Tavalyi ZH Ajándék, annak, aki a végére ért...

DIGSZAR LEHETSÉGES 2. ZH KÉRDÉSEK

- Rajzolja fel a PCI busz burst olvasásának idődiagramját! A periféria sebssége nem csak a kezdeményező sebessége idézhet elő késleltetést!
- Rajzolja fel egy FPGA ált. felépítését! Az IOB-t részletezze!
- Adja meg a PLD-k típusait és rajzolja le őket!
- Ismertesse a lapozásos címezést! Rajzolja fel egy lapozásos címfrdító áramkör blokkdiagramját! A lapok 1024 bájtosak, a laptábla 4096 bejegyzést tartalmazhat. 1 GB fizikai memória címezhető.
- Rajzolja fel egy FIR szűrő részletes adatút blokkdiagramját! Magyarázza a jelek jelentését!
- Jellemezze a RISC számítógépet és adjon legalább egy példát.
- Adja meg a minimális olvasási időt egy RAM-nál, adottak az alábbi katalógus paraméterek t[RCMIN]=12 ns, t[ACSMAX]=12 ns, t[OEMAX]=6 ns
- Rajzolja le a hand shaking folyamatot írás esetén!
- Definiálja a Moor autómata modellt!
- Mikroprogramozott vezérlőegység feladata, vertikális mikroprogramozás (ábra, előnyök, hátrányok)+horizontális
- vezérlőegység programozható makrocellákkal (fajta, ábrák, jellemző tul.)
- ismertesse, hogy hogyan programozható egy programozható VLSI (rajz+előnyök, hátrányok)
- Minimum hány lábra van szükség egy 1024*4-es 1 CS-sel és közös I/O-val rendelkező RAM megvalósításához
- Ismertesse a dinamikus memória frissítési folyamatát (idődiagram)
- Ismertesse a statikus memória írási folyamatát (idődiagram)
- Milyen átviteli technológiát használnak a PCI busz jelei és milyen jelcsoportok találhatók a PCI csatlakozón, továbbá milyen csatlakozó felépítése?
- Rajzoljon fel egy PCI brust write tranzakció idődiagramját és a diagramon mutasson példát a várakozási ciklusra mindkét egység részéről.
- Mekkora minimális hozzáférési idő az ötödik memóriabank eléréséhez, ha t[TRANSLATION]=20 ns, t[HOZZÁFÉRÉS]=20 ns
- Hogyan használhaó a memórából több független program?

- Vezérlőegységek feladata, fajtái. Present state register, next state logic előállítása multiplexeres módszerrel. Rajzoljon példát és valósítsa meg!
- Sorrendi vezérlő egységek megvalósítása shift regizterekkel, átlapoló impulzusok elve. Rajzoljon egy példát egy vezérlő jel megvalósítására!
- Aszinkron handshake protokoll írási és olvasási művelet. Ábrázolja a folyamatokat és adja meg az egyes jelek jelentését!
- I/O csatornák. Milyen fajtáik vannak? Hogy működnek?
- DRAM struktúrája és működése
- Sorolja fel a cache-k fajtáit. Rajzolja fel részletesen a legrugalmasabb logikai felépítést!
- Rajzolja fel a Xilinx FPGA-k fejlesztői rendszerének felépítését!
- Rajzoljon fel egy PCI buszos számítógép architektúrát EISA és SCSI busszal
- Többprocesszoros rendszerek csoportosítása (Flynn), példák
- 29 bittel mekkora memóriát lehet megcímezni?(12 kbyte?)

- Jellemezze a CISC számítógépet, és adjon legalább egy ilyen processzor típust

- Rajzolja fel és röviden ismertesse az arbitrációt és fajtáit
- Definiálja a Mealy autómata modellt
- Rajzolja fel a Xilinx FPGA felépítését (CLB és I/O)
- ismertesse a dinamikus memória írási-olvasási folyamatát (idődiagram)
- Ismertesse a statikus memória olvasási folyamatát (idődiagram)
- Rajzolja fel egy PCI burst read tranzakció idődiagramját és a diagramon mutasson példát a várakozási ciklusra mind a két egység részéről.
- hasonlítsa össze a szegmentálást és a lapozást!
- Rajzolja fel egy I/O interfész egység blokkdiagramját. Azonosítsa a jelentősebb blokkokat és a különböző adatátviteli módokhoz tartozó jeleket, regisztereket.
- Ismertesse az SRAM-ok jellemőit tábláztos formában. Rajzolja fel felépítésüket, alapcelláját.
- Rajzolja fel egy szegmentált virtuális memória címzéses rendszerben címfordító hardver blokkdiagramját! a rendszerben egyszerre 64 program futhat és 32 szegmensből állhat (a hiányzó adatokat értelemszerűen válassza meg!)
- Rajzoljon fel egy VL buszos számítógép archtektúrát, ISA, és SCSI busszal.

- Mi a regiszter ablakozás? (rajz, magyarázat)
- Ismertesse a statikus memória írási-olvasási folyamatát (idődiagramok)
- Rajzoljon fel egy 2-way set associative cache-t a következő paraméterekkel: addres 32 bit, cache line: 32 byte, méret: 1024 kbyte.
- Adja meg egy PCI express Transacton Layer Packet formátumát!

- Definiálja aHarvard-féle száítógép architektúrát
- Rajzolja fel egy CPLD (complex programmable Logic Device) belső felépítését!
- Rajzoljon fel egy fxpontos DSP architektúrát és adja meg néhány jellemző tulajdonságát!
- Rajzoljon fel egy szegmentált virtuális memóriakezelő áramkört memóriavédelemmel! Szükséges paraméterek: virtuális cím: 32 bit, szegmensek száma: 8, fizikai cím: 28 bit
- Ismertesse a szinkron dnamikus memória (SDRAM) írási-olvasási folyamatát (idődiagramok) Adja meg a memóriaelérés tipikus idejét!

- Miért támogatja a RISC architektúra a pipeline feldolgozást?
- Aszinkron hand-shaking írás esetén.
- Definiálja a Moore-féle autómata modellt (3 halmaz, 2 leképzés, 1 rajz)
- ismertesse a szinkron statikus memória (ZBT SSRAM) írási folyamatát!Ismertesse a SCSI buszt! Hogyan zajlik a SCSI buson az adatátvitel?

2011-es... bocs, csak most találtam...

A csoport

- 1.Rajzolja fel a Moore automata modellt és adja meg jellemzőit (halmazok, függvények)! (2p)
- 2.Rajzoljon fel egy vertikális mikroprogramozott vezérlő egységet! Jellemeze mind a horizontális, mind a vertikális típusok előnyös, hátrányos tulajdonságait is! (2p)
- 3.A VLSI alkatrészek programozható összeköttetéseiben, milyen lehetséges programozási módok léteznek? Adja meg tulajdonságaikat, felépítésükkel együtt! (3p)
- 4.Rajzolja fel a XILINX FPGA egy konfigurálható logikai blokkján belüli slice felépítését! Mire használhatóak? (3p)
- 5.Mi az az arbitráció? Rajzolja fel a lekérdezéses (polling) arbitráció blokkdiagrammját!

(2p)

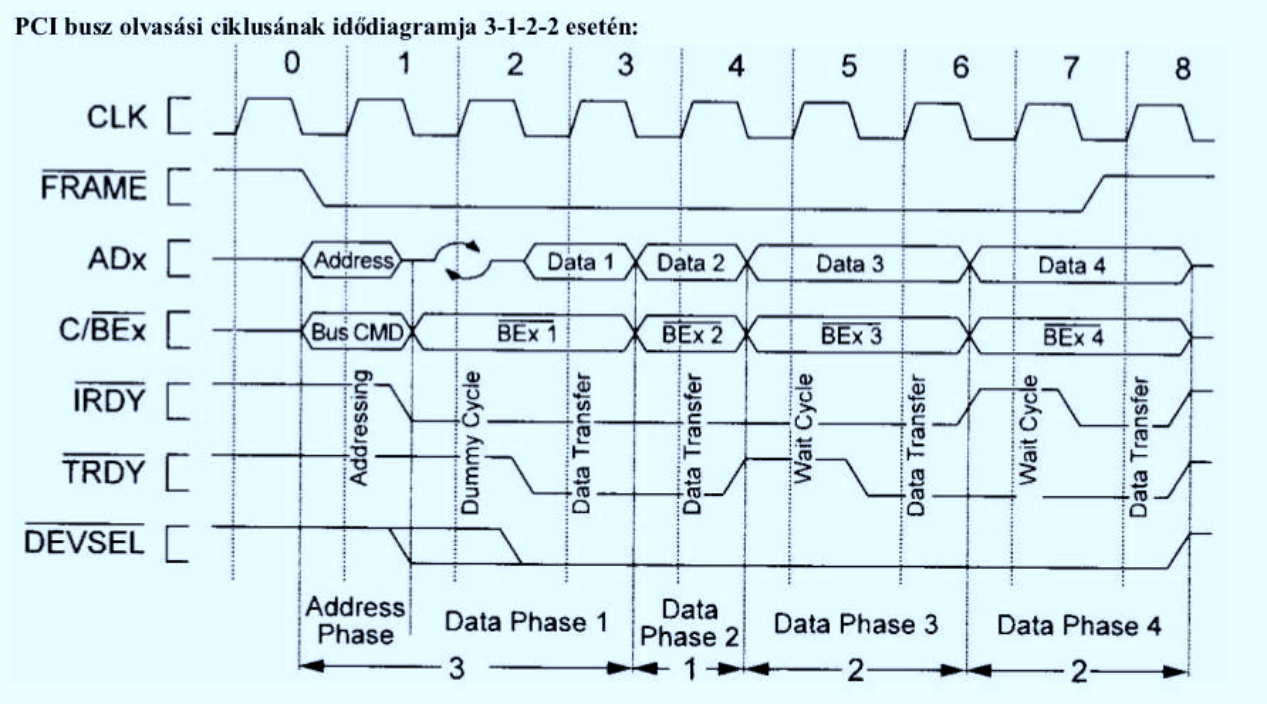
- 6.Rajzolja fel az optimális PCI read burst tranzakció idődiagramját, a fontosabb jelek megadásával! Mitől függenek az egyes adatrészek átviteli idejei? (3p)
- 7.Adja meg egy aszinkron dinamikus memória cella belső felépítését, tulajdonságait, és röviden írja le működését! Adja meg a memória elérés tipikus idejét! Ha szükséges, mennyi időnként kell frissíteni? (3p)
- 8.Jellemezze a DDR memóriákat (tulajdonság, működési elv, fizikai paraméterek)! Mi a különbség a SDRAM-hoz képest? (2p)
- 9.Rajzoljon fel egy szegmentálásos virtuális memóriakezelő áramkört! Határozza meg a blokkok funkcióját! Mit tárolunk a szegmens táblában? Hogyan történik a címfordítás? (3p)
- 10.Mik azok a cache memóriák? Mit jelent a cache hit, cache miss? (2p)

B csoport

- 1.Rajzolja fel a Mealy automata modellt és adja meg jellemzőit (halmazok, függvények)! (2p)
- 2.Rajzoljon fel egy horizontális mikroprogramozott vezérlő egységet! Jellemeze mind a horizontális, mind a vertikális típusok előnyös, hátrányos tulajdonságait is! (2p)
- 3.Ismertesse a Programozható Logikai Eszközök (PLD) típusait, tulajdonságait, rajzokkal együtt! (3p)
- 4.Rajzolja fel a XILINX FPGA általános belső felépítését! Sorolja fel, hogy milyen logikai illetve dedikált erőforrások találhatóak az FPGA-alapú programozható eszközön? (3p)
- 5.Mi az az arbitráció? Rajzolja fel a soros (Daisy Chain) arbitráció blokkdiagrammját! (2p)
- 6.Rajzolja fel az 2-1-1-2 PCI write burst tranzakció idődiagramját, a fontosabb jelek megadásával! Mitől függenek az egyes adatrészek átviteli idejei, és mikor lenne optimális a művelet? (3p)
- 7.Adja meg egy aszinkron statikus memória cella belső felépítését, tulajdonságait, és röviden írja le működését! Adja meg a memória elérés tipikus idejét! Ha szükséges, mennyi időközönként kell frissíteni? (3p)
- 8.Jellemezze a DDR3 memóriákat (tulajdonság, működési elv, fizikai paraméterek)! Mi a különbség az előző generációhoz képest? (2p)
- 9.Rajzoljon fel egy lapozásos virtuális memóriakezelő áramkört! Határozza meg a blokkok funkcióját! Hogyan történik a címfordítás, és mit tartalmaz egy laptábla? (3p)
- 10.Mi az a cache koherencia, mondjon rá egy példát! (2p)

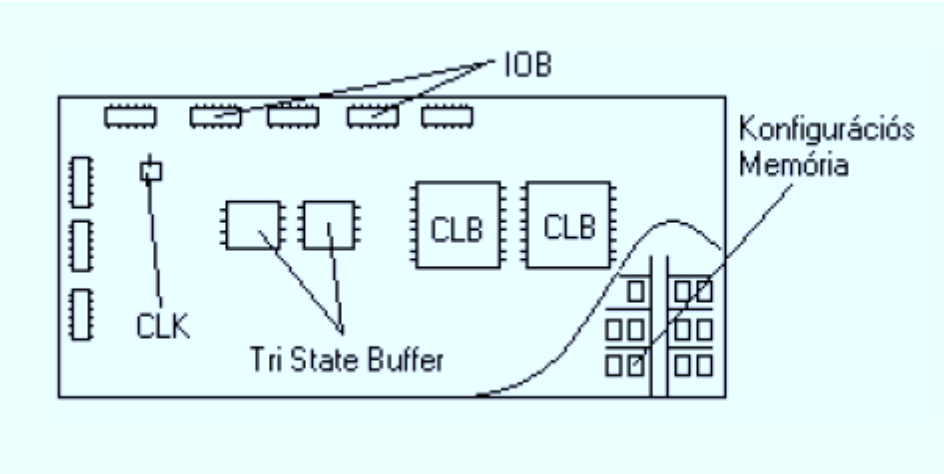
Databázén jól megtanultuk,hogy a redunncia nem tesz jót az érzékeny pici lelkünkenk, így az ismétlődő kérdéseket önhatalmúlag kitöröltük. A “kidolgozott” feladatokra jellemző,hogy csak kopipészt történt a szigósegédanyagból. Talán így kevesebbet kell használni a ctrl+f billentyűparancsot ;)

- **Rajzolja fel a PCI busz burst olvasásának idődiagramját!** A periféria sebssége nem csak a kezdeményező sebessége idézhet elő késleltetést! és a diagramon mutasson példát a várakozási ciklusra mindkét egység részéről.



