Aritmetikai áramkörök

Aritmetikai áramköröknek nevezzük azokat az áramköröket, amelyek egy-egy teljes szón (logikailag összetartozó, rögzített bitszámú, bit együttesen) hajtanak végre műveleteket.

Ezeknek az áramköröknek két fő csoportja létezik:

1. Aritmetikai áramkörök: Elsődlegesen számolási műveletek (összeadás, kiegészítő érték képzés, vagyis komplementálás, kivonás, szorzás, osztás stb.) elvégzésére szolgáló áramkörök. Ezen áramkörök felépítése függ a használatos számrendszertől.
2. Aritmetikai jellegű áramkörök: Egy szón egyforma műveletcsoportot elvégző áramkörök. Például párosság képző, azonosság és eltérés (<, >) vizsgáló, valamint a kódátalakító áramkörök.

Az aritmetikai műveletek közös jellemzője, hogy 1 számjeggyel végzett műveleteknél a keletkező eredmény nem minden esetben ábrázolható egy számjeggyel, ilyenkor létrejön még egy számjegy, az átvitel, amit a következő helyi értéken lévő számjegyekkel végzett műveleteknél kell figyelembe venni.

pl.
0 plusz 0 = 0
0 plusz 1 = 1
1 plusz 0 = 1
1 plusz 1 = 0 és keletkezik a következő helyiérték összeadásánál figyelembe veendő 1-es átvitel.

Fontos megjegyzés: Jelölések

A digitális technikában használatos aritmetikai műveleteknél, mivel a BOOLE-algebrai leírásban „+” jelet a logikai VAGY művelet jelölésére használjuk, ezért az összeadást, mint aritmetikai műveletet a „plus” szócskával jelöljük. Kivonásnál tulajdonképpen lehetne használni a hagyományosan megszokott „-” jelölést, de többnyire az egységesség miatt a „minus” szócskával jelöljük. Ugyanezen okból, az AB leírás, a logikai szorzást, vagyis a logikai ÉS műveletet jelöli.

Léptetőregiszter felépítése

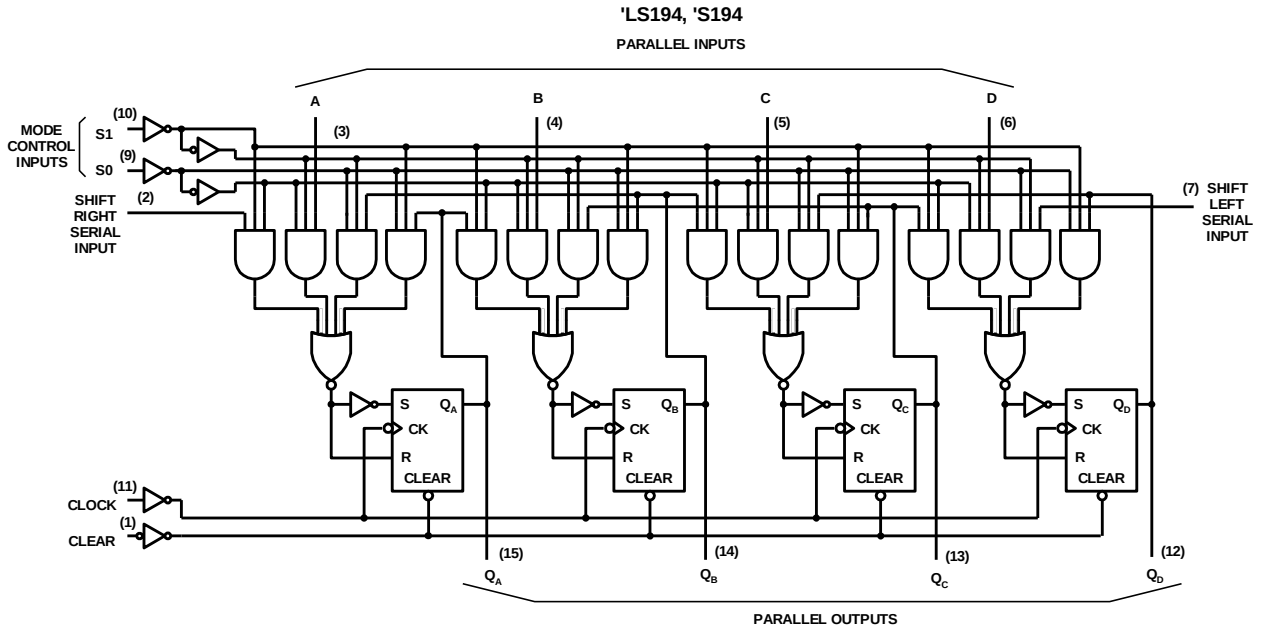
A műveletvégző egységek használatához feltétlen szükségünk van legalább az elvégzett művelet eredményét tároló regiszterre. Ugyanis nem egyedi műveletek, hanem műveletsorok elvégzésére használjuk és ezért feltétlen meg kell őriznünk a korábbi művelet eredményét. Általános igény, hogy a műveletvégzővel mindkét irányú léptetést tudjunk végrehajtani. Az ALU csak egyirányú léptetést tud megvalósítani. Ezért az eredménymegőrző regisztert célszerű léptető regiszterből választani.

Regiszternek nevezzük a több bites, azonos feladatot ellátó tárolók halmazát. Ezek lehetnek, ha csak mintavételezett tárolásra van szükség egyszerű latchek, vagy ha egyéb művelet is szükséges D tárolókból felépített egységek. Léptetőregiszterről beszélünk, ha a párhuzamos beíráson kívül lehetőség van a nagyobb, vagy mint a két helyi érték felé való eltolásra is.

Az univerzális léptetőregiszter üzemmódjai:

- Párhuzamos beírás. Minden bithez önálló bemenet.
- Soros léptetés a nagyobb helyi érték felé. A legkisebb helyi értéknél külső bemenet.
- Soros léptetés a kisebb helyi érték felé. A legnagyobb helyi értéknél külső bemenet.
- Tartás (az órajelre önmaga korábbi értékét tölti be)

Az órajellel léptethető regiszter többnyire rendelkezik aszinkron törlő bemenettel is. A léptetőregiszterek egyes típusait tri-state kimenettel látják el. Egy lehetséges megoldás az alábbi ábrán látható.



S1	S0	Q_A	Q_B	Q_C	Q_D	SR	SL	Művelet megnevezése
0	0	Q_{An-1}	Q_{Bn-1}	Q_{Cn-1}	Q_{Dn-1}	X	X	Tartás (hold)
0	1	0	Q_{An-1}	Q_{Bn-1}	Q_{Cn-1}	0	X	Léptetés nagyobb helyiérték felé (Shift right)
0	1	1	Q_{An-1}	Q_{Bn-1}	Q_{Cn-1}	1	X	“
1	0	Q_{Bn-1}	Q_{Cn-1}	Q_{Dn-1}	0	X	0	Léptetés kisebb helyiérték felé (Shift left)
1	0	Q_{Bn-1}	Q_{Cn-1}	Q_{Dn-1}	1	X	1	“
1	1	A	B	C	D	X	X	Betöltés (Load)

Egy valós, 4 bites, univerzális léptetőregiszter logikai rajza és igazságtáblája.

Elsődlegesen számolási műveleteket elvégzésére szolgáló áramkörök

Vezérelt inverter, vezérelt negáló

Aritmetikai áramkörökben gyakran van szükségünk az előző eredmények negáltjára. Vezérelt negálónak a már ismert KIZÁRÓ-VAGY (vagy duális tulajdonságai miatt az EKVIVALENCIA) kaput használhatjuk.

Működésének igazságtáblája:

V	Be	Ki
0	0	0
0	1	1
1	0	1
1	1	0

Ahol V, a vezérlő bit:

ha V = 0, akkor a kimenet a bemenet értékének ponáltja

ha V = 1, akkor a kimenet a bemenet értékének negáltja

Be = Módosítandó bemenet

Ki = Módosított érték

XOR kapu használata vezérelt inverterként.

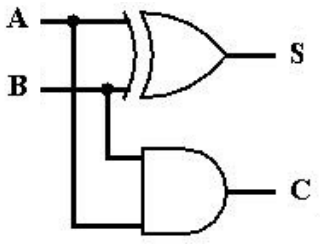
Félösszeadó

Nagyon lényeges része minden CPU-nak (központi egységnek) egy összeadást végrehajtó áramkör. Feladata a bemenetére vezetett A és B bitek összeadása. A félösszeadó két kimenő bitet képez. Egyszer az összegzés (Szummázás) eredményét (S), és a kimeneti átvitel C = Carry output bitet (C).

Az 1 bites egészek összeadásának igazságtáblázatát láthatjuk az alábbi ábrán. Két kimenet van feltüntetve: az A és B bemenő jelek összege (S), valamint az átvitel (carry, C), az a bit, amely továbblép a következő helyi értékű pozícióba. Az igazságtábla alapján látható, hogy az összegzést egy kizáró VAGY, XOR kapu végzi, az átvitelképzést pedig egy AND kapu.

A félösszeadó működésének igazságtáblája és kapcsolási rajza:

A	B	S:Összeg	C: Átvitel
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



A félösszeadó igazságtáblázata és kapcsolási rajza

A félösszeadó hátránya, hogy nem veszi figyelembe az előző helyiértéken keletkező átvitelt.

Teljes összeadó

Feladata a bemenetére vezetett A és B bitek összeadása a bemenetre vezetett átvitel Ci bit figyelembe vételével. A teljes összeadó két kimenő bitet képez. Egyszer az összegzés (Szummázás) eredményét, az Si kimenetet, és a kimeneti átvitel Co = Carry output bitet, Ci+1 formában. Az i index, a helyi érték jelölésére szolgál.

A teljes összeadó igazságtáblája:

A	B	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A működés Karnaugh táblái:

S_i

A \ B \ Ci		1
	1	
		1
1		

C_{i+1}

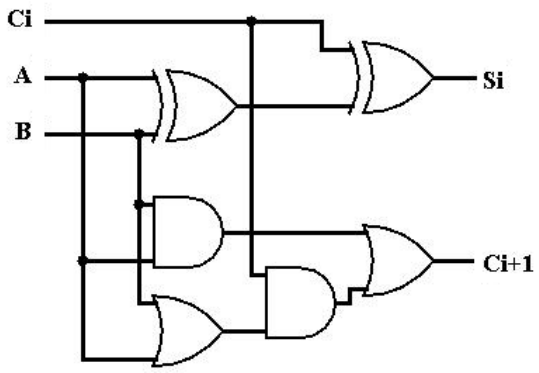
A \ B \ Ci		1
		1
1	1	

A Karnaugh tábla alapján a kimeneti függvények:

$$Si = \overline{A}*\overline{B}*Ci + \overline{A}*B*\overline{Ci} + A*B*Ci + A*\overline{B}*\overline{Ci} = A \oplus B \oplus Ci$$

$$C_{i+1} = A*B + B*C_i + A*C_i$$

Az alábbi ábrán a fenti egyenletet megvalósító hálózat, a teljes összeadó egyik lehetséges megoldása látható:



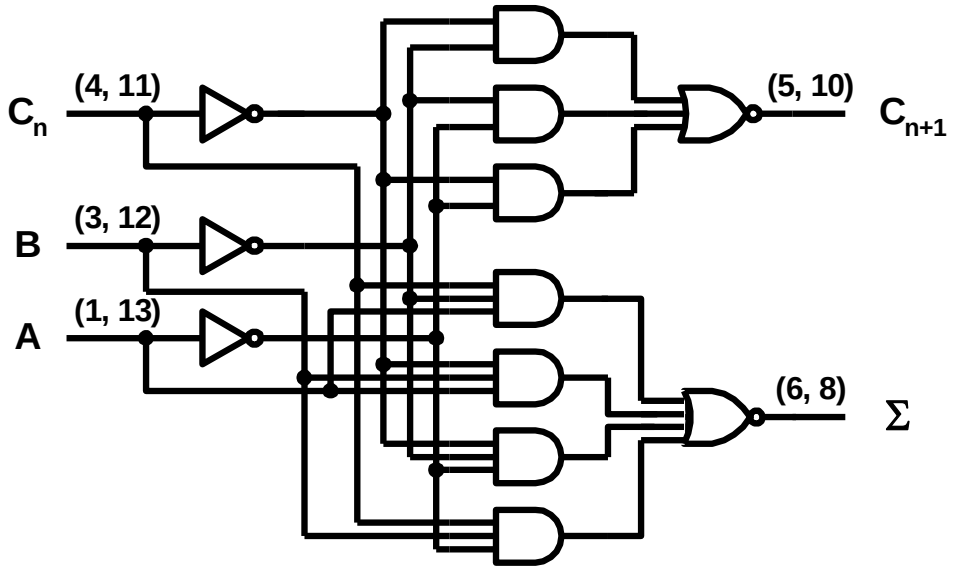
A teljes összeadó egyik lehetséges megoldása

