MEMÓRIÁK I-II.

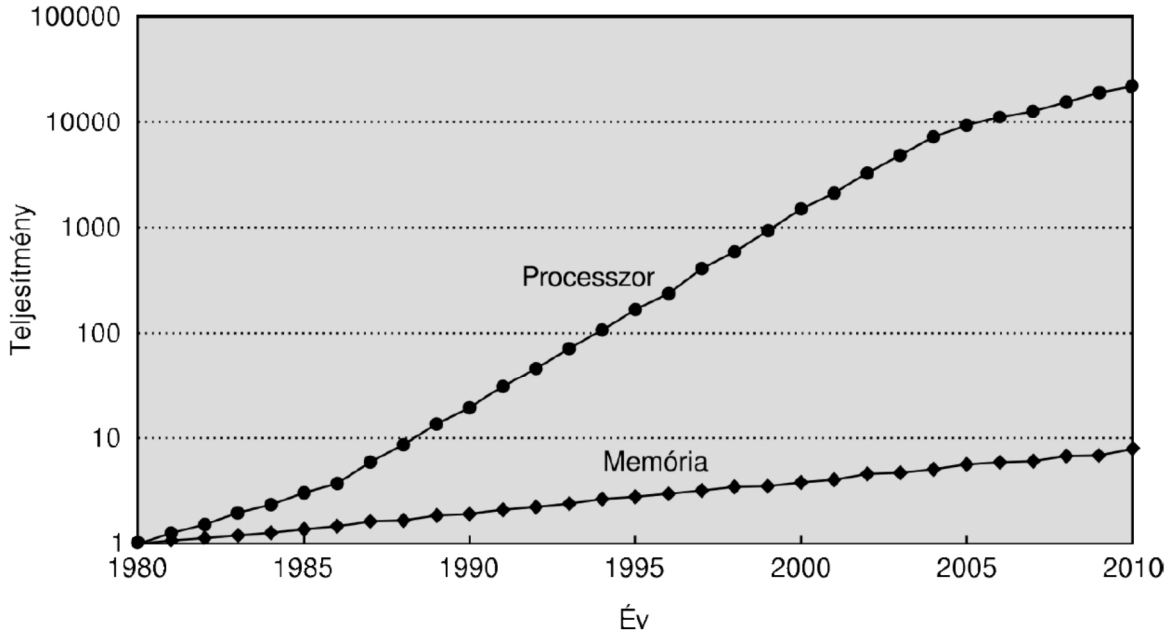
Tartalom

Bevezetés

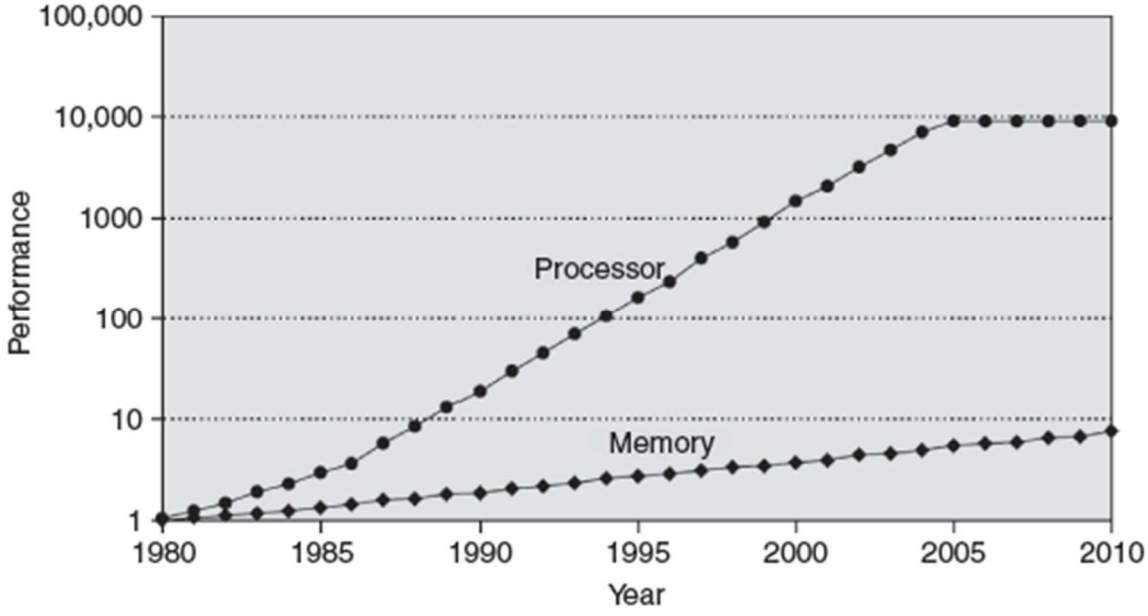
Memória és perifériák illesztése processzorhoz

Forrás BME

- Szűk keresztmetszet:
  - **Memória-sávszélesség!**

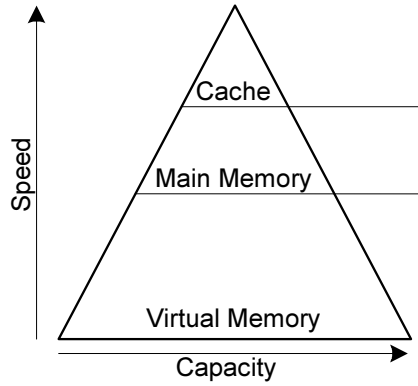


Forrás ELSEVIER -- nem ugyanaz?



Moore törvény → Sima Dezső cikke "Core-count Law"

Forrás: Elsevier

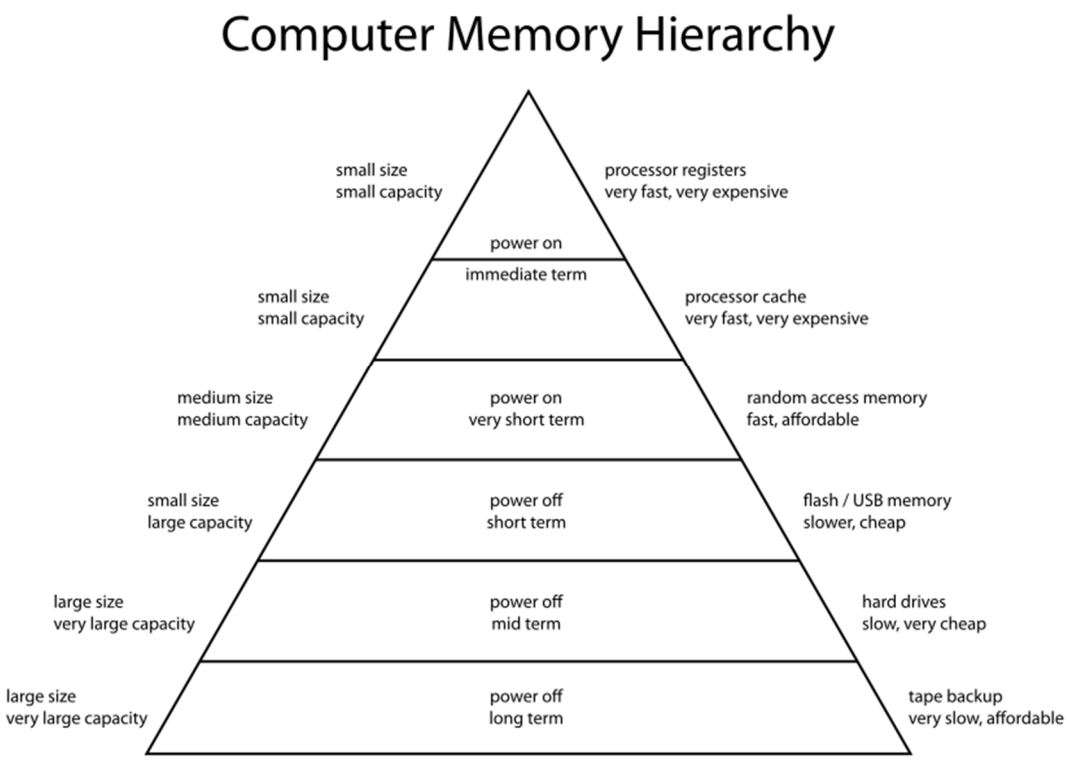


Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	$10,000	1	25+
DRAM	$10	10 - 50	10
SSD	$1	100,000	0.5
HDD	$0.1	10,000,000	0.1

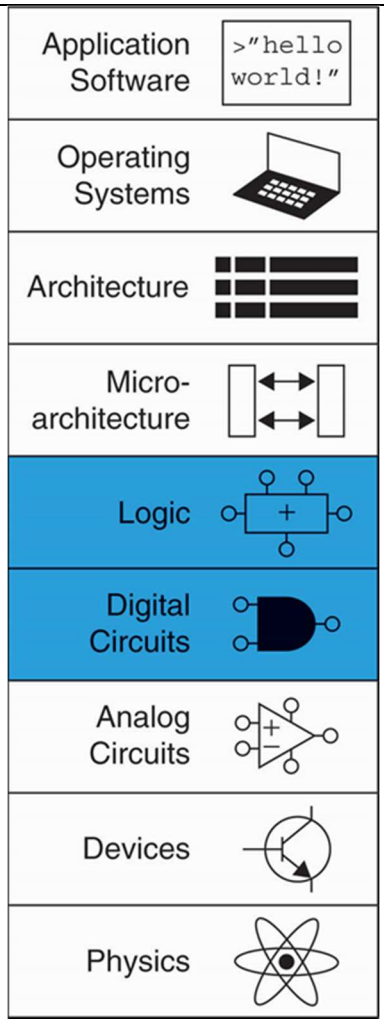
- **Physical Memory:** DRAM (Main Memory)
- **Virtual Memory:** Hard drive
  - Slow, Large, Cheap

A CPU címtartományánál több memória kellene
→ **tárbővítés**
Nincs annyi memoriánk amennyit a CPU címezni tud, a programok azt hiszik, van annyi
→ **virtuális tárkezelés**

Forrás: Wikipedia …



Forrás: Elsevier …



Application Software	programs
Operating Systems	device drivers
Architecture	instructions registers
Micro-architecture	datapaths controllers
Logic	adders memories
Digital Circuits	AND gates NOT gates
Analog Circuits	amplifiers filters
Devices	transistors diodes
Physics	electrons

Nagykapacitású adattárolók: diszk, CD-ROM, DVD

Memóriafajták
Csoportosítás és jellemzők

Felhasználás és elérési idő valamint kapacitás szerint

- Operatív memória – ma tipikusan félvezetős, régebben ferritgyűrűs
- Háttértár – tipikusan mágneses (mágneslemez, szalag)

Címzés módja szerint

- Hely szerint címezhető (dekóderrel kiválasztjuk, melyik memória rekesz)
- Tartalom szerint címezhető Content Addressable Memory (a tárolt információ egy része alapján azonosítjuk/találjuk meg) – speciális alkalmazások (lásd majd: cache, virtuális memória kezelés)

Tárolás jellege (maradandósága) szerint

- Read Only Memory (ROM) - állandó tartalmú (a felhasználás közben)
    - Maszk programozott ROM: A gyártó maszkkal írja.
    - PROM (Programmable ROM): A felhasználó egyszer, speciális eszközzel írhatja. Technológiától függően fuse (dióda elégetés) vagy anti fuse (szigetelőréteg vezetővé tétele) típusú megoldás.
    - UV EPROM (Ultra Violet Erasable and Electrically Programmable ROM) Programozó eszközben írható, UV fénnyel törölhető. Lebegő gate-es FET tranzisztor lebegő elektródájára felvitt töltéssel lehet annak állapotát programozni. A töltések eltávolítása UV fénnyel lehetséges.
    - EEPROM, E^2PROM (Electrically Erasable and Electrically Programmable ROM) a felhasználás helyén elektromosan extra időzítéssel többször (pl. 10000) újraírható. Működése: mint az EPROM, de a töltés elektromos úton távolítható el.
- Read Write Memory - változtatható tartalmú memória
- Technológia és az információ megtartása szerint
    - Statikus (Static) RAM (SRAM) (az információt bistabil multivibrátor tárolja) – a tápfeszültség szükséges és elégséges a beírt információ megtartásához.
    - Dinamikus (Dynamic) RAM (DRAM) (az információt elektródák közötti kapacitásban tárol töltés vagy annak hiánya hordozza) – néhány ms-onként frissítés szükséges.
    - Non Volatile RAM (NVRAM) Statikus RAM és EEPROM„ bitenkénti egyesítése”.

A változtatható tartalmú memória lehet

- A hozzáférés módja szerint
    - Serial Access Memory (shift regiszterként sorosan hozzáférhető)
    - Random Access Memory (bármelyik rekesz tartalma ugyanannyi idő alatt érhető el)

ROM, SRAM és NVRAM műveletek a bemenetek függvényében

ROM

MŰVELET (állapot)	$\overline{CE}$	$\overline{OE}$
standby	1	x
read	0	0

SRAM

MŰVELET (állapot)	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$
standby	1	x	x
write	0	1	0
read	0	0	1

NVRAM

MŰVELET (állapot)	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	$\overline{NE}$
standby	1	x	x	x
write	0	1	0	1
read	0	0	1	1
store	0	1	0	0
recall	0	0	1	1

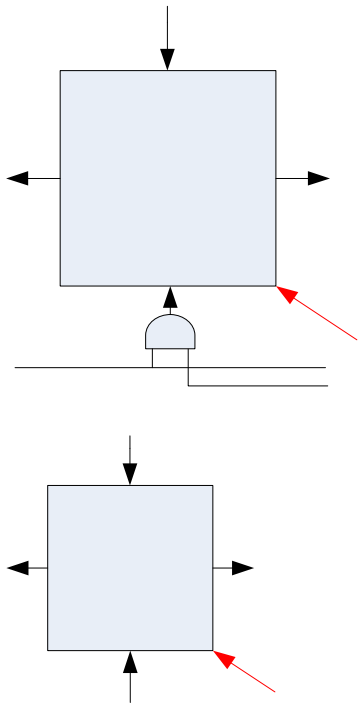
Memóriaszervezési megoldások, memória bővítés

Egy általános RAM esetén - egy memóriacellát D tárolónak tekintve - a memória szervezése lehet:

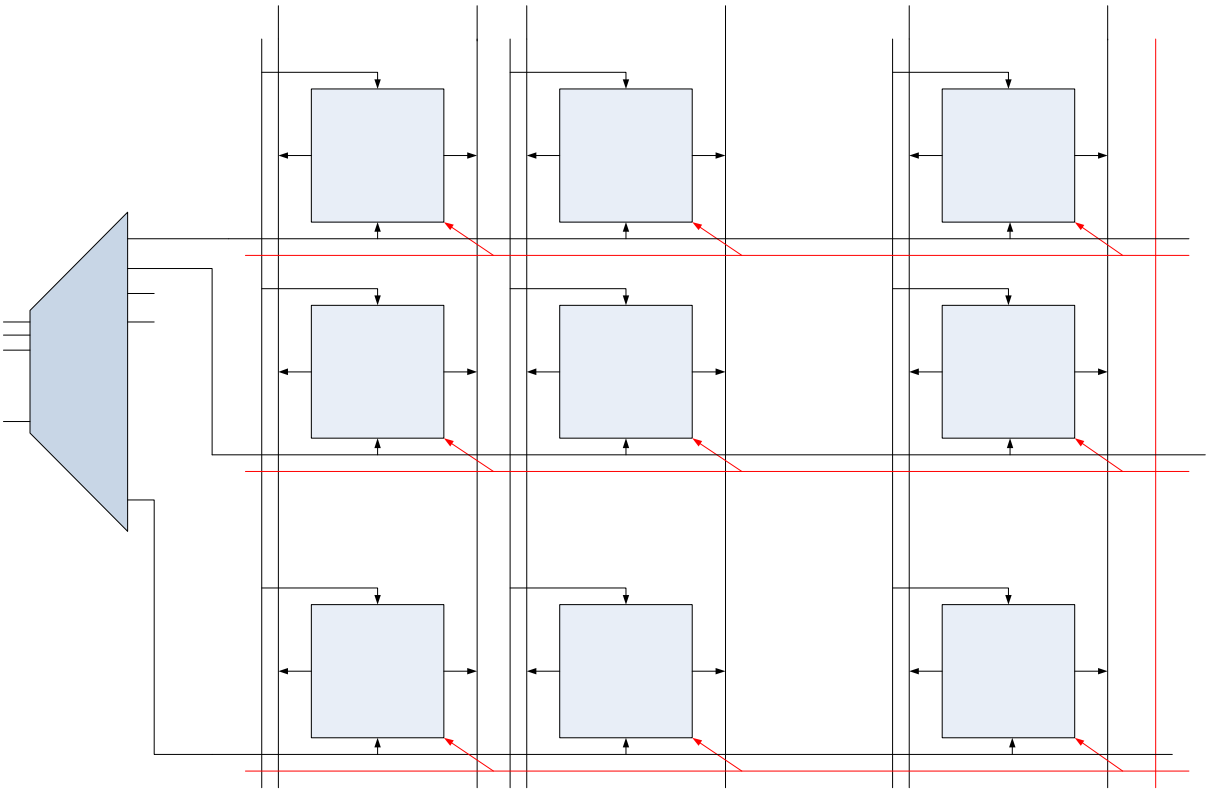
- Szószervezésű (1D)
- Bit szervezésű (2D)
- Módosított bit szervezésű (2,5D)

