



Digitális Rendszerek és Számítógép Architektúrák

7. előadás: I/O műveletek
PCI, PCI Express, SCSI buszok

Előadó: Dr. Vörösházi Zsolt
voroshazi.zsolt@virt.uni-pannon.hu

Jegyzetek, segédanyagok:

- Könyvfejezetek:

- <http://www.virt.uni-pannon.hu>

- ⇒ Oktatás ⇒ Tantárgyak ⇒ Digitális Rendszerek és Számítógép Architektúrák (Nappali)

- (chapter06.pdf + további részek, amik a könyvben nem szerepelnek: PCI_bus.pdf, SCSI_bus.pdf)

- Fóliák, óravázlatok .ppt (.pdf)

- Feltöltésük folyamatosan



I/O műveletek

- **Aszinkron protokoll**
- **Szinkron protokoll**
- **Arbitráció (döntési mechanizmus)**
- **Megszakítás – kezelés (operációs rendszerek)**
- **Buszok – Buszrendszerek:**
 - PCI,
 - PCI-Express,
 - SCSI buszok

I/O egységek

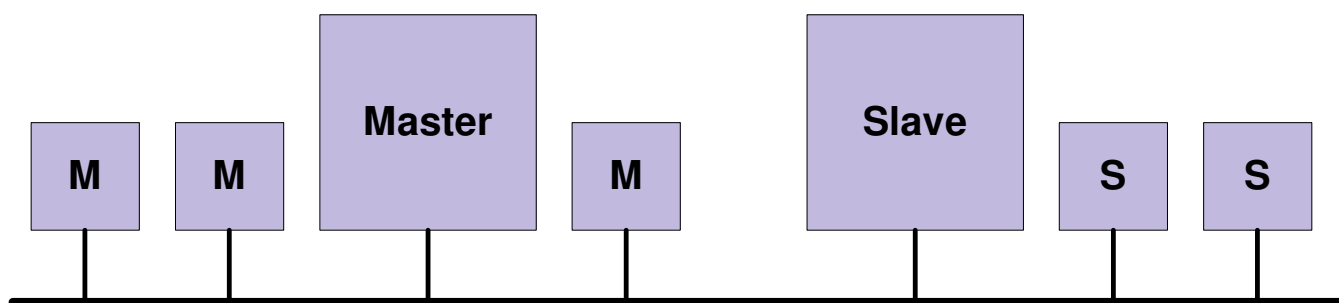
- A számítógép a külvilággal, *perifériákkal* az I/O egységeken keresztül tartja a kapcsolatot. Az információ továbbítását az egységek között buszok végzik, amelyek interfészekkel kapcsolódnak egymáshoz.
- **Interface**: azon szabályok összessége, amelyek mind a fizikai megjelenést, kapcsolatot, mind pedig a kommunikációs folyamatokat leírják. Egy busz általában 3 fő kommunikációs vonalból állhat:
 - vezérlőbusz,
 - adatbusz, és
 - címbusz.

I/O kommunikációs protokollok:

- Kommunikáció során megkülönböztetünk egy eszközpárt: a *Master*-t és *Slave*-t. A Master (pl. CPU) általában, mint kezdeményező, birtokolja és ellenőrzi a buszt, és átadja (ír / olvas) az adatokat a Slave-nek (pl. Memória).
- A kommunikációhoz előredefiniált **protokollokra** (szabályok és konvenciók gyűjteménye) van szükség, amelyek meghatározzák az események sorrendjét és időzítését. A kommunikáció feltétele a másik egység állapotának pontos ismerete. Lehetséges több M-S modul (pl. multi-master rendszer több kezdeményezővel) is egy rendszerben.
- **Két alapvető protokoll különböztethető meg:**
 - 1.) *Aszinkron* kommunikációs (pl. SCSI busz), és
 - 2.) *Szinkron* kommunikációs (pl. PCI busz) protokoll.

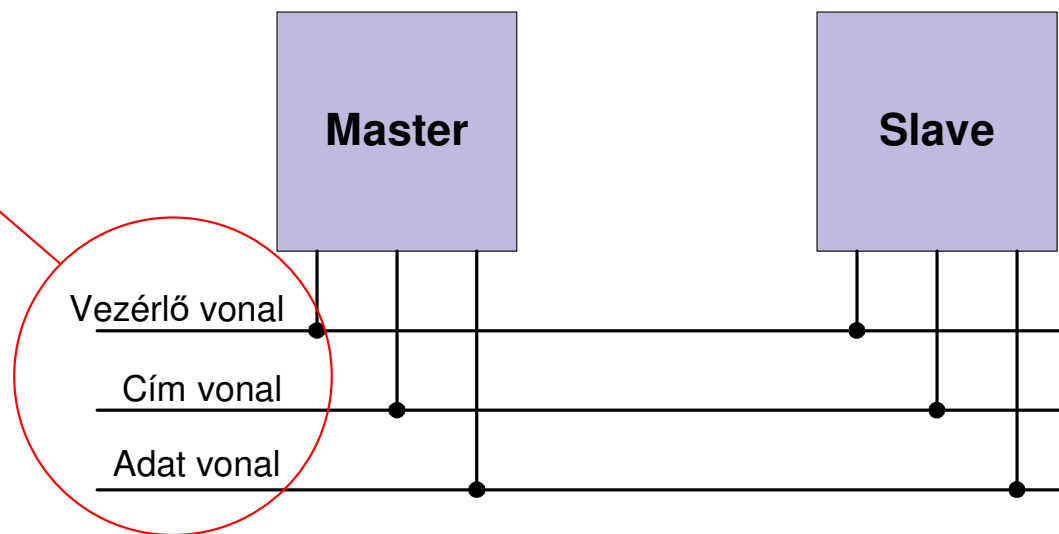
Busz rendszer Master-Slave moduljainak szervezése

■ Modul-szervezés



Busz-Master (busz-vezérlő) kommunikál a Busz-Slave-el (responder).

■ Busz-szervezés: Master - Slave





1.) Aszinkron busz protokollok

Aszinkron busz protokollok

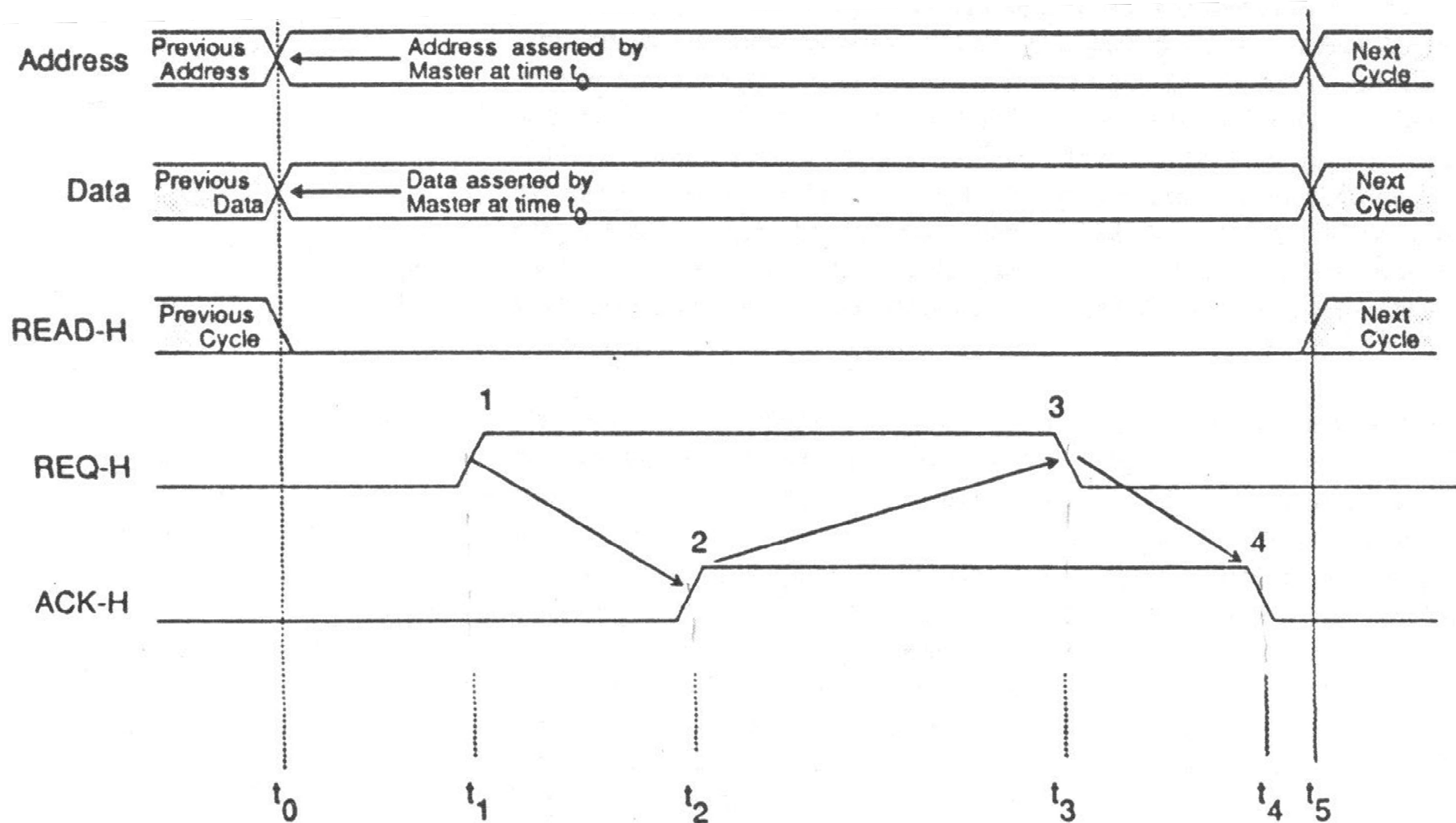
- Aszinkron handshaking: „2x-es kézfogás”
- Ebben az esetben a Master-Slave modulok **nem** közös órajelet (SysCLK-t) használnak: azaz ha az egyik egység végez, elindít egy másik tranzakciót. A Master, mint (commander) kezdeményező, aktiválja a megfelelő vezetékeket. A Slave (responder) válaszol. A Slave címének azonosítására egy külön egység szolgál.
- **Vezérlőjelekkel** zajlik a kommunikáció: *írási / olvasási* tranzakciókat különböztetünk meg. Ilyen vezérlőjelek:
 - ☐ READ, (ha pl. READ-H = Master olvas a Slave-től, ha READ-L= Master ír a Slave-nek, vagyis a Slave olvas.),
 - ☐ REQUEST (REQ)
 - ☐ ACKNOWLEDGE (ACK)
- *Előnye:* egyszerűen felépíthető, nem kell (közös) órajel, gyors átvitelt biztosít (modulok sebességétől függően)
- *Hátránya:* belső késleltetések (skew-propagation time), ill. csak korlátozott hosszúságú buszok /vezetékek használhatóak.

