Digitális elektronika

Összeállította: Bagoly Zsolt és Varga Dezső

Wed Jun 19 16:21:30 CEST 2013

1.1.1. Ha Boole-algebra
1.1.3. Nem?, áram logikai értékre zárás alatt?

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$

A Boole algebra a két logikai érték közötti ÉS (AND) kapcsolatot a szorzás analógiájaként kezeli:

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

A negálás vagy tagadás, azaz a NEM művelete egy számot az ellentétre változtat:

$$\bar{0} = 1$$
$$\bar{1} = 0$$

A Boole összeadás (VAGY művelet) és szorzás (ÉS művelet) alapszabályai ellenőrizhető módon a következőek:

$$A + 0 = A$$
$$A + 1 = 1$$
$$A + A = A$$
$$A + \bar{A} = 1$$
$$A \cdot 0 = 0$$
$$A \cdot 1 = A$$
$$A \cdot A = A$$
$$A \cdot \bar{A} = 0$$

A Boole algebrában a szorzás és összeadás kommutatív és asszociatív, valamint az összeg szorzására a disztributivitás szabálya érvényes:

1.1.3. Digitális szabványok

A logikai funkcióknak számos áramköri megvalósítása van, összességükben ezeket fogjuk digitális rendszereknek nevezni. A fizikai megvalósítás valamilyen tényleges feszültségtartományhoz fog logikai 1-es és 0-t rendelni, ami alapján szabványos áramköri családok jöttek létre. Ezek a családok az áramkörök illesztése (a feszültségszintek), a gyártástechnológia, a fogyasztás és a különböző eltérő fejlesztési irányok következtében kaptak létjogosultságot.

A leggyakoribb áramköri családok a következők:

- TTL: Transistor-Transistor-Logic, bipoláris tranzisztorokra épül. A 90-es évekig szinte egyeduralkodó volt, de mint szabványt most is elterjedten használják.
- CMOS (Complementer MOS): komplementer MOS-FET tranzisztorokból van felépítve, a kapuk (l. XXX fejezet) teljesítmény igénye kicsi, sebességük az elmúlt évtizedekben jelentősen meghaladta a klasszikus TTL rendszerekét.
- ECL (Emitter-Coupled-Logic): nagysebességű kapuk, jelentős fogyasztással. A külső zajoknak jól ellenáll mert differenciális erősítőket alkalmaz.
- NIM

A 1.1 táblázat összefoglalja a leggyakoribb típusok paramétereit. Egyik fontos szempont a tápfeszültség, másik pedig a késleltetés. Ez utóbbi azt jelenti, hogy egy fizikailag megvalósított alkatrész esetén véges időt kell várni amíg a kimeneten megjelenik a bemenet által meghatározott feszültség.

típus	késleltetés (ns)	teljesítmény (mW)	tápfeszültség.
CMOS	2-200	1	3.3-12
TTL	5-10	10	5 (4.75-5.25)
ECL	1-2	25-60	-5.2

1.1. táblázat. Logikai áramkörcsaládok összehasonlítása.

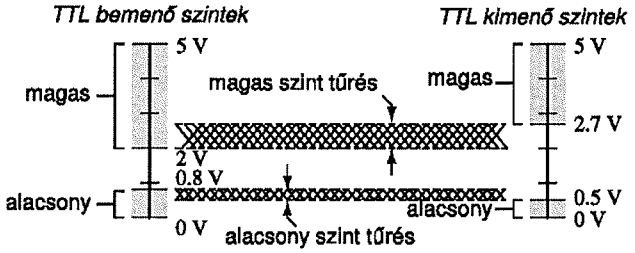
1.1.4. Logikai jelszintek

A logikai áramkör-rendszerek alapelemei az inverterek, illetve a kapcsolódó ÉS és VAGY áramkörök. Ezekek kapuknak is szokás hívni. A XXX fejezetben már példát láttunk egy inverter transzfer karakterisztikájára. Azt is láttuk, hogy egy kapu nem-lineáris erősítőként működik. A feszültségtartományokhoz hozzárendelt digitális szintek (pl. +5V→ 1, 0V→0) csak az egyszerűsítést szolgálják.

A következő ábrán emlékeztetőként egy CMOS inverter kapcsolási rajzát láthatjuk:

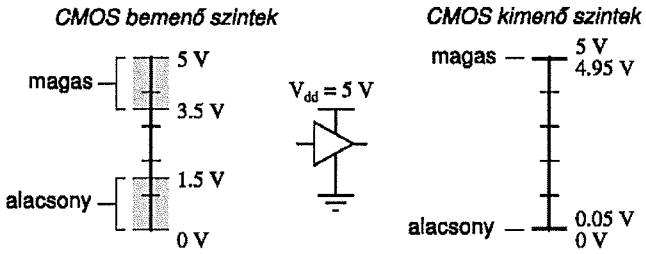
véges szélességű feszültségtartomány felel meg. Az tehát hogy egy-egy kapu kimenetén vagy bemenetén éppen mekkora a feszültség, az irreleváns, ami számít, hogy benne legyen a 0-nak vagy 1-nek megfelelő *tartományban.* Egy másik apróság szintén fontos: ha egy logikai rendszerben adottak a bemenetek által értelmezett logikai tartományok, akkor a kimenetek által kiadott, 0 vagy 1 logikai szintnek megfelelő feszültség mindig a bemeneti tartományokon belül van. Ez azért fontos hogy egy apró áramköri hiba vagy külső zaj miatt ne csúszhasson a kimenet éppen a következő bemenet által értelmezhető tartományon kívülre. Ez a fajta tűréstartomány is a szabványok része.

A TTL kapuk esetén a szintek 0-(5 ± 0.25)V között vannak:



Látható, hogy a bemeneten 0.8V alatti jel biztosan 0, a 2V feletti pedig biztosan 1 értékűnek lesz értelmezve. A kimenetek által adott feszültségtartományok viszont ennél szűkebbek.

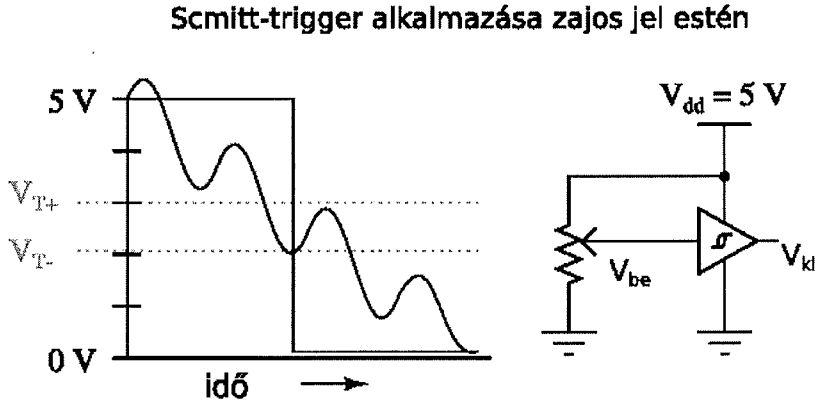
A CMOS kapuk a TTL kapuknál szélesebb feszültségtartományban képesek működni. Egy 5V-os tápfeszültségről működő CMOS kapu a következő szinteket használja:



A kimenet a TTL kapunál sokkal kisebb toleranciával működik, a bemenet viszont csak 3,5V-tól tekinti 1-nek a jelet. A bemeneti szintek különbözősége miatt erre ügyelni kell a TTL-CMOS áramkörök összekapcsolásánál.

A CMOS kapukat általában $15V$ tápfeszültségről is üzemeltethetjük. Ez természetesen eltolja a szinteket, sokkal érzéketlenebbé teszi a zajra (láthatóan még 4 V-os eltérés esetén is jól működik a kapu):

A megoldás erre a XXX fejezetben megismert Schmitt trigger alkalmazása a bemeneten:



Láthatjuk, hogy a hiszterézisnek köszönhetően megnőtt a zajjal szembeni immunitás. A Schmitt trigger bemenetű kapukat a fenti kapura is rárajzolt hiszterézisre emlékeztető szimbólummal jelölik.

1.1.6. Háromállapotú kimenet és buszvonal

A digitális áramkörök kimeneteit nem szabad összekötni, hiszen ha éppen ellentétes állapotú lenne a két kimenet, akkor nincs szabvány szerinti útmutatás hogy melyik legyen a domináns: a kimenet ilyenkor hibás működésű lesz.

Fontos kivétel ez alól a szabály alól az ún. tri-state (három állapotú) kimenet: vagy logikai 1, vagy logikai 0 szint van rajta, vagy pedig határozatlan de nagyon nagy kimenő ellenállásúvá (high-Z) válik. Ez utóbbi azt jelenti hogy egy másik kimenet meghibásodás nélkül tetszőleges (0 vagy 1) logikai szintet kényszeríthet a vezetékre. A high-Z állapot engedélyezését (enable) vagy tiltását egy külső logikai bemenet vezérli. Tri-state kimenetek összekötésekor továbbra is teljesülnie kell annak, hogy egyetlen kimenet lehet ami nem éppen high-Z állapotú.

A tri-state kimenetek esetén a kimenet ellenütemű tranzisztorait egy külön vezérlővezetékkel szabályozott tranzisztorok zárják le, a kimenet ekkor szabadon lebeghet.

A vezérlés természetesen lehet negált is:

rajzon csak a kapu szimbolumát tüntetjük fel, esetleg a kapcsolódó IC lábainak megjelölésével.

A kapuk logikai szinteket dolgoznak fel: a feszültségeket az áramköri nulla ponthoz (föld) mérjük, ez közös minden kapunál. Természetesen minden kapunak tápfeszültséget kell adni (ezt nem szokták feltüntetni a kapcsolási rajzon, ahogy ezt a műveleti erősítőknél is láttuk).

A kapuk működése - ugyanúgy, mint a logikai függvényeknél - a bemenetek és kimenetek közötti igazságtáblázattal írható le. Látni fogjuk, hogy az egyszerű logikai függvények mellett találhatunk memóriával bíró eszközöket is: ezeknél az igazságtáblázat a korábbi állapotokai is tartalmazhatja.

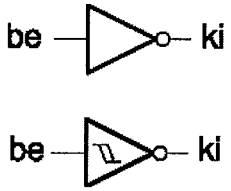
Az invertertáló (NOT) és a puffer kapu

A puffer (buffer) kapu egyszerűen lemásolja a bemenetet a kimenetre, ilyen módon a gyakorlatban is nagyon nagy számú kaput meghajthatunk egy kimenetről. A másik előny, hogy pl. egy buszt puffer kapun keresztül egy csatlakozóra vezetve az esetleges rövidzár, stb. nem okoz meghibásodást a busz működésében.

A puffer áramköri jele a következő:

be	ki
0	0
1	1

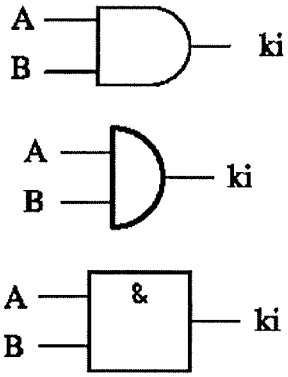
Az inverter az XXX fejezetben megismert áramkörhöz hasonlóan a bemeneti logikai szint ellentettjét adja ki a kimenetén (a hiszterézisre emlékeztető szimbólum Schmitt trigger bemenetű kaput jelöl):



be	ki
0	1
1	0

Fontos megjegyezni, hogy a *kis karika mindig az invertálás jele* lesz a továbbiakban.

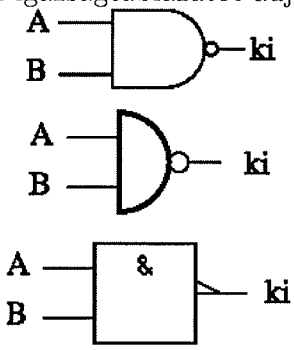
Alternatív jelölésként a bemenetre is tehetik az invertálást jelentő karikát - jóllehet jelen jegyzetben igyekszünk ezt elkerülni:



A	B	ki
0	0	0
0	1	0
1	0	0
1	1	1

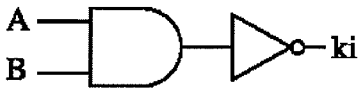
A kimenet csak akkor 1, ha *mindegyik* bemenet értéke 1 (egyik ÉS másik).

A NEM-ÉS (NÉS, NAND) kapu kimenete akkor 1, ha *bármely* bemenet értéke 0. A kimeneti értékek pontosan az ellentettjei az ÉS kapuénak, így egy ÉS kapu után kapcsolti inverter ugyanezt az igazságtáblázatot adja:



A	B	ki
0	0	1
0	1	1
1	0	1
1	1	0

ekvivalens áramkör



Emlékeztetőképpen: a NÉS kapu kimenetén a kis kör az invertálást jelenti az ÉS kapuhoz képest.

A Kizáró-VAGY (XOR) kapu

A Kizáró-VAGY (XOR) kapu funkciója az összehasonlítás: a kimenete 0, ha a bemenetek megegyeznek (akár 00, akár 11), és 1, ha különböznek:

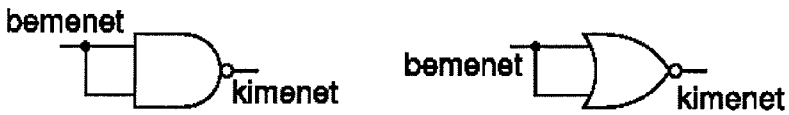
$$(A+B)(A+C) = A \cdot A + B \cdot A + A \cdot C + B \cdot C = A + B \cdot C$$

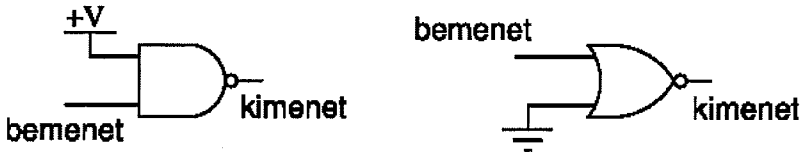
Az egyszerűsítés előnye a csökkent elemszám és az átláthatóság, ami olcsóbb és megbízhatóbb áramkört jelent. Egyszerűsíteni néhány változó esetén „ránézésre", heurisztikusan is lehet. Nagyon sok bemenet esetén, ha a függvény áttekinthetetlenné válik, ravasz matematikai eljárásokat dolgoztak ki, mint pl. Veitch-Karnaugh módszer, melyek a számítógépes tervezési eljárások (CAD, Computer Aided Design) részeivé váltak.