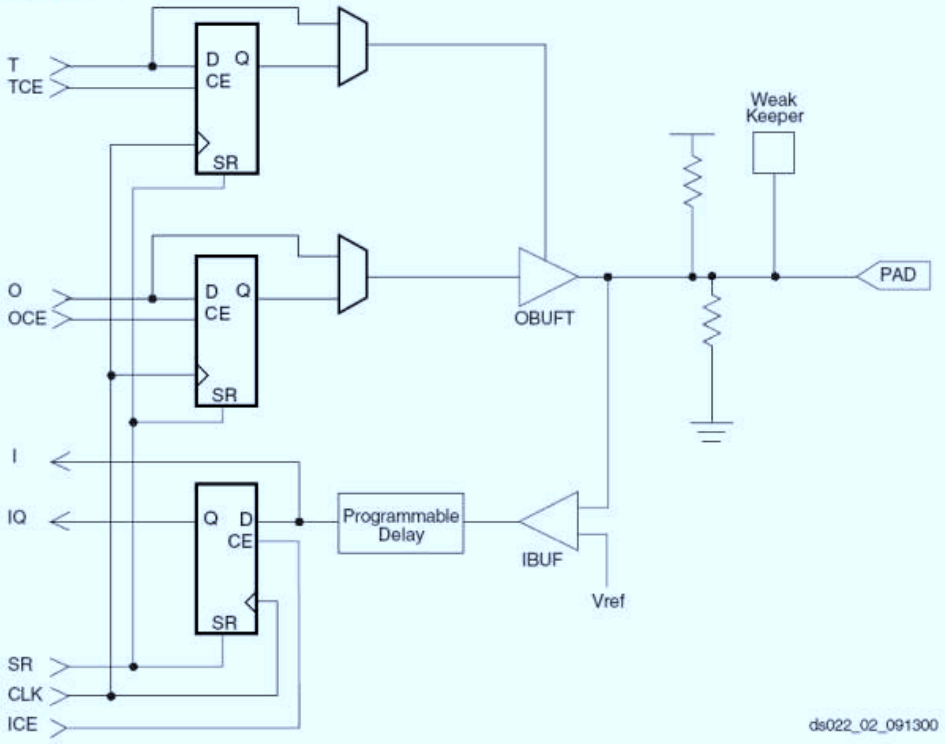
Rajzolja fel egy FPGA ált. felépítését! Az IOB-t részletezze!

FPGA: általánosan



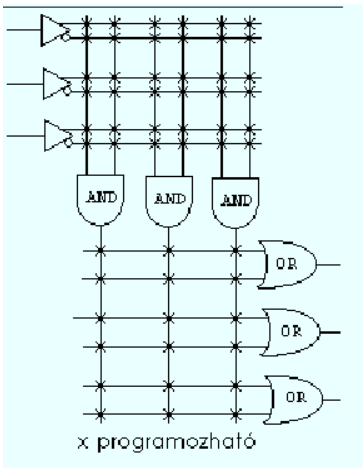
IOB: programozható be/ki meneti blokk.kapcsolat a belső vonalak és a tokozás lábai (pad) között (lehet bementi /kimeneti / kétirányú (tristate) kapcsolatként definiálni). Szintén D-FF-kel rendelkezik, belső puffere (buffere) van, nem kell az adatokat a külső lábon (pad) tartani. A kimenet lehet negált, vagy ponált.

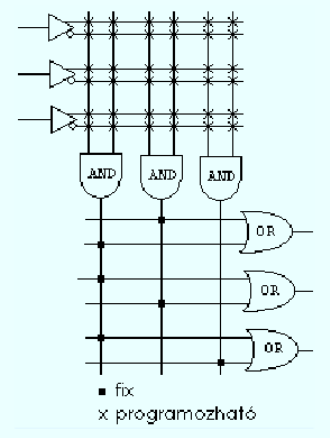
Virtex IOB - I/O Blokk felépítése:

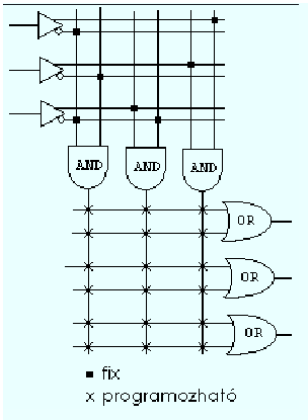


Adja meg a PLD-k típusait és rajzolja le őket!

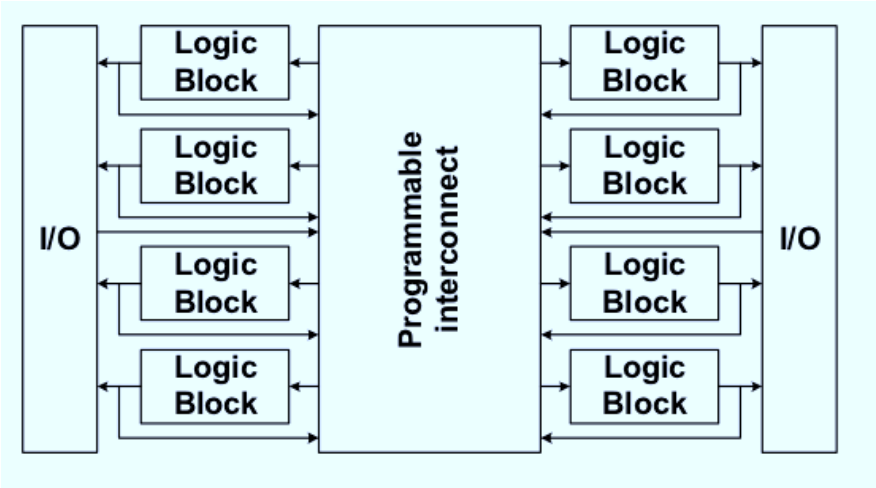
PLA PAL, PROM,







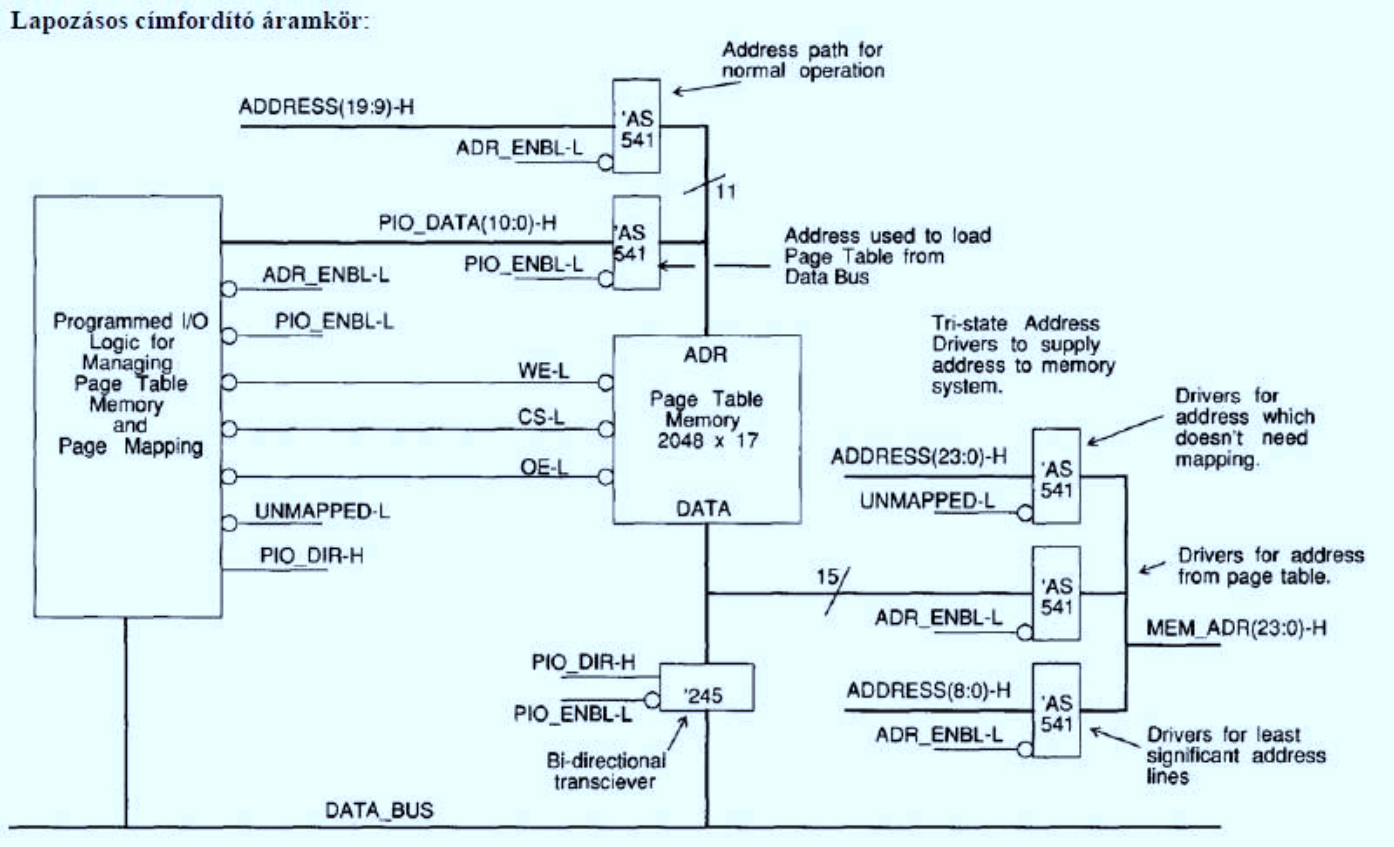
CPLD



~~FPGA is~~ ~~ide tartozik még.~~

- **Ismertesse a lapozásos címezést! Rajzolja fel egy lapozásos címfrdító áramkör blokkdiagramját! A lapok 1024 bájtosak, a laptábla 4096 bejegyzést tartalmazhat. 1 GB fizikai memória címezhető.**

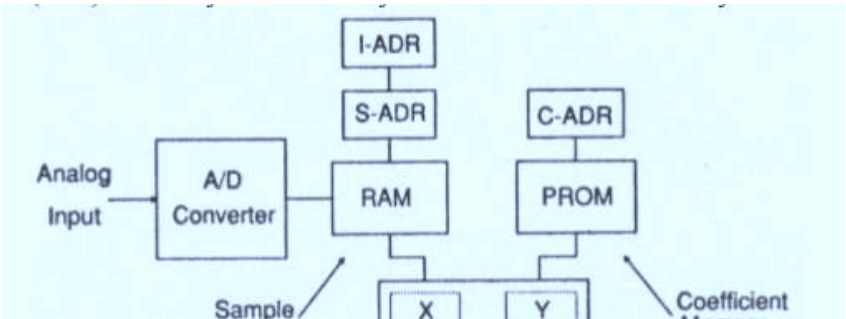
Fizikai Cím=Lap Báziscíme + Lap eltolás (offset) címe. A „+ „itt konkatenációt / összefűzést, nem pedig összeadást jelent. Ezáltal sokkal gyorsabb lesz a fizikai cím leképezése, mivel nem kell összeadót használni, a konkatenációt egyszerű huzalozással megoldhatjuk. A hexadecimális címek egy-egy számjegyét 4 bites bináris számmá kódoljuk, és összefűzzük.



gyönyörű

- **Rajzolja fel egy FIR szűrő részletes adatút blokkdiagramját! Magyarázza a jelek jelentését!**

- MAC:(Multiplier/Accelerator): elvégzi a szorzást, és az összeadást minden egyes órajel ciklusban. Három regiszterből áll: két bemeneti (X és Y) és egy kimeneti regiszterből (P).
- Együttható- memória (Coefficient memory): C_MEM tárolja a Ci konstansok értékeit. Ez egy kisméretű PROM memória.
-



Együttható- memóriacímző regiszter: (C_ADDR) egy számláló, amely a következő együtthatót azonosítja. 0-ról indul minden egyes iterációkor és az együtthatók címei egyesével inkrementálódnak.

- Minta- memória (S_MEM): az éppen aktuális és az azt megelőző 24 mintát tároljuk el. RAM, amely 32 értékből csak 25-t használ.
- Minta- memóriacímző regiszter: (S_ADDR) Ez egy számláló, amely az aktuális minta címét inicializálja, és következő utasítás címére inkrementálódik.
- Kezdő minta cím regiszter: (I_ADDR): azonosítja az algoritmus kezdetét.
- A/D konverter: (ADIN) Minden iterációban ezen az egységen keresztül kapjuk az új adatot.
- D/A konverter: (DAOUT) A kimeneti regiszterből kapja az adatot, és analóg jellé alakítja.
- Kimeneti regiszter: (OUT) MAC-ből jövő értéket tárolja és a D/A konverternek továbbítja.