Aszinkron Handshaking protokoll (kétszeres „kézfogás”)

- A buszrendszer moduljai nem közös órajellel működnek, hanem **vezérlő jelek** segítségével. (READ-H, REQ, ACK)
- Háromféle buszvonalat ismerünk: *cím-, adat- és vezérlő-* buszt. A Master által *címvonala*ra rakott cím (kezdeményezés) egyértelműen meghatározza a tranzakció célállomását. Meghatározott idő áll rendelkezésre, hogy a slave modulok összehasonlítsák saját címükkel a célcímet. Ha megegyezik, akkor válaszol a master-nek, és a tranzakciót a *vezérlővonalak* megfelelő beállításaival szabályozza. A vezérlővonalakat tehát a master-slave közötti kommunikáció szinkronizálására használjuk. Ezt nevezzük **handshaking** protokollnak.
- Címvonalak csoportjából (Address), adatvonalak csoportjából (Data), és három vezérlőjelből (READ-H, REQ-H, ACK-H) áll. Ha READ-H magas, akkor a Master olvas a Slave-ről, ha alacsony, akkor ír a Slave-re. A REQ (kérés) és ACK (nyugtázás) vezérlőjelek határozzák meg az események időzítését és pontos sorrendjét.

Handshaking – írási ciklus

WRITE (READ L) ciklus: „a Master modul ír a Slave-nek”

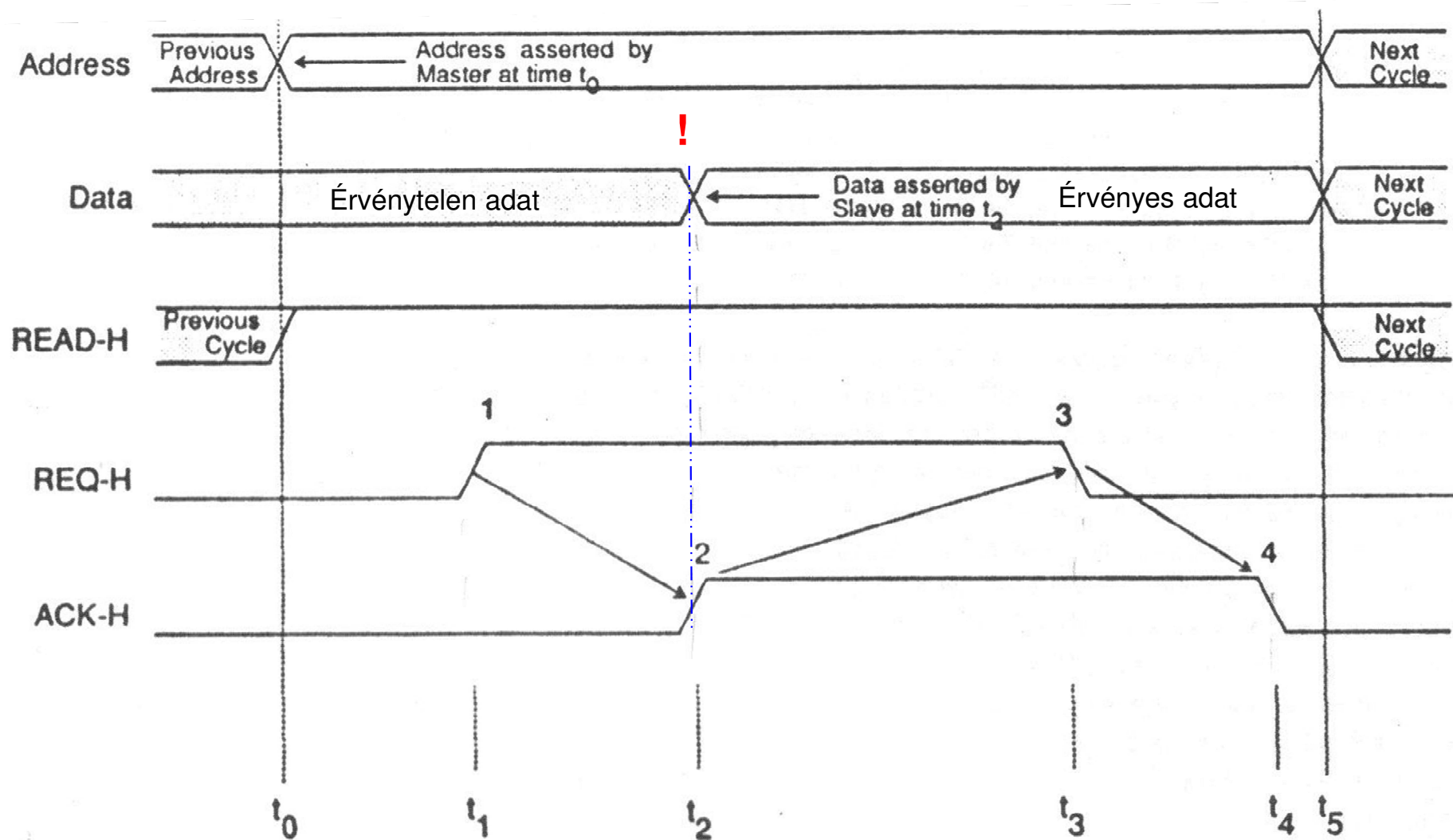


Írási (Write) ciklus lépései:

- t_0 -ban a master (amely az aktuális arbitráció után már megkapta a busz vezérlését) megadja a kívánt slave címét.
- véges idő kell hogy a jel a slave modulokhoz érjen, azok dekódolják a címet, ezért a master vár bizonyos ideig mielőtt beállítja a request vonalat t_1 -ben. (*skew time*= a slavekhez elsőként ill. utolsóként érkező címek közötti időkülönbség $\Delta t = t_1 - t_0$)
- ezt a request jelet minden slave veszi ugyan, de csak az fog válaszolni, akinek a címe megegyezett a master által kért címmel.
- amikor a slave megkapta az adatokat a mastertől, nyugtázza t_2 -ben.
- a master megkapja a nyugtát, így tudja, hogy az átvitel megtörtént, ezért felszabadítja (alacsony állapotba helyezi) a request vonalat t_3 -ban.
- ezt (a request felszabadítását) érzékeli a slave, és felszabadítja a nyugtázó vonalat t_4 -ben
- a master a címvonalat tartja még egy bizonyos ideig (t_5 -ig) a request vonal felengedése után is, a *cím esetleges megváltozása* miatt (amelyet a Slavek dekódolnak).

Handshaking – olvasási ciklus

READ (READ H) ciklus: „a Master olvas a Slave-től”



