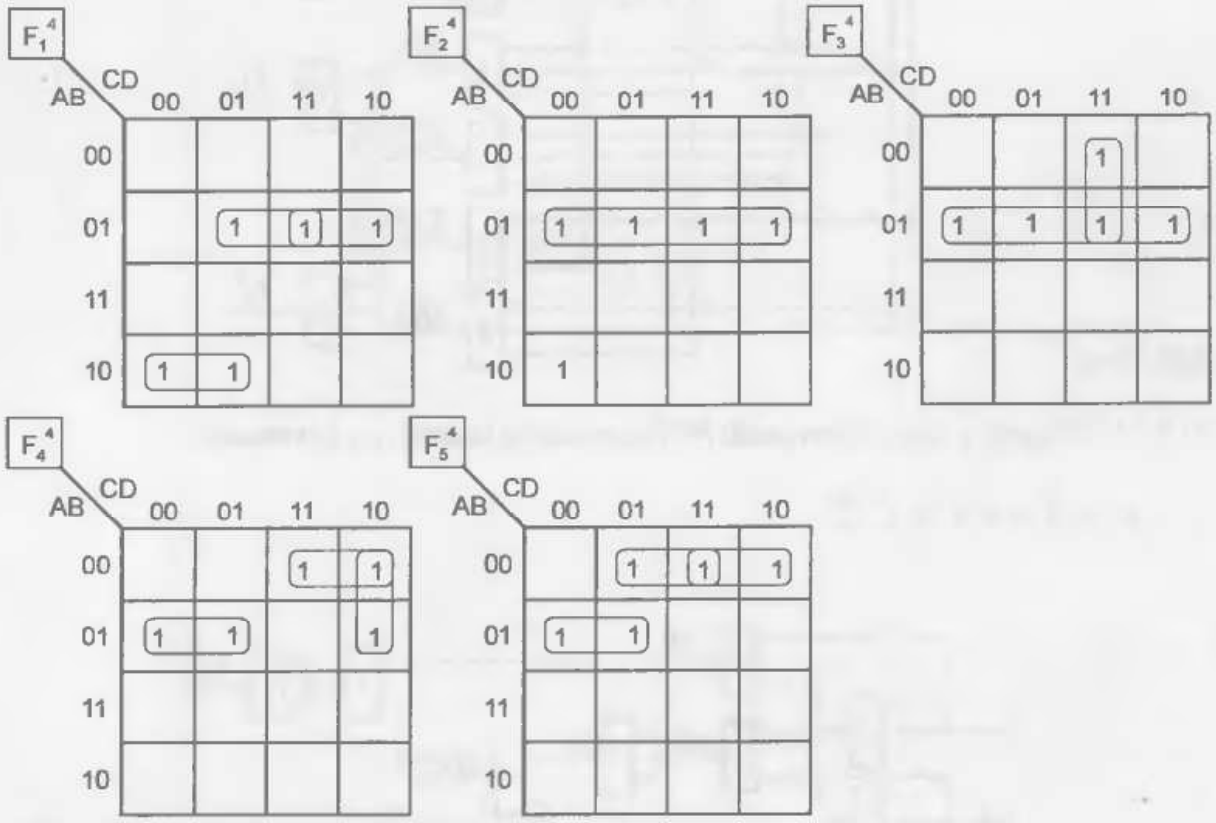
Megoldás:

a) A kódoló áramkör igazságtáblázatát a *4.4. táblázat* tartalmazza:

Decimális szám	BCD-kód súlyozás: 8 4 2 1 A B C D	Johnson-kód				
		F_1^4	F_2^4	F_3^4	F_4^4	F_5^4
0	0 0 0 0	0	0	0	0	0
1	0 0 0 1	0	0	0	0	1
2	0 0 1 0	0	0	0	1	1
3	0 0 1 1	0	0	1	1	1
4	0 1 0 0	0	1	1	1	1
5	0 1 0 1	1	1	1	1	1
6	0 1 1 0	1	1	1	1	0
7	0 1 1 1	1	1	1	0	0
8	1 0 0 0	1	1	0	0	0
9	1 0 0 1	1	0	0	0	0

4.4. táblázat. A kódoló áramkör igazságtáblázata

b) A logikai függvények Karnaugh-tábláit a megfelelő tömbösítéssel a *4.28. ábra* mutatja.



4.28. ábra. A logikai függvények egyszerüsítése

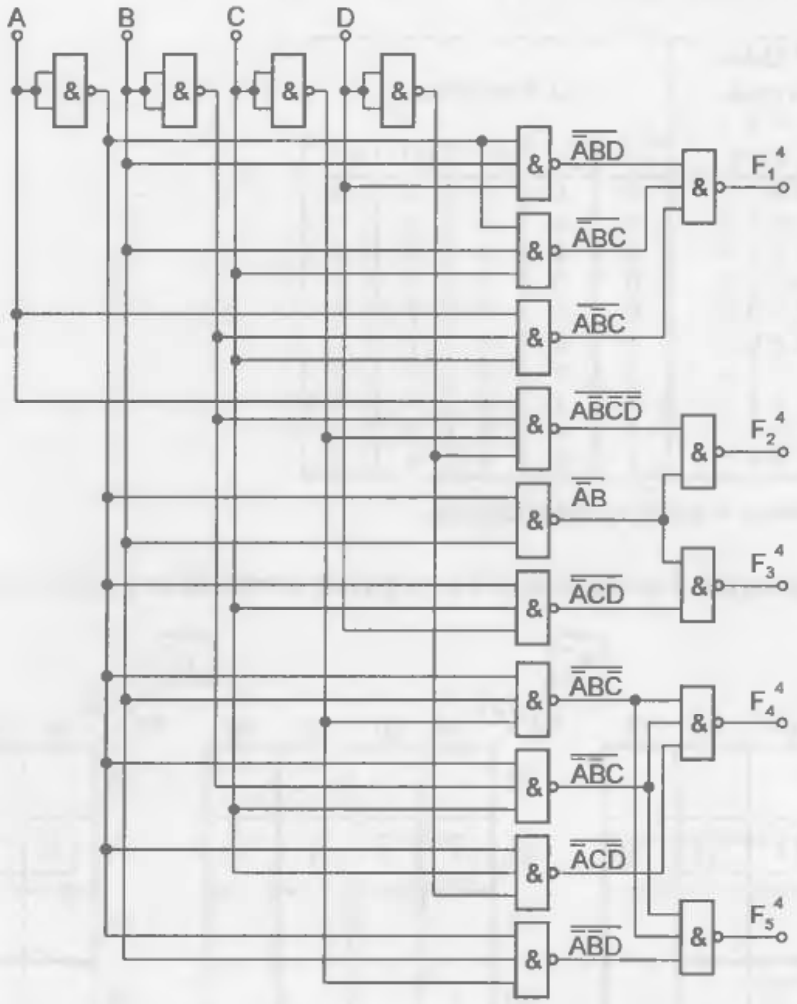
A kimenetek logikai függvényei (a Karnaugh-táblák alapján):

$$F_1^4 = \overline{A} \cdot B \cdot D + \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} \qquad F_2^4 = \overline{A} \cdot B + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$$

$$F_3^4 = \overline{A} \cdot B + \overline{A} \cdot C \cdot D \qquad F_4^4 = \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot C \cdot \overline{D}$$

$$F_5^4 = \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot D + \overline{A} \cdot \overline{B} \cdot C$$

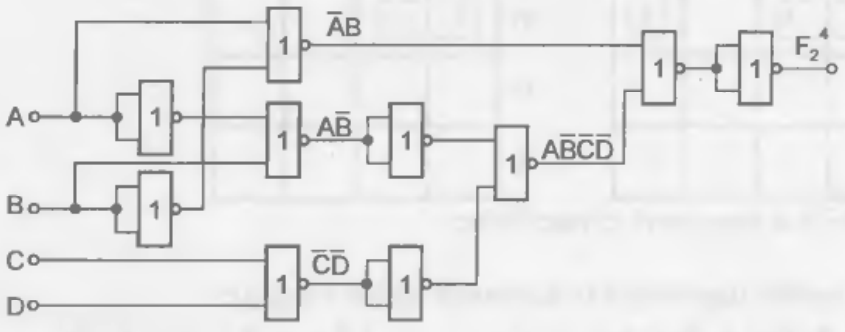
c) A kódoló áramkört *NAND* kapukból megvalósító logikai hálózatot a *4.29. ábra* mutatja.



4.29. ábra.

d) Az F_2^4 függvény csak kétbemenetű *NOR* kapukat felhasználva a következő:

$$F_2^4 = \overline{A} \cdot B + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$$



4.30. ábra.

Összefoglaló kérdések:

1. Mi a különbség a kombinációs- és a szekvenciális hálózat között?
2. Milyen logikai függvényeket megvalósító áramköri elemeket ismer?
3. Mit nevezünk funkcionálisan teljes rendszernek és milyen fajtái vannak?
4. Mi a hálózati szint?
5. Milyen jellemzői vannak a *NAND* kapukkal való logikai függvény-megvalósításnak?
6. Milyen jellemzői vannak a *NOR* kapukkal való logikai függvény-megvalósításnak?
7. Mit nevezünk áramköri hazárdoknak?

Összefoglaló feladatok:

1. feladat:

Egy kombinációs logikai hálózatot a következő logikai függvény jellemzi:

$$F^4 = A \cdot (B + C + D) + A \cdot \overline{B} \cdot C + A \cdot \overline{C}$$

☞

a) Határozza meg a függvény diszjunktív és konjunktív normálalakját decimális (rövidített) formában!

b) Egyszerűsítse a függvényt grafikus eljárást alkalmazva és határozza meg a függvényt megvalósító kombinációs logikai hálózatot!

c) Valósítsa meg az egyszerűsített függvényt kétbemenetű NAND-kapukkal!

d) Valósítsa meg az egyszerűsített függvényt kétbemenetű NOR-kapukkal!

A	B	C	D	F^4
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

4.5. táblázat. Igazságtáblázat

2. feladat:

Egy logikai függvény igazságtáblázatát a *4.5. táblázat* mutatja.

☞

a) Határozza meg a függvény diszjunktív és konjunktív normálalakját decimális (rövidített) formában!

b) Határozza meg a logikai függvény legegyszerűbb alakját!

c) Valósítsa meg az egyszerűsített függvényt kétbemenetű NAND-kapukkal!

3. feladat:
Kétharmados többség megállapítására alkalmas a következő háromváltozós logikai függvény:

$$F^3 = (A \cdot B \cdot C) + (\overline{A} \cdot B \cdot C) + (A \cdot \overline{B} \cdot C) + (A \cdot B \cdot \overline{C}).$$

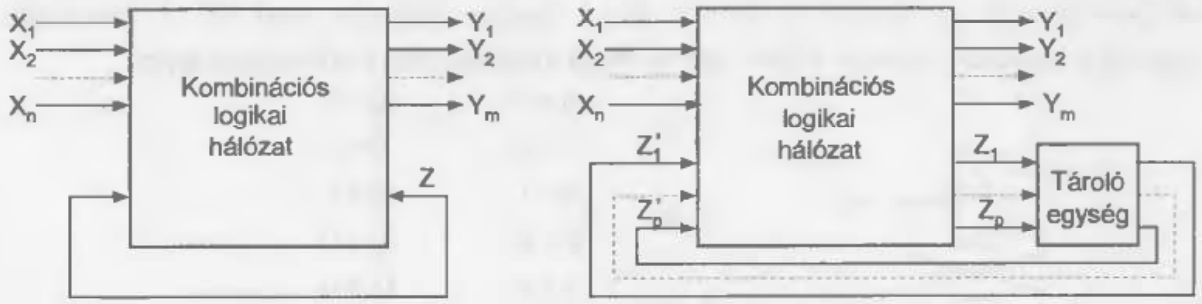
☞

a) Határozza meg a függvény diszjunktív és konjuktív normálalakját decimális (rövidített) formában! Írja fel a függvény igazságtáblázatát!
b) Egyszerűsítse a függvényt grafikus eljárást alkalmazva és határozza meg a függvényt megvalósító kombinációs logikai hálózatot!
c) Valósítsa meg az egyszerűsített függvényt kétbemenetű NOR-kapukkal!
d) Valósítsa meg az egyszerűsített függvényt kétbemenetű NAND-kapukkal!

4.2. Szekvenciális hálózatok

A szekvenciális (*sorrendi*) hálózatok időfüggő logikai függvényeket valósítanak meg. Alapvető jellegzetességük, hogy a kimeneti események alakulását a pillanatnyi bemeneti feltételek mellett a korábbi időpillanatokban bekövetkezett kimeneti események is befolyásolják. Attól függően, hogy az állapotváltozás hogyan következik be, a szekvenciális hálózatok két csoportját különböztetjük meg.

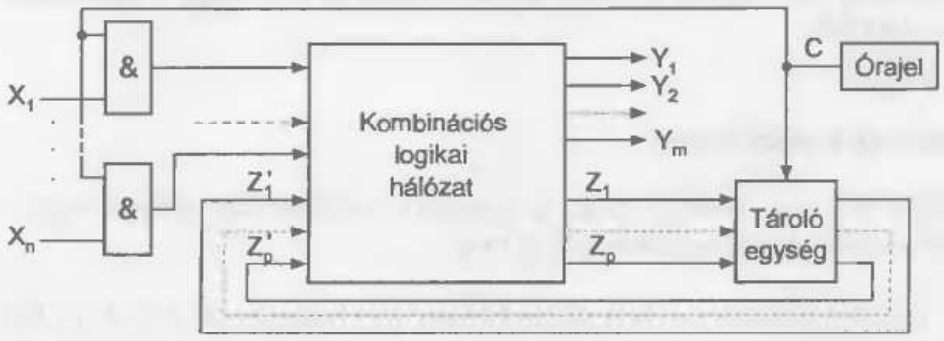
*1. **Aszinkron hálózatok:*** – melyek a kimenet előző állapotától való függését visszacsatolással, vagy tárolókkal valósítják meg (*4.31. ábra*). A kimeneti jellemző (*az áramkör késleltetésétől eltekintve*) valamelyik bemeneti jellemző megváltozására azonnal reagál.



a) *b)*

4.31. ábra. Aszinkron hálózatok tömbvázlata

*2. **Szinkron hálózatok:*** – az állapotváltozás egy ezt engedélyező jel hatására, azzal azonos fázisban zajlik le. Vagyis az állapotváltozás szinkronizálva van egy periodikus vezérlőjellel, amelyet ***C órajelnek*** (*Clock*) vagy *ütemjelnek* neveznek. A kimenet előző állapotától való függést *tárolók* segítségével valósítják meg (*4.32. ábra*).



4.32. ábra. Szinkron hálózatok tömbvázlata

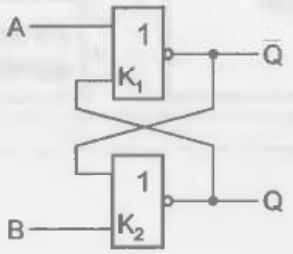
Egy adott órajelperiódus alatt a hálózat S belső állapotát a $Z_1, Z_2, .., Z_p$ jelek képviselik. Ezt az S állapotot az előző órajelperiódus alatti S' állapot, valamint az $X_1, X_2, ..., X_n$ bemeneti jelek határozzák meg. Az S' állapotra jellemző $Z_1', Z_2', ..., Z_p'$ jeleket a hálózat memóriája a szóban forgó órajelperiódus alatt tárolja.

A sorrendi hálózatoknak igen nagy jelentőségük van, mivel a számítástechnikai és automatikai berendezések gyakorlatilag sorrendi hálózatoknak tekinthetők. Közös jellemzője ezeknek a hálózatoknak, hogy kombinációs logikai hálózatot is tartalmaznak.

4.2.1. Integrált tároló áramkörök

A digitális áramkörök legfontosabb adattároló eleme a *bistabil multivibrátor* (billenőáramkör) vagy más nevén ***flip-flop***. A flip-flopnak két fontos tulajdonsága van: két ellentétes állapottal rendelkezik, külső beavatkozás nélkül akármelyiket megtartja, és egy vagy több bemenettel van ellátva, amelyek lehetővé teszik az áramkör egyik vagy másik állapotba való billenését. A flip-flopok a sorrendi hálózatok csoportjába tartoznak. Egy *flip-flop* két állapota révén **1** ***bit*** információ tárolására alkalmas. Az első fejezetben már foglakoztunk a tranzisztorokból felépített egyszerű tárolókkal. A továbbiakban a logikai kapukból felépülő tárolók működését tárgyaljuk.

A tárolás mechanizmusát próbáljuk végigkövetni egy *NOR* kapukból felépített áramkör segítségével (*4.33. ábra*). A kimenetek (Q) egymás negáltjai. Tételezzük fel, hogy bekapcsolás után az áramkör a $Q=0$, $\bar{Q}=1$ logikai állapotot veszi fel. A kimenetek állapotát a bemeneti változók logikai szintje fogja meghatározni a következőképpen:



4.33. ábra. Elemi tároló áramkör

1. Ha $A=0$, $B=0$ $\Rightarrow$ $Q=0$, $\bar{Q}=1$
2. Ha $A=1$, $B=0$ $\Rightarrow$ $Q=1$, $\bar{Q}=0$
3. Ha $A=0$, $B=1$ $\Rightarrow$ $Q=0$, $\bar{Q}=1$
4. Ha $A=1$, $B=1$ $\Rightarrow$ a Q *és* $\bar{Q}$ kimeneten azonos logikai érték adódik, ami nem lehetséges, ezért ez a bemeneti kombináció az áramkör számára nem megengedhető.

A *NOR* kapukból felépített áramkör esetében megállapítható, hogy a Q kimenet logikai értéke a bemeneti kombinációtól függően változik. Ez lehetőséget ad arra, hogy a bemeneten megjelenő információt tároljuk.

4.2.1.1. Tároló áramkörök logikai típusai

A tároló áramkörök esetén – aszerint, hogy a bemeneti kombinációra milyen logikai értéket adnak – többféle logikai típust különböztetünk meg.

R-S típusú tároló: – igazságtáblázata a *4.34.a ábrán* látható. Két bemenettel (S és R) és két kimenettel (Q^{n+1} *és* $\overline{Q^{n+1}}$) rendelkezik. Az ***S***-el jelölt bemenet a ***beíró-bemenet*** (***Set***:-beírás, angol kifejezés), amelyen keresztül a tárolandó információ beírható. A másik ***R***-el jelölt (***Reset***:-törlés, angol kifejezés) bemeneten keresztül a tárolt információ törölhető. Az ***R-S*** tárolóknál az S=1, R=1 bemeneti kombináció határozatlan értékű kimenetet eredményez (*tiltott bemeneti kombináció*). Ez természetesnek tekinthető, mivel egyidőben beírni és törölni értelmetlen.

Inverz R-S típusú tároló: – igazságtáblázata a *4.34.b ábrán* látható. Ez a tároló áramkör az ***R-S*** tároló inverz függvényét valósítja meg.

J-K típusú tároló: – igazságtáblázata a *4.34.c ábrán* látható. A ***J-K*** flip-flop kiküszöböli az ***R-S*** tárolók azt a hibáját, hogy az S=1, R=1 bemeneti kombináció határozatlan értékű kimenetet eredményez. Ebben az esetben a beíró bemenet a ***J***, és a törlő-bemenet a ***K***.

T típusú tároló (trigger): – egy adatbemenettel és egy vezérlőbemenettel rendelkezik. A ***T*** típusú tároló jellegzetessége, hogy ha a bemenetére – valamilyen periodicitás szerint – logikai **1** szintet juttatunk, a kimenet állapota a vezérlőjel ütemében változik **0** vagy **1** logikai értékre. A flip-flop elnevezése (*trigger*) is azt tükrözi, hogy *„triggerel"* (*billen*). Igazságtáblázata a *4.34.d ábrán* látható. Megfigyelhető, ha a ***J-K*** flip-flop esetén kizárjuk a $J \neq K$ logikai értékkombinációkat (*vagyis ha egy J-K tároló bemeneteit összekötjük*) akkor a ***T*** típusú tároló igazságtáblázatát kapjuk.

D típusú tároló: – a bemenetére adott információt a kimenetén egy vezérlőjel időtartamával késleltetve jelenteti meg. Az elnevezése is innen ered (***Delay***: -késleltetés, angol kifejezés). Igazságtáblázata a *4.34.e ábrán* látható. Megfigyelhető, ha a ***J-K*** flip-flop esetén kizárjuk a $J = K$ logikai értékkombinációkat (*vagyis ha egy J-K tároló bemenetei közé egy invertert kötünk*), akkor a ***D*** típusú tároló igazságtáblázatát kapjuk.

S	R	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	x

a) R-S tároló

$\overline{S}$	$\overline{R}$	Q^{n+1}
0	0	x
0	1	1
1	0	0
1	1	Q^n

b) inverz R-S tároló

J	K	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	$\overline{Q^n}$

c) J-K tároló

T	Q^{n+1}
0	Q^n
1	$\overline{Q^n}$

d) T tároló

D	Q^{n+1}
0	0
1	1

e) D tároló

4.34. ábra. Tároló áramkörök igazságtáblázata

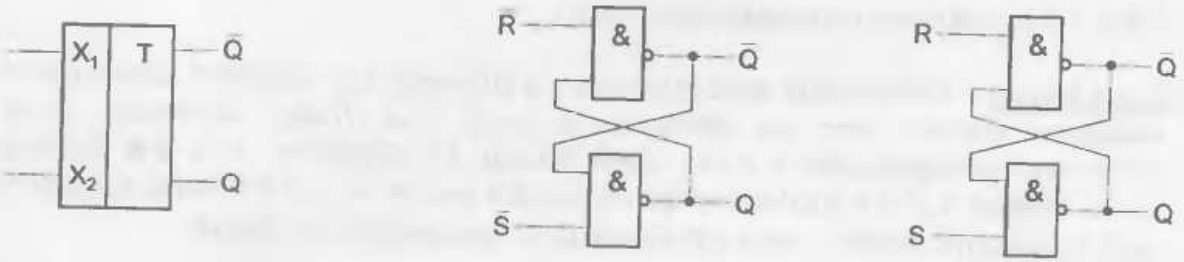
4.2.1.2. Tároló áramkörök vezérlés jellegétől függő típusai

A tároló áramkörök állapotváltozását a bemeneti logikai kombinációk mellett egy külső vezérlőjel (órajel) is befolyásolhatja. Ezt a körülményt is figyelembe véve (amit a tároló *vezérlésének* nevezünk) a tároló áramköröket négy fő csoportra oszthatjuk:

- ***statikus tárolók***,
- ***statikus kapuzott tárolók***,
- ***kétfokozatú (master-slave) tárolók***,
- ***élvezérelt tárolók***.

Statikus tárolók

A bemenetre kapcsolt információ hatására a kimenet logikai értéke (az áramkör jelkésleltetésétől eltekintve) azonnal megváltozik. A statikus tárolók hátránya, hogy a bemenetre kerülő véletlen jel (esetleg zaj) is állapotváltozást eredményez. A *4.35.a ábra* a statikus tároló áramkörök általános jelölését mutatja (X_1 és X_2 a statikus bemenetek és Q, $\overline{Q}$ a kimenet). Statikus ***R-S*** és ***inverz R-S*** tároló kapukból felépített kapcsolási rajza látható a *4.35.b* és *4.35.c ábrán.*

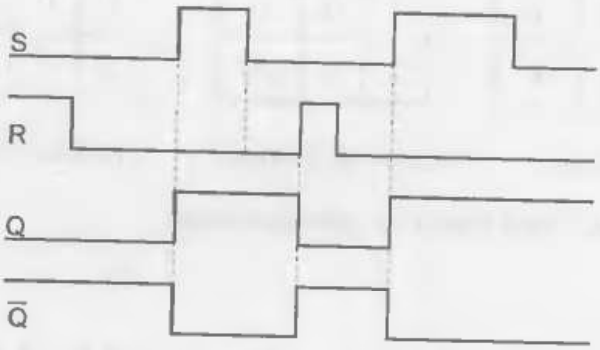


a) Statikus tárolók jelölése *b) R-S tároló* *c) inverz R-S tároló*

4.35. ábra. Statikus tárolók

Az ***RS*** flip-flop a legegyszerűbb aszinkron sorrendi hálózat. Jellegzetessége, hogy az állapota billenésszerűen a bemenő jelkombináció változására azonnal változik (*4.36. ábra*). Ez a tulajdonság sok esetben alkalmatlanná teszi ezt a típusú flip-flopot a szinkron sorrendi hálózatok felépítésére.

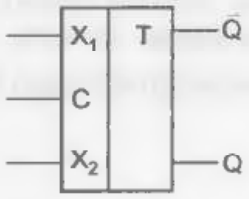
S	R	Q_{n+1}	Q_{n+1}
0	0	Q_n	Q_n
0	1	0	1
1	0	1	0
1	1	*	*



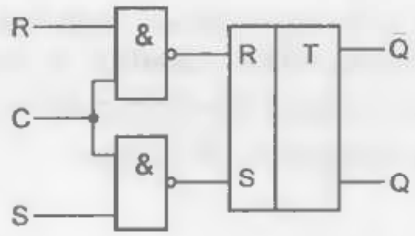
4.36. ábra. RS flip-flop igazságtáblázata. és jelalakjai

Kapuzott statikus tárolók

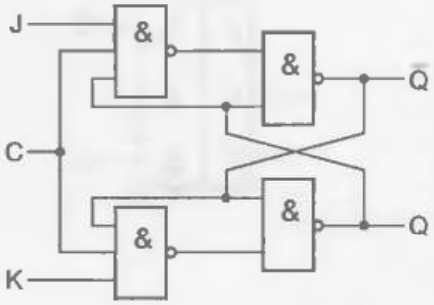
A kimenet logikai értéke, a bemenetre kapcsolt jelkombináción kívül egy *kapuzó jel* (órajel: ***C***) értékétől is függ. A kapuzó jel a bemenetre kerülő véletlen jel hatását küszöböli ki. A vezérelt flip-flop állapota csakis a ***C*** kapuzó jellel szinkronban változhat. A *4.37.a ábra* a kapuzott statikus tároló áramkörök általános jelölését mutatja (X_1 és X_2 a statikus bemenetek, ***C*** a kapuzó jel és Q, $\overline{Q}$ a kimenet). Mind az ötféle logikai típusú tároló megvalósítható kapuzott statikus tárolóként. Kapuzott statikus ***R-S*** tároló a *4.37.b* ábrán, ***J-K*** tároló a *4.37.c ábrán*, ***T*** tároló a *4.37.d ábrán* és ***D*** tároló a *4.37.e ábrán* látható.



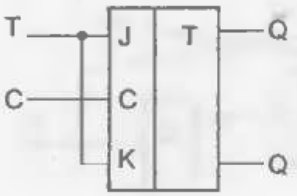
a) általános jelölés



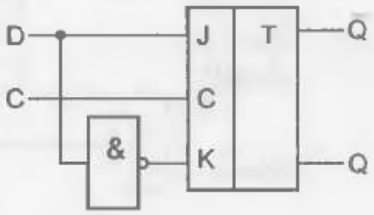
b) kapuzott statikus RS tároló



c) kapuzott J-K tároló



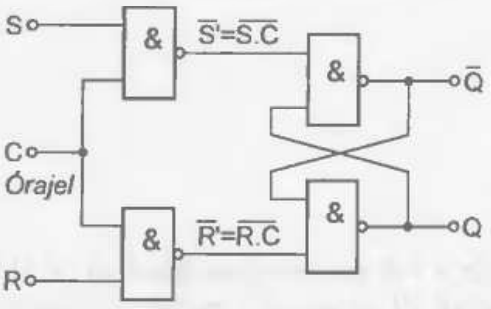
d) kapuzott T tároló



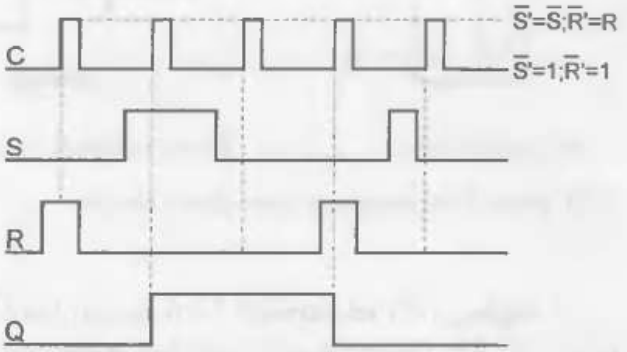
e) kapuzott D tároló

4.37. ábra. Kapuzott (órajel-vezérlésű) statikus tárolók

A *4.38. ábrán* bemutatott órajellel vezérelt *RS* flip-flop állapota csakis a *C* órajellel szinkronban változhat. Két órajel-impulzus között (amikor $C = 0$) a *NEM-ÉS* kapukból felépített flip-flop $\overline{S}'$ és $\overline{R}'$ bemeneti jelei $\overline{S}' = \overline{R}' = 1$, és függetlenek az *S* és *R* jelektől. Ezért két órajel-impulzus között a *Q* és $\overline{Q}$ kimenetek nem változnak meg. A rövididejű órajel-impulzus alatt (amikor $C = 1$) $\overline{S}' = \overline{S}$ és $\overline{R}' = \overline{R}$. Ekkor a bemeneti jelek vezérlik a *NEM-ÉS* kapukból felépített flip-flopot. A flip-flop állapota az *4.34.b ábrán* levő igazságtáblázat segítségével határozható meg. A flip-flop akkor billen az *S* és *R* jelek által meghatározott állapotba, amikor megjelenik az órajel-impulzus ($C = 0 \rightarrow 1$). Abban az esetben, ha $S = R = 0$, az órajel nem változtat a flip-flop állapotán. Az $S = R = 1$ bemenő jelkombináció az órajel-impulzus alatt elkerülendő, mivel a flip-flop állapota az órajel-impulzus után meghatározhatatlan.



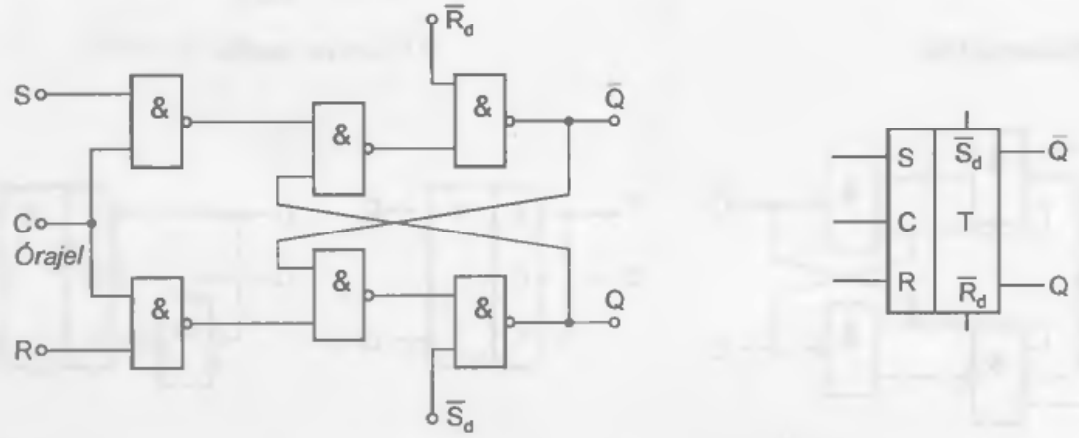
a) kapcsolási rajz



b) impulzusdiagram

4.38. ábra. Kapuzott *RS* flip-flop

Egyes áramkörökben szükséges, hogy a flip-flop nemcsak az órajellel szinkronban működjön, hanem akármikor, az órajeltől függetlenül is lehessen vezérelni. Az *4.39. ábra* egy ilyen típusú flip-flopot mutat be, amelynek erre a célra két közvetlen (*direkt*) beíró ($\overline{S}_d$) és törlő bemenete ($\overline{R}_d$) is van.



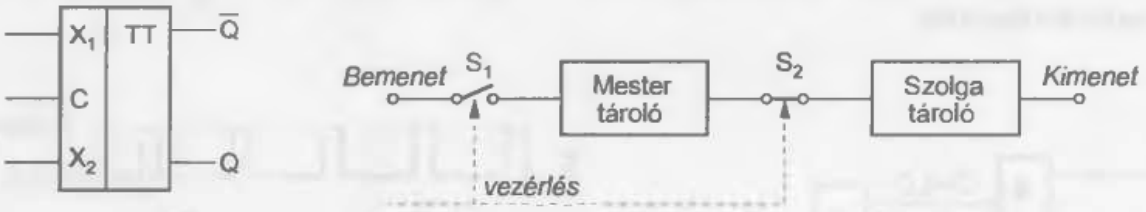
a) felépítés *b) rajzjel*

4.39. ábra. Órajel- és közvetlen vezérlésű RS flip-flop

Kétfokozatú tárolók

A szakirodalom ***master-slave*** (*mester-szolga*) tárolónak is nevezi. Általános jelölésmódját és elvi működési vázlatát a *4.40. ábra* szemlélteti. A kétfokozatú tároló működése a következő (*4.40.b ábra*):

- Az S_1 és S_2 kapcsoló mindig ellenfázisban működik. Ha S_1 záródik a bemeneti információ az első (***mester***) tárolóba kerül, mivel S_2 nyitva van így az információ nem jut el a második (***szolga***) tárolóba.
- Ha S_1 nyit a mester tárolóba nem lehet információt bevinni; mivel S_2 zár az információ a mesterből átíródik a szolga tárolóba és megjelenik a kimeneten.



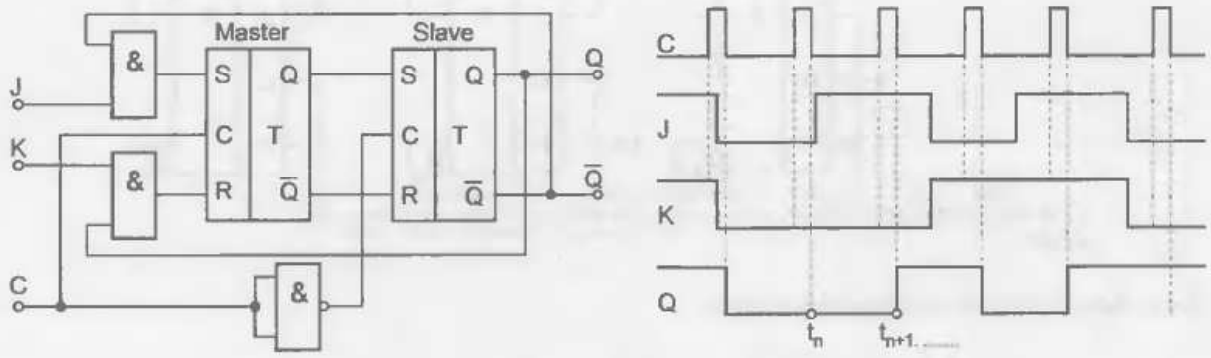
a) általános jelölés *b) elvi működési vázlat*

4.40. ábra. Kétfokozatú (master-slave) tárolók

A legnagyobb jelentőségű kétfokozatú tároló áramkör a *J-K* master-slave flip-flop (*4.41. ábra*). Ez két összekapcsolt órajelvezérlésű *RS* flip-flopból áll, az egyik a master a másik a slave. Működése elemzésének első lépéseként a master flip-flop *S* és *R* bemeneteire kapcsolt *ÉS* kapukat ne vegyük figyelembe.

Az áramkör működése a *4.41 ábra* alapján követhető.

- Az órajelimpulzus alatt, amíg $C = 1$, a master (*mester*) billenésszerűen azonnal követi az S és R jelkombináció által meghatározott állapotot. Ugyanakkor a slave (*szolga*) flip-flop órajele $C' = \overline{C} = 0$, és ennek következtében állapota teljesen független a master flip-flop állapotától.



a) felépítése *b) jelalakjai*

4.41. ábra. A *JK* kétfokozatú (master-slave) tároló

- Két órajelimpulzus között (amikor $C = 0$) a master flip-flop tartja az órajelimpulzus utolsó pillanatában felvett állapotot. Ez idő alatt a slave flip-flop órajele $C' = \overline{C} = 1$, és ezáltal átveszi a master állapotát. Tehát az S és R bemenet csak akkor írja be az információt a master flip-flopba, amikor $C = 1$, és a slave flip-flop csak akkor veszi át ezt az információt, amikor $C = 0$. Sok esetben zavaró, hogy az $S = R = 1$ jelkombináció tiltott. Ezt meg lehet szüntetni a bemenetekre kapcsolt *ÉS* kapuk segítségével. Abban az esetben, ha fennáll a $J = K = 1$ jelkombináció, akkor $S = \overline{Q}$ és $R = Q$. Tehát $S = \overline{R}$, ami azt jelenti, hogy S és R nem lehetnek egyidejűleg logikai **1** szinten.

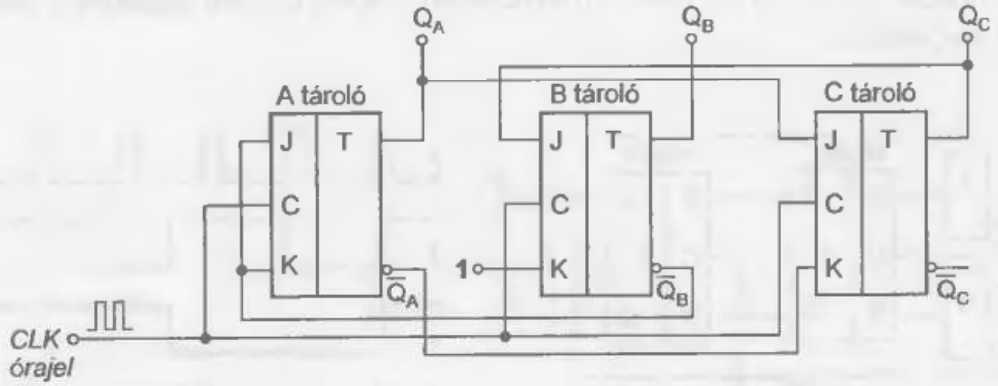
4.2.2. Szekvenciális hálózatok vizsgálata

A szekvenciális hálózatok vizsgálata során – az egyszerűség kedvéért – olyan áramkörökkel foglalkozunk, amelyeknél figyelmen kívül hagyhatók az áramkörök instabilitásával és átmeneti jelenségeivel összefüggő problémák. Egy szekvenciális áramkör *instabilitása* alatt azt értjük, hogy nem határozható meg előre az áramkör kimenetének logikai értéke (pl. egy R-S tárolónál az R=S=1 esetén kialakuló állapot ilyennek tekinthető). Az *átmeneti jelenség* azt jelenti, hogy az áramkör az egyik stabil állapotból a másik stabil állapotba való átmenet közben milyen köztes állapotot vesz fel.

Mivel ezek a problémák főleg az aszinkron hálózatokra jellemzők, a következőkben az áttekinthetőbb működésű szinkron hálózatokkal fogunk részletesebben foglalkozni. Az áramkörök vizsgálata során a szekvenciális hálózat leírására az állapotdiagramot és az ütemdiagramot fogjuk használni.

4.2.2.1. Szinkron hálózatok vizsgálata

Feladat: A *4.42. ábrán* látható szinkron szekvenciális hálózat elemzése.



4.42. ábra: Szinkron szekvenciális hálózat

a) Az áramkör vizsgálata állapotdiagram felhasználásával:

- tételezzük fel, hogy a kiinduló állapotban a tárolók kimenetei a $Q_A = 0$, $Q_B = 0$. és $Q_C = 0$ logikai értékeket veszik fel.
- A *4.42. ábra* alapján a tárolók vezérlési függvényei:

$$J_A = \overline{Q}_B \qquad K_A = \overline{Q}_B$$
$$J_B = Q_C \qquad K_B = 1$$
$$J_C = Q_A \qquad K_C = \overline{Q}_A$$

- Megfigyelhető, hogy az *A* jelű J-K tároló, ebben az esetben *T* típusú tárolóként fog működni. Behelyettesítve a kiinduló állapotnak megfelelő logikai értékeket a vezérlési függvényekbe:

J_A	K_A	Q_A^{n+1}
1	1	1

J_B	K_B	Q_B^{n+1}
0	1	0

J_C	K_C	Q_C^{n+1}
0	1	0

- Az órajel hatására a Q^{n+1}-gyel jelölt állapotot veszi fel a logikai hálózat. Az eredményként kapott logikai állapotok új értékeit ismét beírjuk a vezérlő függvényekbe:

J_A	K_A	Q_A^{n+1}
1	1	0

J_B	K_B	Q_B^{n+1}
0	1	0

J_C	K_C	Q_C^{n+1}
1	0	1

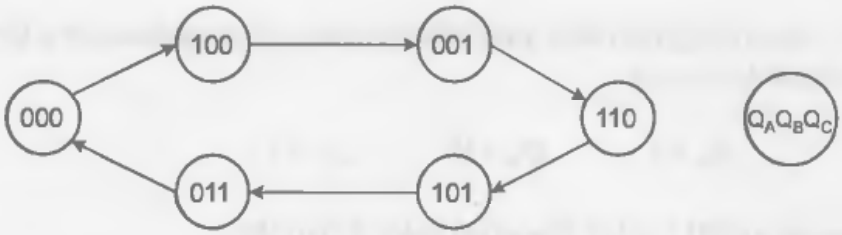
- Ezt a módszert addig folytatjuk, amíg valamelyik előző logikai állapotot fel nem veszi a szekvenciális hálózat.

J_A	K_A	Q_A^{n+1}
1	1	1
0	0	1
1	1	0
0	0	0

J_B	K_B	Q_B^{n+1}
1	1	1
0	1	0
1	1	1
1	1	0

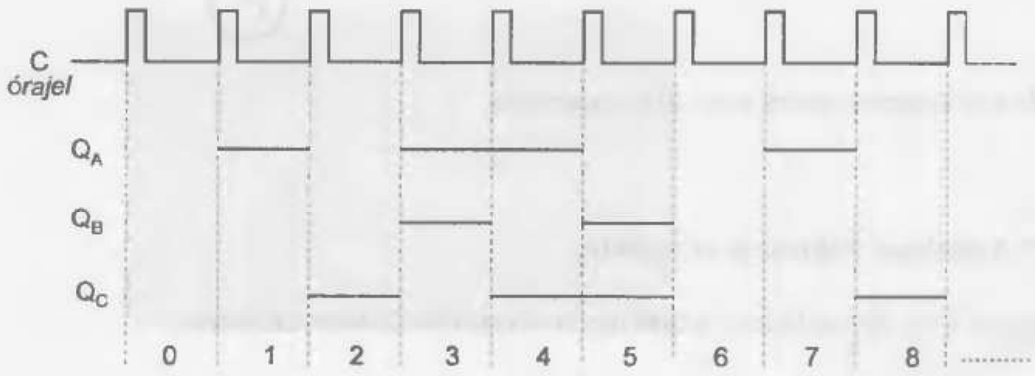
J_C	K_C	Q_C^{n+1}
0	1	0
1	0	1
1	0	1
0	1	0

• A *4.43. ábra* a kimenetek Q^{n+1} állapotait szemlélteti.



4.43. ábra. A szinkron hálózat állapotdiagramja

b) Az áramkör vizsgálata ütemdiagram felhasználásával:
– az ***ütemdiagram*** olyan idődiagram, ahol a vezérlőjeleket és a hatásukra létrejövő kimenőjeleket tüntetik fel az idő függvényében. Az egyes ütemeket nem idő mértékegységgel jellemezzük, hanem a vezérlő jelek változása határoz meg egy működési fázist. Hogy a logikai rendszertől és az áramköri megvalósítástól független diagramot kapjunk, nem feszültségszinteket ábrázolunk, hanem csak egy vízszintes vonallal jelöljük azokat a működési fázisokat, ahol a vizsgált jel logikai értéke **1**. A *4.42. ábrán* látható szinkron hálózat ütemdiagramját a *4.44. ábra* szemlélteti.



4.44. ábra. A szinkron hálózat ütemdiagramja

A *4.44. ábrát* tanulmányozva megfigyelhető, hogy bármilyen – a zárt ciklusban szereplő – kiinduló állapotot is választunk, az állapotsorozat a *4.43. ábra* szerinti marad.

A vizsgált szinkron hálózatban három flip-flop van, amelyek segítségével nyolc különböző állapot hozható létre. Az eddigi elemzésekben viszont csak hat állapot szerepelt. A két hiányzó állapot:

1. $Q_A = 0 \qquad Q_B = 1 \qquad Q_C = 0.$

A vezérlő függvényekbe való behelyettesítés után, az áramkör a következő állapotot veszi fel:

$$Q_A = 0 \qquad Q_B = 0 \qquad Q_C = 0,$$

vagyis a **000** logikai állapotnál belép a ciklusba.

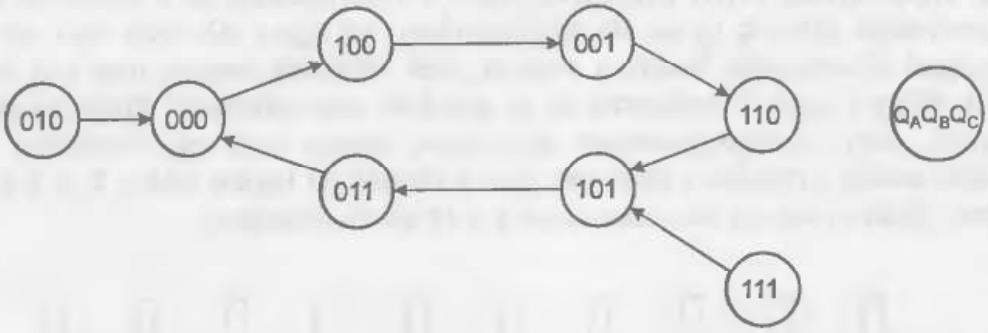
2. $Q_A = 1 \qquad Q_B = 1 \qquad Q_C = 1.$

A vezérlő függvényekbe való behelyettesítés után, az áramkör a következő állapotot veszi fel:

$$Q_A = 1 \qquad Q_B = 0 \qquad Q_C = 1.$$

vagyis az **101** logikai állapotnál belép a ciklusba.

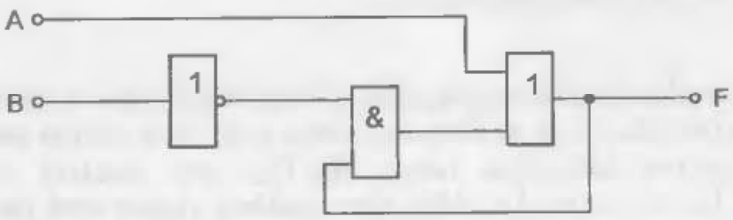
A szinkron hálózat állapotdiagramja – ezen eseteket is figyelembe véve – a *4.45 ábrán* látható.



4.45. ábra. A szinkron hálózat teljes állapotdiagramja

4.2.2.2. Aszinkron hálózatok vizsgálata

Feladat: A *4.46. ábrán* látható aszinkron szekvenciális hálózat elemzése.



4.46. ábra. Aszinkron hálózat

Az ábrán látható aszinkron hálózat elemzésénél figyelembe kell venni, hogy a kimeneti *F* jel logikai értékét nemcsak az *A*, *B* bemeneti jelek, hanem a kimenet előző állapota is befolyásolja. A működés leírására az előbbiekben alkalmazott egyszerűsített állapotdiagram nem használható. Az itt alkalmazható állapotdiagramban fel kell tüntetni a kimenetek logikai állapotain kívül a vezérlőjel kombinációkat is, melyek az egyik állapotból a másik állapotba való átmenet feltételei.

Az aszinkron hálózat kimeneti logikai függvénye:

$$F^{n+1} = A + \overline{B} \cdot F^n .$$

A kimeneti függvény egyenlete alapján a következő kombinációk esetén az áramkör nem változtatja meg a logikai állapotát:

$F^{n+1} = 1$, ha $A=1$ és $B=1$ és $F^n=1$
ha $A=0$ és $B=0$ és $F^n=1$
ha $A=1$ és $B=0$ és $F^n=1$

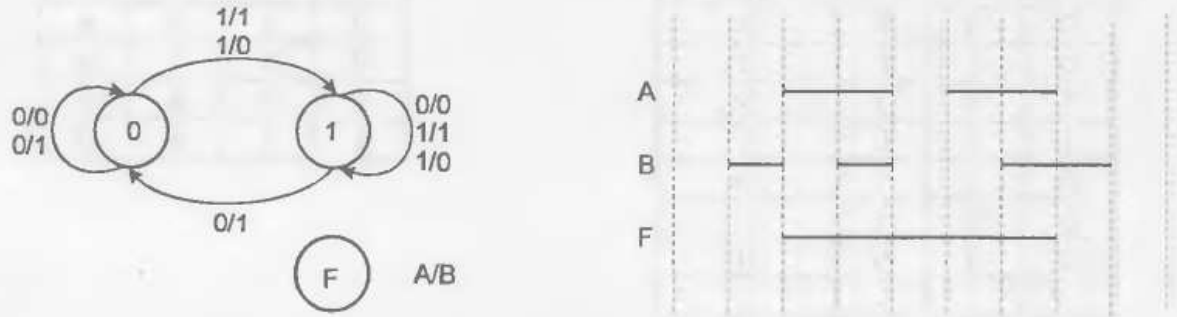
$F^{n+1} = 0$, ha $A=0$ és $B=1$ és $F^n=0$
ha $A=0$ és $B=0$ és $F^n=0$

A következő kombinációk esetén az áramkör megváltoztatja logikai állapotát:

$F^{n+1} = 1$, ha $A=1$ és $B=1$ és $F^n=0$
ha $A=1$ és $B=0$ és $F^n=0$

$F^{n+1} = 0$, ha $A=0$ és $B=1$ és $F^n=1$

Az aszinkron hálózat állapotdiagramját és ütemdiagramját a *4.47.ábra* mutatja.

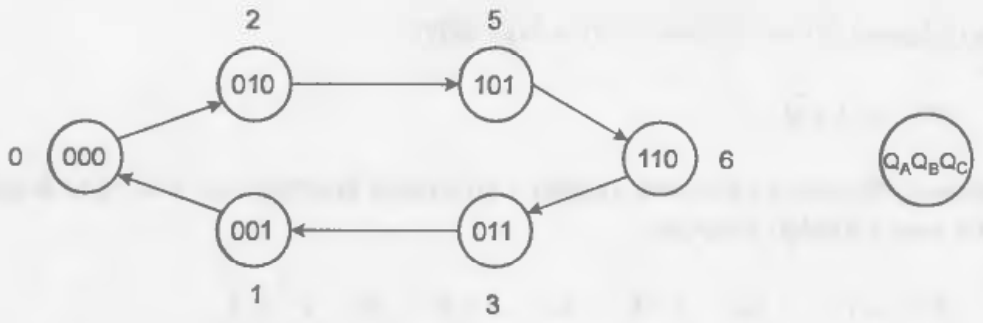


a) *b)*

4.47. ábra. Az aszinkron hálózat állapotdiagramja (a) és ütemdiagramja (b)

4.2.3. Szekvenciális hálózatok megvalósítása

Valósítsunk meg egy szinkron szekvenciális hálózatot, amelynek állapotdiagramja a *4.48. ábrán* látható. A hálózat megvalósítására *J-K* flip-flopok és kétbemenetű *NOR* kapuk használhatók.



4.48. ábra. A szinkron hálózat állapotdiagramja

Megoldás:

• Mivel a különböző állapotok száma 6, az állapotok bináris kódolásához három flip-flop elegendő (A, B és C tároló – Q_A, Q_B, Q_C kimenetekkel).

• Kijelöljük a tetszőlegesen kiválasztható kezdő állapotot (feltételezzük, hogy a kimenetek a $Q_A = 0, Q_B = 0, Q_C = 0$ logikai értékeket veszik fel). Felírjuk az állapotok sorrendjét binárisan valamint decimálisan kódolva (*4.6. táblázat*).

A flip-flopok megfelelő állapotváltozásához minden J és minden K bemenethez kombinációs hálózatot kell kialakítani. Ebben segít a *4.7. táblázat*, amely szemlélteti, hogy milyen vezérlőjeleket kell kapcsolni egy flip-flopra, hogy a kívánt állapotváltozás bekövetkezzen.

Q_A	Q_B	Q_C	*Decimális érték*
0	0	0	0
0	1	0	2
1	0	1	5
1	1	0	6
0	1	1	3
0	0	1	1
0	0	0	0
..	..	..	..

4.6. táblázat. A kimenetek állapotai

Q^n	Q^{n+1}	J	K
0	0	0	***h***
0	1	1	***h***
1	0	***h***	1
1	1	***h***	0

4.7. táblázat. Szükséges vezérlőjelek

A *4.7. táblázatban* foglaltak értelmezése a következő:

- ahhoz, hogy a flip-flop **0** állapota ne változzon: beírni nem szabad ($J = 0$), a K jelű törlés bemenet logikai állapota pedig közömbös (vagyis **0** vagy **1** közül bármelyik logikai értéket rákapcsolhatunk). Azt, hogy egy logikai változó **0** vagy **1** értékű lehet és ezek az állapotok a kimenet értékét nem befolyásolják, a továbbiakban *határozatlannak* vagy *közömbösnek* nevezzük és „*h*"-val jelöljük;
- ahhoz, hogy a flip-flop **0**-ról **1**-re billenjen: be kell írni ($J = 1$), míg a törlés bemenetre kapcsolt logikai érték közömbös;
- ahhoz, hogy a flip-flop **1**-ről **0**-ra billenjen: törölni kell ($K = 1$), míg a J beíró bemenetre kapcsolt logikai érték közömbös;

– ahhoz, hogy a flip-flop **1** állapota ne változzon: nem szabad törölni ($K = 0$), míg a J beíró bemenetre kapcsolt logikai érték közömbös.

A kombinációs hálózatok logikai függvényeit ezek után – flip-floponként – Karnaugh-tábla segítségével írjuk fel. Az eredményül kapott Karnaugh-táblákat a *4.49. ábra* mutatja.

• A *4.6. és 4.7. táblázat* felhasználásával meghatározzuk a J és K vezérlési táblákat:

– a hálózat $0 \rightarrow 2$ átmenetéhez:

Q_A-nak **0**-ban kell maradnia: $J_A = 0$ és $K_A = h$

Q_B-nek **1**-re kell változnia: $J_B = 1$ és $K_B = h$

Q_C-nek **0**-ban kell maradnia: $J_C = 0$ és $K_C = h$

– a hálózat $2 \rightarrow 5$ átmenetéhez:

Q_A-nak **1**-re kell változnia: $J_A = 1$ és $K_A = h$

Q_B-nek **0**-ra kell változnia: $J_B = h$ és $K_B = 1$

Q_C-nek **1**-re kell változnia: $J_C = 1$ és $K_C = h$

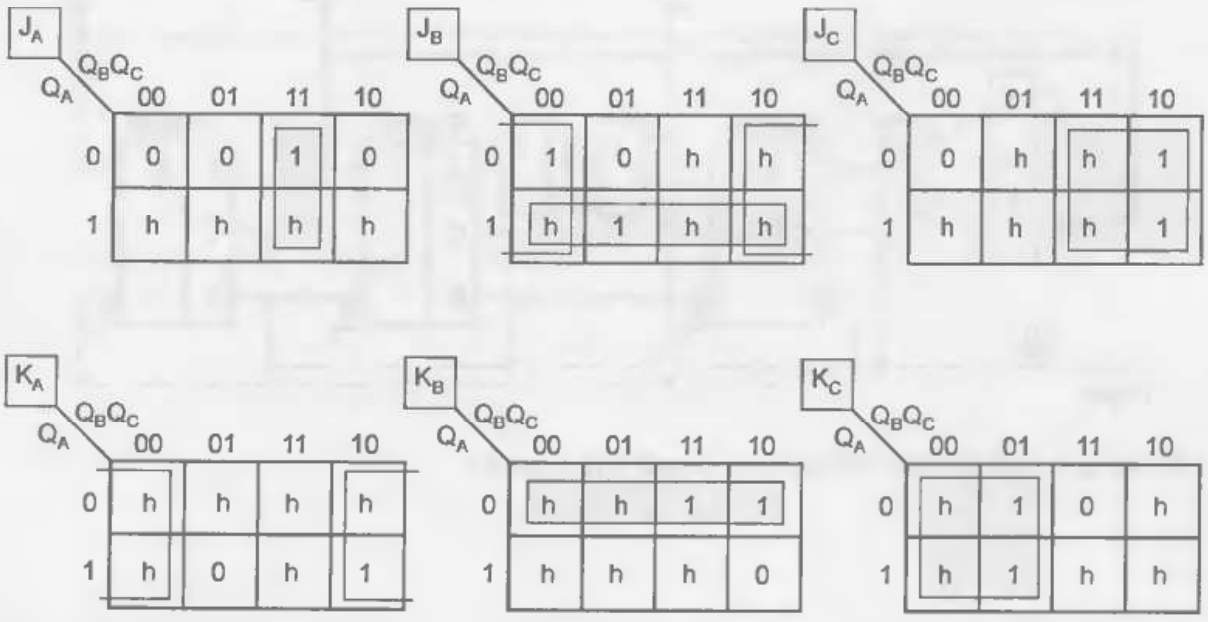
– a hálózat $5 \rightarrow 6$ átmenetéhez:

Q_A-nak **1**-ben kell maradnia: $J_A = h$ és $K_A = 0$

Q_B-nek **1**-re kell változnia: $J_B = 1$ és $K_B = h$

Q_C-nek **0**-ra kell változnia: $J_C = h$ és $K_C = 1$

Ezt az eljárást mindaddig folytatjuk, míg minden állapotból biztosítottuk a következő állapotba való átmenet feltételeit. A hálózat állapotai között nem szereplő állapotoknál – az előzőekben már ismertetett – tiltott kombinációkra vonatkozó elvek alkalmazhatók. Ebben az esetben a függvények egyszerűsítésére használjuk fel ezeket az állapotokat (vagyis a megfelelő cellákba „*h*" jelet írunk).

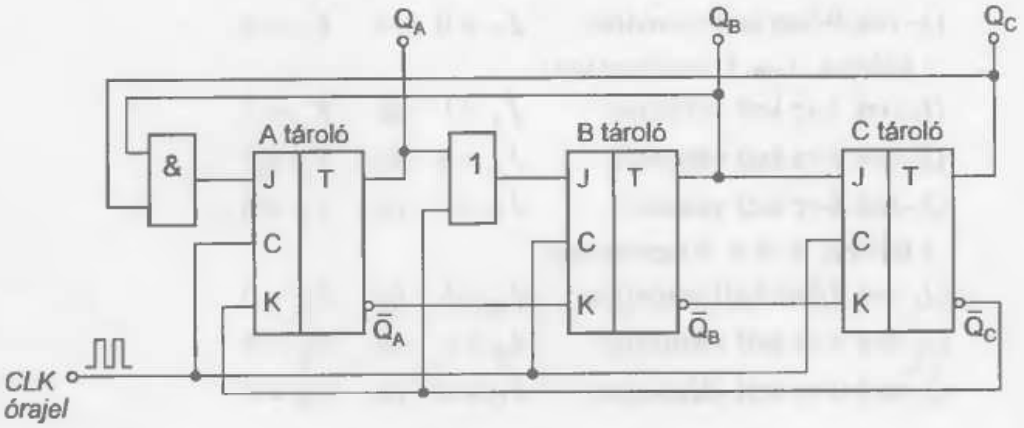


4.49. ábra. A kombinációs hálózat függvényei flip-floponként

- Az egyszerűsítések után a tárolók vezérlési függvényei:

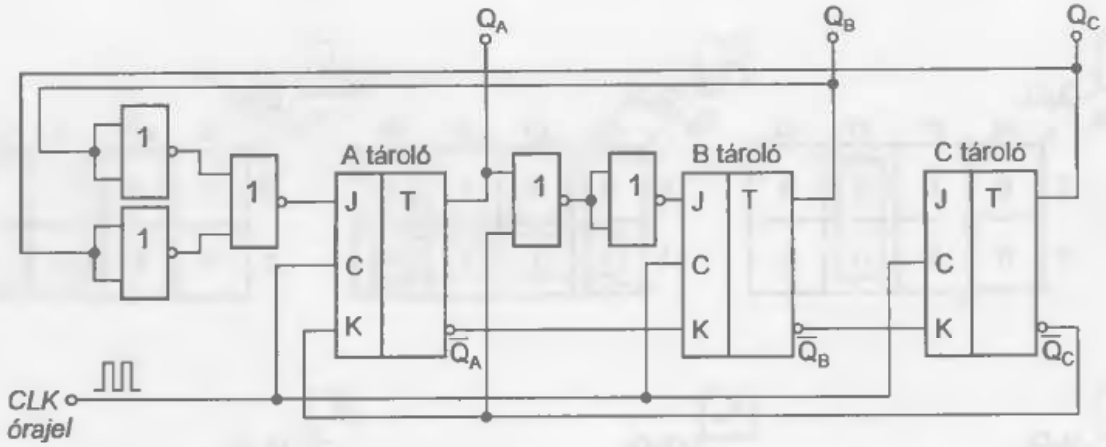
$$J_A = Q_B \cdot Q_C \qquad K_A = \overline{Q}_C$$
$$J_B = \overline{Q}_C + Q_A \qquad K_B = \overline{Q}_A$$
$$J_C = Q_B \qquad K_C = \overline{Q}_B$$

A megvalósított szekvenciális hálózat a *4.50. ábrán* látható.



4.50. ábra. A megvalósított szekvenciális hálózat

A hálózatot kétbemenetű *NOR* kapuk felhasználásával a *4.51. ábra* mutatja.



4.51. ábra. A szekvenciális hálózat kétbemenetű *NOR* kapukkal

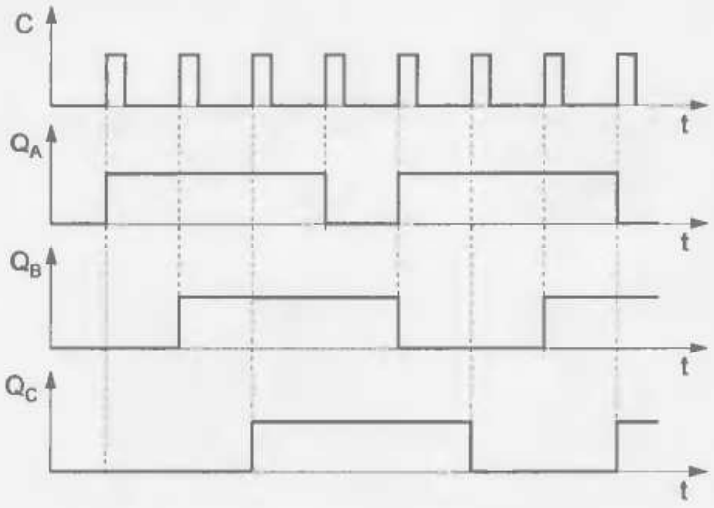
Összefoglaló feladatok:

1. feladat:
Valósítson meg egy szinkron szekvenciális hálózatot, amely a következő – decimálisan kódolt – állapotokkal rendelkezik:

a) 0, 3, 5, 7, 1, 0, ...
b) 0, 7, 4, 5, 3, 0, ...
c) 0, 2, 1, 6, 5, 4, ...
d) 0, 1, 2, 3, 5, 7, ...

A megvalósításhoz *J-K* flip-flopok állnak rendelkezésre!

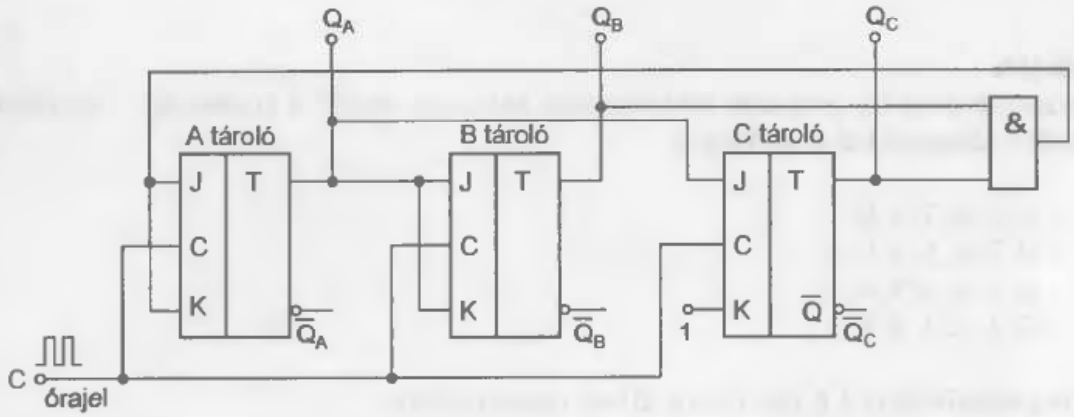
2. feladat:
Tervezzen olyan szinkron szekvenciális hálózatot, amely a *4.52. ábrán* látható jelalakokat állítja elő a kimenetein ciklikusan! A megvalósításhoz *J-K* flip-flopok használhatók!



4.52. ábra. A szekvenciális hálózat impulzusdiagramja

a) Írja fel a jelalakok igazságtáblázatát!
b) Írja fel a vezérlési függvényeket!
c) Rajzolja meg a hálózat kapcsolási rajzát!

3. feladat:
Elemezze a *4.53. ábrán* látható szinkron hálózat működését!



4.53. ábra.

a) Írja fel a *J* és *K* bemenetek logikai függvényeit!
b) Készítse el a szekvenciális áramkör állapotdiagramját

4.3. Digitális jelek szétválasztása és egyesítése

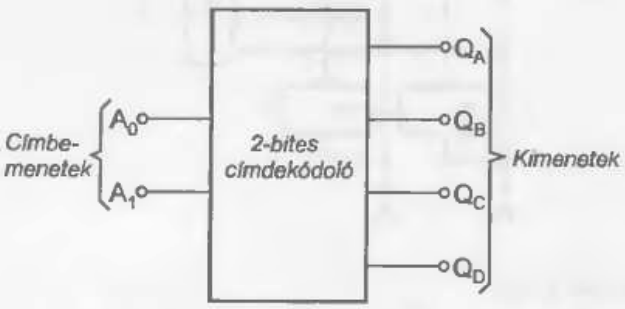
A digitális technikában gyakori feladatnak tekinthető a több vonalon érkező információból való válogatás, vagyis több bemeneti jel időben egymás után történő rákapcsolása az áramkör kimenetére. Ez a kapcsolástechnikai feladat az *adatszelekció, adatválasztás.*

Az ***adatszelektor*** feladata, hogy a különböző adatok sorozatából a kívánt adatokat kiválassza és a kimeneten keresztül továbbítsa. A bemeneti adatok időben egymás után, az úgynevezett *időmultiplex* eljárással továbbíthatók. Azt az áramkört, amely a független bemeneti jeleket időben egymás után a kimenetre továbbítja, ***multiplexernek*** nevezzük. A multiplexer tulajdonképpen egy időtől függően vezérelt adatszelektornak tekinthető. Az érkező adatok időben egymás után, különböző kimenetekre szét is oszthatók. Azt az áramkört, amely a bemenetén megjelenő adatokat adott utasítással az egyik meghatározott kimenetre kapcsolja, ***demultiplexernek*** nevezzük.

4.3.1. Címdekódolók

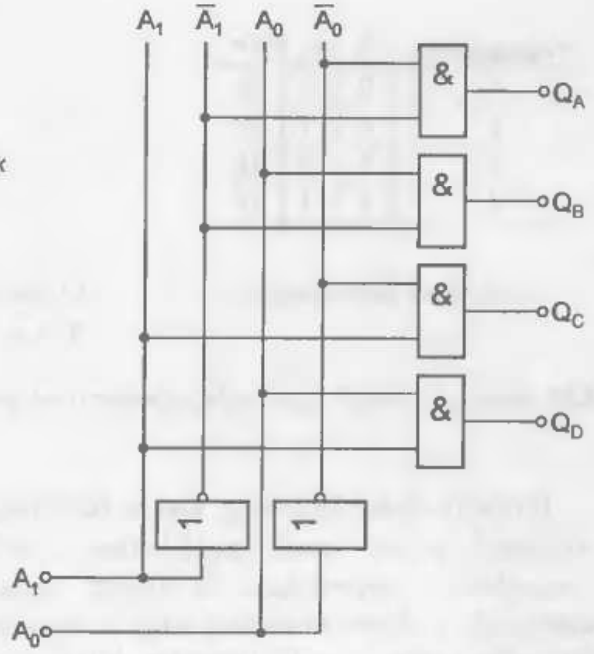
A digitális technikában a különböző elemek vezérelt elérésére úgynevezett ***címre*** van szükség, amely alatt meghatározott hosszúságú bináris jelsorozatot (bináris szót) értünk. Ennek megfelelően vannak, pl. 2-bites, 4-bites, 8-bites, stb. címek.

A ***címdekódoló*** egy olyan többkimenetű áramkör, amelynek – a címbemenet alapján kiválasztandó – kimenetén logikai **1** szint jelenik meg. A *4.56. ábra* egy 2-bites címdekódoló tömbvázlatát és kapcsolási rajzát mutatja.



Cím	A_1	A_0	Q_A	Q_B	Q_C	Q_D
1	0	0	1	0	0	0
2	0	1	0	1	0	0
3	1	0	0	0	1	0
4	1	1	0	0	0	1

a) tömbvázlat és igazságtáblázat



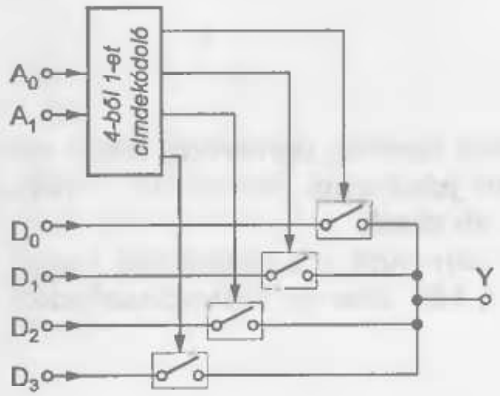
b) kapcsolási rajz

4.54. ábra. 2-bites címdekódoló

4.3.2. Adatszelektorok (multiplexerek)

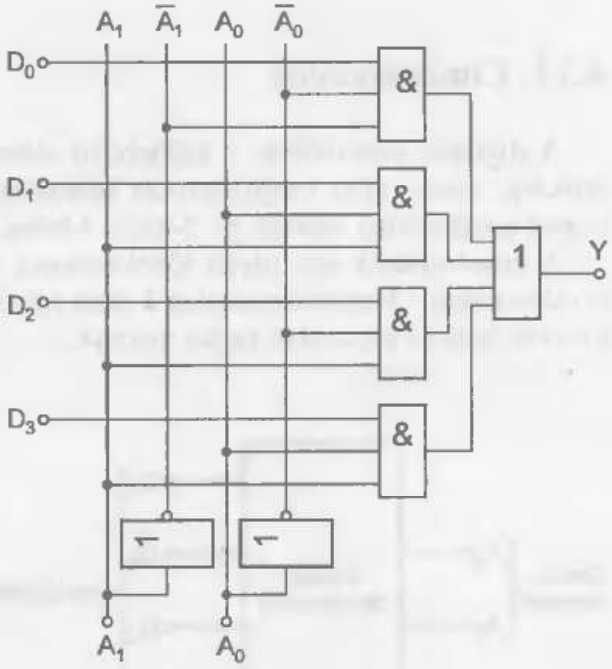
Az adatszelektort, vagy multiplexert abban az esetben alkalmazzuk, amikor több vonal jeléből egyet kell kiválasztani. Azt, hogy melyik csatorna jelét kapcsoljuk a kimenetre, a *címmel* jelöljük ki.

A *4.55.a ábra* az „1 a 4-ből" típusú adatszelektor tömbvázlatát és igazságtáblázatát mutatja. Az adatszelektornak négy bemenete van, amelyek bármelyikét össze lehet kapcsolni az *Y* kimenettel. Tulajdonképpen az adatszelektor úgy működik, mint egy 4 állású fokozatkapcsoló. A digitális multiplexereket természetesen nem mechanikus kapcsolókkal valósítják meg, hanem kombinációs logikai áramkörökkel. A kapcsolófokozat beállítása a vezérlőbemenetekkel történik. A négy különböző kapcsolófokozat digitális vezérléséhez 2 vezérlőbemenet (*címbemenet*) szükséges (A_0, A_1), mivel két vezérlőbittel négy különböző utasítás állítható elő. Az „1 a 4-ből" típusú adatszelektor logikai kapukkal megvalósított változatát a *4.55.b ábra* szemlélteti.



Kapcsolóállás	A_1	A_0	Y=
1	0	0	D_0
2	0	1	D_1
3	1	0	D_2
4	1	1	D_3

a) tömbvázlat és igazságtáblázat

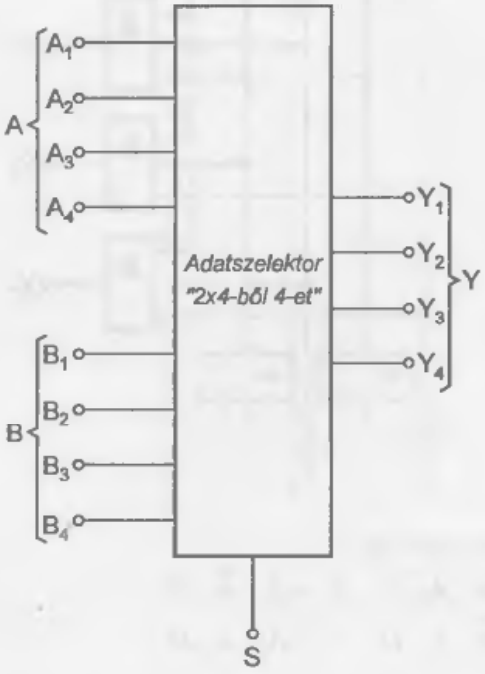


b) kapcsolási rajz

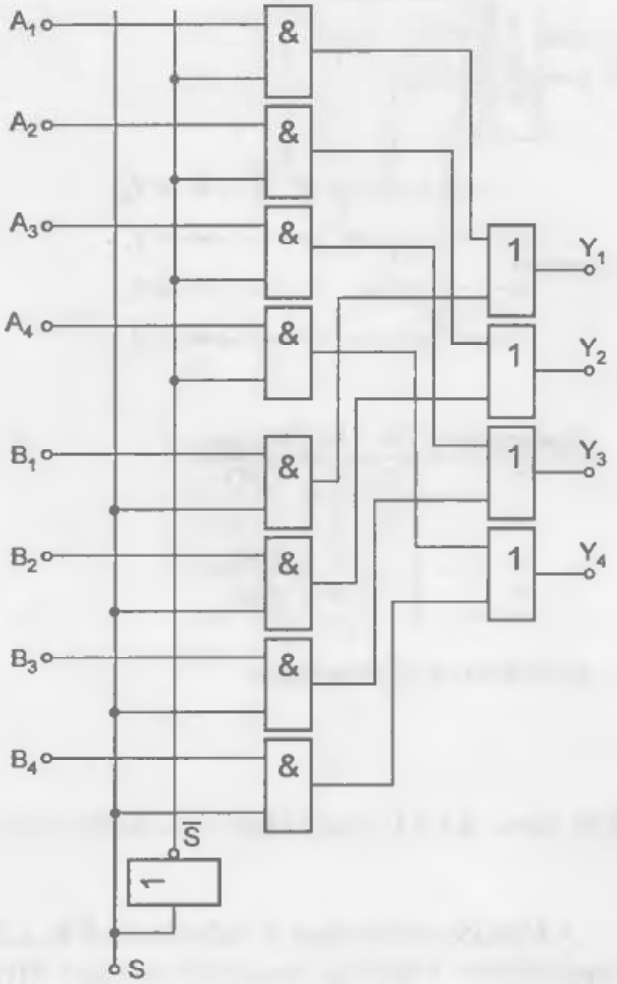
$$Y = \overline{A}_0 \cdot \overline{A}_1 \cdot D_0 + A_0 \cdot \overline{A}_1 \cdot D_1 + \overline{A}_0 \cdot A_1 \cdot D_2 + A_0 \cdot A_1 \cdot D_3$$

4.55. ábra. „1 a 4-ből" típusú adatszelektor (multiplexer)

Hasonló elven lehetőség van a bemenetek bitszámának tetszőleges növelésére, ha a kimenetek számát ennek megfelelően növeljük. A *4.56. ábra* egy „4 a 2×4-ből" típusú adatszelektor tömbvázlatát és logikai kapukkal megvalósított kapcsolását mutatja. Az adatszelektor ebben az esetben vagy a négybites *A* bemenetet, vagy a négybites *B* bemenetet kapcsolja a négybites *Y* kimenetre. Mivel csak két kapcsolóállás lehetséges, ezért egyetlen vezérlőjel (*S*) is elegendő.



Kapcsolóállás	S	Y_{1-4}=
1	0	A_{1-4}
2	1	B_{1-4}



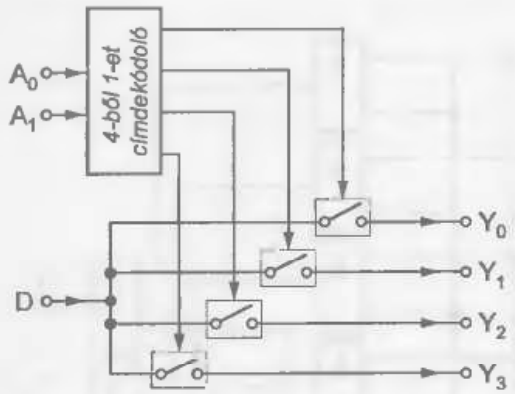
a) tömbvázlat és igazságtáblázat *b) kapcsolási rajz*

4.56. ábra. „4 a 2×4-ből" típusú adatszelektor (multiplexer)

4.3.3. Adatelosztók (demultiplexerek)

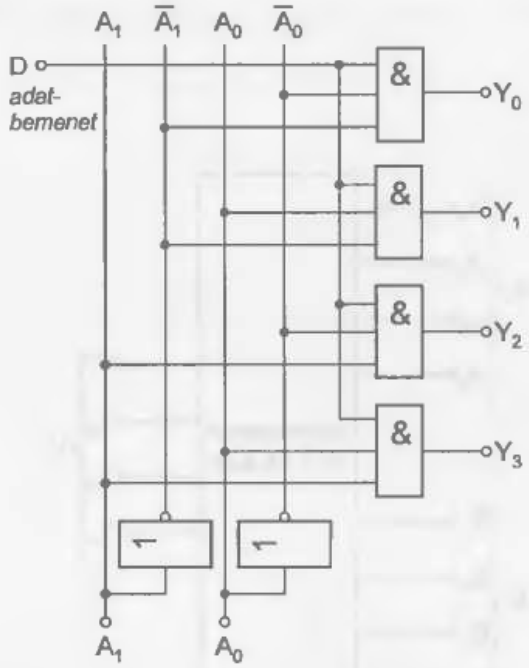
Az adatelosztók vagy más néven *demultiplexerek* egy *D* bemeneti információt – a cím által kiválasztott – kimenetre juttatnak. A működés tehát fordított, mint a multiplexer esetén, de meg kell jegyezni, hogy logikai kapu-hálózattal megvalósított multiplexer nem alkalmas fordított jel-iránnyal való használatra. Egy „4-ből 1-et" típusú demultiplexer elvi működését a *4.57.a ábra* szemlélteti, a *4.57.b ábra* pedig a logikai kapukkal megvalósított kapcsolástechnikai megoldást mutatja. A négy kimenet kiválasztásáraz A_0, A_1 címbemenetek minden kombinációját kihasználjuk.

Amennyiben $D = 1 = állandó$, akkor a vizsgált demultiplexer „4-ből 1-et" típusú dekódolóként működik.



Kapcsolóállás	A_1	A_0	Kimenet
1	0	0	$Y_0=D$
2	0	1	$Y_1=D$
3	1	0	$Y_2=D$
4	1	1	$Y_3=D$

a) tömbvázlat és igazságtáblázat

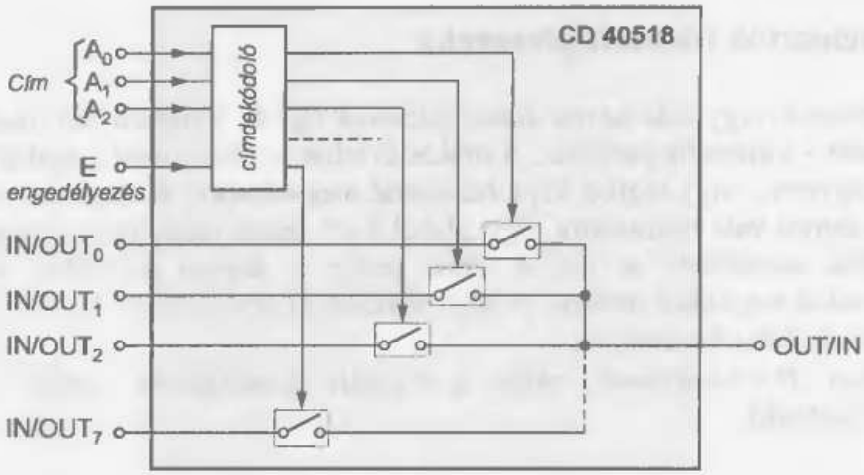


b) kapcsolási rajz

$$Y_0 = \overline{A_0} \cdot \overline{A_1} \cdot D, \quad Y_1 = A_0 \cdot \overline{A_1} \cdot D,$$
$$Y_2 = \overline{A_0} \cdot A_1 \cdot D, \quad Y_3 = A_0 \cdot A_1 \cdot D$$

4.57. ábra. „4 ből 1" típusú adatelosztó (demultiplexer)

A CMOS-technikában a multiplexerek és a demultiplexerek mind kapukkal, mind analóg kapcsolókkal (angolul: *transmission gate*) kivitelezhetők. Analóg kapcsolók alkalmazása esetén a jelátvitel kétirányú lehet, és lehetőség van nemcsak digitális, hanem analóg jelek átvitelére is. Ezért ebben az esetben a multiplexer elvileg megegyezik a demultiplexerrel. Az analóg kapcsolókkal kivitelezett változatot ***analóg multiplexernek*** (***demultiplexernek***) nevezzük. A *4.58. ábra* egy CMOS technológiájú analóg multiplexer/demultiplexer (típusjele: CD 40518) elvi felépítését szemlélteti.



4.58. ábra. Analóg multiplexer/demultiplexer tömbvázlata (CD 40518)

4.4. Regiszterek

A számítógépek és egyéb digitális berendezések működésének elengedhetetlen feltétele az információ – átmeneti vagy hosszabb ideig tartó – tárolása, és az információhoz való gyors hozzáférés. A tárolók fontos jellemzőjének tekinthető a tárolt információ mennyisége és a hozzáférési idő. A nagy tárolási kapacitás és a rövid hozzáférési idő általában ellentmondó követelmények. A különböző tároló áramkörökkel részletesebben majd a 6. fejezetben fogunk foglalkozni.

Abban az esetben, amikor csak kevés információ tárolására (100 bites nagyságrendig) és gyors hozzáférésre van szükség (10÷20 ns), akkor a feladat megoldására flip-flopos tárolók alkalmazhatók. Az ilyen flip-flopos tárolókat – amelyeket általában csoportosan használják egy nagyobb információ egység (pl. byte) tárolására – ***regisztereknek*** nevezzük.

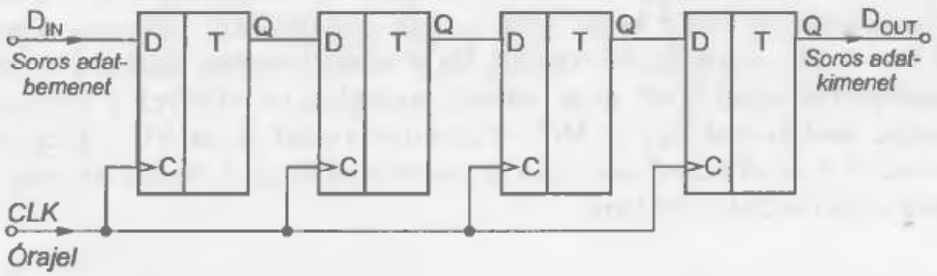
A regiszterek az információbevitel (beírás) és az információkiolvasás szempontjából *soros* vagy *párhuzamos hozzáférésűek* lehetnek. Soros beírásnál és kiolvasásnál a regiszter első és utolsó flip-flopjához lehet hozzáférni. Ebben az esetben szükséges, hogy az információt a regiszterben léptetni lehessen. Ezek a regiszterek a ***léptetőregiszterek***.

Párhuzamos beírásnál és kiolvasásnál az információt a regiszter minden flip-flopjába egyszerre írják be, illetve onnan egyszerre olvassák ki. Mivel ezeknél a regisztereknél léptetés nem szükséges, a regiszter csak tárolási feladatra alkalmas. Ezeket a típusokat ***átmeneti tároló*** vagy ***közbenső tároló*** (*puffer*) ***regisztereknek*** nevezik.

4.4.1. Léptetőregiszterek

A léptetőregiszter flip-flopok olyan lánca, amely lehetővé teszi, hogy a bemenetére adott információ minden egyes órajel hatására egy flip-floppal tovább lépjen. A léptetés irányának a megadására a *jobbra*, illetve *balra* megjelöléseket alkalmazzák. A bemeneti jel áthaladva a láncon késleltetve, de egyébként változatlanul jelenik meg a kimeneten. A léptetőregiszter bináris jelek tárolására szolgál és minden egyes flip-flop, amely a regisztert alkotja egy bit információ tárolására alkalmas. A léptetőregiszterekben a megfelelő működés érdekében (minden léptetési parancsra egy és csakis egy léptetés) feltétlenül órajelvezérelt flip-flopokat kell alkalmazni.

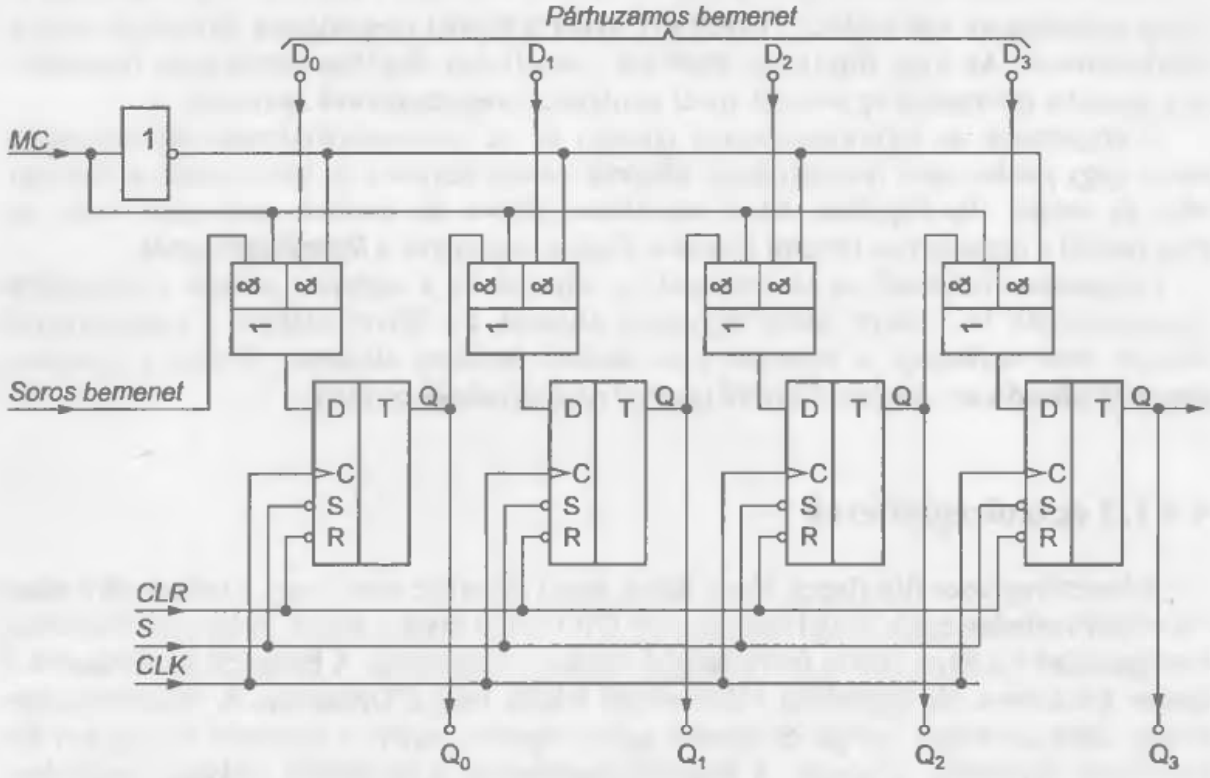
A léptetőregiszterek esetén a soros és párhuzamos beírás és kiolvasás, valamint a kétféle léptetési irány lehetőségének variációival sokféle típus állítható elő. A *4.59. ábra* – a legegyszerűbb változatnak tekinthető – egyirányban (*jobbra*) léptethető, soros beírású és kiolvasású 4-bites áramkör kapcsolását szemlélteti.



4.59. ábra. 4-bites soros léptetőregiszter

Ha a *4.59. ábrán* látható áramkör minden flip-flopját kimeneti kivezetéssel látjuk el, a léptetőregiszter párhuzamos kiolvasásra is alkalmassá válik.

A flip-flopok statikus beíró és törlő bemeneteit is felhasználva lehetővé válik, hogy a léptetőregiszter párhuzamos beírásra is alkalmassá váljon. A *4.60. ábra* egy 4-bites soros/párhuzamos kiolvasású és beírású léptetőregiszter kapcsolási rajzát szemlélteti. A párhuzamos beírás és a léptetés funkcióját az ÉS-VAGY elemek választják szét. A beírás előtt szükséges törlés az *CLR* (törlés) bemeneten keresztül történik logikai **0** szinttel. Beírni az az *S* bemenetről logikai **0** szinttel lehet.



