Tartalomjegyzék

TL;DR verziók

Multiplexerek

Útválasztó, vagyolja azokat a bemeneteket, amikre engedélyező jelet (EN) kap. EN lehet kódolt, amit dekóder szed szét, ez beépíthető az áramköri elembe. Lehet hierarchikus (fa) is: LSb módon páros/páratlan módon bontás, így rétegenként csoportos kiesés, MSb módon felezés, negyedelés…

Dekóderek

Előállítja az összes lehetséges ÉS kapcsolatot a bementek és negáltjaik közötti párokból. Mivel az egyik bemenet mindig aktív, érdemes engedélyező jelet használni. Mivel N bemenetből 2^N kimenet lesz, érdemes hierarchikusan (dekóder kimeneteire kötött dekóderekkel) vagy mátrixosan (sor/oszlopdekóderekkel) használni nagy méretekben.

Komparátorok

Egyenlőség: a két bemenő bitvektor minden bitjére egy XNOR kapu, majd ezek éselése. Teljes összehasonlító komparátor: egyszerre egyenlőség, kisebb/nagyobb. Bitenkénti blokkokban halad mindhárom vizsgálat eredménye, kezdetben mind igaz, és ha buknak, 0-k maradnak. A digit jegyzetünket noob írta, mert amúgy kisebb = !nagyobb * !egyenlő, így elég lenne két vizsgálat.

Számlálók

Aktív bemenetre növel egyet, körbefut. Sok van, legegyszerűbb a bitenkénti TFF-es bináris számláló: minden bit egy sorosan kapcsolt T flip-flop, ami 0 bemenetre tartja az értékét, 1-re pedig vált. TC-nek hívjuk egy számláló végértékjelzését, ami aktív, ha minden bit 1. Mire jó: frekvenciaosztás, időzítés, pulzusszámlálás, általános vezérlési feladatok.

Bináris aritmetikai műveletek

Összeadás: 0+0=0, 0+1=1, 1+1=1+átvitel, 3 bitre ugyanez 0+0+0=0, 0+0+1=1, 0+1+1=0+átvitel, 1+1+1=1+átvitel. Kivonó A+(-B) elven létezik. Negálás = minden bit inverze, majd +1.

Teljes összeadókból felépülő összeadó/kivonó

(Remélem, jó elemről írok.) A második operandus a kivonás jelzés függvényében invertálódik, egy összeadóba. Az invertálás bitenként egy XNOR.

Túlcsordulás kezelése

Adj hozzá egy extra bitet, ahová az utolsó carry-t vezeted, ez általában a CPU carry-re van kötve.

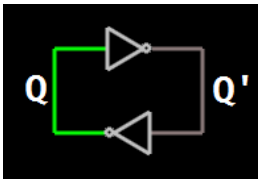
BCD számok összeadása

Minden bitnégyeshez +6, majd az így kapott bitvektoron összeadás. Amelyik négyes nem fut carry-be, abból ki kell vonni 6-ot.

Szorzás megvalósítása összeadó áramkörökkel

Vagy helyiértékenkénti összeadás, vagy shiftelve pörgetett számmal iteratív összeadás.

Memória áramkörök: írható és olvasható memóriák
Fogalmi összefoglaló.

Statikus RAM felépítése


Kvázi flip-flop, két inverter keresztben két tranzisztor közt, de ez 6 tranzisztorral is kirakható, ez a leggyakoribb. Addig marad meg a tartalma, amíg kap áramot, de nem kell periodikusan frissíteni. Kiolvasáshoz a BL sínre és inverzére áram megy, ha BL-en marad, akkor 1 a bit, ha az inverzén, akkor 0. Kezdeti állapota random.

Dinamikus RAM felépítése
Egy kondi tárolja az adatot, ami kiolvasással és idővel is elszivárog, ezért olvasás után és periodikusan is újra kell írni. Olvasáshoz BL 0 és 1 közti állapotra feszül, WL 1-re, majd erre az egyetlen tranzisztor rárántja az adatot BL-re, törölve azt. Tápfesz ráadásakor *lényegében* random.

ROM jellegű memóriák
- ROM: Read-Only Memory (csak olvasható, maszkprogramozott)
- PROM (Programmable ROM): egyszer programozható
- EPROM (Erasable-Programmable ROM): UV-fénnyel törölhető
- EEPROM (Electrically EPROM): elektronikusan törölhető

Sínrendszerek definíciója, kialakulása, osztályozása
Magyarul: busz. Egy IC-vel beszélő vezetékek csoportja.

Vezetékek csoportosítása, az egyes vezetékcsoportok feladata
Cím/adatbusz, vezérlőjelek (ALE, READY, IO/M, utóbbi jelentése, hogy I/O- és memóriakezelés is lehetséges a WR és RD vezetékeken, de nem egyszerre, így nincs DMA), táp és órajel (Vcc, Vss, CLK), interrupt (INTR, INTA), alaphelyzet (RST/RESET), memóriakezelés (MWR, MRD), DMA (HOLD, HLDA, és hogy az eszköz DMA-képes legyen, kell még **külön** memória és I/O vezetékcsoport), I/O (IOWR és IORD és/vagy SID és SOD).

A digitális számítógép felépítése, működése
Neumann-architektúra: a program az adatokkal a memóriában van, ez önmódosító és programot készítő (fordító)programokat tett lehetővé. Fő részei a CPU, RAM, és perifériák. A CPU vezérlője olvassa az utasításokat és vezérli az ALU-t, ami műveleteket végez. A perifériák a CPU akkumulátorába írnak és abból olvasnak.

A mikroszámítógépek általános felépítése, blokkvázlata, a funkcionális egységek jellemzői, a működés modellje
Lásd kettővel ezelőtti témakör (remélem).

Alaphelyzetbe állítás, órajel generálás
Alaphelyzet: programszámláló 0 lesz, a buszok szintén, az IC minden eleme resetel
Órajel generálás: oszcillátor (ismert rezgésszámú kristály) + erősítő + néha osztó v. szorzó

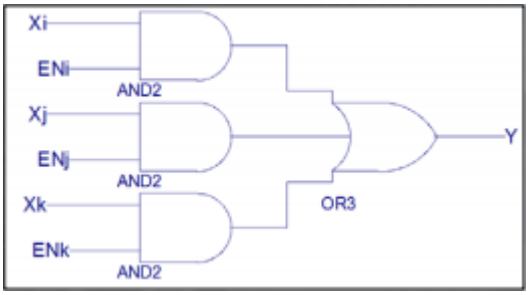
Memóriák illesztése sínrendszerekhez
Nem lehet belőle TL;DR-t csinálni, nézd meg a kifejtett verziót.

Időzítési viszonyok a memóriák szempontjából
DRAM bank: egy IC a RAM modulon, mátrixosan (sorok, oszlopok) tárol biteket. A bank-ek mellett egy SPD chip időzítési információkat tárol, hogy milyen gyorsan áll elő adat a mátrixból:
• T_{RCD} (Row-to-Column Delay): megnyitás parancs → sor érzékelő erősítőkre kerülése
• CL v. T_{CAS} (Column Access Strobe): olvasás parancs és az adat buszra kerülése közti idő
• T_{RP} (Row Precharge): sor bezárása, erősítők alaphelyzete, bitvezetékek töltése ennyi idő
• T_{RAS} (Row Active Time): minimum eddig nyitva kell legyen egy sor (kondik újratöltenek)

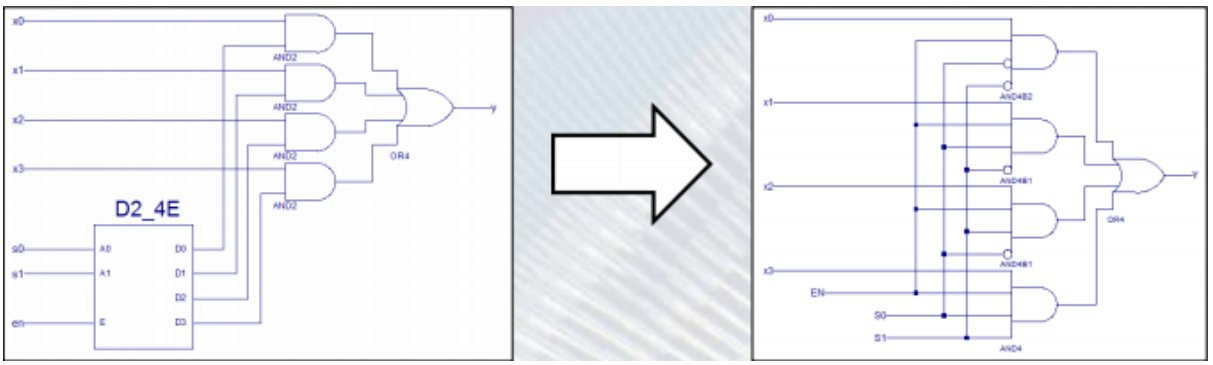
Az adatszélesség növelése
N bitből lesz N*2...