Memóriabővítés: szóhossz (szóhosszbővítés) és szószám (kapacitásbővítés) bővítésével

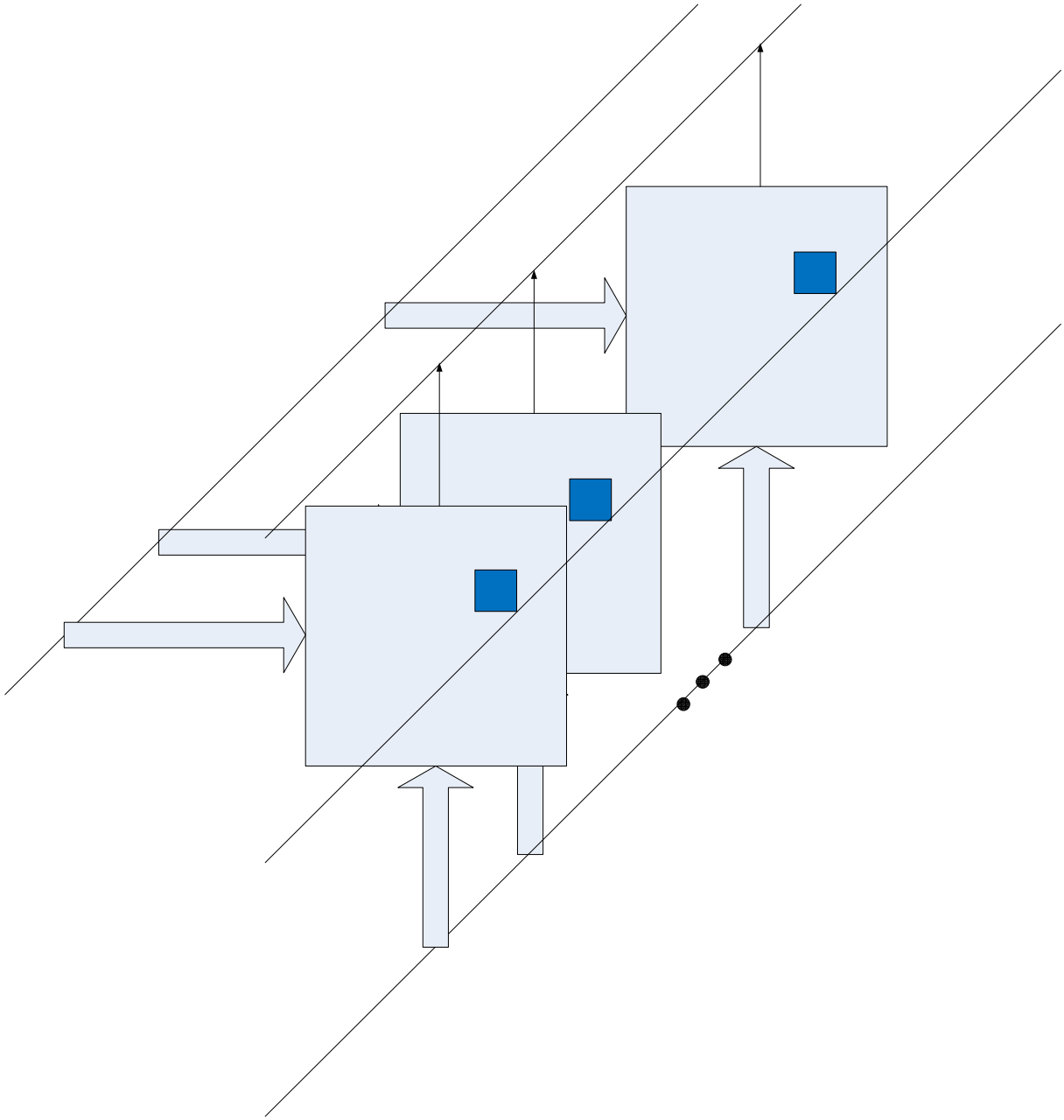
Memória cellák



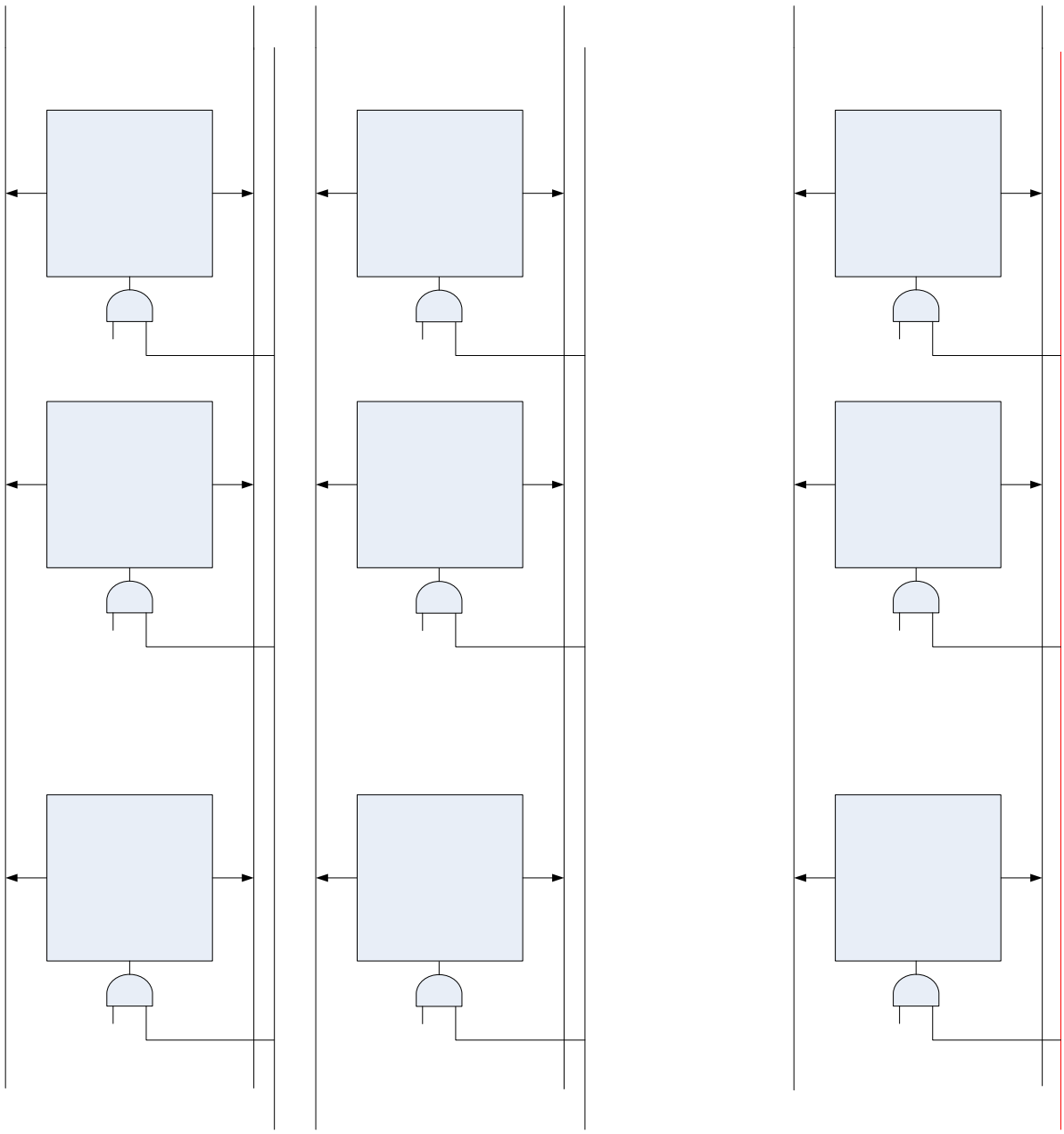
Szó ---- 2D



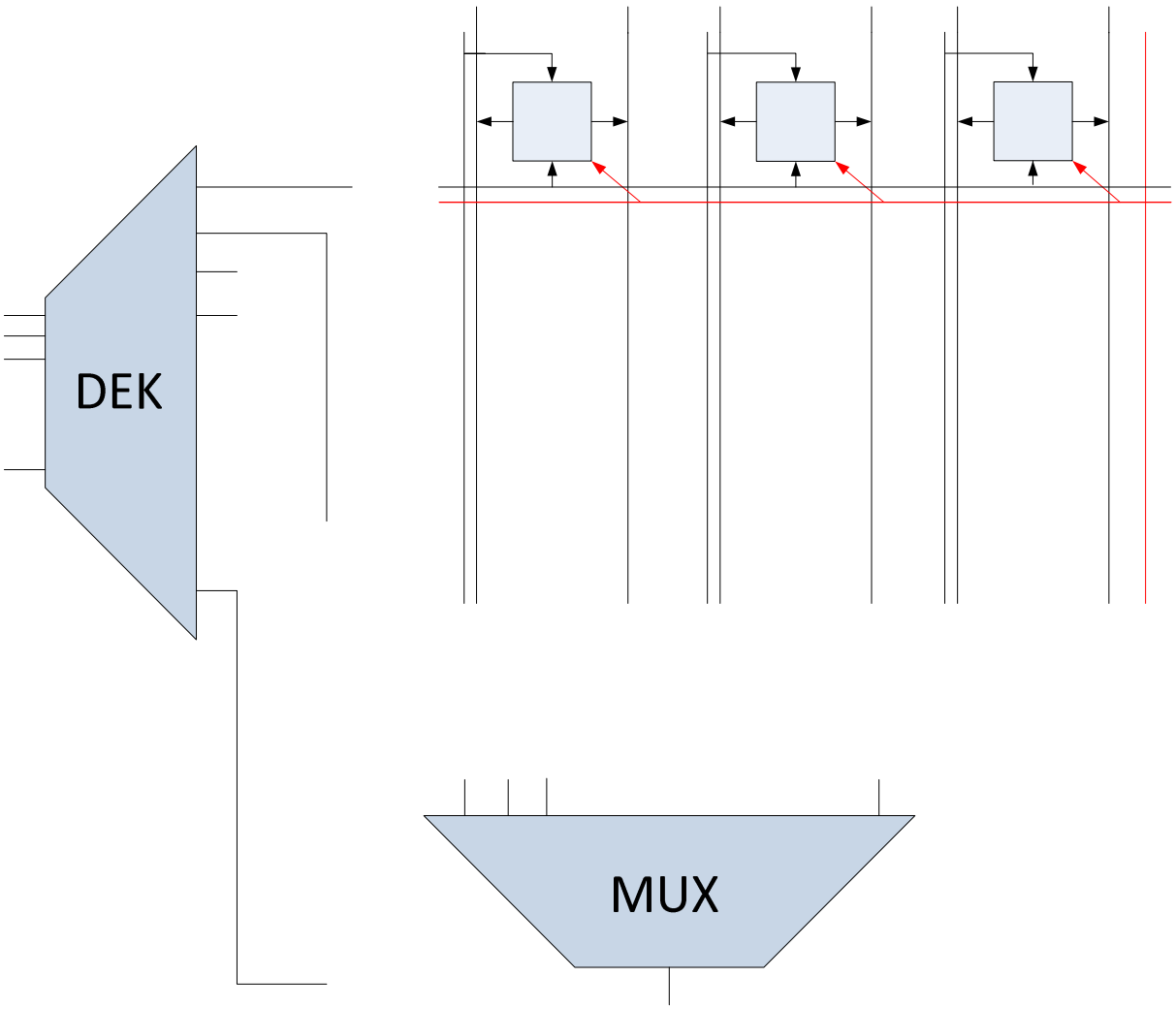
Bit ---- 3D



Bit ---- 3D

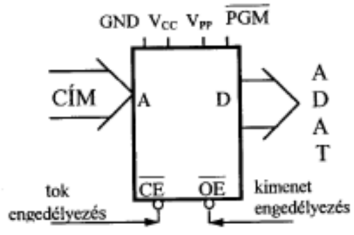


Vegyes ---- 2,5 D

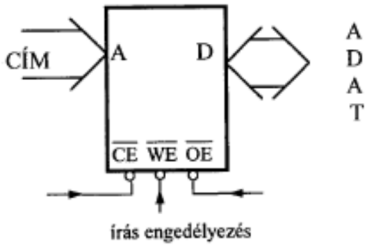


Egyes memóriatípusok külső illesztési felülete és időDiagramjai

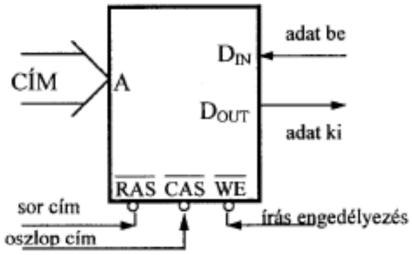
Forrás Horváth L



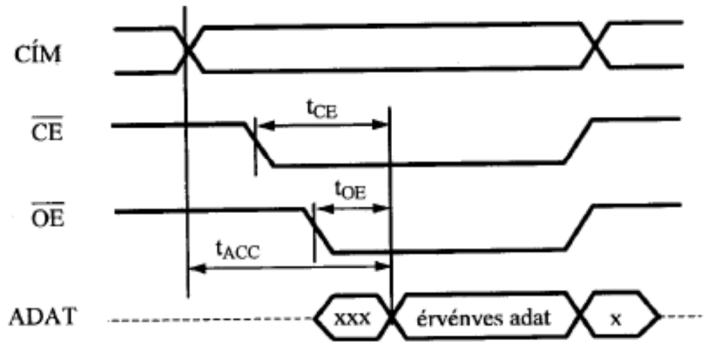
4.3. ábra UV EPROM /EEROM rajzjele



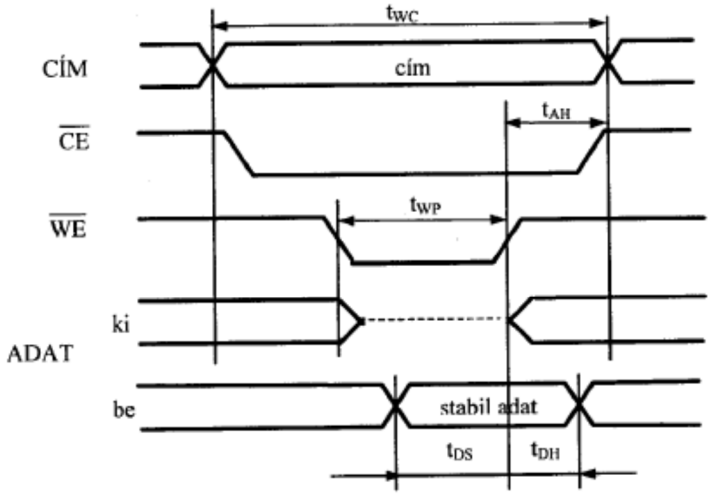
4.11. ábra Statikus RAM rajzjele



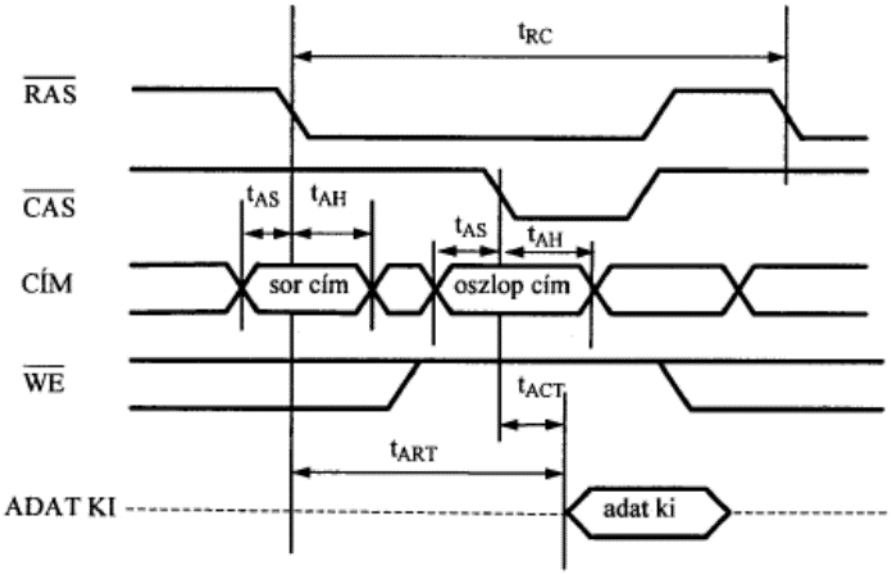
4.17 ábra Dinamikus RAM rajzjele



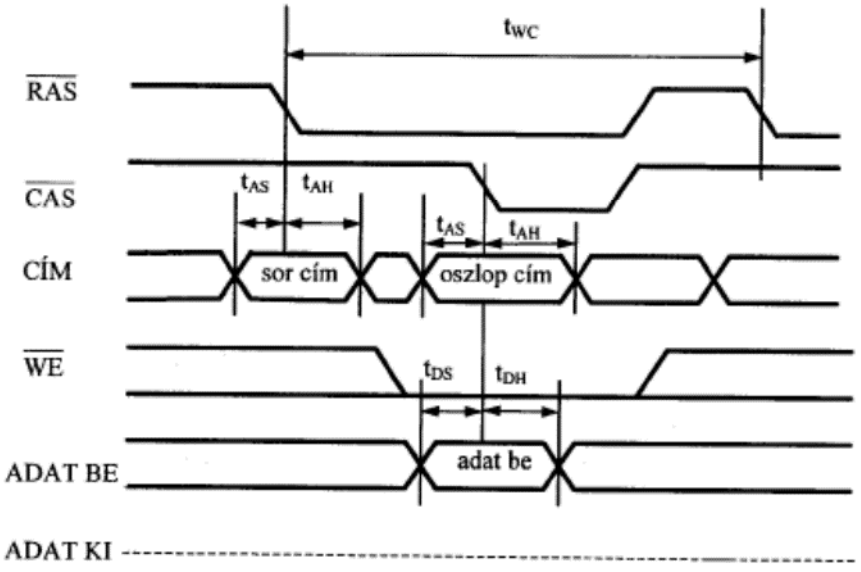
4.4. ábra Permanens memória olvasása



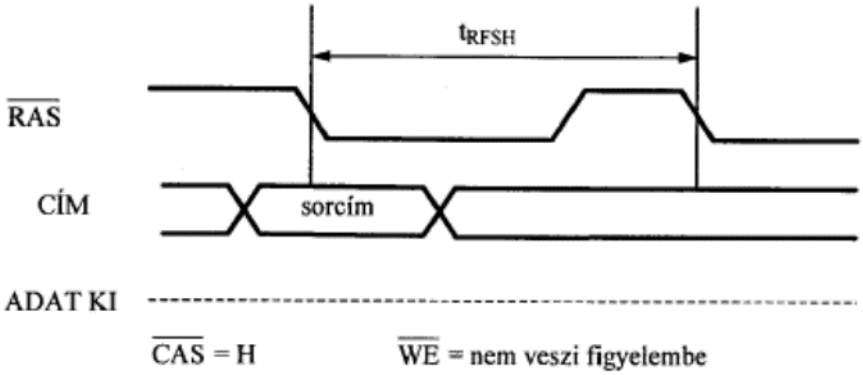
4.10. ábra Statikus RAM írási ciklusa



4.20. ábra Dinamikus RAM olvasási ciklusa



4.21. ábra Dinamikus RAM írási ciklusa



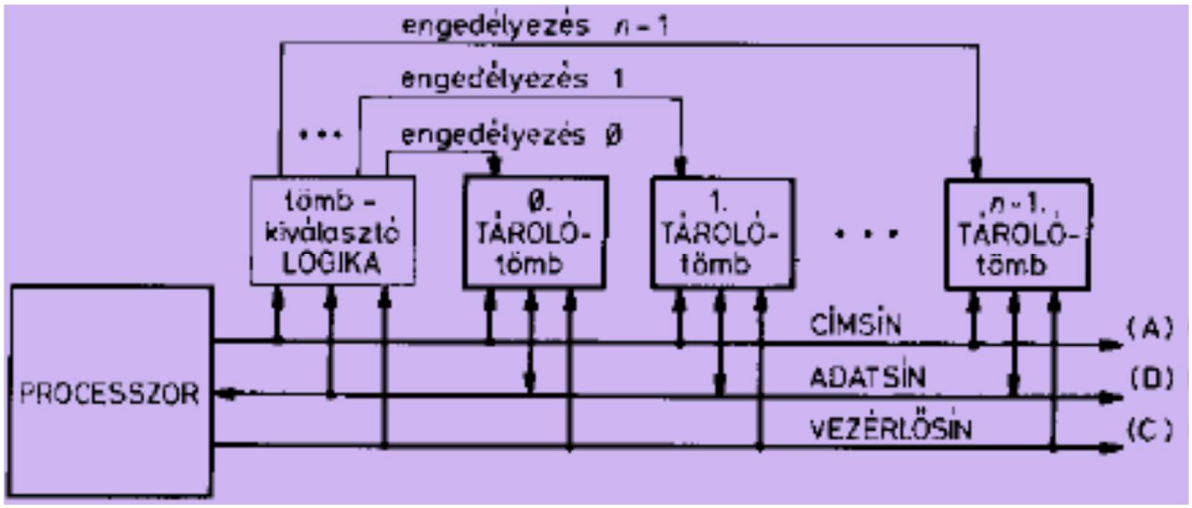
4.19. ábra Dinamikus RAM frissítési ciklusa[11]

Memóriabővítés - tovább

Tárolókezelés

Forrás: Németh Gábor, Horváth László.: Számítógép-architektúrák,
2. kiadás, Akadémiai Kiadó, 1993.

Forrás Németh G



Memóriaillesztés

1. feladat: 8085' processzorhoz illesszünk a 0000h címtől 8kB ROM-ot, a 2000h címtől pedig 2x8kB RAM-ot! Rajzoljunk memóriatérképet és készítsük el a blokkvázlatot! (közösen)
2. feladat: 8085' processzorhoz illesszünk a 0000h címtől 16kB ROM-ot, a 8000h címtől pedig 3x8kB RAM-ot. Rajzoljunk memóriatérképet és készítsük el a blokkvázlatot! (önállóan)

Függelék: Tárolókezelési módszerek

Probléma: A megcímezhető memóriatartomány mérete kisebb a szükségesnél (például 16 biten 64kB címezhető meg, de nagyobbra van szükség).

Tömbkapcsolás

Megoldás alapötlete: A szükséges méretű memóriát a processzor által megcímezhető méretű tömbökre bontjuk (n darab ilyen tömb lesz), a processzorhoz pedig egy perifériát illesztünk, amibe **out** utasítással kivisszük az általunk éppen használni kívánt tömb sorszámát. A megoldást bemutatja a 2.1. ábra.
Az átkapcsolás egyszerű, de a megoldás rugalmatlan.
Finomítás: A processzor által megcímezhető címtartomány egy része ne legyen átkapcsolható, ez alkalmas paraméterátadásra, megszakítások kezelésére, stb.

Feladat: Illesszünk 8085' processzorhoz a 0000h címre 16kB EPROM-ot, a 4000h címre 16kB RAM-ot, a 8000h címre pedig 8x 32kB RAM-ot úgy, hogy a 00h címre beírt 0-7 közötti értékkel lehessen közülük a megfelelő sorszámút engedélyezni. (Az most nem követelmény, az a 00h címrőlaz aktuális érték visszaolvasható is legyen.)

Indexelt leképzés

Az indexelt leképzés célja is a címtartomány kiterjesztése. A tárolót rögzített méretű lapokra bontja, ezek közül egyszerre néhány érhető el. A megoldást a 2.2. ábrán mutatjuk be.
A processzor által kiadott *logikai cím*et (LA: *Logical Address*) két részre bontjuk: *index* (*x* bites) és *eltolás* (*d* bites) mezők.
Az index mező *x* bitjével az indexregiszter tömb *2x* regisztere közül választunk ki egyet. Ennek a regiszternek két mezője van: a *z* bitből álló *vezérlés* mezőben jogosultságok adhatók meg, az *n* bites(*n>x*) címkiterjesztés mező pedig a kiválasztott lap kezdőcímét adja meg az operatív tárban. A lapon belüli címzésre a logikai cím eltolás mezőjének értékét használjuk. A fizikai cím (PHA: *Physical Address*) mérete így *n+d* bit.
A teljes *2n+d* méretű tárból egyszerre így is csak *2x+d* méretű részt látunk, de az indexregiszterek feltöltésével rugalmasan választhatjuk ki, hogy mely *2x* számú (*2d* méretű) lapot kívánjuk látni.
Az indexregisztereket perifériaként írhatjuk.
Itt is szükséges néhány lapot (pl. megszakításrutinok, az indexregisztereket kezelő operációs rendszer funkciók tárolására) állandóan elérhetővé tenni.

Feladat: Adott egy 8085' processzor és egy 64 elemű indexregiszter tömb. Az indexregiszterek 8 bites vezérlés és 16 bites címkiterjesztés mezőből állnak. Az indexregiszter tömb 64 perifériacímet foglal el, és az indexregiszterek: vezérlés, címkiterjesztés alsó bájt, címkiterjesztés felső bájt sorrendben írhatók. Mekkora lapokat használjunk, és így összesen legfeljebb mekkora operatív memóriánk lehet?

Probléma: a processzor által megcímezhetőnél kisebb memória áll rendelkezésünkre.

Virtuális tárkezelés

