ELEKTRONIKAI TECHNIKUS KÉPZÉS

2 0 1 3

DIGITÁLIS ELEKTRONIKA II.

LOGIKAI HÁLÓZATOK

ÖSSZEÁLLÍTOTTA
NAGY LÁSZLÓ
MÉRNÖKTANÁR

Tartalomjegyzék

Alapfogalmak

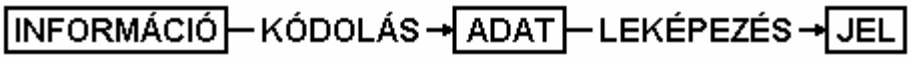
Az információ fogalma alatt a különféle hírek, üzenetek, közlemények tartalmi jelentését, más megfogalmazásban az adatokon végzett gondolati műveletek eredményét értjük. Az információ jellemzője, hogy új ismereteket hordoz, bizonytalanságot szüntet meg, döntésre illetve magatartásunk megváltoztatására késztet, létrehozható és megsemmisíthető. Az információ mennyisége mérhető, mértékegysége a bit, azaz egy eldöntendő kérdésre adható igen-nem válasz. Az adat a különféle események nem értelmezett, de értelmezhető megjelenési formája, kódolt információ. A közlés formai szabályainak összessége a szintaktika, tartalmi szabályainak összessége a szemantika.

A különféle fizikai mennyiségek, például: a hőmérséklet, az erő, az áram stb. értékének számszerű megállapításához, olyan más fizikai mennyiségre való átalakítás szükséges, melyet az ember közvetlenül érzékelni képes. Azt a műveletet mely valamilyen fizikai jellemzőt egy másik fizikai jellemzőre alakít át leképezésnek nevezzük. A két fizikai mennyiség között ismert és kölcsönösen egyértelmű függvénykapcsolatnak kell fennállnia. Az átalakítást valamilyen technikai eszközök, mérőátalakitók, műszerek végzik például: Deprez műszer, higanyos hőmérő, stb. A leképezés lehet:

analóg: a leképező mennyiség folyamatosan követi a leképzendő mennyiség változásait. Az analóg leképezés eredménye az analóg jel, mely egy adott tartományon belül tetszőleges értéket vehet fel. A leképzendő mennyiség felbontása végtelen (pl.: gépkocsi kilométeróra)

digitális: a leképező mennyiség ugrásszerűen követi a leképzendő mennyiség változásait. A digitális leképezés eredménye a digitális jel, mely egy adott tartományon belül csak meghatározott, diszkrét értéket vehet fel. A leképzendő mennyiség felbontása véges (pl.: gépkocsi kilométer számláló).

Az információ, valamely fizikai jellemzőhöz hasonlóan leképezhető.



Valamely információ átalakítását egyezményes jelekké kódolásnak nevezzük. A kód megállapodás szerinti jelek vagy szimbólumok rendszere, mellyel valamely információ egyértelműen megadható. A kód az alkalmazott jelkészleten kívül a hozzárendelés szabályait is tartalmazza. A jelek száma a kód értékét határozza meg, például a 10-es számrendszer 10 értékű kód. A kódszavak lehetnek állandó és változó hosszúságúak.

Digitális technikában alkalmazott számrendszerek

A számrendszerek csoportjai: additív (például római számok)
helyértékes (például a 10-es számrendszer)

A mindennapi életben a tízes számrendszert használjuk.

Egy számot leíró összefüggés általánosan:

$$N_r = \pm \sum_{k=-m}^{k=+j} a_k r^k$$

- N_r egy tetszőleges számrendszerbeli szám;
- r a számrendszer alapszáma (radixa), az egy helyértéken megkülönböztethető számjegyek száma;
- m negatív (tört) helyértékek száma;
- j pozitív (egész) helyértékek száma;
- a_k az egyes helyértékek együtthatói, = 0, 1, 2, 3, ... r-1;
- k hatványkitevő, = 0, ±1, ±2, ...

A digitális technikában a 2-es (bináris), 8-as (oktális), 16-os (hexadecimális) számrendszereket alkalmazzuk. A 16-os számrendszerben a 9 feletti számokat tehát 10-től 15-ig, a tömörebb írásmód érdekében A, B, C, D, E, F betűkkel jelöljük. Egy decimális szám átalakításának lépései „r" alapú számrendszerbe:

Decimális egészrész átalakítása:

1. osztás az alapszámmal, 2. a hányados törtrészének r szerese az r alapú szám 0-ik vagy következő helyértéke, 3. a hányados egészrésze a következő decimális egészrész, 4. ha a decimális egészrész kisebb mint az r, akkor kész az átalakítás, különben vissza az 1. lépéshez.

Decimális törtrész átalakítása: 1. szorzás az alapszámmal, 2. a szorzat egészrésze az r alapú szám −1-ik vagy következő helyértéke, 3. a szorzat törtrésze a következő decimális törtrész, 4. ha a decimális törtrész egyenlő 0-val, akkor kész az átalakítás, különben vissza az 1. lépéshez. A törtrész ismétlődése esetén tetszőleges számú lehet az átalakított tört.

Egy r alapú szám decimális értéke az általános összefüggés alapján határozható meg. Minta átalakítások a mellékletben találhatók.

Digitális technikában alkalmazott kódok

A numerikus kódok csak számokat tartalmaznak. A bináris kód a numerikus információt a bináris számrendszer szabályai szerint - bináris számokkal és bináris helyértékekkel - képezi le. A Binary Coded Decimal vagy röviden a BCD kód a tíz decimális számjegyet négy vagy öt esetleg több bináris elemből alkotott kódszavakkal képezi. Több számjegy esetén a kódszavakat a decimális számszerkezet, azaz a helyértékrendszer megtartásával kell összefűzni. Gyakoribb 4, 5 és 7 bites BCD kódok:

Bináris vagy 8421-es súlyozású kód: a decimális szám bináris megfelelője.

Aiken vagy 2421 súlyozású kód: 0-tól 4-ig bináris, 5-től 9-ig 6-tal több bináris.

Stibitz vagy 3-többletes kód: 0-tól 9-ig 3-mal több bináris.

Gray kód: két szomszédos kódszó csak egy bitben különbözik. Képzése történhet tükrözéssel vagy egy 0 + a szám négybites bináris kódjával bitenként képzett XOR művelettel.

5-ből-2 vagy 74210 súlyozású kód: minden kódszóban 2db. 1-es van. A 0 a decimális 11.

Johnson kód: két szomszédos kódszó csak egy bitben különbözik. Képzése 1-esek és 0-ák

beléptetésével történik. Az Aiken és a Stibitz kódok önkomplementálók. Ez azt jelenti, hogy 4. és 5. kódszavak közötti elhelyezett képzeletbeli szimmetria tengely mentén a táblát összehajtva a 0-ák és 1-esek találkoznak, a kódszavak egymás komplemensei.

	8421	Aiken	Stibitz	Gray	5-ből 2	Johnson	Hamming
0	0000	0000	0011	0000	11000	00000	0000000
1	0001	0001	0100	0001	00011	00001	1101001
2	0010	0010	0101	0011	00101	00011	0101010
3	0011	0011	0110	0010	00110	00111	1000011
4	0100	0100	0111	0110	01001	01111	1001100
5	0101	1011	1000	0111	01010	11111	0100101
6	0110	1100	1001	0101	01100	11110	1100110
7	0111	1101	1010	0100	10001	11100	0001111
8	1000	1110	1011	1100	10010	11000	1110000
9	1001	1111	1100	1101	10100	10000	0011001

Hamming kód: hibajavításra alkalmas. Két kódszó közötti Hamming távolságnak nevezzük azoknak a biteknek a számát, amelyek a két kódszóban különböznek. Egy kód Hamming távolságának nevezzük a kódot alkotó kódszavak közötti legkisebb Hamming távolságot.
Ha a kód Hamming távolsága 1, akkor a kódszóban előforduló hiba nem ismerhető fel és nem javítható. Ha a kód Hamming távolsága ≥ 2, akkor a kódszóban jelenlévő hiba felismerhető (Error Detecting Code) de nem javítható (pl. az 5-ből 2 vagy a paritás elemes kód, melyben a kódszóhoz kapcsolt paritásbit értéke az 1-esek számát párosra vagy páratlanra egészíti ki). Ha a kód Hamming távolsága ≥ 3, akkor a kódszóban előforduló egy hiba felismerhető és a hiba javítható (Error Correcting Code).

	1	2	3	4	5	6	7
E2				Pc	D3	D2	D1
E1		Pb	D4			D2	D1
E0	Pa		D4		D3		D1

A Hamming kód képzése:
1,2,3,4,5,6,7 a bitek sorszáma;
D4,D3,D2,D1 8421 súlyozású adatbitek;
Pa,Pb,Pc az adatbiteket páros paritásra kiegészítő bitek;
E2,E1,E0 súlyozott bitcsoportok.
Hibajavítás: ha a bitcsoportok képzésekor páratlan paritás keletkezik, akkor a bitcsoportok súlyozott összege a hibás bit sorszámát adja meg, ami invertálással javítható.
Az alfanumerikus információt leíró jeleket karaktereknek nevezzük. A karakterek lehetnek: kis és nagybetűk, decimális számjegyek, írásjelek, műveleti jelek és vezérlő karakterek. TELEX 5 bites kód: betű ill. számváltó karakterrel összesen 62 jelet tartalmaz. ASCII (Amerikai Szabványos Kód Információ Átalakítására) 7 bites kód: 127 jelből 32 vezérlő a többi betű, szám és írásjel. ASCII 8 bites kód: 255 jelben grafikus, keretrajzoló és más szimbólumok is vannak. Néhány karakter cserélhető (kódlapok). EBCDIC (Bővített Binárisan Kódolt Decimális Átváltó Kód) 8 bites kód: lyukkártyás rendszerekben alkalmazott Hollerith kódból fejlődött ki, IBM nagyszámítógépeknél volt használatos.

A logikai függvény fogalma, csoportjai

A formális logika a helyes gondolkodás alaki törvényszerűségeivel, a helyes következtetéssel és a következtetésekkel kapcsolatos bizonyításokkal foglalkozó tudomány. A következtetések folyamán alapfeltételekből indulunk ki és ítéletek alkotásával jutunk el a következtetésig. Mind az ítélet, mind a belőlük levont következtetés lehet igaz vagy hamis. Megállapodás szerint az igaz ítélet logikai értéke "1", hamis ítélet logikai értéke "0". A ítéletet jelentő mondatokhoz - melyek állítást vagy tagadást fejezhetnek ki - az ABC-ből választott betűvel logikai változót rendelünk. Az állító mondatot jelölés nélküli - vagy ponált - (pl. A), ugyanezen állítást tagadó mondatot felülvont - vagy negált - ($\bar{A}$) betűvel jelöljük. A logikai változókkal különböző műveletek végezhetők. Ezen műveletek eredménye a logikai változók értékétől valamint a közöttük levő kapcsolattól függ. A kapcsolatok logikai függvényekkel írhatók le. A logikai függvény a független változónak tekintett események hatására kialakuló, függő változónak tekintett következmények közötti kapcsolatot fejezi ki.

A logikai függvények csoportosítása:

a változók száma szerint: egy, kettő és több változós. A többváltozós függvények visszavezethetők kétváltozós függvényekre.

a változók időbeli függése szerint: időfüggetlen vagy kombinációs hálózat:
a függő változók értékei, csak a független változók pillanatnyi értékétől függenek.

időfüggő vagy szekvenciális hálózat:
a függő változók értéke a független változók egy adott idő pillanatban felvett értékétől és a függő változók korábbi állapotától függenek.

A logikai függvény megadási módjai

- Szövegesen: mikor, mi történik, ha ..., akkor ..., stb.
- Igazságtáblázattal: a logikai változók által felvehető értékek összes kombinációjának és az ezekhez tartozó függő változó(k) értékének táblázatba foglalt formája.

AB	X	Y
00	1	0
01	1	1
10	0	0
11	1	0

- Algebrai alakban: a matematikai kifejezésekhez hasonlóan, az eseményekhez hozzárendelt logikai változókkal fejezzük ki a feltétel rendszert. Az "és" műveletet a szorzás, a "vagy" műveletet az összeadás jele jelképezi. A logikai változók azonos műveleti jellel

összekapcsolt csoportja a "terminus" vagy rövidem term, melyben minden változó, ponált vagy negált formában egyszer előfordul. A logikai változók "és" művelettel összekapcsolt csoportját minimális termnek vagy egyszerűen mintermnek, a "vagy" művelettel összekapcsolt csoportját maximális termnek vagy maxtermnek nevezzük. A logikai függvény -a dualitásból következően- lehet diszjunktív vagy konjunktív normál alakú. Normál alakúnak nevezzük azt a függvényt, melyben minden termben, minden változó, ponált vagy negált formában egyszer előfordul. A diszjunktív normál alakot a mintermek "vagy" kapcsolatai, a konjunktív normál alakot a maxtermek "és" kapcsolatai alkotják. (az "és" művelet jelét általában elhagyjuk!)

Diszjunktív normál alak: $X^2 = \overline{A}\,\overline{B} + \overline{A}\,B + A\,B$ $Y^2 = \overline{A}\,B$

Konjunktív normál alak: $X^2 = \overline{A} + B$ $Y^2 = (A + B)(\overline{A} + B)(\overline{A} + \overline{B})$

- Decimális alakban: a normál alakú függvények egyszerűsített megadási módja, melyben a mintermek ill. a maxtermek sorszámai vannak felsorolva.

  Diszjunktív sorszámos alak: $X^2 = \Sigma^2 (0, 1, 3)$; $Y^2 = \Sigma^2 (1)$

  Konjunktív sorszámos alak: $X^2 = \Pi^2 (1)$; $Y^2 = \Pi^2 (0, 1, 3)$
- Grafikusan: a Veitch és a Karnaugh tábla alapvetően egy koordináta rendszer, melynek tengelyein a logikai változók kombinációi Gray kódban, a képződő cellákban a függő változó értékei vannak feltüntetve. A két táblázat csak a jelölés módjában tér el egymástól. Mindkét tábla alkalmas a mintermes vagy a maxtermes függvények ábrázolására. A V-K táblák a logikai függvények minimalizálásának szemléletes eszközei.
- Állapot átmeneti tábla: a szekvenciális hálózatok igazságtáblázata, melyben a bemenő változók n-ik időpillanatban fennálló értékeinek függvényében mutatja be a kimenő változók n+1-ik időpillanatban létrejövő értékeit.
- Állapot diagram: a szekvenciális hálózat kimenő változóinak különböző időpillanatban létrejövő értékei vannak körökben feltüntetve. Az egyes állapotok sorrendjét a köröket összekötő irányított vonalak jelzik.
- Ütemdiagram: egy logikai változó értékét mutatja az idő vagy egy órajel függvényében.
- Logikai vázlat: a logikai függvények rajzjeleinek összerajzolt formája.

Logikai alapfüggvények

A lehetséges függvénykapcsolatok száma a független változók számától függ:

független változók száma	1	2	3	4
kombinációk száma	2	4	8	16
függvénykapcsolatok száma	4	16	256	65536

Ha a független változók száma: n

akkor a kombinációk száma: 2^n

és a függvénykapcsolatok száma: 2^{2^n}

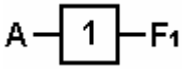
Egyváltozós logikai függvények

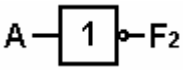
Egy logikai változó két értéket vehet fel, a lehetséges válaszok száma négy.

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

A függvények algebrai alakja, megnevezése és rajzjele:

F0 = 0 soha földpont vagy GND

F1 = A ismétlő 

F2 = $\bar{A}$ negáló 

F3 = 1 mindig tápfeszültség vagy Vcc

A kimeneten a kis kör a negálást jelzi. A rajzjelekre vonatkozó szabvány megengedi a bemeneti negálást is. Az ismétlő függvénynek a meghajtó áramköröknél van jelentősége.

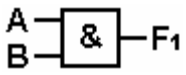
Kétváltozós logikai függvények

Két logikai változónak négy kombinációja van, a lehetséges válaszok száma tizenhat.

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

A függvények algebrai alakja, megnevezése és rajzjele:

$F_0 = 0$ soha

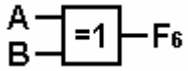
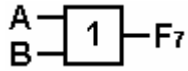
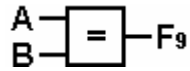
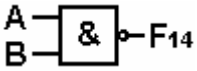
$F_1 = A \cdot B = (A + B) \cdot (A + \overline{B}) \cdot (\overline{A} + B)$ ÉS, AND, konjunkció 

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

$F_3 = A$ ismétlő

$F_4 = \overline{A} \cdot B = (A + B) \cdot (\overline{A} + B) \cdot (\overline{A} + \overline{B})$ inverz inhibíció

$F_5 = B$ ismétlő

$F_6 = \overline{A}B + A\overline{B} = (A + B)(\overline{A} + \overline{B})$	ANTIVALENCIA, XOR, kizáró vagy	
$F_7 = \overline{A}B + A\overline{B} + AB = A + B$	VAGY, OR, diszjunkció	
$F_8 = \overline{A}\,\overline{B} = (A + \overline{B})(\overline{A} + B)(\overline{A} + \overline{B}) = \overline{A + B}$	NEMVAGY, NOR, sem	
$F_9 = \overline{A}\,\overline{B} + AB = (A + \overline{B})(\overline{A} + B) =$	EKVIVALENCIA, megengedő és	
$F_{10} = \overline{B}$	negáló	
$F_{11} = \overline{A}\,\overline{B} + A\overline{B} + AB = A + \overline{B}$	inverz implikáció	
$F_{12} = \overline{A}$	negáló	
$F_{13} = \overline{A}\,\overline{B} + \overline{A}B + AB = \overline{A} + B$	implikáció	
$F_{14} = \overline{A}\,\overline{B} + \overline{A}B + A\overline{B} = \overline{A} + \overline{B} = \overline{AB}$	NEMÉS, NAND	
$F_{15} = 1$	mindig	

Az F_6 és F_9 függvényekhez külön műveleti jel is tartozik, az antivalencia jele a "$\oplus$ ", és az ekvivalencia jele a "$\odot$ ". Az A antivalens B algebrai alakja: $A \oplus B$, és az A ekvivalens B algebrai alakja az $A \odot B$.

A kétváltozós függvények közül az ÉS, NEMÉS, VAGY, NEMVAGY, ANTIVALENCIA és az EKVIVALENCIA függvényeknek van gyakorlati jelentősége, mivel a többi vagy egy változós ismétlő vagy negáló illetve a felsorolt hat valamelyikével előállítható.

Függvények értelmezése több, tehát 3, 4, 5, stb. változóra: A kétváltozós ÉS és NEMVAGY függvényben csak 1db. 1-es, a VAGY és NEMÉS függvényekben csak 1 db. 0-ás van. Ezek a tulajdonságok a logikai működés alapján kettőnél több bemenő változó esetén is igazak, tehát az ÉS a VAGY a NEMÉS és a NEMVAGY függvényeket több változóra is ugyanúgy értelmezzük mint két változó esetén. Az ANTIVALENCIA és az EKVIVALENCIA függvények kétváltozósra való visszavezetése helytelen eredményt ad. A többváltozós ANTIVALENCIA függvény akkor igaz, ha páratlan számú bemenő változó igaz (1 értékű). A többváltozós ekvivalencia függvény akkor igaz, ha páros számú bemenő változó igaz (1 értékű).

Logikai algebra szabályai, azonosságai

A matematikai algebrával összehasonlítva a logikai algebrában három eltérés van:

- csak három művelet, az ÉS a VAGY és a NEGÁLÁS létezik,
- az ÉS és a VAGY művelet azonos prioritású,
- egy változó többször is felhasználható.

Műveletek 0-val és 1-el:

$0+0=0$	$0+1=1$	$1+1=1$
$0\cdot 0=0$	$0\cdot 1=0$	$1\cdot 1=1$
$0\oplus 0=0$	$0\oplus 1=1$	$1\oplus 1=0$
$0\odot 0=1$	$0\odot 1=0$	$1\odot 1=1$

Műveletek egy változóval és egy állandóval:

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

Műveletek egy változóval:

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

Felcserélhetőségi (kommutatív) szabály:

$$A+B=B+A \qquad A\cdot B=B\cdot A$$

Társítási (asszociatív) szabály:

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

Kiemelési (disztributív) szabály:

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

Beolvasztási (abszorpciós) szabály:

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

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

De Morgan szabály:

$$\overline{A + B} = \bar{A} \cdot \bar{B} \qquad \overline{A \cdot B} = \bar{A} + \bar{B}$$

Két logikai változó VAGY műveletének negáltja, a két változó negáltjának ÉS műveletével azonos.

Két logikai változó ÉS műveletének negáltja, a két változó negáltjának VAGY műveletével azonos.