4.60. ábra. 4-bites soros/párhuzamos kiolvasású és beírású léptetőregiszter
MC (Mode Control) – üzemmód vezérlő
CLR (Clear) – törlés
S (Set) – beírás
CLK (Clock) – órajel

Ha az üzemmódvezérlő (angolul: ***MC***– *Mode Control*) bemenetén logikai **0** szint van, a kettős *ÉS* kapucsoportok kapui közül azok vannak engedélyezve, amelyek a párhuzamos bemenetről (D_0, .., D_3) veszik az információt. Ha a vezérlőbemenet logikai **1**-es szinten van, az *ÉS* kapucsoportok kapui közül azok vannak engedélyezve, amelyek a szomszédos flip-flop kimenetére csatlakoznak. Így az $MC = \mathbf{0}$ párhuzamos beírás, az $MC = \mathbf{1}$ léptetés, illetve a soros bemenet felhasználásával soros beírás üzemmódot biztosít. Természetesen bármelyik működés csak órajel hatására jön létre.

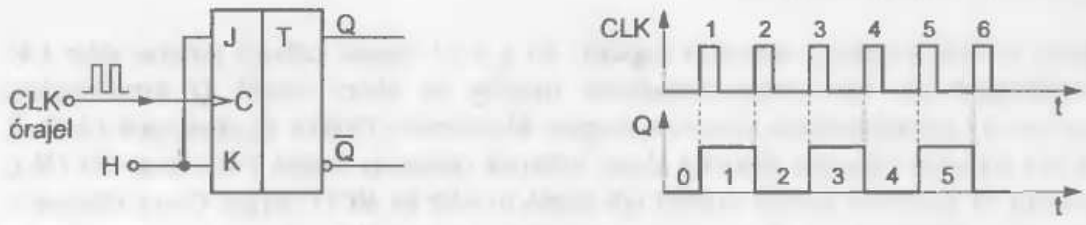
4.5. Számláló áramkörök

A szekvenciális hálózatok egyik igen fontos csoportjának tekinthetők a számláló áramkörök. A számláló áramkörök feladata, hogy rögzítsék és jelezzék (számlálják) a bemenetükre jutó impulzusok számát. Elvileg számláló áramkör lehet minden olyan áramkör, amelynél – bizonyos korlátokon belül – a beérkezett impulzusok száma és a kimeneti változók állapota között egyértelmű kapcsolat van.

A számláló áramköröknek tárolniuk kell a beérkezett impulzusok számlálásának az eredményét és ezt újabb impulzus hatására megfelelő módon meg kell változtatniuk. Egy számláló áramkörnek legalább annyi egymástól különböző állapottal kell rendelkeznie, amennyi a számlálandó impulzusok számának a maximuma.

A számláló áramkörök legtipikusabb alapáramkörének a *T* flip-flop tekinthető. A J-K flip-flopok alkalmazása alapáramkörként nagyobb lehetőséget nyújt különleges logikai függvénykapcsolatok megvalósítására. Minden bit tárolására egy flip-flop szükséges. Egy négybites számláló (amely tehát 4 flip-floppal rendelkezik) maximálisan $N = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 15$ -ig számlálhat.

A *4.61. ábra* egy *T* flip-floppal kivitelezett 1-bites számláló kapcsolását és a számlálás idődiagramját szemlélteti. Megfigyelhető, hogy a *T* flip-flop minden órajelre ellenkező állapotba billen át. Mivel csupán két megkülönböztethető állapota van, *csak két órajelimpulzus* egyértelmű leszámlálására alkalmas. Kettőnél több impulzus leszámolásához természetesen kettőnél több különböző állapot szükséges, ami a számlálóban alkalmazott filp-flopok számának növelésével érhető el. Mivel minden kimeneti változó csak két értéket vehet fel, ezért *n* kimenet esetén 2^n kombináció fordulhat elő (vagyis a számláló 2^n darab impulzus leszámlálására képes). Gyakran a lehetséges kombinációk csak egy részét használják fel.



a) kapcsolása *b) kimeneti állapotainak időábrája*

4.61. ábra. 1-bites számláló

A számláló áramkörök nagyon sok típusa ismeretes, ezért osztályozásuk igen sokféle szempont szerint történhet. A következőkben a rendszerezés kedvéért megemlítjük a legfontosabb szempontokat.

- A számlálót alkotó flip-flopok órajellel való vezérlése szempontjából megkülönböztetünk:
  - ***aszinkron számlálót:*** az órajelek a számlálónak általában csak az egyik, többnyire a legkisebb helyiértéket képviselő flip-flopját vezérlik. A többi flip-flop az órajelet egymástól kapja, így billenésük nem azonos időpontban történik (vagyis a működés aszinkron);
  - ***szinkron számlálót:*** az órajelek a számlálót alkotó összes flip-flopot egyszerre vezérlik. Ebben az esetben minden flip-flop billenése azonos időpontban történik (vagyis a működés szinkron).

♦ A számláló számábrázolása (a tárolt impulzusszám kódolása) szerint, megkülönböztetünk:

• ***bináris számlálót:*** a tárolt impulzusszám kódolása bináris számrendszerben történik;
• ***decimális számlálót:*** a tárolt impulzusszám kódolása (*általában 8421 kódú*) decimális számrendszerben történik.

♦ A számlálás iránya (sorrendje) szerint, megkülönböztetünk:

• ***előre-számlálót:*** a tárolt impulzusszám növekvő sorrendű (angol elnevezése: up counter);
• ***vissza-számlálót:*** a tárolt impulzusszám csökkenő sorrendű (angol elnevezése: down counter);
• ***reverzibilis-számlálót:*** a számlálás iránya megfordítható (angol elnevezése: up-down counter).

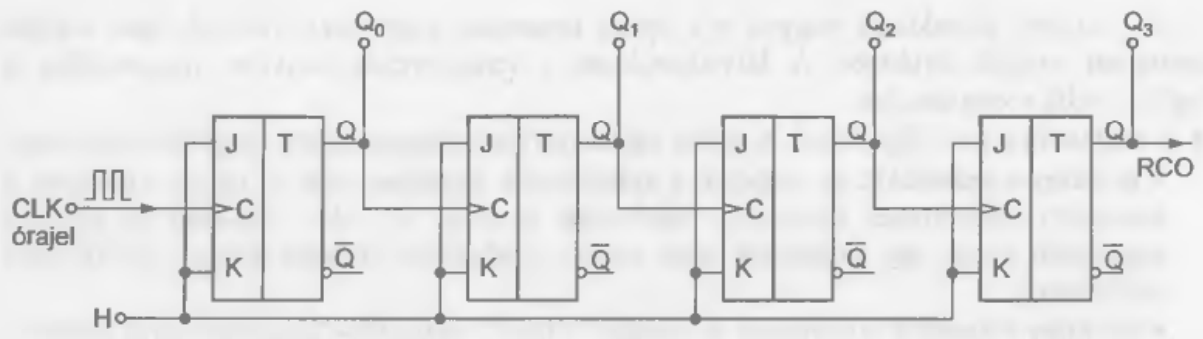
A csak csökkenő sorrendű számlálók áramköri megvalósítása (integrált áramköri formában) nem szokásos, általában a visszaszámlálás megvalósítására reverzibilis számlálókat alkalmaznak.

4.5.1. Bináris számláló áramkörök

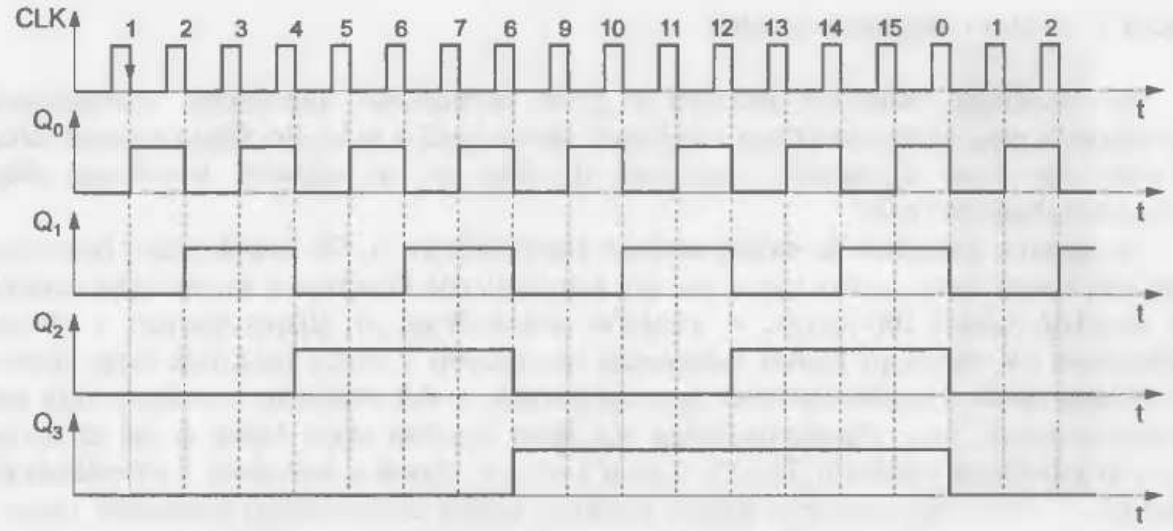
A számláló áramkörök legegyszerűbb felépítésű változatai a bináris számlálók. Jellegzetességük, hogy a tárolt impulzusszám kódolása bináris számrendszerben történik

4.5.1.1. Aszinkron bináris számlálók

Aszinkron bináris számláló áramkört kapunk, ha a *4.62. ábrán* látható módon több J-K tárolót sorbakötünk, és ezek órajel-bemenetét mindig az előző tároló *Q* kimenetéhez csatlakoztatjuk. Az előreszámlálás elérésére negatív élvezérlésű tárolót alkalmazunk (*J=K=1* feltétellel), így a tároló kimeneti állapota akkor változik mikor az órajel **1**-ről **0**-ra vált (*H-L* átmenet esetén). A számláló tetszés szerint bővíthető tovább az ***RCO*** (**R**iple **C**arry **O**utput – átvitelkimenet) segítségével (pl. 10 tárolóval már 1023-ig számlálhatunk).



4.62. ábra. 4-bites aszinkron bináris számláló



4.63. ábra. Aszinkron bináris számláló kimeneti állapotainak időábrája

N	Q_3	Q_2	Q_1	Q_0
	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16→0	0	0	0	0

4.8. táblázat. A kimenetek állapota az impulzusszám függvényében

A *4.63. ábrán* látható időábra a számlálandó impulzusok (órajel) és az alkalmazott négy flip-flop kimeneteinek állapotai közötti összefüggést szemlélteti. A *4.8. táblázat* a beérkezett impulzusok N száma és a Q_i kimeneti változók értékének összerendelését mutatja (természetesen ezek az összefüggések a *4.63. ábrán* látható idődiagramból is kiolvashatók). A táblázatot felülről lefelé vizsgálva két törvényszerűség ismerhető fel:

1. egy Q_i kimeneti változó akkor változtatja meg az értékét, ha az előtte álló kisebb helyértékű változó (Q_{i-1}) logikai **1**-ről **0**-ra vált;
2. egy Q_i kimeneti változó mindig akkor változtatja meg az értékét, ha minden kisebb helyértékű változó ($Q_{i-1},\ldots,Q_0$) logikai **1** értékű, és egy újabb impulzus érkezik.

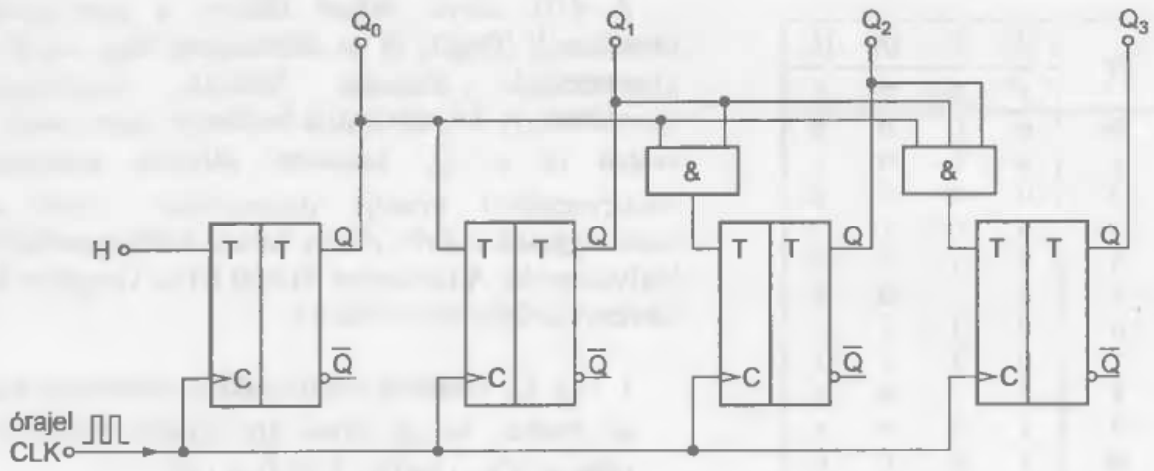
Vannak esetek mikor olyan számláló áramkörre van szükség, amelynél a számláló állását minden egyes számlálandó impulzus 1-gyel csökkenti (visszaszámláló). Egy ilyen visszaszámláló törvényszerűségei a *4.8. táblázatból* kiolvashatók, ha a sorokat alulról felfelé tekintjük át.

1. Egy Q_i kimeneti változó visszaszámlálásnál mindig akkor változtatja meg az értékét, ha a következő kisebb helyértékű változó (Q_{i-1}) logikai **0**-ról **1**-re vált.
2. Egy visszaszámláló esetében a Q_i kimeneti változó mindig akkor változtatja meg az értékét, ha az összes kisebb helyértékű változó ($Q_{i-1},\ldots,Q_0$) logikai **0** értékű, és egy újabb impulzus érkezik.

4.5.1.2. Szinkron bináris számlálók

Az aszinkron számlálók hátránya a gyors ismétlődésű impulzusok számlálásánál mutatkozik meg. Bizonyos szintet meghaladó sebességnél a számláló állapota hamis lehet. Ennek oka, hogy a nagyobb helyértékű flip-flop felé az előbbiek késleltetési ideje láncszerűen adódik össze.

A *szinkron számlálók* az előbbi hátrányt küszöbölik ki. A flip-flopok órajel bemenetei egymás között össze vannak kapcsolva, így a számlálandó impulzusok szinkronban vezérlik a számláló összes flip-flopját. A szinkron számlálóknál az állapotváltozást a tárolók (általában J-K flip-flop) logikai bemenetein alkalmazott vezérlés határozza meg. Bináris számlálás esetén a logikai feltételek igen egyszerűek. A *4.8. táblázatot* tanulmányozva már megállapítottuk, hogy előreszámláláskor egy adott flip-flop akkor billen át, ha az összes kisebb helyértéket képviselő flip-flop logikai **1**-et tárol. Ennek a feltételnek a teljesülését *ÉS* kapukkal lehet ellenőrizni. Egy 4-bites szinkron bináris előreszámláló kapcsolási rajzát a *4.64. ábra* mutatja. A számláló pozitív élvezérelt *T* flip-flopokból épül fel. Az órajel ebben az esetben csak az állapotváltozások ütemét határozza meg.

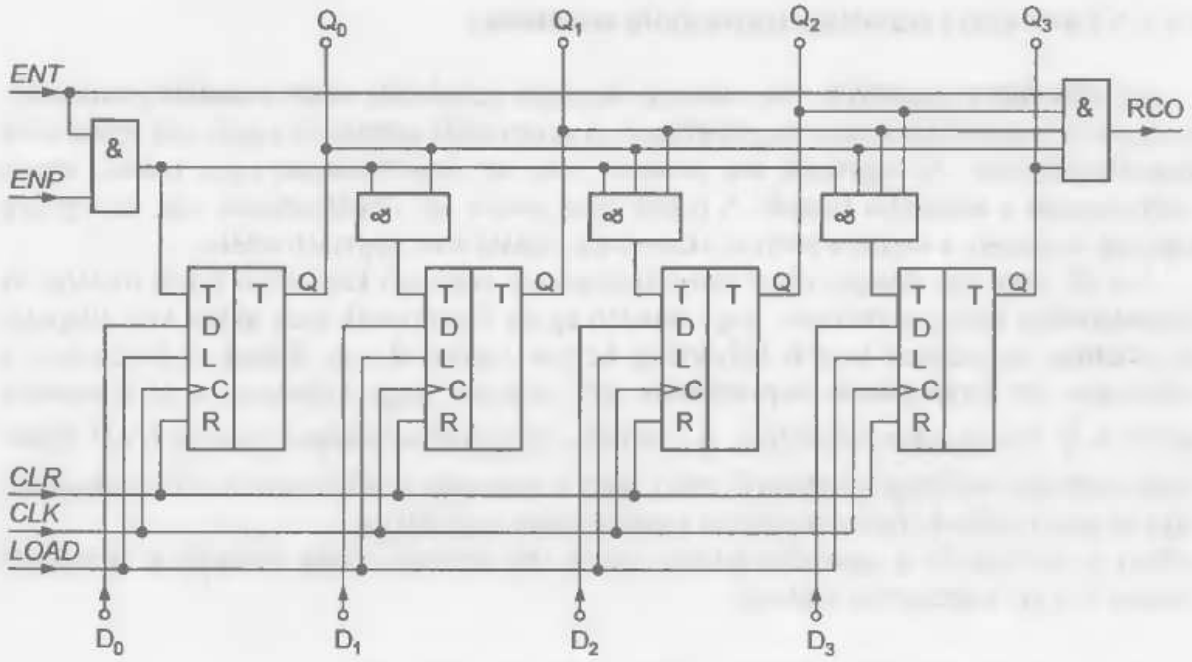


4.64. ábra. Szinkron bináris előreszámláló

Annak ellenére, hogy a bemeneti impulzusok egyidejűleg minden *C* óra jelbemenetre rákerülnek, mégsem változik minden órajel hatására az összes tároló állapota. A tárolók csak akkor billennek át, ha a *T* vezérlőváltozó értéke logikai **1**-es.

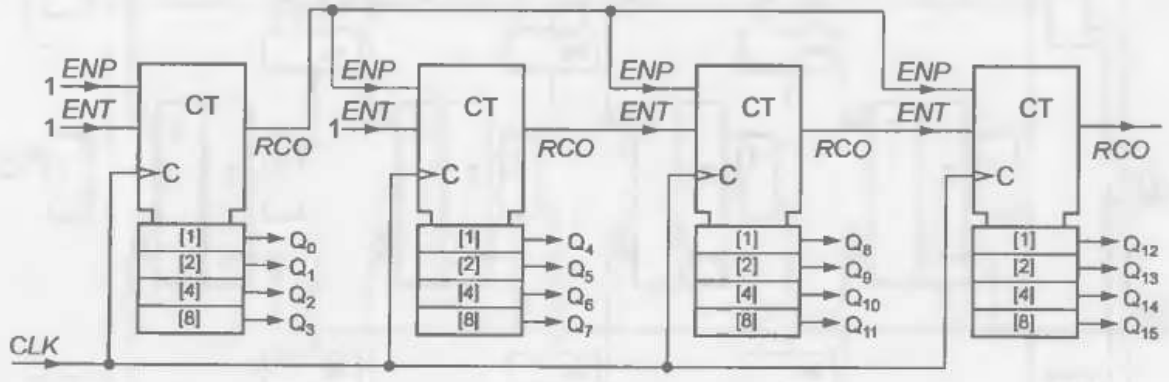
Az integrált szinkron számlálóknak még további be- és kimenetei vannak, amelyek szerepét és alkalmazását a *4.65. ábra* alapján megérthetjük.

- A ***CLR*** (Clear) törlő bemenet alkalmas az egész számlánc nullázására.
- A ***LOAD*** beíró bemeneten keresztül vezérelve a számlálóba tetszőleges *D* szám írható be.
- Az ***RCO*** (Riple Carry Output) átvitelkimenet segítségével többhelyértékű számlálót kapcsolhatunk össze. Az átvitelkimenetnek akkor kell logikai **1**-nek lennie, ha a számláló állása eléri az **1111**-et, és minden kisebb helyértékű tároló-egység ugyancsak átvitelt ad.
- Az ***ENT*** (**En**able **T**) *T* engedélyező bemenet.
- Az ***ENP*** (**En**able **P**) *P* engedélyező bemenet.



4.65. ábra. Integrált szinkron bináris előreszámláló kapcsolása

Többhelyértékű számlálót, pl. több 4-bites számláló kaszkád kapcsolásával alakíthatunk ki. A fokozatokat az *RCO* kimenetek és az *ENT* engedélyező bemenetek összekötésével kapcsolhatjuk össze. A sorbakapcsolt *ÉS* kapuk késleltetési ideje ebben az esetben összegeződik és emiatt csökkenne a maximális számlálási frekvencia. Ezért jobb megoldásnak tekinthető, ha a szükséges *ÉS*-függvénykapcsolatokat minden egyes számláló fokozatban párhuzamosan alakítjuk ki. E célból kihagyjuk a legkisebb helyértékű számláló fokozatot a soros *RCO-ENT* láncból és engedélyezzük párhuzamosan a nagyobb helyértékű számláló fokozatokat az ENP bemeneteken. Ily módon a *4.66. ábrán* látható elrendezéssel a párhuzamos *ÉS*-függvényeket külső logikai kapuk nélkül valósíthatjuk meg.



4.66. ábra. Szinkron bináris előreszámláló fokozatok kaszkád kapcsolása

CT (Counter) *– számláló*
ENT (Enable T) *– T engedélyező bemenet*
ENP (Enable P) *– P engedélyező bemenet*
CLK (Clock) *– órajel*
RCO (Riple Carry Output) *– átvitelkimenet*

4.5.1.3. Előre-hátra számlálók (reverzibilis számlálók)

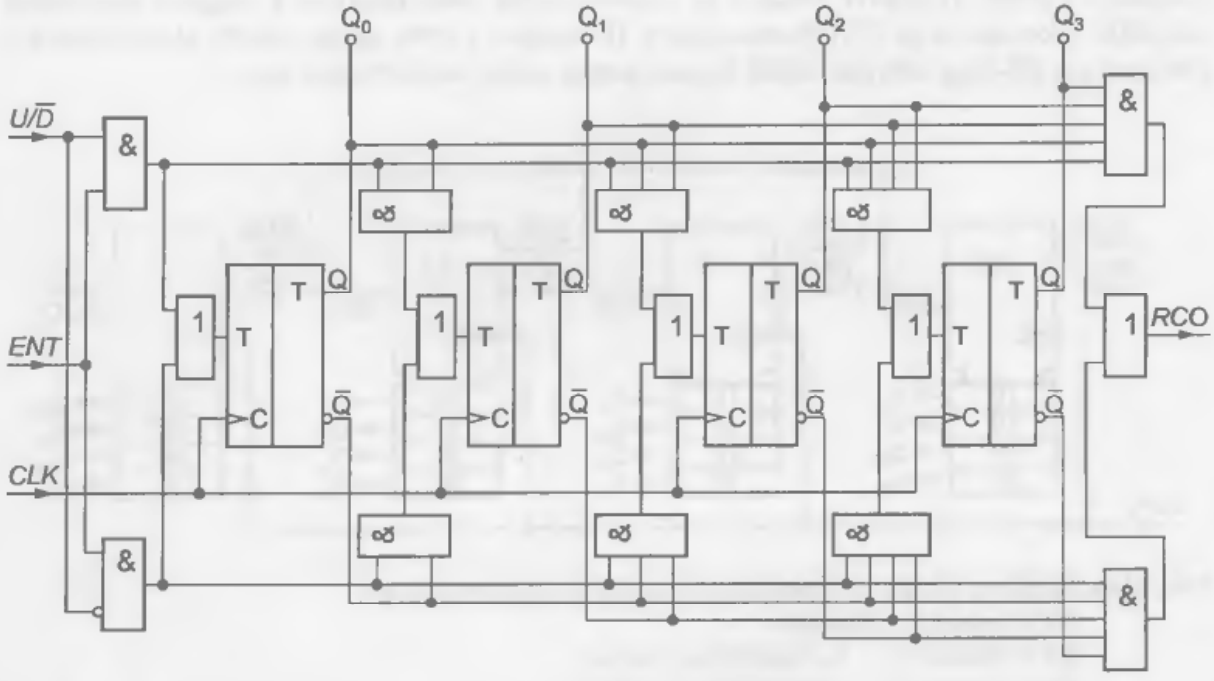
Az ***előre-hátra számlálók*** (más néven *reverzibilis számlálók*) olyan számláló áramkörök, amelyeknél a számlálás iránya megfordítható. A reverzibilis számlálók esetén két típust lehet megkülönböztetni. Az egyiknél két bemenet van, az órajelbemenet és a másik, amely meghatározza a számlálás irányát. A másik típus esetén két órajelbemenet van, az egyikre kapcsolt jel növeli, a másikra pedig csökkenti a számláló által képviselt értéket.

A *4.67. ábra* egy átkapcsolható számlálási irányú számláló kapcsolási rajzát mutatja. A hátraszámlálás billenési feltétele, hogy minden egyes flip-flopnak csak akkor kell állapotot változtatnia, ha minden kisebb helyértékű helyen logikai **0** van. Ennek a feltételnek a dekódolása az ábrán látható kapcsolásban úgy történik, hogy kapuzásra a Q kimenetek helyett a $\overline{Q}$ kimeneteket használjuk. A számlálás irányának az átkapcsolása az $U/\overline{D}$ előre-hátra vezérlőjel segítségével történik, mely vagy a kapcsolás felső részének (*előreszámlálás*) vagy az alsó részének (*hátraszámlálás*) kapuhálózatát engedélyezi.
Abban az esetben ha a számlálót tovább szeretnénk bővíteni, akkor átvitelre a következő fokozat felé két esetben van szükség:

1. ha a számláló állása: **1111** ($U/\overline{D}=1$, előreszámlálásnál).
2. ha a számláló állása: **0000** ($U/\overline{D}=0$, hátraszámlálásnál).

Az átviteli változóra a következő logikai függvény adódik:

$$RCO=\left[Q_0\cdot Q_1\cdot Q_2\cdot Q_3\cdot U/\overline{D}+\overline{Q}_0\cdot\overline{Q}_1\cdot\overline{Q}_2\cdot\overline{Q}_3\cdot\overline{U/\overline{D}}\right]\cdot ENT$$



4.67. ábra. Átkapcsolható számlálási irányú bináris számláló

4.5.2. Decimális számláló áramkörök

A bináris számlálók esetén megfigyelhető, hogy a három helyértékű változat 7-ig számlálhat, a négy helyértékű számláló pedig 15-ig. A természetes *BCD* ábrázolású számjegyeket négy helyértékű bináris szám képviseli, ahol a számjegyek súlya 2^3, 2^2, 2^1, 2^0, ezért a BCD ábrázolást 8421 kódnak is nevezik. Ennek megfelelően egy 8421 kódú *BCD* számláló megvalósításához négy bináris helyérték szükséges.

N	Q_3	Q_2	Q_1	Q_0
	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10→0	0	0	0	0

4.9. táblázat. BCD kód állapotdiagramja

A *4.9. táblázat* a 8421 típusú BCD kód állapotdiagramját szemlélteti (N a bemenő impulzusok számát képviseli). A táblázat a **9.** számjegyig megegyezik a *4.8. táblázat* szerinti bináris számlálókra vonatkozó táblázattal, a **10.** számot viszont ismét a **0000** érték képviseli.

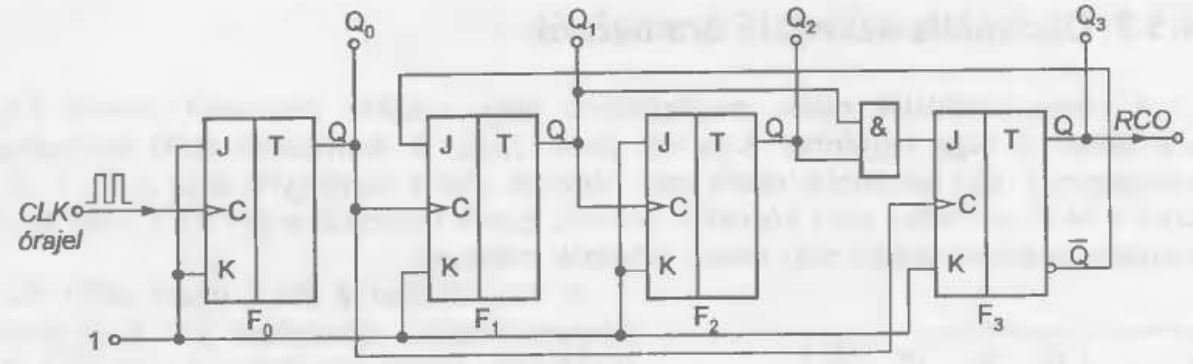
A decimális számláló csak abban különbözik a bináris számlálótól, hogy a tizedik impulzus hatására nullázódik, és átvitelt képez. A kapott átvitellel a következő, eggyel nagyobb helyértékű decimális számláló áramkört vezérelhetünk. Ahhoz, hogy a számlálót a tizedik bemeneti impulzus kezdeti állapotába juttassa egy megfelelő kiegészítő logika is szükséges, amelyet kombinációs hálózattal könnyen meg lehet valósítani.

4.5.2.1. Aszinkron decimális számláló áramkörök

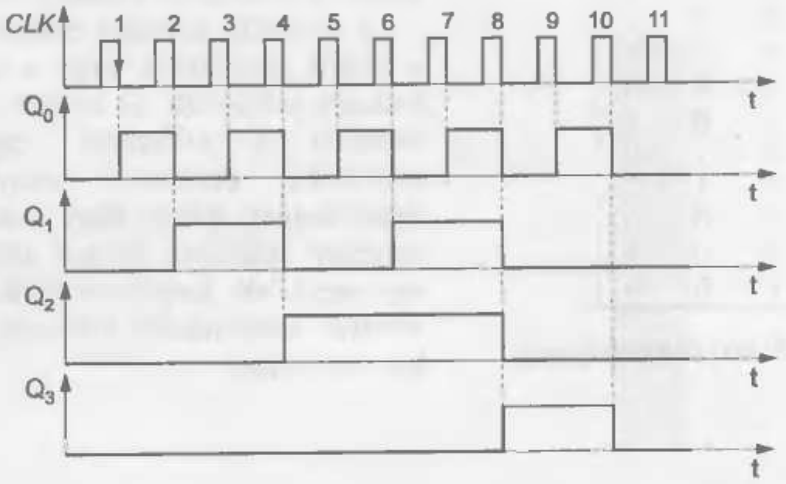
Decimális számláló áramkörök megvalósítására célszerű *J-K* flip-flopokat alkalmazni, mivel szélesebb körű vezérlési lehetőségeket biztosítanak. Aszinkron decimális számlálót kapunk a *4.60. ábrán* látható bináris változat kapcsolásának módosításával, ha biztosítjuk a következő feltételek teljesülését:

- a Q_1 kimenet maradjon logikai **0** akkor, ha a Q_3 előzőleg már logikai **1** szintű lett (vagyis a Q_1 kimenetű tárolónak nem szabad átbillennie a tizedik impulzus hatására);
- a Q_3 kimenetnek a tizedik impulzus hatására logikai **1**-ről **0**-ra kell átváltania.

Megoldás: – a második F_1 jelű flip-flop ***J*** bemenete nem állandó logikai 1-es szinten van, hanem a negyedik F_3 jelű flip-flop negált kimenetéről kapja a logikai vezérlőjelét *(4.68. ábra)*. Ennek következtében az F_1 jelű flip-flop *J* bemenete a számláló 0, ..., 7 állapotaiban logikai **1**-es szinten van, a ***nyolcadik*** impulzus hatására (Q_3 = **1**) logikai **0** szintre vált és az első három tároló törlődik. A következő ***kilencedik*** impulzus csak az első F_0 jelű tárolót billenti át, míg a ***tizedik*** impulzus a bemeneti logikai feltételek miatt csak az első flip-flopot billenti Q_0 = **1** logikai állapotba (a Q_1 kimenet logikai **0**-ban marad). Mivel a Q_1 kimeneten nem volt állapotváltozás, a harmadik flip-flop nem kap órajelet (Q_2 = **0**) és a negyedik flip-flop pedig törlődik (Q_3 = **0**). Következésképpen a számláló a tizedik órajel hatására alapállapotba (**0000** logikai állapotba) kerül.



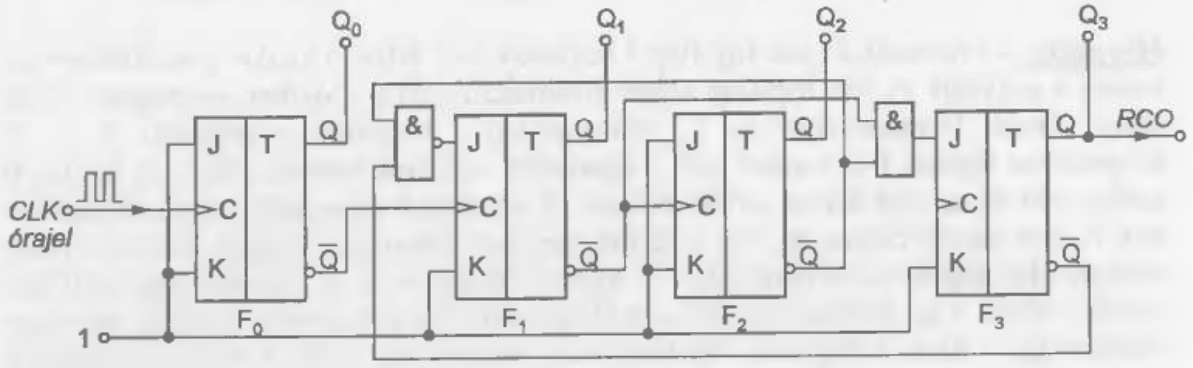
4.68. ábra. Aszinkron decimális számláló



4.69. ábra. Aszinkron decimális számláló kimeneteinek idődiagramja

Az aszinkron decimális előreszámláló kapcsolás kimeneti állapotainak időábráját a *6.69. ábra* szemlélteti.

Aszinkron decimális visszaszámláló áramkör kapcsolását mutatja a *4.70. ábra*. Az eltérés a *4.68. ábrán* látható változattól abban nyilvánul meg, hogy ebben az esetben – a visszaszámlálás követelményeinek megfelelően – a flip-flopok vezérlése a negált kimenetekről történik. A *NAND* kapuval előállított $J = 0$ logikai szint megakadályozza az F_1 jelű flip-flop átbillenését a 0→9 átmenetnél.



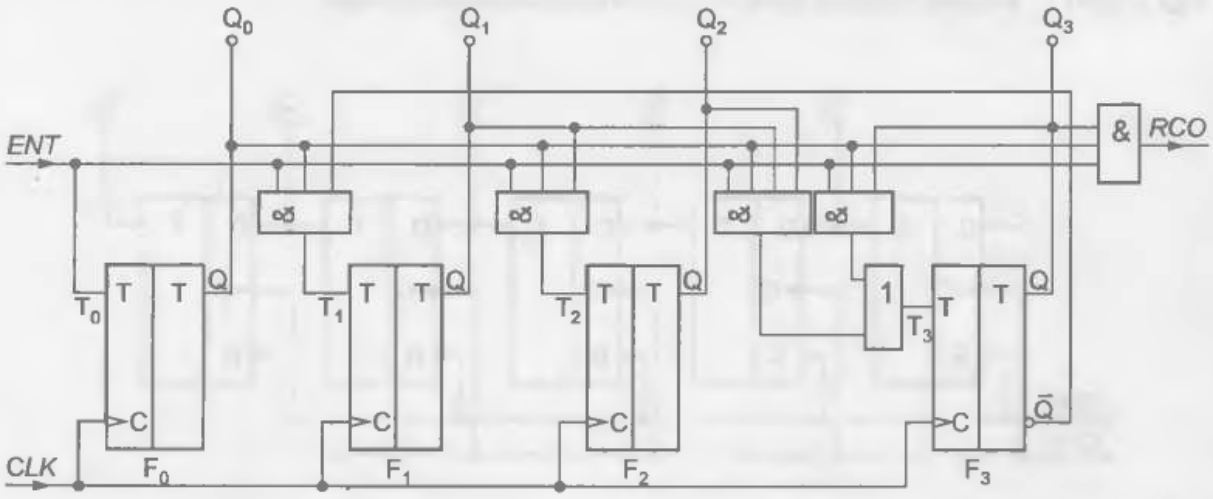
4.70. ábra. Aszinkron decimális visszaszámláló

4.5.2.2. Szinkron decimális számláló áramkörök

A *4.71. ábrán* látható szinkron decimális előreszámláló áramkör a *4.64. ábrán* látható bináris változat megfelelője a következő kiegészítésekkel:

- az F_3 jelű tároló negált kimenete vezérli az F_1 jelű tároló T bemenetét (a megfelelő logikai hálózaton keresztül), amely a 9→0 átmenetnél megakadályozza, hogy az F_1 jelű tároló átváltson;
- az F_3 jelű tároló visszaállítását a T bemenetén keresztül a **9** dekódolása segítségével oldjuk meg.

A *4.10 táblázat* az egyes flip-flopok logikai vezérlőjeleinek alakulását szemlélteti, amelynek segítségével nyomon követhetjük a számláló működését. A táblázatból megfigyelhető, hogy a **9**-es állapot felvétele után a számláló a következő órajelre a **0**-s állapotba kerül (tehát a számláló decimális típusú).



4.71. ábra. Szinkron decimális előreszámláló

		k							*(k+1)*				
Decimális érték	*ENT T engedélyező*	Q_3 Q_2 Q_1 Q_0 2^3 2^2 2^1 2^0				T_0	T_1	T_2	T_3	Q_3 Q_2 Q_1 Q_0 2^3 2^2 2^1 2^0				*RCO*
0	1	0	0	0	0	1	0	0	0	0	0	0	1	0
1	1	0	0	0	1	1	1	0	0	0	0	1	0	0
2	1	0	0	1	0	1	0	0	0	0	0	1	1	0
3	1	0	0	1	1	1	1	1	0	0	1	0	0	0
4	1	0	1	0	0	1	0	0	0	0	1	0	1	0
5	1	0	1	0	1	1	1	0	0	0	1	1	0	0
6	1	0	1	1	0	1	0	0	0	0	1	1	1	0
7	1	0	1	1	1	1	1	1	1	1	0	0	0	0
8	1	1	0	0	0	1	0	0	0	1	0	0	1	0
9→0	1	1	0	0	1	1	0	0	1	0	0	0	0	1

4.10. táblázat. A flip-flopok logikai vezérlőjeleinek alakulása

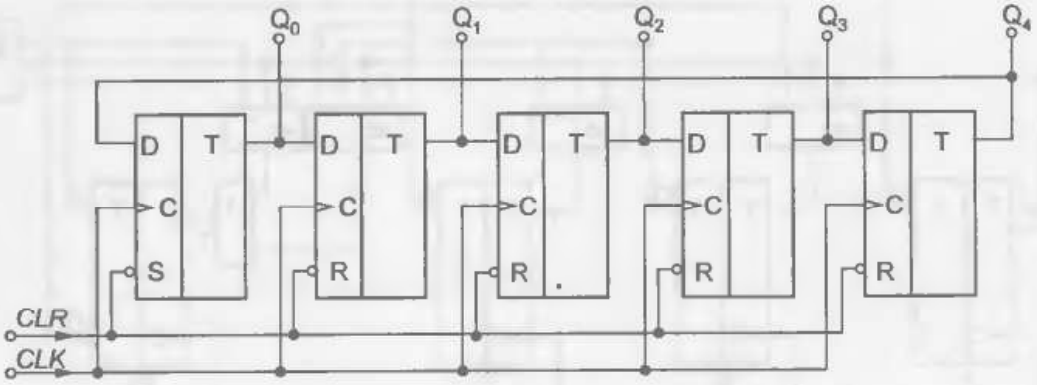
4.5.3. Gyűrűs számláló áramkörök

A gyűrűs számláló áramkörök felépítése és működése eltér az eddig ismertetett számlálókétól. A gyűrűs számlálók tulajdonképpen visszacsatolt léptetőregiszterek, amelyekben az állapotok számát a visszacsatolás módja határozza meg. Mivel a léptetőregiszterek flip-flopjai közös órajelet kapnak (vagyis egyszerre változtatnak állapotot), a gyűrűs számlálók működési sebessége a szinkron számlálókkal egyenértékű.

A gyűrűs számlálókat a kódolás szempontjából csoportosíthatjuk, a következőképpen:

- „*N*-ből 1" kódban működő számlálók;
- Johnson-számlálók;
- maximális hosszúságú számlálók.

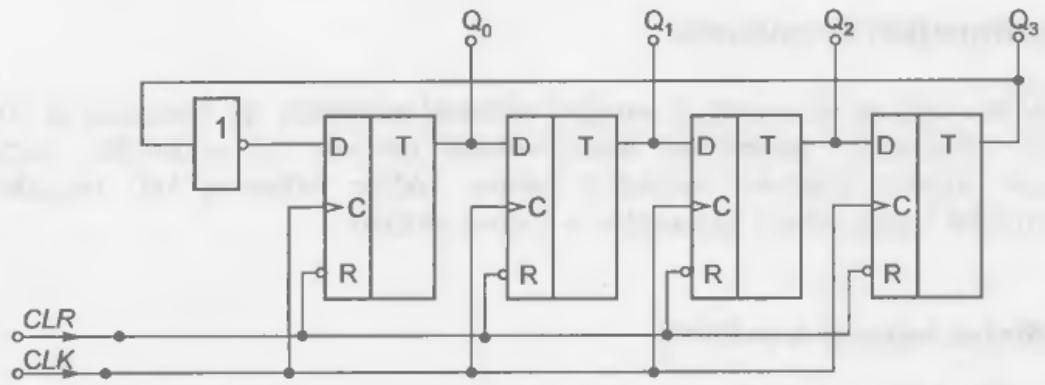
Az „*N*-ből 1" kódban működő számlálók információtartalma 1 bit kivételével azonos. A számláló állapotát az jelzi, hogy a többitől eltérő bit melyik flip-flopban van. A *4.72. ábra* egy „5-ből 1" kódban működő gyűrűs számláló kapcsolását mutatja.



4.72. ábra. *N*-bites léptetőregiszter
CLR (clear) – törlés (logikai 0-ban aktív)
CLK (clock) – órajel

A *CLR* bemenetre adott logikai **0** érték, nullázza a számlálót (vagyis $Q_0 = \mathbf{1}$ lesz, a többi kimenet pedig logikai **0** állapotba kerül). Ezután minden órajelre a regiszter tartalma egy helyértékkel jobbra lép. Az így kapott számláló áramkör előnye, hogy kombinációs hálózatot nem tartalmaz és ennek következtében működése gyorsabb, mint a kapuáramköröket is tartalmazó szinkron számlálóké. Ugyanakkor előnynek tekinthető, hogy a számlálás eredményét nem *BCD* kódban kapjuk meg, hanem a számlálás eredményét a logikai **1**-es szintű kimenet pozíciója közvetlenül jelzi. Hátránya az áramkörnek (ami miatt ritkán használják), hogy a megkülönböztethető állapotok száma (*modulusa*) azonos a flip-flopok számával, tehát nagy állapotszám esetén igen sok flip-flop kell a megvalósításhoz.

A Johnson-számlálókkal gazdaságosabban alakíthatók ki nagy modulusú számláló áramkörök. A *4.73.a ábra* egy 4-bites Johnson-számláló kapcsolását, a *4.73.b ábra* a számláló állapotdiagramját mutatja.



a) kapcsolás

b) állapotdiagram

4.73. ábra. *4*-bites Johnson-számláló
CLR (clear) – törlés
CLK (clock) – órajel

Az áramkör számlálási ciklusa 8 állapotot tartalmaz. Az állapotsorozat jellegzetessége, hogy a számláló balról jobbra logikai 1-ekkel, majd 0 bitekkel töltődik fel. *N* számú flip-flop esetén az **1** és **0** értékű bitekkel való feltöltődés *N-N* különböző állapotot eredményez.

Megállapítható, hogy *N* számú flip-flop (N-bites léptetőregiszter) esetén a különböző állapotok száma:

$$2 \cdot N .$$

A Johnson-számláló az „*N*-ből 1" kódhoz képest tehát kétszer több állapotot biztosít. Hátránya viszont az áramkörnek, hogy kijelzés céljaira dekóder áramkör alkalmazása szükséges.

4.6. Aritmetikai áramkörök

Azokat a digitális áramköröket, amelyek számtani műveletek (pl. összeadás és kivonás) végzésére alkalmasak ***aritmetikai áramköröknek*** nevezik. Az aritmetikai áramkörök bemenetét képező számokat megfelelő bináris kódban kifejezve kell megadni. Az eredményként kapott számok ugyanabban a kódban adódnak.

4.6.1. Bináris összeadó áramkörök

A bináris összeadás két egybites szám összeadására vezethető vissza. Az *A* és *B*, két egybites szám összeadását a következő szabály (*4.11. táblázat*) szerint lehet elvégezni:

				C	*(átvitel)*
A	+	*B*	=	*S*	*(összeg)*
0	+	0	=	0	0
0	+	1	=	0	1
1	+	0	=	0	1
1	+	1	=	1	0

4.11. táblázat. Két bináris szám összeadása

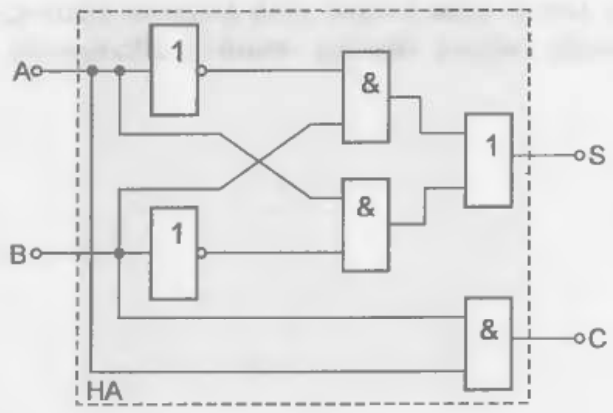
A	*B*	*C*	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

4.12. táblázat. Félösszeadó igazságtáblázata

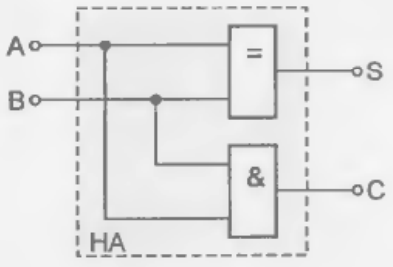
Ha a 0 számjegyet a bináris **0**, az 1 számjegyet a bináris **1** állapothoz rendeljük, akkor a *4.12. táblázat* szerinti igazságtáblázat adódik. Az ***S*** összeg és ***C*** átvitel kifejező logikai függvények segítségével is:

$$S = (A \cdot \overline{B}) + (\overline{A} \cdot B) = A \oplus B \qquad (4.1)$$
$$C = A \cdot B$$

A két logikai függvényt a *4.74. ábra* szerint megvalósító kapcsolást ***félösszeadónak*** nevezik. A félösszeadó (amely két bináris számot tud összeadni), a legegyszerűbb aritmetikai áramkörnek tekinthető.



a) alapkapukból felépített változat



b) KIZÁRÓ-VAGY kapus változat

4.74. ábra. Félösszeadó (jelölése– *HA*)

A félösszeadó önmagában nem alkalmas több bites számok helyértékenkénti összeadására, mert az átvitelt nem tudja figyelembe venni. Ez egy összeadási példából is látható (*4.13. táblázat*):

4.	3.	2.	1.	0.	*oszlop*
1	1	0	0		*átvitelbitek*
	1	1	1	0	} *összeadandó számok*
	1	1	0	0	
1	1	0	1	0	*eredmény*

4.13. táblázat.

A 0., 1. és 2. oszlop összeadását el lehet végezni félösszeadóval is, de a 3. oszlopét a keletkező átvitel miatt már nem.

Az átvitelt is figyelembe vevő összeadó az úgynevezett ***teljes összeadó***. Ennek működése a következő összefüggéseken alapszik (*4.14. táblázat*):

C_{i-1}	+	A_i	+	B_i	=	C_i	S_i
0	+	0	+	0	=	0	0
0	+	0	+	1	=	0	1
0	+	1	+	0	=	0	1
0	+	1	+	1	=	1	0
1	+	0	+	0	=	0	1
1	+	0	+	1	=	1	0
1	+	1	+	0	=	1	0
1	+	1	+	1	=	1	1

4.14. táblázat.

ahol A_i és B_i az i. oszlop összeadandó bitjei, C_{i-1} az $(i-1)$. oszlop összeadása után keletkező átvitel, S_i az összeg, C_i pedig a keletkező átvitel.

A teljes összeadót a következő logikai függvények írják le:

$$S_i = C_{i-1} \oplus P_i \tag{4.2}$$

$$C_i = C_i' + G_i$$

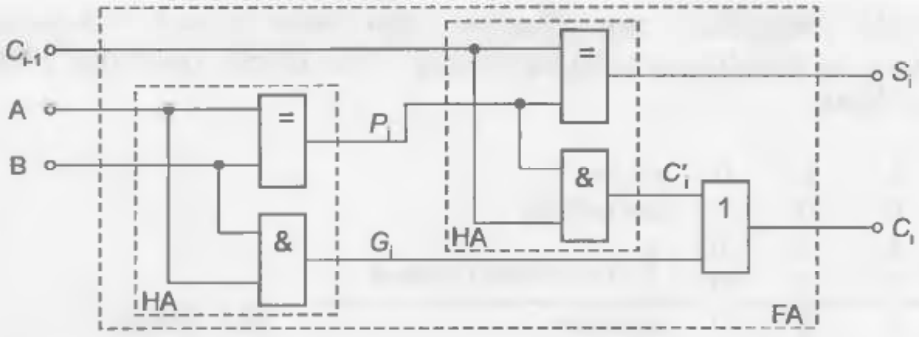
ahol

$S_i = C_{i-1} \oplus P_i$ *a terjedő (propagate) átvitel,*

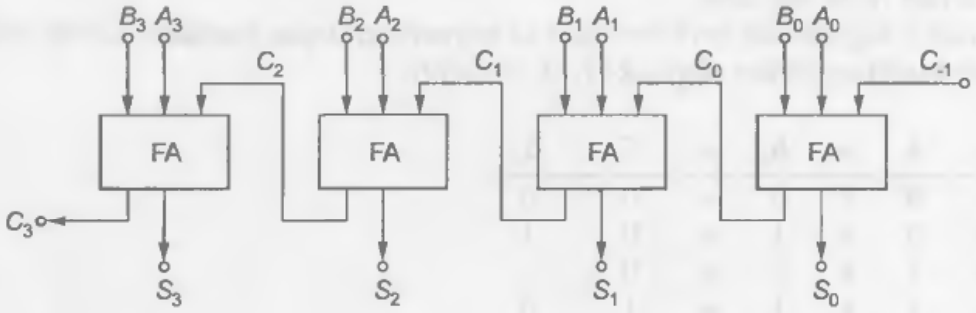
$G_i = A_i \cdot B_i$ *a keletkező (generate) átvitel.*

$C_i' = C_{i-1} \cdot P_i$

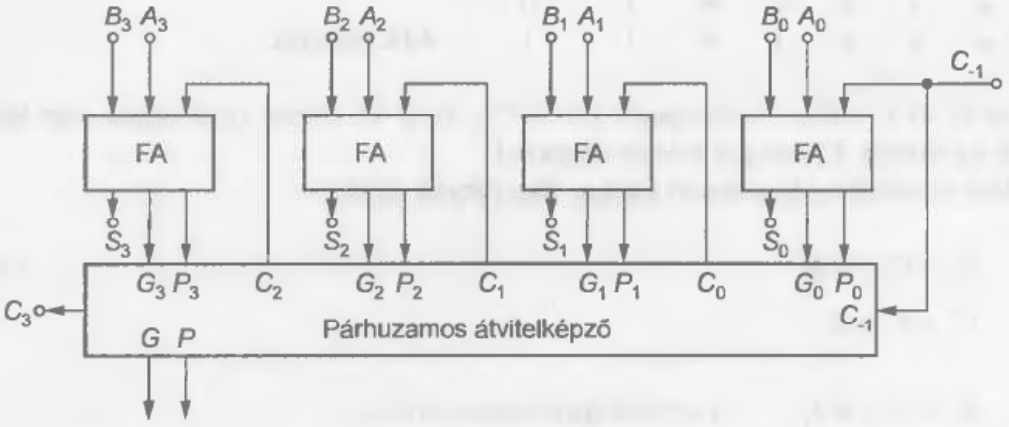
Egy teljes összeadó, amint a fenti összefüggések alapján is látható, két félösszeadóból és egy VAGY kapuból alakítható ki (*4.75. ábra*). A teljes összeadó a mikroprocesszor aritmetikai és logikai egységének alapvető építőeleme. A *4.76. ábra* egy 4-bites összeadó egység felépítését mutatja be. A több bites összeadó egység számolási ideje hosszabb mint, mint az ezt felépítő egybites teljes összeadóé. Ennek a magyarázata az, hogy a nagyobb helyértékű bitek összeadásánál várni kell az alacsonyabb helyértékű bitek átvitelképzésére. A késleltetési idők láncszerűen összeadódnak. A számlási idő lecsökkentését párhuzamos átvitelképző áramkör (angolul: *look k-ahead carry generator*) segítségével lehet elérni (*4.77. ábra*).



4.75. ábra. Teljes összeadó két félösszeadóból (jelölése *FA*)



4.76. ábra. 4-bites teljes összeadóegység



4.77. ábra. 4-bites összeadóegység párhuzamos átvitelképzéssel

Az átvitel általános kifejezése a teljes összeadót leíró függvényekből (4.2) származik:

$$C_i = G_i + P_i \cdot C_{i-1} \tag{4.3}$$

Ennek alapján a *4.76. ábrán* bemutatott négybites összeadó egység átvitelei:

$$C_0 = G_0 + P_0 \cdot C_{-1} \tag{4.4}$$
$$C_1 = G_1 + P_1 \cdot C_0 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_{-1}$$
$$C_2 = G_2 + P_2 \cdot C_1 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_1$$

$$C_3 = G_3 + P_3 \cdot C_2 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 +$$
$$+ P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_{-1} = G + P \cdot C_{-1}$$

ahol:

$$G = G_3 + P_3 \cdot G_2 + G_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 \quad \text{és}$$
$$P = P_3 \cdot P_2 \cdot P_1 \cdot P_0$$

A párhuzamos átvitelképző a fenti összefüggések szerint párhuzamosan állítja elő a C_0, C_1, C_2 és C_3 átviteleket, valamint a további lánckapcsoláshoz szükséges G és P jeleket.

4.6.2. Bináris kivonó áramkörök

Az 1-es és 2-es komplemens segítségével a kivonás összeadásra vezethető vissza. Két n-bites szám különbsége:

$$D = A - B \text{ , vagyis } D = A + (-B) \tag{4.5}$$

ahol $(-B)$ 1-es vagy 2-es komplemensű.

A (2.6) és (2.10) egyenlőségek alkalmazásával a fenti különbség kifejezése a következőképpen írható:

$$D^{(1)} = A^{(1)} + (-B)^{(1)} = A + (2^n + -1 - B) \tag{4.6}$$

az 1-es kiegészítős számábrázolás esetén, és

$$D^{(2)} = A^{(2)} + (-B)^{(2)} = A + (2^n - B) \tag{4.7}$$

2-es kiegészítős számábrázolás esetén.

A különbség kiszámítása 2-es komplemensű számábrázolásban sokkal célszerűbb, mint 1-es komplemensű számábrázolásban. A legnagyobb helyérték összeadása után keletkező átvitelt (2^n a (4.6) és (4.7) kifejezésekben) az n-bites eredmény nem veszi figyelembe. Ezért a 2-es komplemensű számábrázolás esetében a különbség valós értékét kapjuk meg (ugyancsak kettes komplemensben). Az 1-es komplemensű ábrázolás esetében ahhoz, hogy a különbség valós értékét kapjuk meg, az eredményhez még 1-et kell hozzáadnunk. Például számítsuk ki a $6_{10} - 3_{10}$ különbség eredményét először négybites 2-es komplemensben (*4.15. táblázat*) és azután négybites 1-es komplemensben (*4.16. táblázat*).

1	1	0	0	*(átvitel)*	
	0	1	1	0 *(kisebbítendő)*	$+6_{10}$
	1	1	0	1 *(kivonandó)*	-3_{10}
	0	0	1	1 *(eredmény)*	$+3_{10}$

4.15. táblázat. Különbség számítása négybites 2-es komplemensben

1	1	0	0	*(átvitel)*	
	0	1	1	0 *(kisebbítendő)*	$+6_{10}$
	1	1	0	0 *(kivonandó)*	-3_{10}
	0	0	1	0	$+2_{10}$
		+	1		$+1_{10}$
	0	0	1	1 *(eredmény)*	$+3_{10}$

4.16. táblázat. Különbség számítása négybites 1-es komplemensben

Az alábbi példák az ún. 2-es komplemensű aritmetika tulajdonságait szemléltetik. Végezzük el a következő összeadásokat és kivonásokat:

$+4_{10}+3_{10}$, $+4_{10}-3_{10}$, $+3_{10}-4_{10}$ és $-3_{10}-4_{10}$.

A két legnagyobb helyértékű bit összeadásánál keletkező átvitelt (C_{n-2} és C_{n-1}) feljegyeztük, ugyanis a továbbiakban ennek jelentősége van:

C_{n-1}	C_{n-2}				
0	**0**				
	0	1	0	0	$+4_{10}$
	0	0	1	1	$+3_{10}$
	0	1	1	1	$+7_{10}$

C_{n-1}	C_{n-2}				
1	**1**				
	0	1	0	0	$+4_{10}$
	1	1	0	1	-3_{10}
	0	0	0	1	$+1_{10}$

C_{n-1}	C_{n-1}				
0	**0**				
	0	0	1	1	$+3_{10}$
	1	1	0	0	-4_{10}
	1	1	1	1	-1_{10}

C_{n-2}	C_{n-2}				
1	**1**				
	1	1	0	1	-3_{10}
	1	1	0	0	-4_{10}
	1	0	0	1	-7_{10}

Végül végezzük el a következő műveleteket: $+3_{10}+6_{10}$ és $-3_{10}-6_{10}$. Az eredmény ábrázolására nem elég 4 bit. Ezekben az esetekben túlcsordulás keletkezik, és az eredmény csak akkor helyes, ha a legnagyobb helyértéknél keletkező C_{n-1} átvitelt is figyelembe vesszük:

C_{n-1}	C_{n-2}				
0	**1**				
	0	0	1	1	$+3_{10}$
	0	1	1	0	$+6_{10}$
0	1	0	0	1	$+9_{10}$

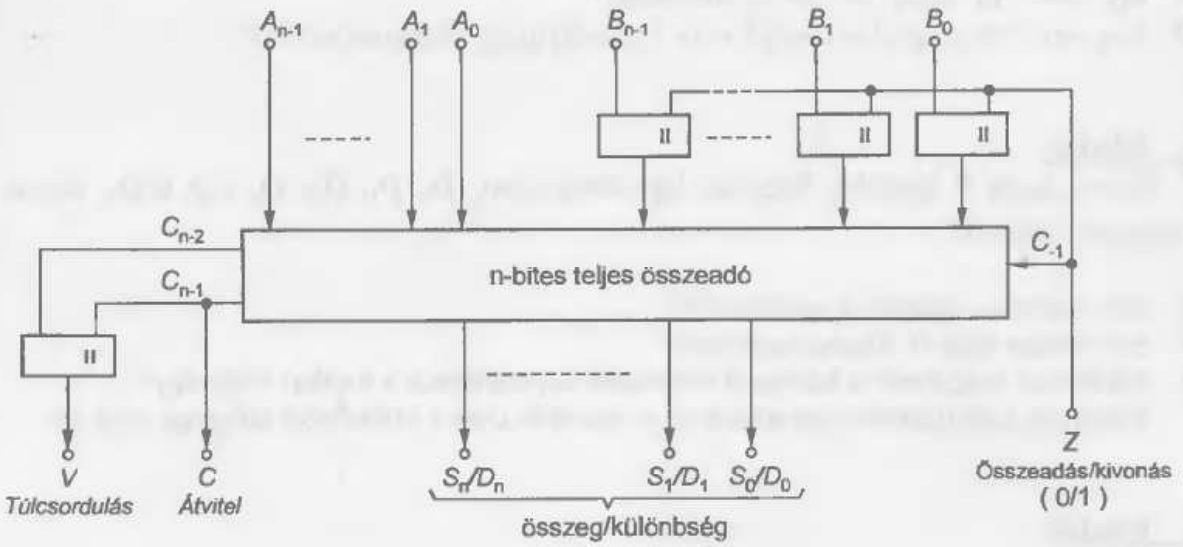
C_{n-1}	C_{n-2}				
1	**0**				
	1	1	0	1	-3_{10}
	1	0	1	0	-6_{10}
1	0	1	1	1	-9_{10}

Így az eredményt nem n-bites számként értelmezzük, hanem $(n+1)$-bitesként (ebben az esetben 5-bitesként). A *túlcsordulás* (angolul:*owerflow*) keletkezése a C_{n-2} és C_{n-1} átvitelek segítségével mutatható ki.

Abban az esetben, ha $C_{n-2} = C_{n-1}$, akkor nem keletkezik túlcsordulás. Viszont, ha $C_{n-2} \neq C_{n-1}$, akkor túlcsordulás keletkezik. Tehát a V túlcsordulást a következő *KIZÁRÓ-VAGY* függvény jelzi:

$$V = C_{n-2} \oplus C_{n-1} \tag{4.8}$$

A *4.78. ábra* egy n-bites összeadó/kivonó egységet mutat be. Az áramkör alapja egy n-bites teljes összeadó. Az A bináris szám $A_{n-2},...A_1$ és A_0 bitjei közvetlenül az összeadó egyik bemeneteire, míg a B szám $B_{n-1},...B_1$ és B_0 bitjei a *KIZÁRÓ-VAGY* kapukon keresztül az összeadó másik bemeneteire vannak csatolva. Ha egy *KIZÁRÓ-VAGY* kapu egyik bemenete logikai **0**, akkor a kimenete azonos a másik bemenetével; és abban az esetben, ha az egyik bemenete logikai **1**, akkor a kimenete a másik bemenetének komplemense. Így, ha az összeadás/kivonás vezérlő bemenet Z = **0**, akkor az egység összeadóként működik ($S = A + B$), egyébként, ha Z = **1**, akkor az egység kivonóként működik ($D = A - B$). Az utóbbi esetben a *KIZÁRÓ-VAGY* kapuk az 1-es komplemenst képezik, amely a $C_{-1} = 1$ átvitel hozzáadásával válik 2-es komplemensűvé.



4.78. ábra. n-bites kettes komplemensű összeadó-kivonó egység

Összefoglaló kérdések és feladatok:

1. Fogalmazza meg mi a különbség a multiplexer és a demultiplexer között!
2. Mit nevezünk analóg multiplexernek?
3. Mit nevezünk regiszternek és milyen regisztertípusok vannak?
4. Hogyan lehet megvalósítani egy párhuzamos beírású és kiolvasású léptetőregisztert?
5. Milyen szempontok szerint lehet csoportosítani a számláló áramköröket és milyen típusok vannak?
6. Milyen jellegzetességei vannak a bináris számláló áramköröknek?
7. Rajzolja fel egy szinkron bináris számláló kapcsolását és elemezze a működését! Szerkessze meg az állapotdiagramját!
8. Milyen lehetőségek vannak többhelyértékű (nagyobb mint 4) bináris számlálók realizálására?
9. Milyen tulajdonságokkal rendelkeznek a reverzibilis számlálók?
10. Miben különböznek a decimális számláló áramkörök a bináris változatoktól?
11. Mit nevezünk gyűrűs számláló áramköröknek és milyen változatai vannak?
12. Mire alkalmasak az aritmetikai áramkörök?
13. Rajzolja le egy félösszeadó áramkör kapcsolási rajzát és írja fel a kimeneti logikai függvényeket!
14. Mit nevezünk teljes összeadó áramkörnek?
15. Hogyan lehet megvalósítani a bináris kivonást digitális áramkörökkel?

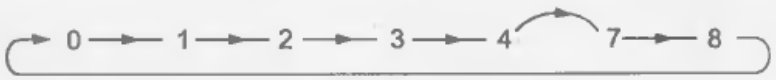
1. feladat:

Három darab D tárolóból felépített léptetőregisztert (D_0, D_1, D_2) $D_0 = Q_1 \oplus Q_2$ visszacsatolással látunk el.

a) Készítse el az áramkör logikai rajzát!
b) Szerkessze meg az állapotdiagramot!
c) Határozza meg annak a hálózatot kiegészítő kapcsolásnak a logikai függvényét, amelynek a hálózathoz kapcsolásával az áramkör csak 4 különböző állapotot vesz fel!

2. feladat:

Tervezzen olyan BCD kódú számlálót J-K flip-flopok felhasználásával, amelynek számlálási ciklusa a *4.79. ábrán* látható!



4.79. ábra. Számlálási ciklus

a) Írja fel a tervezéshez szükséges igazságtáblázatot!
b) Olvassa ki az állapotábrából a J-K bemenetek vezérlési függvényeit és rajzolja le a kapcsolást!

5. Logikai alapáramkörök

A digitális készülékek első látásra viszonylag bonyolultnak tűnnek, pedig néhány egyszerű logikai alapkapcsolás (alapáramkör) többszöri alkalmazásával megvalósíthatók. Az alapkapcsolásokból a feladat megoldásához szükséges rendszert tisztán formális módszerekkel alakíthatjuk ki a Boole-algebra segítségével. Az előző fejezetekben már logikai áramköröket használtunk anélkül, hogy ismertük volna a belső felépítésüket. Ez a tárgyalási mód azért indokolt, mert napjainkban a digitális elektronikában majdnem kizárólag integrált áramköröket használnak, amelyeknek a tápfeszültség-csatlakozási pontokon kívül csak bemeneti és kimeneti kivezetései vannak.

Egy-egy alapáramkör megvalósítására több áramkör-technikai megoldás létezik, amelyek teljesítményfelvételben, tápfeszültség-igényben, logikai szintekben, sebességben és kimeneti terhelhetőségben térnek el egymástól. Az áramkörtípusok helyes megválasztása érdekében ismernünk kell nagy vonalakban belső felépítésüket is.

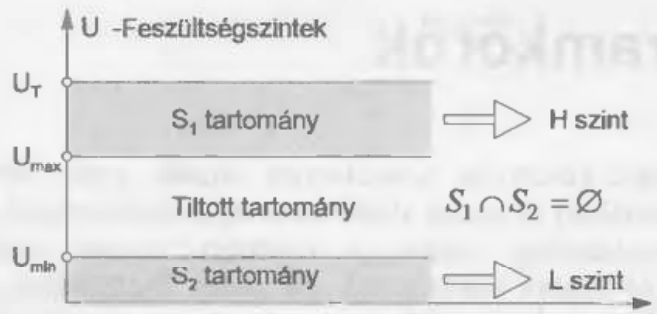
5.1. Logikai változók fizikai megjelenítése

A Boole-algebrában a logikai változókat – az eseményeket – rendszerint nagybetűkkel jelölik, és logikai jeleknek is nevezik. Logikailag két eset lehetséges: vagy bekövetkezik egy esemény, vagy nem. Ha bekövetkezik, akkor a logikai értéke igaz, és ezt az egyszerűség kedvéért „**1**"-gyel jelölik, ha pedig nem következik be, akkor a logikai értéke hamis, és ezt „**0**"-val jelölik. Az „**1**" és a „**0**" jelek a logikai algebrában nem számok, hanem célszerű szimbólumok. Az *5.1. ábrán* levő kétállapotú áramkör a logikai szimbólumok áramköri alkalmazását szemlélteti.

a) *kapcsoló : be*, *égő : világít* } *igaz* $\Rightarrow K = 1$ *b)* *kapcsoló : ki*, *égő : nem világít* } *hamis* $\Rightarrow K = 0$

5.1. ábra. A logikai szimbólumok áramköri alkalmazása

A digitális áramkörökben a logikai változók **0** és **1** értékeinek két különböző feszültségszint felel meg (*5.2. ábra*). Ezek a feszültségszintek természetesen csak bizonyos pontossággal tarthatók be, ezért célszerűbb feszültségtartományról beszélni. Pontos feszültségértékek meghatározása gyakorlati szempontból nem célszerű, mivel az ezt megvalósító áramkör igen bonyolult lehet. Azt a feszültségtartományt, amelyik U_{max} értéknél nagyobb abszolút értékű feszültségértékeket tartalmaz (S_1: $-U_{max}$ és U_T között) **H** (high: – magas) ***szintnek***, amelyik U_{min} értéknél kisebb abszolút értékű feszültségértékeket tartalmaz (S_2: – U_{min} és 0V között) **L** (low: – alacsony) ***szintnek*** nevezzük. Nagyon fontos feltételként jelentkezik, hogy a két feszültségtartomány diszjunkt legyen ($S_1 \cap S_2 = \varnothing$).



5.2. ábra. Logikai szimbólumok megfeleltetése feszültségszinteknek

A két tartományhoz (S_1, S_2) tartozó feszültségszintekre érvényesek a következő összefüggések: bármely $U_1 \in S_1$ *és* $U_2 \in S_2 \Rightarrow U_2 < U_1$

Az S_1 és S_2 tartomány közötti feszültségértékek logikailag nem értelmezettek ezért tiltott feszültségtartománynak tekinthetjük. Attól függően, hogy a **H** és **L** szintekhez milyen feszültségtartományt rendelünk, megkülönböztetünk:

- ***negatív feszültségű rendszert***: – mind a **H**, mind az **L** logikai szintekhez negatív feszültségtartományt rendelünk;
- ***pozitív feszültségű rendszert***: – mind a **H**, mind az **L** logikai szintekhez pozitív feszültségtartományt rendelünk.

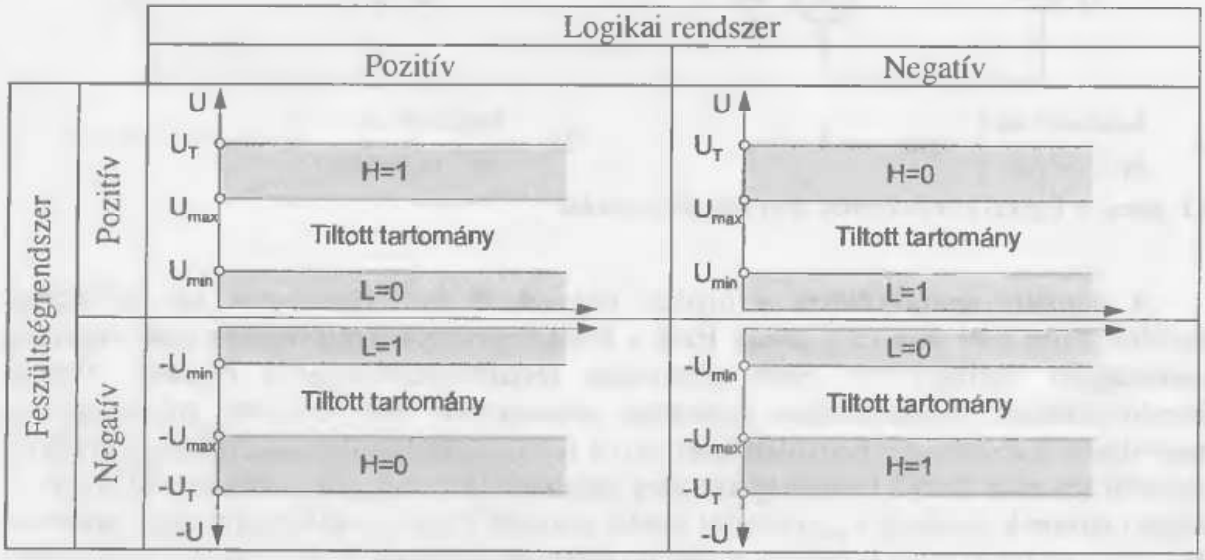
Egy adott logikai rendszeren belül a **H** és **L** szintekhez egyenként kétféle logikai értéket rendelhetünk:

$$\left.\begin{matrix} H=1 \\ L=0 \end{matrix}\right\} \Rightarrow \text{pozitív logika} \qquad \left.\begin{matrix} H=0 \\ L=1 \end{matrix}\right\} \Rightarrow \text{negatív logika}$$

Pozitív logika: a logikai **1** értékhez tartozó feszültségszint pozitívabb a logikai **0** értékhez tartozó feszültségszintnél.

Negatív logika: a logikai **1** értékhez tartozó feszültségszint negatívabb a logikai **0** értékhez tartozó feszültségszintnél.

Az *5. 3. ábra* a logikai rendszerek csoportosítását szemlélteti a fenn említett elvek alapján.



5.3. ábra. Logikai rendszerek

Vizsgáljuk meg egy kétbemenetű logikai kapu esetén a hozzárendeléseket. A logikai kapu pozitív feszültségrendszerben dolgozik és az *5.1.a* táblázatban látható igazságtáblázattal rendelkezik.

A	B	F^2
L	L	L
L	H	L
H	L	L
H	H	H

a)

A	B	F^2
0	0	0
0	1	0
1	0	0
1	1	1

b)

A	B	F^2
1	1	1
1	0	1
0	1	1
0	0	0

c)

5.1. táblázat.

- Pozitív logika esetén L=0, H=1 logikai értékeket rendelhetünk a feszültségszintekhez (*5. 1. b táblázat*). Megfigyelhető, hogy a logikai kapu *ÉS* kapcsolatot valósít meg.
- Negatív logika esetén L=1, H=0 logikai értékeket rendelhetünk a feszültségszintekhez (*5. 1. c táblázat*). Megfigyelhető, hogy a logikai kapu ebben az esetben *VAGY* kapcsolatot valósít meg.

A kapott eredmény a már előzőleg megismert *duál tételt* igazolja.

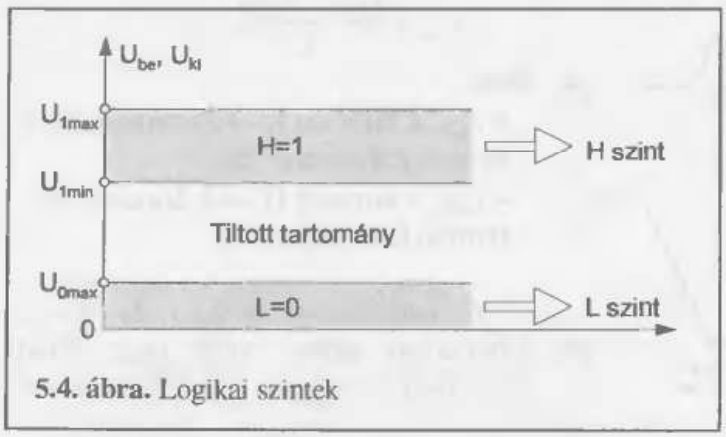
5.2. Logikai áramkörök jellemző adatai

A logikai függvények fizikai megvalósításához a gyártó cégek bizonyos alapáramkör választékot alakítanak ki. Az alapáramkörök széleskörű felhasználhatóságát, az áramkörök jellemző adatainak katalógusokban való közlése és egységesítése teszi lehetővé. A digitális technikában a pozitív logika terjedt el, és az áramkörök pozitív feszültségrendszerben dolgoznak. A következőkben pozitív feszültségű rendszerben működő, pozitív logikájú logikai áramkörök legfontosabb jellemző adataival foglalkozunk.

Tápfeszültség

A ***tápfeszültség*** az áramkör megfelelő működtetéséhez szükséges feszültség, a megengedhető tűrés feltüntetésével. Jele: U_T.

Logikai szintek



5.4. ábra. Logikai szintek

A logikai szintek azon feszültségtartományok, melyek a logikai **1**, illetve **0** értéket jellemző feszültségértékként még megengedhetők, mind a bemeneten, mind a kimeneten.
A logikai szinteket pozitív logika esetén az *5.4. ábra* szemlélteti.

Zajtartalék

A ***zajtartalék*** (angolul: *noise margin*), vagy ***zajérzéketlenség*** (angolul: *noise immunity*) az a feszültségtartomány, amelyen belül a feszültség változása nem változtatja meg a kapu logikai állapotát

Bemeneti terhelhetőség

A terhelhetőség fogalma alatt egy áramkör bemenetének és kimenetének áramviszonyait értjük.

A ***bemeneti terhelhetőség*** (*FAN IN*) alatt azt a maximális áramértéket értjük, amelyik az áramkör bemenetén átfolyhat mind logikai **1**, mind logikai **0** állapotban. A bemeneti terhelhetőség értékét a katalógusok egy dimenzió nélküli számmal adják meg, amely az ***egységterhelésnek*** felel meg.

*Az **egységterhelés** megadja, hogy az áramkör egy specifikusan megállapított érték hányszorosát képes úgy elviselni, hogy a logikai szintek a megadott határokon belül maradjanak.*

Kimeneti terhelhetőség

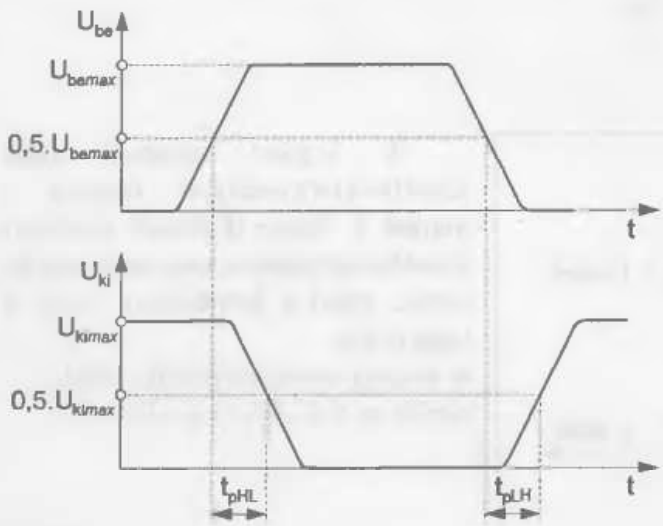
A ***kimeneti terhelhetőség*** (*FAN OUT*) azon bemenetek számát jelenti, amelyeket egy áramkör kimenete – károsodás nélkül – képes árammal ellátni (meghajtani). A kimeneti terhelhetőség értékét a katalógusok ugyancsak egy dimenzió nélküli számmal adják meg.

Teljesítményfelvétel

A katalógusok az áramkörök teljesítményigényét nem adják meg egységesen. Egyes katalógusok a P_d disszipációt (vagyis az 50 %-os kitöltésű impulzus által vezérelt hővé alakuló teljesítményt) adják meg, mások pedig a különböző logikai szintekhez tartozó tápáramigényt. Ez utóbbiból számítással határozható meg a tényleges teljesítményigény.

Jelkésleltetési idő

Egy logikai áramkör bemenetére adott jel és a hatására a kimeneten létrejövő jel megjelenéséhez időre van szükség. Ezt az időt a ***jelterjedési idővel*** (t_{pLH}, t_{pHL}), vagy az ***átlagos jelkésleltetési idővel*** (t_{pd}: – *propagation delay time*) jellemzik. Az *5.5. ábra* egy bemeneti impulzus hatására a kimeneten létrejövő impulzust szemléltet a létrejövő késleltetési időkkel együtt.



5.5. ábra. Jelkésleltetési idő

Az átlagos késleltetési idő a következő összefüggéssel számítható ki:

$$t_{pd} = \frac{t_{pHL} + t_{pLH}}{2};$$

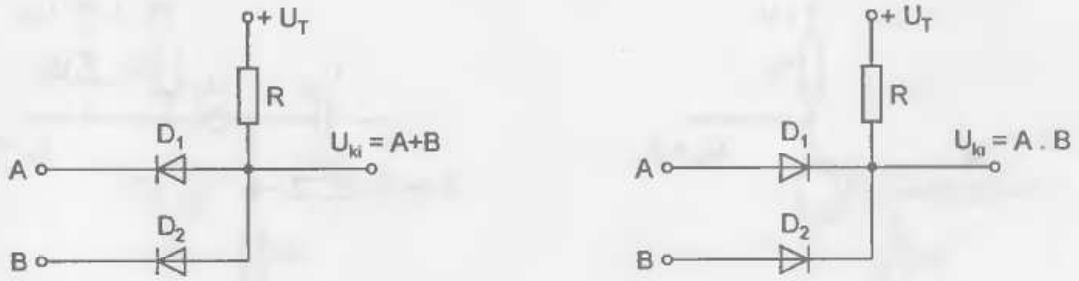
ahol:

- t_{pLH}, a kimenet L → H átmenetéhez tartozó jelterjedési idő.
- t_{pHL}, a kimenet H → L átmenetéhez tartozó jelterjedési idő.

A katalógusok a t_{pLH} és a t_{pHL} jelterjedési időket adják meg, ebből számítható az átlagos jelkésleltetési idő.

5.3. Diódás kapuáramkörök

Logikai áramköröket legegyszerűbben diódákkal és ellenállásokkal valósíthatunk meg. A diódás kapuáramkörök a logikai *ÉS* illetve *VAGY* kapcsolat megvalósítására használhatók. Az 5.6.a ábra egy pozitív logikában dolgozó *ÉS* kaput az *5.6.b. ábra* egy *VAGY* kaput mutat.



a) *b)*

5.6. ábra. Diódás kapuáramkörök

Vizsgáljuk meg az *5.6.a ábrán* látható *ÉS* kapu működését:

- Az *A* és *B* bemenetre kapcsolt **L** szint (L=0V) a D_1 és D_2 diódát nyitóirányban feszíti elő, így a diódák vezetni fognak és az áramkörben létrejövő áram a bemenetek felé folyik. Mivel a nyitóirányban előfeszített diódán kicsi feszültség esik, a bemenet és a kimenet közel ekvipotenciálissá válik, ami a kimeneten **L** szintet eredményez.
- Az *A* bemenetre **L** szintet a *B* bemenetre **H** szintet kapcsolva, a D_1 jelű dióda vezetni fog, a D_2 jelű dióda lezár. A létrejövő áram a vezető diódán a bemenet felé elfolyik, így a kimenet **L** szinten lesz. (Az *A* = **H** és *B* = **L** esetben az eredmény az előbbivel azonos).
- Mind a két bemenetre **H** szintet kapcsolva a D_1 és D_2 jelű diódák lezárnak, amely a kimeneten közel tápfeszültséget, tehát **H** szintet eredményez.

Az áramkör működésének igazságtáblázatát – L = 0 és H = 1 helyettesítésekkel – az *5.2 táblázat* szemlélteti.

A	B	F^2
L	L	L
L	H	L
H	L	L
H	H	H

A	B	F^2
0	0	0
0	1	0
1	0	0
1	1	1

5.2. táblázat.

A	B	F^2
L	L	L
L	H	H
H	L	H
H	H	H

A	B	F^2
0	0	0
0	1	1
1	0	1
1	1	1

5.3. táblázat.

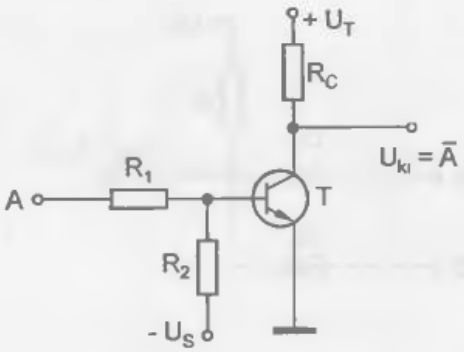
Hasonló gondolatmenetet alkalmazva az *5.6.b ábrán* látható áramkörre, belátható, hogy VAGY kapcsolatot valósít meg. Az áramkör működésének igazságtáblázatát az *5.3 táblázat* szemlélteti.

A diódás kapuárámkörök egyik hátránya, hogy a bemenetek számának növelésével a diódák záróirányú árama összeadódik, ami a kimeneti szint eltolódását eredményezi (vagyis téves logikai szintek jöhetnek létre). Egy másik hátrányként említhető, hogy a bemenő jel polaritását, fázisát nem lehet megfordítani, így *NEM* kapcsolat nem valósítható meg.

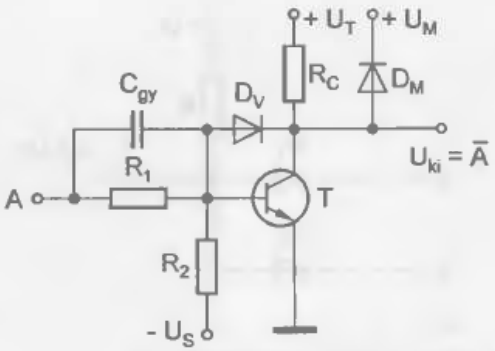
A digitális elektronikában a diódás kapuáramköröket az előnyösebb tulajdonságokkal rendelkező tranzisztoros kapuáramkörök szorították ki.

5.4. Inverterek

A negáció műveletet legegyszerűbben egy kapcsoló üzemmódban működő emitter-kapcsolású tranzisztorral valósíthatjuk meg (*5.7. ábra*). Ezt az áramkört ***inverternek*** nevezzük.



5.7. ábra. Egyszerű inverter kapcsolás



5.8. ábra. Javított inverter kapcsolás

A $-U_s$ segédfeszültség a tranzisztor stabil lezárását biztosítja (alapállapotban a tranzisztor bázisán az emitterhez képest negatív feszültség van). Vizsgáljuk meg az áramkör működését.

- Ha az *A* bemenet **L** szinten van (L=0V) a tranzisztor zárt állapotban van és kollektorán közel tápfeszültség (**H** szint) mérhető (*terheletlenül*).
- Ha az *A* bemenetre pozitív feszültség (**H** szint) kerül a tranzisztor vezetni fog. A vezető tranzisztor kollektor-emittere között csak az U_{CEsat} szaturációs feszültség mérhető (vagyis az **L** szintnek megfelelő állapot).

Eredményképpen megállapítható, hogy teljesül a NEM kapcsolat.

Az 5.7. ábrán látható kapcsolásnak több hátránya is van:

- az inverter kimeneti feszültsége függ a terhelő áram nagyságától, így a kimeneti feszültség viszonylag széles tartományban változhat. Ezért ezt az inverter-kapcsolást ***szabad szintű inverternek*** nevezik;
- az inverter üzemmódjából következően (teljesen zárt állapotból azonnal teljesen nyitott állapotba kell kerülnie) nagy a jelkésleltetési ideje.

Ezeket a hibákat küszöböli ki az 5.8. ábrán látható kapcsolás.

- A kimeneti feszültség terheléstől való függetlenségét egy D_M jelű ún. *„megfogó"* diódával küszöböljük ki, mely dióda a kimeneti feszültséget egy $U_{ki} = U_M + 0{,}6$ V szinten rögzíti. Ha a terhelő áram megnő, a dióda kinyit, így a kimenet feszültségét a terhelő áramtól függetlenül állandó értéken tartja. A D_M megfogó diódát tartalmazó invertert ***megfogott szintű inverternek*** nevezzük.
- A jelkésleltetési idő csökkentését a C_{gy} gyorsító kondenzátor, valamint a D_V telítésgátló dióda valósítja meg. A C_{gy} gyorsító kondenzátor a bemeneti négyszögjelet differenciálja, így a tranzisztor bázisára tűimpulzus kerül. A tranzisztor képes elviselni a rövid ideig tartó túlvezérlést, viszont a túlvezérléssel lehetőség nyílik a többségi töltéshordozók nagymennyiségű injektálására, ami által a tranzisztor gyorsan kerül vezetési állapotba. A túlvezérlésnek egy hátrányos következménye, hogy a tranzisztor telítésbe kerül, ami a tranzisztor lezárásánál jelent komoly időveszteséget.

  A telítési állapot elkerülésére egy ún. „telítésgátló" diódát (D_V) alkalmazunk, amely tulajdonképpen egy feszültségfüggő negatív visszacsatolást valósít meg. Ha a tranzisztor kollektorpotenciálja negatívabbá válna a bázis potenciáljánál, a D_V dióda

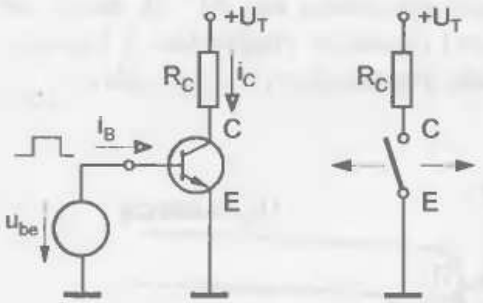
nyitóirányú előfeszítést kap és a felesleges bázisáram a diódán és a tranzisztor kollektorán keresztül a föld felé folyik. Ez meggátolja a tranzisztor telítésbe kerülését.

5.5. Logikai áramköri rendszerek

A diódás kapuáramkörök csak az *ÉS* és a *VAGY* kapcsolatot, az inverter csak a *NEM* kapcsolatot képes megvalósítani. Tetszőleges logikai függvénykapcsolatot viszont csak a funkcionálisan teljes rendszerek képesek megvalósítani. A következőkben olyan logikai áramkör-rendszerekkel foglalkozunk, amelyek kielégítik a funkcionálisan teljes rendszerek követelményeit.