A teljes megcímezhető tárterület tartalmát háttértárolón tároljuk, ennek egy időben csak egy (kis) részét tudjuk leképezni az operatív tárolóra. A tárkezelő egység (MMU – *Memory Management Unit*) feladata, hogy a felhasználó számára láthatatlan módon a felhasználó által éppen elérni kívánt részeket a háttértárolóról az operatív tárolóba behozza. *Lapszervezés* esetén a memóriát fix méretű lapokra, *szegmensszervezés* esetén változó méretű szegmensekre bontva kezeljük.

Lapszervezésű virtuális tároló

A lapszervezésű virtuális tároló működésének kulcsfontosságú eleme a *laptábla*, amelyet (nagy mérete miatt) az operatív memóriában helyezünk el.

A *virtuális cím* (VA: Virtual Address) alkalmas az egész (háttértárolón elhelyezkedő) virtuális memória megcímzésére. A virtuális cím két mezőből áll: *lapsorszám* (PN: Page Number) és lapon belüli *eltolás* (D: Displacement).

A laptábla operatív memóriabeli kezdőcímét a *laptáblamutató* (PTP: Page Table Pointer) tartalmazza, amihez hozzáadva egy adott virtuális cím lapsorszám részét a *lapcímmutató* (PAP: Page Address Pointer) értékét nyerjük, ami egy laptábla bejegyzés operatív tárbeli kezdőcíme. Egy ilyen bejegyzés két részből áll: *lapkezdőcím* (PA: Page Address) és *vezérlés* mezők. A vezérlés mezőben a laphoz különféle jogosultságokat adhatunk meg, és ennek a mezőnek a megfelelő bitjéből derül ki az is, hogy a kívánt lap éppen bent van-e az operatív memóriában:

- Amennyiben igen, akkor a lapkezdőcím értékéhez a virtuális cím eltolás mezőjének értékét hozzáadva nyerjük azt *fizikai cím*et (PHA: Physical Address) ami megadja a kívántinformáció helyét az operatív memóriában.
- Ha a lap nincs bent, akkor a MMU feladata, hogy az operációs rendszerrel behozassa azt a háttértárolóról. Ehhez esetleg szükség lehet valamely lapnak a háttértárolóra való kiírására(aminek a helyére behozzuk).

Azt az esetet, amikor az elérni kívánt lap nem található az operatív memóriában, *laphibá*nak (page fault) nevezzük. A laphiba NEM kezelhető megszakítással, ugyanis megszakításnál a végrehajtás alatt álló utasításnak be kell fejeződnie, itt azonban erre nincs lehetőség. Ezt az esetet, amikor laphiba miatt az utasítás végrehajtása kényszerűen félbeszakad (felfüggesztődik), *kivétel*nek (exception) hívjuk. A processzornak képesnek kell lennie a félbeszakadt utasítás végrehajtását (megfelelő állapotmentéssel) később, a kívánt lap behozása után folytatni.

Annak a döntésnek a támogatására, hogy egy lapot melyik lap helyére hozzunk be, több információt is felhasználunk. Minden lapról nyilvántartjuk, hogy milyen régen használtuk utoljára – erre elvileg egy számlálót használhatnánk, de egyszerűségi megfontolásból inkább csak egy bitet (accessed) használunk, amit periodikusan törlünk, valamint azt is, hogy módosult-e a háttértárolóról való behozása óta (dirty bit). (Ezek a bitek is a vezérlés mezőben helyezkednek el.) Lehetőség szerint egy nem accessed és nem dirty lap helyére hozunk be másikat, amit így nem kell kiírnunk.

Amennyiben a laphibák túl gyakoriak (például az összes memóriahivatkozások több, mint 1%-a laphibát okoz), akkor a rendszer működése során az idő túl nagy részében fog a laphibák kezelésével foglalkozni, ezt vergődésnek (trashing) nevezzük [6]. Ahhoz, hogy egy task kielégítő sebességgel fusson, a végrehajtáshoz éppen szükséges lapjainak (working set: [8]) nagy valószínűséggel az operatív memóriában kell lenniük.

A lapméret megválasztásánál figyelembe vesszük, hogy a háttértár elérési ideje a néhány ms nagyságrendjébe esik, az átvitel viszont meglehetősen gyors, így ma kb. 4kB a szokásos lapméret.

Mivel a laptáblákat az operatív memóriában tároljuk, a virtuális címhez a fizikai cím meghatározása (címfordítás) miatt minden memóriaművelet két memória hozzáférést igényelne, ami felére csökkentené az operatív tár elérésének sebességét. Ennek kiküszöbölésére használják a *Translation Lookaside Buffer*t (TLB) amelyet az MMU-ban (ami ma tipikusan a CPU része) egy néhány (néhányszor tíz) bejegyzést tárolni képes tartalom szerint címezhető memóriában (CAM Content Addressable Memory) helyeznek el: ez tárolja a néhány utoljára kiszámított PN – PA párost, és PN alapján villámgyorsan megadja PA-t.

Szegmensszervezésű virtuális tároló

A változó méretű szegmensek használata jobban igazodik egy feladat szerkezetéhez (függvények, adatszerkezetek egymástól való szétválasztása), viszont a kezelésük lényegesen bonyolultabb, ezért csak nagy gépeken szokták használni. Mi bővebben nem foglalkozunk a témával, érdeklődők K2-ben elolvashatják.

Probléma: az operatív tároló nem elég gyors (a processzorhoz képest).