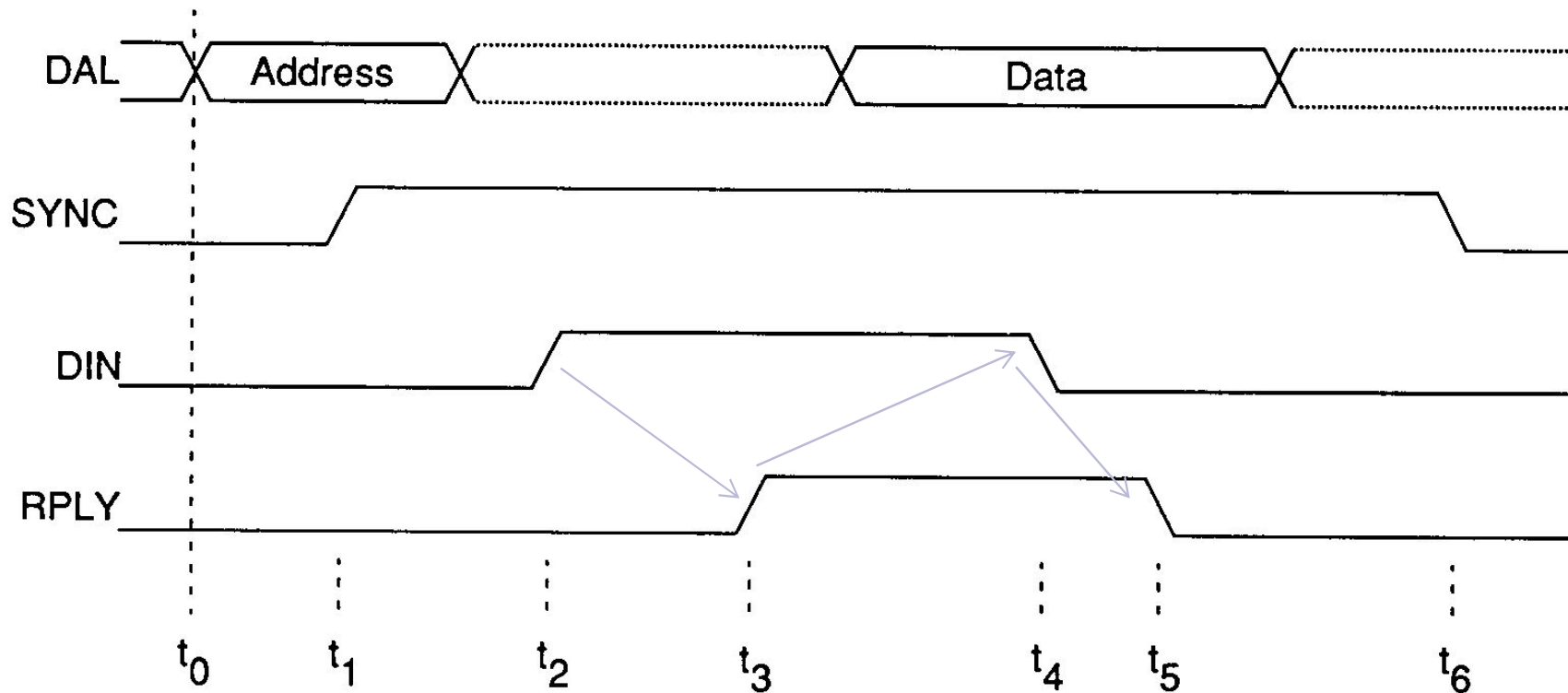
Olvasási (Read) ciklus lépései:

- Nagyon hasonlít az írási folyamathoz, de abban különbözik, hogy a READ-H jel magas szinten van, és az adatvonalat a slave állítja be. Ez jelenti az olvasást.
- t_0 hasonlóan történik, a master (amely az arbitráció után már megkapta a busz vezérlését) megadja a kívánt slave címét
- véges idő kell hogy a jel a slave modulokhoz érjen, azok dekódolják a címet, ezért a master vár bizonyos ideig mielőtt beállítja a request vonalat t_1 -ben. (*skew time* = a slavekhez elsőként ill. utolsóként érkező címek közötti időkülönbség $\Delta t = t_1 - t_0$). Tehát t_1 -ben a master a request vonal beállításával a megcímzett slave-től adatot kér, olvasni szeretne
- t_2 -ben veszi a kérést a slave, nyugtázza és beállítja magas szintre a nyugtázó vonalat.
- t_3 -ban a master megkapja az adatot a slave-től, felszabadítja a request vonalat.
- t_4 -ben érzékeli a slave, hogy a master felszabadította a requestet, ezért így ő is felszabadítja a nyugtázó vonalat.
- végül a master felszabadítja a címvonalat (t_5 -ben)

Példa.) Q-Bus: idő-multiplexált aszinkron protokoll

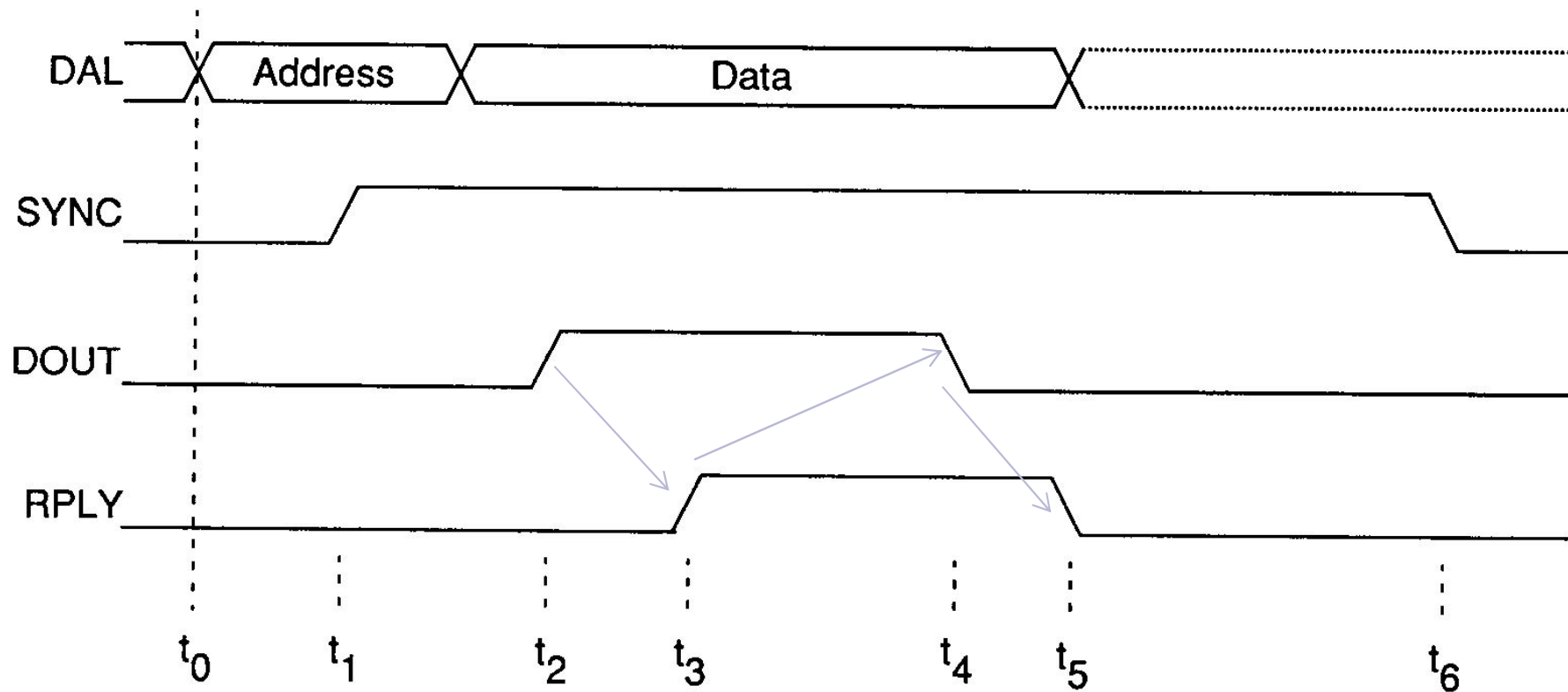
- DEC PDP / VAX gépeihez tervezték
 - DEC QBus: multiplexált adat/cím busz rendszer
- Olvasás: (Read) M olvas az S-től
- Írás: (Write) M ír az S-nek
- Jelei:
 - DAL: Data / Address Line (közös) (16 ... 22 bites)
 - SYNC: Slave eszközöknél a cím esetleges tárolására („latch”: tárolóelemben), amelyet a *Master* kezel.
 - DIN: Master bemenő vonala (olvasáskor $\sim$ REQ)
 - DOUT: Master kimenő vonala (írásakor $\sim$ REQ)

Q-Bus: aszinkron olvasás (read)



- M beállítja a címvonalat (DAL) t_0 -ban. (+ propagációs és skew time)
- S beállítja SYNC-et t_1 -ben
- M felszabadítja az címnél a DAL vonalat, hogy később beállíthassa a DAL vonalat az adatahoz is (címezett Slave-től fog olvasni)
- Ezt követi a 4-lépéses tranzakció, ahol DIN jelenti az igénylést t_2 -ben
- A Slave nyugtázza az igényt RPLY-vonalon t_3 -ban. (adat átvitel ez után kezdődhet)
- M válaszol erre, a DIN felengedésével t_4 -ben.
- S észreveszi ezt és t_5 -ben felszabadítja a RPLY, majd pedig DAL vonalat is.
- SYNC felszabadítása (t_6 -ban) a Master által

Q-Bus: aszinkron írás (write)



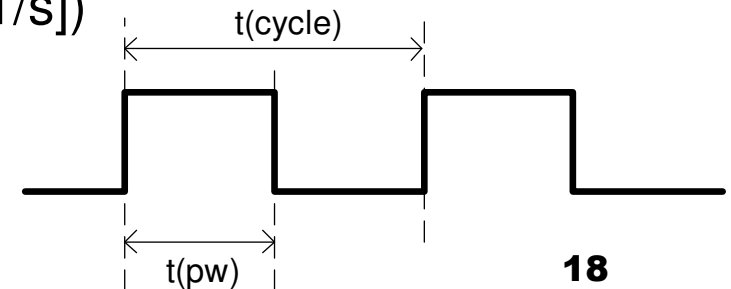
- M beállítja a címvonalat (DAL) t_0 -ban. (+ propagációs és skew time)
- M beállítja SYNC-et t_1 -ben
- Miután M felszabadítja az címnél a DAL vonalat, és már egyből beállítja a DAL vonalat az adathoz is (címezett Slave-nek fog írni)
- Ezt követi a 4-lépéses tranzakció, ahol DOUT jelenti az írási igényt t_2 -ben
- A Slave nyugtázza az igényt RPLY-vonalon t_3 -ban. (adat átvitel ekkor kezdődik a Slave felé)
- M válaszol erre, a DOUT felengedésével t_4 -ben.
- S észreveszi ezt és t_5 -ben felszabadítja a RPLY, és egyszerre a DAL vonalat is.
- Busz protokoll kérheti a Mastert hogy tartsa egy ideig a SYNC-t, majd felszabadítja (t_6 -ban)