A gyakorlati megvalósítás során mivel az összetett kapu felépítéséből adódóan, az ÉS-VAGY-NEM kapu egyfokozatú késleltetést valósít meg, szemben a kétfokozatú késleltetést adó ÉS-VAGY kapuval, a valós áramköröknél a sebesség igény miatt is, a 0-ra megvalósító áramkört használják. Ennek megfelelően a valós áramkörből megvalósított kapcsolás logikai egyenletei:

$$S = \overline{\overline{A}*\overline{B}*\overline{C_i} + \overline{A}*B*C_i + A*B*\overline{C_i} + A*\overline{B}*C_i}$$

$$C_{i+1} = \overline{\overline{A}*\overline{B} + \overline{A}*\overline{C_i} + \overline{B}*\overline{C_i}}$$

A megvalósítását, vagyis a 74-s sorban létrehozott áramkör (74H183) logikai rajza látható az alábbi ábrán. A kapcsolási rajzon a gyári adatlapnak megfelelő jelzések vannak feltüntetve. A megfeleltetések: Ci = Cn; Ci+1 = Cn+1; és Si = ∑



A gyors működésű egy bites összeadó logikai rajza.

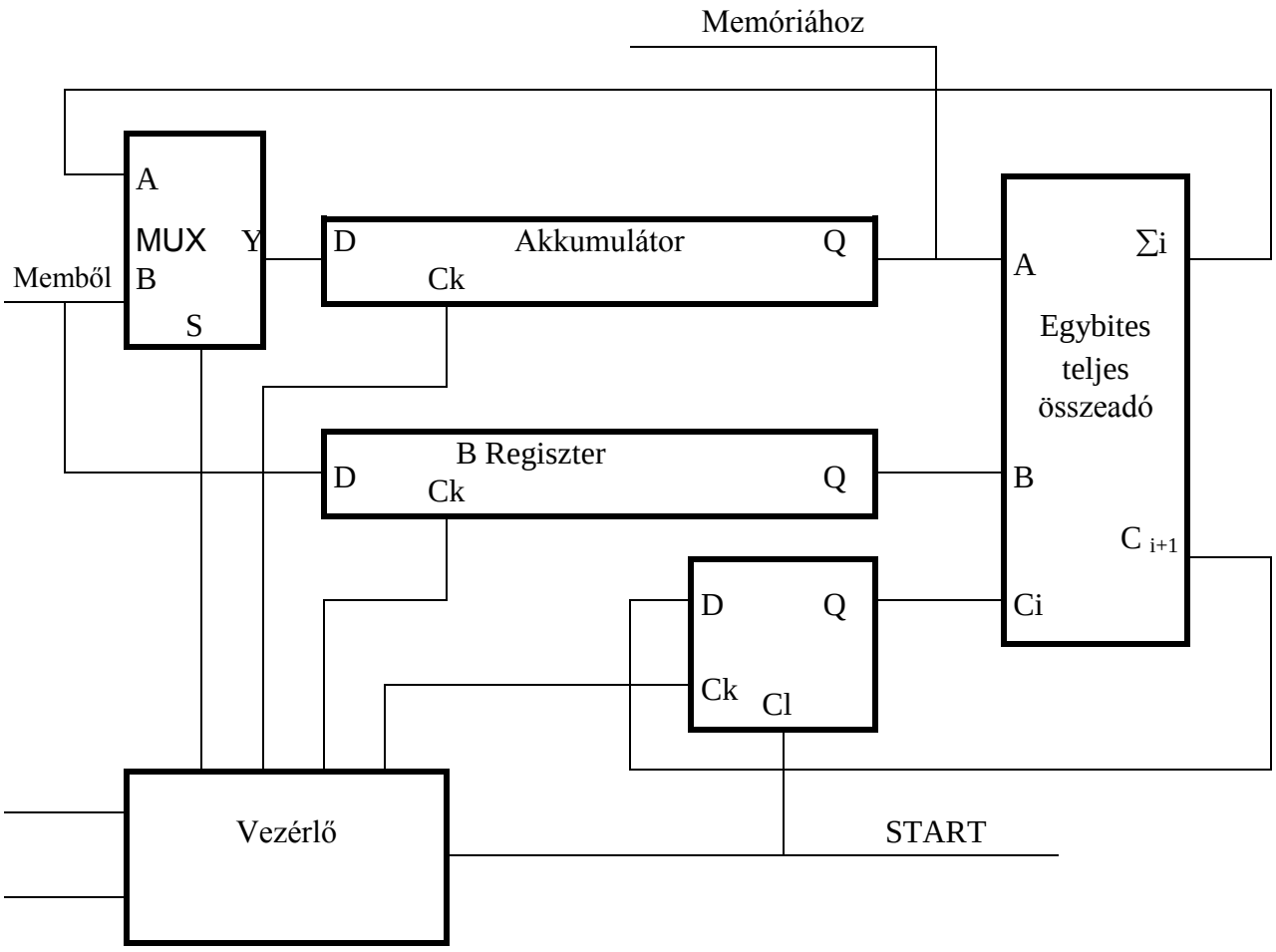
Az 1 bites összeadó, mint alapelem felhasználásával n bitre, alapvetően kétféle módon készíthetünk összeadót.

1. Soros üzemmód: Ekkor az összeadandókat egy-egy léptetőregiszterből (annak kimeneteiről) vesszük, a bemeneti átvitelt pedig egy erre a célra beépített D tárolóból. Az összeadáshoz a legkisebb helyiértékű bitet léptetjük be először az 1 bites teljes összeadóba. Az eredményt az „A" Akkumulátor regiszter belépési pontjához vezetjük, a kimenő átvitelt pedig, a D tár D bemenetéhez. Órajellel léptetjük a rendszert, és az n bites adatot, n lépés után kapjuk meg. A nagyon egyszerű aritmetikai elem használatának előnye mellett, hátránya a rendszernek a lassúsága.
2. Párhuzamos üzemmód soros átvitelképzéssel: Az összeadónak n bitre való megvalósításával párhuzamos összeadóhoz jutunk. Itt az egyes bitek közötti soros átvitel, még lényegesen lassíthatja a működést. Pl. egy 8 bites összeadónál 7 helyen van a szón belül átvitel. E miatt a szó összeadási ideje hétszerese az egy bit összeadási idejének.

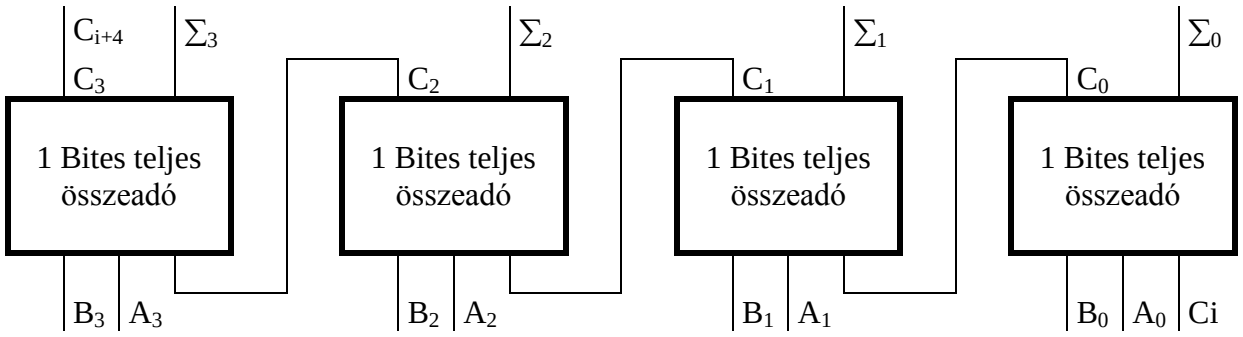
Lényegesen gyorsíthatjuk a működést, ha több bites teljes összeadót valósítunk meg oly módon, hogy a kimeneten az átvitel egy lépésben keletkezzen.

Az alábbi ábrákon a soros és a párhuzamos üzemmód megvalósítását, illetve egy 4 bites gyorsított átvitelű teljes összeadó (a 74.283) kialakítása látható.

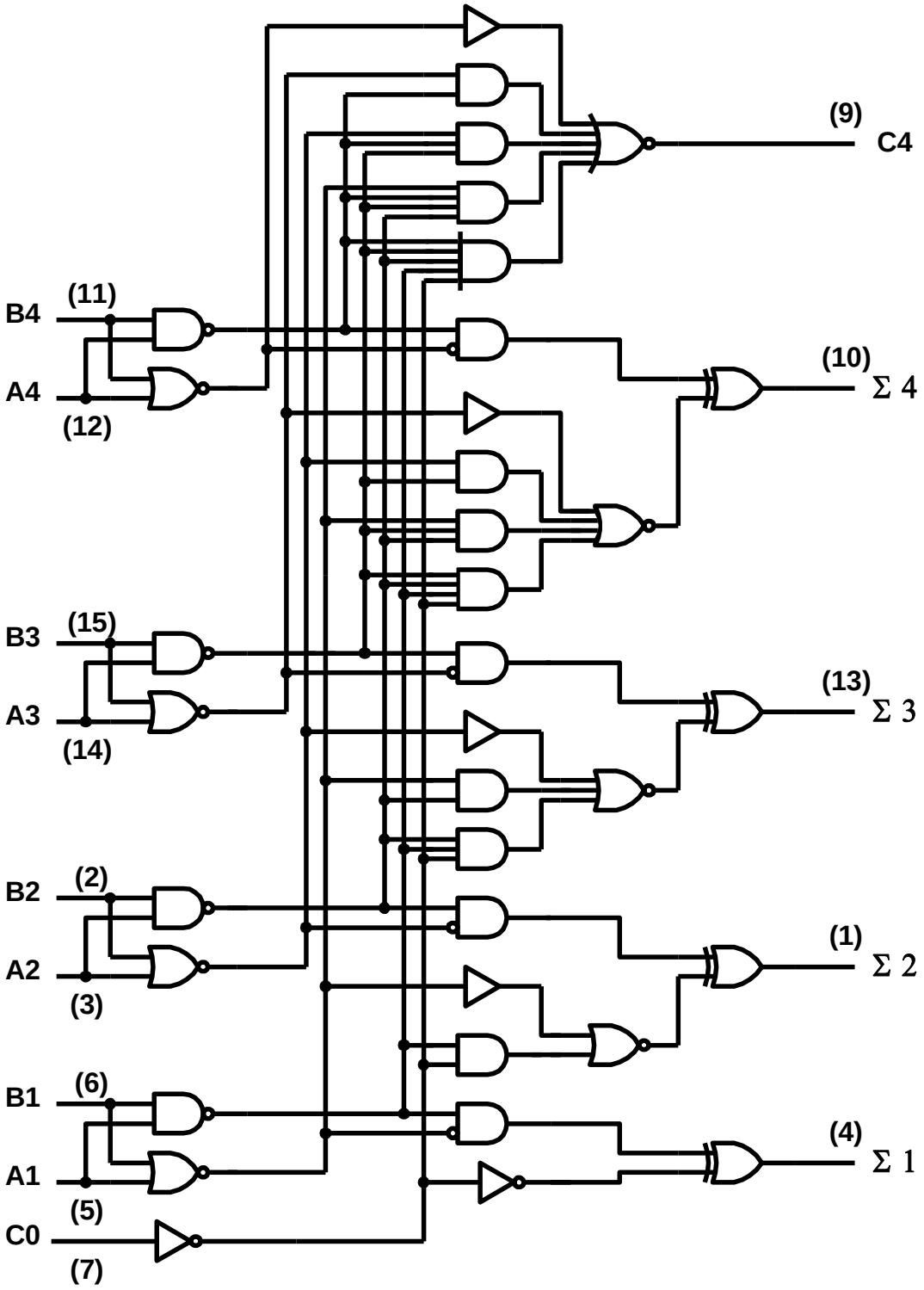
A valós áramkör kapcsolási rajzán megfigyelhető, hogy itt a legrosszabb esetben is 3 fokozattal megoldott kapcsolásról van szó. Természetesen a 4 bites eszközökből, a tetrádokból felépített összeadónál a teljes bitszámra nézve az összeadás idejét itt is az átvitel (Carry) terjedési ideje fogja megszabni.