Az azonosságok több változóra, illetve több termre is alkalmazhatók:

$$\overline{A + B + C} = \bar{A} \cdot \bar{B} \cdot \bar{C} \qquad \overline{A \cdot B \cdot C} = \bar{A} + \bar{B} + \bar{C}$$

Mintermes függvény átalakítása: (második lépésben csak NAND -et tartalmaz !)

$$F^3 = X\bar{Y}Z + XY\bar{Z} = \overline{\overline{X\bar{Y}Z} + XY\bar{Z}} = \overline{\overline{X\bar{Y}Z}}\cdot\overline{\overline{XY\bar{Z}}} = \overline{(\bar{X} + Y + \bar{Z})\cdot(\bar{X} + \bar{Y} + Z)}$$

Maxtermes függvény átalakítása: (második lépésben csak NOR -t tartalmaz !)

$$F^3=(\overline{X}+Y+Z)\cdot(X+\overline{Y}+Z)=\overline{\overline{(\overline{X}+Y+Z)\cdot(X+\overline{Y}+Z)}}=\overline{\overline{\overline{X}+Y+Z}+\overline{X+\overline{Y}+Z}}=\overline{\overline{X}\,\overline{Y}\,\overline{Z}+\overline{X}\,\overline{Y}\,\overline{Z}}$$

Az átalakítás lépései mindkét háromváltozós függvény esetén:

- a függvény kétszeres felülvonása;
- az egyik felülvonást elviszi a De Morgan, a termek negálódnak és a termek közötti műveletek átfordulnak;
- a termek felülvonását elviszi a De Morgan, a változók negálódnak és a változók közötti művelet átfordul.

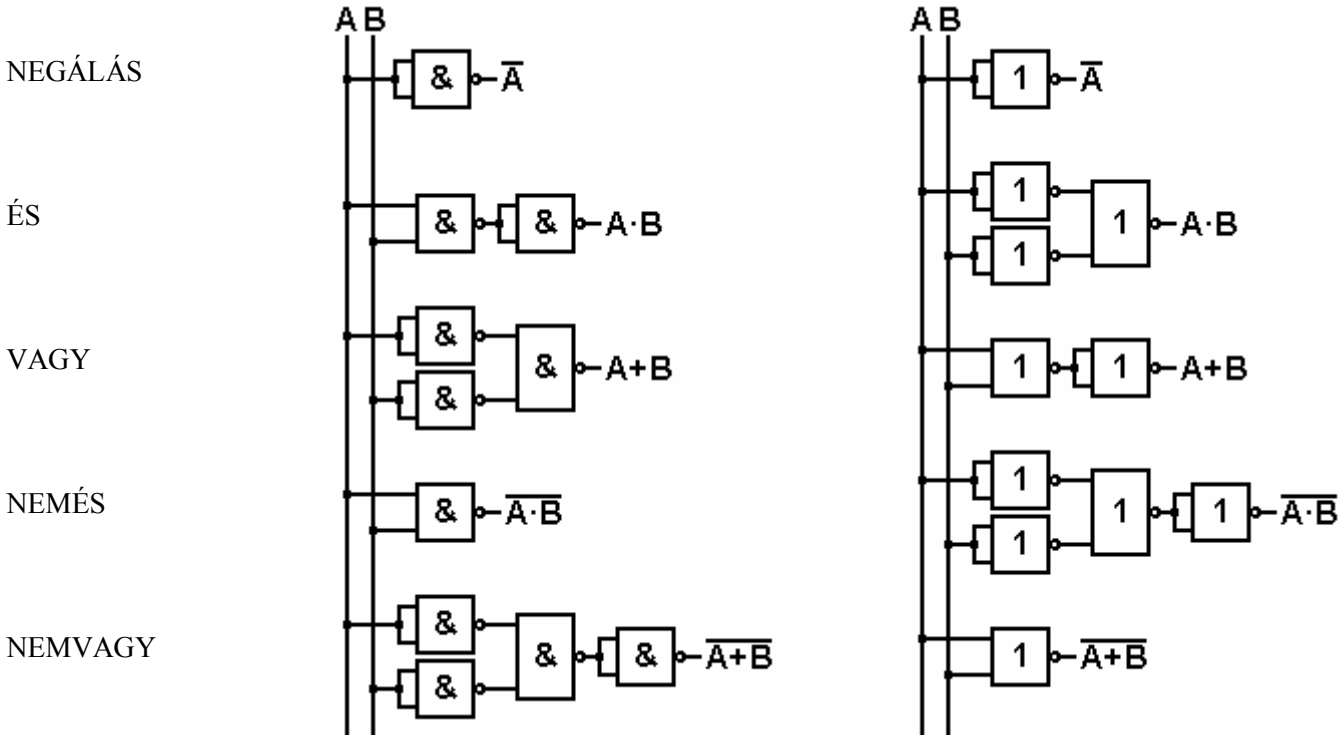
Megjegyzés: a felülvonás zárójelként működik!

Funkcionálisan teljes rendszerek

Funkcionálisan teljes rendszernek nevezzük a kapuk azon csoportját, melyekkel tetszőleges logikai függvény előállítható. Funkcionálisan teljes rendszert alkot:

a NEM az ÉS és a VAGY kapuk együtt (NÉV rendszer),
a NEMÉS kapuk önállóan (NAND rendszer),
a NEMVAGY kapuk önállóan, (NOR rendszer).

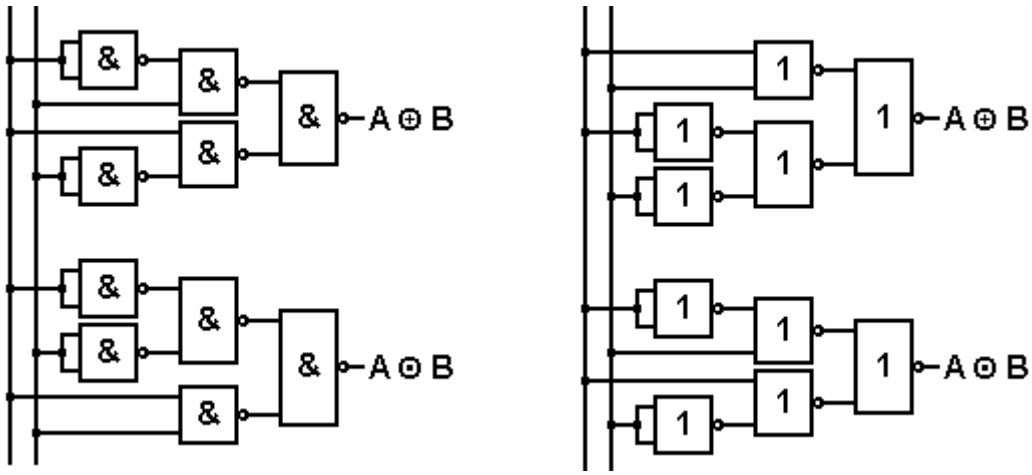
A logikai alapfüggvények megvalósítása NEMÉS és NEMVAGY univerzális kapukkal:



Ha két bemenetű ÉS kapu egyik bemenetét 1-re, illetve a két bemenetű VAGY kapu egyik bemenetét 0-ra kötjük, akkor a másik bemenetére nézve ISMÉTLŐ művelet keletkezik.
Ha két bemenetű NEMÉS kapu egyik bemenetét 1-re, illetve a két bemenetű NEMVAGY kapu egyik bemenetét 0-ra kötjük, akkor a másik bemenetére nézve NEGÁLÓ művelet keletkezik.

ANTIVALENCIA

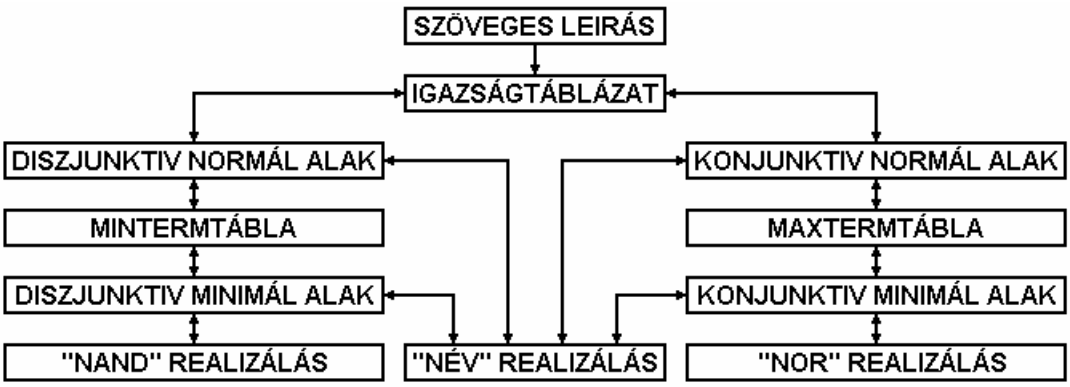
EKVIVALENCIA



Ha a két bemenetű ANTIVALENCIA kapu egyik bemenetét 0-ra kötjük, akkor a másik bemenetre nézve ISMÉTLŐ, ha 1-re kötjük, akkor NEGÁLÓ művelet keletkezik.
A funkcionálisan teljes rendszerek közötti átalakítás a De Morgan tétel alkalmazásával hajtható végre. Az alapfüggvényeket a NEMÉS rendszerbe a diszjunktív alak alapján, a NEMVAGY rendszerbe a konjunktív alak alapján lehet átalakítani. Dualitás: ha egy logikai függvényben az ÉS műveletet VAGY művelettel, a 0-kat 1-el helyettesítjük és viszont, akkor az eredeti függvény duál függvényét kapjuk. Ha egy logikai függvény egy esemény bekövetkezését állítja, akkora duál függvénye minden más esemény bekövetkezését tagadja.

Kombinációs hálózatok tervezésének és elemzésének folyamata

Kombinációs hálózatok tervezésének és elemzésének folyamat ábrája:



A tervezés alapja a szöveges megadás. A leírásban megadott technológiai eseményekhez logikai változókat kell rendelni. Az események közötti kapcsolatokból fel kell állítani az igazságtáblázatot. A táblázatban meg kell jelölni azokat a termeket, melyek üzemszerű működés esetén nem jönnek vagy nem jöhetnek létre. Ilyen akkor fordul elő, ha a változókhoz egymást kizáró események vannak rendelve, például egy motor nem foroghat egyidejűleg két irányba vagy egy memória cellát nem lehet egyidejűleg beírni és kiolvasni. Ezeket a határozatlan termeket a minimalizálás során lehet felhasználni. Az egyes függő változókra a

normál alakú függvényeket az igazságtábla alapján lehet felírni. A diszjunktív normál alakú függvényt a változók ponált értékéből képzett azon mintermek állítják elő, ahol a függvény igaz. A konjunktív normál alakú függvényt a változók negált értékéből képzett azon maxtermek állítják elő ahol a függvény hamis.

Diszjunktív alak: $X^2=\overline{A}\,\overline{B}+\overline{A}B+AB$

$Y^2=\overline{A}B$

Konjunktív alak: $X^2=\overline{A}+B$

$Y^2=(A+B)(\overline{A}+B)(\overline{A}+\overline{B})$

A B	X	Y
0 0	1	0
0 1	1	1
1 0	0	0
1 1	1	0

A diszjunktív normál alakú függvény ismeretében a konjunktív alak, illetve a konjunktív normál alak ismeretében a diszjunktív alak is előállítható. Ehhez első lépésben a hiányzó termekből álló függvényt kell meghatározni, ami az eredeti függvény negáltja, majd ezt a negált függvényt még egyszer negálva és a De Morgan azonosságot felhasználva áll elő a duál függvény, ami az eredetivel teljesen azonos.
Diszjunktív alakból hiányzó mintermek, tehát a függvény negáltja:

$$\overline{X^2}=A\overline{B} \qquad \overline{Y^2}=\overline{A}\,\overline{B}+A\overline{B}+AB$$

A negált függvény második negálása az eredeti függvény duálja:

$$\overline{\overline{X^2}}=\overline{A\overline{B}} \qquad \overline{\overline{Y^2}}=\overline{\overline{A}\,\overline{B}+A\overline{B}+AB}$$

A De Morgan azonosságot felhasználva:

$$X^2=\overline{A}+B \qquad Y^2=\overline{\overline{A}\,\overline{B}}\cdot\overline{A\overline{B}}\cdot\overline{AB}$$

$$Y^2=(A+B)(\overline{A}+B)(\overline{A}+\overline{B})$$

Az egyes függő változókra a decimális alakú függvényeket szintén az igazságtábla alapján lehet felírni. A diszjunktív decimális alakú függvényben azon mintermek sorszámai vannak feltüntetve, ahol a függvény értéke igaz. A mintermek sorszáma a változók ponált értékéből képzett bináris kód decimális értéke. A konjunktív decimális alakú függvényben azon maxtermek sorszámai vannak feltüntetve, ahol a függvény értéke hamis. A maxtermek sorszáma a változók negált értékéből képzett bináris kód decimális értéke. Ha az igazságtáblázatban a változók által felvehető értékek kombinációi a bináris kód természetes növekvő sorrendjében vannak kitöltve, akkor a mintermek 0-tól 2^n-1-ig tartó sorszámokat, a maxtermek 2^n-1-től 0-ig tartó sorszámokat veszik fel. Az összefüggésben n a változók száma! A diszjunktív sorszámos alak ismeretében a konjunktív alak, illetve a konjunktív sorszámos alak ismeretében a diszjunktív alak is meghatározható. Ehhez első

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

lépésben a függvény negáltját jelentő, hiányzó sorszámokat kell megkeresni, majd ezeket a legnagyobb ábrázolhatóból, tehát 2^n-1-ből kivonva kapjuk meg a duál alakot. Az átalakítást a J és K függvények mutatják be.

A diszjunktív sorszámos alak: $J = \Sigma^3(1,2,4,6,7)$
A hiányzó sorszámok: $\overline{J} = \Sigma^3(0,3,5)$
A konjunktív sorszámos alak: $J = \Pi^3(7,4,2)$

A konjunktív sorszámos alak: $K = \Pi^4(3,5,9,10,11,13,14,15)$
A hiányzó sorszámok: $\overline{K} = \Pi^4(0,1,2,4,6,7,8,12)$
A diszjunktív sorszámos alak: $K = \Sigma^4(15,14,13,11,9,8,7,3)$

Az algebrai alak ismeretében a decimális alak közvetlenül is előállítható. A termek sorszámai a változók bináris kódnak értelmezett értékéből állapíthatók meg. Az átalakítás feltétele, hogy minden termben és az igazságtáblázatban azonos legyen a változók sorrendje. A decimális alak ismeretében az algebrai alak közvetlenül is előállítható. A termekben szereplő változók értékét a sorszám bináris megfelelője szerint kell figyelembe venni. A változók a bináris megfelelő azonos helyérték szerinti 0 értékénél negáltak, egyébként ponáltak. Az algebrai és decimális áttérést a G és H háromváltozós függvények mutatják be.

$$G^3 = \overline{A}\overline{B}C + ABC + ABC \Leftrightarrow G = \Sigma^3(1,2,7)$$

$$H^3 = (\overline{A} + B + C)(A + \overline{B} + \overline{C})(A + B + \overline{C}) \Leftrightarrow H = \Pi^3(3,4,6)$$

Logikai függvények egyszerűsítése

A normál alakok birtokában a függvény NEM ÉS VAGY, NAND illetve NOR rendszerben már megvalósítható. Gazdaságossági és műszaki szempontból célszerűbb a kevesebb termet tartalmazó függvényt megvalósítani. Ez azt jelenti, hogy ha a függvényben kevesebb az 1, akkor diszjunktív alakot, ha kevesebb a 0, akkor a konjunktív alakot kell felhasználni. Előfordulhat, hogy a normál alakú függvénynek létezik egy egyszerűbb, kevesebb termből vagy változóból álló ún. minimál ekvivalense, melynek működése azonos az eredeti függvénnyel. Egy logikai függvény legegyszerűbb alakjának megkeresésére irányuló eljárást minimalizálásnak nevezzük. A minimál alak alapján elkészített áramkör kevesebb alkatrészből áll, kisebb a tömege és a térfogata, kisebb az energiaigénye és kisebb az építési költsége. Az egyszerűbb alak megkeresésére spekulatív és szisztematikus eljárások léteznek. A spekulatív úton való minimál alak megkeresése a „józan paraszti ész” elvén azaz az átlátható eseményrendszer nyilvánvaló egyszerűsítésének felismerésén alapul, például: ha esik, ha nem, focimeccsre megyek, tehát a focimeccsre menés ténye nem függ az időjárástól Az eljárás eredménye összetettebb kapcsolatrendszer esetén nem megbízható. Az algebrai egy-

szerűsítés alapja az abszorpciós szabály, mely szerint, ha két term csak egy logikai változó értékében tér el egymástól, akkor függvény ettől a változótól nem függ és két term a közös változókkal egyesíthető.
Abszorpciós szabály alkalmazása két változóra:

$$P^2 = \overline{A}B + AB = (\overline{A} + A)B = 1 \cdot B = B$$
$$Q^2 = (A + \overline{B})(A + B) = A + \overline{B}B = A + 0 = A$$

Abszorpciós szabály alkalmazása három változóra:

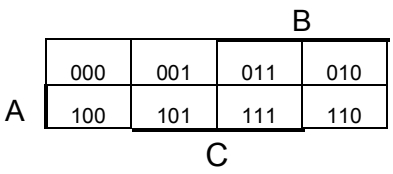
$$X^3 = \overline{A}\,\overline{B}C + \overline{A}BC = \overline{A}C(\overline{B} + B) = \overline{A}C \cdot 1 = \overline{A}C$$
$$Y^3 = (\overline{A} + B + \overline{C})(A + B + \overline{C}) = B + \overline{C} + \overline{A}A = B + \overline{C} + 0 = B + \overline{C}$$

A beolvasztási szabály négy term esetén is használható, ha a négy termben két változó értékének összes kombinációja szerepel.

$$J^3 = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C} + ABC = B$$
$$K^3 = (\overline{A} + \overline{B} + C)(\overline{A} + B + C)(A + \overline{B} + C)(A + B + C) = C$$

Az azonosság alkalmazásából következik, hogy kettő hatványa szerinti darabszámú tehát 2, 4 , 8, 16, stb. term összevonható, ha azokban 1, 2, 3, 4, stb. változó összes kombinációja előfordul. Az algebrai úton való minimalizálás nagy figyelmet igényel és a legegyszerűbb alak megtalálása sem biztos.
A grafikus egyszerűsítés eszköze a Karnaugh tábla vagy a Veitch diagram. Mindkét tábla 2^n darab cellából áll, ahol az n a változók száma. Az egyes cellák egy-egy termnek felelnek meg. A két tábla csak a változók feltüntetésének módjában tér el. Karnaugh tábla esetén a kontúrokon a sorokhoz és oszlopokhoz tartozó változók értékei vannak feltüntetve binárisan, Gray kód szerinti sorrendben. Veitch diagram esetén az egyes változók igaz értékéhez tartozó sorok és oszlopok vannak megjelölve. A cellákba a függő változó értéke kerül. A szakirodalom a diszjunktív függvényekhez a minterm tábla, a konjunktív függvényekhez a maxterm tábla használatát írja le. A kettő között csak a független és a függő változók értékében van eltérés. A minterm táblában a változók ponált, a maxterm táblában negált értékkel szerepelnek. Az ábra a háromváltozós Karnaugh tábla és a háromváltozós Veitch diagram mintermes formáját mutatja be a cellákhoz tartozó mintermekkel.

A \ BC	00	01	11	10
0	000	001	011	010
1	100	101	111	110



A grafikus eszköz jellegzetessége, hogy az egymás melletti cellák termjei csak egy változó értékében térnek el egymástól, így a beolvasztási szabály alkalmazhatóságát grafikusan - azaz elhelyezkedés alapján - lehet felismerni. A minimalizálás tehát 2, 4, 8, stb. egymás

melletti termek összevonásából, tömbösítéséből áll. Az összevonás a tömbben lévő ponált és negált értékkel szereplő változók elhagyását illetve a közös változók feltüntetését jelenti. A cél, a lehető legkevesebb tömbbel az összes 1 értékű cella lefedése. A minimalizált függvényhez egy cella többször is felhasználható és minden cellának szerepelnie kell.

Az összevonás általános szabályai:

- két egymás melletti cella összevonható,
- sorok és oszlopok végein lévő cellák összevonhatók,
- négy egymás melletti cella összevonható,
- teljes sorok és oszlopok cellái összevonhatók,
- négy sarokban lévő cella összevonható,
- nyolc egymás melletti cella összevonható,
- teljes tábla összevonható ha minden cellában azonos érték van.