2.) Szinkron busz protokollok

Szinkron busz protokollok

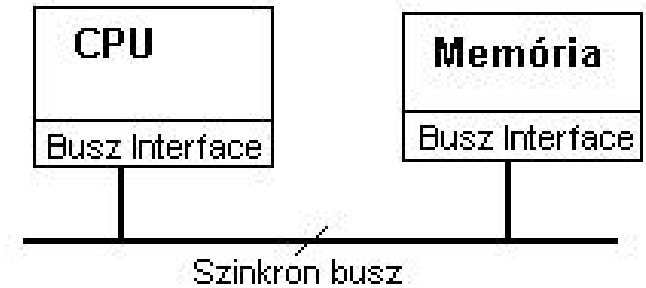
- Ebben az esetben a Master-Slave modulok **közös órajelet** (CLK-t): használnak. Itt a Master, mint Commander (kezdeményező), a Slave pedig (Responder) válaszol.
- *Előnye*: Tehát a műveleti időt az órajelciklus határozza meg. Mivel nincs szükség párbeszédre (pl. handshaking), gyorsabb lesz az aszinkron működésénél.
- *Hátránya*: Az órajelet mindig a leglassabb (legtávolabb lévő) egységhez kell igazítani.
- A digitális áramkörökben az események történésének *sorrendje kritikus* (megfelelő időzítés kell $\Rightarrow$ órajel vezérléssel)
- **Óra**: impulzusok sorozatát bocsátja ki, pontosan meghatározott szélességgel $[t(pw)]$, és időintervallummal.
- **Ciklus-idő** (clock-cycle): két egymást követő pulzus élei közötti időintervallum $[t(cycle)]$.
 - Példa: Órajel Frekvencia: $f=100\ 000\ 000$ [Hz] ($[1/s]$)
 - Ekkor $T=1/f=1 / 100\ 000\ 000 = 10$ [ns]
- Kristály-oszcillátor szolgáltatja ált. az órajelet.



Szinkron írási / olvasási ciklus

• Írási ciklus (pl. CPU → MEM)

időlépések

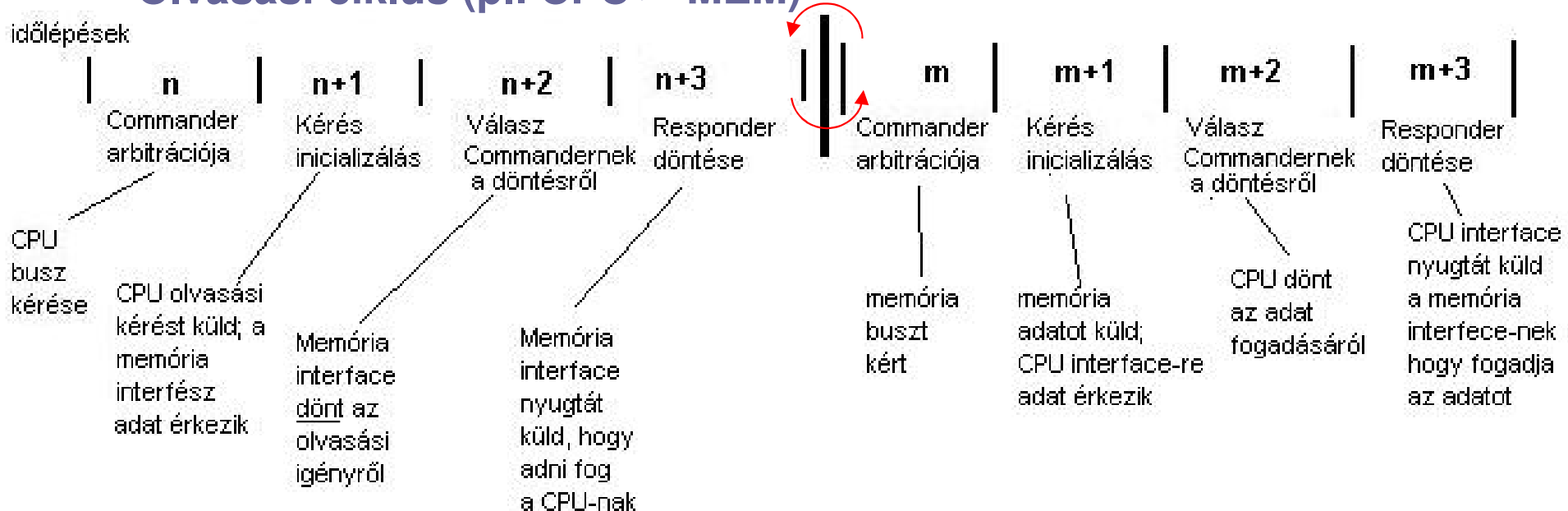


Szinkron adatátvitel 4 fő lépése:

1. Commander arbitrációja, kiválasztása
2. átvitel megkezdése (de *még nem fogad*)
3. válasz a kérésre, döntéshozás
4. nyugta, Responder *veszi* az adatot

• Olvasási ciklus (pl. CPU ← MEM)

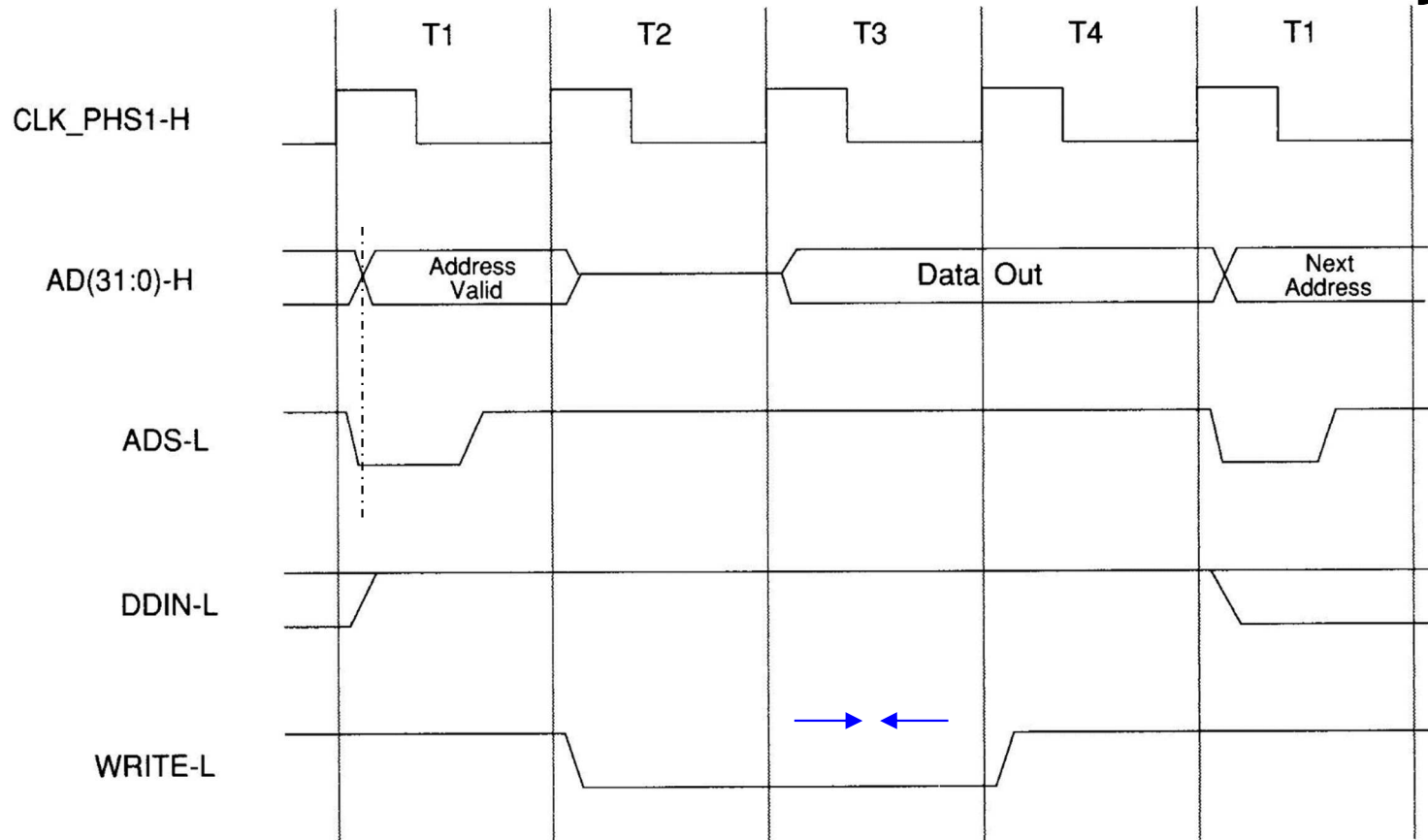
időlépések



Példák: Szinkron busz protokollokra

- 1.) **Általános célú busz:** CPU közvetlen vezérlése alatt áll (határozza meg a forrást ill. célt)
- 2.) **Motorola 68020:** nagyteljesítményű rendszer. Dinamikus buszméret. CLK-val szinkron kommunikáció. Ha a Slave rövid időn belül nem képes egy kérést kielégíteni, akkor a CPU automatikusan egy tétlen („idle”) állapotot, *várakozó* ciklust generál, amíg a Slave nem válaszol.
- 3.) **NS32332** buszrendszer: 32-bites szinkron rendszer. Idő-multiplex Addr/Data kommunikáció (IC lábszám csökkentése miatt).
- 4.) **PCI buszrendszer!** (Lásd később)

