2014 ZH A csoport:

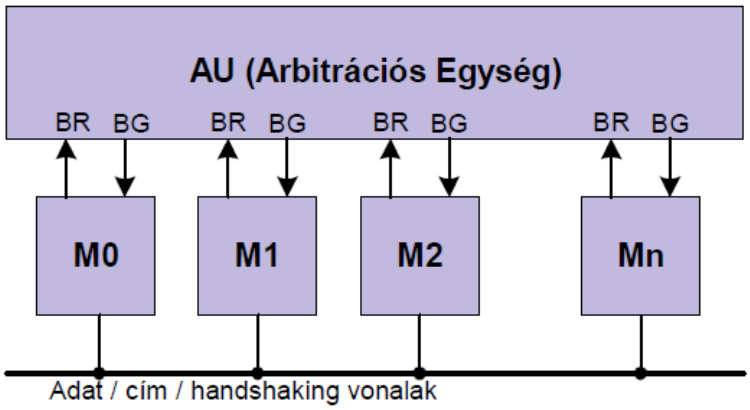
1. feladat: szinkron írási protokoll

Közös órajellel működnek a buszra csatlakozó egységek → a műveleti időt az órajelciklus határozza meg. (Master/Slave) Az órajelet mindig a leglassabbhoz egységhez kell igazítani.
írási ciklus: (CPU → MEM)
n: Commander arbitrációja/kiválasztása
n+1: átvitel megkezdése, memóriához adat érkezik
n+2: memória dönt az adatok elfogadásáról, válaszol a Commandernek
n+3: Slave/Responder nyugtát küld vissza

2. feladat: Arbitráció és fajtái

I/O művelet esetén több Master is egyszerre akarja a busz irányítását. Az arbitrációs eljárás egy döntési folyamat, amely az aktuális adatátvitel befejezése előtt eldől, hogy melyik következő master adhat.
Az arbitrációnak három típusa van: párhuzamos, soros, lekérdezéses.
Párhuzamos:

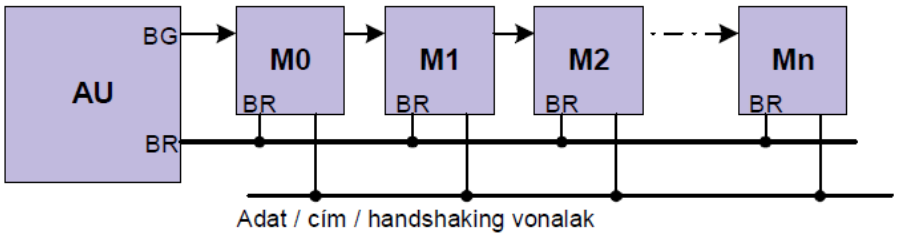
a.) Párhuzamos:



Ez a leggyorsabb, legdrágább módszer és az M-ek száma korlátozott. Az arbitráció lehet: FIFO, round robin, prioritásos alapú.

Soros:

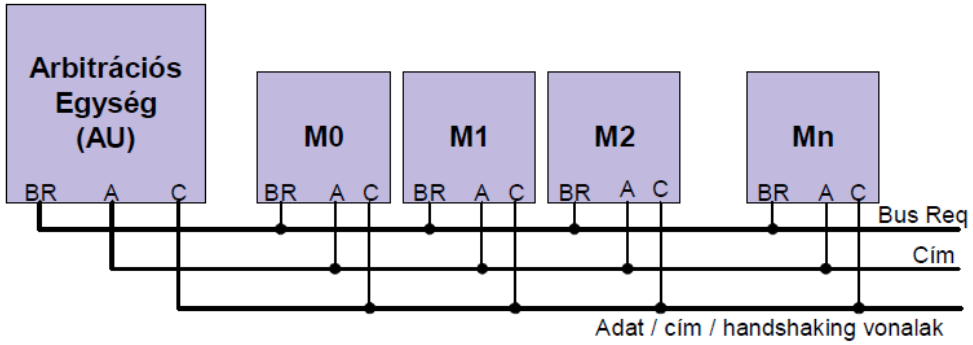
b.) Soros „Daisy Chain”:



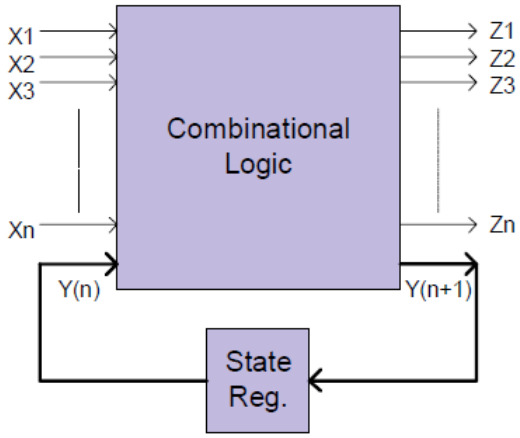
Bármennyi eszközt sorba köthetünk (nincs felső korlát). Az arbitrációs idő egyenes arányban van a sorba kapcsolt M-ek számával. BG vonal sorba van kötve. M_0 legmagasabb prioritásút kérdezzük meg először,igényelt-e. Ha igen, minden esetben megkapja a buszt.

Lekérdezéses:

c.) *Polling / Lekérdezéses technika:*



Mindegyik Master közös BR buszon igényelhet. Az AU minden ciklusban megnézi, ki a legnagyobb prioritású igénylő (FIFO, round robin).



3. feladat: Mealy automata modell definiálása

A sorrendi hálózatok egyik alapmodellje. A kimenetek nem csak a pillanatnyi bementtől, de a korábbi állapotoktól is függnek. Három halmaza van: bemenetek – X, kimenetek – Z, állapotok halmaza – Y

Két leképezési szabály van a halmazok között:

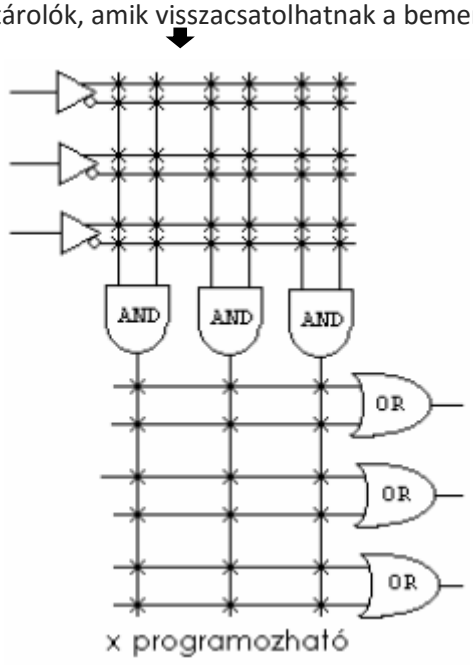
$\delta (X_n, Y_n) \rightarrow Y_{n+1}$:következő állapot meghatározása