A Z négyváltozós függvény minimalizálása, minterm és maxterm tábla használata:

A B C D	Z	Y
0 0 0 0	0	0
0 0 0 1	0	1
0 0 1 0	1	0
0 0 1 1	1	1
0 1 0 0	0	0
0 1 0 1	0	0
0 1 1 0	0	0
0 1 1 1	1	1
1 0 0 0	0	0
1 0 0 1	1	0
1 0 1 0	0	1
1 0 1 1	1	1
1 1 0 0	1	0
1 1 0 1	1	0
1 1 1 0	0	1
1 1 1 1	1	1

AB \ CD	00	01	11	10
00			1	1
01			1	
11	1	1	1	
10		1	1	

$$Z^4 = \overline{A}\overline{B}C + AB\overline{C} + AD + CD$$

AB \ CD	11	10	00	01
11	1	1		
10	1	1		1
00				1
01	1			1

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

A felső táblában az összevonás eredménye négy hiányos mintermből álló diszjunktív, az alsó táblában az összevonás eredménye három hiányos maxtermből álló konjunktív függvény. A két függvény teljesen azonos egymással. Ezt úgy lehet bizonyítani, hogy a mintermes alak áll elő, a maxtermes alak kijelölt műveleteinek elvégzésével. A változók számától függetlenül, a minterm és a maxterm tábla a bal felső cella értéke alapján különböztethető meg. Ennek a cellának az értéke minterm tábla esetén 0, maxterm tábla esetén 15. A két táblában a többi cella értéke 2^n-1-re egészítik ki egymást. A legfelső sort tekintve, a mintermes cellák értéke 0, 1, 3, 2 a maxtermes cellák értéke 15, 14, 12, 13. A minterm táblában a 0 értékű független és függő változók a maxterm táblában 1 értékkel szerepelnek

és viszont. Ennek figyelembe vételével, létrehozható egy olyan tábla, amely alkalmas a akár a mintermes, akár a maxtermes minimál függvény előállítására. Ehhez mindössze a minterm táblát kell a független változók maxtermes étékeivel egy második, külső kontúron kiegészíteni. Az előzőkhöz viszonyítva eltérés, hogy a maxtermes minimál alakot a függő változók 0 értékeinek összevonásából lehet előállítani. Az Y függvény minimalizálásának első lépése a függő változók értékének cellákba írása, minterm táblának megfelelően. A diszjunktív minimál alakot a függő változók 1 értékének összevonásával a független változókat a belső kontúrról leolvasva kapjuk meg. A konjunktív minimál alakot a függő változók 0 értékének összevonásával a független változókat a külső kontúrról leolvasva kapjuk meg.

Mintermes alak:

$$Y^4 = \overline{A}\,\overline{B}D + CD + AC$$

Maxtermes alak:

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

AB \ CD		**11**	**10**	**00**	**01**	MAX.
		00	01	11	10	min.
11	00	0	1	1	0	
10	01	0	0	1	0	
00	11	0	0	1	1	
01	10	0	0	1	1	
MAX.	min.					

Előfordulhat, hogy a függvény olyan termet is tartalmaz, amely nem vonható össze semmivel sem. Ilyenkor ezt a termet fel kell tüntetni a függvényben, mivel annak része.

Minimalizálás határozatlan termek felhasználásával

Egy logikai függvény megvalósításakor előfordulhat, hogy a független változók értékének egy vagy több kombinációja normális, üzemszerű működés során nem jöhet létre. Ezeket a termeket nevezzük határozatlannak. A határozatlan termek akkor jelenhetnek meg, ha a független változókhoz egymást kizáró események vannak rendelve. Ebben az esetben a függő változó értéke nem határozható meg, tehát határozatlan. Az egyszerűsítés során a határozatlan termeket bármilyen értékkel figyelembe lehet venni.

Példák egyszerűsítésre határozatlan termekkel:

AB \ CD	00	01	11	10
00		1	h	1
01	1	1	h	
11	h	1	h	
10		h	1	1

$$Q1^4 = ABC + B\overline{C} + D$$

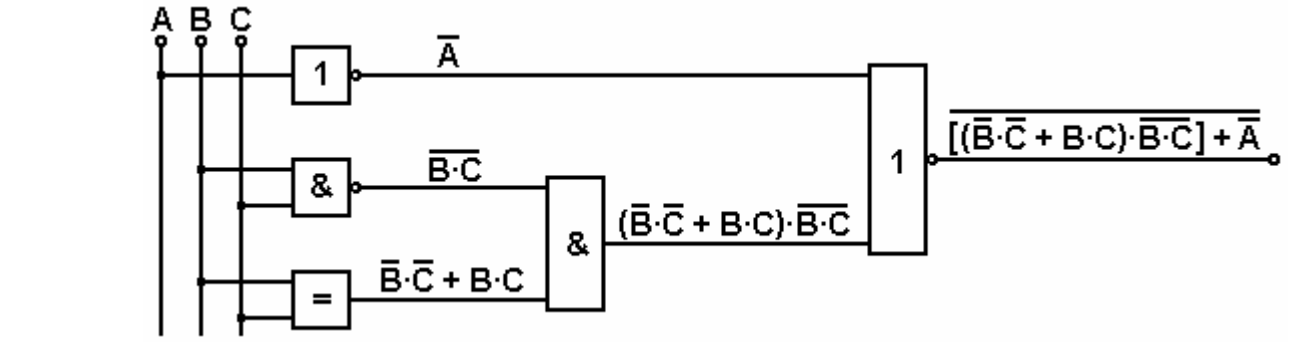
AB \ CD	11	10	00	01
11	h	1	1	
10	1	h	h	
00	1		1	
01	1			

$$Q2^4 = (A + C)(C + D)(\overline{B} + \overline{C} + \overline{D})$$

A határozatlan termeket csak értékes termekkel lehet összevonni. Egyesített táblában is felhasználhatók.

Kombinációs hálózatok elemzése

Kombinációs hálózatok elemzésére akkor van szükség, ha egy kész hálózat működését kívánjuk megismerni vagy esetleg át kell tervezni más kapuáramkörök felhasználásával. Előfordulhat, hogy a helyes működés ellenőrzéséhez vagy hibakereséshez, javításhoz kell az áramkör működését megismerni. Az elemzés alapja minden esetben egy logikai vázlat, azaz egy kapcsolási rajz. A hálózat működésének megismeréséhez a realizált algebrai alakra van szükség. A kapcsolási rajzból az algebrai alakot többféle módon lehet visszafejteni. Az egyik lehetőség, hogy a kapuk kimenetéhez feljegyezzük a bemeneti változók közötti függvénykapcsolatot. Ezt mutatja be a mellékelt ábra.



A hálózat kimenetén adódó, általában hiányos termekből álló függvényt, az algebrai azonosságok felhasználásával normál alakúra hozzuk. A bemutatott hálózat esetén:

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

Az algebrai alak felderítésének egy másik módja, hogy a kimenetről a bemeneti változók felé haladva, faág szerűen végigjárjuk az összeköttetéseket és feljegyezzük a jelzett műveleteket és végül a változókat. Ezt a módszert a mellékelt ábrára alkalmazva, elsőként rajzolunk egy hosszú negálást egy VAGY művelettel. A VAGY művelet baloldalára rajzolunk egy negálást (az inverterét) és alá írjuk az A változót. Mivel elértünk a bemeneti változókat, visszahaladunk a NEMVAGY kapuig és most az alsó ágon folytatjuk a visszafejtést. Az első VAGY művelet jobboldalára feltüntetünk egy ÉS műveletet, majd szintén a felső ágon haladunk tovább. Rajzolunk egy invertálást egy ÉS művelettel, beírjuk hozzá a két bemeneti változót, majd visszahaladunk az utolsó elágazásig és az ÉS kapu alsó bemenetén haladunk tovább. Következik egy VAGY és két ÉS művelet, majd a változók beírása. Mivel több elágazás nincs ezért kész a függvény visszafejtése, már csak rendezett formába kell átírni. A visszafejtés során figyelni kell az egyes műveletek zárójelezésére is és a műveleti jelek közötti elegendő helyre. A normál alakot az előzővel megegyező módon hozzuk létre.

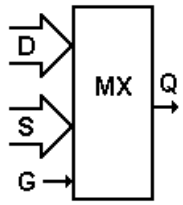
Kombinációs hálózatokra épülő funkcionális egységek

A funkcionális egységeknek olyan logikai hálózatok, melyek egy jól meghatározott felada-

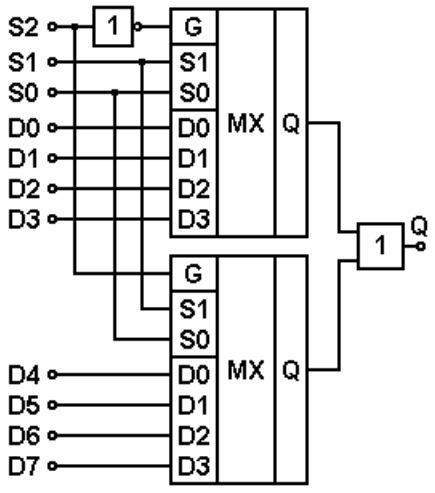
tot látnak el vagy egy meghatározott műveletet végeznek el. Kombinációs hálózatra épülő funkcionális egységek: az adatválasztók, a kódolók, a komparátorok és az aritmetikai egységek.

Multiplexerek vagy adatválasztók

A multiplexerek vagy adatválasztók olyan kombinációs hálózattal felépített funkcionális egységek, melyek egy n-bites adatból egyet választanak ki, azaz az S (SELECT) bemenetekkel kijelölt D (DATA) bemenet közül egynek a tartalmát a Q kimenetre juttatják. Ha az adatbemenetek (D0, D1, ..., Dn–1) száma n és a kiválasztó (Sk-1, ..., S1, S0) bemenetek száma k, akkor az $n=2^k$ összefüggésnek kell teljesülnie. Az S bemenetek bináris kódban tartalmazzák a kiválasztott D bemenet sorszámát. Szokványos megadási mód a D/S/Q számhármas. Tipikus kialakítások: 4/2/1, 8/3/1 és 16/4/1. A multiplexerek bővített változatával, a bemeneti adatok egy csoportját (D10, D11; D20, D21; ...; D40, D41) lehet a (Q0, Q1) kimenetekre kiválasztani. Tipikus kialakítások: 2x2/1/2, 2x4/1/4 és 4x2/2/2. Az áramköri kialakítás szerint a kimenet lehet TP, OC vagy TS (leírás a digitális áramkörökben!), logikai érték szerint lehet ponált vagy negált. A multiplexerek rendelkezhetnek egy G vagy EN - ponált vagy negált aktív szintű - bemenettel, mellyel a kimenet letiltható vagy esetleg HZ-be állítható.

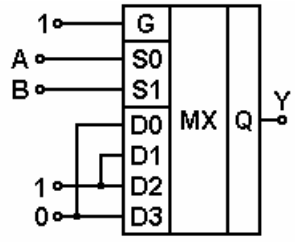


A multiplexerek adatok kiválasztására és n-bites párhuzamos adat sorosra alakítására használhatók. Több (2, 4, 8, ...) azonos D/S/Q multiplexer, az engedélyező bemenetek felhasználásával összekapcsolható. A G bemenetekre egy demultiplexer kimenetei csatlakoznak, így egyidejűleg csak egy multiplexer kimenete engedélyezett, míg a többi a D=0-hoz tartozó értéket veszi fel, ponált kimenet esetén 0-át, negált kimenet esetén 1-et. A kiválasztott adat az egyes kimeneteket összekötő - negált kimenet esetén ÉS, ponált kimenet esetén VAGY - kapu kimenetén jelenik meg. Az inverter 1-ről 2 vonalra dekódol/ demultiplexel! TS kimenetű multiplexerek esetén ez a kimeneteket összekapcsoló kapu elhagyható és a kimenetek közösíthetők.



Egy S számú kiválasztó bemenettel rendelkező multiplexerrel tetszőleges, S vagy S+1 változós logikai függvény realizálható. Egy S változós függvény realizálása az igazságtáblázat alapján történik. A bemenő változókat a szelekciós bemenetekre, a kimenő változó értékeit pedig az adatbemenetre kell kötni. Az S_0 bemenetre a 2^0 helyértékű bemenő változót, az S_1 bemenetre a 2^1 helyértékű bemeneti változót, és így tovább kell csatlakoztatni. A D_0 adatbemenetre a 0. minterm, a D_1 adat-

B A	Y
0 0	0
0 1	1
1 0	1
1 1	0

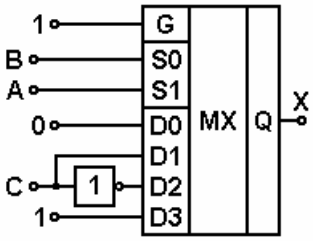


$$Y^2 = \overline{B}A + B\overline{A}$$

bemenetre az 1. minterm, és így tovább, függővváltozójának értékét kell kötni. A multiplexer kimenetén a szelekciós bemenet által meghatározott bemenet logikai értéke jelenik meg, ami a éppen a bemeneti kombinációhoz tartozó függő változó értéke. Egy 16/4/1 multiplexerrel négyváltozós függvény minimalizálás nélkül, egyszerűen csak bekötéssel realizálható. Negált kimenetű multiplexer adatbemeneteire, a függő változó negáltját kell az kötni.

Egy S+1 változós logikai függvény esetén, a legkisebb helyértékű bemenő változó kivételével a bemenő változókat ugyanúgy kell bekötni, mint az előző esetben. Az adatbemenetek bekötéséhez a 2^0 helyértékű bemenő változó és a függő változó kapcsolatát kell megvizsgálni. Az ábrán az igazságtáblázatban a szaggatott vonal azokat a sorokat választja szét, melyekben az A és B változó értéke azonos és a C változó 0-val és 1-el szerepel. Ehhez a 0-hoz és 1-hez tartozhat 00 (első sor) és 11 (utolsó sor) függő változó érték. A kiválasztott adatbemenetre tehát 0-t vagy 1-t kell kötni. A C változó 0 és 1 értékéhez tarozhat 01 (második sor) és 10 (harmadik sor) függő változó érték. A kiválasztott adatbemenetre tehát a C-t, vagy a C negáltat kell kötni. Egy 16/4/1 multiplexerrel egy ötváltozós függvény minimalizálás nélkül, egyszerűen csak bekötéssel realizálható. Negált kimenetű multiplexer adatbemeneteire, a függő változó negáltját kell az kötni.

ABC	X
000	0
001	0
010	0
011	1
100	1
101	0
110	1
111	1

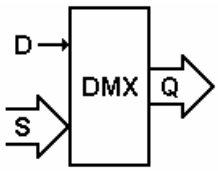


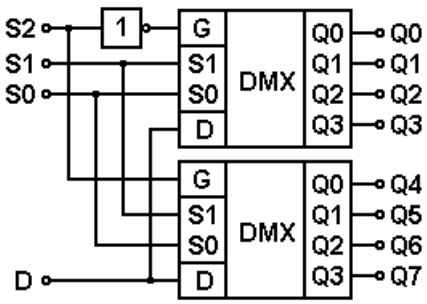
$X^3 = A\overline{C} + BC$

Multiplexerrel párhuzamos-soros átalakítási művelet is elvégezhető. Ehhez egy S bites számláló kimenetét kell a szelekciós bemenetre kötni, az adatbemeneteken lévő értékek a számláló órajelének ütemében, megjelennek a multiplexer kimenetén.

Demultiplexerek vagy adatszétosztók

A demultiplexerek vagy adatszétosztók olyan kombinációs hálózattal felépített funkcionális egységek, melyek a multiplexeléssel ellentétes műveletet végeznek, azaz a D (DATA) bemenet tartalmát az S (SELECT) bemenetek által kijelölt Q kimenetre juttatják. Ha az adatkimenetek (Q0, Q1, ..., Qn–1) száma n és a kiválasztó (Sk-1, ..., S1, S0) bemenetek száma k, akkor az $n=2^k$ összefüggésnek kell teljesülnie. Az S bemenetek bináris kódban tartalmazzák a kiválasztott Q kimenet sorszámát. Szokványos megadási mód a D/S/Q számhármas. Tipikus kialakítások: 1/2/4, 1/3/8 és 1/4/16. A kimenetek áramköri kialakítás szerint lehetnek TP, OC vagy TS, logikai érték szerint ponáltak vagy negáltak. A demultiplexerek rendelkezhetnek egy G vagy EN, ponált vagy negált aktív szintű bemenettel, mellyel a kimenetek letilthatók vagy esetleg HZ-be állíthatók. A demultiplexereket adatok szétosztására és - mivel, egyidejűleg csak egy kimenet aktív - dekódolásra használ-

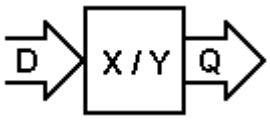




hatók. Több (2, 4, 8, ..., db.) azonos D/S/Q demultiplexer, az engedélyező bemenetek felhasználásával összekapcsolható. A G bemenetekre szintén egy demultiplexer/dekóder kimenetei csatlakoznak, így egyidejűleg csak egy demultiplexer kimenetei engedélyezettek, míg a többi a D=0-hoz tartozó értéket veszi fel, ponált kimenet esetén 0-át, negált kimenet esetén 1-et. A TS kimenetű demultiplexerek minden esetben negált kimenetűek. A demultiplexereket soros - párhuzamos átalakítóként ritkán alkalmazzák, mivel minden kimenet állapotát, az átalakítási ciklus időtartamára, egy elemi tárolóval kellene megőrizni.

Kódolók

A kódolók vagy angol nevükön encoder-ek, olyan kombinációs hálózatból felépített funkcionális egységek, melyek decimális adatokból valamilyen 4, 5 vagy 7 bites BCD kódot állítanak elő. A kódolás általános értelemben áttérés egy nagyobb terjedelmű kódból egy kisebb terjedelmű kódba. A számítástechnikai gyakorlatban a kódolás betűket, írásjeleket, decimális számokat tartalmazó információ, alfanumerikus kódra való átalakítását jelenti. A kódolás és a kódátalakítás célja, hogy az adatok a számítástechnikai készülékekkel feldolgozhatóak legyenek.



Kódoló és kódátalakító áramkörök főbb típusai:

decimális → bináris kódoló: 10 bemenet, 8421 súlyozású BCD kimenet, csak egy és mindig csak egy bemenet lehet aktív.

prioritás kódoló: 10 bemenet, 8421 súlyozású BCD kimenet, több aktív bemenet esetén a legnagyobb decimális értékű bemenethez tartozó kód jelenik meg, külön kimenet jelzi, ha egy bemenet sem aktív.

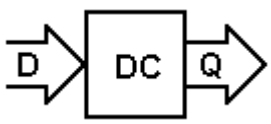
8421 BCD → Aiken, Gray, Stibitz, 5-ből 2, Hamming kódátalakító

paritás kódoló: egy n bites adatban lévő 1 értékű bitek számának párosságát vagy páratlanságát jelzi.

Kódoló egységek szekvenciális hálózatokból is felépíthetők (pl. Johnson számláló), de működésük lassúbb.

Dekódolók

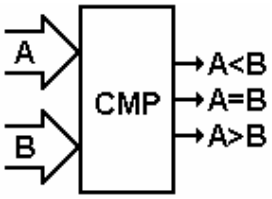
A dekódolók olyan kombinációs hálózattal felépített funkcionális egységek, melyek a bemenetükre érkező n bites kódszavakat (D0, D1, ... Dn-1) átalakítják Ł 2^n hosszúságú, decimális adatként értelmezhető kódszavakra (Q0, Q1, ..., Q9). Egy kimenet akkor aktív, ha a neki megfelelő kódszó van a bemeneten. A kimenet áramköri kialakítás szerint lehet TP, OC, logikai érték szerint lehet ponált vagy negált. Szokványos megadási mód a D/Q számpáros.



Tipikus kialakítások:

8421 BCD → decimálisra: 4/10, negált TP vagy OC kimenet, lámpa, LED, NIXI meghajtó
Stibitz → decimálisra: 4/10
421 → decimálisra: 3/8, negált TS kimenet
8421 BCD → 7 vonalra: 4/7, OC kimenet (7 szegmens meghajtó), automatikus "0" kioltás, lámpa teszt, fényerő vezérlés
Egyes áramkörök a dekóderen kívül a meghajtót és a kijelzőt is tartalmazzák, pl. TIL 311 4x7 pontos hexadecimális kijelző, közös tokban.

Komparátorok

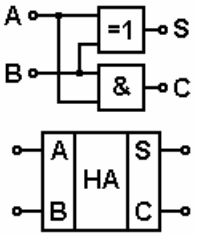


A komparátorok két több bites adatot hasonlítanak össze. Az összehasonlás eredménye három kimeneten jelenik meg. A kimenetek a kisebb, nagyobb vagy egyenlő eredményt jelzik általában ponált értékkel. A bemenetek jelölése A_0, A_1 ... A_n és B_0, B_1 ... B_n bináris helyértékeket követi. Tipikus a négybites komparátor, melynek három kaszkád bemenetével több komparátor összekapcsolható és négynél több bites adatok is összehasonlíthatók.