A NÉS és NVAGY kapuk speciálisak, mivel univerzálisak: egy matematikai állítás az, hogy minden logikai áramköri rendszer felépíthető belőlük, ezért önmagukban is alkalmasak logikai elemrendszer megvalósítására. Ennek technikai jelentősége az, hogy elvben elegendő számú csak NOR, vagy csak NAND áramkörörrel tetszőleges logikai rendszert fel lehet építeni: pl. az Apollo űrhajók Holdra szálló moduljában a vezérlő számítógép csak 3 bemenetű NOR kapukat tartalmazott. Nézzük meg példaképp, hogy a többi megismert kapu hogy állítható elő NÉS-ből illetve NVAGY-ból.

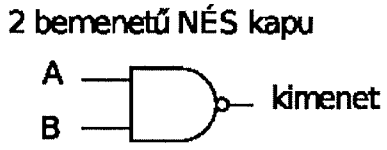
Pl. invertert könnyű előállítani:

bemenet	kimenet
0	1
1	0

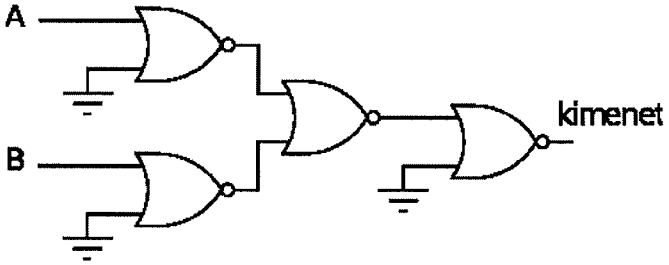




Az ÉS áramkört a következő kapcsolások valósítják meg NÉS ill. NVAGY áramkörökkel:



A	B	kimenet
0	0	1
0	1	1
1	0	1
1	1	0

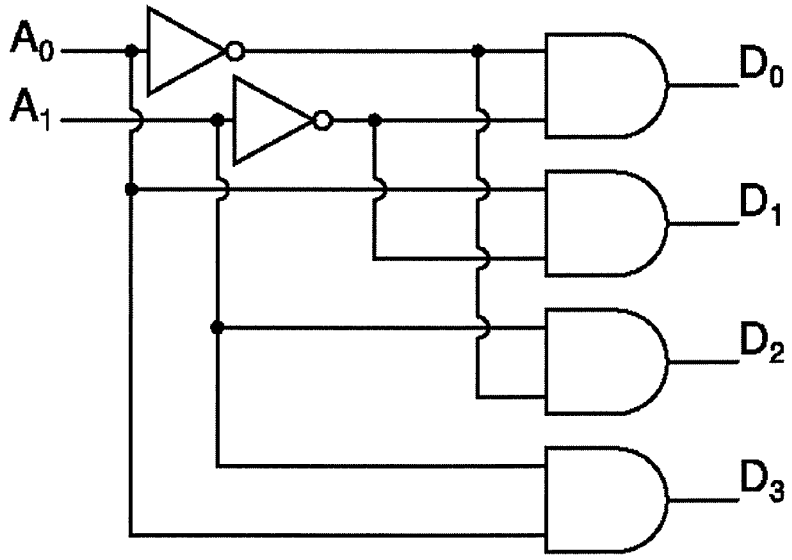


1.2. Kombinációs logikai hálózatok

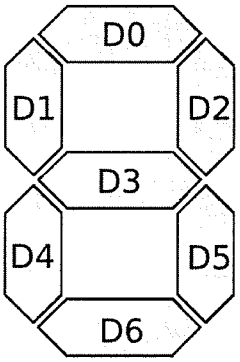
A kombinációs logikai hálózatok logikai kapukból álló összetett rendszerek, amelyek logikai függvényeket valósítanak meg. Egy ilyen rendszerre mindig igaz, hogy a kimenet(ek) csak bemenetek pillanatnyi értékeitől függnek, nem számít, hogy a bemeneti értékek milyen korábbi értékeket vettek fel. Fogalmazhatunk úgy is, hogy a kombinációs hálózatoknak nincs memóriájuk.

Minden korábban tárgyalt logikai kapu (ÉS, VAGY, NÉS, NVAGY, kizáró VAGY, negálás) önmagában is kombinációs hálózat.

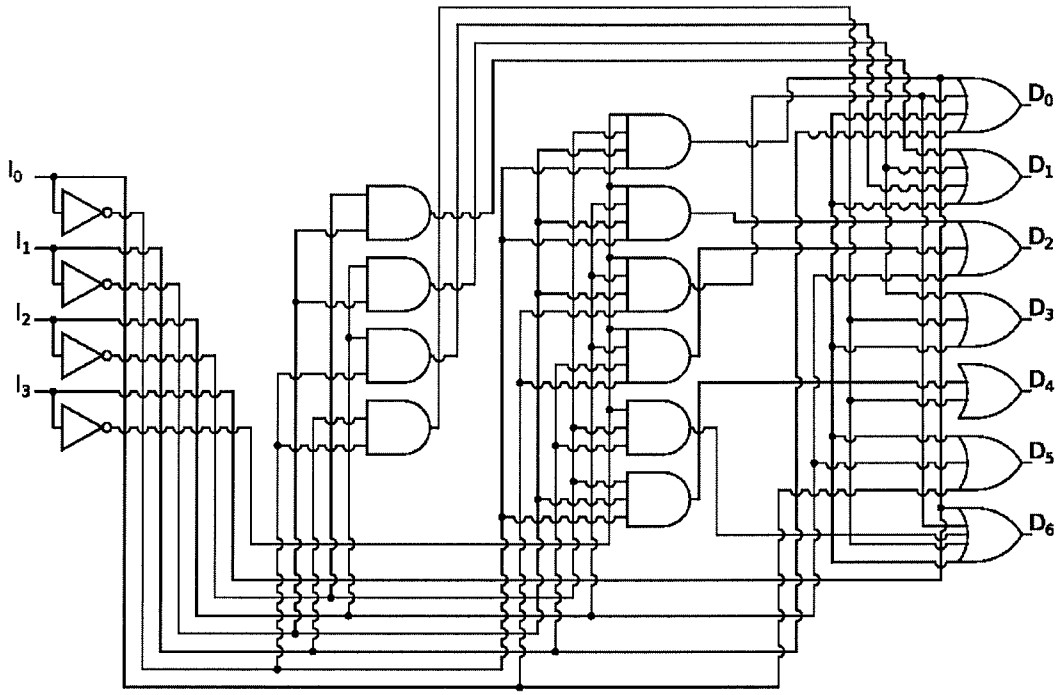
Az összetett függvények logikai kapukkal való megvalósítása nem mindig egyszerű feladat. Egy tetszőleges igazságtáblázat alapján szorzattá átírhatjuk a logikai 1 kimenethez tartozó sorokat, amit aztán VAGY művelettel egyesíthetünk. Az így elkészített logikai függvényt természetesen tovább egyszerűsíthetjük, ha szükség van rá. A következő tábázatban ezt az eljárást mutatja egy példa:



Egy másik példa a binárisról 7-szegmensre kódoló áramkör. A bemenet itt is kettes számrendszerben értelmezett négy Boole-szám, ami az arab számok értékét kódolja 0 és 9 között (a négy bináris szám által adott lehetőségek száma 16, ebből csak a 10 legkisebb értéket használjuk). A kimenet a számok megjelenítésére általánosan használt 7 szegmenses kijelző, ami egy nagyon elterjedt vizuális számábrázolási módszer. A kijelző bármelyik eleme, szegmense lehet bekapcsolt (1) vagy kikapcsolt (0) állapotban. Tekintsünk egy lehetséges szegmensbeosztást:



A bemeneten 4 értékkel meg tudjunk adni a kijelezni kívánt bináris számot. A 4 értékhez felírható kijelezni kívánt mintákat (a számokat) leíró igazságtáblázat is (pl a 8-as számnak az felel meg amikor minden szegmens be van kapcsolva):



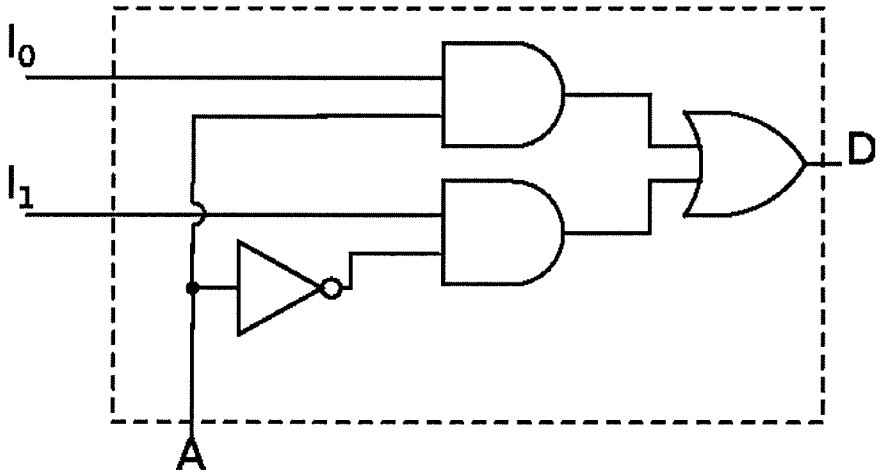
Erre a funkcióra a különbözű eszközökben gyakran szükség van, ezért külön integrált áramkört is gyártanak erre a célra (például SN7446).

1.2.2. Demultiplexer és multiplexer

A demultipexer (dmux) egy I bemeneten megjelenő logikai értéket kapcsol több kimenet közül a vezérlőbemenet által megadottra. Azaz, a demultiplexer kimenetei közül egyik éppen I értékű, a többi kimenet mint 0. Az áramkör hasonlít a fentiekben szerepelt, n jegyű bináris számot 2^n szám közül egyet 1-re író dekódolóra. Itt egyszerű logikai aktiválás helyett a bemenő jel értékét kapja a kimenet. Az 1-ről 2-re demultiplexáló áramkör igazságtáblázata a következő:

I	A	D_0	D_1
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Az ezt megvalósító áramkör:



A logikai függvények megvalósításának egyik érdekes lehetősége a multiplexerek felhasználása.

Tegyük fel, hogy adva van egy igazságtáblánk, ami három változós (F, G, H) függvényt ír le, tehát 8 sorból áll.

Az igazságtábla értékeit kössük egy multiplexer adat (D) bemeneteire, a megfelelő sorrendben. Ha ekkor az A_0, A_1, A_2 címbemenetekre sorban F, G és H értékét kötjük, akkor a Q kimenet éppen a D vezetékek közül a megfelelő sorszámú lesz – azaz éppen az igazságtábla szerinti függvénykimenet. Ez azon múlik hogy az igazságtábla sorait éppen az F,G,H mint bináris szám szerint sorszámozzuk.

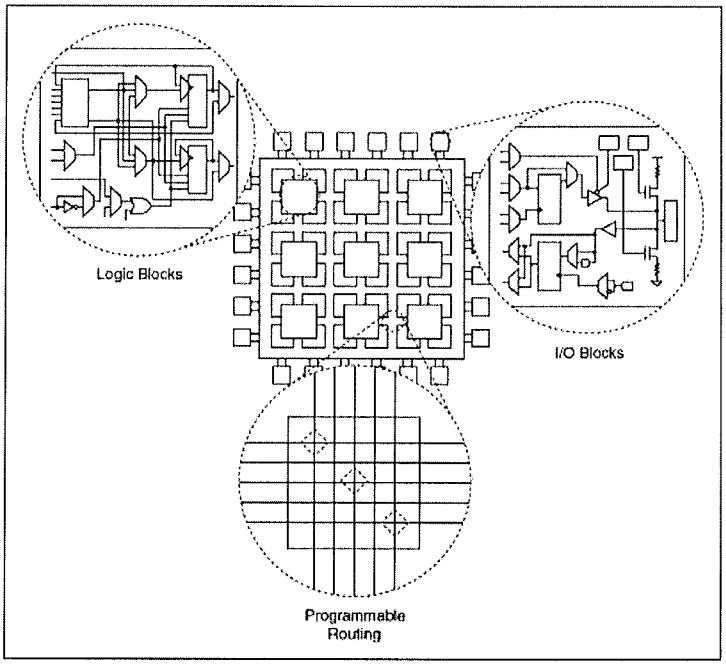
F	G	H	Q
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

A 8 bemenetű multiplexert így felhasználhatjuk az összes lehetséges háromváltozós Boole függvény előállítására. Az adott függvény kiválasztása a multiplexer bemeneteinek „programozásával" történik: ha szükséges akkor az áramkör fizikai átalakítása nélkül is kicserélhetjük.

1.2.3. Programozható logikai hálózatok: PLA, FPGA

A 7 szegmenses kijelző vezérlésénél láthattuk, hogy már ehhez a viszonylag egyszerű feladathoz is sok kapuból álló rendszerre volt szükség.

Ritkábban előforduló feladatoknál a logikai függvények megvalósítása egyedi kapukból történik, ami jelentős számú áramköri egység felhasználását jelentheti. Ennek a feladatnak



A felhasználó a logikai blokkokat, mint építőkockákat kötheti össze a programozás során, tetszés szerinti komplex digitális áramköröket létrehozva.

1.2.4. Összeadó áramkör

A digitális rendszerekben a Boole-típusú változókból bináris (kettes számrendszerű) számokat rakhatunk össze, és elsősorban ilyen számokkal dolgozunk mind a bemeneteken, mind a kimeneteken. A bináris számok tetszőleges jegyből állhatnak, gyakori a 8, 16, 32 vagy 64 számjegyű egység. Gyakori elnevezés, ha a szám egy-egy jegyét (ami tehát Boole-algebrai szám) bitnek nevezzük. Szintén gyakori konvenció, hogy a legértékesebb jegyet MSB rövidítéssel (Most Significant Bit), a legkevésbé értékes jegyet LSB rövidítéssel (Least Significant Bit) jelölik. Például a 11010 bináris szám, ami tízes (decimális) számrendszerben 26-tal egyenlő, MSB-je 1, LSB-je 0.

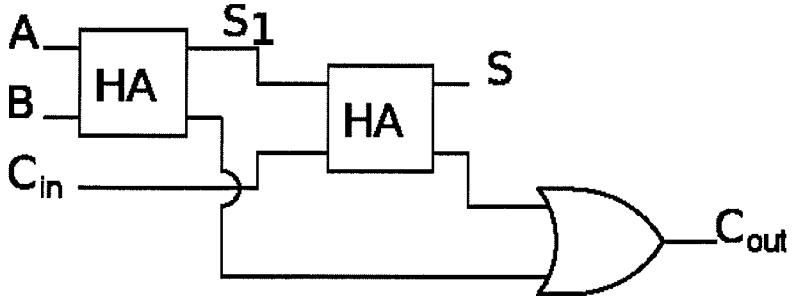
Ha műveleteket végzünk, akkor ilyen többjegyű számokkal kell tulajdonképpen a műveletet is elvégezni. Legegyszerűbb kérdés: hogyan valósítjuk meg logikai kapukkal két bináris szám összeadását?