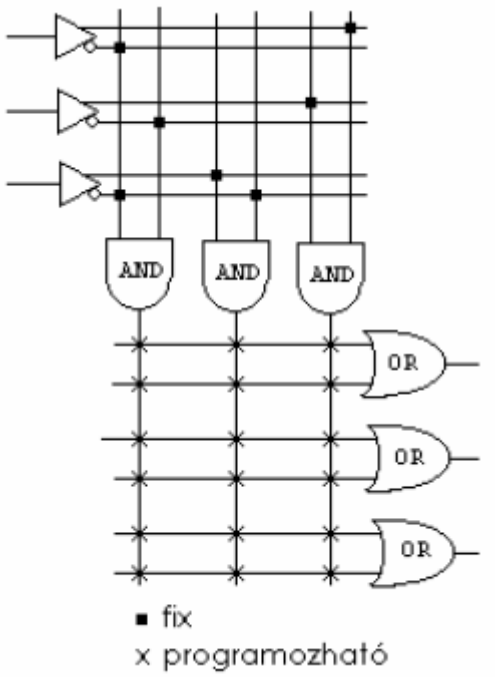
$\mu(X_n, Y_n) \rightarrow Z_n$: kimenet meghatározása

4. feladat: Vezérlőegység programozható makrocellákkal:

PROM: AND fix, OR programozható. OR kapuk kimeneténél tároló regiszterek, amik visszacsatolódnak a bemenetekre. →

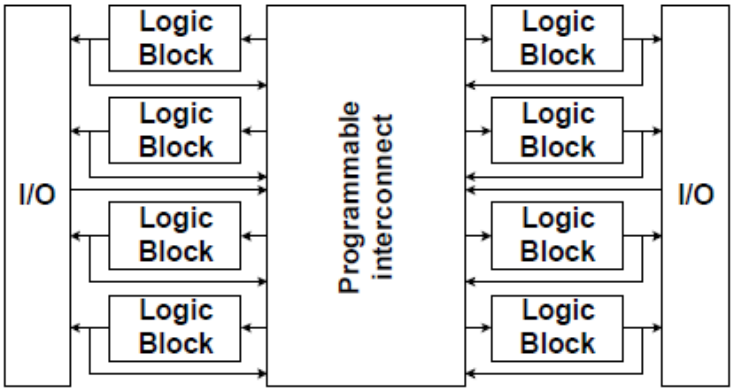
PLA: AND és OR programozhatók. OR kapuknál D-tárolók, amik visszacsatolhatnak a bemenetekre.

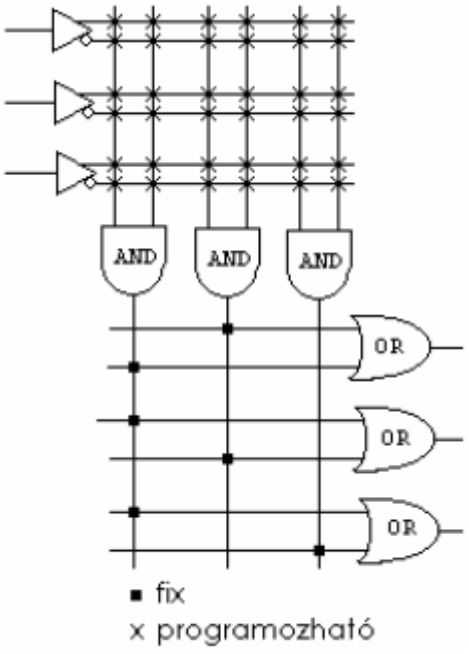




PAL: AND kapuk programozhatók, OR kapuk fixek. Gyorsabb, mint a PLA. Kimeneten D-tárolók, visszacsatolhatnak. →

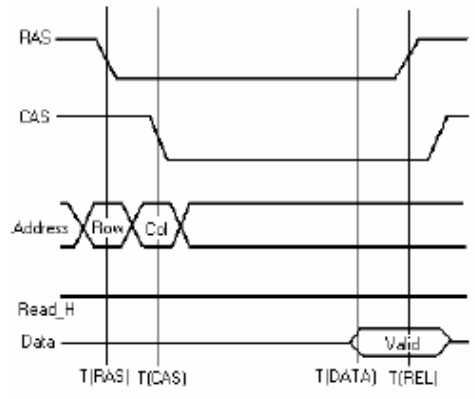
CPLD: Logikai blokkban: PAL/PLA, Regiszterek.
Interconnect lehet: Teljes vagy Részleges. ⬇



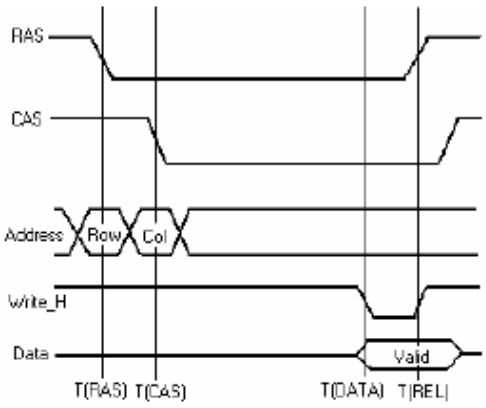


6. feladat: Dinamikus memória írási/olvasási folyamata (időDiagram)

Olvasási ciklus:



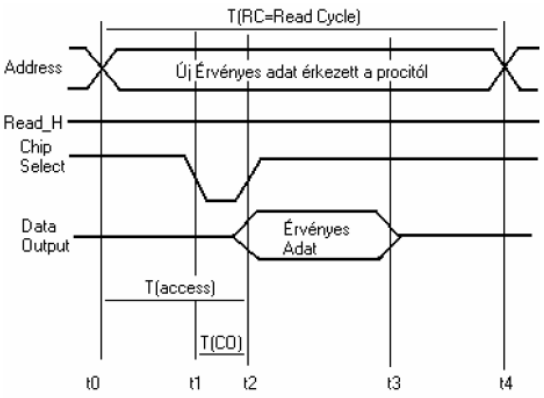
Írási ciklus:



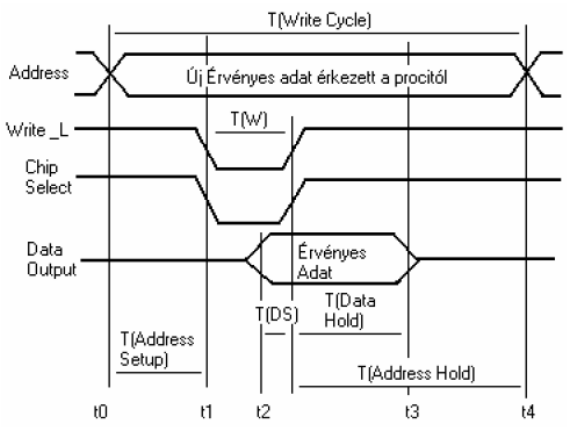
(sor cím: RAS, oszlop cím: CAS)

7. feladat: Statikus memória írási/olvasási folyamata (időDiagram)

Olvasási ciklus:

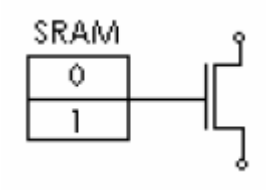


Írási Ciklus:

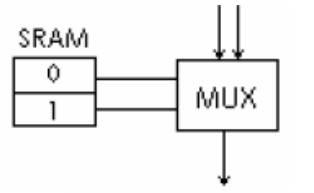


8. feladat: Hogyan programozható egy VLSI?

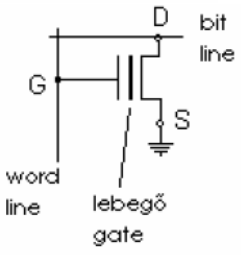
SRAM-os:
- végtelenszer újra programozható
- SRAM-ot nem kell frissíteni
- Táp kikapcsolásakor a SRAM elveszti tartalmát
- Bekapcsoláskor fel kell programozni
- Sok tranzisztor, nagy méret



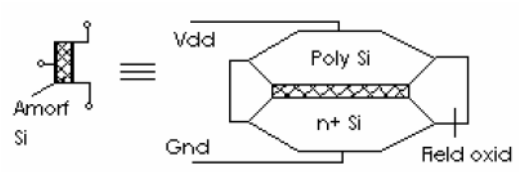
Muliplexeres: A SRAM-ban tárolt 0 agy 1 bitet használunk a Multiplexer vonalának kiválasztásához.



Floating Gate:
- többször törölhető UV fénnyel
- kikapcsoláskor is megőrzi a tartalmát

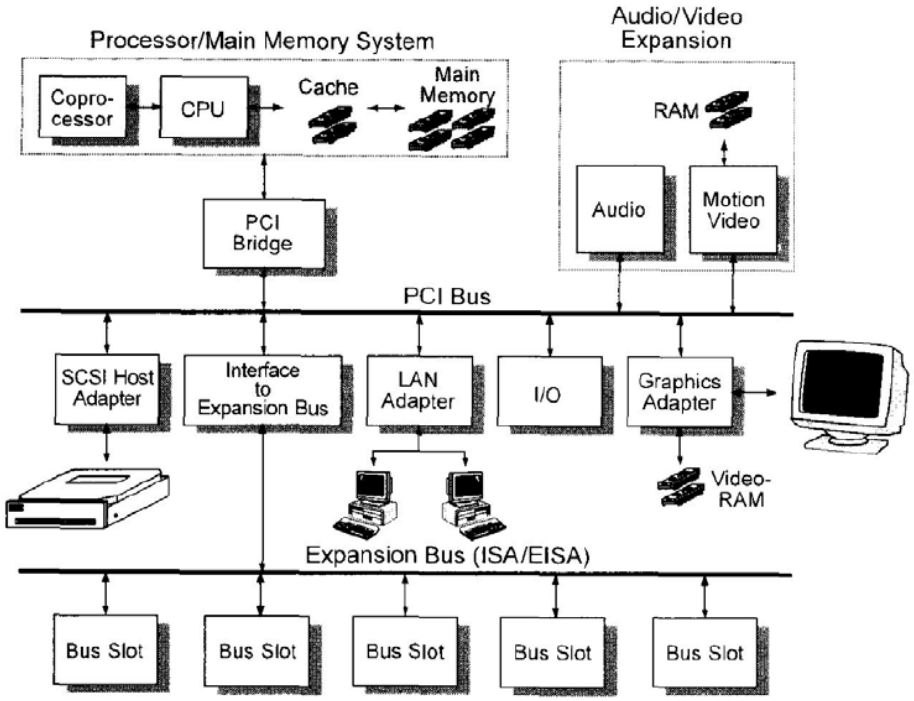


Antifuse:
- egyszer programozható
- kis méret
- drága előállítani

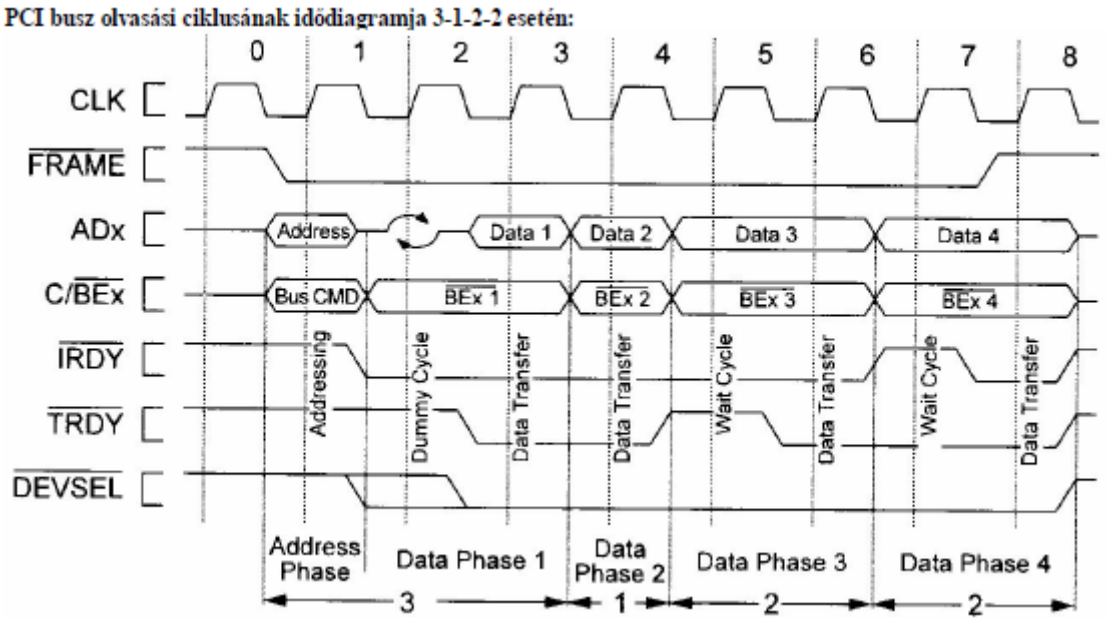


EPROM/EEPROM/FLASH-os:
- Floating Gate-es technológiát használ.
- megőrzi a tartalmát
- végtelen sokszor írható/törölhető (EEPROM, FLASH)
- UV-fénnyel törölhető (EPROM)
- drága