- **Jellemezze a RISC számítógépet és adjon legalább egy példát.**

RISC: Reduced Instruction Set Computer (Csökkentett utasításkészletű számítógép):
-Csak a kívánt alkalmazásra jellemző utasítástípusokat tartalmaz, az utasításkészlet összetettségének csökkentése végett kihagytak olyan utasításokat, amelyeket a program amúgy sem használ, ezáltal nő a sebesség.
-*Minimális utasításkészletet és címzési módot* (csak amit gyakran használ), gyors HW elemeket, optimalizált SW használ.
-Azonban, hogy a programozási nyelvek komplex függvényei leírhatók legyenek (ahogyan az a CISC-nél működik) szubrutinokra, és hosszabb utasítássorozatokra (sok egyszerű utasítás) van szükség.
-Hogyan tudjuk a rendszer erőforrásait hatékonyan kihasználni? Gyorsabb működés érhető el (MIPS), egyszerűbb architektúra megvalósítására kell törekedni.
-*Azonos hosszúságú utasításformátum* (korlátozott utasításformátum miatt a tárolt programú gépeknél az *F-D-E* folyamatban a dekódolás minimális idejű lesz (nullának feltételezzük), amely során azonosítani kell

a végrehajtandó utasítást)
-*Huzalozott (hardwired) utasításdekódolás* (a hardveres dekódolás megvalósításához kombinációs logikát használ, azonban a mai memóriaalapú mikro-kódú gépeknél ez lassabb).
-*Egyszeres ciklusvégrehajtás*: (minden egyes ciklusban egy utasítást hajt végre, ha ezt sikerülne elérni optimális lenne az erőforrás kihasználás - VLSI technológiafüggő. Egy lebegőpontos művelet rendkívül kis idő alatt végrehajtható. Hátránya, hogy vannak bizonyos műveletek, amelyeket egy ciklus alatt nem kapunk meg: pl. a memóriában lévő érték inkrementálásakor az értéket előbb ki kell venni, frissíteni, majd visszaírni a memóriába).
-*LOAD/STORE memóriaszervezés*: 2 művelet – tölt és tárol (regiszter <-> memória). Regiszterre azért van szükség, mivel a betöltött adatot sokkal gyorsabban tudjuk kiolvasni, mint a memóriából. Az aritmetikai/logikai utasítások a regiszterekben tárolódnak. A regiszterek gyorsabbak, mint a memória-intenzív műveletek.
-További architektúra technikák: **pipeline** (utasítás feldolgozás párhuzamosítása), többszörös adatvonalak, nagyszámú gyors regiszterek alkalmazásával.

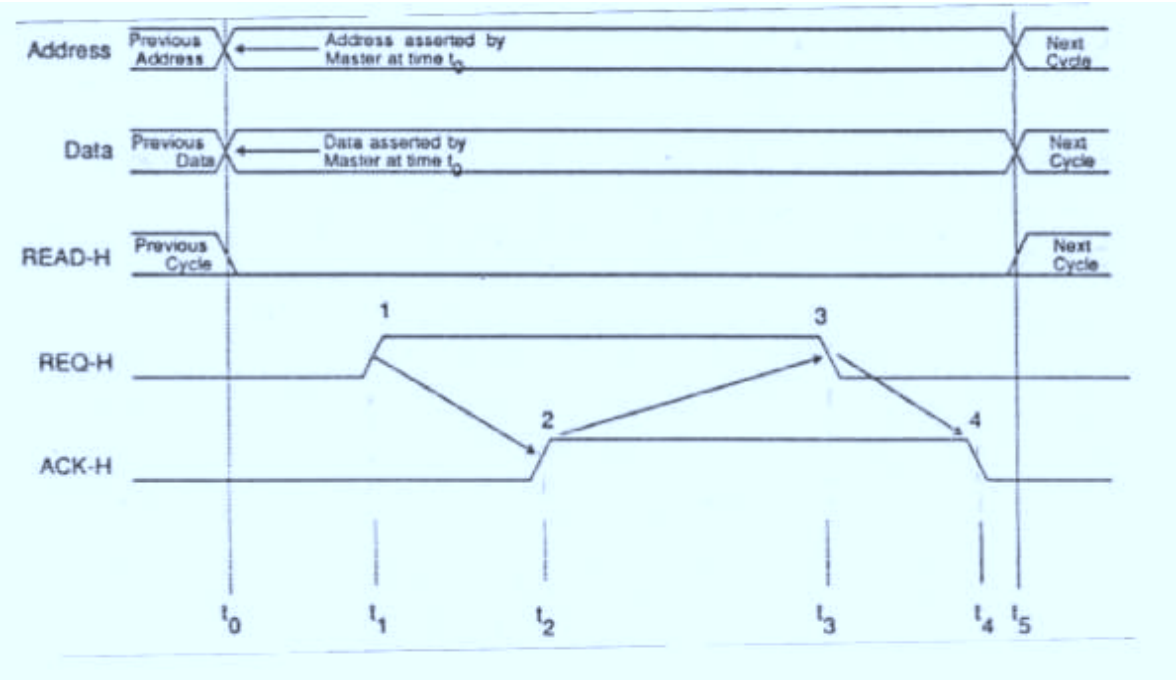
Például: Motorola 88000 RISC rendszere, vagy Berkeley Egyetem RISC-I rendszere.

- **Adja meg a minimális olvasási időt egy RAM-nál, adottak az alábbi katalógus paraméterek t[RCMIN]=12 ns, t[ACSMAX]=12 ns, t[OEMAX]=6 ns**

Aszinkron handshake protokoll írási és olvasási művelet. Ábrázolja a folyamatokat és adja meg az egyes jelek jelentését!

Adress: címvonal
Data: adatvonal
vezérlőjelek:
- READ-H: olvasás (ha magas akkor a Master olvas a Slaverről,ha alacsony akkor ír a Slavere)
- REQ-H: kérés
- ACK-H:nyugtázás

Olvasás:

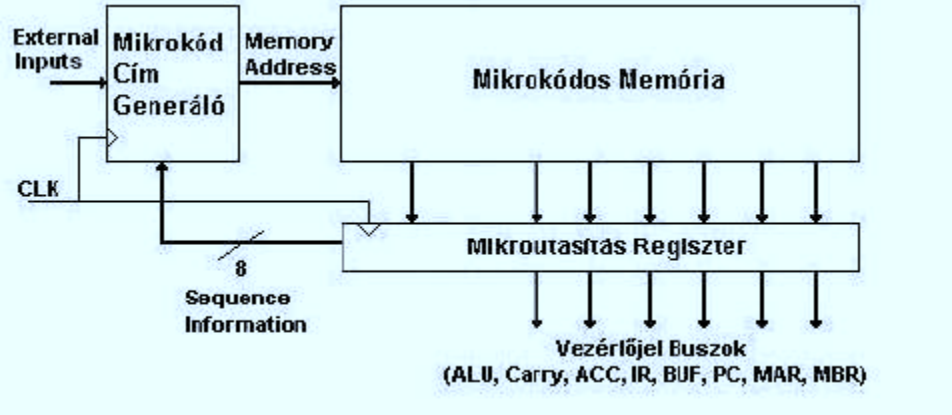


Írás: a diagrrammon annyi a különbség,hogy a READ-H vezérlőjel magasan van és **az adat csak t2 után érkezik a Slaveből a Masterbe!!**.

- **Definiálja a Moor autómata**
-

Itt a kimenetek közvetlenül csak a pillanatnyi állapottól függenek (a bemenettől függetlenek). Tehát a kimenetet nem a bemenethez, hanem az állapotoknak megfelelően szinkronizáljuk. Ezért használunk állapotregisztert. Három halmaza van: X – a bemenetek, Z – a kimenetek, Y – az állapotok halmaza. Visszacsatolni az állapotregisztert a késleltetés miatt kell. Halmazok közötti leképezési szabály: μ(Yn)→ Z : kimenet meghatározása! δ(Xn,Yn)→Yn+1 : következő állapot meghatározása /Input – Next State – Present State -Output/

- **Mikroprogramozott vezérlőegység feladata, vertikális mikroprogramozás (ábra, előnyök, hátrányok)+horizontális**



Hotizontális: Minden egyes vezérlőjelhez saját vonalat rendelünk, ezáltal horizontálisan megnő a mikro-utasításregiszter kimeneteinek száma, (=horizontálisan megnő a mikrokód). Minél több funkciót valósítunk meg a vezérlőjelekkel, annál szélesebb lesz a mikrokód. Ennek köszönhetően ez a leggyorsabb mikrokódos technika, mivel minden bit független egymástól ill. egy mikrokóddal többszörös (konkurens) utasítás is megadható. Pl: a megfelelő regisztereket (memória, ACC) egyszerre tudjuk az órajellel aktiválni, ezáltal egy órajelciklus alatt az információ mindkét irányba átvihető. **Növekszik a sebesség**, mivel nincs szükség a vezérlőjelek dekódolását végző dekódoló logikára. Így **minimál**isra csökken a műveletek **ciklusid**eje. Azonban **nagyobb** az **erőforrás szükséglete**, és **fogyasztása**.

Vertikális:
- külön dekódoló
- lassabb, energiatakarékosabb

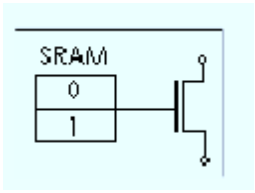
-

- **vezérlőegység programozható makrocellákkal (fajta, ábrák, jellemző tul.)**

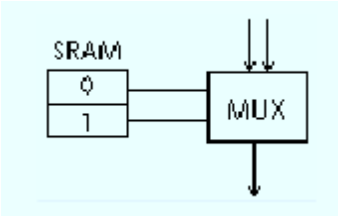
Lásd előző feladatnál PLD-k (ebben nincs benne az FPGA)

Ismertesse, hogy hogyan programozható egy programozható VLSI (rajz+előnyök, hátrányok)

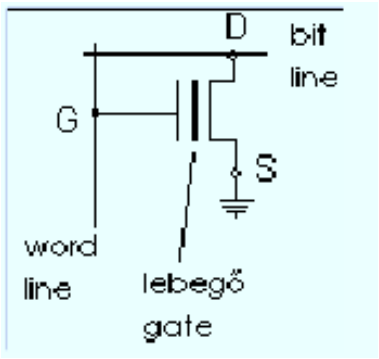
SRAM-os:
végtelen sokszor újraprogramozható (statikus RAM)
- táp kikapcsolása után az SRAM elveszti tartalmát
- bekapcsoláskor (inicializáláskor) a programot be kell tölteni, fel kell programozni
- az SRAM cellára egy áteresztő tranzisztor van csatolva. A tranzisztor vagy kinyit (vezet), vagy lezár. Az SRAM értéke, ami egy bitet tárol (’0’ vagy ’1’) letölthető. Összeköttetéseket, vagy MUX-ok állását is eltárolja.
- 1 bit tárolása az SRAM-ban (min. 6 tranzisztorból áll)
- sok tranzisztor (standard CMOS), nagy méret, nagy disszipáció
- nem kell frissíteni az SRAM-ot
- nagy 0.5-2 kΩ átmeneti ellenállás
- nagy 10-20 fentoF parazita kapacitás



Multiplexeres: Működése hasonló az SRAM celláéhoz.

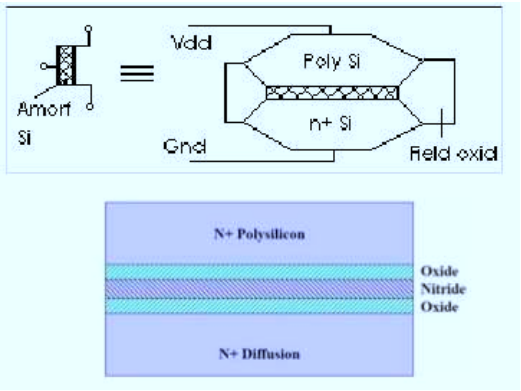


Floating Gate:
programozása: töltéseket viszünk fel a word-line felől a lebegő gate-re, kinyit a tranzisztor (a control gate befolyásolja a működését)
- többször törölhető (kis ablakon keresztül UV fénnyel)
- írás: nagy feszültség hatására töltést viszünk fel a lebegő gate-re
- kikapcsoláskor is megőrzi tartalmát, töltések nem sülnek ki (akár 99 évig)
- nagy 2-4 kΩ átmeneti ellenállás
- nagy 10-20 fentoF parazita kapacitás



Antifuse:

- az„átégetés” irreverzibilis folyamat, nem lehet újraprogramozni
- kis méreten megvalósítható, kis disszipáció
- kis átmeneti ellenállás 300 Ω
- kis parazita kapacitás 1.1-1.3 fentoF
- előállításához sok maszkréteg szükséges, drága technológiát igényel
- Típusai: Amorf Si vagy ONO (Oxid-Nitrid-Oxid)



EPROM/EEPROM/FLASH-os: (Lattice)

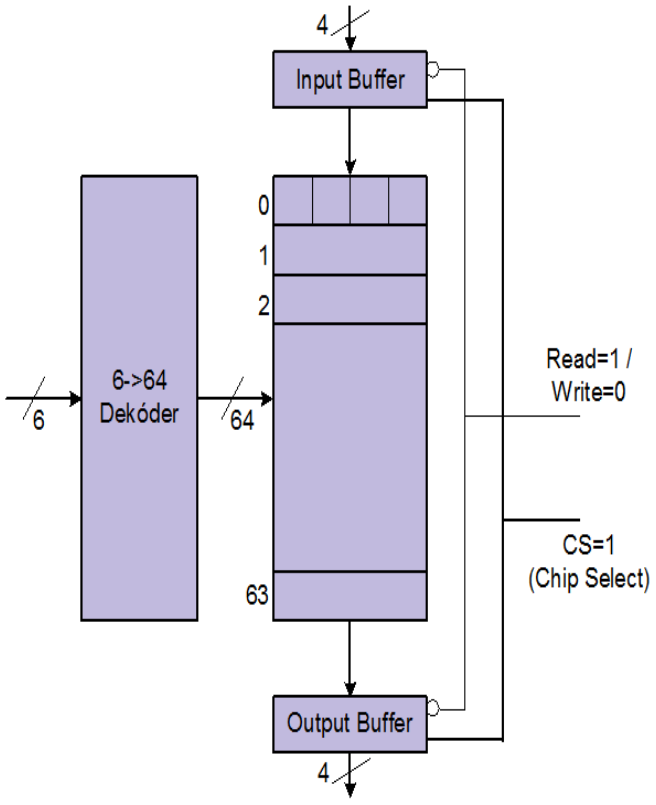
- A floating-gates technológiát alkalmazza!
- Megőrzi tartalmát (non-volatile)
- elektromosan írható/törölhető végtelen sokszor (EEPROM, FLASH)
- UV-fénnyel törölhető (EPROM)
- nagy átmeneti ellenállás 2-4 kΩ
- nagy parazita kapacitása
- További gyártástechnológia (sok maszk-réteg alkalmazása szükséges) - drága

Minimum hány lábra van szükség egy 1024*4-es 1 CS-sel és közös I/O-val rendelkező RAM megvalósításához

CS =>SRAM-ról van szó
4 db: I/O
10 db: cím [2^10=1024]
1 db R/W
1 db VDD (betáp)
1 db GND (föld)
1 db CS
Összesen: 4+10+4= **~18**