Az összeadó áramkör vizsgálatát kezdjük a félösszeadó áramkörrel (half-adder, HA): a 1.2 igazságtáblázat foglalja össze a félösszeadó működését. Az A és B egyjegyű bináris számokat (biteket) adjuk össze, az eredmény az S (sum) bitben van, míg a következő bithez az átvitelt („1+1=0, marad az 1") a C (carry) bit mutatja.

A 1.1 ábrán a félösszeadó áramköri megvalósítását láthatjuk. A felépítése rendkívül egyszerű: az 1.2 igazságtáblázat alapján a C és az S oszlop a logikai ÉS és kizáró-VAGY műveleteknek felel meg, azaz egy XOR kapuval képezünk egy összeg, míg egy ÉS kapuval

c_{i-1}	x_i	y_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	0	0
1	1	0	0	0
1	1	1	1	1

1.3. táblázat. A teljes összeadó igazságtáblázata.



1.2. ábra. A teljes összeadó felépítése kapukból.

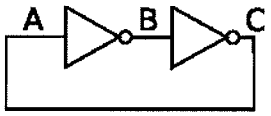
1.2.5. Logikai hazárdok

Eddig a Boole algebra szerint bináris jelekkel dolgoztunk, amelyek a kapukon azonnal áthaladnak. Valójában minden kapubemenet parazita kapacitása a hozzávezető vezetékkel és a meghajtó fokozat nem elhanyagolható kimeneti ellenállásával aluláteresztő *RC* szűrőként viselkedik, azaz a jelek véges felfutásúak lesznek. Magának a logikai kapunak is van egy működési sebessége, ami tovább bonyolítja a helyzetet. Egy logikai áramkörben a bemenet és a kimenet között általában különböző számú kapu van, ezért az egyes részáramkörök eltérő ideig késleltetik a jeleket (erre példát látunk majd a szinkron számlálóknál a XXX fejezetben). Különösen a gyors jeleknél okozhat problémát az, hogy figyelni kell a jelek véges felfutási és lefutási idejére.

Tekintsünk egy és áramkört, amelynek egyik bemenetére a másik negáltját vezetjük:

emlékezőképességére utal, olyan értelemben használjuk, hogy a rendszerben vannak belső, kívülről nézve esetleg rejtett logikai értékek, amelyek a kimenet értékébe beleszólhatnak. Arra, hogy ennek a visszacsatoláshoz köze lehet, a Schmitt-trigger volt jó példa: létezett ott olyan tartomány, ahol a kimenet egy adott bemeneti feszültségnél több értéket felvehetett. Itt most mind a kimenetek, mind a bemenetek, mind a belső állapotok kizárólag digitálisak, azaz mint Boole-algebrai számok értelmezhetők.

A legegyszerűbb 1 bites memória a bistabil multivibrátor, aminek két stabil állapota lehet. Az áramkör érdekessége, hogy külső bemenettel egyátalán nem rendelkezik, önmagában tehát nincs is haszna. A legegyszerűbb esetben két szembekapcsolt inverterből áll:

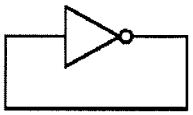


bistabil tároló

Az alábbi táblázatban láthatjuk, hogy a rendszernek két stabil állapota van:

A=C	B
0	1
1	0

Egyszerűen látható, hogy legalább kettő (és páros!) inverter kell a stabil állapothoz. Egy inverter visszacsatolva nem stabil:



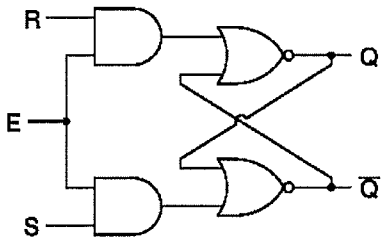
visszacsatolt inverter

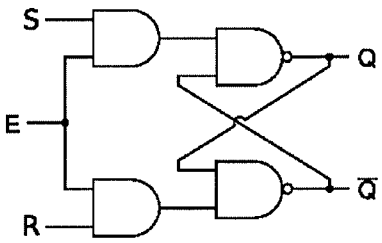
Ha a bemeneten 0 érték van, akkor az a kimeneten 1-et ad a kapu késleltetési ideje után, ez viszont megegyezik a bemenettel. Azaz akkor a bemeneten 1 van, és a kimenet 0-t vesz fel (szintén a késleltetési idő elteltével): a ciklus folytatódik, és a rendszer folyamatosan vált, oszcillál a $0-1-0-1\ldots$ állapotok között. A rezgés frekvenciáját a kapu késleltetése határozza meg. Világos módon ez a viselkedés páratlan számú inverter sorbakapcsolása esetén ugyanúgy megjelenik.

1.3.1. RS tároló

Az RS tároló két kapuáramkörből alakítható ki. Ha a bistabil multivibrátor két inverterét lecseréljük egy-egy kapura (NAND vagy NOR típusúra), akkor külső bemenete lesz a rendszernek.

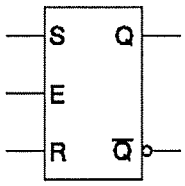
Tekintsük a következő megoldást, ahol két NVAGY vagy NÉS kapu kimenetét visszacsatoljuk a bemenetekre:





E	S	R	Q	$\overline{Q}$
0	0	0	tárol	tárol
0	0	1	tárol	tárol
0	1	0	tárol	tárol
0	1	1	tárol	tárol
1	0	0	tárol	tárol
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0

Ennek áramköri jelölése a következő:



Ebben az áramkörben az E jel 1 értéke kapuzza az R és S értékét: csak ennek $E = 1$ esetben íródik be a tárolóba a bemenetek értéke, ami azután az RS állapotát megszabja.

1.3.3. Engedélyezett D tároló

Fontos típus az ún. D (Delay) bistabil. Ez egy olyan S-R tároló, ahol egy inverter állítja elő az S-ből az R értékét, így nincs tiltott állapot. Az előző eset R és S bemenetéből tehát egyetlen bemenet lett. Van viszont egy másik külső bemenet is: az engedélyező E bemenet.

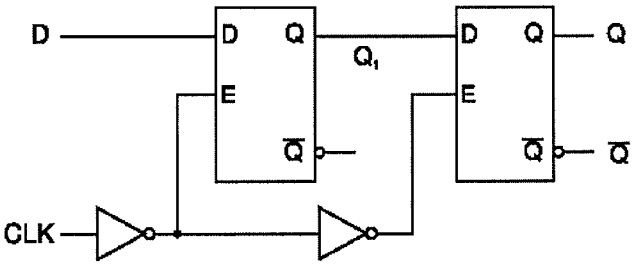
A D tároló lehetséges áramköri felépítése NOR és NAND kapukkal, valamit az igazságtáblázat a következő:

1.3.4. Élvezérlés: a hazárdok kiküszöbölése

Az előző alfejezet gondolatmenete az volt, hogy lényeges feladat megoldani szinkronizálást, azaz, definiálni azt a pillanatot amikor a digitális jel értékét beolvassuk (például egy RS vagy D tárolóba). Viszont fent a legegyszerűbb, nem mindig kielégítően jó megoldást láttuk: ekkor az órajel engedélyezésként szerepelt, az RS tárolóba beírhattunk bármit amíg magas volt az órajel – nem egy pillanatot, hanem egy véges szélességű időtartományt definiáltunk.

Van egy másik, sokkal egyértelműbben meghatározott megoldás, amit a digitális alkalmazások nagy részében kialakítanak. Az akitvitás idejét az órajel szélességétől legegyszerűbben úgy tudjuk függetleníteni, ha a *bemenet az órajel megfelelő irányú változásának a pillanatában (felfutó vagy lefutó élének nagyon rövid idejére korlátozva) aktív. Ezt élvezérlésnek (edge trigger) nevezzük.*

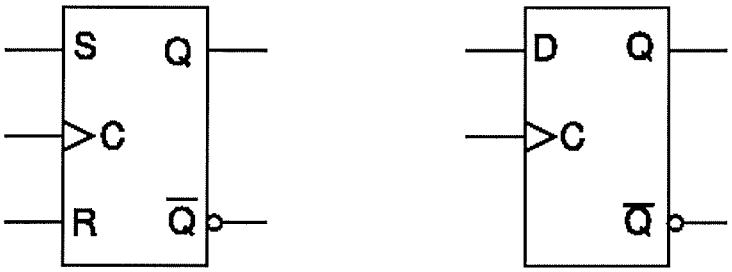
Élvezérelt D tárolót építhetünk két engedélyezett D tároló egymás után kapcsolásával. Az engedélyezés az elsőnél invertált, a másodiknál kétszer invertált. Kimenetnek a második D tároló Q kimenetét tekintjük, de a működés szempontjából hasznos követni az első D tároló Q1 kimenetét is.



A működés lényegét könnyű megérteni, ebben segít az alábbi példaként tekinthető idődiagram. Ha CLK alacsony, akkor az első D tároló engedélyezett (az invertálás miatt). A Q_1 kimenet ekkor szabály szerint követi a D bemenet értékét. Mivel viszont a második D tároló nem engedélyezett, a Q kimenet stabil, a második D tároló aktuális tárolt értékét mutatja. Váltson most CLK alacsonyból magasba. Ebben a pillanatban az első D tároló lesz zárolt, azaz Q_1 értéke rögzített, a második pedig lemásolja Q_1 értékét a Q kimenetre. Bárhogyis változik a D bemenet, az a Q_1 értékére már nincs hatással, ennélfogva a Q kimenetre sem.

Az élvezérelt multivibrátorokat *flip-flop*-oknak is nevezik. Ekkor az S, R és D bemenetek az ún. *szinkron* bemenetek, mivel csak akkor hatásosak, amikor az órajel megfelelő éle megjelenik. Tekintve hogy az órajel változásának pillanata a meghatározó, ezért hasznos ha az órajel mint időfüggő feszültség zajmentes, és változása 0-ból 1-be a lehető leggyorsabb.

A flip-flop áramkörök jelölése kicsit különbözik a tároló kapukétól. Az órajel melletti háromszög jelzi az élvezérlést:



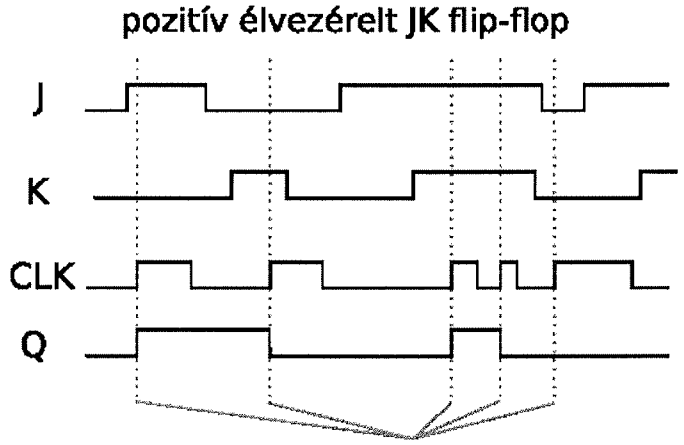
A digitális berendezések döntő többsége ún. szinkron hálózat, a számítási lépések itt szinkronizálódnak az órajelhez. A késleltetési idők miatti hibás működés problémáját úgy oldják meg, hogy az áramkörök egy periódikus vezérlőjelet (szinkronjel, órajel, clock, C, CL,CLK) kapnak. Ennek időtartamát általában az áramkör legnagyobb elképzelhető időkésése határozza meg. A kapuk csak az órajel felfutásának pillanatában tudnak a bemeneteken beolvasni. Ha a kapukésleltetések nagyok vagy az áramkör bonyolult, akkor az órajel frekvenciáját csökkenteni kell hogy helyes (de lassabb) működést kapjunk.

Az élvezérlés azt jelentette, hogy az órajel felfutásának pillanatában dől el a bemenetek aktuális értéke alapján hogy mi lesz a tárolt érték. Ha a bemenetek mégiscsak változnak a felfutás igen rövid de véges ideje alatt, akkor hibás működés állhat elő. Ez kiküszöbölhető, ha a felfutás előtt-után megfelelő tervezéssel elegendő időt (tipikusan az 1-10 ns tartományban) hagyunk amíg a bemenetek garantáltan stabilak, figyelembe véve a lehetséges késleltetési időket. Az az időtartam amíg a stabil bemeneteket a tervezőnek garantálnia kell, fontos paraméterei a ténylegesen kialakított rendszereknek, ezért a gyártók a fizikailag megvalósított eszközöknél specifikálják értékeiket.

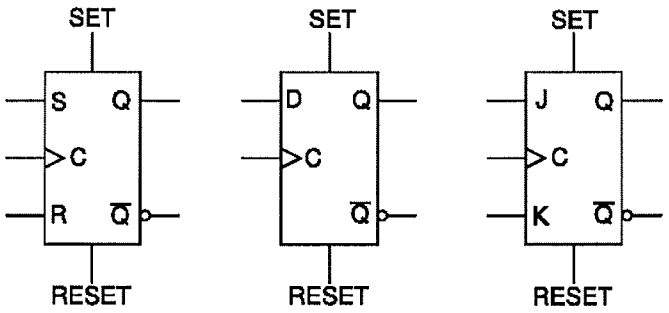
1.3.5. Az élvezérelt JK flip-flop tároló

Az RS tárolónak tiltott bemenetei voltak: ha az R és S egyszerre 1, a kimenet nem volt jól definiált. Az RS tároló tiltott bemeneti állapotai miatt külön figyelni kell arra, hogy tiltott kombinációk ne fordulhassanak elő. Ezt a problémát küszöböli ki a JK bistabil egy újabb visszacsatolási irány hozzáadásával. A J és a K az R és S bemenetekhez hasonlító bemenetet jelöl.

Az áramkör követi az RS működését, azonban az RS esetén tiltott egyidejű 1 bemenetre egy új funkciót vezetünk be: a kimenet az előző állapothoz képest invertálódik (átvált, angolul a toggle szót használjuk).