Flash memóriák alkalmazástechnikája
Lebegő kapus tranzisztorokba zárunk elektronokat, ha a kapu teli, a bit 0, ha üres, akkor 1. Több bit is tárolható, a bezárt elektronok mennyiségétől függően single/multi/triple-level cell-ről beszélünk. 1-ből 0-ba (programozás) úgy lépünk, hogy a source-t földeljük, és onnan nagy árammal húzzuk át az elektronokat a vezérlő gate felé, 0-ból 1-be (törlés) pedig ugyanez fordítva.

Források

Multiplexerek


• A legfontosabb funkcionális elem
• Alapvető feladata az adatforrás, adatút választás megvalósítása
• Xi, Xj, Xk az adatbemenetek
• ENi, ENj, ENk engedélyezés
Y = Xi*ENi + Xj*ENj + Xk*ENk
• Természetesen tetszőleges számú bemenetre
• Helyes működés feltétele: az EN bitvektor legyen 1-az-N-ből kódolású → Vagyis egy dekóder kimenete
• Megjegyzés: Az AND-OR hálózat helyettesíthető 3 állapotú Hi-Z kimenetű meghajtókkal is
• A MUX tehát lényegében egy DEK+AND-OR
• Azonban a MUX jelentősége miatt a beépített DEK funkciót külön nem szoktuk azonosítani



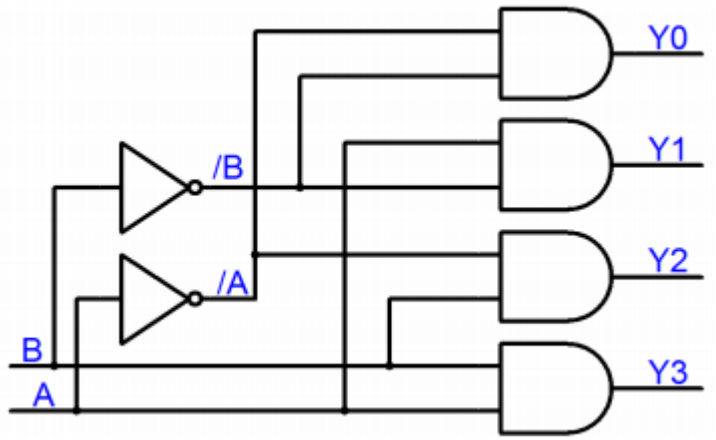
• Jellemző méretek: 2:1, 4:1, 8:1, 16:1
• A MUX is felépíthető hierarchikusan, fa struktúrában
• A struktúra generálható az LSb vagy az MSb felől
• LSb-vel kezdve első lépés a páros-páratlan bitek közti választás, majd így haladunk tovább, mindig az aktuális szomszédok között választva
• MSb-vel kezdve első lépés az adatbemenetek felezése (első-második fele), majd tovább a maradék felezése (negyedelés) és í.t. az utolsóig
• A fa felépíthető 4:1, 8:1 vagy nagyobb lépésekben is, ekkor kevesebb szint kell, de több bemenetűek a kapuk
• A választott megoldás sok más tényező függvénye is lehet, pl. egyszerűen a huzalozás megoldhatósága

Dekóderek

• A dekóder a binárisan értelmezett bemeneti jelekből az általános logikai függvény összes mintermjét előállítja (az igazságtábla minden sorához van „1”)

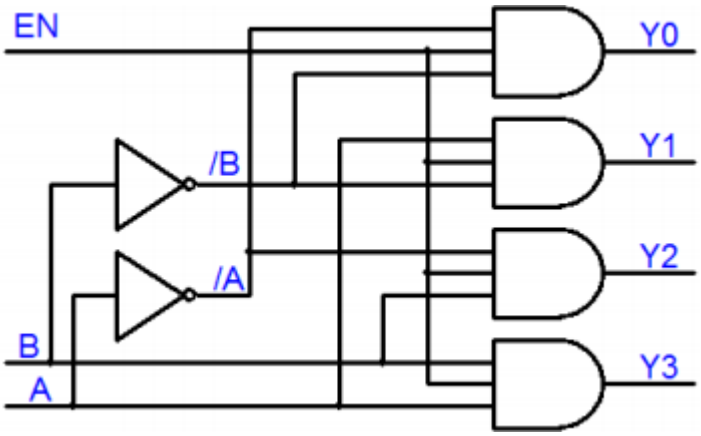
KIM = f (BEM)

• Ha a BEM egy n bites binárisan kódolt bemenet, akkor KIM egy 2^n méretű, 1-a-2^n-ből kódolású bitvektor, ezért hívjuk DE-KÓDOLÁSNAK
• A tipikus méretek: 1:2, 2:4, 3:8, 4:16. (de lehet 4:10 is)
• Lehet nagyobb méretben is, de esetleg érdemesebb az egyszintű dekóder helyett a többszintű sor-oszlop vagy hierarchikus fa struktúrájú felépítést használni.
• Egyszintű, közvetlen dekóderek felépítése
• A dekóder kimeneti bitjei egyenként megfelelnek az n bemenetű általános logikai függvények egy-egy mintermjének, azaz minden változó szerepel benne, ponált vagy negált értelemben.
• Minden kimenet egy n bemenetű ÉS kapu kimenete
• Pl. 2 bemenetre:
    • n bitre n INV és 2^n db n bemenetű ÉS
    • (Egy bitre csak 1 INV)

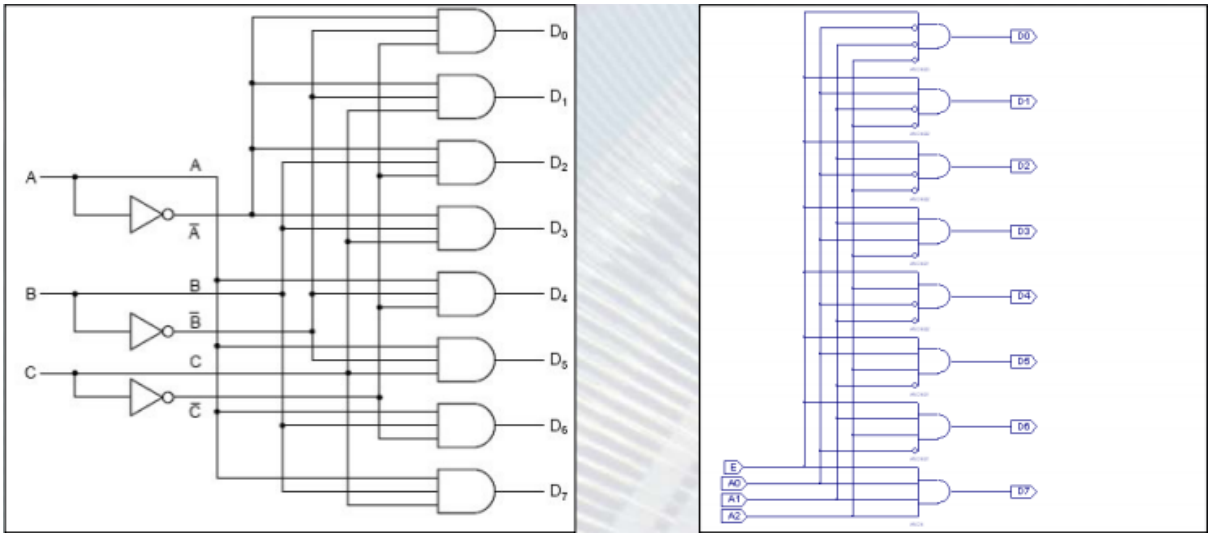


• A dekóder egyik kimenete értelemszerűen mindig aktív (A bemeneten mindig van egy kombináció)