Cache tároló

Ez a probléma az operatív tároló méretének növekedésével szükségszerűen előáll, hiszen a tároló kapacitásának négyzetgyökével arányos az elérési ideje. (Memóriáknál láttuk, hogy miért.)
A nagy és lassú operatív tár átlagos elérési idejét jelentősen csökkenthetjük azzal, hogy a processzor és az operatív memória közé egy kicsi és gyors *rejtett tár*at (cache) iktatunk be. A rejtett tár az operatív tár bizonyos részeinek másolatát tartalmazza, működése a felhasználó számára átlátszó, azaz a felhasználó közvetlenül nem érzékeli a létét, csupán az operatív tárat látja gyorsabbnak: amikor a keresett információ a cache-ben található, lényegesen rövidebb az elérési idő, amikor nem, akkor meg kell várni az operatív tárat. Vegyük észre, hogy a virtuális tárkezelés és a cache alkalmazása a tárhierarchia más szintjén ugyan, de ugyanazt szolgálja: egy nagyobb méretű, lassabb tárat (majdnem) egy kisebb méretű, gyorsabb tár sebességével szeretnénk elérni!
Mivel azonban az elérési idők nagyságrendi eltérése különböző (cache és operatív memória mindegyike félvezető, operatív memóriával szemben a háttértár mechanikus eszköz), így míg a virtuális memóriánál 1% körül van az elfogadható laphiba arány, a cache esetében ez (*cache miss* rate – a találat pedig: *cache hit*) 10% körül van, és a szokásos blokkméret is jóval kisebb, 16 vagy 32 bájt körül van. (A cache blokkjait gyakran úgy hívják, hogy *cache line*.) A cache esetén a gyakorlatban több szintű hierarchiát is használnak. (Lásd DEC Alpha 21164-nél 3 szint: [10])
Több szempontból is értelmes külön utasítás és adat cache-t használni (2.5. ábra).
A külvilág szempontjából a cache vagy az operatív tár vagy processzor részének tekinthető, mindkét megoldásnak megvannak az előnyei és hátrányai (2.6. ábra).
A cache vezérlésnek többféle módja van (és ezeknek más-más előnye és hátránya van, ami alapján a gyakorlatban adott célra egyiket vagy másikat választhatjuk), most mi azzal foglalkozunk, amelyik tartalom szerint címezhető tárolót (CAM) alkalmaz. Működésének egyik lényeges lépését mutatja be a 2.7. ábra. Itt látható, hogy egy cache blokk memóriabeli címét (BAM) a CAM egyszerre összehasonlítja az összes benne tárolt bejegyzéssel és találat esetén azonnal megadja, hogy az adott blokk hol helyezkedik el a cache-ben (BAC). Az ábra nem tünteti fel, hogy mi történik, ha nem volt egyezés. Ilyenkor célszerű a legrégebben használt blokk helyére betölteni az újat. Ennek egy jó megvalósítása a CAM shiftregiszterként való használatán alapul: egyezés esetén az egyező sort kiveszik, a fölötte levő összes sort eggyel lefele léptetik és az egyező sort a legfelső sor helyére teszik. Ha nincs egyezés, akkor az összes sor eggyel lejjebb lép, és a legfelső sorba kerül az éppen a cache-be behozott blokkot leíró BAM-BAC páros. Így a legrégebben használt (LRU: Least Recently Used) blokk helyére hoztuk be az újat.
Megjegyzés: a 2.7. ábra úgy tiszta és érthető, ha a VA helyére PHA kerül: az operatív tárbeli fizikai cím!
Az egy másik kérdés, hogy a virtuális tárkezeléssel való együttes használat érdekében a címfordítás kiküszöbölésére VA is tárolható. Ekkor azonban oda kell figyelni, hogy mi történik, ha az adott cache blokkot tartalmazó lap (vagy szegmens) kikerül az operatív memóriából!
A fenti megoldást *teljesen asszociatív* (fully associative) leképzésnek nevezzük. A másik véglet a *direkt leképzés* (direct mapping), amikor az operatív memóriát a cache memória méretével megegyező méretű részekre bontjuk, és ezek cache line méretű blokkjainak fixen adódó helye van a cache-ben. Ilyenkor az operatív memória két különböző cache memória méretű részének azonos sorszámú cache line méretű blokkja ugyanarra a cache line-ra képeződik le, így az egyiknek a cache-be történő behozatala szükségszerűen kiüti a másikat! Ilyenkor csak azt kell nyilvántartanunk, hogy az adott cache line az operatív memóriának éppen melyik cache méretű részéből származik. A megoldás egyszerűsége miatt olcsóbb, de kevésbé hatékony (kisebb a cache hit rate).
A fenti megoldás javítása az *N-utas asszociatív* (N-way set associative) leképzés: ilyenkor a direkt leképzés hátrányait úgy csökkentjük, hogy N darab cache memóriát használunk, így az egymást kiütő blokkokból legfeljebb N példány még egyidejűleg bent lehet az N darab cache memória valamelyikében (a neki megfelelő rögzített helyen). Ekkor minden egyes cache sorhoz kell egy N méretű tartalom szerint címezhető memória, aminek a segítségével mind az N darab cache memóriára nézve egyszerre

meg tudjuk mondani, hogy egy adott operatív tárbeli címen található blokk benne van-e. Tipikusan 2 vagy 4 utas megoldást szoktak használni.
További kérdés, hogy mi történjen íráskor (write policy). Erre több megoldást is használnak:
• write through: az írási művelet mind a cache-ben, mind az operatív tárban megtörténik
• write back: íráskor a cache blokk dirty jelölést kap, és később kerül csak átírásra az operatív tárban – legkésőbb a cache blokk felülírásakor meg kell történnie!

Függelék: Memória szervezés RAM-okkal

Forrás: - haszn --- KÖNYV különösen --MEMORY -- Principles of Computer Architecture Murdocca Heurig - 1999.pdf

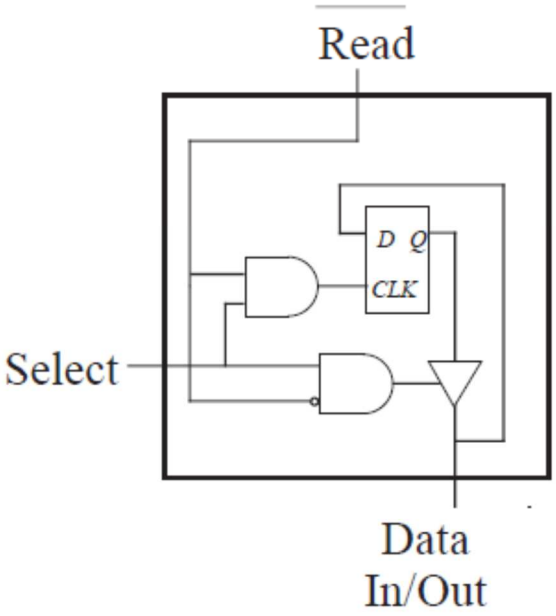


Figure 7-2 Functional behavior of a RAM cell.

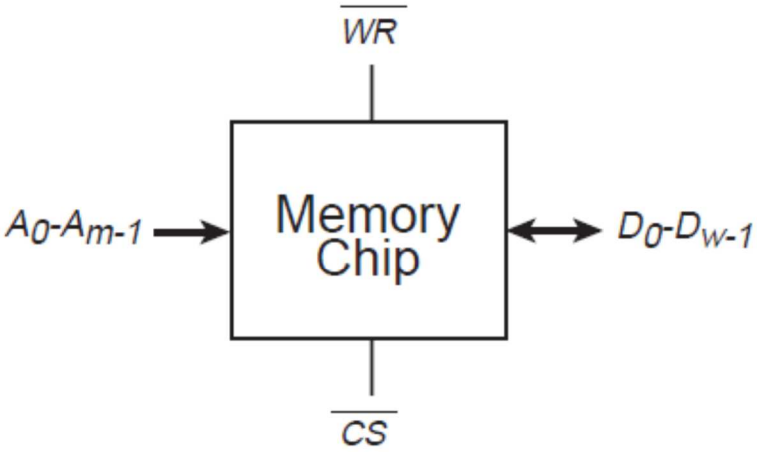


Figure 7-3 Simplified RAM chip pinout

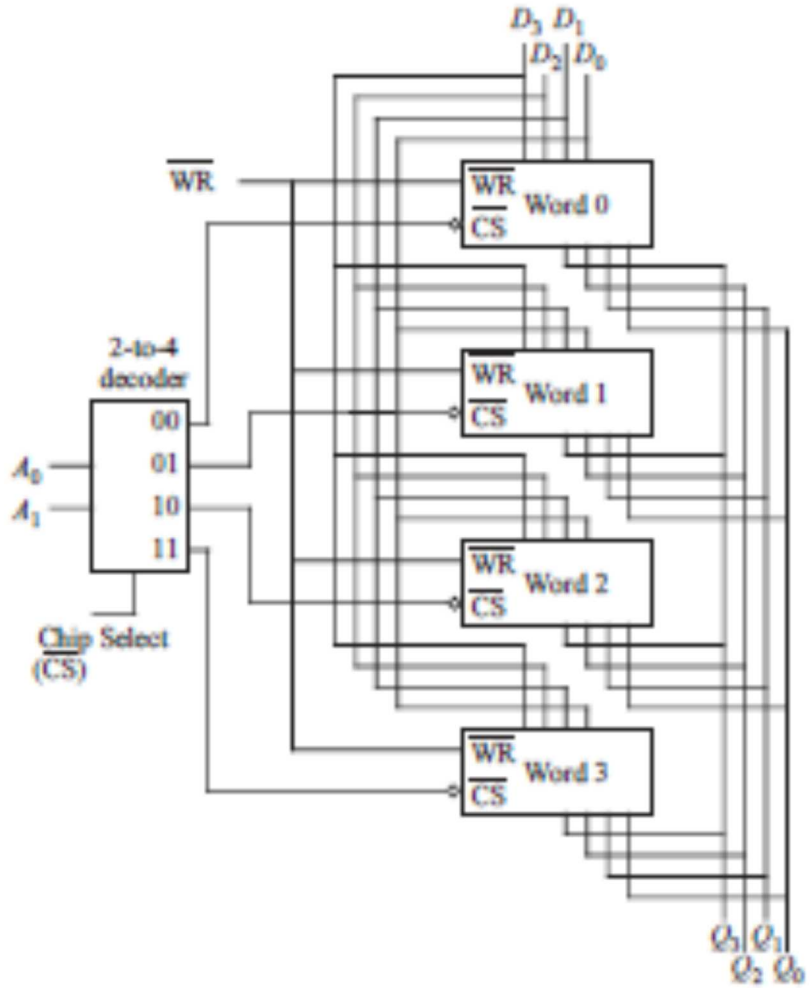


Figure 7-4 A four-word memory with four bits per word in a 2D organization.

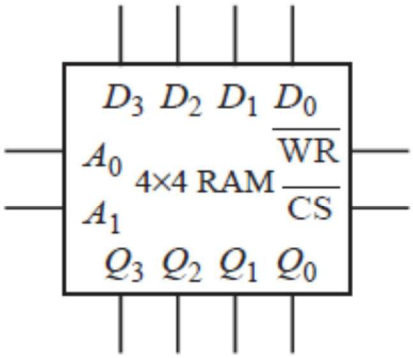


Figure 7-5 A simplified version of the four-word by four-bit RAM.

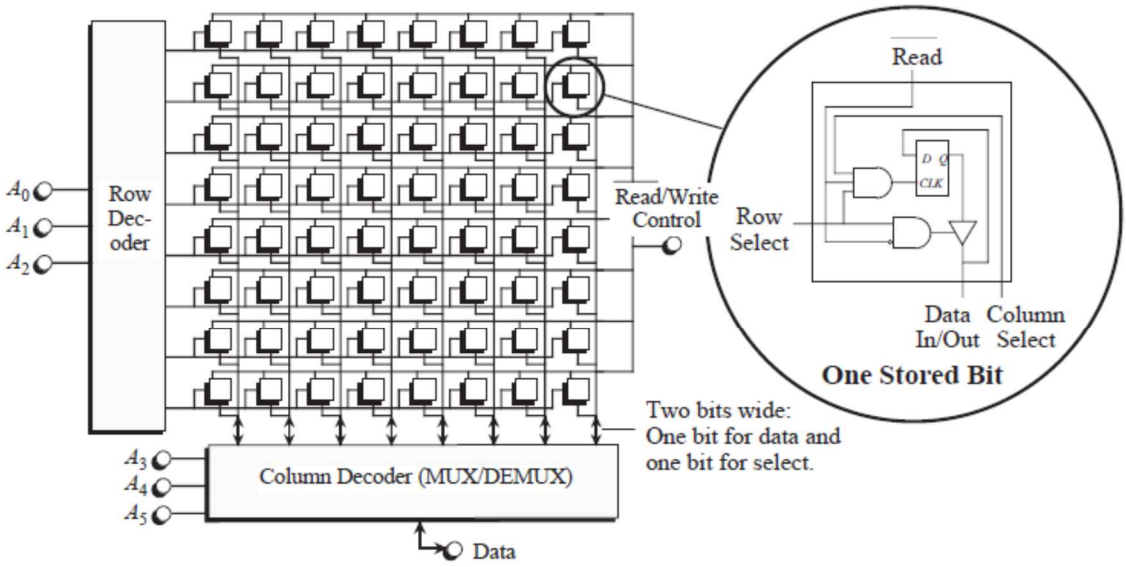


Figure 7-6 2-1/2D organization of a 64-word by one-bit RAM.

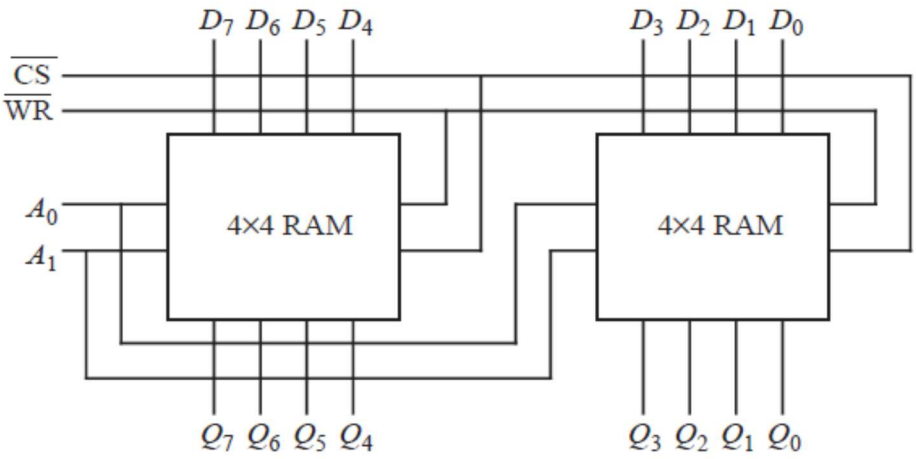


Figure 7-7 Two four-word by four-bit RAMs are used in creating a four-word by eight-bit RAM.

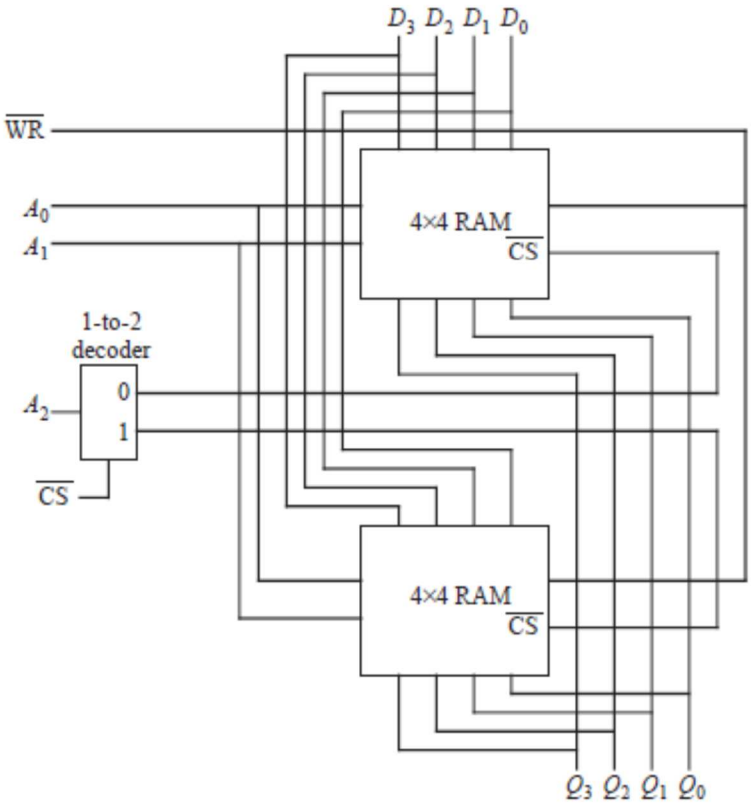


Figure 7-8 Two four-word by four-bit RAMs are used in creating an eight-word by four-bit RAM.