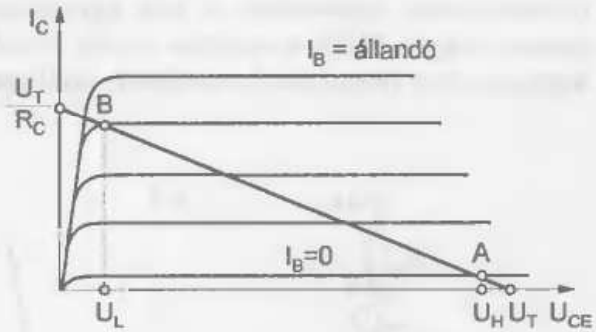
5.5.1. Bipoláris és MOS logikai integrált áramkörök

A logikai áramkörök alapvető építőeleme a tranzisztor. Bipoláris és unipoláris tranzisztorokat lehet megkülönböztetni.



a) elvi kapcsolás *b) helyettesítő kapcsolás*

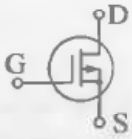
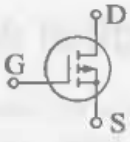
5.9. ábra. A bipoláris tranzisztor mint kapcsolóelem



5.10. ábra. Bipoláris tranzisztor kimeneti jelleggörbe-serege

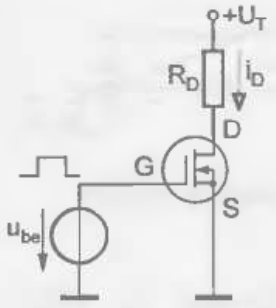
A bipoláris tranzisztor, amely működéséhez negatív (*elektronok*) és pozitív töltéshordozókra (*lyukak*) egyaránt szükség van, *NPN* vagy *PNP* típusú lehet. A bipoláris tranzisztor a logikai áramkörökben rendszerint közös emitterű alapkapcsolásban, mint kétállapotú kapcsolóelem működik (*5.9. ábra*). A tranzisztor telített és lezárt állapota megfelel egy bekapcsolt, illetve kinyitott kapcsolónak. Az R_C kollektor ellenállás munkaegyenese a tranzisztor kimeneti jelleggörbéiből kimetszi a telített, illetve a lezárt állapotoknak megfelelő *A* és *B* munkapontokat (*5.10. ábra*). A telített tranzisztor kollektorán egy kis U_L maradék feszültség lép fel. Emiatt az alacsonyabb logikai feszültségszint értéke, U_L sohasem lehet pontosan 0 V, hanem annál egy kissé pozitívabb. A lezárt tranzisztoron keresztül mindig folyik a kollektor-emmitter maradékáram. Ezért a nagyobb feszültségszint értéke U_H sohasem éri el az U_T tápfeszültség értékét, hanem a maradékáram által az R_C munka-ellenálláson létrehozott feszültségeséssel ennél kisebb.

Az unipoláris tranzisztor, amelynek működésében kizárólag egy polaritású töltéshordozó játszik szerepet, lehet MOSFET (**M**etal-**O**xid-**S**emiconductor **F**ield-**E**ffect-**T**ransistor) térvezérlésű tranzisztor, vagy JFET (**J**unction-**FET**) záróréteges térvezérlésű tranzisztor.
Az integrált logikai áramkörökben a MOS térvezérlésű tranzisztort alkalmazzák. A MOS tranzisztoroknak négy altípusa ismeretes: – a *P*- és *N*-csatornás tranzisztorok növekményes és kiürítéses üzemmódú változatai (*5.4. táblázat*).

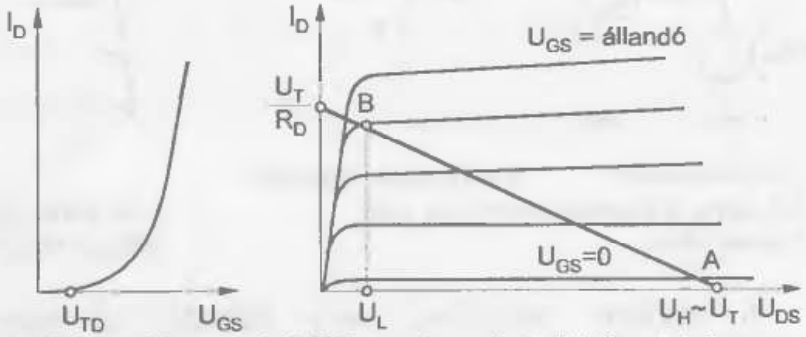
MOSFET TRANZISZTOROK TÍPUSAI			
Kiürítéses típusú MOSFET		*Növekményes típusú MOSFET*	
N csatornás	*P csatornás*	*N csatornás*	*P csatornás*
			

5.4. táblázat. MOS térvezérlésű tranzisztorok felosztása, áramköri jelölései

Az *S source* (forrás) és a *D drain* (nyelő) közötti vezetőcsatorna lehet *P*-, illetve *N*-típusú szilícium, attól függően, hogy a többségi töltéshordozók lyukak, vagy elektronok. A vezetőcsatorna keresztmetszete a *G gate* (vezérlőelektróda) és az *S source* közé kapcsolt U_{GS} feszültséggel befolyásolható. Ha nulla U_{GS} feszültségnél létezik vezetőcsatorna, akkor a tranzisztor *kiürítéses* (depletion) üzemmódú. Abban az esetben, ha a gate-elektródára adott U_{GS} feszültséggel előzetesen vezetőcsatornát kell létrehozni, akkor a tranzisztor *növekményes* (enhancement) üzemmódú. A *gate* egyenáramú bemeneti ellenállása kb. 10^{14} Ω, amely azt jelenti, hogy a MOS tranzisztor ideális feszültségvezérelt elemként viselkedik. A bipoláris tranzisztorhoz hasonlóan használható kétállapotú kapcsoló üzemmódban is (*5.11. ábra*).



5.11. ábra. A MOS tranzisztor mint kapcsolóelem

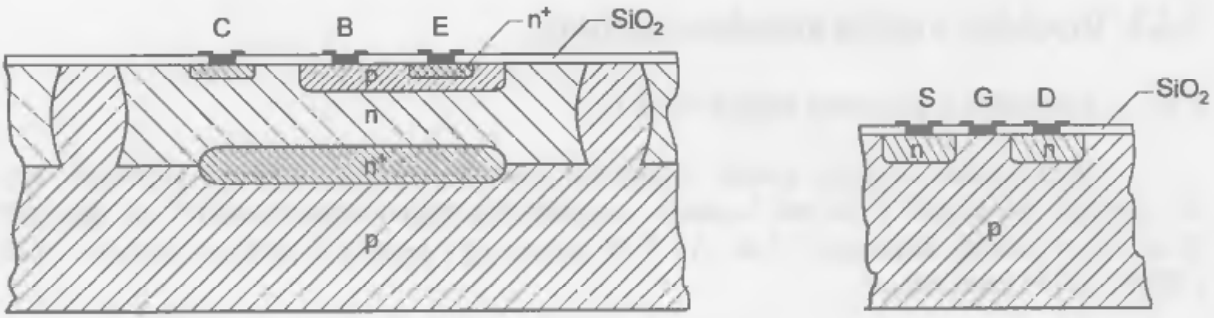


5.12. ábra. N-csatornás MOS tranzisztor átviteli jelleggörbéi

A kapcsolóelem szerepét betöltő tranzisztor működését a jelleggörbéi alapján lehet követni (*5.12. ábra*).

- Ha U_{GS} kisebb mint az U_{TO} küszöbfeszültség, akkor a MOS tranzisztor lezárt állapotban van; a *drain*-feszültség az U_T tápfeszültséggel egyenlő.
- Abban az esetben, ha U_{GS} egy adott mértékben nagyobb U_{TO}-nál, akkor a tranzisztor telítési tartományban vezet, és a *drain*-feszültsége majdnem nulla.

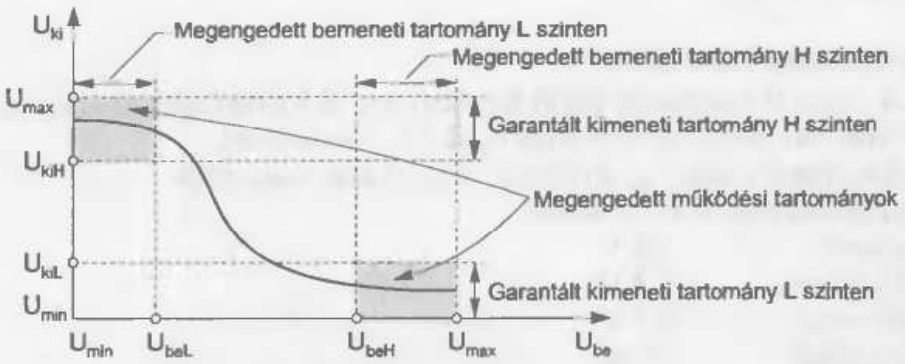
Egy bipoláris *NPN* tranzisztor és egy MOS *N*-csatornás növekményes térvezérlésű tranzisztor keresztmetszetét szemlélteti az *5.13 ábra*. A bipoláris tranzisztoros integrált áramköröket ***bipoláris integrált áramköröknek***, az unipoláris MOS térvezérlésű tranzisztoros integrált áramköröket ***MOS integrált áramköröknek*** nevezik.



a) Bipoláris NPN *b) MOS N-csatornás növekményes*

5.13. ábra. Integrált tranzisztor keresztmetszetek

Az integrált logikai áramköröknek a bemenetre és a kimenetre vonatkoztatva két feszültségtartománya van, amelyek megfelelnek a **0** és az **1** logikai állapotoknak. Bipoláris vagy MOS integrált áramköri kapuk jellegzetes átviteli jelleggörbéjét, valamint a két megengedett működési tartományát az *5.14. ábra* szemlélteti.



5.14. ábra. Az integrált áramkörös kapuk jellegzetes átviteli jelleggörbéje

A bemeneti **L** alacsony feszültségszint (pozitív logikában **0**-nak felel meg) U_{min}-től U_{beL}-ig terjed, és a bemeneti **H** magasabb feszültségszint (pozitív logikában **1**-nek felel meg) U_{beH}-tól U_{max}-ig terjed. A kimeneti feszültségszintet az egyedi átviteli jelleggörbe függvényében a következő fontosabb tényezők határozzák meg: a bemeneti feszültségszint, a kimenetet terhelő bemenetek száma és a környezeti hőmérséklet. Ezeknek figyelembevételével a kimeneti feszültség L szintnél garantáltan U_{min} és U_{kiL} között van, míg **H** szintnél U_{kiH} és U_{max} között. Egy logikai rendszert alkotó kapuk együttesének helyes működése feltételezi azt, hogy:

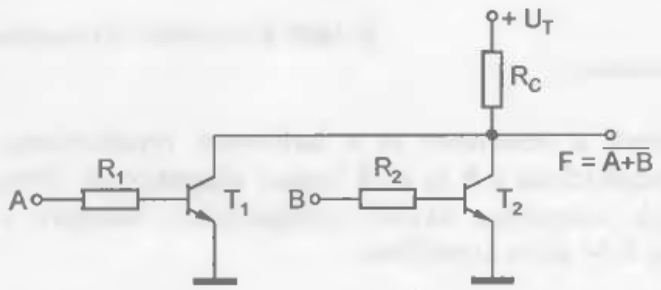
$$U_{kiL} < U_{beL} \quad \text{és} \quad U_{beH} < U_{kiH}$$

Az $U_{beL} - U_{kiL}$ és $U_{kiH} - U_{beH}$ különbségek az **L** szint, illetve a **H** szint zavartávolságát jelentik. A zavarok a logikai rendszerekben külső és belső zavarforrásokból, a jelek és a tápfeszültség vezetékei útján kerülhetnek be. Ezek rátevődnek a logikai jelszintekre és hibás működést eredményezhetnek. Minél nagyobb egy logikai áramkör-rendszer zavartávolsága, annál biztosabb a működése.

5.5.1. Bipoláris logikai áramkörcsaládok

5.5.1.1. Ellenállás-tranzisztor logika (RTL)

Az *RTL* logikai áramkör család a digitális technika kezdeti szakaszát képviseli; egy átmenetnek tekinthető a telített logikájú tranzisztoros billenőkapcsolásoktól az integrált technikával készült áramkörök felé. Az *5.15. ábrán* egy pozitív logikában dolgozó *NOR (NEM-VAGY)* kapu látható.



5.15. ábra. RTL NOR kapu

Az áramkör működése a következő:

- ha az *A*, vagy *B* bemenetek közül legalább egy **H** logikai szinten van, akkor a hozzá tartozó tranzisztor kinyit, és a kimenet logikai **L** szintre kerül.

Az *RTL* logikai áramköröket a gyakorlatban ma már nem használják.

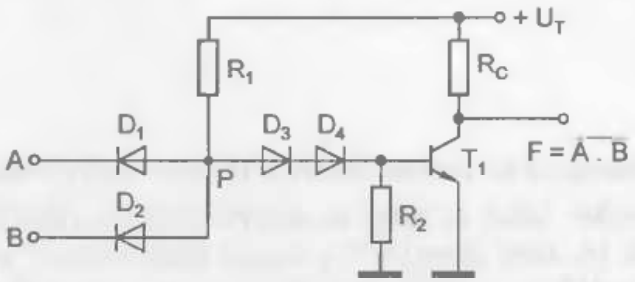
Az *RTL* kapu-áramkörök szokásos adatai:

Tápfeszültség:	3,6 V
H logikai szint:	1,6 V
L logikai szint:	0,2 V
Kapukésleltetés:	25 ns
Teljesítményfelvétel:	5 mW

5.5.1.2. Dióda-tranzisztor logika (DTL)

A *DTL* áramköri rendszer kétfokozatú. Az első fokozat egy ÉS, vagy egy VAGY kapcsolatot megvalósító diódás kapu. A második fokozat egy inverter, amely az első fokozat kimeneti jelét negálja, vagyis *NEM* kapcsolatot valósít meg.

Az *5.16. ábrán* egy *NAND (NEM-ÉS)* kapcsolatot megvalósító pozitív logikás *DTL* áramkör látható.



5.16. ábra. DTL NAND kapu

Elemezzük az *5.16. ábrán* látható *DTL* kapu működését.

- Ha az *A* vagy *B* bemenetek közül bármelyikre **L** logikai szint kerül, a kapcsolás *P* jelű pontján **L** szint van, ami a T_1 tranzisztort lezárja, így a kollektorán a tápfeszültséghez közeli feszültség (vagyis logikai **H** szint) jelenik meg.

- Ha mindkét bemenet feszültsége **H** szintű, mind a D_1, mind a D_2 dióda le van zárva és a kimeneti tranzisztor bázisárama az R_1 ellenálláson folyik. Ebben az esetben a T_1 tranzisztor kinyit és kimenete **L** szintű lesz.

A kapcsolásban a D_3 és D_4 jelű diódák az áramkör zavarérzéketlenségét növelik, mivel a T_1 tranzisztor csak akkor nyit, ha a *P* jelű ponton 1,2 V-nál nagyobb feszültség van (*feltételezzük, hogy a diódák nyitóirányú feszültsége 0,6 V*).
Az *DTL* kapu-áramkörök szokásos adatai:

Tápfeszültség:	5 V
H logikai szint:	3,5 V
L logikai szint:	0,3 V
Kapukésleltetés:	30 ns
Teljesítményfelvétel:	15 mW

5.5.1.3. TTL (Transistor-Transistor Logic) Tranzisztor-tranzisztor logika

A szabványos TTL áramkörcsalád együtt a *H* (High speed – Nagy sebességű TTL), *S* (Schottky-diódás TTL), *L* (Low power – alacsony fogyasztású TTL) és *LS* (Low power Schottky – Alacsony fogyasztású Schottky-diódás TTL) változatokkal néhány évvel ezelőtt a legelterjedtebb bipoláris integrált logikai áramköröknek számítottak. Leginkább kis és közepes bonyolultságú integrált áramkörök (*SSI* és *MSI*) formájában terjedtek el.

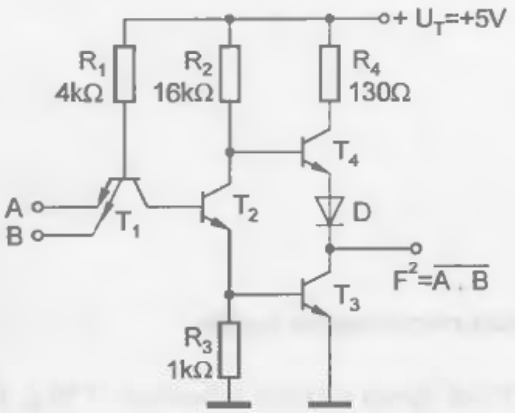
Ahogy az *5.5. táblázatban* látható, ezek az áramkörök kompromisszumot teremtenek a működési sebesség és a teljesitményfelvétel között. A logikai áramkörökben a kapcsolási és a terjedési-késleltetési idők szabják meg az elérhető legnagyobb működési sebességet. A kapunként disszipált teljesítmény nagysága a felépítendő logikai rendszer bonyolultságának korlátot szab.

TTL áramkörcsalád	Egy kapura eső átlagos fogyasztás [mW]	Terjedési-késleltetési idő [ns]
Szabványos	20	10
H (High Speed-Nagy sebességű TTL)	30	6
S (Schottky-diódás TTL)	20	3
L (Low Power-alacsony fogyasztású TTL)	2	35
LS (Low Power Schottky-Alacsony fogyasztású Schottky-diódás TTL)	2	15

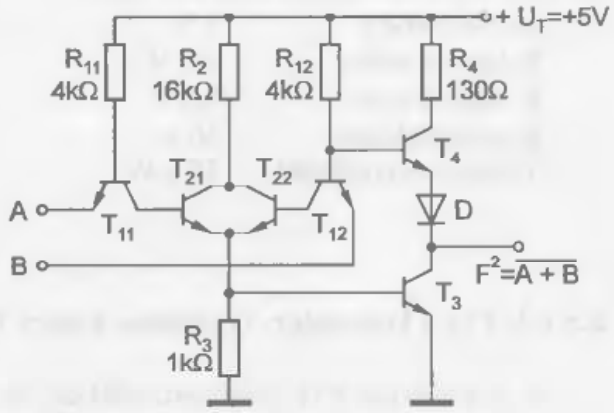
5.5. táblázat. TTL áramkörök összehasonlítása

Egy szabványos TTL NEM-ÉS kapu áramkörét az *5.17. ábra* szemlélteti.
A kapcsolás működése az ábra alapján tárgyalható.

- Ha a T_1 multiemitteres tranzisztor bármelyik emitterére **0** logikai szint (U_{beL}=0,8 V) kerül, akkor az normál üzemmódban működik. Az R_1 ellenállás által meghatározott bázisáram biztosítja, hogy a tranzisztor kollektorfeszültsége kis értékű legyen. Ezért a T_2 tranzisztor lezárt állapotban van. A T_3 tranzisztor is lezárt állapotban van mivel a bázisfeszültsége gyakorlatilag nulla. A T_4 jelű tranzisztor bázisa az R_2 ellenálláson keresztül kapja a telített állapothoz szükséges bázisáramot. Így a kimeneti feszültség valamivel kisebb mint a tápfeszültség, vagyis logikai **1**-nek felel meg (U_{kiH}=2,4 V).
- Abban az esetben, ha a T_1 mindegyik emitterére logikai **H** szint kerül akkor ez fordított üzemmódban működik (az emitter és a kollektor szerepe felcserélődik), és biztosítja a T_2 tranzisztor telített állapotát. Ezáltal T_3 telített, míg T_4 lezárt állapotba kerül. Így a kimeneti feszültség nagyon kis értékű. Ez logikai **L** szintnek felel meg (U_{kiL}=0,4 V)



5.17. ábra. TTL *NEM-ÉS* (*NAND*) kapu



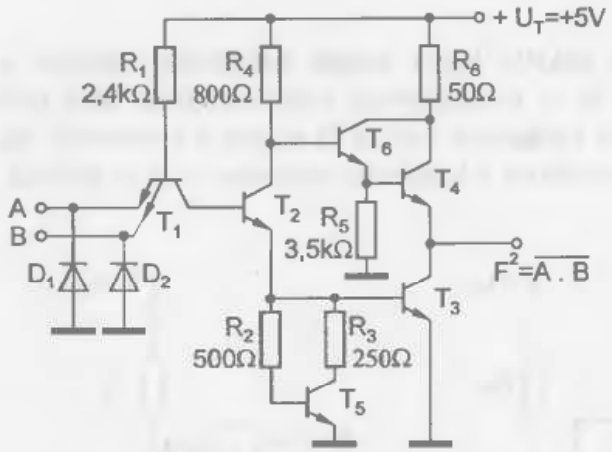
5.18. ábra. TTL *NEM-VAGY* (*NOR*) kapu

Egy TTL *NEM-VAGY* kaput az *5.18. ábra* mutat be. Ennek a működésére az előbbiekben tárgyalt NEM-ÉS kapu működésének ismeretében könnyen lehet következtetni.

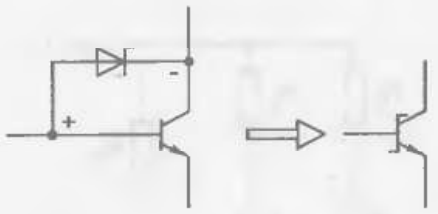
Nagy sebességű TTL (H-TTL)

A szabványos TTL áramkörök kapcsolását módosítva a működési sebesség növelésére nyílik lehetőség. Az 5.19 ábra egy nagy sebességű NAND kapu kapcsolását mutatja, amely az *5.17. ábrán* látható kapcsolás módosított változatának tekinthető. Az új elemek megjelenése 7 ns körüli értékre csökkenti a késleltetési-terjedési időt. A módosító elemek szerepe a következő:

- a T_5 tranzisztor az R_2 és R_3 ellenállással egy nemlineáris ellenállásnak tekinthető:
  - kis U_{B3} feszültségek esetén az ekvivalens ellenállás nagyon nagy, a BE_3 átmenet söntölési hatása kicsi lesz, ami gyorsítja a T_3 tranzisztor bekapcsolását;
  - olyan U_{B3} feszültségek esetén amelyek a T_3 tranzisztor telítését eredményezik, az ekvivalens ellenállás kicsi (lesöntöli a BE_3 átmenetet), ami megvédi a T_3 tranzisztort a túlvezérléstől. Ez a bázison tárolt töltésmennyiség csökkenését eredményezi, vagyis gyorsítja a T_3 tranzisztor bekapcsolását;
- a T_6 és a T_4 tranzisztor egy *Darlington* kapcsolást alkot; következésképpen a T_6 tranzisztor bázisa nagy impedanciásnak tekinthető, a T_2 tranzisztor kapacitív terhelése csökken, ami növeli a T_2 tranzisztor kapcsolási sebességét; ugyanakkor a kimeneti impedancia a T_4 emittere felől kisebb, vagyis az áramkör kevésbé érzékeny kapacitív terhelésekre.



5.19. ábra. H-TTL NEM-ÉS (*NAND*) kapu



5.20. ábra. Schottky tranzisztor

Schottky-diódás TTL áramkörök

Ezeknek az áramköröknek a kapcsolása megegyezik a nagy sebességű TTL áramkörökével, de telítetlen tranzisztorokat használnak a működés során. A telítés elkerülésére Schottky-diódákat használnak, az *5.20. ábrán* látható módon.
Minden egyes tranzisztor kollektor-bázis átmenetével párhuzamosan egy Schottky-dióda van kapcsolva. A Schottky-dióda nyitóirányú feszültségesése (kb.0,3 V) kisebb, mint a bázis-kollektor átmeneté. Ezért a tranzisztor nem kerül telítésbe, ami azt eredményezi, hogy a tranzisztor kapcsolási ideje lényegesen csökken.

Az alacsony fogyasztású Schottky-diódás TTL áramkörök (***LS***) szerkezete ugyanilyen, de a tranzisztorokon folyó áramok kisebbek, amit az ellenállás-értékek növelésével valósítanak meg. A kapcsolási idők természetesen nőnek az alaptípushoz képest. Mivel ezeknek a kapuknak a bemeneti árama kisebb, lehetőség van MOS logikai áramkörökkel való vezérlésükre.

Nyitott kollektoros és háromállapotú TTL áramkörök

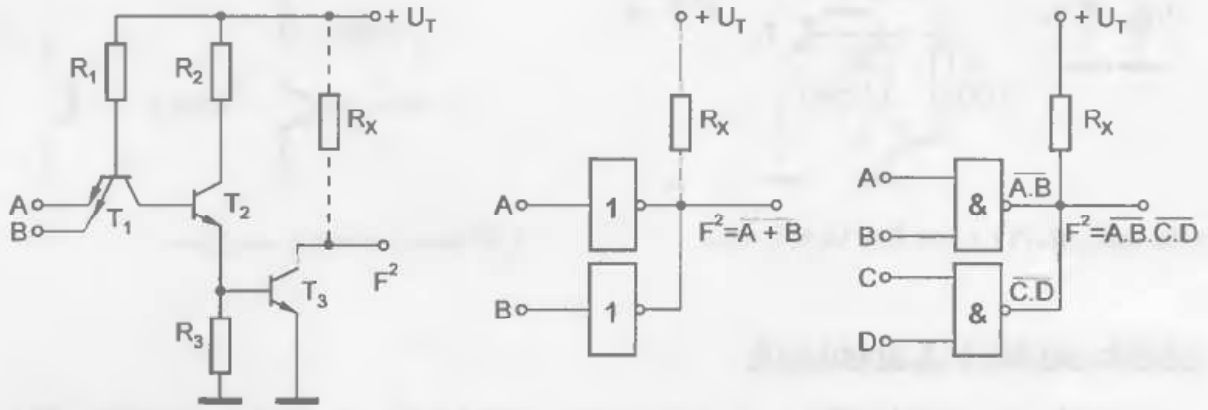
A TTL logikai áramkörök háromféle kimenettel rendelkezhetnek:

- olyan kimenet, amely **H** és **L** logikai szinteket biztosít, mint az *5.17, 5.18, 5.19 ábrán* látható áramkörök esetén;
- *nyitott kollektoros kimenet*, amely esetén a T_4 tranzisztor és a D dióda hiányzik; egy ilyen áramkört mutat az *5.21. ábra*;
- *háromállapotú* (three-state, tri-state) *kimenet*, amely esetén lehetőség van a kimenet leválasztására úgy, hogy az áramkör kimenete nagy impedanciás állapotba kerül; egy ilyen áramkört mutat az *5.22. ábra*.

Két olyan logikai áramkör kimenete, amelyik nem nyitott kollektoros vagy nem háromállapotú, nem köthető össze, mivel egy logikai **L** szint az egyik kimeneten és ugyanakkor egy logikai **H** szint a másik kimeneten egy konfliktus-helyzetet eredményez. Egy konfliktus-helyzet esetén a kimeneten létrejövő feszültségszint nem tartja be a **H** és **L** szintre érvényes értékeket; ugyanakkor a kimeneti áramkörök túlterheléséhez és tönkremeneteléhez vezethet.

Nyitott kollektoros TTL áramkörök

Az *5.21.a ábra* egy nyitott kollektoros NAND kaput mutat. Működése hasonló a szabványos TTL kapuéval, ha csak a T_1, T_2, és T_3 tranzisztorára vonatkoztatjuk. Meg kell említeni, hogy ennél az áramkörnél csak akkor kaphatunk logikai **H** szintet a kimeneten, ha beiktatjuk az R_X jelű ellenállást (a lezárt T_3 tranzisztor a kimenetet szabadon hagyja anélkül, hogy ennek a potenciálját rögzítené).



a) kapcsolási rajz *b) huzalozott VAGY kapcsolat* *c) huzalozott ÉS kapcsolat*

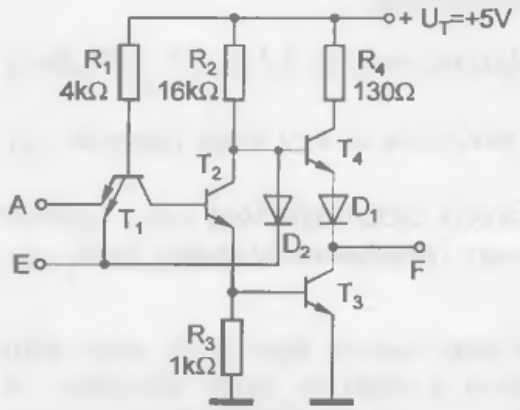
5.21. ábra. Nyitott kollektoros TTL kapu

Az R_X ellenállás méretezésénél figyelembe kell vennünk – telítés esetén – a T_3 tranzisztor maximális kollektoráramát:

$$R_X \geq \frac{U_T}{I_{C_{3\,sat\,max}}}$$

Több nyitott kollektoros kapu összekapcsolásával úgynevezett huzalozott *VAGY* (Wired-OR) és huzalozott *ÉS* (Wired-AND) kapcsolat alakítható ki. A „huzalozott" elnevezés arra utal, hogy a kapuk kimeneteit vezetékkel összekötjük és a közös pontot ellenállás közbeiktatásával tápfeszültségre kötjük. Az *5.21.b ábrán INVERTER*-ekből kialakított huzalozott *VAGY* kapcsolás, míg az *5.21.c ábrán NAND* kapukból kialakított huzalozott *ÉS* kapcsolás látható.

Háromállapotú TTL áramkörök

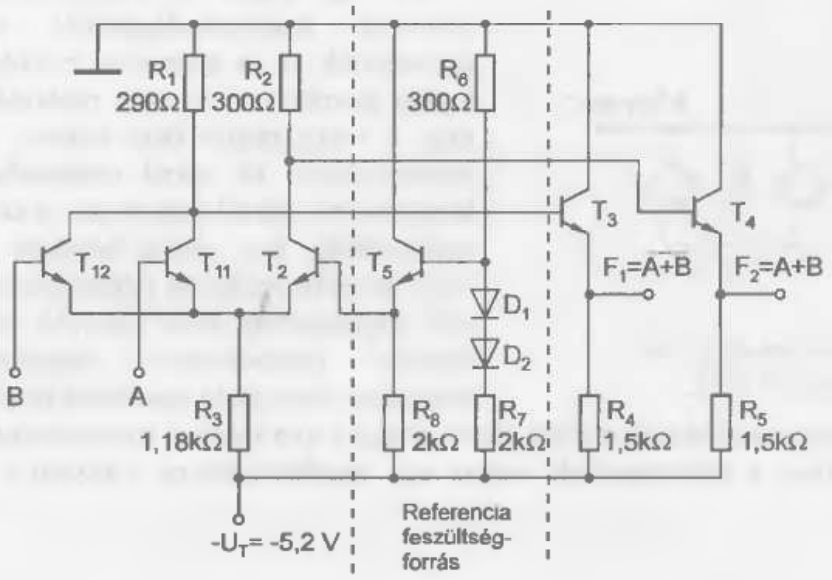


5.22. ábra. TTL háromállapotú inverter

Végül egy háromállapotú TTL inverter kerül bemutatásra (*5.22. ábra*). Amíg az ***E*** engedélyező/tiltó bemenet logikai **1**-et kap, addig az áramkör rendes TTL inverterként működik ($F = \overline{A}$). Ha az ***E*** bemenet **0**-át kap, akkor az a kaput harmadik állapotba vezérli. Ebben az esetben T_1 tranzisztor függetlenül az ***A*** adatbemenettől normál üzemmódban működik. Ezáltal T_2 lezárt állapotban van, amely biztosítja, hogy T_3 is lezárt állapotban legyen. A D_2 anódfeszültsége, amely a dióda nyitóirányú feszültségesésével nagyobb a katódján levő **0** logikai szintnek megfelelő feszültségnél, biztosítja, hogy T_4 is lezárt állapotban legyen. A T_3 és T_4 kimeneti tranzisztorok lezárt állapota eredményezi az adatátvitelt letiltó nagy impedanciájú kimenetet, amely a kapu harmadik állapotának felel meg.

5.5.1.4. ECL (Emitter-Coupled Logic) Emittercsatolású logika

Egy logikai áramkör terjedési-késleltetési ideje jelentősen lecsökken, ha a tranzisztorok telítéses üzemmódját kiküszöböljük. Az előbbiekben megemlített Schottky-diódás TTL áramkörökön kívül az ECL áramköröknél sem kerülnek a tranzisztorok telített állapotba.



5.23. ábra. ECL *VAGY/NEM-VAGY* kapu

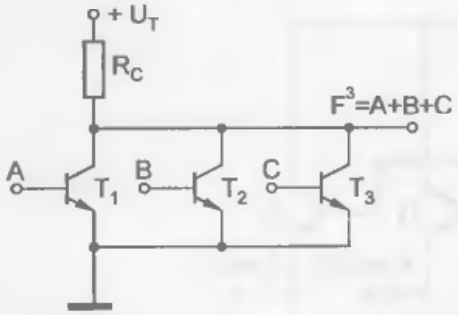
Az *5.23. ábra* egy tipikus ECL VAGY/NEM-VAGY kaput mutat be. Az áramkör egy differenciálerősítőhöz hasonló kapcsolásból (T_{11}, T_{12} és T_2), egy referenciafeszültség-forrásból (T_5) és két kimeneti emitterkövetőből (T_3 és T_9) áll. Ha mindkét (A és B) bemeneten logikai **0** ($U_{beL} = -1{,}325$ V) található, akkor a $U_R = -1{,}175$ V referenciafeszültség biztosítja, hogy T_2 vezessen (nem telített állapotban), és hogy T_{11}, valamint T_{12} le legyen zárva. Ezért F_1, illetve F_2 kimenet logikai szintje: **1** ($U_{beH} = -1{,}025$ V) és **0** ($U_{kiL} = -1{,}5$ V).
Abban az esetben, ha valamelyik, vagy mindkét bemeneten logikai **1**($U_{beH} = -1{,}025$ V) található, akkor valamelyik vagy mindkét bemeneti-tranzisztor (T_{11}, T_{12}) vezet, és T_3 lezárt állapotban van.
Ezáltal $F_1 = \mathbf{0}$, és $F_2 = \mathbf{1}$. Az átkapcsolás a T_{11}, T_{12}, T_2 közös emittereihez csatlakozó R_3 ellenálláson megjelenő jel segítségével történik (innen származik ennek az áramkörcsaládnak az elnevezése).

Az ECL áramkörös logikai hálózatokban nincs szükség inverterre. Ugyanis az ECL, kapuk két komplemens kimenettel rendelkeznek.

Valamennyi bipoláris áramkörcsalád közül az ECL kapcsolások kapukésleltetési ideje a legkisebb. Egy kapu terjedési-késleltetési ideje átlagosan 2 nsec, teljesítményfelvétele 25 mW. Még a Schottky-TTL kapcsolásoknál is gyorsabbak, amelyek ugyancsak telítésmentesen működnek. Azért van ez a különbség, mert a nyitott tranzisztor kollektor emitter feszültsége az ECL-nél nagyobb, sohasem kerül 0,6 V alá. Ezáltal egyrészt nagyobb a telítési feszültségtől való eltérés, másrészt a kollektor-bázis záróréteg-kapacitás is kisebb. Az ECL kapcsolások gyors működésének másik oka a kis, 0,8 V-os szintváltozás. amely átkapcsolásnál lép fel. Ezáltal az elkerülhetetlen parazita kapacitások gyorsan áttöltődnek.

5.5.1.5. IIL (Integrated Injection Logic) Integrált injekciós logika

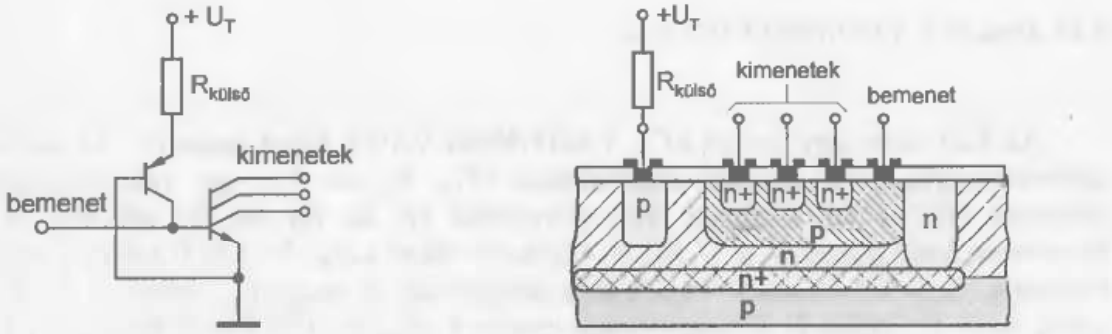
Az IIL (vagy I^2L) kapu integrált áramköri felületigénye sokkal kisebb, mint a TTL vagy az ECL kapué. Körülbelül azonos a MOS kapu felületigényével. Ezért az IIL, a legalkalmasabb bipoláris áramkörcsalád a nagy bonyolultságú integrált áramkörök (*LSI*) előállítására. Kedvező az IIL kapu jóval kisebb teljesítményfelvétele is.



5.24. ábra. Közvetlen csatolású tranzisztor logika (DCTL)

Az IIL kaput a legegyszerűbb közvetlen csatolású tranzisztorlogikából (*5.24. ábra*) fejlesztették ki. A közvetlen csatolású tranzisztor logika áramkörben a kapu működése lényegesen függ a tranzisztorok bázis-emitter feszültségének azonosságától. Ez annál nehezebben teljesíthető követelmény, minél inkább növekszik az áramköri bonyolultság. Egy másik hátrányt jelent a kapu nagy integrált áramköri felületigénye. Ezt főleg az elvi kapcsolásban nem szereplő, de az integrált áramkör működéséhez szükséges kollektor-ellenállást elszigetelő átmenetek okozzák.

Jelentős felületmegtakarítást lehet elérni az egy kaput felépítő tranzisztorok emittereinek, illetve bázisainak a közösítésével, vagyis egy multikollektoros tranzisztor kialakításával (*5.25. ábra*).



5.25. ábra. IIL kapu és integrált áramköri keresztmetszete

A kollektor ellenállás szerepét a következő kapu báziskörében levő, áramforrásként viselkedő *PNP* tranzisztor tölti be. Ez az ún. ***injektor***. Abban az esetben, ha egy adott kapu bemenetén logikai **1**-es szint van (az előző kapu multikollektoros tranzisztora lezárt), akkor az injektor biztosítja a kapu multikollektoros tranzisztorának telített állapotához szükséges bázisáramot. Ha a kapu bemenete logikai **0**, akkor az injektor áramát az előző kapu telített állapotban vezető tranzisztora veszi át. Ezért a szóban forgó kapu multikollektoros tranzisztora lezárt állapotban van.

Az IIL kapu terjedési-késleltetési ideje – 100 nW teljesítményfelvétel mellett – 20 nsec alatt van. Nagyobb teljesítményfelvételnél a terjedési késleltetési idő csökken.

5.5.2. MOS logikai áramkörcsaládok

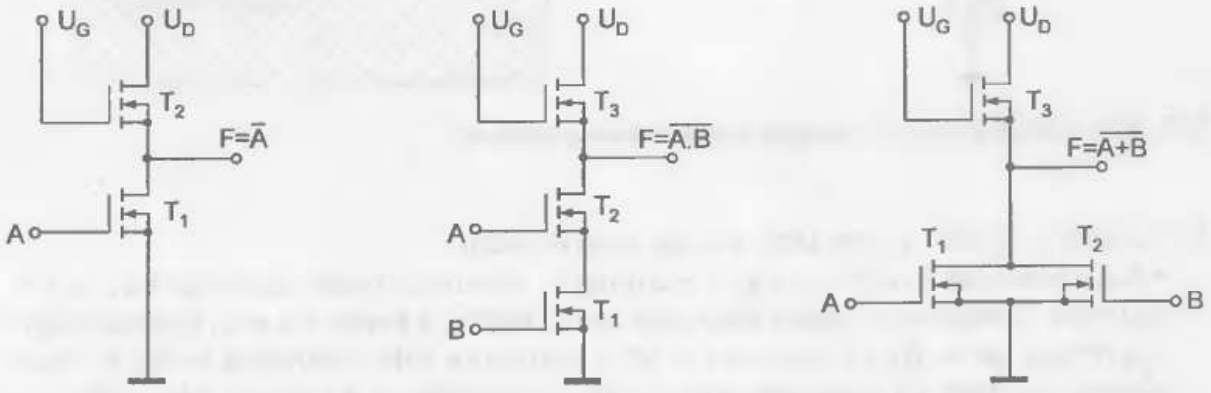
Egy MOS tranzisztor felületigénye szembetűnően kisebb, mint egy bipoláris tranzisztoré. Ezt szemlélteti az *5.13. ábra* is, ahol az arányokat igyekeztünk megőrizni. Tehát egy adott felületre lényegesen több MOS tranzisztort lehet integrálni, mint bipolárist. Ezért egy ***LS***-MOS chip területe nagyjából azonos egy MSI bipoláris chipével (*kivétel az IIL áramkör*). A MOS integrált áramkörök gyártástechnológiája is előnyösebb, mint a bipolárisoké, mivel a technológiai lépések száma kevesebb. Egy másik előnye a MOS áramköröknek a jelentősen kisebb teljesítményfelvétel is. A térvezérlésű tranzisztorok típusától függően megkülönböztethetők a következő MOS áramkörök: az *N*-csatornás MOS (*N*-MOS), a *P*-csatornás MOS (*P*-MOS), és a komplementer MOS (*C*MOS) áramkörök.

5.5.2.1. N-MOS logikai áramkörök

A hetvenes évek közepéig *P*-csatornás MOS térvezérlésű tranzisztoros integrált áramköröket gyártottak. Később sikerült kidolgozni az *N*-csatornás MOS tranzisztorok stabil és megbízható működéséhez megfelelő oxidréteg kialakításának technológiáját. Ezután, előnyös tulajdonságaik alapján, az *N*-csatornás áramkörök rövidesen kiszorították az addig használt *P*-csatornás áramköröket.

Az *N*-csatornás tranzisztorok működésében részt vevő negatív töltéshordozók (az elektronok) mozgékonysága majdnem háromszor nagyobb, mint a *P*-csatornás tranzisztorok, pozitív töltéshordozóinak (*a lyukak*) mozgékonysága. Ezáltal egy azonos meredekségű *N*-csatornás tranzisztor felületigénye körülbelül fele egy *P*-csatornásénak. Az innen származó kisebb gate-kapacitás, az *N*-MOS áramkörök nagyobb működési sebességét eredményezi. Csökken a tranzisztor U_{TO} küszöbfeszültsége is, amely alacsonyabb tápfeszültség alkalmazását teszi lehetővé. Ezért az *N*-MOS áramkört könnyebb illeszteni a széles körű használatnak örvendő TTL-áramkörhöz.

Az *5.11. ábrán* bemutatott kapcsolás tulajdonképpen egy egyszerű inverter, amelynek bemenetét és kimenetét a MOS tranzisztor ***G*** gate-je, illetve ***D*** drain-je képviseli. A telített állapotban vezető tranzisztor alacsony drain feszültségét a nagy értékű R_D ellenállás (100 kΩ felett) biztosítja. Egy nagy értékű integrált ellenállásnak a felületigénye is nagy. Ezért célszerűbb az ellenállást egy MOS tranzisztorral helyettesíteni, amely nemcsak térkihasználás szempontjából kedvező, hanem nagyobb egyenértékű drain ellenállást enged meg (*5.26.a. ábra*). Ezzel magyarázható a MOS áramkörök kis áramfelvétele.



a) inverter *b) NEM-ÉS kapu* *c) NEM-VAGY kapu*

5.26. ábra. MOS kapuk

Az *N*-MOS kapuk, amint az *5.26. ábra* is szemlélteti, kizárólag aktív elemeket tartalmaznak: *N*-csatornás növekményes MOS térvezérlésű tranzisztorokat. A NEM-ÉS, valamint a NEM-VAGY kapu felépítése az inverterére származtatható vissza. A NEM-ÉS kapu kimenetén csak akkor jelenik meg logikai **0**, ha a T_1 és T_2 tranzisztor is vezet, vagyis ha mindkét bemenete logikai **1**-et kap. A NEM-VAGY kapu kimenetén logikai **1** csak akkor jelenik meg, ha T_1 és T_2 is lezárt állapotban van, vagyis ha mindkét bemenete logikai **0**-t kap. A drain ellenállás szerepét betöltő tranzisztor gate-elektródájára kapcsolt U_G feszültség legalábbis a tranzisztor U_{TO} küszöbfeszültségével kell hogy nagyobb legyen, mint a U_D tápfeszültség. Zavarvédettségi szempontból ugyanis célszerű, ha a kimeneti feszültség logikai **1** szinten közel U_D értékű. Ez a kimeneti feszültség a drainközi terhelőtranzisztor gate-source feszültségével kisebb, mint U_G. Másfelől a gate-source feszültség nem sokkal nagyobb, mint U_{TO}.

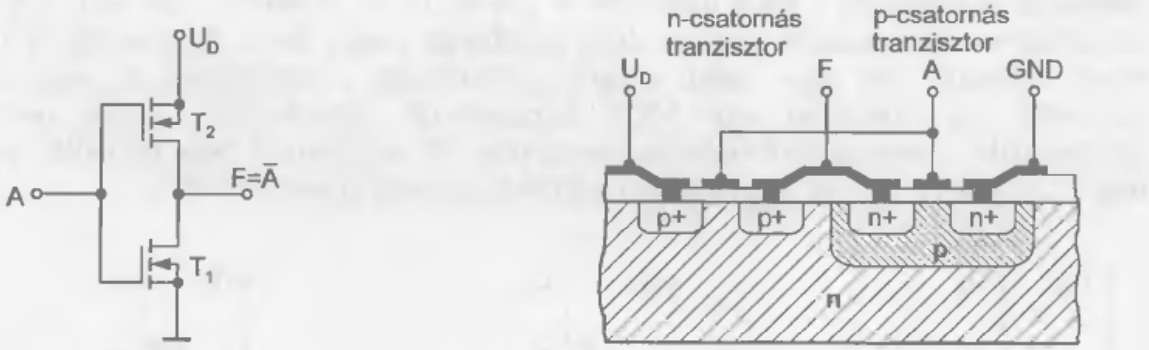
A MOS kapu egyenáramú bemeneti ellenállása nagyon nagy értékű. A gate úgy viselkedik, mint egy kis szivárgási árammal rendelkező kapacitás fegyverzete. A bemeneti feszültség változása a gate-kapacitást töltő és kisütő áramot hoz létre. Ez a rövid idejű áramimpulzus nagyobb, mint a szivárgási áram. Ennek ellenére úgy lehet venni, hogy a MOS tranzisztor nem terheli le az előző kapu kimenetét.

Az *N*-MOS kapuk terjedési-késleltetési ideje átlagban 15 nsec körüli, teljesítményfelvétele pár száz μW-ig emelkedik.

5.5.2.2. Komplementer MOS (CMOS) áramkörök

A komplementer MOS áramkört, amint az elnevezése is mutatatja, *P*-csatornás és *N*-csatornás növekményes MOS tranzisztorpárok alkotják. A CMOS áramkör jellegzetessége a rendkívülien kis áramfogyasztás, széles működési tápfeszültség-tartomány és a nagy zavarvédettség.

A CMOS áramkörök alapeleme az inverter (*5.27. ábra*).



5.27. ábra. CMOS inverter és integrált áramköri keresztmetszete

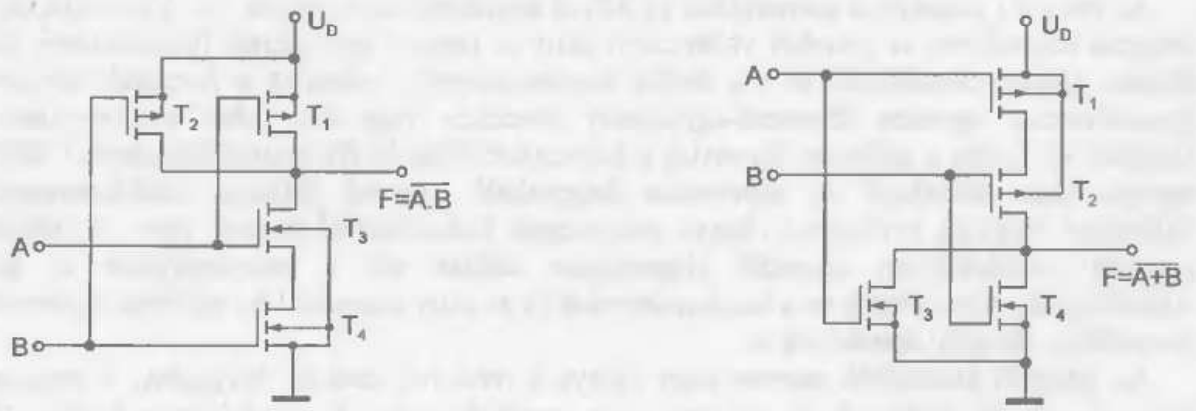
A kapcsolás működése a fenti ábrák alapján magyarázható.

- Nulla bemeneti feszültségnél az *N*-csatornás T_1 tranzisztor lezárt állapotban van, és a *P*-csatornás T_2 tranzisztor telített állapotban vezet. Ezáltal a kimenet a $+U_D$ tápfeszültségre van rövidre zárva. Ha a kimenetet egy MOS tranzisztor gate-elektródája terheli le, akkor elhanyagolhatóan kis veszteségi áram, folyik a tápegységből a kimenet felé, valamint a lezárt T_1 tranzisztoron keresztül a föld felé.

• Abban az esetben, ha a bemeneti jel megközelítőleg $+U_D$ értékű, akkor T_1 vezet, és T_2 lezárt. A kimenet a T_1 tranzisztoron keresztül földpotenciálra kerül. Ebben az esetben is a tápegységből felvett áram a veszteségi-szivárgási áramokból tevődik össze. Jelentősebb áramfelvétel csak akkor lép fel, amikor a kapu az egyik állapotból a másikba kapcsol át. Ekkor ugyanis az egyik tranzisztor még nem zárt le teljesen, és a másik vezetni kezd. Ezenkívül még fel kell tölteni, vagy kisütni a következő kapu bemeneti gate-kapacitását. Ezek miatt az átkapcsolás alatt a tápáramfogyasztás egy ún. áramtüskével növekszik.
Ez magyarázza meg, hogy miért nő meg számottevően a CMOS áramkörök teljesítményfelvétele, ha a működési frekvencia növekszik.

Az *5.28. ábrán* két bemenetű CMOS *NEM-ÉS*, valamint *NEM-VAGY* kapu látható. A *NEM-ÉS* kapu esetén a *P*-csatornás tranzisztorok (T_1 és T_2) párhuzamosan kapcsolódnak, míg az *N*-csatornásak (T_3 és T_4) sorosan. Az áramkör működése az ábrák alapján magyarázható.

• Ha mindkét bemeneten logikai **1** van, akkor a telített állapotban vezető T_3 és T_4 tranzisztor a kimenetet közel földpotenciálon tartja, vagyis logikai **0** szinten. Minden más bemeneti jelkombináció esetén, vagyis ha az egyik bemeneten vagy- mind a kettőn logikai **0** van, akkor az egyik vagy mind a két *P*-csatornás, illetve *N*-csatornás tranzisztor kinyit, illetve lezár. A kimeneti feszültség közel $+U_D$ értékű; amely logikai **1** szintnek felel meg.

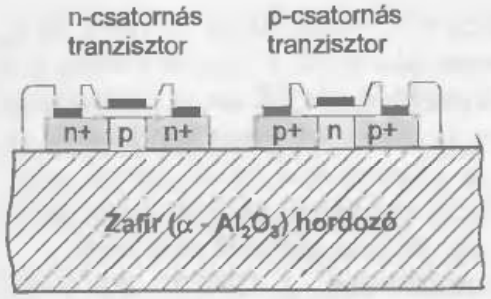


a) NEM-ÉS kapu *b) NEM-VAGY kapu*

5.28. ábra. CMOS kapuk

A *NEM-VAGY* kapu esetén a *P*-csatornás tranzisztorok (T_1 és T_2) sorba, az *N*-csatornás tranzisztorok (T_3 és T_4) párhuzamosan kapcsolódnak. A kapu működésére az előbbi *NEM-ÉS* kapu működésének alapján könnyen lehet következtetni. A CMOS áramkörök tápfeszültsége +3 és +15 V közötti értékeket vehet fel. Egy kapu terjedési-késleltetési ideje átlagosan 25 nsec, és a nyugalmi teljesítményfelvétel 50 nW körül van.

Nagy jelentőségű változata a CMOS áramköröknek az SOS (*Silicon on Sapphire*) áramkör (*5.29. ábra*). Előnyös tulajdonságokkal rendelkező áramkör származik, ha a szilícium helyett zafír hordozóra alakítják ki a komplementer MOS tranzisztorokat. Az áramköri elemek között a szigetelési ellenállás nagyon nagy értékű. A hordozó parazita-kapacitásainak látványos csökkenése miatt az SOS áramkörök működési sebessége egy nagyságrenddel nagyobb, mint a szilícium alapú CMOS áramköröké. Egy kapu terjedési-késleltetési ideje 1-2 nsec. Jelentősen csökken az áramkör nyugalmi áramfelvétele is.



5.29. ábra. Zafírra növesztett szilícium (SOS) komplementer MOS tranzisztorok

Az áramköri elemek felületi sűrűsége körülbelül négyszer akkora, mint a hagyományos CMOS áramkörök esetén.

Az SOS típusú CMOS áramkör keresztmetszetét az *5.29. ábra* szemlélteti. A szigetelő hordozó kristályszerkezete, a kristályrács és mérete hasonló kell, hogy legyen a tranzisztorokat felépítő szilíciuméval. Ezeket a krisztallográfiai követelményeket jó szigetelő tulajdonságai mellett a zafír teljesíti. A zafír helyett spinellt (*magnéziumaluminát*) is lehet használni. A spinell szerkezete még közelebb áll a szilíciumhoz, megmunkálása is könnyebb.

5.6. Az integrált áramkörök gyártástechnológiája

A digitális technika fejlődésére az alkatelemek gyártástechnológiája döntő befolyást gyakorol. Az eddigi legszámottevőbb fejlődés a mikroelektronikának köszönhető. A mikroelektronika tulajdonképpen az integrált áramkörökkel vált fogalommá.

Az integrált áramkörök ***monolitikus*** és ***hibrid*** áramkörökre oszthatók fel. A monolitikus integrált áramkörben az áramkör valamennyi aktív és passzív építőelemét (tranzisztorok és diódák, illetve ellenállások és kis értékű kondenzátorok), valamint a hozzájuk tartozó összekötéseket egyetlen félvezető-egykristály lemezkén vagy ún. „***chip***"-en (*morzsán*) alakítják ki. Eddig a szilícium bizonyult a legmegfelelőbbnek. Az áramkörök elemeit több egymás után következő és egymáshoz kapcsolódó gyártási fázisban adalékanyagok különböző mértékű bevitelével, illetve zárórétegek kialakításával hozzák létre. A hibrid integrált áramkörökben szigetelő alapanyagon állítják elő a vezetőpályákat és az ellenállásokat. Abba ültetik be a kondenzátorokat és az aktív elemeket. Az utóbbiak lehetnek monolitikus integrált áramkörök is.

Az integrált áramkörök nagyon nagy előnye a rendkívül csekély helyigény, a nagyon kedvező gyártási költségek és az igen nagy megbízhatóság. A chipfelületen kialakított funkciók sűrűsége, ill. az integrált áramkör mérete szerint a bonyolultság négy fokozatát különböztetjük meg:

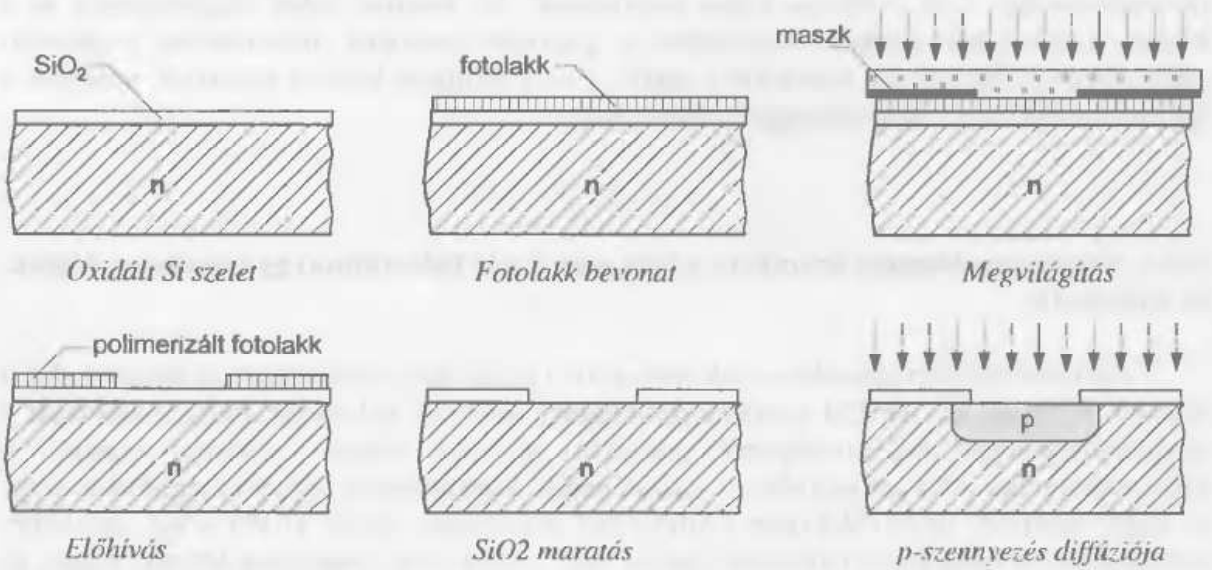
- ***SSI*** (Smal Scale Integration): – kis bonyolultságú fokozat, 50 tranzisztornál (vagy 50 kapunál) kevesebbet tartalmaz;
- ***MSI*** (Medium Scale Integration): – közepes bonyolultságú fokozat, 50-nél több, de 500-nál kevesebb tranzisztort (vagy 200 kaput) tartalmaz;
- ***LSI*** (Large Scale Integration): – nagy bonyolultságú fokozat, 500 tranzisztornál több de 10 000 ... 20 000-nél kevesebb tranzisztort tartalmaz;
- ***VLSI*** (Very Large Scale Integration) nagyon nagy bonyolultságú fokozat, 10 000 ... 20 000-nél több tranzisztort foglalnak magukba.

A következőkben a monolitikus digitális integrált áramkörök gyártástechnológiáját ismertetjük röviden.

A félvezető eszközök előállítására különféle anyagokat lehet használni, eddig azonban a szilícium bizonyult a legmegfelelőbbnek. Az integrált áramkörök gyártástechnológiájának első lépése a nagy tisztaságú szilícium-egykristály rúd növesztése. Ezután ebből vékony tárcsákat szeletelnek. A szeletelés a kristályrács meghatározott irányában történik. Egy szeleten egymás mellett több tucat integrált áramkört lehet előállítani.

Még mielőtt a szeletet chipekre (morzsákra) darabolnák fel, minden egyes integrált áramkört egy letapogató, szondás berendezéssel megvizsgálnak. A rossznak minősített áramkörök megkülönböztető jelzést kapnak, így a szelet feldarabolása után könnyen fel lehet őket ismerni, és végül kiselejtezni. A feldarabolást a törési vonal mentén egy gyémántcsúccsal való karcolás segíti elő. A feldarabolás után keletkező minden egyes chipet egy-egy megfelelő alapra helyeznek. A kivezetéseket bekötik, és végül tokozással légmentesen lezárják. Az utolsó lépés a kész integrált áramkör részletes vizsgálata.

Ebből a bonyolult gyártástechnológiai eljárásból csak egy részt ismertetünk: – az integrált áramkör kialakításának lépéseit a szilícium chipen. A bipoláris és MOS integrált áramköröket nagypontosságú planáris technológiával állítják elő. Ennek kulcsa a szilícium szelet felületén létrehozott szilícium-dioxid (SiO_2) rétegből fotolitográfiai és maratási eljárásokon alapuló *oxidmaszk-kialakítás*. Az oxidréteg a legtöbb szilícium technológiában használatos szennyező adalékanyag számára áthatolhatatlan. Így az oxidmaszk nyílásain keresztül. Az adalékanyag diffúziója során *P*- vagy *N*-típusú adalékoltságú rétegeket lehet kialakítani (*5.30. ábra*). A szilícium szelet felületének oxidálását különleges kályhában végzik. Itt vízgőzzel telített oxigénáramban nő a szilícium szelet felületén az egyenletes vastagságú oxidréteg. A fotólitográfiai eljárás első lépése az oxidréteg felületének vékony fényérzékeny lakkréteggel (ún. fotóreziszt) való bevonása. A fotólakk egyenletes vastagságát a szelet nagy sebességű centrifugálásával érik el. A lakk felületére nagy pontosságú, állítható befogószerkezettel helyezik rá a maszk fotónegatívját, amelyen keresztül megvilágítják a fotólakkréteget. A megvilágított helyeken a lakkréteg polimerizálódik. Előhíváskor ez a lakkréteg nem oldódik, csak ott, ahol nem volt megvilágítva. A fedetlen helyeken az oxidréteget egy megfelelő savkeverékkel eltávolítják. A savkeverék sem a polimerizált lakkréteget, sem a tiszta szilíciumot nem támadja meg. Végül a polimerizált lakkréteg eltávolítása után a maszkírozott szelet készen áll a megfelelő átmenet diffúziós kialakítására.



5.30. ábra. Az oxidmaszk előállítása fotólitográfiai eljárással és egy *P*-adalékoltságú rétegdiffúzió létrehozása

Az adalékanyagok diffúziója különleges csőkemencében 1150–1350 °C hőmérsékleten történik. Ez a hőmérséklet főleg az adalékanyag típusától, a kialakítandó adalékolt réteg mélységétől és a diffúziós időtartamtól függ. Az integrált áramköri elemek előállításánál minden diffúziós lépésnél a megfelelő adalékanyagot sajátos oxid-maszkon keresztül diffundáltatják.

A diffúziós módszert a MOS integrált áramköri technológiában sok esetben kiszorítja az ***ionimplantációs technika***. Az ionimplantáció légüres térben történik. A szennyező adalékanyag ionjai nagy sebességgel ütköznek a szelet kívánt részéhez. Sebességüktől függően az ionok bizonyos mélységig hatolnak be. Ezután a szilícium lemezt felhevítik kb. 800°C-ig, hogy az ionok a kristályrács megtelelő helyeit foglalják el. Az ionimplantáció legfontosabb előnye az adagolási szint egyszerű, pontos és reproduktibilis szabályozása. Az ionimplantált MOS tranzisztor kevesebb felületet foglal el, és sokkal nagyobb a működési sebessége, mint a diffúziós MOS tranzisztornak.

Az *epitaxiális rétegnövesztés* a nagy sebességű integrált áramkörök gyártástechnológiájának fontos lépése. Ez az eljárás elvileg abban áll, hogy ha egy félvezető kristály olvadáspontjához közeli hőmérsékleten érintkezik az alapanyaga vegyületének gőzével (a szilícium esetén $SiHCl_3$ vagy $SiCl_4$), akkor ebből egy bonyolult. lecsapódási folyamat révén a félvezető kristály továbbépül. Ha a gőzbe adalékanyagot visznek be, akkor az epitaxiális réteg megfelelő mértékben *P*- vagy *N*-típusú adalékoltságú lesz. A vékony epitaxiális rétegben diffúzióval készítik a tranzisztorokat.

Az előbbiekben ismertetett gyártástechnológiai folyamat eredményeként a chipben kialakított tranzisztorok keresztmetszetét az *5.13. ábrán* láthatjuk. Az áramkörök felépítésében a tranzisztorokon kívül diódák és ellenállások is találhatók. A diódák a bipoláris tranzisztorok bázis-emitter vagy bázis-kollektor átmenetének egyikéből alakíthatók ki. Az integrált áramköri ellenállásokat úgy állítják elő, hogy adalékanyagok diffúziójával egy adott térfogatban megváltoztatják' a szilícium vezetési típusát.

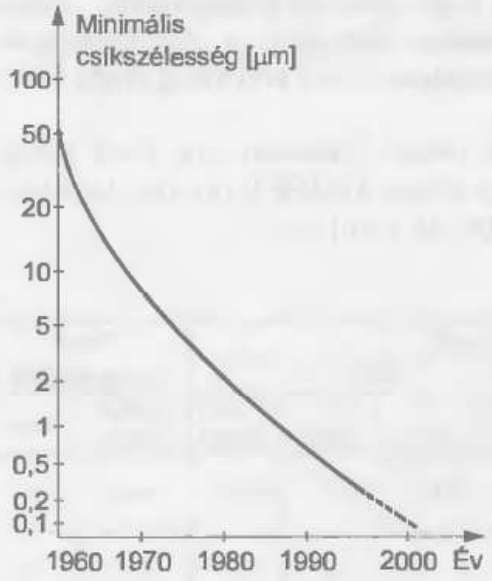
Az egyes integrált áramköri elemeket összekötő vezetékeket a chip felületén alakítják ki. A teljesen oxidált felületen a kontaktusok helyén eltávolítják az oxidréteget, és az egész felületet vékony fémréteggel vonják be. Ez vákuumpárologtatással valósul meg. Ezután a megfelelő huzalozási alakzatot szintén fotólitográfiai eljárással alakítják ki.

Az előbbiekben leírt gyártástechnológiai lépések a különböző logikai áramkörcsaládok megvalósításáról csak általános képet nyújthatnak: Az áramkörcsalád függvényében és a kívánt integráltsági foknak megfelelően a gyártástechnológiai folyamatban különböző eltérések mutatkoznak. Ez leginkább a maszkok és a diffúziós lépések számával; valamint a minimális diffúziós csíkszélességgel jellemezhető.

5.6.1. Nagy bonyolultságú áramköri (LSI-Large Scale Integration) gyártástechnológiák és fejlődésük

A mikroszámítógép alkatelemeinek legnagyobb részét nagy bonyolultságú integrált (LSI) áramkörök teszik ki. Az LSI gyártástechnológiák, valamint fejlődésük összehasonlítását a mikroszámítógépek teljesítményéből származó tényezők alapján célszerű végezni. A mikroelektronika erőteljes fejlődése a számítógépek teljesítményét annyira megnövelte hogy az összehasonlítás lépcsőfoka nem a különböző számítógéptípusok között a legvitatottabb, hanem a számítógép és az élőlények „működését" szabályozó rendszerek között. Ennek az összehasonlításnak a gondolatát először Norbert Wiener vetette fel. Később 1956-ban

Neumann János két alapvető összehasonlítási tényezőt fogalmazott meg. Az egyik a rendszerek térfogategységéhez viszonyított számítási teljesítmény, amely az alapvető építőelem (*logikai kapu, vagy neuron*) működési sebességének és térbeli sűrűségének szorzatával fejezhető ki. A másik tényező az egy számítási műveletre jellemző energiafogyasztás. Neumann mindkét tényező szempontjából az emberi agyat a számítógépnél körülbelül tízezerszeresen felsőbbrendűnek találta. Azóta a technológia fejlődése ezt az arányt csökkentette, és tovább csökkenti. Az egyedüli bipoláris technológia, amely valamelyest versenyképes maradt a MOS technológiával, az az IIL.



5.31. ábra. Az integrált áramkörök gyártástechnológiájának fejlődése

Az alapvető logikai építőelem térbeli sűrűségét bonyolultsága (egy kapura jellemző áramköri elemek száma), valamint az egy tokban elhelyezhető logikai elemek száma határozza meg. A MOS technológia a MOS kapu egyszerűsége és a MOS tranzisztor kis felületigénye miatt a legelterjedtebb ***LSI*** technológia. A MOS technológiák közül a legnépszerűbb az *N*-MOS és CMOS, ugyanakkor feljövőben van az SOS (zafír alapú CMOS).

Az integrált áramkör leegyszerűsítve úgy tekinthető, mintha a félvezető egy keskeny rétegében létrehozott különböző adalékoltságú és alakú csíkokból tevődnék össze. Mivel a chip méretét funkcionális és gyártástechnológiai tényezők korlátozzák, a chipre integrálható áramköri elemek számát a csíkszélesség határozza meg. A nagy sorozatban gyártott integrált áramkörök esetén a legkisebb megvalósítható csíkszélesség 2 μm. A fotólitográfiai eljárás alkalmazása kb. 1 μm csíkszélességig terjedhet ki. A látható fény hullámhossza ugyanis már nem elegendően kicsi 1 μm-nél keskenyebb csíkok előállítására. Kisebb csíkszélességet egyelőre csak laboratóriumi körülmények között lehet Röntgen-, elektron-, vagy ionsugarakkal megvalósítani. Ezeknek a sugaraknak a hullámhossza sokkal kisebb mint a látható fényé. A Röntgen (*X*) sugarak segítségével 0,2 μm csíkszélességet is sikerült előállítani. Ez már a baktériumok és a vírusok méreteihez hasonlítható. Az elektronsugaras litográfia 0,3 μm körüli csíkszélesség megvalósítását teszi lehetővé. Nagy előny ennek az eljárásnak, hogy nem igényel exponálási maszkot. Az elektronsugár számítógéppel vezérelt eltérítésével nagy pontossággal követhető a diffúziós alakzat. Elektronsugár helyett ionsugarat is lehet alkalmazni. Ennek az az előnye, hogy ugyanazzal a berendezéssel nemcsak a fotólakk exponálását lehet elvégezni, hanem az ionplantálást is. Az ionsugárral 0,1 μm körüli csíkszélesség valósítható meg.

Az *5.31. ábra* a nagy sorozatban gyártott integrált áramkörök minimális csíkszélességének rohamos vékonyodását szemlélteti.

A geometriai méretek csökkentésével a terjedési-késleltetési idő és a teljesítményfelvétel is csökken. Újabb típusú áramkörökkel ezek a tényezők többszörösen csökkenthetők. A kutatási eredmények a gallium-arzenid (*GaAs*) félvezetők terén kecsegtetőek. Ezzel egy egészen új típusú integrált áramkör megjelenésének vagyunk a tanúi: a nagyon nagy sebességű integrált áramköröknek – ***VHSIC*** (Very High Speed Integrated Circuits).

A gallium-arzenidben az elektronok mozgékonysága körülbelül ötször nagyobb, mint egy azonos adalékoltságú szilíciumban és mintegy tízszer nagyobb, mint egy N-MOS tranzisztor csatornájában. Ezenkívül biztató az, hogy a szilícium alapú integrált áramkörökhöz képest kisebb a teljesítmény-felvétele, és nagyobb az áramköri sűrűsége. A gallium-arzenid hátránya, hogy nem sikerült a felületére oxidréteget növeszteni. Ebből kifolyólag GaAs MOS tranzisztorokat nem lehet előállítani.

Így a GaAs alapú integrált áramkörök MESFET (Metal-Semiconductor Field Effect Transistor) és JFET tranzisztorokból épülnek fel. Egy GaAs MESFET inverter terjedési-késleltetési ideje 70 psec, teljesítményfelvétele pedig 500 μW körül van.

Jellemzők	Jelenlegi technológiák								Jövő technológiák	
	Bipoláris				MOS					
	TTL	S-TTL	ECL	IIL	P-MOS	N-MOS	C-MOS	C-MOS (SOS)	C-MOS (SOS)	GaAs
A gyártástechnológia bonyolultsága (a technológiai folyamat lépéseinek száma)	18-20	18-23	19-23	13-17	8-14	9-15	14-17	14-20	14-20	16
A logikai alapelem bonyolultsága (a kétbemenetű kapu alkatrészeinek száma)	12	12	8	3-4	3	3	4	4	3-4	2
Logikai elemek sűrüsege [kapu/mm^2]	10-20	20-40	15-20	75-150	75-150	100-200	40-90	100-200	200-500	300-1000
Terjedesi késleltetesi idő [ns] (tipikus érték)	6-30 (10)	2-10 (5)	0,7-2 (2)	7-50 (20)	30-200 (100)	4-25 (15)	10-35 (20)	4-20 (10)	0,2-0,4 (0,3)	0,05-0,1 (10)
Terjedesi idő x teljesítmény-felvétel [pJ]	30-150	10-60	15-80	0,2-2,0	50-500	5-50	2-40	0,5-30	0,1-0,2	0,01-0,1
Tökéletesítési és fejlesztési lehetőség	kis	kis	kis	közepes	kis	nagy	közepes	nagy	nagy	nagy

5.5. táblázat. Monolitikus integrált áramkörök jellemzői

Az *5.5. táblázat* áttekintő összehasonlítást nyújt a félvezető alapú integrált áramkörök különböző típusai között. Az integrált áramkör egységnyi felületéhez viszonyított számítási teljesítményt a logikai alapelem-sűrűség és a terjedési-késleltetési idő hányadosa tükrözi. Ez legkisebb a TTL, és a *P*-MOS áramkörök esetén. A CMOS áramköröknél az előbbiekhez viszonyítva megduplázódik, majd ismét körülbelül a kétszerese a Schottky TTL, és az ECL áramköröknél. Jelenleg az *N*-MOS és az SOS áramkörök esetében a legnagyobb. Ezt a tényezőt az SOS áramkörök fejlesztési lehetőségei majdnem két nagyságrenddel növelhetik meg. A GaAs alapú integrált áramkörök esetében viszont majdnem három nagyságrendű növekedéssel is számolhatunk.

A technológia erősen befolyásolja a számítógép teljesitményének a növekedését. Ezt az *5.6. táblázat* tükrözi. Sok tekintetben a nyolcvanas évek számítógépei is lemaradnak az emberi agy teljesítőképességétől. A működési sebesség terén messzemenően túlszárnyalják az emberi agy képességeit, viszont az építőelem-sűrűség és különösen a memória kapacitása tekintetében a lemaradás eléggé jelentős. Az átlagos emberi agy 62,5 millió géppel írott oldalnak (2000 jel/oldal) megtelelő információt tud tárolni. Ez egymillió Mbitet jelent. Ehhez az optikai lemez 20 000 Mbit, a videoszalag és a videolemez 150 000 Mbit kapacitása áll a legközelebb. Ezért úgy tűnik, hogy a biológiai logikai rendszerek terén a kutatás nagyon is indokolt.

Jellemző tulajdonságok	A számítógép körülbelül:				Az emberi agy
	1956-ban	1980-ban	1990-ben	1997-ben	
Alkatelemek sürüsége [áramkörök, vagy neuronok/cm^4]	10^{-1}	10	10^5	10^6	10^7
Teljesítményfelvétel [W/kapu]	10^{-1}	3.10^{-2}	10^{-6}	10^{-8}	10^{-9}
Sebesség [ciklus/sec]	10^6	10^8	10^9	10^{10}	10^2
Kapcsolás/sec/cm^4	10^5	10^9	10^{14}	10^{16}	10^9
Kapcsolás/Joule	10^7	3.10^9	10^{15}	10^{17}	10^{11}
Memóriasűrüség [bit/cm^4]	5.10^2	5.10^4	5.10^7	5.10^8	10^{16}
Az egység összes alkatelemeinek száma [áramkörök vagy neuronok]	10^5	10^5	10^5	10^7	10^{10}
Számítási teljesítmény [kapcsolás/sec]	10^{11}	10^{13}	10^{14}	10^{16}	10^{12}

5.6. táblázat. A számítógépek az emberi agyhoz viszonyítva

Az IBM 1997-ben egy olyan új ***rézalapú*** gyártási módszert mutatott be, amely valószínűleg forradalmasítja a nagybonyolultságú integrált áramkörök gyártástechnológiáját. Az új technika gyorsító hatását az okozza, hogy a réz felhasználásával kisebb, energiatakarékosabb áramkörök (pl. olcsóbb mikroprocesszorok és *ASIC*-ek építhetők; az *ASIC* az alkalmazásspecifikus integrált áramkör angol rövidítése). A réz ellenállása kisebb mint az alumíniumé, ezért kisebb (vékonyabb a szükséges huzalozás) és gyorsabb lapkák készíthetők belőle. Ugyanakkor azonban a réz szennyezi a szilíciumot. Az IBM mérnökei úgy oldották meg a szennyezési problémát, hogy elkülönítették a réz áramkört a szilíciumtól, majd egy speciális bevonattal szigetelték a rezet.

A réz már 1998-ban lehetővé teszi a 0,20 μm-es vezetősáv szélességű gyártást, később pedig az alumínium alsó mérethatárának tekintett 0,13 mikron alá is be lehet menni.

Megjegyzés: – Az Intel 386-os processzorcsaládja 1,5 μm-es technológiával, a Pentium II-es család pedig 0,35 μm-es technológiával készült.

Összefoglaló kérdések:

1. Mit nevezünk pozitív- és negatív feszültségű rendszernek?
2. Milyen jellemző adatai vannak a logikai áramköröknek?
3. Milyen hátrányai vannak a diódás kapuáramköröknek?
4. Mit feltételez egy logikai rendszert alkotó kapuk együttesének helyes működése?
5. Milyen jellemzői vannak az RTL- és DTL logikai áramköröknek?
6. Milyen változatai vannak a TTL áramkörcsaládnak?
7. Hasonlítsa össze a H-TTL és S-TTL áramköröket?
8. Milyen előnyei vannak a nyitott kollektoros TTL áramköröknek?
9. Az ECL logikai áramkörök miért tekinthetők a leggyorsabb bipoláris áramkörcsaládnak?
10. Milyen jellemzői vannak az IIL logikai áramköröknek?
11. Milyen előnyökkel rendelkeznek a MOS logikai áramkörcsaládok a bipoláris változatokhoz képest?
12. Milyen jellemzői vannak az N-MOS logikai áramköröknek?
13. Milyen jellemzői vannak a C-MOS logikai áramköröknek?
14. Az integrált áramkörök milyen bonyolultsági fokozatai ismeretesek?

6. Memóriák

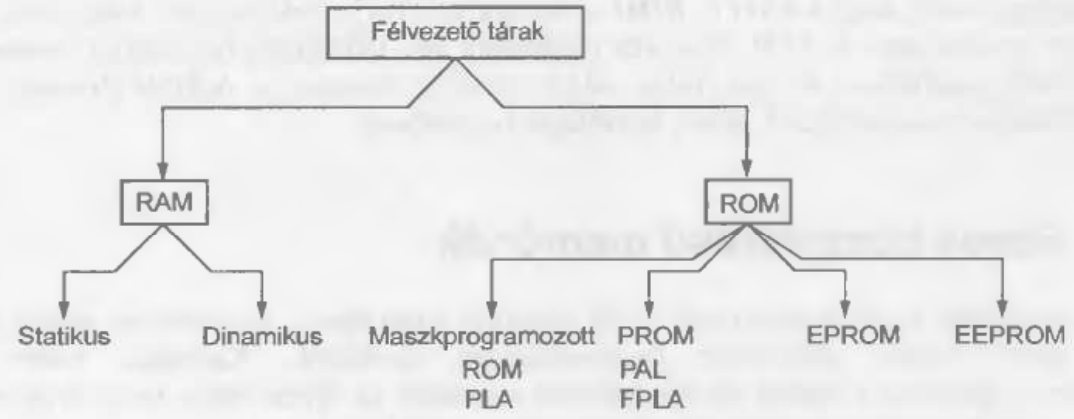
A memória vagy tár azon eszközök összessége, amelyek az információkat tetszés szerinti ideig megőrzik, és ahonnan azokat bármikor ki lehet olvasni. A számítógép utasításai által feldolgozandó bitek, bájtok és karakterek azon sorát, amely a memóriában is tárolható szónak (*Word*) nevezik. A ***szó hosszúsága*** (amelyet rendszerint bitekben adnak meg), a számítógépek és egyben a memóriák egyik legfontosabb jellemzője.

A memóriában minden szó tárolására külön rekesz áll rendelkezésre. A rekeszt a cím (*Address*) azonosítja. A megcímezhető rekeszek száma és a rekeszben tárolható szó hosszúsága a memória szervezését mutatja. A ***memória kapacitását***, amelyet bitben fejeznek ki, a rekeszek számának és a tárolható szó hosszúságának szorzata fejezi ki.

A címzés után csak egy bizonyos időintervallum eltelte után lehet a tárolt adatot (*Data*) kiolvasni, vagy újabbat beírni. Ez az intervallum az ún. ***hozzáférési*** vagy ***elérési idő*** (*Acces time*). A kapacitás és a hozzáférési idő a memóriák két fontos jellemzője.

A memóriákat technológiai felépítésük és funkcionális jellemzőik szerint osztályozzák.

Technológiai felépítésük szerint a következő legelterjedtebb memóriatípusok különböztethetők meg: ***félvezető***, ***mágneses***, ***optikai*** és ***magneto-optikai*** memóriák. A mikroszámítógépek esetében a félvezető memóriáknak van a legnagyobb jelentőségük. Ezért a következőkben részletesebben csak a félvezető memóriákat fogjuk tárgyalni (*6.1 ábra*).



6.1. ábra. Félvezető tárak típusai

A memóriák egyik funkcionális jellemzője a megcímzett rekesz hozzáférési módja. Így megkülönböztethetők:

- ***tetszőleges*** (véletlenszerű) hozzáférésű memóriák,
- ***soros*** (szekvenciális) hozzáférésű memóriák,
- ***asszociatív*** memóriák.

A tetszőleges hozzáférésű memóriában bármely adat, függetlenül a címétől, ugyanolyan rövid idő alatt érhető el. Az ilyen típusú memória rövidített elnevezése ***RAM*** (az angol Random Acces Memory kifejezés rövidítése, azaz a véletlen hozzáférésű táré). A soros hozzáférésű memória esetében a keresett adat hozzáférési ideje különböző és függ a címétől, valamint a keresés kezdő címétől.

Az asszociatív memória bemenetén levő szót egyidejűleg összehasonlítja az egyes címeken tároltakkal, és azt a címet adja meg, amely által kijelölt rekeszben a tárolt szó megegyezik a bemeneti szóval. Az asszociatív memória használatos elnevezése ***CAM*** (az angol *Content Addressable Memory* kifejezés rövidítése).

Az információ beírhatósága szempontjából két típust lehet megkülönböztetni:

- ***végleges*** és
- ***módosítható memóriát***.

A végleges beírás legtöbbször irreverzibilis szerkezeti változást hoz létre a memóriában, és utána a tartalma nem változtatható meg. Ebből a típusból az információt csak kiolvasni lehet. Rövidített elnevezése ***ROM*** (az angol ***Read Only Memory*** kifejezés rövidítése, azaz a permanens – csak olvasható – táré).

Az információ megőrzésének szempontjából:

- ***statikus*** és
- ***dinamikus memóriákat***

különböztetünk meg.

A statikus memória az információt korlátlan ideig megőrizheti, feltéve, ha a tápfeszültsége nem szűnik meg. A dinamikus memória viszont időnként "felfrissítést" igényel, másként tartalma véglegesen törlődik.

A memóriákat még egyéb funkcionális jellemzőik szerint is meg lehet különböztetni, például a tápfeszültség kiesésekor megsemmisülő (RAM), illetve megmaradó tartalom (ROM) vagy a vezérlőjelek összetettsége alapján.

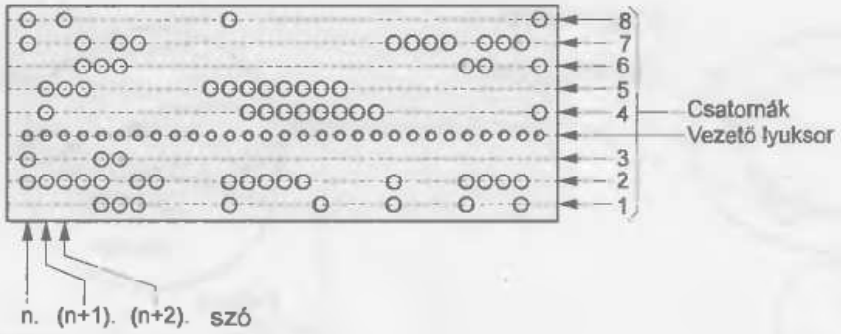
Megjegyzendő, hogy a ***RAM*** és ***ROM*** széles körben elterjedt elnevezések használata nem teljesen következetes. A *RAM* elnevezés rendszerint egy tetszőleges hozzáférésű memóriát jelöl, amely módosítható beírású. Tehát a *RAM* írható és olvasható is. A *ROM* elnevezés egy csak olvasható memóriát jelöl, amely tetszőleges hozzáférésű.

6.1. Soros hozzáférésű memóriák

A soros vagy szekvenciális hozzáférésű memória olyan típusú, amelyben az adatok csak egy adott lineáris sorrendben (*szekvenciában*) tárolhatók. Kiolvasás esetén az olvasóberendezés az elhaladó lineáris adatszekvenciából az éppen előtte levőt olvassa ki. Ezért a különböző címeken tárolt adatok hozzáférési ideje különbözik.

6.1.1. A lyukszalag

A soros hozzáférésű memóriák legklasszikusabb példája – bár jelenleg csak történeti jelentőséggel bír – a lyukszalag (*Paper tape*). Ennek az alapanyaga papír vagy műanyag. Az információ a szalag megfelelő kilyukasztásával tárolható (*6.2. ábra*:). A lyukszalag továbbítását az információhordozó lyukaknál kisebb vezető vagy továbbító lyukak biztosítják. A lyukszalagon tárolt szó a szalag hosszirányára merőleges lyuksorból áll. A szó minden egyes **1** vagy **0** értéket felvevő bitjét egy kilyukasztott, illetve lyukasztatlan hely képviseli. A kilyukasztott és a lyukasztatlan helyek kombinációja a szó kódolása szerint történik. A szalag hosszirányával párhuzamos lyuksort csatornának nevezik. Az alkalmazott kódrendszer függvényében 5–, 6–, 7– vagy 8–csatornás lyukszalag használatos.



6.2. ábra. Lyukszalag

A lyukszalagolvasó a $t=0$ időpontban az első szót olvassa ki. Ezután a szalagot egy szóval továbbítják, és $t=0+T$ időpontban a lyukszalagolvasó a második szót olvassa ki. A T időköz a szalag továbbítására és a kiolvasó erősítők előkészítésére szolgál. A folyamat lépésenkénti megismétlése a szükséges információhalmaz kiolvasását eredményezi.
A lyukszalagot a mágnesréteges memóriák teljesen kiszorították.

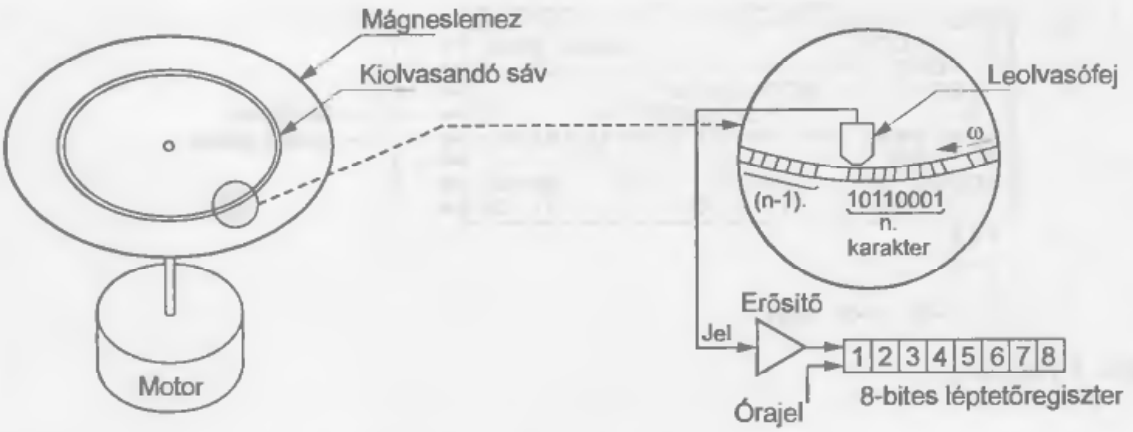
6.1.2. Mágnesréteges memóriák

A mágnesréteges memóriák tárolókapacitása messze felülmúlja a lyukszalagét (például egy hajlékony mágneslemez oldalainak egyikén tárolt 1 Mbájt egy 2,5 km hosszúságú lyukszalagot venne igénybe).

A mágnesréteges memóriák hozzáférési ideje egy-két nagyságrenddel kisebb, mint a lyukszalagé. A mágnesréteges memóriák működési elve hasonlít a magnetofonéhoz. Az írófej pólusait az információtól függő íróáram mágnesezi, amely a fej tekercsén folyik keresztül. A fej légrésében keletkező mágneses tér a mozgó tárközeg mágnesezhető rétegén maradandó mágnesezettséget alakít. ki. Ez a beírott adattól függően változó irányú, és ha elhalad az olvasófej előtt, akkor ennek a tekercsében áramimpulzust indukál. A felerősített impulzusokból alakítják ki a kiolvasott szót. A mágneses réteg hordozóanyagától függően megkülönböztethetők: mágnesszalagos, mágneskártyás, mágneslemezes és mágnesdobos memóriák. A mikroszámítógépek esetén a kazettás kivitelű mágnesszalagos (angolul: *Casette tape*), hajlékony mágneslemezes (angolul: *Floppy-disk*) és a mágneskártyás memóriák (angolul: *Magnetic card*) használatosak.

A kazettás mágnesszalagos memória hasonló felépítésű és méretű kazettás szalagot használ, mint a közismert kazettás magnetofon. A kettő között csak minőségi és tárterületi különbség van. Egy zenei felvételben a szalag minősége miatt keletkező zavarokat alig vesszük észre. Ezek viszont a tárolt információkat meghamisíthatják. Ennek elkerülésére a tárolási célokat szolgáló kazetták minősége jobb. Egyes kazettás mágnesszalagos memóriák az információt hang formájában tárolják. Vagyis egy bizonyos tónus logikai **0**-t, míg egy másik logikai **1**-et jelent. Ennek a módszernek a hátránya, hogy egy szalagon kevesebb információt lehet tárolni, mint az előbbiekben felvázolt digitális felvételi módszer alkalmazásával.

A ***hajlékony mágneslemezes memória*** esetén a mágnesezhető réteg egy gyorsan forgó műanyag lemez mindkét felére van felvive. A mágneslemez felületén koncentrikus sávok mentén tárolják az információt. Az olvasófej úgy mozgatható, hogy akármelyik sávon tárolt információt leolvashassa.