• Ezért érdemes bevezetni egy ezt felülbíráló jelet: ENGEDÉLYEZÉS.
    • Ha EN = 0, minden kimenet inaktív
    • Így könnyen szabályozható a kimenetek által vezérelt egységek működése
    • Egyébként is ez a DEK legfontosabb funkciója
    • Könnyű a többszintű bővíthetőség

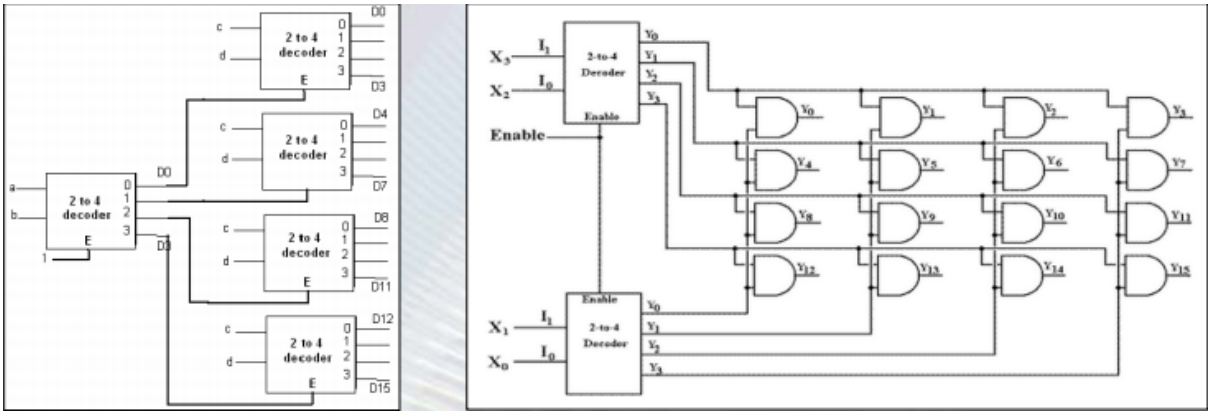


• Egyszintű, közvetlen dekóderek felépítése
• Ábrázolása inverterekkel vagy a bemeneteken jelölve a negált aktív szintet (rajzolhatóság, olvashatóság)



• Nagyméretű dekóderek
    • 60 kimenetre 60 db 6 bites ÉS kapu
    • Egy memóriában pl. 2^{17} kimenetre 131072 db 17 bemenetű ÉS kapu
• Más stratégia kell
    • Memória: sor-oszlop dekóder, mátrixszerű lokális engedélyezés, 2 bemenetű ÉS kapukkal
    • A korábban bevezetett engedélyező bemenet használatával többszintű, hierarchikus felépítés
• Nagyméretű dekóderek (moderált méretben mutatva)
    • 4:16 dekóder, engedélyező bemenettel, közvetlenül
    • 2:4 méretű, engedélyezhető dekóderből felépítve

HIERARCHIKUSAN 2D, SOR-OSZLOP módban



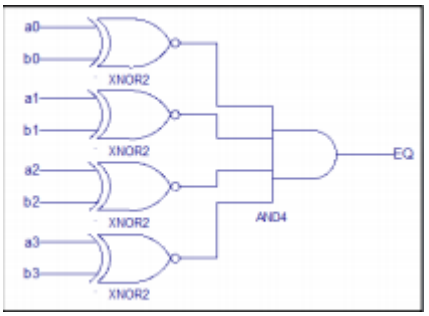
Komparátorok

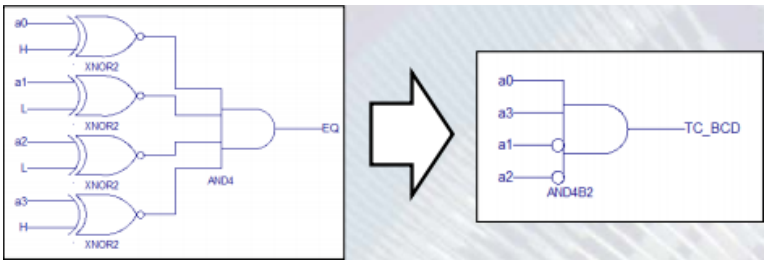
• Értékek, adatok összehasonlítása
  • Egyenlőség komparátor
  • Teljes funkciójú komparátor

a	b	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

Egyenlőség komparátor

• Logikailag a XNOR műveleten alapul
• Két bitvektor azonos pozíciójú bitjeit vizsgálja, hogy minden biten teljesül-e az egyenlőség feltétel
• a, b adatvektorok esetén n db két bemenetű XNOR kapu + 1 db n bemenetű ÉS kapu
• Tetszőleges kódolásra működik
• Fix érték vizsgálatára XNOR kapuk egyik bemenete fix 0 vagy 1 → programozott inverter
• (Pl. a 9 vizsgálata BCD számlálónál)



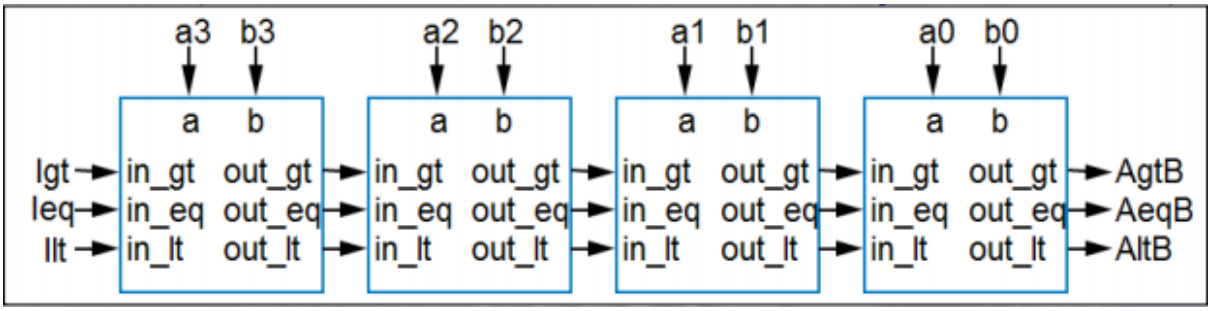


a	FIX	XNOR
a	0	/a
a	1	a

• Használhatunk komparátort is, de a 4 bemenet ÉS kapu természetesebb. Ilyenkor azt mondjuk, hogy dekódoljuk a bináris 9 értéket, mert valójában ez a teljes 4 változós függvény m_9 sorszámú mintermje, azaz egy 4:16 dekóder 9. kimeneti jele.

Teljes összehasonlító komparátor

• Valódi nagyság szerinti összehasonlítás
• 3 kimenet, ai = bi, ai > bi, ai < bi, (nem függetlenek)
• „Előző” bitpozícióról hasonló értelmű bemenetek
• Melyik az előző? Hogyan kaszkádosítsunk? Melyik jobb?
• Lehetséges MSb → LSb, de LSb → MSb irányba is
(Az összeadás csak az LSb → MSb irányban működik!!!)



• Tehát egy adott szinten: 3 + 2 = 5 bemenet, 3 kimenet

• MEGTERVEZHETŐ, egy adott bitpozícióra felírhatók az összefüggések → néhány kapu (nem tervezzük meg!)