Függelék: Memória technológiák

Forrás Dr. lencse Gábor BME előadása

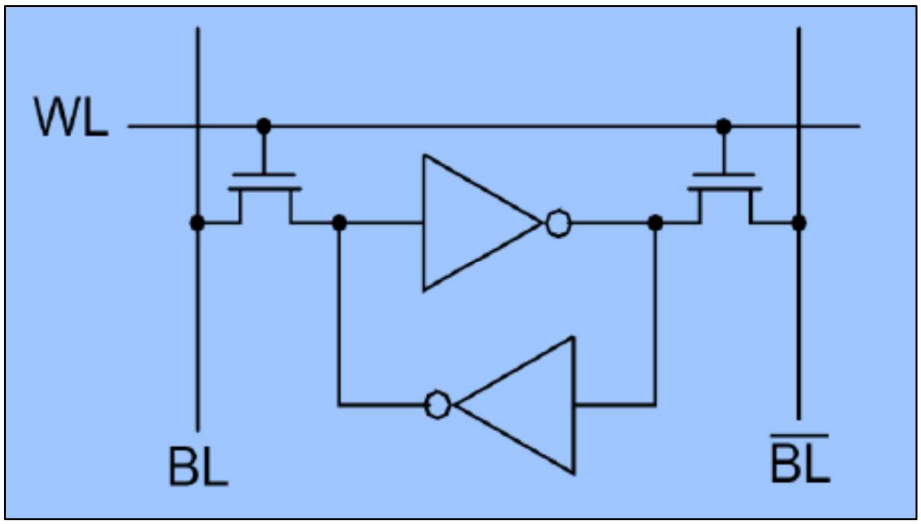
Tartalom

SRAM ÉS DRAM FELÉPÍTÉSE ÉS ENNEK KÖVETKEZMÉNYEI

SRAM felépítése és működése

Az SRAM egy cellájának felépítése – 1

- A statikus RAM egy cellája (ami egy bit tárolására képes) – logikai áramkörként tekintve – két keresztbe kötött inverterből áll:
  - az egyik inverter bemenete a bitet,
  - a másiké annak inverzét tárolja.
- Ez az áramkör mindaddig képes megőrizni a bit értékét, amíg tápfeszültséggel ellátjuk.
- A cella kiolvasását és írását további két tranzisztorral valósítják meg, ezeket *hozzáférési* (access) tranzisztoroknak hívjuk.

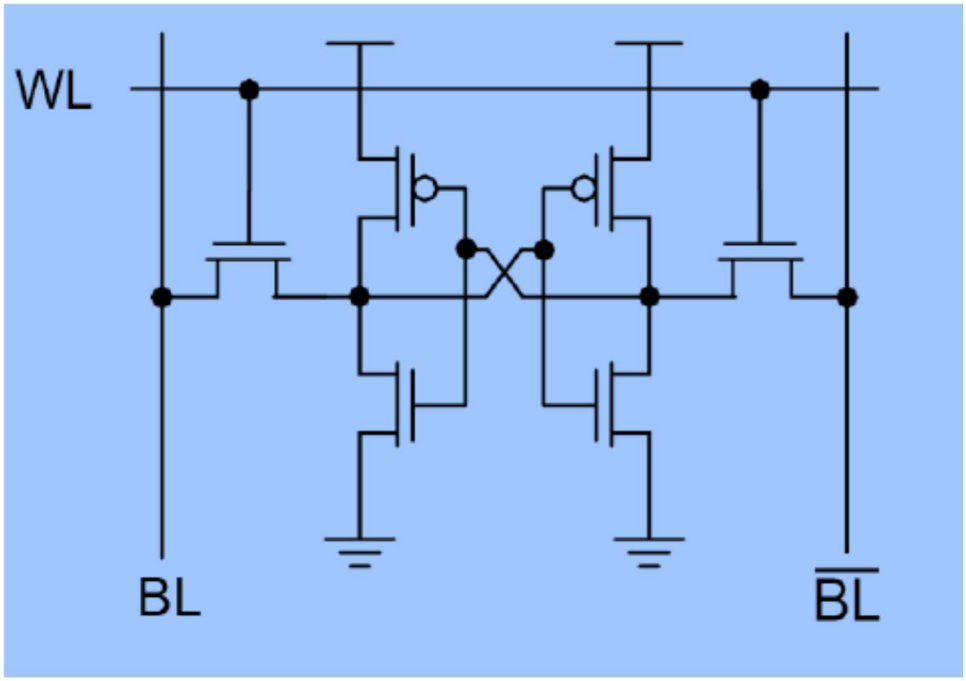


6

Az SRAM egy cellájának felépítése – 2

- Mivel az inverterek megvalósításához is 2-2 tranzisztor szükséges, így cellánként összesen 6 tranzisztort használnak fel.
- Az ábrán WL (**W**ord **L**ine) a *szóvezeték*et, BL (**B**it **L**ine) a *bitvezeték*et jelenti.

7



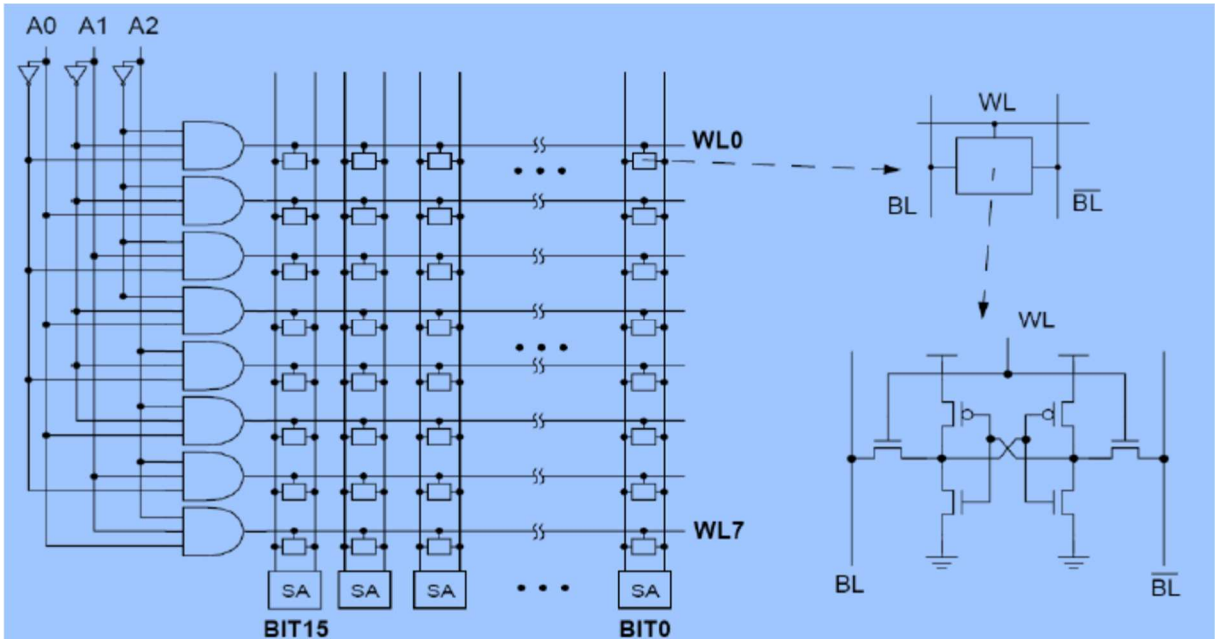
Az SRAM felépítése – 1

- A cellákat mátrixba rendezik. Az egyszerűség kedvéért most a *szószervezésű* megoldással foglalkozunk; legyen a *címbitek* száma n, így a *szavak száma* $k=2n$, a *szavak szélessége* pedig m bit, ekkor a mátrix k sorból és m oszlopból áll.
- A következő dián egyszerű példaként egy nagyon kis méretű memóriát látunk a következő értékekkel: címbitek száma $n=3$, dekódolt címvezetékek (az ábrán WL0-WL7 szóvezetékek) száma $k=8$, a szószélesség $m=16$.

- A bitmezőtől balra egy inverterekkel és ÉS kapukkal megvalósított 3/8-as dekódert látunk; a cellák tartalmát pedig a jobb oldalon kinagyítva és kirészletezve találjuk.

Az SRAM felépítése – 2

- Az *érzékelő erősítők*et SA-val (sense amplifier) jelöltük.
- A rajzon nem szerepelnek, de minden bitvezetékhez tartoznak még a *előfeszítő* (precharge) áramkörök is.



9

Az SRAM olvasása – 1

1. Az előfeszítő áramkörök a ponált és negált bitvezetékeket egyaránt logikai 1 szintre húzzák fel, majd a bitvezetékekről lekapcsolódnak. A bitvezetékek kapacitása miatt a bitvezetékek feszültségszintje megmarad. (Ez azért hasznos, mert a celláknak könnyebb/gyorsabb egy bitvezetéket 0-ra lehúzni, mint 1-re felhúzni.)
2. Ha az *i*. memóriaszót szeretnénk kiolvasni ($0<=i<2n$), akkor a WLi szóvezetékre logikai 1 értéket adunk. (Ezt az ábrán a bitmezőtől balra található dekóderrel valósítjuk meg.)

Az SRAM olvasása – 2

3. Az *i*. memóriaszó összes bitjének hozzáférési tranzisztorain át a bitek értéke megjelenik a bitvezetékeken:
   - amelyik bit értéke Q=1, ott BL értéke 1 marad, /BL-t pedig 0-ra húzza le a bit inverze (/Q);
   - a Q=0 értékű bitnél pedig a helyzet éppen fordított.

4. Az érzékelő erősítők érzékelik BL és /BL különbségét, és a kimenetükön rendelkezésünkre áll az *i*. memóriaszó bitjeinek értéke. (Minél érzékenyebbek, annál gyorsabb a kiolvasás.)
- Az olvasás a kiolvasott memóriaszó értékét nem változtatja meg.
- A bitek felismerését segíti (és gyorsítja), hogy nem abszolút jelszinteket, hanem BL és /BL különbségét kell felismerni.

Az SRAM írása – 1

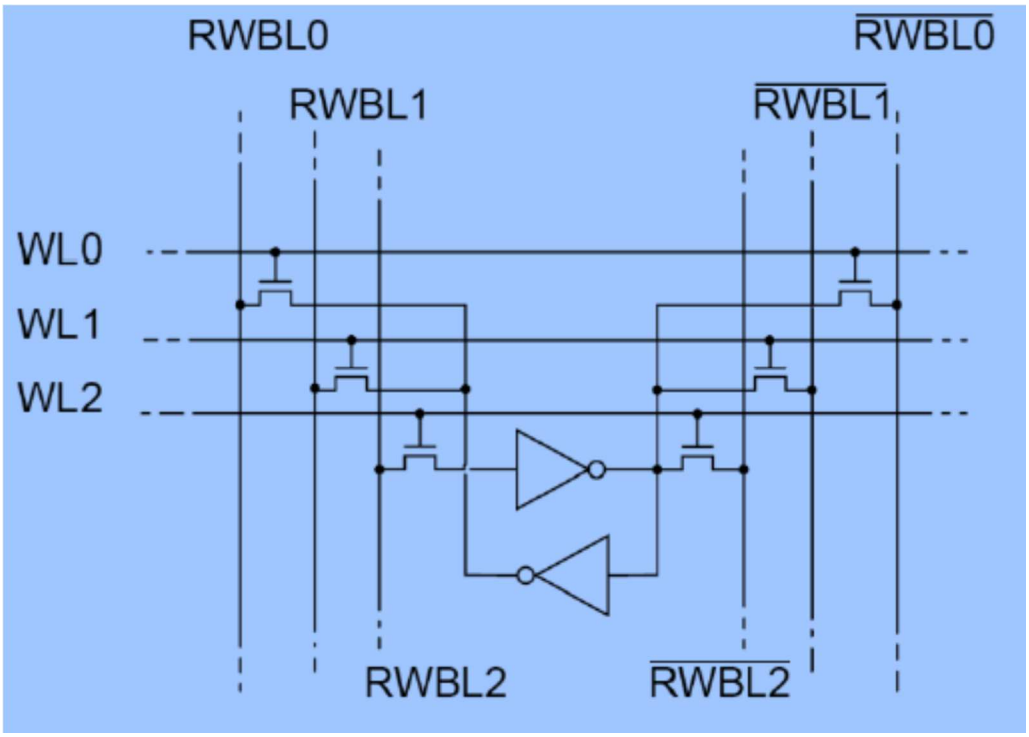
1. A ponált (BL) és negált (/BL) bitvezetékekre ráadjuk a beírandó bitek logikai értékének megfelelő feszültséget:
    - ha a *j*. bit értéke 1, akkor BLj=1, /BLj=0;
    - ha a *j*. bit értéke 0, akkor BLj=0, /BLj=1.
2. Ha az *i*. memóriaszót szeretnénk írni ($0<=i<2n$), akkor a WLi szóvezetékre logikai 1 értéket adunk.
3. Az *i*. memóriaszó összes bitjének értéke a hozzáférési tranzisztorain keresztül felveszi a bitvezetékek által rákényszerített értéket, mivel a bitvezetékek meghajtó tranzisztorai erősebbek, mint a cellák tranzisztorai.

Az SRAM írása – 2

- A megfelelő működéshez természetesen a tranzisztorok megfelelő méretezésére van szükség.
- Az írásnál szintén fontos szerepe van a két bitvezetéknek: mindkét inverter bemenetét megfelelő szintre beállítva billentjük át a bistabil multivibrátort.

Több portos SRAM megvalósítása

- Amennyiben több portos memóriát szeretnénk (ha egyidejűleg több eszköznek hozzá kell férnie), akkor a hozzáférési tranzisztorok száma nő: portonként egy pár tranzisztorra van szükség.
    - Tehát például 3 db írásra és olvasásra egyaránt használható port esetén 4+3x2=10 tranzisztorra van szükség, és egy cella a így néz ki:



Az SRAM makrostruktúrája

- Az SRAM struktúrájának bemutatásakor szószervezést használtunk. Ez komoly probléma, hiszen például egy kifejezetten kis kapacitásúnak számító 8kB-os SRAM esetén a címbitek száma n=13, a szóvezetékek (más néven dekódolt címvezetékek) száma k=213=8192, míg a szószélesség m=8; tehát a bitmező 8192 sorból és 8 oszlopból áll, ami meglehetősen aránytalan.
- Erre a problémára viszonylag kis méret esetén elégséges megoldás lehet a *bitszervezés*, ahol a címbitek felét sorcímként, a másik felét oszlopcímként használjuk és a dekódolt címvezetékek metszéspontjában egy-egy ÉS kapuval engedélyezzük a cellát, de mivel ez bitenként egy-egy ÉS kaput igényelne, ezért a gyakorlatban a *módosított bitszervezés*t szokták használni, ahol *sor dekóder*t és *oszlop multiplexer*t alkalmaznak.
- Ezt a megoldást a DRAM struktúrájánál fogjuk bemutatni.
- Bizonyos méret felett azonban nem egyetlen bitmezővel, hanem hierarchikus felépítésű struktúrával valósítjákmeg az SRAM-ot (és a DRAM-ot is).