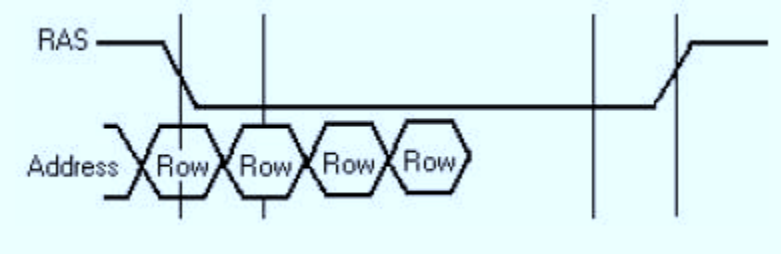
Példa: 64x4-es SRAM felépítése

- Feladat: Mekkora **lábszámot** kell biztosítani az SRAM működéséhez?
- **6+4+1(CS)+1(R/W)+ 1(Vdd)+1(Gnd) = 14**
- Tehát összesen 14 lábra (pin) van szükség!
- (64 rekesz, 4bit/szó)



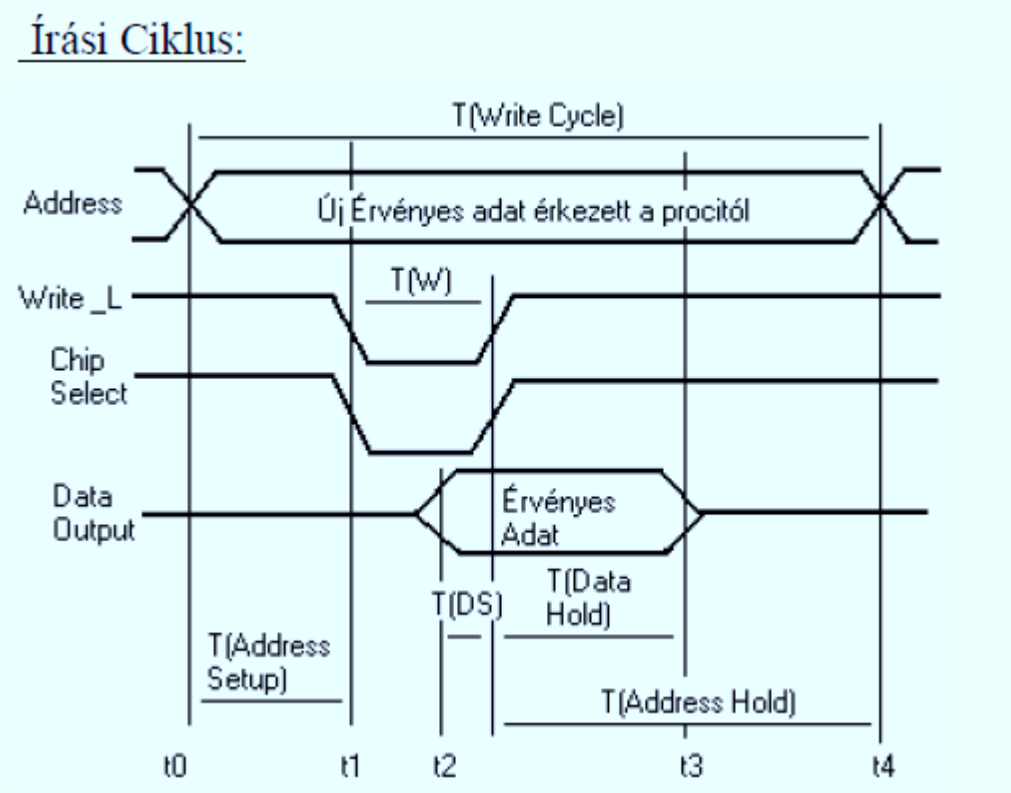
23

Ismertesse a dinamikus memória frissítési folyamatát (idődiagram)



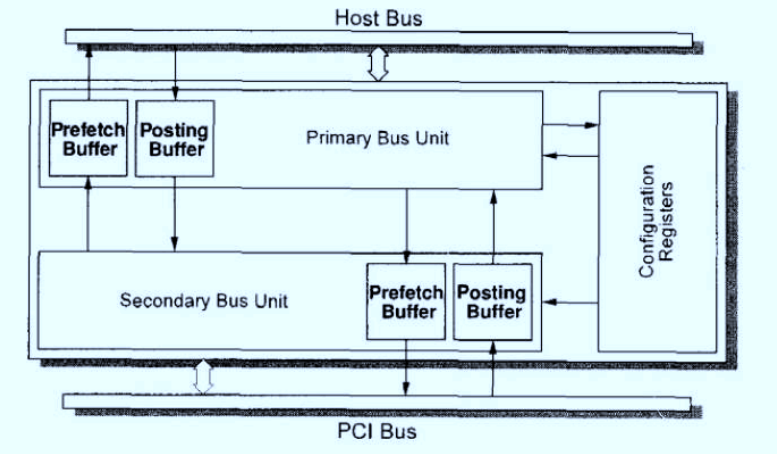
Frissíteni kell 2-10 ms-onként mert kondin van az info melynek feszültsége exponenciálisan csökken ergó kisül.

Ismertesse a statikus memória írási folyamatát (idődiagram)



Milyen átviteli technológiát használnak a PCI busz jelei és milyen jelcsoportok találhatók a PCI csatlakozón, továbbámilyen csatlakozó felépítése?

szinkron, PCI 32 bites multiplexált adat/cím jeleket alkalmaz
A csatlakozó 188 lábú, oldalanként 94-94 lábbal. Sok GND található a zavarások elkerülése végett. A tápról jön 12V, 5V, 3.3V-os jel is.



Mekkora minimális hozzáférési idő az ötödik memóriabank eléréséhez, ha t[TRANSLATION]=20 ns, t[HOZZÁFÉRÉS]=20 ns
MO: t[HOZZÁFÉRÉS] + 5 * t[TRANSLATION] = 20ns + 5*20ns = 120ns

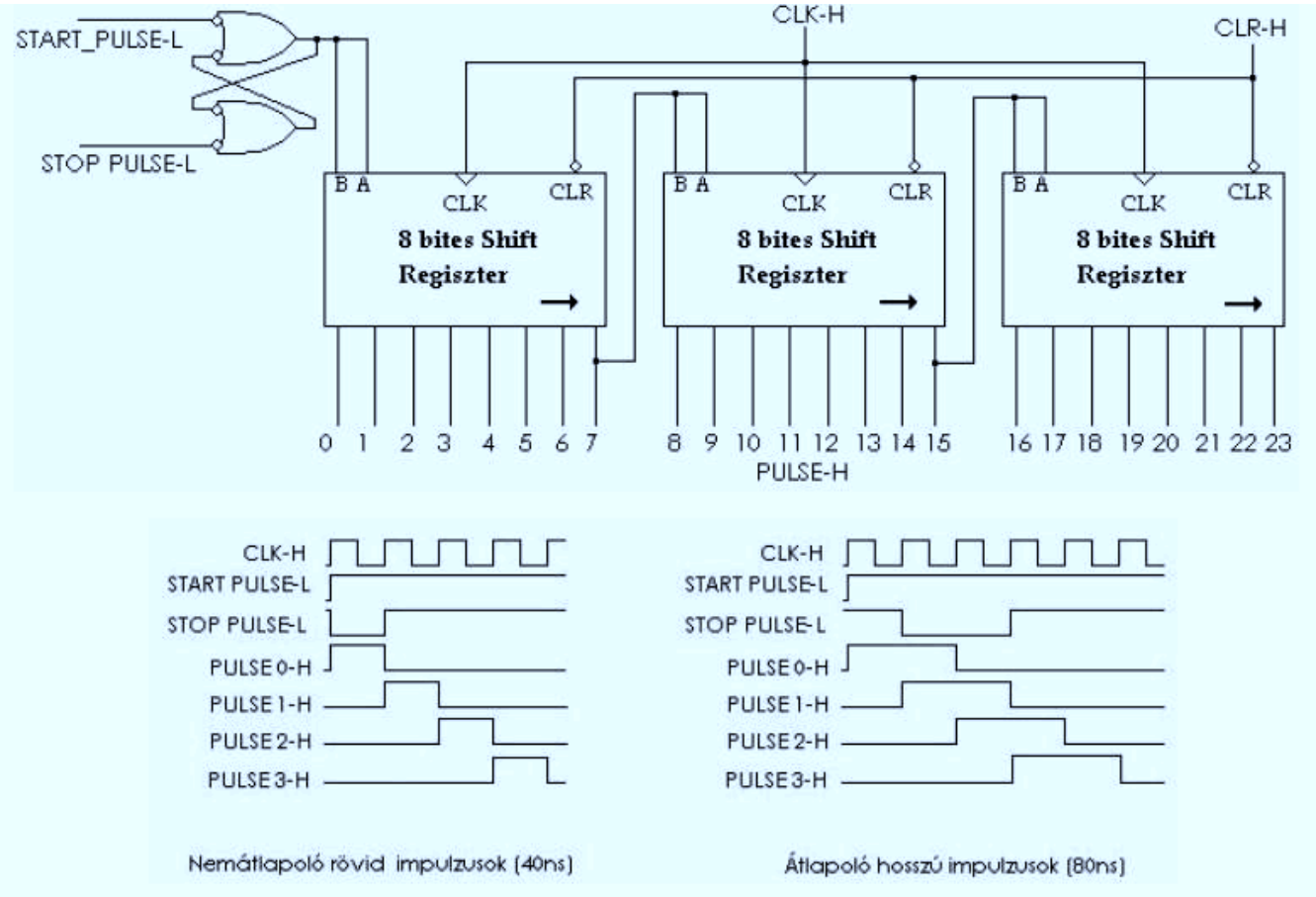
Hogyan használható a memórából több független program?

minden programnak saját logikai/bázis címtartománya van, amellyel hivatkozni lehet rá.

Vezérlőegységek feladata, fajtái. Present state register, next state logic előállítása multiplexeres módszerrel. Rajzoljon példát és valósítsa meg!

Sorrendi vezérlő egységek megvalósítása shift regizterekkel, átlapoló impulzusok elve. Rajzoljon egy példát egy vezérlő jel megvalósítására!

A shift regiszteres megvalósítás az egyedi késleltetéses módszerhez áll közel. Adatút diagrammot használunk a vezérlőjelek azonosítására, folyamat diagrammot a regiszter-transzferek (RTL) ábrázolására, míg idő diagrammot a vezérlőjelek kölcsönhatásának leírására. Kapcsolásunk 3db 8-bites shift-regiszterrel (74ALS164) állítja elő az impulzusokat (közvetve vezérlő jeleket is). Inicializáláskor a kívánt impulzus előállítását a START_PULSE_L beállításával érhetjük el, amelyet a teljes folyamat végéig „L" alacsony aktív szinten tartunk. A következő órajelciklusban a PULSE_0-H jel állítódik be magas jelszintre rövid 40 ns-os impulzus ideig. A STOP_PULSE-L vezérlőjelet a PULSE_0-H jel negálásával kapjuk meg (abban az esetben, ha egy ciklus, azaz 40ns ideig tart). Az órajel minden egyes felfutó élére shiftelődik az impulzus. Ekkor nem lapolódnak át, mivel egymás utáni 40ns –os részekből állnak össze, és a megfelelő PULSE_XX kimeneti vezérlőjelek OR kapcsolatából képződnek. Azonban bizonyos esetekben impulzushibák ún. tüskék keletkezhetnek (pl. ha az egyik jel alacsony szinten marad, a másik viszont magas jelszintre vált). Ezeket a nem megfelelő jelváltásokat vagy SET-RESET flip-flopok-kal küszöbölhetjük ki, vagy pedig alkalmazni kell az átlapoló impulzusok technikáját. Ezt az idődiagramot a jobboldali ábra mutatja. Két egység hosszú impulzusokat (80ns) egyszerűen létrehozhatunk a PULSE_1-H jel és az invertált STOP_PULSE-L jel OR kapcsolatával (a bal oldali ábra jeleiből !). Az így kapott impulzus mentes lesz a hibáktól, és kiküszöbölhetők a hazárdjelenségek



I/O csatornák. Milyen fajtáik vannak? Hogy működnek?

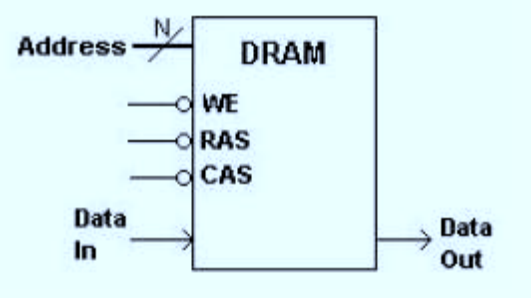
Aszinkron:
Ebben az esetben a Master-Slave modulok nem használnak közös órajelet (CLK-t): ha az egyik egység végez, elindít egy másik tranzakciót. A Master, mint kezdeményező, aktiválja a megfelelő vezetékeket. A Slave válaszol. A Slave címének azonosítására egy külön egység szolgál. Vezérlőjelekkel zajlik a kommunikáció: írási / olvasási folyamatokat különböztetünk meg. Ilyen vezérlőjelek lehetnek a READ, REQUEST, ACKNOWLEDGE (ha READ-H = Master olvas a Slave-ről, ha READ-L= Master ír a Slave-re) + egyszerűen felépíthető, nem kell (közös) órajel, gyors átvitelt biztosít (modulok sebességétől függ) - beépített késleltetések, csak korlátozott hosszúságú buszok /vezetékek lehetnek

Szinkron:
Közös órajellel működnek a buszra csatlakozó egységek. Tehát a műveleti időt az órajelciklus határozza meg. Mivel nincs szükség párbeszédre (pl. handshaking), gyorsabb az aszinkron működésénél. Jel: Master=Commander, a Slave=Responder.- Az órajelet mindig a leglassabb (legtávolabb lévő) egységhez kell igazítani.

talán ezek lehetnek a fajták:
I/O interfacek fejlődése:
1. CPU által közvetlenül vezérelt
2. I/O kontrollerrel vezérelt modul: a CPU programozott I/O utasításokat használ, (még megszakítások nélkül)
3. Interruptos kontroller: megszakítás vezetékekkel egészül ki
4. DMA- közvetlen memória elérésű, saját cpu-val rendelkezik, tehermentesíti a CPU-t, perifériát közvetlenül eléri
5. I/O csatornák: az eszköz saját utasításkészlettel rendelkezik, I/O program a memóriában van
6. I/O processzorok: saját elkülönített RAM-al rendelkeznek (nem a központi memóriát használja)

DRAM struktúrája és működése

MOS technológiával készülnek. A tápfeszültség alatt is frissíteni kell 2-10 ms-ként, mivel idővel elvesztik tartalmukat. Nagyobb kapacitású, mint az SRAM, de méretben is nagyobb, bonyolultabb és lassabb is. A hozzáférési idő kétszer nagyobb a memória R/W ciklusidejénél: 2*T(R/W Cycle)=T(Access Time). Integritási sűrűsége 4x jobb. (IRAM: az időzítő elektronika a DRAM-ra van integrálva, úgy látszik, mintha SRAM lenne, a frissítést tekintve). A CS=Chip Select jelet két részre osztottuk fel: RAS=sorkijelölő, és CAS=oszlopkijelölő komponensekre.