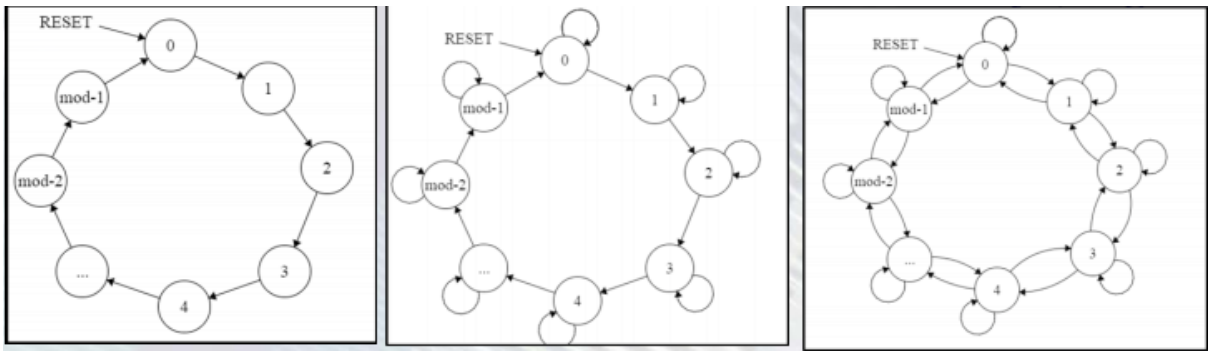
Számlálók

• Tetszőlegesen ciklikus állapotgépek (pl. a HF1 is 3 db számláló jellegű egység)
• Bináris mellett vannak shiftregiszter alapú számlálók (Gyűrűs, Johnson, LFSR)

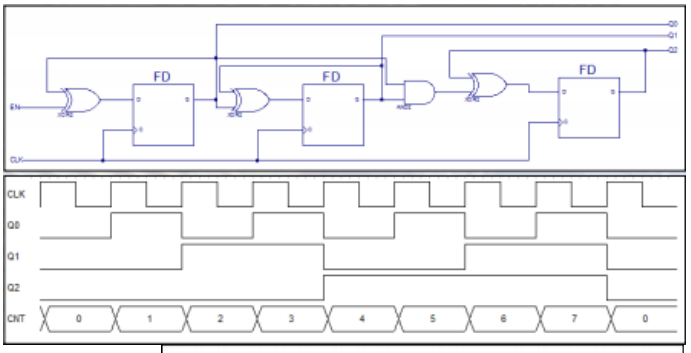
Bináris számlálók

• Bináris, Gray, BCD (0..9)
• A legfontosabb számláló típus
• n bit 2n állapot: 0 → (2n – 1), kompakt reprezentáció
• Állapotdiagramok a működési opciókra

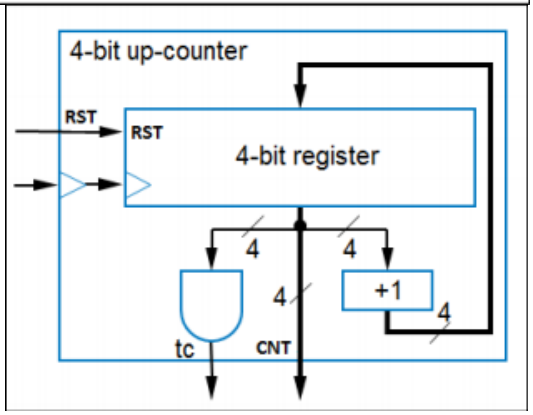
Egyszerű | Engedélyezhető | Kétirányú, eng.



• A tölthetőséget nem rajzoljuk fel, túl sok állapotátmenet, áttekinthetetlen lenne
• n bit 2n állapot: 0 → (2n – 1), kompakt reprezentáció
• Sokféle lehetséges realizálási lehetőség
    • A legegyszerűbb a bitenkénti TFF (XOR + DFF)
    • A TFF vált, ha a bemenete 1, egyébként tart
    • Az ÉS kapu biztosítja a (2x – 1) → 2x átmenetet, azaz az átvitelt, ha már minden kisebb helyiértékű bit értéke 1.
    • Szimmetrikus négyszögjelek
    • EN bemenet



• Általános modell
    • Egyszerű BIN számláló: REG + INC
    • INCREMENTER: Mint egy ADDER, de az egyik operandusa 0, és a CIN0 pedig 1, azaz CNT + 1
    • TC a végérték jelzés TC = 1, ha CNT = 1111
    • A 4 bites regiszter szinkron töltésű, törlésű (közös órajel)
    • A 16-os számláló az 1111 érték után átfordul és újra indul 0-ról



• Bináris számlálók alkalmazásai
    • Frekvenciaosztás
        • Pulzus kimenet kapuzott ütemezett működtetésre
        • Közel szimmetrikus négyszögjel tetszőleges felhasználásra

- Időzítés
    - Adott késleltetés/időzítés után egy pulzus kiadás
    - Adott időtartamú/szélességű pulzus előállítása
- Pulzusszámlálás
    - Él detektálással az engedélyező bemeneten
- Általános vezérlési feladatok
    - n bit, 2^n állapot, törléssel, töltéssel, engedélyezéssel vezérelt állapotátmenetek (RESET,CONT, JUMP)
    - Feltételjelek alkalmazása a megfelelő vezérlő bemenetnél
    - Több utas ugrás megoldása nem gazdaságos

Bináris aritmetikai műveletek

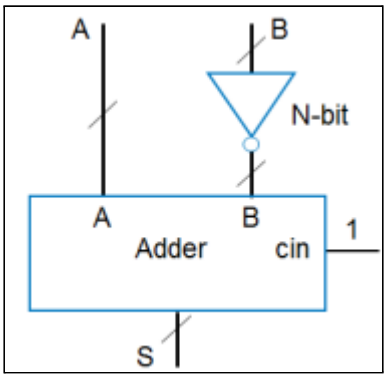
Összeadás

Összeadás szabályai (általában 2 operandus között): 0 + 0 = 0, 1 + 0 = 1, 0 +1 = 1, 1 + 1 = 10, ahol az 1 az átvitel a következő, eggyel magasabb helyiértékre. Ugyanez 3 bitre (a kaszkádolt átvitelhez): 0+0+0=0, 0+0+1=1, 0+1+1=0, és van átvitel, végül 1+1+1=1 és van átvitel.

		1		
	0	1	1	0
+	0	0	1	1
	1	0	0	1

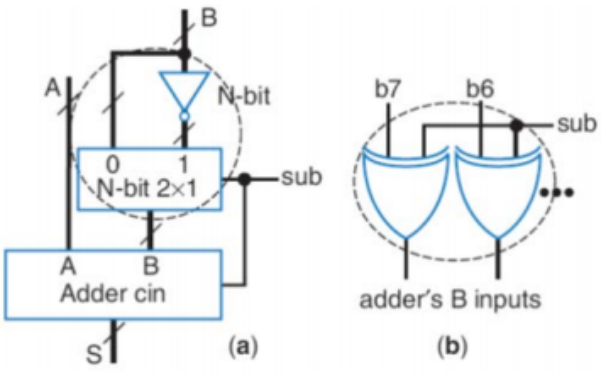
Kivonás

- Szerepelt korábban az 1 bites teljes összeadó
- Ebből kaszkádosítással készítettünk több biteset
- Az összeadó jól működik pozitív és 2's komplemenes negatív számokra is! (Ezért terjedt el a 2's komplemens számábrázolás)
- Tudjuk képezni egy szám (-1) szeresét, ezért lehetséges az A-B művelet végrehajtása az ismert A + ((-1)*B) összefüggés alapján.
- (-1)*B → Minden bitet invertálunk és hozzáadunk 1-et, ami pontosan a 0. pozíció Cin jele.
- Ezt használjuk komparátor helyett!