Összeadó és kivonó áramkörök

Az összeadó és kivonó áramkörök, olyan kombinációs hálózatból felépített funkcionális egységek, melyek kettes számrendszerbeli számok összegét illetve különbségét állítják elő. Az összegzésben résztvevő számok kódolása alapján az áramköri egységek bináris vagy BCD összeadók lehetnek. A bináris összeadók működésének alapját, a bináris számok összeadásának szabályai képezik, melyek a 0+0=0, a 0+1=1 és az 1+1=10. Két egyjegyű bináris szám összeadásakor egy Summa összeg és egy Carry átvitel keletkezik. A szabályok szerint felvett igazságtáblázat alapján az összeadási műveletet, egy ANTIVALENCIA és egy ÉS kapu valósítja meg. Az áramköri egységet - mivel az előző helyértékről érkező átvitelt nem veszi figyelembe és így többjegyű számok összeadására nem alkalmas - fél összeadónak, angol szakkifejezéssel Half Adder-nek nevezzük. Az előző helyértékről érkező C_0 átvitel az összeadás műveletét úgy módosítja, hogy akkor is keletkezik átvitel ha csak az egyik tag értéke igaz (5. és 6. minterm), tehát nem csak akkor, ha mindkét tag értéke igaz (3. és 7. minterm).

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



$S = A \oplus B \quad C = AB$

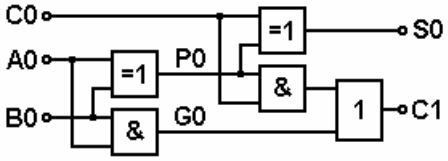
Az S_0 és a C_1 minimalizált függvényei:

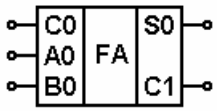
$$S0 = (A0 \oplus B0) \oplus C0 = P0 \oplus C0$$

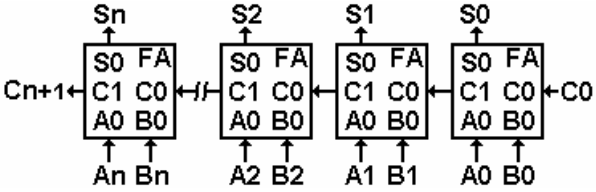
$$C1 = A0 \cdot B0 + (A0 \oplus B0) \cdot C0 = G0 + P0 \cdot C0$$

C0	A0	B0	C1	S0
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

Mivel az A_0 és B_0 változókkal, valamint a C_0 és P_0 belső változókkal ugyanazt a fél összeadási műveletet kell elvégezni, ezért az S0 és C1 logikai függvényeket két fél összeadóval és egy VAGY kapuval lehet realizálni. Az előző helyértéken keletkező átvitelt is fegyelembe vevő áramköri egységet teljes összeadónak, angol szakkifejezéssel Full Adder-nek nevezzük. Egyes irodalmakban SM betűk jelzik a műveletet. Többjegyű számok összeadásához a helyértékeknek megfelelő számú teljes összeadó szükséges, melyek átviteli bitjeit láncba kell kapcsolni. Az összeadás elvégzését lassítja, hogy minden helyértéken meg kell várni amíg előáll az előző helyértéken a helyes átvitel. Ez az úgynevezett átviteli terjedési idő, a számjegyek számától és azok értékétől is függ. Az átvitel képzés módja korlátozza ennek az egységnek az alkalmazását. Az átvitel előállításának gyorsítására lehetőséget ad, hogy az átviteli biteket elő lehet állítani az összeadandó számok bitjeiből még az összeadás elvégzése előtt. Az n+1. helyértéken keletkező átvitel, a teljes összeadó igazságtáblázata alapján: $Cn + 1 = Gn + Pn \cdot Cn$. A Gn belső változó azt adja meg, hogy az n. helyértéken az összeadásból keletkezik vagy generálódik-e átvitel, a Pn változó pedig azt, hogy az előző helyértékről érkező átvitel elnyelődik vagy továbbterjed azaz propagálódik-e. Az n. helyértéken keletkező átvitel: $Cn = Gn-1 + Pn-1 \cdot Cn-1$, és így tovább. A Cn+1-be a Cn-t, a Cn-be a Cn-1-t, stb. helyettesítve állnak elő a párhuzamos átvitel, angolul Paralell Carry Logic függvényei.

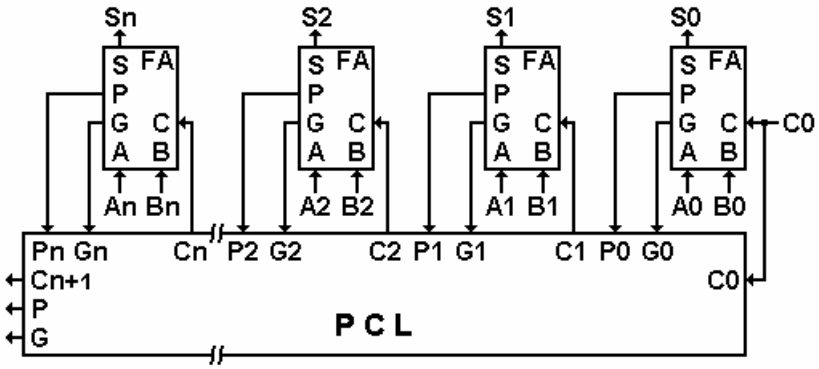




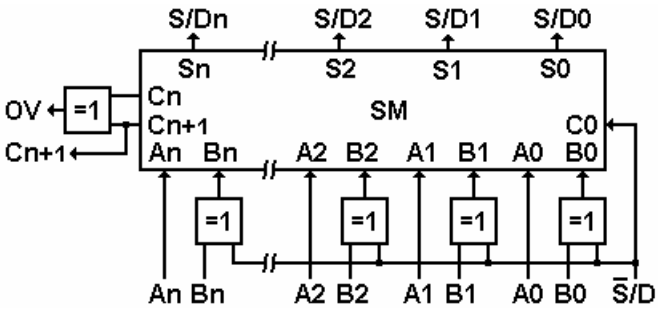


$$C1 = G0 + P0\ C0$$
$$C2 = G1 + P1\ G0 + P1\ P0\ C0$$
$$C3 = G2 + P2\ G1 + P2\ P1\ G0 + P2\ P1\ P0\ C0$$
$$C4 = G3 + P3\ G2 + P3\ P2\ G1 + P3\ P2\ P1\ G0 + P3\ P2\ P1\ P0\ C0$$
$$\ldots$$

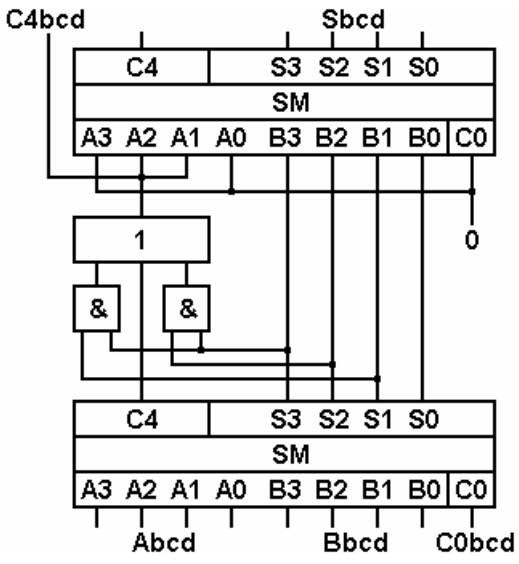
Az átvitel gyorsítás módja tehát az, hogy minden helyértéken az összes alacsonyabb helyértékből egyszerre képződik az átvitel. Mivel a PCL logikai függvényei kétszintű hálózattal előállíthatók, ezért minden helyértéken azonos áramköri késleltetéssel jön létre az átvitel. Standard TTL áramkörrel 16 bites összeadás ideje kb. 20ns! Hátrányként könyvelhető el, hogy egyre több bemenetű ÉS és VAGY kapu szükséges. PCL-el kiegészített összeadó sémájában a P, G valamint Cn+1 kimenet egy további - még egy n bites, tehát két n bites összeadót összekapcsoló - PCL megfelelő bemeneteire csatlakozhat.



A bináris kivonó áramkör az A – B = A + (–B) azonosság felhasználásával, a kivonandó kettes komplemensének hozzáadásával végzi el a különbségképzést. A kivonandó egyes komplemensét kétbemenetű ANTIVALENCIA kapukkal, kettes komplemensét a C0 = 1 hozzáadásával lehet előállítani. Az S/D bemenet értékétől függően az áramkör összeget (Summa) vagy különbséget (Differencia) képez. Kettes komplemens ábrázolás esetén az n. bit az előjel, aminek ha 0 az értéke, akkor pozitív és ha 1 az értéke, akkor negatív számról van szó. Két tetszőleges előjelű szám összege túllépheti az ábrázolási tartományt és ilyenkor túlcsordulás, angolul Overflow, keletkezik. Ebben az esetben az előjelbit értéke és az eredmény is helytelen lehet. Az OV túlcsordulás az előjel helyére belépő Cn és az innen kilépő Cn+1 átvitel különbözőségéből mutatható ki.

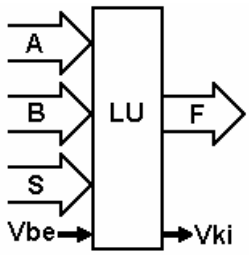


A BCD összeadók két 8421 súlyozású bináris szám összegét képezik. Bináris összeadó alkalmazása esetén, hogy az összeg is BCD kódban legyen, az átvitelt és az összeget korrigálni kell. Ha az összeg Ah ÷ Fh tartományba esik, akkor a 10-es és a 16-os számrendszer alapszáma közötti különbséget, tehát 6-ot kell az összeghez még hozzáadni. Az intervallumba eső összeg figyelését egy Q^4 = S3 S2 + S3 S1 logikai függvényt megvalósító kombinációs hálózattal lehet megoldani. Ha az összeg 10h ÷ 12h tartományba esik, akkor az összeghez szintén 6-ot kell hozzáadni. Ezt az intervallumot a C4 átvitelbit 1 értéke jelzi. A BCD kód érvényessége miatt a kombinációs hálózat ilyenkor nem jelez! A korrekciót egy második 4 bites bináris összeadó végzi el, melynek egyik bemenetére a két BCD szám bináris összege kerül, a másik bemenetére 0h vagy 6h, a korrekció szükségességétől függően. Mivel a második összeadó C4 átvitelbitje csak bináris átvitelt jelez, így nem használható fel BCD átvitelként. A helyes BCD átvitelt, ami éppen korrekció esetén képződik, a kombinációs hálózat állítja elő (C4bcd). A BCD kivonó áramkörök 5 bites bináris összeadóval oldhatók meg, mely a kettes komplemens ábrázolás előjelét is figyelembe veszi. A felépítése a bináris kivonó áramkörrel azonos, a helyes BCD összeghez ill. különbséghez szintén korrekció szükséges.



Logikai műveletvégző egységek

A logikai műveletvégző egységek több, általában négy bites adatokkal végeznek különféle



logikai műveleteket. A blokkvázlatban az A és B a bemeneti adatok, az S szintén négybites művelet kiválasztó bemenet, az F a négybites kimeneti függvény. Ezen felül van még a működést befolyásoló Vbe bemeneti vezérlőjelek valamint a műveletvégzés során keletkező Vki vezérlőjelek, mint például az eredmény egyelő nulla, vagy páros,vagy negatív. A négybites S bemeneten 16 féle műveletet lehet beállítani, melyek lehetnek az ÉS a VAGY a NEMÉS, a NEMVAGY, az XOR, stb. de találhatunk valamelyik változó negálása, vagy valamelyik változó eggyel növelése vagy csökkentése műveleteket is. A négybites adatok közötti logikai műveletvégzés helyérték helyesen és bitenként történik.

A logikai egységeket többnyire egy aritmetikai egységgel helyezik közös tokozásba. Az így létrehozott áramkört nevezzük aritmetikai és logikai egységnek, angolból átvéve Arithmetic Logic Unit. Egy bemeneti vezérlőjellel lehet kiválasztani, hogy aritmetikai vagy logikai műveletnek kell értelmezni a műveletkiválasztó bemeneten lévő adatot. Az ALU-k megtalálhatók önálló áramköri kivitelben pl. SN74181 és a mikroprocesszorokban beépítve.

Szekvenciális hálózatok alapelemei

A szekvenciális hálózatok alapelemei a tároló áramkörök. A tároló áramkörök egy logikai változó értékének megőrzésére, tárolására szolgálnak. A tárolt logikai változó értéke a bemenő változók pillanatnyi állapotától, azok időbeni változásától és a tárolók korábbi állapotától függ.

A tároló a fizikai vezérlés alapján lehet:

- statikus vagy szintvezérelt: ha a bemeneti változók állapotának változásait a kimeneti változó értéke azonnal követi.
- dinamikus vagy élvezérelt: ha a bemeneti változók állapotának változásait a kimeneti változó értéke, egy vezérlőjel meghatározott irányú megváltozásának hatására követi.

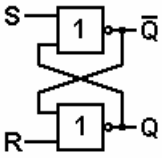
A tároló a logikai vezérlés alapján lehet:

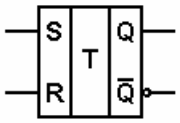
- RS tároló
- kapuzott RS tárol
- D tároló
- JK és T tároló

Szintvezérlésű tárolók

A szintvezérelt vagy statikus tárolók alapelve, két sorosan kapcsolt és pozitívan visszacsatolt inverter működésére vezethető vissza. A vezérelhetőség érdekében két bemenetű univerzális kapuk szükségesek.

RS tároló felépítése, rajzjele, vezérlési táblázata:

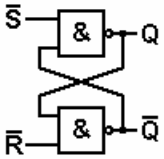


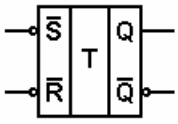


R S	Q^{n+1}
0 0	Q^n
0 1	1
1 0	0
1 1	tiltott

Az RS tároló S = R = 0 esetén kerül vezérlésmentes, vagy tárolási állapotba. A Q kimenete S(et) = 1-el beírható, R(eset) = 1-el törölhető. Kettős vezérlés tiltott, mivel S = R = 1-ből, S = R = 0-ba való átmenet után a tároló a beírt vagy törölt állapotot véletlenszerűen veszi fel.

Inverz RS tároló felépítése, rajzjele, vezérlési táblázata:

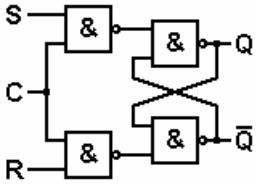


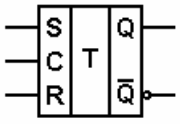


R S	Q^{n+1}
0 0	tiltott
0 1	0
1 0	1
1 1	Q^n

Az inverz RS tároló S = R = 1 esetén kerül vezérlésmentes, vagy tárolási állapotba. A Q kimenete S = 0-val beírható, R = 0-val törölhető. Kettős vezérlés tiltott, mivel S = R = 0-ból, S = R = 1-be való átmenet után a tároló a beírt vagy törölt állapotot véletlenszerűen veszi fel.

Kapuzott RS tároló felépítése, rajzjele, vezérlési táblázata:

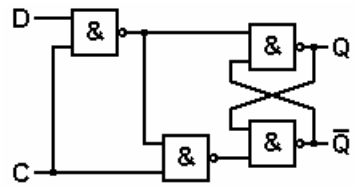


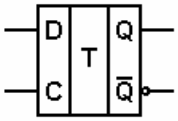


C R S	Q^{n+1}
0 x x	Q^n
1 0 0	Q^n
1 0 1	1
1 1 0	0
1 1 1	tiltott

A kapuzott RS tároló C = 0 esetén kerül vezérlésmentes vagy tárolási állapotba. Ha a C kapuzó bemenet értéke 1, akkor a tároló Q kimenete S = 1-el beírható és R = 1-el törölhető. A C bemenet 0-ra váltásakor, rögzül a kimenetre beírt állapotot és ilyekor az R S bemenetek is hatástalanok. Kettős vezérlés tiltott.

D tároló (LATCH) felépítése, rajzjele, vezérlési táblázata:

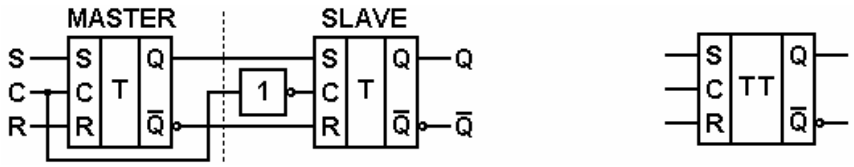




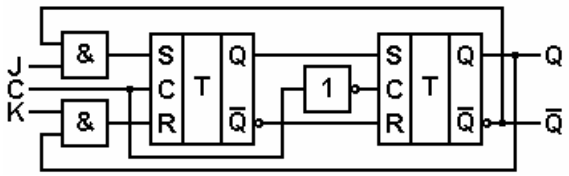
C D	Q^{n+1}
0 x	Q^n
1 0	0
1 1	1

Ha a kapuzott RS tároló S és R bemenetei közé egy invertert helyezünk el, akkor a fennmaradó S bemenet vezérlésével csak beírás vagy törlés műveletek jöhetnek létre. Az így létrehozott egyetlen bemenetet D-nek és a tárolót D tárolónak nevezzük. A D tároló tárolási állapota C = 0 esetén jön létre. Ilyekor a D bemenet változtatása hatástalan. A C órajel bemenet 1 értéke estén a tároló átlátszik, azaz a Q kimenet követi a D bemenet állapotát. A C bemenet 0-ra váltásakor, a D bemenet pillanatnyi értéke rögzül a kimeneten.

Kétfokozatú RS-MS tároló felépítése, rajzjele:



A kétfokozatú tárolók alapelve két sorosan kapcsolt, de ellenfázisban engedélyezett kapuzott RS tároló működésére vezethető vissza. C = 1 estén a MASTER fő tároló az RS bemenetek állapotát beolvassa, majd C = 0 esetén ez az állapot a SLAVE segéd tároló kimenetére íródik. A kapuzott RS tárolóhoz viszonyítva a működés abban tér el, hogy a kimeneten a C bemenet 1 értékénél még nem, hanem csak 0-ra váltásakor jelenik meg a kimeneten az állapot változás. Az RS-MS tároló csak elvi jelentőségű, mivel működése megegyezik az egyszerűbb felépítésű RS működésével. Kettősvezérlés szintén tiltott. Az RS-MS tároló kettősvezérlés tiltása feloldható, ha az R-S bemenetek állapotát a visszavezetett kimeneti és a megkülönböztetésül J-K-val jelölt bemeneti változók együttesen határozzák meg.
Kétfokozatú JK-MS tároló felépítése, rajzjele, vezérlési táblázata:

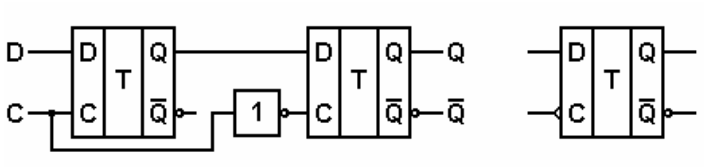


J K	Q^{n+1}
0 0	Q^n
0 1	0
1 0	1
1 1	$\overline{Q^n}$

A kétfokozatú JK_MS tároló beírás és törlési művelete azonos az előző RS_MS tároló működésével. J = K = 1 esetén a C felfutó éle a tároló kimeneti állapotának negáltját írja be a fő tárolóba, a lefutó éle ellenkező állapotba billenti a segéd tárolót. A JK-MS tároló kettős vezérlés esetén minden befejezett órajel után állapotot vált.

Élvezérlésű tárolók

Az élvezérlésű vagy dinamikus tárolók alapelve két sorosan kapcsolt de ellenfázisban engedélyezett D típusú tároló működésére vezethető vissza.
Negatív élvezérelt D tároló felépítése, rajzjele, vezérlési és állapot átmeneti táblázata:



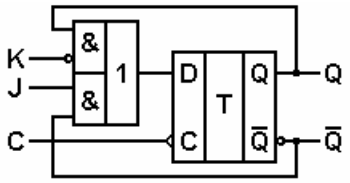
D	Q^{n+1}
0	0
1	1

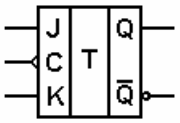
Q^n	D	Q^{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

A bemeneti és a kimeneti tárolók átlátszó vagy rögzített állapota a C órajeltől függően ellentétesen váltakozik. A C = 0 értékénél a bemeneti tároló zárt, a kimeneti tároló nyitott, azaz másolja a bemeneti állapotát. A C = 0-1 átmeneténél a kimeneti tároló lezár, a bemeneti tároló kinyit és a D bemenet állapotát veszi fel. A C = 1-0 átmeneténél és csak ekkor, a D állapota átíródik a kimeneti tárolóba, azaz a Q kimeneten megjelenik. Az élvezérelt tárolókra a nagyobb működési sebesség jellemző, mivel a D bemeneten az érvényes állapotnak

az órajel lefutása előtt (t_{SETUP}) és után (t_{HOLD}) csak kb. egy áramköri késleltetési ideig kell fennállnia.