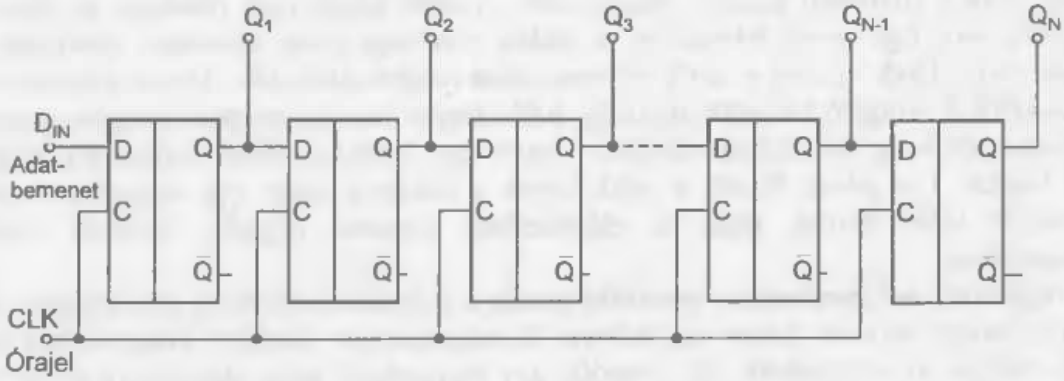
6.3 ábra. Mágneslemezes memóriaegység vázlatos felépítése

A hozzáférési idő két müvelettől függ. elsősorban a leolvasófejet kell a kívánt sávra beállítani, ezután a forgó lemezen a sáv kívánt részének kell az olvasófej elé kerülnie és elhaladnia (*6.3 ábra*). A hajlékony mágneslemez kapacitásának növelését és hozzáférési idejének csökkentését főleg a mágnesezhető réteg jellemzőinek javításával, a fej és a lemez között levő távolság csökkentésével, valamint a lemez fordulatszámának növelésével lehet elérni. Az ún. Winchester-technológiájú mágneslemez-memóriák esetén a lemezt, a fejeket, valamint az ezeket beállító mechanizmust egy közös, zárt, cserélhető kazettába építették be. Így sikerült kiküszöbölni a lemezcserénél fennálló fejbeállítási eltérést, valamint a levegőben levő por és egyéb szennyeződés lerakódását a lemezre.

A ***mágneskártyás memória*** adathordozója, amint a neve is mutatja, egy mágneses réteggel ellátott hajlékony kártya. Az információ tárolása hasonlóképpen történik, mint az előbbi mágnesréteges memóriák esetében. A tárolandó információ felvételére a mágneskártya hosszanti irányban több sávra van felosztva. A mágnesréteges memóriák nagy kapacitásuk, valamint az ehhez viszonyított alacsony áruk miatt a számítógépek háttérmemóriájának alapvető eszközei.

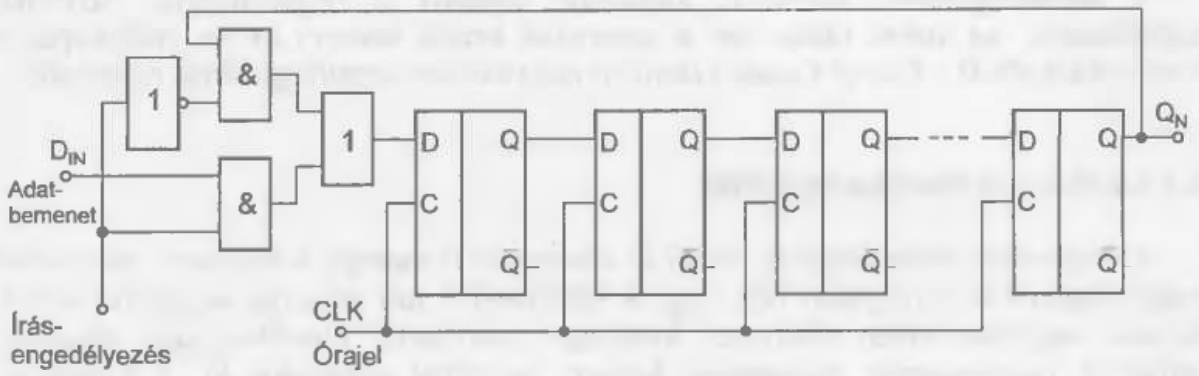
6.1.3. Léptetőregiszteres memóriák

A léptetőregiszter (angolul: *Shift register*) láncszerűen összekapcsolt flip-flopok együttese. Mindegyik flip-flop kimenete a következő bemenetére van kapcsolva (*6.4 ábra*).



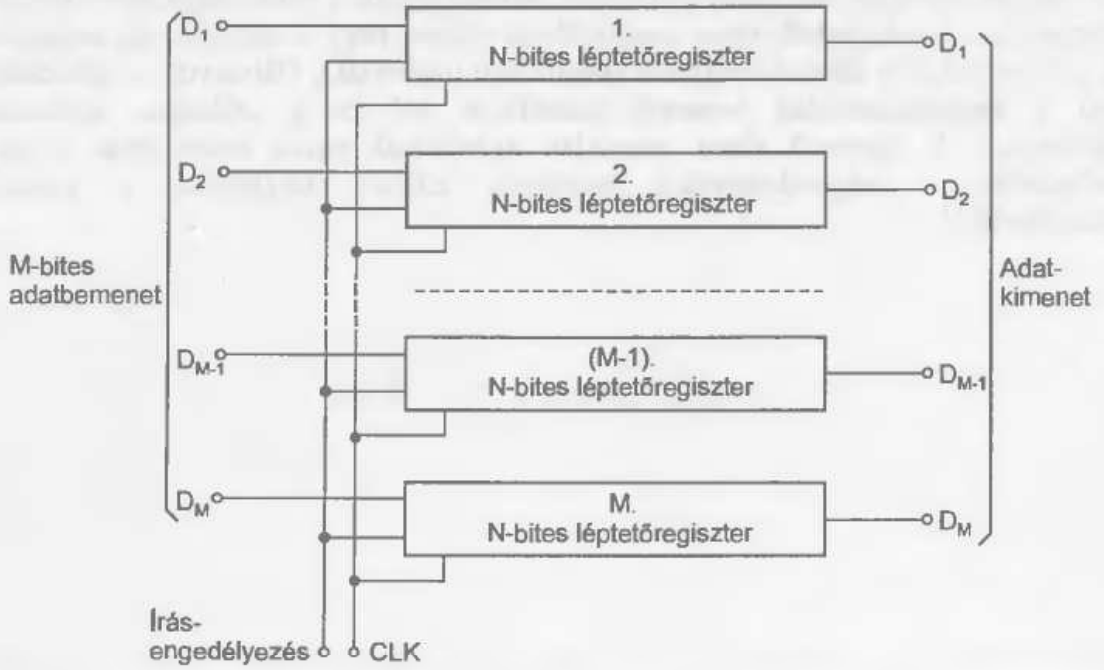
6.4 ábra. *N*-bites léptetőregiszter

Kivételt képez az első flip-flop, amelynek a bemenetére a tárolandó információt (adatot) vezetik. Az órajel, amely az összes flip-flopot egyidejűleg billenti, a tárolt adatot a regiszter kimeneti irányában (jobbra) "lépteti". Abban az esetben ha a léptetések során egy tárolt bit eljutott az utolsó flip-flopig, akkor egy újabb léptetéskor ez elvesztődik, vagyis "kilép" a regiszterből. Tárolásnál ez hátrányt jelent, amelyet gyűrűs léptetőregiszterrel (*Recirculating shift register*) lehet elkerülni. A gyűrűs léptetőregiszter egy olyan típusú léptetőregiszter, amelynek a kimenete vissza van csatolva a bemenetére. (*6.5. ábra*).



6.5. ábra. Gyűrűs léptetőregiszter

A visszacsatolást egy kapurendszer valósítja meg, de csak akkor, ha az írásengedélyező jel logikai **0**. Ellenkező esetben, vagyis ha az írásengedélyező jel **1**, akkor a kapurendszer a kimenetet leválasztja a bemenetről, hogy az utóbbira a beírandó adatokat kapcsolja. Az előbbiekben ismertetett léptetőregiszterekben a tárolt adatot csak jobbra lehet léptetni. Ha egy regisztert balra is lehet léptetni, akkor a hozzáférési idő lerövidül. A jobbra és balra is léptethető regiszter flip-flopjai nem kapcsolódnak egymáshoz közvetlenül, hanem egy-egy kapurendszer biztosítja a léptetési irány megváltoztathatóságát.



6.6. ábra. $M \times M$-bites soros léptetőregiszteres memória vázlatos felépítése

Egy N-bites léptetőregiszterben több M-bites szó tárolható sorosan. Ha $N = k \cdot M$, akkor k számú M-bites szót lehet tárolni. Ebben az esetben a tárolás szervezése bitenként és szavanként is soros. Célszerűbb M darab léptetőregisztert használni.

Ezek párhuzamosan tárolhatják a szó összes M bitjét. Így N-bites léptetőregiszterekkel N számú M-bites szó tárolható (*6.6 ábra*). Ebben az esetben a tárolás szervezése bitenként párhuzamos és szavanként soros. A léptetőregiszterek tartalmának címzését egy számláló segítségével valósítják meg. Ez a léptetőregiszterek léptetésével párhuzamosan számlál. A számláló tartalma tudósít a regiszterek állapotáról.

A léptetőregiszteres memóriák kapacitása lemarad a mágnesréteges memóriák kapacitásától. Az utóbbi időben ezt a lemaradást kezdik behozni az ún. töltéskapcsolt eszközökkel (***CCD** – **Charge Coupled Devices***) megvalósított léptetőregiszteres memóriák.

6.1.1.4. Mágnesburborékos memóriák

A mágnesburborékos memória, amint az elnevezése is mutatja, a tárolandó információt mágnesburborékok formájában őrzi meg. A körülbelül 3 μm átmérőjű mágnesburborékok állandó mágneses térben elhelyezett különleges szerkezetű, filmvékonyságú mágneses anyagban véletlenszerűen mágnesezett keskeny mezőkből alakíthatók ki. A burborékok körülbelül 15 μm távolságra vannak egymástól. Egy burborék jelenléte logikai **1**-et jelent, míg hiánya logikai **0**-t. Egy megfelelően kialakított, változtatható mágneses tér segítségével a burborékokat el lehet mozdítani a helyükből. Az adatokat képviselő mágnesburborékok egy zárt gyűrűben éppen úgy körbe léptethetők a változtatható mágneses tér segítségével, mint a gyűrűs léptetőregiszterben tárolt adatok az órajel segítségével. Ugyancsak ezzel a mágneses térrel újabb mágnesburborékokat lehet generálni, valamint a régebbieket megszüntetni. Ez megfelel a léptetőregiszterbe való adatbeírásnak, illetve törlésnek. Az adatkiolvasás, vagyis egy burborék jelenlétének kimutatása könnyen megvalósítható például a Hall-effektus segítségével.

A mágnesburborékos memóriák szervezése hasonlít a gyűrűs léptetőregiszteres memóriák szervezéséhez. A mágnesburborékos memóriáknak előnye, hogy a tápfeszültség megszűnése után sem vesztik el a tárolt információt (nemfelejtő memóriák). Hátrányuk az utóbbiakhoz képest a nagyságrendekkel hosszabb hozzáférési idő és a szükséges addicionális vezérlőlogika. A félvezető alapú memóriák gyártásának gyors technológiai fejlődése következtében a mágnesburborékos memóriák teljesen kiszorultak a gyakorlati alkalmazásokból.

6.2. Tetszőleges hozzáférésű, írható olvasható memóriák (RAM)

A tetszőleges vagy véletlenszerű hozzáférésű memóriák (RAM - *Random Acces Memory*) megkülönböztető jellemzője, mint ahogy az elnevezésük is mutatja, az, hogy akármelyik címhez azonos (nagyon rövid idő) alatt hozzá lehet férni. Az információ beírása vagy kiolvasása bármelyik címen tetszés szerint megismételhető. Más szóval a memória írható is és olvasható is (*Read-Write Memory*). A RAM áramkörök jellegzetessége még, hogy tápfeszültségük megszűnése esetén elveszítik információtartalmukat.

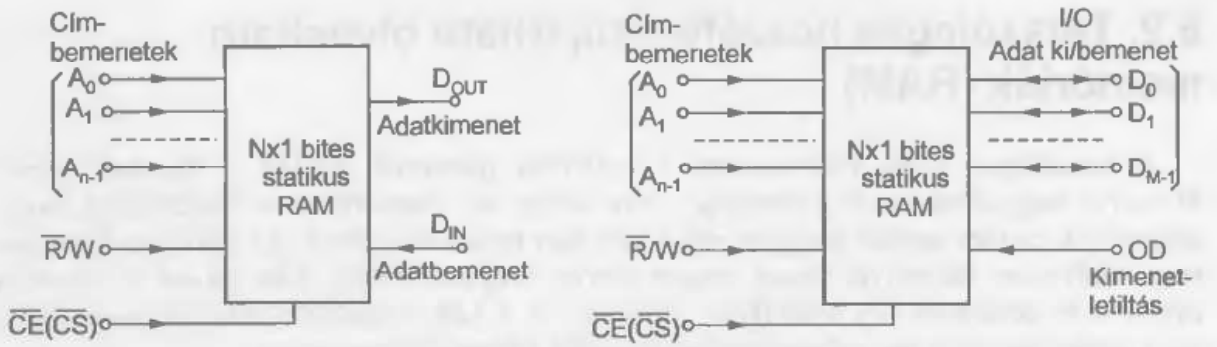
Az első RAM memóriák tároló cellája egy ferritgyűrűre épült fel. A ferritgyűrűs memória nagy hátránya, hogy a kiolvasott információ a memóriából kitörlődik. A félvezetők gyártástechnológiájának fejlődése lehetővé tette a nagy kapacitású és olcsó félvezetős RAM integrált áramkörök kifejlesztését. Előnyük az információ törlésmentes kiolvasása (a kiolvasás után a tárolt információ nem semmisül meg) és a rövid hozzáférési idő (60 nsec. nagyságrendű). Hátrányuk viszont az, hogy a tápfeszültség megszűnésekor a tárolt információt elveszítik.

Statikus és ***dinamikus*** *RAM* áramköröket különböztetünk meg. A statikus memória tároló cellája egy flip-flop, amely egy bit tárolását teszi lehetővé. A dinamikus memória tároló cellája az információt egy integrált áramköri kapacitás által tárolt elektromos töltés formájában őrzi meg. Mivel ez a töltés a kondenzátor veszteségi ellenállásán keresztül kisül, az információ elveszítését a kondenzátor időnkénti újratöltésével, vagyis az ún. felfrissítéssel kerülik el. A dinamikus memóriák cellája kevesebb tranzisztort igényel, mint a statikusoké. Ezért a dinamikus memóriák kapacitása nagyobb, mint az azonos méretű chipen előállított statikus memóriáké. Ezért a nagy kapacitású memóriákat célszerűbb a sokkal olcsóbb dinamikus *RAM* áramkörökkel felépíteni. A kis kapacitású memóriák felépítése gazdaságosabb statikus RAM áramkörökkel, mert nincs szükség az információt felfrissítő ciklusokra, illetve az azokat vezérlő áramkörökre. Annak ellenére, hogy elvileg – a felfrissítés időnkénti közbeiktatása miatt – a dinamikus *RAM* áramköröket nem lenne célszerű gyors memóriaegységek megvalósítására alkalmazni, a jelenlegi technológiával gyártott dinamikus RAM-ok sokkal kisebb hozzáférési idővel rendelkeznek (kb. 10 nsec.) mint a statikus változatok.

6.2.1. Statikus RAM áramkörök

A statikus *RAM* áramkörök tároló cellája egy flip-flopra épül fel. Beíráskor a flip-flop átveszi a tárolandó adat által meghatározott állapotot. Kiolvasáskor a flip-flop állapota az adatkimeneten jelenik meg. Egy cella egy bit tárolását teszi lehetővé. Egy *RAM* áramkör kapacitását a tárolócelláinak száma határozza meg, ugyanis a memóriában tárolható bitek száma egyenlő a cellák számával.

A *RAM* áramkörök szervezését a megcímezhető rekeszek száma és a rekeszben tárolandó szó hosszúsága határozza meg. Egy $N \times 1$ bit szervezésű *RAM* (*6.7.a ábra*) N különböző címen 1 bites szavakat tárolhat. Egy $N \times M$ bit szervezésű *RAM* (6.7.*b* ábra) N különböző címen M–bites szavakat tárolhat. Általában az M értéke 4, vagy 8 szokott lenni. A memória celláinak számát az $N \times M$ szorzat adja meg. A beírásra szánt adatok a D_{IN} adatbemenetekre kerülnek, a kiolvasott adatok pedig a D_{OUT} adatkimeneteken jelennek meg.



a) Nx1-bites RAM *b) NxM-bites RAM*

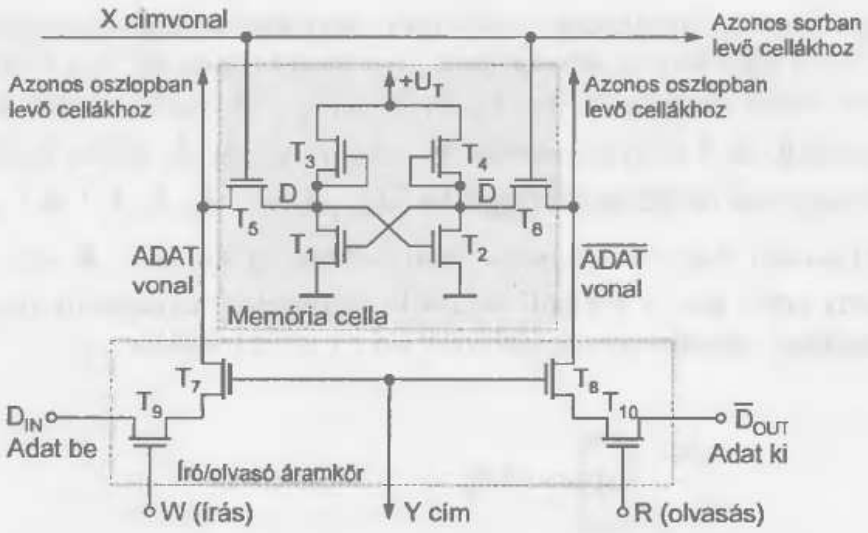
6.7. ábra. Statikus RAM árankörök szervezése

Egyes RAM típusoknál az adatbemenetek közösek az adatkimenetekkel (a *6.7.b ábra* esetében megjegyzendő, hogy nemcsak az $N \times M$ szervezésű *RAM* áramköröknél lehetnek közösek az adatbemenetek az adatkimenetekkel, hanem az $N \times 1$ szervezésű RAM áramköröknél is). A rekesz $A_{n-1}A_{n-2}....A_2A_1A_0$ címe egy bináris szám. A rekeszek száma: N határozza meg n-et, a cím bitjeinek számát, vagyis $2^n = N$.

Az adatbeírást és kiolvasást az $R/\overline{W}$ ($Read/\overline{Write}$) üzemmód-választó bemenet vezérli. Ha ez logikai **0**, akkor az adatbemeneten levő szó a cím által kijelölt rekeszbe íródik be. Ellenkezőleg: ha ez logikai **1**, akkor a cím által kijelölt rekesz tartalma az adatkimeneten jelenik meg. A kimenet-letiltó ***OD*** (***Output Disable***) bemenet segítségével az adatkimenet még a kiolvasás alatt is a harmadik, nagy impedanciájú állapotba hozható. A kimenet letiltását komplemens fogalommal is ki szokták fejezni. Ez az ***OE*** (***Output Enable***) - nincs kimenet engedélyezés. A memóriachip csak akkor írható és olvasható, ha az engedélyező bemenete ***CE*** (***Chip Enable***) vagy más elnevezéssel a kijelölő bemenete ***CS*** (***Chip Select***) logikai **0** szinten van. Ellenkező esetben, ha ez logikai **1**, akkor az adatkimenet a harmadik, nagy impedanciájú állapotba kerül, még abban az esetben is, ha $R/\overline{W} = 1$ (kiolvasás alatt is). A beírás pedig még abban az esetben is gátlás alá kerül, ha $R/\overline{W} = 0$ (beírás vezérlés alatt is). A chipengedélyező bemenet a memóriák címzését, valamint a kisebb kapacitású *RAM* integrált áramkörök nagyobb kapacitású memóriák megvalósítása céljából való összekapcsolását segíti elő.

A statikus *RAM* áramkörök az alapvető aktív építőelem típusának függvényében lehetnek bipoláris vagy *MOS* memóriák. A bipoláris memóriák valamivel gyorsabbak (rövidebb hozzáférési idejűek); mint a MOS memóriák, viszont a MOS tárolócella sokkal kisebb helyigénye miatt egy adott felületű chipen nagyobb kapacitású memória megvalósítását teszi lehetővé.

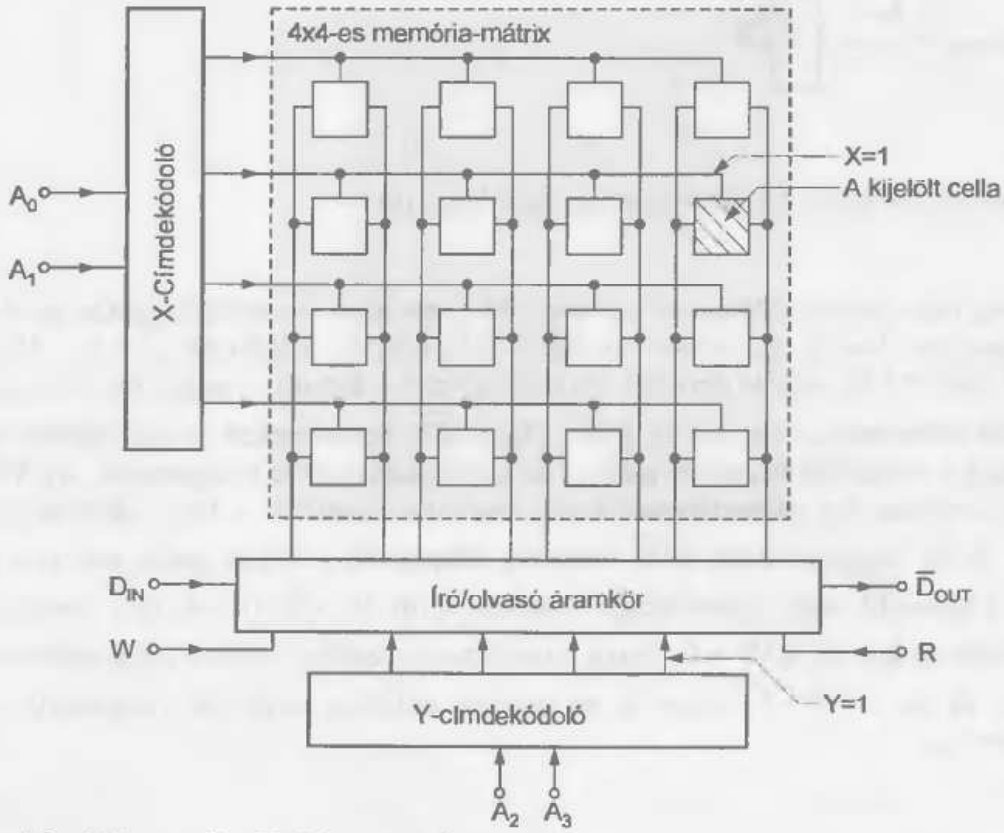
Egy tipikus statikus MOS RAM tárolócella a *6.8. ábrán* látható. A cella összesen 6 darab *N*-csatornás MOS tranzisztorból épül fel. A T_1 és T_2 keresztbekapcsolt tranzisztorok alkotják a flip-flopot. T_3 és T_4 növekményes tranzisztorok az aktív terhelőellenállás szerepét töltik be. A flip-flopot a T_5 és T_6 tranzisztorok kapcsolják az adatvonalakra. A cellákat P sorból és Q oszlopból álló mátrix alakba tömörítik. A memóriamátrix rendszerint négyzetes, vagyis $P = Q$. A mátrix celláit X sorkijelölő és Y oszlopkijelölő címvonalak segítségével választják ki. A megcímzett (kiválasztott) cella az $X = 1$ és $Y = 1$ állapotú címvonalak metszeténél található. Ha valamelyik oszlop Y címvonala logikai **1** szintre kerül, akkor a szóban forgó oszlop *ADAT* és $\overline{ADAT}$ vonalait kapcsoló T_7 és T_8 tranzisztorok vezetésbe lépnek.



6.8. ábra. Statikus MOS RAM tárolócella

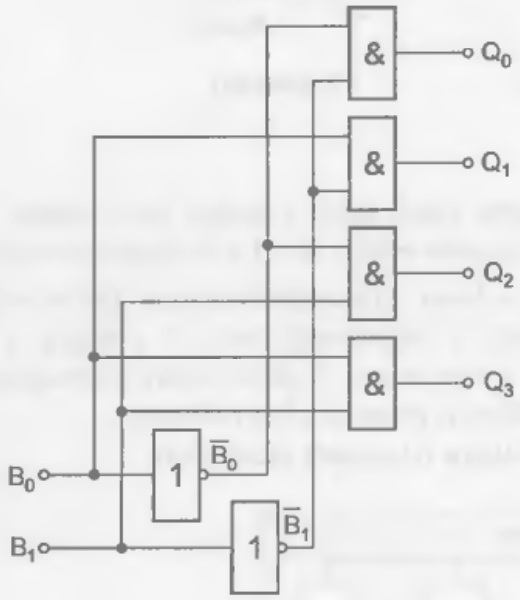
Ha valamelyik sor X címvonala logikai **1** szintre kerül, akkor a szóban forgó sorban levő cellák T_5 és T_6 tranzisztorai vezetésbe lépnek. Kiolvasás esetén, $R = 1$ a T_{10} tranzisztort hozza vezetésbe, amely a megcímzett cella $\overline{D}$ pontját a $\overline{D_{OUT}}$ kimenetre kapcsolja. Beírás esetén, $W = 1$ a T_9 tranzisztort hozza vezetésbe, amely a megcímzett cella D pontjára a D_{IN} bemenetet kapcsolja. Ez a bemenet a D pontot a vele azonos logikai szintre kényszeríti. A flip-flop a beírás megszűnése ($W = 0$) után továbbra is megőrzi a beírt állapotot.

A *6.9. ábra* egy 16×1 bites *RAM* áramkör vázlatos felépítését szemlélteti.



6.9. ábra. 16 × 1 bites statikus RAM vázlatos felépítése

Minden egyes cím a címdekódoló áramkörök segítségével egy-egy tároló cellát tesz hozzáférhetővé. A dekódolók a bemenetükön levő bináris címet az "1 a *K*-ból" (ebben az esetben $K = 4$) kóddá alakítják át. Az $X_0, X_1, X, \ldots X_{K-1}$ K számú kimenet közül csak az egyik nem logikai **0**. Ha Z jelöli a bemeneti $A_{k-1}, A_{k-2}, \ldots A_2, A_1, A_0$ k-bites bináris szám által képviselt cím decimális megfelelőjét, vagyis ha $(A_{k-1}, A_{k-2}, \ldots A_2, A_1, A_0)_2 = Z_{10}$, akkor csak $X_z = 1$. Nyilvánvaló, hogy a kimenetek száma ebben az esetben: $K = 2^k$. A 4×4-es memória mátrix cellái két "1 a 4-ből" dekódoló segítségével címezhetők meg. Egy ilyen dekódoló felépítését, valamint igazságtáblázatát a *6.10. ábra* szemlélteti.



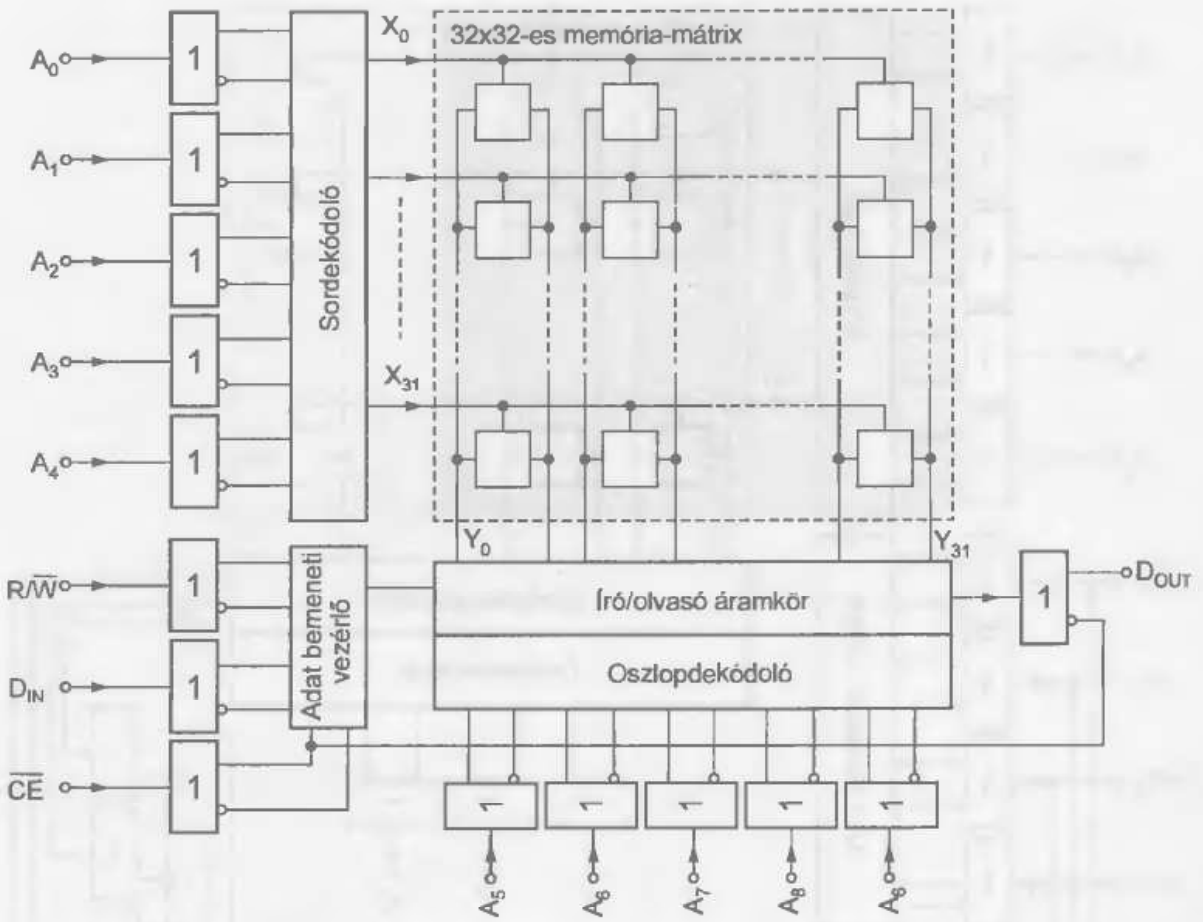
a)

Bemeneti változók		Kimenetek			
B_0	B_1	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

b)

6.10. ábra. Az „1 a 4-ből" dekódoló (a) és igazságtáblázata (b)

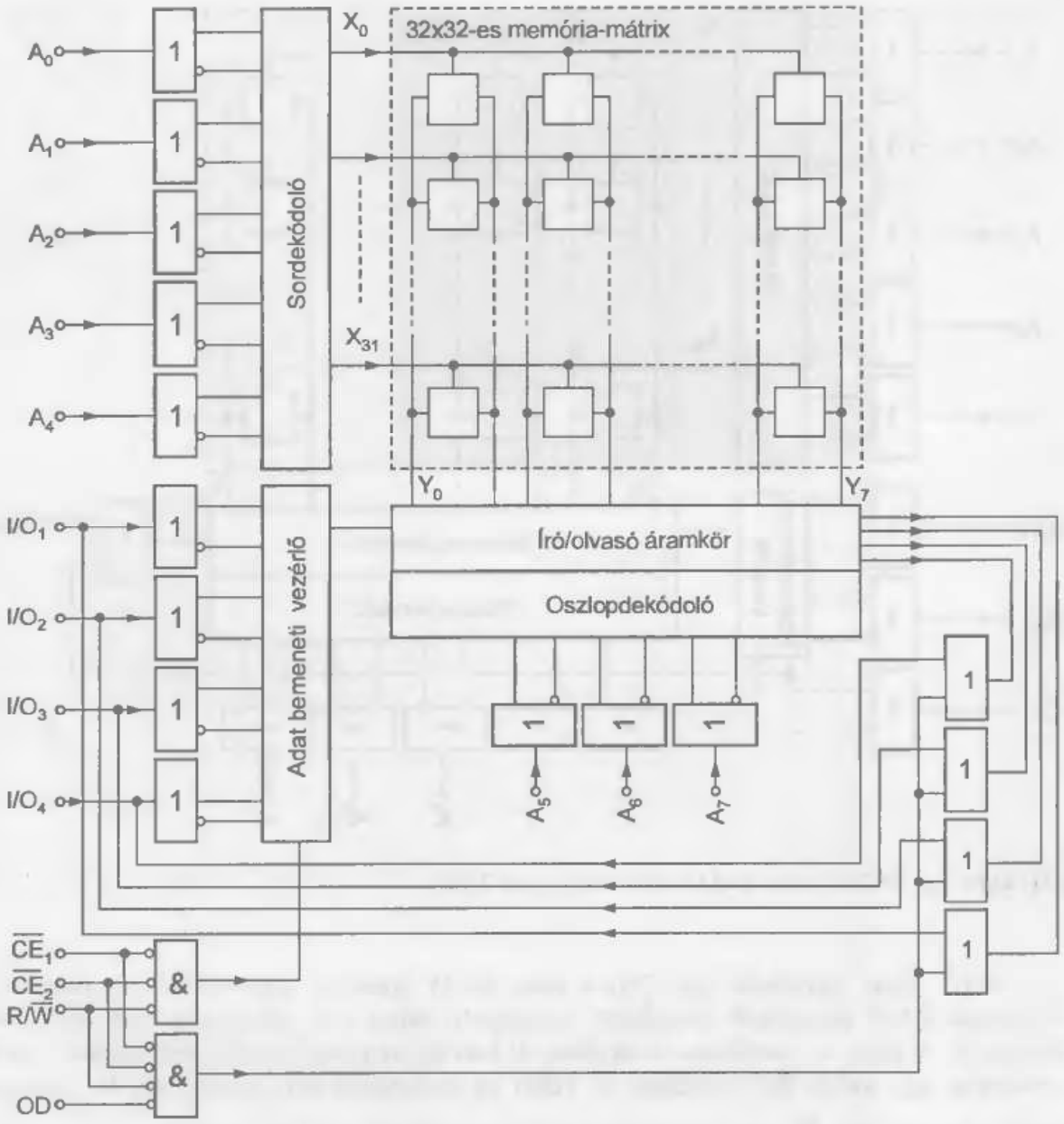
Elvileg nem sokban különbözik egy nagyobb kapacitású memória felépítése az előbbiekben bemutatott kisebb kapacitású memória felépítésétől. Példaként a *6.11. ábrán* egy 1024×1 bites RAM integrált áramkör vázlatos felépítése látható. A sordekódolók, az oszlopdekódolók bemenetein, valamint az $R/\overline{W}$, D_{IN} és $\overline{CE}$ bemeneteken levő kétkimenetű kapuk egyidejűleg a megfelelő bemeneti változót és a komplemensét is szolgáltatják. Az író/olvasó áramkör kimenete egy háromállapotú kapun keresztül csatolódik a D_{OUT} adatkimenetre. Ha $\overline{CE} = 1$, akkor függetlenül az $R/\overline{W}$ bemenet állapotától, a beírás gátlás alá, míg a D_{OUT} kimenet a harmadik, nagy impedanciájú állapotba kerül. Ha $\overline{CE} = 0$, akkor a memória írható és olvasható is. Így, ha $R/\overline{W} = 0$, akkor a megcímzett cellába beíródik a D_{IN} adatbemeneten levő bit; és ha $R/\overline{W} = 1$, akkor a megcímzett cellában tárolt bit megjelenik a D_{OUT} adatkimeneten.



6.11. ábra. Egy 1024×1 bites RAM tömbvázlata (Intel 2102)

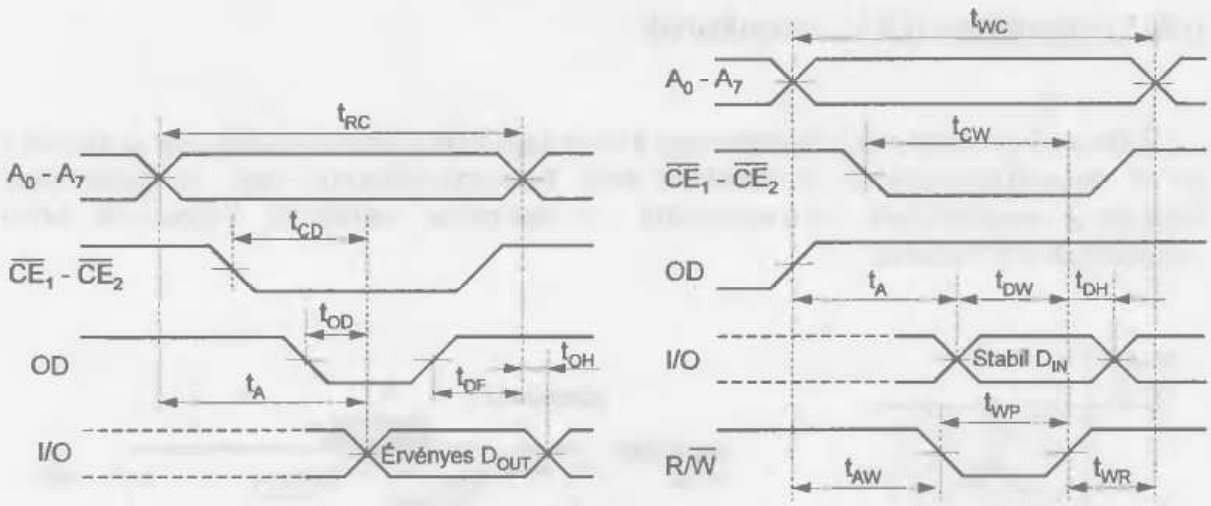
A *6.12. ábra*, példaként egy 256×4 bites RAM áramkör segítségével, az *N*×*M* bit szervezésű RAM áramkörök felépítését szemlélteti. Mivel egy tárolócella csak egy bitet tárolhat, az *M* bites szó tárolására *M* tárolócellát kell egyidejűleg hozzáférhetővé tenni. Ezek rendszerint egy sorban helyezkednek el. Tehát az oszlopdekódoló egyidejűleg *M* oszlopot jelöl ki. Vagyis, $Q'=\frac{Q}{M}$ számú oszlop-csoport alakul ki. Így az oszlopdekódoló „1 a Q'-ből" dekódoló, és egyidejűleg *M* oszlopból álló oszlopcsoportot jelöl ki. Az oszlopcsoportok és a sorok számának szorzata a memória rekeszeinek számát adja, vagyis $P \cdot Q' = N$. A 6.12. ábrán bemutatott memória esetében mivel a szóhosszúság $M = 4$, és $Q = 32$ oszlopból összesen $Q' = 8$ oszlop-csoport alakul ki. Az áramkör adatbemenetei közösek az adatkimeneteivel ($I/O_1 ... I/O_4$). A chip csak akkor engedélyezett, vagyis csak akkor írható és olvasható, ha $\overline{CE}_1 \cdot \overline{CE}_2 = 0$. Kiolvasás esetén, amikor $R/\overline{W} = 1$, az adatkimeneti háromállapotú kapuk csak akkor kapcsolják a közös adatvonalakra a megcímzett rekeszből kiolvasott 4-bites szót; ha a kimenet nincs letiltva ($OD = 0$).

Beírás esetén, amikor $R/\overline{W} = 0$, az adatkimeneti háromállapotú kapuk a harmadik, nagyimpedanciájú állapotba kerülnek.



6.12. ábra. Egy 256×4 bites RAM tömbvázlata (Intel 2111A)

A *6.13. ábra* a beírási és kiolvasási művelet idődiagramját szemlélteti. Ez a hardvertervezésnél elengedhetetlen fontosságú. Az egyszerűség kedvéért az idő-, valamint a feszültség-koordinátatengelyek ábrázolását elhanyagoltuk. Az alacsonyabb feszültségszint a „0" logikai szintet, míg a nagyobb feszültségszint az "1" logikai szintet képviseli (pozitív logika). A harmadik állapotot szaggatott vonal ábrázolja. A két logikai szint fele távolságán futó folytonos vonallal is szokták ábrázolni. Az átkapcsolási időket nem lehet nullával egyenlőnek venni, mint egyes ideális esetekben. Az átkapcsolási görbéket egyenessel közelítettük meg. A jelalakok funkcionális fontosságú egymáshoz való viszonyulását megjelöltük. Értékük a memória adatlapjaiban megtalálható.



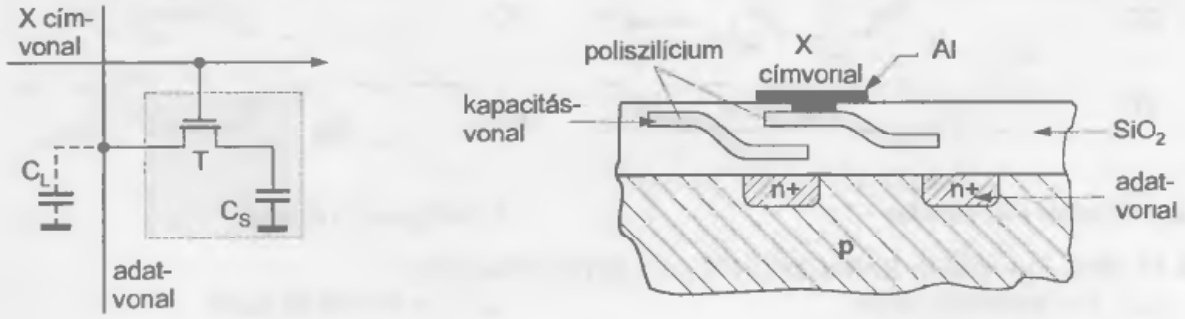
a) jellegzetes olvasóciklus *b) jellegzetes íróciklus*

6.13. ábra. Egy statikus RAM (Intel 2102) jellegzetes idődiagramjai

t_{RC} – adat kiolvasási ciklus
t_A – hozzáférési idő
t_{CO} – chip-engedélyezéstől az adatkimenetig
t_{OD} – kimenettiltástól az adatkimenet harmadik állapotáig
t_{OH} – előzőleg kiolvasott adat a címváltozás utáni fennmaradása
t_{WC} – az adatbeírási ciklus
t_{AW} – íráseltolás
t_{CW} – chip-engedélyezéstől a beírásig
t_{DW} – adat előzetes beállítása
t_{DH} – adat fenntartása
t_{WP} – beíróimpulzus
t_{WR} – beírás feléledési ideje
t_{DS} – kimenetletiltás előzetes beállítása

6.2.2 Dinamikus RAM áramkörök

A dinamikus memóriák jellegzetessége a nagy kapacitás és ehhez viszonyítva az alacsony ár. A dinamikus memória tárolócellája lehet háromtranzisztoros vagy az újabb nagy kapacitású memóriákban egytranzisztoros. A dinamikus memóriák *N*-csatornás MOS technológiával készülnek.

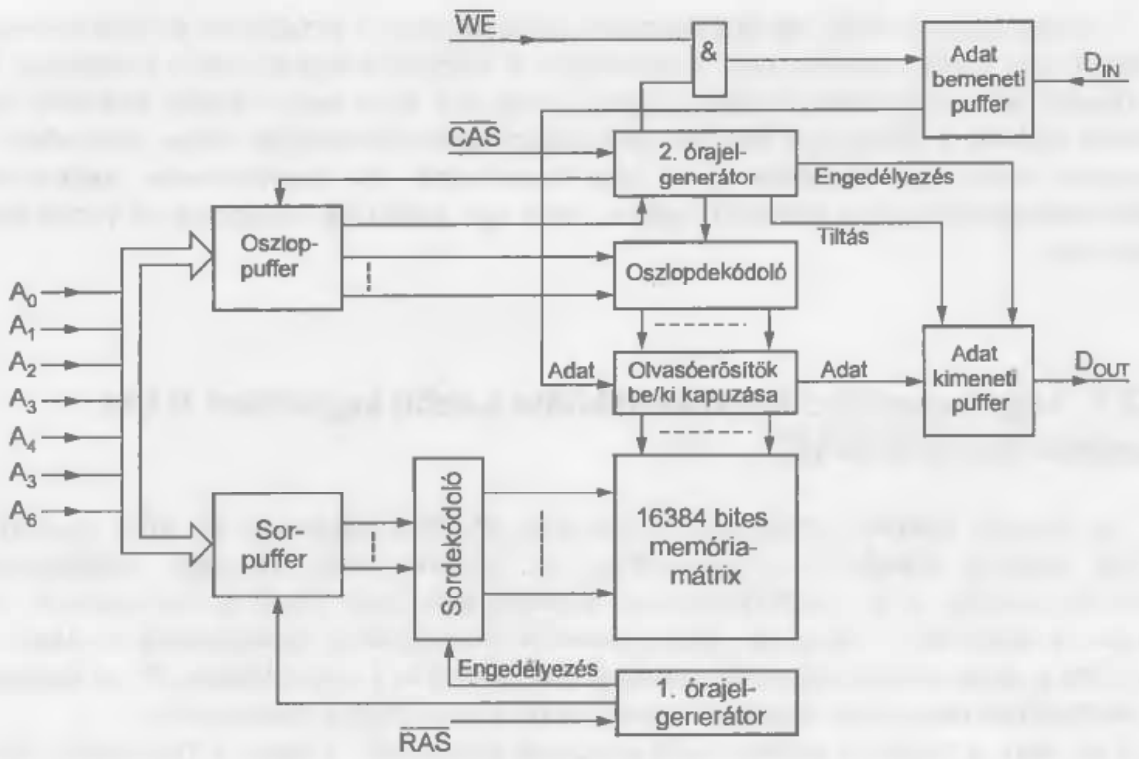


a) *b)*

6.14. ábra. Egytranzisztoros dinamikus RAM tárolócella (a) és integrált áramköri metszete (b)

Egy egytranzisztoros tárolócellát mutat be a *6.14. ábra*. Az információtároló szerepét a C_S kondenzátor látja el. Ezt az ***X*** címvonallal vezérelt *T* tranzisztor kapcsolja az oszlop adatvonalára, amelynek a végére a kiolvasó erősítő kapcsolódik. A feltöltött C_S kondenzátor az erősítő bemenetén a $\frac{C_S}{C_L}$ törttel arányos jelet szolgáltat, ahol C_L az adatvonal kapacitása, és arányos a memória-mátrix felületével. A nagy kapacitású memóriák esetén C_L nagyobb, mint C_S, és ezért az olvasó-erősítő bemenetén csak 10.....100 mV nagyságú jel jelenik meg. Ennek a kis amplitúdójú jelnek az erősítését differenciálerősítővel végzik. Abban az esetben, ha a kapacitásvonal pozitív feszültséget kap, ennek hatására a *P*-szubsztrátban N^+ inverziós-vonal alakul ki. Ez tárolja az információt, és egyben a tranzisztor source-át is képviseli. A címvonalra kapcsolt jel hatására a source és drain között vezetőcsatorna alakul ki. A drain egyben adatvonal is. A bináris információ másik állapotában a kapacitásvonal nem kap pozitív feszültséget, és ezért nem alakul ki vezetőcsatorna.

A C_S tárolókondenzátor idővel veszít a töltéséből. A tárolt információ elvesztését a kondenzátor újratöltésével akadályozzák meg. Ezt a folyamatot felfrissítésnek nevezik, és periodikusan (rendszerint 2 ms-ként) meg kell ismételni. A felfrissítés tulajdonképpen. a tárolt információ kiolvasása és az azonos cellákba való újraírása. Ezt a folyamatot a dinamikus memória felfrissítő erősítői könnyítik meg. A memóriamátrix minden egyes oszlopa egy-egy felfrissítő erősítővel van ellátva. A memóriamátrix egy sorának a megcímzésével az egész sor automatikusan felfrissül. Az egész memória felfrissítése az összes *P* sor felfrissítéséből áll. Míg a statikus *RAM* áramkörök felépítése típusonként kevésbé eltérő, a dinamikus *RAM* áramkörök felépítése a felfrissítési művelet különböző kivitelezése miatt sokkal eltérőbb.



6.15. ábra. Egy 16 kbites dinamikus RAM tömbvázlata (Intel 2116)

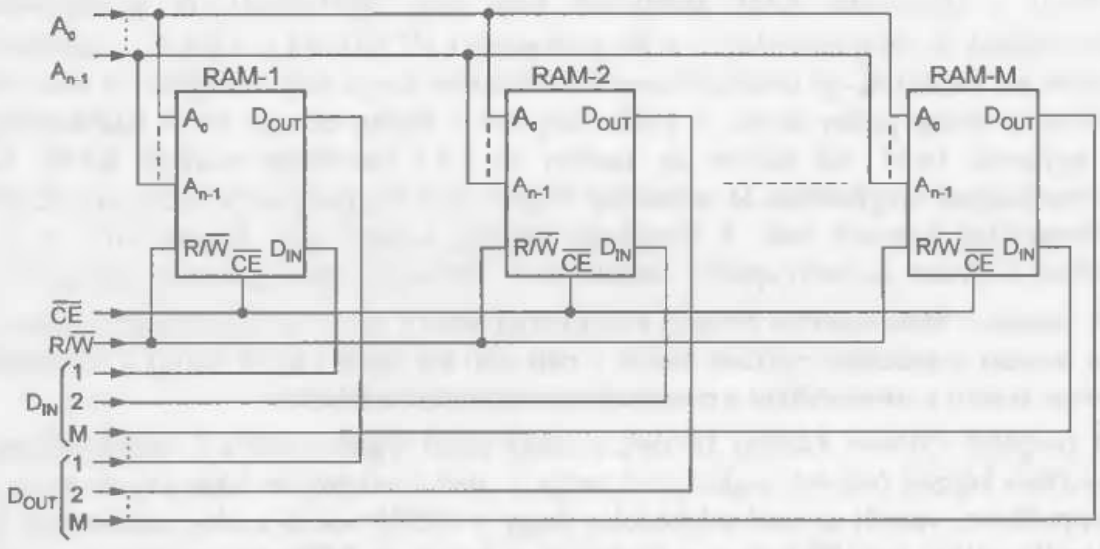
A *6.15. ábra* egy 16 kbites (16384×1 *bit*) dinamikus *RAM* tömbvázlatát szemlélteti (jellemző a dinamikus *RAM* áramkörök elég nagy hányadára). A memóriamátrix tárolócelláinak a megcímzésére 1 k bit szükséges ($2^{14} = 16384$). Ezeket a memória az összesen hét (A_0-tól A_6-ig) címbemeneten két részletben kapja meg. Az egész 14 bites címet a sor- és az oszlop-puffer tárolja. A puffer (angolul: – *buffer, storage latch*) tulajdonképpen egy egyszerű $1 \times M$ bit (ebben az esetben $M = 7$) szervezésű statikus RAM. Ez a szóhosszúságnak megfelelően M számú flip-flopból épül fel, melyeknek közös órajele beírja a bemenetükre kapcsolt szót. A következő órajelig a flip-flopok kimeneteiről ez a szó olvasható le. A sor- és oszlop-puffer órajeleit az 1. illetve a 2. órajel-generátor szolgáltatja. A $\overline{RAS}$ (angolul: ***Row Address Strobe***) külső órajel vezérli az 1. Órajel-generátort, amely az általa képzett órajelekkel egyrészt beírja a cím első hét bitjét (A_0-tól A_6-ig) a sorpufferbe, másrészt vezérli a sordekódolót a megfelelő sor kijelölése érdekében.

$\overline{CAS}$ (angolul: ***Column Address Strobe***), a másik külső órajel vezérli a 2. órajel-generátort. Ez az általa képzett órajelek segítségével beírja a cím következő hét bitjét (A_7-től A_{14}-ig) az oszloppufferbe, vezérli az oszlopdekódolót (hogy a kijelölt sor és oszlop metszeténél levő tárolócella váljon hozzáférhetővé a kiolvasás, a beírás és felfrissítés számára), és végül vezérli az adatkimeneti, valamint az adatbemeneti puffereket is. A beírást vagy a kiolvasást a $\overline{WE}$ (***Write Enable***) beírásengedélyező bemenet vezérli. A kiolvasást $\overline{WE} = 1$ határozza meg, míg a beírást $\overline{WE} = 0$. A $\overline{CAS}$ és $\overline{RAS}$ órajelek megfelelő együttese egy megcímzett sor felfrissítését teszi lehetővé. Az egész memóriamátrix felfrissítése az összes 128 sor felfrissítéséből áll, amelyet legritkábban minden 2 ms-ként kell elvégezni. A dinamikus *RAM* különböző vezérlőjelei között levő bonyolult időbeni összefüggést, amelynek a betartása elengedhetetlen a hardvertervezésnél, az adatlapok tanulmányozásával lehet részletesen megismerni.

A dinamikus memóriák egyedüli hátránya, hogy időnként a tartalmukat fel kell frissíteni. A felfrissítés alatt a memória. nem hozzáférhető. A felfrissítés interferálhat a számítógép író és olvasó ciklusaival, ekkor a számítógépnek várnia kell. Ezt a nemkívánatos időkiesést úgy kerülik el, hogy a felfrissítést azokban az időintervallumokban hajtják végre, amelyekben a memória nincs sem kiolvasás alatt, sem beírás alatt. Ez természetesen komplexebb felfrissítésvezérlő logikai áramkört igényel, mint egy szabályos időközönként végrehajtott felfrissítés.

6.2.3. Nagy kapacitású RAM kialakítása kisebb kapacitású RAM integrált áramkörökből

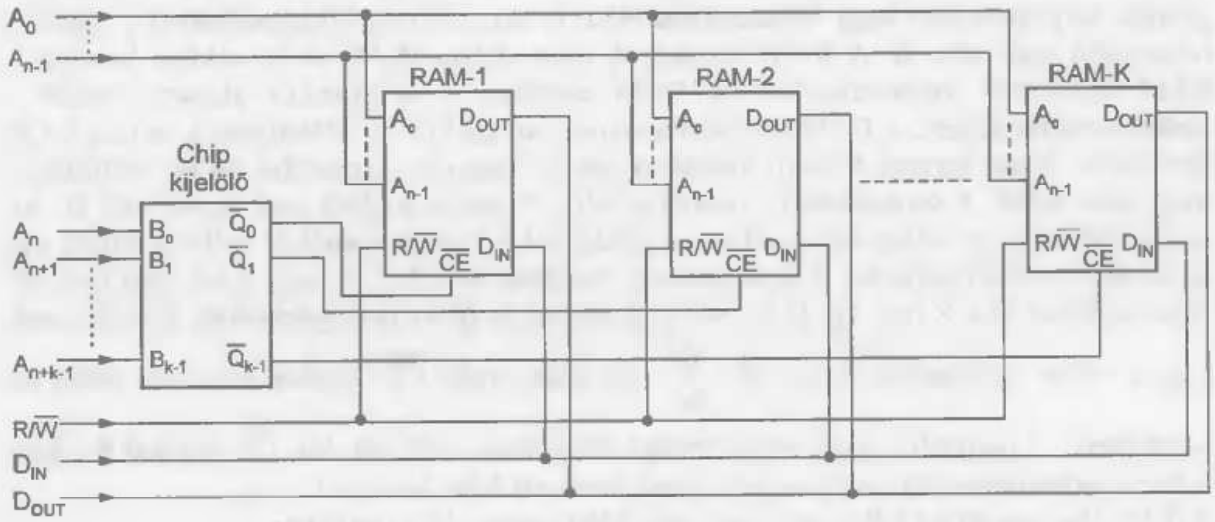
Az integrált áramköri gyártástechnológia még 16 Mbit statikus és 64 Mbit dinamikus RAM integrált áramkörök megvalósítását is lehetővé teszi. A nagy teljesítményű mikroszámítógép *RAM* memóriaegységei kapacitásának még ennél is nagyobbnak kell lennie. A kapacitás növelését egyrészt a memória által tárolható szóhosszúság növelésével, másrészt a megcímezhető rekeszek számának növelésével kell megvalósítani. A két módszert az alábbiakban ismertetjük. Ezeket a legtöbb esetben kombináltan alkalmazzák.
A 6.16. ábra a tárolható szóhosszúság növelését szemlélteti. Legyen a felhasznált RAM áramkör $N \times 1$ bit szervezésű. Ha a szó hosszúsága M bit, akkor M számú RAM áramkör szükséges az $N \times M$ bit szervezésű RAM egység megvalósításához.



6.16. ábra. M számú $N \times 1$-bites RAM integrált áramkörből megvalósított $N \times M$-bit szervezésű RAM memóriaegység

A címbemenetek és a chip engedélyező bemenetek párhuzamos kapcsolása folytán a címek bármelyikén az első RAM áramkör tárolja a szó első bitjét, a második RAM áramkör a szó második bitjét, és így tovább az M. RAM áramkörig. Például egy 1024×8 bit szervezésű RAM egységet nyolc 1024×1 bites RAM áramkörből vagy két 1024×4 bites RAM áramkörből lehet kialakítani.

A *6.17. ábra* szemlélteti a kapacitás növelését egy adott szóhosszúság mellett. Az alapáramkör ugyancsak egy $N \times 1$ bit szervezésű RAM. Az N rekesz megcímzését megvalósító $A_0, A_1, \ldots A_{n-2}$ és A_{n-1} címbemenetek párhuzamosan kapcsolódnak.



6.17. ábra. K számú $N \times 1$ bites RAM integrált áramkörből megvalósított $KN \times 1$-bit szervezésű RAM memóriaegység

Cím				*Chip-engedélyezés*			
A_3	A_2	A_1	A_0	CE_1	CE_2	CE_3	CE_4
0	0	0	0	0	1	1	1
0	0	0	1	0	1	1	1
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	1
0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	1	1	0	1
1	0	0	1	1	1	0	1
1	0	1	0	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	0	1	1	1	0
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0

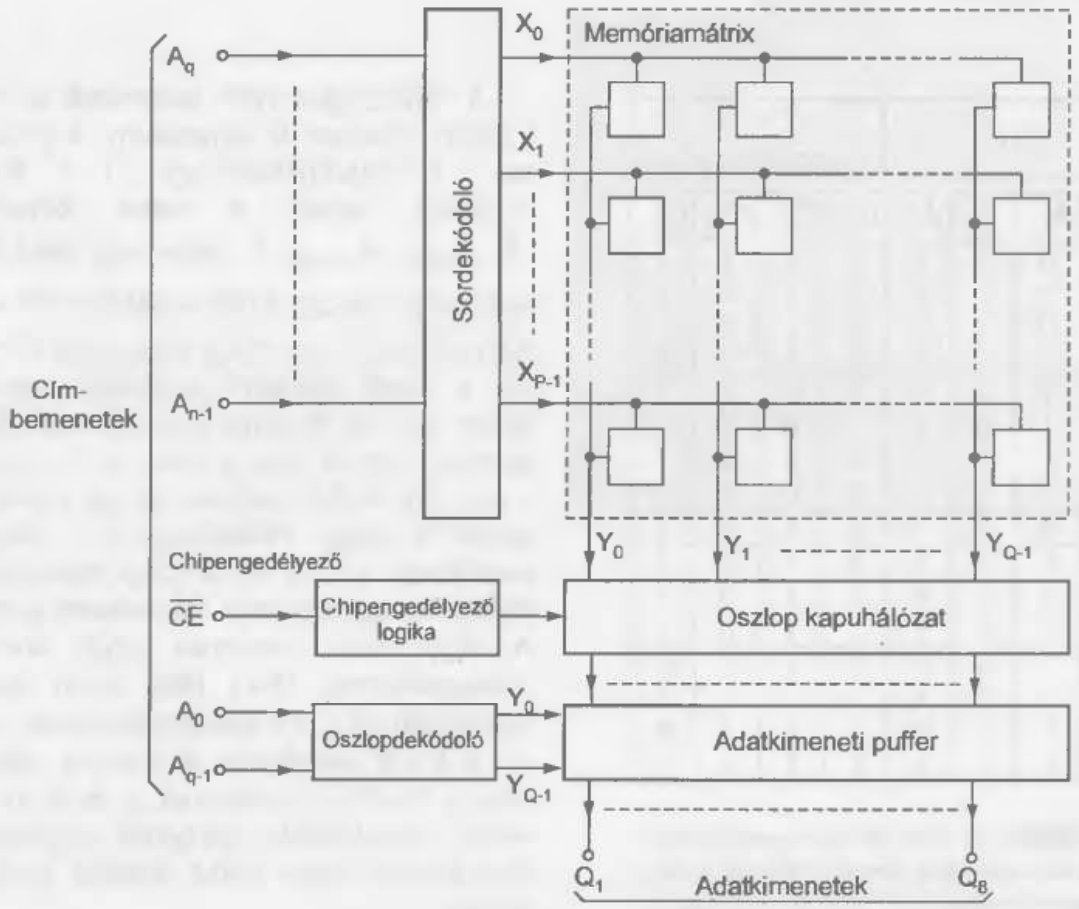
6.1. táblázat. A cím- és chipengedélyezés nagyobb kapacitású memória kisebb kapacitású memóriákból való felépítésénél egy feltételezett esetben

A chip-engedélyező bemenetek a chip-kijelölő áramkör K kimeneteire kapcsolódnak. A chip-kijelölő egy „1 a K-ból" dekódoló, amely a soron következő $A_n, A_{n+1}, \ldots A_{n+(k-1)}$ k címvonalat dekódolja, és mindig csak egy RAM áramkört jelöl ki.

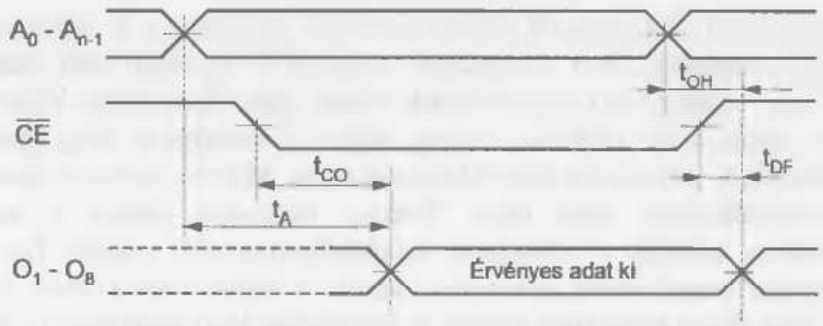
Nyilvánvaló, hogy ebben az esetben $K = 2^k$. Ha a címek növekvő sorrendjét vesszük, akkor az első N cím, csak az első RAM áramkört címezi meg, a második N cím csak a második RAM áramkört, és így tovább az utolsó N címig. Példaként a *6.1. táblázat* szemlélteti a cím és a chip-engedélyezés alakulását egy egyszerű feltételezett esetben. A négy darab 4×1bites RAM áramkör felhasználásával 16×1 bites RAM egység valósítható meg. Az első két címvonal, A_0 és A_1, a RAM áramkörök 4 rekeszét címezik meg. A további két címvonal, A_2 és A_3 az „1 a 4-ből" chip-kijelölő dekódoló segítségével engedélyezi a négy RAM áramkör közül az egyiket.

6.3. Csak olvasható memóriák (ROM)

A csak olvasható memóriákat (***ROM – Read Only Memory***) fix memóriáknak is nevezik. A bennük rögzített információt csak kiolvasni lehet. A kiolvasás szempontjából a ROM memóriák véletlenszerű hozzáférésűek. Az információ rögzítése történhet a memória gyártási folyamata alatt vagy felhasználása előtt. A tápfeszültség kikapcsolásával a rögzített információ nem vész el. A ROM áramkörök tömbvázlata (*6.18. ábra*) sokban hasonlít a RAM áramkörök tömbvázlatához. A ROM esetében is az áramkör alapvető részét a memóriamátrix képezi. A ROM tárolócella viszont sokkal kisebb felületigényű, mint a RAM tárolócella. Ezért azonos felületű kristályra sokkal nagyobb kapacitású ROM valósítható meg, mint RAM. A sordekódoló a memóriamátrix P számú sorából csak egyet jelöl ki. Az oszlopdekódoló az oszlop-kapurendszer segítségével a kijelölt sorból M cella tartalmát írja be az adatkimeneti pufferbe. A szóhosszúság általában $M = 1, 2, 4$ vagy 8 bit lehet (a *6.18. ábra* esetében $M = 8$ bit). Ha Q az oszlopok száma és Q' az oszlopdekódoló kimeneteinek száma, akkor nyilvánvaló, hogy $Q' = \frac{Q}{M}$. Az alatt, amíg $\overline{CE}$ logikai **1**-et kap, addig az adatkimenet a harmadik, nagy impedanciájú állapotban található. Ha $\overline{CE}$ logikai **0**-t kap, akkor az adatkimenetről a megcímzett rekesz tartalmát lehet leolvasni.
A *6.19. ábra* egy ROM jellegzetes kiolvasási idődiagrammját szemlélteti.



6.18. ábra: Csak olvasható memória (*ROM*) tömbvázlata



6.19. ábra: Egy ROM jellegzetes olvasóciklusa

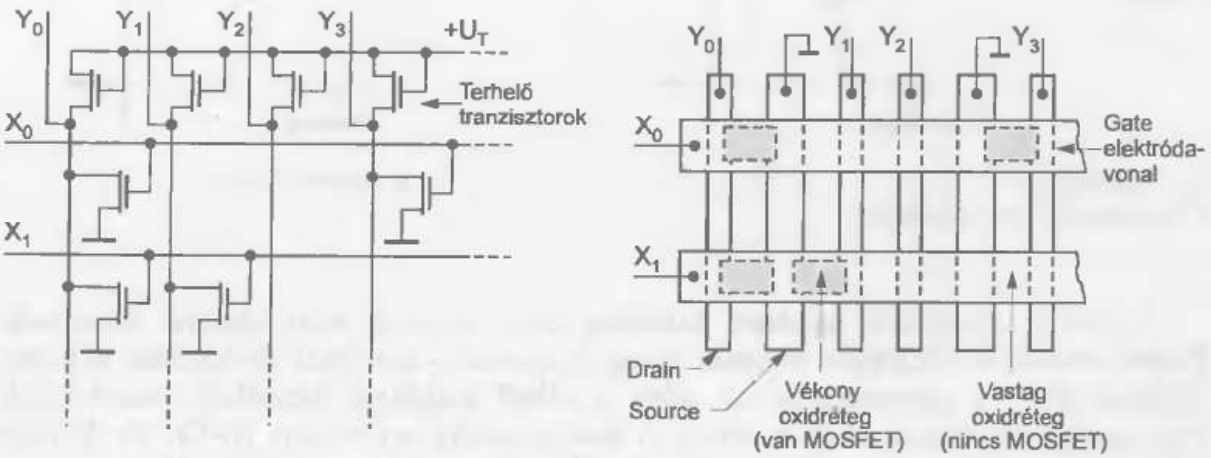
A ROM memóriákat főként a mikroszámítógépes célrendszerek fix programjának tárolására használják. Egyéb tipikus ROM alkalmazások: a különböző matematikai függvények előállítására szolgáló függvénytáblázatok, a különböző szabványos kódokhoz használatos kódgenerátorok, valamint a megjelenítőkhöz szükséges alfanumerikus karaktereket előállító karaktergenerátorok.

A ROM memóriák típusai, amelyek az alábbiakban kerülnek részletesebb bemutatásra, az információ rögzítési módjára utalnak.

6.3.1. Maszkprogramozott ROM áramkörök

A maszkprogramozott ROM áramkörökbe az információt a gyártás során programozzák be, és ez utólag már nem változtatható. A programozást a gyártástechnológiai folyamatban felhasznált maszkok egyikével valósítják meg. Innen származik a *„maszkprogramozott"* elnevezésük is. Nagy felhasználási sorozatok esetén a maszkprogramozott ROM alkalmazása a leggazdaságosabb.

A *6.20. ábra* egy tipikus *MOS* ROM memóriamátrix szerkezetét, valamint fizikai elrendezését szemlélteti felülnézetben.



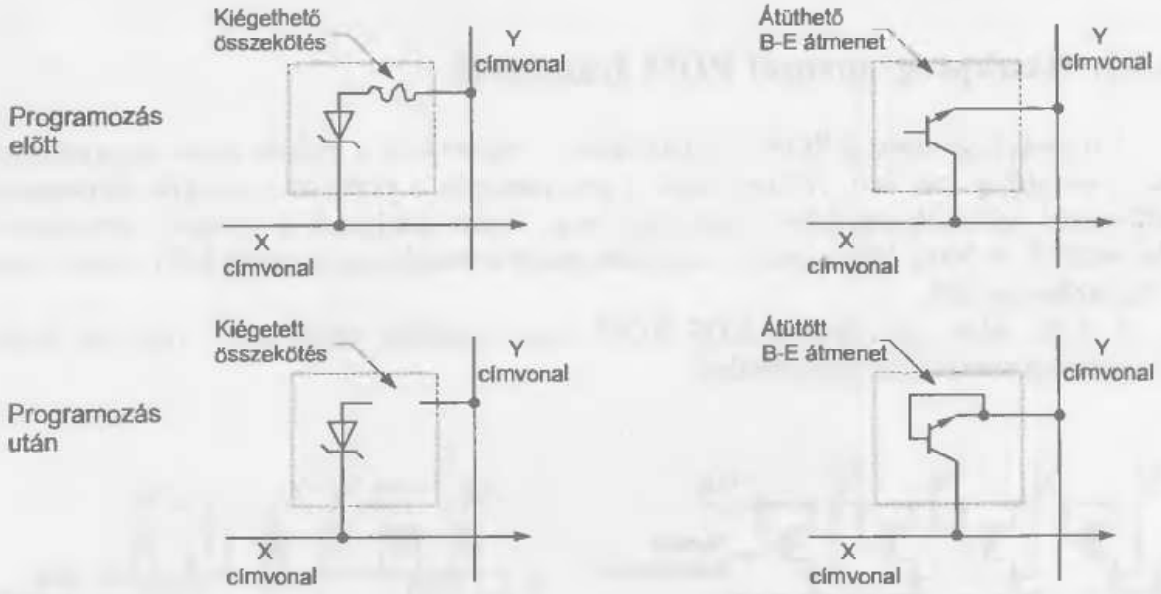
a) áramköri részlet *b) integrált áramköri kivitel*

6.20. ábra: MOS ROM memóriamátrix egy részlete

Egy oszlopban levő tranzisztorok drainjei közösek, egyben az *Y* címvonalat a képezik. Két szomszédos oszlopban levő tranzisztor source-a is közösen van kialakítva. Az *X* címvonalat az egy sorban levő tranzisztorok közös gate elektródája képezi. Ha a cella csatornafelülete fölött lévő oxidréteg vastag, akkor a tranzisztor U_{TO} küszöbfeszültsége meghaladja a kijelölt *X* címvonalra kapcsolt feszültséget. Mivel a csatorna nem jöhet létre, ez a hely tranzisztorhiánynak felel meg. Vékony oxidréteg esetén a tranzisztor U_{TO} küszöbfeszültsége a kijelölt *X* címvonal feszültségszintjénél kisebb. Így a kijelölt X címvonalra kapcsolt tranzisztorok vezetésbe jönnek. A többi, nem kijelölt *X* címvonalakra kapcsolt tranzisztor lezárt állapotban marad. A vezetésben levő tranzisztorok drainjei – tehát az ezeknek megfelelő *Y* címvonalak is – közel földpotenciálon vannak. A terhelőtranzisztorok a többi címvonal feszültségét közel $+U_T$ értéken tartják.

6.3.2. Felhasználáskor programozható ROM áramkörök (PROM)

Ez a típusú ROM rendeltetésének megfelelően a felhasználás előtt programozható. Rövidített jelölése ***PROM*** (**P**rogrammable **ROM**). Kis felhasználási sorozatok esetén alkalmazható előnyösen.



a) „kiégethető" cella *b) „átüthető" cella*

6.21. ábra. PROM tárolócellák

A PROM áramkörök majdnem kizárólag csak bipoláris technológiával készülnek. Programozásuk kétféleképpen történik. Az egyik típusnak – amelynek tárolócellája egyetlen diódából áll – a programozása az ebből a célból kialakított kiégethető összekötések megszakításával történik (*6.21.a. ábra.*). A diódák anódja egy vékony Ni–Cr, Ti–W vagy polikristályos szilícium rétegből készülő ellenállásszakasszal kapcsolódik az Y címvonalra. Ha az ezen átfolyó áramsűrűség értéke meghalad egy kritikus szintet, akkor az anyag hirtelen megolvad, és szakadás áll be. Az újabb áramkörök általában Schottky-diódákkal készülnek, amelyeknek előnye a kis hozzáférési idő.

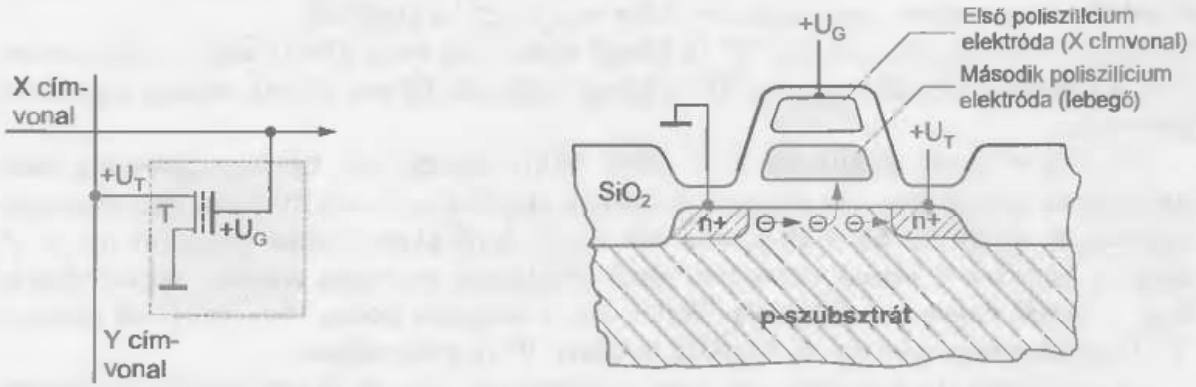
A másik típusnak a cellája egyetlen tranzisztorból áll, amelynek a bázisa nincs bekötve (*6.21.b. ábra*). A programozás előtt az emitter és kollektor között nem folyhat áram. Programozáskor a cellára kisülő feszültség meghaladja a bázis-emitter átmenet letörési szintjét. Az átmeneten nagy áram jön létre, melynek hatására tartós rövidzár keletkezik az emittervezeték és a bázis között. Így az *Y* címvonalra kapcsolt bázis a bázis-kollektor átmenetet képviselő dióda anódja.

Minden egyes PROM áramkörtípus programozási eljárását az adatlapjai részletesen közlik. Az adatok ún. „beégetése" a külön ezt a célt szolgáló automatikus programozókészülékkel történik.

6.3.3. Újraprogramozható ROM áramkörök

Az újraprogramozható ROM egyedi berendezésekben, prototípusokban kerül alkalmazásra, és ott, ahol a programot meg kell tudni változtatni. Rövidített jelölése EPROM vagy REPROM (**E**rasable, illetve **R**eprogramable **PROM**).

Az adatok beégetése a megfelelő programozókészülékkel felhasználás előtt végezhető el. A memória tartalma egészben, egyszerre törölhető. Az egyik legelterjedtebb EPROM típus az, amelynél a törlés ultraibolya (UV) sugárral történik (angolul: UV Erasable PROM). Egy másik, újabb típusnál a törlés is, mint a programozás, elektromosan végezhető. Ezeknek a rövidített jelölése EEPROM (**E**lectrically **E**rasable **PROM**) vagy EAROM (**E**lectrically **A**lterable **ROM**).

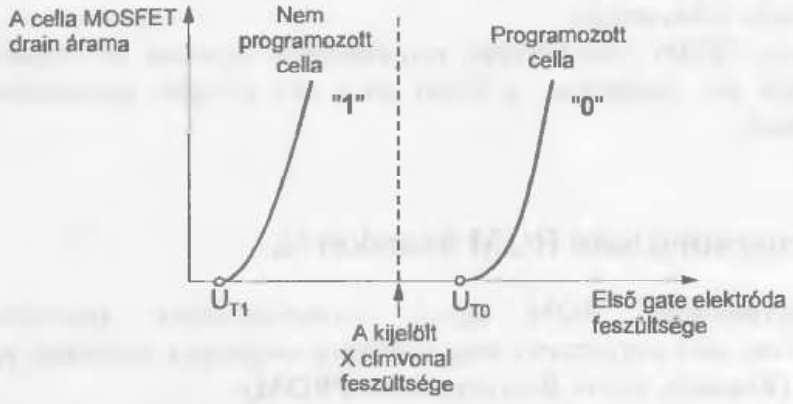


a) tárolócella - FAMOS cella *b) integrált áramköri keresztmetszete*

6.22. ábra. Ultraibolya sugárral törölhető és újraprogramozható ROM (UV EPROM)

A *6.22. ábra* egy ultraibolya sugárral törölhető EPROM tárolócelláját mutatja be. A cella egyetlen lebegő vezérlőelektródájú, lavinainjektálással működő MOS (FAMOS - **F**loating-gate **A**valanche-injection **MOS**) tranzisztorból áll. A polikristályos szilíciumból levő lebegő elektróda a jól szigetelő oxidrétegben minden oldalról körülvéve „lebeg", vagyis nem kapcsolódik az áramkör többi részéhez. A tranzisztor tulajdonképpeni vezérlőelektródája a felső, *X* címvonalra kapcsolt elektróda. A programozás a lebegő elektróda lavinainjektálásával történik. A nagy térerősség hatására a csatornában mozgó elektronok lavinaszerűen átlépik a szilícium-szubsztrát és az oxidréteg-átmenet energiaküszöbét, és eljutnak a lebegő elektródához.

A kiváló szigetelés miatt a lebegő elektródára került töltés nagyon hosszú ideig megmarad (70°C-on 100 év alatt kb. 5%-os töltésveszteség). A lebegő elektródán levő negatív töltés hatására a tranzisztor küszöbfeszültsége megnő (*6.23. ábra*).



6.23. ábra. A FAMOS cella tranzisztorának átviteli jelleggörbéje

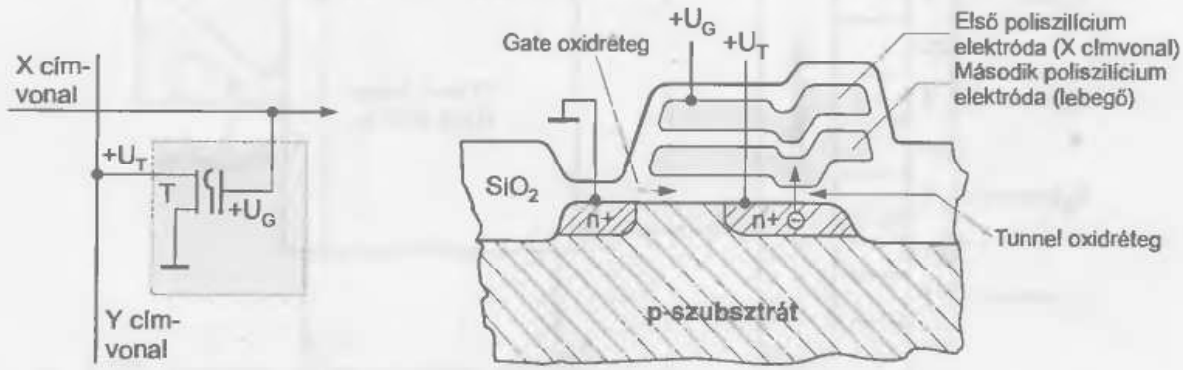
A nem programozott cella tranzisztorának U_{T1} küszöbfeszültsége a kijelölt *X* címvonal feszültségénél kisebb, de a nem kijelölt *X* címvonalak feszültségénél nagyobb. Ezért csak azok a nem programozott tranzisztorok vezetnek, amelyek a kijelölt *X* címvonalon vannak. A programozott cella tranzisztorának U_{T0} küszöbfeszültsége meghaladja a kijelölt *X* címvonal feszültségszintjét. Ezért ez a tranzisztor akkor sem vezet, ha kijelölték.
Tehát, ha a kijelölt cella tartalma "**1**" (a lebegő elektródán nincs töltés), akkor a tranzisztora vezet; és ha a kijelölt cella tartalma "**0**" (a lebegő elektróda fel van töltve), akkor a tranzisztor nem vezet.

A törlés a lebegő elektródán tárolt töltés eltávolításából áll. Ez fotoelektromos hatás segítségével valósul meg. Az áramkört közvetlen ultraibolya ($\lambda = 0{,}2537$ μm hullámhosszú) sugárzásnak teszik ki. Ez a chiphez a tok elején levő kvarcablakon keresztül jut el. A sugárzás hatására a lebegő elektródán tárolt elektronok energiája annyira megnövekszik, hogy el tudják hagyni az elektródát. Törlés után a memória összes tárolócelláinak tartalma "**1**". Programozással a kívánt tárolócellák tartalma "**0**"-ra változtatható.

Az ultraibolya sugaras törlés egy kissé körülményes. Először is egy megfelelő erősségű (kb. 10 Ws/cm^2) ultraibolya fényforrás szükséges. Másodsorban a memória kitörlése elég sok időt vesz igénybe (átlagosan 15-30 percet).

A *6.24. ábra* egy elektromosan törölhető EPROM tárolócellát mutat be. Az elektromos törlés az ultraibolya sugaras törlés hátrányait küszöböli ki. A cella egyetlen MOS tranzisztorból áll, amelynek a törlése alagúthatáson alapul: Ez az ún. FLOTOX (**Flo**ating-gate **T**unnel-**Ox**ide) cella. A FLOTOX cella programozása – hasonlóan mint a FAMOS cella programozása –, a lebegő elektróda lavinainjektálásával történik. A FLOTOX cella törlése az ún. Fowler-Nordheim-féle tunneletfektus segítségével történik. Ha két elektróda között levő szigetelőben (ebben az esetben SiO_2) az elektromos térerő meghalad egy kritikus szintet (kb. 10^7 V/cm), akkor az alagúthatás következtében a negatív elektródán levő elektronok a szigetelőn keresztül eljutnak a pozitív elektródáig.

Ebben az esetben, ha az *X* címvonalra kapcsolt vezérlő elektróda földpotenciált kap, és a drain nagy pozitív potenciált (kb. +20 V-ot), akkor az elektronok a negatív töltésű elektródáról a vékony oxidrétegen keresztül a Fowler-Nordheim-féle effektus alapján eljutnak a drainig. Így a lebegő elektróda elveszti a negatív töltését – a cella kitörlődik.



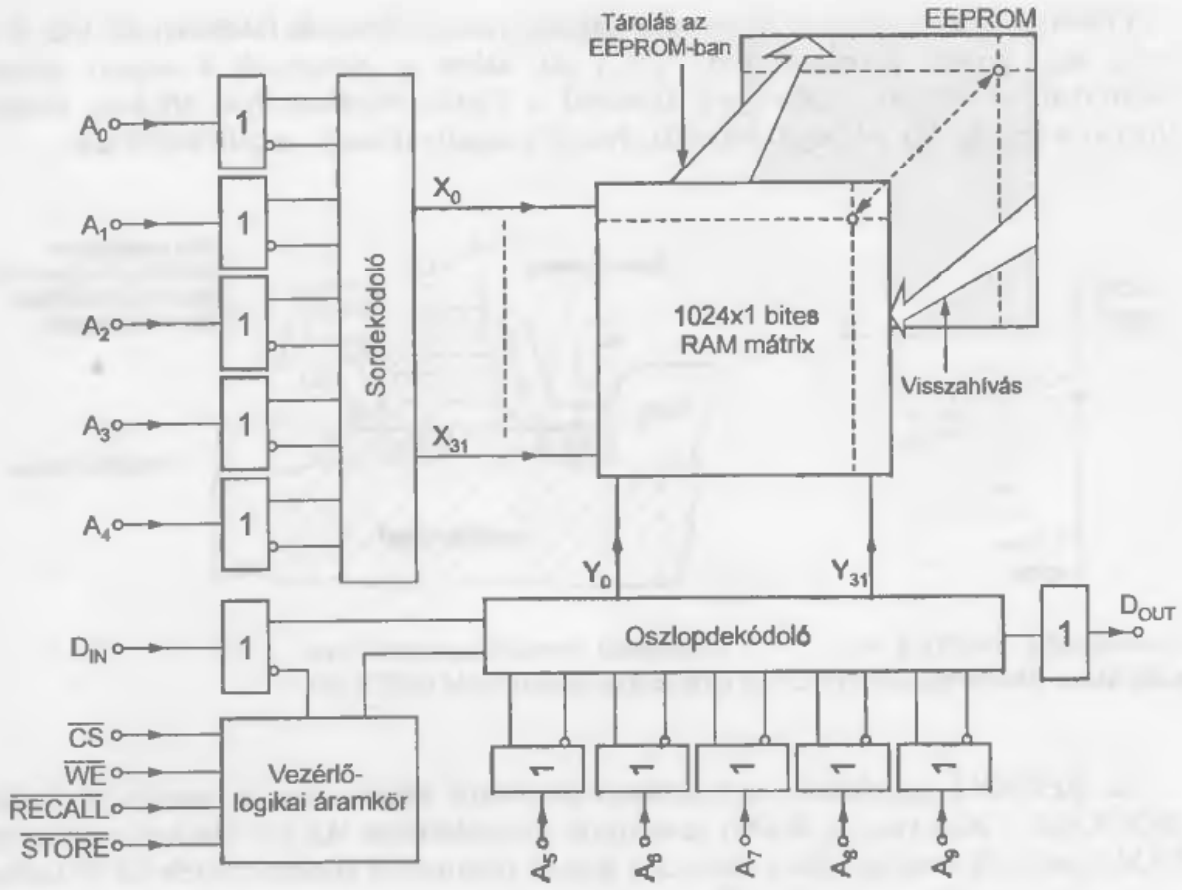
a) tárolócella –FLOTOX cella *b) integrált áramköri keresztmetszete*

6.24. ábra: Elektromosan törölhető és újraprogramozható ROM (EEPROM)

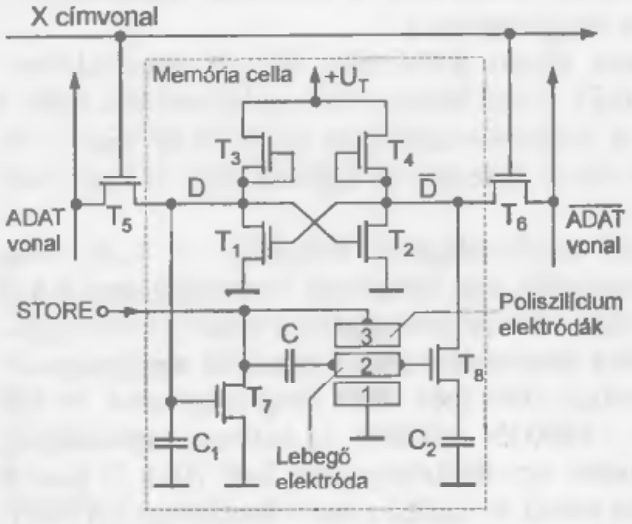
Az EEPROM áramkörök. egy érdekes és fontos alkalmazása a nemfelejtő RAM (NOVRAM – **N**on-volatile **RAM**) áramkörök megvalósítása. Az előbbiekben bemutatott RAM áramkörök a tápfeszültség kiesésekor a tárolt információt elvesztik, elfelejtik. Ott, ahol ez nem megengedett, és kis hozzáférési idejű írható-olvasható memória szükséges, a RAM memóriaegység táplálását a hálózati tápfeszültség megszűnése esetén a külön erre a célra használt akkumulátorból oldják meg. Ez azért szükséges, hogy a logikai áramkör a hálózati tápfeszültség megszűnése esetén automatikusan átkapcsolja a memóriát akkumulátoros táplálásra és fordítva, a hálózati tápfeszültség megjelenésekor.

Az alábbiakban bemutatásra kerülő nem felejtő RAM nem igényel akkumulátoros táplálást. Az áramkör vázlatos felépítése a *6.25. ábrán* látható. Azt a tulajdonságát, hogy a tápfeszültség megszűnése után is megtartja a tárolt információt, az 1024 × 1 bit szervezésű kis hozzáférési idejű RAM memóriamátrixszal párhuzamosan kapcsolt azonos kapacitású EEPROM memóriamátrixnak köszönheti.

Az áramkör összetett RAM–EEPROM tárolócellájának felépítését a *6.26. ábra* szemlélteti. A T_1 – T_6 tranzisztorok az előbbiekben már ismertetett hattranzisztoros RAM cellát (*6.8. ábra*) alkotják. Az EEPROM cellát a három polikristályos szilícium elektródás MOS tranzisztor képezi. A hálózati feszültség megszűnése után a memória tápegységének feszültsége a szűrőkondenzátorban tárolt energia miatt még rövid ideig megmarad. Ez idő alatt a RAM cella tartalma beíródik az EEPROM cellába: A hálózati tápfeszültség megszűnésének pillanatában a STORE bemenet egy feszültségugrást kap. Ha a D pont **0** logikai szinten van; akkor T_7 lezárt állapotban marad, és ezáltal a drain feszültsége a STORE bemenettel egyidejűleg fut fel. Így a kapacitívan kapcsolt lebegő elektróda az első, földelt elektródához képest pozitívvá válik. Ezért a lebegő elektróda a földelt elektródából elektronokkal töltődik fel. Ellenkező esetben, ha a *D* pont logikai **1** szinten van, akkor T_7 vezetni kezd, és a drainje földpotenciálon marad. Így a lebegő elektróda kapacitívan földelt.



6.25. ábra: Nem felejtő RAM (NOVRAM) tömbvázlata



6.26. ábra: A nem felejtő RAM cellája

Ezért a harmadik elektróda, amely az előbbihez képest pozitívvá válik, magához vonzza a másik elektronjait. Ha a memória újra tápfeszültséget kap, akkor a RAM cella állapota a lebegő elektródán tárolt töltés (elektronok) függvényében alakul, vagyis a lebegő elektróda nincs lebegő elektronokkal feltöltve, akkor T_8 kinyit, és a $\overline{D}$ pontra rákapcsolódik a kisült C_2 kondenzátor. Mivel ennek értéke nagyobb, mint a D pontra kapcsolódó ugyancsak kisült C_1 kondenzátoré, a flip-flop állapota a tápfeszültség megjelenésével úgy alakul, hogy $\overline{D} = 0$ és $D = 1$ lesz. Ellenkező esetben, ha a lebegő elektróda fel van töltve elektronokkal, akkor T_8 zárva marad, és a kisült C_1 kondenzátor a flip-flop állapotát úgy alakítja, hogy $D = 0$ lesz.