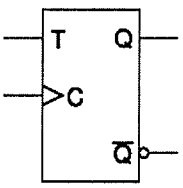


Az áramkör sok esetben a két belső tárolót közvetlenül elérő beállító (set) és törlő (reset) bemenettel is ellátják. Ezeket a bemeneteket aszinkron bemeneteknek is szokták hívni, mivel ezek az órajeltől függetlenül is aktívak (beállítják vagy törlik a tárolót). A beállító (set) és törlő (reset) vezetékeket más kapuknál is alkalmazhatják



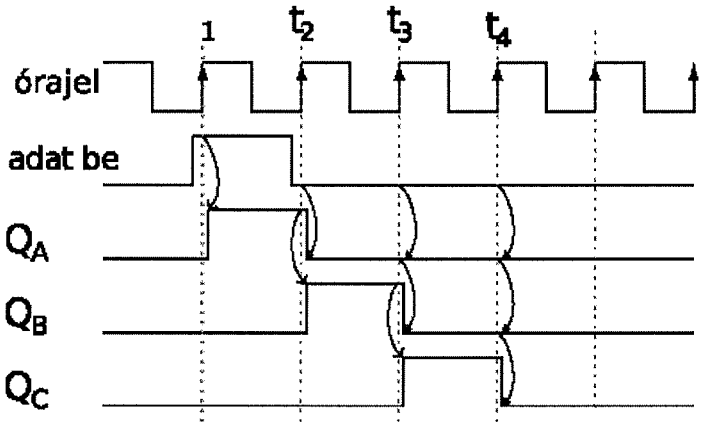
A T-tároló

A JK tárolónak három bemenete van (J, K, órajel). Megtehetjük, ha a J és K bemenetet egymással összekötjük és egyetlen, T-vel jelölt bemenetnek tekintjük – ekkor kapjuk a T tárolónak nevezett flip-flop-ot. Funkcióját tekintve, a T vezetéken át engedélyezzük az órajel váltásakor történő váltást a kimeneten. A T bistabil áramköri jelölése a következő:



A következő táblázat mutatja az igazságtáblázatot:

Ez az áramkör az adat be- és kimenet módja miatt a soros be/soros ki shift regiszter. A shift regiszter idődiagrammja a következő:



A t_1 időpillanatban az adatbemenet beíródik az első kapuba, az A fokozatba (a B és C állapota a korábbi adatokat őrzik, velük most nem foglalkozunk). A t_2 időpillanatban a Q_A érték átíródik a B fokozatba, az adatbemenet pillantnyi értéke pedig az A fokozatba. t_3 ugyanez ismétlődik azzal, hogy a Q_B kimenet beírásra kerül a C fokozatba. Az idődiagrammon látható, hogy az egyes fokozatok között az adatok átírása nem azonnal történik a kapuk késleltetése miatt.

A soros be/soros ki shift regiszter adatokat (bitsorozatot) tud tárolni és ezeket léptetni. Léteznek olyan shift regiszterek is, ahol a bemenet és a kimenet léptetése különböző lehet, ami pl. mérési adatok gyors beolvasását és lassú órajelű kiolvasását teszi lehetővé.

Párhuzamos be/soros ki shift regiszter

A párhuzamos be/soros ki shift regiszter (PISO, parallel-in serial–out shift regiszter) egy olyan shift regiszter, ahol az adatokat egyszerre, párhuzamosan írjuk be a regiszterbe, majd a shift regisztert léptetve sorosan olvassuk ki a kimeneten.

Gyakorlati alkalmazása a párhuzamos formában (pl. CPU regiszter, busz kimenet) érkező adatok soros átalakításában rejlik.

A következő ábrán egy D tárolókból felépített 3 tagú PISO felépítését láthatjuk. Minden egység egy ÉS-VAGY adatválasztót tartalmaz (ez igazából egy egy bites multiplexer!), amivel kiválasztható a beírás ill. a shift (léptetés) funkció. Beolvasás esetén a $SHIFT/\overline{LD} = 0$, azaz ekkor az adatok párhuzamos D_A, D_B, D_C bemenetről olvasódnak be az órajelre:

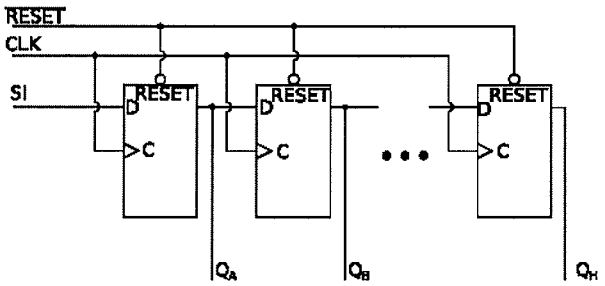
Az *SI* és *SO* be- és kimenet lehetővé teszi az áramkörök egymás után kötését (kaszkádolását), így tetszőleges méretű párhuzamos be/soros ki shift regisztert is létrehozhatunk.

Soros be/párhuzamos ki shift regiszter

A soros be/párhuzamos ki shift regiszter (SIPO, Serial-in, parallel-out shift regiszter) hasonlít a soros be/soros ki shift regiszterhez, mivel itt is folyamatos lépteti be a bemeneten megjelenő adatok a belső tárolóregiszterbe.

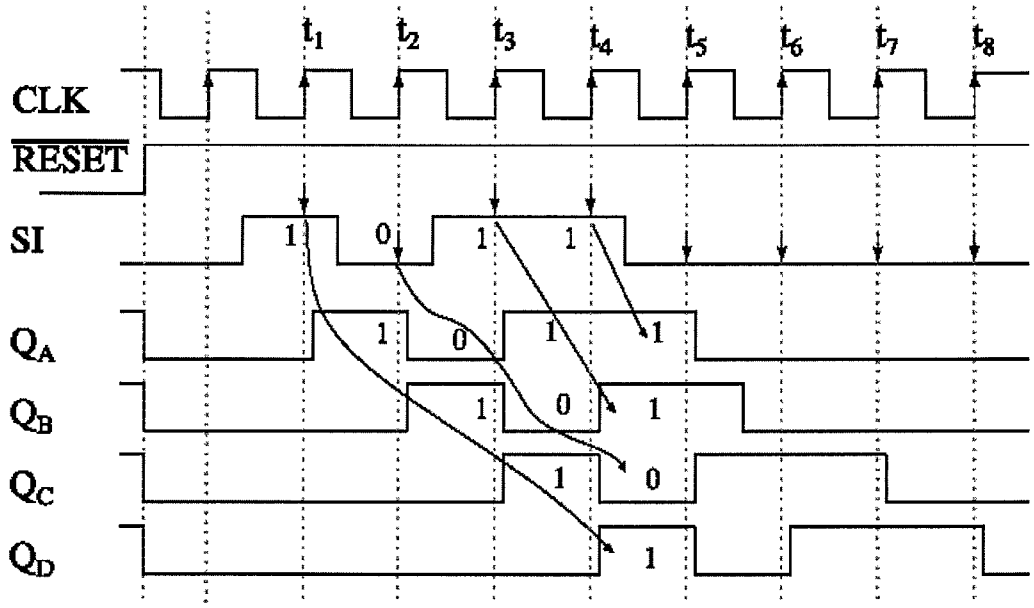
Gyakorlati alkalmazása a soros formában érkező adatok párhuzamos átalakításában rejlik. A kimenet lehet pl. egy CPU regiszter vagy egy buszmeghajtó áramkör.

A SIPO pl. D tárolókból áramkörökből építhető fel, az egyes kimeneteket a tárolók Q kimenete adja:



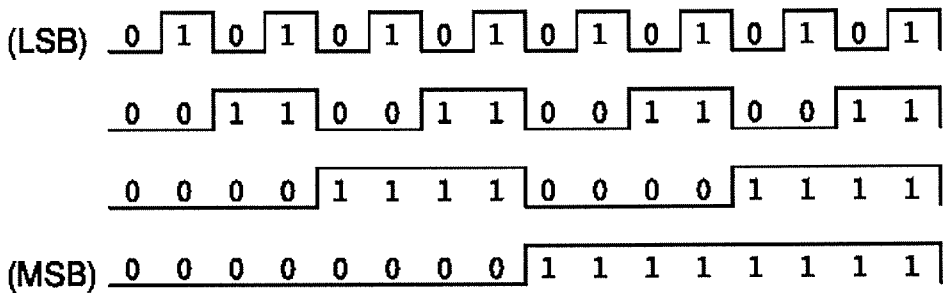
8 bites soros be/párhuzamos ki shift register

Az áramkör időbeli diagrammja a következő:

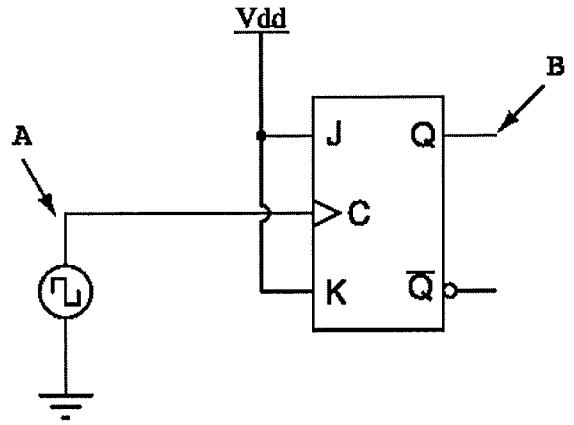


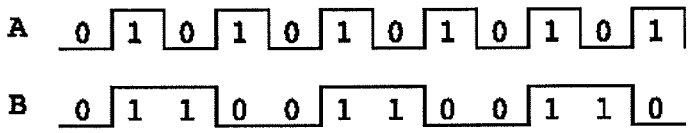
soros be/párhuzamos ki shift regiszter idődiagramm

Látható, hogy az egyes fokozatok az órajalek felfutásakor léptetik át az adatokat az



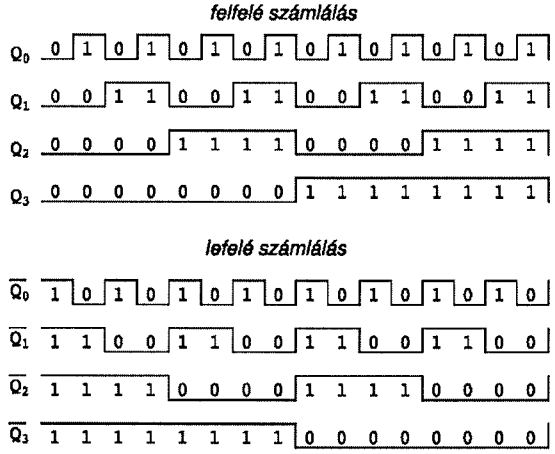
Egy JK flip-flop pontosan ilyen állapotváltozást mutat, ha J=K=1 állapotban órajelet kap:



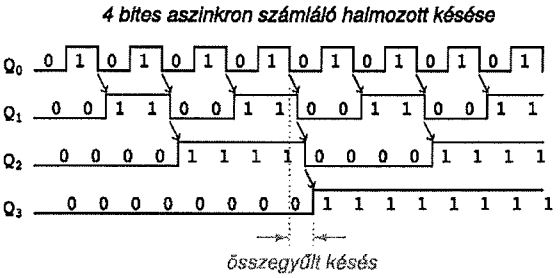


Aszinkron számlálók

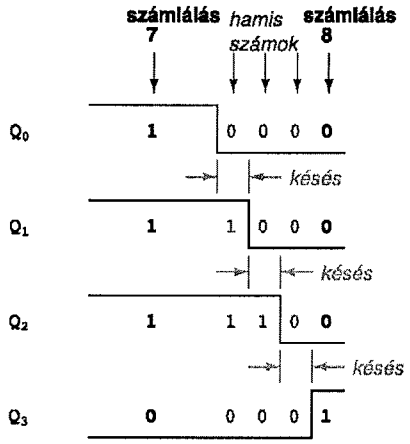
Negatív élvezérlésű JK flip-flopok egymás utáni kötésével (az órajelet az előző fokozat Q kimenetére kötve) megvalósíthatjuk a számláló által kívánt jelalakot (a J és K bemenet logikai 1-en van):



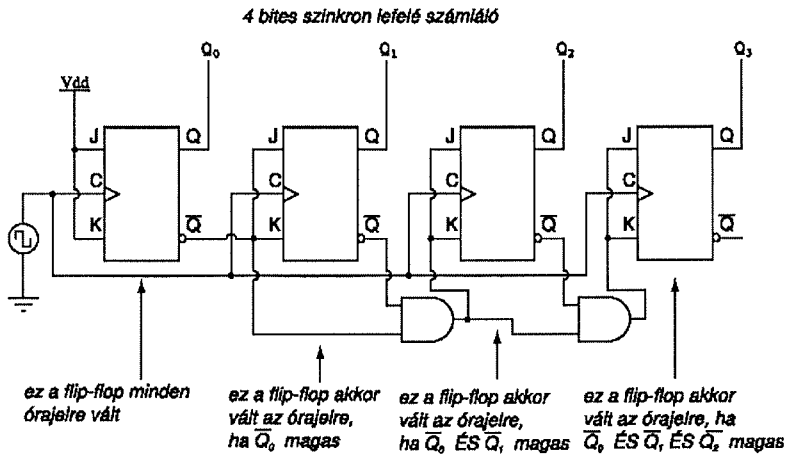
Sajnos ezeknek a számlálóknak kimenete időben furcsa hullámzást (ripple) mutat a számlálás során: ez a jelenség a bitszám növekedésével egyre erősebb lesz. Amikor egy flip-flop Q kimenetén megjelenik az 1-0 átmenet, a következő flip-flop átvált. Az átváltás csak a JK flip-flop késleltése után történik meg, azaz a átváltások időben elcsúsznak, éppen ezért ezeket a számlálókat *aszinkron* számlálóknak hívjuk:



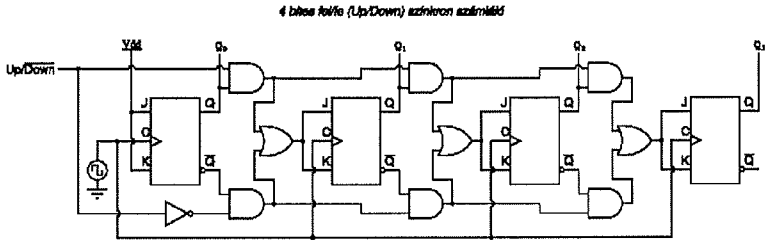
Látható, hogy az effektus erősödik az LSB - MSB irányba. Példaképpen nézzük meg az átmenetet a 0111 állapotból az 1000 állapotba:



A számláló a tiszta 0111 → 1000 átmenet helyett a 0111 → 0110 → 0100 → 0000 →

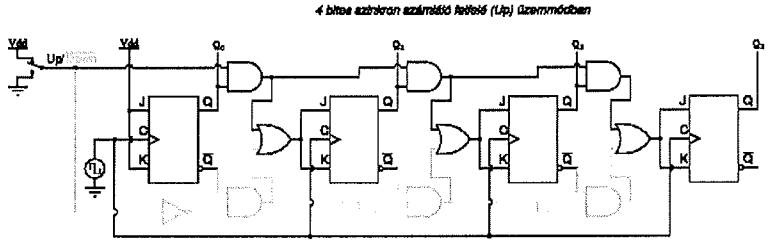


A két áramkört kombinálhatjuk, és így előáll a 4 bites, univerzálisan ténylegesen is használt szinkron fel-le számláló:



Ez az áramkör nem annyira komplex, mint elsőre látszik. A számlálás irányát a fel/$\overline{\text{le}}$ (Up/$\overline{\text{Down}}$) vezeték szabályozza. Minden bit esetén két ÉS és egy VAGY áramkör képez egy 1 bites demultiplexert, ami vált a két üzemmód között.