9. feladat: PCI buszos PC blokkdiagram



10. feladat: PCI burst read idődiagram



3-1-2-2=3 órajelciklust vár az első adatig, utána 1-2-2 ciklusonként jönnek az adatok.(közben wait ciklusok vanak)

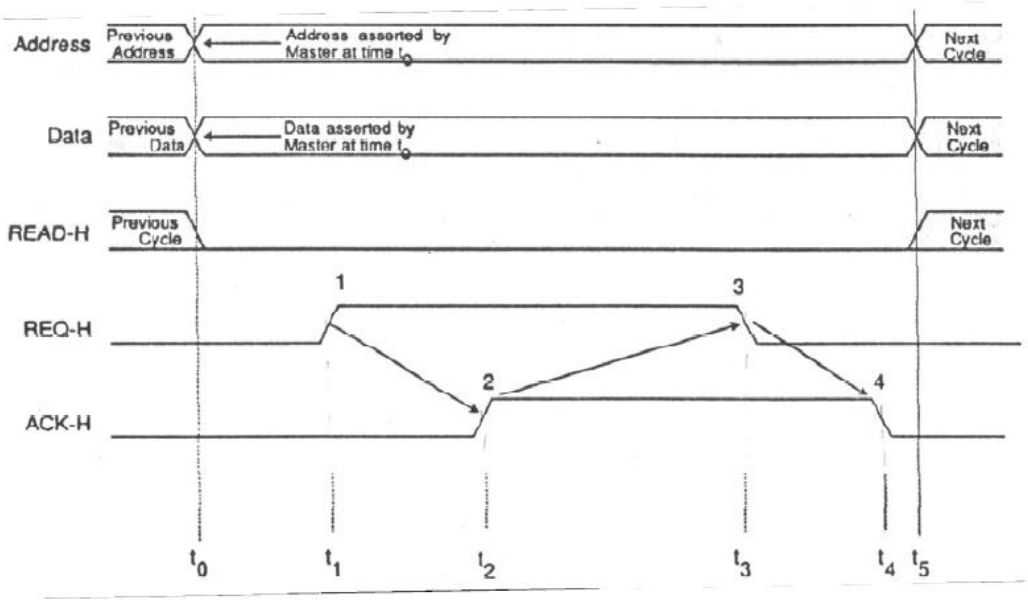
2014 ZH B csoport:

1. feladat: szinkron olvasási protokoll

Olvasási ciklus: (CPU ← MEMÓRIA)
n: Commander arbitrációja, CPU busz lekérése
n+1: kérés inicializálása (CPU olvasási kérést küld, memóriához adat érkezik)
n+2: Válasz a Commandernek, memória dönt az olvasási igényről
n+3: Responder döntése (Memória nyugtát küld CPU-nak)

2. feladat: Aszinkron Hand Shaking folyamat írás esetén

Write: Master ír Slavennek.
t0- ban master megadja a kívánt slave címét.
t1-ben master beállítja a request vonalat (minden slave veszi, de csak a toban kiválasztott válaszol)
t2-ben slave nyugtázza, hogy megkapta az adatokat
t3-ban master is megkapja a slave nyugtáját, felszabadítja a request vonalakat
t4-ben slave érzékeli és

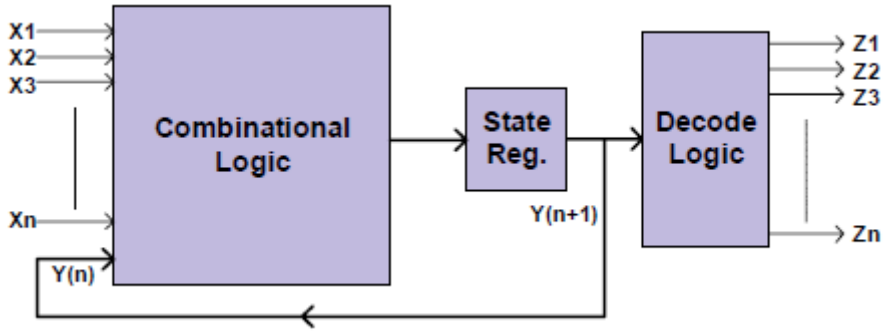


felszabadítja a nyugtázó vonalat
t5-ig a master a címvonalat tartja még (a cím esetleges megváltozása miatt)

3. feladat: Moore automata modell

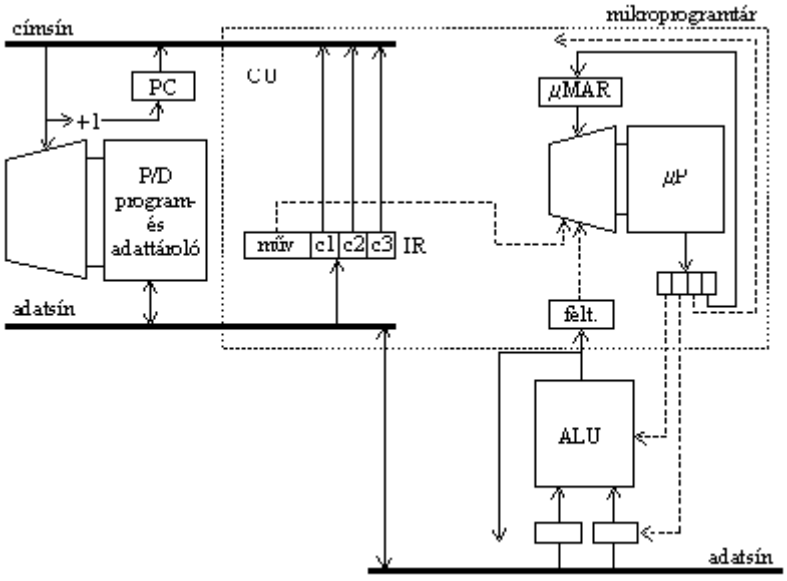
Kimenetek közvetlenül csak a pillanatnyi állapottól függ. Három halmaza van: bemenetek – X, kimenetek – Z, állapotok halmaza – Y
Két leképezési szabály van a halmazok között:

$\delta (X_n, Y_n) \rightarrow Y_{n+1}$: következő állapot meghatározása
$\mu(Y_n) \rightarrow Z_n$: kimenet meghatározása



4. feladat: Mikroprogramozott vezérlő feladata

Feladat: a memóriában lévő gépi kódolású utasítások részműveletekre bontása és funkcionális egységek vezérlése
Mikroprogramozott: Az adatútvonal vezérlési pontjait (ROM) memóriából kiolvasott vertikális- vagy horizontális mikrokódú utasításokkal állítjuk elő.
Előny: Takarékos az erőforrásokkal a vertikális mp. (fogyasztás, bitek száma)
Hátrány: lassabb, egyszerre egy bitet állít be
Ábra: (?)