Teljes összeadókból felépülő összeadó/kivonó

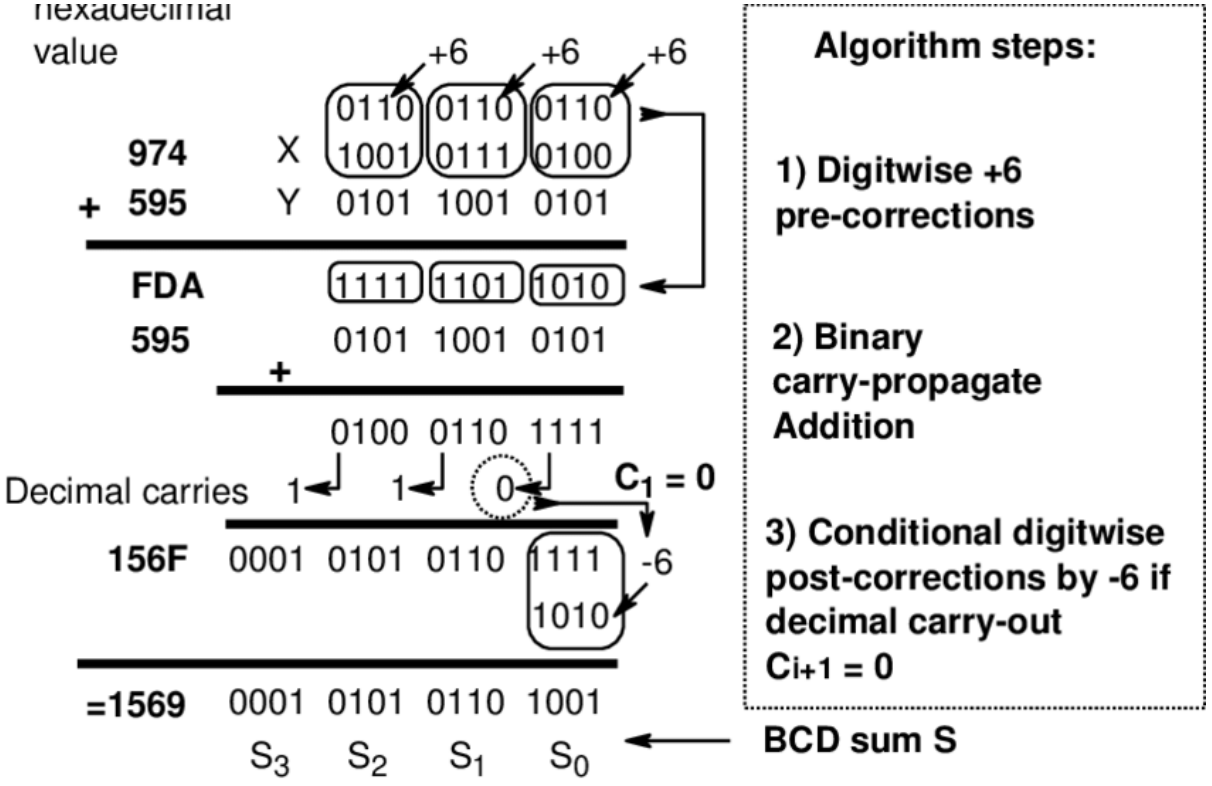
- Művelettől függően B normál vagy invertált értéke jut az ADDER bemenetére, mialatt a Cin 0 vagy 1.
- Megvalósítás: ADD inverz/SUB vezérlő bemenet ADD inverz/SUB = 0 összeadás, ADD inverz/SUB =1 kivonás
- B bemenet kialakítása: B, B inverz és 2:1 MUX
- XOR kapu, mint vezérelhető INVERTER
- A Cin közvetlenül az ADD inverz/SUB vezérlőjel



Túlcsordulás kezelése

Nincs hozzá anyag, de végtelen módszer van, a legegyszerűbb +1 bitet felvenni, ami a gyakorlatban a processzor carry-je.

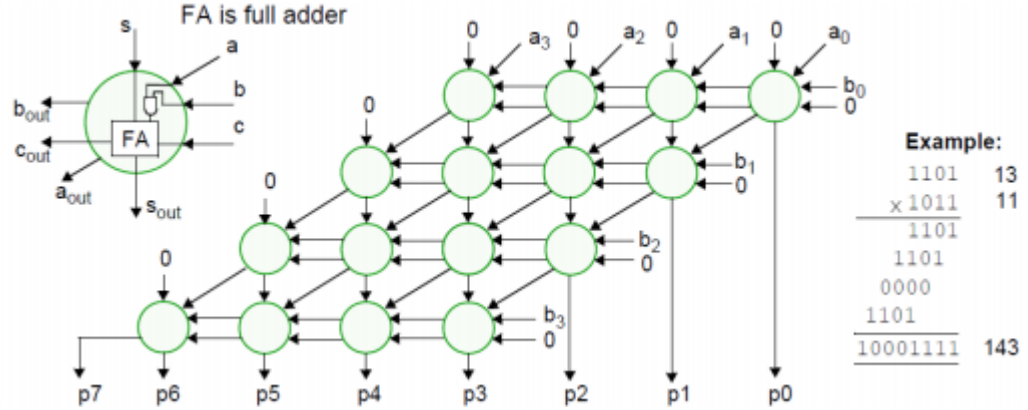
BCD számok összeadása



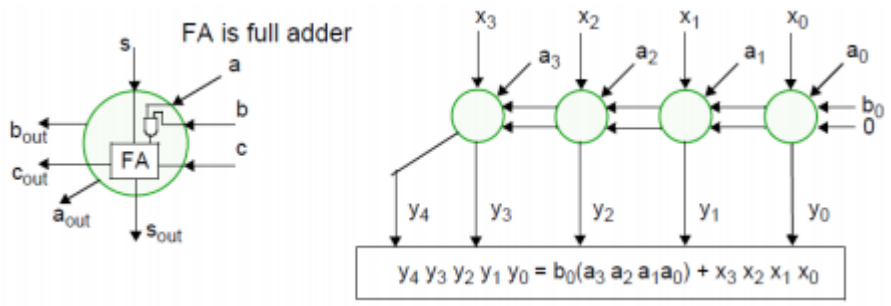
Szorzás megvalósítása összeadó áramkörökkel

$$P = \sum_i^{N-1} \sum_j^{M-1} a_i * 2^i * bj * 2^j = \sum_i^{N-1} \sum_j^{M-1} a_i * bj * 2^{i+j}$$

• Párhuzamos tömbszorzó (HW)



• Iteratív soros-párhuzamos szorzó (HW vagy SW)

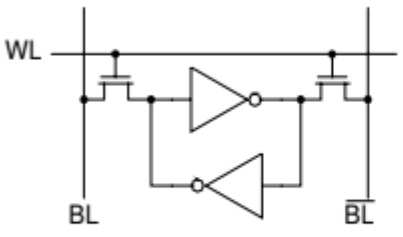


Memória áramkörök: írható és olvasható memóriák

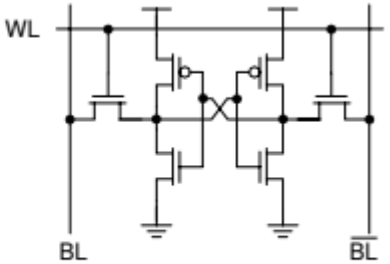
• Logikailag a regisztertömbre hasonlít
• Dimenzióban jelentősen eltér
• Jellemző méretek:
    • Adatszélesség: (4), 8-16-32 bit
    • Adatszavak száma: 210 – 220, (1 Ki – 1 Mi adatszó), technológia függő
        • Hogyan lehetséges? Nem DFF bittároló, hanem kifejezetten a legegyszerűbb megoldások a nagy adatsűrűség érdekében (6T, 1T)
• Fő típusok (használat szerint):
    • ROM: csak olvasható memória (Read Only Memory), az adatok programozással kerülnek bele
    • RAM: írható-olvasható memória (Random Access Memory) Tetszőlegesen elérhető (címezhető) memória

Statikus RAM felépítése

Az egyetlen bit tárolására képes SRAM tárolócella (ábra) nagyon emlékeztet egy egyszerű flip-flop-ra: a keresztbe kötött inverterek egy bistabil multivibrátort (lol) valósítanak meg. Ez az áramkör mindaddig képes megőrizni a bit értékét, amíg ellátjuk tápfeszültséggel. Az ábrán az inverterek mellett két tranzisztor is látható, melyek szerepe, hogy az inverterek kimenetét a bitvezetékre (BL, bit line) vezessék, ha a szóvezetékkel (WL, word line) a tárolócellát olvasás vagy írás műveletre jelöljük ki. Mivel egy invertert két tranzisztorral meg lehet valósítani, az SRAM memóriában egyetlen cella tárolásához 6 tranzisztor szükséges (lásd ábra, jobb oldal), amit a szakirodalomban ”6T” cellának neveznek. (Léteznek 4T és 10T SRAM cella struktúrák is, de ezekkel most nem foglalkozunk, a 6T messze a legelterjedtebb).