Negatív élvezérelt JK tároló felépítése, rajzjele, vezérlési és állapot átmeneti táblázata:

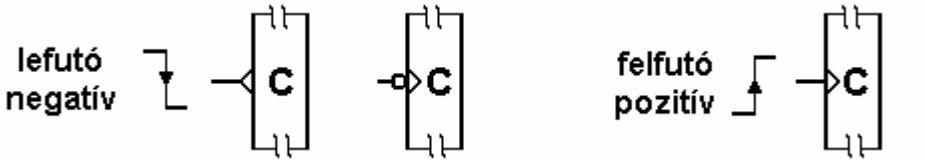




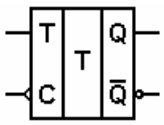
J K	Q^{n+1}
0 0	Q^n
0 1	0
1 0	1
1 1	$\overline{Q^n}$

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

Az élvezérelt JK tároló működése teljesen azonos a JK-MS tároló működésével. Az élvezérelt D tároló valamint a D bemenetre csatlakozó kombinációs hálózat a J, K és Q^n bemenő változók függvényében együttesen valósítja meg a J-K jellegű logikai működést. A kimenetek állapotának megváltozása a C órajel 1-0 átmenetnél következik be. Pozitív élvezérlésű, tehát a C bemenet 0-1 átmenetre működő tárolók működése az előzőekkel azonos, de a rajzjelben a dinamikus működést jelző háromszög a doboz belsejében, befelé mutatóan van feltüntetve. Létezik olyan rajzjel is ami a pozitív élvezérlés rajzjelét egy bemeneti invertálással fordítja meg negatív élvezérlésűre.



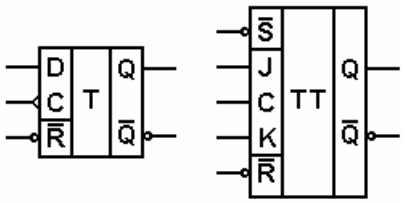
A JK tárolók J-K bemeneteinek közösítésével T tároló alakul ki. A közösítés miatt a vezérlési táblázatból csak az első és az utolsó sor marad meg. Az órajel hatására T = 0 esetén a tároló az előző állapotban marad és T = 1 esetén állapotot vált. A negatív élvezérlésű T tároló rajzjele és vezérlési táblázata:



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

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

A kétfokozatú és élvezérelt tárolók gyakran el vannak látva törlő (CLEAR, RESET) és beállító (PRESET, SET) bemenetekkel. A törlő - beállító bemenetek működése lehet aszinkron, azaz a megjelenő aktív szint az órajeltől függetlenül azonnal vezérli a tárolót, és szinkron működésű, azaz a megjelenő aktív szint a következő órajel megjelenésekor vezérli a tárolót. Ezek a vezérlő bemenetek használhatók fel, a bekapcsoláskor véletlenszerűen beálló tárolók alaphelyzetbe állítására. A tárolók a szekvenciális hálózatok, regiszterek, számlálók alapelemei.



Számlálók

Számlálóknak nevezzük azokat a szekvenciális hálózatokat, melyek az órajel bemenetükre beérkezett impulzusokat megszámlálják és az eredményt párhuzamos, helyértékes kimene-

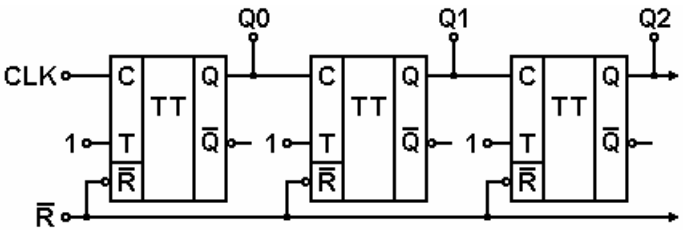
teken, valamilyen kódolt formában jelenítik meg. A számlálók feladata kettős, egyrészt tárolják az órajelek számlálásának eredményét, másrészt a következő órajel hatására ezt az eredményt előírt módon megváltoztatják. Ha az órajelek száma eléri a számláló kimenetén az egymástól megkülönböztethető állapotok számát, akkor a következő órajel hatására a számláló valamelyik, már korábban előfordult állapotba kerül, azaz a számlálást "újra kezdi". A kimeneten megjelenő egymástól megkülönböztethető állapotok száma a számláló modulo száma vagyis modulusa. Egy n bites (n kimenetű) számláló modulusa legfeljebb 2^n lehet. A számlálók kétütemű (Master-Slave) vagy élvezérelt JK ill. T tárolókból építhetők fel. A számlálók csoportosíthatók:

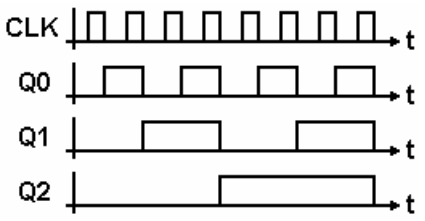
- a tárolók vezérlési módja szerint: - aszinkron működésű számlálók
  - szinkron működésű számlálók
- a számlálás kódja szerint: - bináris számlálók
  - decimális (BCD) számlálók
  - tetszőleges modolusú (programozható) számlálók
  - tetszőleges sorrendű számlálók
- a számlálás iránya szerint: - előre (felfelé) számlálók
  - hátra (lefelé) számlálók
  - kétirányú (reverzibilis) számlálók

Aszinkron számlálók

Aszinkron működésű számlálókban a számlálandó órajel csak a legkisebb helyértéket előállító tároló órajel bemenetére csatlakozik. Minden további tároló az órajelét az előző helyértékű tároló kimenetéről kapja. Az összekapcsolás módjából következik, hogy a tárolók "dominó" szerűen működtetik egymást.

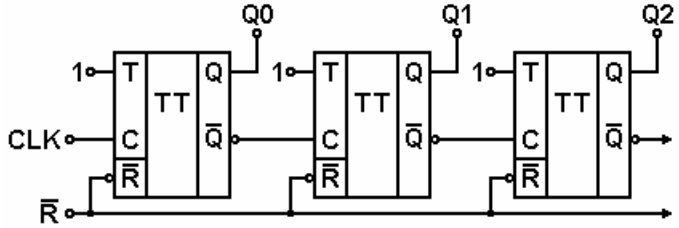
Bináris előre számlálóban a negatív élvezérelt vagy kétütemű T tárolók ponált kimenete csatlakozik a következő, nagyobb helyértékű tároló órajel bemenetére. Alaphelyzetbe (0. állapot) állítás (RESET) után, az első tároló minden egyes (2^0), a második tároló minden második (2^1), a harmadik tároló minden negyedik (2^2), a n-ik tároló minden 2^{n-1}-ik befejezett órajelnél vált állapotot. A kimeneteken a beérkezett órajelek száma természetes bináris kódban jelenik meg. Egy n bites számláló számlálási határa 2^n-1 és a megjelenő kimeneti állapotok száma 2^n. A számlálási határ elérése után a tárolók alaphelyzetbe (0. állapotba) állnak és új számlálási ciklus kezdődik.



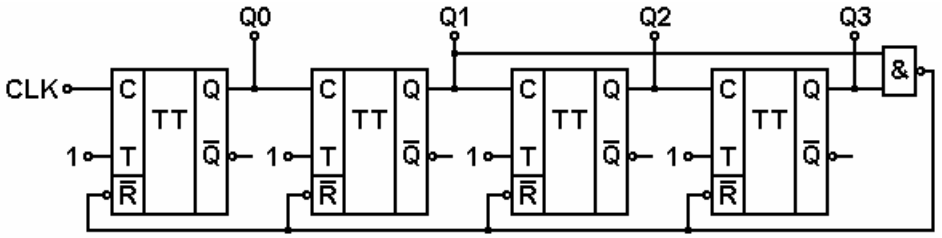


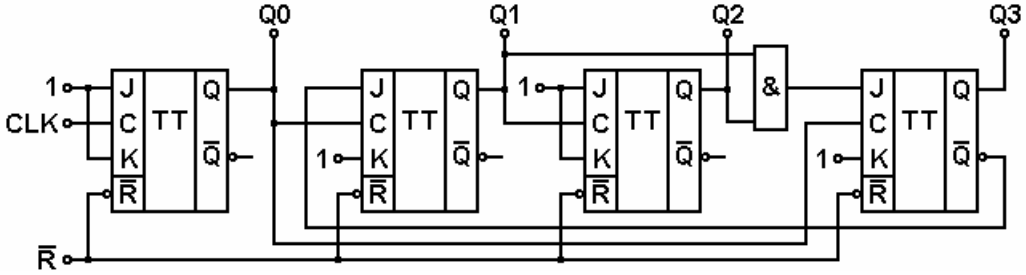
Bináris hátra számlálóban a negatív élvezérelt vagy kétütemű T tárolók negált kimenete

csatlakozik a következő, nagyobb helyértékű tároló órajel bemenetére. Alaphelyzetbe (0. állapot) állítás (RESET) után az első órajel hatására a legkisebb helyértékű tároló állapotot vált, aminek következtében az állapot váltás az összes tárolón végighalad, azaz a kimeneteken a számlálási határ felső értéke jelenik meg. A további órajelek a természetes bináris kódot csökkenő sorrendben hozzák létre tároló kimenetén. A 0. állapot elérése után új számlálási ciklus kezdődik. (pozitív élvezérlésű tárolók esetén az ellentétes kimeneteket kell használni !)

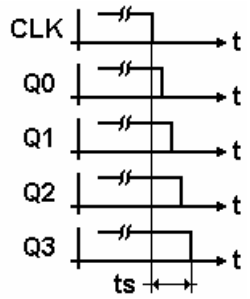


Decimális vagy 8421 súlyozású BCD előre számlálót egy 4 bites bináris számláló ciklusrövidítésével vagy a JK bemenetek megfelelő kapuzásával lehet előállítani. A ciklusrövidítéshez általában egy - többnyire bináris - számlálót egy kombinációs hálózattal kell kiegészíteni, melynek bemenetére a számláló kimenetei, a kimenete a számláló aszinkron törlő (CLEAR, RESET) vagy beállító (PRESET) bemeneteire csatlakoznak. Működéskor, egy meghatározott számlálási állapotnál a kombinációs hálózat a számlálót a kezdőállapotra állítja be. Ciklusrövidítés esetén tehát az összes lehetséges kimeneti állapotnak csak egy része jön létre. Ciklusrövidítéssel megvalósított decimális előre számlálóban a Q1 és Q3 kimenetekre kapcsolódó NAND kapu, a 11. azaz az 1010 kimeneti állapotnál törli a számláló tárolóit. A kialakítás hátránya egyrészt, hogy az áramköri késleltetések által meghatározott időtartamra a 11. állapot is megjelenik, másrészt a RESET bemenetek foglaltsága miatt a számláló kívülről nem állítható alaphelyzetbe. A JK bemenetek kapuzásával megvalósított decimális előre számlálóban a JK tárolók vezérlési lehetőségeinek (beírás, törlés, állapot váltás) felhasználásával lehet megoldani, hogy csak az első tíz bináris állapot jöhessen létre. Ennek érdekében a JK tárolókból felépített aszinkron bináris előre számlálót három helyen kell módosítani: a J1-re a Q3 negáltat, a J3-ra a Q1&Q2-öt valamint a C3-ra - mivel az 1001→ 0000 állapot váltásnál az előző helyértéken nem képződik órajel - a Q0-t kell kapcsolni. RESET után a 210 tárolók az első 7 órajel alatt, a J1 = Q3 = 1 miatt, bináris számlálóként működnek. A 8. órajel a 210 tárolókat 0-ra, a 3 tárolót - mivel előző-





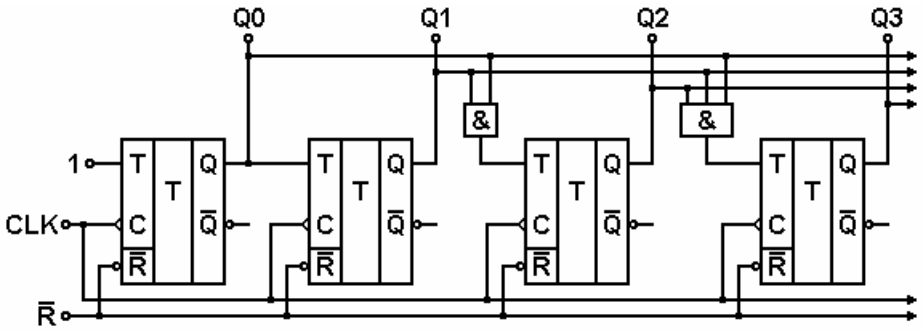
leg J3=1 volt - 1-be állítja. A 9. órajelre a 0 tároló 1-be, a 10.-re 0-ba billen. Az 1 és így a 2 tároló is, a J1 = 0 miatt 0-ban marad, a 3 tárolót a Q0 lefutó éle - a J3=0 miatt - törli. Az 1001 állapot után tehát a 0000 állapot következik. A RESET bemenettel a felhasználó szabadon rendelkezhet. Az aszinkron decimális hátra számlálók is megvalósíthatók, az előre számlálók elvi felépítése alapján.



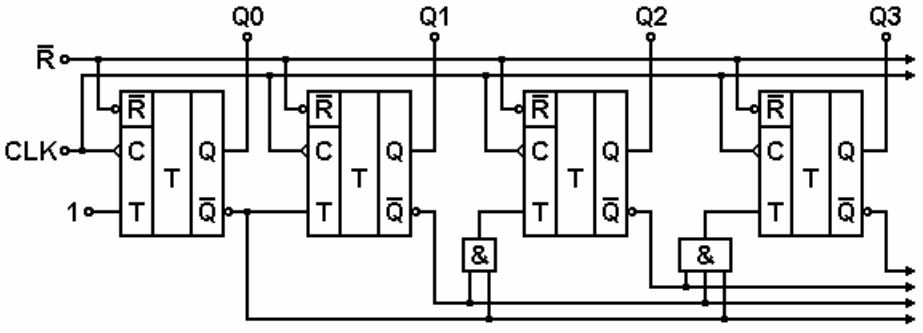
Az aszinkron számlálók felépítéséből következik, hogy az egyes helyértékeken a tárolók állapotváltása nem azonos időpillanatban következik be. Egy négybites bináris számláló esetén az 1111 - 0000 állapot váltáskor a 1110, 1100, 1000 hazárd állapotok is megjelennek. A jelenség egyrészt korlátozza az órajel frekvenciáját, másrészt késlelteti a kimeneti adat további feldolgozását. Az aszinkron számlálók egyszerűbb esemény számlálási feladatokra használhatók fel.

Szinkron számlálók

Szinkron működésű számlálókban a számlálandó órajel minden egyes tároló órajel bemenetét egyszerre, szinkronban vezérli. Mivel az órajel csak az állapotváltás ütemét határozza meg, ezért az egyes helyértékeken az állapotváltást a tárolók logikai bemenetén kell megfelelő vezérlő jelekkel létrehozni. A vezérlő jeleket, a számláló egymás után következő állapotainak függvényében, az előző helyértékű tárolók állapotából egy-egy kombinációs hálózat állítja elő. A szinkron számlálók élvezérelt - esetleg kétütemű - JK ill. T tárolókból építhetők fel.



Bináris előre számlálóban az n. helyértékű tároló billenésének feltétele, hogy az összes előző helyértékű tároló beírt állapotban legyen. A T tárolók azon tulajdonságát felhasználva, hogy T = 0 esetén a tároló az előző állapotban marad és T = 1 esetén állapotot vált, az n. tároló vezérlő jelét egy n-1 bemenetű ÉS kapuval lehet előállítani. A 0. helyértékű tároló minden órajelnél billen, ezért T0 = 1.

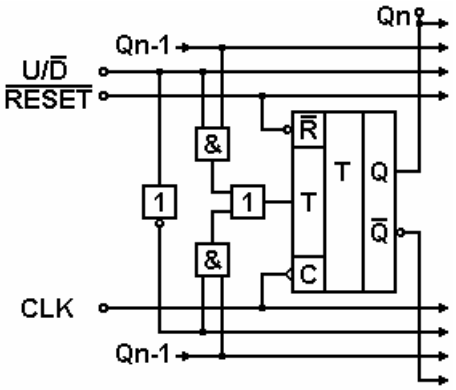


Bináris hátra számlálókban az n. helyértékű tároló billenésének feltétele, hogy az összes előző helyértékű tároló törölt állapotban legyen. A vezérlő jeleket, szintén n-1 bemenetű ÉS ka-

puval lehet előállítani, de ebben az esetben a törölt állapotot 1 szinttel jelző Q negált kimeneteket kell felhasználni.

A szinkron bináris számlálók maximális órajel frekvenciáját csak az alkalmazott tároló és a vezérlő jelet előállító kapuk együttes késleltetési ideje határozza meg. Szinkron számlálók működési sebességét nem korlátozza az alkalmazott tárolók száma, így egy szinkron számlálónak lényegesen nagyobb (TTL esetén kb. 50 MHz) lehet az órajel frekvenciája, mint egy azonos számlálási határra megvalósított aszinkron számlálónak.

Kétirányú vagy reverzibilis bináris szinkron számlálók kétféle módon valósíthatók meg. Az egyik megoldás, hogy két külön órajel bemenet szolgál az előre (CLKUP) illetve a hátra (CLKDOWN) számlálásra. Alkalmazása körülményes, mivel a pontos működés érdekében, külső áramkörökkel le kell tiltani az órajelek egyidejűségét. A másik és a gyakorlatban elterjedtebb megoldás, hogy a számláló egy iránykijelölő (UP/DOWN) bemenettel egészül ki és az órajel mindkét irányban közös. A számláló előnyös tulajdonsága, hogy az órajel bármelyik fázisában reverzálható az iránykijelölő bemenettel.



Tetszőleges sorrendű számlálók egyedi tervezésű szinkron hálózatok. A tárolók dinamikus működésű JK, T és D típusúak lehetnek. A logikai bemenetek vezérlő jeleit egy-egy kombinációs hálózat állítja elő az egymás után következő kimeneti állapotok függvényében. Általában a JK tároló alkalmazása a célszerűbb, mivel állapot átmeneti táblázatának minden sorában szerepel egy határozatlan term, ami a vezérlési függvények minimalizálása során jól felhasználható. A számlálási állapotok száma és sorrendje tetszőleges lehet. A szinkronhálózatok tervezésének lépései:

- ha a feladat szövegesen adott, akkor az állapot diagram felvétele;
- ha a feladat állapot- vagy ütem diagrammal adott, akkor ez a tervezés alapja;
- állapot átmeneti tábla felvétele a n. és n+1. állapotok alapján;
- állapot átmeneti tábla kitöltése, a tárolók állapot átmeneti táblái alapján;
- vezérlési függvények meghatározása, minimalizálása V_K táblával;
- ha van erre vonatkozó előírás, akkor a vezérlési függvények átalakítása;
- kapcsolási rajz elkészítése.

Egy szinkron hálózat elemzésének lépései:

- kapcsolási rajz alapján, a vezérlési függvények kiolvasása;
- állapot átmeneti tábla felvétele, egy adott vagy választott kezdeti állapottal;
- a vezérlési függvények megoldása a kezdő vagy az n. állapot alapján;
- vezérlési táblázatok alapján az n+1. állapot meghatározása;
- a műveletsor folytatása az n+1 állapottal, az összes állapot végigjárásáig;
- állapotdiagram megadása.

Elágazásosnak nevezzük azokat a tetszőleges sorrendű szinkron hálózatokat, melyek egy vagy több külső vezérlőjel értékétől függően, eltérő állapot sorozatokat vesznek fel. Az elágazásos hálózatok tervezésének lépései azonosak a tetszőleges sorrendű hálózatok tervezésével, de a vezérlési függvények minimalizálása során a külső vezérlőjeleket is figyelembe kell venni. A vezérlési függvények minimál alakját, a kombinációs hálózatokra alkalmazott összevonási szabályok szerint kell meghatározni. Ha a táblában csak 1-h vagy 0-h értékű cellák vannak, akkor a vezérlési függvény 1 vagy 0 lesz. Ha a táblában 1-0-h értékű a cellák tartalma, akkor a vezérlési függvény diszjunktív vagy konjunktív minimál alakja, a tömböket alkotó azonos értékű változókkal és a felhasználható határozatlan termek bevonásával megadható. Ha a táblában külső változó is van, akkor első lépésben a határozatlan termek bevonásával 1-re vagy 0-ra el kell végezni a lehetséges összevonásokat, majd a külső változókat 1-nek vagy 0-nak tekintve a táblában lévő 1-esekkel vagy 0-lákkal, 1-re vagy 0-ra újabb tömböket kell képezni, melyekben a külső változó a tömböt alkotó termhez kapcsolódik. A külső változó mindig a táblában lévő értékével szerepel a vezérlési függvényben. Több azonos értékű külső változó szintén összevonható. Több eltérő értékű külső változót vagy több külső változót külön-külön kell összevonni. A vezérlési függvényeket célszerű mintermes és maxtermes alakban is meghatározni, majd az egyszerűbb, kevesebb változót tartalmazó alkot felhasználni.
(mintafeladatok a fejezet végén a mellékletben találhatók!)