Soros összeadómű elvi vázlata.



Egybites összeadókból, soros átvitellel kialakított párhuzamos összeadó.



Belső párhuzamos átvitelképzésű 4 bites teljes összeadó

A kivonás folyamata

A kivonáshoz szükségünk van komlemensképzésre is, mivel a kivonást a legtöbb esetben a kivonandó 2-s komplemensével való összeadásra vezetik vissza.

A kivonás helyett a kivonandó, a 3-as szám, kettes komplemensét, ami 1101 adjuk hozzá a 4-hez 0100, vagyis a megvalósítás: Pl. 4-3=?

0100 kisebbítendő
0011 kivonandó
0001 eredmény

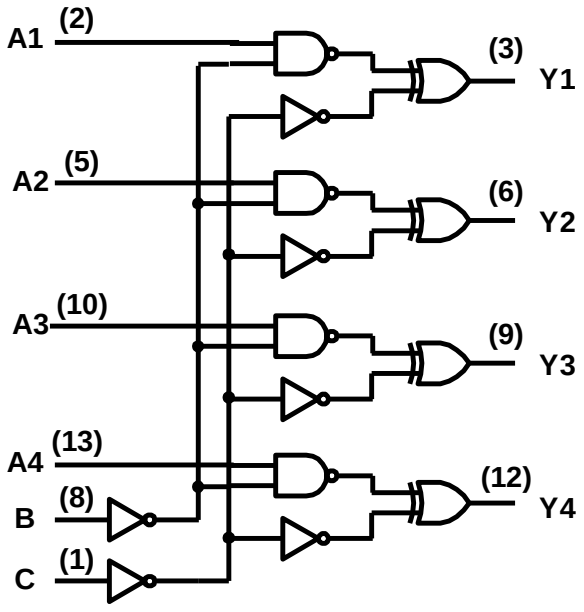
0100 kisebbítendő
1101 kivonandó 2-s komplemense
(1)0001 eredmény az átvitellel

A zárójelben megadott 1-s, a túlcsordulás, amit ha van további nagyobb számérték, ott figyelembe veszünk, ha nincs, akkor az eldobandó kifutó túlcsordulás.

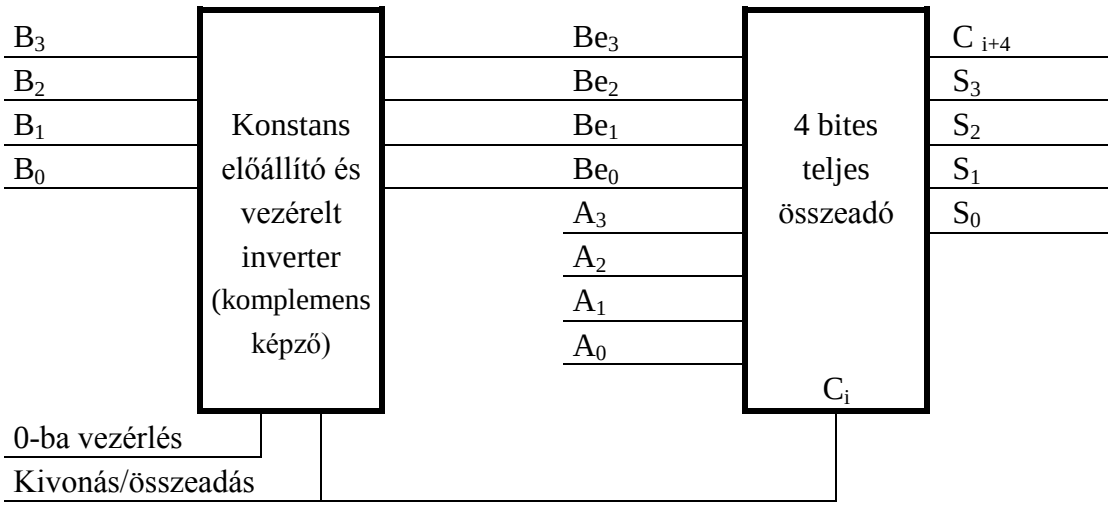
Az 1-es komplemens a binárisan felírt szám bitenként vett negációjával állítható elő. A 2-s komplemens előállítása a következő nagyságrendből való kivonással állna elő, de ezzel egyenértékű, ha a bitenként vett negációhoz (vagyis az egyes komplemenshez) hozzáadunk 1-t. A gyakorlatban ez a megoldás a használatos, mégpedig oly módon, hogy a hozzáadás a bemeneti Ci bit felhasználásával, az összeadással egy lépésben történik meg.

A vezérelt komplemensképző és alapérték állító elem kapcsolási rajza és vezérlési táblázata látható az alábbi ábrán.

Vezérlés		Kimenetek			
B	C	Y_1	Y_2	Y_3	Y_4
L	L	$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$
L	H	A_1	A_2	A_3	A_4
H	L	H	H	H	H
H	H	L	L	L	L



4 bites vezérelt komplemensképző és 0/1 előállító áramkör igazságtáblája és logikai rajza



4 bites vezérelt összegző, kivonó áramkör vázlata

Az univerzális műveletvégző elem, az ALU

Az aritmetikák között fontos szerepet tölt be az univerzális műveletvégzésre készült áramkör az Aritmetikai Logikai Elem, az úgynevezett ALU (Arithmetic Logic Unit/Function Generator)

Ez az összetett elem a kiválasztó (M, S3, S2, S1, S0 Selection és a Ci) bemenetek vezérlésének függvényében képes aritmetikai – vagyis helyi érték eltolódást is okozó műveletek – és bitpáronként logikai műveletek végrehajtására. Regiszterrel és adatirányítóval kiegészítve a számítógépek központi végrehajtóművének (CPU) a fő egysége.

A gyakorlati felhasználás során a kimeneti szinttartományok és a kettes számrendszer 0, 1 értékeinek az összerendelése választás kérdése. Így beszélhetünk magas szint (H) aktív logikáról (ez a + tápfeszültségű rendszerben a pozitív logika), amikor is az L szint képviseli a 0 számértéket, és a H szint az 1-t, illetve alacsonyszint (L) aktív logikáról (ez a + tápfeszültségű rendszerben a negatív logika), amikor a H szint képviseli a 0-t és az L szint az 1-t. Az ALU működését – a felhasználást megkönnyítendő – mindig mindkét logika szerint megadják.

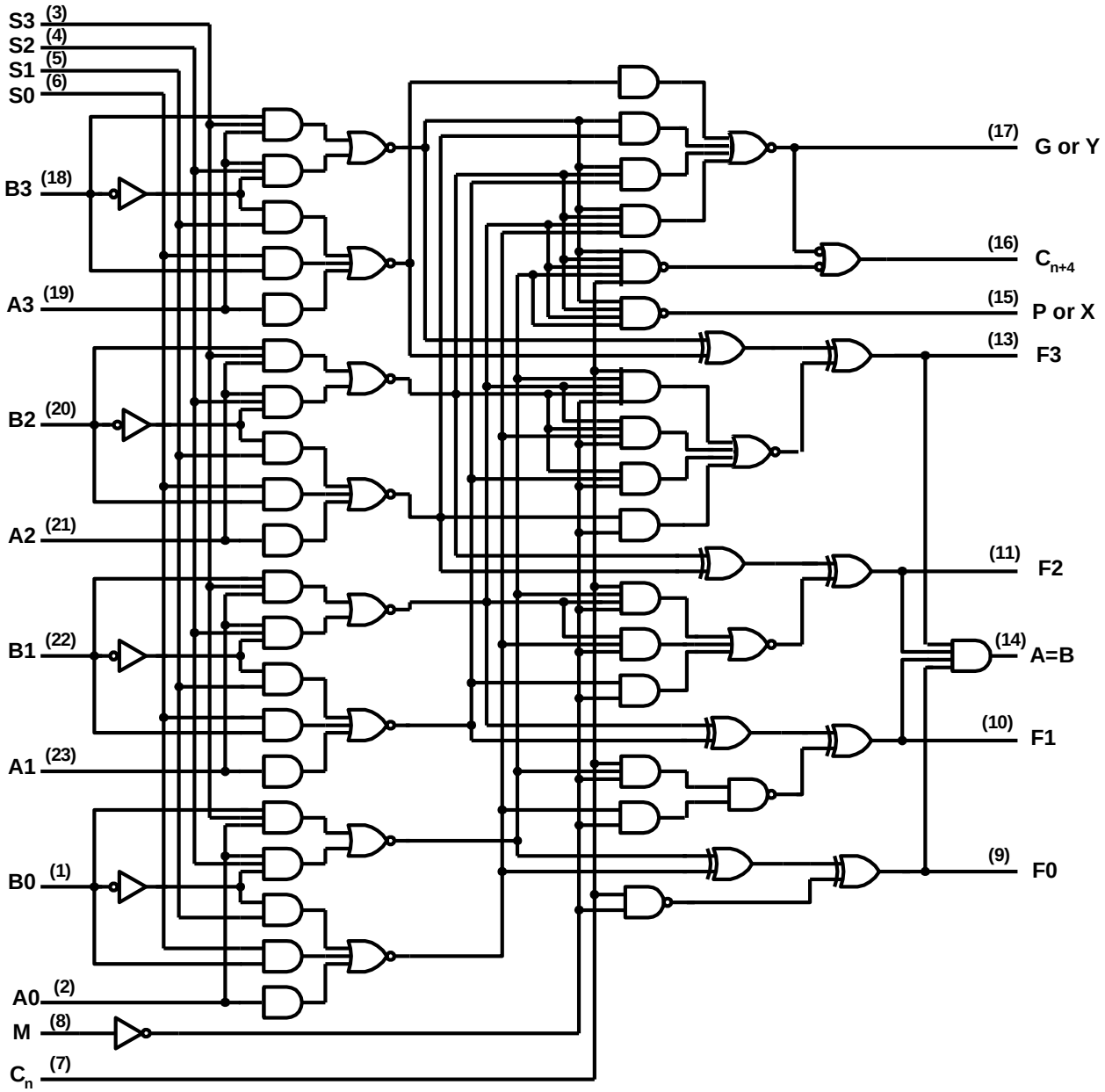
Az ALU tartalmaz egy nyitott kollektoros megoldású kimenetet a minden kimenőbit (F3 – F0) 1-s értékű detektálására. Használatánál így a teljes szóra, huzalozással létrehozhatunk egy ÉS kapcsolatot, de természetesen a helyes működéshez feltétlen kell a kimenetre kötött munkaellenállás. Ez az A=B kimenet. Kézenfekvő használata az EKVIVALENCIA művelet eredményének vizsgálatánál van. A két bemenő számérték egyezik, ha az A = B kimenet, H értékű. Természetesen ez a kimenet akkor is H, ha az összes kimenet bármely okból H értéket vesz fel.