Inverterekkel



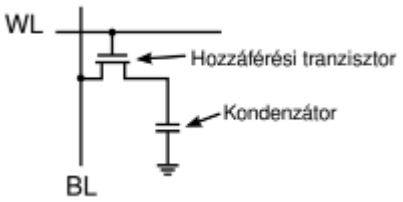
Tranzisztorokra lebontva

Az olvasás a következőképpen zajlik. Először az előfeszítő áramkörök a ponált és a negált bitvezetékeket egyaránt logikai 1 szintre húzzák fel, majd a bitvezetékekről lekapcsolódnak. A bitvezetékek kapacitása miatt a bitvezetékek feszültségszintje megmarad. (Ez az előkészítő lépés azért hasznos, mert a celláknak könnyebb/gyorsabb egy

előre feltöltött bitvezetéket 0-ra lehúzni, mint 1-re felhúzni.) A bit értékének kiolvasásához a WL szóvezetékre logikai 1 értéket adunk. A hozzáférési tranzisztorokon át a bitek értéke megjelenik a bitvezetékeken: ha a bit értéke 1, akkor a BL 1 marad, BL inverzt (disclaimer: ahol nagybetűk után inverzt írok, az a föléhúzással jelölt inverz, csak ez a platform nem tud olyat) pedig 0-ra húzza le a bit inverze; a 0 értékű bitnél pedig a helyzet éppen fordított. A BL és a BL inverz különbségét az érzékelő erősítők érzékelik, és a kimenetükön rendelkezésünkre áll a bit értéke. Minél érzékenyebbek az érzékelő erősítők, annál gyorsabb a kiolvasás. A bitek felismerését az is segíti (és gyorsítja), hogy nem abszolút jelszinteket, hanem csak a BL és BL inverz különbségét kell felismerni. Az olvasás az SRAM állapotát (a tárolt bit értékét) nem változtatja meg.

Az írás hasonlóképpen zajlik. A ponált és negált bitvezetékekre ráadjuk a beírandó bitek logikai értékének megfelelő feszültséget: ha a bit értékét 1-be szeretnénk állítani, akkor BL = 1, BL inverz = 0; ha 0-ba, akkor BL = 0, BL inverz = 1. Ezt követően a WL szóvezetékre logikai 1 értéket adunk. A cella által tárolt bit értéke a hozzáférési tranzisztorain keresztül felveszi a bitvezetékek által rákényszerített értéket, mivel a bitvezetékek meghajtó tranzisztorai erősebbek, mint a cellák tranzisztorai.

Dinamikus RAM felépítése



Dinamikus RAM esetén a bitek tárolása teljesen másképp történik: a flip-flop helyett egy kondenzátor töltöttsége hordozza az információt (lásd ábra). A kondenzátor melletti hozzáférési tranzisztor szerepe az, hogy a kondenzátort összekösse a bitvezetékkel, írás vagy olvasás céljából. Mivel egyetlen bit tárolásához 1 tranzisztor és 1 kondenzátor szükséges, a DRAM cellák felépítését ”1T1C” névvel is illetik.

Ha ki szeretnénk olvasni a tárolt bit tartalmát, először is elő kell feszíteni a bitvezetéket, mégpedig logikai 0 és 1 szint közé ”félútra”. Ezután a szóvezetékre logikai 1-et adunk, aminek a hatására a hozzáférési tranzisztor zár, és a kondenzátor összekapcsolódik a bitvezetékkel, töltése (vagy a töltés hiánya) pedig módosítja a bitvezeték feszültségszintjét. A bitvezeték végén elhelyezkedő érzékelő erősítők érzékelik ezt a (kondenzátor apró mérete miatt egyébként igen kis) változást, és előállítják a megfelelő logikai 0 vagy 1 szintet. A DRAM tárolócellák nagyon fontos tulajdonsága, hogy a kiolvasás destruktív, hiszen például egy 1-es bit kiolvasáskor a kondenzátor töltése a bitvezetéken át távozik, ezért a kiolvasás után egy külön lépésben rögtön vissza is kell állítani számára az elveszett töltését.

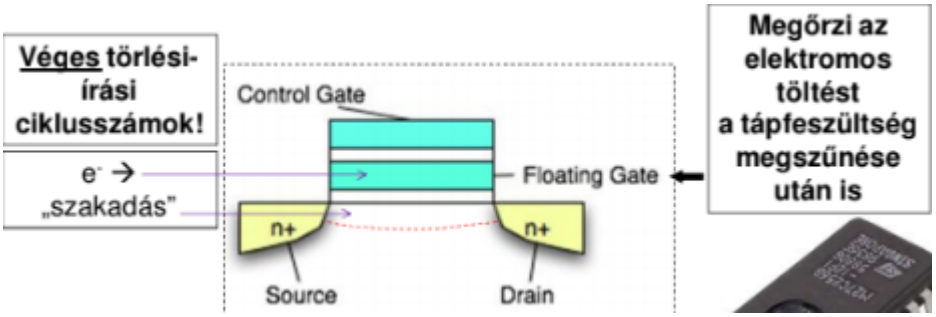
A tárolócella írása hasonlóan történik. A szóvezeték ”kijelöli” a cellát (a hozzáférési tranzisztor a kondenzátort a bitvezetékre kapcsolja), majd a tárolni kívánt logikai szintnek megfelelő töltést a bitvezetéken keresztül a kondenzátorba kényszerítjük.

Sajnos a DRAM tárolócellák a bitek kiolvasása nélkül is elveszítik tartalmukat, mert a kondenzátor töltése képes magától elszivárogni. Emiatt időnként (néhányszor 10 ms-onként, jellemzően 64 ms-onként) a tárolt információ frissítésre szorul, ami abból áll, hogy ki kell olvasni és vissza kell írni a tárolt bitet. Ezt a fajta megoldást pont azért hívják dinamikus RAM-nak, mert periodikus frissítésre van szüksége.

ROM jellegű memóriák

- ROM: Read-Only Memory (csak olvasható, maszkprogramozott)
- PROM (Programmable ROM): egyszer programozható (égethető be) a tartalom

- EPROM (Erasable-Programmable ROM): UV-fénnyel törölhető
- EEPROM (Electrically EPROM): elektronikusan törölhető
- elektronszakadás: a gate-ek közé szorított elektronok kiszabadulhatnak és ott ragadhatnak, ezzel módosíthatatlanná téve a bitet



Sínrendszerek definíciója, kialakulása, osztályozása

Magyarul: busz. Egy IC-vel beszélő vezetékek csoportja.

Vezetékek csoportosítása, az egyes vezetékcsoportok feladata

A 8085-ön bemutatva:

Cím/adatbusz:

- A_0-A_{15}: adat- és címbusz, az adat mindkét irányból írható/olvasható, a cím read only

Vezérlőjelek:

- ALE (address latch enable): a címbusz alsó 8 bitjét állítja adatbusszá, ha 1
- READY: 1, ha a periféria tud adatot küldeni/fogadni
- IO/~M (I/O or memory), néha IO/M vagy IO/M': ha ilyen vezeték van, akkor a memória- és I/O-műveletek közös vezetékeket használnak, egy write és read vezeték van WR (write) és RD (read) néven, és egy ciklusban csak az egyik írható. DMA adatátvitel akkor lehetséges, ha az IC tud egyszerre memóriát és I/O-t olvasni/írni, tehát van MWR, MRD, IOWR, és IORD vezeték. Ez IO/M által kombinált WR és RD mellett nem megoldható.

Táp és órajel:

- Vcc: 5V
- Vss: föld
- CLK (clock): órajel, felfutó és/vagy lefutó éle triggerel változást a rendszerben

Interrupt:

- INTR (interrupt request): megszakítás kérés
- INTA (interrupt acknowledgment): az IC válasza az INTR-re

Alaphelyzet:

- RST vagy RESET: ha aktív, a következő órajelre alaphelyzetbe áll az IC
- RESET OUT: jelzi, hogy az IC reset alatt áll

Memóriakezelés:

- MWR (memory write): az adatsín memóriába íródik
- MRD (memory read): az adatsínre olvasódik a memória

DMA:

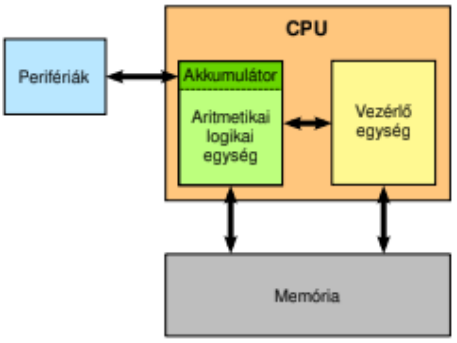
- HOLD: az IC nem csinál semmit, amíg az adatsínét DMA-nak használják
- HLDA: jelzi, hogy az IC elfogadta a HOLD állapotot
- Hogy az IC DMA-kompatibilis legyen, kell még hozzá memória- **és** I/O-kezelés is

I/O-kezelés:

- IOWR (I/O write): az adatsín I/O-ra íródik
- IORD (I/O read): az adatsínre olvasódik az I/O
- Soros I/O esetében SID és SOD van (serial input/output device)

A digitális számítógép felépítése, működése

A Neumann-architektúra legnagyobb újítása elődeihez képest, hogy a programot nem mechanikus kapcsolósorok segítségével kell beállítani, hanem az az adatokkal együtt a memóriában van eltárolva. Ezt a megoldást az motiválta, hogy a mechanikus kapcsolókhoz képest így sokkal könnyebben és gyorsabban lehet a programot (az utasítássorozatot) bevinni, megváltoztatni, illetve egymás után több különböző programot lefuttatni. Emellett annak, hogy a program és az adatok is egyazon memóriában vannak elhelyezve, van még egy fontos járulékos hozadéka: lehetővé válik a ”programot generáló programok” készítése, ekkor indult el tehát a fordítóprogramok karrierje is.