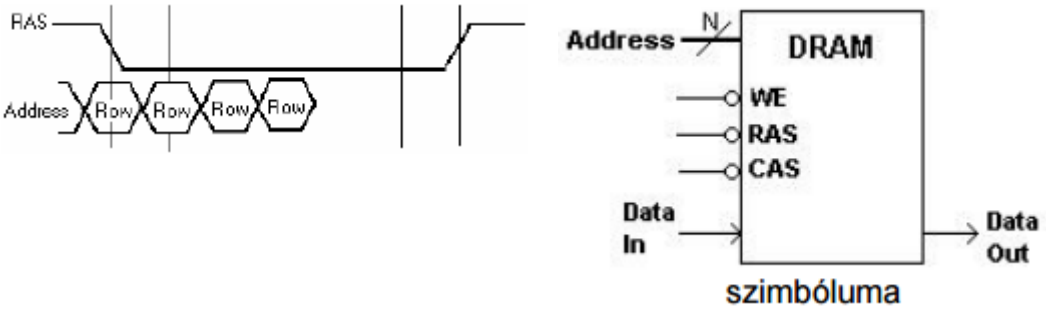
Mikroprogramozott vezérlési struktúra

5. feladat: 4A

6. feladat: 8A

7. feladat: Dinamikus memória frissítési folyamata (idődiagram)

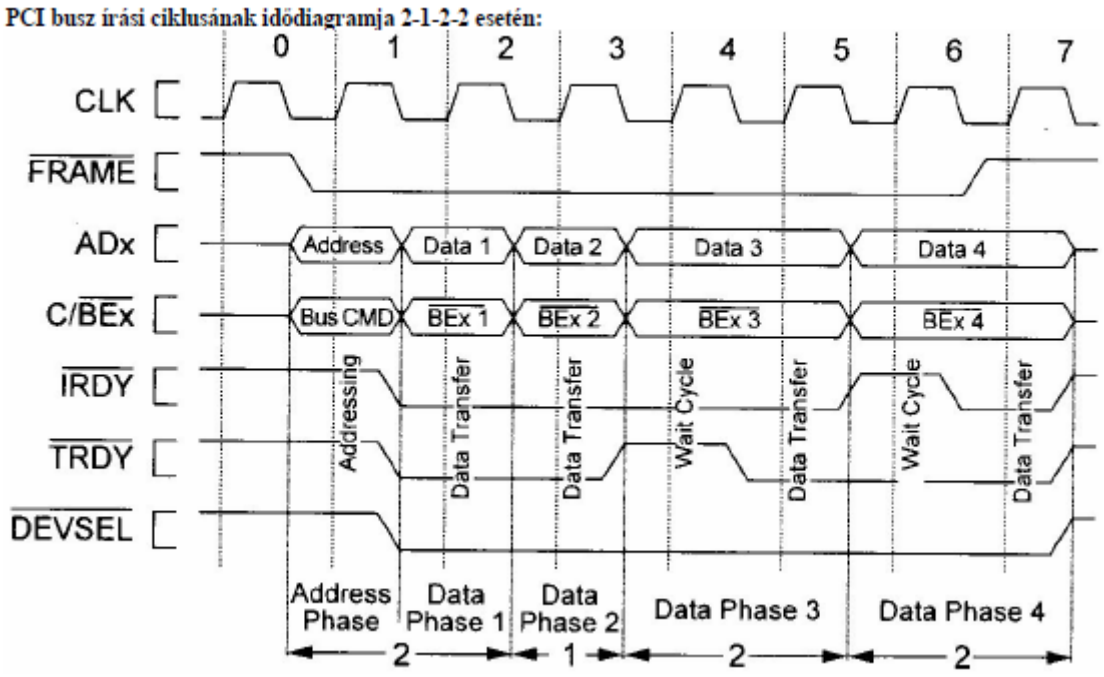
Frissíteni kell, mivel kisméretű kondenzátoron tároljuk az információt, melynek feszültsége idővel exponenciálisan csökken (kisül).



Egyetlen CAS oszlopcímet adunk ki, majd RAS-al az összes sorcímet, hogy az összes cella frissítésre kerüljön.

8. feladat: 7A

10. feladat: PCI burst write (PCI-Express)