6.3.4 Logikai tömbök (PLA és PAL áramkörök)

A csak olvasható (fix) tárak egy dekódoló (*a címek dekódolására*) és egy kódoló (*a címek alatt található szavak tárolására*) összekapcsolásával keletkeznek. A ROM-ban tárolt információ tulajdonképpen egy igazságtáblázat, amelyben a címváltozók a bemeneti változókat képviselik. A bemeneti változók minden állapotkombinációjához a kimeneti változó egy állapota tartozik. Ezt a tár a megfelelő cím alatt tárolja. A dekódoló rész n bites címhosszúságnál a $0, 1, \ldots, 2^{n-1}$ címeket tartalmazza. A kódoló részben ezért 2^n szót lehet elhelyezni; m bites szóhosszúságnál ezért a ROM kapacitása 2^n számú m-bites szó, vagyis $m \cdot 2^n$ bit.

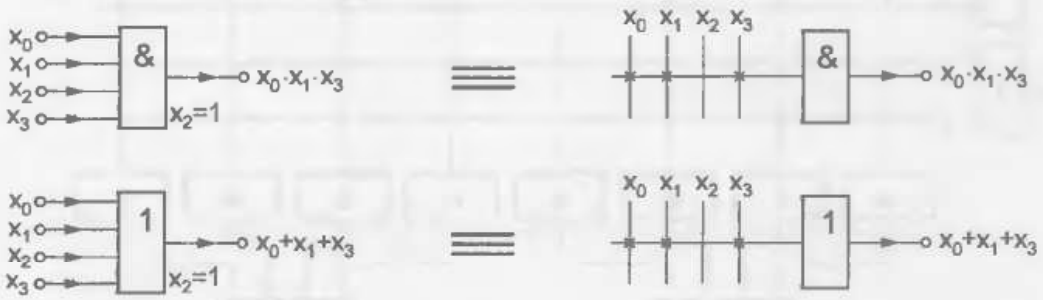
Gyakran előfordul, hogy az igazságtáblázatban (vagyis a memória mátrix celláiban) viszonylag kevés helyen van logikai **1**-es érték. Ilyenkor gazdaságosabb, ha nem az egész igazságtáblázatot tároljuk, hanem a logikai függvényeket képezzük. A logikai függvények képzése legegyszerűbben logikai kapukkal valósítható meg. Ha diszjunktív normálalakból indulunk ki, akkor a kimeneti változók legyenek például:

$$y_0 = \overline{x_1} \cdot x_3 + x_0 \cdot \overline{x_2} \cdot x_3 + x_0 \cdot x_2 + \overline{x_0} \cdot \overline{x_1}$$

$$y_1 = \overline{x_0} \cdot x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_0 \cdot x_1 \cdot x_2 \cdot x_3 + \overline{x_0} \cdot \overline{x_1} \qquad (6.1)$$

Az adott logikai függvények előállíthatók: – képezve először az *ÉS*-kapcsolatokat és utánna a *VAGY*-kapcsolatokat. Gyakorlatilag olyan mátrixot kell kialakítani, amelyben a bemeneti változók és ezek negáltjai között szükséges *ÉS*-kapcsolatokat egyszerűen kereszteződő vezetékek összekötésével valósítjuk meg. Egy másik mátrixban – az *ÉS*-kapuk kimenetei felhasználva – kialakíthatjuk a szükséges *VAGY*-kapcsolatokat. Ezt ***programozott logikai mátrixelrendezésnek*** (**PLA** – **P**rogrammable **L**ogic **A**rray) nevezzük.

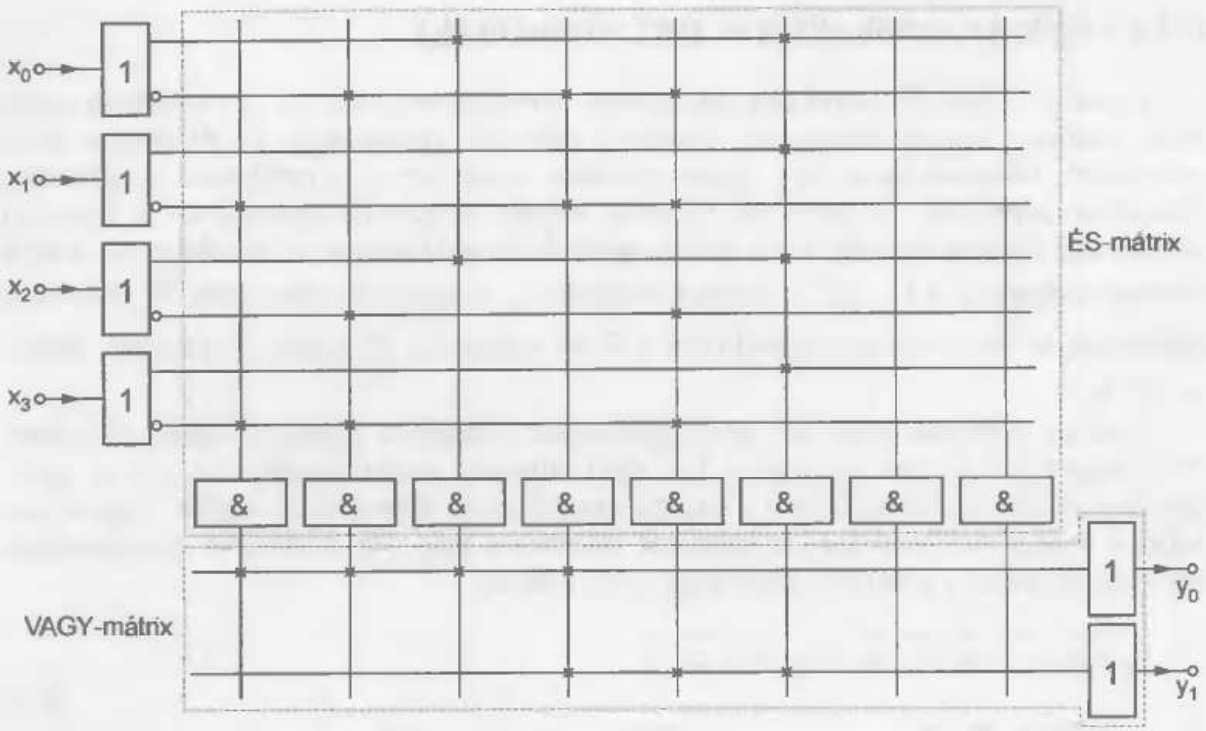
A szemléletes ábrázolás érdekében a *6.27. ábrán* látható egyszerűsített jelöléseket alkalmazzuk az *ÉS*-, illetve *VAGY*-kapukra.



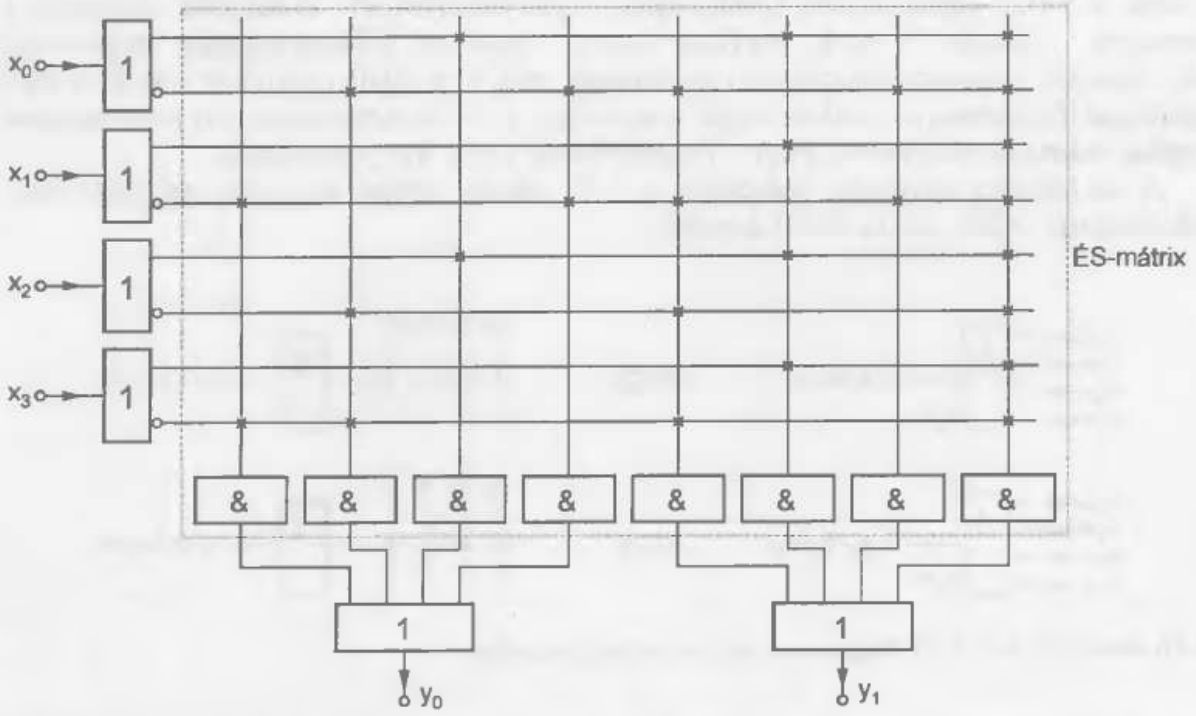
6.27. ábra: ÉS- és VAGY függvények egyszerűsített ábrázolása

A (6.1) egyenletekben szereplő függvények megvalósítását PLA-val a *6.28. ábra* szemlélteti.

A PLA áramkörök esetén – hasonlóan a ROM-okhoz – megkülönböztetünk maszkprogramozott és felhasználó által programozható típusokat. A felhasználó által programozható típusokat FPLA-val (**F**ield **P**rogrammable **L**ogic **A**rray) jelölik. A programozás hasonlóan történik, mint a ROM-ok esetén, de a szükséges programozó berendezés meglehetősen drága.



6.28. ábra. PLA elvi kapcsolása



6.29. ábra. PAL elvi kapcsolása

Ennek a hátránynak a kiküszöbölésére olyan egyszerűbb felépítésű FPLA áramköröket fejlesztettek ki, amelyeknél csak az *ÉS* mátrix programozható. A *VAGY* mátrixot már a gyártás során kialakítják. Az ilyen FPLA áramköröket **PAL** (**P**rogrammable **A**rray **L**ogic) kapcsolásoknak nevezik. Ezek programozása nem igényel különleges és drága programozó

készüléket. Az előre programozott VAGY mátrix nem okoz különösebb korlátozást, mivel nagyon sokféle típus kapható, amelyek VAGY mátrixa különböző.

A (6.1) egyenletekben szereplő függvények megvalósítását PAL kapcsolással a *6.29. ábra* szemlélteti.

6.4. Hibafelismerés és hibajavítás

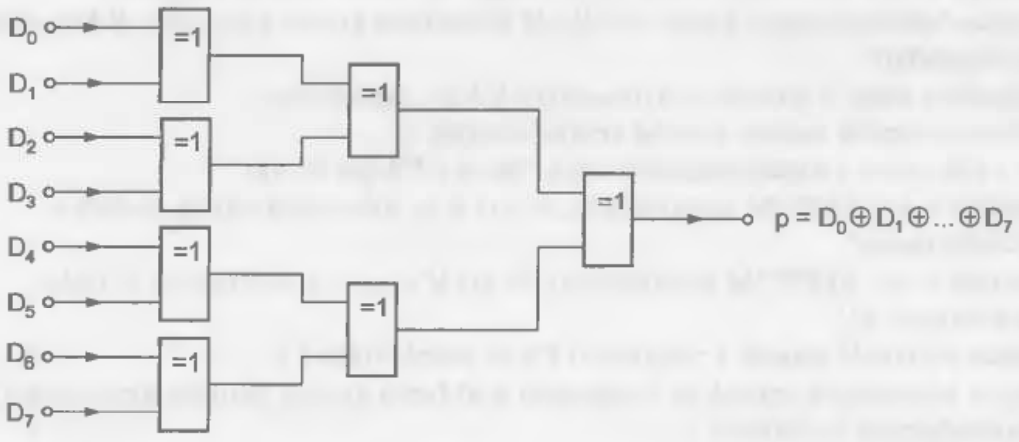
Az információ RAM-ban történő tárolása során fellépő hiba kétféle lehet: – *állandó* és *véletlen hiba*. Állandó hibákat (angolul: *hard errors*) okozhat maga a tár integrált áramköre vagy a működésben szerepet játszó vezérlő logika meghibásodása. Véletlen hibák (angolul: *soft errors*) csak időnként lépnek fel és ezért nem reprodukálhatók. Ilyen hibákat főleg a kozmikus sugárzás α részecskéi okoznak. Áttölthetik a dinamikus RAM-ok tárolókondenzátorait, ugyanakkor a statikus RAM tárolócelláját is átbillenthetik. Véletlen hiba keletkezhet az áramkörön belüli zavarimpulzusok hatására is.

A tárhibák fellépése nagyon komoly következményekkel járhat. Nemcsak hibás eredményhez vezethet egy számítógépnél a tár ilyen hibája, hanem magát a programot is tönkre teheti. Ezért olyan eljárásokat fejlesztettek ki, amelyek a hibák fellépését jelzik. E célból az információs biteket egy vagy több redundáns ellenőrző bittel egészítik ki. Minél több ellenőrző bitet használnak, annál több hiba ismerhető fel, sőt javítás is lehetségessé válhat.

A digitális adatok ellenőrzésének és javításának elméleti alapjaival a második fejezetben már foglalkoztunk.

6.4.1. Paritásbit

A hibafelismerés legegyszerűbb módja egy p paritásbit átvitele. Páros vagy páratlan paritást írhatunk elő. Páros paritás esetén a paritásbit **0**, ha az adatszóban előforduló egyesek száma páros, és logikai **1** értékű, ha az adatszóban előforduló egyesek száma páratlan. A páros paritásbit az adatbitek számjegyeiből képzett (modulo-2) összegként is értelmezhető. Ezt a számjegyösszeget kizáró VAGY-függvénnyel számíthatjuk ki az adatbitekből. Egy paritásképző egység megvalósítását páros paritásra a *6.30. ábra* szemlélteti. A kizáró VAGY-kapuk elhelyezkedési sorrendje tetszőleges.



6.30. ábra. 8 bites paritásgenerátor páros paritásra

Hibafelismerés céljából a paritásbitet az adatbitekkel együtt tárolják. Adatolvasásnál ismét képezik a paritásbitet és ennek végeredményét összehasonlítják a tárolt paritásbittel. Ha ezek különböznek, akkor hiba lépett fel. Hibajavítás azonban mégsem végezhető, mert nem ismert a hiba helye. Ha több hibás bit is van, akkor csak a páratlan hibaszám ismerhető fel, a páros nem.

6.4.2 Hamming-kód

A Hamming-kódnál több ellenőrző bit használata lehetővé teszi a hibafelismerés finomítását, sőt egyszeres hibáknál nemcsak hibajelzés, hanem a helymeghatározás is megtörténhet. Ha bináris kódnál a hibás bit helye ismert, akkor ez invertálással javítható.

A 2.4. szakaszból tudjuk, hogy ha n információs bithez k paritásbitet adunk, és a kódrendszer egy hibát tud javítani, akkor a következő egyenlőtlenségnek kell fennállnia:

$$2^k \geq (n+k)+1$$

A gyakorlatilag fontos eseteket a *6.2. táblázat* foglalja össze. Megfigyelhető, hogy az ellenőrző bitek aránya a hasznos szóhosszhoz képest annál kisebb, minél nagyobb a szóhossz.

Adatbitek száma	n	1÷4	5÷11	12÷26	27÷57	58÷120	121÷247
Ellenőrző bitek száma	k	3	4	5	6	7	8

6.2. táblázat: A szükséges ellenőrző bitek száma egyetlen hiba felismeréséhez és kijavítására a szóhossz függvényében

Kérdések:

1. Mit nevezünk memóriának, és milyen lehetőségek vannak a memóriák osztályozására?
2. Milyen jellemzői vannak a mágnesréteges memóriáknak?
3. Mit nevezünk gyűrűs léptetőregiszternek?
4. Milyen jellemzői vannak a RAM áramkörörknek?
5. Rajzolja le egy dinamikus RAM tárolócelláját, és írja le hogyan működik!
6. Hogyan lehetséges nagy kapacitású RAM kialakítása kisebb kapacitású RAM integrált áramkörökből?
7. Hasonlítsa össze a statikus és a dinamikus RAM áramköröket!
8. Milyen jellemzői vannak a ROM áramkörörknek?
9. Mi a különbség a maszkprogramozott ROM és a PROM között?
10. Rajzolja le egy EPROM tárolócelláját, és írja le az információtárolás és törlés mechanizmusát!
11. Rajzolja le egy EEPROM tárolócelláját, és írja le az információtárolás és törlés mechanizmusát!
12. Milyen jellemzői vannak a nemfelejtő RAM áramköröknek?
13. Milyen lehetőségek vannak az információ RAM-ban történő tárolása során fellépő hibák felismerésére és javítására?

7. Digitális-analóg (D/A) és analóg-digitális (A/D) átalakítók

A digitális rendszerekben sokszor előfordul, hogy analóg jeleket kell feldolgozni, vagy átalakítani. Ha egy folytonos analóg jelet digitálisan kell feldolgozni, akkor az analóg bemeneti jelet megfelelő bináris számokká kell alakítani. A feladatot analóg-digitális átalakítóval (A/D átalakító, A/D konverter, **ADC**) lehet megoldani. A digitális értékek visszaalakítása ezzel arányos feszültséggé vagy árammá digitális-analóg átalakítóval (D/A átalakító, D/A konverter, **DAC**) lehetséges.
Az átalakítók működésének a bemutatására a következő fogalmakat szükséges definiálni:

- ***kvantálás:*** – egy analóg mennyiség érték-változási tartományának felosztása meghatározott számú egységre (*kvantumra*);
- ***MSB:*** – a legnagyobb helyértékű bit (angolul: **M**ost **S**ignificant **B**it) az a bit, amely a legnagyobb súlyozású a bináris számok felírása során;
- ***LSB:*** – a legkisebb helyértékű bit (angolul: **L**ast **S**ignificant **B**it) az a bit, amely a legkisebb súlyozású a bináris számok felírása során.

Egy n-jegyű binárisan kódolt, D digitális jelet n darab D_1, ..., D_n együtthatókkal az alábbi összefüggés írja le:

$$D = D_1 \cdot 2^{-1} + D_2 \cdot 2^{-2} + ... + D_k \cdot 2^{-k} + ... + D_{n-1} \cdot 2^{-(n-1)} + D_n \cdot 2^{-n}$$

A legkisebb helyértékű hely $2^{-n} = \frac{1}{2^n}$ és az ennek megfelelő *legkisebb helyértékű bit* (*LSB*) D_n. A legnagyobb helyértékű hely $2^{-1} = \frac{1}{2}$ és az ennek megfelelő *legnagyobb helyértékű bit* (*MSB*) D_1.
A D_{max} érték akkor lép fel, ha az n hely összes D_k együtthatója **1** értékű:

$$D_{max} = 2^{-1} + 2^{-2} + ... + 2^{-n} = 1 - 2^n \approx 1$$

7.1. Digitális-analóg (D/A) átalakítók

7.1.1. Digitális-analóg átalakítók alapelvei

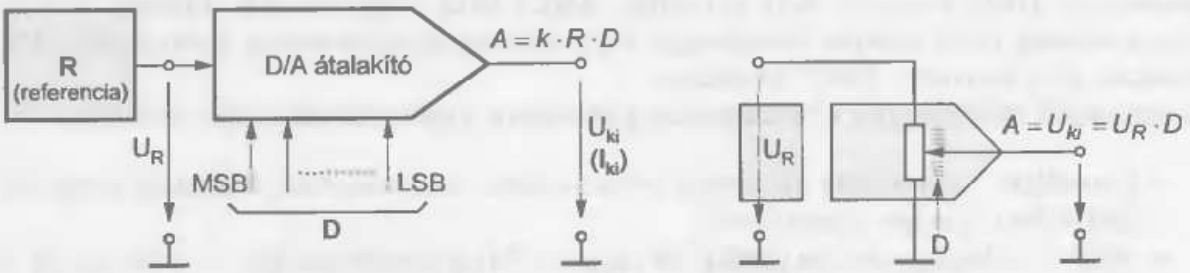
A ***digitális-analóg átalakítók*** (angolul: ***DAC*** – **D**igital to **A**nalog **C**onverter) feladata, hogy a bemenetére érkező „*D*" számadatnak megfelelő „*A*" analóg jelet (általában áramot vagy feszültséget) állítson elő a kimenetén (*7.1.a ábra*).

Egy ideális digitális-analóg átalakító esetén felírhatjuk, hogy:

$$U_{ki} = n \cdot \Delta U = n \cdot U_{LSB} \quad \text{vagy} \quad I_{ki} = n \cdot \Delta I = n \cdot I_{LSB},$$

feszültségkimenet, illetve áramkimenet esetén. Az n változó a bemenetre adott számot képviseli, ΔU illetve ΔI pedig egy bemenetre adott egységnek megfelelő feszültség- illetve áram-lépcsőket (pl. $n = 1$ esetén $U_{ki} = \Delta U$) jelenti.

A működéshez szükséges egy U_R referenciafeszültség (általában egy nagyon pontos feszültségforrás), amelyből a kimeneti feszültséget származtatjuk és ez határozza meg a kimeneti feszültség maximális értékét (*végkitérését*) is (angolul: *Full Scale* – F.S.).



a) tömbvázlat *b) elvi működés*

7.1 ábra. D/A átalakító

Az áramkör működése – feszültségkimenetet feltételezve – hasonlít egy potenciométeres feszültségosztó működéséhez, amelynek a leszedő-érintkezőjének a helyzetét digitális vezérléssel határozzuk meg (*7.1.b ábra*).

A digitális vezérléstechnikában az adatok többnyire bináris alakban állnak rendelkezésre, valamilyen meghatározott kódban kifejezve. Ezt a kódot a D/A átalakítónak ismernie kell. A D/A átalakítók csak meghatározott bináris kód szerinti jeleket képesek analóg jellé átalakítani. Bizonyos bináris kódok nem alkalmasak digitális-analóg átalakításra. Ezek az érték nélküli kódok. *Érték nélkülinek nevezünk azt a kódot*, amelynek elemeihez nincsenek adott számértékek hozzárendelve. A bináris- vagy a BCD-kód pl. *értékkel rendelkező kódok* (minden helyértékhez a kettőnek valamilyen hatványa van hozzárendelve). A Gray-kód ezzel szemben érték nélküli kód, elemeihez nincs konkrét számérték hozzárendelve. Az érték nélküli kódokat a digitális-analóg átalakítás előtt értékkel rendelkező kóddá kell alakítani, amely megfelelő kódátalakítókkal nehézség nélkül megoldható.

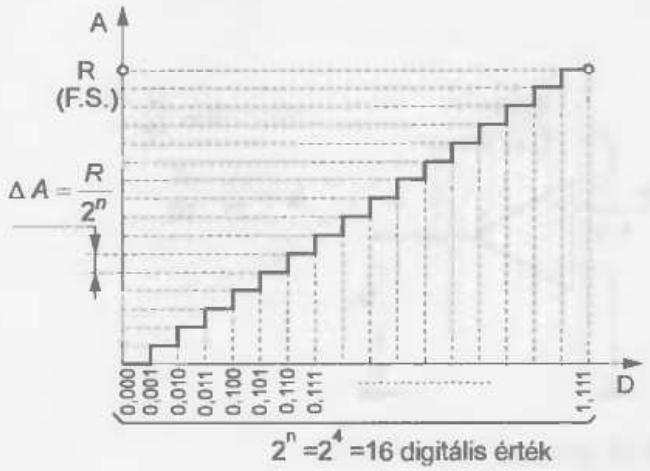
Az ideális D/A átalakítók kimeneti jele egyenesen arányos a bemenetükre digitálisan adott szám értékével. Vagyis a *7.2. ábrán* látható módon a D digitális érték és a kimeneti A analóg jel között egyenes arányosság áll fenn ($D < 1$). Egy n-bites digitális érték maximum:

$$N = 2^n$$

analóg jelértéket eredményez. A legkisebb, elemi lépcső az analóg jelben:

$$\Delta A = \frac{R}{2^n}, \qquad \text{(feszültségkimenet esetén } \Delta A = \frac{U_R}{2^n} = U_{LSB}\text{)}.$$

Megfigyelhető, hogy egy analóg jel annál pontosabban állítható elő, minél kisebb egy elemi lépcső. Az elemi lépcső méretét az átalakító *bitszáma* („***felbontása***") határozza meg.



7.2 ábra. A D/A átalakítás elve

Az átalakítók felbontásának a növelése az áramköri megvalósítást nehezíti, drágítja. A kis pontosságigény esetén általánosan használt digitális-analóg átalakítók felbontása **8 bit** (256 elemi lépcső), nagyobb pontosságot biztosítanak a **10, 12, 14 bites** (1024, 4096, 65536 elemi lépcsővel) átalakítók, a **16,18, 20 bites** átalakítókat főleg nagy pontosságú rendszerekben alkalmazzák).

A D/A átalakítók egy digitális jelet (számot) vele arányos feszültséggé alakítanak. A D/A átalakítók az átalakítás elve szerint lehetnek:

- ***közvetlen elvű átalakítók***: amelyek vezérelt feszültség- vagy áramosztás, illetve erősítés útján állítják elő az analóg jelet a bemeneti digitális jelből;
- ***közvetett elvű átalakítók***: amelyek az analóg jel előállításának folyamatába egyéb mennyiségeket (pl. impulzus-időt, idő-arányt, frekvenciát) iktatnak be.

Attól függően, hogy a digitális jel bitjei időben mikor fejtik ki hatásukat az átalakítás során megkülönböztetünk:

- ***párhuzamos működésű átalakítókat***: amelyekben a digitális jel bitjei egyidőben fejtik ki hatásukat (az átalakító bitszámával egyező ismétlődő hálózatrész található bennük);
- ***soros működésű átalakítókat***: amelyek egymást követő ismétlődő ciklusokban, bitenként alakítják át a digitális jelet analóggá.

7.1.2. Közvetlen D/A átalakító, összegző erősítővel

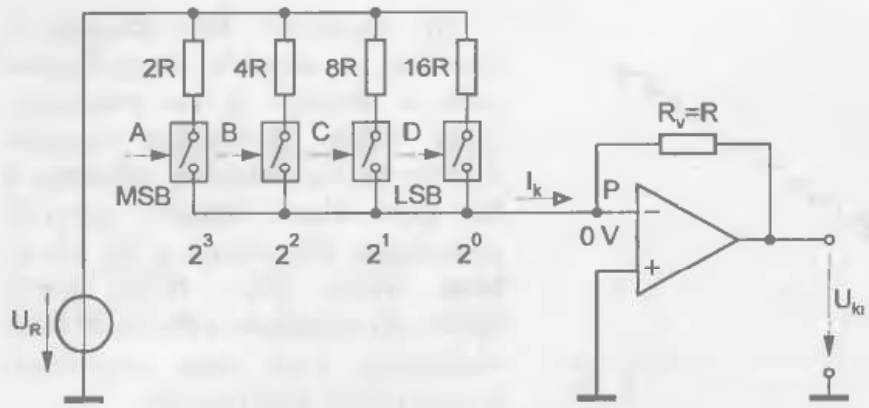
A *7.5. ábrán* látható műveleti erősítős összegző kapcsolás, bináris számok velük arányos feszültséggé való átalakítását végzi (*az egyszerűség kedvéért 4 bites digitális kódot szemléltetünk*). Az ellenállások értékeit úgy határozzuk meg, hogy zárt kapcsolóállásnál olyan áram folyjon rajtuk keresztül, amely megfelel az adott helyérték súlyának. A kapcsolókat akkor zárjuk, ha az adott helyértéken logikai **1** van. A műveleti erősítő R_v visszacsatoló ellenállásán létrejövő negatív visszacsatolás miatt a ***P*** összegzőponton mindig 0 V feszültség van. A részáramok tehát nem befolyásolják egymást.

Ha a legkisebb helyértékű bitnek (*LSB* – Last Significant Bit) megfelelő kapcsoló zárva van (a bináris érték: **0001**), akkor a kimeneti feszültség:

$$U_{ki} = U_{LSB} = -U_R \cdot \frac{R_v}{16 \cdot R} = -\frac{1}{16} \cdot U_R .$$

Ha a legnagyobb helyértékű bitnek (*MSB* – Most Significant Bit) megfelelő kapcsoló zárva van (a bináris érték: **1111**), akkor a kimeneti feszültség:

$$U_{ki} = U_{ki\max} = -U_R \cdot \frac{15 \cdot R_v}{16 \cdot R} = -\frac{15}{16} \cdot U_R .$$



7.5. ábra. Összegző erősítővel működő 4 bites D/A átalakító

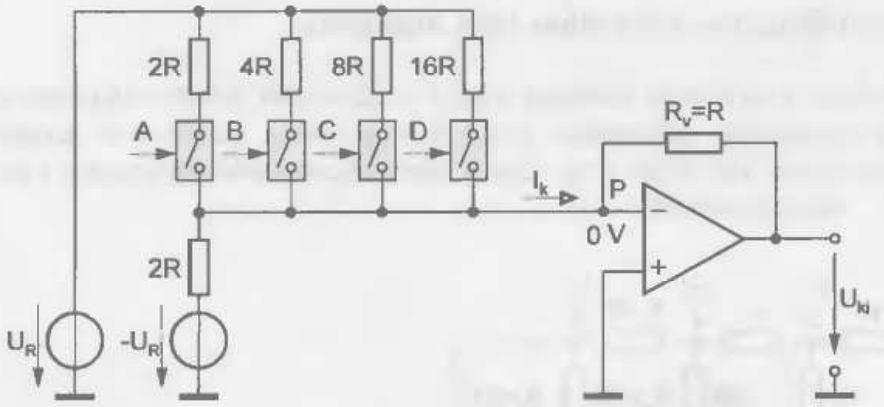
Általános esetben: $U_{ki} = D \cdot U_{LSB} = -U_R \cdot \frac{D}{D_{\max} + 1}$.

Az átalakító működését a *7.1. táblázat* foglalja össze.

A 2^{-1}	B 2^{-2}	C 2^{-3}	D 2^{-4}	U_{ki} *számpélda* $U_R = -16\,\mathrm{V}$
0	0	0	0	$0 = 0 \cdot U_{LSB} = 0\,\mathrm{V}$
0	0	0	1	$-\left(\frac{1}{16} \cdot U_R\right) = 1 \cdot U_{LSB} = 1\,\mathrm{V}$
0	0	1	0	$-\left(\frac{1}{8} \cdot U_R\right) = 2 \cdot U_{LSB} = 2\,\mathrm{V}$
0	0	1	1	$-\left(\frac{1}{8} \cdot U_R + \frac{1}{16} \cdot U_R\right) = 3 \cdot U_{LSB} = 3\,\mathrm{V}$
.....				.....
1	1	1	1	$-\left(\frac{1}{2} \cdot U_R + \frac{1}{4} \cdot U_R + \frac{1}{8} \cdot U_R + \frac{1}{16} \cdot U_R\right) = 15 \cdot U_{LSB} = 15\,\mathrm{V}$

7.1. táblázat. A D/A átalakító működése

A *7.5. ábrán* látható kapcsolás kismértékű kiegészítésével olyan D/A átalakítót készíthetünk, amelyik pozitív-negatív polaritású feszültséget képes előállítani. A módosított kapcsolás a *7.6. ábrán* látható. A műveleti erősítő invertáló bemenetére egy újabb ágat csatlakoztatunk, amely az eredetivel azonos abszolút értékű, de ellenkező irányú referencia feszültségforrást tartalmaz. Eredményképpen olyan átalakítót kapunk, amely nem változtatja meg a feszültséglépcsők nagyságát csak a kezdőpont (a **0000** kódnak megfelelő kimeneti feszültség) tolódik el 8 egységgel (*7.2. táblázat*). Megfigyelhető, hogy a kimeneten 0V akkor áll elő mikor csak az *A* bitnek (előjel-bit) megfelelő kapcsolót zárjuk (ilyenkor a $+U_R$ és $2R$ illetve a $-U_R$ és $2R$ ág éppen kiegyenlíti egymást). A kimeneti feszültséglépcsők száma nem változik, továbbra is 16 különböző állapot van (**8** a negatív feszültség-lépcsőknek, **1** a nullának és **7** a pozitív feszültség-lépcsőknek).



7.5. ábra. Pozitív-negatív kimenetű, összegző erősítővel működő 4 bites D/A átalakító

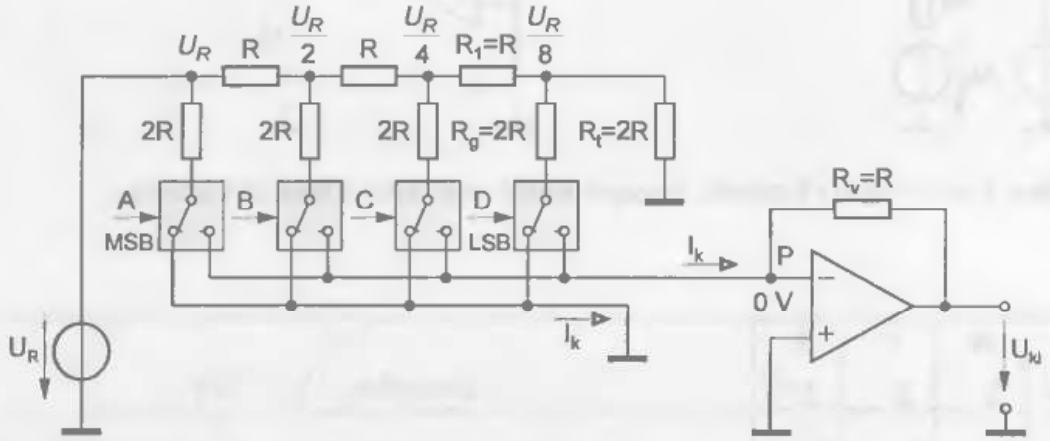
A előjel	B 2^{-1}	C 2^{-2}	D 2^{-3}	U_{ki} *számpélda* $U_R = -16\,\mathrm{V}$
0	0	0	0	$-\frac{1}{2}\cdot U_R = -8\cdot U_{LSB} = -8\,\mathrm{V}$
0	0	0	1	$-\frac{1}{2}\cdot U_R + \frac{1}{16}\cdot U_R = -7\cdot U_{LSB} = -7\,\mathrm{V}$
0	0	1	0	$-\frac{1}{2}\cdot U_R + \frac{2}{16}\cdot U_R = -6\cdot U_{LSB} = -6\,\mathrm{V}$
.....				.....
1	0	0	0	$-\frac{1}{2}\cdot U_R + \frac{1}{2}\cdot U_R = -0\cdot U_{LSB} = 0\,\mathrm{V}$
.....				.....
1	0	0	1	$-\frac{1}{2}\cdot U_R + \frac{9}{16}\cdot U_R = +1\cdot U_{LSB} = +1\,\mathrm{V}$
.....				.....
1	1	1	1	$-\frac{1}{2}\cdot U_R + \frac{15}{16}\cdot U_R = +7\cdot U_{LSB} = +7\,\mathrm{V}$

7.2. táblázat. Pozitív-negatív kimenetű, 4 bites D/A átalakító működése

Az összegző erősítővel működő D/A átalakítók egyik hátránya, hogy nagyon széles értéktartományú ellenállás-sorozat szükséges a felépítéséhez (pl. 10 bit esetén *R*-től 1024*R*-ig). Ez nagyon megnehezíti a megfelelő pontosságú megvalósítást. Ugyancsak hátrányként említhető, hogy az elektronikus kapcsolókon nagy feszültségváltozások lépnek fel. A nyitott kapcsolón U_R feszültségkülönbség van, záráskor a feszültségkülönbség 0 lesz. Ezért minden átkapcsolás során át kell tölteni a kapcsoló parazita kapacitásait.

7.1.3. Ellenállás-létrahálózatos közvetlen D/A átalakító

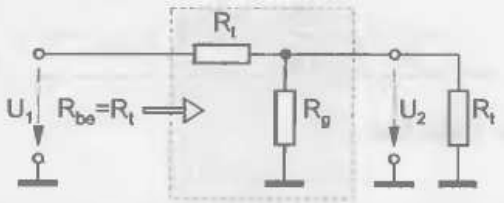
Az integrált D/A átalakítók gyártásánál általában a nagy értékszórású, pontos ellenállások megvalósítása okozza a legnagyobb nehézséget. Ezért a súlyozást a helyértékek szerint gyakran ellenállás-létrahálózattal valósítják meg. Egy 4 bites ellenállás-létrahálózatos D/A átalakító kapcsolását a 7.7. ábra szemlélteti.



7.7. ábra. Ellenállás-létrahálózatos közvetlen D/A átalakító

Egy ilyen létrahálózat alapeleme a *7.8. ábrán* látható terhelt feszültségosztó, amely a következő tulajdonságú:

- ha R_t terhelő-ellenállással terheljük, akkor az R_{be} bemeneti ellenállásának R_t-vel egyenlőnek kell lennie;
- az $\alpha = \frac{U_2}{U_1}$ feszültségátvitelnek ennél a terhelésnél egyenlőnek kell lennie egy előre megadott értékkel.



7.8. ábra. A létrahálózat egy fokozatának felépítése

A méretezési egyenletek az előbbi két feltétel felhasználásával:

$$R_1 = \frac{(1-\alpha)^2}{\alpha} \cdot R_g$$

$$R_t = \frac{(1-\alpha)}{\alpha} \cdot R_g$$

Bináris kód esetén $\alpha = 0{,}5$. Feltételezve, hogy $R_g = 2 \cdot R$, akkor: $R_1 = R$ és $R_t = 2 \cdot R$ a *7.7. ábrán* szemléltetett adatokkal összhangban. A referenciafeszültség-forrást az $R_{be} = 2 \cdot R \times 2 \cdot R = R$ ellenállás terheli. Az összegző műveleti erősítő kimeneti feszültsége, pedig:

$$U_{ki} = -R_v \cdot I_k = -U_R \cdot \frac{R_v}{16 \cdot R} \cdot (8 \cdot A + 4 \cdot B + 2 \cdot C + D) = -U_R \cdot \frac{R_v}{16 \cdot R} \cdot D$$

A *7.7. ábrán* látható digitális-analóg átalakító kapcsolásban a $2R$ értékű ellenállások két R értékű ellenállás sorba kapcsolásával helyettesíthetők. Ezért az áramkör nagyon jól gyártható monolit integrált áramkörös technológiával. A szükséges együttfutási pontosság így egyszerűen biztosítható, mivel az ellenállások pontos értéke nem lényeges, csupán a relatív hibát kell minél kisebbre csökkenteni. A pontos kimeneti feszültség biztosítása céljából az R_v visszacsatoló ellenállást is ugyanazon a monolit chipen kell megvalósítani. Így az A/D átalakító kimeneti feszültségének értéke ($R_v = R$ helyettesítést alkalmazva):

$$U_{ki} = -U_R \cdot \frac{1}{16} \cdot D.$$

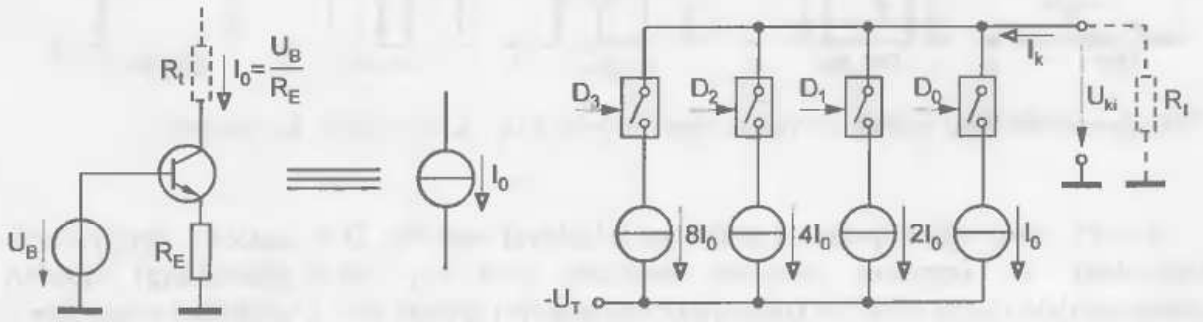
7.1.4. Kapcsolt áramgenerátoros közvetlen D/A átalakító

A bipoláris technológiájú digitális-analóg átalakítók egyszerűen áramgenerátorokkal valósíthatók meg, amelyek az eredő kimeneti áram összetevőit alkotják. Ha egy bipoláris tranzisztor bázisára stabil egyenfeszültséget kapcsolunk és emitter-ellenállással látjuk el, akkor a kollektor-terhelésen állandó áram folyik (*7.9.a ábra*). Ha ilyen áramgenerátorból többet ki-be kapcsolható módon párhuzamosan kötünk, és áramukat **2** hatványai szerinti értékűre állítjuk be, akkor D/A átalakítót valósíthatunk meg (*7.9.b ábra*). Ha a bázisfeszültségek egyenlőek, és minden emitter-ellenállást közös $-U_T$ pontra csatlakoztatunk, akkor az ellenállásoknak a helyérték súlyával fordított arányban kell változniuk. Ez a bipoláris integrált áramkörös technikában is megnehezíti a kellő pontosságú megvalósítást.

Emiatt az áram eloszlását itt is célszerű létrahálózattal megoldani. Az elvet a *7.10. ábrán* látható kapcsolás szemlélteti.

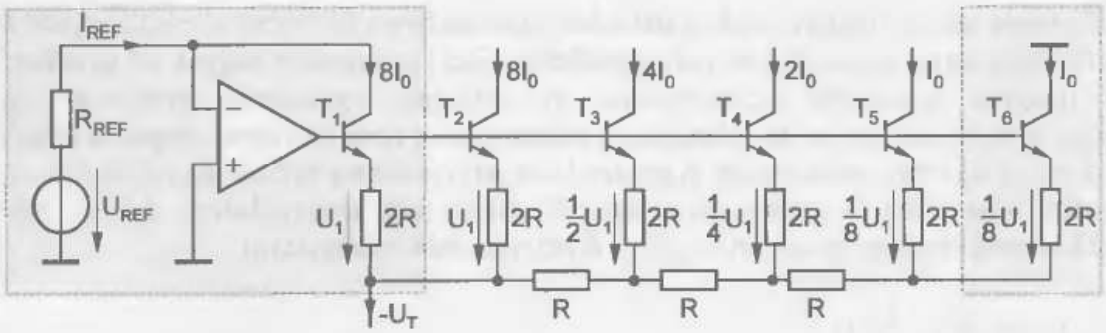
A T_1, ..., T_6 tranzisztorok bázisfeszültsége egyenlő. A bázisfeszültséget a műveleti erősítő úgy állítja be, hogy a T_1 referencia tranzisztoron $I_{REF} = \frac{U_{REF}}{R_{REF}}$ áram folyjék. Ekkor $U_{REF} = 2 \cdot R \cdot I_{REF}$. Ha a többi tranzisztor emitter-bázis feszültsége egyenlő T_1-évei, akkor az emitter-ellenállásokon a *7.10. ábrán* bejelölt feszültségesések jelentkeznek, tehát megfelelő súlyozott áramok keletkeznek.

Nem adódnak teljesen egyenlő bázis-emitter feszültségek még akkor sem, ha a tranzisztorok egyformák, mert különböző áramok folynak át rajtuk. Hogy minden tranzisztor kollektorárama azonos legyen, annyi tranzisztort kell párhuzamosan kapcsolni, hogy minden tranzisztoron csak I_0 áram folyjon. Integrált áramkörökben ez egyszerűen megvalósítható, mivel csak az árammal arányosan kell növelni a tranzisztorfelületet.



a) tranzisztoros áramgenerátor *b) elvi kapcsolás*

7.9. ábra. Kapcsolt áramgenerátoros közvetlen D/A átalakító



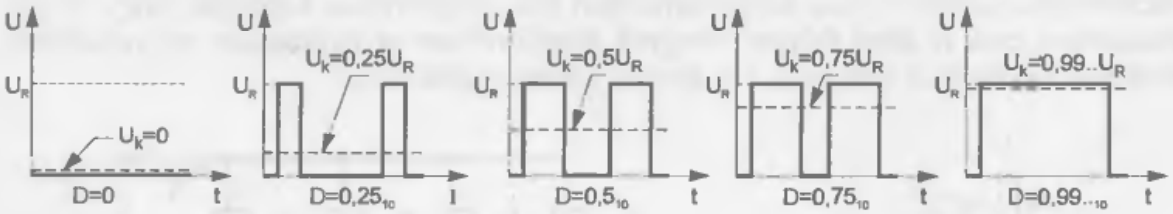
7.10. ábra. Súlyozott áramok előállítása

A *7.10. ábra* létrahálózatának $2R$ lezáró ellenállását nem szabad a fölhöz csatlakoztatni, hanem erre a célra emmitterpotenciálú pontot kell keresni. Erre való az egyébként nem használt T_6 tranzisztor. A T_6 tranzisztor emitterét T_5-el párhuzamosan köthetjük, és a két emitterellenállást egyetlen R ellenállássá egyesíthetjük.

7.1.5. Közvetett D/A átalakító impulzus-kitöltés modulációval

A közvetett elvű D/A átalakítók közül az impulzus-kitöltés modulációval (angolul: ***PDM** – **Pulse Duration Modulation***) működő átalakító bír a legnagyobb gyakorlati jelentőséggel. Olyan helyeken célszerű az alkalmazása, ahol az átalakítás sebessége nem kritikus.
A működés alapelve a *7.11. ábra* alpján tárgyalható.

- Olyan – logikai szintű – négyszögjelet állítunk elő, amelynek kitöltési tényezője megegyezik az átalakítandó digitális értékkel.
  *Megjegyzés: – **Kitöltési tényezőnek** nevezzük a négyszögjel logikai 1-es értékének- és teljes periódusidejének viszonyát.*
- A digitális értékkel egyező kitöltési tényezőjű periodikus jelet egy megfelelő áramkörrel U_R (*referencia feszültség*) amplitúdójú négyszögjellé alakítjuk (a *7.11. ábra* különböző kitöltési tényezőjű jelalakokat szemléltet);
- A kapott U_R amplitúdójú négyszögjelet egy *középérték-képzőre* kapcsoljuk. A kimeneti feszültség egyenesen arányos a bemeneti digitális értékkel ($0 \le D \le 1$).

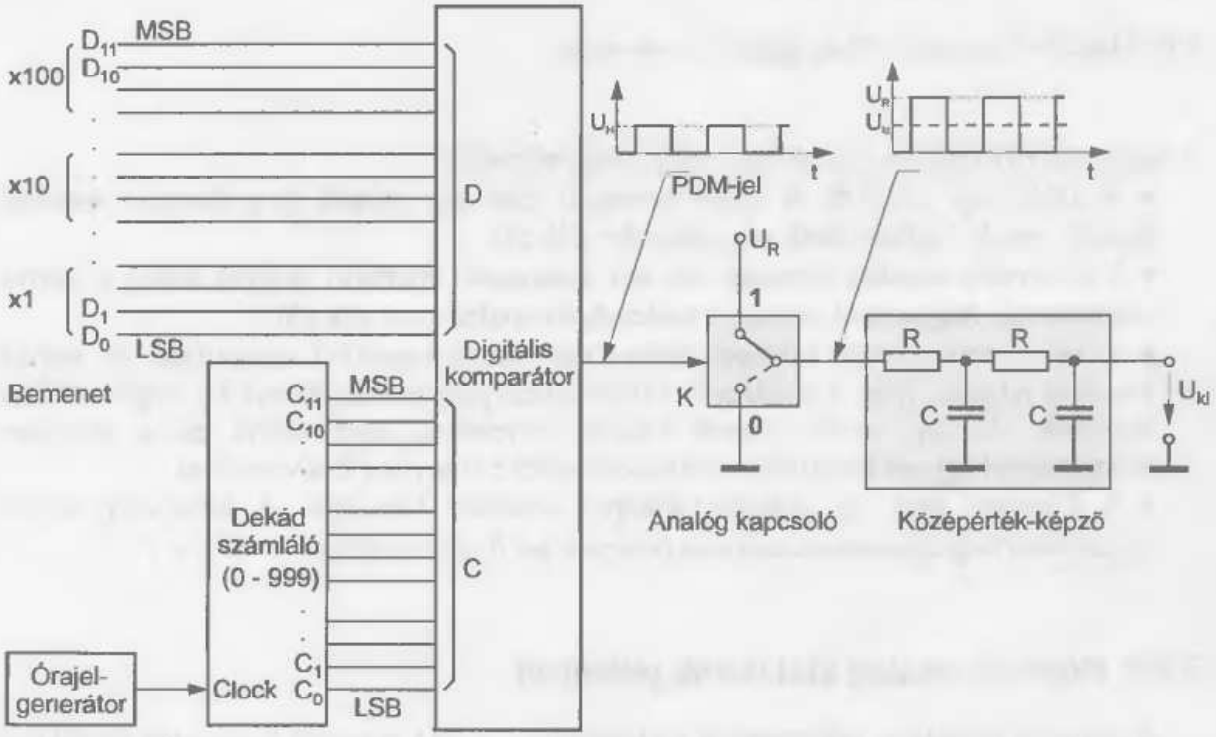


7.11. ábra. Különböző kitöltési tényezőjű négyszögjelek (U_k – a négyszögjel középértéke)

A *7.12. ábra* egy impulzus-kitöltés modulációval működő D/A átalakító tömbvázlatát szemlélteti. Az impulzus szélesség modulált jelet egy órajel-generátorral vezérelt dekádszámláló és egy digitális komparátor segítségével állítjuk elő. A digitális komparátor a számláló (***C***) és a bemenet (***D***) jelét hasonlítja össze. Az átalakító működése a *7.12. ábra* alapján követhető.

• Amíg a számláló az előre számlálás közben (maximális értéke 999) nem éri el a bemeneti jel értékét addig a komparátor kimenetén logikai **1**-es szint van (a *PDM* jel ilyenkor logikai **1** szinten van). Amikor a számláló kimenetén levő ***C*** digitális érték megegyezik a bemeneti ***D*** értékkel, a komparátor kimenete logikai **0**-ra vált, és ez a logikai szint marad a ciklus végéig (a *PDM* jel ilyenkor logikai **0** szinten van). Megállapítható, hogy a komparátor kimenetén keletkező négyszögjel (*PDM jel*) kitöltési tényezője – ezrelékben kifejezve – megegyezik a bemeneten levő digitális jel értékével.

• A *PDM* jellel vezérelt *K* analóg kapcsoló kimenetén olyan négyszögjelet kapunk, amely U_R és 0 V értékeket vesz fel a vezérlésnek megfelelő kitöltési tényezővel. Ennek a jelnek a középértékét úgy kapjuk meg, hogy aluláteresztő szűrőn (középérték-képzőn) vezetjük át.



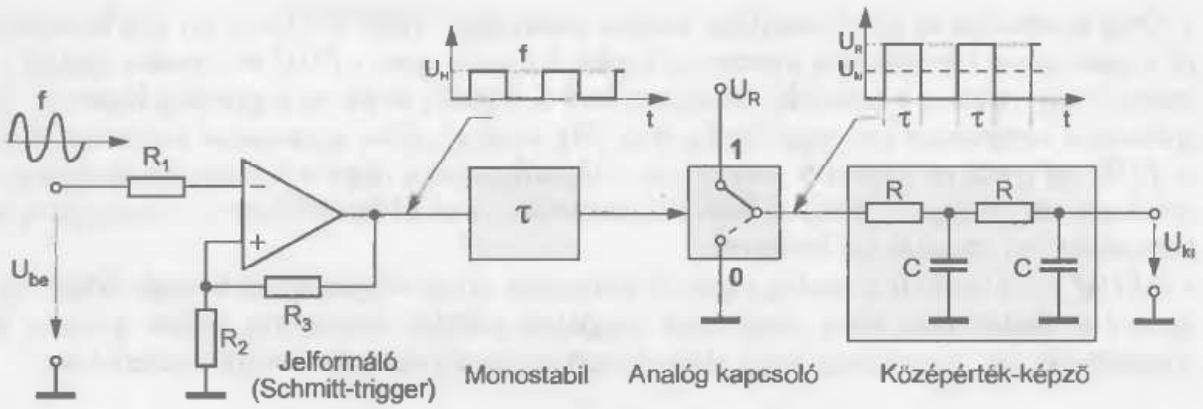
7.12. ábra. Impulzus-kitöltés modulációval működő D/A átalakító tömbvázlata

A tárgyalt D/A átalakító nagy előnye, hogy pontossága elvileg csak az U_R referencia-feszültség pontosságától függ, egyéb nagypontosságú áramköri elemekre nincs szükség. Ezért a D/A átalakításnak ezt a módszerét a gyakorlatban nagyon sok helyen alkalmazzák (pl. nagypontosságú egyenfeszültségek előállítása, CD lemezjátszók D/A átalakító részeként).

7.1.6. Frekvencia-feszültség közvetett D/A átalakító

A digitális technikában gyakran egy jel frekvenciáját használják fel mint a digitális jel értékét, mivel az iparban igen elterjedtek a frekvencia-kimenetű érzékelők, távadók.

A *7.13. ábra* egy frekvencia-feszültség átalakító tömbvázlatát szemlélteti, amely alkalmas a bemenetére érkező jel-frekvencia egyenfeszültséggé alakítására. Fontos követelmény, hogy a bemeneti jel-frekvencia és a keletkező egyenfeszültség között szigorúan lineáris összefüggés legyen.



7.13. ábra. Frekvencia-feszültség átalakító tömbvázlata

A kapcsolás működése a *7.13. ábra* alapján magyarázható.

- A különböző jelszintű és alakú bemeneti jelet egy erősítő és jelformáló áramkör fogadja, amely logikai szintű négyszögjelet állít elő.
- A jelformáló áramkör kimeneti jele egy monostabil áramkört vezérel, amely a bejövő frekvenciától függetlenül azonos, τ szélességű impulzusokat állít elő.
- A monostabil áramkör kimeneti jelével egy analóg kapcsolót vezérelünk. Az analóg kapcsoló feladata, hogy a τ szélességű impulzusok pontosan egyforma U_R amplitúdójúak legyenek. Az ily módon kapott azonos szélességű, amplitódójú és a bemeneti frekvenciával egyező impulzus-sorozat középértéke arányos a frekvenciával.
- A kimeneti jelet egy középérték-képző áramkör biztosítja. A kimeneten kapott egyenfeszültség egyenesen arányos a bemeneti jel frekvenciájával: $U_{ki} \sim f$.

7.1.7. Digitális-analóg átalakítók jellemzői

A kimeneti feszültség változásának tartománya: – a D/A átalakító kimenetén jelentkező analóg mennyiség változásának maximális tartományát képviseli.

Felbontás (*angolul:* ***rezolution***): – egy D/A átalakító kimenetén fellépő legkisebb analóg lépcső értéke; meg kell jegyezni, hogy a felbontás megadható vagy a kimeneten fellépő legkisebb elemi lépcső értékével, vagy a kimeneten jelentkező különböző elemi lépcsők maximális számával.

Offszethiba: – az átviteli karakterisztikának a kezdőponthoz viszonyított elmozdulása; az offszethiba mértékegysége az *LSB*.

Linearitáshiba: – a D/A átalakító valós karakterisztikájának az ideális karakterisztikától (*egy egyenes*) való eltérését fejezi ki; a linearitáshiba mértékegysége az *LSB*.

Konverziós (átalakítási) idő: – azt az időtartamot képviseli, ami egy átalakításhoz szükséges; az átalakítás akkor tekinthető befejezettnek, mikor a kimeneti analóg mennyiség a végleges értéket felveszi $\pm\frac{1}{2}LSB$.

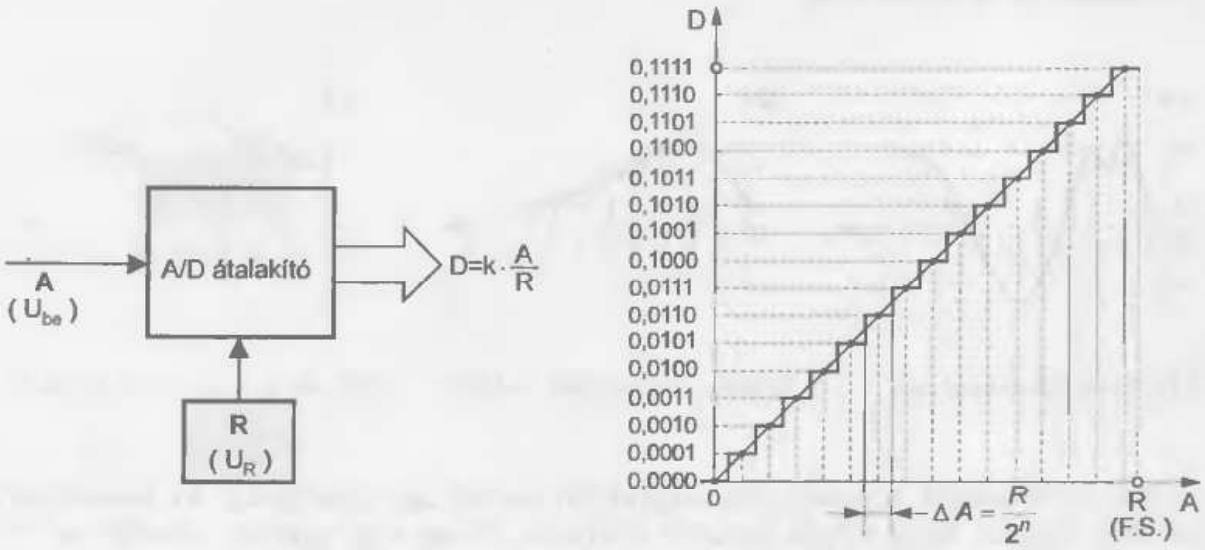
Átalakítási sebesség: – a D/A átalakító által egy másodperc alatt végrehajtott átalakítások számát adja meg.

7.2. Analóg-digitális (A/D) átalakítók

7.2.1. Analóg-digitális átalakítók alapelvei

Az ***analóg-digitális átalakító*** (angolul: ***ADC*** – **A**nalog to **D**igital **C**onverter) feladata, hogy a bemenetére érkező „*A*" analóg jelnek megfelelő „*D*" digitális jelet állítson elő a kimenetén (*7.15.a ábra*).

A működéshez szükséges egy ***R*** referencia (általában egy U_R referenciafeszültség), amelyhez az A/D átalakítók az ***A*** analóg mennyiséget viszonyítják ($D = k \cdot \frac{A}{R}$) és amely a kimeneti feszültség maximális értékét is meghatározza.



a) tömbvázlat *b) átviteli karakterisztika*

7.15. ábra. A/D átalakítók elvi működése

A *7.15.b ábra* az A/D átalakító átviteli karakterisztikáját szemlélteti. Megfigyelhető, hogy az analóg jel bizonyos számú (az ábrán 16 érték) amplitúdó-értékkel közelíthető (vagyis az A/D átalakító *kvantál*). Az analóg jelet – vagyis a 0 .... *R* tartomány minden pontjában értelmezett jelet (*feszültséget*) – egyenlő részekre kell osztani és minden egyes szakaszhoz egy-egy számot (*digitális mennyiséget*) kell rendelni a lehető legkisebb hibával. Az analóg jel tengelyének egy-egy elemi szakasza (ΔA), amelyhez ugyanaz a digitális kód tartozik, a ***kvantum***. Az analóg jelben a kvantum, vagy elemi lépcső mérete (*bináris kódú átalakítóknál*):

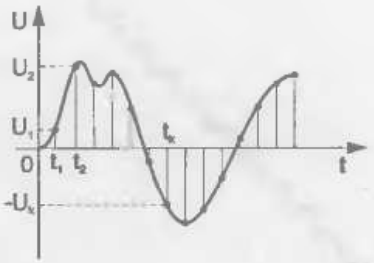
$$\Delta A = \frac{R}{2^n}, \text{ vagy feszültség esetén } \Delta U = U_{LSB} = \frac{U_R}{2^n}.$$

Egy analóg jel annál pontosabban ábrázolható digitális értékekkel, minél kisebb egy elemi lépcső, vagy kvantum nagysága. Mivel az elemi lépcsők méretét az átalakító *felbontása* határozza meg, a kvantálás finomságának a növelése érdekében növelnünk kell az átalakító felbontását. Az A/D átalakítónak annál nagyobb a felbontóképessége, minél több bit áll rendelkezésre a számábrázoláshoz.

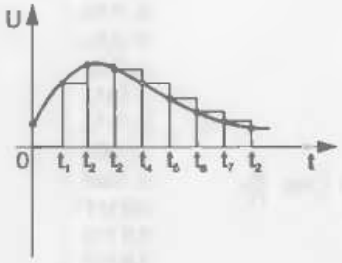
Az átalakítók felbontásának a növelése az áramköri megvalósítást nehezíti, drágítja. A kis pontosság-igény esetén általánosan használt analóg-digitális átalakítók felbontása **8 bit** (256 elemi lépcső), nagyobb pontosságot biztosítanak a **10, 12, 14 bites** (1024, 4096, 65536 elemi lépcsővel) átalakítók, a **16, 18, 20 bites** átalakítókat főleg nagy pontosságú rendszerekben alkalmazzák).

7.2.2. Mintavételezés

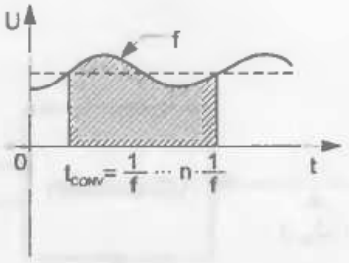
Az analóg jelábrázolással ellentétben, ahol a mérendő mennyiségeket folyamatos mérőjellé alakítják át, a digitális mérési jel ábrázolásánál csak diszkrét mérőjelek fordulnak elő, amelyeket mintavételezéssel, kvantálással és kódolással állítanak elő. A kvantálásnál elkerülhetetlen az információveszteség. Az ésszerű kvantálás a fizikai mérőjel fajtájától és az előirányzott alkalmazástól függ.



7.15. ábra. Mintavételezés



7.16 ábra. Pillanatérték átalakító



7.17. ábra. Középérték átalakító

Az A/D átalakítók bemeneti jele a legritkább esetben egyenfeszültség. Az átalakítandó jel, attól függően, hogy milyen forrásból származik, lassan vagy gyorsan változhat az idő függvényében. Az időben változó analóg jelek esetén – ahhoz, hogy a jelből képzett digitális információ megfelelő legyen – meghatározott időközönként a jelből mintát kell venni (*7.16. ábra*). A mintavételezés gyakoriságát annál nagyobbra kell venni, minél gyorsabban változik az analóg jel. Általánosságban érvényes:

- az analóg jel mintavételezési gyakoriságának legalább kétszer akkorának kell lennie, mint a jel változásának legnagyobb frekvenciája.

Az A/D átalakítókat aszerint, hogy az analóg jel mely értékeit dolgozzák fel, két alapvető típusba sorolják.

- ***Pillanatérték átalakítók:*** – amelyek bizonyos időközökben az adott időpillanatban felvett jel-értéket alakítják át digitális információvá (*7.16. ábra*). A pillanatérték átalakítók nem veszik figyelembe a jel megváltozását az átalakítás ideje alatt. Ugyanakkor igen nagy a zavarérzékenységük. Ezeket az átalakítókat legtöbbször gyors (μs nagyságrendű, vagy rövidebb átalakítási idővel) átalakításokhoz alkalmazzák.
- ***Átlagérték átalakítók:*** – amelyek az átalakítás ideje alatt a jel átlagértékét (középértékét) veszik figyelembe. Alkalmazásuk lassú átalakítások (ms nagyságrendű átalakítási idővel) esetén célszerű és olyan esetekben, amikor a hasznos jelre zavarfeszültség szuperponálódik (*7.17. ábra*).

Ha az átalakítás ideje (t_{CONV}) megegyezik a zavaró jel periódusidejével, akkor ez a zavar nem befolyásolja lényegesen a digitális eredményt. Ez különösen fontos, pl. ipari környezetben, mérőátalakítók, távadók jelének digitális feldolgozásakor.

7.2.3. Átalakítási elvek és áramköri megvalósításuk

Nagyszámú konkrét A/D átalakítótípus létezik, amelyek az áramköri kiépítésben, a használt elemekben, az elvégezhető műveletek sorrendjében, konstrukciós és technológiai megoldásokban térnek el egymástól. Ezek alapját azonban viszonylag kis számú bázismódszer (elv) képezi. Különböző közelítésben lehet vizsgálni ezen elvek kiemelését és egy megfelelő osztályozási rendszer kialakítását, amely az A/D átalakítók sokféleségét tükrözné.

Az egyik osztályozási módszer alapjául az A/D átalakítókban a visszacsatolás alkalmazásának a vizsgálata szolgál. Ebből a szempontból kiindulva az átalakitókat ***zárt*** és ***nyitott rendszerűekre*** oszthatjuk fel.

Hasonlóan a D/A átalakítókhoz, az A/D átalakítók is lehetnek ***közvetlen***, vagy ***közvetett*** működésűek aszerint, hogy nem iktatunk be, vagy beiktatunk az átalakítás folyamatába egyébb mennyiségeket.

Az utóbbi időben legelterjedtebbé vált osztályozás lényege az analóg mennyiség digitális átalakítási folyamata idődiagramjának vizsgálata. Már említettük, hogy a jelek mintavételi értékeinek egyértékű digitális értéksorba alakításához a kvantálási és kódolási műveletek szolgálnak alapul. Ezek az átalakítandó mennyiség soros, párhuzamos vagy soros-párhuzamos közelítésével valósíthatók meg. Ennek alapján az A/D átalakítók felépítési lehetőségeit célszerű ***soros***, ***párhuzamos*** és ***soros-párhuzamos*** módszerekre osztani. Ehhez még figyelembe kell venni, hogy a soros eljárás lehet *számláló típusú*, illetve megvalósítható *súlyozott bináris közelítéssel* is.

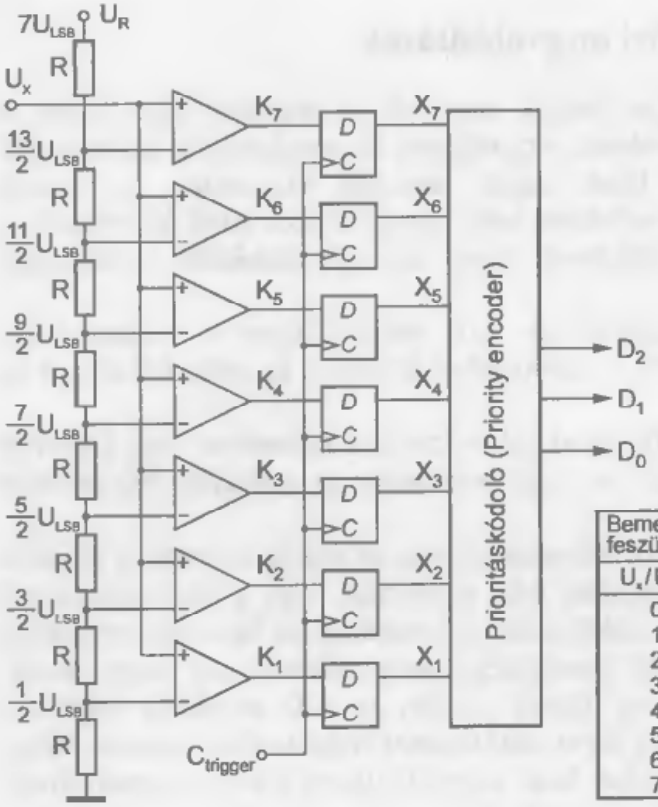
7.2.3.1 Közvetlen A/D átalakító

Egy közvetlen A/D átalakító megvalósítását – 3 bites számra – a *7.18.a ábra* szemlélteti. Az áramkör működése az amplitúdószelekció elvén alapul.

- Az U_x bemeneti jelet egy komparátor-sor hasonlítja össze egy ellenállás-osztó által előállított lépcső feszültséggel.
- Amelyik komparátor nagyobb bemeneti feszültséget érzékel mint a hozzá tartozó lépcső feszültség, az logikai **1**-et, amelyik kisebbet, az logikai **0**-át ad a kimenetére ($K_1 - K_7$). Ha a bemenetre, pl. $\frac{5}{2}U_{LSB}$ és $\frac{7}{2}U_{LSB}$ közti bemeneti feszültség kerül, akkor az 1...3 komparátor kimeneti állapota logikai **1**-es, a 4...7 komparátoré logikai **0** szintű lesz. A *7.18.b ábrán* látható táblázatban az összetartozó komparátor-állapotok és a bináris kódú eredmények összefoglalása látható.

A szükséges átalakítást prioritásdekódolóval oldhatjuk meg. A prioritásdekódolót nem lehet közvetlenül a komparátorok kimenetére csatlakoztatni, mert ha a bemeneti feszültség nem állandó, akkor időlegesen teljesen hamis eredményt kaphatunk. Példaként megvizsgálhatjuk a **3**-ról **4**-re való átváltást, amely bináris kódban **011**-ből **100**-ba való átváltást jelent.

Ha a legnagyobb helyértékű helyen a változás kisebb késleltetése miatt gyorsabban megvalósul, mint a többin, akkor egy átmeneti időre **111** jelenik meg, amely **7**-nek felel meg. Ez fél végkitérésnek megfelelő mérési hibát jelent.



$$Z = \frac{U_x}{U_{LSB}} = 7 \cdot \frac{U_x}{U_R} = Z_{max} \cdot \frac{U_x}{U_R}$$

Bemeneti feszültség	Komparátorállapotok							Bináris szám			Decimális megfelelő
U_x/U_{LSB}	K_7	K_6	K_5	K_4	K_3	K_2	K_1	D_2	D_1	D_0	Z
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	1	1
2	0	0	0	0	0	1	1	0	1	0	2
3	0	0	0	0	1	1	1	0	1	1	3
4	0	0	0	1	1	1	1	1	0	0	4
5	0	0	1	1	1	1	1	1	0	1	5
6	0	1	1	1	1	1	1	1	1	0	6
7	1	1	1	1	1	1	1	1	1	1	7

a) tömbvázlata *b) állapotai a bemeneti feszültség függvényében*

7.18. ábra. Közvetlen (párhuzamos) A/D átalakító (3-bites)

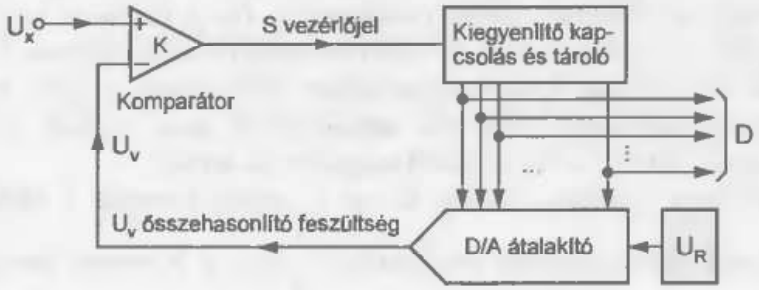
Ha a mérés időtartamára a komparátorok kimeneti feszültségét állandó értéken tartjuk, akkor ezt a hibát kiküszöböljük. A probléma megoldására a komparátorok kimenetén digitális tárolást használhatunk. Az ábrán ezt a célt szolgálják a minden komparátor-kimenetre kapcsolt élvezérelt *D* tárolók. Így biztosítható, hogy a prioritásdekódolóra egy teljes órajel-periódus alatt állandó bemeneti jel kerüljön. A következő triggerjel (jelölése: $C_{trigger}$) beérkezése előtt így a prioritásdekódoló kimenetén az állandósult állapotnak megfelelő adatok állnak rendelkezésre. A közvetlen átalakítási megoldás különleges előnye éppen a digitális mintavevő-tartó áramkör alkalmazásának lehetősége. Ez a nagysebességű A/D átalakítás előfeltétele.

A bemutatott közvetlen átalakító hátrányaként említhető, hogy annyi komparátor és ellenállás kell, ahány lépcső van (pl. egy 10 bites átalakítóhoz 1023 komparátor szükséges). Előny viszont, hogy a közvetlen módszerrel rendkívül gyors átalakítót lehet készíteni (ezért is hívják *angolul villám konverternek – flash converter*).

Az 1990 óta rendelkezésre álló integrálási technikákkal 12 bit felbontású, párhuzamos átalakítók előállítása lehetséges. Ennél tehát 4095 komparátort és a referenciafeszültségek előállításához szükséges szerkezeti elemeket, az átkódoló, valamint a kimeneti tárolót kell egy chipre integrálni. Ezeknek a flash konvertereknek a tipikus frekvenciája 100 – 200 MHz értékű. A hozzá tartozó átalakítási idők tehát 5 – 10 ns értékűek.

7.2.3.2. Kompenzációs elv szerinti A/D átalakítás

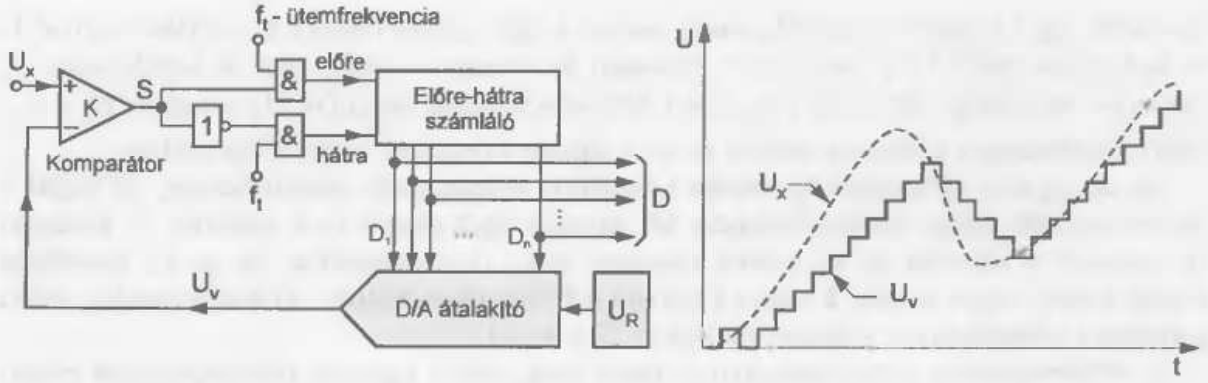
A kompenzációs elv szerinti analóg-digitális átalakító a *7.19. ábra* szerint egy D/A átalakítót tartalmaz a visszacsatolásban. Egy kiegyenlítő kapcsolás segítségével a D/A átalakító digitális *D* bemeneti jelét megfelelő módon addig változtatják, amíg az U_v analóg kimeneti jel gyakorlatilag teljes mértékben kompenzálja az átalakítandó U_x analóg bemeneti jelet. A szükséges S vezérlőjelet a kiegyenlítő kapcsolás egy olyan *K* komparátortól kapja, mely logikai **1** jelet szállít mindaddig, amíg az átalakítandó U_x bemeneti feszültség nagyobb, mint a visszacsatolt U_v összehasonlítási feszültség. Kiegyenlített állapotban a D/A átalakító digitális bemeneti jele azonos a teljes A/D átalakító digitális kimeneti jelével.



7.19. ábra. Kompenzációs rendszerű A/D átalakító elve

Követő átalakító kétirányú számlálóval

A kompenzációs elv szerinti legegyszerűbb A/D átalakító az egyirányú számlálóval rendelkező növekményes átalakító. Mivel az ilyen átalakítók vagy csak növekvő vagy csak csökkenő bemeneti feszültséget tudnak követni, ezért általában kétirányú számlálóval rendelkező növekményes átalakítót szoktak beépíteni. Ezek a követő átalakítók mind növekvő, mind csökkenő bemeneti jeleket tudnak követni.



a) elvi kapcsolás *b) idődiagram*

7.20. ábra. Követő átalakító kétirányú számlálóval

A *7.20. ábra* egy kétirányú számlálóval rendelkező követő átalakító tömbvázlatát és idődiagramját szemlélteti. Az áramkör működése az ábra alapján magyarázható.

- Megfelelő logikával a számláló előreszámoló bemenetét addig vezérlik, amíg a bemeneti jel nagyobb, mint a D/A átalakító visszacsatolt U_v jele.
- A reverzibilis számláló visszaszámláló bemenetét akkor vezérlik, ha az átalakítandó bemeneti feszültség kisebb, mint U_v. A digitális kimeneti jel mindig egy egy egységgel ugrik ide-oda, mivel a két számlálóbemenet egyikén állandóan ütemimpulzusok lépnek fel. Ezt az ide-oda ugrálást el lehet kerülni, ha komparátorként ún. ablakkomparátort használnak, amely egy meghatározott holtzónán belül az egyik számláló bemenetét sem vezérli.

A *7.20.b ábra* szerinti jelleggörbe megmutatja, hogyan követi egy követő átalakító a bemeneti feszültség növekedését illetve csökkenését. Ha a rendszer maximális átalakítási sebességét túllépjük, csak akkor követi késéssel az átalakító az U_x változó feszültséget.

Ha a követő átalakítónak késleltetésmentesen kell tudnia a jelet követnie, akkor a bemeneti feszültség maximális változási sebességének nem szabad az összehasonlítási feszültség maximális változási sebességénél nagyobbnak lennie.

Példa: – Egy 10-bites átalakító esetén, ha az f_i *ütemfrekvencia* 1 MHz és a bemeneti feszültség váltakozó összetevőjének amplitúdója $\frac{1}{2}U_R$, a bemeneti feszültség maximális, megengedhető frekvenciája kb. 310 Hz.

Fokozatos közelítésű A/D átalakító

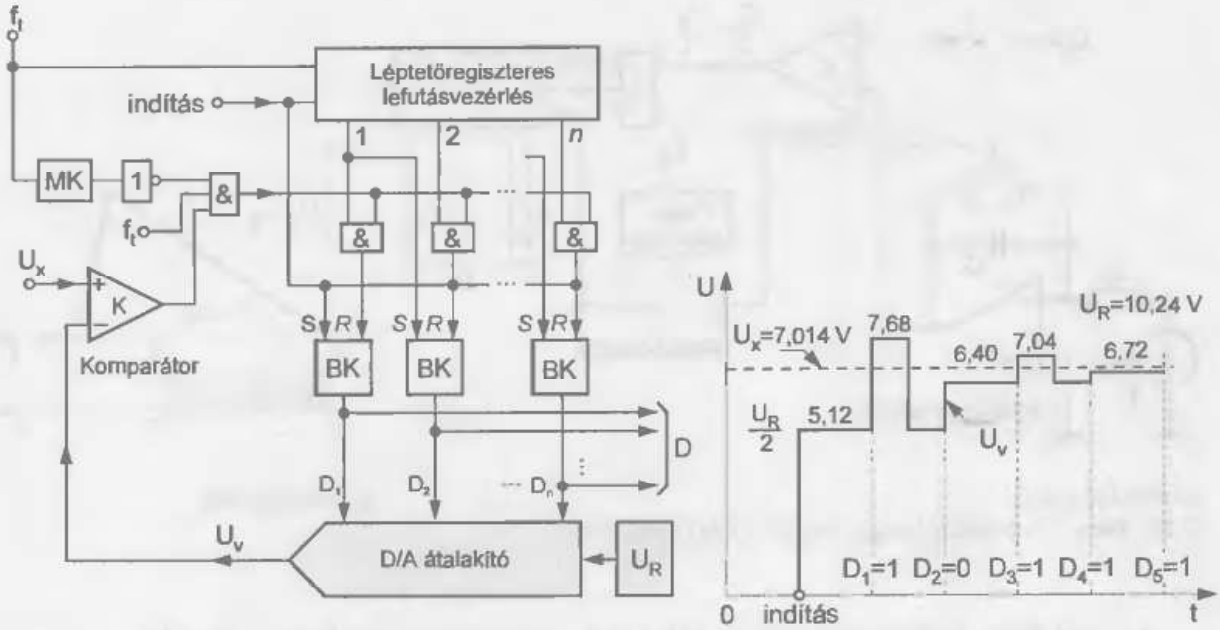
Az analóg-digitális átalakítási eljárások között nagyon elterjedt a fokozatos közelítés módszere. Ezek az átalakítók az olyan ütemvezérlésű, soros átalakítókhoz tartoznak, melyeknél minden ütemperiódusban a *D* digitális kimeneti jel egy helyértéke alakul ki (angolul: *one bit at a time*). Egy *n*-bites átalakítónál tehát az átalakításhoz *n* lépés szükséges.

Egy fokozatos közelítés elve (angolul: *successive approximation – szukcesszív aproximációs*) szerint működő A/D átalakító tömbvázlatát a *7.21.a ábra* mutatja.

Az átalakító működésének alapjául a dichotómia elve szolgál, vagyis a mért érték sorozatos összehasonlítása a lehetséges maximális érték (U_R) $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ stb. részével. Az átalakítás egy kísérlettel kezdődik, amely szerint a legmagasabb értékű helyértékre logikai **1**-et kell beírni. Ha a D/A átalakító U_v kimeneti feszültsége kisebb, mint az átalakítandó U_x bemeneti feszültség, akkor ezt a logikai **1** értéket a rendszer megőrzi. Ha azonban $U_x < U_v$, akkor a komparátor kimenete átbillen és ezt a lépcsőt a rendszer visszaállítja nullára.

Ez az eljárás folytatódik az ezután következő legmagasabb értékű helyen, és végül a legalacsonyabb értékű helyen fejeződik be. Minden lépés után a D/A átalakító U_v kimeneti feszültségét a rendszer az U_x analóg bemeneti jellel összehasonlítja. Ha az U_x feszültség kisebb értékű, akkor logikai **1** marad a bistabil *BK* billenőfokozatban. Túlkompenzálás esetén azonban a billenőfokozat visszaáll nullára (*7.21.b ábra*).

A lefutásvezérlést léptetőregiszterrel oldják meg, amely egyrészt túlkompenzálás esetén nyitja az *ÉS*-kaput a billenőfokozat törlésére, másrészt pedig az ezt követő kisebb értékű hely billenőfokozatát logikai **1**-re állítja. A monostabil *MK* billenőfokozat a komparátor jelét elegendő hosszú ideig késlelteti ahhoz, hogy ezáltal az átmeneti folyamatok beállása kivárható legyen.



a) elvi kapcsolás *b) idődiagram*

7.21. ábra. A/D átalakító fokozatos megközelítéssel

A *7.21.b ábrán* $U_x = 7{,}014\,\text{V}$ bemeneti feszültség példáján bemutatjuk az átalakítás kezdetét $U_R = 10{,}24\,\text{V}$ referenciafeszültség esetén.

A gyors átalakítók ezen elv szerint 1 MHz-es ütemfrekvenciával dolgoznak. Ez 1 µs-os ütemperiódusnak felel meg. Egy 10-bites jel átalakítására ekkor 10 µs szükséges

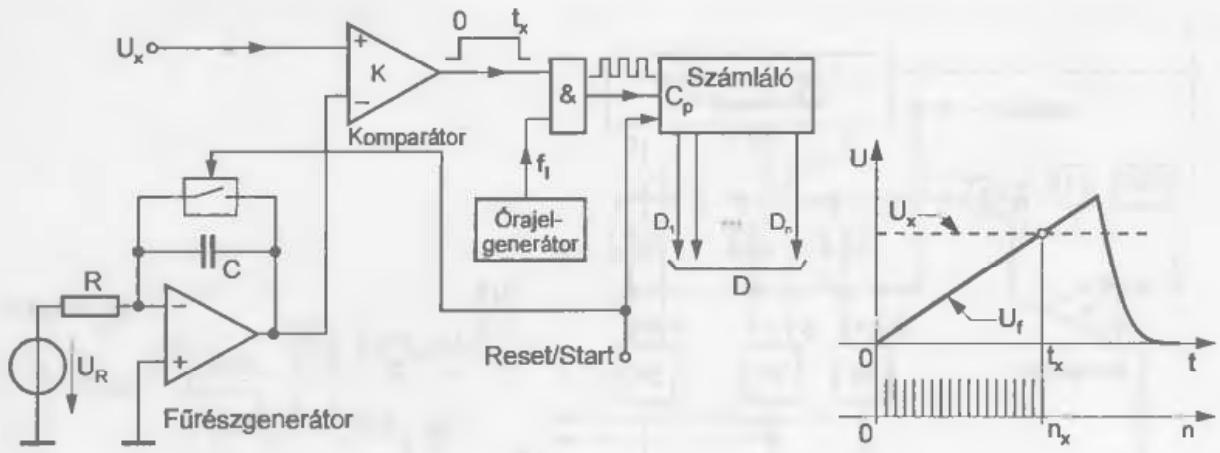
7.2.3.3. Analóg-digitális közvetett átalakítás

Egy sor gyakorlati alkalmazási esetnél, (pl. laboratoriumi digitális voltmérőknél) az analóg-digitális átalakítás sebességével szemben nem támasztanak nagy követelményeket. Ezeknél előnyösen lehet alaklamazni azokat a közbenső mennyiségként idővel vagy frekvenciával dolgozó átalakítási eljárásokat, amelyek részben nagyon nagy pontosságot tesznek lehetővé.

Fűrészfeszültség-eljárással működő közvetett A/D átalakító

Az időtranszformációs átalakítók legegyszerűbb változata a fűrészfeszültség-eljárással működő változat. A *7.22. ábrán* látható tömbvázlata nem tartalmaz D/A átalakítót. Működése azon az elven alapul, hogy az átalakítandó bemeneti feszültséget először értékével arányosan idővé alakítjuk.

Az átalakítás a fűrészjelet előállító integrátor elindításával kezdődik. A komparátor kimenete ilyenkor logikai **1**-es értéken van, és az órajelgenerátor által szolgáltatott impulzusokat – a nyitott *ÉS*-kapun keresztül – egy előre-számláló számolja. Amikor az U_f lineáris fűrészfeszültség eléri a bemeneti jel U_x értékét, a komparátor kimenete logikai **0**-ra vált és letiltja az *ÉS*-kapun keresztül az órajel impulzusokat (vagyis a számlálás megáll).



a) elvi kapcsolás *b) idődiagram*

7.22. ábra. Fűrészfeszültséggel működő A/D átalakító

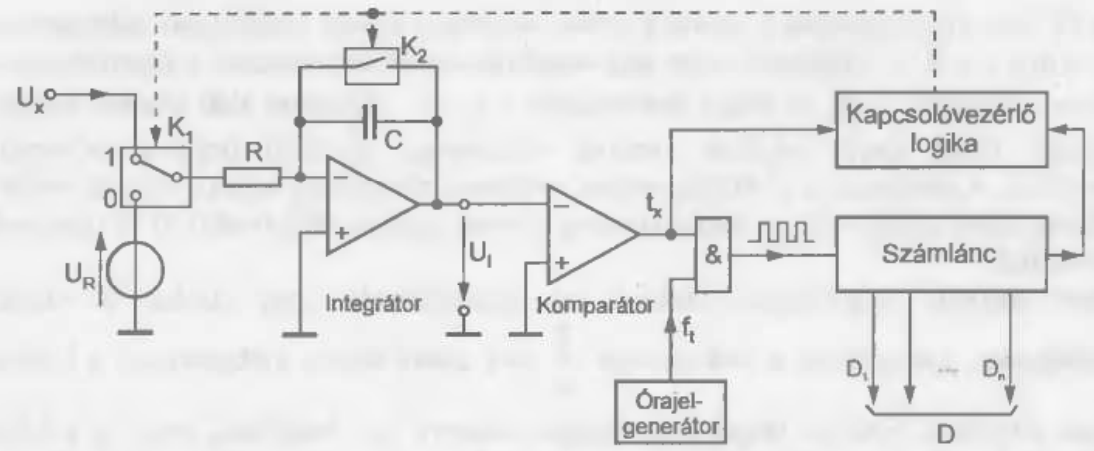
Az átalakítás kezdetétől eltelt t_x idő (és a számlálóban tárolt impulzusok n_x száma), egyenesen arányos a bemeneti U_x feszültség értékével. Minden mérés után nulláznunk kell a számlálót és a fűrészgenerátort is vissza kell állítani kezdeti állapotába.

Megjegyzés: A fűrészgenerátor kisebb módosításával lehetőség nyílik pozitív és negatív feszültségek átalakítására is. Ennek feltétele, hogy a fűrészgenerátor U_f feszültsége negatívtól pozitív értékek irányában változzon.
Ha negatív feszültség kerül a bemenetre, akkor az U_f feszültség először a mérendő feszültséget éri el, majd a **0** feszültséget lépi át. Ebből a sorrendből adódik a mért feszültség előjele. A mérés időtartama nem függ a vizsgált feszültség előjelétől, csak az abszolút értékétől.

A fűrészfeszültséggel működő átalakítók pontossága nagymértékben függ a felhasznált alkatrészek minőségétől. A fűrészfeszültség jellemzőit az *RC* tag, a műveleti erősítő és az U_R referencia-feszültség együttesen határozza meg, ezért ezek minősége kritikusnak tekinthető. Ugyanakkor az átalakítás pontosságát ezen alkatrészek hőmérsékletfüggése és hosszú idejű stabilitása is befolyásolja (ez szükségessé teszi pl. az automatikus kalibrálást). Ezért 0,1 %-nál nagyobb átalkítási pontosság csak nehezen érhető el. Ezeknek az átalakítóknak hátránya, hogy zavarérzékenységük viszonylag nagy (mivel pillanatérték átalakítók), ami korlátozza a felbontóképességet, és rendszerint ezt 8...10 helyérték szintjén tartja.