Egy Neumann-architektúrájú számítógép 3 fő komponensből áll (ábra): CPU, memória, bemeneti/Kimeneti perifériák.

A processzor (CPU) tartalmazza egyrészt a vezérlő egységet (control unit), másrészt az aritmetikai-logikai egységet (arithmetic-logic unit, ALU). Az ALU végzi a tényleges számításokat: összead, kivon, szoroz, oszt, logikai és, vagy műveleteket végez. A vezérlő egység értelmezi a memóriából lehívott utasításokat, meghatározza az utasítások végrehajtási sorrendjét, előállítja az ALU részére a vezérlőjeleket.

A bemeneti/kimeneti perifériák biztosítják a kapcsolatot a számítógép és a külvilág között. Az ALU-ban lévő akkumulátor (operandus ill. eredménytároló) tartalma egy kimeneti perifériára kiírható, vagy egy bemeneti perifériából beolvasható.

A memória adott bitszélességű adategységek tárolója, melyben minden adategység egy egyedi címmel rendelkezik. Már említettük, hogy a memóriában tároljuk az utasításokat és az adatokat is. Azonban ezek a memóriában semmilyen módon nincsenek megkülönböztetve, az, hogy egy memóriabeli objektum utasítás-e, vagy adat, attól függ, hogy a processzor utasítást hív-e le onnan, vagy adatot. Ugyanez igaz az adatokra is: a memóriában nincs letárolva az adatok típusa. Egy memóriatartalom interpretálható egészként, karakterként, vagy lebegőpontos számként egyaránt, attól függően, hogy azon egész, karakter, vagy lebegőpontos utasítások dolgoznak-e.

A mikroszámítógépek általános felépítése, blokkvázlata, a funkcionális egységek jellemzői, a működés modellje

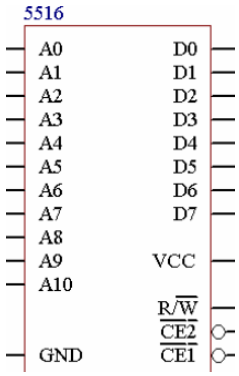
Lásd kettővel ezelőtti témakör (remélem).

Alaphelyzetbe állítás, órajel generálás

Alaphelyzet: a következő órajelre a programszámláló 0-ra áll, a buszok szintén, és az IC összes eleme (pl. számlálók) is kap a jelből.

Órajel generálás: oszcillátor (ismert rezgésszámú kristály) + erősítő. Az erősítő általában visszavezeti a jel inverzét az oszcillátorba, hogy a rezgést fenntartsa. Az órajel osztható és szorozható.

Memóriák illesztése sínrendszerekhez

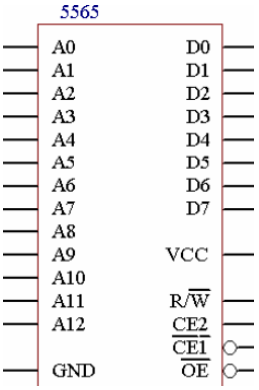


5516

Ugyanaz, mint a 2716, csak az RD-t külön rá kell kötni R/W-re, illetve A11-et a CE1-re és CE2-re.

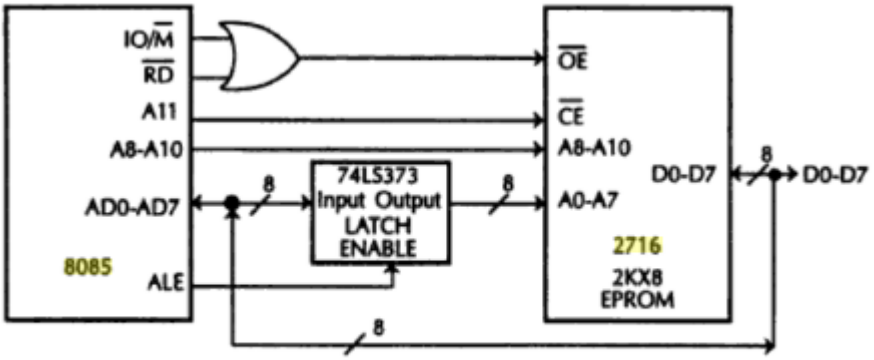
5565

Ugyanaz, mint a 2716, csak az RD-t külön rá kell kötni R/W-re, illetve A11-et a CE1-re és inverteren át a CE2-re.



2716 és 2732

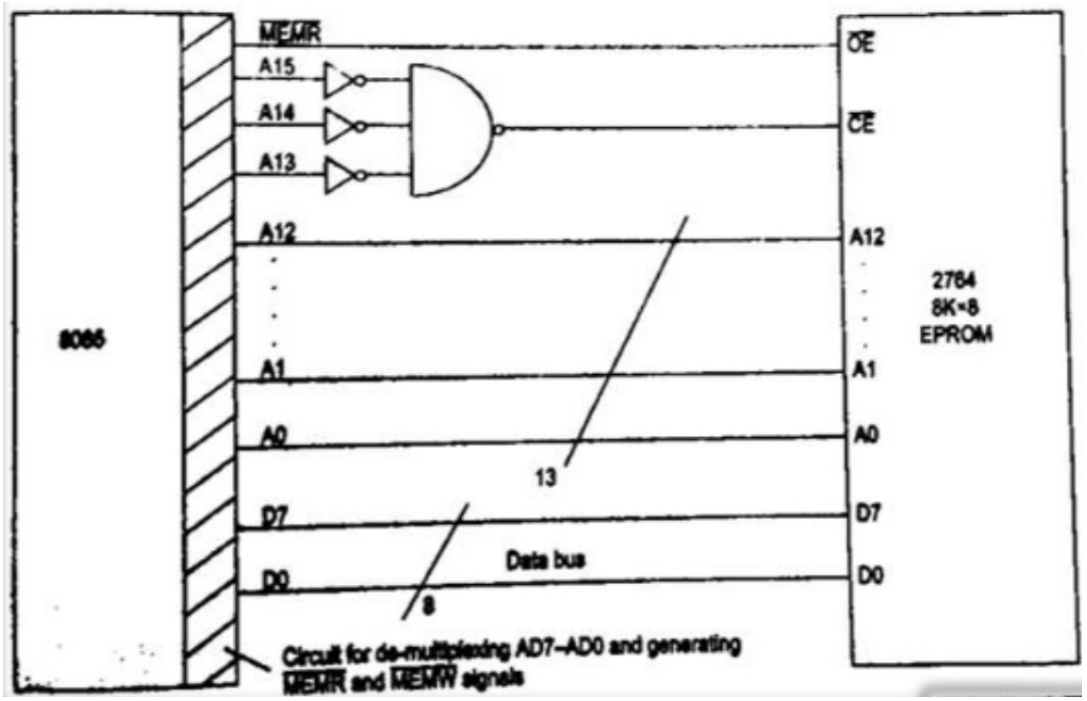
A 74LS373 egy 8-as latch, a papírra max annyit írj.



(a) 8155 - 2716 interface using internal latches.

2764 és 27128

A MEMR valójában RD, AD7-AD0 kiválasztás kell ALE alapján, a 2716 módján.