Az SRAM interfész lehetőségei

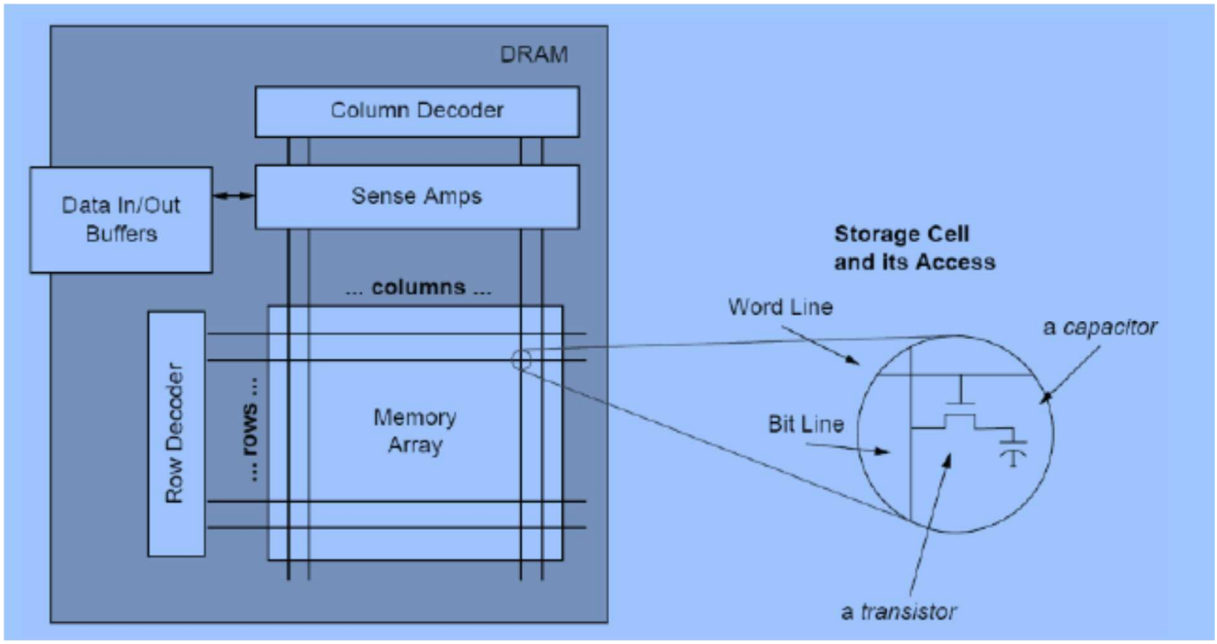
- Szinkronizációs szempontból az SRAM-nak kétféle interfésze lehet:
  - Aszinkron interfész
  - Szinkron interfész
- Aszinkron interfész
  - Az első gyakorlaton bemutatott illesztési felület aszinkron: órajeltől függetlenül, a /CE, /OE, /WE jelekkel vezérelve, a lehető legrövidebb időn belül teljesíti a kapott parancsot.
  - Amennyiben az SRAM-ot mikroprocesszorhoz (vagy mikrokontrollerhez) illesztve operatív tárként használjuk, ez az interfész a szokásos megoldás.

- Szinkron interfész
    - A szinkron interfésznél a korábban bemutatott illesztési felület kiegészül még egy (esetleg két) órajel bemenettel (CLK).
    - Ez a megoldás lehetővé teszi, hogy az átvitel *pipelined burst mode*ban történjen: például olvasáskor nemcsak a megadott címről, hanem a rákövetkező bizonyos számú címről is megtörténik a kiolvasás, és az egymást követő memóriarekeszekből való olvasás és a CPU felé való átvitel egyes fázisai egymással átlapolódnak.
    - Ezt a megoldást cache esetén szokták használni, hiszen így egy teljes cache sort igen hatékonyan át lehet vinni.

DRAM felépítése és működése

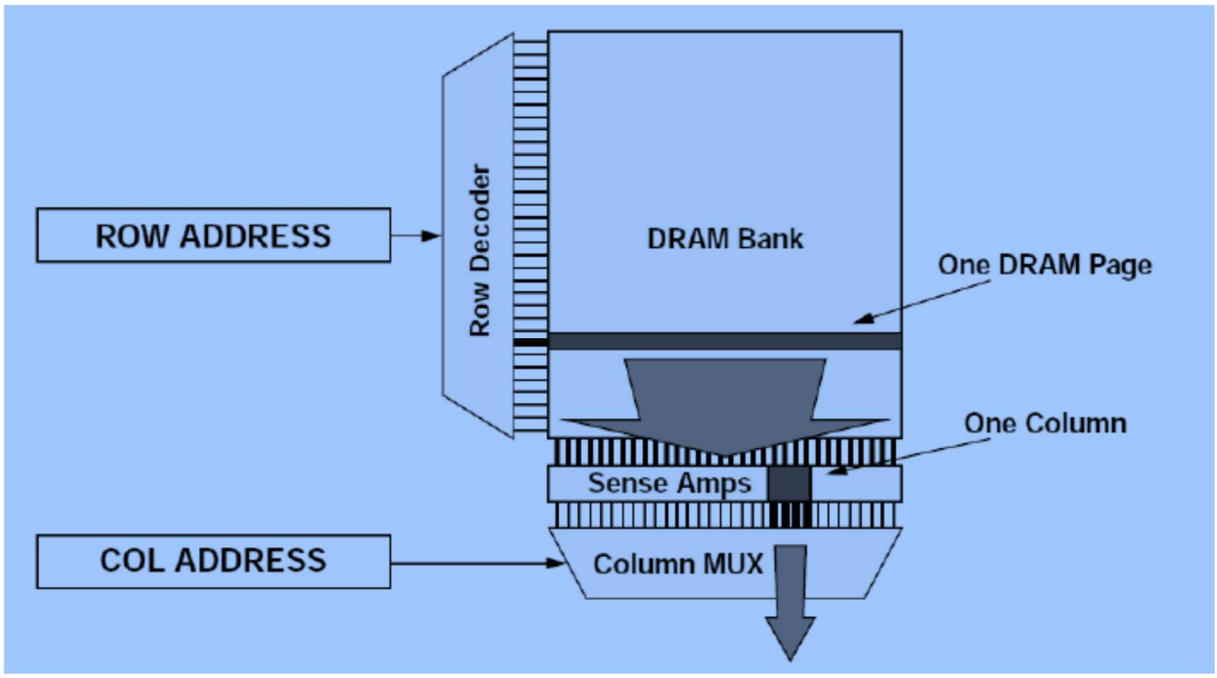
A DRAM felépítése

- A dinamikus RAM egy cellája (ami egy bit tárolására képes) logikailag egy tranzisztorból és egy kondenzátorból áll.
- A cellákat mátrixba rendezik; egy adott bitet a sor- és oszlopszáma azonosít.



20

- A cellákból egyszerre mindig egy sort lehet kezelni, ezt a sorcím alapján a sor dekóder választja ki; a sor bitjei közül pedig szükség esetén az oszlop multiplexerrel választunk. (Ezt módosított bit szervezésnek hívják.)



A DRAM olvasása

1. Az SRAM-hoz hasonlóan a bitvezetékeket itt is előfeszítjük, de az SRAM-mal ellentétben nem logikai 1- re, hanem csak a logikai 0 és 1 szintje közé "félútra".
2. A sorcím által kijelölt dekódolt sorvezetékre (a bevezető ábrán Word Line néven szerepel) logikai 1-et adunk, ennek hatására a kiválasztott sor tranzisztorai összekapcsolják a kondenzátorokat a hozzájuk tartozó bitvezetékkel, így a kondenzátor töltése módosítja a bitvezeték feszültségszintjét.

Az olvasás az SRAM-mal ellentétben destruktív!

3. Az érzékelő erősítők érzékelik a feszültségnek az előfeszített szinthez képest való eltérését, majd előállítják a korrekt logikai szintet, ami a kondenzátorokba vissza is íródik.
    - Így regeneráljuk a tönkretett tartalmat.
    - Amíg a /RAS jel alacsony szintje fennáll, a kiválasztott sor *aktív* állapota fennmarad. Angolul úgy fejezik ki, hogy "the page is open".
4. Az oszlop multiplexer az oszlopcím alapján kiválasztja a kért bitet és az megjelenik a DRAM kimenetén.
5. A /RAS jel felfutó élének hatására a sor aktív állapota megszűnik.

A DRAM adatszélessége

- A kívánt adatszélesség elérésére többféle módunk van.
    - Lehet tokonként csak egy bitet kezelni és egy memóriamodulra a kívánt számú DRAM tokot rátenni.
    - Lehet DRAM tokonként több bitet is kezelni;
        - akár úgy, hogy egymással párhuzamosan működtetünk több bitmezőt,
        - de úgy is, hogy egy oszlopcím hatására egymás mellett több bitet (pl. 4-et) választunk ki.

A DRAM írása

1. A sorcímmel kiválasztott sor aktiválása ugyanúgy történik, mint olvasás esetén (lásd ott: 1-3. lépések).
2. Az oszlopcímmel kiválasztott bit értékét az érzékelő erősítőre az adatbemeneten kapott értéknek megfelelően "rákényszerítjük".
3. Az egész sor visszaíródik (frissül), benne az oszlopcímmel kiválasztott bit az új értéket veszi fel.
4. A /RAS jel felfutó élének hatására a sor aktív állapota megszűnik.

A DRAM frissítése – klasszikus

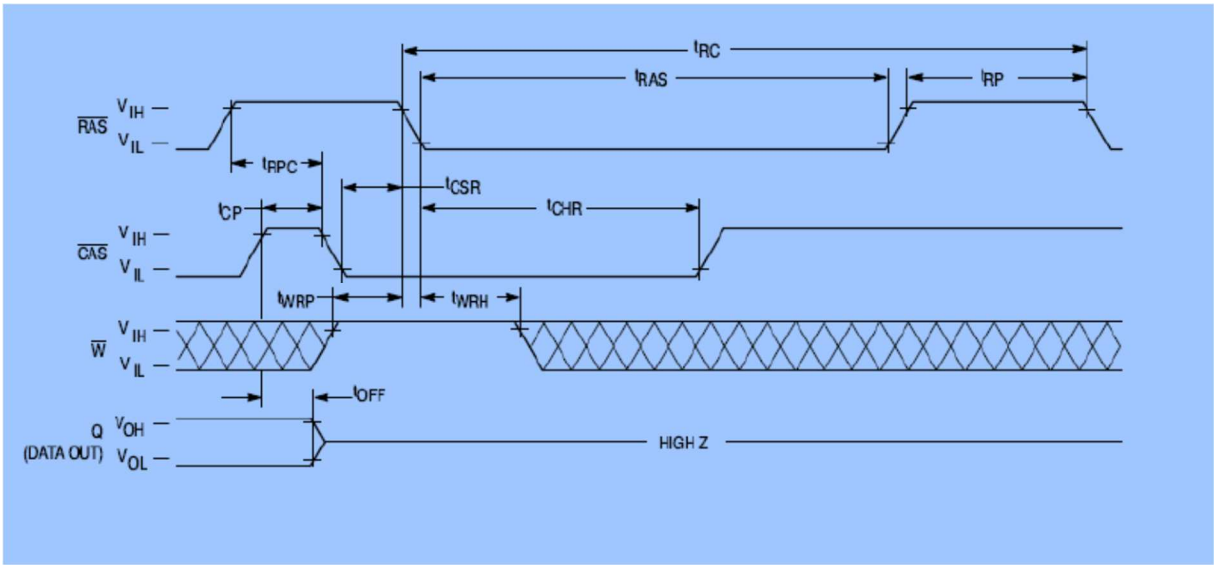
- Az eredeti megoldás szerint (amikor a DRAM lényegében semmi támogatást sem ad a frissítéshez) a frissítés az olvasás 1-3. és 5. lépésével történik.
- Az olvasáshoz képest a 4. lépés azért marad el, mert a /RAS=0 tartama alatt nincs /CAS=0.
- Ehhez a megoldáshoz szükséges, hogy a DRAM kívülről megkapja a frissítendő sor címét.
    - A címet az egyszerű, 8 bites gépek idején tipikusan a CPU-nak kellett előállítania; ehhez egy számlálót használt, amiben nyilvántartotta, hogy éppen melyik sor frissítése következik. Adott időközönként egy megszakítási rutin elvégezte a következő esedékes sor frissítését.
    - Később a frissítés memória vezérlő feladata lett.

A DRAM frissítése – korszerű

- DRAM chipeket kiegészítették egy számlálóval, ami nyilvántartja, hogy melyik sort kell frissíteni, és a frissítés protokollját is módosították:
    - Amennyiben a /CAS jel szintje 0 lesz még a /RAS jel 0 szintje előtt, majd ezt követi a /RAS=0, akkor ez (a korábban illegális eset) azt jelenti, hogy a kapott címet figyelmen kívül hagyva, a DRAM-nak a belső számlálója szerinti sort kell frissítenie.
    - Ilyenkor természetesen növeli is a számláló értékét.

Ezt *CBR frissítés*nek (CAS-before-RAS refresh) nevezik.

- Az alábbi ábra egy katalógusból származik, pusztán a CBR frissítés illusztrációjának szánjuk, az egyes időzítésekkel most nem foglalkozunk.



SRAM és DRAM összehasonlítása

- Költség szempontjából
  - Mivel egy SRAM cella 6 tranzisztorból áll, egy DRAM cella pedig egy tranzisztorból és egy kondenzátorból, ráadásul az SRAM-nál bitenként két bitvezeték van, a DRAM-nál pedig csak egy, ezért adott méretű lapkán DRAM esetén több (6-8x annyi) bit fér el, következésképpen olcsóbb.
  - Szintén az olcsóságot segíti elő, hogy DRAM-nál a címet két részletben viszik át, ami csökkenti a tok lábszámát, de a DRAM fent megismert működése miatt további lassulást nem okoz.

- Sebesség szempontjából
  - Működési sebességet tekintve azonban az SRAM nyer, ennek oka szintén a belső felépítésében keresendő: a bistabil multivibrátor állapota gyorsabban érzékelhető (néhány ns), mint a kondenzátor csekélyke töltésének állapota (néhányszor 10ns).
- Felhasználás egyszerűsége szempontjából
  - Az illesztés is SRAM esetén kényelmesebb: a cím egyben kezelhető; ráadásul frissítésre sincs szükség.

- Célszerű felhasználási terület szempontjából
  - A fentiekből következően számítógépek operatív memóriája céljára DRAM-ot használnak.
  - Az SRAM-ot cache céljára és (cache nélküli) kis rendszerek operatív memóriájaként is alkalmazzák.

- A CPU és az SRAM azonos technológiával gyártható, ezért azokat könnyű egy lapkára integrálni (belső cache), míg a DRAM gyártási technológiája eltérő.

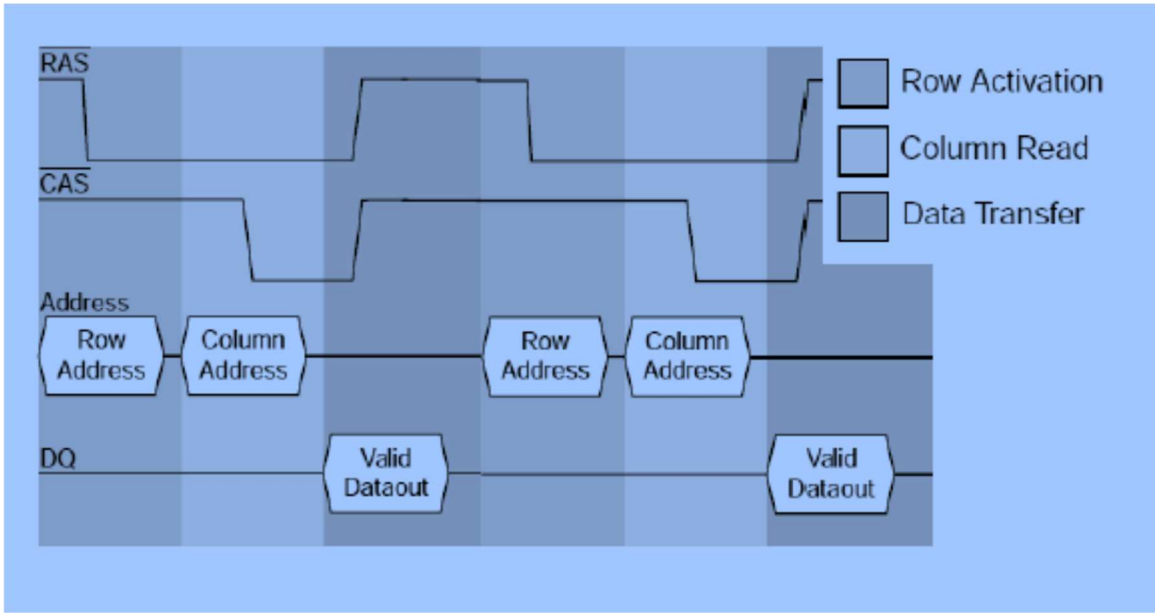
- Elvi korlát az elérési időre
  - Felmerül a kérdés, hogy ha az SRAM és DRAM közötti sebességkülönbség oka a technológiában keresendő, akkor miért van értelme több szintű cache alkalmazásának - különösen akkor, ha ezek mindegyikét a CPU-val azonos lapkára integrálják.
  - A válasz nagyon egyszerű: a jelterjedési késleltetés miatt. A biteket tároló cellákat (akár egy, akár több bitmezőt használunk is) lényegében mindenképpen két dimenzióban kell elrendeznünk, így a befoglaló négyzet oldala a bitszám (azaz a memóriakapacitás) négyzetgyökével arányos: így ha például a cache kapacitását két nagyságrenddel növeljük, akkor a jelterjedési késleltetés egy nagyságrenddel nő.