Kettős meredekséggel integráló közvetett A/D átalakító

A kettős meredekségű integrálás (angolul: *Dual Slope Integration*) elvén működő átalakító a legelterjedtebb típusok közé tartozik, amely mentes az előző átalakító hibáitól.
A *7.22. ábra* egy ilyen – egyféle polaritású feszültség mérésére alkalmas – átalakító tömbvázlatát mutatja. A teljes működési ciklus két szakaszból áll. A mérés kezdetén a kapcsolóvezérlő logika a számláncot törli, a K_1 kapcsolót az 1-es állásba billenti és a K_2 kapcsolót pedig nyitja. Ezáltal az U_x mérendő feszültséget egy t_0 állandó ideig integrálják. Ha a mérendő jel pozitív, az integrátor kimenete negatív lesz, és a komparátor engedélyezi az órajelgenerátor jelét.

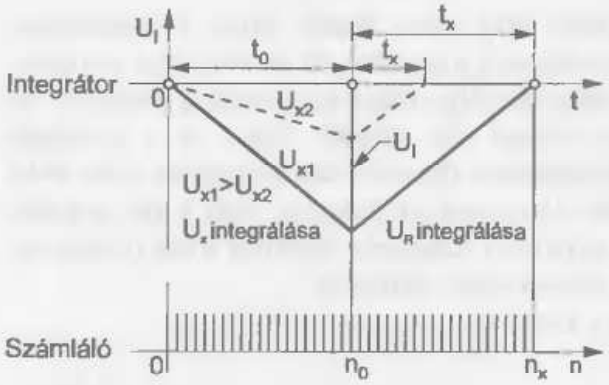


7.23. ábra. Dual slope A/D átalakító tömbvázlata

Ez a mérési időtartam akkor ér véget, amikor a számláló $Z_{max}+1$ periódus után túlcsordul, és ezáltal a számláló által mutatott érték ismét 0 lesz. Ezen szakasz végén az integrátor kimeneti feszültsége:

$$U_I = -\frac{1}{RC} \cdot U_x \cdot t_0 \qquad (7.1)$$

Ezt követően a negatív referenciafeszültség integrálására kapcsol át a vezérlő logika. Így a kimeneti feszültség abszolút értéke csökkenni kezd, amint ez a *7.24. ábrán* látható. Amikor az integrátor feszültsége eléri a zérust, a számlálás megszűnik, és a számlálóban az n_x végeredmény tárolódik.



7.24. ábra. Az integrátor kimeneti feszültségének időábrája

Megfigyelhető, hogy minél nagyobb az U_x feszültség, annál meredekebb a jel-integrálási szakasz, annál nagyobb a vissza-integrálás ideje és ezzel együtt az n_x értéke is.

$$t_x = \frac{U_I}{U_R} \cdot RC \qquad (7.2)$$

A *7.1* és *7.2* egyenleteket felhasználva kapjuk a kettős meredekséggel integráló átalakítók alapegyenletét:

$$-\frac{U_x}{U_R} = \frac{t_x}{t_0} = \frac{n_x}{n_0}. \qquad (7.3)$$

A számlálóban tárolt végeredmény: $$n_x = -n_0 \cdot \frac{U_x}{U_R}. \qquad (7.4)$$

A (7.3) és (7.4) egyenletek szerint a kettős integrálási eljárás szembeötlő jellegzetessége, hogy sem a $\tau = R \cdot C$ időállandó, sem az f_t órajelfrekvencia nem szerepel a végeredményben. Csak az szükséges, hogy az órajel frekvenciája a $t_0 + t_x$ időtartam alatt állandó legyen. A szükséges rövid idejű stabilitás azonban viszonylag egyszerű órajel-generátorral is biztosítható. A pontosságot tehát lényegében a referenciafeszültség toleranciája, az integrátor és a komparátor nullponthibája határozza meg. Emiatt ezzel az eljárással 0,01 % pontosságot is elérhetünk.

Igen előnyös tulajdonsága ezeknek az átalakítóknak, hogy azokat a váltakozó feszültségeket, amelyeknek a frekvenciája $\frac{1}{t_0}$-nak egész számú többszörösei, a kapcsolás teljesen elnyomja. Ezér az órajel frekvenciáját célszerű úgy beállítani, hogy t_0 a hálózati váltakozó feszültség periódusidejével vagy annak egész számú többszörösével legyen egyenlő. Ekkor a búgófeszültség hatását teljesen kiküszöbölhetjük.

Mivel a kettős integrálási eljárás olcsó és pontos, zavarelnyomása is nagy, ezért főként digitális feszültségmérőkben használják. Ezeknél a viszonylag nagy átalakítási idő nem zavaró. A *7.23. ábrán* látható számlálónak nem kell feltétlenül bináris számlálónak lennie, a működési elve változatlan, ha pl. *BCD* számlálót alkalmazunk. Ezt a lehetőség ki is használható a digitális voltmérőkben, mert ekkor nem kell bináris-decimális átalakítót használni.

A vizsgált A/D átalakító működési sebessége a feszültség-frekvencia átalakító ismert maximális frekvenciája mellett a szükséges helyértékek számával határozható meg.

Feszültség-frekvencia közvetett átalakítók

Az ilyen rendszerű A/D átalakítók működési elve azon alapul, hogy a feszültséget előzetesen átalakítja a feszültséggel arányos frekvenciájú sorozattá: ezt az impulzus sorozatot a vezérlőegységben rögzített mérési idő alatt megszámlálja, majd ez a szám a bemeneti jel digitális ekvivalenseként kerül kivitelre (*7.25 ábra*). Jól látható, hogy ez a folyamat megegyezik a bemeneti jel integrálásával egy ugyanolyan rögzített időintervallum alatt, ezért ez az AD átalakító az integráló átalakítók olyan előnyeivel rendelkezik, mint a kis sztatikus hiba és a nagy zajtűrő képesség. Az átalakító működési sebessége azonban kicsi (Hz-ekben mérhető), ami eleve meghatározza az átalakító felhasználási területeit.

Az átalakító működése a következő képlettel írható le:

$$N_x = f \cdot t_x = \alpha \cdot U_x \cdot t_x , \qquad (7.5)$$

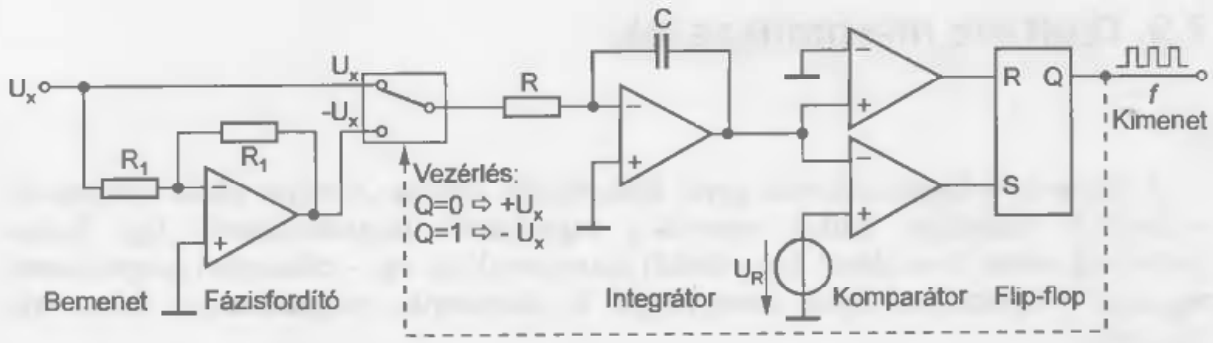
ahol: N_x – a digitális jelnek megfelelő szám;

$\alpha = \frac{f}{U_x}$ – a feszültség-frekvencia átalakítási jelleggörbe meredekségének inverze;

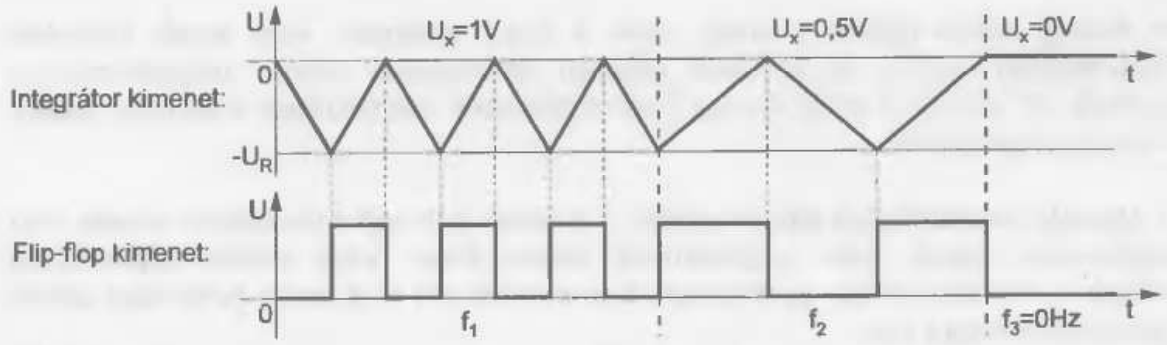
f – a feszültség-frekvencia átalakító egység kimeneti frekvenciája;

t_x – a mérési időintervallum.

Mint ahogyan a (7.5) összefüggésből következik, az A/D átalakító sztatikus hibáját a feszültség-frekvencia átalakításnak a hibája és a t_x mérési intervallum hibája határozza meg. Ennek következtében pontosságát a gyakorlatban R, C és U_R közvetlenül befolyásolja. Az ilyen A/D átalakítók előnye, hogy ezekben lényegében az analóg és a digitális rész elválasztódik, és működésük aszinkron.



a) elvi kapcsolási rajz



b) az integrátor és a flip-flop kimeneti feszültségének időábrája

7.25. ábra. Feszültség-frekvencia átalakító

Ez lehetővé teszi az A/D átalakító és az adatfeldolgozási rendszerbe tartozó számítástechnikai eszközök közötti kapcsolat könnyű megszervezését. Ennek érdekében szétválasztható az A/D átalakító analóg és digitális része úgy, hogy az analóg részt az analóg adók mellett helyezik el, a digitális részt pedig a számítástechnikai eszközök bemenetén. És végül: optikai illesztőáramkörök felhasználásával az A/D átalakítók analóg és digitális részeit galvanikusan szét lehet választani, és biztosítani lehet kapcsolatukat olyan feltételek között, amikor ezek a részek különböző feszültségen működnek.

7.2.4. Analóg-digitális átalakítók jellemzői

• ***Felbontás***: – a kvantálási intervallumok maximális számát képviselő *bitek számával* adható meg; ugyanakkor egy A/D átalakító felbontása kifejezhető egy kimeneti egységnek megfelelő bemeneti lépcsővel is.

• ***A bemeneti feszültség változásának tartománya***: – az A/D átalakító bemenetén jelentkező analóg mennyiség változásának maximális tartományát képviseli.

• ***Nemlinearitási hiba:*** – az A/D átalakító valós karakterisztikájának az ideális karakterisztikától való maximális eltérését fejezi ki; a nemlinearitási hiba mértékegysége az *LSB*.

• ***Átalakítási sebesség:*** – azt az időintervallumot adja meg, amely egy átalakításhoz szükséges; mértékegysége [μs] vagy [ms]; az átalakítási sebesség kifejezhető még az időegységenkénti átalakítások számával is.

• ***Offszethiba:*** – az átviteli karakterisztikának a kezdőponthoz viszonyított eltérése; az offszethiba mértékegysége az *LSB*.

7.3. Digitális mérőműszerek

A mérés az információszerzés egyik legfontosabb forrása. A mérés révén információt nyerhetünk valamilyen fizikai mennyiség nagyságáról, megváltozásáról. Egy fizikai mennyiség mérése azon alapul, hogy értékét összehasonlítjuk egy – célszerűen megválasztott nagyságú – ugyanolyan fizikai mennyiséggel és számszerűen meghatározzuk ahhoz való viszonyát.
Egy fizikai mennyiség mérése kétféle módszer szerint történhet.

- ***Analóg mérési eljárás***: – amely során a fizikai jellemzőt, vagy annak változását folyamatosan mérjük, és a fizikai jellemző tetszőlegesen kicsiny megváltozásához tartozik egy új mérési érték. Pl. egy Deprez-műszerrel megvalósított árammérés analóg eljárásnak tekinthető.

- ***Digitális (mintavételes) mérési eljárás***: – a fizikai jellemző változásának mérése nem folyamatos, hanem csak meghatározott időpontokban (vagy időintervallumokban) történik. A *mintavételezés* gyakorisága, a *mintavételi idő* és a mérés pontossága között szoros összefüggés van.

Ha összehasonlítjuk a kétféle mérési eljáráson alapuló mérőműszereket, akkor jellemzőiket kell megvizsgálni.

Pontosság

Az analóg műszerekkel elérhető mérési pontosság általában 1 ÷ 3 % körül van (speciális műszerek esetén ennél jobb is lehet). A digitális műszerek egyik legnagyobb előnye a nagy mérési pontosság, amely természetesen a kvantálás finomságától függ. Frekvenciamérés esetén 10^{-6} ÷ 10^{-9}, más fizikai mennyiségeknél 0.01 ÷ 0.001 % körüli pontosság valósítható meg.

Kijelzés

Analóg mérés esetén a mutatós műszer leolvasása leolvasási hibát eredményez. Ez a hiba a számjegykijelzésű digitális műszereknél nem jelentkezik, mivel a mért érték leolvasosása során nem követhetünk el hibát.

A mért adatok feldolgozhatósága

Analóg műszerek esetén a jelek továbbítása és feldolgozása körülményes, általában csak manuális úton valósítható meg. Digitális műszerek esetén az információ digitális jelek formájában rendelkezésre áll, így ezekkel műveleteket végezhetünk, információveszteség nélkül feldolgozásra továbbíthatjuk, és számítógépes rendszerrel összekapcsolva bonyolult feladatokat hajthatunk végre.

7.3.1. Digitális feszültségmérők

7.3.1.1. A digitális feszültségmérők felépítése

A digitális feszültségmérők (*angolul: Digital VoltMeter = DVM*) a mérendő analóg jelet (egyenfeszültséget) digitális információvá (számjegyekké) alakítják. Ennek megfelelően minden digitális feszültségmérő tartalmaz egy analóg-digitális átalakítót. Egy analóg-digitális átalakító önmagában még nem műszer, mivel a megfelelő feszültségtartományban való méréshez, az eredmények kijelzéséhez és más szolgáltatásokhoz egyéb kisegítő áramköri elemek is szükségesek. Egy digitális feszültségmérő egyszerűsített tömbvázlatát a *7.21. ábra* mutatja. A részegységek feladatai az ábra alapján tárgyalhatók:

- A bemeneti fokozat feladata az analóg jel megfelelő impedancián való fogadása manuálisan beállítható, vagy automatikusan beálló méréshatárral.
- A kijelző egység feladata az A/D átalakító digitális kimenő jelének átalakítása leolvasható decimális értékké.
- A vezérlő egység irányítja az egész digitális feszültségmérő működését. A modern műszerek vezérlőegysége sok, az automatizált méréshez tartozó feladatot is el tud látni. Ilyen például az *automatikus méréshatár váltás*, az *automatikus nullázás*, az *automatikus kalibrálás*, a mérési eredményekkel való *műveletvégzés*, a kijelzőn járulékos információk megjelenítése, valamint a kapcsolattartás a külvilággal, amelynek segítségével lehetőség nyílik automatizált mérőrendszer építésére. A bonyolult, intelligens műszerek vezérlőegysége ma már kivétel nélkül mikroprocesszoros (mikroszámítógépes) vezérlésű.

7.22. ábra. Digitális feszültségmérő tömbvázlata

A digitális feszültségmérők közvetlenül csak egyenfeszültséget mérnek. Ha váltakozó feszültséget is szeretnénk mérni, akkor a műszer elé egy AC/DC átalakítót (egyenirányítót) kell kapcsolni. A váltakozó feszültség leggyakrabban használt jellemzője az *effektív érték*. Ennek mérése általában bonyolult és költséges, ezért csak kevés digitális feszültségmérő alkalmas *valódi effektív érték* mérésére. Legtöbbjükben átlagértékmérő kapcsolás található, de a kijelzés effektív értékre átszámolt. Ezek azonban csak tiszta szinuszos jelek esetén mutatnak helyes értéket.

A digitális feszültségmérők tartalmazhatnak ellenállás/feszültség konvertert is, amivel ellenállásmérés valósítható meg. Az ellenállás értékének a meghatározása ismert áram (I_1) által létrehozott feszültségmérés útján valósítható meg:

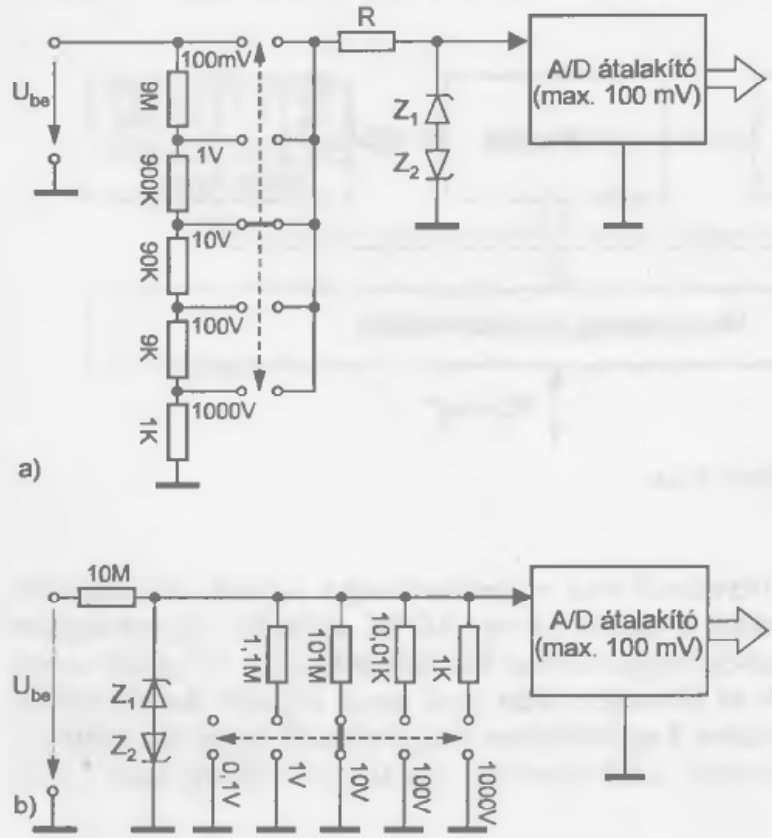
$$R = \frac{U}{I} \quad \Rightarrow \quad R_x = \frac{U_x}{I_1} = k \cdot U_x \qquad \text{ahol:} \quad k = \frac{1}{I_1} = állandó$$

R_x – a mérendő ellenállás értéke
U_x – a mért feszültség értéke

Hasonló módon árammérés is megvalósítható. A különböző típusú A/D átalakítók alkalmazásával a digitális műszer pontossága – váltakozó feszültség és ellenállás mérésnél – csökken, és lényegesen rosszabb, mint egyenfeszültség mérésénél. Az ilyen. több mérési célra is használható műszert *digitális multiméternek* nevezik.

7.3.1.2. Bemeneti fokozatok

Az analóg-digitális átalakítók csak meghatározott feszültségtartományban képesek egyenfeszültséget fogadni. Az átalakító bemeneti feszültsége általában ± 0,1 V, vagy ± 1 V között változhat). Ezért a bemenetre a méréshatár változtatása céljából osztót, vagy osztóval kombinált erősítőt kell elhelyezni. Minden esetben gondoskodni kell a megfelelő túlfeszültség elleni védelemről is. A *7.23. ábra* passzív egyenfeszültségű méréshatár váltó kapcsolásokat szemléltet.

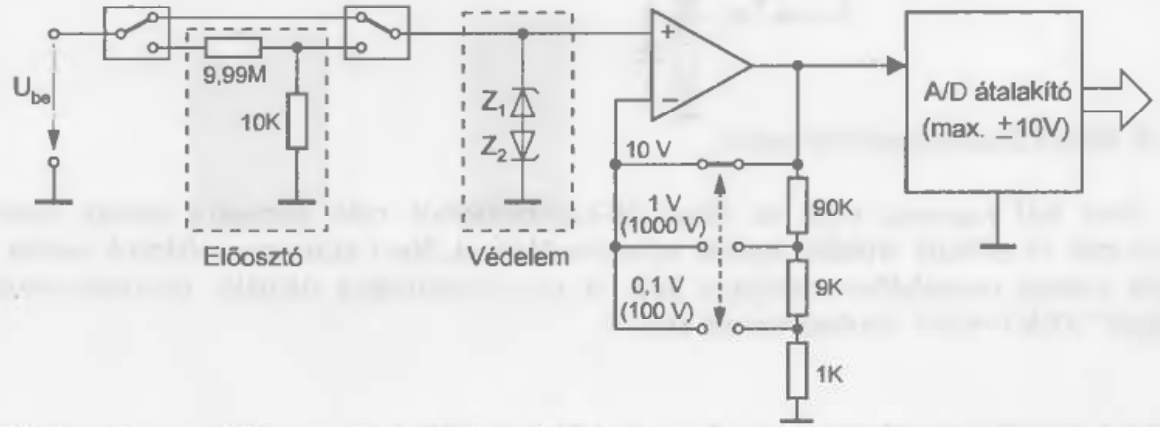


A *7.23.a ábrán* látható kapcsolás előnye, hogy bemeneti ellenállása minden méréshatárban azonos (10 MΩ). A Z_1 és Z_2 – kis maradékáramú – Zener-diódák a túlfeszültség ellen védenek.
A *7.23.b ábrán* látható kapcsolás előnye, hogy a Zener diódák feszültségkorlátozó hatása miatt a méréshatár váltó kapcsolókon nem lép fel nagy feszültség, ezért ez a kapcsolás alkalmas félvezetős kapcsolókkal való megvalósításra. Ez az automatikus méréshatár váltásnál nagy előnynek tekinthető.

7.23. ábra. Passzív (DC) méréshatár-váltó kapcsolások

A nagy pontosságú laboratóriumi digitális feszültségmérőkben gyakran alkalmaznak ± 10 V végkitérésű A/D átalakítókat. Ebben az esetben szükség van olyan precíziós bemeneti erősítőre, amely képes teljesíteni a kellő pontossági követelményeket. Egy ilyen kapcsolás tömbvázlatát szemlélteti a *7.24. ábra*. A bemeneten egy előosztó van (általában 0,1 Ω ellenállást beiktató jelfogók biztosítják az átkapcsolást) és az ezt követő „programozható" erősítő áramkör biztosítja a minden méréshatárban beállítható megfelelő kimeneti feszültséget. Megfigyelhető, hogy a feszültségmérő áramkör bemeneti ellenállása nem állandó:

– az előosztó bekapcsolt állapotában 10 MΩ, kikapcsolt állapotában (a 10 V és az alatti bemeneti feszültség-tartományokban) $10^9 \div 10^{12}$ Ω is lehet.



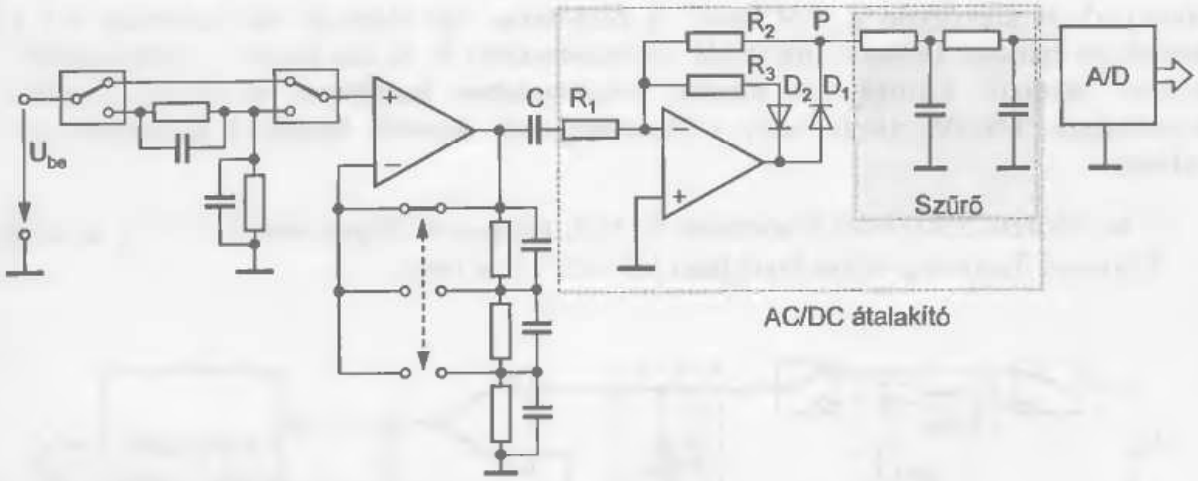
7.24. ábra. Bemeneti fokozat erősítő áramkörrel

A bemeneti erősítőt a megfelelő pontosság elérése érdekében általában ellátják automatikus nullázással (angolul: autozero) is. Kihasználható, hogy az A/D átalakító működése szakaszos, ezért amikor éppen nem történik mintavételezés, lehetőség van a bemeneti erősítő nullázására.

Váltakozó feszültség mérése esetén a bemeneti fokozat jelét egy nagypontosságú egyenirányítón (AC/DC átalakítón) kell átvezetni, mielőtt az A/D átalakító bemenetére csatlakozunk. A *7.25. ábra* egy váltakozó feszültség mérésére alkalmas bemeneti áramkör egyszerűsített kapcsolási rajzát mutatja. A pontos *AC* erősítés-leosztás érdekében a méréshatár váltó ellenállásokkal párhuzamosan kötött kompenzáló kondenzátorokat kell beiktatni (az elemek *RC* időállandójának azonosnak kell lennie). A műveleti erősítővel kivitelezett átlagérték-egyenirányító a D_1, D_2 diódán nyitóirányban létrejövő kb. 0,6 V feszültségesést gyakorlatilag nullára csökkenti. Mivel a dióda a negatív visszacsatoló ágban van, R_2 kimeneti oldalán pontosan a bemeneti jel negatív félperiódusának „másolatát" kapjuk $-\frac{R_2}{R_1}$-szeres erősítéssel. Ennek *R-C* tagokkal szűrt középértékét mérjük az A/D átalakítóval. Az erősítést mindig úgy választják meg, hogy szinuszos jel mérésekor annak effektív értékével egyező középérték-egyenfeszültség keletkezzen:

$$\frac{U_{be}}{\sqrt{2}} = \frac{\hat{U}_{ki}}{\pi} \quad \Rightarrow \quad A = \frac{\hat{U}_{ki}}{\hat{U}_{be}} = \frac{\pi}{\sqrt{2}} = 2{,}22$$

Vagyis egyutas egyenirányítás esetén $R_2 = 2{,}22 \cdot R_1$-nek kell lennie.



7.25. ábra. Váltakozó feszültség mérése

Meg kell jegyezni, hogy az átlagérték-egyenirányítók csak szinuszos jelalak esetén képeznek az effektív értékkel egyező egyenfeszültséget. Nem szinuszos jelalakok esetén a mért értéket megfelelően korrigálni kell. A nagy pontosságú digitális feszültségmérők „igazi" effektív-érték mérővel vannak ellátva.

7.3.1.3. Digitális feszültségmérő műszerek A/D átalakítói

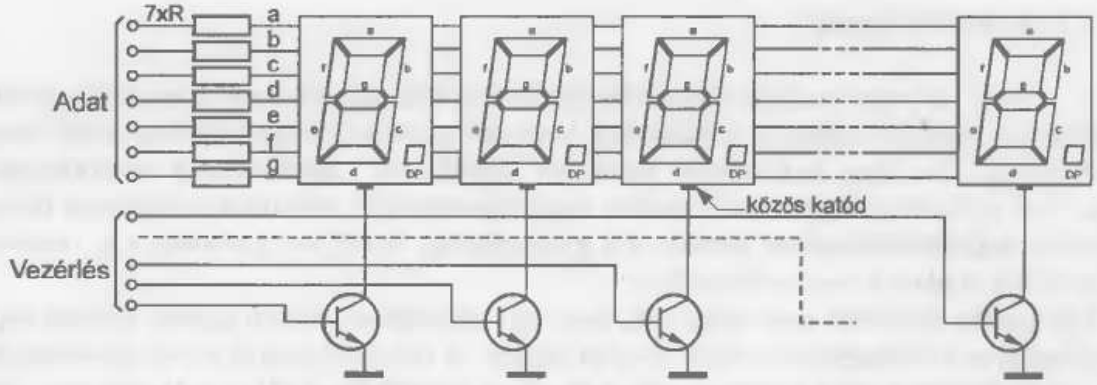
A digitális feszültségmérő műszerekben a leggyakrabban integráló típusú A/D átalakítókat alkalmaznak, mivel ezek a legkevésbé érzékenyek a zajra, a hálózatból eredő búgófeszültségre. A mai modern, nagypontosságú műszerek mikroprocesszoros vezérlésűek és a legtöbbször a processzor is aktívan részt vesz az analóg-digitális átalakítás folyamatában. Olyan műszerek esetében amikor fontos a nagyon nagy felbontás és a nagy sebesség, kompenzáló típusú átalakítókat építenek be.

7.3.1.4. Optikai kijelző egységek

A két leggyakrabban alkalmazott optikai kijelző egység, amellyel a digitális feszültségmérő műszerekben találkozunk:

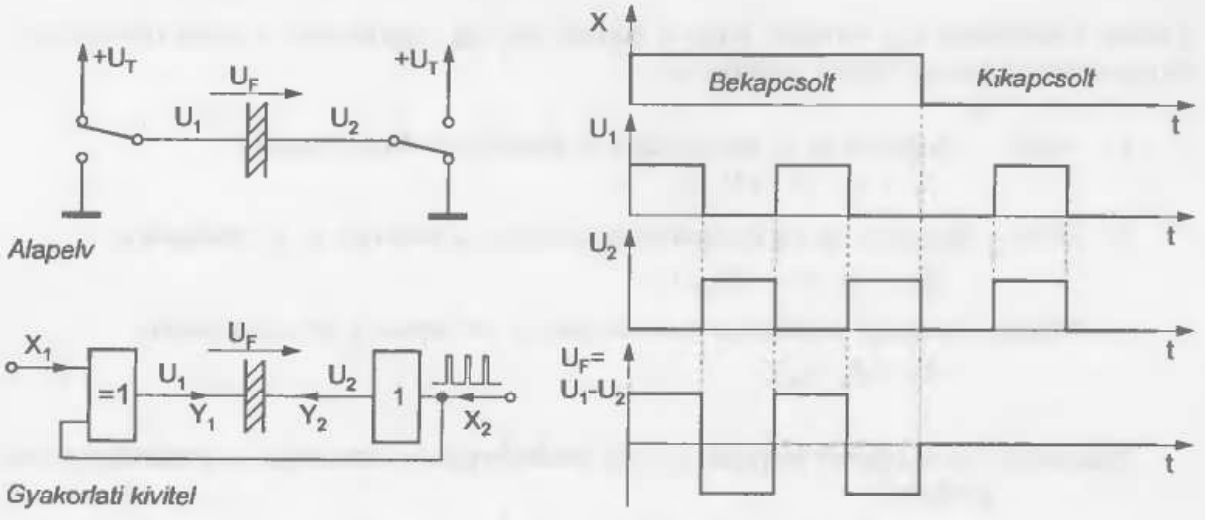
- a fénydiódás (*LED*-es) és a folyadékkristályos (*LCD*) kijelzők numerikus és alfanumerikus megjelenítésre alkalmas változatai.

A fénydiódás kijelzők előnyének tekinthető – a passzív folyadékkristályos változatokhoz viszonyítva –, hogy saját fényt bocsátanak ki és ezért rosszabb megvilágítási körülmények között is könnyen leolvashatók. Hátrányuk viszont, hogy meghajtásukhoz relatív nagy áram szükséges (szegmensenként 4÷20 mA). Nagyszámú kijelzőelem gazdaságos vezérlése multiplexeléssel oldható meg (*7.26.a ábra*).



7.26. ábra. Hétszegmenses kijelzők (*LED*) multiplex vezérlése

A passzív folyadékkristályos kijelzők eltérően a fénydiódáktól nem bocsátanak ki fényt, hanem csak külső megvilágítás esetén láthatók. Előnyük viszont a nagyon kicsi (gyakorlatilag elhanyagolható) teljesítmény-felhasználás (ez főleg térvezérlésű *LCD* esetében van így). A térvezérlésű folyadékkristályos kijelzők különösen alkalmasak emiatt telepes műszerekhez, amelyekben nagyon kritikus a kijelzőegység energia-felhasználása. A vezérléshez váltakozó feszültséget használnak (olyan szimmetrikus váltakozófeszültséget, amelynek egyen-középértéke nulla), mivel az egyenfeszültség elektrolízist indít el, ami a folyadékkristály élettartamát jelentősen csökkenti. A *7.27. ábrán* látható megoldásban, ha X_1=0, akkor Y_1=Y_2=X_2; tehát a kijelző mindkét kapcsolója az X_2 négyszögjel ütemében azonos fázisban kapcsol. X_1=1 esetén $Y_1 = \overline{X}_2$, és a kijelző ellenfázisú feszültséget kap. Az X_2 négyszögjel frekvenciája legalább akkora kell legyen, hogy a szem számára ne legyen érzékelhető a villódzó hatás.



a) megvalósítás *b) idő diagram*

7.27. ábra. Folyadékkristályos kijelző (*LCD*) vezérlése

Az utóbbi időben egyre nagyobb teret hódítanak az aktív mátrixos (*TFT*) folyadékkristályos kijelzők, amelyek energia fogyasztása valamivel nagyobb mint a passzív változatoké, de külső megvilágítás nélkül, nagy térszögben is jól leolvashatók.

7.3.1.5. A vezérlő egység

A vezérlő egység feladata a digitális feszültségmérő egységeinek irányítása, bizonyos automatikus funkciók ellátása, eredmények korrigálása, műveletvégzés és kapcsolat tartása a „külvilággal". Az hogy egy műszer mennyire „intelligens", alapvetően a vezérlőegységtől függ. Szó volt róla, hogy a mai modern nagyteljesítményű műszerek irányítását beépített egytokos mikroszámítógépre bízzák. Ez gyakorlatilag korlátlan lehetőségeket biztosít az automatikus funkciók megvalósításánál.

Egy példán keresztül megvizsgáljuk, hogy egy intelligens vezérlő egység miként képes a hagyományos feszültségmérés elvét megváltoztatni. A feszültségmérés során elkövetett hibát egy, az U_x mérendő jellel sorosan beiktatódó ofszet feszültséggel (U_{of}), valamint egy „hibás" erősítés paraméterrel (A_u) modellezhetjük (*7.28. ábra*).

7.28. ábra. „Intelligens" vezérlő egység működése

A hibák korrigálása úgy történik, hogy a vezérlő egység –ugyanazzal a mérő-rendszerrel – három egymást követő mérést végeztet el:

1. mérés: – megméri az U_x feszültséget és eltárolja az E_1 eredményt:
$$E_1 = A_u \cdot (U_x + U_{of}) .$$
2. mérés: – megméri az U_R referenciafeszültséget és eltárolja az E_2 eredményt:
$$E_2 = A_u \cdot (U_R + U_{of}) .$$
3. mérés: – megméri a kimeneti feszültséget 0V-ra kapcsolt bemenet esetén:
$$E_3 = A_u \cdot U_{of} .$$

Számítás: – az elvégzett mérések eltárolt eredményeiből kiszámítja az eredmény pontos értékét:

$$E = \frac{E_1 - E_3}{E_2 - E_3} = \frac{A_u \cdot (U_x + U_{of}) - A \cdot U_{of}}{A_u \cdot (U_R + U_{of}) - A \cdot U_{of}} = \frac{U_x}{U_R}$$

Megfigyelhető, hogy a műszer által kijelzett ***E*** eredmény pontossága csak az U_R referenciafeszültségtől függ.

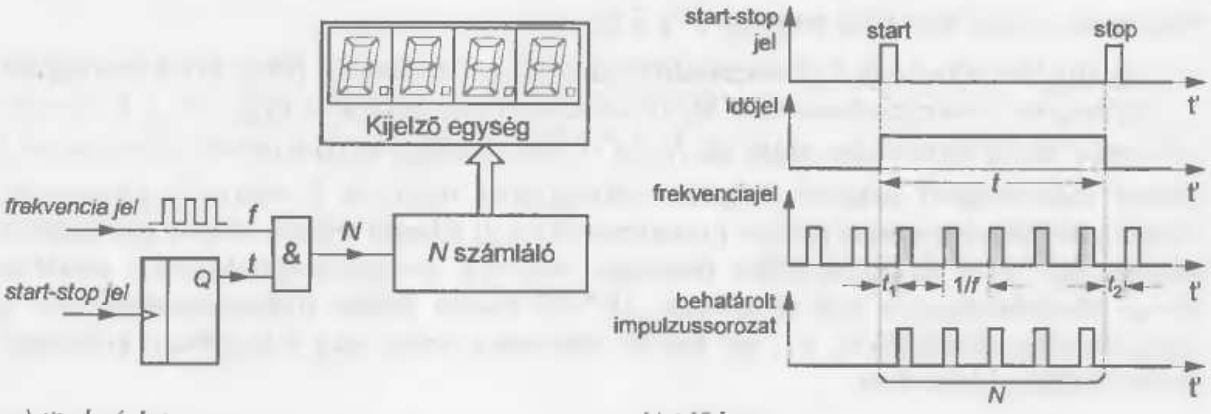
7.3.2. Digitális frekvencia- és időmérők

7.3.2.1. A digitális frekvencia- és időmérés elve

A digitális frekvencia- és időmérők univerzális műszerek. Különböző fizikai mennyiségeket lehet velük megmérni; így frekvenciát, periódusidőt, időtartamot, frekvencia-arányt, fázisszöget. Mivel ezeknek a mennyiségeknek a mérése ugyanazon elven történik, az alapvető funkciójú egységek megfelelő összekapcsolásával a kívánt mérések elvégezhetők.

A frekvencia és az ezzel összefüggő fizikai mennyiségek mérése a legpontosabban és a legtipikusabban megvalósítható digitális mérési eljárás. A műszer feladata jelimpulzusok (azaz kvantum-egységek) megszámlálása, amelyhez mérési hibát nem okozó digitális áramkörök (*számlálók*) állnak rendelkezésre.

Digitális időmérésnél egy ismert frekvenciájú jelből származó impulzusokat számlálnak ismeretlen ideig egy számlálóval. Digitális frekvenciamérésnél fordítva, ismeretlen frekvenciájú jelből származó impulzusokat számlálnak ismert ideig. A *7.29. ábra* a digitális frekvencia és időmérés elvét szemlélteti.



a) tömbvázlat *b) idődiagram*

7.29. ábra. A digitális frekvencia- és időmérés elve

A *7.29.a ábra* szerint addig jutnak a számlálandó impulzusok a frekvencia-bemenetről az impulzus-számlálóba, amíg logikai **1** jelet kapcsolnak az *ÉS*-kapu időbemenetére.

A *7.29.b ábra* szerinti időfüggvényben a flip-flop bemenetén t időtávolságra lévő indító és leállító jeleket, a t időtartamú jelet a flip-flop kimenetén, az f frekvenciájú illetve az $1/f$ periódusidejű frekvenciajelet, valamint az *ÉS*-kapu kimenetén azt a behatárolt impulzus-sorozatot tüntettük fel, amely az impulzusszámlálóba kerül. A frekvenciajelnél rövid ideig tartó impulzusokat feltételezve, az indítási időpont tetszés szerinti helyzeténél az időtartam az alábbi összefüggés szerinti:

$$t = N \cdot \frac{1}{f} + (t_1 - t_2) = N_{várt} \cdot \frac{1}{f},$$

ahol: N – az egész számú impulzusszáma annak a behatárolt impulzus-sorozatnak, melyet a számlálóra kapcsolnak;
$1/f$ – a frekvenciajel periódusideje;
t_1, t_2 – az indítójel illetve a leállítójel és a frekvenciajel legközelebbi impulzusa közötti rövid maradékidő.

Az $N_{várt}$ elvárt érték egy törtszám, amely azt adja meg, hogy milyen gyakran fordul elő a reciprok $1/f$ periódusidő a t mérési időben. Ha az előbbi egyenletet f frekvenciával megszorozzuk az alábbi összefüggést kapjuk:

$$f \cdot t = N + f \cdot (t_1 - t_2) = N_{várt}$$

Ha a maradékidők $t_1 - t_2$ különbségéhez képest a mérési idők nagyok, azaz $t >> t_1 - t_2$, akkor a számláló állása az

$$N = f \cdot t .$$

összefüggés szerinti.
Az abszolút kvantálási hiba:

$$F_q = N - N_{névleges} = f \cdot (t_2 - t_1) .$$

Mivel a $t_2 - t_1$ értéke az $1/f$ reciprok frekvenciánál nem lehet nagyobb, ezért a kvantálási hiba értéke egynél nem lehet nagyobb ($F_q \leq 1$).

Egy digitális idő- vagy frekvenciamérés pontossága a kvantálási hibán kívül lényegében az alkalmazott referenciafrekvencia ill. referenciaidő pontosságától függ. Ha a kvantálási hibát nem veszik figyelembe, akkor az $N = f \cdot t$ számlálóállás mind a mérési idővel, mind a mérési frekvenciával arányos. Digitális időmérésnél tehát az f referencia-frekvenciát, digitális frekvenciamérésnél pedig a t referenciaidőt kell állandó értéken tartani. Ezt mindkét esetben egy olyan kvarcoszcillátor biztosítja, melynek frekvencia-stabilitásával szemben magas követelményeket kell támasztani. 10^{-4}-nél kisebb relatív frekvencia-eltéréseket a legegyszerűbb eszközökkel, 10^{-8}-nál kisebb eltéréseket pedig még elfogadható költséggel (termosztálás) el lehet érni.

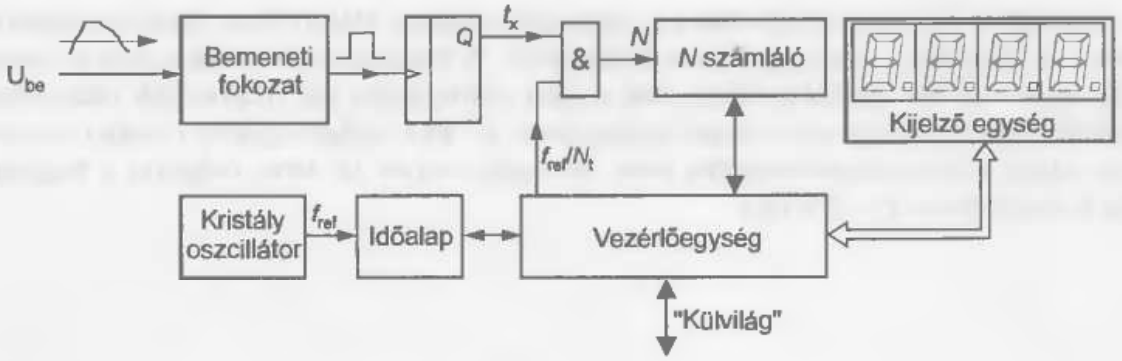
7.3.2.2. Digitális időmérés

Digitális időmérésnél a *7.30. ábra* szerint azokat az ismert frekvenciájú impulzusokat számlálják, melyeket a t_x mérendő idő alatt kapcsolnak egy számláló bemenetére. Mivel a bemeneti jel legtöbbször nem logikai szintű digitális jel, olyan bemeneti fokozatra kell vezetni, amely lehetőleg nagy bemenet ellenállással és megfelelő (beállítható) erősítéssel digitális impulzusjelet állít elő. A mérendő idő a

$$t_x = \frac{N_f}{f_{ref}} \cdot N$$

összefüggés szerinti. Az elérhető időfelbontás a kvarcfrekvenciától függ és pl. $f_{ref} = 10\,\text{MHz}$ és $N_f = 1$ esetén $\frac{1}{f_{ref}} = 0{,}1\,\mu s$.

Hosszabb idők méréséhez egy egész számú N_f osztástényezővel rendelkező, digitális frekvenciaosztót kapcsolnak a kvarcoszcillátor után.



7.30. ábra. Digitális időmérő tipikus tömbvázlata

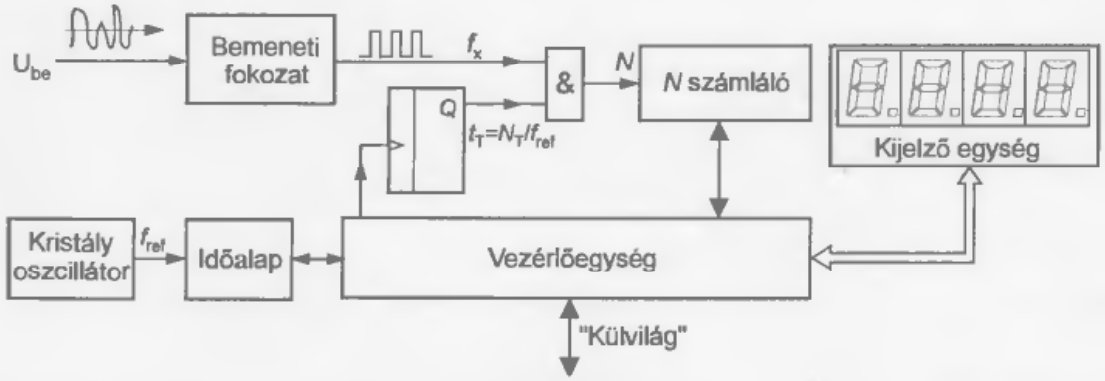
Egy jel periódusidejének digitális méréséhez a jelet először Schmitt-trigger segítségével négyszögjellé alakítják és azután, mint az időkülönbség mérésnél, indító és leállító jelek képzésére használják fel. Kis frekvenciákat előnyös periódusidő elven mérni, mert így kis mérési időket kapjunk

7.3.2.3. Digitális frekvenciamérés

Digitális frekvenciamérésnél a *7.31. ábra* szerint azokat az f_x ismeretlen frekvenciájú, N impulzusokat számlálják, melyeket a t_T ismert időn (kapuidő) keresztül kapcsolnak egy számláló bemenetére. A t_T kapuidő ebben az esetben a frekvencia periódusidejével egyenlő, amely úgy keletkezik, hogy az f_{ref} kvarcfrekvenciát digitálisan elosztják az N_T tényezővel. A $t_T = \frac{N_T}{f_{ref}}$ behelyettesítéssel az ismeretlen frekvencia az

$$f_x = \frac{N_T}{t_T} = \frac{f_{ref}}{N_T} \cdot N$$

összefüggésből adódik.



7.31. ábra. Digitális frekvenciamérő tipikus tömbvázlata

Az elérhető frekvencia-felbontás a t_T kapuidőtől (mérési időtől) függ. Gyakran dinamikai okokból a kapuidőt 1 s-ra vagy 10 s-ra korlátozzák. A frekvencia-felbontás ebben az esetben 1 Hz illetve 0,1 Hz. 10 MHz feletti, már a GHz tartományba eső frekvenciák méréséhez a mérendő frekvenciát egy gyors osztó segítségével, pl. *ECL*-technológiával (emitter-coupled-logic) olyan frekvencia-tartományba lehet leosztani, melyet fel lehet dolgozni a hagyományos technológiával (5-10 MHz).

Összefoglaló kérdések

1. Ismertesse a digitális –analóg átalakítók elvi működését!
2. Milyen elvek szerint lehet osztályozni a D/A átalakítókat és milyen típusai vannak?
3. Milyen előnyei és hátrányai vannak az összegző erősítővel működő D/A átalakítónak?
4. Ismertesse az ellenállás-létrahálózatos közvetlen D/A átalakító működési elvét!
5. Rajzolja le az impulzus-kitöltés modulációval működő közvetett D/A átalakító tömbvázlatát és ismertesse működési elvét!
6. Hogyan használják fel a digitális technikában egy jel frekvenciáját digitális értékként?
7. Ismertesse az analóg-digitális átalakítók elvi működését!
8. Mi a kompenzációs elv szerinti analóg-digitális átalakítás?
9. Ismertesse a kettős meredekséggel integráló közvetett A/D átalakító működési elvét?
10. Milyen jellemzői vannak az analóg-digitális átalakítóknak?
11. Hasonlítsa össze a mért adatok feldolgozhatósága szempontjából az analóg és a digitális műszereket!
12. Rajzolja le egy digitális feszültségmérő tömbvázlatát és ismertesse az egyes részegységek feladatát!
13. Milyen előnyei és hátrányai vannak a folyadékkristályos kijelzőknek?
14. Ismertesse a digitális frekvenciamérés elvét!
15. Ismertesse a digitális időmérés elvét!
16. Milyen tényezőktől függ a frekvencia és időmérés pontossága?

8. Mikroszámítógépek

8.1. Bevezetés

Az emberi tevékenység egyik fő célkitűzése a bennünket körülvevő világ megismerése. A természeti törvények és törvényszerűségek felfedezésének és tanulmányozásának gyökerei az ókori görögökig nyúlnak vissza. Ők kezdték el a mai értelemben vett tudomány alapjait lerakni. Különösen Püthagoraszról és a róla elnevezett pithagoreusok tevékenységéről kell megemlékeznünk. Az egész pithagorászi szemlélet a következő állításban foglalható össze: a dolgok természete, lényege a szám. A valóság és a matematika kapcsolatába vetett hitet konkrét eredményekkel támasztották alá. Püthagorasznak és iskolájának köszönhetjük a Pitagorasz-tételt, az irracionális számok létezésének bizonyítását, a lebegő, gömb alakú Föld képét, a zenei hangok és a húrok hossza közötti összefüggést, hogy csak a legfontosabbakat említsük.

Szintén a görögök nagy érdeme az a felismerés, hogy a törvények felírásához absztrakció, idealizáció kell. Úgy tűnik, hogy az absztrakció nehéz, de nagyon fontos lépés volt a tudományok fejlődésében. Az absztrakció a jelenség olyan leegyszerűsítése, amely a jelenség alapvető jellegét nem változtatja meg, ugyanakkor kvantitatív tárgyalásra alkalmas.

A matematikai modellalkotás szerepét elsőnek Galileo Galilei (1564–1642) ismerte fel. Az 1623-ban megjelent „Il Saggiatore" című munkájában a következőket írja: „*A filozófia abban a nagy könyvben van írva, amely nyitva áll mindenkor szemeink előtt: az univerzumra gondolok; de nem olvashatjuk mindaddig, míg meg nem tanultuk a nyelvét, és nem barátkoztunk meg a jelekkel, amelyekkel írva van. A matematika nyelvén van írva, és a betűi háromszögek, körök és más geometriai alakzatok, amelyek ismerete nélkül lehetetlen egyetlen szót is megérteni.*" A matematikai modellen végzett számításokról később James Clerk Maxwell (1831–1879) így ír: „*Minthogy minden matematikai tudomány a fizika törvényei és a számok törvényei közötti relációkon alapul, így az egzakt tudomány célja az, hogy a természetben fellépő problémákat mennyiségek meghatározására redukálja – számokon végzett műveletek segítségével.*"

A számítási eljárások gyakorlati megvalósításának megkönnyítésére az emberiség az idők folyamán, a civilizációs foknak megfelelően, különböző bonyolultságú segédeszközöket szerkesztett. Az első ilyen segédeszköz az úgynevezett abakusz volt, amelyet az ókori görögök és főként a rómaiak használtak. Bizonyos számolóköveket a helyi érték szerint csúszóvájatba illesztettek, és ide-oda tologatásukkal végezték el az egyszerű műveleteket.

A XVI. és különösen a XVII. században beindult órásmesterség erőteljes fejlődése adta az első döntő impulzust a számolási segédeszközök célszerűsítéséhez és tökéletesítéséhez. Az első, mai értelemben vett számológép születése az ismert francia tudós, Blaise Pascal (1623-1662) nevéhez fűződik. Pascal gépe, amelyet tizenkilenc évesen, 1642-ben néhány tehetséges órással együttműködve épített meg, egy folyamatos tizes áttételű mechanikus összeadó- és kivonógép volt. Később, 1671-ben G. W. Leibniz (1646-1716) egy olyan számológépet szerkesztett, amely szorozni is tudott.

Továbbá megemlítjük Ch. P. Babbage (1792-1871) angol matematikus munkásságát; ő 1822-ben készített egy olyan gépet, amellyel táblázatokat lehetett kiszámítani. Később, 1825-ben, egy ún. „analitikus gép" tervét vetette fel, amely már sok, napjaink elektronikus számítógépjeire jellemző megoldást tartalmazott.

A gépnek 1000 szavas tára lett volna, egyenként 50 decimális számjegy befogadóképességgel. Adatbevitelre a J. M. Jacquard által feltalált és szövőgépek vezérlésére használt lyukkártyákat alkalmazta volna. A terv élete végéig foglalkoztatta; de a tervezést befejezni nem tudta, tervének gyakorlati kivitelezés is korlátokba ütközött, de szellemileg egy századdal megelőzte korát.

A modern számítógépig vezető úton haladva elsősorban A. M. Turing matematikust említhetjük meg. Turing 1936-ban fejlesztette ki az automatikus tárolt programozású univerzális számítógép elméleti (matematikai) modelljét. Bebizonyította, hogy az olyan gép, amely el tud végezni néhány alapműveletet, elvileg bármilyen számítás végrehajtására alkalmas. Turing használta először a „*to compute*" (kiszámítani) igét. Innen származik a „*computer*" (számítógép) elnevezés is. Ezután, 1948-ban Norbert Wiener a „Cybernetics or Control and Communication in the Animal and the Machine" (A kibernetika, az élő és élettelen rendszerek folyamatainak irányítás-elméletében) című művében megírja, hogy még 1940-ben kidolgozott öt alapelvet, amelyek később a programvezérlésű elektronikus számítógépekben megvalósultak.

A legelső kizárólag elektronikus elemekkel működő számítógép az ***ENIAC*** (**E**lectronic **N**umerical **I**ntegrator and **C**alculator) volt. Ezt Neumann János (1903-1957) matematikus elgondolása alapján J. N. Brainerd, J. P. Eckert, J. W. Mauchly és H. M. Goldstine építették meg, és 1944 végén állították működésbe. Neumann 1946-ban tartott előadásaiban tette közzé a korszerű számítógép megépítésének alapelveit. Először is rámutatott arra, hogy az akkori gépek, amelyekben még lényeges szerepet játszottak a mechanikus alkatrészek, sem túlságos lassúságuk, sem megbízhatatlanságuk miatt nem felelhettek meg rendeltetésüknek. Ezért teljesen elektronikus gépet javasolt. Másodsorban, kifejtette, hogy míg a tízes számrendszer a mechanikus szerkezeteknél tökéletesen megfelelő volt, az elektronikus gépeknél a kettes számrendszert sokkal célszerűbb alkalmazni. Ez a számrendszer kétállapotú elektronikus eszközök segítségével, könnyen megvalósítható, és ezenkívül a száminformációk tárolása minimális alkatrészt igényel. A fenti elvek alkalmazásával a gép számítási gyorsasága olyan nagy lesz, hogy értelmetlenné válik az emberi beavatkozás a gép működésébe, ez ugyanis tetemesen több időt venne igénybe; mint a műveletek elvégzése: Logikusan következik a *harmadik javaslat*: a belső memória (tár) létrehozása. Ebben a számítási részeredmények tárolhatók, és így a gép több műveletsorozatot is automatikusan el tudna végezni. A legfontosabb javaslat a *negyedik* volt, amely a számítógépek logikai irányítására vonatkozott. Ha a gép belső memóriája nemcsak adatokat tárol, hanem műveleti utasításokat is, akkor képes lépésről lépésre önállóan haladni. Minden lépés után a gépet saját memóriája utasítja a további teendőkre anélkül, hogy emberi beavatkozásért. kellene időznie.

Az utasításoknak azt a sorozatát, amely meghatározott feladatot old meg, ***programnak*** hívják. A számítógép programozhatósága tekinthető az igazán lényeges áttörésnek, amely megkülönbözteti a modern számítógépet a régi számológépektől. A tárolt program koncepciója vezetett azután az önmagukat módosító számítógépek kifejlesztéséhez.

A Neumann felsorolt alapelvei szerint működő első gép az ***EDVAC*** (**E**lectronic **D**iscrete **V**ariable **A**utomatic **C**omputer) volt, amelyet 1949-ben állítottak működésbe. Az EDVAC volt az első elektronikus digitális komputer, amit belső programtárolási koncepciónak megfelelően építettek meg. Az EDVAC-ot a Moore School of Electrical Engineering munkatársai tervezték, hasonlóan mint az ENIAC-et, de az EDVAC igen jelentősen különbözött attól. A legfontosabb eltérés az volt, hogy ez a gép tárolt programozású volt. Az EDVAC-ba beépítettek 19 különböző típusú, 3563 vákuumcsövet, 8000 kristály diódát, 1325 mágneses elemet, közelítőleg 5500 kondenzátort, 12000 ellenállást és 320 neont.

Az EDVAC 50 kW-ot fogyasztott és közel 50 négyzetméter területet foglalt el. A berendezés közel 8 tonna súlyú volt.

A mikroprocesszorokkal megvalósított mikroszámítógépek az elvi működés szempontjából elég kevés eltérést mutatnak a Wiener, valamint Neumann javaslatai alapján megépült, tárolt programozású univerzális számítógépektől. Ellenben annál nagyobb a különbség a megvalósítási technológia terén. Ennek a jelentőségét nem hanyagolhatjuk el, ugyanis a technológia fontossága már a számítógépek fejlődésének történetében megmutatkozott. A XVII. században a finommechanika fejlődése tette lehetővé Pascalnak az első mechanikus számológép megalkotását. A mechanikus és a későbbi elektromechanikus számológépek korszakának a végét a következő felfedezések jelentették: az elektronemisszió (Edison, 1883-ban), a mágnesesvezérlésű elektroncső (R. von Lieben, 1906-ban) és az audiondetektor (L. de Forest, 1907-ben). Az első elektronikus számítógép, az *ENIAC*, 18000 elektroncsövet tartalmazott. A gép gyorsasága körülbelül ezerszeresen felülmúlta az elektromechanikus gépek gyorsaságát. Ennek ellenére megmutatkoztak az elektroncsövek hátrányai. Az *ENIAC* 18 000 elektroncsövének felszerelésére 500000 ponton kellett forrasztani. Csupán a forrasztási helyek előkészítése két évig tartott. A gép 100-150 kW elektromos energiát fogyasztott. Elhelyezésére 9 × 15 méter nagyságú teremre volt szükség, súlya pedig 30 tonna körül volt.

Talán semmihez nem kötődik úgy a mai számítástechnika, mint a félvezetők fizikájához, technológiájához. A mikroprocesszorok technológiai előzményei 1948-ig nyúlnak vissza, amikor a félvezetők több évtizedes kutatása eredményeként John Bardeen, Walter H. Brattain és William Shockley, a Bell Laboratórium munkatársa felfedezték a tranzisztort. Ez nagy jelentőségű fordulópont volt, ugyanis az elektroncső működése a vákuumban mozgó elektronok vezérlésén, míg a tranzisztoré – amely majd később teljesen kiszorította az elektroncsöveket – a félvezető kristályszerkezetében lejátszódó jelenségeken alapszik. Érdekes megemlíteni, hogy 1948-ban John W. Turkey először használja a számítástechnika egyik legfontosabb és leggyakrabban használt egységét, a ***bitet***, azaz a „**bi**nary digi**t**"-ből rövidített fogalmat. A tranzisztor és a félvezetőipar fejlődésében igen jelentős eseménynek tekinthető, hogy 1952-ben az amerikai ipar nagyértékű megbízást kap a hadseregtől mind a gyártásra mind a fejlesztésre. Ez már szükségessé tette részben a gyártás automatizálását, részben a szabványosítást. A kutatás, a technológiafejlesztés, a gyártásautomatizálás és a tömegtermelés itt kezd szoros szimbiózisban élni, hatnak egymásra, táplálják egymást. A félvezetőkből készült elektronikus eszközök nemcsak képesek mindarra, amire az elektroncsövek, hanem azokhoz képest több döntő előnyük is van:

- térfogatuk az elektroncső térfogatának törtrésze;
- sokkal nagyobb a tartósságuk és a megbízhatóságuk;
- messzemenően kisebb energiát igényelnek működésükhöz;
- nagyobb sebességgel tudnak dolgozni.

Félvezetők nélkül a mai napig sem lett volna sima leszállás a Vénuszra és a Holdra. A valódi miniatürizáláshoz főképpen a rakéta és ürrakétaprogramok adtak ösztönzést. Érthető okokból a szakemberek rá voltak kényszerülve, hogy kicsi és könnyű elektronikus berendezéseket építsenek. A növekvő követelményeket a nyomtatott áramkörök sem elégítették ki. Újabb nehézségeket is ki kellett küszöbölni: a nagyfokú zavarérzékenységet és a csökkenő megbízhatóságot, amelyeket elsősorban a nagyszámú forrasztási pont okozott.

A miniatürizálás és a performanciák, valamint a megbízhatóság terén további nagy ugrást jelentett az *integrált áramkör*. Az integrált áramkörben a különböző rendeltetésű aktív és passzív áramköri építőelemeket, valamint a hozzájuk tartozó összekötéseket egyetlen gyártási folyamatban közös félvezető alapon állítják elő. Az integrált áramkört 1960-ban fejlesztették ki a Fairchild és a Texas Instruments cégek szakemberei: R. Noyce, valamint J. St. Clair Kilby. G. Moore 1965-ben megállapítja, hogy évenként megduplázódik az egy félvezető (szilícium) lapkán levő integrált tranzisztorok száma. Az ekkor megjósolt tendencia napjainban sem veszített érvényességéből.

Az integrált áramkörök technológiájának kimagasló eredménye a mikroprocesszor megvalósítása. Az első mikroprocesszort az Intel cég szakemberei fejlesztették ki 1971-ben. ***Mikroprocesszoron*** *egy (vagy legfeljebb néhány) olyan integrált áramkörből álló rendszert értünk, amely elvégzi a programozható számítógép központi egységének a feladatkörét. A mikroprocesszoron alapuló* ***mikroszámítógép*** *pedig az a rendszer, amely a mikroprocesszoron kívül rendelkezik a szükséges memóriával, valamint ki-, illetve bemeneti információcserére alkalmas egységgel.*

A számítógépekkel kapcsolatosan két szorosan összefüggő, de mégis jól elhatárolható részt különböztetnek meg: a ***hardvert*** és a ***szoftvert*** (az angol „hardware" és „software" kifejezések kiejtésének megfelelő írásmódja). A hardver a számítógép összes műszaki (fizikai) míg a szoftver a számítógép működését és a feladatok megoldását biztosító programokat foglalja magába.

A számítási sebesség növelése céljából a legújabb típusú számítógépek felépítése nem követi Wiener és Neumann javaslatait. 1983-ban a NEC Corporation japán cég kifejlesztette a világ első nem Neumann típusú, ultrasebességű számítógépét. E számítógép, amely nem volt nagyobb, mint egy klasszikus Neumann-típusú mikroszámítógép, másodpercenként 53 millió számítási műveletet végzett, vagyis 50-100 -szor olyan gyorsan számolt, mint az előbbi. A sebességét annak köszönhette, hogy míg a Neumann-típusú számítógépeknél minden számítási műveletnél az adatokat a memóriából el kell juttatni az aritmetikai egységbe és onnan vissza, addig az új típusú számítógépnél nem kell minden egyes számítási műveletnél adattovábbítást végezni.

A nyolcvanas évek elejétől a számítástechnika és az informatika minden képzeletet felülmúló fejlődésének vagyunk szemtanúi. Egyre gyorsabb, nagyobb teljesítményű és kisebb méretű digitális számítógépek és mikroprocesszoros rendszerek jelennek meg. Ezeknek a rendszereknek az ára – a nagy sorozatban való gyártásnak, az automatizálásnak és a magas fokú integráltságnak köszönhetően – fokozatosan csökken. A különböző mikroprocesszoros rendszerek ezáltal a mindennapi élet nélkülözhetetlen részeivé válnak.

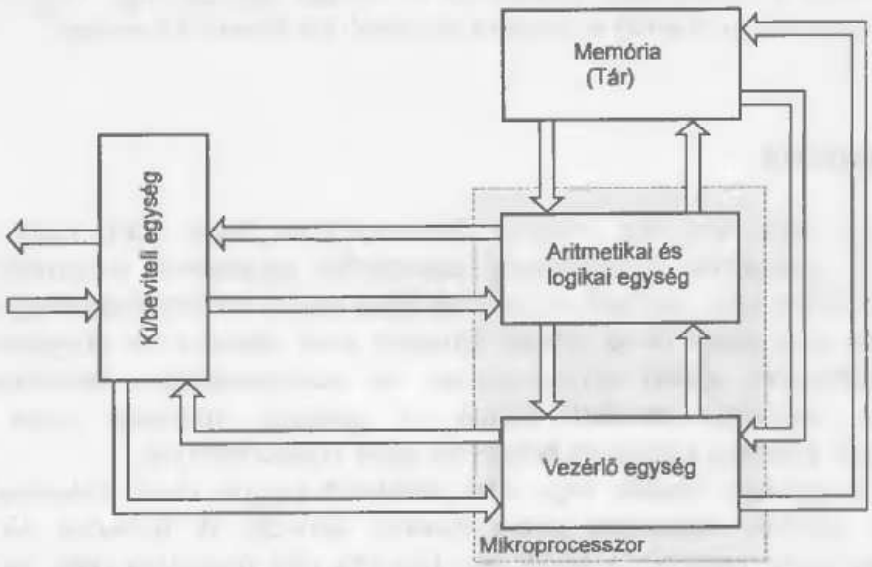
8.2. A mikroszámítógépek felépítése és működése

A mikroprocesszoros mikroszámítógépek működésének alapelveit a klasszikus értelemben vett (Neumann-féle) univerzális számítógép felépítésének és működésének tanulmányozásával érthetjük meg.

Az egymásba láncolt feladatmegoldási lépéssorozatok elemzésével a számítógép működése könnyen érthetővé válik. Adott feladat megoldása a megfelelő matematikai modell felállításával kezdődik. Ezután meg kell szerkeszteni a modellen alapuló megoldási lépések szekvenciáját. Ezt az úgynevezett programot a számítógépnek tudnia kell tárolni. Tárolásra szorulnak a feldolgozás alatt, valamint az utána keletkező rész-, illetve végeredmények. Mielőtt a gép hozzálátna a feladat megoldásához, biztosítani kell a megfelelő adatok betáplálását és tárolását. A feladat megoldása után a számítógépnek az eredményt könnyen érthető és kezelhető formában kell közölnie a külvilággal.

A felsorolt feladatok elvégzését biztosítja a számítógép négy alapegysége (*8.1. ábra*):

- az aritmetikai és logikai egység;
- a memória- vagy táregység;
- a vezérlőegység;
- a ki/bevitelí egység.



8.1. ábra: A klasszikus felépítésű univerzális számítógép rendszer-tömbvázlata

Az alábbi részletes elemzés megvilágítja az egységek működését, valamint a számítógépen belüli egymáshoz való funkcionális viszonyulását.

8.2.1. Az aritmetikai és logikai egység

Az aritmetikai és logikai egység (*ALU* – Arithmetic and Logic Unit), amint az elnevezése is mutatja, azon aritmetikai és a logikai műveletek végrehajtását teszi lehetővé, amelyekkel a program által meghatározott számolási és logikai műveletek sorozata végezhető el.

Minél többféle művelet elvégzésére képes egy számítógép aritmetikai és logikai egysége, annál könnyebben tudja megoldani a bonyolultabb feladatokat. Általában az aritmetikai és logikai egység a következő alapvetőnek számító bináris műveleteket végezheti el: két szám összeadása, valamint kivonása, egy szám jobbra vagy balra léptetése (helyérték eltolás), két számmal végzett logikai *ÉS*, *VAGY*, valamint *KIZÁRÓ-VAGY* művelet, egy szám komplemensének képzése és két szám összehasonlítása. E műveletek segítségével más komplexebb műveletek is elvégezhetők. Például a szorzást ismételt összeadás és helyérték eltolás segítségével, az osztást pedig ismételt kivonás és helyértékeltolás segítségével lehet elvégezni.

A számok összehasonlítása, valamint az aritmetikai és logikai műveletek együttese a legbonyolultabb döntést (feltételes ugrást) meghatározó műveletek elvégzését teszi lehetővé. A döntési műveletek a számítógép „intelligens" működésének alapját képezik.

Az aritmetikai és logikai egység központi regisztere az úgynevezett ***akkumulátor*** (angolul: *accumulator*). Ebben a művelet végrehajtása előtt az egyik operandus, a művelet végrehajtása után pedig az eredmény található. Az akkumulátoron kívül az aritmetikai és logikai egységben még egy ***regisztertömb*** (angolul: *scratch pad, register array*) is van. Ennek a regiszterei bizonyos utasítások végrehajtásában vesznek részt. Egy-egy operandust vagy eredményt tárolnak. Az akkumulátor és a regisztertömb a memóriából és a ki/beviteli egységen keresztül kapja az adatokat. Feladatuk elvégzése után tartalmukat a memóriában tárolják, vagy a ki/beviteli egységen keresztül szolgáltatják ki.

Egy műveletet a számítógép aritmetikai és logikai egysége egy vagy több utasítás segítségével végez el. Az utasítás a program legkisebb funkcionális egysége.

8.2.2. A memória

A memória vagy táregység (angolul: Memory Unit, Store Unit) nagy funkcionális fontosságú, és rendszerint a számítógép legnagyobb egységének tekinthető. Memórián azokat az eszközöket értik, amelyek az információkat tetszés szerinti ideig megőrzik.

A memória a program és az adatok tárolását teszi lehetővé. A programmemóriában található a programot alkotó utasítássorozat. Az adatmemóriában találhatók a feladat megoldásához szükséges kezdeti adatok, a program futtatása során keletkezett részeredmények, valamint a program befejezése utáni végeredmények.

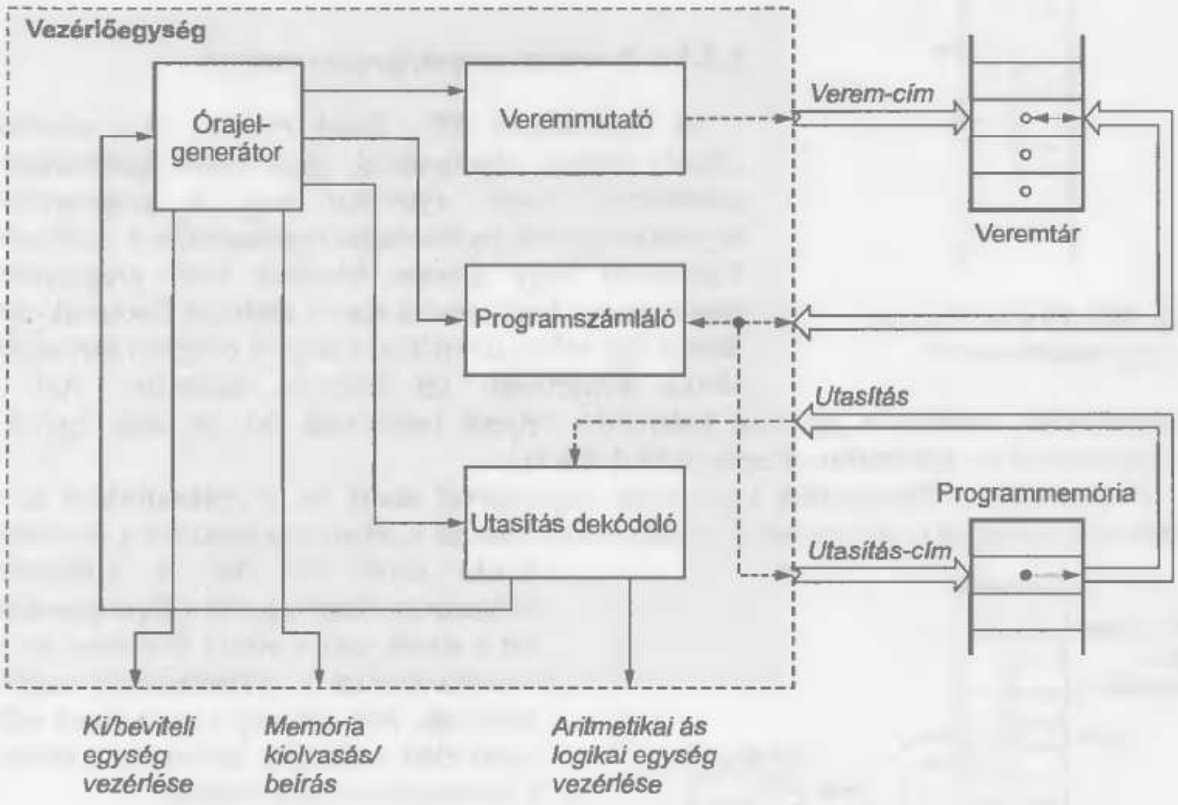
A memória egy-egy utasítás vagy adat tárolására kijelölt részét ***rekesznek***, a rekesz azonosítására szolgáló sorszámot pedig ***címnek*** nevezik. A memória által tárolható információmennyiség mutatója a ***kapacitás***. Egy-egy cím megadása után, amíg a rekesz tartalmát ki lehet olvasni vagy adatot lehet beírni, bizonyos időnek kell eltelnie. Ez az úgynevezett *hozzáférési* vagy *elérési idő*. A kapacitás és a hozzáférési idő a memóriák lényeges jellemzői.

A *8.1. ábrán* bemutatott tömbvázlatban levő memória a számítógép úgynevezett ***operatív memóriája*** (angolul: *main memory*). Használatosak a központi, belső vagy elsődleges memória elnevezések is. A számítógép teljesítményének növelése céljából az operatív memóriát kiegészítik úgynevezett ***külső*** vagy ***másodlagos memóriával*** is (secondary memory). Ennek a memóriának az elérése a ki/beviteli egységen keresztül valósul meg. Szükség esetén a külső memóriában tárolt információt át lehet vinni az operatív memóriába, vagy fordítva. a külső memória kiegészíthető a nagy adattömeget tároló ***háttér memóriával*** (angolul: *mass storage*). Ennek kapacitása nagyobb, de az előbbiekhez képest a hozzáférési ideje is jóval nagyobb.

8.2.3. A vezérlőegység

A vezérlőegység (angolul: *Control Unit*) szerepe a számítógép működésének, tehát a műveletek program szerinti végrehajtásának az irányítása is. Ez az egység rendszerint a következő négy részegységből épül fel (*8.2. ábra*):

- a programszámláló;
- a veremtár- (memória-cím) mutató;
- az utasításdekódoló;
- az órajelgenerátor.

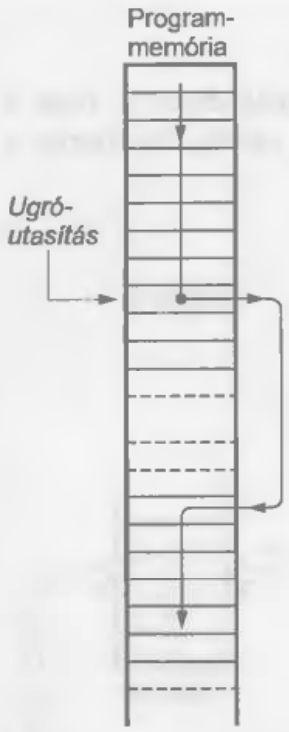


8.2. ábra: A vezérlőegység felépítése és funkcionális összeköttetései

Az egység működését a felsorolt részegységek funkcionális elemzése alapján érthetjük meg.

8.2.3.1. A programszámláló

A programszámláló (*PC* – Program Counter) a soron következő utasítás címét jelöli ki. Minden egyes utasítás kiolvasása után az órajelgenerátor a programszámláló tartalmát eggyel megnöveli. A programszámláló így biztosítja az utasítások lépésről lépésre való elérését. Abban az esetben, ha a gép egy ún. ***elágaztató*** vagy ***ugróutasításhoz*** ér, amelynek az a szerepe, hogy megszakítsa az utasítások növekvő sorrendű elérését, akkor a programszámláló tartalma egy adott címre cserélődik.



8.3. ábra. Programelágazás egy ugróutasítás esetén

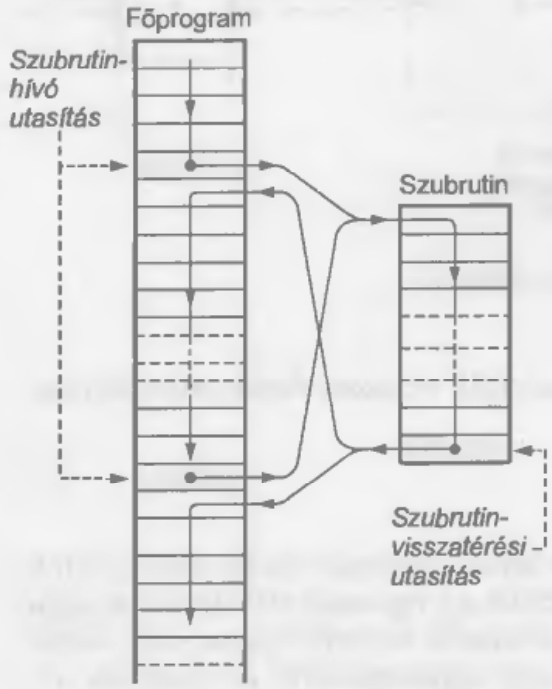
Ez után az ugrás után a programszámláló az új címtől kezdve lépésről lépésre tovább növeli tartalmát (*8.3. ábra*).

Az ugróutasítások lehetnek feltételhez kötöttek vagy feltétel nélküliek. A ***feltételes ugróutasítást*** a számítógép egy döntési művelet után hajtja végre. A döntés eredményétől függően, amely – mint láttuk – egyszerűbb vagy bonyolultabb összehasonlítási műveletekből származik, programelágazás jöhet létre. A ***feltétlen ugróutasítás*** esetén a gép program végrehajtásában feltétel nélküli ugrás következik.

8.2.3.2. A verem (memória-cím) mutató

A veremmutató (***SP*** – **S**tack **P**ointer) és a veremtár (*Stack*) szerepe ***alprogramok*** (más néven ***szubrutinok***) alkalmazása esetén nyilvánul meg. A programozás egyszerűsítésének legfontosabb segédeszköze a *szubrutin*. Különböző vagy azonos feladatot leíró programban megtörténhet, hogy azonos részek többször fordulnak elő. Ezeket úgy célszerű felállítani, hogy a program bármelyik részén közvetlenül fel lehessen használni. Azt a programrészt, amelyet a program különböző helyein használnak fel, de csak egyszer programoznak be, ***szubrutinnak*** nevezik (*8.4. ábra*).

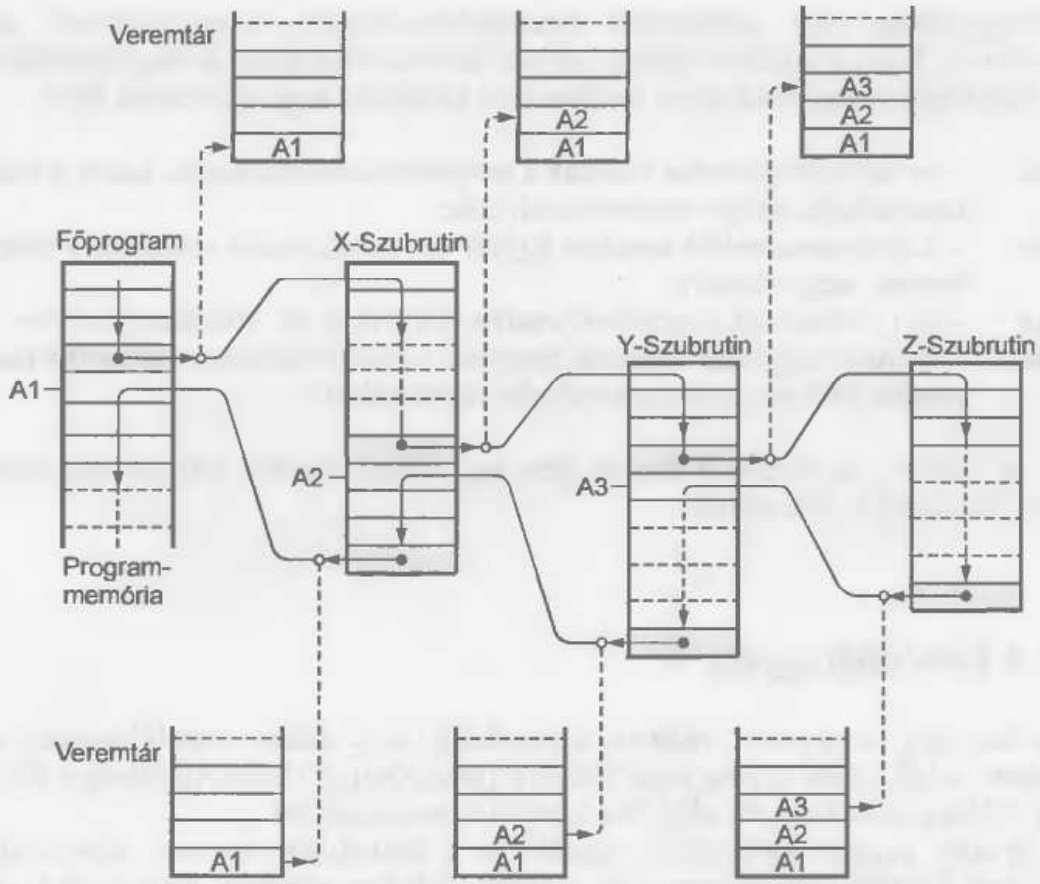
A szubrutint a főprogramba két utasítás segítségével ékelik be: a ***szubrutinhívó*** és a ***szubrutin-visszatérési*** utasítással. A szubrutinhívó utasítás a programszámlálóba a szubrutin kezdő címét írja be, a szubrutin-visszatérési utasítás pedig a főprogramnak azt a címét, ahová vissza kell térni. Ez a cím rendszerint a szubrutinhívó utasítás utáni cím, amelyet még a szubrutinra való ugrás előtt szükséges tárolni. Ez a tárolás a veremmemóriában történik.



8.4. ábra. Egy szubrutin felhasználása a főprogram különböző helyein

A veremtár beírási sorrendben tárolja a címeket, kiolvasáskor pedig fordított sorrendben adja vissza őket. Ezért a veremmemória egy ún. LIFO (**L**ast – **I**n, **F**irst – **O**ut: az utolsó – be, az első – ki) típusú memória. A legutolsó beírt adat címét, tehát a legelsőnek kiolvasandó adatcímet a veremmutatóból lehet kiolvasni (*8.2. ábra*).

A veremmemória és a veremmutató hasznossága főleg a több szintű egymásba skatulyázott szubrutinok esetén nyilvánul meg (*8.5. ábra*).



8.5. ábra. A veremtár szerepe a többszintű, egymásba skatulyázott szubrutinok használatával

Ebben az esetben a szubrutinhívó utasítások hatására a veremtár a visszatérési címeket a szubrutin hívások sorrendjében tárolja, visszatéréskor pedig fordított sorrendben adja vissza őket. Ily módon a visszatérés mindig az utoljára megszakított programra (szubrutinra vagy főprogramra) történik. A veremmemóriában nemcsak címeket lehet tárolni, hanem adatokat is, mint például az akkumulátor és a regiszterek tartalmát.

8.2.3.3. Az utasításdekódoló

Az utasításdekódoló (angolul: Instruction Decoder) a vezérlőegység legfontosabb része. Szerepe abban nyilvánul meg, hogy az utasításokat ábrázoló kódszámokat megfelelő vezérlőjelekké alakítja át. A vezérlőjelek egy része közli az aritmetikai és logikai egységgel a végrehajtandó művelet típusát. A vezérlőjelek másik része a számítógépen belüli információáramlást ellenőrzi és szabályozza.

8.2.3.4. Az órajelgenerátor

Az órajelgenerátor (Clock Generator, Timing Unit) állítja elő a gép időbeni működéséhez szükséges vezérlőjeleket. Ezek rendeltetései a következők: az aritmetikai és logikai egység vezérlése (az utasításdekódoló által jelzett művelet elvégzése), az információk kiolvasása és beírása a memóriába, a ki/beviteli egység működésének vezérlése és a vezérlőegység időbeni működésének irányítása.

Az órajelgenerátor által szolgáltatott vezérlőjeleket részben a végrehajtandó utasítás határozza meg. Ezért a dekódoló egység irányító információkat közöl az órajelgenerátorral.

A számítógép időbeni működését rendszerint a következő négy gépi ciklus írja le:

1. ciklus – az ún. címregiszterbe beíródik a programszámláló tartalma, amely a végrehajtandó utasítás programmemóriabeli címe;
2. ciklus – a programszámláló tartalma eggyel növekszik; ezzel előkészül a következő utasítás megcímzésére;
3. ciklus – az 1. ciklus alatt megcímzett utasítás beíródik az ún. utasításregiszterbe;
4. ciklus – az utasításregiszter tartalmát értelmezi az utasításdekódoló, amely biztosítja az utasítás által meghatározott művelet végrehajtását.

Abban az esetben, ha az előbbi utasítás nem egy leállító utasítás volt, az órajelgenerátor indítja a következő 1. gépi ciklust.

8.2.4. A ki/beviteli egység

A számítógép és az ember, valamint a számítógép és az általa vezérelt berendezés közti kapcsolatot a ki/beviteli egység teszi lehetővé (Input/Output Unit). Általában a ki/beviteli egység felelős a számítógép és a külvilág közti információcseréért.

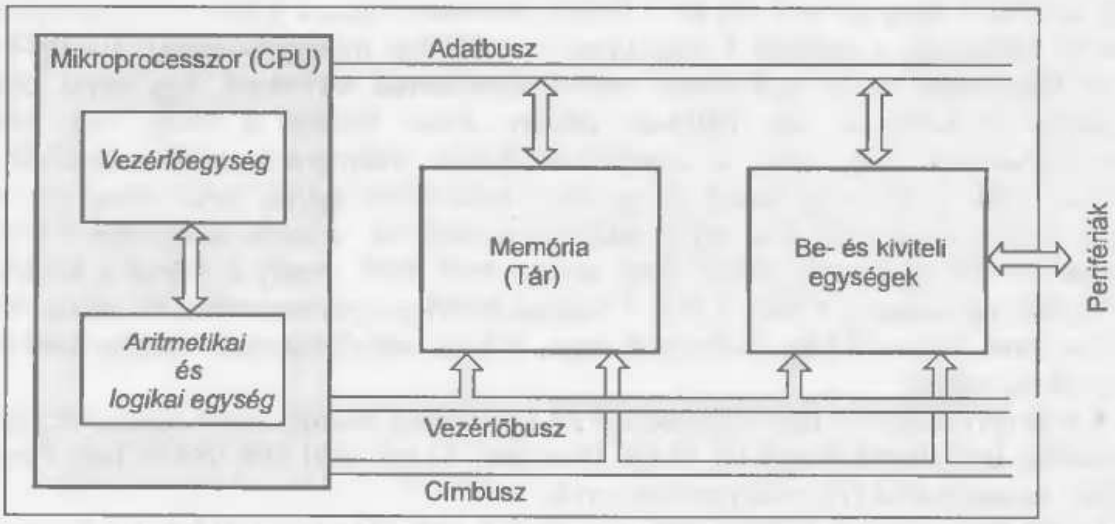
A beviteli egység segítségével visszük be a számítógép operatív memóriájába a megoldandó feladatot leíró programot és az ehhez szükséges adatokat. A bevitelre kerülendő információ előzőleg lyukszalagon, lyukkártyán, mágneslemezen, mágneskártyán vagy mágnesszalagon lehet tárolva. Innen ezt az információt a sajátos leolvasómű a gép számára kompatibilis elektromos jellé alakítja át. Ezen kívül a számítógép alkalmas az információ kézi (manuális) betáplálására is. Erre szolgál a billentyűzet vagy más nevén a klaviatúra. Abban az esetben, ha a számítógép ipari folyamatot szabályoz, a beviteli egységnek a folyamat paramétereit figyelő érzékelők és átalakítók által szolgáltatott elektromos jeleket is kell; tudnia venni. Legtöbbször ez a jel a megfigyelés alatt álló paraméterrel arányos analóg jel. Ilyenkor analóg-digitális konverter (átalakító) alkalmazása szükséges.

A kiviteli egységnek a számítási folyamat eredményeit további felhasználásra is alkalmas formában kell megadnia. Ezt elektromos írógép, nyomtató, rajzolókészülék, lyukszalag-, vagy lyukkártya-lyukasztó segítségével lehet megvalósítani. A számítógépeknél népszerű a katódsugárcsöves megjelenítő használata is. Ennek segítségével nemcsak írott szövegként (számok és betűk) kaphatjuk kézhez az eredményt, hanem rajz formájában is. Ipari folyamatok szabályozása esetén a kiviteli egység olyan elektromos jeleket is kell, hogy szolgáltasson, amely motorokat, elektromechanikus csapokat és más vezérlőműveket is képes működtetni.

A számítógép ki/beviteli egységéhez kapcsolt ki-, illetve beviteli készülékeket (például: lyukszalagolvasó, billentyűzet, egér, nyomtató, katódsugárcsöves megjelenítő) perifériáknak nevezik. A perifériák egy sajátos illesztőegységen, ún. interfészen keresztül csatolódnak a ki/beviteli egységhez.

8.3. Mikroprocesszorok

A mikroprocesszor tulajdonképpen a számítógép „agya". A mikroprocesszor szó a számítógépes terminológiában már korábban is ismert volt. A mikroprogramozható számítógépek mikroprogramját feldolgozó központi egységének a processzorát értették mikroprocesszoron. Jelenleg ezt a kifejezést tágabb értelemben használják. Tekintsünk vissza az univerzális számítógép elvi felépítéséhez.