Az átvitel bit alapvetően negatív, vagyis a pozitív logikánál akkor van átvitel, ha a Ci bit L szintű. Negatív logikánál természetesen a H szint jelenti az átvitelt. Fontos megjegyezni, hogy a kivonásnál az átvitelbit tartalma fordított az összeadásnál használtnak. A belső megoldás párhuzamos átvitelbit képzést tartalmaz. A következő 4 bites egység(ek) esetén azonban az átvitel soros. Több összekapcsolt ALU esetén, ami a szokásos 16, 32 bites rendszereknél 4, 8 db. 4 bites ALU alkalmazását jelenti, ez már erősen lassíthatja a működést. Ezért az ALU, az alapvető működéshez szükséges kimeneteken kívül megvalósít két segédfüggvényt is. A segédfüggvények bemenetként szolgálnak egy átvitelgyorsításra kifejlesztett áramkörhöz. Az átvitel gyorsító az egyes ALU-któl

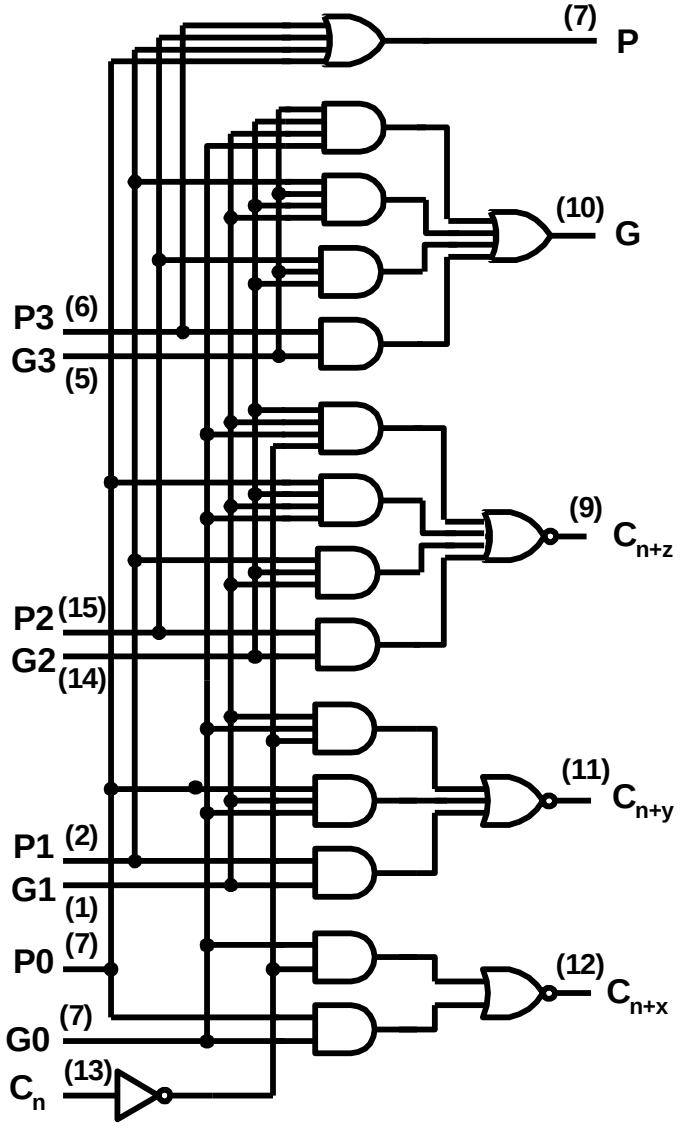
kapott segédfüggvények segítségével előállítja a következő három fokozat bemeneti átvitel bitjét. Ily módon egy 16 bites számértéknél az átvitel már gyorsul. 32 és 64 bites számoknál kétfokozatú átvitelgyorsítót kell használnunk, amire az átvitelgyorsító saját kimeneti segédfüggvényeivel lehetőséget biztosít. Ily módon egy átvitel legfeljebb 3 áramköri fokozat késleltetését jelenti, a 32 bitesnél egyébként fennálló 7 fokozatú késleltetéssel szemben. Az alábbi 3 ábrán az ALU, az átvitel gyorsító, és a két egység összekapcsolása látható.

A bemeneti, kimeneti túlcsordulás bitek összehasonlításával, az A és a B bemeneti érték relációjához jutunk:

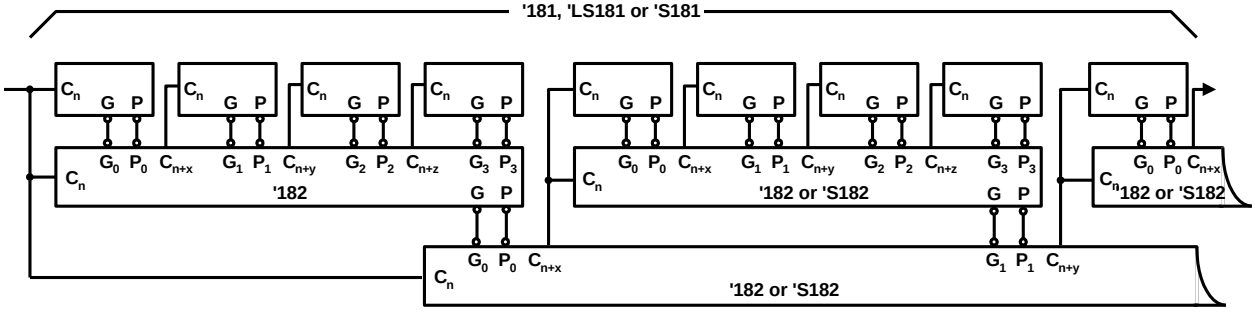
Bemeneti C_i	Kimeneti C_{i+4}	Aktív magas szintű logika	Aktív alacsony szintű logika
H	H	$A \leq B$	$A \geq B$
H	L	$A > B$	$A < B$
L	H	$A < B$	$A > B$
L	L	$A \geq B$	$A \leq B$



Aritmetikai Logikai Elem (ALU) logikai rajza



Átvitel gyorsító áramkör logikai rajza.



ALU és átvitelgyorsító együttes használata.

Az ALU működési táblázatai

Magasszintű logikához tartozó vezérlési táblázat:

KIVÁLASZTÓ BEMENETEK				AKTIV MAGAS SZINTŰ LOGIKA		
				M = H LOGIKAI FUNKCIÓK	M = L ARIMETIKAI MŰVELETEK	
S3	S2	S1	S0		Cn = H (Nincs átvitel)	Cn = L (Van átvitel)
L	L	L	L	F = $\overline{A}$	F = A	F = A plus 1
L	L	L	H	F = $\overline{A + B}$	F = A + B	F = (A+B) plus 1
L	L	H	L	F = $\overline{A}$ B	F = A + $\overline{B}$	F = (A+ $\overline{B}$) plus 1
L	L	H	H	F = 0	F = minus 1 (2-s kompl.)	F = zero
L	H	L	L	F = $\overline{A\ B}$	F = A plus A $\overline{B}$	F = A plus A $\overline{B}$ plus 1
L	H	L	H	F = $\overline{B}$	F = (A+B) plus A $\overline{B}$	F = (A+B) Plus A $\overline{B}$ Plus 1
L	H	H	L	F = A $\oplus$ B	F = A minus B minus 1	F = A minus B
L	H	H	H	F = A $\overline{B}$	F = A $\overline{B}$ minus 1	F = A $\overline{B}$
H	L	L	L	F = $\overline{A + B}$	F = A Plus AB	F = A plus AB plus 1
H	L	L	H	F = A $\oplus$ B	F = A plus B	F = A plus B plus 1
H	L	H	L	F = B	F = (A+ $\overline{B}$) plus AB	F = (A+ $\overline{B}$) plus AB plus 1
H	L	H	H	F = A B	F = AB minus 1	F = AB
H	H	L	L	F = 1	F = A plus A*	F = A plus A plus 1
H	H	L	H	F = A + $\overline{B}$	F = (A+B) plus A	F = (A+B) plus A plus 1
H	H	H	L	F = A + B	F = (A+ $\overline{B}$) plus A	F = (A+ $\overline{B}$) plus A plus 1
H	H	H	H	F = A	F = A minus 1	F = A

* = eggyel léptetés (shift) a nagyobb helyiértékű bitek felé

Túlcsordulás bitek értelmezése, a relációk megállapításához:

- Ha az A=B kimenet H, ez a megfelelő műveletkor (Ekvivalencia) azonosságot jelent.
- Az A=B kimenet Pozitív logika esetén jelzi az előfordulható legnagyobb számérték az Fh meglétét, illetve a negatív logikánál a 0 értéket.

Az ALU-nál a logikai műveletcsoport kiválasztásakor (M mód választó vezérlő bit H) rendelkezésünkre áll mind a 16 lehetséges művelet kombináció.

Alacsonyszintű logikához tartozó vezérlési táblázat:

KIVÁLASZTÓ BEMENETEK				AKTIV ALACSONY SZINTŰ LOGIKA		
				M = H LOGIKAI FUNKCIÓK	M = L ARITMETIKAI MŰVELETEK	
S3	S2	S1	S0		Cn = L (Nincs átvitel)	Cn = H (Van átvitel)
L	L	L	L	F = $\overline{A}$	F = A minus 1	F = A
L	L	L	H	F = $\overline{A B}$	F = A $\overline{B}$ Minus 1	F = A $\overline{B}$
L	L	H	L	F = $\overline{A}$ + B	F = A $\overline{B}$ minus 1	F = A $\overline{B}$
L	L	H	H	F = 1	F = minus 1 (2-s kompl)	F = zero
L	H	L	L	F = $\overline{A + B}$	F = A plus (A+ $\overline{B}$)	F = A plus (A+ $\overline{B}$) plus 1
L	H	L	H	F = $\overline{B}$	F = A plus (A+ $\overline{B}$)	F = AB plus A + $\overline{B}$) plus 1
L	H	H	L	F = $\overline{A \oplus B}$	F = A minus B minus 1	F = A minus B
L	H	H	H	F = A + $\overline{B}$	F = A + $\overline{B}$	F = (A+ $\overline{B}$) plus 1
H	L	L	L	F = A B	F = A plus (A+B)	F = A plus (A+B) plus 1
H	L	L	H	F = A $\oplus$ B	F = A plus B	F = A plus B plus 1
H	L	H	L	F = B	F = A $\overline{B}$ plus (A+B)	F = A $\overline{B}$ plus (A+B) plus1
H	L	H	H	F = A + B	F = A + B	F = (A+B) plus 1
H	H	L	L	F = 0	F = A plus A *	F = A plus A plus 1
H	H	L	H	F = A $\overline{B}$	F = A B plus A	F = A B plus A plus 1
H	H	H	L	F = A B	F = A $\overline{B}$ plus A	F = A $\overline{B}$ Plus A plus 1
H	H	H	H	F = A	F = A	F = A plus 1

* = egyel léptetés (shift) a nagyobb helyiértékű bitek felé

Az Aritmetikai műveletcsoportban a következő műveletek állnak rendelkezésre:

- Összeadás, kivonás. A művelet értelmezése az átvitel bit függvénye is. Csak az A mínusz B értelmezett, a B-ből nem tudok kivonni.
- A plus 1, inkrementálás
- A mínusz 1, dekrementálás
- Komplementálás
- A-nak önmagával való összeadása, ami a nagyobb helyi érték felé való léptetést jelenti (balra léptetés). Ez a művelet a B bemenetről nem értelmezett, valamint nem tud az ALU a kisebb helyi érték felé (jobbra) léptetni, amennyiben ez a művelet is szükséges, akkor külön áramkörrel oldják meg.

Az aritmetikai műveletcsoportban jó néhány logikai művelet is található, amit egyszerűbb a logikai műveletcsoportból kiválasztani, mert ott nem kell figyelni a Carry beállítására. Ezenkívül összetett aritmetikai, logikai műveletek teszik teljessé a választékot.

Az ALU használata

Az ALU alapvetően a számítógépek központi műveletvégzőjeként kerül használatra.