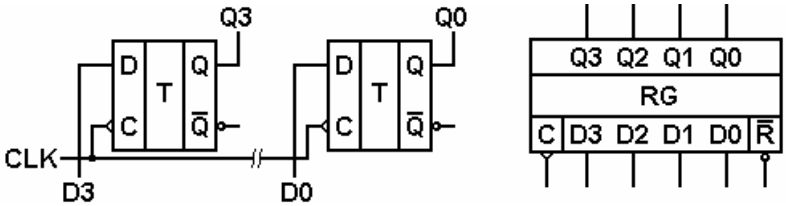
Regiszterek

A regiszterek, közös vezérlőjellel működő, több elemi tárolóból álló, olyan szekvenciális hálózatok, melyek több bites adat átmeneti tárolására vagy valamilyen előírás szerinti átalakítására szolgálnak. A regiszterek a végzett művelet valamint az adat be- és kimeneteik alapján lehetnek:

- átmeneti tárolók párhuzamos bemenet - párhuzamos kimenet
- léptetőregiszterek soros / párhuzamos bemenet - soros / párhuzamos kimenet

Átmeneti tárolók

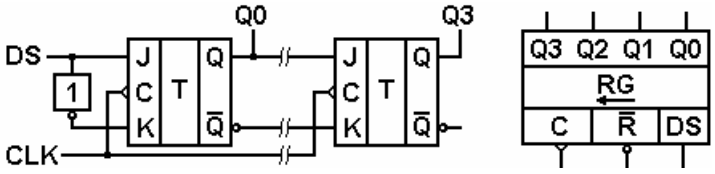
Az átmeneti tárolók vagy puffer regiszterek szintvezérelt D, pozitív vagy negatív élvezérelt D vagy JK, esetleg JK-MS tárolókból építhetők fel. Általában 4 vagy 8 bites adatot tárolnak. Az egyes tárolók órajel és törlő bemenetei közösítettek. A D_n - D_0 párhuzamos adat az órajel hatására kerül a Q_n - Q_0 kimenetekre. A kimenet TP vagy TS kialakítású lehet. Átmeneti tárolókból épülnek fel a µP-ok regiszter tömbjei is.



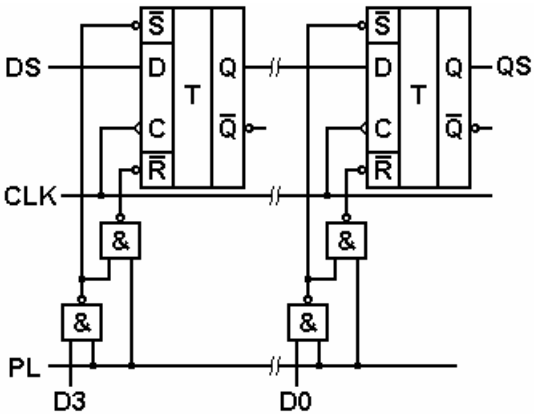
Léptető regiszterek

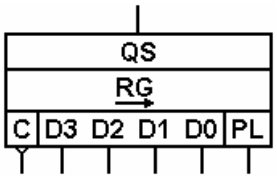
A léptető- vagy shift regiszterek soros adatokat párhuzamosra vagy párhuzamos adatokat sorosra alakítatnak át. A léptetés az órajel ütemében történik, azaz n bites adat ki- ill. beléptetéséhez n db. órajel szükséges. Az órajel hatására az adat bitjei helyértékről helyértékre vándorolnak. A léptetés iránya alapján léteznek egy- és kétirányú léptetőregiszterek. A jobbra léptetés mint aritmetikai művelet kettővel való osztást, a balra léptetés kettővel való szorzást jelent. A léptetőregiszterek JK-MS, élvezérelt D vagy JK tárolókból építhetők fel.

A soros-párhuzamos átalakítókban az órajel a DS soros adatbemenet pillanatnyi értékét lépteti be a párhuzamos kimenetre. Jobbra léptetéshez a nagyobb helyértékű tároló kimenetét kell a kisebb helyértékű tároló bemenetére kapcsolni. A soros adat (DS) a legkisebb helyértékű bitjével érkezik a legnagyobb helyértékű tároló bemenetére. Balra léptetéshez a kisebb helyértékű tároló kimenetét kell a nagyobb helyértékű tároló bemenetére kapcsolni. A soros adat (DS) a legnagyobb helyértékű bitjével érkezik a legkisebb helyértékű tároló bemenetére. A Q_0 ill. a Q_{n-1} soros adatkimenetként is felhasználható. A rajzjelben a léptetési irányt a funkciójelölés alatt feltüntetett nyíl mutatja.



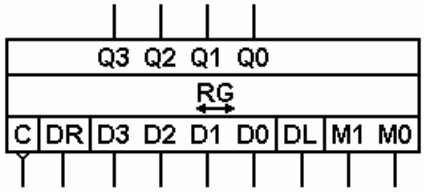
A párhuzamos-soros átalakítókban az órajel a tárolókba beírt D_0 - D_{n-1} párhuzamos adatot lépteti a QS soros kimenetre. A párhuzamos beíráshoz (PL) a tárolók aszinkron törlő vagy beállító bemenetei használhatók fel. A tárolók összekapcsolása a jobbra ill. balra léptetéshez, azonosak az előzőkkel. Jobbra léptetés esetén a D_{n-1}, balra léptetés esetén D_0 lép ki utoljára a QS kimeneten. A DS soros adatbemenetként is felhasználható.





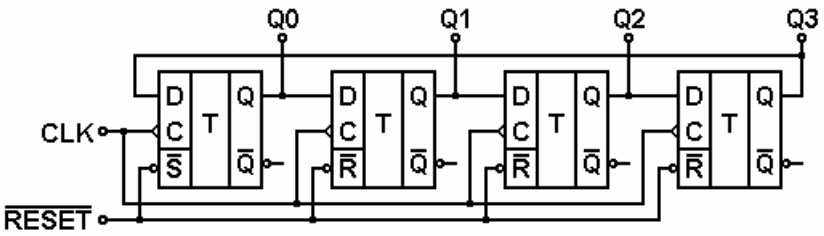
Kétirányú vagy univerzális léptetőregiszterekben, minden tároló bemeneti adatát egy kombinációs hálózat vagy egy-egy multiplexer állítja elő. A kombinációs hálózat bemenő változói az előző és a következő helyértékű tárolók kimenetei (Q_0 - Q_{n-1}), a soros adatbemenetek, DL(eft) - soros adat balra léptetéshez, illetve DR(ight) - soros adat jobbra léptetéshez, valamint az üzemmódot meghatározó M1, M0 bemenetek, a multiplexer S1, S0 bemenetei. Az üzemmód bemenetekkel kiválasztható funkciók egy lehetséges megoldása:

M1	M0	funkció
0	0	aszinkron törlés
0	1	léptetés balra
1	0	léptetés jobbra
1	1	párhuzamos betöltés



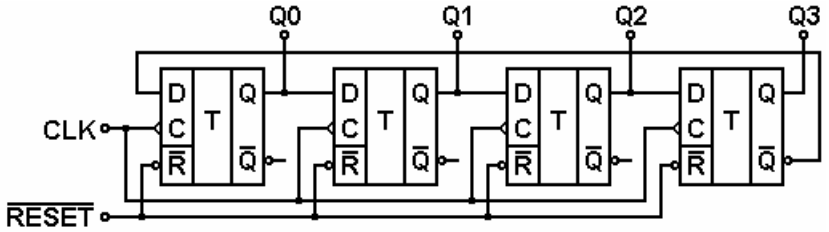
Gyűrűs számlálók

A gyűrűs számlálók, valamilyen visszacsatolással ellátott léptetőregiszterekből állnak. A léptetőregiszter szinkron működésű, élvezérelt JK vagy D tárolókból felépített soros-párhuzamos átalakító. A léptetési irány a gyűrűs számlálók megvalósítása szempontjából közömbös. A számlálási állapotokat a tárolók száma és a visszacsatolás módja határozza meg.



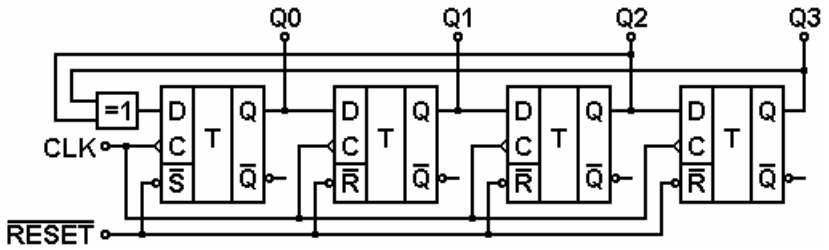
N-ből 1 kódban működő számláló tárolóinak állapota egy kivétellel azonos. A számlálási állapotokat az különbözteti meg, hogy az eltérő bit éppen melyik helyértéken van. Az n-ből 1 számláló egy n bites balra vagy jobbra léptető shiftregiszterből áll, melynek legnagyobb vagy legkisebb helyértékű kimenete van visszacsatolva a soros adatbemenetre. Az eltérő bitet a RESET jellel kell beállítani. A számláló modolusa a tárolók számával (M = n) egyezik meg. A számláló előnyös tulajdonsága a dekódolt azaz decimális (!) kimenet. Egy 4 bites n-ből 1 gyűrűs számláló $Q_3Q_2Q_1Q_0$ kimenetén megjelenő állapotok:

$$\rightarrow 0001 \rightarrow 0010 \rightarrow 0100 \rightarrow 1000 \rightarrow$$



Johnson, más néven Möbius számláló megegyezik az n-ből 1 kódban működő számláló felépítésével, de a soros adatbemenetre a legnagyobb vagy legkisebb helyértékű kimenet negáltja van visszacsatolva. A RESET jel hatására minden tároló törlődik, majd az órajel ütemében a tárolók rendre 1-el, majd a negált visszacsatolás miatt 0-val töltődnek fel. A számláló modolusa M = 2n a tárolók számának kétszeresével egyezik meg. Az 5 bites Johnson számláló a Johnson BCD kódot állítja elő. Egy 4 bites Johnson számláló $Q_3Q_2Q_1Q_0$ kimenetén megjelenő állapotok:

$$0000 \rightarrow 0001 \rightarrow 0011 \rightarrow 0111 \rightarrow 1111 \rightarrow 1110 \rightarrow 1100 \rightarrow 1000.$$



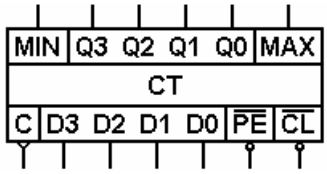
Maximális hosszúságú számláló szintén megegyezik az n-ből 1 kódban működő számláló felépítésével, de a soros adatbemenetre a két legnagyobb helyértékű kimenet antivalencia kapcsolata van visszacsatolva. A számláló a 0. állapot kivételével az összes lehetséges állapotot (ál véletlenül) felveszi. Egy 4 bites számláló $Q_3Q_2Q_1Q_0$ kimenetén RESET után meg-jelenő álla-

potok decimálisan:

1—>2—>4—>9—>3—>6—>13—>10—>5—>11—>7—>15—>14—>12—>8.

Programozható számlálók

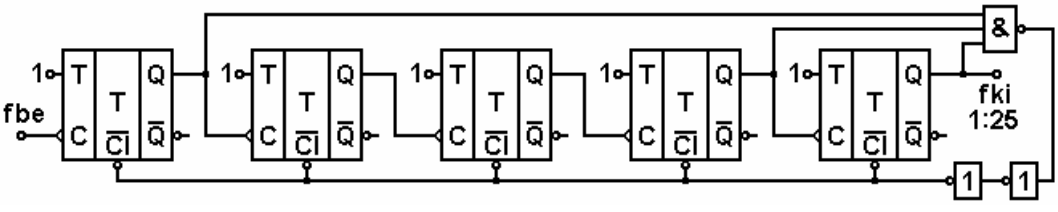
A programozható számlálók általában egy szinkron, négybites bináris előre vagy hátra számlálót tartalmaznak, melynek tárolói a CLear-rel törölhetők, a Paralell Enable-el a D bemenetek értékére beállíthatók. A programozhatóság azt jelenti, hogy egy számlálási állapot elérésekor egy külső kombinációs hálózat a számlálót törli vagy új kezdőértékre állítja be. A CL és a PE bemenetek szinkron vagy aszinkron működésűek lehetnek. Szinkron működés esetén a törlés vagy a párhuzamos betöltés az órajellel szinkronozottan, az aktív jelszint megjelenése után a következő órajel hatására következik be. Aszinkron működés esetén a törlés vagy a párhuzamos betöltés az órajeltől függetlenül, az aktív jelszint megjelenésekor azonnal megtörténik A számlálási kezdőértéket (0000) a MIN, a végértéket (1111) a MAX kimenetek jelzik egy órajel periódus időtartamára. A programozható számlálókkal különböző frekvenciájú és kitöltési tényezőjű impulzus sorozat állítható elő.



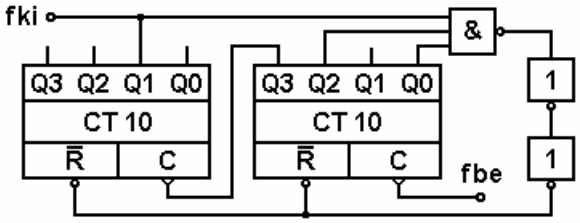
Frekvenciaosztók

A frekvenciaosztók olyan szekvenciális hálózatok, melyek 1 db. kimeneti impulzust állítanak elő N db. beérkezett órajel hatására. Mivel a kimeneti impulzussorozat periódus ideje az órajel periódus idejének N-szerese, azaz $T_{ki} = N \cdot T_{be}$, ezért a kimeneti impulzussorozat frekvenciája az órajel frekvenciájának N-ed része, tehát $f_{ki} = f_{be} / N$. A frekvenciaosztók jellemzője az $f_{ki} : f_{be} = 1 : N$ frekvenciaosztási tényező. Felépítésükre nézve, alapvetően szinkron vagy aszinkron működésű, bináris vagy decimális számlálók, melyeknek csak a legnagyobb helyértékű kimenete kerül felhasználásra. A megfelelő frekvenciaosztási tényezőt ciklusrövidítéssel, a kimeneti impulzussorozat kitöltési tényezőjét ciklusmódosítással $1 / N$ -től $(N - 1) / N$ -ig lehet beállítani, de a kitöltési tényező általában közömbös.

Bináris számlálóval való megvalósítás esetén - szinkron vagy aszinkron működéstől függetlenül - az $N \le 2^n - 1$ összefüggés alapján, n db. T típusú tárolóra van szükség, azaz amennyi az N decimális szám bináris alakú számjegyeinek száma. A ciklusrövidítő hálózat egy NAND kapu, melynek bemeneteire azoknak a tárolóknak a ponált kimenetei csatlakoznak, melyek az N szám bináris kódjának elérésekor beírt állapotban vannak, és a kimenete a tárolók a Clear bemenetét vezérli. Ha az N 2 hatványa, akkor nem kell visszacsatoló hálózat.



Decimális számlálóval való megvalósítás esetén annyi dekadikus számlálóra van szükség, amennyi az N decimális szám számjegyeinek száma. A visszacsatoló hálózat ebben az esetben is NAND kapu, melynek bemeneteire a számlálók azon kimenetei csatlakoznak, melyek az N szám egyenes BCD kódjának elérésekor 1 értékűek, és a kimenete a számlálók Reset bemenetét vezérli. Ha az N 10 hatványa, akkor nem kell visszacsatoló hálózat.



Mindkét kapcsolási rajz 1:25 frekvenciaosztót mutat be. Az N számnak megfelelő állapot az áramköri késleltetések időtartamára megjelenik a tárolók kimenetein. A stabil működés érdekében a törlőimpulzust célszerű két inverter beiktatásával megnyújtani. A bemeneti frekvencia maximális étékét a számlálók és a visszacsatoló hálózat késleltetési ideje alapján lehet meghatározni.

Számrendszerek közötti átalakítás

Feladat: $47,4_{(10)}$ átalakítása r = 3 számrendszerbe.

Egészrész átalakítása:

47 / 3	=	15,6666	0,6666	* 3 = 2
15 / 3	=	5,0	0,0	* 3 = 0
5 / 3	=	1,6666	0,6666	* 3 = 2
1				

Törtrész átalakítása:

0,4	* 3 =	1,2	1
0,2	* 3 =	0,6	0
0,6	* 3 =	1,8	1
0,8	* 3 =	2,4	2
0,4	* 3 =	1,2	1 (ismétlődik)

Megoldás: $47,4_{(10)} = 1202,10121..._{(3)}$

Ellenőrzés: $1*3^3 + 2*3^2 + 2*3^0 + 1*3^{-1} + 1*3^{-3} + 2*3^{-4} + 1*3^{-5} = 47,39917...$

Feladat: $9A3,52F_{(16)}$ átalakítása r = 2-as és r = 8-es számrendszerbe.

Megoldás:

a bináris átalakításhoz a hexadecimális számjegyek bináris kódját kell összefűzni:

az oktális átalakításhoz a bináris számot hármas csoportokra kell bontani:

9 A 3 , 5 2 F	hexadecimális
1001 1010 0011 , 0101 0010 1111	bináris
100 110 100 011 , 010 100 101 111	bináris
4 6 4 3 , 2 4 5 7	oktális

A bemutatott átalakítás a három számrendszer között, visszafelé is elvégezhető!

Számítógépekben alkalmazott számábrázolások

A számítógépben a numerikus információ leképezése, BCD vagy bináris kód hozzárendelésével lehetséges. A BCD kód meglehetősen terjengős, egy byte-on ún. pakolt BCD formátumban, mindössze 0-tól 99-ig terjedő decimális tartomány képezhető le. Az előjel megadásához külön bitekkel kell gondosodni. A műveletek (összeadás, kivonás, szorzás, osztás) végzéséhez programot kell írni, bár a processzorok utasítás készletében létezik BCD összeadás és túlcsordulás esetén az eredményt korrigáló művelet is. (lásd: Wikipédia BCD műveletek)

A bináris kóddal történő leképezés a gyakorlatban elterjedtebb, mivel ugyanakkora bitszámmal, lényegesen nagyobb decimális tartomány fedhető le. Alapvetően két leképezés lett kifejlesztve: a fixpontos és a lebegőpontos.

A fixpontos ábrázolás esetén a bináris tizedespont, azaz a kettedes pont helye rögzített.

- Ha a kettedes pont a bináris szám jobb oldalán (a legkisebb helyértéken) helyezkedik el, akkor az ábrázolás fixpontos egész típusú.
- Ha a kettedes pont valahol a bináris számban, azt egész illetve tört részre kettéosztva helyezkedik el, akkor az ábrázolás fixpontos tört(es) típusú.
- A kettedes pont teljesen balra helyezésével, kizárólag tört számokat lehet ábrázolni. Gyakorlati jelentősége a lebegőpontos ábrázolásban van.

A fixpontos egész típusú ábrázolás esetén a bináris számot, a decimális értékéhez rendeljük hozzá. Ha regiszter (a processzor műveletvégző egységének) mérete 8 bites, akkor ez a 0_d-hoz a 00_h és a 255_d-hoz az FF_h hozzárendelését jelenti. Ez a hozzárendelés nagyon egyszerű, de csak pozitív számok leképezésére alkalmas.

255_d	FF_h
...	...
0_d	00_h

Negatív számok ábrázolásához az a lehetőség kínálkozik, hogy 8 biten az összes lehetséges, tehát a 256 féle kombinációból, az egyik fele 0-val, a másik fele 1-el kezdődik. A legnagyobb helyérték (Most Significant Bit) alkalmas lehet az előjel megkülönböztetésére. A pozitív számot az MSB=0, a negatív számot az MSB=1 jelenti. Ezt az ábrázolást nevezzük előjeles (vagy elő jegyes) abszolút értékes típusúnak. Kedvezőtlen az ábrázolási tartomány ±127-re csökkenése és a negatív 0 megjelenése is. További jelentős hibája ennek az ábrázolási módnak, hogy pl. +1 és –1 összege nem egyenlő 0-val. Műveletvégzésre tehát nem alkalmas, de jól használható a lebegő pontos ábrázolásnál a mantissza megadására. Ehhez az ábrázoláshoz hasonló eredményhez jutunk, ha egy 0-val kezdődő bináris szám komplemensét tekintjük negatívnak. Egy bináris szám komplemense alatt a teli egyesre, azaz a legnagyobb ábrázolhatóra való kiegészítőt jelenti. Négybites példán bemutatva tehát az 1100-t az 1111-re, a 0011 egészíti ki. Ez a kiegészítés, azaz az egyes komplemens képzése, végső soron bitenkénti invertálás művelettel

-127_d	FF_h
...	...
-0	80_h
127_d	$7F_h$
...	...
0_d	00_h

hajtható végre, ami áramkörileg (inverterekkel) egyszerűen megoldható. Ezt az ábrázolást nevezzük egyes komplemens típusúnak. Megfigyelhető, hogy az előjeles abszolút értékes ábrázolás negatív tartománya fordított sorrendben rendelődik hozzá a decimális értékekhez. A negatív 0 változatlanul megmaradt, matematikailag értelmezhetetlen. Ha ebben az ábrázolásban egy bináris számot a saját komplemenséhez, azaz a negatív előjellel vett értékéhez adunk hozzá, akkor teli egyest, azaz negatív 0-t kapunk. A negatív 0 egy 1-es hozzáadásával pozitív 0-vá alakítható. Egy bináris számot a negatív előjelű értékéhez hozzáadva helyes eredményt, tehát pozitív 0-t kapunk, ha a negatív számot az egyes komplemens előállításán kívül 1 hozzáadásával képezzük. Másképpen fogalmazva egy bináris pozitív szám negatívját a szám következő nagyságrendre (8 biten 1 0000 0000-ra) kiegészítőjeként értelmezzük. Ezt az ábrázolást nevezzük kettes komplemens típusúnak. A negatív decimális értékekhez is 1-et hozzáadva, a negatív nulla –1-re, a –127 pedig –128-ra módosul. A kettes komplemens ábrázolásban az FF_h-t -1_d-nak, a 80_h-t pedig -128_d-nak értelmezzük.