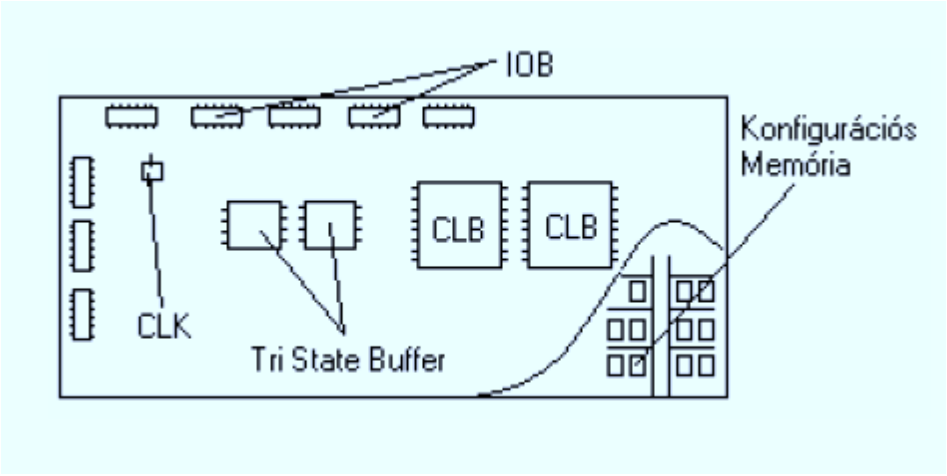
Sorolja fel a cache-k fajtáit. Rajzolja fel részletesen a legrugalmasabb logikai felépítést!

- közvetlen leképezésű: a központi memória minden egyes blokkja csak egy sorban (rögzített / direct sorindex) szerepelhet a cache-ben. Alacsony hit rate.
- teljesen asszociatív: a központi tár bármelyik blokkja a cache-be bárhova betölthető, a blokk címe pedig bekerül a cache toldalék részébe. Gyors és Drága hw-t igényel.
- n-utas csoport asszociatív: Olyan cachetároló, amely több, "n" sorból álló csoportokra van osztva, és az egy csoporthoz tartozó cachetárolórész önmagában teljesen asszociatív tárolóként működik. Annak megállapítása, hogy egy cachebe írandó blokk melyik csoporthoz tartozik, a közvetlen leképezésű cachehez hasonlóan, a memóriacímből képzett indexszel kerül meghatározásra. (előző két módszer ötvözete)

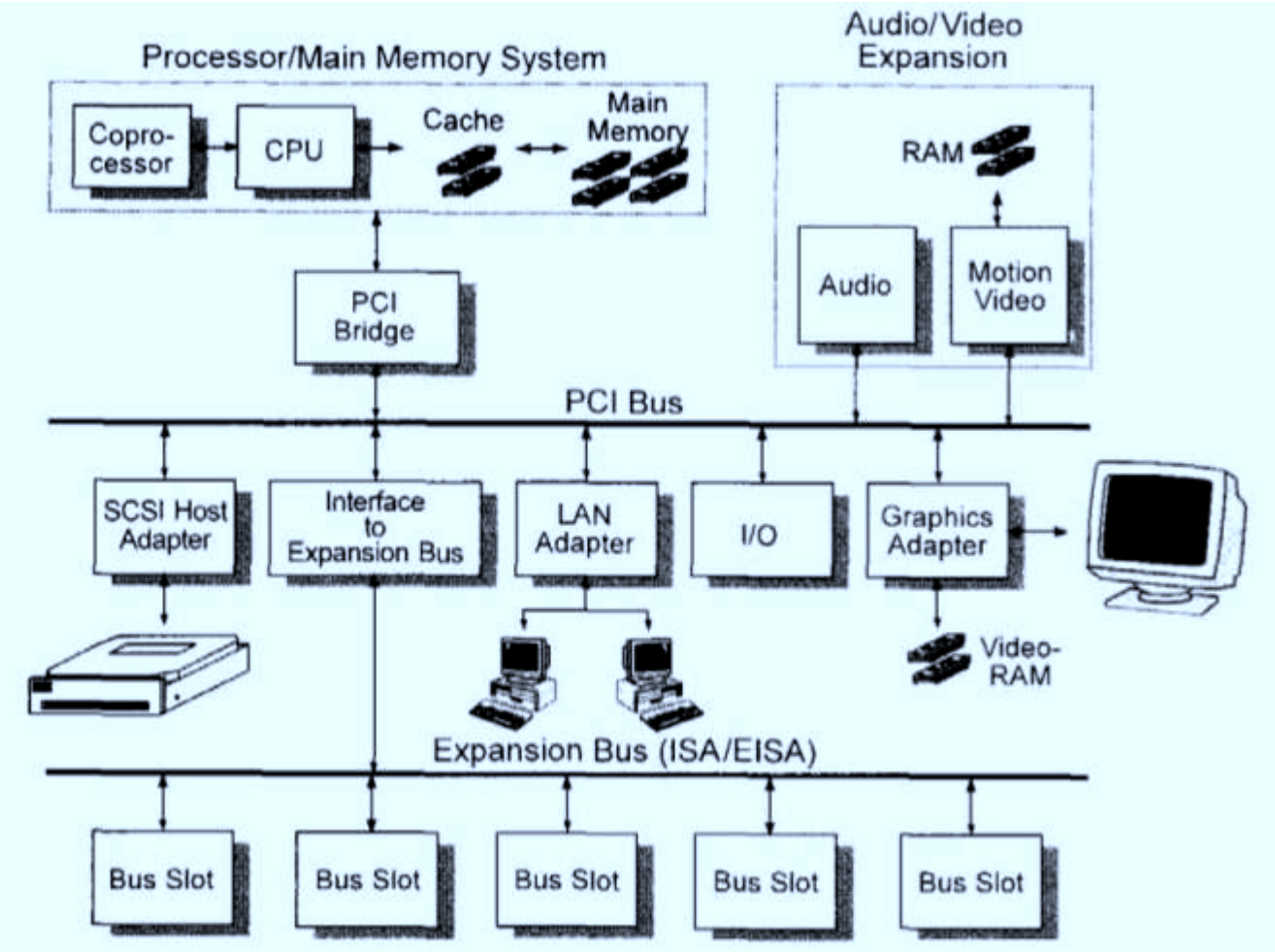
Itt valszeg az n-utas csop. asszoc cache-t kell felrajzolni, mert az a két másik ötvözete.

Rajzolja fel a Xilinx FPGA-k fejlesztői rendszerének(??) felépítését!

Tipp:
Ez az ábra a fejlesztői rendszer amiről beszél, de nem tuti ezt még meg kéne kérdezni...



Rajzoljon fel egy PCI buszos számítógép architektúrát EISA és SCSI busszal



(habár,ennek az árbának nem kell az egésze**)**

Többprocesszoros rendszerek csoportosítása (Flynn), példák (??)

Gúgliból...

http://en.wikipedia.org/wiki/Flynn%27s_taxonomy

Flynn-féle osztályzási modell:

- *SISD (Single Instruction Single Data):* Az végrehajtó egység egy órajel ciklus alatt egy utasítást végez egy adaton. Klasszikus szekvenciális rendszer. Az ilyen elvű egységeket *skalár* feldolgozónak nevezzük. - Példa: Neumann-féle számítógép.
- *SIMD (Single Instruction Multiple Data):* Az végrehajtó egység egy órajel ciklus alatt egy utasítást végez több adaton. Az ilyen elvű egységeket *vektor* feldolgozónak nevezzük. - Példa: Vektorprocesszor, Tömbprocesszor.
- *MISD (Multiple Instruction Single Data):* Az végrehajtó egység egy órajel ciklus alatt több utasítást végez egy adaton. Létezése vitatható, hiszen eleve furcsa, hogy több utasítást adunk ki ugyanarra az adatra. Némi megkötéssel besorolhatók ide a csővezetékes gépek.
- *MIMD (Multiple Instruction Multiple Data):* Az végrehajtó egység egy órajel ciklus alatt több utasítást végez több adaton. Az ilyen elvű egységeket *szuperskalár* feldolgozónak nevezzük. - Példa: Multiprocesszor, Multiszámítógép.

29 bittel mekkora memóriát lehet megcímezni?(12 kbyte?)

Innen kell valószinűleg kiszürni:) :
A lapok mérete 512 byte-os, a laptábla 2.048 bejegyzést (lapot) tartalmaz.
A megcímezhető max memória 512x2048= 2^20. (1Mbyte) 20 bit a virtuális címtartomány.
A cím lapcímből és lapon belüli eltolási címből tevődik össze. Egy lap 512 (2^9) byteos = 9 bitet használunk (ADDRESS (0:8)) a lapon belüli hely azonosítására. További „fennmaradó” (2048 = 2^11) 11 bitet a megfelelő lap címének azonosítására (ADDRESS (9:19)). A laptábla tehát 2.048x17 nagyságú, 17 bittel címezhető (17 = 15 bit a címzésre + 2 státuszbit). A címfordítás csak 15 bites. Az egyik státuszbit jelzi, hogy a lap a főmemóriában található-e, ill. a másik jelzi, hogy a betöltés óta módosult-e a lap.

hogy következik a pirosból a zöld?? :(

Jellemezze a CISC számítógépet, és adjon legalább egy ilyen processzor típust

CISC: *Complex Instruction Set Computer*

-*Nagyszámú utasítás-típust*, és címzési módot tartalmaz, egy utasítással több feladatot végre tud hajtani. Változó méretű utasításformátum miatt a dekódolónak először azonosítania kell az utasítás hosszát, az utasításfolyamból kinyeri a szükséges információt, és csak ezután tudja végrehajtani a feladatát.

-A korai gépeknek egyszerű volt a felépítése, de bonyolult a nyelvezete. Összetett problémákat kívántak vele megoldani, a gépi kódnál magasabb szintű nyelven. *Szemantikus* rés= a gépi nyelv és felhasználó nyelve közötti különbség. Ennek áthidalására új nyelvek születtek: Fortran, Lisp, Pascal, C, amelyek bonyolultabb problémákat is egyszerűen képesek voltak kezelni. Komplexebb gépek születtek, amelyek gyorsak, sokoldalúak voltak.

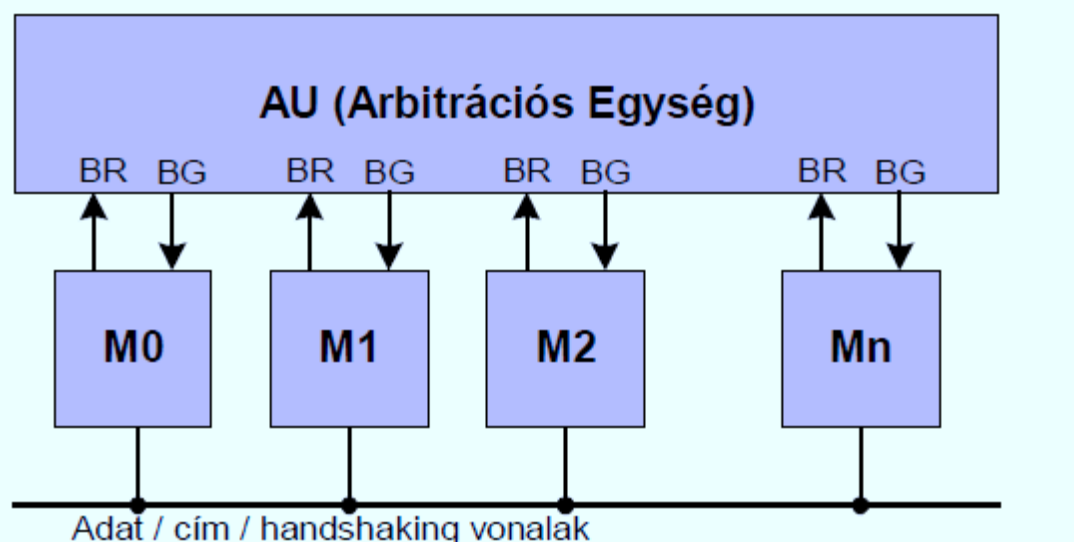
-*Compiler = Fordító:* a bemenetén a probléma felhasználói nyelven van leírva, míg a kimenetén a megoldást gépi nyelvre fordítja le.

-Megfigyelték, hogy a processzor munkája során a rendelkezésre álló utasításoknak csak egy részét használja (20%-os használat, az idő 80%-ában).