Ha felfele számolunk, akkor az Up/$\overline{\text{Down}}$ vonal aktív, logikai 1-et kap:



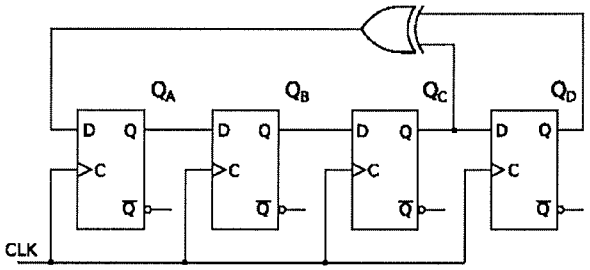
Az ábrán látható, hogy ekkor az alsó ÉS kapuk inaktívak, a felső ÉS kapukon a jel a normál szinkron számlálóval megegyező utat jár be.

Ha lefele számoláskor az Up/$\overline{\text{Down}}$ vonal logikai 0-át kap a felső ÉS kapuk inaktívak, az alsók a lefele számlálás logikáját engedélyezik:

kapukat használunk, ez a többi kaput is nullázza).

Ezen az elven tetszőleges N számrendszerben számoló számlálókat építhetünk, azonban figyelni kell a fejezetben tárgyalt időzítési problénákra.

A számlálók egy érdekes verzióját kaphatjuk, ha egy n elemű shift regiszter bemenetére visszavezetjük a regiszter néhány bitjének (átvitel nélküli) összegét. Összeget XOR kapukkal tudunk képezni, tehát az áramkör XOR kapun keresztül csatolja vissza a biteket. Pl. 4 bitnél:

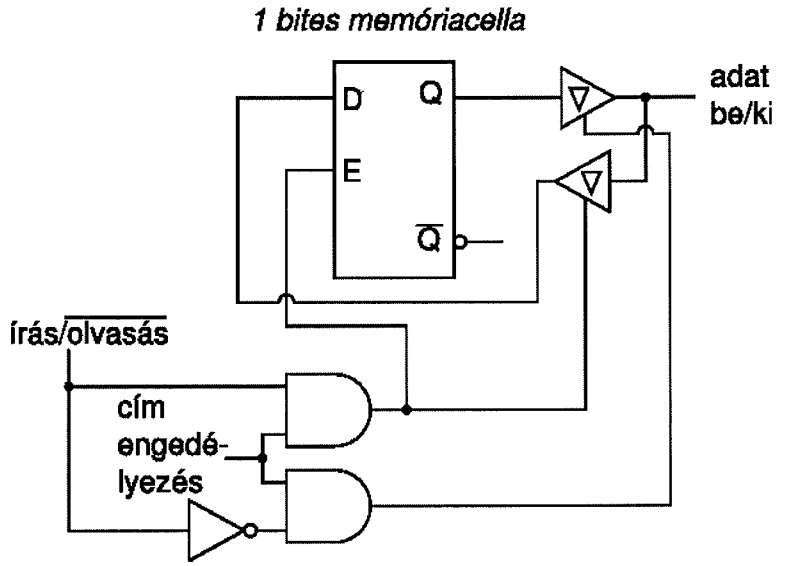


Q_A	Q_B	Q_C	Q_D
0	0	0	1
1	0	0	0
0	1	0	0
0	0	1	0
1	0	0	1
1	1	0	0
0	1	1	0
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
	⋮		

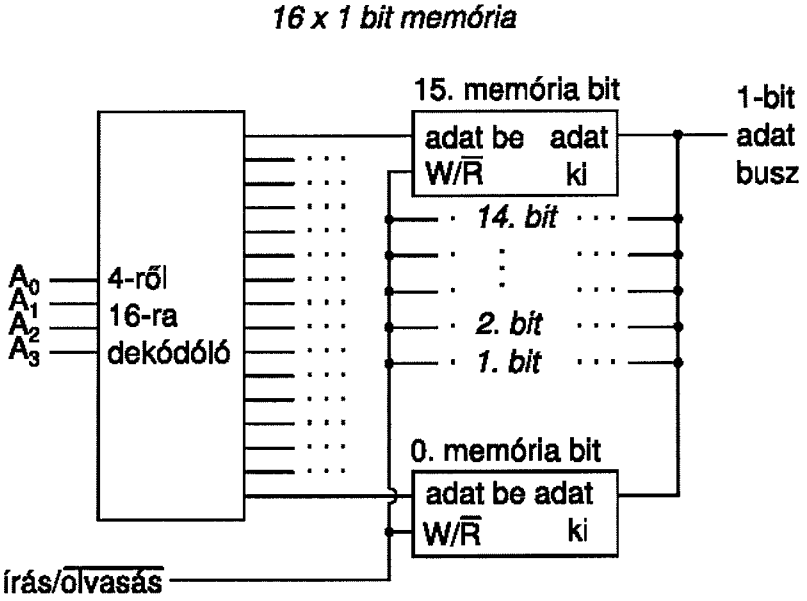
Az áramkör érdekessége, hogy ha a visszacsatolt biteket ügyesen választjuk, akkor a rendszer a 0000 állapot kivételével minden lehetséges állapoton át fog menni (természetesen ciklusonként csak egyszer), mire újra visszatér a kiinduló állapotba. A 0000 állapot a rendszer fixpontja.

A különböző shift regiszter hosszaknál különböző pontokat kell kiválasztani a maximális ciklushosszhoz, egy adott regiszternél több verzió is lehet. Pl. 16 fokozatnál a (16, 15, 13, 4) vagy a (16, 14, 13, 11) kombináció is jó (összesen 26 ilyen van). 32 fokozatnál jó

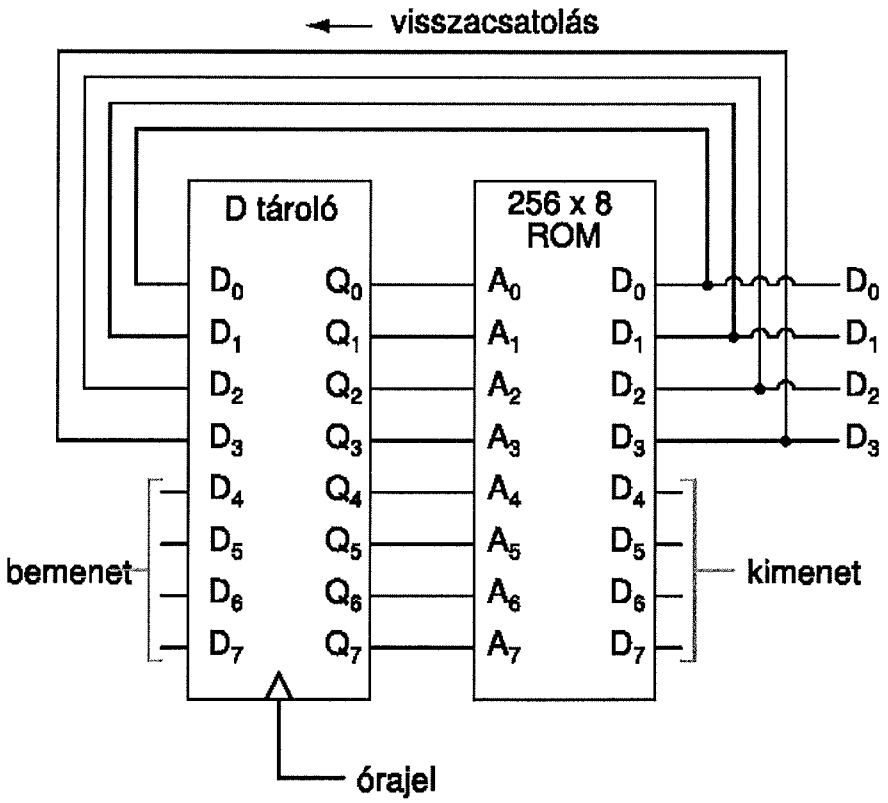
írást/olvasást így egyetlen Write vezetékkel tudjuk kiválasztani, míg az egész áramkör működését a cím engedélyező Enable vezeték vezéreli (amikor 0, akkor az áramkör külső kapcsolatai inaktívak):



Nagyobb memória esetén az egyes áramköröket egy $n \to 2^n$ dekódoló egységgel (l. XXX fejezet) választhatjuk ki, amit a cím engedélyező vezetékre kötünk. Például egy 16 bites memória elrendezése az alábbi ábrán látható. Nagy kapacitású memória esetén a dekódoló nagyon nagy számú vezetéket kell kezeljen.



A dekódoló egység N bit esetén N kimenetet kell, hogy tartalmazzon. A dekódolás egyszerűsítésére a valódi memóriacellák szervezése általában oszlop-sor felépítésű, azaz



Az ábrán láthatjuk, hogy a $D_0 - D_3$ vezetékek kizárólag a ROM visszacsatolására vannak használva, míg a $D_4 - D_7$ vezetékek külső bemenetként vezérlik az áramkört. A bemenetek sokkal nagyobb szabadsági fokot adnak a rendszernek, rajtuk át pl. kapcsolót vagy digitális szenzort is használhatunk az automata állapotainak vezérléséhez. A ROM kimenetén a $D_4 - D_7$ vezetékek kimenetként viselkednek, értékük nincs visszacsatolva.

Jelzőlámpa vezérlés véges állapotú automatával

Példaként nézzünk egy egyszerű feladatot: készítsünk egy közlekedési jelzőlámpa vezérlésére alkamas digitális gépet, amelyik egy külső x jellel vezérelhető. A x=0 esetben az üzemmód sárga villogó, míg x=1 a normál működést jelenti (piros, piros-sárga, zöld, sárga).

A rendszer működését ún. állapot-diagrammon ábrázolhatjuk:

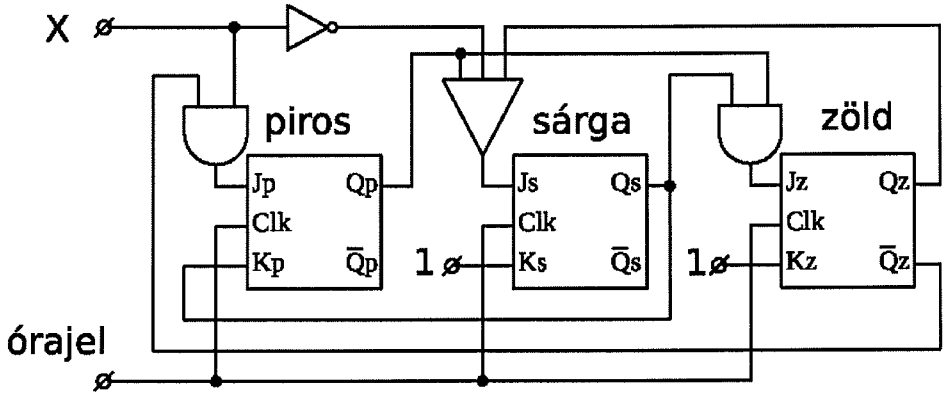
X	Q_P^n	Q_S^n	Q_Z^n	Q_P^{n+1}	Q_S^{n+1}	Q_Z^{n+1}	J_P	K_P	J_S	K_S	J_Z	K_Z
0	0	0	0	0	1	0	0	x	1	x	0	x
0	0	1	0	0	0	0	0	x	x	1	0	x
1	0	0	0	1	0	0	1	x	0	x	0	x
1	1	0	0	1	1	0	x	0	1	x	0	x
1	1	1	0	0	0	1	x	1	x	1	1	x
1	0	0	1	0	1	0	0	x	1	x	x	1
1	0	1	0	1	0	0	1	x	x	1	0	x

itt az x jelölés azt jelenti, hogy nem számít az adott bit állapota.

A JK flip-flop igazságtáblázata és a fenti tábla elemzése alapján meg lehet határozni a szükséges kombinációs logikai hálózat függvényeit (az eljárást magát nem tárgyaljuk). A függvények azért ilyen egyszerűek, mert kihasználjuk mind a J, mind a K bemenetet:

$$
\begin{aligned}
J_p &= X \cdot \overline{Q}_z \\
K_p &= Q_s \\
J_s &= \overline{X} + Q_p + Q_z \\
K_s &= 1 \\
J_z &= Q_p \cdot Q_s \\
K_z &= 1
\end{aligned}
$$

A kész áramkör mindössze 3 JK és 4 egyéb logikai kaput tartalmaz:



A három JK kimenetével kell vezérelnünk a három lámpát. A rendszer az órajel ütemében vált, az X bemenet pedig kiválasztja az üzemmódot.