-127_d	80_h
...	...
-0	FF_h
127_d	$7F_h$
...	...
0_d	00_h

-128_d	80_h
...	...
-1	FF_h
127_d	$7F_h$
...	...
0_d	00_h

Példa a kettes komplemens képzésre:

a pozitív bináris szám:	01010110_b	$+86_d$
az egyes komplemens:	10101001_b	
+ 1:	1_b	
a kettes komplemens:	10101010_b	-86_d

A pozitív bináris számból egyszerűen és gyorsan képezhető a kettes komplemens a következő eljárással: jobbról balra haladva leírom az összes 0-t és az első 1-est, majd bitenként invertálom! Példa a gyors kettes komplemens képzésre:

a pozitív bináris szám: 01100100b kettes komplemens: 10011100b

Egy kettes komplemens ábrázolású negatív szám értékét ismételt kettes komplemens átalakítással lehet meghatározni. Példa a negatív szám abszolút értékének meghatározására:

a kettes komplemens: 10110000b abszolút értéke: 01010000b 80d

(megjegyzés: a 80_h-nak 80_h a kettes komplemense !)

Végül a fixpontos egész típusú ábrázolás utolsó típusa az ún. eltolt nullapontos. Ebben az ábrázolásban a 0 valahol az értéktartomány közepén helyezkedik el, ettől fölfelé a számokat pozitívnak lefelé pedig negatívnak értelmezzük. A 0-hoz tartozó eltolási érték (displace-ment) esetenként változik. Például 7 bites esetben az eltolás 63_d vagy 64_d, 8 bites esetben 127_d vagy 128_d, 10 bites esetben 511_d vagy 512_d lehet. Az eltolt nullapontos leképezés tipikusan a lebegőpontos ábrázolásnál a karakte-

$+127_d$	FF_h
...	...
$+1_d$	81_h
0_d	80_h
-1_d	$7F_h$
...	...
-127_d	00_h

risztika megadására használatos.

A fixpontos tört(es) ábrázolásnál a kettedes pont helye rögzített és a bináris szám egész és tört részét választja el. Az egész illetve a tört rész számára fenntartott bitek száma egyedileg változhat. A negatív számok ábrázolása kettes komplemens kódban történik. Az előjelet az egész rész MSB-je az S bit tartalmazza. Egy n bites fixpontos tört általános sémája:

2^{n-1}	2^{n-2}	2^0
S	Egész	Tört

A felhasználás szempontjából a számok ábrázolásának két fontos jellemzője van:

az ábrázolható tartomány

az ábrázolás pontossága

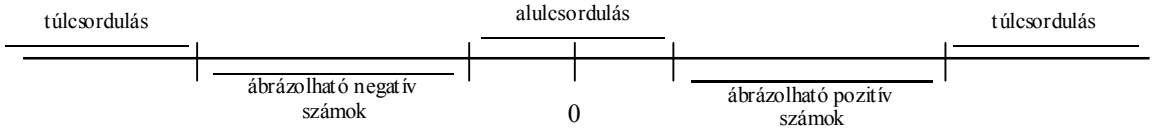
A két jellemzőt a kettedes pont helye határozza meg és az egyik csak a másik rovására növelhető. Ha a kettedes pontot jobbra mozgatjuk, akkor növekszik az egész rész bitjeinek száma és így az ábrázolási tartomány is, de csökken a tört rész bitjeinek száma és így az ábrázolás pontossága is. Ha a kettedes pontot balra mozgatjuk, akkor csökken az ábrázolási tartomány, de növekszik az ábrázolás pontossága. Az, hogy hány biten, mekkora tartománnyal és pontossággal történik az ábrázolás, a programozó döntésén múlik.

Egy 16 bites lehetséges kialakítás pl. 10 bit az egész rész és 6 bit a tört rész:

b_{15}		b_0
S(1)	Egész(9)	Tört(6)

az ábrázolható legkisebb pozitív szám:	0 000000000,000001	0,015625
az ábrázolható legnagyobb pozitív szám:	0 111111111,111111	511,984375
az ábrázolható legkisebb negatív szám:	1 000000000,000000	-512,0
az ábrázolható legnagyobb negatív szám:	1 111111111,111111	-0,015625

Ennek a 16 bites fixpontos tört ábrázolásnak a pontossága $0{,}015625_d$, ami azt jelenti, hogy ennél kisebb szám már nem ábrázolható és két szomszédos szám közötti különbség is csak ennyi lehet. Más szóval ez az érték a decimális szám felbontását jelenti, azaz bármely decimális számot ennek a többszöröseinek összegéből tudjuk összerakni. Az ábrázolási tartomány közelítőleg $\pm 512_d$. A tartományt számegyenesen jelölve:



A fixpontos tört ábrázolásnál a kettes komplemens képzése a korábban leírt módon, ugyanúgy történik, mint az egész típusú ábrázolásnál. Az átalakításkor figyelmen kívül kell hagyni a kettedes pontot, és a „leírom az összes nullát és az első 1-est, majd bitenkénti "invertálás” műveletet jobbról balra, a törtrész legkisebb helyértékű bitjével kell kezdeni. Érdemes megfigyelni, hogy míg az egész típusú ábrázolás tartományának szélső értékei között ab-

szolút értékben 1 az eltérés (pl. 128-127=1), addig az előző példában ez az eltérés 1/64=0,015625, azaz a legkisebb helyértékű biten ábrázolt szám.
A negatív számok kettes komplemens kódban való alkalmazása, azzal a jelentős előnnyel jár, hogy a kivonás művelete összeadásra vezethető vissza. Ebből az következik, hogy a processzorok aritmetikai egységét elég csak összeadásra „megtanítani". Gyakorlatilag a kivonás műveletet, a kivonandó kettes komplemensének hozzáadásával lehet elvégezni:

kisebbítendő - kivonandó = különbség $\rightarrow$ kisebbítendő + kivonandó2k = különbség

Néhány példa az összeadás és kivonás műveletekre:

két pozitív szám összege:
(+a) + (+b) = c
(az eredmény pozitív)

01001001	73
+ 00101101	+ 45
01110110	118

egy nagyobb és egy kisebb szám különbsége,
(+a) - (+b) = (+a)+(-b2k) = d
(a túlcsordulás figyelmen kívül hagyható)

01001001	73
+ 11010011	- 45
1 00011100	28

egy kisebb és egy nagyobb szám különbsége:
(+a) - (+b) = (+a)+(-b2k) = e
(az eredmény negatív, abszolút értéke 2k-el határozható meg)

00101101	45
+ 10110111	- 73
11100100	- 28

két negatív szám összege:
(-a2k) + (-b2k) = f
(az eredmény negatív, abszolút értéke 2k-el határozható meg)

11010011	- 45
+ 10110111	- 73
1 10001010	-118

BASIC, PASCAL, C+, stb. alkalmazott egész típusú számábrázolások határértékei:

8 biten	+127	-128
16 biten:	+32 767	-32768
32 biten:	+2 147 483 647	-2 147 483 648

A lebegőpontos ábrázolás igen nagy (pl. 1 C = 6,25 10^{18} db. e^-) és igen kicsi (pl. ε_0= 8,86 10^{-12} As/Vm) számok ábrázolását teszi lehetővé. Az ábrázoláshoz az értékes számjegyeket azaz a mantisszát, a hatványkitevőt azaz a karakterisztikát, és a szám előjelét külön bináris adatok rögzítik. Egy decimális szám leképezéséhez első lépésben az előjelétől függetlenül, a bináris megfelelőjét kell előállítani. A kapott bináris szám fixpontos egész, fixpontos tört esetleg előjeles abszolút értékes lehet. Ezt a bináris számot az egységes ábrázolás érdekében normalizálni kell. A normalizálással a bináris számot, a kettedes pontjának áthelyezé-

sével, olyan fixpontos törtre alakítjuk, amelynek az egész része 0 vagy 1. A kettedes pont áthelyezésének száma adja meg azt a hatványkitevőt, mellyel az eredeti bináris számot később elő lehet állítani. Lebegőpontos ábrázolás képletei:

egészre normalizált: $N = (-1)^S \cdot 1,M \cdot 2^{C-D}$

törtre normalizált: $N = (-1)^S \cdot 0,M \cdot 2^{C-D}$

Jelölések értelmezése: N - az ábrázolandó bináris szám
S - előjel bit (0: pozitív, 1: negatív)
M - mantissza (általában előjeles abszolút értékes)
C - eltolt nullapontos karakterisztika
D - a karakterisztika eltolási értéke

A lebegőpontos ábrázolás általános n bites formátuma:

2^{n-1}		2^0
S	C	M

Egészre vagy törtre normalizálás után a kettedes ponttól balra vagy jobbra minden esetben 1-es áll. Ezt az 1-est implicit bitnek nevezzük. Mivel csak a műveletvégzéshez szükséges, ezért eltárolás előtt kivéve a bitsorból, a mantissza pontossága 1 bittel növelhető.

Példák a lebegőpontos ábrázolásra: 1. példa:

a decimális szám: - 159,1875

a formátum: S előjel, 1 bit
C 127-el eltolt nullapontos karakterisztika, 8 bit
M abszolút értékes, törtre normalizált mantissza, 23 bit

átalakítás binárisra: $159{,}1875_d \rightarrow 10011111{,}0011b$

normalizálás: $10011111{,}0011 \rightarrow 0{,}100111110011 \cdot 2^8$

karakterisztika: C - D = exp. $\rightarrow$ C = exp. + D = 8 + 127 = 10000111b

a szám negatív, az előjel: S = 1

az ábrázolt szám bitképe: 1 10000111 00111110011000000000000

hexadecimális alakban: C39F3000

2. példa:

a decimális szám: 0,025

a formátum: S előjel, 1 bit
C 128-al eltolt nullapontos karakterisztika, 8 bit
M abszolút értékes, egészre normalizált mantissza, 23 bit

átalakítás binárisra: $0{,}025_d \rightarrow 0{,}0000011001100110011001100110011_b...$

normalizálás: $0{,}0000011001100... \rightarrow 1{,}100110011... \cdot 2^{-6}$

karakterisztika: C - D = exp. $\rightarrow$ C = exp. + D = -6 + 128 = 01111010b

a szám pozitív, az előjel: S = 0

az ábrázolt szám bitképe: 0 01111010 10011001100110011001100

hexadecimális alakban: 3D4CCCCC

Szabványosított (Intel, Motorola, stb. által támogatott) ábrázolási formátumok:

szimpla pontosságú:	(32 bites)	S(1)	C(8)	D=127	M(23)
dupla pontosságú:	(64 bites)	S(1)	C(11)	D=1023	M(52)
belső pontosságú:	(80 bites)	S(1)	C(15)	D=16383	M(64)

(bővebben: Vikipédia - IEEE lebegőpontos számformátum)

Lebegőpontos számokkal való műveletvégzéshez, a memóriából kivett bitsorozatba, vissza kell helyezni az implicit bitet, majd közös karakterisztikára kell állítani a számokat. A nagyobb szám hatványkitevőjét először meg kell növelni 1-el, hogy helyet biztosítsunk a mantisszában esetleg keletkező átvitelhez, majd ehhez a hatványkitevőhöz kell a kisebb szám karakterisztikáját hozzáigazítani. A dolog magyarázatául figyeljük meg a következő példát: $2 \cdot 10^3 + 5 \cdot 10^2 = 2 \cdot 10^3 + 0{,}5 \cdot 10^3 = 2{,}5 \cdot 10^3$. Összeadás esetén a mantisszákat össze kell adni, kivonáskor a kivonandó mantisszájának a kettes komplemensét kell hozzáadni kisebbítendőhöz. A műveletvégzés után az eredményt újra normalizálni kell.

Szimpla pontos ábrázolás határértékei:

$$-3{,}402823 \cdot 10^{38} \leftarrow -2{,}802597 \cdot 10^{-45} \leftrightarrow 2{,}802597 \cdot 10^{-45} \rightarrow 3{,}402823 \cdot 10^{38}$$

Kombinációs hálózat tervezése

Egy logikai egységnek két X és Y vezérlő valamint két A és B adat bemenete van. A mellékelt táblázat tartalmazza, hogy a vezérlő jelek állapotának függvényében, milyen műveletet kell az adatokkal végezni. Az X legyen a legnagyobb helyértékű változó.

Feladatok: függvény ábrázolása igazságtáblázattal, diszjunktív minimál alak meghatározása V-K táblával, realizálás NAND elemekkel.

XY	F
00	$\overline{A}$
01	$\overline{A \cdot B}$
10	$A \oplus B$
11	$\overline{A} \cdot B$

XYAB	Q
0000	1
0001	1
0010	0
0011	0
0100	1
0101	1
0110	1
0111	0
1000	0
1001	1
1010	1
1011	0
1100	0
1101	1
1110	0
1111	0

XY \ AB	00	01	11	10
00	**1**	**1**		
01	**1**	**1**		**1**
11		**1**		
10		**1**		**1**

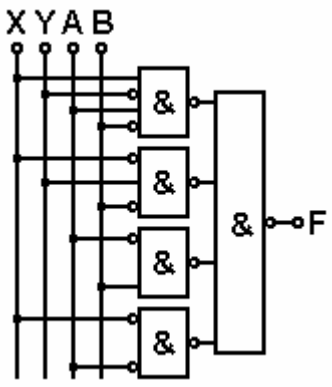
Q

$$F^4 = \overline{X} \cdot \overline{A} + \overline{A} \cdot B + \overline{X} \cdot Y \cdot \overline{B} + X \cdot \overline{Y} \cdot A \cdot \overline{B}$$

átalakítás nand elemekre:

$$F^4 = \overline{\overline{\overline{X} \cdot \overline{A}} \cdot \overline{\overline{A} \cdot B} \cdot \overline{\overline{X} \cdot Y \cdot \overline{B}} \cdot \overline{X \cdot \overline{Y} \cdot A \cdot \overline{B}}}$$

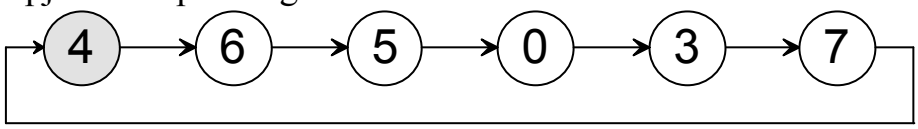
logikai vázlat:



Szinkron hálózat tervezése

Tetszőleges ciklusú szinkron hálózat tervezése, negatív élvezérelt JK (2^2), D (2^1), és T (2^0) tárolókkal. A kimenetek jelölése Qa, Qb és Qc! Kezdő állapot a 4.

1. a tervezés alapja az állapot diagram:



2. az állapot átmeneti tábla és a kitöltéshez szükséges állapot átmeneti táblák:

n.				n+1.
Qa Qb Qc	Ja Ka	Tb	Dc	Qa Qb Qc
1 0 0	h 0	1	0	1 1 0
1 1 0	h 0	1	1	1 0 1
1 0 1	h 1	0	0	0 0 0
0 0 0	0 h	1	1	0 1 1
0 1 1	1 h	0	1	1 1 1
1 1 1	h 0	1	0	1 0 0

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

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

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

Az állapot étmeneti tábla kitöltése az állapot diagram alapján az n. és az utána következő n+1. állapotok bejegyzésével történik. A kezdő állapot a 4. ahonnan 6. állapotba lép a számláló, majd a 6.-ból az 5.-be és így tovább. A hurkot a 7.-ből a 4.-be lépés zárja be. Az n. oszlopban az állapotok sorrendje valójában közömbös, értékük növekvő vagy csökkenő sorrendjében is bejegyezhetők, de az n+1. állapotnak is ehhez kell igazodnia. A vezérlő bemenetek értékei a tárolók állapot átmeneti táblázataiból másolhatók ki. Az első sorban a Qa tároló 1-ből, 1-be lép, amihez a JK bemenetek értéke h0. Az utolsó előtti sorban a Qa tároló 0-ból, 1-be lép, amihez a JK bemenetek értéke 1h. Az első sorban a Qb tároló 0-ból, 1-be lép, amihez a T bemenet 1 értéke tartozik. A harmadik tárolóval is ugyanígy kell eljárni. A teljesen kitöltött táblázat alapja a vezérlési függvények meghatározásának.

3. a vezérlési függvények egyszerűsítése V-K táblával:

A táblák kitöltése az n. állapot alapján történik. A nem definiált, jelen esetben a 001 és 010 állapotokat, határozatlannak lehet tekinteni.

Qa \ QbQc	00	01	11	10
0	0	h	1	h
1	h	h	h	h

$$Ja = Qb$$

Qa \ QbQc	00	01	11	10
0	h	h	h	h
1	0	1	0	0

$$Ka = \overline{Qb} \cdot Qc$$

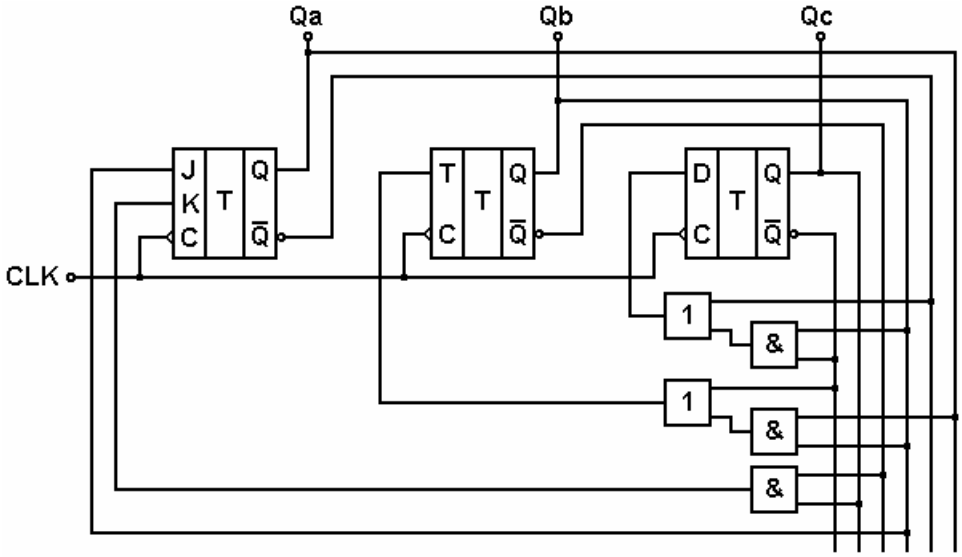
Qa \ QbQc	00	01	11	10
0	1	h	0	h
1	1	0	1	1

$$Tb = \overline{Qc} + Qa \cdot Qb$$

Qa \ QbQc	00	01	11	10
0	1	h	1	h
1	0	0	0	1

$$Dc = \overline{Qa} + Qb \cdot \overline{Qc}$$

4. a logikai vázlat:



5. a nem definiált állapotok vizsgálata:

Az állapot diagramban nem szereplő állapotok határozatlan termként lettek figyelembe véve a minimalizálás során. A már megtervezett hálózat vezérlési függvényei alapján meghatározhatók, hogy a hiányzó vagy nem definiált állapotokban mit fog tenni a hálózat. Ehhez szintén az állapot átmeneti táblázat szükséges, de most az n. állapotba a hiányzó állapotokat kell bejegyezni. A vezérlési függvények megoldásával kapjuk meg a vezérlő bemenetek értékét. Az n+1. állapot az adott tároló vezérkési táblázatából olvasható ki. Ez az eljárás, egyben a szinkron hálózatok elemzésének a módszere is.

n.				n+1.
Qa Qb Qc	Ja Ka	Tb	Dc	Qa Qb Qc
0 0 1	0 1	0	1	0 0 1
0 1 0	1 0	1	1	1 0 1

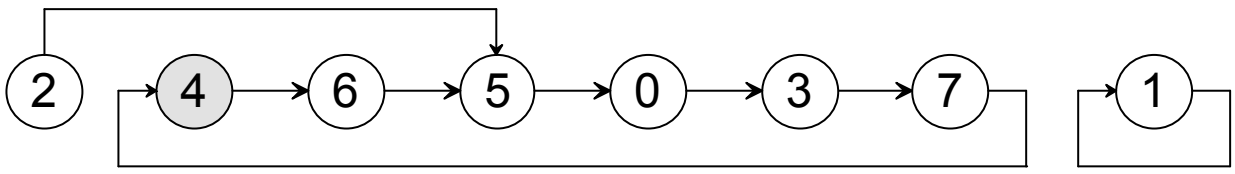
J K	Q^{n+1}
0 0	Q^n
0 1	0
1 0	1
1 1	$\overline{Q^n}$

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

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

A hálózat 1. állapotból önmagába, 2. állapotból, 5. állapotba lép.