A számítógép központi műveletvégzőjéhez szükségünk van:

- ALU-ra
- A műveletek eredményének a következő lépésig tartó tárolására. Ezt a kimenetre épített regiszterrel oldjuk meg. Ennek a regiszternek a neve: Akkumulátor (gyűjtő).
- A jelző kimeneti bitek tároló flip-flopjaira.
- Egyes esetekben a bemenetre adott adatokat tároló regiszterekre.
- Az adat utak vezérelt összekötését biztosító adatirányítókra, vagyis multiplexerekre.
- Vezérelhető órajel-generátorra.
- A teljes műveletvégző működését ütemező vezérlő egységre, mely önmagában is egy összetett, bonyolult áramkör, most csak fekete dobozként, funkcionálisan fogjuk tárgyalni.
- Szükség van a bemeneti adatok forrására és a kimeneti adatok tárolására, amit tipikusan a memória lát el. Most csak jelzésszerűen fogjuk említeni.

Digitális szorzás

A szorzás megvalósítható az ALU használatával, de sokbites számok esetén igen sok lépésre van szükség. Számtalan alkalmazásban szükség van nagy értékű számokkal végzett tömeges szorzások elvégzésére.

Két egyjegyű bináris szám szorzata:

Szorzandó	Szorzó	Eredmény
0	0	0
0	1	0
1	0	0
1	1	1

Láthatjuk, hogy a szorzás igazságtáblája megfelel a kétbemenetű ÉS kapu igazságtáblájának, tehát elemi szorzónak használható a logikai elem.

Nézzük, hogy hogyan lehet egy többjegyű szám szorzását egyszerűen elvégezni.

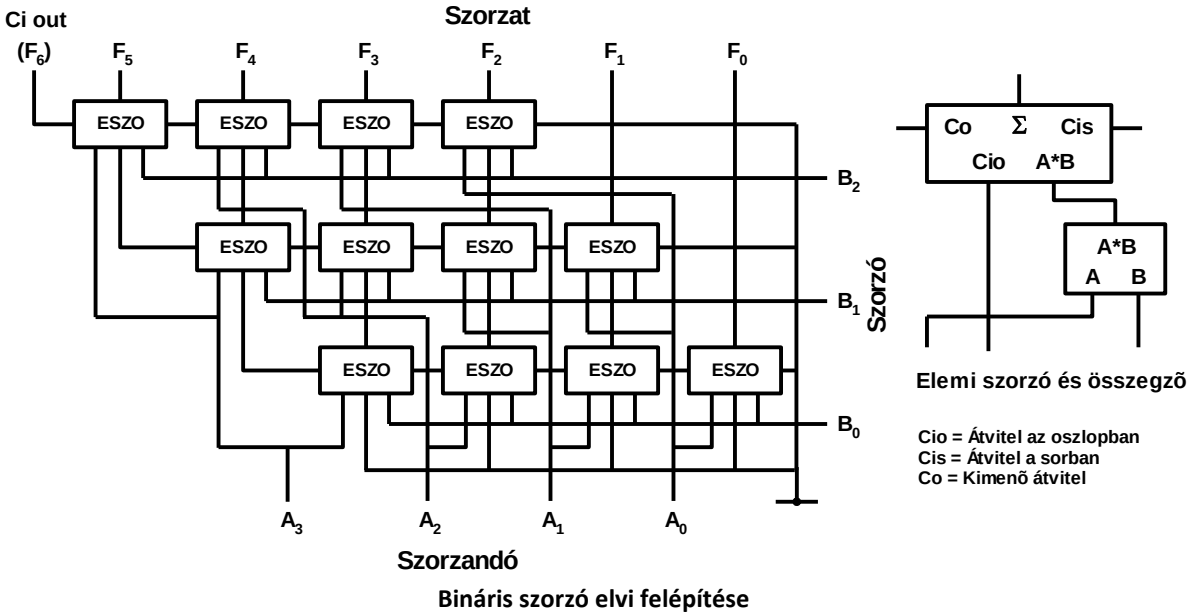
Például szorozzuk össze 7h és az Eh számokat

Bináris formában felírva:

```
0 1 1 1 • 1 1 1 0
0 1 1 1
  0 1 1 1
    0 1 1 1
      0 0 0 0
-----------------
1 1 0 0 0 1 0
```

```
  7 • E
-------
9 8
```

Az egyszerű “szorzómű” mellett jelentős az összeadók száma. A felépítés vázlatát 4•4 bites esetre az Arit.14. ábrán mutatjuk be. A gyakorlatban a szorzók nagy bitszámmal dolgoznak. 64-80-128 bit. Ez rendkívül nagyméretű és az összegzés és az átvitel miatt egyre lassúbb áramköröket jelent. A sebességigény miatt sokféle, gyorsabb átvitelű szorzót dolgoztak ki.



Bináris szorzó elvi felépítése

Hiba ellenőrző és hibajavító áramkörök

Minden tárolt, vagy átvitt információ, a környezeti hatások és az áramkörműködés bizonytalanságai miatt, csekély, de figyelembeveendő valószínűséggel sérülhet. Az alkalmazásokhoz tulajdonképpen mindig hibátlan adatokra lenne szükségünk. Minimális igényként legalább azt szeretnénk tudni, hogy a használt adat jó-e.

A hibafelismeréshez és a javításhoz plusz bitek használatára, redundancia beépítésére van szükség. A legegyszerűbb kiegészítés egy párosság (paritás) bit képzése és a felhasználáskor ellenőrzése. A használt kiegészítő bitek számától függően csak jelezni tudjuk a hibát, vagy a redundancia növelésével (a plusz bitek számának növelésével) egy, más megoldásoknál több bit hibáját javítani tudjuk.

A hibajavító kódok a javítható bitszámnál nagyobb számú hibát még jelezni képesek.

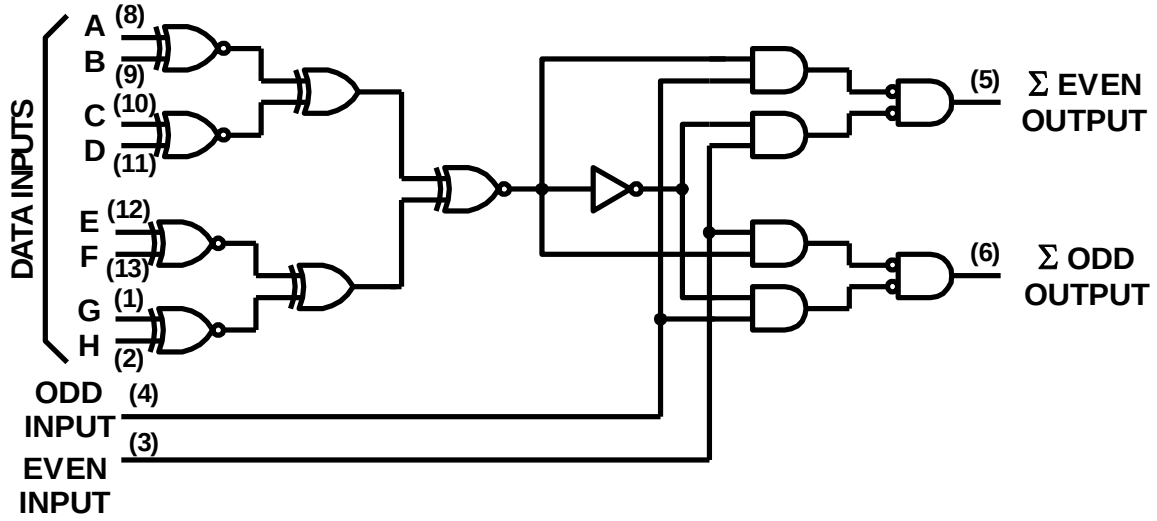
Párosság képzés és vizsgálat. Paritás-képző és ellenőrző áramkörök

A paritásképző áramkör általában egy bájt bitjeinek a párosságát vizsgálja, és a kiegészítő, az úgynevezett paritásbitet teszi 9. bitként az adathoz. Az ellenőrzéskor újra képezzük a paritás bitet és összehasonlítjuk a kiolvasottal. Olvasás alatt itt most a tárolt, vagy valamilyen módón átvitt adat felhasználáshoz való „elővételét” értjük. Amennyiben az olvasáskor előállított párosság bit azonos a kiolvasottal, akkor az adat jó, ha nem, akkor hibás az adat.

Az egyszerű párosság vizsgálat csak páratlan számú hibát tud jelezni, alapvetően 1 bit hibáját, vagyis ellenkezőre fordulását. A páros számú hiba a vizsgálat elvéből adódóan nem állapítható meg.

A paritásvizsgáló áramkör felépítése és működési táblája látható az alábbi ábrán.

Bemenetek			Kimenet	
∑ H szint számossága	Vezérlés bemenetek		∑ páros	∑ páratlan
	Páros	Páratlan		
Páros	H	L	H	L
Páratlan	L	L	L	H
Páros	L	H	L	H
Páratlan	L	H	H	L
X	H	H	L	L
X	L	L	H	H



Párosság (Paritás) előállító és vizsgáló áramkör kapcsolási rajza és működésének igazságtáblája.

A soros átvitelnél a legegyszerűbb paritásképzés egy T flip-flop felhasználásával nyerhető. Az adatok rákapcsolása előtt a flip-flopot nullázni kell. A T bemenetre vezetett adatfolyam minden 1-es értéknél kiváltja a flip-flop ellenkező értékre billenését, vagyis az előző érték negálását. Ebből adódóan, ha az utolsó adatbit után a flip-flop 0-ban van, az adatfolyam páros, míg ha 1-ben, az adatfolyam páratlan számú egyesekből épült fel. Az utolsó adatbit után a párosság vizsgáló flip-flop tartalmát hozzáadjuk az adatfolyamhoz, mint utolsó elküldendő bitet. A soros adatátvitelnél többnyire a párosra kiegészítő paritás bitet használjuk. A vételi oldalon egy azonos áramkör bemenetére vezetjük az adatfolyamot és beléptetjük az adatbiteket és a paritás bitet. Hibátlan átvitel esetén a paritásbit belépése után a vizsgáló flip-flop kimenete 0-n áll, hiba esetén 1-n.

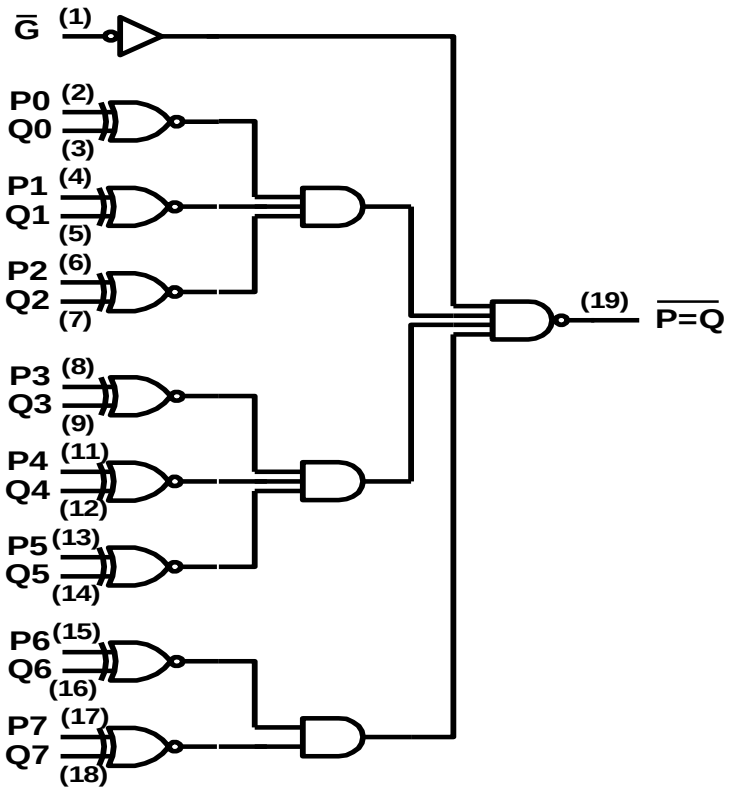
Aritmetikai jellegű áramkörök

Az aritmetikai jellegű áramkörök közé soroljuk a szón egyforma műveletcsoportot elvégző elemeket. Ilyenek pl. összehasonlító (<, =, >), prioritás (elsőbbség) vizsgáló, valamint a kódátalakító (kódkonverter) áramkörök. Bizonyos esetekben ide szokás sorolni az előzőleg tárgyalt paritásvizsgáló áramköröket is.