1.4.5. Aritmetikai-logikai egység

A FSM végtelen nagy memóriával megépítve ekvivalens az ún. Turing-géppel, azaz akár számítógépnek is tekinthető, rajta minden program lefuttatható. A jelzőlámpás példán

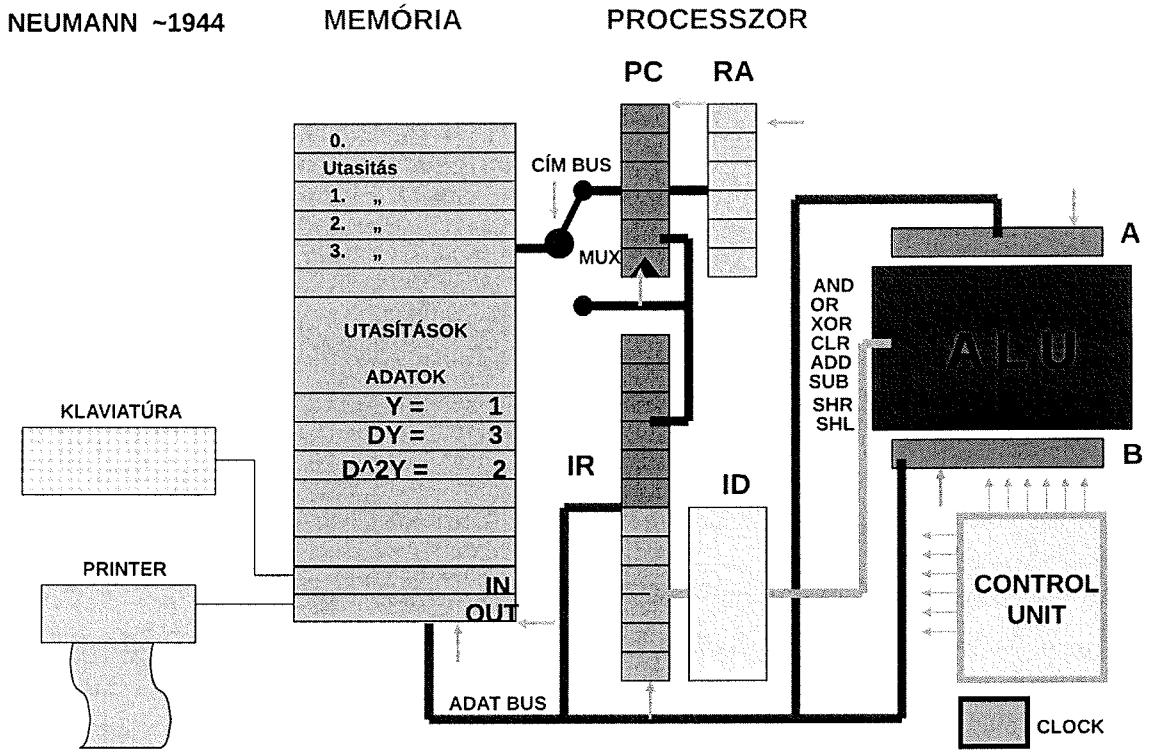
művelet	
CLA	A regiszter törlés
CMA	A komplementere (0-1 felcserélése)
AND	ÉS művelet $A \wedge B$
IOR	VAGY művelet $A \vee B$
RAL	A bitenkénti eltolása balra (szorzás 2-vel)
RAR	A bitenkénti eltolása jobbra (osztás 2-vel)
ADD	összeadás $A + B$
LD	olvasás az adatbuszról az A regiszterbe

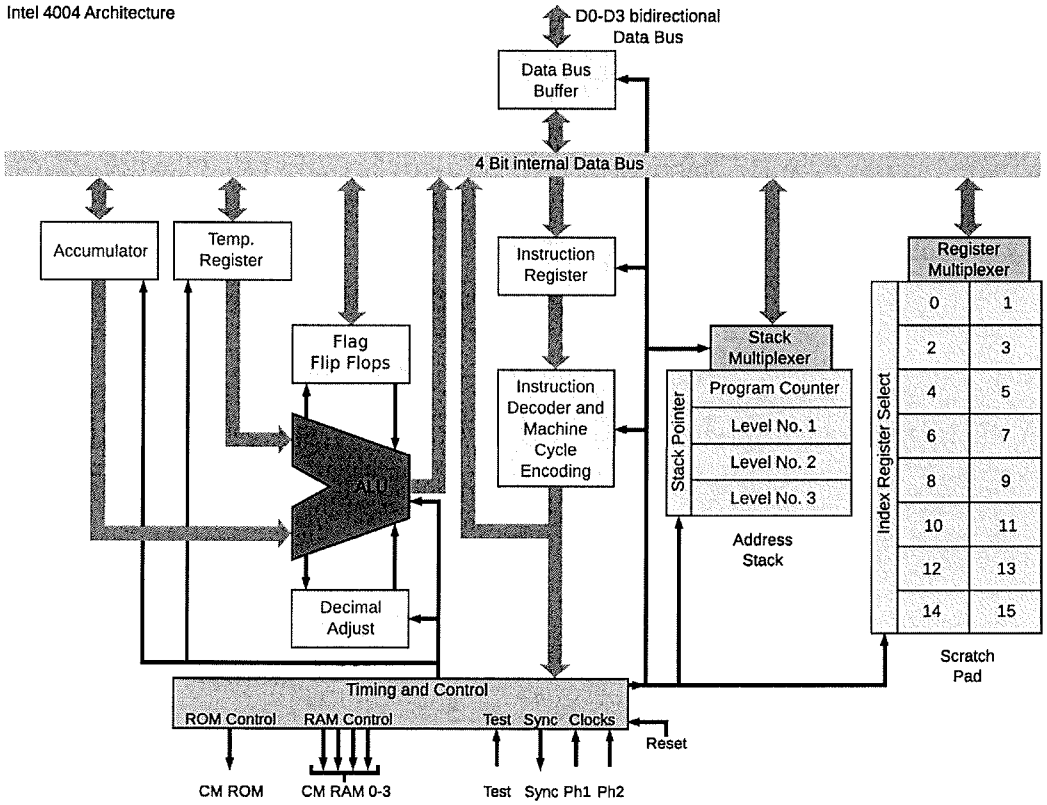
Figyeljük meg, hogy a tri-state/high-Z kapu segítségével az A regiszter tartalmát kiírhatjuk az adatbuszra. Az írás/olvasás vezérlése külön történik, itt most nem részletezzük.

Megjegyezzük, hogy shift regiszterek használatával építhető a műveleteket sorosan elvégző ALU is: ilyenkor csak 1 bites műveletvégző egységre van szükség, ami kevés kapuból egyszerűen megépíthető. A rendszer ekkor tetszőleges bitszámmal tud működni, hátránya, hogy lassabb a párhuzamos egységnél.

1.4.6. Számítógépek felépítésének vázlata

Az előzőekben megismert tárolóregiszterek, a memória és az ALU segítségével megérthető a Neumann-felépítésű számítógép struktúrája, aminek egy egyszerű példája a következő:





Megjegyzendő, hogy manapság komplex egy chip-es mikroszámítógépeket is gyártanak. Pl. 8 lábú IC tokozással néhány száz Ft (2013) körüli áron olyan IC vásárolható (ATtiny45), amelyben A/D konverter, 4kbyte memória, órajel generátor, időzítő stb. található egy 20 MHz-es 8 bites CPU köré építve.

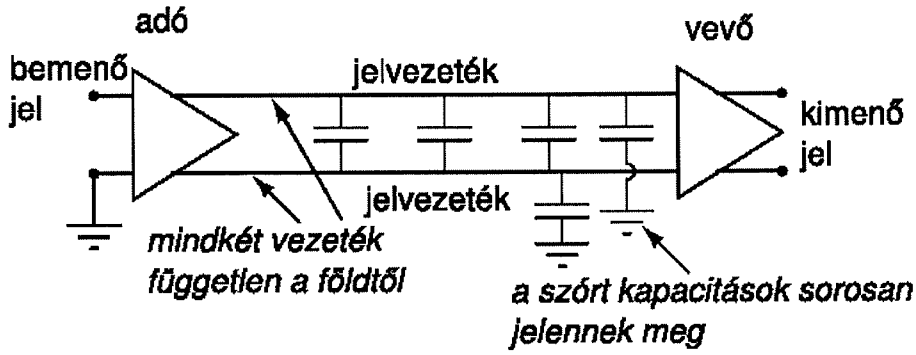
1.4.7. Információátvitel

Digitális információnak a továbbítását kétféleképpen végezhetjük

- *párhuzamos* módon: itt az adatok a regiszterekből egyszerre kerülnek átvitelre a buszon keresztül (l. XXX fejezet). A busz méretét vezetékek száma adja meg, (pl. 4, 8, 16, 32, 64 bit). A feszültségszintek jól meghatározottak, és általában az átvitelt vezérlővezetékek szabályozzák (l. tri-state kimenetek).

- *soros* átvitelnél az átviendő adatokat a kimeneten pl. egy shift regiszterrel (pl. párhuzamos be/soros ki shift regiszterrel) alakítjuk át, míg a bemeneten pl. soros be/párhuzamos ki átalakítást végezhetünk.

A soros átvitel kevesebb vezetéket igényel, de időben lassabb. Egy áramkörön, készüléken belül általában párhuzamos átvitelt használnak, távolabbra pedig sorost. Az



Itt a szórt kapacitások sorba vannak kötve, ráadásul a jelvezetékekből csavart érpárat is ki lehet alakítani, ami tovább növeli a zavarvédelmet és a sebességet. Hasonló áramköröket alkalmaznak pl. az UTP csatlakozásoknál a számítógépes hálózatokban.

1.5. Analóg és digitális mérések elvei

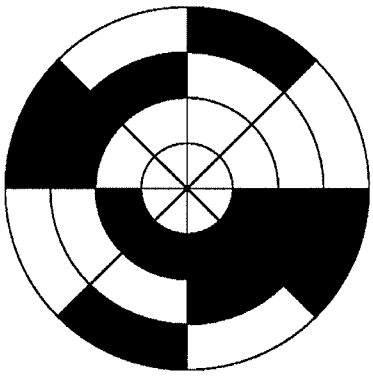
Az analóg és digitális mérési eljárásokat régóta ismerik és használják. Pl. az időmérésnél az analóg mérés egyszerű, de hosszú idők mérésére pontatlan eszköze a homokóra, míg pontos időalap (pl. inga, mint mechanikus oszcillátor) segítségével a hosszú időtartamokat is pontosan lehet mérni a periódusok leszámlálásával (fogaskerekekkel).

Az analóg mérések általában nem egészen analóg folyamatok: súrlódás és más nemlinearitások miatt kicsi változás esetén már nem követik a mérendő mennyiséget, gondoljunk pl. egy mutatós műszerre: egy elektronnal több másodpercenként valószínűleg nem mozdítja meg a mutatót. Digitális mérésnél a mérési kvantumnál kisebb bemeneti változatok nem változtatják meg a kimeneti számértéket. A mérendő értéktartományt és a mérési kvantumot a kívánt mérési pontosságnak megfelelően kell megválasztani. Mély filozófiai kérdés, hogy hiszünk-e a valódi analóg jelekben, vagy világot kvantált felépítésűnek képzeljük, de ezt sok esetben nem vizsgáljuk: pl. a hétköznapi méréseknél az áramot értékét annak ellenére folytonosnak tekintjük, hogy tudjuk, hogy az diszkrét elektronok folyama.

1.5.1. Digitális mérőátalakítók

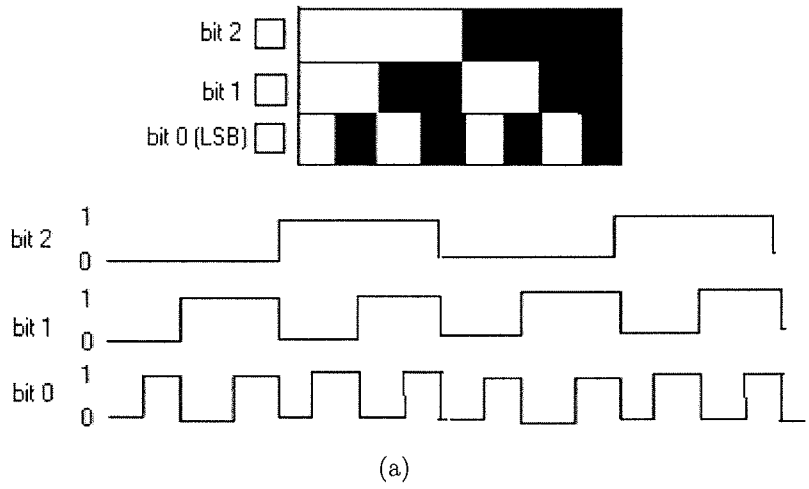
A digitális mérőátalakítók a bemenetükön kapott jelből digitális jelet állítanak elő. A legegyszerűbb ilyen átalakító a kapcsoló, amely be/ki állapotát pl. egy pozíció jelzésére használhatjuk fel. Elektromos jeleknél a komparátor (l. XXX fejezet) használható arra, hogy egy feszültségről eldöntsük, hogy kisebb vagy nagyobb egy adott értéknél.

A mechanikai pozíciómérés egyik elterjed eszköze a kódtárcsás mérőátalakító, ennek egy példája látható a 1.3 ábrán. A berendezéssel együtt elforgó tárcsa fogazata két fénysugár útját szakítja meg: a fénysugár intenzitásának változásait diszkriminátor érzékeli, melynek kimenőjele digitális.



(a)

1.4. ábra. Bináris kódolású kódtárcsa.



(a)

1.5. ábra. Bináris kódolás működési diagramja.

010-000 sorrendben billenek át a bitek (ez hasonló az aszinkron számláló problémájához, l. a XXX fejezetben). Ez azt jelenti, hogy az érzékelők bizonyos pozíciókban (a szektorok határán) egyszerűen rossz értéket is kiadhatnak.

A rendszer pl. úgy lehet működésképessé tenni, ha elérjük, hogy a szektorhatárokon csak egy bit változzon: ez az ún. Gray-kód lényege. A Gray kódolású kódtárcsa mintázatát a 1.7 ábra mutatja. A Gray kódolás működési diagramja az 1.7 ábrán látható.

A Gray kód azzal a tulajdonsággal rendelkezik, hogy az egymás után következő értékek csak egyetlen helyiértékben (bitben) különböznek, Ez azt jelenti, hogy a lassan/bizonytalanul forgó tárcsa két szomszédos szektor határán csak vagy az egyik, vagy a másik értéket

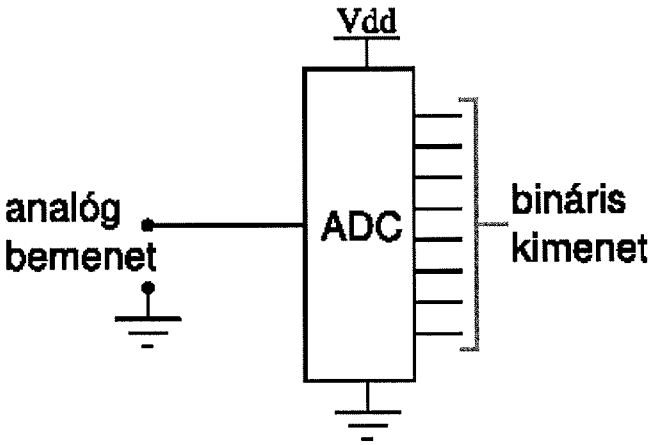
1.5.2. Digitális és analóg jelek átalakítása

A külső érzékelők digitális eszközök való kapcsolása akkor egyszerű, ha az érzékelők maguk is digitálisak (pl.kapcsolók, relék és kódtárcsák): ekkor egy egyszerű szintillesztéssel meg lehet oldani a feladatot.

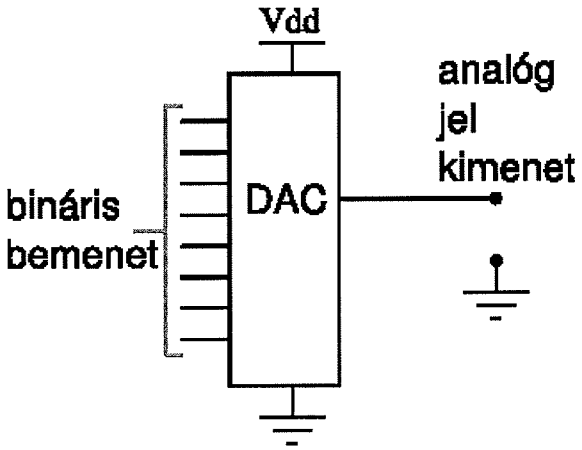
A mérések során használt eszközök nagy része analóg jelet ad ki magából, amit folytonosnak tekintünk. Ez az esetek nagy részében megfelelő, még akkor is, ha az az analógnak tekintett jel diszkrét elektronok árama. Csak néhány esetben (mint pl. a csillagászati CCD-k képalkotása) kell figyelembe venni ezt a kvantálást.