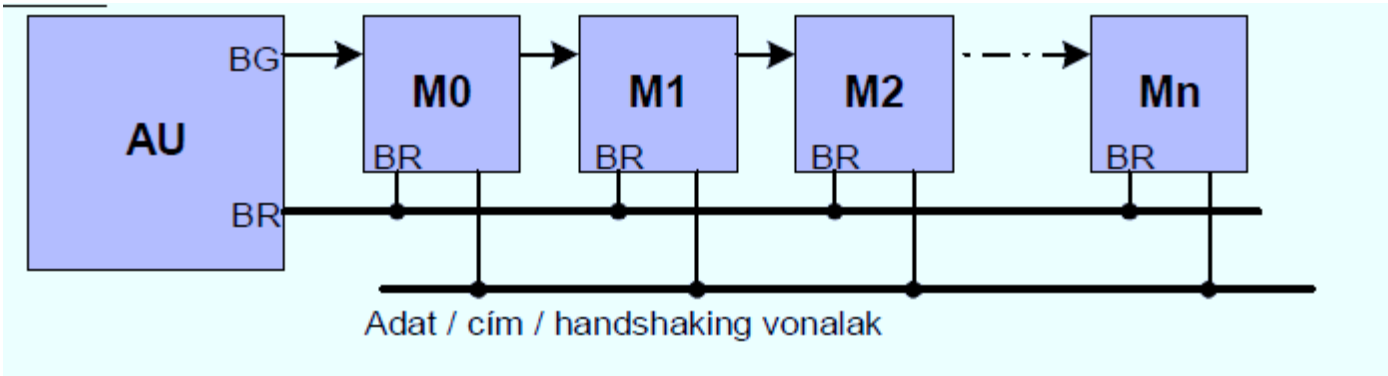
Ugyanaz a komplex program, függvény *kevesebb utasítássorozattal* megvalósítható. Memória, vagy regiszter alapú technikát használ

Például: Digital VAX, Intel IA-32, Motorola 68000

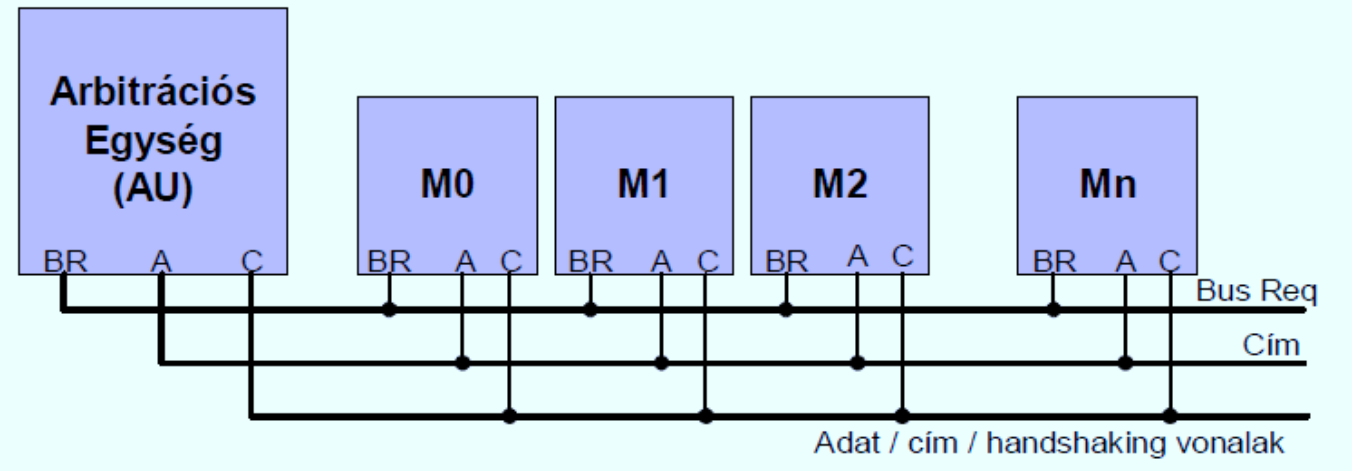
Rajzolja fel és röviden ismertesse az arbitrációt és fajtáit



Párhuzamos: Ez a leggyorsabb módszer, minden M-nek kitüntetett, egyenrangú kapcsolata van az AU-val (két vonalon: BG és BR-en). Az arbitráció lehet (i) first asserted/first served (FIFO), vagy (ii) round robin, vagy (iii) prioritásos alapú. Ezek a módszerek többfajta lehetőséget biztosítanak mind egyszerű, mind pedig komplex esetben. Az AU vezérli a közös buszon az adatátvitelt. Az adat-cím-handshake vonalat a kijelölt master kezeli, az adás befejeztével pedig kiadja ismét a BR-jelet. Hátránya, hogy drágább, mint a többi módszer (a párhuzamos ágak miatt minden M-hez 2 vonal kell), és az M-ek száma korlátozott.

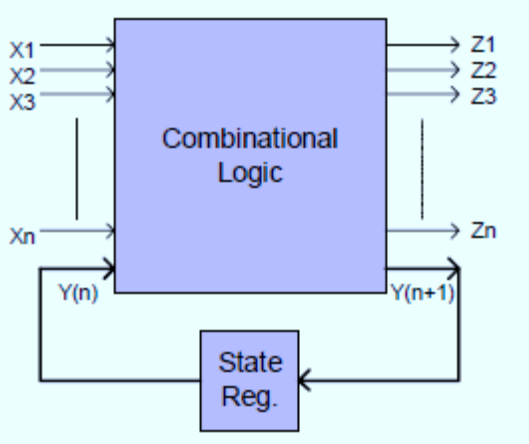


Soros (daisy chain): A BG vonal sorosan van kötve, míg a BR vonal mindegyik M-hez csatlakozik. Az AU nem ismeri, hogy pontosan melyik M kívánja elérni a buszt, ezáltal az átvitel leegyszerűsödik: csupán azt tudja, hogy bizonyos ciklusonként BG-jelet kell kibocsátania. A soros csatlakozás miatt a legelső Master (Mo) rendelkezik a legnagyobb prioritással (fizikai prioritás), így ha ő igényelt, minden esetben megkapja a buszt. Bármennyi eszközt is sorba köthetünk, nincs felső korlátja. Hátránya, hogy az arbitrációs idő (ha a legutolsó M igényel és az előtte lévők nem) egyenes arányban van a sorba kapcsolt M-ek számával.



Lekérdezéses(polling): Mindegyik Master egy közös BR buszon igényelhet, és az AU eldönti, hogy melyik Master-től kapta a kérést. Annak a címét az address vonalra rakja. Minden ciklusban megnézi az igényléseket, és a legmagasabb prioritással rendelkezőt fogadja el. Itt is többféle prioritásos módszer megvalósítható: pl. (FIFO, round robin). Hátránya, hogy nagyobb az időszükséglete a párhuzamosnál, így ritkábban alkalmazzák (pl: I/O kérések arbitrációjánál), segítségével a processzor egy program futtatásakor lekérheti az I/O eszközöket, megszakítást engedélyezhet.

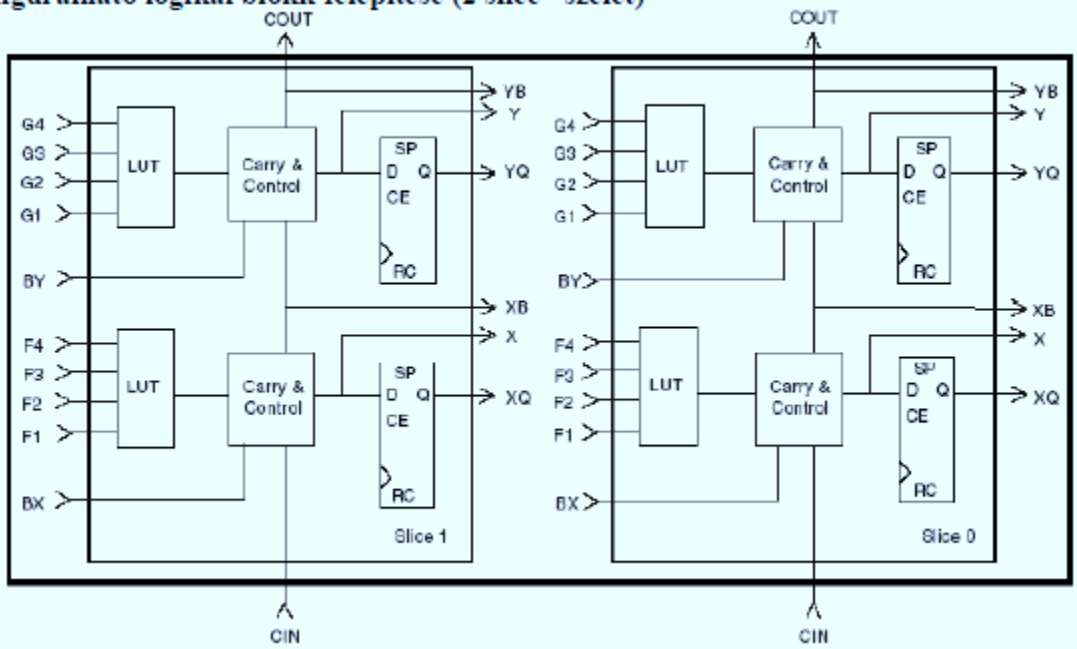
Definiálja a Mealy autómata modellt



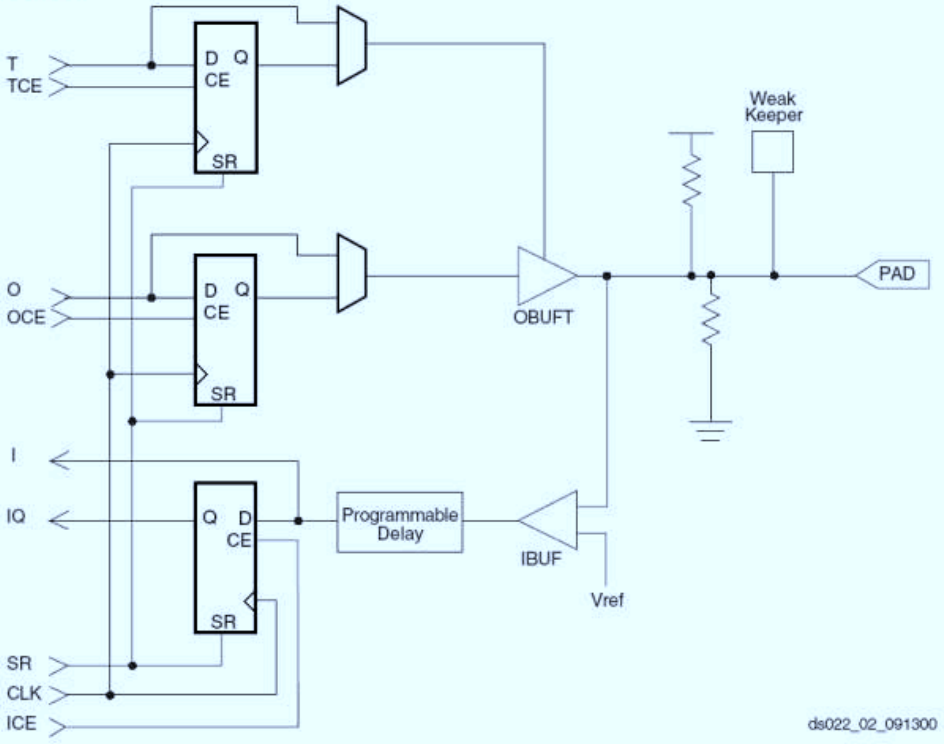
A sorrendi hálózatok egyik alapmodellje. (A KH. = kombinációs hálózatok esetén a kimenetet csak a bemenetek függvényeként kaptuk meg). Azonban a valóságban minden egyes eszköznek van bizonyos minimális késleltetése, és a kimeneten az eredmény véges időn belül jelenik meg, vagy a korábbi értékek visszacsatolódnak a bemenetre. A sorrendi hálózatokat megvalósító vezérlő egységeknél kimenetek nem csak a bemenetek pillanatnyi, hanem a korábbi állapotoktól is függenek. (N memóriaelem esetén 2N lehetséges állapotot tud a rendszer elfogadni.) Három halmaza van: X – a bemenetek, Z – a kimenetek, Y – az állapotok halmaza.
Visszacsatolni az állapotregisztert a késleltetés miatt kell. Két leképezési szabály van a halmazok között:
$\delta(Xn,Yn) \rightarrow Yn+1$: következő állapot meghatározása
$\mu(Xn,Yn) \rightarrow Zn$: kimenet meghatározása
Azonban problémák merülhetnek fel az állapotok és bemenetek közötti szinkronizáció hiánya miatt (változó hosszúságú kimenetet kaphatunk).

Rajzolja fel a Xilinx FPGA felépítését (CLB és I/O)

Virtex CLB: Konfigurálható logikai blokk felépítése (2 slice –szelet)

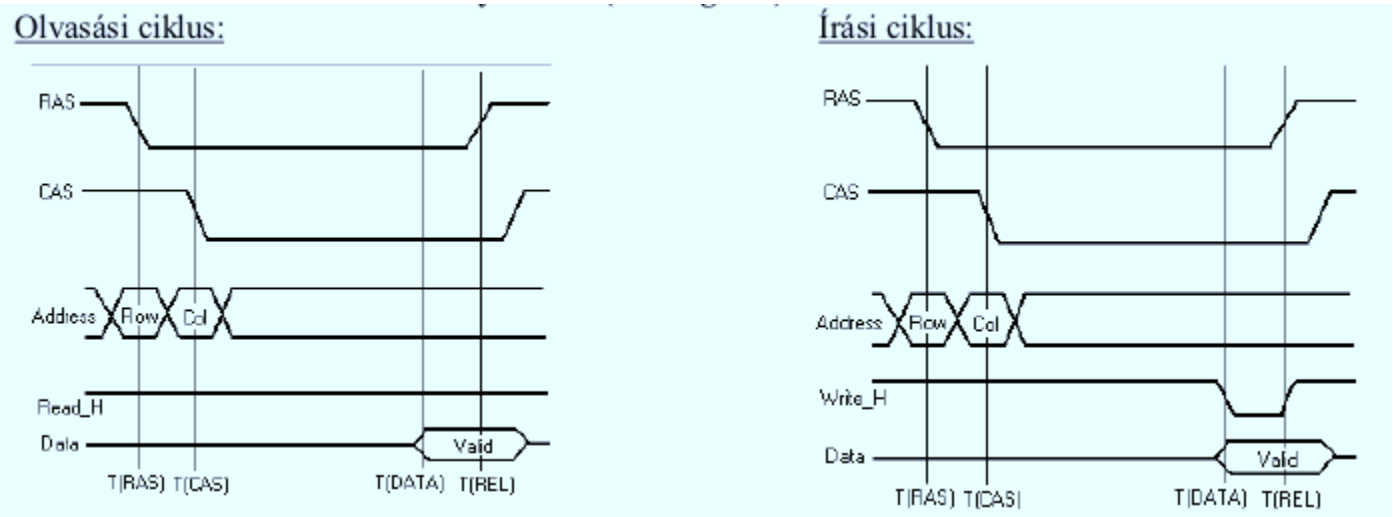


Virtex IOB - I/O Blokk felépítése:



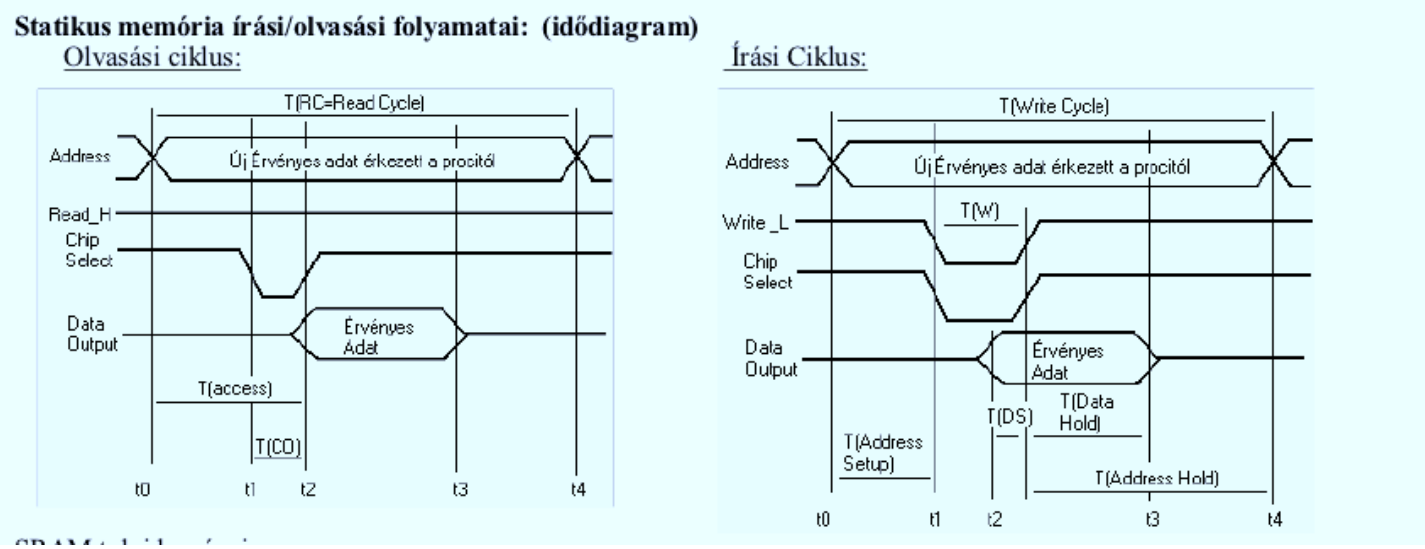
ismertesse a <u>dinamikus</u> memória írási-olvasási folyamatát (idődiagram)

<u>Olvasási ciklus:</u>
A DRAM-oknak több címvonalra van szüksége, mivel a címek két részre, egy sor- és egy oszlop-azonosítóra (RAS-CAS) vannak felosztva. Miután a RAS jelet alacsony szintre állítja be (lefutó élre vezérelt) a sorcím megjelenik a címvonalon. Ezt kitartja rövid ideig, majd a CAS jelet állítja be alacsony szintre, és az oszlopcím jelenik meg. Ezután válik elérhetővé a memória (setup time). Majd a memória elérési idejétől (T access time) függően kis idő múlva az adat érvényessé válik (Valid). Amikor az RAS felszabadul (T (REL)) a kimeneten az adat visszatér a háromállapotú (tri-state)feltételhez.Read_H végig magas aktív,hiszen olvasási ciklust hajt végre.
<u>Írási ciklus:</u>
Az olvasástól csak annyiban különbözik, a WE (write enable), Write_L jel engedélyezi az írást, (az írás lefutó élre történik).



Ismertesse a statikus memória olvasási folyamatát (idődiagram)

Az információ a tápfesz alatt is megmarad, nem kell frissíteni. SRAM esetén T(R/W Cycle) ≈T(Access Time) közel azonos. Megvalósítható bipoláris (0,1) SRAM cellával, nMOS tranzisztorokkal, vagy CMOS tranzisztorokból (6 tranzisztor). Kisebb kapacitású memória építhető fel SRAM-ból, de gyorsabb a DRAM-nál, mivel tápfeszültség alatt sem kell frissíteni, viszont nagy a fogyasztása. Integritási sűrűsége 4x rosszabb a DRAM-nál. Tápfeszültség kikapcsolásával elveszti a tartalmát.
Felhasználása: cache memóriákban, digitális oszcillátorokban, logikai analizátorokban, operatív tárakban (memória).



hasonlítsa össze a szegmentálást és a lapozást!

LAPOZÁS:
Lap: a program és a memória is fizikailag egyenlő méretű darabokra van osztva (fizikai határok). A főprogram, szubrutinok, globális adatok, lokális adatok, verem mind-mind egy-egy azonos méretű lapnak felelnek meg a memóriában. A lapozás gazdaságtalan, mert a kis méretű adatot is egy nagy méretű lapba kell tölteni, nem használja ki a rendelkezésre álló lapméretet. Fizikai Cím=Lap Báziscíme + Lap eltolás (offset) címe. A „+ „itt konkatenációt / összefűzést, nem pedig összeadást jelent. Ezáltal sokkal gyorsabb lesz a fizikai cím leképezése, mivel nem kell összeadót használni, a konkatenációt egyszerű huzalozással megoldhatjuk. A hexadecimális címek egy-egy számjegyét 4 bites bináris számmá kódoljuk, és összefűzzük. Pl: Logikai cím a virtuális memóriában: 2,344= 2. lap száma és a lap 344. rekeszcíme a virtuális memóriában. Legyen a 2. lap címe 2C00. Ezt kell összefűzni a 344-el.

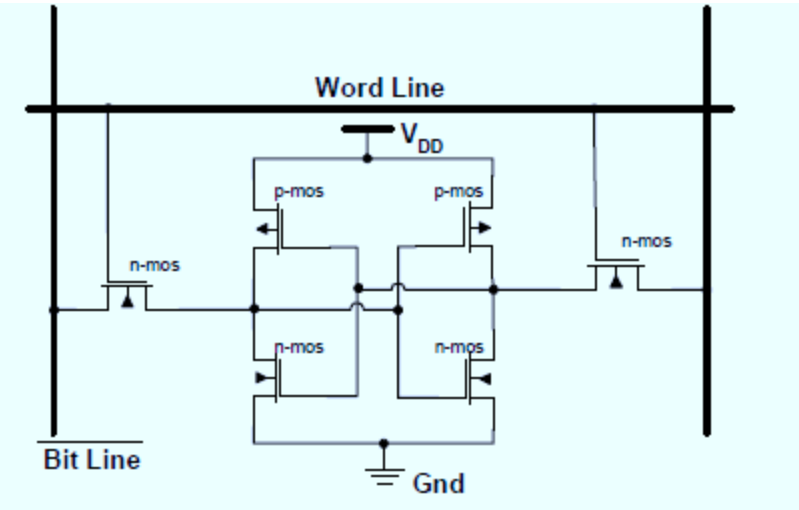
Az OS minden futó processhez hozzárendel egy laptáblát. A laptáblából olvassuk ki a lap számát (hexa érték) követően a hozzá tartozó lapcímet (hexa), továbbá az egyes lapok elérési információit: R/W/X: olvasható /írható /végrehajtható. A fizikai címet a CPU határozza meg. A lapok kis egységek, nagy laptábla kell, célszerű őket a memóriában tartani. Itt a lapcímeken az összeadás helyett konkatenációt használunk: időt nyerünk (a fenti lapcímben a legkisebb helyiértékű bitek csupa nullák, azokat nem kell hozzáadni; amikben pedig nem csupa nulla szerepel, azokat egyszerűen egymás után fűzzük). A lapok mérete általában mindig kisebb, mint a szegmensek mérete, ezért a kisebb méretű lapok száma több kell, hogy legyen (pl 1024)
SZEGMENTÁLÁS:
Szegmens: a program változó méretű logikai egységekre van feldarabolva. A program négy fő szegmense funkciójuk szerint:
- 0. szegmens: főprogramok (olvasható/végrehajtható),
- 1. szegmens: szubrutinok,
- 2. szegmens: csak olvasható adatok,
- 3. szegmens: olvasható / írható adatok.

A program a memóriát közvetlenül nem érheti el, csak a szegmensen keresztül.
FIZIKAI CÍM= Szegmens bázis címe + Szegmensen belüli eltolás (offset) címe. A „+” itt ténylegesen összeadást jelent! HW-es megvalósítás kell az összeadó miatt, és kell egy összehasonlítás, azért hogy nem címeztünk-e ki a szegmens címtartományán kívülre! Hiba esetén megszakítást (interrupt) küld. Szegmenstábla: A szegmenstáblából olvassuk ki a szegmens számát (hexa érték) követően a hozzá tartozó szegmens hosszát, szegmens címet (hexa), továbbá az egyes szegmensek elérési információit: R/W/X: olvasható /írható / végrehajtható.
Pl: Logikai cím a virtuális memóriában: 0,154 = 0 szegmens (főprogram) száma, és a szegmens 154 offsetcíme a virtuális
memóriában. Legyen a 0.szegmens hexa címe 1310. Ezt kell összeadni! a hexa 154-el.
ADDR(REAL)= ADDR(BASE)+ADDR(OFFSET) /Ez a formula általánosan igaz a lapozásra és szegmentálásra is !!!/

Rajzolja fel egy I/O interfész egység blokkdiagramját. Azonosítsa a jelentősebb blokkokat és a különböző adatátviteli módokhoz tartozó jeleket, regisztereket.

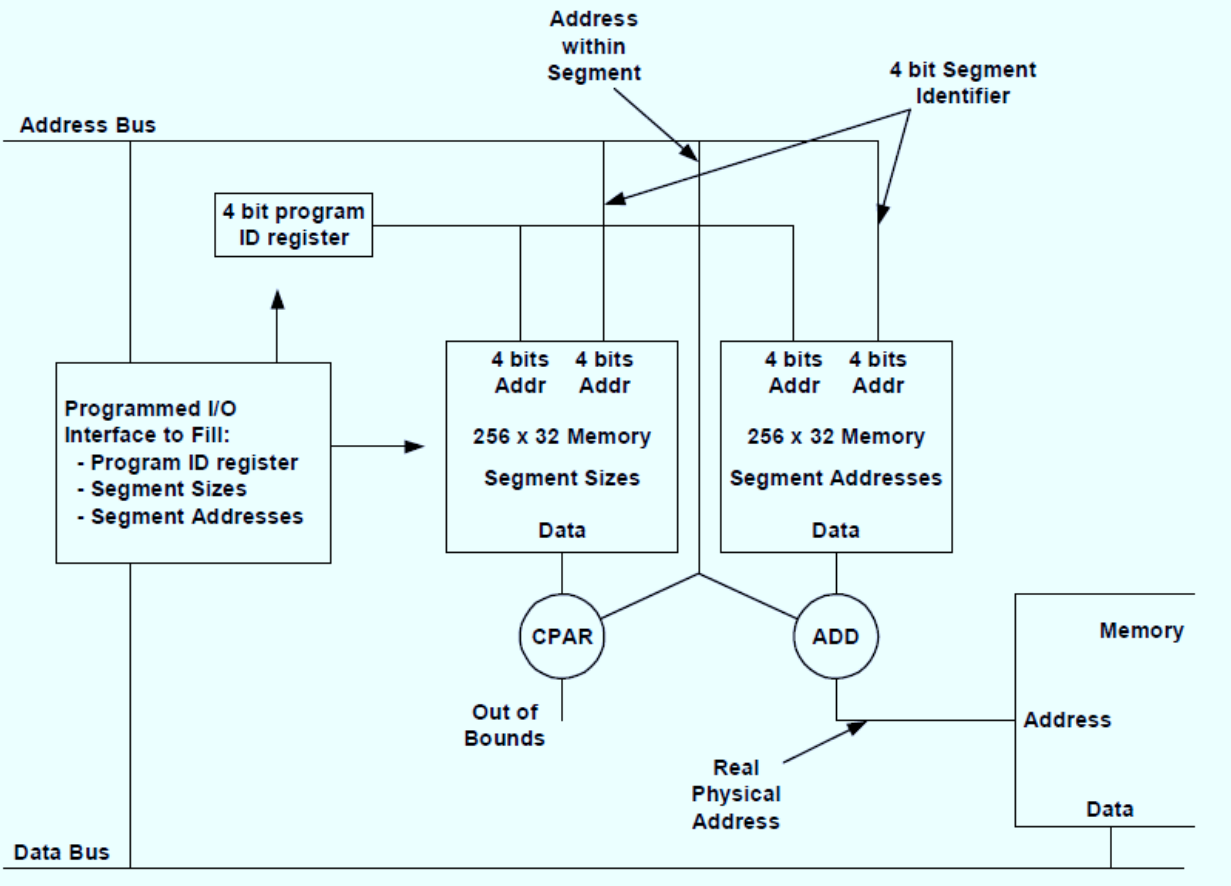
Ismertesse az SRAM-ok jellemőit tábláztos formában. Rajzolja fel felépítésüket, alapcelláját.

SRAM Cella:



Rajzolja fel egy szegmentált virtuális memória címzéses rendszerben címfordító hardver blokkdiagramját! a rendszerben egyszerre 64 program futhat és 32 szegmensl állhat (a hiányzó adatokat értelemszerűen válassza meg!)