Összehasonlító (Magnitude Comparator) áramkörök

Azonosságvizsgáló

A csak azonosság vizsgáló, általában címek (kijelölő bitcsoportok) felismerésére szolgál, és mint ilyen áramkör nagyon sok helyen nyer alkalmazást. Tipikus alkalmazása a rendszer összeállításkor beállítható periféria címek felismerése. A bővíthetőség az egy áramkörrel felismerhetőnél (ez a bemutatott áramkörnél 8 bit) több bites felhasználás esetén szükséges. Jellegénél fogva mindegy, hogy a nagyobb, vagy a kisebb helyiérték van-e az első fokozatba kötve.



74HC688 8 bites azonosságvizsgáló (Identity Comparator) logikai rajza.

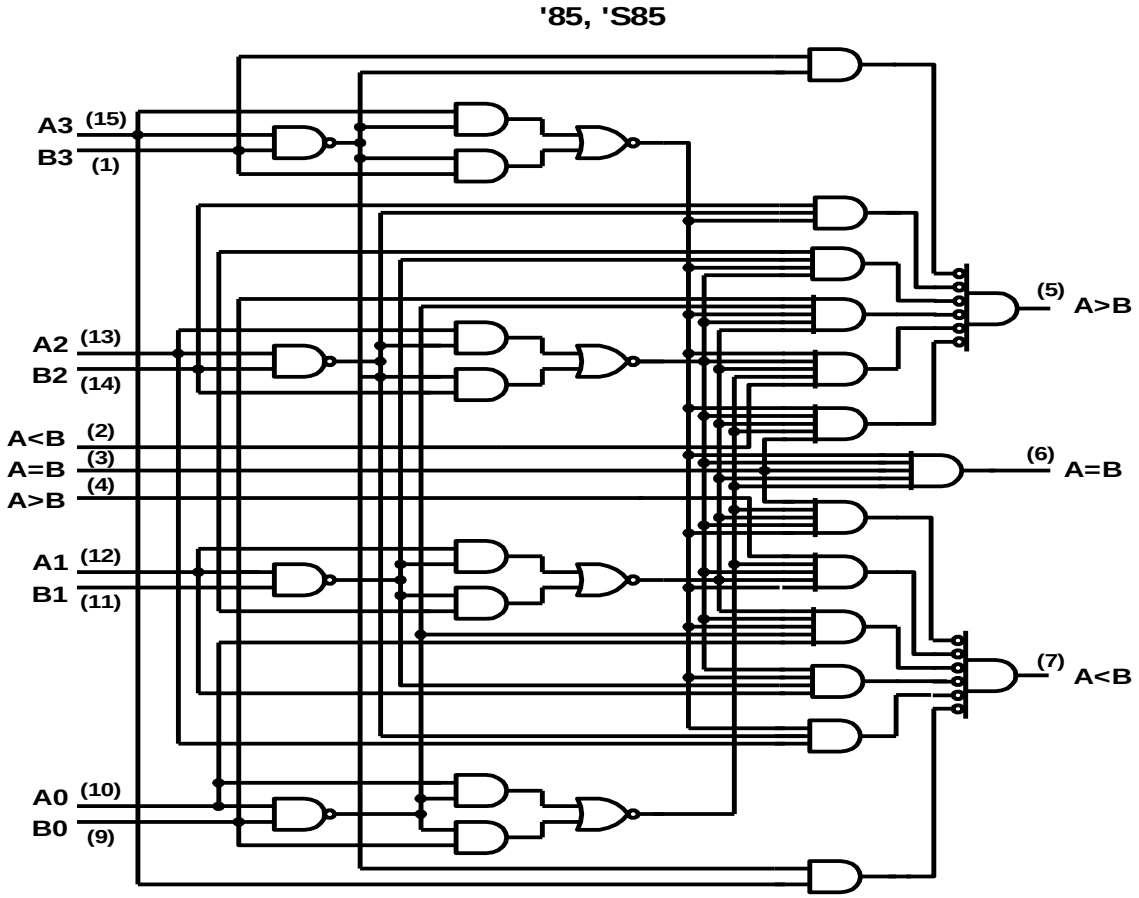
Teljes összehasonlító

Az összehasonlítás, A bemenet kisebb, mint B, illetve A bemenet nagyobb, mint B összehasonlítása alapján történik. Az ilyen áramkörök mindig tartalmaznak A egyenlő B azonosságvizsgálatot is. A vizsgálat az azonos helyiértékű bitek között történik. Az összehasonlítások jellegüknél fogva, mindig a legkisebb helyiértékű biteknél kezdődnek és így haladnak felfelé. A két bemenet arányát mindig a legnagyobb helyiértékű biten lévő adat határozza meg. Láncba kötéskor ezért a láncoló bemenetet csak akkor veszi figyelembe az egység, ha a saját összehasonlító bemenetein egyezést talált, hiszen az arányt most a kisebb helyiértékű biteknek kell meghatározniuk. Az elmondottakból

következik, hogy a bitek sorrendjének keverésekor az áramkör csak az azonosság vizsgálatot tudja elvégezni.

Az áramkör kapcsolása és működésének igazságtáblája látható az alábbi ábrán.

Összehasonlított bemenetek				Bővítő bemenetek			Kimenetek		
A3, B3	A2, B2	A1, B1	A0, B0	A>B	A<B	A=B	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	X	X	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L



74S85 4 bites Összehasonlító (Magnitudo Comparator) igazságtáblája, logikai rajza. (más technológiai sor esetén a belső kapcsolás kialakítása, a tok funkciójának megtartása mellett ettől eltérhet.)

Dekódoló

A dekódoló áramkör olyan több bemenetű kombinációs áramkör, amely a bináris, vagy BCD kódból állít elő un. **1 az N-ből** kódot. Olyan kombinációs áramkör, melynek n bemente és m kimenete van. A bemeneti kombinációk lehetséges száma 2^n, a kimenetek számára az alábbi összefüggés teljesül $m \leq 2^n$. A kimenetek közül mindig csak egy lehet aktív állapotú. Az aktív állapot lehet az 1, akkor csak a kiválasztott kimenet lesz 1-es értékű, az összes többi 0. De lehet a 0 is az aktív állapot, ekkor a kiválasztott kimeneten jelenik meg 0, a többi kimeneten 1 lesz. Az n-bites bináris bemeneti kóddal kiválasztásra kerül egy kimenet az m közül, amely csak ezen bemeneti érték esetén lesz aktív állapotú.

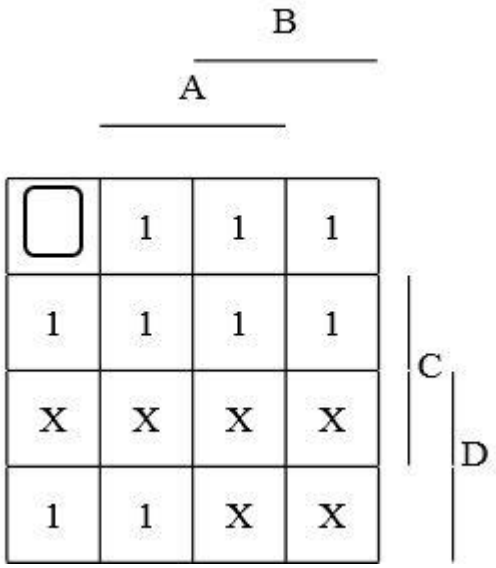
Nézzük meg példaként a BCD-decimális dekódolót.

A BCD kód legyen 4 bites, a decimális számjegyek száma pedig 10. Tehát itt egy 4-ről 10-re dekódolót fogunk megvalósítani. A feladatot úgy oldjuk meg, hogy az aktív kimenet 0 értékű legyen, a többi 1-es. Először írjuk fel az igazságtáblát:

D	**C**	**B**	**A**	**Y9**	**Y8**	**Y7**	**Y6**	**Y5**	**Y4**	**Y3**	**Y2**	**Y1**	**Y0**
0	0	0	0	1	1	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	1	1	0	1	1	1
0	1	0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	1	1	1	1	1	0	1	1	1	1	1
0	1	1	0	1	1	1	0	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1	1	1	1
1	0	0	0	1	0	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	1	1	1
1	0	1	0	X	X	X	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X

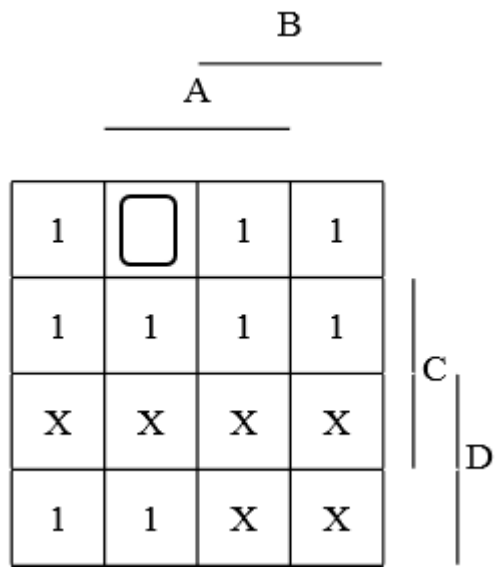
A kimenetek megvalósításához 10 db 4 változós Karnaugh tábla kell.