Példa: NS32332 Busz **írási** tranzakciója



- CLK_PHS1_H: közös rendszer-órajel (felfutó élre)
- AD (31:0)_H : 32-bites multiplexált cím/adat vonal (érvényes adat T_3 -ban)
- ADS_L: érvényes cím érkezését jelzi T_1 ciklusban
- WRITE_L: írás engedélyezés T_2 -ben
- DDIN_L: adatátviteli irány (DIN: -L: out/olvasás, -H: input/írás)
- Minimális tranzakcióhoz 4 ciklus szükséges (T_1 - T_4):

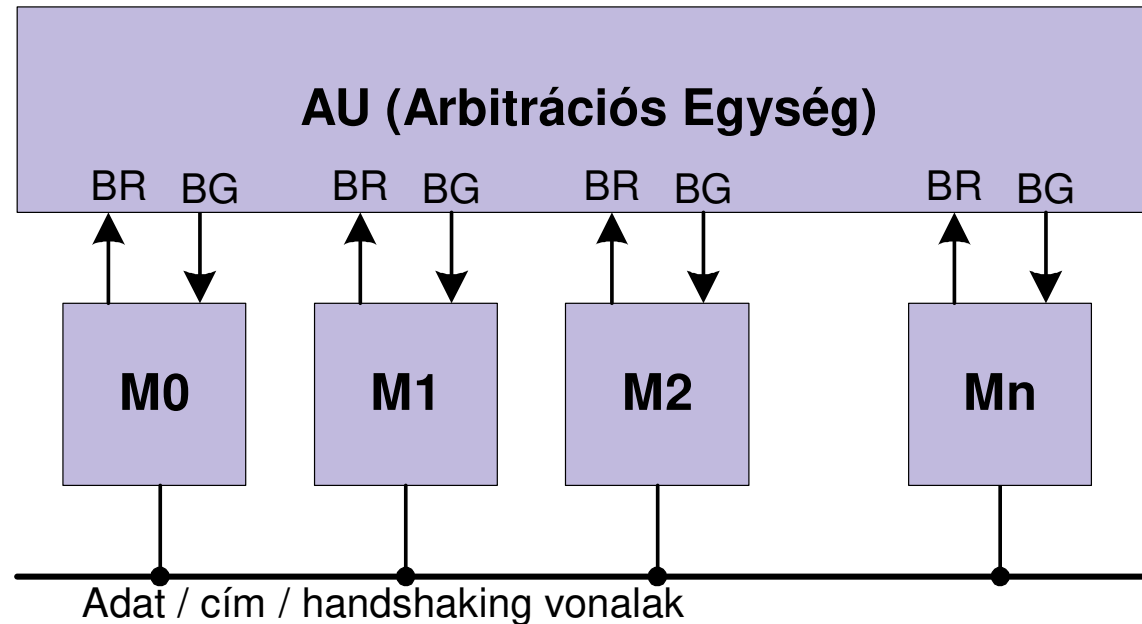


Busz Arbitráció

Arbitráció

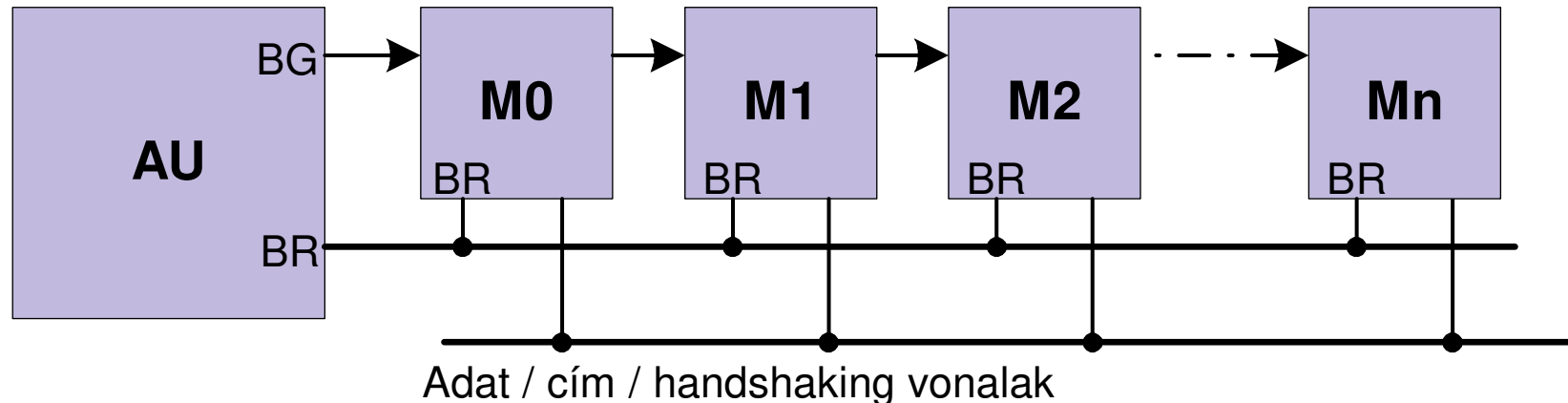
- Egy tetszőleges I/O művelet esetén (aszinkron v. szinkron buszos átvitel) több Master (commander) egység is meg akarja szerezni egyszerre a busz irányítását. Különböző előredefiniált algoritmusok segítségével egyértelműen azonosítható, hogy a „**versenyhelyzetben**” a következő átvitelt (a következő ciklusban) melyik Master fogja megvalósítani. Az **arbitrációs eljárás** egy **döntési folyamat (mechanizmus)**, amely az adatátvitellel párhuzamosan zajlik le, és még az aktuális adatátvitel befejezése előtt eldől, hogy melyik következő master adhat. A Mastereket $M_1 \dots M_n$ -el jelöljük, a *BR: Bus Request* (busz kérése, igénylése) a Master által, míg a *BG: Bus Grant* (igénylés elfogadása, engedélyezés) jelet a központi *AU: Arbitration Unit* (arbitrációs egység) bocsátja ki.
- **Az arbitrációnak 3 típusa van:**
 - ☐ a.) párhuzamos,
 - ☐ b.) soros (daisy chain), és
 - ☐ c.) lekérdezéses (polling).

a.) Párhuzamos



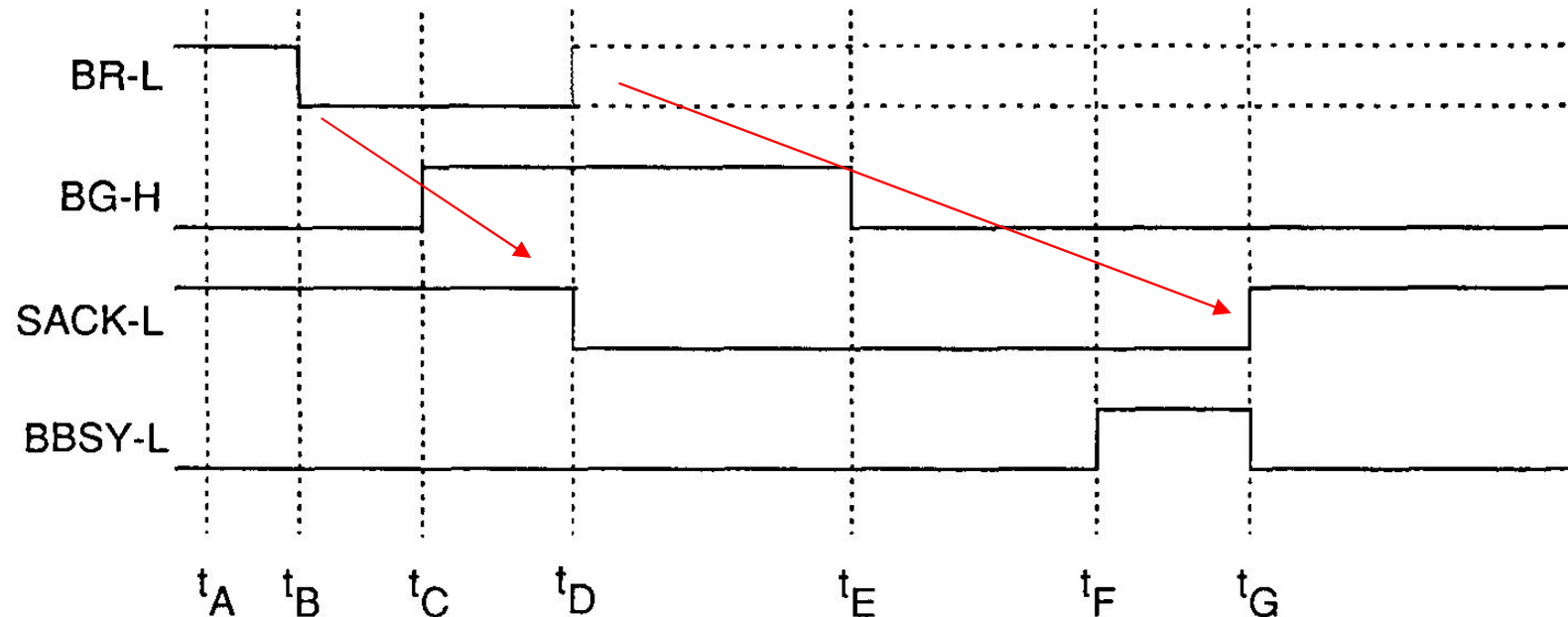
- Ez a leggyorsabb módszer, minden M_i -nek kitüntetett, egyenrangú kapcsolata van az AU-val (két vonalon: BG és BR-en). Az arbitráció lehet (i) *first asserted/first served (FIFO)*, vagy (ii) *round robin*, vagy (iii) *prioritásos alapú*. Ezek a módszerek többfajta lehetőséget biztosítanak mind egyszerű,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

b.) Soros (daisy chain)