8.6. ábra. A mikroszámítógépek egyszerűsített tömbvázlata

A *8.6. ábrán* bemutatott mikroszámítógép tömbvázlata a *8.1. ábrán* látható klasszikus architektúrájú univerzális számítógép tömbvázlatából megfelelő átcsoportosítással alakítható ki. A mikroszámítógép három alapvető egységét lehet megkülönböztetni:

- a memória (Memory),
- a központi feldolgozó egység (CPU – Central Processing Unit) és
- a ki-, illetve beviteli egység (Input/Output Unit).

A *mikroprocesszor* elnevezés a központi feldolgozó egységre vonatkozik. Ez egy olyan LSI áramkör, amely egy vagy néhány tokba van beépítve. Felépítésében két alapvető egység különböztethető meg: az *aritmetikai és logikai egység*, valamint a *vezérlőegység*.

A mikroszámítógép felépítéséhez a mikroprocesszort elsősorban memória- és ki-, illetve beviteli áramkörökkel, valamint többé kevésbé a vezérlést kisegítő áramkörökkel (mint például órajelgenerátorral, állapotdekódolóval, meghajtókkal, stb.) kell kiegészíteni.

A mikroszámítógép építőelemeit párhuzamos vezetékcsoport-együttesből álló, ún. ***busz-*** vagy ***sínrendszer*** köti össze. Az adatáramlás az adatbuszon bonyolódik le; a címeket a mikroprocesszor a címbuszon küldi ki; és végül a vezérlőjeleket a vezérlőbusz juttatja el a mikroszámítógép összes egységéhez. A *8.6. ábra* egy egy adatbusszal rendelkező, hagyományos Neumann-féle, ún. ***SISD*** (Single Instruction – Single Data Stream = egy utasítás – egy adatáramlás) mikroszámítógépet szemléltet. Gyorsabb adatfeldolgozás és adattárolás céljából bonyolultabb szervezésű számítógépeket fejlesztettek ki, mint például az

SIMD (Single Instruction – Multiple Data Stream = egy utasítás – sokszoros adatáramlás) vagy az ***MIMD*** (Multiple Instruction – Multiple Data Stream = többszörös utasítás – többszörös adatáramlás).

Az első mikroprocesszort 1971-ben készítették el az Intel cég szakemberei. Ez az Intel 4004 mikroprocesszor, 4-bites adatok feldolgozására volt képes, és egy japán asztaliszámológép-gyártó cég rendelése nyomán valósították meg. Számítási teljesítménye közepesen aluli volt, inkább egy számológépre emlékeztetett. A következő az Intel cég 8008-as mikroprocesszora volt. Ezt 1972-ben, egy katódsugárcsöves megjelenítő részére dolgozták ki. Közben a megrendelő bipoláris áramkörökkel oldotta meg a feladatát, és lemondta a megrendelést. Utólag az Intel cég ezt a 8-bites processzort piacra dobta, és a nagy igénylés nyomán kidolgozta a második 8-bites típust – a 8080-as mikroprocesszort. Ez 1974-ben került forgalomba, és az első sikeres mikroprocesszornak tekinthető. Egy évvel később megjelent a Motorola cég 6800-as, néhány évvel később a Zilog cég Z80-as mikroprocesszora, hogy csak az elterjedtebb 8-bites mikroprocesszorokat említsük. A hetvenes évek végén megjelentek az egytokos mikroszámítógépek (Microcomputer on a chip). A processzoron kívül a chipen található a memória, valamint a ki- illetve beviteli áramkör is. Az egyik legbeváltabb típus volt az Intel 8048, amely a chipen a következő egységeket tartalmazza: 8-bites CPU, 1 kbájtos ROM programmemória, 64 bájtos RAM adatmemória, három 8-bites ki/beviteli kapu, 8-bites időjelgenerátor, órajelgenerátor és megszakításvezérlő.

A mikroprocesszorok teljesítőképessége az egyidejűleg feldolgozandó adatszó hosszával növekedik. Így jelentek meg a 12, 16 (pl. Intel 286), 32 (pl. Intel 386, 486 és Intel Pentium család), valamint a 64 bites mikroprocesszorok.

A mikroprocesszorok teljesítménye növelhető a ***segédproceszszorokkal*** (Co-processor). Példaként megemlítjük az Advanced Micro Devices cég Am 9511A és Am 9512A valamint az Intel cég 8087, 287, 387 jelű aritmetikai műveletekre specializált segédprocesszorait. Segítségükkel 16, 32, illetve 64 bites fixvesszős, valamint 32, illetve 64 bites formátumú lebegővesszős számokkal a következő műveletek végezhetők el: összeadás, kivonás, szorzás, osztás, négyzetgyökvonás, hatványra emelés, logaritmus és trigonometriai függvények számítása, valamint átalakítások: a fixvesszős formátumból a lebegővesszősbe és fordítva. Ezek a műveletek a segédprocesszorral sokkal gyorsabban végezhetők el, mint nélküle. Ezen kívül számottevően csökkenthető a program memória kapacitása azáltal, hogy nincs szükség a műveleteket elvégző szubrutinokra.

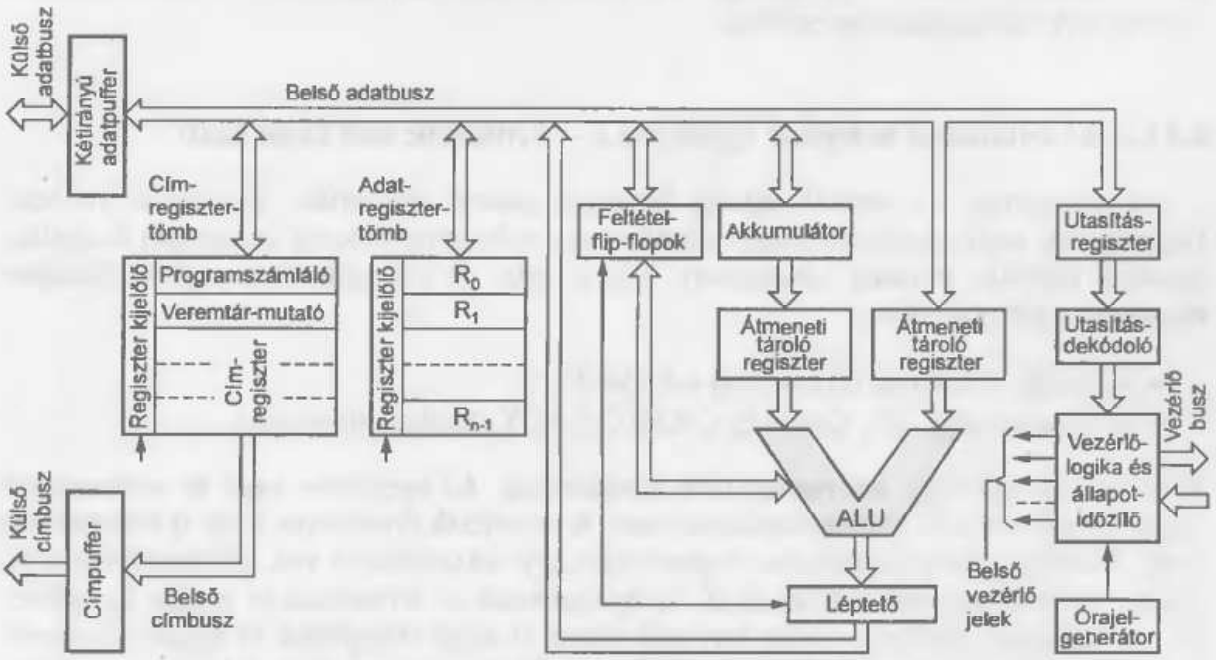
Különböző adatfeldolgozási és vezérlési műveletekre specializált processzorokat is kifejlesztettek. Például az Intel 2920 jelprocesszorát, amely az analóg jelek valós idejű, digitális feldolgozására alkalmas. Az analóg bemeneti jel mintavételezését és digitálissá való átalakítását a chipen levő áramkörök végzik. A digitális jelet a processzor, az ugyancsak a chipen levő 192 × 24 bites EPROM programmemóriába betáplálható algoritmus alapján dolgozza fel. A kimenet előtt levő digitális/analóg átalakító az így kapott digitális jelet analóggá alakítja át. A TRW cég TDC1008/1009/1010 8-, 12, illetve 16 bites gyors jelprocesszorai segédprocesszorként is használhatók. Sebességüket a gyors összeadást, kivonást, valamint szorzást végző aritmetikai egységük biztosítja. Segítségükkel valós idejű gyors Fourier-transzformálás, digitális szűrés és vektorszorzás valósítható meg.

Végül a ***bitszelet mikroprocesszorokat*** (angol elnevezése: *bit-slice microprocessor*) említjük meg, amelyek a mikroprogramozás eszközei. A mikroprocesszor tulajdonképpen úgy tekinthető, mint egy sorrendi logikai hálózat. Bonyolultabb sorrendi hálózatok tervezése legtöbbször komoly nehézségekbe ütközik. Ezenkívül a rugalmasságuk meglehetősen alacsony, és a továbbfejlesztési lehetőségeik korlátozottak. Ezek a hátrányok egy

mikrovezérelt rendszerrel küszöbölhetők ki. A vezérlőinformációt úgynevezett vezérlőmemória tárolja. A pillanatnyi vezérlőmemória-cím tartalma adja az aktuális vezérlési állapotot. A következőt pedig az előző állapot, a vezérlőmemória tartalma és a külső feltételek szabják meg. A mikrovezérelt rendszer vezérlőmemória tartalmának az adott feladat függvényében való előállítását mikroprogramozásnak nevezik. A legbeváltabb az Advanced Micro Devices Am 2900 4-bites bitszelet mikroprocesszor családja. A kívánt szó hosszúságnak megfelelően több bitszelet processzor kapcsolható párhuzamosan.

8.3.1. A mikroprocesszorok felépítése és működése

A *8.7. ábra* a mikroprocesszorok közös funkcionális tulajdonságait veszi figyelembe. A továbbiakban ismertetésre kerülő elemzés alapján a különböző típusok felépítése és működése könnyen érthetővé válik.



8.7. ábra. Tipikus mikroprocesszor-tömbvázlat

8.3.1.1. A buszrendszer (Bus System)

Megkülönböztethető a ***külső*** és a ***belső buszrendszer***. A külső buszrendszer, a mikroszámítógép mikroprocesszorán kívül levő építőelemeit köti össze. A belső buszrendszer a mikroprocesszor belső egységeit köti össze és a chipen van kialakítva.

A buszrendszer három részre tagolódik.

1. Az adatbusz (*Data bus*) a különböző egységek közötti kétirányú adatátvitelt bonyolítja le. A belső adatbusz a kétirányú adatátvitelre alkalmas adatpufferen keresztül csatlakozik a külső adatbuszhoz. Ha a mikroprocesszor adatot kap, akkor az adatpuffer bemenő állapotban van. Adatkiküldés esetén az adatpuffer kimenő állapotba kerül. A külső adatbusz-meghajtók az ún. ***közvetlen memóriahozzáférés*** (DMA – Direct Memory Acces) alatt kerülnek a

harmadik, nagy impedanciájú állapotba. Ekkor a közvetlen memóriahozzáférést lebonyolító egység kerül összeköttetésbe a memóriával.

2. A címbusz (*Address bus*). A címpuffer csatlakoztatja a belső címbuszt a külsőhöz. A címpuffer csak egyirányú jelátvitelre alkalmas. A mikroprocesszor az általa előállított cím segítségével jelöli ki azt az egységet, amellyel éppen érintkezésbe fog lépni. A közvetlen memóriahozzáférés alatt a mikroprocesszor felszabadítja vezérlése alól a címbuszt. A buszmeghajtók a harmadik, nagy impedanciájú állapotba kerülnek. Ezáltal a. memóriát a közvetlen hozzáférést lebonyolító egység címezheti meg.

3. A vezérlőbusz (*Control bus*) szerepe a különböző vezérlő és szinkronizáló jelek továbbítása. A mikroszámítógép vezérlését a mikroprocesszor által előállított jelek biztosítják. Válaszként a számítógép áramkörei állapotelismerő jeleket küldenek vissza. A mikroprocesszor műveletvégrehajtást vezérlő vagy állapotkérő jeleket is kaphat a perifériás áramköröktől. A kérő jel (*request signal*) átvételét elismerő jellel (acknowledge signal) nyugtázza.

A mikroprocesszorok buszmeghajtóit általában egy TTL vagy 5-10 MOS áramkörrel lehet leterhelni. Ezért – az egészen kis számítógépek kivételével – a buszrendszerre teljesítmény-meghajtókat kapcsolnak.

8.3.1.2. Az aritmetikai és logikai egység (ALU – Arithmetic and Logic Unit)

Az aritmetikai és logikai egység bizonyos számú aritmetikai és logikai művelet végrehajtását teszi lehetővé. A nagy teljesítményű mikroprocesszorok aritmetikai és logikai egysége többféle művelet elvégzésére képes, mint a kis teljesítményűeké, általában mindkettő a következőkre:

- összeadás és kivonás (aritmetikai műveletek),
- komplementálás, ÉS, VAGY és KIZÁRÓ-VAGY (logikai műveletek).

A műveletek egy vagy két operandusra vonatkoznak. Az operandus vagy az operandusok egyike az *akkumulátorban* (accumulator) van. A műveletek eredménye is az akkumulátorba kerül. Minden mikroprocesszornak legkevesebb egy akkumulátora van, de létezik olyan is, amelyet több akkumulátorral látnak el. Az operandusok az aritmetikai és logikai egységhez egy-egy átmeneti tárolóregiszteren keresztül jutnak el. Ezek elszigetelik az egység kimenetét a bemeneteitől. Három belső adatbusszal rendelkező mikroprocesszoroknál az átmeneti tárolóregiszterek fölöslegessé válnak. Az operandusok két különálló buszon jutnak az aritmetikai és logikai egységhez. Az eredmény a harmadik buszra kerül.

Az akkumulátor mellett levő ún. feltétel- vagy státuszregiszter (angol elnevezése: Flag, Status-register) flip-flopjainak állapota az éppen végrehajtott művelet eredményétől függően alakul. Ezért jelző biteknek is szokták nevezni. Feltételes ugróutasítások előtt a mikroprocesszor egy adott feltételbit értékét hasonlítja össze a logikai változó **0** vagy **1** értékével. Az összehasonlítás eredményének függvényében alakul a program további végrehajtása.

A feltételregiszter általában a következő bitekből áll:

1. **CY** (Carry) – átvitelbit, amely úgy tekinthető mint az n-bites akkumulátort bővítő $(n+1)$ legnagyobb helyértékű bit. Ez a bit tartalmazza a legnagyobb helyértékű oszlop összeadása után keletkező átvitelt. Segítségével n-nél nagyobb szóhosszúságú adatok összeadása is lehetséges.

2. **V** (Overflow) – a túlcsordulásbit, amelyet **O**-val is szoktak jelölni, két előjeles szám összeadása után keletkező átvitelt mutatja. Ha értéke **1**, akkor a két n-bites 2-es komplemensű szám összege egy $(n+1)$-bites szám. A legnagyobb helyértékű oszlop, valamint a közvetlenül előtte levőnek az összeadása után keletkező átvitelekkel végzett KIZÁRÓ-VAGY művelet eredményezi a túlcsordulást.

3. **N** (Negative) vagy **S** (Sign) – az előjelbit. Ez a 2-es komplemensben ábrázolt számok előjelét mutatja. Ha értéke **1**, akkor az akkumulátor negatív számot tartalmaz. Ha **0**, akkor ellenkezőleg, az akkumulátor tartalma pozitív vagy nulla. A 2-es kiegészítős számábrázolás figyelembevételével az előjelbit egyenlő az akkumulátor legnagyobb helyértékű bitjével.

4. **H** (Half-carry) vagy **AC** (Auxiliary-carry) a BCD kódolású számokkal végzett műveleteknél játszik szerepet, és az első elnevezése a 8-bites mikroprocesszorokra vonatkozik. Két decimális számjegyű két BCD szám legkisebb helyértékű számjegyének az összeadásánál keletkező átvitelt jelöli. Tulajdonképpen a legkisebb helyértékű decimális számjegy legnagyobb helyértékű oszlopának (a 8-bit 4. bitje) az összeadásánál keletkezik.

5. **Z** (Zero) – a zerobit, amelynek értéke **1**, ha egy művelet eredményeként az akkumulátor tartalma nullává válik.

6. **P** (Parity) – a paritásbit. Ez a bit jelzi az akkumulátorban levő **1**-esek páros vagy páratlan számát. Páros számú egyesek esetében a paritásbit értéke **1**. A paritásbit fontossága az adatátvitelnél nyilvánul meg. A külső zavarok miatt hamisan érkezett információ detektálását teszi lehetővé. Ha a beérkezett adat paritásbitje azonos a küldött adatéval, akkor a helyességének valószínűsége nagy.

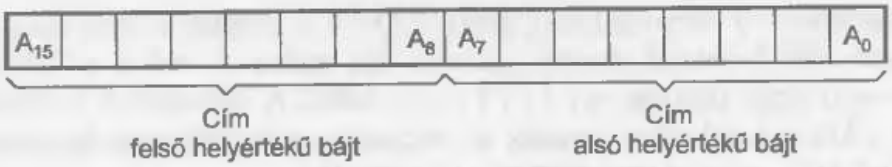
7. **I** (Interrupt) – megszakításbit, amelynek szerepe a program megszakítás-ellenőrzésénél nyilvánul meg.

Az aritmetikai és logikai egység léptetési műveleteket (angolul: shift, rotate) is végez. Ezeket a ***léptető*** (shifter) hajtja végre.

8.3.1.3. A mikroprocesszor belső regiszterei (Internal registers)

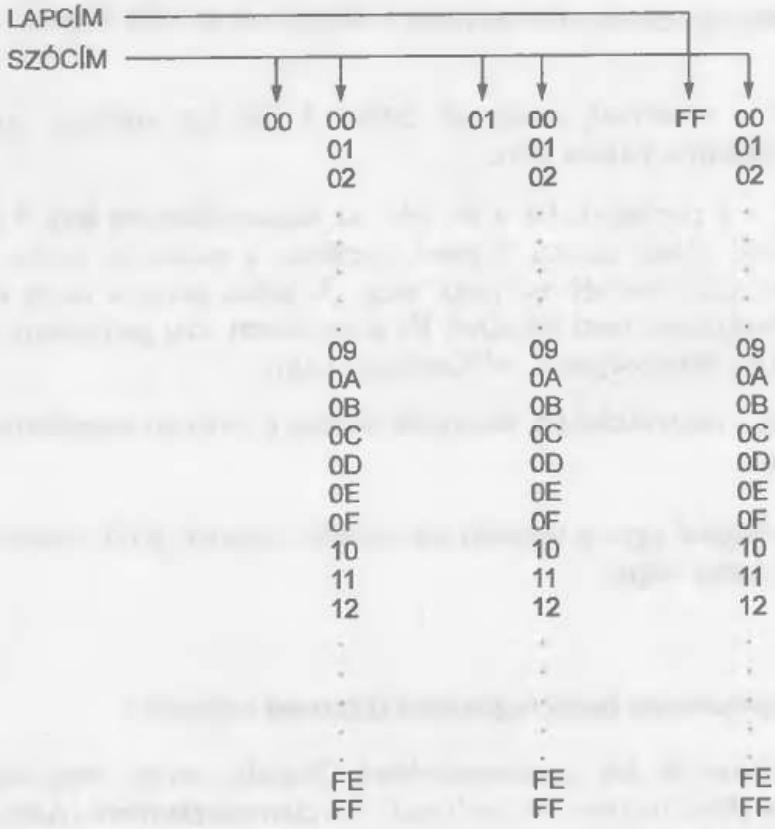
A mikroprocesszorok két regisztertömbbel (Register array) rendelkeznek, ezek: az ***adatregisztertömb*** (Data registers, Scratch pad) és a ***címregisztertömb*** (Address registers).
Az adatregisztertömb rendeltetése egy-egy operandus vagy eredmény átmeneti tárolása, és az akkumulátorral azonos hosszúságú regiszterekből tevődik össze. A regiszterkijelölő áramkör a végrehajtás alatt álló utasítás függvényében a regiszterek egyikét választja ki, és az adatbuszhoz kapcsolja. A regisztereket általában kettesével vagy négyesével is össze lehet kapcsolni. Ilyenkor a regiszterkijelölő egy kétszer vagy egy négyszer hosszabb regisztert választhat ki. Ezáltal könnyebb a nagyobb szóhosszúságú adatok feldolgozása.

A címregisztertömb legfontosabb regisztere a ***programszámláló*** (Program counter), és ez akármelyik mikroprocesszorban megtalálható. A programszámláló a soron következő utasítás címét tartalmazza, amelyet a címpuffer juttat a külső címbuszra. A 8-bites mikroprocesszorok esetén a címszó általában 16 bites. Ezzel összesen $2^{16} = 65\,536$ címhely különböztethető meg.



8.8. ábra. 8-bites mikroprocesszorok 2-bájtos címe

A felső helyértékű címbájttal (*8.9. ábra*) 256, ***lap***nak is nevezett memóriamező címezhető meg. Ezért a felső helyértékű címbájtot ***lapcím***nek is szokták nevezni. Az alsó helyértékű címbájttal egy lap keretén belül 256 rekesz címezhető meg, Mivel egy rekeszben egy adatszó tárolható, az alsó helyértékű címbájtot ***szócímnek*** is nevezik. A címet rendszerint hexadecimális karakterekkel ábrázolják (*8.2. táblázat*).



8.2. táblázat. A 8-bites mikroprocesszorok 16-bites címei hexadecimális karakterekkel kifejezve

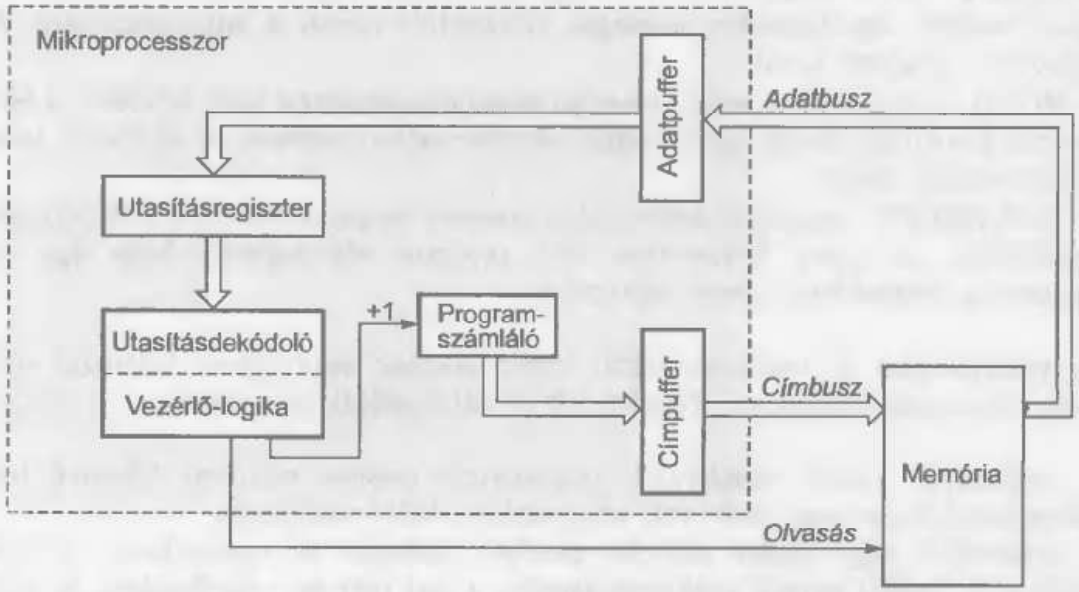
A ***veremtár-mutató*** (angolul: Stack Pointer) egy másik fontos címregiszter. A veremtár-mutató a veremmemória (stack memory) legfelső rekeszének címét tartalmazza.

A címregiszterek közül még megemlítjük az ***indexregisztert***. Ennek szerepével a továbbiakban, a címzési módszerek bemutatásánál ismerkedhetünk meg.

8.3.1.4. Az utasításregiszter (Instruction Register) és az utasításdekódoló (Instruction Decoder)

A mikroprocesszor időbeni működésére jellemző az *utasításciklus*. Ezt két egymásutáni utasítás kezdete határolja el, és három alapvető fázisra tagolódik:

- az *utasításelérés*,
- *utasításdekódolás* és az
- *utasításvégrehajtás*.



8.9. ábra. Az utasítás kiolvasása és dekódolása

Az *utasításelérés* alatt a mikroprocesszor a programszámláló által kijelölt címről kiolvassa az utasítást. Ez az adatpufferen keresztül eljut az utasításregiszterbe (*8.9. ábra*). A következő fázisban az utasításdekódoló az utasításregiszter tartalmát dekódolja. Végül, az utolsó fázisban, a vezérlő logika a dekódolt utasítás függvényében irányítja a mikroprocesszor, valamint a mikroszámítógép egységeit az utasítás végrehajtása céljából. Az utasításdekódolás folyamata, ha a mikroprocesszort rendszer-alkatelemként tekintjük, elkülöníthetetlen a végrehajtás folyamatától (*8.10. ábra*).

8.10. ábra. A mikroprocesszor tipikus utasítás ciklusa

Egyes mikroprocesszorok működési sebességének növelése céljából az átlapolt processzorciklusú (angol elnevezése: overlapped processing cycle) megoldást használják. Ez nagyjából abban áll, hogy a mikroprocesszor, miközben az utasítást a program-memóriából olvassa ki, az azelőtti utasítás végrehajtását is végzi.

8.3.1.5. A vezérlőlogika és állapotidőzítő (Control and Timing Unit)

A vezérlőlogika és állapotidőzítő egy szinkron sorrendi hálózat. Ennek az órajelét az órajelgenerátor szolgáltatja. Bemeneteire a mikroprocesszorhoz érkező külső vezérlőjelek, valamint az utasításdekódoló kimenőjelei vannak csatolva. A következő négy külső vezérlőjelet említjük meg mint legfontosabbakat:

- **RESET** – *jel*, a kezdeti feltételek beállítására (mint például a programszámláló nullázása, az akkumulátor és a státusregiszter törlése);
- **READY** – *jel*, a visszajelzés arról, hogy a memória vagy a ki/beviteli áramkör kész a program további végrehajtására; nemleges visszajelzés esetén a mikroprocesszor WAIT (várakozási) – állapotba kerül;
- **HOLD** – *állapot kérő jel*, a közvetlen memóriahozzáférést teszi lehetővé; a HOLD-állapotban a mikroprocesszor felszabadítja vezérlése alól a címbuszt, az adatbuszt, valamint a vezérlőbusz egy részét;
- **INTERRUPT** – *megszakításkérő jel* (a program megszakítására); a mikroszámítógép megszakíthatja az éppen folyamatban levő program végrehajtását, hogy egy másik, pillanatnyilag fontosabb műveletet végezzen el.

A vezérlőlogika az utasításdekódoló kimenőjeleinek segítségével határozza meg az utasítás végrehajtásának lépéseit. Feladatkörét az alábbi példák szemléltetik:

– regiszterek közötti adatátvitelt meghatározó utasítás esetében lehetővé teszi a forrásregiszterből a célregiszterbe való adatáramlást a belső adatbuszon;
– aritmetikai vagy logikai művelet esetében biztosítja az operandusok eljutását az aritmetikai és logikai egység tárolóregisztereibe, a kért művelet végrehajtását, és végül az eredmény átvitelét az akkumulátorba (kivételes esetekben egy regiszterbe vagy egy memóriarekeszbe);
– memóriából való adatkiolvasás esetében a memóriarekesz címét a címbuszra, az olvasó jelet a vezérlőbuszra és végül a kiolvasott adatot az adatbuszon a célregiszterig juttatja;
– memóriába való adatbeírás esetében a memóriarekesz címét a címbuszra, a tárolandó adatot az adatbuszra és végül az írásjelet a vezérlőbuszra juttatja.

A vezérlőlogika és állapotidőzítő az utasítások végrehajtásán kívül növeli a programszámláló tartalmát, amely a soron következő utasítás címe. Kivételt képeznek az ugróutasítások.

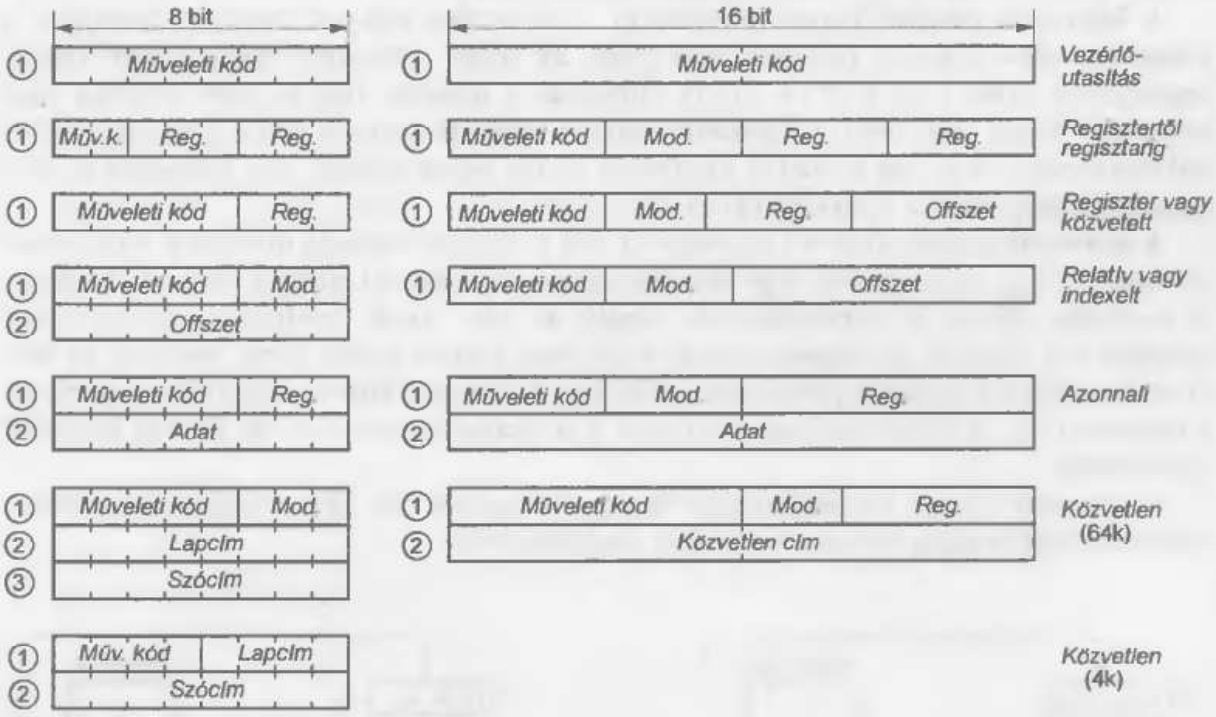
8.3.1.6. Az órajelgenerátor (Clock generator)

Egyes mikroprocesszorok órajelgenerátorral vannak ellátva. Másoknál az órajelgenerátor különálló egység. Az órajel frekvenciáját a legtöbb esetben egy kvarckristály adja meg. Amikor a frekvencia-stabilitási igények nem túl nagyok, akkor megfelel egy egyszerű RC áramkör is.

8.3.2. A mikroprocesszorok utasításai

Az utasítások (*Instruction*) a programmemóriában az adatokhoz hasonlóan bináris formában vannak tárolva. A szoftvertervezésnél az ***emlékeztető szimbólumokkal*** (*Mnemonics*) való jelölésüket használják. Ezek az utasítások angol nyelvű elnevezésének megfelelő rövidítései. A programmemóriába való beírásának megkönnyítése végett az utasítások hexadecimális ábrázolását használják (amint láttuk, a hexadecimális karakterekkel ábrázolt számot könnyen át lehet alakítani binárissá).

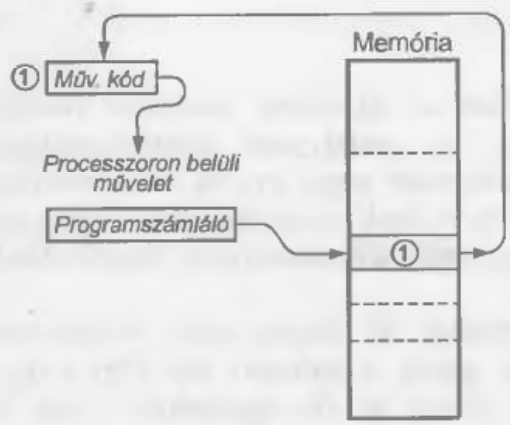
Az utasítások hossza változó: egyszavas, kétszavas és háromszavas utasításokat különböztethetünk meg (*8.11. ábra*). Legfontosabb részük a ***műveleti kód*** (Op code). Általában az utasítás tartalmazza az operandusok címeit is. Az operandusok vagy a mikroprocesszor regisztertömbjében, vagy a memóriában, vagy a beviteli áramkörben találhatók. Az utasítás formátumát a mikroprocesszor architektúrája határozza meg, amely lehet regiszter-, illetve memória-orientált. A regiszterorientált architektúrájú mikroprocesszorok utasításai a regiszterek megcímzését, míg a memóriaorientált architektúrájúaké a memória megcímzését teszik lehetővé.



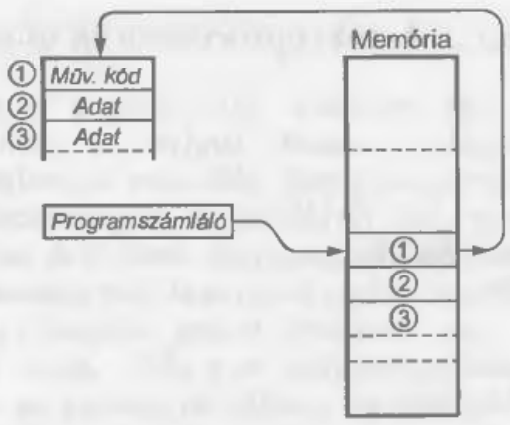
8.11. ábra. Tipikus 8-bites és 16-bites utasításformátumok

Az alábbiakban az ismertebb címzési módszerek kerülnek bemutatásra (a vonatkoztatások a 8-bites mikroprocesszorokra érvényesek, de kis eltéréssel kiterjeszthetők a 16 vagy 32-bitesekre is).

Az utasításban foglalt címzés (*Inherent-, Implied addressing*) esetében az utasításból egyértelműen következnek az operandusok. Ezt a típusú címzést használó utasítások egybájtosak (*8.12. ábra*). Általában egy processzoron belüli műveletet hajtanak végre. Ezért a hatékonyságuk a regiszterorientált mikroprocesszoroknál nyilvánul meg.



8.12. ábra. Utasításban foglalt címzés



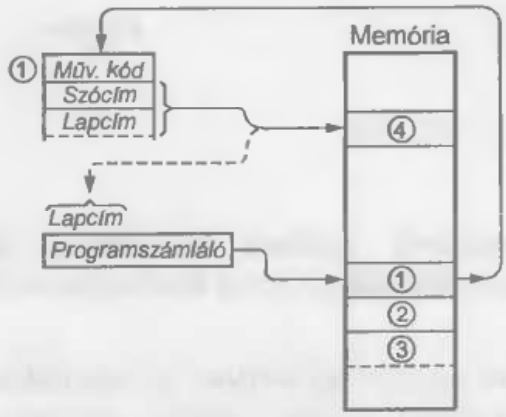
8.13. ábra. Közvetlen beírás

A közvetlen beírás (*Immediate addressing*). Az ezt a típusú címzést használó utasítások két- vagy- hárombájtosak (*8.13. ábra*). A műveleti kód után következő egy- vagy két bájt az operandust képviselő egy-, illetve kétbájtos adat. A ROM típusú programmemóriájú mikroszámítógépeknél ez az adat csak egy fix érték lehet.

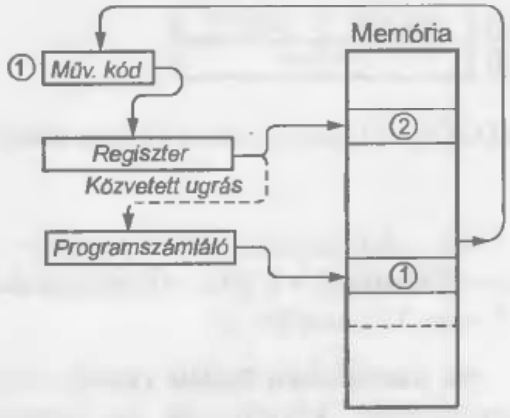
A közvetlen címzés (*Direct addressing*). A közvetlen címzést használó utasítások a memóriát közvetlenül a műveleti kód vagy az utána következő bájt, illetve bájtok segítségével címezik meg (*8.14. ábra*). Általában a második bájt az adat címének alsó helyértékű bájtja (szócíme), a harmadik pedig a felső helyértékű bájtja (lapcím). Egyes mikroprocesszoroknál, ha az adat és az utasítás azonos lapon vannak, nem szükséges az adat lapcímét is megadni. Ez a rövidített közvetlen címzés.

A közvetett címzés (*Indirect addressing*). Ezt a címzést használó utasítások a memóriát közvetetten, egy regiszter vagy egy memória-rekesz segítségével címezik meg (*8.15. ábra*). A regisztert, illetve a memóriarekeszt, amely az adat címét tartalmazza, az egybájtos műveleti kód jelöli ki. Az *azonnali címzés* a közvetett címzés sajátos esete, amelynél az adat címét tartalmazó regiszter a programszámláló. Egy különleges közvetett címzés az, amelynél a regiszter (vagy a memória-rekesz) tartalmát a programszámláló veszi át. Ez egy közvetett ugróutasítás.

A közvetett címzést használó utasítások nagyon hatékonyak. Egy egybájtos utasítással a mikroszámítógép teljes operatív memóriája megcímezhető.



8.14. ábra. Közvetlen címzés

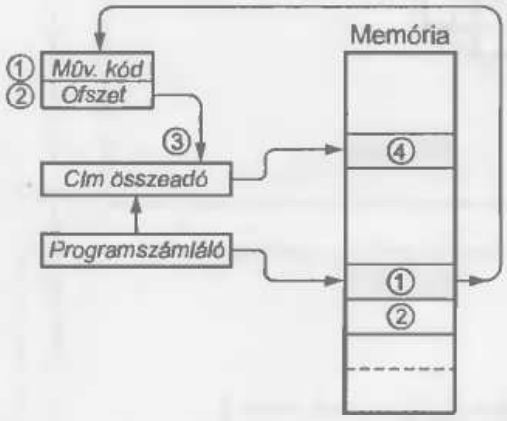


8.15. ábra. Közvetett címzés

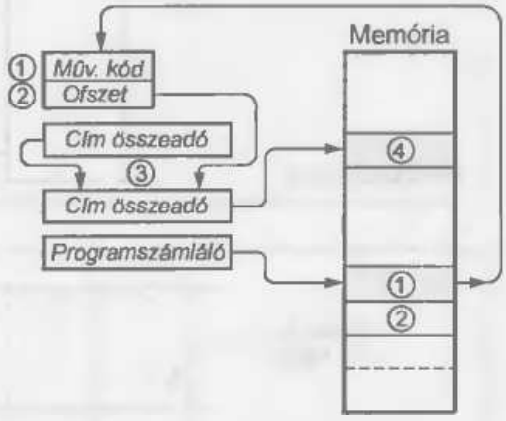
A relatív címzés (*Relative addressing*) esetében az adatot tartalmazó memóriarekesz címét a mikroprocesszor számítás után kapja meg úgy, hogy egy referenciacímhez hozzáadja az ofszet-, vagy eltoláscímet. A referenciacím származhat a programszámlálóból, egy címregiszterből vagy egy memóriarekeszből. Ha a programszámlálóból származik, akkor ***program-relatív címzésről*** beszélünk (*8.16. ábra*). Ez nagyon előnyös az *áthelyezhető programok* (Relocatable Program) szerkesztésénél, amelyek az ugróutasítások által megadott címek megváltoztatása nélkül a programmemória bármelyik részébe áthelyezhetők.

Ha a referenciacim egy címregiszterből vagy egy memóriarekeszből származik, akkor ***bázis-relatív címzésről*** beszélünk. Ennek a fontossága az adatkezelés esetében nyilvánul még. Segítségével áthelyezhető adatkezelő programok szerkeszthetők. Adattárolás céljából fenntartott bizonyos memóriamező az operatív memória akármelyik részébe áthelyezhető. Ezt ***dinamikus memóriafelosztásnak*** (Dynamic Memory Allocation) nevezik.

A relatív címzést használó utasítások végrehajtási idejének csökkentése végett a címet az adatkezeléssel elfoglalt aritmetikai és logikai egységtől különálló címösszeadó végezheti el.



8.16. ábra. Program-relatív címzés

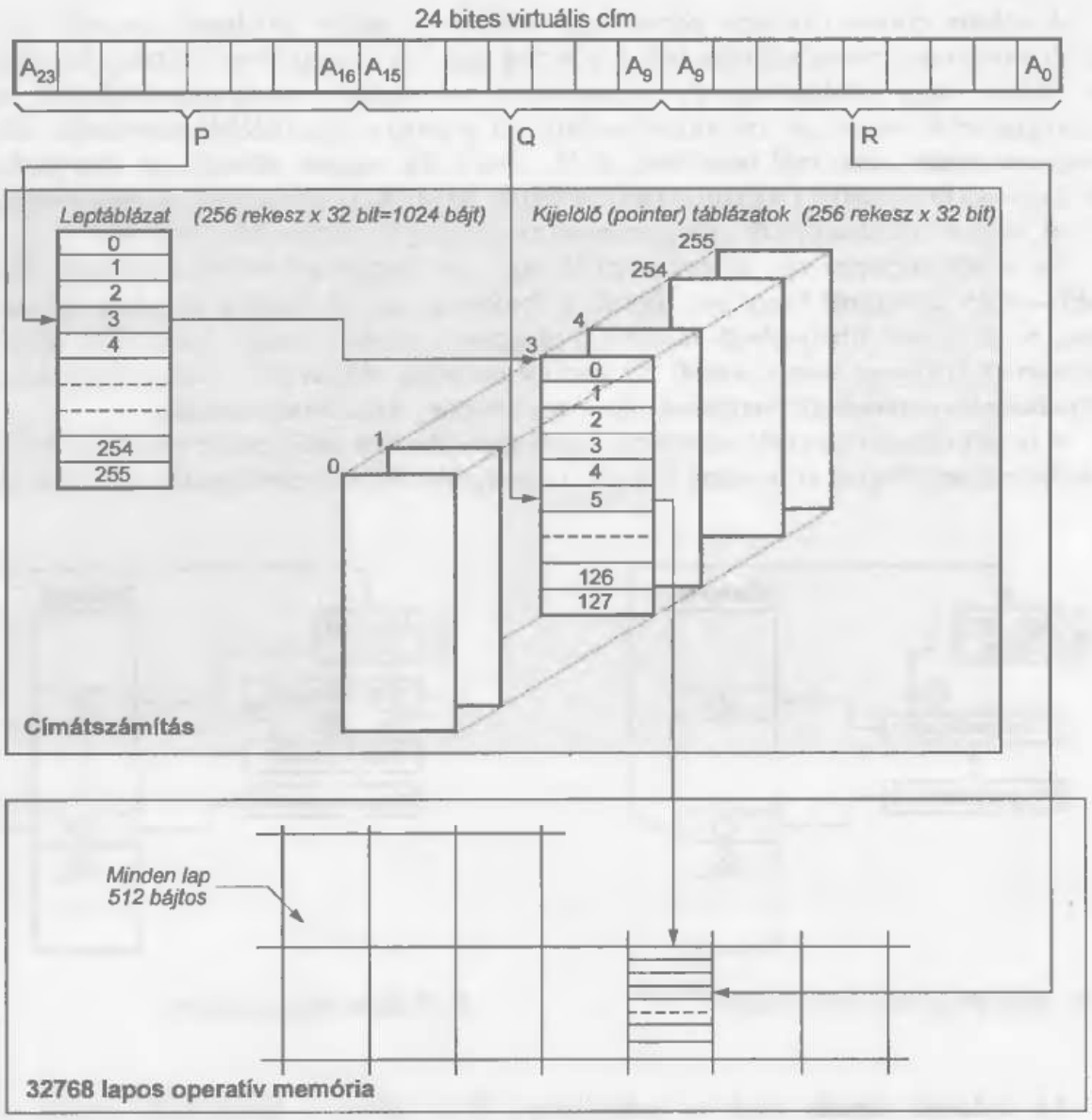


8.17. ábra. Indexelt címzés

Az indexelt címzés (*Indexed addressing*) (*8.17. ábra*) a bázisrelatív címzés egy változata. A bázis-relatív címzésnél a referenciacím fix, míg az indexelt címzés esetében a referenciacím növelhető vagy csökkenthető. A referenciacimet tartalmazó regisztert ***indexregiszternek*** nevezik.

A virtuális címzés (*Virtual addressing*) a nagyobb teljesítményű, 16-bites és 32-bites mikroprocesszorokra jellemző. A virtuális memória és a fizikailag meglevő operatív memória azonos nagyságú lapokra van felosztva (a 16 bites mikroprocesszoroknál a lap általában nagyobb, mint a 8-biteseknél szabványos 256 bájt). Az egész virtuális memória tartalma a háttérmemóriában van tárolva. Az operatív memóriába a virtuális memória azon lapjai kerülnek, melyek az éppen futó programban közvetlenül szükségesek. A virtuális memória segítségével a felhasználó hosszabb programokat szerkeszthet anélkül, hogy figyelembe venné az operatív memória fizikai határait.

A virtuális címzést a *8.18. ábra* szemlélteti. A 24 bites címszó felső 15 bitje ($A_{23}A_{22}....A_{11}A_{10}A_9$) egy lap virtuális címét tartalmazza. A virtuális címből a valós cím kiszámítása a laptáblázatok alapján történik. A virtuális cím ***P*** része a laptáblázat 256 rekesze közül egyet jelöl ki. Ennek a rekesznek a tartalmával a 256 kijelölő táblázatokból egy választható ki.



8.18. ábra. Virtuális címzés

A virtuális cím Q része a kijelölő táblázat egyik rekeszét címezi meg. Ennek a rekesznek a tartalma a fizikailag meglevő memória egyik lapját jelöli. A virtuális cím utolsó, R része ($A_8A_7....A_2A_1A_0$) a kijelölt lap egyik rekeszét címezi meg.

A továbbiakban a jellegzetes mikroprocesszor-utasításokat ismertetjük. A különböző mikroprocesszor-utasítások emlékeztető szimbólumainak jobb megértése végett az utasítások angol nyelvű elnevezését használjuk.

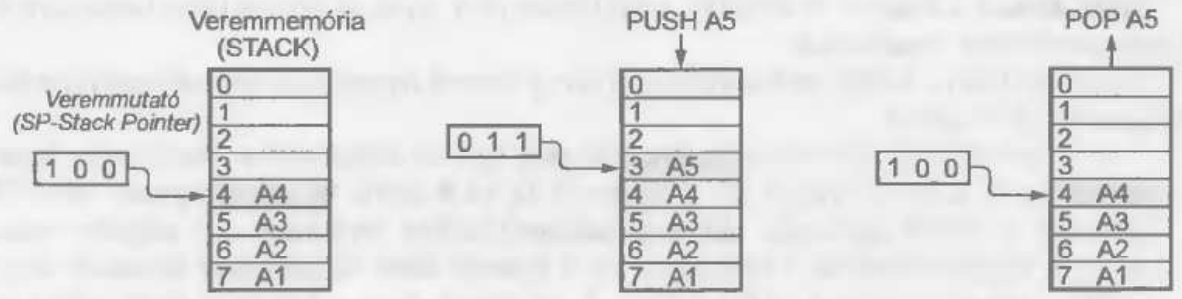
8.3.2.1. Adatmozgató utasítások

Load, Move utasítások hatására a processzor a kijelölt forrásregiszter tartalmát átmásolja a kijelölt célregiszterbe.

Store utasítás eredményeképpen a processzor a forrásregiszter tartalmát egy memória-rekeszben tárolja.

Exchange utasítás után két regiszter tartalma helyet cserél.

PUSH és **POP** (vagy **PULL**) két, speciális adatmozgató utasítás, melyek a veremmemóriába való adatbeírást, illetve kiolvasást eredményezik. Szemléltetésük a *8.19. ábrán* levő hipotetikus 8-rekeszes veremmemóriával történik. A PUSH utasítás hatására, az A5 adat beíródik a veremmutató által megcímzett veremmemória 3. rekeszébe.



8.19. ábra. A veremmemória és a veremmutató a PUSH, valamint a POP művelet után

A **POP** utasítás hatására a mikroprocesszor a veremmutató által megcímzett veremmemória 3. rekeszében tárolt A5 adatot kiolvassa. A POP utasítás végrehajtása után, a veremmutató a soron következő 4. rekesz címét tartalmazza.

Input utasítás egy adatbeviteli áramkör regiszterének tartalmát egy regiszterbe vagy egy memóriarekeszbe írja be.

Output utasítás egy regiszter vagy egy memóriarekesz tartalmát egy adatkiviteli áramkör regiszterébe írja be.

8.3.2.2. Aritmetikai utasítások

Add az összeadás műveletét jelzi.

Add with carry utasítás is az összeadást jelzi. Az operandusok összegéhez az ezelőtti művelet során keletkező átvitelbitet is hozzáadja.

Subtract a kivonás műveletét jelzi.

Subtract with carry utasítás is a kivonást jelzi. Az operandusok különbségéből az ezelőtti művelet során keletkező átvitelbitet is levonja.

Increment utasítás hatására egy regiszter vagy egy memória-rekesz tartalma 1-gyel növekszik.

Decrement utasítás hatására egy regiszter vagy egy memória-rekesz tartalma 1-gyel csökken.

Multiply a szorzás műveletét jelzi, és csak a nagyobb teljesítményű mikroprocesszorok utasításkészletében lelhető fel.

Divide az osztás műveletét jelzi, és fentihez hasonlóan csak a nagyobb teljesítményű mikroprocesszorok utasításkészletében lelhető fel.

Decimal Adjust a decimális helyesbítést végző utasítás. Segítségével a tiszta bináris számokkal dolgozó aritmetikai és logikai egység BCD kódolású számokkal is tud műveleteket végezni.

8.3.2.3. Logikai utasítások

And a logikai *ÉS* műveletet jelzi. Az eredmény minden egyes bitjét a két operandus megfelelő helyértékű bitjei között elvégzett *ÉS* művelet adja.

Or a logikai *VAGY* műveletet jelzi.

Exclusive – Or a logikai KIZÁRÓ-VAGY műveletet jelzi.

Complement az operandus komplemensét képezi.

Compare az összehasonlítás utasítása. Az összehasonlítás eredményét a feltételbitek megfelelő beállítása jelzi.

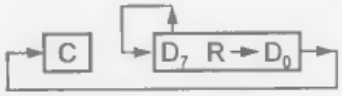
Shift, Rotate a léptetési és forgatási műveleteket jelzi. Ezek az akkumulátor tartalmára és a carry átvitelbitre vonatkoznak.

Az alábbiakban a 8-bites mikroprocesszorokra jellemző léptetési és forgatási műveleteket ismertetjük (*8.20. ábra*).

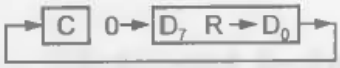
– *Aritmetikai léptetés balra* (Arithmetic shift up): az akkumulátor összes bitjei egyet lépnek balra, a D_7-bit átlép a *CY* átvitelbe és D_0-ba **0** íródik be. Ha a léptetés után CY átvitelbit és D_7-bit azonosak, akkor az akkumulátorban levő szám egy előjeles szám, amely a léptetés előttinek a kétszerese. Ha a léptetés előtti akkumulátor tartalmát előjel nélküli számnak tekintjük, akkor a $CY = 1$ azt jelenti, hogy a kétszeres érték túllépi az akkumulátor kapacitását.

– *Aritmetikai léptetés jobbra* (Arithmetic shift down): az akkumulátor összes bitjei egyet lépnek jobbra, a D_0 bit átlép a *CY* átvitelbe, és a D_7-bit változatlan marad. A léptetés eredménye fele a léptetés előtti számnak. Ha a léptetés után $CY = 0$, akkor a felezés eredménye maradék nélküli. Ellenkező esetben, ha $CY = 1$, akkor a felezés nem maradék nélküli. Az eredmény lefelé kerekített.

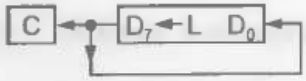
a) aritmetikai léptetés balra (szorzás 2-vel)



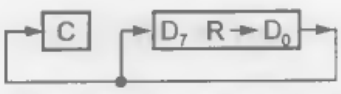
b) aritmetikai léptetés jobbra (osztás 2-vel)



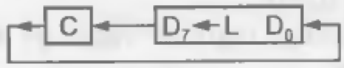
c) logikai léptetés jobbra



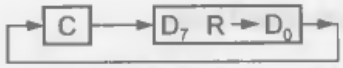
d) ciklikus léptetés (forgatás) balra



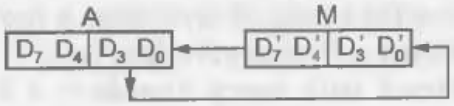
e) ciklikus léptetés (forgatás) jobbra



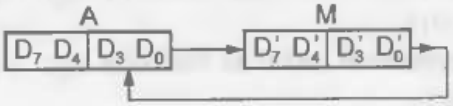
f) ciklikus léptetés az átvitelbiten keresztül balra



g) ciklikus léptetés az átvitelbiten keresztül jobbra



h) ciklikus tetrád-léptetés balra



i) ciklikus tetrád-léptetés jobbra

8.20. ábra. Léptetési műveletek

– *Logikai léptetés jobbra* (Logical shift right): ez a léptetési művelet abban különbözik az előbbitől, hogy a D_7-be **0** íródik be. A léptetés eredménye fele a léptetés előtti előjel nélküli számnak. A maradékot, az előbbihez hasonlóan, a *CY* átvitelbit tartalmazza.

– *Ciklikus léptetés (forgatás) balra* (Cyclic shift left, Rotation left): az akkumulátor összes bitjei ciklikusan lépnek egyet balra, a D_7-bit átlép a D_0-ba, és átmásolódik a *CY* átvitelbe.

– *Ciklikus léptetés (forgatás) jobbra* (Cyclic shift right, Rotation right): ez a léptetési művelet az előbbihez hasonló, csak a léptetési irány különbözik.

– *Ciklikus léptetés (forgatás) az átvitelbiten keresztül balra* (Rotation through carry left): a *CY* átvitelbitet az akkumulátor 9. bitjeként kezeli. A léptetési műveletben D_7 és D_0 között helyezkedik el. Az összes bitek egyet lépnek balra, a *CY* átvitelbit D_0-ba lép, az átvitelbitbe pedig D_7 lép át.

– *Ciklikus léptetés (forgatás) az átvitelbiten keresztül jobbra* (Rotation through carry right): ez a léptetési művelet az előbbihez hasonló, csak a léptetési irány különbözik.

– *Ciklikus tetrádléptetés balra*: ez az utasítás ciklikusan léptet három tetrádot balra. Ezek közül kettő egy memóriarekeszben van, a harmadik pedig az akkumulátorban. Az akkumulátor legnagyobb helyértékű tetrádja változatlan marad.

– *Ciklikus tetrádléptetés jobbra*: ez az utasítás hasonló az előbbihez, a BCD kódolású számokkal végzett műveleteknél van jelentősége, és csak a léptetési irány különböző.

8.3.2.4. Vezérlő utasítások

A vezérlő utasítások lehetnek *feltételesek* és *feltétel nélküliek*. A feltételes utasítás végrehajtása egy feltétel flip-flop állapotától függ.

Jump' Skip utasítások a programszámláló folyamatos növekedésében egy ugrást iktatnak be. A Jump ugróutasítás a legtöbb mikroprocesszornál feltétel nélküli, míg a Skip feltételhez kötött. Ha a feltétel teljesül, akkor a program további végrehajtása az ugróutasítással jelzett címtől folytatódik. Ellenkező esetben, ha a feltétel nem teljesül, akkor az ugrás nem következik be, a program végrehajtása a soron következő utasítással folytatódik tovább.

Call Subroutine' Jump Subroutine szubrutinhívási utasítások. Ezek is lehetnek feltételesek és feltétel nélküliek. A programszámláló tartalmát, amely a soron következő utasítás címe, a veremmemória a szubrutinra való ugrás előtt elmenti. Ezután a programszámlálóba a szubrutin kezdő címe íródik be.

Return a szubrutinból való visszatérési utasítás. Ez is lehet feltételes vagy feltétel nélküli. A programszámlálóba a szubrutinra való ugrás előtti, a veremmemóriába elmentett tartalom íródik be.

A Halt utasítás megállítja a programszámlálót, és ennek következtében a program további végrehajtását is. A megállítás addig tart, amíg egy külső jel hatására a mikroprocesszor kikerül a *Halt* állapotból, és újraindítja a programszámlálót.

8.3.3. A megszakítás

Miközben a mikroprocesszor a főprogram végrehajtásával van elfoglalva, előállhat egy fontosabb feladat megoldása, amelynek eredménye történetesen kihathat a főprogramra is. Ebben az esetben a főprogramot meg kell szakítani, hogy a mikroprocesszor áttérjen arra a programra, amely az éppen előállt feladat megoldására szolgál.

A mikroprocesszor kap egy megszakításkérő jelet (*Interrupt Request*). A megszakítást a megszakítás-elismerő jellel nyugtázza (*Interrupt Acknowledge*). A megszakítás idejére a programszámláló valamint a belső regiszterek tartalmát elmenti a veremmemóriába.

Bonyolult rendszer esetében a mikroprocesszor több helyről is kaphat megszakításkérést. A megszakítási módszer lehet ***lekérdező*** (*Polled Interrupt*) vagy **vektorizált** (Vectorized Interrupt). Az első esetben a mikroprocesszor a perifériás áramkörök megszakításkérő kimeneteit sorra lekérdezi. Ha valamelyikben megjelenik a megszakításkérő jel, akkor végrehajtja a megfelelő megszakítási programot, és továbblép. A megszakítás elsőbbségét a periferikus áramkörnek a lekérdezési szekvenciában levő relatív helyzete határozza meg. A vektorizált megszakítás hatékonyabb, mint a lekérdezéses. Ebben az esetben a mikroprocesszor válaszként a megszakításkérésre az adatbuszon egy periféria-azonosító címet küld ki. Ez az ún. *vektor*. A megszakításkérő periféria azonosítása után elvégzi a megfelelő megszakítási programot.

8.3.4. Jellegzetes mikroprocesszor típusok

A jelenleg gyártott sok különböző mikroprocesszor közül két nagy családot emelhetünk ki. Az egyik a Motorola 6800-as típusára épül, a másik az Intel 8080-as áramkörére.

A *8.1. táblázat* – a teljesség igénye nélkül – a 6800-as család típusait foglalja össze. A 6802-es típus a 6800as továbbfejlesztése. A 6802-es utasításkészlete azonos, ezért a programok változtatás nélkül csereszabatosak. A 6802-es típust a gyártók órajelgenerátorral és 128 byte RAM-mal egészítették ki.

A 6809-esnek két címregisztere van. Utasításkészlete különösen indexelt címzésnél jelentősen kibővült.

A 68000-es típusok teljesen új generációhoz tartoznak. Minden programsíkon azonos utasításkészletük van, ami a 68010 és 68020 típusoknál csupán néhány kiegészítéssel bővül. A 68000-es típusok belül 32 bites mikroprocesszorként épülnek fel. A 68000 és 68010 esetén a külső adatbusz 16 bit széles. A 68008-nál a buszillesztő 8 bites adatbuszhoz illeszkedik. Ezért számottevő hardver-változtatás nélkül 8 bites rendszerben is működhet. A 68000 és 68010, 24 bit hosszú címbuszával 2^{24} byte = 16 Mbyte cím címezhető. Ezzel már nagyobb folyamatirányító számítógép teljesítőképességének nagyságrendjét érjük el. Utasításkészletének különleges jellegzetessége a viszonylag kevés utasítás kombinációja sok címzésmóddal. Tehát nagyon egyszerű és mégis hatékony programozásra alkalmas.

A 68010 és 68020 típusok virtuális táras üzemmódban is működtethetők (*Virtual Memory, VM*). Ebben az üzemmódban a felhasználó úgy fordulhat a háttértár adataihoz (pl. merevlemezről vagy floppy lemezről), mintha azok közvetlenül a RAM-ban lennének tárolva. A háttértár kezelését az operációs rendszer teljesen automatikusan végzi. A program futása közben előfordulhat, hogy olyan operandus kell, amelyik nem a RAM-ban, hanem külső háttértárban van. Ebben az esetben az utasítás végrehajtása félbeszakad, az operációs rendszer azt az adatszegmenst, melyben az operandus van, áttölti a háttértárból a RAM-ba, és ezután a megszakított utasítás végrehajtása is befejeződik.

A 68020-as típus 32 bit szóhosszú belső adatbusza kívülről is hozzáférhető. Ez kétszeres adatátviteli sebességet eredményez a 16 bit szóhosszal működő típusokhoz képest olyan esetekben, amikor az utasítások és az operandusok szóhossza nagyobb 16 bitnél.

A 68020-as típus ezenkívül „aritmetikai társprocesszor interfésszel" (*Coprocesszor Interface*) is rendelkezik, mellyel speciális számítógépekkel párhuzamosan képes működni, ilyen pl. a 68881 aritmetikai processzor (numeric data processor). A két processzor az utasításkód alapján felismeri, hogy melyiknek kell az utasítást végrehajtani.

Típus	Tipikus teljesítmény-felvétel (mW)	Adatbusz szóhossz bit	Cím-tartomány, byte	Adat-regiszter, bit	Cím-regiszter, bit	Utasítások száma	Órajel frekven-ciája, (MHz)	Különleges tulajdonságok
6800	600	8	64 k	2×8	3×16	72	1 ÷ 2	
6802	600	8	64 k	2×8	3×16	72	1 ÷ 2	128 byte RAM
6809	650	8	64 k	2×8	5×16	59	1 ÷ 2	
68000	1200	16	16 M	2×32	10×32	56	6 ÷ 12	
68008	1200	8	1 M	2×32	10×32	56	6 ÷ 12	
68010	1500	16	16 M	2×32	11×32	58	6 ÷ 12	VM
68020	1500	32	4 G	2×32	11×32	92	12 ÷ 16	VM, CP, CA

8.1. táblázat. A 6800-as mikroprocesszor-család áttekintése

Típus	Tipikus teljesítmény-felvétel (mW)	Adatbusz szóhossz bit	Cím-tartomány, byte	Adat-regiszter, bit	Cím-regiszter, bit	Utasítások száma	Órajel frekven-ciája, (MHz)	Különleges tulajdonságok
8080	780	8	64 k	8×8	6×8	78	2 ÷ 3	
8085	650	8	64 k	8×8	2×16	80	3 ÷ 5	
8086	1200	16	1 M	4×16	6×8	104	5 ÷ 10	CP
8088	1200	8	1 M	4×16	9×16	104	5 ÷ 8	CP
80186	1500	16	1 M	4×16	9×16	114	8 ÷ 10	CP, T, DMA, IRC
80188	1500	8	1 M	4×16	9×16	114	8 ÷ 10	CP, T, DMA, IRC
80286	2000	16	16 M	4×16	9×16	129	8 ÷ 10	CP, MMU, VM
80386		32	4 G	4×32	9×32		16 ÷ 40	CP, MMU, VM
80486		32	16 G				16 ÷ 133	ACP, MMU, VM

CP=koprocesszor interfész; T=időzítő (timer); DMA=közvetlen tárhozzáférés (Direct Memory Access); IRC=megszakításvezérlő (Interrupt Controller); MMU=tárkezelőegység (Memory management unit); VM=virtuális tár (Virtual memory); CA=cache memória; ACP=beépített aritmetikai társprocesszor

8.2. táblázat. A 8080-as mikroprocesszor-család áttekintése

A 8080-as családról a *8.2. táblázat* ad áttekintést. A 8080A alaptípus még egy régebbi technológiával készült, melynél az áramkör még három tápfeszültséget igényelt. Ezt a típust teljesen kiszorította utódja, a 8085-ös típus amely – két utasítástól eltekintve – azonos utasításkészlettel és azonos gépi kóddal működik. A 8085-ös processzor hátránya, hogy bizonyos utasítások hiánya miatt aritmetikai teljesítménye igen korlátozott.

A 8086-os típus adatbusza és aritmetikája 16 bites. Az utóbbi a szorzást és osztást is tartalmazza. A 8088 utasításkészlete azonos, azonban adatbusza 8 bites. A 80186 esetén a 8086-hoz képest néhány vezérlő funkciót is integráltak, amit különösen nagyobb rendszerekben a felhasználónak kellene csatlakoztatni. Tartalmaz továbbá egy időzítőt (*timer*), egy DMA-vezérlőt (közvetlen tárhozzáférés vezérlőt) a gyors adatátvitelhez és egy megszakításvezérlőt több megszakítás (*interrupt*) feldolgozásához.

A 80286 a 8086 továbbfejlesztett változatának tekinthető. Hasonlóan a 68000-es családhoz különleges operációs rendszerutasításokkal működhet. Ezenkívül a 80286-ot ellátták tárkezelő-egységgel (angolul: memory management unit), és a 68020-hoz hasonlóan virtuális tárkezelésre is alkalmas. A 8086-os processzor család minden típusa aritmetikai társprocesszorral is képes együttműködni, mint pl. a 8087 aritmetikai vagy a 8099 ki/beviteli (I/O) processzorral.

A 80386-nál a cím- és adatbusz szóhossza 32 bit, amely a 80286 továbbfejlesztése. Ez megfelel a 68020-nak és majdnem eléri a hatékonyságát is.

A 80486 esetén az aritmetikai társprocesszor nem külső áramkörként csatlakozik a rendszerhez, hanem ugyanazon a chip-en van integrálva. A 80486-os áramkör a 80386 továbbfejlesztésének tekinthető.

A korszerű mikroprocesszorok teljesítményfelvétele nagyságrendekkel kisebb, mint a korábbi TTL technikával készült processzoroké. Ennek ellenére telepes működés számára mégis túl nagy a teljesítményfelvétel. Ezért nagyon fontos, hogy a legtöbb egyszerű mikroprocesszor hardver- és szoftverkompatibilis CMOS változatban is készül, amelyekből néhányat a *8.3. táblázat* foglal össze.

NMOS típusok	CMOS típusok	Teljesítmény-felvétel (mW)	Az órajel frekvenciája (MHz)	Gyártók
6802	MD 68SC02	75	5	Mitel
6809	HD 6309	80	5	Hitachi
8085A	MSM 80C85	50	3	Oki
8085	HD 80C86	150	8	Harris
8088	HD 80C88	150	8	Harris

8.3. táblázat. A 6800-as és a 8080-as család CMOS mikroprocesszorai

A CMOS technológiával készült processzoroknak belső statikus táruk van. Ezért megengedett, hogy órajel frekvenciáját tetszőlegesen csökkentsük. Mivel az áramfelvétel arányos a frekvenciával, ezért kisebb frekvenciájú órajellel áramot takaríthatunk meg. Az órajel frekvenciája akkora kell legyen, hogy a működési sebesség még éppen kielégítse a követelményeket.

8.3.5. Az Intel 8085 mikroprocesszor

A 8080 és 8085 mikroprocesszorok szoftver szempontjából összeegyeztethetőek. Az utóbbi két addicionális utasítását kivéve az utasításkészletük azonos. Hardver szempontjából viszont e két 8-bites mikroprocesszor között jelentősebb eltérés van. A 8085 mikroprocesszor a 8080 továbbfejlesztett változata. Ugyanazon a chipen, több egységet is integráltak (pl. központi feldolgozó egység, órajelgenerátor) amelyek feladatát a 8080 esetében külső áramkörök valósítottak meg. A műveleti sebességét jelentősen növelték. Az átlagos utasításciklus 2 μs-ról 1,3 μs-ra csökkent. Nem igényel három tápfeszültséget, csak egyet a szabványos +5 V-ost.

A 8085-ös mikroprocesszor tömbvázlatát a *8.21. ábra* szemlélteti. Azért, hogy a mikroprocesszor egy 40 lábas tokban kapjon helyet, az alsó helyértékű adatbájt és címbájt multiplex üzemű buszrendszerre került. A gépi ciklus első órajelperiódusa alatt az $AD_0 \div AD_7$ adat/címbuszon az alsó helyértékű címbájt ($A_0 \div A_7$) jelenik meg. Ezután $AD_0 \div AD_7$ az adatbusz szerepét tölti be.

Az óragenerátor az időzítő- és vezérlőegység egy része. Az X_1 és X_2 kivezetésekre az órajelfrekvenciát meghatározó kvarckristályt kell kapcsolni. A *CLK OUT* kivezetésen az órajel vehető le, frekvenciája fele a kvarckristály párhuzamos rezonancia frekvenciájának, amely legkevesebb 1 MHz, legtöbb 6 MHz lehet. Az időzítő- és vezérlőegység kívülről a következő jeleket kapja: *READY*, $\overline{RESET\ IN}$ és *HOLD*. Kifelé a $\overline{RD}$, $\overline{WR}$, *ALE* (*Address Latch Enable*), *HLDA* és *RESET OUT* vezérlőjeleket valamint S_0, S_1 és $IO/\overline{M}$ státuszjeleket (állapotjeleket) szolgáltatja. Az utóbbiak jelentése a *8.4.a táblázatban* van összefoglalva.

S_0	S_1	*Státusz*
0	0	HALT (megállás)
0	1	WRITE (írás)
1	0	READ (olvasás)
1	1	FETCH (utasításelérés)

$IO/\overline{M}$	$\overline{RD}/\overline{WR}$ *vonatkozások*
0	Memória
1	Ki/bemeneti áramkör

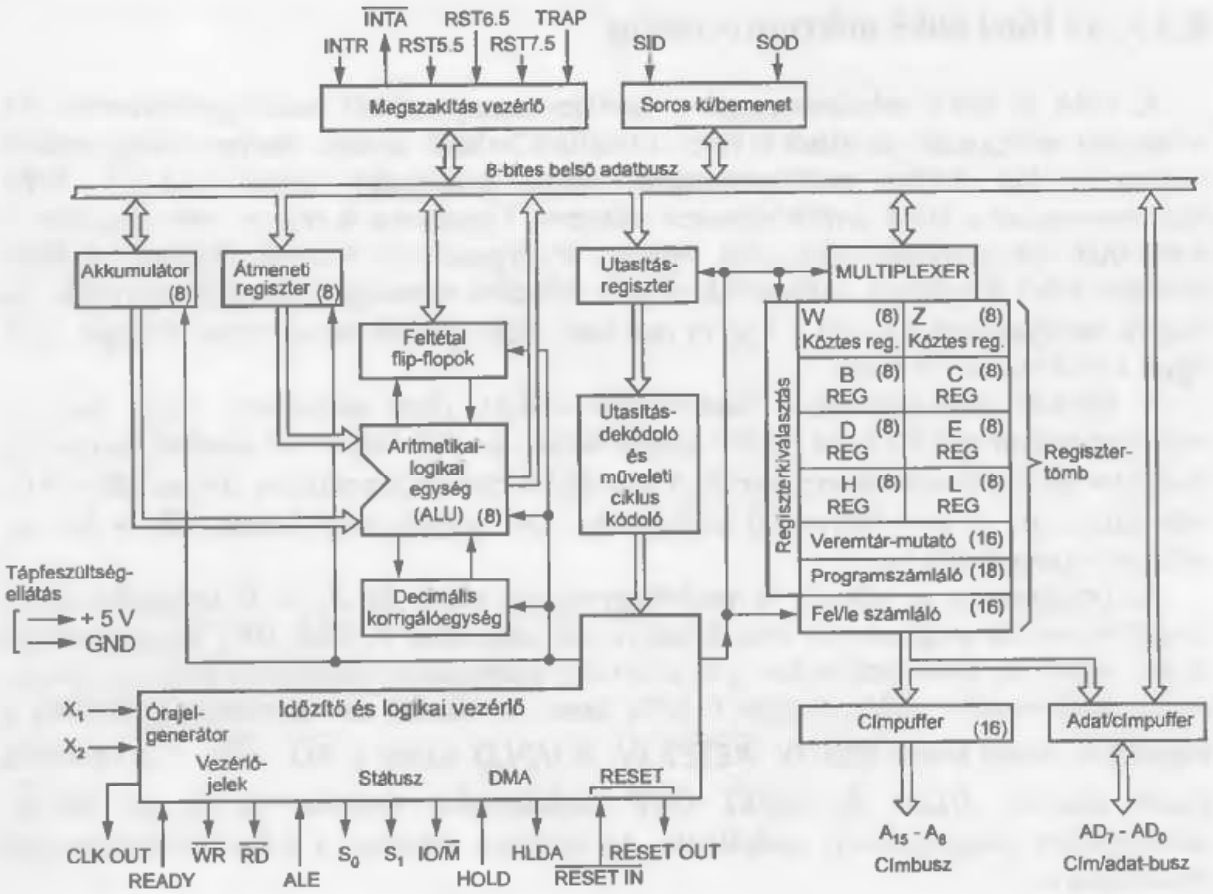
a) státuszinformációja

Elnevezés	*Restart cím (Hexa)*
TRAP	0024
RST 5.5	002C
RST 6.5	0034
RST 7.5	003C

b) restart (újraindítás) címei

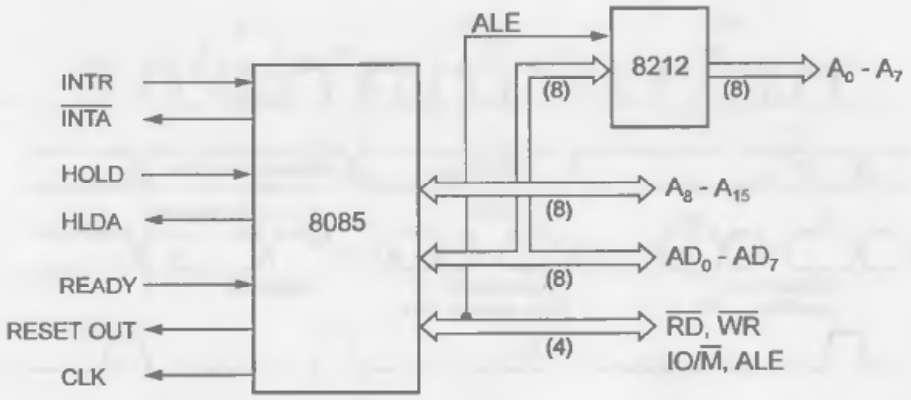
8.4. táblázat. A 8085 mikroprocesszor

A megszakítás-vezérlő egységnek öt megszakításkérő bemenete van. Az *INTR* (*Intrerrupt Request*) a szabványos megszakításkérő bemenet. Az *INTA* (*Interrupt Acknowledge*) kimeneten jelzi a megszakítás elismerését. A további négy megszakításkérő bemenet *TRAP*, *RST* 5.5, *RST* 6.5 és *RST* 7.5. A *TRAP* nem elfedhető, és a legnagyobb prioritású, a legelsőnek kiszolgált megszakításkérés. Az *RST* 5.5, *RST* 6.5 és *RST* 7.5 (*Restart Interrupts*) elfedhető (maszkolható) megszakítás-kérő bemenetek, és növekvő prioritás szerint vannak felsorolva. Az utóbbi négy megszakítás esetében a mikroprocesszor a program további végrehajtását a *8.4.b táblázat*ban megadott címeken folytatja. Ezek tulajdonképpen a megfelelő megszakítást kiszolgáló szubrutinok kezdő címei.



8.21. ábra. A 8085 mikroprocesszor tömbvázlata

$A_8 \div A_{15}$ – adatbusz - felső helyértékű bájt (háromállapotú kimenet)
$AD_7 \div AD_0$ – multiplexelt cím/adatbusz (bemenet/ háromállapotú kimenet)
ALE – címregiszter-engedélyező kimenet
S_0, S_1 – az adatbusz státuszát (állapotát) jelző kimenetek
$\overline{RD}$ – olvasójel (háromállapotú kimenet)
$\overline{WR}$ – írójel (háromállapotú kimenet)
READY – kész (bemenet)
HOLD – HOLD állapotkérő bemenet
HLDA – HOLD állapotelismerő kimenet
INTR – megszakításkérő bemenet
INTA – megszakítás-elismerő kimenet
RST 5.5, *RST* 6.5, *RST* 7.5 – Restart (újraindítás) típusú megszakításkérő bemenetek
TRAP – nem maszkolható, legnagyobb prioritású *Restart* típusú megszakításkérő bemenet
RESET IN – RESET bemenet
RESET OUT – RESET állapotjelző kimenet
X_1, X_2 – a (kvarc) kristály vagy az R/C hálózat csatlakozási pontjai
CLK – órajelkimenet
IO/M – az írás valamint az olvasás a ki/beviteli egységre vagy a memóriára való vonatkozását jelző kimenet
SID – soros adatbemenet
SOD – soros adatkimenet



8.22. ábra. A 8085 mikroprocesszoron alapuló mikroszámítógépek buszrendszere

A soros adat ki/beviteli áramkör, amint elnevezése is mutatja, a soros adatátvitelt teszi lehetővé. A kimeneti vonal az *SOD* (*Serial Output Data*), míg a bemeneti vonal az *SID* (*Serial Input Data*).

A 8085 mikroprocesszor buszrendszerét a *8.22. ábra* szemlélteti. Az $AD_0 \div AD_7$ multiplexelt cím/adatbuszon megjelenő címbájtot az *ALE* jel írja be a 8212 áramkör 8-bites regiszterébe, amely azt megtartja a gépi ciklus további órajelperiódusai alatt is, amikor feltétlenül szükséges a memóriák és a be/kiviteli áramkörök számára. Az Intel cég ki/beviteli áramkörökkel ellátott speciális memóriákat fejlesztett ki. A 8155 típusnak RAM memóriája, a 8355, valamint a 8755 típusoknak ROM, illetve EPROM memóriájuk van. Ezeknél az alsó helyértékű címbájtot egy 8-bites regiszter tárolja.

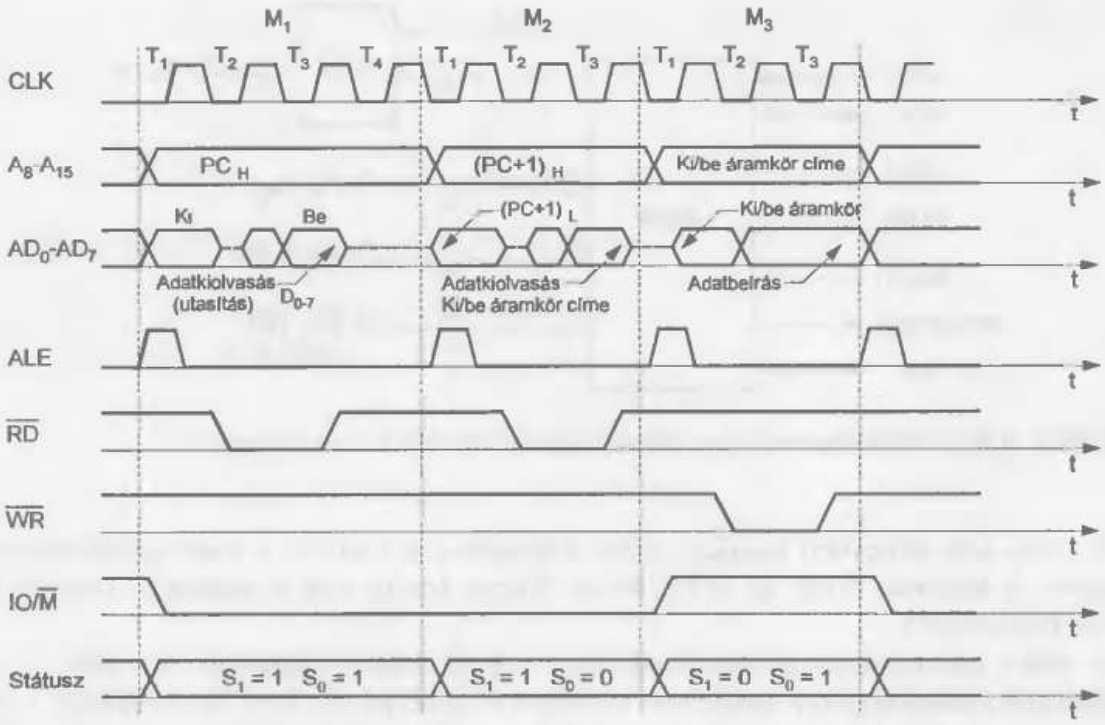
8.3.5.1. A 8085 mikroprocesszor utasításciklusa

A 8085 mikroprocesszor utasításciklusa egytől ötig terjedő gépi ciklusból tevődik össze. Egy gépi ciklus 3, 4 vagy 5 állapotból állhat, amelyben minden állapot a kétfázisú órajel periódusával egyenlő. Kilencféle gépi ciklust különböztetünk meg: INSTRUCTION FETCH (utasítás-elérés), MEMORY READ (memóriaolvasás), MEMORY WRITE (memóriaírás), STACK READ (veremtár-olvasás), STACK WRITE (veremtár-írás), INPUT (bemenet), OUTPUT (kimenet), INTERRUPT (megszakítás) és HALT (megállás). A gépi ciklusok szekvenciája a végrehajtás alatt álló utasítás függvényében alakul. Minden utasításciklus első gépi ciklusa az utasítás-elérés.

A 8085-ös mikroprocesszor olvasó-, illetve íróciklusának részletes idődiagramját a *8.23. ábra* szemlélteti.

Az olvasóciklus kezdetén, a 8085-ös mikroprocesszor a címbuszra a 16-bites címet küldi ki először. Az alsó helyértékű címbájtot az $AD_0 - AD_7$ multiplexelt cím/adatbuszon, a felső helyértékű címbájtot pedig az $A_8 - A_{15}$ címbuszon küldi ki. Ezután a vezérlőbuszon az *ALE* jelet, valamivel később az olvasójelet ($\overline{RD}$) küldi ki. A cím/adatbusz harmadik, nagy impedanciájú állapotba kerül, és így lehetővé teszi a megcímzett egység számára, hogy az adatot a cím/adatbuszra helyezze. Végül, az olvasójel ($\overline{WR}$) megszünte után új ciklus kezdődik.

A 8085-ös mikroprocesszor íróciklusa kezdetén, az olvasóciklusához hasonlóan, a 16-bites címet küldi ki. Később az adatbusz állapota kimenetire vált és kiküldésre kerül az adat. Ezután a vezérlőbuszon az *ALE* jelet és valamivel később az írójelet küldi.



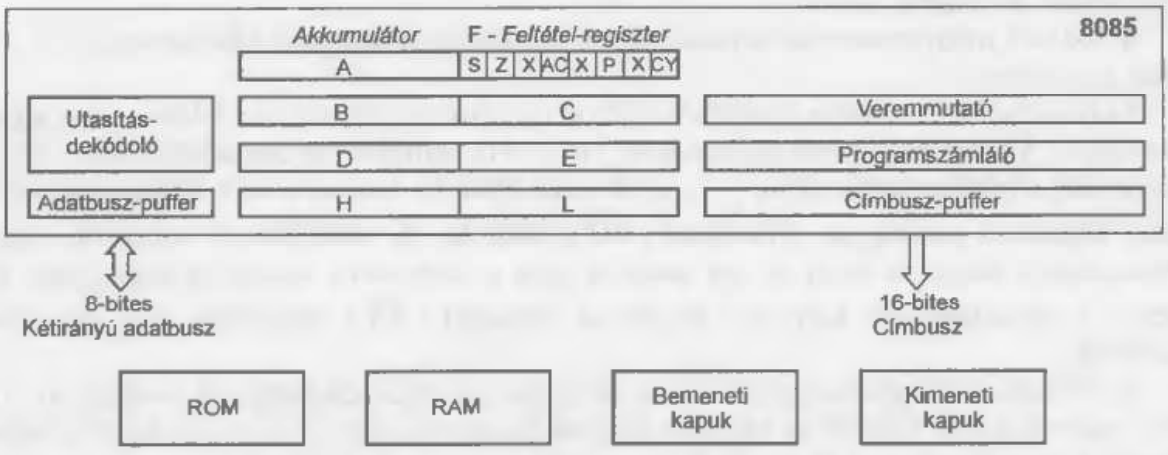
8.23. ábra. A 8085 mikroprocesszor tipikus idődiagramja

8.3.5.2. A 8080 és 8085 mikroprocesszorok utasításkészlete

A 8080 és 8085 mikroprocesszorok utasításkészlete öt csoportba rendszerezhető:

- adatmozgató utasítások;
- aritmetikai utasítások;
- logikai utasítások;
- programszámláló tartalmát változtató utasítások;
- veremtár, ki/beviteli és gépvezérlő utasítások.

Az adatok és utasítások formátumával az előzőekben már foglalkoztunk. Az utasítások vonatkoztatásai a mikroprocesszor programozási modelljén követhetők (*8.24. ábra*).



8.24. ábra. A 8085 mikroprocesszor programozási modellje

A következőkben a *8.5. táblázatban* összefoglalt szimbólumokat és rövidítéseket használjuk.

Szimbólumok	Jelentésük
akkumulátor	az A regiszter
r, r1, r2	az A, B, C, D, E, H, L regiszterek egyike
ddd, sss	a cél illetve forrásregiszter utasításkódbeli jelölése ddd vagy sss — a kijelölt regiszter 111 — A 000 — B 001 — C 010 — D 011 — E 100 — H 101 — L
rp	regiszterpár: B a B–C regiszterpárt, D a D–E regiszterpárt, H a H–L regiszterpárt, SP a 16-bites veremmutatót jelöli
RP	a regiszterpár utasításkódbeli jelölése RP — a kijelölt regiszterpár 00 — B – C 01 — D – E 10 — H – L 11 — SP
PSW	programstátusz-szó: A–F regiszterpár
rh	a regiszterpár felső helyértékű regisztere: B, D vagy H
rl	a regiszterpár alsó helyértékű regisztere: C, E vagy L
PC	a 16-bites programszámláló (PC_H és PC_L a programszámláló felső- illetve alsó helyértékű 8-bites regiszterét jelöli ki)
SP	a 16-bites veremmutató (SP_H és SP_L a veremmutató felső- illetve alsó helyértékű 8-bites regiszterét jelöli ki)
(r), (M)	a zárójelben levő regiszter vagy memóriarekesz tartalma
r_m	az r regiszter *m* bitje
Z, S, P, CY, AC	feltétel bitek:
cím	16-bites cím
adat	8-bites adat
adat 16	16-bites adat
<B_2>, <B_3>	az utasítás 2. illetve 3. bájtjának tartalma
kapu	a ki/beviteli egység 8-bites címe
[ ]	a zárójelben levő cím által kijelölt memóriarekesz tartalma
←	az adatmozgás jelölése
↔	az adatcsere jelölése
∧	logikai *ÉS*-kapcsolat
∨	logikai *VAGY*-kapcsolat
∀	logikai *KIZÁRÓ-VAGY*-kapcsolat
+	összeadás
–	kivonás a 2-es kiegészítős aritmetika szabályai szerint
*	szorzás
n	az RST utasítások számozása 0-tól 7-ig
nnn	*n* bináris alakja

8.5. táblázat. Használt szimbólumok és rövidítések

Adatmozgató utasítások

Az adatmozgató utasítások az adatokat az egyes regiszterek illetve a regiszterek és a memória között mozgatják. A feltétel bitek értékét nem változtatják meg.

- MOV r1, r2; MOV r, M; MOV M, r
  - A 2. operandust az elsőbe írja be. Az M memóriarekesz címét a HL regiszterpárban kell megadni.
- MVI r,adat; MVI M, adat
  - Az utasítás 2. bájtja által képviselt adatot beírja az utasítás által megjelölt regiszterbe vagy a HL regiszterpár által kijelölt M memóriarekeszbe.
- LXI rp, adat 16
  - Hárombájtos utasítás. Hatására a 2. és 3. bájt tartalma beíródik az utasítás által kijelölt regiszterpár alsó-, illetve felső helyértékű 8-bites regiszterébe.
- LDA cím
  - Az utasítás 2. és 3. bájtja által megcímzett memóriarekesz tartalmát beírja az akkumulátorba.
- STA cím
  - Az akkumulátor tartalmát beírja az utasítás 2. és 3. bájtja által megcímzett memória-rekeszbe.
- LHLD cím
  - Az utasítás 2. és 3. bájtjával megcímzett memóriarekesz tartalmát beírja az L regiszterbe, az ezután következőét pedig a H regiszterbe.
- SHLD cím
  - Az utasítás 2. és 3. bájtjával megcímzett memóriarekeszbe beírja az L regiszter tartalmát és az ezután következőbe a H regiszterét.
- LDAX rp
  - A kijelölt regiszterpár tartalma által megcímzett memóriarekesz tartalmát beírja az akkumulátorba.
- STAX rp
  - A kijelölt regiszterpár tartalma által megcímzett memóriarekeszbe beírja az akkumulátor tartalmát.
- XCHG
  - A H és L, regiszterek tartalmát megcseréli a D, illetve E regiszterével.

Aritmetikai utasítások

Az aritmetikai utasítások a regiszterek valamint a memóriarekeszek adataival végeznek aritmetikai műveleteket. Ha nincs külön feltüntetve, ezek a műveletek a Z, S, P, CY és AC feltétel flip-flopok állapotát az eredménynek megfelelően változtatják meg. A kivonás a kettes komplemensű aritmetika szabályai szerint történik.

- ADD r; ADD M
  - A regiszter, illetve a HL regiszterpár által megcímzett memóriarekesz tartalmát az akkumulátoréhoz adja hozzá. Az eredményt az akkumulátorba helyezi.
- ADI adat
  - Az utasítás 2. bájtja által képviselt adatot az akkumulátorhoz adja hozzá. Az eredményt az akkumulátorba helyezi.