2015 ZH A csoport:

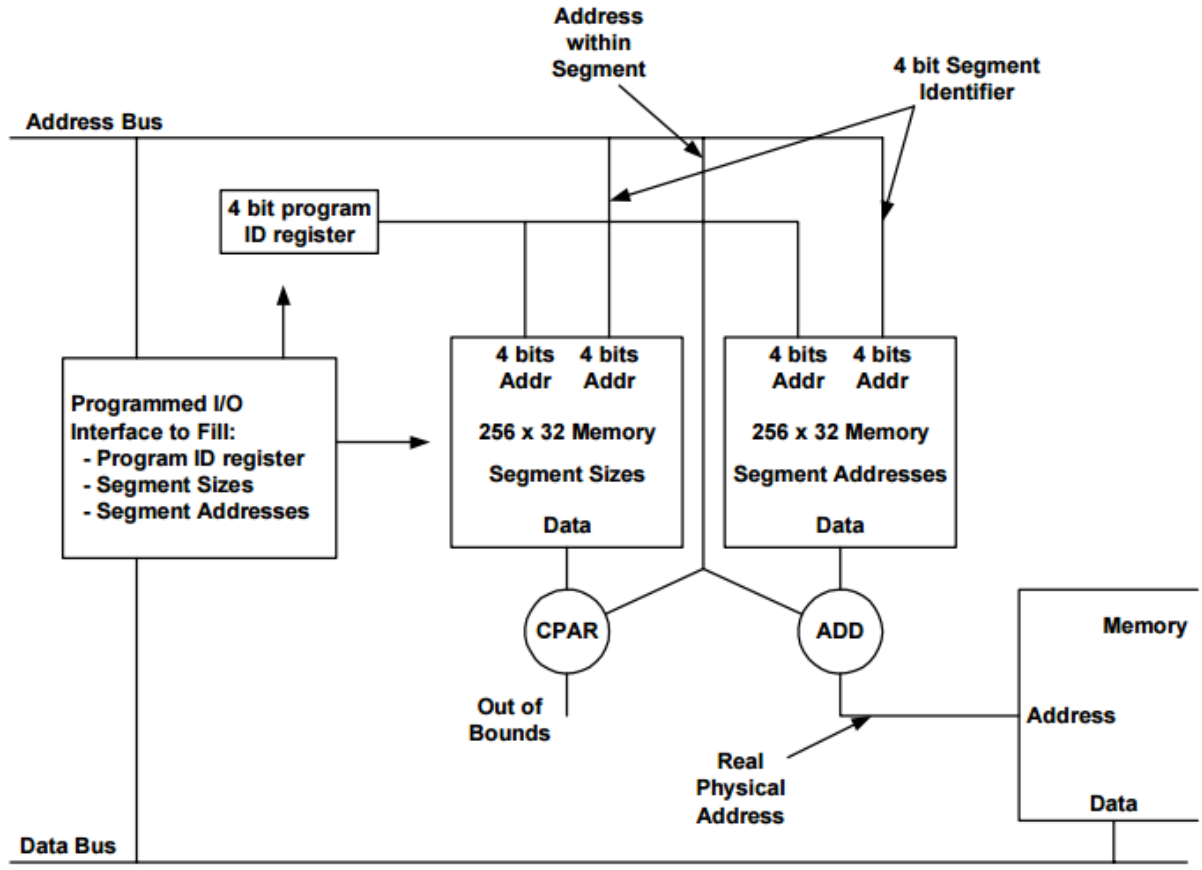
11. feladat: Virtuális memóriakezelés, típus: szegmentálás

Szegmens: a program változó méretű logikai egységekre van feldarabolva. A program 4 fő szegmense:
- 0.: főprogramok (olvasható/végrehajtható)
- 1.: szubrutinok
- 2.: csak olvasható adatok

- 3.: olvasható/írható adatok
A program a memóriát közvetlenül nem érheti el, csak a szegmensen keresztül.
FIZIKAI CÍM = Szegmens bázis címe + Szegmensen belüli eltolás címe (ez tényleg összeadás!!!)
Szegmenstáblából olvassuk ki a szegmens számát követően a hozzátartozó szegmens hosszát, szegmens címet, továbbá z egyes szegmensek elérési információit: (R/W/X) olvasható/írható/végrehajtható.

Szegmentálás hátránya: külső töredezettség
Szegmenstábla kezelése az OS feladata.

Szegmentálásos címfordító áramkör:

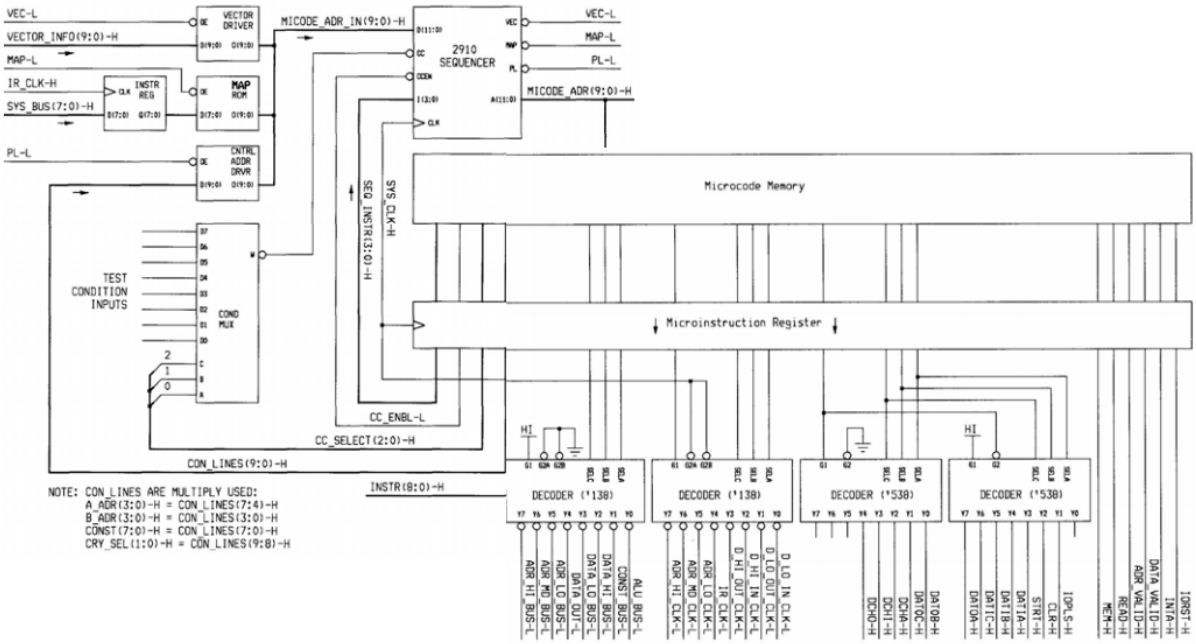


2015 ZH B csoport

4. feladat: Vertikális mikroprogramozás

Előny:
- takarékos az erőforrásokkal (fogyasztás, mikrokódban a bitek számával
Hátrány:
- lassú
- korlátozott a műveletek párhuzamos végrehajtása

Dekódolás: N bites buszt, $\log_2(N)$ számú bittel próbálunk kódolni. Több mikroutasítás szükségeltetik → mikrokódú memóriát vertikális irányban kell megnövelni