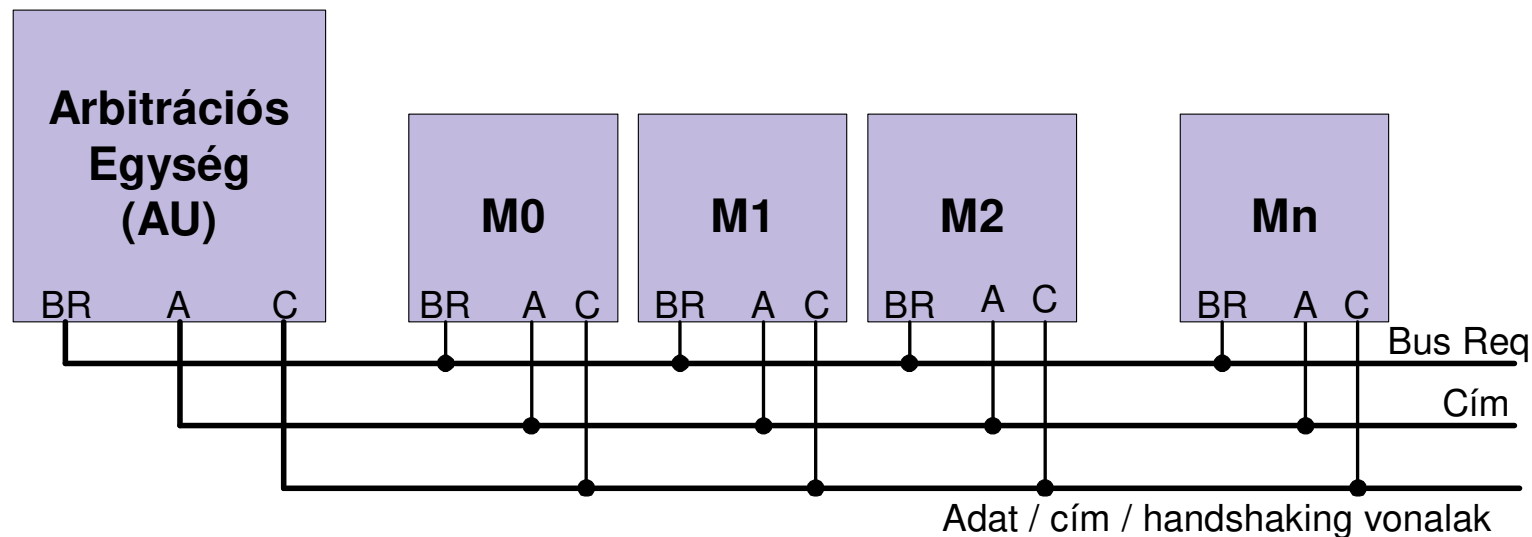
- A BG vonal sorosan van kötve, míg a BR vonal mindegyik M_i -hez csatlakozik. Az AU nem ismeri, hogy pontosan melyik M kívánja elérni a buszt, ezáltal az átvitel leegyszerűsödik: csupán azt tudja, hogy bizonyos ciklusonként BG-jelet kell kibocsátania. A soros csatlakozás miatt a legelső Master (M_0) 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_i igényel és az előtte lévők nem) egyenes arányban van a sorba kapcsolt M_i -k számával.

Példa: UNIBUS - Soros arbitrációs rendszer



- Jelei: BR_L, BG_H, SACK_L (selection ack), BBSY_L (bus busy - foglalt)
- Lépései:
 - t_A : AU felismeri, hogy egy új arbitrációs ciklusban vagyunk (SACK inaktív)
 - t_B : Master igényel (BR-L)
 - t_C : AU nyugtázza a kérést, veszi az M igényét (BG-H)
 - t_D : Master BG jelet kap az AU-tól, SACK-L küldése
 - t_E : AU a tranzakció végén elengedi a BG-H jelet ('0')
 - t_F : „aktuális” Master befejezi a ciklust, felengedi a BBSY-L jelet: '1' (a **sorban** következő Masternek adja át a döntési mechanizmusnak megfelelően)
 - t_G : új busztranzakció kezdete a Master által (új ciklus)

c.) Lekérdezéses (polling)

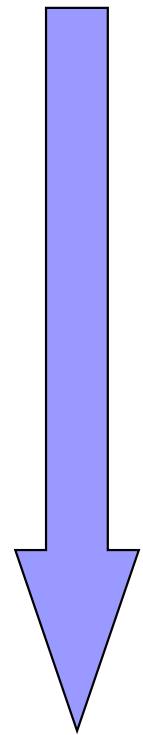


- Mindegyik Master egy **közös** BR buszon igényelhet (ID), és az AU dönti el, hogy melyik Masternek adja a buszt. Annak az M-nek a **címét** az *address* vonalra rakja. Minden ciklusban megnézi az igényléseket, és az aktuálisan legmagasabb prioritással rendelkezőt fogadja el. Itt is többféle prioritásos módszer megvalósítható: pl. (FIFO, round robin stb.).
- Hátránya, hogy nagyobb az időszükséglete a párhuzamosnál, így lassabb periféria műveletek esetén alkalmazzák (pl: I/O kérések arbitrációjánál), a processzor egy program futtatásával folyamatosan monitorozza az I/O eszközök busz kéréseinek állapotát
 - (a megszakításos I/O műveletek ennél a módszernél jobbak).



Adatmozgatás: I/O kommunikációs technikák

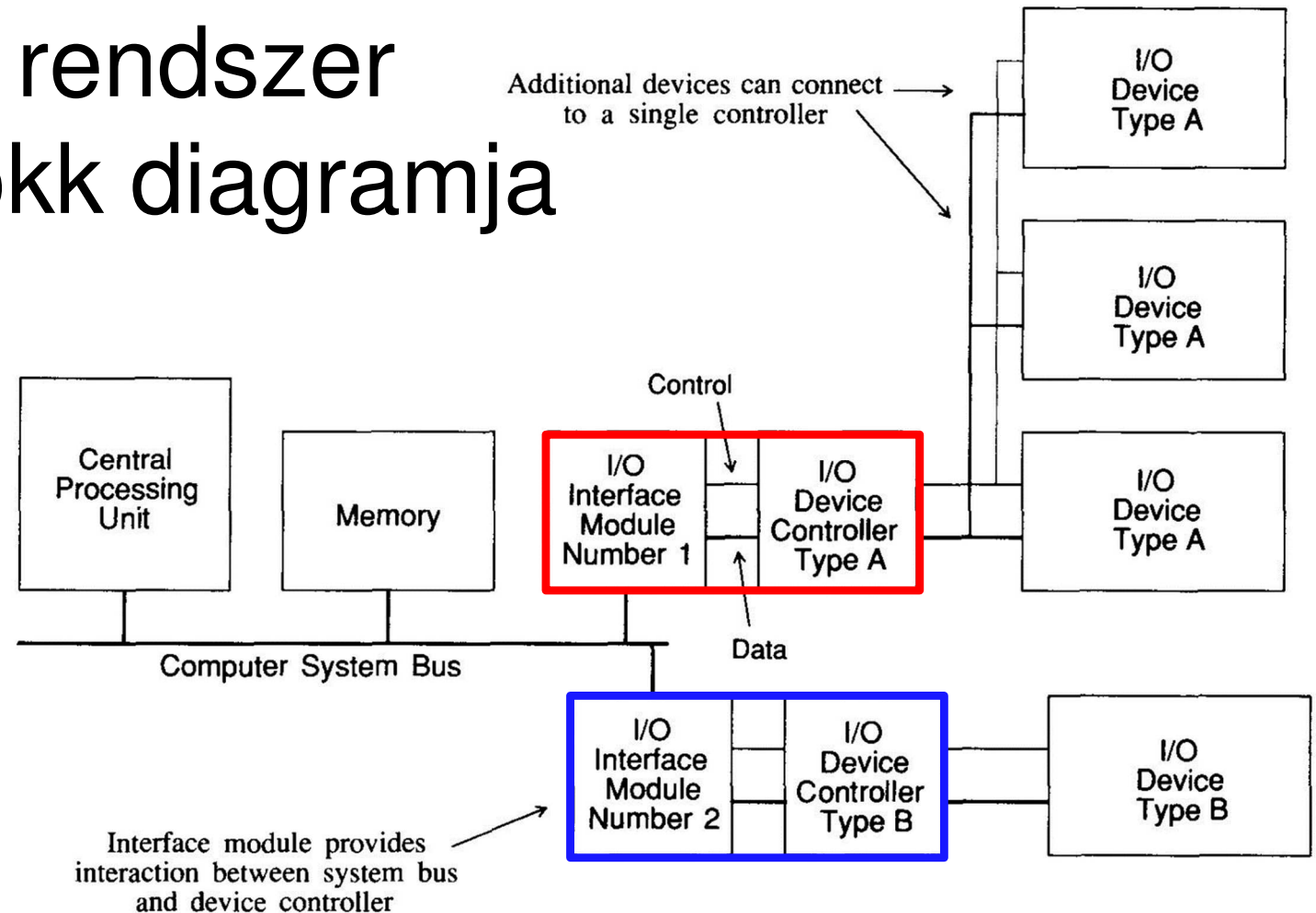
I/O adatátvitel típusai:



- Programozott I/O átvitel (polling)
- Megszakításos (interrupt) átvitel
- Direct Memory Access (DMA) -
Közvetlen memória-hozzáféréssel
rendelkező átvitel
- IOP - I/O processzoros átvitel

Feladat: CPU
tehermentesítése

I/O Interfész rendszer általános blokk diagramja



Két fő részre osztható: 1. **rendszerbusz** (CPU, memória) 2. **I/O eszközvezérlők** a különböző típusú I/O eszközökkel tartják a kapcsolatot. A rendszerbuszt és az I/O eszközvezérlőket az I/O interfacek kapcsolják (illesztik) össze. Az eszközök gépi kódú utasításokkal (assembly) vezérelhetők. Egy új I/O eszközt (A v. B típusú) a megfelelő típusú eszközvezérlőkhöz kell kapcsolni.

1.) Programozott I/O átvitel (Polling)

- Legegyszerűbb technika
- De leginkább ez a módszer terheli a CPU-t (adatátvitel teljes ideje alatt) – lassú eszközök esetén
 - Teljes vezérlésért, adatmozgatásért felel
 - Pl. periféria állapotának ciklikus lekérdezés folyamatosan terheli
- Csak a CPU közbeiktatásával érheti el a periféria a memóriát
- Lehet Memory Mapped I/O: amikor a program és az I/O eszköz is ugyanazt a címtartományt használja (cím leképezés = „mappelés”)

2.) Megszakításos (interrupt) átvitel

- Megszakítással jelezhető a CPU-által az I/O eszköz számára az adatátviteli igény, illetve az adatátvitel befejeződése
- Megszakítás kérelem (interrupt request) I/O eszköz által – **IRQ** szintek
 - Megszakítási vektorok (maszkolható megszakítások – SW interrupt)
- Bővebben: Operációs rendszerek tárgyból

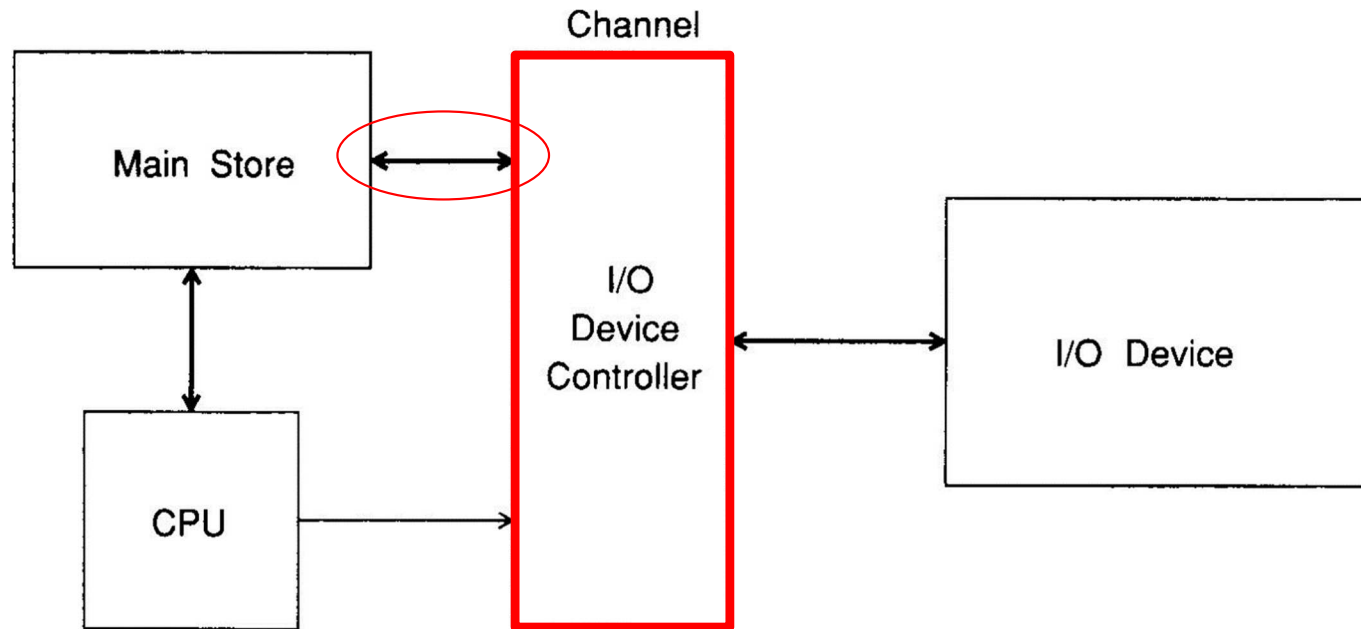
3.) Direct Memory Access (DMA)

- Közvetlen memória hozzáférés:
az I/O eszköz/periféria és memória közötti adatátvitelt a processzortól függetlenül egy DMA-controller/vezérlő (eszközvezérlő) végzi el.
- Cél a CPU tehermentesítése a tranzakció idejére
- CPU feladatai csupán:
 - az átvitel előkészítése (kezdőcím és adat(ok) hossza, száma),
 - minimális vezérlés: eszköz állapot vizsgálata (busy)
 - és a befejezett művelet hibátlanságának ellenőrzése (megszakításos alapú is egyben)
- Gyors módszer

4.) I/O processzor (I/O csatornák)

- A cpu átadja az I/O műveletet és a végrehajtáshoz szükséges összes adatot ez intelligens eszközvezérlőnek = I/O (társ)processzornak, amely teljesen önállóan szabályozza a tranzakciót
- Főleg a mainframek-re jellemző módszer
- Rendkívül gyors
- I/O csatornák: eszköz sebessége szerinti osztályozás

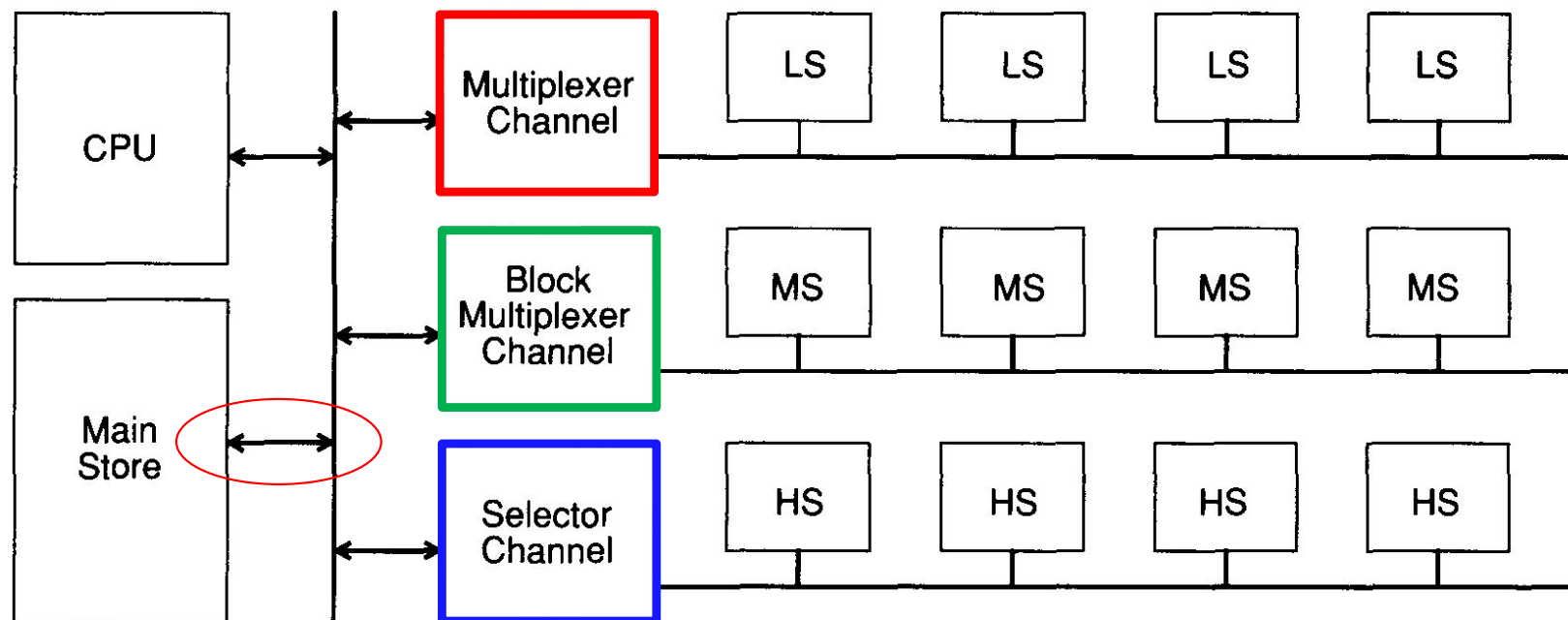
a.) I/O channels – I/O csatornák



- A cpu csak az eszközvezérlőn keresztül (közvetetten) érheti el a perifériát.
- Channel = I/O Device Controller: általános célú processzáló elem.
- Channel feladata:
 - Konverzió
 - Adatmozgatás
 - Hibaellenőrzés és kezelés