Y0



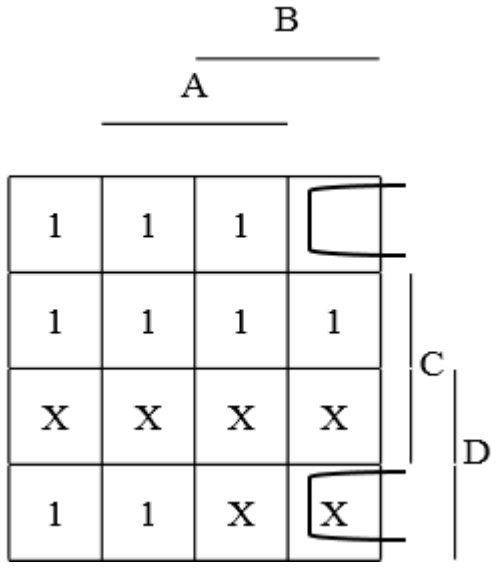
Y0 _____

Y1



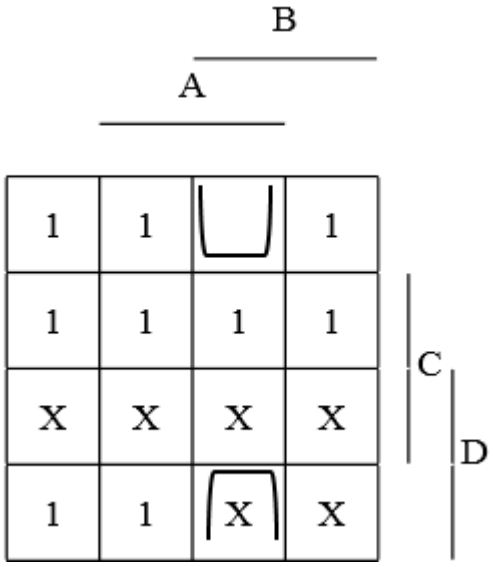
Y1 _____

Y2



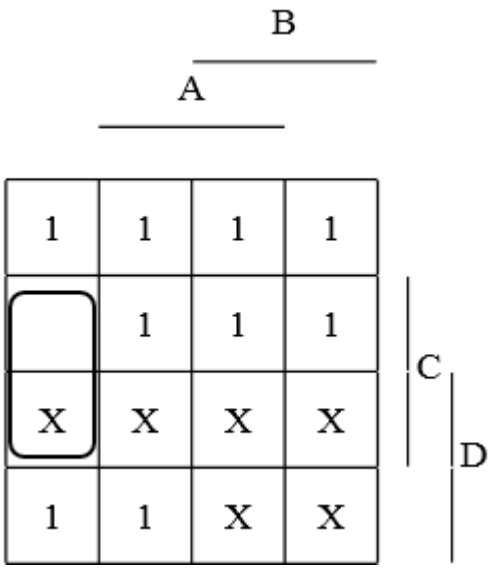
Y2 _____

Y3



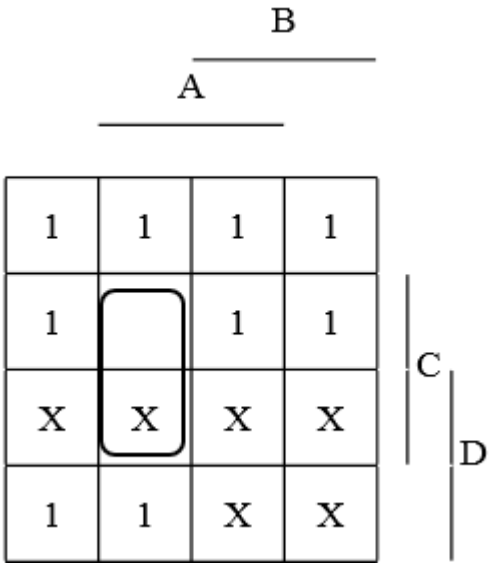
Y3 _____

Y4



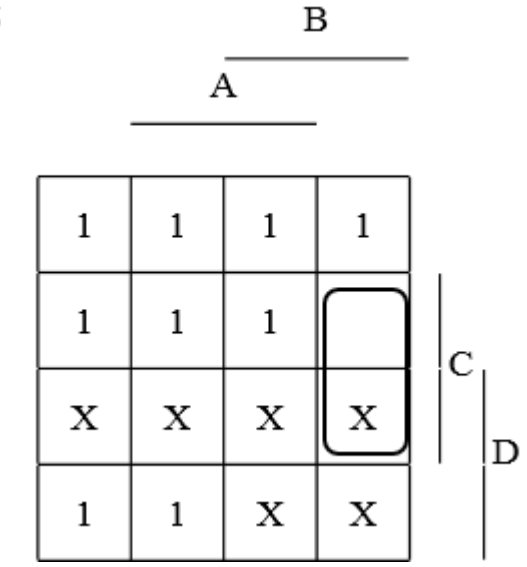
Y4 _____

Y5



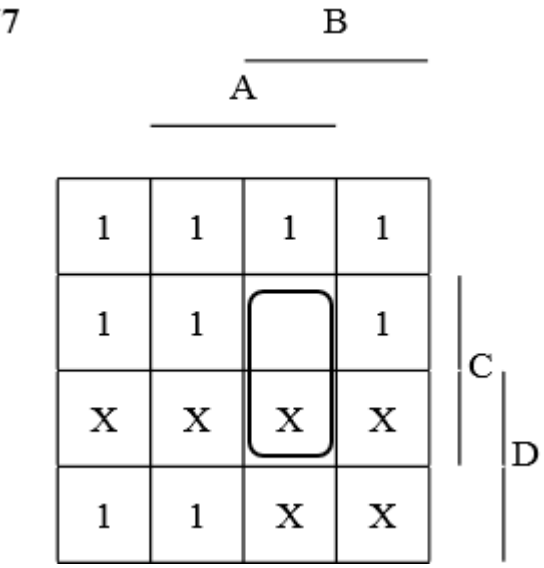
Y5 _____

Y6



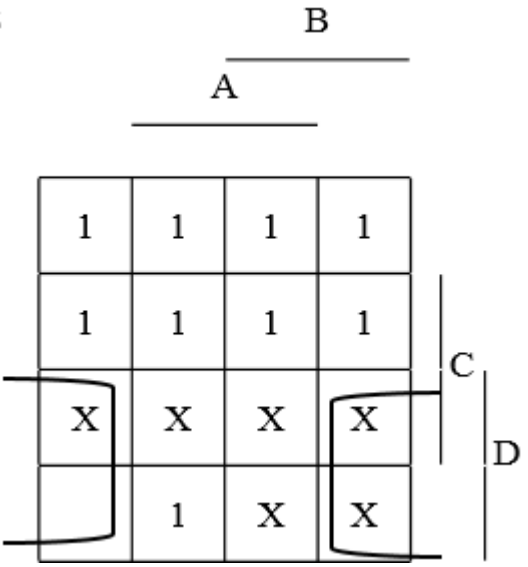
Y6 = _____

Y7



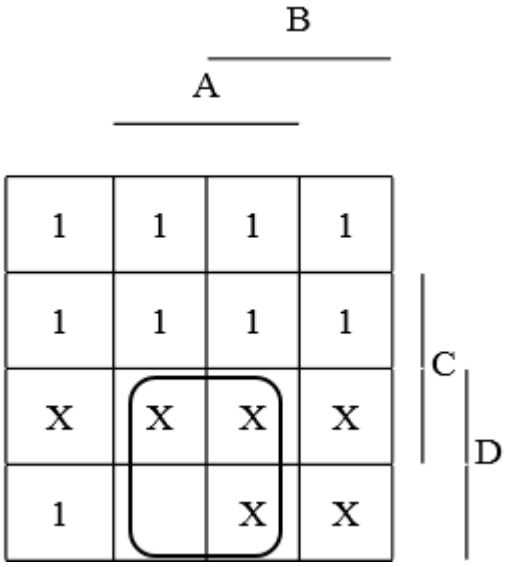
Y7

Y8



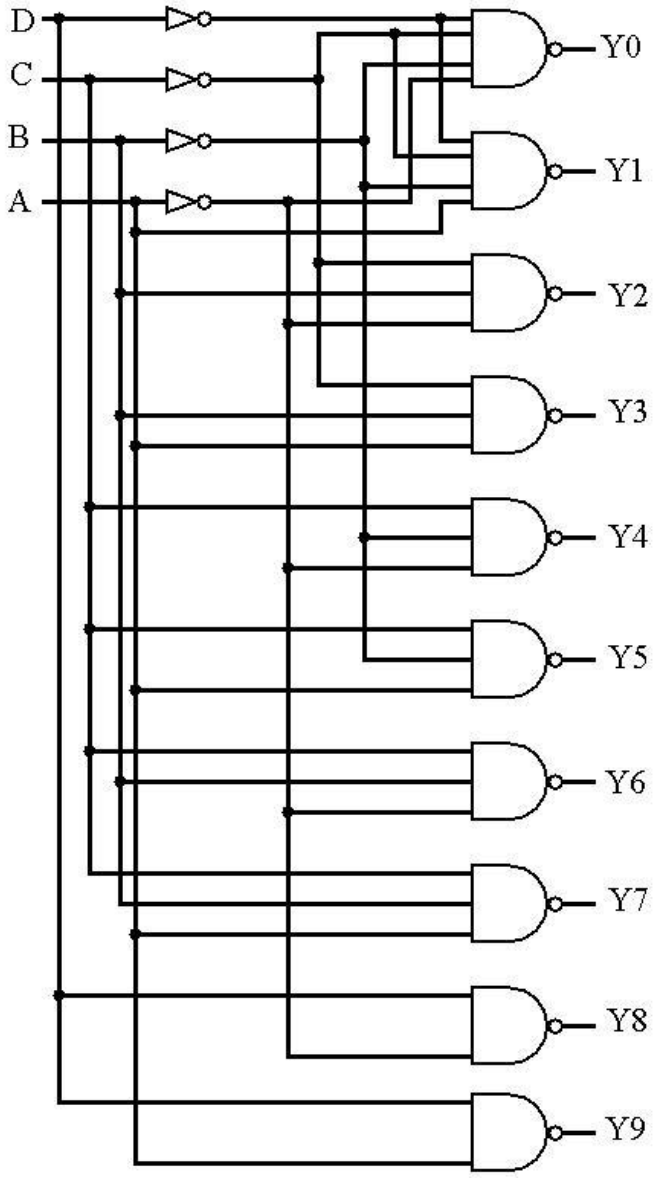
Y8 = ____

Y9



Y9

A BCD-decimális dekódoló:



Kódoló

A dekódoló ellentéte: 2^N bemenete és N kimenete van. A kimeneteken azon bemenet bináris sorszáma jelenik meg, amelyik bemenet aktív szintű. Mindig csak egy bemenet lehet aktív szintű, ami lehet 0 vagy 1 is.

Példaként nézzük meg az oktális-bináris kódolót. Az aktív bemeneti szint az 1-es. Igazságtáblája:

I0	I1	I2	I3	I4	I5	I6	I7	C(2^2)	B(2^1)	A(2^0)
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

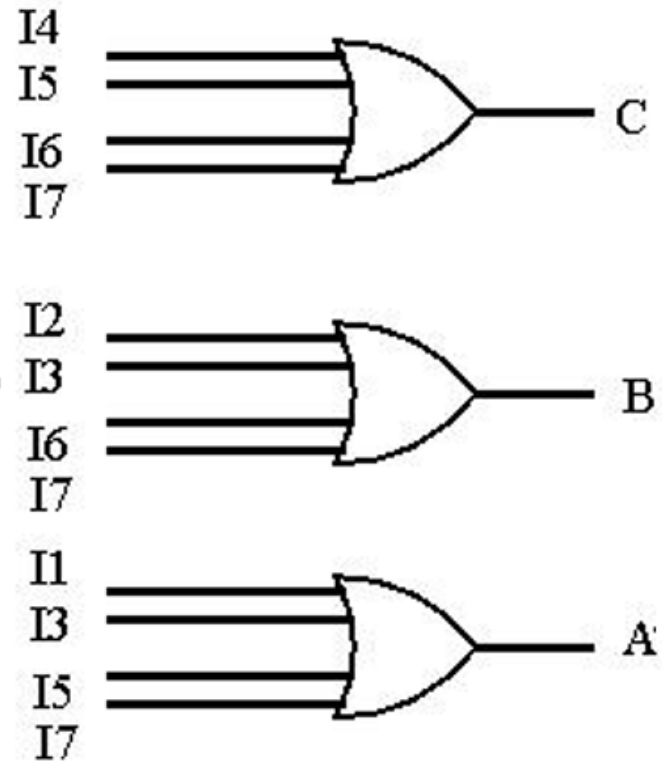
Az igazságtábla alapján nézzük a kimenetek egyenleteit:

C=I4+I5+I6+I7

B=I2+I3+I6+I7

A=I1+I3+I5+I7

Az egyenletek alapján rajzoljuk fel a hálózatot:



Kódátalakítók

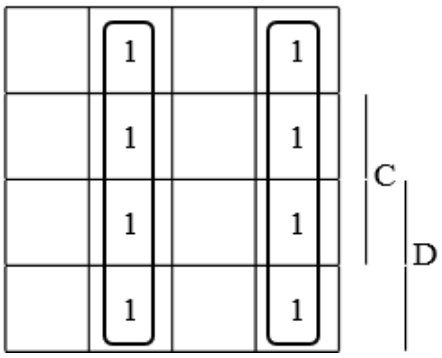
A kódátalakító áramkörök két különböző kódrendszer közötti konverziókat végzik el. Alapvetően BCD kódok egymás közötti, illetve a hexadecimális kód közötti átváltására dolgozták ki ezeket az áramköröket. Ma többnyire a kódkonverziót mikroprocesszoros alkalmazásokban programozott úton hajtják végre. Amennyiben mégis szükség van hardverműködésű kódátalakítóra, azt PROM-ban, vagy programozható logikában oldják meg.

Nézzük meg példaként a Bináris-Gray kódátalakítót. Első lépésben írjuk fel az igazságtáblát. 4 bites kódokkal dolgozunk.