A számítógép (mikrovezérlő vagy egyéb elektronikus eszköz) vezérlelheti is a külvilágot (mérőberendezést) pl. egy feszültséggel, ezért az átalakítóknak általában mindkét, a digitális-analóg és az analóg-digitális irányú változatára is szükségünk lesz.

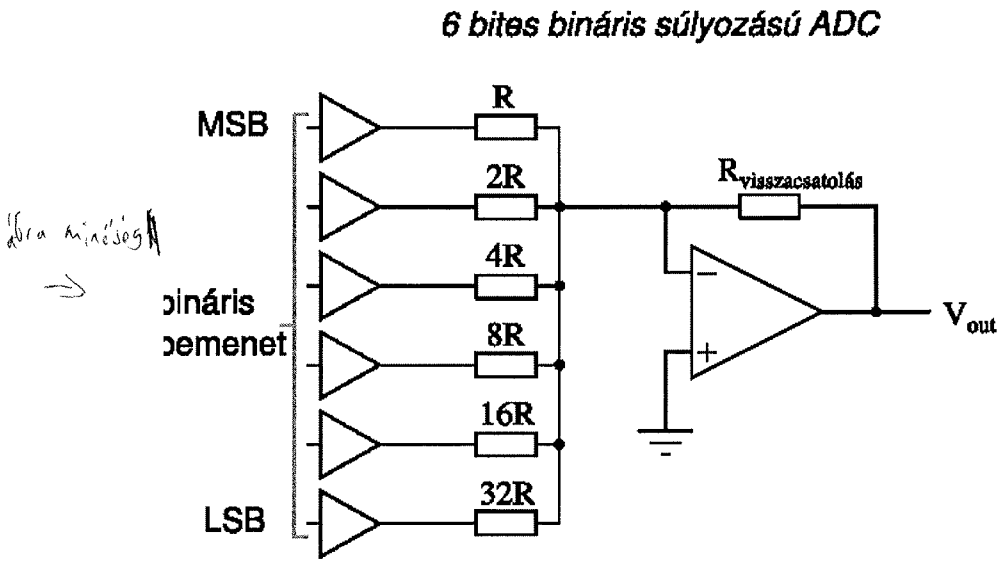
Az analóg-digitális átalakító (Analog Digital Converter, ADC) az analóg jeleket alakítja digitálissá:



A digitális-analóg átalakítók (Digital Analog Converter, DAC) a digitális jelből állítanak elő analógot:



Pl. egy 6 bites DAC a következőképpen nézhet ki (LSB: Least Significant Bit, legkisebb helyiérték, MSB: Most Significant Bit, legnagyobb helyiérték):

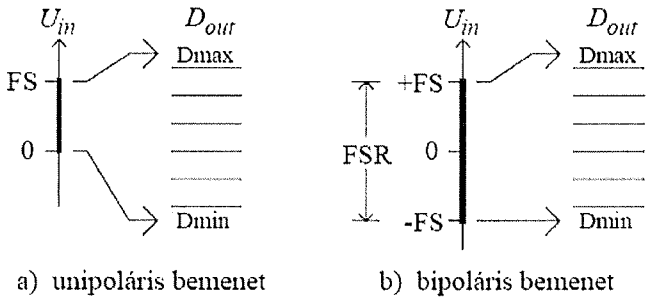


Az R_i ellenállások nagysága az egyes bitek helyiértékevel arányos, ezért a biteknek megfelelő I_i áramok is ugyanígy súlyozódnak (a műveleti erősítő bemenete virtuális földpont!). A kimeneten az áramok $\sum_i I_i$ összegével arányos $U = -R_{\text{feedback}} \sum_i I_i$ feszültség jelenik meg.

A kimeneti tartomány nagysága az R_{feedback} ellállással állítható be: pl. ha a 0001000 bemenetre a kimeneti feszültség -4 V, akkor az ellenállást felére csökkentve a kimeneti feszültség is felére csökken ugyanakkora bemenet esetén.

A DAC felbontását növelni további bináris bemenetek hozzáadásával növelhetjük. Meg kell jegyezni, hogy az egyes bitekhez tartozó V_i bemenő feszültségek digitális kapuk kimenetei, azaz értékük közel 0 vagy az U_T tápfeszültség. A kapuk kimenő feszültségének a DAC felbontásán belül meg kell egyezni, máskülönben az átalakítás nem pontos.

Az R/2^nR DAC magas bitszám esetén különleges ellenállásokat igényel, ezért nehezen (drágán) állítható elő. Egy alternatív kapcsolással (R/2R létra) ugyanezt az áramösszegzést lehet elérni csupán kétfajta ellenállás felhasználásával:



1.8. ábra. Az AD átalakító méréshatárának értelmezése.

ami csökkenti a gyártási költségeket, mikrovezérlőkhöz is könnyebben illeszthető ill. igény esetén (pl. orvási műszerek) egyszerű galvanikus elválasztást tesz lehetővé,

A digitális műszerekhez, voltmérőkhöz gyártott AD átalakítók kimenete decimális kódolású, és párhuzamosan vagy számjegyenként soros formában adja ki az értékeket. A számjegyenként soros kimenet a multiplexelt LED vagy LCD kijelzők meghajtását egyszerűsíti, és a kimenetek számát is csökkenti.

Egy ADC felbontóképességének azt az analóg jelváltozás nevezzük, ahol a kimenet vált, azaz ahol a változás megkülönböztethető a digitális kimeneten is. A felbontóképesség egy n bites bináris kódolású konverter esetén elvileg megegyezik $q = FSR/2^n$ kvantumnagysággal. A felbontóképességet általában bitekben adják meg, pl. 8, 10, 12, 16 bites stb. típusokat vásárolhatunk.

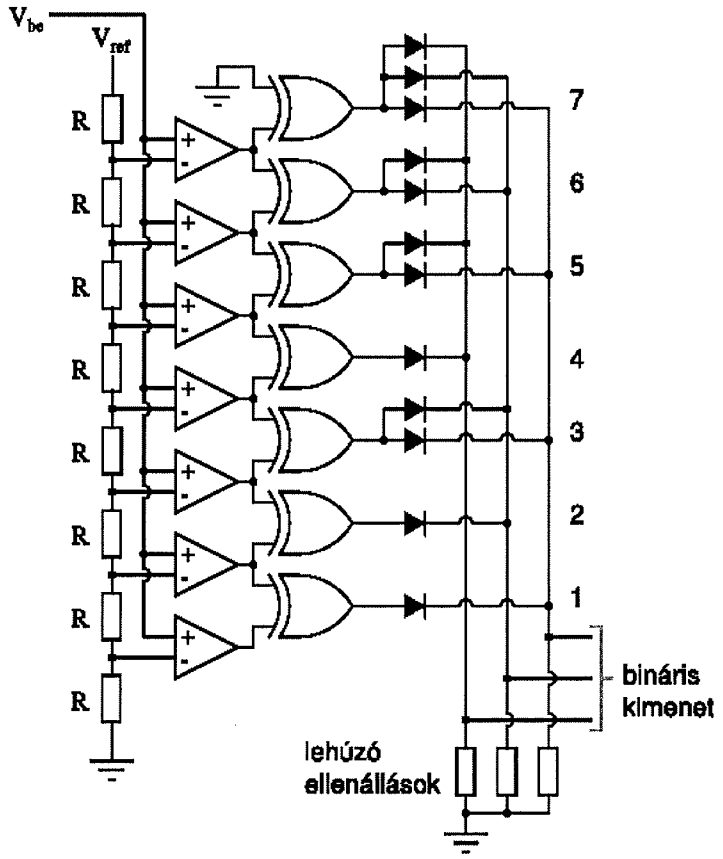
Pl. egy 8 bites ADC $2^8 = 256$ különböző szintet tud megkülönböztetni a teljes tartományban. Egy 14 bites ADC viszont már $2^{14} = 16384$ szintre osztja a bemeneti jeltartományt.

AD konvertereket igen változatos technikai adatokkal hoznak forgalomba, rendszerint egyetlen chip formájában. A konverzió időtartama érthetően függ a felbontástól, tipikusan általában 1 μs - 1 sec közötti ideig tart egy mérés. Az extrém gyors flash konverterek (néhány nsec környéke) eléggé drágák, a kisebb pontosságú, lassabban működő chipek egészen olcsók (gondoljuk csak arra, hogy digitális multimétereket, digitális lázmérőket, stb. milyen viszonylag kicsiny összegért vásárolhatunk).

A következőkben végigtekintjük az AD konverterek legfontosabb típusait.

Flash AD konverter

A párhuzamos (flash) ADC az elvileg talán legegyszerűbb, a leggyorsabb és egyben a legdrágább (legtöbb alkatrészt igénylő) átalakító. Egy 3 bites párhuzamos ADC áramkört láthatunk az ábrán:

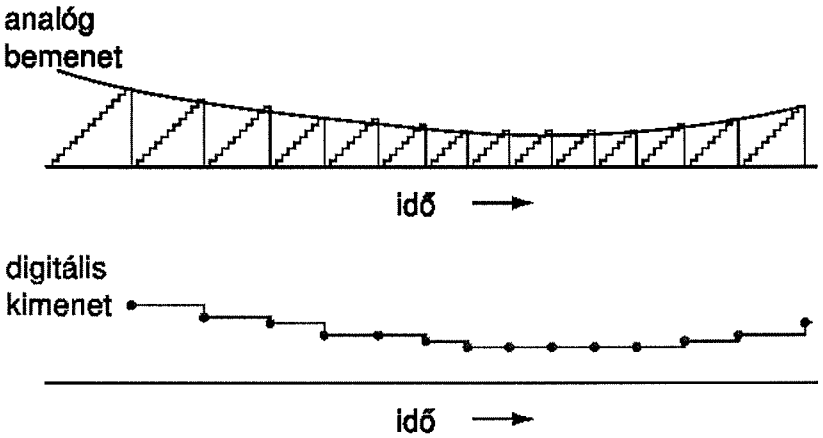


A flash konverter - mégha nagyobb bitszám esetén sok alkatrészt is igényel - a leggyorsabb ADC, mivel az átalakítás egy órajel alatt megtörténik (igazából nincs is szükség órajelre)! A sebességet csak a komparátorok és a kódoló áramkör késleltése korlátozza, ezért a sebesség nagyon nagy ($\approx$ 300 megaminta/s) is lehet: ilyeneket konvertereket használnak videojelek digitalizálásánál, tárolós oszcilloszkópokban, stb..

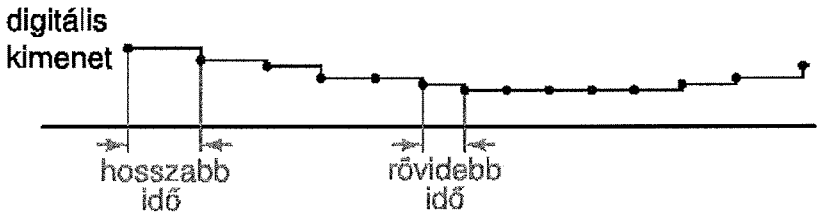
Egy érdekes tulajdonsága a flash konverternek, hogy a digitalizálási szintek csak az ellenálláslánctól függenek, azaz igény esetén nem-lineáris skálát is lehet alkalmazni az analóg-digitális átalakításnál az ellenállások megváltoztatásával.

Számláló AD konverter

A számláló AD konverter az egyik legegyszerűbb AD konverter:

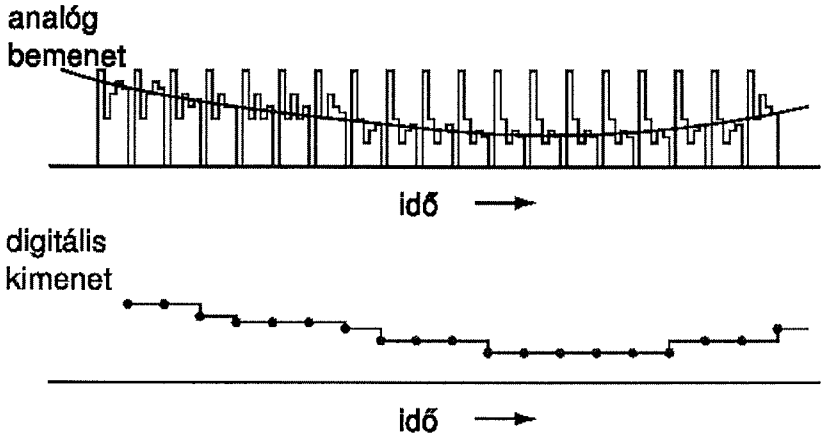


Ez az ingadozás a számítógépes feldolgozást (pl. teljesítményspektrum meghatározás) bonyolítja, a digitális szűrést pedig megakadályozza.



Szukcesszív approximációs ADC

A szukcesszív approximációs konverter javítja a számláló AD eljárását, gyorsítja és a mérendő mennyiség értékétől függetlenné teszi a mérési (átalakítási) időt. Viszonylag egyszerű felépítése, pontossága és sebessége miatt ez az egyik legelterjedtebb átalakító a számítógépes mérésadat-gyűjtő berendezésekben.



Az átalakítást a külső áramkör a SAR törlésével indítja, a konverzió végét az áramkör jelzi (pl. EOC, End Of Conversion).

Kettős meredekségű AD átalakító

Mérőműszerekben és máshol is sok helyen használják az ún. kettős meredekségű AD konvertert. Ez az egyszerű felépítésű robosztus ADC akár nagypontosságú (11-12 bites) átakakítást is végezhet, általában mérsékelt sebességgel.

A méréshez egy integráló áramkör (l. XXX fejezet) bemenetére kötik a a mérendő U_{be} bemeneti feszültséget (l. 1.9 ábra). Egy meghatározott T_0 idő után (amit pl. egy fixfrekvenciájú (pl. kvarc) oszcillátor impulzusainak számlálásával mérnek) az integrátor bemenetére negatív előjelű, konstans értékű U_{ref} feszültséget vezetnek, azaz az integrátort kisütik.

A T_m kisülési időt szintén egy számlálóra vezetett impulzusokkal mérik, és a számlálást akkor állítják meg, amikor az integráló áramkör kimenete eléri a 0-t. Ekkor az integrátoron az össztöltés 0, azaz $T_0 U_{be}/R = -T_m U_{ref}/R$. Innen $T_m = -U_{be} T_0 / U_{ref}$, a T_m idő (és az impulzusok száma is) arányos az U_{be} bemenő jellel.