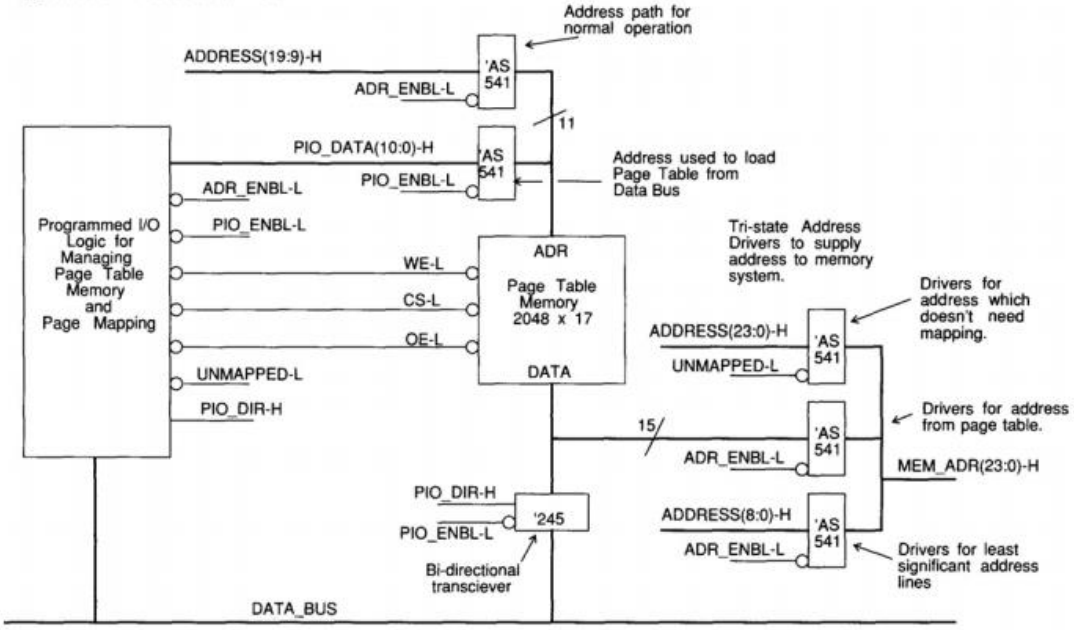
11. feladat: virtuális memóriakezelés, típus: lapozásos

Lap: a program és a memória is fizikailag egyenlő méretű darabokra van osztva (fizikai határok). Főprogram, szubrutinok, globális/lokális adatok, verem mind-mind egy-egy azonos méretű lapnak felelnek meg.

Hátránya:
- gazdaságtalan (nem használja ki a rendelkezésre álló lapméretet)
Előny:
- gyorsabb fizikai cím leképezés (nem kell összeadót használni)

Fizikai Cím = Lap Báziscíme + Lap eltolás címe (itt a + konkatenáció/összefűzés, NEM összeadás)
A konkatenációt huzalozással oldhatjuk meg.
Laptáblából olvassuk ki a lap számát követően a hozzátartozó lapcímet, továbbá az egyes lapok elérési információit: (R/W/X).
Fizikai címet a CPU határozza meg. Lapokat célszerű a memóriában tárolni.

Lapozásos címfordító áramkör:

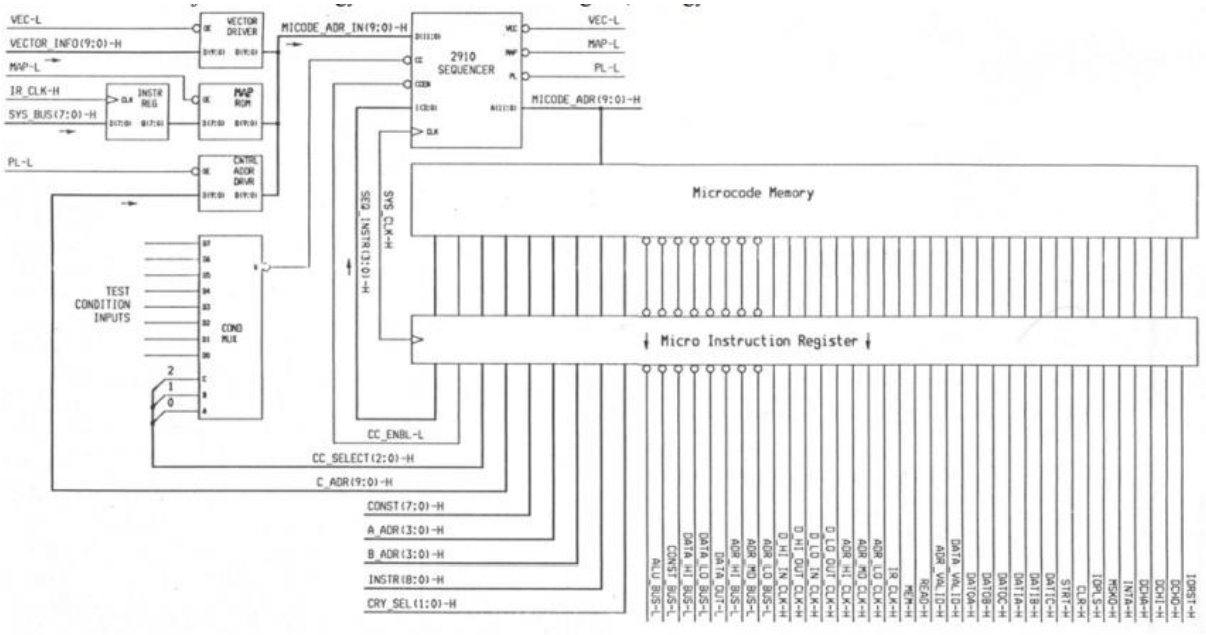


+.) Horizontális mikrokódos vezérlő

Mindenegyes 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.

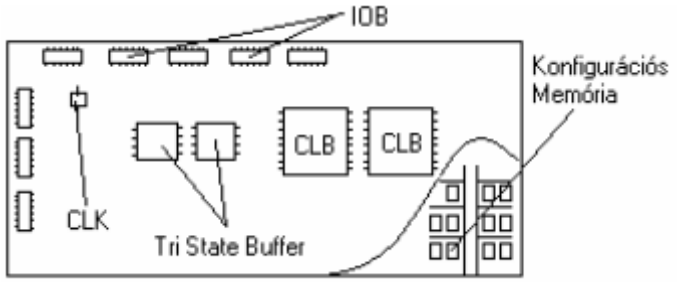
Előny:
- leggyorsabb mikrokódos technika
- minimális a műveletek ciklusideje