D	C	B	A	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

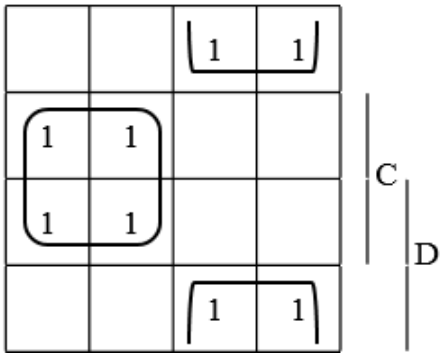
Az igazságtábla alapján adjuk meg a G3-G0 kimenetek megvalósítását:

G0



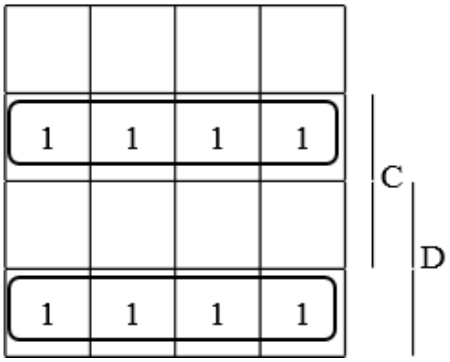
$$G0 = \overline{B} * A + B * \overline{A}$$

G1



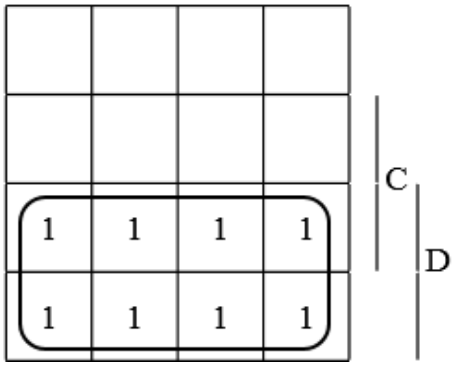
$$G1 = \overline{C} * B + C$$

G2



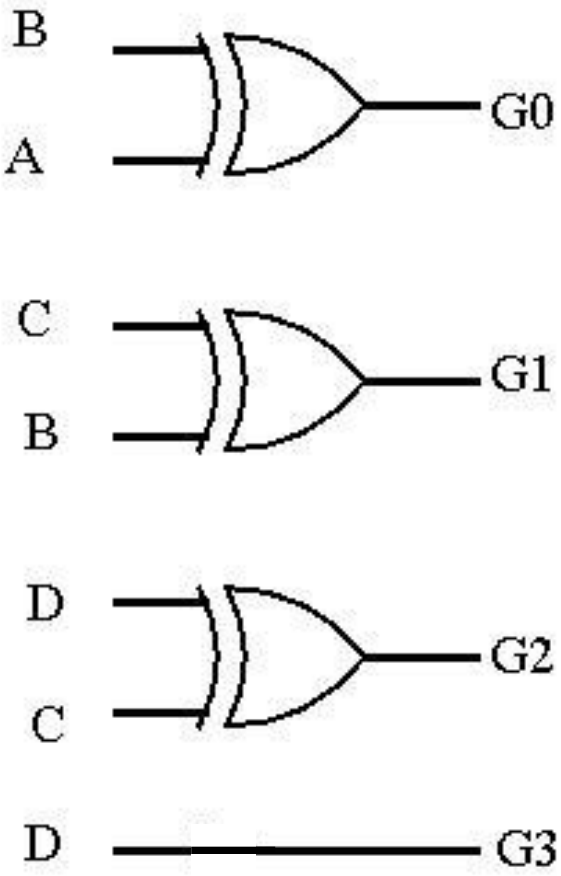
$$G2 = \overline{D} * C + D * \overline{C}$$

G3



$$G3$$

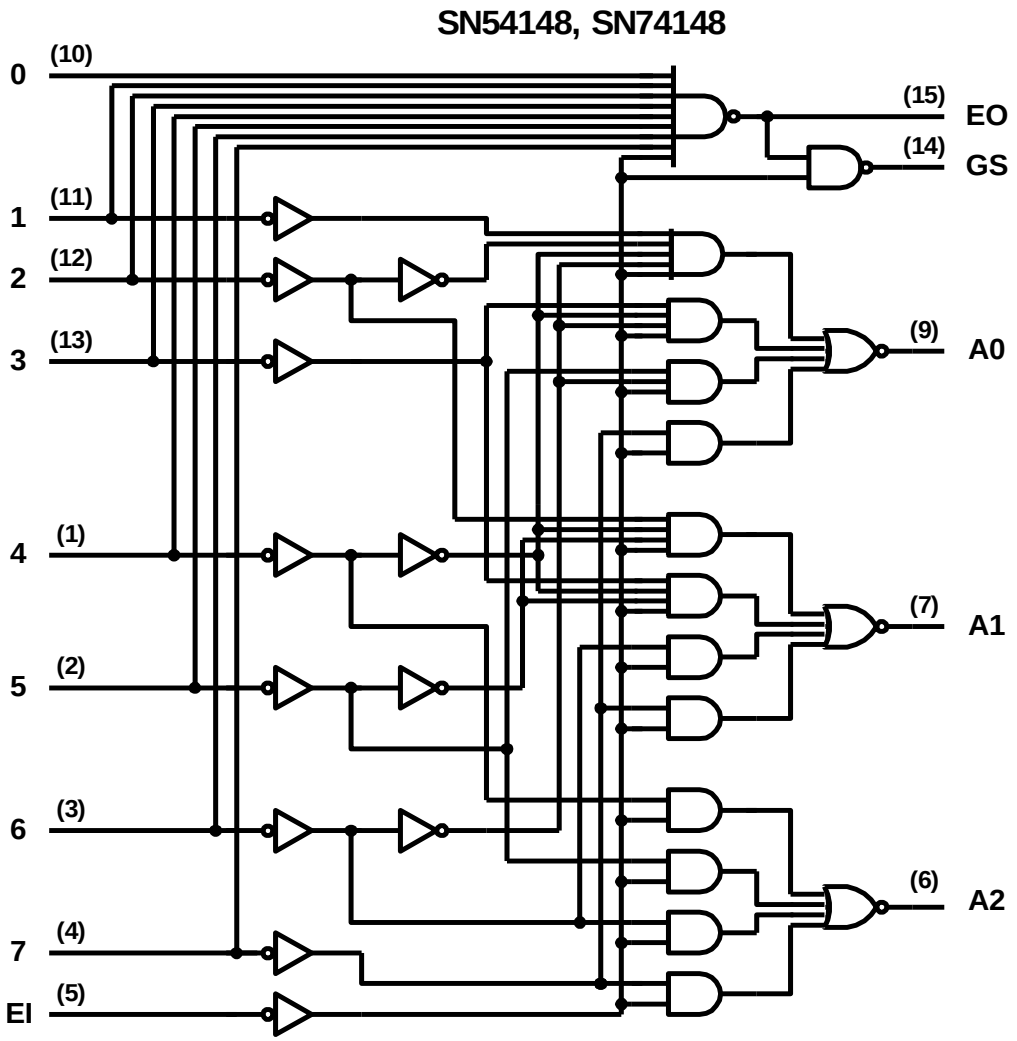
A Bináris-Gray átalakító:



Prioritásvizsgáló

Egyes alkalmazásokban szükséges a bemenő jelek erősorrendjének, illetve fontossági, elsőbbségi sorrendjének a megállapítása. Pl. a régibb típusú klaviatúráknál több, egyidejűleg lenyomott billentyű esetén a kód kialakításához. Vagy egy processzor a hozzá beérkező kérelmek közül egyszerre csak egyet tud kiszolgálni. Ekkor egy előre meghatározott kiszolgálási sorrend kialakításához meg kell határozni, hogy a több egyidejűleg előálló kérelem közül melyik a rendszerben a legerősebb, és az annak megfelelő kódot kell közölni a vizsgálatot végző eszközzel, példánkban a processzorral. Az erősorrend vizsgálat alapötlete: az elsőbbségi helyen lévő kérelem letiltja az összes nála kisebb sürgősségű jelek bemenő kapuját, és kialakítja a kimeneten a sorrendiségének megfelelő kódot. Kialakításuk lehetővé teszi több egység összekapcsolását is. A nagyobb prioritású EO kimenetét kell a kisebb prioritású EI bemenetére kötni. A GS kimenet felhasználható a következő kimenő Ax érték előállításához. Léteznek a kérelmet eltároló típusok is.

Bemenetek									Kimenetek				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	E0
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H



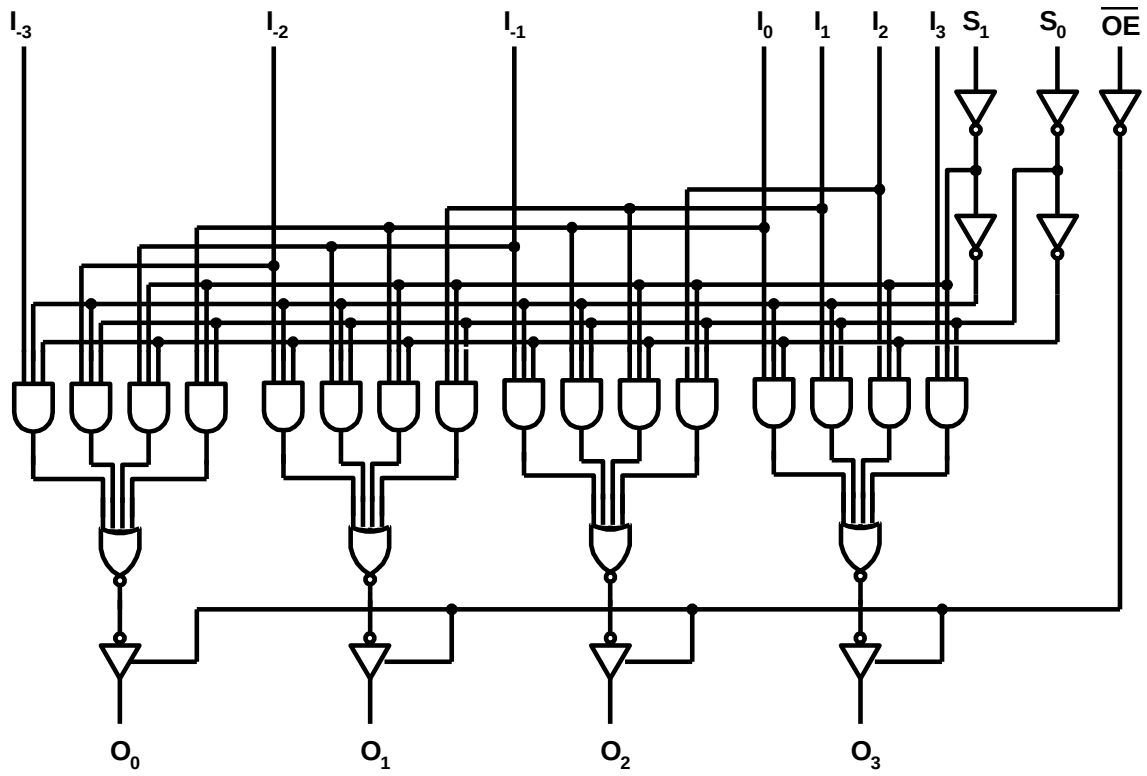
Prioritás encoder igazságtáblája és kapcsolási rajza.

Az aritmetikai működést kibővítő elemek

Léptetők (Shifterek)

Egyes aritmetikáknál sűrűn előforduló feladat lehet a több helyi értékkel való léptetés. A szokásos megoldás esetén minden helyi értékkel való léptetés egy óraütemet igényel. Szerkeszthető azonban olyan áramkör, mely egy lépésben megvalósítja a kívánt lépésszámot. Ez az áramkör a vezérelt léptető (shifter). Lényegileg egy erre a célra ügyesen bekötött multiplexerről van szó, amely a bejáratán lévő adatot a beállított vezérlés szerinti kimenő bitre kapuzza. A tri-state kimeneti megoldás a funkció bővíthetőségét biztosítja.

BEMENETEK VEZÉRLÉSE			KIMENETEK			
$\overline{OE}$	S_1	S_0	Q_0	Q_1	Q_2	Q_3
H	X	X	Z	Z	Z	Z
L	L	L	I_0	I_1	I_2	I_3
L	L	H	I_{-1}	I_0	I_1	I_2
L	H	L	I_{-2}	I_{-1}	I_0	I_1
L	H	H	I_{-3}	I_{-2}	I_{-1}	I_0



Vezérelt léptető, úgynevezett Shifter

Felhasznált irodalom

[1] Grosz Imre: Élő Digitronika Arit_1_2_5 (Aritmetikai áramkörök felépítése)