A DRAM FEJLŐDÉSE

Aszinkron DRAM típusok

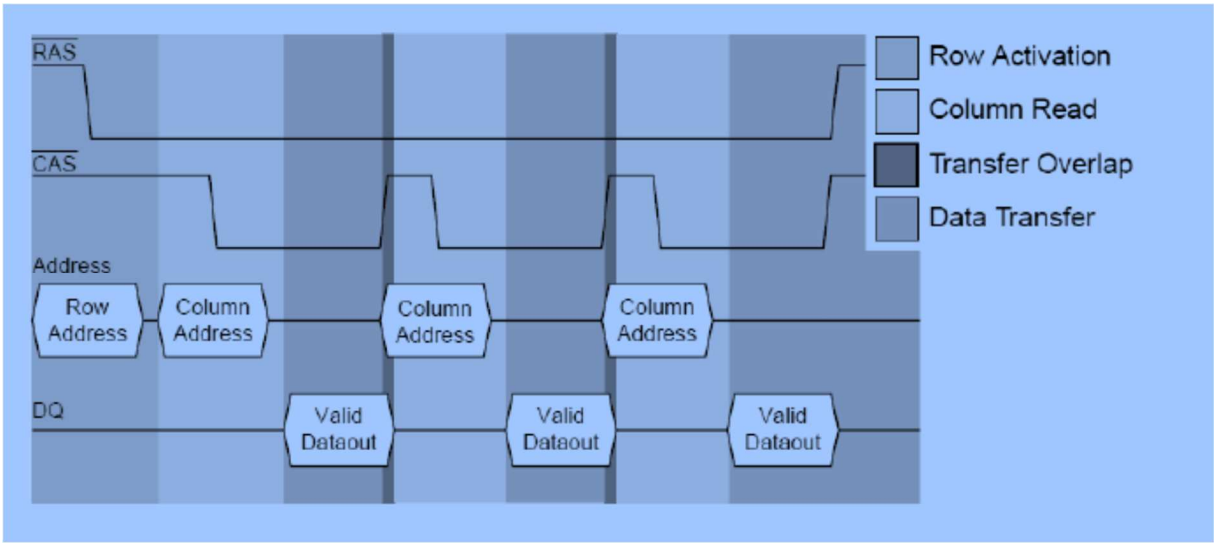
A klasszikus aszinkron DRAM

- Teljesítményének fokozására több, általában egymásra épülő megoldás is született, a továbbiakban ezeket vesszük sorra. Összehasonlítási alapként tekintsük az alábbi ábrát; itt minden olvasásánál újra és újra meg kell adni először a sorcímet, majd az oszlopcímet.



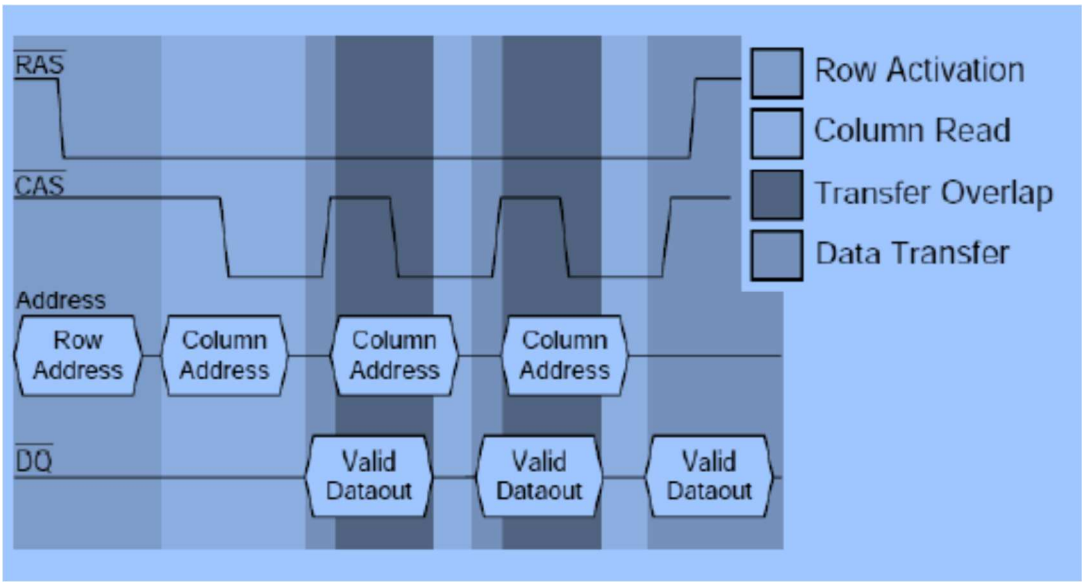
FPM DRAM

- A DRAM olvasásának átlagos ideje lerövidíthető, ha megengedjük, hogy azonos sorcímről történő olvasás esetén a /RAS jel alacsonyan tartása mellett, csupán a CAS jellel vezérelve egymás után több oszlopcímről is olvashassunk.
- Láttuk, hogy amíg a /RAS=0 fennáll, a sorcím által kiválasztott sor összes bitje elérhető az érzékelő erősítőkben, tehát csak ki kell választani közülük a számunkra még szükséges további biteket.
- Ezt a megoldást (gyors) *lapolvasás*nak (**F**ast **P**age **M**ode) nevezzük; az azonos sorcímhez tartozó oszlopokat egy memória "lap"-nak tekintjük.
- Figyeljük meg a megoldás nyereségét az alábbi ábrán!
    - A sorcímet laponként egyszer elegendő átvinni.
    - Apró átlapolás is megfigyelhető az érvényes adat és az új oszlopcím átvitelénél, de ennek mértéke nem jelentős.



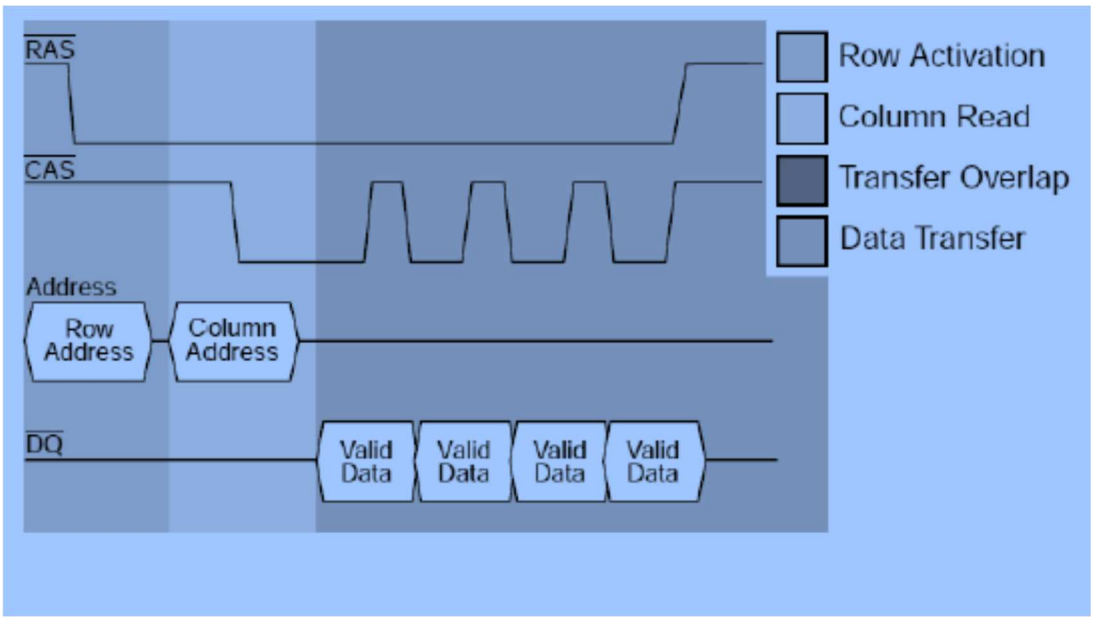
EDO DRAM

- A következő lépés az EDO (**E**xtended **D**ata **O**ut) DRAM, ami az FPM DRAM kiegészítése egy kimeneti latch-csel; így az adat kívülről még elérhető marad, miközben már új oszlopcímet adunk meg. Ezzel igen jelentős átlapolás érhető el a bitmezőből kiolvasott adat átvitele és az új oszlopcím megadása között.
- Megjegyzés: Az EDO DRAM-nál bevezették az /OE bemenetet, a CPU (vagy memória vezérlő) ezzel tudja jelezni, hogy meddig van szüksége az adatra.
- Figyeljük meg az átlapolást az érvényes adat és az új oszlopcím átvitele között!



BEDO DRAM

- Ismét egy újabb lépés a BEDO (Burst-Mode EDO) DRAM, ahol már nem is kell megadni az új oszlopcímet, hanem az oszlopcímet egy belső számláló egyesével növeli a kezdeti értékről.
- Így természetesen csak egymást követő címek érhetők el, de a gyakorlatban úgyis ez a szokásos eljárás.
- (Ez a megoldás is kifejezetten hasznos, de kevésbé terjedt el.)
- Figyeljük meg, hogy az időegységenként átvitt adatok mennyisége tovább nő!



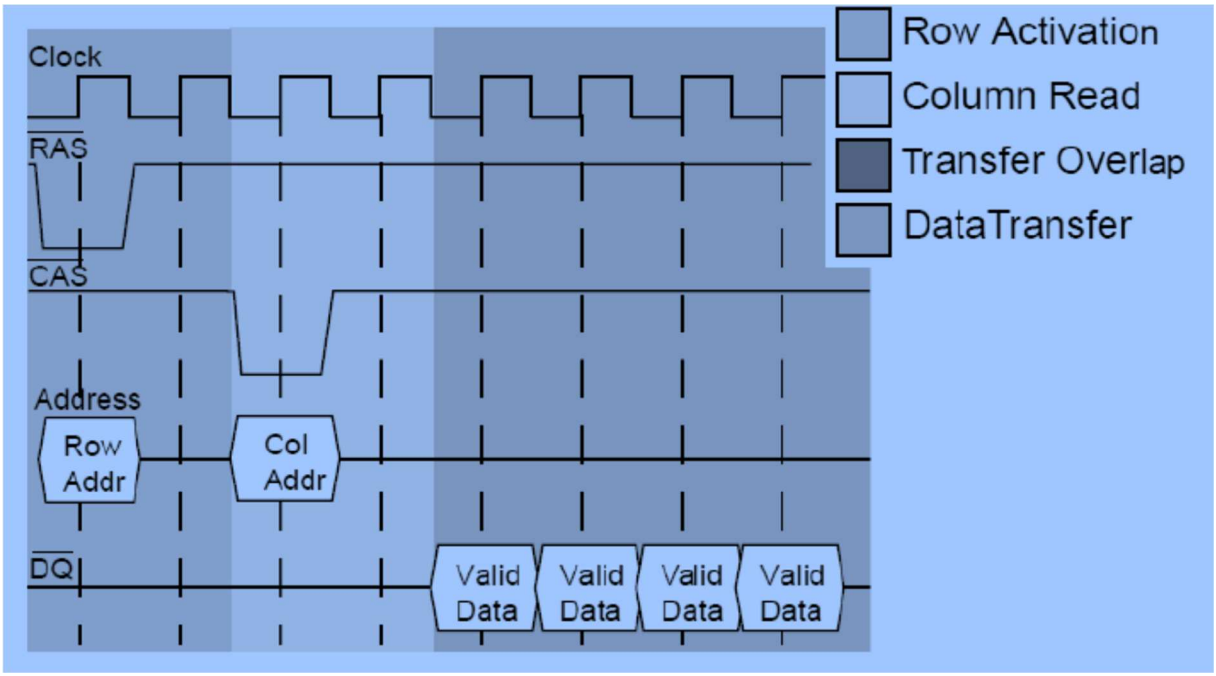
Szinkron DRAM típusok

SDRAM

Az SDRAM (**S**ynchronous **DRAM**) egy igen jelentős fejlődési lépés. Míg korábban a /CAS és /RAS bemenetek értéke tetszőleges időpontban változhatott, és azok közvetlenül

vezérelték a DRAM belső működését, addig az SDRAM-ban minden bemeneti változás egy külső órajellel szinkronizáltan történik.
Ez azért előnyös, mert az SDRAM tervezői így kihasználhatják, hogy az IC nem akármikor, hanem csak előre meghatározott időpontokban kaphat valamilyen parancsot.

- A BEDO DRAM-hoz hasonlóan az SDRAM is képes automatikusan növekvő oszlopcímekről egymás után több adat átvitelére (burst), de attól eltérően itt már ezt nem kell a /CAS jellel újra és újra igényelni, hanem az adatok megjelenítése az órajel ütemére történik, és az SDRAMban van egy programozható regiszter, amiben az egymás után átviendő adatok számát be lehet állítani (például a cache sor hosszának függvényében).
- Bár az SDRAM megjelenésekor teljesítményében nem volt jobb a BEDO DRAM-nál, hosszú távon nyertes lett, mert a szinkronizált működés egyre nagyobb működési frekvenciát tett lehetővé.
- Figyeljük meg, hogy az órajellel ütemezetten az újabb adatok megjelennek a kimeneten –minden további kérés nélkül!



- A további, ma is általánosan elterjedt típusok mind az SDRAM továbbfejlesztett változatai, ezeket röviden áttekintjük, de előbb megismerjük a legfontosabb időzítéseket.
- Megjegyzés: A későbbi, DDRx SDRAM-októl való megkülönböztetés érdekében az eredeti SDRAM-ot SDR (**S**ingle **D**ata **R**ate) SDRAM-nak is szokták nevezni.

44

Legfontosabb időzítések szinkron DRAM-oknál

SDRAM időzítések

- Mivel minden órajellel szinkronizáltan történik, ezért az időzítéseket nem ns-ban, hanem a szükséges órajel ciklusok számában szokták megadni. Természetesen ezek az órajel frekvenciájának függvényében értelmezhetők, és szükség esetén ns-ra is átszámíthatók. (Aszinkron DRAM esetében pedig természetesen ns-ban adják meg őket.)
- A következőkben megadjuk azt a négy időzítést, amivel az egyes SDRAM memória modulokat jellemezni szokták.
- Az időzítések értékét a következőkben használt sorrendben, a számértékeket egymástól kötőjellel elválasztva szokták feltüntetni, például: 3-4-4-8.

CAS Latency

- Definíció
  - A /CAS=0-tól az adat megjelenéséig eltelt idő.
    (Természetesen egy már aktív sor esetén.)
- *Magyarázat*
  - *Eddig tart, amíg az érzékelő erősítőben már jelen levő bitek közül az oszlopcím által kiválasztott bit a kimeneten megjelenik.*
- *Jelölések*
  - t_{CL}, t_{CAS}, (t_{ACT})
- Alternatív megnevezés
  - **A**ccess **C**olumn **T**ime

RAS to CAS Delay

Definíció

- A /RAS=0-tól legalább ennyi időnek el kell telnie a /CAS=0 megjelenéséig.