ADC r; ADC M

- A regiszter, illetve a HL regiszterpár által megcímzett memóriarekesz tartalmát, valamint a CY átvitel bitet az akkumulátoréhoz adja hozzá. Az eredményt az akkumulátorba helyezi.

- ACI adat
  - Az utasítás 2. bájtja által képviselt adatot, valamint a CY átvitel-bitet az akkumulátorhoz adja hozzá. Az eredményt az akkumulátorba helyezi.
- SUB r; SUB M; SUI adat
  - Az akkumulátorból az r regisztert, a HL regiszterpár által megcímzett memóriarekesz, tartalmát illetve az utasítás 2. bájtját vonja ki. Az eredményt az akkumulátorba helyezi.

- SBB r; SBB M; SBI adat
  – Az akkumulátorból az r regisztert, a HL regiszterpár által megcímzett memóriarekesz tartalmát illetve, az utasítás 2. bájtját, valamint a CY átvitelbitet is kivonja. Az eredményt az akkumulátorba helyezi.
- INR r; INR M
  – Az r regiszter, illetve a HL regiszterpár által megcímzett memóriarekesz tartalmát eggyel növeli (inkrementálja). Valamennyi feltételbitet a CY átvitel bit kivételével az eredménytől függően állítja be.
- DCR r; DCR M
  – Az r regiszter, illetve a HL regiszterpár által megcímzett memóriarekesz tartalmát eggyel csökkenti (dekrementálja). Valamennyi feltétel bitet a CY átvitelbit kivételével az eredménytől függően állítja be.
- INX rp; DCX rp
  – Az rp regiszterpárban levő 2-bájtos számot eggyel növeli, illetve csökkenti. A feltételbitek változatlanok maradnak.
- DAD rp
  – Az rp regiszterpár tartalmát a HL regiszterpáréval összegezi, és az eredményt a HL regiszterpárba helyezi. Csak a CY átvitelbiten változtathat.
- DAA
  – Decimális helyesbítést végző utasítás. Az akkumulátor tartalmát tétszámjegyű BCD kódolású számként tekinti, és a következő műveleketet hajtja végre:
  - ha a kisebb helyértékű BCD decimális szám nagyobb, mint 9, vagy ha AC = 1, akkor ehhez 6-ot ad hozzá;
  - ha a nagyobb helyértékű BCD decimális szám nagyobb, mint 9, vagy ha CY = 1, akkor ehhez 6-ot ad *hozzá;*
  Az összes feltételbit értéke megváltozhat.

Logikai utasítások

A logikai utasítások a regiszterek, a memóriarekeszek valamint a feltételregiszter bitjeivel végeznek logikai műveleteket. Általában valamennyi feltételbit a szokásos szabályok szerint változik meg. A kivételeket külön tüntetjük fel.

- ANA r; ANA M; ANI, adat
  – Az utasítás operandusa és az akkumulátor tartalma között bitenként logikai *ÉS* műveletet végez. Az eredményt az akkumulátorba helyezi. A művelet után CY = 0 és AC = 1.
- XRA r; XRA M; XRI adat
  – Az utasítás operandusa és az akkumulátor tartalma között bitenként *KIZÁRÓ-VAGY* műveletet végez. Az eredményt az akkumulátorba helyezi. A művelet után CY = 0 és AC = 0.
- ORA r; ORA M; ORI adat
  – Az utasítás operandusa és az akkumulátor tartalma között bitenként logikai *VAGY* műveletet végez. Az eredményt az akkumulátorba helyezi. A művelet után CY = 0 és AC = 0.
- CMP r; CMP M; CPI adat
  – Az operandust és az akkumulátort kivonás segítségével hasonlítja össze. Mindkettő változatlan marad. A feltétel-bitek a kivonás eredményének függvényében alakulnak. Ha az operandus egyenlő az akkumulátor tartalmával, akkor Z = 1. Ha az akkumulátor tartalma kisebb, mint az operandus, akkor CY = 1. Ellenkező esetben CY = 0.
- RLC; RRC; RAL; RAR
  – Ezek az utasítások az akkumulátor tartalmát léptetik egy bittel balra (RLC és RAL) vagy jobbra (RRC és RAR). RAL és RAR a CY-bitet az akkumulátor 9. bitjeként kezeli. A feltétel bitek közül kizárólag csak a CY-bit értékét változtathatják meg.
- CMA
  – Az akkumulátor tartalmát komplementálja. A feltétel bitek változatlanok maradnak.

- CMC
  - A CY átvitelbitet komplementálja. A többi feltételbit változatlan marad.
- STC
  - A művelet a CY-átvitelbitet 1-re állítja, míg a többi feltételbit nem változik.

A programszámláló tartalmát változtató utasítások

A programszámláló tartalmát változtató utasítások csoportjába az ugróutasítások, a szubrutinhívó és a szubrutin-visszatérési utasítások tartoznak. Lehetnek ***feltételesek*** és ***feltétel nélküliek***. A feltételesek az utasítás D_5 D_4 D_3 bitjei segítségével a következő feltételek teljesülését vizsgálják meg:

Feltétel jelölése	A feltételbit vizsgált értéke	Az utasításkód bitjei D_5 D_4 D_3
NZ – nem zéró	Z = 0	0 0 0
Z – zéró	Z = 1	0 0 1
NC – nincs átvitel	CY = 0	0 1 0
C – van átvitel	CY = 1	0 1 1
PO – páratlan	P = 0	1 0 0
PE – páros	P = 1	1 0 1
P – pozitív szám	S = 0	1 1 0
N – negatív szám	S = 1	1 1 1

- JMP cím
  - Feltétel nélküli ugróutasítás. A programszámláló az utasítás 2. és 3. bájtja által jelölt címre ugrik.
- J (feltétel) cím
  - Feltételes ugróutasítás. A programszámláló csak akkor ugrik az utasítás 2. és 3. bájtja által jelölt címre, ha a feltétel teljesül. Másként a soron következő utasítással folytatja.
- CALL cím
  - Feltétel nélküli szubrutinhívó utasítás. A veremmutató tartalma eggyel csökken. A veremmemóriának erre a címére beíródik a programszámláló felső helyértékű bájtja. Azután a veremmutató tartalma ismét csökken eggyel, és a veremmemóriába beíródik a programszámláló alsó helyértékű bájtja. Végül a programszámlálóba az utasítás 2. és 3. bájtja által tartalmazott cím kerül be.
- C (feltétel) cím
  - Feltételes szubrutinhívó utasítás. Az utasításban foglalt feltétel teljesülése esetén a szubrutinhívás végrehajtásra kerül. Másként a program a soron következő utasítással folytatódik.
- RET
  - Feltétel nélküli szubrutinból való visszatérési utasítás. A veremmutató által megcímzett veremmemória rekeszének tartalma beíródik a programszámláló alsó helyértékű bájtjába. Ezután a veremmutató tartalma eggyel nő, és erről a címről a programszámláló felső helyértékű bájtja íródik be. A veremmutató tartalma ismét nő eggyel.
- R (feltétel)
  - Feltételes szubrutin-visszatérési utasítás. Ha a feltétel teljesül, a szubrutinból a visszatérés bekövetkezik, ellenkező esetben a program a soron következő utasítással folytatódik.
- RST n
  - Speciális szubrutinhívó utasítás a programmemória 0. lapjának nyolc különböző helyére. A szubrutin kezdő címét az utasítás műveleti kódja adja meg.
- PCHL
  - A HL regiszterpár tartalma a programszámlálóba kerül. A program további végrehajtása ezen a címen folytatódik.

Veremtár és ki/beviteli műveletek valamint gépvezérlő utasítások

Ezek az utasítások a feltétel biteket nem változtatják meg. Egyedüli kivétel a POP PSW utasítás.

- PUSH rp; PUSH PSW
  - Az rp regiszterpár tartalmát, illetve a programstátusz szót beírja a veremtárba. A felső helyértékű regiszter, illetve az akkumulátor tartalma a veremtár azon rekeszébe íródik be, amelynek címe eggyel kisebb a veremmutatóban levő címnél. Az alsó helyértékű regiszter, illetve a feltétel regiszter tartalma a veremtár azon rekeszébe íródik be, amelynek címe kettővel kisebb a veremmutatóban levő címnél. Végül a veremmutató tartalma kettővel csökken. Az rp regiszterpár nem lehet a veremmutató.
- POP rp; POP PSW
  - A veremtár két egymás utáni rekeszének tartalmát beolvassa az utasítás által kijelölt regiszterpárba, illetve az A-F regiszterpárba. A veremmutató címe által kijelölt rekesz tartalma beíródik az alsó helyértékű regiszterbe, illetve a feltétel regiszterbe, az eggyel kisebb című rekesz tartalma pedig a felső helyértékű regiszterbe, illetve az akkumulátorba. A veremmutató tartalma kettővel növekszik. Az rp regiszterpár nem lehet a veremmutató.
- XTHL
  - A HL regiszterpárnak és a veremmutató által megcímzett rekesznek, valamint az utána következő rekesznek a tartalmát kicseréli. A veremmutató változatlan marad.
- SPHL
  - A HL regiszterpár tartalmát beírja a veremmutatóba.
- IN kapu
  - Beviteli utasítás. A bemeneti kapu tartalmát beírja az akkumulátorba. A bemeneti kapu 2-bájtos címét az utasítás 2. bájtjának megduplázásával képezi.
- OUT kapu
  - Kiviteli utasítás. Az akkumulátor tartalmát beírja a kimeneti kapuba. A kimeneti kapu 2-bájtos címét az utasítás 2. bájtjának megduplázásával képezi.
- EI
  - A megszakításkérést a következő utasítás végrehajtása után engedélyezi.
- DI
  - A megszakításkérés elfogadását közvetlenül az utasítás végrehajtása után letiltja.
- HLT
  - A processzor utasítás-végrehajtását leállítja. Egyik regiszter tartalma sem változik.
- NOP
  - A processzor egy gépi ciklus idejére semmit sem végez. A regiszterek tartalma változatlan marad.

A 8085-ös addicionális utasításai

- RIM
  - Az utasítás a SID bemenet állapotát, az RST 7.5, RST 6.5 és RST 5.5 megszakítások elvégzését valamint, az elfedési maszkjait írja be az akkumulátorba.
- SIM
  - Ez az utasítás az akkumulátor tartalmával a SOD kimenet állapotát, az RST 7.5, RST 6.5 és RST 5.5 megszakítások elfedési maszkjait állítja be.

Az Intel 8080/8085 mikroprocesszor utasításkészletét a jobb áttekinthetőség céljából a *8.6. táblázat* foglalja össze.

8.6. táblázat. A 8080/8085 mikroprocesszor utasításkészlete

Emlékeztető szimbólum	Az utasítás angol nyelvű elnevezése	Az utasítás kódja		Órajel-ciklusok	A művelet leírása
		Bináris $D_7D_6....D_1D_0$	Hexa		
1	2	3	4	5	6
	Adatmozgató utasítások				
MOV r_1, r_2	Move register to register	0 1 d d d s s s	*	4 (+1)	$(r_1) \leftarrow (r_2)$
MOV M,r	Move register to memory	0 1 1 1 0 s s s	*	7	(M) ← (r)
MOV r, M	Move memory to register	0 1 d d d 1 1 0	*	7	(r) ← (M)
MVI r	Move immediate register	0 0 d d d 1 1 0	*	7	(r) ← <B_2>
MVI M	Move immediate memory	0 0 1 1 0 1 1 0	36	10	(M) ← <B_2>
LXI B	Load immediate register pair B & C	0 0 0 0 0 0 0 1	01	10	(C) ← <B_2>; (B) ← <B_3>
LXI D	Load immediate register pair D & C	0 0 0 1 0 0 0 1	11	10	(E) ← <B_2>; (D) ← <B_3>
LXI H	Load immediate register pair H & L	0 0 1 0 0 0 0 1	21	10	(L) ← <B_2>; (H) ← <B_3>
LXI SP	Load immediate stack pointer	0 0 1 1 0 0 0 1	31	10	(SP) ← <B_3><B_2>
STAX B	Store A indirect	0 0 0 0 0 0 1 0	02	7	[(B)(C)] ← (A)
STAX D	Store A indirect	0 0 0 1 0 0 1 0	12	7	[(D)(E)] ← (A)
LDAX B	Load A indirect	0 0 0 0 1 0 1 0	0A	7	(A) ← [(B)(C)]
LDAX D	Load A indirect	0 0 0 1 1 0 1 0	1A	7	(A) ← [(D)(E)]
STA	Store A direct	0 0 1 1 0 0 1 0	32	13	[<B_3> < B_2>] ← (A)
LDA	Load A direct	0 0 1 1 1 0 1 0	3A	13	(A) ← [<B_3> < B_2>]
SHLD	Store H & L direct	0 0 1 0 0 0 1 0	22	16	[<B_3> <B_2>] ← (L); [<B_3> <B_2>+1] ← (H)
LHLD	Load H & L direct	0 0 1 0 1 0 1 0	2A	16	(L) ← [<B_3> <B_2>]; (H) ← [<B_3> <B_2>+1]
XCHG	Exchange D & E, H & L registers	1 1 1 0 1 0 1 1	EB	4	(H) ↔ (D); (L) ↔ (E)
	Zsákmemória - műveletek				
PUSH B	Push register pair B & C on stack	1 1 0 0 0 1 0 1	C5	12 (–1)	[SP – 1] ← (B), [SP – 2] ← (C); (SP) = (SP) – 2
PUSH D	Push register pair D & E on stack	1 1 0 1 0 1 0 1	D5	12 (–1)	[SP–1] ← (D), [SP-2] ← (E); (SP) = (SP) – 2
PUSH H	Push register pair H & L on stack	1 1 1 0 0 1 0 1	E5	12 (–1)	[SP – 1] ← (H), [SP – 2] ← (L); (SP) = (SP) – 2
PUSH PSW	Push A and Flags on stack	1 1 1 1 0 1 0 1	F5	12 (–1)	[SP – 1] ← (A), [SP – 2] ← (F); (SP) = (SP) – 2
POP B	Pop register pair B & C off stack	1 1 0 0 0 0 0 1	C1	10	(C) ← [SP], (B) ← [SP + 1]; (SP) = (SP) + 2
POP D	Pop register pair D & E off stack	1 1 0 1 0 0 0 1	D1	10	(E) ← [SP], (D) ← [SP + 1]; (SP) = (SP) + 2
POP H	Pop register pair H & L off stack	1 1 1 0 0 0 0 1	E1	10	(L) ← [SP], (D) ← [SP + 1]; (SP) = (SP) + 2
POP PSW	Pop A and Flags off stack	1 1 1 1 0 0 0 1	F1	10	(F) ← [SP], (A) ← [SP + 1]; (SP) = (SP) + 2
XTHL	Exchange top of stack, H & L	1 1 1 0 0 0 0 1	E3	16 (+2)	(L) ↔ [SP], (H) ↔ [SP + 1]
SPHL	H&L to stack pointer	1 1 1 1 1 0 0 1	F9	6 (–1)	(SP) ← (H)(L)

A 8.6. táblázat folytatása

1	2	3	4	5	6
	Ugróutasítások				
JMP	Jump unconditional	1 1 0 0 0 0 1 1	C3	10	(PC) ← <B_3><B_2>
JC	Jump to carry	1 1 0 1 1 0 1 0	DA	7/10 (+3/0)	Ha CY=1
JNC	Jump to no carry	1 1 0 1 0 0 1 0	D2	7/10 (+3/0)	Ha CY=0
JZ	Jump to zero	1 1 0 0 1 0 1 0	CA	7/10 (+3/0)	Ha Z =1
JNZ	Jump to no zero	1 1 0 0 0 0 1 0	C2	7/10 (+3/0)	Ha Z =0 ⇒ (PC) ← <B_3><B_2>; ha
JP	Jump to pozitive	1 1 1 1 0 0 1 0	F2	7/10 (+3/0)	Ha S =0 nem akkor (PC) ← (PC) + 3
JM	Jump to minus	1 1 1 1 1 0 1 0	FA	7/10 (+3/0)	Ha S =1
JPE	Jump to parity even	1 1 1 0 1 0 1 0	EA	7/10 (+3/0)	Ha P =1
JPO	Jump to parity odd	1 1 1 0 0 0 1 0	E2	7/10 (+3/0)	
PCHL	H & L to program counter	1 1 1 0 1 0 0 1	E9	6 (–1)	(PC) ← (H)(L)
	Szubrutinhívó utasítások				
CALL	Call unconditional	1 1 0 0 1 1 0 1	CD	18 (–1)	[SP – 1][SP – 2] ← (PC), (SP) ← (SP) + 2, (PC) ← <B_3><B_2>
CC	Call on carry	1 1 0 1 1 1 0 0	DC	9/18(+2/-1)	Ha CY =1
CNC	Call on no carry	1 1 0 1 0 1 0 0	D4	9/18(+2/-1)	Ha CY =0 ⇒ [SP-1][SP-2] ← (PC),
CZ	Call on zero	1 1 0 0 1 1 0 0	CC	9/18(+2/-1)	Ha Z =0 (SP) ← (SP)-2,
CNZ	Call on no zero	1 1 0 0 0 1 0 0	C4	9/18(+2/-1)	Ha S =1 (PC) ← <B_3><B_2>:
CP	Call on pozitive	1 1 1 1 0 1 0 0	F4	9/18(+2/-1)	Ha S =0
CM	Call on minus	1 1 1 1 1 1 0 0	FC	9/18(+2/-1)	Ha P =1 ha nem akkor
CPF	Call on parity even	1 1 1 0 1 1 0 0	EC	9/18(+2/-1)	Ha P =0 (PC) ← (PC) + 3
CPO	Call on parity odd	1 1 1 0 0 1 0 0	E4	9/18(+2/-1)	
	Visszatérési utasítások szubrutinból				
RET	Return	1 1 0 0 1 0 0 1	C9	10	(PC)←[SP][SP + 1]; (SP) ← (SP)+2
RC	Return on carry	1 1 0 1 1 0 0 0	D8	6/12(-1/-1)	
RNC	Return on no carry	1 1 0 1 0 0 0 0	D0	6/12(-1/-1)	Ha CY =1
RZ	Return on zero	1 1 0 0 1 0 0 0	C8	6/12(-1/-1)	Ha CY =0 ⇒ akkor
RNZ	Return on no zero	1 1 0 0 0 0 0 0	C0	6/12(-1/-1)	Ha Z =1 (PC) ← [SP][SP + 1],
RP	Return on pozitive	1 1 1 1 0 0 0 0	F0	6/12(-1/-1)	Ha Z =0 (SP) ← (SP) + 2
RM	Return on minus	1 1 1 1 1 0 0 0	F8	6/12(-1/-1)	Ha S =1
RPE	Return on parity even	1 1 1 0 1 0 0 0	E8	6/12(-1/-1)	Ha S =0 ha nem akkor
					Ha P =1 (PC) ← (PC) + 1
RPO	Return on parity odd	1 1 1 0 0 0 0 0	E0	6/12(-1/-1)	Ha P = 0
RST	Restart	1 1 n n n 1 1 1	*	12 (–1)	[SP – 1][SP – 2] ← (PC), (SP) ← (SP) – 2 PC = 00000000 00nnn000
	Ki / Beviteli utasítások				
IN	Input	1 1 0 1 1 0 1 1	DB	10	(A) ← (bemenő adat)
OUT	Output	1 1 0 1 0 0 1 1	D3	10	(kimenő adat) ← (A)
	Inkrementáló és Dekrementáló utasítások				
INR r	Increment register	0 0 d d d 1 0 0	*	4 (+1)	(r) ← (r) + 1
DCR r	Decrement register	0 0 d d d 1 0 1	*	4 (+1)	(r) ← (r) – 1
INR M	Increment memory	0 0 1 1 0 1 0 0	34	10	(M) ← (M) + 1
DCR M	Decrement memory	0 0 1 1 0 1 0 1	35	10	(M) ← (M) – 1
INX B	Increment B & C registers	0 0 0 0 0 0 1 1	03	6 (–1)	(B)(C) ← (B)(C) + 1
INX D	Increment D & E registers	0 0 0 1 0 0 1 1	13	6 (–1)	(D)(E) ← (D)(E) + 1

A 8.6. táblázat folytatása

1	2	3	4	5	6
INX H	Increment H & L registers	00100011	23	6 (–1)	(H)(L) ← (H)(L) + 1
INX SP	Increment stack pointer	00110011	33	6 (–1)	(SP) ← (SP) + 1
DCX B	Decrement B & C	00001011	0B	6 (–1)	(B)(C) ← (B)(C) – 1
DCX D	Decrement D & E	00011011	1B	6 (–1)	(D)(E) ← (D)(E) – 1
DCX H	Decrement H & L	00101011	2B	6 (–1)	(H)(L) ← (H)(L) – 1
DCX SP	Decrement stack pointer	00111011	3B	6 (–1)	(SP) ← (SP) – 1
	Összeadási utasítások				
ADD r	Add register to A	10000sss	*	4	(A) ← (A) + (r)
ADC r	Add register to A with carry	10001sss	*	4	(A) ← (A) + (r) + (CY)
ADD M	Add memory to A	10000110	86	7	(A) ← (A) + (M)
ADC M	Add memory to A with carry	10001110	8E	7	(A) ← (A) + (M) + (CY)
ADI	Add immediate to A	11000110	C6	7	(A) ← (A) + <B_2>
ACI	Add immediate to A with carry	11001110	CE	7	(A) ← (A) + <B_2> + (CY)
DAD B	Add B & C to H & L	00001001	09	10	(H)(L) ← (H)(L) + (B)(C)
DAD D	Add D & E to H & L	00011001	19	10	(H)(L) ← (H)(L) + (D)(E)
DAD H	Add H & L to H & L	00101001	29	10	(H)(L) ← (H)(L) + (H)(L)
DAD SP	Add stack pointer to H & L	00111001	39	10	(H)(L) ← (H)(L) + (SP)
	Kivonási utasítások				
SUB r	Subtract register from A	10010sss	*	4	(A) ← (A) – (r)
SBB r	Subtract register from A with borrow	10011sss	*	4	(A) ← (A) – (r) – (CY)
SUB M	Subtract memory from A	10010110	96	7	(A) ← (A) – (M)
SBB M	Subtract memory from A with borrow	10011110	9E	7	(A) ← (A) – (M) – (CY)
SUI	Subtract immediate from A	11010110	D6	7	(A) ← (A) – <B_2>
SBI	Subtract immediate from A with borrow	11011110	DE	7	(A) ← (A) – <B_2> – (CY)
	Logikai utasítások				
ANA r	And register with A	10100sss	*	4	(A) ← (A) ∧ (r)
XRA r	Exculsive Or register with A	10101sss	*	4	(A) ← (A) ∀ (r)
ORA r	Or register with A	10110sss	*	4	(A) ← (A) ∨ (r)
CMP r	Compare register with A	10111sss	*	4	(A) – (r); ha (A) = (r) akkor Z = 1, ha (A) < (r), akkor CY = 1
ANA M	And memory with A	10100110	A6	7	(A) ← (A) ∧ (M)
XRA M	Exclusive Or memory with A	10101110	AE	7	(A) ← (A) ∀ (M)
ORA M	Or memory with A	10110110	B6	7	(A) ← (A) ∨ (M)

A 8.6. táblázat folytatása

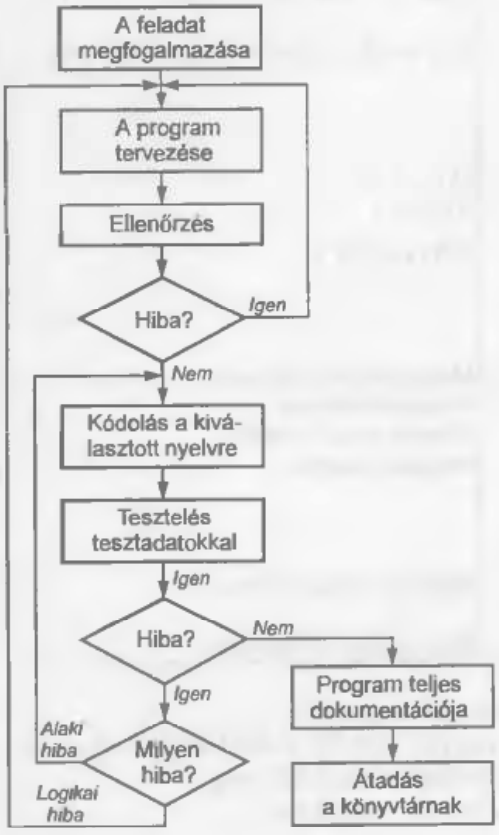
1	2	3	4	5	6
CMP M	Compare memory with A	10111110	BE	7	(A) – (M) ; ha (A) = (M), Z = 1 és ha (A) < (M); CY = 1
ANI	And immediate with A	11100110	E6	7	(A) ← (A) ∧ <B_2>
XRI	Exclusive Or immediate with A	11101110	EE	7	(A) ← (A) ∀ <B_2>
ORI	Or immediate with A	11110110	F6	7	(A) ← (A) ∨ <B_2>
CPI	Compare immediate with A	11111110	FE	7	(A) – <B_2>; ha (A) = <B_2>, Z = 1 és ha (A) < <B_2>, CY = 1
	Léptetési utasítások				
RLC	Rotate A left	00000111	07	4	(A_{m+1}) ← (A_m), (A_0) ← (A_7), (CY) ← (A_7)
RRC	Rotate A right	00001111	0F	4	(A_m) ← (A_{m+1}), (A_7) ← (A_0), (CY) ← (A_0)
RAL	Rotate A left through carry	00010111	17	4	(A_{m+1}) ← (A_m), (A_0) ← (CY), (CY) ← (A_7)
RAR	Rotate A right through carry	00011111	1F	4	(A_m) ← (A_{m+1}), (A_7) ← (CY), (CY) ← (A_0)
	Egyéb utasítások				
CMA	Complement A	00101111	2F	4	(A) ← ($\overline{A}$)
STC	Set carry	00110111	37	4	(CY) ← 1
CMC	Complement carry	00111111	3F	4	(CY) ← ($\overline{CY}$)
DAA	Decimal adjust A	00100111	27	4	
	Vezérlőutasítások				
EI	Enable Interrupts	11111011	FB	4	Megszakításengedélyezés
DI	Disable Interrupts	11110011	F3	4	Megszakításletiltás
NOP	No-opeartion	00000000	00	4	Művelet nélküli utasítás
HLT	Halt	01110110	76	5 (+2)	Megállító utasítás
	8085 utasítástöbblet				
RIM	Read Interrupt Mask	00100000	20	4	Megszakítási maszkolvasás
SIM	Set Interrupt Mask	00110000	30	4	Megszakítási maszkállítás

Megjegyzés: *Az órajelciklusok száma a 8085 mikoprocesszorra van megadva.*
A 8080 mikroprocesszor azon órajelciklusai, amelyek eltérőek a 8085-ösétől, az utóbbi órajelciklusaiból a zárójelben levő számok hozzáadásával kaphatók meg.
Az órajelciklusoknak a feltételes utasítások esetében két értékük van.

8.4. A mikroszámítógépek szoftverfejlesztése

Az előző részben a mikroszámítógépek hardverfelépítését mutattuk be. Ez a rész a mikroszámítógépek programozásával foglalkozik, amelyet ***szoftvernek*** (hard = kemény, soft = lágy) neveznek.

A szoftverfejlesztés fontosabb lépéseit ismertetjük röviden. Ennek logikai folyamatábráját a *8.25. ábra* mutatja. A szoftverfejlesztő munkája az adott feladat matematikai és logikai alakban való megfogalmazásával kezdődik. A következő lépés a program tervezése, amely főleg a feladat megoldás alkalmas algoritmus kiválasztásából áll. ***Algoritmuson*** az aritmetikai és logikai műveletek azon sorozatát értjük, amely lehetővé teszi a feladatnak a számítógép által való megoldását. A program tervének rögzítése a könnyebb áttekinthetőség céljából a *folyamatábrával* történik. Ez szabványos szimbólumokat használ, amelyeket a *8.26. ábra* mutat be.



8.25. ábra. A szoftverfejlesztés folyamatábrája

8.26. ábra. A folyamatábra szabványos szimbólumai (DIN 66 001 (MSZ 7784))

A program ellenőrzését az íróasztalnál végzik. A program menetét egy vagy több példán keresztül lépésenként megvizsgálják. Ha a vizsgálat eredménye megnyugtató, akkor a programot a számítógép számára érthető nyelvre kell kódolni.

A számítógépek programozása három különböző nyelven történhet: *gépi nyelven, gépre orientált nyelven* és *feladatra orientált nyelven*.

A gépi nyelven való programozás esetében a programozó az utasításokat a gépnek közvetlenül gépi kódban adja meg. Az utasítások kódjait, amelyeket a programmemória bináris alakban tárol, a kényelmesebb írásmód miatt általában hexadecimális vagy oktális formában adják meg. Előre meg kell határozni az utasítások, valamint az adatok címeit, vagyis a programban az utasításokra és az adatokra csak az abszolút címükkel lehet hivatkozni (lásd az ún. abszolút címzést!). A programmódosítások az összes cím megváltoztatását teszik szükségessé, ami rendkívül időigényes, és sok hiba forrása is lehet. A gépi nyelven való programozás azért is nehézkes és időigényes, mivel a programozó egy bonyolultabb műveletet csak sok egyszerű műveletet elvégző gépi utasítással hajthat végre, vagyis a gondolkozási módját gépi szintre kell áttennie. Ezért arra törekedtek, hogy a programokat a gépi nyelvtől függetlenül fejlesszék. Így a programok megfogalmazására különböző nyelveket alakítottak ki, amelyek lehetnek gépre és feladatra orientáltak.

A gépre orientált nyelvek közül az ***assembly*** nyelvet említjük meg, amely egy szimbolikus nyelv. Ennek az utasításai a hozzátartozó berendezés gépi nyelv utasításaival meggyezőek vagy ahhoz hasonló szerkezetűek. Az utasítás nevét könnyen megjegyezhető. módon rövidítik az ún. mnemotechnikai kódokkal. A mikroszámítógép programozása az assembleren keresztül történik. Az ***assembler*** egy fordítóprogram segítségével az assembly nyelven írt programokat gépi nyelvre fordítja. Az utasítások, valamint az adatok címei az assemblernek szimbolikus formában is megadhatók. A címet könnyen megjegyezhető módon betükből és számokból álló szimbolikus nevek, illetve ezekhez viszonyított relatív címek adják.

Végül megemlítjük a feladatra orientált nyelveket (más elnevezéssel a magas szintű nyelvek), amelyek lehetővé teszik a feladatnak a problémához igazodó megfogalmazását. Ezeket magas szintű nyelveknek is nevezik. A következő legismertebb magas szintű nyelveket említjük meg: FORTRAN, ALGOL, COBOL, BASIC, PASCAL, C, C++.

A FORTRAN (**For**mula **Tran**slation – képletfordítás) rendkívül gyakran alkalmazott nyelv, amely a képletírásra épül, és elsősorban műszaki-tudományos jellegű feladatok megoldására fejlesztették ki 1957-ben. Azóta is állandóan bővítik és fejlesztik. Az ALGOL (Algorithmic Language – algoritmikus nyelv) szintén műszaki és tudományos alkalmazásoknál használt, feladatra orientált nyelv, amely az algoritmusok leírásának egyszerűsítését teszi lehetővé. A COBOL (**C**ommon **B**usiness **O**riented **L**anguage – általános üzletvitelre orientált nyelv) nyelvet, mint elnevezése is mutatja, gazdasági ügyviteli célra fejlesztették ki. Minden állítása a kereskedelem területén használt stilizált angol szavakból származik. A BASIC (**B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode – kezdők általános szimbolikus utasításkódja) nagyon egyszerű programozási nyelv, amely hamar megtanulható. Egyszerűbb műszaki tudományos feladatok programozásához alakították ki. Az előnye különösen az időosztásos rendszereknél nyilvánul meg. A PASCAL nyelvet az ALGOL nyelv alapján fogalmazták meg, és egy korszerű nyelv követelményeinek tesz eleget. A mikroszámítógépeknél hamar tért hódított. A C és továbbfejlesztett változata a C++ egyike a leghasználtabb magas szintű programozási nyelveknek, amelyet 1962-ben fejlesztettek ki. A szakirodalom szerint ez a nyelv a jövőben is tekintélyes szerephez fog jutni.

A programozónak magas szintű nyelv használata esetén nem kell ismernie a gép egyéni nyelvét. A kompilátor a magas szintű nyelven megfogalmazott programot az adott gép gépi nyelvére fordítja le. A kompilátor működése sok tekintetben hasonlít az assembler működéséhez, de ennél sokkal bonyolultabb, mert a magas szintű nyelv szintaxisa sokkal kötetlenebb. A kompilátor a fordítást rendszerint több lépesőben hajtja végre, közbenső nyelvek felhasználásával, amelyek egyike gépre orientált programozási nyelv is lehet.

A magas szintű nyelv használata sok előnyt jelent az assembly nyelvhez képest. Legfontosabb a programok szerkesztéséhez szükséges idő körülbelül egy nagyságrenddel való csökkenése. Egy adott magas szintű nyelven írt program akármilyen típusú mikroprocesszorra épülő mikroszámítógépen futtatható, amely rendelkezik a megfelelő kompilátorral. A magas szintű nyelv használata egyedül a memória kihasználásának szempontjából hátrányos. Egy azonos feladatmegoldásra írt program magas szintű nyelven körülbelül kétszer annyi memóriahelyet igényel, mint az assembly nyelven megírt program. Ezért a magas szintű nyelvek használata csak a nagyobb gépeknél gazdaságos.

A szoftverfejlesztés következő lépése a program tesztelése, amely során a hibákat kiküszöbölik. Kétféle hiba lehetséges: *alaki* és *logikai hiba*. Az alaki hibát az adott programozási nyelv formai jellegű szabályainak a megsértése okozza. A legtöbb alaki hiba a program fordításakor kiderül. A logikai hiba a program helytelen felépítéséből származik. A program próbafuttatását megfelelő tesztadatokkal kell végezni. Ezeket úgy kell megválasztani, hogy az eredményt a program futtatása nélkül könnyen ki lehessen értékelni, és minden programrész kipróbálását lehetővé tegyék. Minden hiba kijavítása után újabb próbafuttatást kell végezni.

A programokat a programkönyvtárban összefoglalva rendszerezik. A programnak, mielőtt a könyvtárba kerülne, el kell készíteni a teljes dokumentációját. Ez a szoftverfejlesztés utolsó lépésének tekinthető.

Programozás a 8080/8085 mikroprocesszor assembly nyelvén

A közepes bonyolultságú alkalmazásoknál az assembly nyelv bizonyul a leghatékonyabbnak. Az assembly nyelven való programozás elsajátítása elősegíti a magas szintű nyelvek megtanulását.

A mikroprocesszor assembly nyelvén írt programot az ***assembler*** (az assembly nyelv fordítóprogramja) fordítja le gépi nyelvre megkímélve a hosszadalmas és sok hibalehetőséggel járó munkától.

Az assembly nyelvben az utasításokat és a címeket szimbólumokkal jelölik. Ez megkíméli a programozót attól, hogy megjegyezze az utasítások gépi kódjait, valamint a címek abszolút értékeit. Egy szimbólum – az assemblertől függően – négytől hatig terjedő, különböző karakterkombinációból állhat, amelynek kötelezően az első karaktere betű. Az assembly nyelv állításai a következő négy mezőben csoportosulnak: *címke, utasítás, operandus vagy cím* és *jelentés*.

A mezők elhatárolása a köztük meghagyott egy vagy több üres karakterrel vagy különböző lezáró karakterrel (vessző, pont, pontosvessző, kettőspont, választóvonal, dőltvonal) történik. A címke az utasítást vagy az adatot tároló memóriahely szimbolikus címét jelöli ki. Ebből kifolyólag a programban akárhol meg lehet címezni a címke által kijelölt utasítást vagy adatot. A programnak nem minden egyes utasítása van címkével ellátva, rendszerint csak azok, amelyek végrehajtása ugróutasítás után következhetik.

Az utasítások emlékeztető (mnemotechnikai) szimbólumai a következő mezőben kapnak helyet. Az utasítást követi az operandus vagy a cím. Az utolsó mezőbe az utasítások jelentése és a megjegyzések kerülnek. Ezeket az assembler fordítóprogramja elhanyagolja, kizárólag csak a program dokumentálását szolgálják.

Az assembly nyelven írt programot ***forrásprogramnak*** (source program) is nevezik. A gépi nyelvre fordított forrásprogramot pedig ***tárgy***- vagy ***célprogramnak*** is nevezik.

Az assemblerek lehetővé teszik az úgynevezett makroutasítások alkalmazását. A makroutasítás egy olyan assembly nyelven írt utasítássorozat, amely a programban gyakran fordul elő.

Az assembler fontos szerepet játszik a forrásprogramban levő hibák felfedezésében és kijavításában is. Informálja a felhasználót a hiba helyéről és típusáról hibaüzenet formájában, vagy a kiírt programlistát a megfelelő helyeken hibakódokkal látja el.

A következőkben egy pár jellegzetes programot mutatunk be, amelyek a 8080/8085 mikroprocesszor assembly nyelvén készültek. A programok teljes megértéséhez szükséges a *8.6. táblázatban* összefoglalt utasításkészlet is.

Elágazás nélküli programok

Az elágazás nélküli vagy más néven lineáris programok jellemzője, hogy a program utasításai a tárban tárolt sorrendben hajtódnak végre. Egy utasítás végrehajtása után az utasításszámláló értéke eggyel megnő, és a következő címen levő utasítás kerül végrehajtásra. Ez a program utolsó utasításának végrehajtásáig folytatódik.

☞ **1. példa:** 8-bites összeadó-rutin

A 0040 és 0041 memóriarekeszekben tárolt 8-bites számokat adja össze. Az eredményt a 0042 címen tárolja. A program egy „végtelen hurokkal" zárul, amelyből RESET segítségével lehet kilépni.

A forrásprogram:

```
        LXI     H, 40H
        MOV     A, M        Az első operandus beolvasása.
        INX     H
        ADD     M           A második operandus hozzáadása.
        INX     H
        MOV     M, A        Az eredmény tárolása.
VHUR:   JMP     VHUR
```

A tárgyprogram:

Címke:	*Cím*	*Utasításkód*	*Utasítás-szimbólum*
	0000	21	LXI H, 40H
	0001	40	
	0002	00	
	0003	7E	MOV A, M
	0004	23	INX H
	0005	86	ADD M
	0006	23	INX H
	0007	77	MOV M, A
VHUR:	0008	C3	JMP VHUR
	0009	08	
	000A	00	

☞ **2. példa:** a nagyobbik szám kiválasztása

A 0040 és 0041 memóriarekeszekben levő 8-bites előjel nélküli számok körül a nagyobbikat a 0042 című rekeszben tárolja.

A forrásprogram:

```
        LXI     H, 40H
        MOV     A, M        Az első operandus beolvasása.
        INX     H
        CMP     M           Nagyobb-e a második operandus?
        JNC     NOPT
        MOV     A, M        Igen, a második operandus beolvasása.
NOPT:   INX     H
        MOV     M, A        A nagyobbik operandus tárolása.
VHUR:   JMP     VHUR
```

A tárgyprogram:

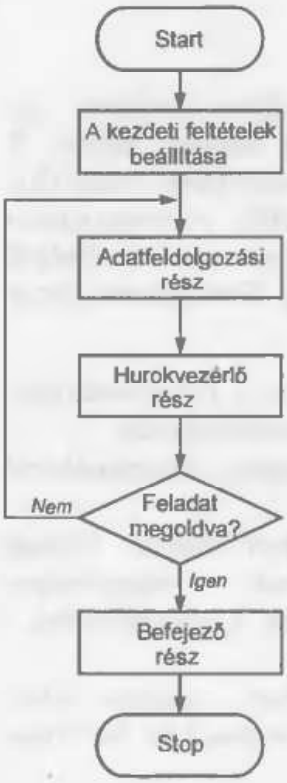
Címke:	*Cím*	*Utasításkód*	*Utasítás-szimbólum*
	0000	21	LXI H, 40H
	0001	40	
	0002	00	
	0003	7E	MOV A, M
	0004	23	INX H
	0005	BE	CMP M
	0006	D2	JNC NOPT
	0007	0A	
	0008	00	
	0009	7E	MOV A, M
NOPT:	000A	23	INX H
	000B	77	MOV M, A
VHUR:	000C	C3	JMP VHUR
	000D	0C	
	000E	00	

Egyszerű programhurkok

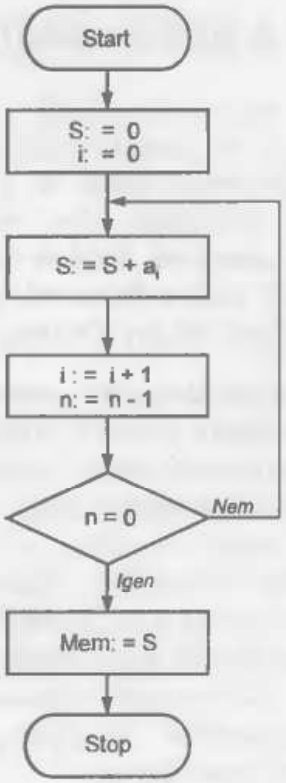
A programhurok egy olyan utasításlánc, amelyet a feladat megoldása érdekében egymásután többször futtatunk. Egymást követő ismétlések során az előző ismétlés eredményei képezik a következő ismétlés kiinduló adatait. A programhurok négy elkülöníthető részből áll:

- a *kezdeti feltételek beállítása*, amelyen a számlálók, a címregiszterek és az egyéb változók kezdeti értékének a megadását értjük;
- az *adatfeldolgozási rész*, amelynek többszöri ismétlése a probléma megoldásához vezet;
- a *hurokvezérlő rész*, amely a számlálók és a címregiszterek tartalmát a következő iteráció számára állítja be;
- *a program befejező része*, amely az eredményt elemzi és tárolja.

A számítógép a hurok első és utolsó részén csak egyszer fut át, míg a másodikon és harmadikon többször is. A hurok előállítható feltétel nélküli, vagy elágazásos döntéssel összekapcsolva feltételes ugrással. A *8.27. ábra* a programhurok általánosított folyamatábráját szemlélteti. Az alábbiakban néhány egyszerűbb programhurkot mutatunk be.



8.27. ábra. A programhurok általánosított folyamatábrája



8.28. ábra. Folyamatábra egy számsorozat összegének kiszámítására

☞ **Példa:** Egy számsorozat összegének a kiszámítása

Az a_1, a_2, ..., a_i, ..., a_n számsorozat első tagja a 0042 című memóriarekeszben, míg a sorozat következő tagjai a következő rekeszekben találhatók; *n*-et, a sorozat tagjainak számát a 0041 című rekeszben kell megadni. Az *S* eredményt a program a 0040 címre tárolja. A sorozat tagjait úgy kell megválasztani, hogy az összességük ne haladja meg a 8-bites formátumot. A program folyamatábrája a *8.28. ábrán* látható.

A forrásprogram:

```
        LXI     H, 41H
        MOV     B, M        n beírása a B regiszterbe.
        XRA     A           Az akkumulátor nullázása.
SUMA:   INX     H           A sorozat a_i tagjainak címe.
        ADD     M           Az összeg képzése az akkumulátorban.
        DCR     B           A sorozat tagjainak számlálása.
        JNZ     SUMA        Ha n > 0, az összeadás folytatása.
        STA     40H         Ha n = 0, az eredmény tárolása.
VHUR:   JMP     VHUR
```

8.5. A mikroszámítógépek alkalmazása

A mikroprocesszorok és a mikroszámítógépek alkalmazási területe rendkívül tág. Jelenleg ott tartunk, hogy az alkalmazási lehetőségeknek nincsenek műszaki határai. A mikroprocesszoroknak az a jelentősége, hogy a számítógépekkel megoldható feladatokat sokkal olcsóbban lehet velük megoldani, és ezáltal lehetőség nyílik a programozott célrendszereknek újabb és újabb területeken való alkalmazására. Az alkalmazási lehetőségek korlátait csak a felhasználó hozzáértése és képzelőereje szabja meg. Szemléltetésképpen felsorolunk néhány alkalmazási lehetőséget:

- *adatfeldolgozás:* zsebszámítógép, személyi számítógép, munkahelyi miniszámítógép, intelligens terminál, programozási segédeszközök, decentralizált adatfeldolgozás;
- *híradástechnika:* távmásoló, elektronikus telefonközpont, képújság, kommunikáció műholdak segítségével;
- *mérés-, vezérlés- és szabályozástechnika:* az iparban – ipari adatok, üzemek folyamatvezérlése, automatikus mérlegek; az orvostudományban – számítógépes tomográfia, EEG és EKG feldolgozása, pszichológiai tesztrendszerek; a közlekedésben – forgalomtól függő közlekedési lámpavezérlés;
- *szórakoztató elektronika:* programozható elektronikus játékok, sakkautomaták, magnetofon- és képmagnetofon-vezérlés, hangszintetizátorok, automatikus beállítású fényképezőgépek;
- *háztartási elektronika:* automatikus mosógépek és mosogató gépek, fűtés és fényszabályozás, varrógépek, háztartási számítógépek;
- *járműelektronika:* üzemanyag-fogyasztást optimalizáló motorvezérlés, a kerekek leblokkolását gátló fékrendszer (ABS), automatikus sebességváltó, összeütközések elkerülését elősegítő radar, fedélzeti számítógépek a repülőgépeken, hajókon valamint az űrhajókon.

A mikroszámítógépeknek az adatfeldolgozásban való alkalmazásával a számítástechnikai munka decentralizálható. Lehetővé válik, hogy az adatelőkészítés és előfeldolgozás már a munkahelyen megtörténjen, ahol az „adatok keletkeznek". A mikroprocesszorok egyik legismertebb alkalmazási példája *a személyi számítógép.* A határok a professzionális szintű alkalmazásoktól a gazdasági területeken át a számítógépes játékokig terjednek.

A híradástechnika sem zárkózhat el a digitális elektronika által nyújtott sokrétű lehetőségektől. A televíziókészülék a telefonnal és új szolgáltatásokkal együtt átfogó információs rendszerré fejlődik. Az előfizetők – ha kívánják – aktív (párbeszédes) módon kapcsolódhatnak az adásokba.

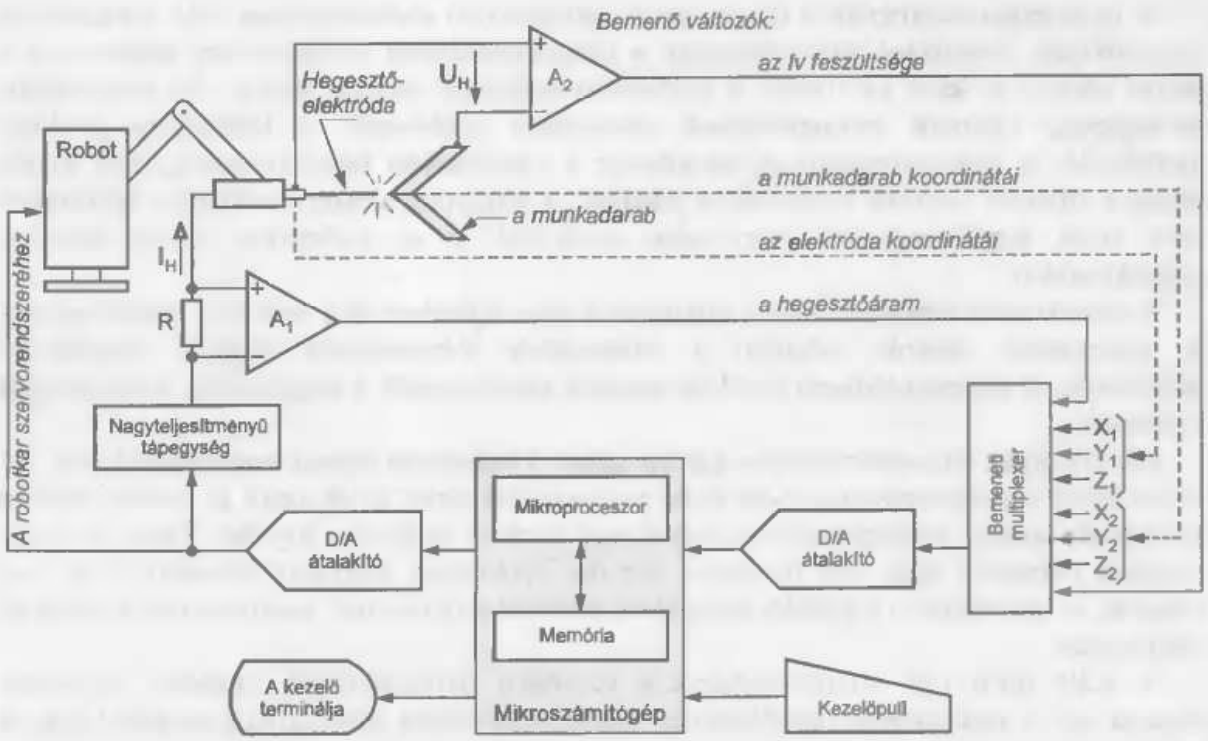
A mérés-, vezérlés- és szabályozástechnika a mikroprocesszorok és mikroszámítógépek legfontosabb alkalmazási területének egyike. Elsősorban *a robottechnikát* említjük meg, amely a műszaki élet egyik legdivatosabb területe. Maga a „robot" szó 1920-ból, a kiváló cseh írótól, Karel Čapektől származik. A köznyelv robotnak nevez minden önállóan működő műszaki rendszert, a legegyszerűbb automatától a korszerű elektronikus irányítórendszerig. A műszaki értelmezés legáltalánosabban elfogadott változata szerint *a robot* olyan eszköz, mely különböző, kézzel végezhető (manipulációs) vagy- helyzetváltoztatási feladatok automatikus elvégzésére beprogramozható. Míg e meghatározás szerint a robotok tevékenységi körébe mind az alsó, mind a felső végtagok műveleteinek egy része beletartozik, az *ipari robotok* tevékenységi köre a felső végtagok tevékenységi körére korlátozódik. A robotok egyik alapvető része *a manipulátor,* más néven *a robotkar.*

A robotkar mozgása programozható, és vele a programban rögzített művelet számtalanszor megismételhető.

Az első generációbeli robotoknak nem volt semmiféle kapcsolatuk a külvilággal, mert nem voltak érzékelőik. Ez különösen a munkadarabnak az elhelyezésekor és rögzítésekor okozott gondot. Az évtizedünkben megjelent második generációbeli robotok érzékelőik révén már kommunikálhatnak a környezetükkel. Az érzékelőktől érkező jeleket az általában több processzoros rendszerek dolgozzák fel. Így a robot képes „látni", „hallani" és „tapintani" is. A műveleteket, a mindenkori körülményeket figyelembe véve hajtja végre, bizonyos mértékben alkalmazkodva a környezet változásaihoz. A kifejlesztés alatt levő harmadik generációbeli robotok bonyolult, összetett feladatok ellátására is képesek. Ezek környezetükből érzékelőikkel összegyűjtik az információkat, logikájukkal feldolgozzák azokat, és – ez a legfőbb jellemzőjük – önálló döntéseket hoznak. Így bonyolult folyamatokban cselekvően vehetnek részt, vagyis alapjaiban alakíthatják át az üzemek termelési rendjét.

A robot az egyhangú, a veszélyes és a nehéz fizikai munka alól felszabadítja az embert. Az ipari termékek választéka csak úgy bővíthető, ha egy-egy terméket kisebb sorozatban és gazdaságosan gyárthatnak. Ez robotokkal könnyen megvalósítható, hiszen a robot rövid idő alatt és egyszerűen átállítható az új termék gyártására. Egyébként a robot nagyon termelékeny eszköz: három műszakban is működhet, és munkaidejének 95%-ban tevékeny.

A *8.29. ábra* egy ipari hegesztőrobotot mutat be vázlatosan. A robot a hegesztő elektródával követi a munkadarab összehegesztendő vonalát. Ezért a mikroprocesszor a bemeneti multiplexeren és az analóg-digitális átalakítón keresztül az elektróda X_1, Y_1 és Z_1 és a munkadarab X_2, Y_2 és Z_2 térbeli koordinátáját kapja. A jó hegesztés fontos követelménye az elektróda és a hegesztendő felület között levő pontos távolság betartása. Ezt a hegesztőáram pontos értéken való tartásával lehet biztosítani.



8.29. ábra. Ipari hegesztőrobot vázlatos felépítése

A nagy teljesítményű hegesztő tápegység árama az R ellenálláson feszültségesést okoz, amely az A_1 erősítőn, a bemeneti multiplexeren és az analóg-digitális átalakítón keresztül a mikroprocesszorhoz kerül. A mikroprocesszor feldolgozza a bemenő információt, és a kimeneti digitális-analóg átalakítón keresztül vezérli a hegesztő elektródát tartó robotkart és a nagy teljesítményű hegesztő tápegységet.

A mikroprocesszorok és a mikroszámítógépek az orvostudományban is fontos szerephez jutottak. A legtipikusabb alkalmazásuk a számítógépes tomográfia, amely lehetővé teszi az emberi test belső felépítését ábrázoló keresztmetszetek készítését. A transzmissziós tomográfia esetében egy nagy erejű röntgensugárnyaláb metszi a testet, amelynek „árnyékát" a sugárforrással szemben elhelyezett érzékelők fogadják. Ez azonban nem más, mint az egymás mögött levő szervek és szövetek képe. A léptető automata ezért egy bizonyos szöggel elfordítja a sugárnyalábot és az érzékelőt. Így újabb információk jutnak be a számítógépbe. Miután a sugárnyaláb lépésenként elvégezte a 180°-os elfordulást, a számítógép feldolgozza az így kapott adatokat, és meghatározza a keresztmetszeti képek elemeit. Megemlíthetjük az emissziós számítógépes tomográfiát is. Ebben az esetben a beteg szervezetébe juttatott radioaktív izotópból emittáló sugárzást használják fel a keresztmetszet feltérképezésére. A számítógépes tomográfia a keresztmetszet különböző fedettségű részeit nem a szürke árnyalataival különbözteti meg, hanem a színképskála különböző árnyalataival. Így a keresztmetszet testrészei jobban láthatóvá válnak, és ez megkönnyíti a diagnosztizálást.

A mikroszámítógépek másik jelentős alkalmazása az orvostudományban és az egészségügyben a *betegfelügyelet*. A mikroszámítógép állandóan figyelemmel tartja a beteg szervezetének különböző paramétereit, például a lázát, pulzusát, vérnyomását stb. Ha valamelyik vagy az egymáshoz viszonyulásuk eltér a normálistól, a gép azonnal riaszt.

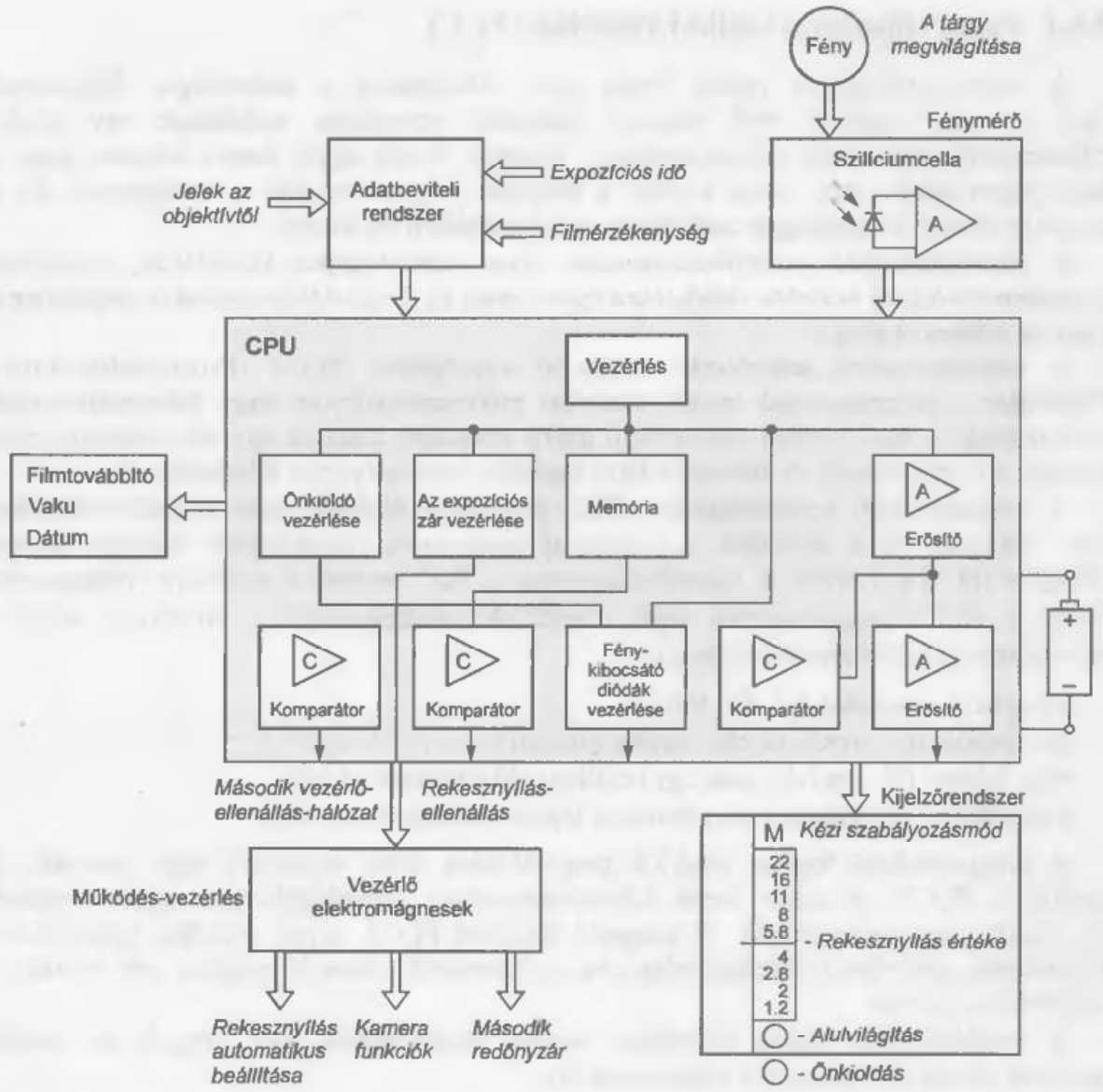
A gépjárműveknél a klasszikus autovillamosságról már áttértek a mikroprocesszoros automatikára. A megnövekedett biztonság és az utazási komfort mellett a járműgyártás olyan céljai is megvalósíthatók, mint az optimális energiafelhasználás és a kisebb szennyezőanyag kibocsátás.

A mikroprocesszoroknak a háztartási és szórakoztató elektronikában való alkalmazása igen sokrétű. Ízelítőként megemlíthetjük a programvezérlésű varrógépeket, amelyeknél a varrás közben is lehet változtatni a különböző bonyolult varrásmintákat. Az automatikus mosógépek, valamint mosogatógépek lényegesen csökkentik a háztartásra fordított munkaidőt. A mikroprocesszorok jelentősége a háztartásban hasonló ahhoz, amit annak idején a villamos motorok megjelenése jelentett. A mikroprocesszoros háztartási készülékek azon kívül, hogy magasabb színvonalon szolgálják ki az embereket, sokkal energiatakarékosabbak.

A szórakoztató elektronikában a mikroprocesszoros játékok és a személyi számítógépekre szerkesztett játékok, valamint a multimédiás alkalmazások nagyon elterjedtnek tekinthetők. A kikapcsolódáson kívül az emberek önkéntelenül is megtanulják a számítógép kezelését.

Létezik olyan mikroprocesszoros karóra, amely a kimondott szónak engedelmeskedik. Az órába épített mikroprocesszor 95-98 %-os pontossággal ismer fel és alakít át vezérlő jelekké különböző szóbeli parancsokat, amelyeket egy parányi mikrofon fog fel. Téves reagálás esetén a parancsot meg kell ismételni. Az óra viszonylag könnyen felismeri, hogy hol hibázott, és hozzászokva a gazdája hangjához, a következőkben már pontosan hajtja végre az utasításokat.

A *8.30. ábra* egy mikroprocesszoros vezérlésű fényképezőgép vázlatos felépítését mutatja be. A mechanikus vezérlőelemek legnagyobb részét elektronikus megfelelőjükkel helyettesítették.



8.30. ábra. Mikroprocesszoros vezérlésű fényképezőgép tömbvázlata

A fényképezőgép elektronikus részének alapját egy CMOS technológiával készült mikroprocesszor képezi. A filmre kerülő fény erősségét egy szilícium fotodióda méri. A fotodióda jelét egy logaritmikus átviteli karakterisztikájú BI-MOS műveleti erősítő erősíti fel és küldi a mikroprocesszornak. A használt film érzékenysége és a kívánt expozíciós idő másik két paraméter, amelyek segítségével a mikroprocesszor kiszámítja a helyes rekesznyílást. Ezt a rekesznyílás nagyságát meghatározó speciális ellenállás-hálózat és a rekesznyílást változtató elektromágnesek segítségével állítja be automatikusan. Az így beállított érték a keresőn keresztül, a fényképezendő tárgy mellett leolvasható. Ha az expozíciós időt túlságosan rövidre állítottuk, úgy hogy a rekesznyílás kívánt értéke meghaladja az objektív fényerejét, a mikroprocesszor másik fontos szerepe, hogy a beállított expozíciós idő szerint működteti a redőnyzárat. Végül a fényképezőgépre szerelhető automatikus filmtovábbító vaku és dátumrögzítő működésének vezérlését is biztosítja.

Az előbbiekben felsorolt példák csak közelítőleg érzékeltetik a mikroprocesszorok és a velük megépített mikroszámítógépek által beindított hatalmas innovációs hullámot.

8.5.1. Programozható logikai vezérlők (PLC)

A mikroszámítógépek másik fontos ipari alkalmazása a technológiai folyamatok vezérlése. Az üzemben levő minden fontosabb berendezés működését egy kisebb teljesítményű specializált mikroszámítógép irányítja. Ennek egyik fontos feladata, hogy a feldolgozott adatok egy részét közölje a központi folyamatirányító számítógéppel. Ez a nagyteljesítményű számítógép csak bizonyos helyzetekben lép közbe.

A tárprogramozott vezérlőberendezések olyan számítógépes készülékek, amelyeket programozható ipari vezérlési feladatokra fejlesztettek ki. Teljesítőképességüket alapvetően a szoftver határozza meg.

A tárprogramozott vezérlések feldolgozó egységeként PLC-t (**P**rogramable **L**ogic **C**ontroller = programozható logikai vezérlőt) mikroszámítógépet vagy folyamatterminált alkalmaznak. A PLC-k olyan elektronikus üzemi eszközök, amelyek egy alkalmazásorientált nyelven programozhatók és képesek ellátni vezérlési és szabályozási feladatokat is.

A programozható vezérlőlogikát a PLC programtárolójában, mint vezérlőutasításokat lehet megadni. Ezek felváltják a kapcsolatprogramozott vezérléseknél használt logikai elemeket (pl. *ÉS, VAGY*). A vezérlőutasításokat a PLC értelmező szoftvere (interpretere) dolgozza fel. A programozható logikai vezérlők utasításkészlete a következő vezérlőutasításokat mindenképpen tartalmazza:

- logikai kapcsolatok (pl. *ÉS, VAGY*),
- számlálás (pl. impulzusszám meghatározása),
- késleltetés (pl. egy jelet csak egy beállított idő elteltével ad ki),
- tárolás (pl. egy kimenet vagy bemenet logikai értékének tárolása).

A programozható logikai vezérlők megvalósítása lehet *moduláris* vagy *kompakt*. A moduláris PLC-k az adott üzemi követelményekhez illeszthetők. Az egyes modulok bővíthetők vagy kicserélhetők. A kompakt kiépítésű PLC-k olyan vezérlési feladatokhoz alkalmasak, amelyeknél előreláthatólag a be- és kimenetek száma lényegesen nem változik a későbbiek folyamán.

A vezérlőlogikát három különböző módon határozhatjuk meg (vagyis az eszköz programozására három nyelvet fejlesztettek ki):

- utasításlista (**St**atement **l**ist – STL),
- létradiagram (**L**adder **d**iagram – LAD),
- funkcionális terv (**C**ontrol **s**ystem **f**lowchart – CSF)

Az egyes gyártók különböző eszközöket bocsátanak rendelkezésre a PLC-hez a vezérlőprogram bevitelére. Ezek lehetővé teszik a vezérlőprogram megadásának legalább az egyik módját a három közül. A jelenleg legelterjedtebb eszköz a programozás és programfejlesztés megvalósítására a megfelelő szoftverrel ellátott számítógép (PC – Personal Computer). A gyártók gyakran használnak egyedi vezérlőutasításokat a programok (utasításlisták, létradiagramok vagy funkcionális tervek) beviteléhez.

A *8.31. ábra* egy PLC tipikus felépítését mutatja be. A főbb részegységek a következők:

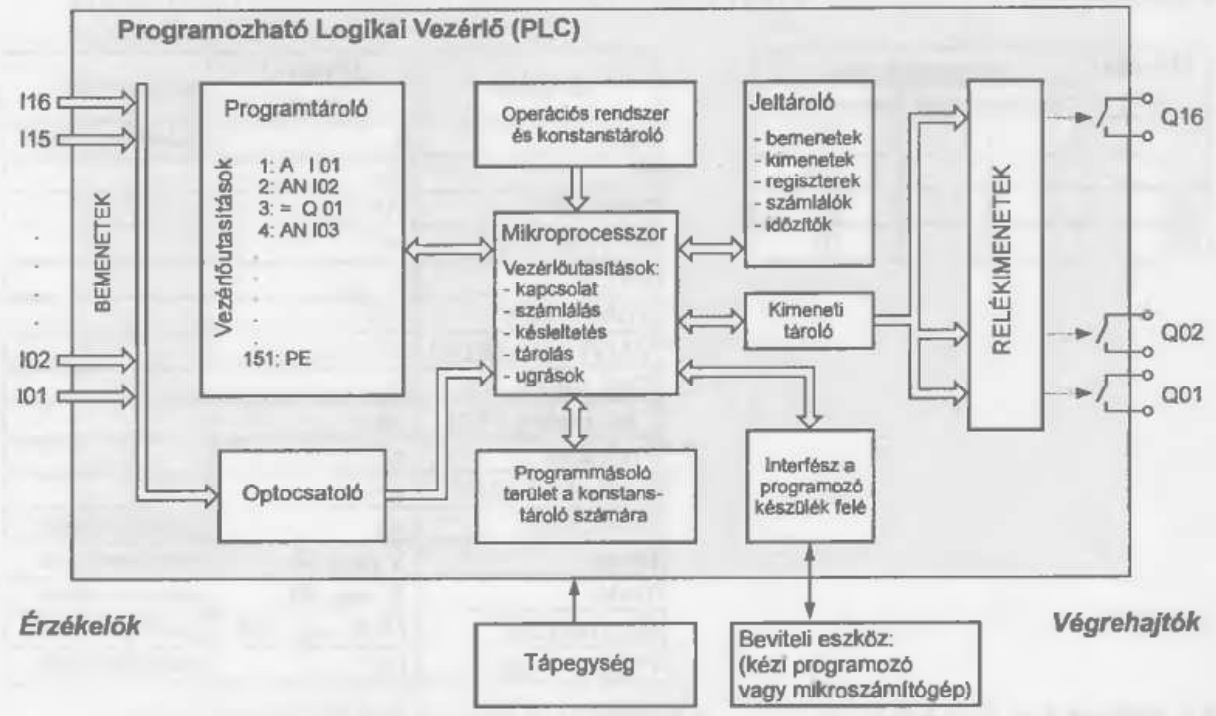
- feldolgozóegység:
  - a mikroprocesszorral, a konstanstárolóval, a jeltárolóval, a programtárolóval;
- bemeneti interfész;
- kimeneti interfész;
- kimeneti interfész a programozókészülék felé;
- hálózati csatlakozás.

A bemenetek – galvanikus leválasztást biztosító – optocsatolón keresztül kapcsolódnak a mikroprocesszor bemeneti interfészéhez. Ezen keresztül valósul meg az érzékelők jelszintjének illesztése a PLC üzemi feszültségéhez. A relékimenetek látják el ebben az esetben a teljesítmény interfész feladatát. Meg kell jegyezni, hogy a PLC-k kimeneti- és bemeneti interfésze – kiépítéstől függően – képes lehet analóg és digitális jelek kibocsátására ill. fogadására is. Ebben az esetben az analóg bemeneti jelek feldolgozását A/D átalakítók végzik. A mikroprocesszor digitális vezérlőjeleit D/A átalakítók alakítják át analóg kimenő jellé.

Az utasításlistát (vezérlőlogikát) a beviteli egység (más néven programozókészülék) az interfészen keresztül juttatja el a PLC programtárolójába. A konstanstárolóban levő speciális operációs rendszer vezérli úgy a mikroprocesszort, hogy a kapott adatok a programtárolóba kerüljenek. Ez az operációs rendszer döntően meghatározza a PLC teljesítőképességét.

A vezérlőprogram indításakor a bemenetek aktuális logikai állapota eltárolásra kerül (ezeket az értékeket veszi figyelembe a vezérlőlogika a program végrehajtása során) és az utasításszámláló az első utasítássorra áll (az ábrán az 1 sor). A mikroprocesszor kiolvassa az első utasítássort a programtárolóból és az operációs rendszer segítségével, végrehajtja azt. Az "**A I01**" parancs hatására az **I01** bemenet – előzőleg már letárolt – logikai állapota betöltődik, és a jeltárolóba kerül. Ezt követi a következő utasítássor végrehajtása ("**AN I02**"). Az **I02** bemenet logikai állapota is a jeltárolóba kerül. A második sorban lévő "**A**" utasítás hatására az operációs rendszer a két bemenet eltárolt állapotát logikai *ÉS*-kapcsolatba hozza.

Az **I01** és **I02** bemenet *ÉS*-kapcsolatának eredményét (amely logikai **1** vagy **0** értéket vehet fel) szintén eltárolja a mikroprocesszor. Ezután a következő utasítássor kerül végrehajtásra ("**= Q01**"). Az utasítás a logikai *ÉS*-kapcsolat eredményét a **Q01** kimenethez rendeli hozzá. Az eredmény betöltődik a jeltárolóba. Közben a kimeneti tároló kiadja a jelet (logikai 0 vagy 1) a **Q01** kimenet számára. Ha az **I01** és **I02** bemenet logikai **1**-es szintet kap az érzékelőktől az **Q01** kimeneten levő relé behúz, vagyis a terhelési oldal áramköre záródik. Ezzel a programtárolóban levő első "logikai mondat" végrehajtása befejeződik.



8.31. ábra. A PLC alapvető felépítése

A mikroprocesszor folytatja a negyedik, ötödik, stb. sorban levő utasítás végrehajtását mindaddig, amíg a program végére nem ér. A processzor számára a program végének elérését a **PE** (**P**rogram **E**nd) utasítás jelzi. Az utasításszámláló ismét a program elejére áll vissza. A PLC elkezdi ismét az utasításlista soronkénti végrehajtását. A **Q01** kimeneten addig áll fenn a logikai **1**-es jel, amíg az **I01** vagy **I02** bemenet bármelyikén logikai **1**-es szint van

Megállapítható, hogy a PLC működését a program határozza meg. A program nem más, mint utasítások sorozata, amelyeket folyamatosan egymás után (szekvenciálisan) hajt végre a mikroprocesszor.

8.5.1.1. A PLC programozása

A nemzetközi szabványok nem rögzítik a programozható logikai vezérlők utasításkészletének sem a legkisebb sem a legnagyobb terjedelmét. A különböző PLC-gyártók (pl. Siemens, AEG, Festo, Omron, Mitsubishi) utasításkészletei ezért gyakran eltérnek egymástól. Az adott PLC kézikönyvében a gyártók megadják a megfelelő programozási előírásokat.

Programozás utasításlista (STL) segítségével

Az utasításlistával történő programozás esetén a működési egyenleteket előírt formában soronként kell megadni. Az utasításlista tulajdonképpen a vezérlőutasítások egy sorozata. A *8.7. táblázatban* látható egy utasítássor felépítése. Az **01**-el és **02**-vel jelölt bemenet (**I**) *ÉS*-kapcsolata (**A**) valamint *ÉS-NEM*-kapcsolata valósul meg a példában.

A különböző programozható logikai vezérlők utasításkészlete a *8.8. táblázatban* feltüntetett parancsokat (műveleteket) mindenképpen tartalmazza. A régebbi gyártású (általában 1993 előtti) PLC-k esetében egy utasításlista sornak mindenképpen az **L** (töltés) paranccsal kell kezdődnie és a **PE** (program vége) paranccsal kell végződnie. Az újabb fejlesztésű PLC-k esetében az első sorban levő **A** (*ÉS*) utasítás automatikus programbetöltést eredményez.

Műveleti rész	operandus rész	
	operandus jel	*paraméter*
A	I	01
A	I	02
AN	I	01
AN	I	02

8.7. táblázat. Egy utasításlista-sor formai megjelenése

Jelentés	Műveleti rész	Megjegyzés
töltés	L	gyártóspecifikus
ÉS	A	
NEM-ÉS	AN	
ÉS nyitózárójel	A(	
VAGY	O	
NEM-VAGY	ON	
VAGY nyitózárójel	O(	
Záró zárójel	)	
Záró zárójel NEM	)N	
Értékadás	=	
Értékadás NEM	=N	
Idő-bemenet	=T	gyártóspecifikus
Beírás	S vagy SL	gyártóspecifikus
Törlés	R vagy RL	gyártóspecifikus
Üres művelet	NOP vagy NO	gyártóspecifikus
Program vége	PE	gyártóspecifikus

8.8. táblázat. A műveleti rész és annak jelentése

A PLC gyártók a dokumentációban meghatározzák, hogy melyik parancs szükséges az indításhoz.

A be- és kimenetek mellett a műveletek operandusaként szóba jöhetnek a különböző regiszterek, számlálók és időzítők.

A ***regiszterek*** itt 1-bites tárolók, amelyeket arra használnak, hogy a logikai kapcsolatok kiértékelésének eredményeit eltárolják a következő utasítások számára.

Időzítők felhasználásával a folyamatirányító berendezés időterv-vezérlésekre is alkalmassá válik. Ha a beállított idő letelt, az időzítő logikai 1-es jelet ad ki.

Számlálók felhasználásával különböző impulzusok számlálása valósítható meg. Ha a számláló bemenetére kerülő impulzusszám túllépi a beállított értéket a számláló kimenetén logikai 1-es jel jelenik meg.

Jelentés	Operandus-jelölések
bemenet	I (pl. I01-tól I16-ig)
kimenet	Q (pl Q01-től Q16-ig)
regiszter	M (M01-től M256-ig)
időzítő	T (T01-től T1024-ig)
számláló	C (C01-tól C256-ig)

8.9. táblázat. Operandusjelölések és azok jelentései

Ahogy a *8.7. táblázatban* is megfigyelhető volt, minden operandus-jelet egy megfelelő paraméternek kell meghatároznia. A paraméter adja meg a bemenet, a kimenet, a regiszter vagy az időzítő sorszámát. A rendelkezésre álló paraméterek számát a PLC típusa és kiépítettsége (pl. a bemeneti- és kimeneti modulok száma) határozza meg. A *8.9. táblázat* az operandusok tipikus jelölését és jelentését szemlélteti.

Programozás létradiagram (LD) segítségével

A létradiagram szerkezete egy adott vezérlés áram-út tervének elvét követi. Az áramút-tervtől abban különbözik, hogy az áramutakat vízszintesen rendezi, és más szimbólumokat használ, amelyek a számítógépes ábrázolásban könnyen és szemléletesen megvalósíthatók. Ezzel a létradiagramokat egyszerűen lehet megjeleníteni a képernyőn vagy a nyomtatón. A *8.10. táblázat* a létradiagramok tipikus szimbólumait mutatja be.

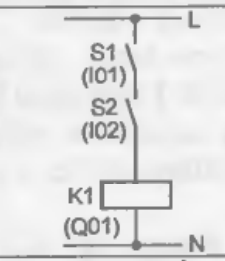
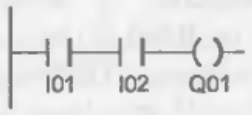
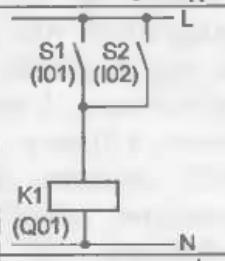
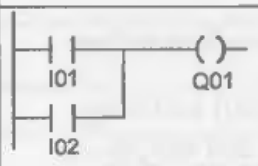
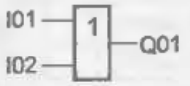
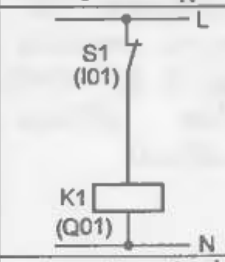
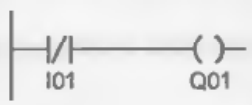
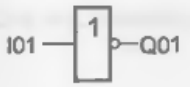
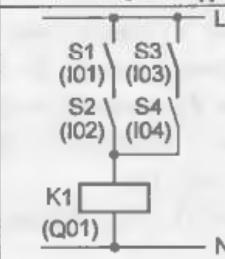
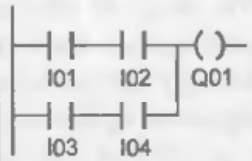
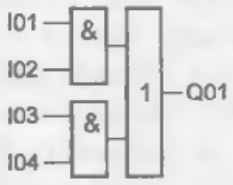
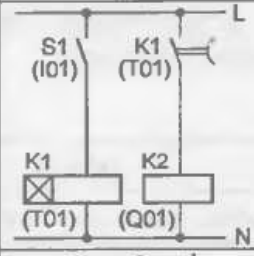
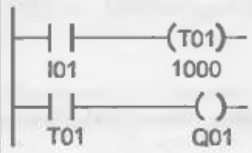
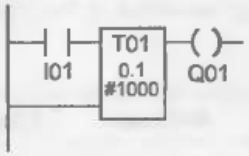
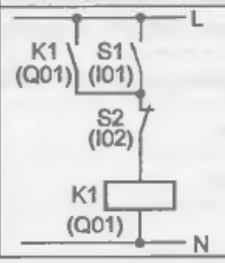
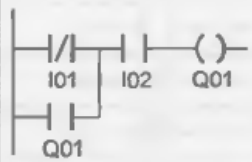
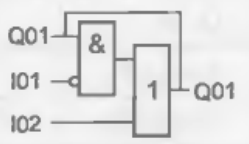
A *funkcionális terv* alapjaiban a logikai tervvel egyezik meg. Ezzel terjedelmi okokból részletesen itt nem foglalkozunk.

A programozható logikai vezérlők tehát utasításlistával, létradiagrammal vagy funkcionális tervvel programozhatók. A három programozási mód tulajdonképpen ugyanannak a vezérlő-programnak a különböző meghatározási módja.

Művelet	*Létradiagram-szimbólumok*	*Jelentés*
L, A, O	—| |—	Alapállapotban nyitott (NO) érintkező (bemenet)
AN, ON	—|/|—	Alapállapotban zárt (NC) érintkező (bemenet)
Q, M, T	—| |—	Alapállapotban nyitott (NO) érintkező (bemenet)
S vagy SL	—|S|—	Beíró bemenet
R vagy RL	—|R|—	Törlő bemenet
=	—( )—	Kimenet

8.10. táblázat. Létradiagramok szimbólumai

A *8.11. táblázat* az *ÉS*, *VAGY* és *NEM* logikai müveletek és további példák különböző technológiai megvalósítását mutatja be.

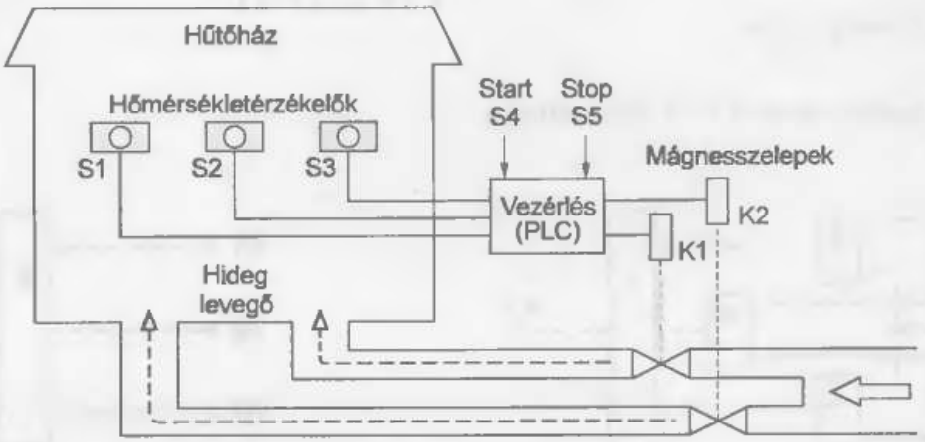
Jelölés Működési egyenlet	Elektromechanikus megvalósítás	PLC		
		STL	LAD	CSF
ÉS $I01 * I02 = Q01$		A I01 A I02 = Q01		
VAGY $I01 + I02 = Q01$		A I01 O I02 = Q01		
NEM $\overline{I01} = Q01$		AN I01 = Q01		
Programozás zárójelekkel $(I01 * I02) +$ $+ (I01 * I02) = Q01$		A(A I01 A I02) O(A I03 A I04) = Q01		
Késleltetett bekapcsolás (a késleltetés 1000*100 ms)		A I01 = T01 AN T01 = Q01		
Egy kimenet öntartása		A I01 O Q01 AN I02 = Q01		

8.11. táblázat. Tipikus programozási modulok

Megoldott feladat:

Egy hűtőházban (*8.32. ábra*) a **K1** mágnesszelep csak akkor engedi a hideg levegő bevezetését, ha az **S1, S2, S3** hőmérsékletérzékelők közül legalább kettő a beállított hőmérsékleti határérték túllépését jelzi. Ha a három érzékelő együtt jelez, akkor egy második **K2** mágnesszelepen további hideg levegő áramolhat be.

A vezérlés öntartó bekapcsolását az **S4** kézi nyomógomb (záró), kikapcsolását pedig az **S5** kézi nyomógomb (bontó) kell, hogy biztosítsa. A hűtőház vezérlését programozható logikai vezérlő (*PLC*) alkalmazásával szeretnénk biztosítani.



8.32. ábra. A hűtőház vezérlésének egyszerűsített tömbvázlata

a) Tervezze meg a vezérlés binárisan kódolt vezérlőtábláját, bekapcsolt állapotot feltételezve!

b) Rajzolja meg - az a) pontban kapott vezérlőtáblát felhasználva - a megfelelő logikai tervet!

c) Készítse el a vezérlés hozzárendelési listáját! A programozható logikai vezérlő felhasználható elemei és jelölései a hozzárendelési lista készítésénél a következők:

- I 01 ... I 16 bemenetek;
- Q 01 ... Q 16 kimenetek;
- M 01 ... M 16 regiszterek.

d) Valósítsa meg a vezérlés működését biztosító vezérlőprogramot létradiagramban!

e) Készítse el a vezérlés működését biztosító vezérlőprogramot utasításlistás formában!

Megoldás:

a) A vezérlés vezérlőtábláját a *8.12. táblázat* szemlélteti:

S1	S2	S3	K1	K2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	**1**	0
1	0	0	0	0
1	0	1	**1**	0
1	1	0	**1**	0
1	1	1	**1**	**1**

8.12. táblázat. Vezérlőtábla

b) A K1 logikai függvény Karnaugh-tábláját (a megfelelő tömbösítéssel) a *8.33. ábra* szemlélteti.

S1 \ S2S3	00	01	11	10
0	0	0	1	0
1	0	1	1	1

8.33. ábra. Karnaugh-tábla

Az egyszerüsített függvény, vagyis a K1 vezérlését biztosító müködési egyenlet:

$$K1 = S2 \cdot S3 + S1 \cdot S3 + S1 \cdot S2 .$$

A K2 működési egyenlete:

$$K2 = S1 \cdot S2 \cdot S3 .$$

A vezérlés logikai tervét a *8.34. ábra* mutatja.

8.34. ábra. A vezérlés logikai terve

c) A hozzárendelési listákat a *8.35. ábra* mutatja.

Érzékelő	S1	S2	S3	S4 (Start)	S5 (Stop)
PLC bemenet	I 01	I 02	I 03	I 04	I 05

a) bemenetek hozzárendelési listája

Végrehajtószerv	K1	K2
PLC kimenet	Q 01	Q 02

b) kimenetek hozzárendelési listája

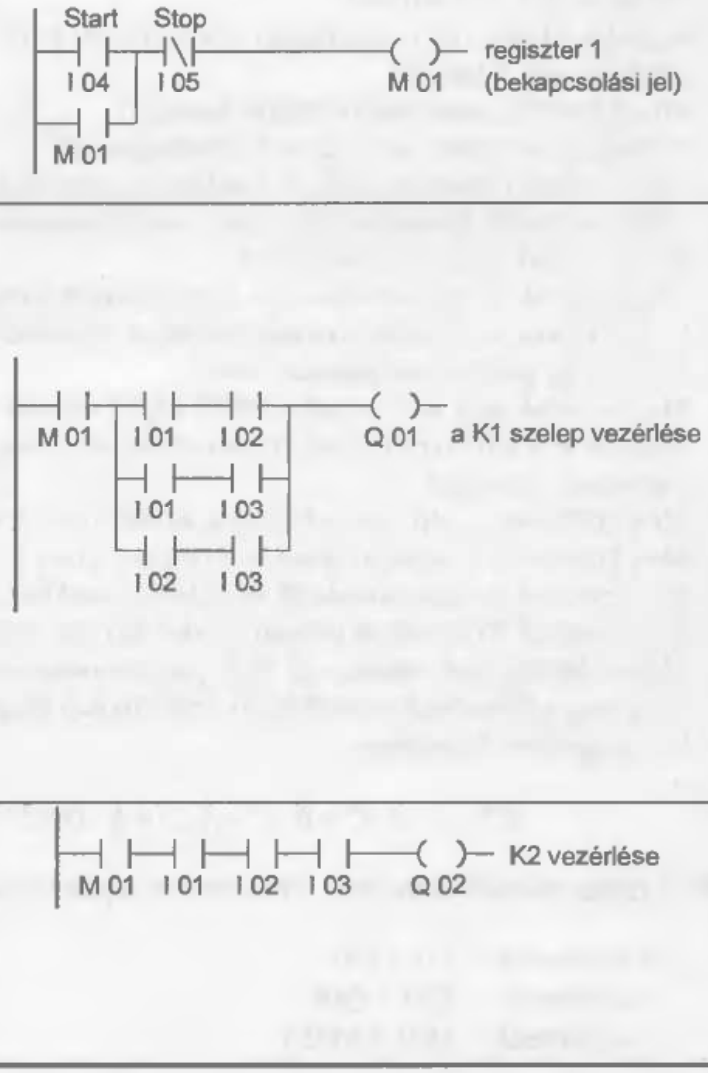
8.35. ábra. Hozzárendelési lista

d), e)
A vezérlés működését biztosító vezérlőprogramot utasításlistában és létradiagramban a *8.36. ábra* mutatja.

Sor	Utasítás	Operandus
1:	A	I 04
2:	O	M 01
3:	AN	I 05
4:	=	M 01
5:	A	M 01
6:	A(	
7:	A	I 01
8:	A	I 02
9:	)	
10:	O(	
11:	A	I 01
12:	A	I 03
13:	)	
14:	O(	
15:	A	I 02
16:	A	I 03
17:	)	
18:	=	Q 01
19:	A	M 01
20:	A	I 01
21:	A	I 02
22:	A	I 03
23:	=	Q 02

Utasításlista

Létradiagram



8.36. ábra. A vezérlőprogram

Összefoglaló kérdések és feladatok:

1. Milyen alapegységei vannak a Neumann-féle univerzális számítógépnek?
2. Milyen feladatokat lát el az aritmetikai logikai egység és hogyan működik együtt a vezérlőegységgel?
3. Mi a szerepe a programszámlálónak?
4. Magyarázzuk el a veremtár elvi működését!
5. Mit nevezünk szubrutinnak?
6. Rajzolja fel egy mikroszámítógép egyszerüsített tömbvázlatát és magyarázza el a részegységek feladatát!
7. Mi a közvetlen memóriahozzáférés lényege?
8. Milyen jelző-biteket tartalmaz a feltételregiszter?
9. Milyen belső regiszterei vannak a mikroprocesszornak?
10. Milyen alapvető fázisokra tagolódik a mikroprocesszor utasításciklusa?
11. Mit nevezünk virtuális címzésnek?
12. Magyarázzuk meg a müveleti rész és az operandus rész fogalmakat?
13. Hogyan csoportosíthatók a mikroprocesszor utasításai?
14. Írjuk le egy programmegszakítás elvét!
15. Magyarázzuk meg mi történik a **MOV r1, r2** utasítás hatására!
16. Rajzolja le a szoftverfejlesztés folyamatábráját és magyarázza el a folyamatábra egyes lépéseinek lényegét!
17. Mi a különbség a gépi nyelv és magas szintű nyelv között?
18. Mi a különbség a forrásprogram és a tárgyprogram között?
19. Mit nevezünk programhuroknak és milyen részekből áll?
20. Mit nevezünk PLC-nek és milyen feladatokat tud ellátni egy PLC?
21. Milyen lehetőségek vannak egy PLC programozására?
22. Adja meg a következő vezérlőfüggvényt (logikai függvényt) utasításlistás és létradiagramos formában:

$$F^4 = A \cdot B \cdot C + \overline{B} \cdot C + A \cdot D + B \cdot \overline{D} \cdot \overline{C}$$

A PLC felhasználható bemenetei, kimenetei és lépéstárolói (regiszterei) a következők:

- bemenetek: I01 ÷ I16
- kimenetek: Q01 ÷ Q08
- regiszterek: M01 ÷ M256

Ára: 750 Ft

Kovács Csongor

DIGITÁLIS ELEKTRONIKA