Hátrány:
- nagy az erőforrás szükséglete
- nagy fogyasztás



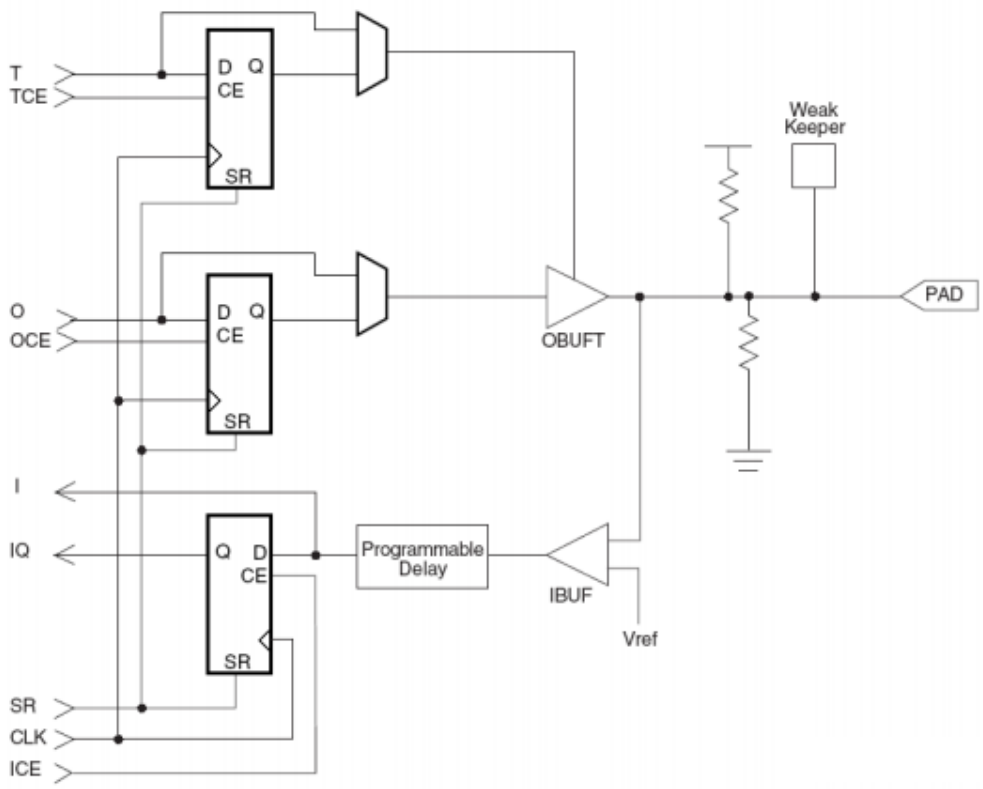
XILINX FPGA felépítése:

Általános felépítés:



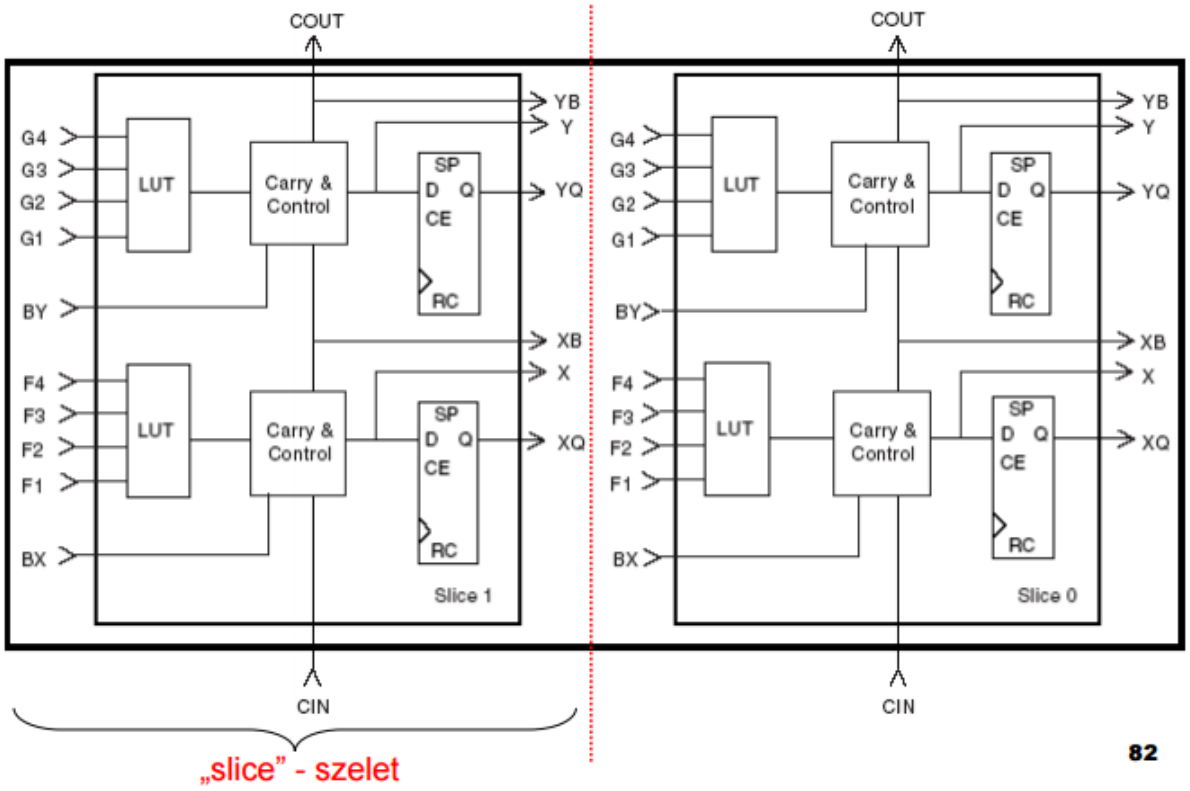
IOB Input/Output Blokk (programozható be/kimeneti blokk)

- kapcsolat a belső vonalak és a tokozás lábai között.
- kimenet lehet negált vagy ponált.



Virtex – CLB (Konfigurálható Logikai Blokk)

Szükséges logikai kapcsolatok megvalósítása egy logikai tömbben.



++.) 64x4es SRAM működéséhez mekkora lábszámot kell biztosítani?

6+4+1(CS)+1(R/W)+1(Vdd)+1(Gnd) = 14 → tehát összesen 14 lábra (pin) van szükség.

++.) I/O csatornák típusai:

- Multiplexeres : lassú
- Blokkos: közepesen gyors
- Selector Channel: nagysebességű

+++.) Aszinkron Handshaking protokoll – olvasási ciklus

t0-ban a master megadja a kívánt slave címét.
t1-ben a master vár egy bizonyos ideig mielőtt beállítja a request vonalat. (master a slavetől adatot kér, olvasni szeretne)
t2-ben a slave veszi a kérést, nyugtázza és beállítja magas szintre a nyugtázó vonalat.
t3-ban a master megkapja az adatot a slavetől, felszabadítja a request vonalat.
t4-ben a slave érzékeli, hogy a master felszabadította a request vonalat -> felszabadítja a nyugtázó vonalat.
t5-ben a master felszabadítja a címvonalat.