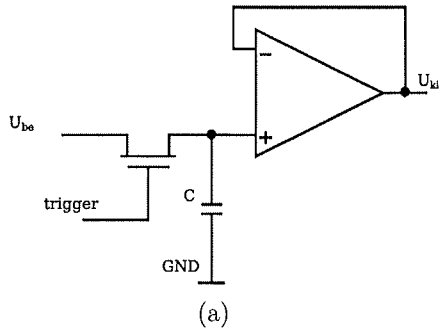
Világos módon ha az oszcillátor frekvenciája nem változik egy konverzió alatt (ezt könnyű teljesíteni), akkor a mérés pontossága nagy lehet. A konvertálási idő ~~hasonlóan a~~ itt is függ a mérendő feszültségtől. Ez az áramkör a bemenő jel változásra kevésbé érzékeny, mint a szukcesszív approximációs konverter.

1.5.5. Változó jelek mérése, mintavételezési törvény

Gyorsan változó jelek átalakításánál fontos a magas AD átalakítási frekvencia (conversion rate, f_{cr}), ami egyben általában magas jel mintavételi frekvenciát (f_m) ill. rövid a mintavételi periódusidőt (T_m) is jelent. Az f_m mintavételi frekvenciát *Nyquist frekvenciának* is szokták hívni.

Közönséges átalakítóknál a mintavétel ritkábban történik, mint a konverzió, de az

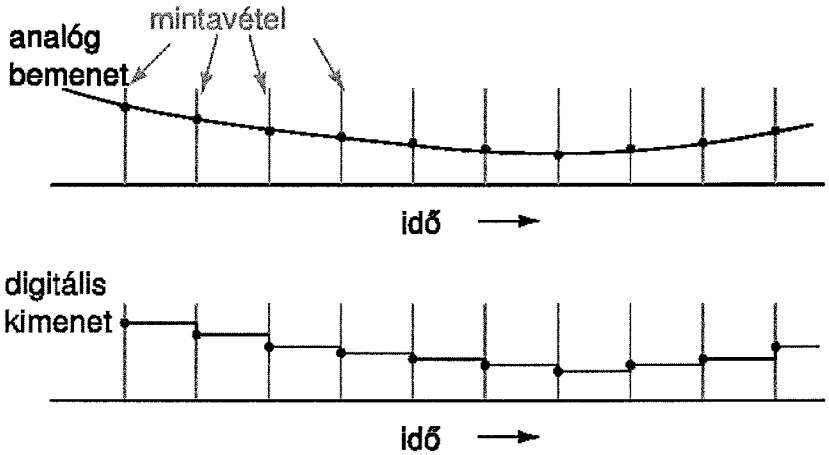
tárolja a feszültséget.



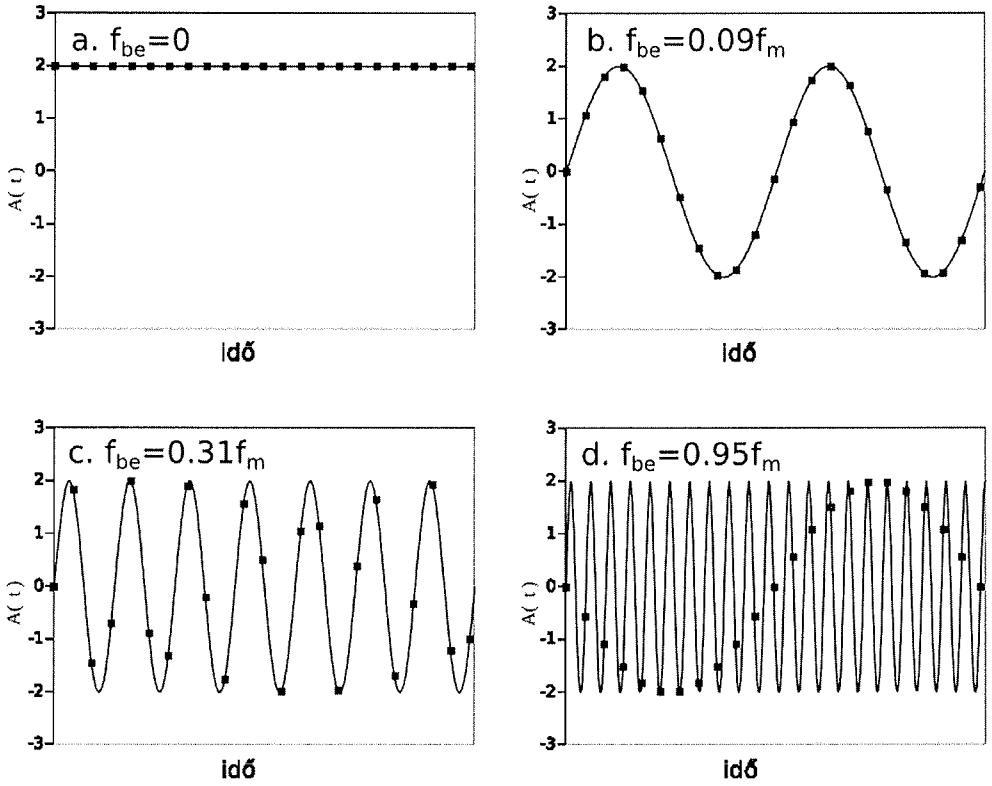
1.11. ábra. Egy egyszerű mintavevő és jelnyújtó (Sample and Hold) áramkör felépítése.

Egy egyszerű mintavevő és jelnyújtó (Sample and Hold) áramkör látható a 1.11 ábrán. A trigger bemeneten keresztül kapcsolható a FET, amely nyitott állapotában az U_{be} feszültségre tölti a C kondenzátort. A jel nyújtását a nagyon nagy bemenő ellenállású, feszültségkövetőként használt műveleti erősítő végzi. A mintavételezési időt a FET ellenállása és a kapacitás RC időállandója határozza meg.

Az ADC szakaszos működése egyben azt is jelenti, hogy a bemenő jel nem változhat a T_m mintavételi időnél gyorsabban. Lassú jelek esetén nincsen probléma:



Gyorsan változó jelek esetén teljesen torzul a mérés:



(a)

1.12. ábra. Az *aliasing* jelensége nem más, mint a lebegés megjelenése a mintavételi folyamatban. A bemenő jelet egyértelműen vissza lehet állítani az *a-c* esetekben. A *d* ábrán sérül a mintavételi törvény, és látható módon a mintavételezett jelek (pontok) az eredeti frekvencia helyett a $f_m - f_{be}$ különbségi frekvenciát adják vissza.

valószínűsége kicsiny volt (gondoljunk csak hírekre: öntudatlanul is átsiklunk a „megszokott" híreken, de felfigyelünk a rendkívüli dolgokra!).

Minden nehézség nélkül értelmezhetjük egy eseményhalmaz megfigyelése során nyert átlagos információt is, amit az adott eseményhalmaz entrópiájának nevezünk (belátható, hogy ez annyira szorosan kapcsolódik a termodinamika és a statisztikus fizika entrópiájához, hogy megegyezik vele, nem kell a két fogalmat megkülönböztetni!):

$$H = -\sum_i p_i \log_2 p_i \tag{1.3}$$

Egy igen/nem döntés akkor jelenti a legnagyobb átlagos információmennyiséget, ha mindkét esemény bekövetkezése ugyanannyira valószínű $p_i = 0.5$, ekkor az átlagos információ

sávszélességet.

Az emberi beszéd (másodpercenként 3 - 8 betű/hang) 40-60 bit/sec hozamú.

A jó minőségű digitális hangrögzítő rendszerek a sztereo mikrofon jeléből másodpercenként legalább 44200-szer vesznek mintát, a mintavételek 16 bit pontosságúak - a fluxus kb. 177 kbyte/sec.

1.6.2. Tömörítési eljárások

Az elektronikus eszközökkel létrehozott adatok (programok, képek, állományok) közül sok tartalmaz részben ismétlődő szakaszokat, részeket. Ezek az adatok általában különböző tömörítési algoritmusokkal (pl. a közismert `pkzip` vagy `arj`) eredeti méretük töredékére zsugoríthatók, így csökkenthető az adatok átviteléhez szükséges idő, redukálható a tárolásához szükséges hely. A tömörítés jobb, ha a kimenet zajszerű, azaz magas az entrópiája.

Az eljárások közül megkülönböztetünk veszteség nélküli és veszteséges eljárásokat: ez utóbbiak ugyan jobban tömörítenek, de az eredeti jelek tökéletes visszaállítása itt nem lehetséges. Veszteséges tömörítés pl. a jpeg vagy az mp3 kódolás: ezeket általában az emberi érzékszervek működéséhez illesztik. A jpeg esetén az élek megőrzésére fektettek hangsúlyt (szemünk nagyon érzékeny erre), míg az mp3 esetén azt használták ki, hogy egy hangos hang mellett fülünk a halk kísérőhangokat nem, vagy csak rosszul tudja észlelni - ezeket felesleges tehát eltárolni.

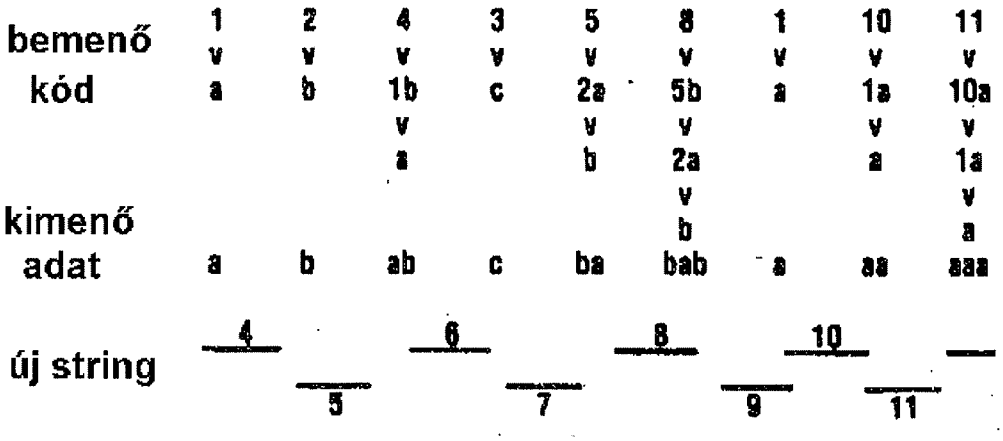
Filózófiai szempontból maga a mérési folyamat is egy veszteséges tömörítési eljárásnak tekinthető: pl. az LHC részecskedetektorainak adatait hatékonyan lehet néhány bitbe tömöríteni (Higgs bozon létezési valószínűsége, tömege).

A veszteségmentes tömörítő eljárások közül legegyszerűbb a futási hossz kód. Ilyenkor pl. a 127 decimális érték alatti karakter jelzi, hogy hányszor kell az utána jövő karaktert megismételni, a 128 feletti karakterek pedig a 128 feletti értékkel megadott számú, egymás után jövő különböző karaktert jeleznek (hasonló eljárást használ pl. a fax is).

Ez az eljárás ugyan egyszerű, de nem mindig hatásos: pl. egy ABABAB... szöveg esetén nem ismeri fel az ismétlődő mintát. Ilyen esetekre az ún. Lempel-Ziv-Welch (LZW) eljárás tekinthető a módszer egyfajta kiterjesztésének: ekkor folyamatosan egy szótár épül fel sorban, dinamikusan a kódolandó adatokból. Minden egyes bejövő adat (pl. byte) a korábbi adatokkal együtt egy új szót hoz létre. Ezt a szótár elemeivel összehasonlítjuk, és a maximális egyezésű szótárelem az új adattal együtt egy újabb szótárelemet hoz létre.

Nézzük a következő példát! A szótárat a 1.13 ábrán láthatjuk, a kódolás pedig a 1.14 ábrán követhető nyomon. A bejövő adatok balról jobbra kerülnek vizsgálatra. Az első karakter a `a`. Mivel nincs hoszabb egyező szó a szótárban, ezért a `ab` szó bekerül a szótárba, 4-es kóddal.

A tömörített kódot a maximális egyezésű szótárelem pozíciója adja, a kimenetre csak az így megadott szótár–kód kerül.



1.15. ábra. Az LZW kódok dekódolása.

A szótár a méretét tipikusan 2^{12}-2^{16} szóra választják. Amikor a szótár megtelik, akkor akkor azt letörlik (pl. byte méret esetén csak az első 256 szó marad meg), majd újra indul az algoritmus.

A dekódoláshoz a bejövő kódok alapján fel kell újra építeni a szótárat. Ez (a kódoláshoz hasonlóan) folyamatosan elvégezhető, azaz az eljárással az adatokat folyamatosan tömöríthetjük ill. állíthatjuk vissza. Több, adatfolyamban is használt program is (pl. `gzip, compress`) részben ezen az algoritmuson alapul.

Példánk dekódolását láthatjuk a 1.15 ábrán. Minden kód rekurzívan helyettesítődik a prefix kódjával (szótárelem kódja) és a követő karakterrel. A végeredmény az eredeti adatfolyam.

Az eljárás hardware segítségével is könnyen megvalósítható (így valós idejű diszk tömörítés hozható létre!).

Bizonyos esetekben, amikor a bejövő jelek gyakoriság eloszlása előzetesen pontosan ismert, a fenti algoritmusoknál esetlegesen jobban tömörítő ún. Huffman-kódolást is használhatunk. Ennek lényege a 1.16 ábrán látható: az eljárás az egyes jeleket (pl. karakterek) előfordulásuk valószínűségében sorbarendezi. Ezek után az algoritmus a két legkisebb valószínűségű jelet helyettesíti egy olyan új jellel, amelynek valószínűsége a két jel valószínűségének összege. Ezután újra rendez, majd megismétli az egész folyamatot egészen addig, amíg csak két jel nem marad.

A kódoláshoz megfordul a rendezési folyamat: pl. a nagyobb valószínűségű jelhez rendeljük a 0-t, míg az 1-et a kisebbhez. Ezek után visszalépünk egyet a rendezésben, és újra 0-t írunk a nagyobb, 1-et a kisebb jel kódja után (prefix kódolás). A visszalépéseket addig ismételjük, amíg minden elemi jelhez hozzá nem rendelünk egy kódot (l. 1.17 ábra). Az eljárás eredményeként a gyakran előforduló jelek kódolása mindössze néhány bittel történik, míg eközben a nagyon ritka jelek akár 10-14 bites kódot is használhatnak. Érdekes megfigyelni, hogy az így előálló kód egyértelműen dekódolható, nem szükséges