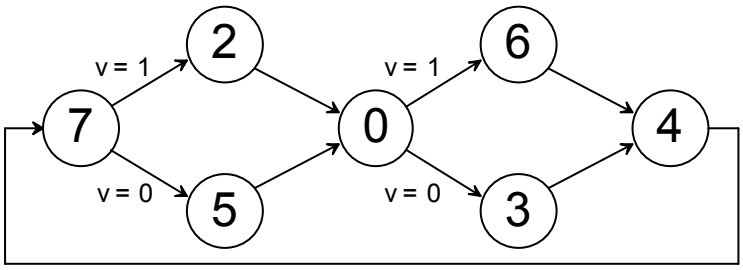
6. a teljes állapotdiagram:



Elágazásos szinkron hálózatok tervezése

Elágazásos hálózat tervezése az adott állapotdiagram szerint, negatív élvezérelt JK tárolókkal és tetszőleges kapuáramkörökkel.

1 a tervezés alapja az állapot diagram:



Az állapot diagramban két elágazás van. Mindkettő a v vezérlőjel értékétől függően ágazik el. A vezérlőjel vizsgálata a 7-es és a 0-ás állapotnál történik. Az elágazáskor v = 0 esetén 7-ből 5-be, illetve 0-ból 3-ba, v =1 esetén 7-ből 2-be, illetve 0-ból 6-ba lép a hálózat. Ha a vezérlőjel bármikor megváltozik, akkor a hálózat működésének követni kell ezt a változást.

2. az állapot átmeneti tábla:

Az állapot átmeneti tábla csak annyiban tér el az előzőktől, hogy az n+1. állapot nemcsak a külső változóktól független, - jelen esetben a középső - oszlopot tartalmazza, hanem az elágazások bejegyzésére alkalmas oszlopokat is. Ezen oszlopokat a külső vezérlőjel, azaz az elágazási feltétel értékei különböztetik meg. A tábla több külső vezérlőjel esetén is alkalmazható, de a vezérlő jeleket ebben az esetben indexelni kell és az indexet a adott elágazási sor végén lehet feltüntetni. A logikai vezérlő bemenetek értékének kitöltése, szintén az adott tároló állapot átmeneti táblája alapján történik, de elágazás

n.					n+1.	
Qa Qb Qc	Ja Ka	Jb Kb	Jc Kc	v = 0	Qa Qb Qc	v = 1
1 1 1	h v	h $\overline{v}$	h v	1 0 1		0 1 0
1 0 1	h 1	0 h	h 1		0 0 0	
0 1 0	0 h	h 1	0 h		0 0 0	
0 0 0	v h	1 h	$\overline{v}$ h	0 1 1		1 1 0
0 1 1	1 h	h 1	h 1		1 0 0	
1 1 0	h 0	h 1	0 h		1 0 0	
1 0 0	h 0	1 h	1 h		1 1 1	

esetén figyelembe kell venni a külső vezérlőjel értékét is. Ehhez célszerű egy segéd táblázatot felállítani. A 7. állapotban az "a" tároló v = 0-nál 1-ből, 1-be és v = 1-nél 1-ből , 0-ba lép. A segédtábla a vezérlő bemenetek értékével:

v = 0	v = 1
h 0	h 1

Ahhoz, hogy az elágazás a "v" vezérlőjeltől függjön a JK bemenetek értékének hv-nek kell lenni. A többi tárolónál ugyanígy kell eljárni. Ha az elágazási két állapotban azonos az "érkezési" érték, például "b" tároló 0-ás állapotban, akkor nem függ a külső jeltől a tároló n+1. állapota.

3. vezérlési függvények meghatározása:

A vezérlési függvényeket az előzőkkel megegyező módon kell V-K táblában ábrázolni a külső vezérlő jelekkel együtt. Az összevonási műveleteket az "Elágazásos szinkron hálózatok" c. részben leírtak szerint elvégezni. Az 1-es vagy 0-ás termeket a lehetséges határozatlan termekkel lehet összevonni, majd a külső változókat tartalmazó cellákat 1-nak vagy 0-nak tekintve az előző 1-es vagy 0-ás összevonásokkal egyesíteni. A külső változó mindig a táblázatbeli értékével szerepel, függetlenül a attól, hogy mintermes vagy maxtermes az összevonás. Az index a mintermes vagy maxtermes alakra utal.

Qa \ QbQc	00	01	11	10
0	v	h	1	0
1	h	h	h	h

$$Ja_1 = Qc + \overline{Qb} \cdot v$$

Qa \ QbQc	00	01	11	10
0	h	h	h	h
1	0	1	v	0

$$Ka_0 = Qc \cdot (\overline{Qb} + v)$$

Qa \ QbQc	00	01	11	10
0	1	h	h	h
1	1	0	h	h

$$Jb_1 = \overline{Qc}$$

Qa \ QbQc	00	01	11	10
0	h	h	1	1
1	h	h	$\overline{v}$	1

$$Kb_1 = \overline{Qa} + \overline{Qc} + \overline{v}$$

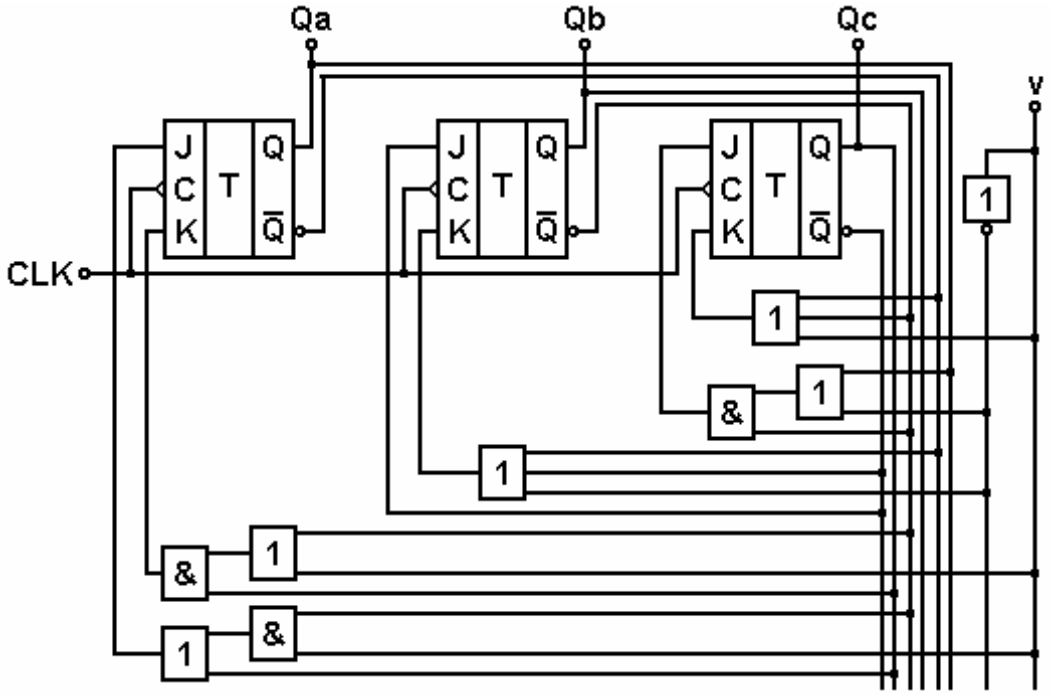
Qa \ QbQc	00	01	11	10
0	$\overline{v}$	h	h	0
1	1	h	h	0

$$Jc_0 = \overline{Qb} \cdot (Qa + \overline{v})$$

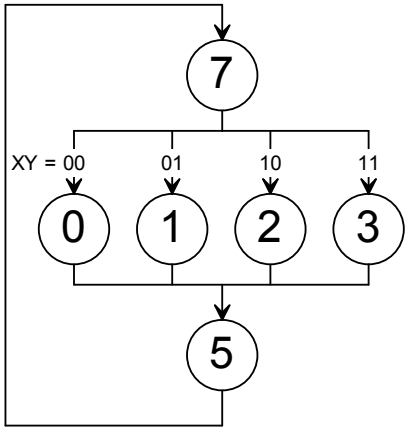
Qa \ QbQc	00	01	11	10
0	h	h	1	h
1	h	1	v	h

$$Kc_1 = \overline{Qa} + \overline{Qb} + v$$

4. a logikai hálózat



Elágazásos hálózat tervezése két külső vezérlőjel esetén:
1. állapotdiagram:



Az egyszerűség érdekében az elágazási állapotok megegyeznek az elágazási feltétellel, tehát XY=00 estén 111-ből a 000 állapotba, XY=01 esetén 111-ből a 001 állapotba és így tovább, lép a hálózat.

Az elágazásos sort kiemelve és az egyes tárolók jeleit egyenként vizsgálva:

Elágazási feltételek:				XY=00	XY=01	XY=10	XY=11
QaQbQc (n)	Ja Ka	Jb Kb	Jc Kc	QaQbQc (n+1)			
111	h 1	h X'	h Y'	000	001	010	011

Az „a" tároló 1-ből 0-ba lép a vezérlőjelektől függetlenül, tehát JaKa = h1.
A „b" tároló a külső vezérlőjelek függvényében veszi fel az állapotokat. XY független változókra és Jb Kb függő változókra igazság táblázatot felállítva, a tároló vezérlő jelei kiolvashatók: Jb = h és Kb = Xnegált (X').
A „c" tároló szintén a külső vezérlőjelek függvényében veszi fel az állapotokat. Az előző igazságtáblázatot Jc Kc - vel kiegészítve: Jc = h és Kc = Ynegált (Y').

XY	Jb Kb	Jc Kc
00	h 1	h 1
01	h 1	h 0
10	h 0	h 1
11	h 0	h 0

Az állapot-átmeneti táblázat az adott állapot-diagramra felírva:

n.						n+1.		
QaQbQc	Ja Ka	Jb Kb	Jc Kc	XY=00	XY=01	QaQbQc	XY=10	XY=11
111	h 1	h X'	h Y'	000	001		010	011
000	1 h	0 h	1 h			101		
001	1 h	0 h	h 0			101		
010	1 h	h 1	1 h			101		
011	1 h	h 1	h 0			101		
101	h 0	1 h	h 0			111		

A hiányzó állapotokat határozatlannak tekintve, a minimalizált vezérlési függvények:

$$Ja = 1 \qquad Jb = Qa \qquad Jc = 1$$

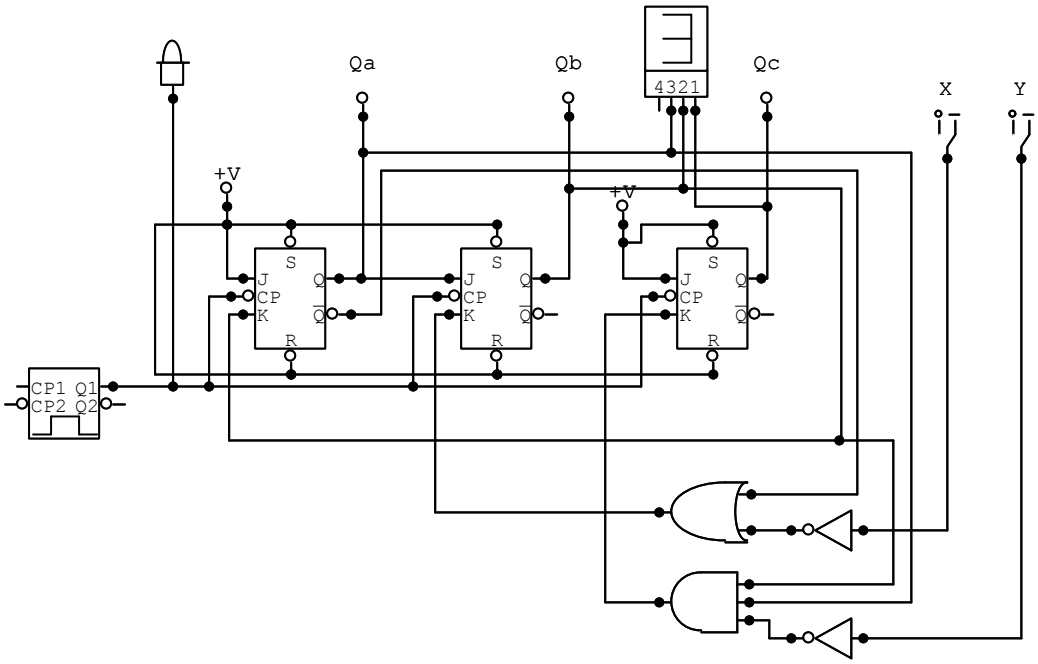
$$Ka = Qb \qquad Kb = \overline{Qa} + \overline{X} \qquad Kc = Qa \cdot Qb \cdot \overline{Y}$$

A minimál alakok a V-K tábla alapján állíthatók elő. Elsőként, a külső vezérlőjelet tartalmazó cellákat figyelmen kívül hagyva, az összes 1-t (0-t) tartalmazó cellákat kell összevon-

ni egymással vagy a h-t tartalmazó cellákkal. Másodszor a külső vezérlőjelet tartalmazó cellákat 1-nek (0-nak) tekintve az 1-et (0-t) és h-t tartalmazó cellákkal kell összevonni. A külső vezérlőjelek minterm esetén „és” kapcsolatban, maxterm esetén „vagy” kapcsolatban vannak az összevonható cellákkal. A vezérlő jelek mindkét estben a cellába írt értékkel szerepelnek a függvényben. Kiegészítve a vezérlési függvényeket:

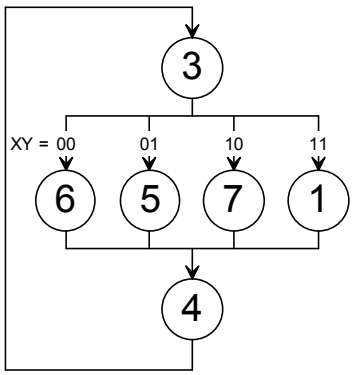
$$Kb = \overline{Qa} + 1 \cdot \overline{X} \qquad Kc = Qa \cdot Qb \cdot (0 + \overline{Y})$$

A kapcsolási rajz: (XY=11)



2. állapotdiagram:

Az előző feladatban az elágazásnál az „érkezési” oldalon 2 db. 1-es és két db. 0-ás bit szerepelt. Ebben a feladatban páratlan számú 1-es illetve 0-ás szerepel az n+1 oldalon. Az elágazásos sort kiemelve és az egyes tárolók jeleit egyenként vizsgálva:



Elágazási feltételek:				XY=00	XY=01	XY=10	XY=11
QaQbQc (n)	Ja Ka	Jb Kb	Jc Kc	QaQbQc (n+1)			
011	(XY)’ h	h Y	h (X+Y)’	110	101	111	001

Az „a” tároló 0-ból indul, 3 helyen 1-be és egy helyen 0-ba érkezik. Az előzőkben felállított igazságtáblát alkalmazva, a vezérlő jelek: Ja = (XY)’ (XnandY) és Ka = h. A „b” tároló 1-ből indul, 2 helyen 0-ba és 2 helyen 1-be érkezik. Az igazság táblázat alapján a vezérlő jelek: Jb = h és Kb = Y. A „c” tároló 1-ből indul, egy helyen 0-ba és 3 helyen 1-be érkezik. A vezérlő jelek szintén az igazság táblázatból: Jc = h és Kc = (X+Y)’ (XnorY).

XY	Ja Ka	Jb Kb	Jc Kc
00	1 h	h 0	h 1
01	1 h	h 1	h 0
10	1 h	h 0	h 0
11	0 h	h 1	h 0

Az állapot-átmeneti táblázat:

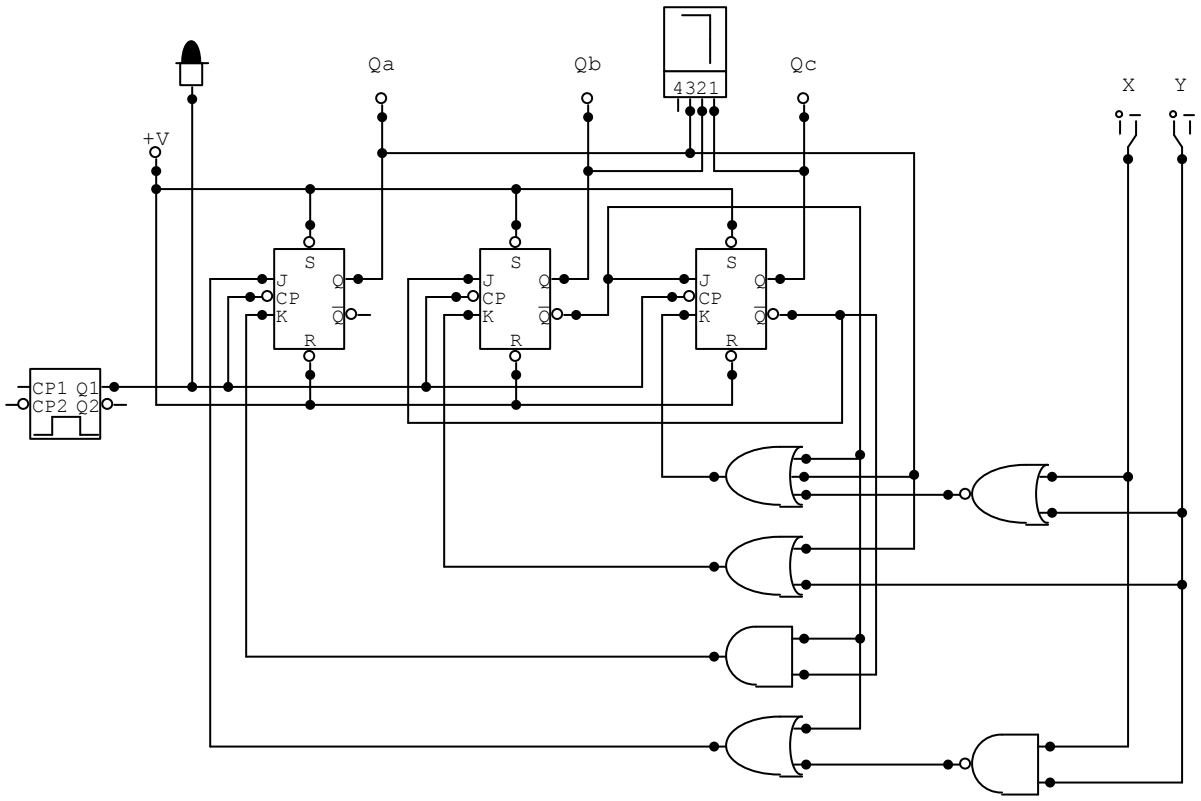
n.						n+1.		
QaQbQc	Ja Ka	Jb Kb	Jc Kc	XY=00	XY=01	QaQbQc	XY=10	XY=11
011	(XY)' h	h Y	h (X+Y)'	110	101		111	001
110	h 0	h 1	0 h			100		
101	h 0	0 h	h 1			100		
111	h 0	h 1	h 1			100		
001	1 h	0 h	h 1			100		
100	h 1	1 h	1 h			011		

A vezérlési függvények:

$$Ja = Qb + \overline{X \cdot Y} \qquad Jb = \overline{Qc} \qquad Jc = \overline{Qb}$$

$$Ka = \overline{Qb} \cdot \overline{Qc} \qquad Kb = Qa + Y \qquad Kc = Qa + Qb + \overline{X \cdot Y}$$

A kapcsolási rajz: (XY=10)



Megjegyzés: ha a V-K táblában több cellában fordul elő külső vezérjel, akkor azokat különkülön egyenként kell összevonni a lehetséges 1-kel vagy 0-kal. Ha nincs összevonási lehetőség, akkor a kérdéses cellát meghatározó termhez kell hozzáfűzni. Például: a teli 0-ás cellában van egy 1-es, a teli 1-es cellában egy „V" külső vezérlőjel, a többi cella értéke 0, tehát mintermes összevonás nem lehetséges. A vezérlési függvény mintermes alakja:

$$Jn = \overline{Qa} \cdot \overline{Qb} \cdot \overline{Qc} + Qa \cdot Qb \cdot Qc \cdot V$$

***