Időzítési viszonyok a memóriák szempontjából

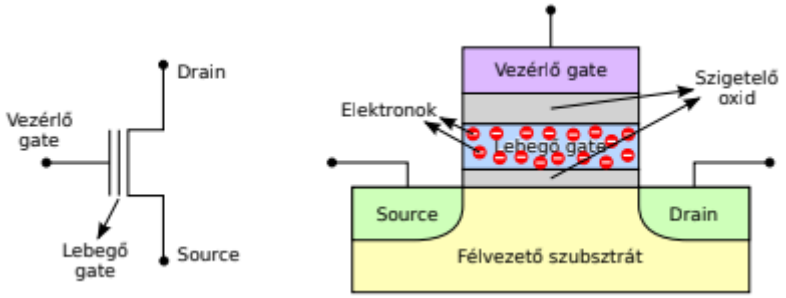
DRAM bank: egy IC a RAM modulon, mátrixosan (sorok, oszlopok) tárol biteket.
A DRAM-nak a parancsok végrehajtásához idő kell, melyet a memóriavezérlőnek figyelembe kell vennie, miközben memóriaműveleteket végez. Egy DRAM memória modul a DRAM chip-eken felül tartalmaz még egy IC-t (Serial presence detect, SPD), mely ezeket az időzítési információkat tárolja, a rendszer indulásakor a memóriavezérlő innen szerez tudomást a parancsok késleltetéséről.
A sok-sok különféle időzítési érték (timing) közül érdemes kiragadni az alábbi négyet, mert egyrészt ezek a legfontosabbak, másrészt ezekkel szokták jellemezni a DRAM chip-ek késleltetését:

• T_{RCD} (Row-to-Column command Delay): Ennyi ideig tart egy sor megnyitása, vagyis ennyi idő telik el a megnyitás parancstól számítva addig, amíg a sor tartalma megjelenik az érzékelő erősítőkben.
• CL, vagy T_{CAS} (Column Access Strobe time): Egy nyitott sor egy oszlopának a kiolvasásához szükséges idő. Az olvasás parancs után ennyi idő múlva jelenik meg az (első) adat a modul adatbuszán.
• T_{RP} (Row Precharge): A PRECHARGE parancs végrehajtásához szükséges idő. PRECHARGE: az aktuális, nyitott sor bezárására szolgál. Az érzékelő erősítők alaphelyzetbe állnak, egyúttal feltöltik (precharge) az összes bitvezetéket is, hogy a következő sormegnyitás művelet, ha majd szükség lesz rá, gyorsan megtörténhessen.
• T_{RAS} (Row Active Time): Az a minimális idő, amíg egy sor nyitva lehet (ennyi idő kell a kondenzátorok töltésének regenerálásához). Ha figyelembe vesszük, hogy egy új sor megnyitása előtt elő-feltöltés is szükséges, megkapjuk, hogy a sor megnyitási parancsok nem követhetik gyakrabban egymást, mint $T_{RC} = T_{RAS} + T_{RP}$.

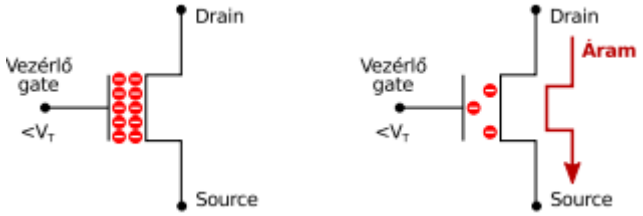
Az adatszélesség növelése

N bitből lesz N*2...

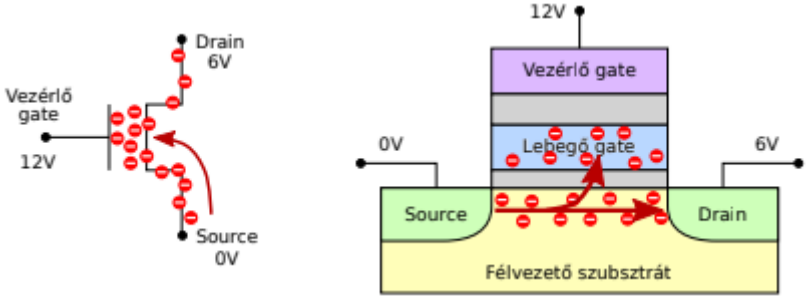
Flash memóriák alkalmazástechnikája



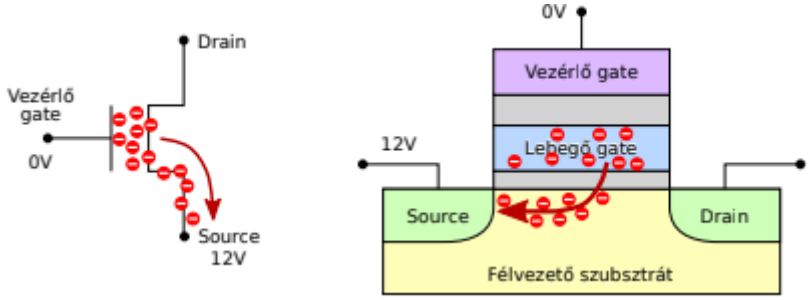
Lebegő kapus tranzisztorokba zárt elektronok jelentenek 0-s bitet, az üres kapu pedig 1-et. Úgy olvashatjuk ki a tárolt bit értékét, hogy kis feszültséget adunk a vezérlő gate-re, és megnézzük, hogy folyik-e áram a source és a drain között. Ha igen, akkor 1-es bitet, ha nem folyik áram, 0-ás bitet tárol a tranzisztor.



Váltás 1-ből 0-ba: ennek a műveletnek programozás a neve. Ehhez a váltáshoz a lebegő gate-et elektronokkal kell feltöltenünk. Ezt úgy érhetjük el, hogy a source-ot leföldeljük, a vezérlő gate-re és a drain-re nagy feszültséget adunk. Ilyenkor a nagy energiájú elektronok átjutnak a lebegő gate és a szubsztrát közötti szigetelő rétegen, és a lebegő gate-ben is maradnak.



Váltás 0-ból 1-be: ezt a műveletet törlésnek hívják. A vezérlő gate-et le kell földelni, a drain-t szabadon lebegve hagyni, és a source elektródára nagy feszültséget kell adni. Ekkor az alagúteffektusnak köszönhetően az elektronok a szigetelőrétegen átlépve elhagyják a lebegő gate-et.



A programozásnak és a törlésnek azonban van egy igen kellemetlen mellékhatása: a lebegő gate-be kényszerített, illetve az azt elhagyó elektronok néha beleragadnak a szigetelőrétegbe, és ott is maradnak. Ettől a szigetelőrétegnek megváltoznak az elektromos tulajdonságai, a tranzisztor zárásához szükséges feszültség pedig emiatt eltolódik. Minél többször töröljük a tranzisztort, annál több elektron szorul a szigetelőbe, ami egyre kevésbé fog szigetelni. A lebegő gate-es tranzisztor tehát öregszik (kopik). Egyszer aztán eljön az a pont, amikor már nem képes adatot tárolni. Az, hogy ez hány törlési ciklus után következik be, a gyártástechnológiától függ: minél kisebb a tranzisztor, annál hamarabb.

Ahogy láttuk, a lebegő gate-en lévő töltés jelenléte határozza meg, hogy a tranzisztor 0-ás, vagy 1-es bitet tárol-e. Ha vannak töltések, akkor 0-ás, ha nincsenek, 1-es bitet feleltetünk meg neki. Egy egyszerű ötlettel azonban meg lehet oldani, hogy egy lebegő gate-es tranzisztor több bitet is tárolni tudjon. A módszer lényege, hogy nem csak a ”van töltés” – ”nincs töltés” állapotokat különböztetjük meg, hanem több töltési szintet is használunk. Így a legegyszerűbb esetben négy töltési szinttel 2 bitet is tárolni lehet: ”11” bitek felelnek meg annak az esetnek, amikor nincs töltés a lebegő gate-en, ”10” bitet tárol a tranzisztor, ha kis, ”01”-át, ha több, és ”00”-át, ha sok töltés van rajta. A töltés mennyisége

határozza meg a tranzisztor zárásához szükséges feszültséget, minél több a töltés, annál nagyobb feszültség zárja a tranzisztort. Ha egy bit fér egy tranzisztorba, azt hívjuk SLC-nek (Single-Level Cell), ha kettő, akkor MLC-nek (Multi-Level Cell), ha három, akkor TLC-nek (Triple-Level Cell).