DDR4



Mi a regiszter ablakozás? (rajz, magyarázat)

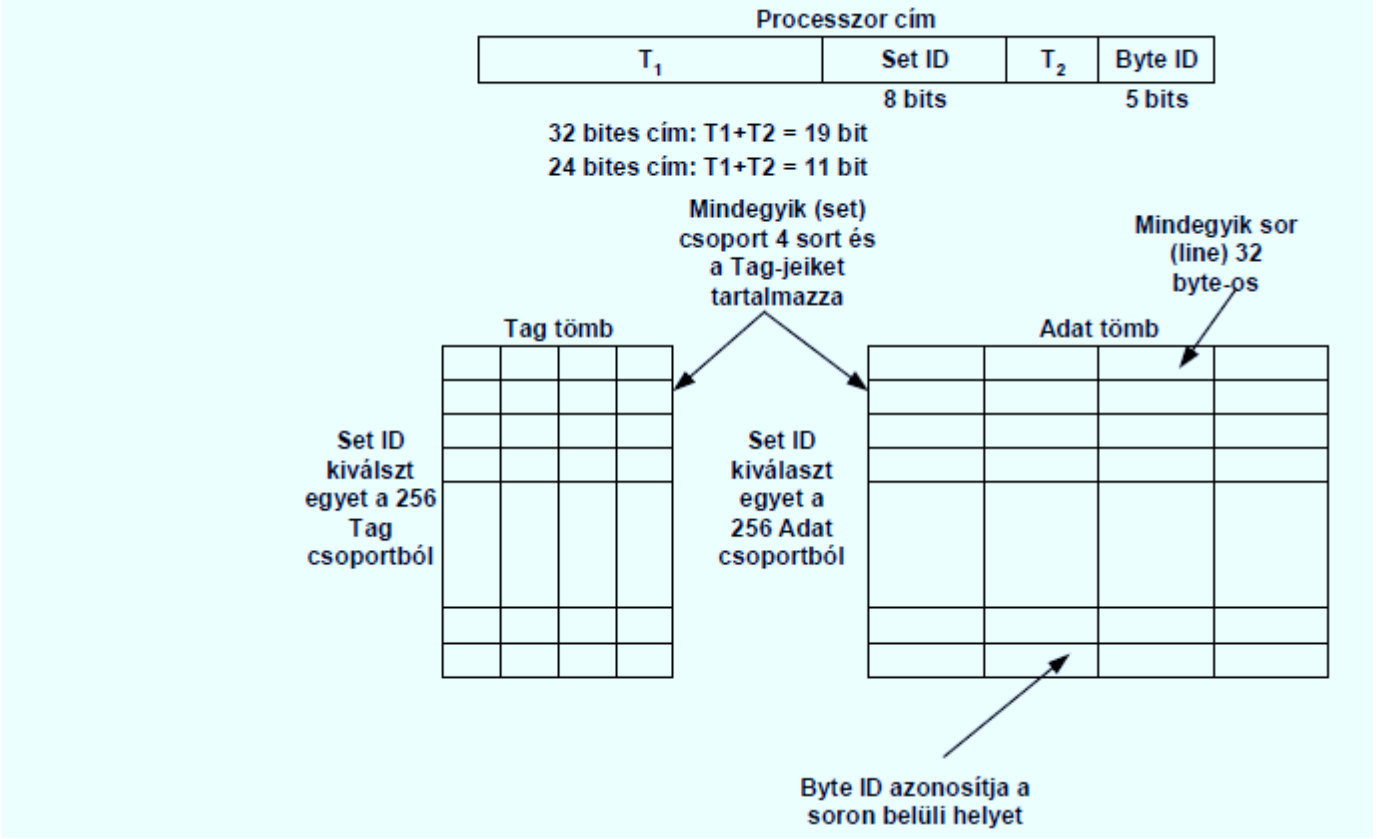
Regiszterablakozási technika: (RISC-I) Az eljáráshívás és az abból való visszatérés során a szubrutinok nagy processzoridőt használnak, mivel memóriában tárolnák az adatokat. A felhasznált processzoridő minimalizálására találták ki a regiszterablakos technikát.
A nagyszámú regiszterhalmazból egyidőben csak adott, korlátozott számú regisztert használ a rendszer. Ezt a korlátozott számú regisztert egy ablakkal azonosítják, és elkülönítik a többi regisztertől. Az ablak megváltozásakor egy pointer azonosítja, hogy az aktív regiszterek helyett új regiszterkészletet kell egy ablakba fogni. Amikor az ablak átlapolódik a szubrutinok, alprogramok között, akkor a paraméterek átadását globális regisztereken keresztül oldja meg, amelyek az összes szubrutin által elérhetőek. Az utasításkészletben 5 bittel azonosíthatjuk az utasításnál használni kívánt regisztert, ezáltal 32 különböző regiszter (R0-R31) címezhető meg (5 bit segítségével, mivel 25=32) Tehát vannak:
- közös (globális) regiszterek: R0-R9, minden szubrutin által elérhetők
- rutin specifikus regiszterek: R10-R31 mely további három részből áll (a regiszterek között történhet átlapolódás!)

a.) Alacsonyszintű regiszterek: R10-R15
b.) Lokális regiszterek: R16-R25
c.) Magasszintű regiszterek: R26-R31

Ez az eljárás mindaddig jól működik, ameddig a paraméterek száma kisebb a regiszterek méreténél, mivel nem igényel memória-intenzív stack műveletet.

Rajzoljon fel egy 2-way set associative cache-t a következő paraméterekkel: addres 32 bit, cache line: 32 byte, méret: 1024 kbyte.

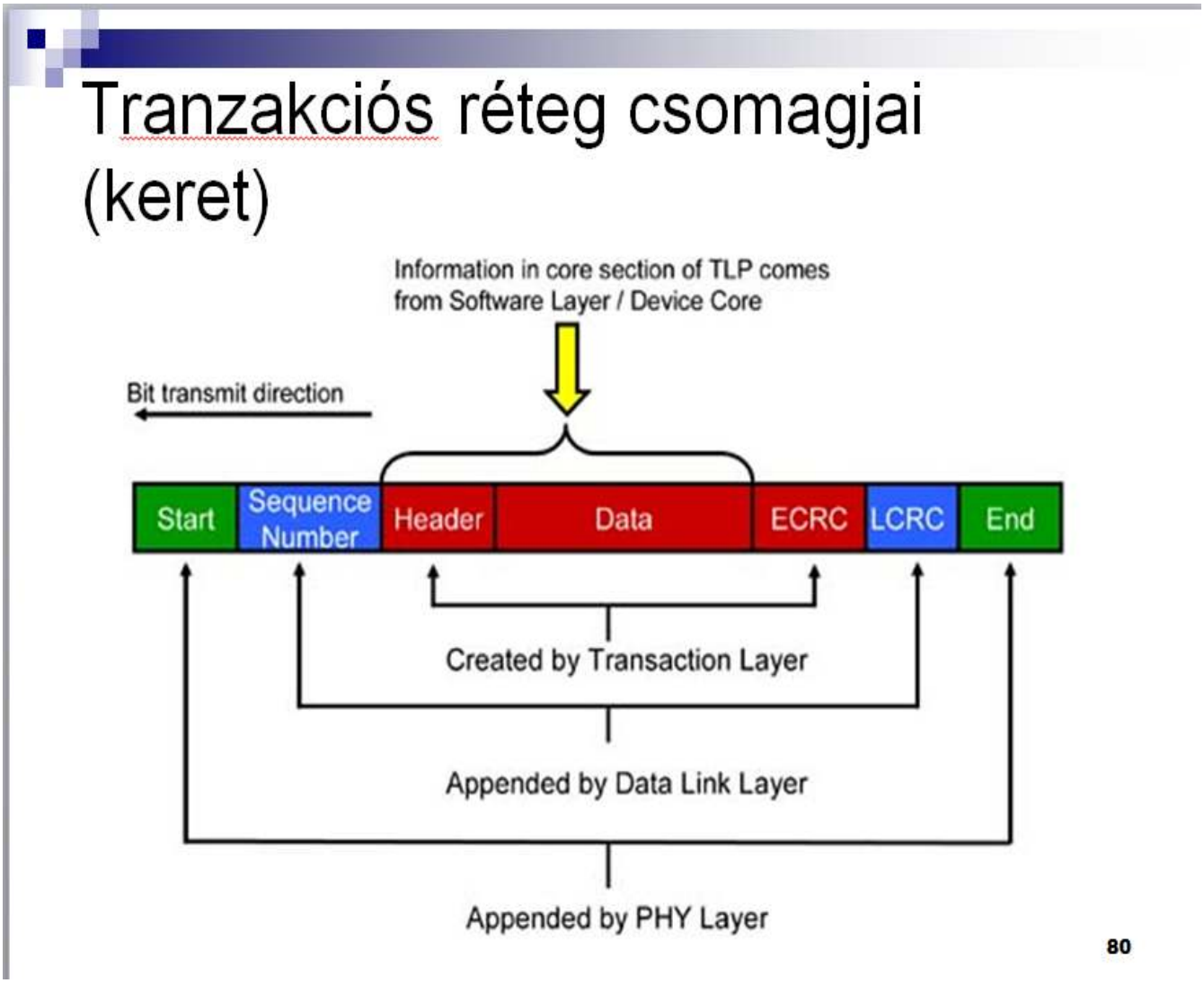
Pl. 4-utas csoport asszociatív cache:



Ez a négy utas innen kéne kitalálni: HAJRÁ!

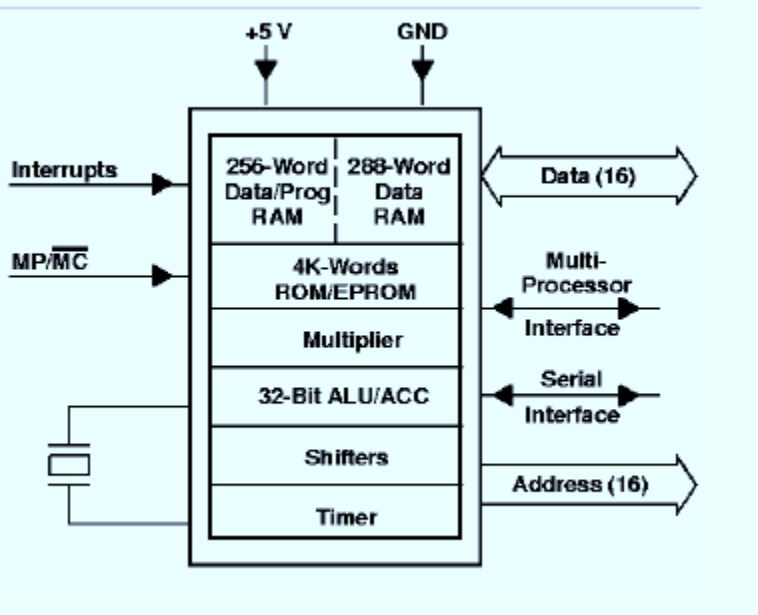
Adja meg egy PCI express Transacton Layer Packet formátumát!

Nincs benne!
A diasorban benn van...



Definiálja a Harvard-féle száítógép architektúrát
Szét van választva a memóriában az adatterület és a programterület.

Rajzoljon fel egy fxpontos DSP architektúrát és adja meg néhány jellemző tulajdonságát!



ezek közül lehet váláogatni:
40MHz frekvencia; fixpontos 16 bites, 68 lábú jelfeldolgozó processzor. CMOS technológiával készült->kis teljesítménydisszipáció (500mW); 80-100ns-os utasításvégrehajtási ciklusidő; 1 ciklus alatt szorzás/tárolás; 5V tápellátás; módosított Harward architektúra, két adatmemóriája van, amelyből az egyik variálható, 32-Bites ALU, Blokkos adatátvitel, Rugalmas,nagysebességű (12.5 MIPS), processzortömbbe szervezhető chip. Párhuzamos architektúra, hatékony utasításkészlet (133 utasítással), HW-esen implementált funkciók; AR0-AR7: 8 külső/kiegészítő tároló. Magas szintű C nyelven programozható

Ismertesse a szinkron dnamikus memória (SDRAM) írási-olvasási folyamatát (idődiagramok) Adja meg a memóriaelérés tipikus idejét!(??)

Miért támogatja a RISC architektúra a pipeline feldolgozást?

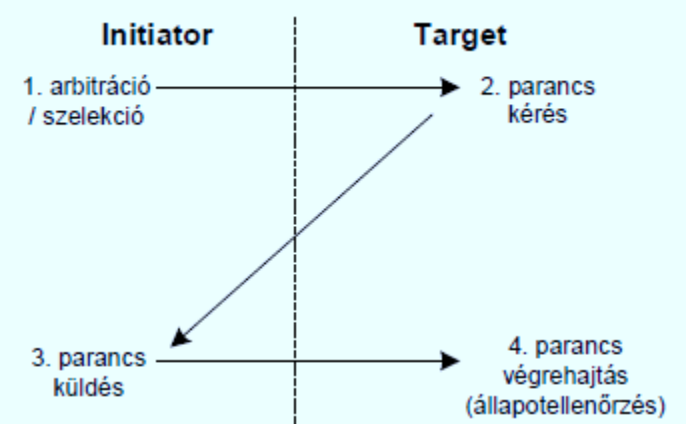
a soros feldolgozással ellentétben, egy feladat egymástól független részei (fázisai) a rendszer különböző pontjain egyszerre, egyidőben hajtódnak végre, ezáltal növekszik a sebesség. Az operandusokat gyors regiszterekben tároljuk. Azonos hosszúságú utasítások gyors F-D-E eljárása. Egy ciklusban egyszerre történik különböző utasításrészek Fetch-Decode-Execute feldolgozása (dekódolás, fetch rendkívül gyors).

Mert RISC architetúra esetén minden utasítás azonos hosszú, tehát lehet taktikázni azzal, hogy ha az egyik szál már végzett az első feladatrésszel, a második is elkezdheti stbstb, szép táblázatos kép volt róla.

Ismertesse a SCSI buszt! Hogyan zajlik a SCSI buson az adatátvitel?

SCSI= Small Computer Standard Interface: komplex, intelligens, sínorientált eszköz (merev-, hajlékonylemez, CD-ROM, szalagos egység, scanner...) interface. Különféle perifériák illesztésére, a processzor tehermentesítésére fejlesztették ki, az ooperációs rendszertől független felülettel rendelkezik. Sokoldalú eszköz, mivel nemcsak PC-s környezetbe, hanem UNIX munkaállomásokba és Apple-Mac gépekbe is beépíthető.

A saját processzorral és memóriával rendelkező intelligens SCSI egységek kezdeményezőként (initiator) és fogadóként (target) is működhetnek! Maximálisan 8 initiator, és 8 target működhet egyszerre. A kezdeményező adja ki az utasításokat, a cél pedig feldolgozza, és végrehajtja azokat.



A teljes SCSI busz fázisok a következők:
- busz szabad?
- arbitráció / szelekció (ki adjon?)
- üzenetküldés
- parancs elmegy
- adatkapcsolat
- állapotellenőrzés (status)
- üzenetzárás, kapcsolatbontás

ma burger kingben kajáltam

Aki eljutott idáig, annak 3 **ajándék:**

1. **első:)**
2. **második;D** http://imm.io/oR79
3. **harmadik: https://www.youtube.com/watch?v=3lkgXNUZurM**

:-)