Magyarázat

- *Eddig tart a sor "megnyitása": a sor kiválasztásától addig eltelő idő, amíg a kiválasztott sor tartalma megjelenik az érzékelő erősítőben. Ez után lehet olvasni vagy írni.*

Jelölés

- t_{RCD}

Row Precharge time

- Definíció
  - A /RAS jelnek legalább ennyi ideig magas szintűnek kell lennie, azaz egy aktív sor lezárásától kezdve ennyi időnek kell eltelnie, mire egy másikat meg lehet nyitni.
- *Magyarázat*
  - *Ennyi idő szükséges a bitvezetékek előfeszítéséhez.*
- *Jelölés*

- t_{RP}

Row Address Strobe time

- Definíció
  - A /RAS jelnek legalább ennyi ideig alacsony szintűnek kell lennie.
- *Magyarázat*
  - *Ez az idő ahhoz szükséges, hogy a sor kiolvasása és visszaírása rendben megtörténjen.*
- *Jelölés*
  - t_{RAS}
- Alternatív megnevezések
  - **R**ow **A**ctive **T**ime
  - Active to Precharge (time)

DDR SDRAM

- A DDR (**D**ouble **D**ata **R**ate) SDRAM azonos órajel esetén az SDR SDRAM-hoz képest (közel) kétszer akkora átviteli sebességre képes.
- Ezt úgy oldja meg, hogy az órajelnek mind a felfutó, mind a lefutó élénél végez adatátvitelt. Míg az SDR SDRAM minimális burst hossza 1, a DDR SDRAM-é 2.
- A szabványos nevében már a fizikai órajel frekvenciájának a dupláját használja, így a DDR-200
- SDRAM valójában 100MHz-en működik.
  - A moduljai 64 bit szélesek, és a nevükben figyelembe veszik azt is, hogy hány byte információt visznek át párhuzamosan. Például egy 200MHz-en működő 64 bit széles DDR-400-at PC-3200 néven lehet megvásárolni, és maximum 3200MB/s átviteli sebességre képes.

DDR2 SDRAM

- A DDR2 SDRAM belső órajele csak fele a külsőnek, így adott belső órajel mellett a külső kétszer akkora lehet, és a külső órajelciklusonként visz át két bitet, tehát a belső órajel minden periódusa alatt 4-et.
- Míg a DDR-nél 2 bites volt, itt már 4 bit-es a prefetch puffer (a DDR3-nál pedig 8 bites).
- Nézzünk itt is egy tipikus számpéldát.
  - Egy DDR2-800 SDRAM-nak a belső órajele 200MHz, a külső 400MHz, a "800" úgy jön ki, hogy a külső órajel fel- és lefutó élénél is van adatátvitel; a 64 bites szélesség miatt pedig PC2-6400 modulként adják el, és maximum 6400 MB/s átviteli sebességre képes.

DDR3 SDRAM

- A DDR3 SDRAM belső órajele már csak negyede a külsőnek, így adott belső órajel mellett a külső négyszer akkora lehet, és a külső órajelciklusonként visz át két bitet, tehát a belső órajel minden periódusa alatt 8-at.
- Nézzünk itt is egy tipikus számpéldát.

- Egy DDR3-1600 SDRAM-nak a belső órajele 200MHz, a külső 800MHz, az "1600" úgy jön ki, hogy a külső órajel felés lefutó élénél is van adatátvitel; a 64 bites szélesség miatt pedig PC3-12800 modulként adják el, és maximum 12800 MB/s átviteli sebességre képes.

DDRx SDRAM

- A DDR sorozat tagjai egymással sem mechanikailag sem feszültségszint tekintetében nem kompatibilisek.
  - A disszipáció csökkentése érdekében egyre kisebb feszültségszinteket használnak.
- A DDR4 piaci megjelenése 2012-re várható.

A GDDR sorozat

- A GDDR (Graphics DDR) sorozat videokártyákhoz készült.
- A számozás 1-gyel (vagy kettővel) előbbre tart a DDR sorozatnál, így például a GDDR3 közelítőleg a DDR2-nek felel meg. Ahhoz képest van néhány előnye, ebből rendszertechnikailag számunkra az az érdekes, hogy egyidejűleg írható és olvasható.
- A GDDR4 a DDR3-ra épül és a GDDR3 utódjának szánták, de nem igazán váltotta be a hozzá fűzött reményeket.
- A GDDR5 is DDR3-ra épül, és 2008 óta kapható.

A Rambus irányzat

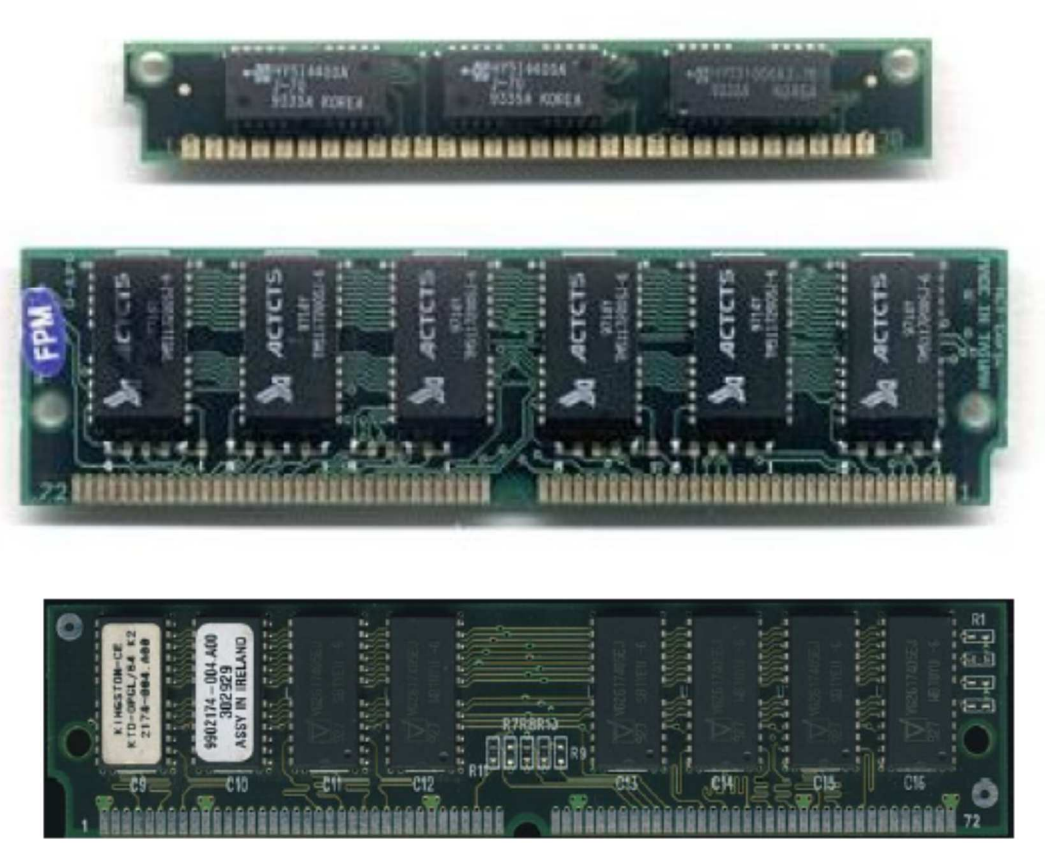
- A Rambus cég nem gyárt memóriát csak fejleszt és licencel. Első termékével az RDRAM-mal (Rambus DRAM) forradalmi újításokba fogott (például lényegesen keskenyebb buszt használ; a parancsait nem 1 bites vonalak jelei, hanem több bites üzenetek hordozzák), de a teljesítményhez képest az ár túl magas volt, így bár egy ideig terjedt, aztán az olcsóbb DDR sorozat kiszorította.
- A következő terméke az XDR (eXtreme Data Rate) DRAM jelenleg a DDR2 és a GDDR4 vetélytársa. Olyan esetekben van esélye, ahol a kis lábszám előnyt jelent.
- 2008-ban elkészült az XDR2 DRAM specifikációja is, de azóta sem talált gyártóra.

MEMÓRIA MODULOK

SIMM

- A SIMM (**S**ingle **I**n-line **M**emory **M**odule) memória modulok csak az egyik oldalon tartalmaznak memória ICket és bár a csatlakozási felületnél mindkét oldalon vannak érintkezőik, ezek redundánsak.
- Főleg az 1980-as és 90-es években használták őket, 30 és 72 érintkezős kivitelben készültek, az előbbiek 8/9 bitesek, az utóbbiak 32/36 bitesek voltak - attól függően, hogy tartalmaztak-e hibajavításra (ECC) 8 bitenként +1 bitet. Ezek általában még aszinkron DRAM-ok voltak, de készült még DDR SDRAM is SIMM kivitelben.

- Felül 8/9 bites aszinkron DRAM, alatta 32/36 bites FPM DRAM, legalul 32/36 bites EDO DRAM

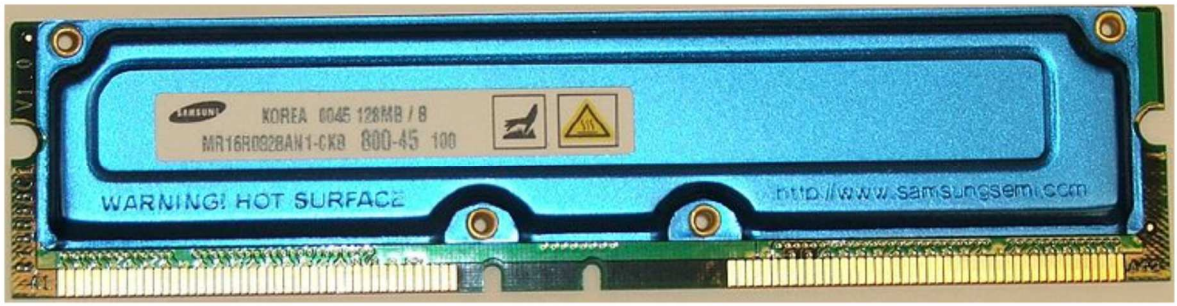


DIMM

- A DIMM (**D**ual **I**n-line **M**emory **M**odule) memória modulok mindkét oldalon tartalmaznak memória IC-ket, adatszélességük is 64 bit, és kihasználják mindkét oldalon az érintkezőket.
    - Az Intel Pentiummal terjedtek el, így már nem kellett őket párban használni, mint a SIMM modulokat.
- Bár készültek FPM és EDO DRAM-ok is DIMM kivitelben, de a DIMM-ek már főleg szinkron DRAM-ok, azon belül bejárták a technológiai fejlődést. Az érintkezők számában is változatosak.
- Az alábbiak közül a felső egy 168 érintkezős SDRAM, az alsó egy 184 érintkezős DDR SDRAM:

RIMM

- Bár az **RDRAM** (**R**ambus **DRAM**) vagy **DRDRAM** (**Direct Rambus DRAM**) modulok is SIMM vagy DIMM kivitelűek, megkülönböztetésül mégis **RIMM**-nek (**R**ambus **I**n-line **M**emory **M**odule) szokták őket nevezni.
    - Jól látható jellemzőjük az integrált hűtő - de integrált hűtő lehet más SIMM/DIMM modulon is!

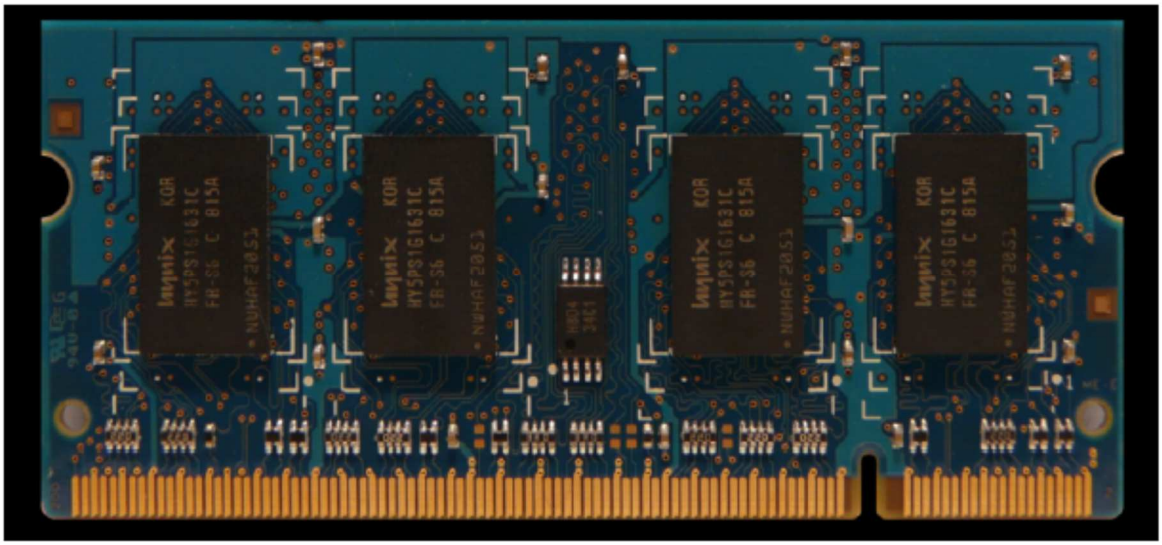


XDR DRAM

- Az **RDRAM** utódja az **XDR** (e**X**treme **D**ata **R**ate) **DRAM** az elődjéhez hasonlóan szintén integrált hűtővel rendelkezik:

SO-DIMM

- A **SO-DIMM** (**S**mall **O**utline **DIMM**) a DIMM helytakarékos változata. Többféle érintkezőszámmal (100, 144 és 200) készül és DDR/DDR2/DDR3 SDRAM egyaránt található köztük. Az alábbi képen egy PC2-6400 DDR2 SO-DIMM látható.



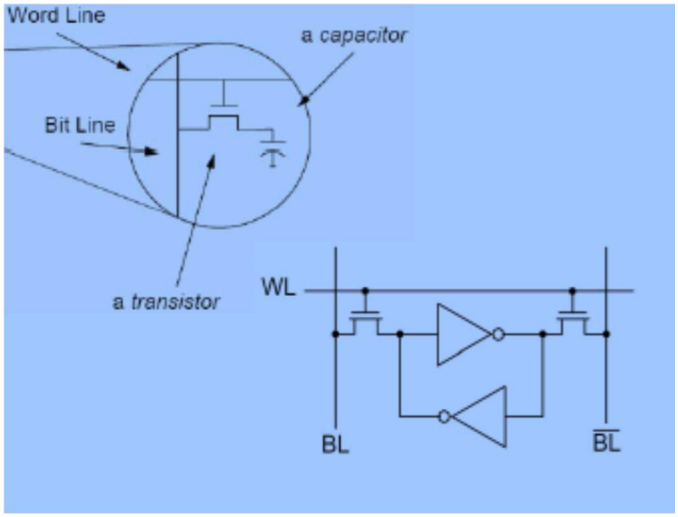
Önálló hallgatói munka

1. Számítsa ki a bitmezőt befoglaló négyzetekben létrejövő jelterjedési késleltetés arányát egy 16kB-os L1 és egy
1MB-os L2 cache esetén!
2. Van egy SDRAM chipünk. Az időzítései: 3-3-3-6. Feltéve, hogy előfeszített (precharged) állapotban van, és két bitet kell belőle kiolvasnunk, amelyek sorcíme különböző; a kezdéstől számítva mennyi idő múlva jelenik meg a második bit a kimeneten, ha a chip 100MHz frekvenciájú órajelet kap?
3. Oldja meg az előző feladatot 6-6-6-18 időzítéssel és 200MHz-es órajellel!
4. Egy PC3-6400-ként eladott DDR3 memóriamodul milyen belső és milyen külső órajellel működik?

Összefoglalás

- SRAM és DRAM felépítése és ennek következményei

- SRAM felépítése és működése
- DRAM felépítése és működése
- SRAM és DRAM összehasonlítása



- DRAM fejlődése
  - Aszinkron DRAM típusok
  - Szinkron DRAM típusok
  - Legfontosabb időzítések szinkron DRAM-oknál
- Memória modulok