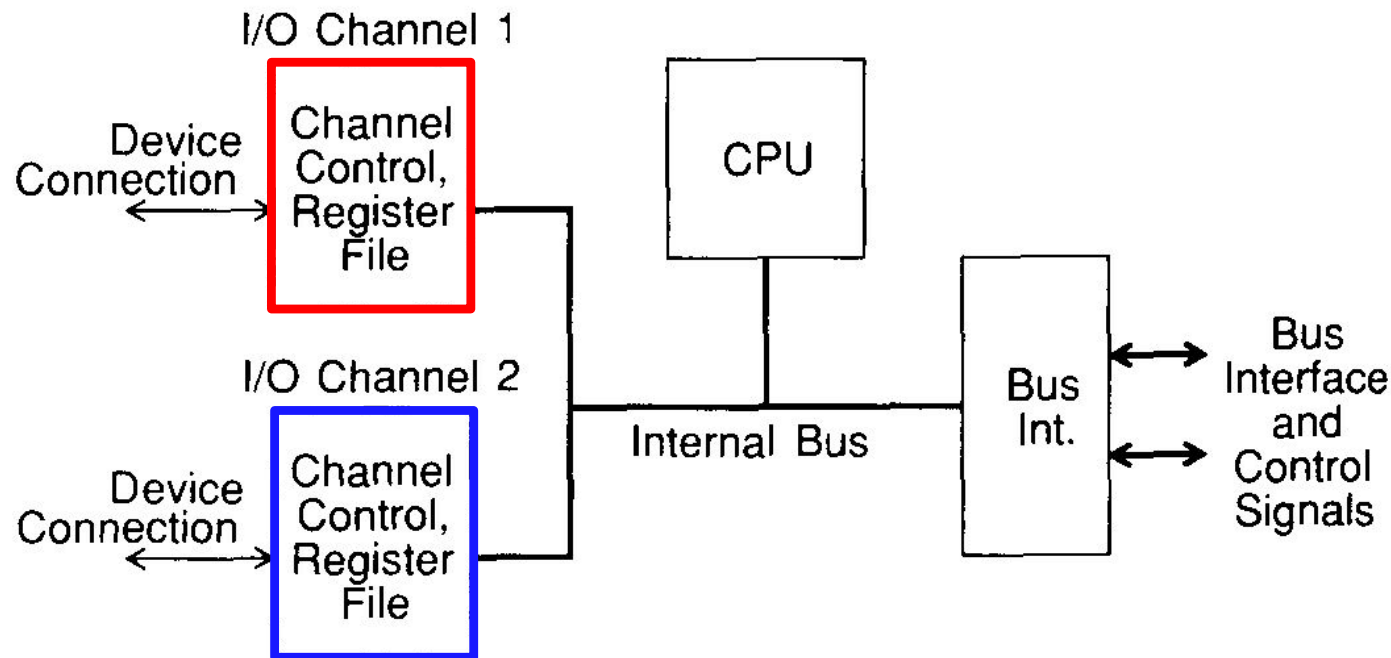
Példa: I/O channel – I/O csatornák

- I/O csatornák típusai (sebességük szerinti kategóriákat képeznek):
 - Multiplexeres: LS (lassú)
 - Blokkos: MS (közepesen gyors)
 - Selector channel: HS (nagysebességű)



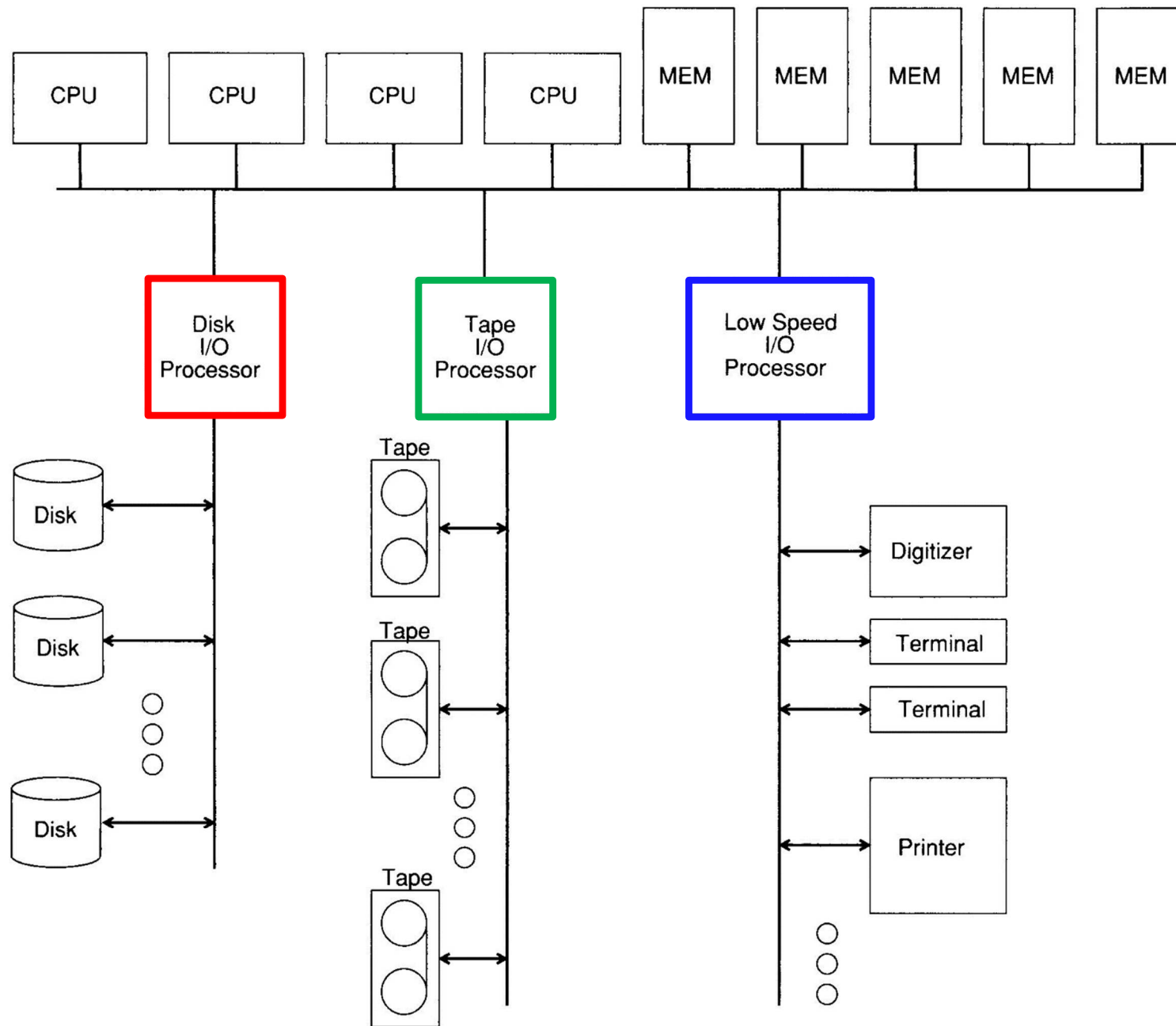
LS = Low Speed device
MS = Medium Speed device
HS = High Speed device

b.) I/O Processzorok (IOP)



- IOP: Intelligens eszközvezérlők/processzorok
- Saját, dedikált funkciókkal rendelkeznek (vezérlést, és interfészt biztosít más rendszerekkel – különböző sebességen)
- Példa: SCSI rendszer is egy IOP

Példa: több I/O-processzoros eszköz





PCI busz

PCI busz



- **PCI= Peripheral Component Interconnect (1992).** újszerű, a korábbiaktól önálló szabvány (ver 2.1, 2.2 és 2.3) volt: 33MHz-es órajel támogatás. Többprocesszoros rendszereket is támogat. Párhuzamos sín!
- Kapcsolatot a *processzor* és a *PCI sín* között egy *PCI HOST BRIDGE* biztosítja. Támogatja a többprocesszoros rendszereket, kompatibilis az ISA, EISA, MCA régebbi rendszerekkel. A PCI csatlakozóhelyekre speciális intelligens kártyák helyezhetők, amelyek képesek önálló adatátvitelt végrehajtani a processzor tehermentesítése céljából, így gyorsabb működés érhető el. (3.3 – 5V szabvány).
- Csatlakoztatható eszközök: hálózati-, hang-, videó-kártya./ Konfigurálása szoftveres úton, a BIOS-on keresztül történik. / Arbitrációs mechanizmust és **szinkron** protokollt használ.

PCI busrendszer tulajdonságai:

- A PCI 32 bites **multiplexált** címadat vonalat alkalmaz
 - Maximális átviteli sebessége (4-es „**burst**-onként” - löketszerűen) 133Mbyte/sec (= 4 byte*33.3MHz).
- (64 bites változata is létezik, főként szerverekben alkalmazzák. Jele. PCI-X!).
 - Ekkor a maximális átviteli sebessége 266Mbyte/sec (8byte*33.3MHz)
- Régebbi alaplapon az ISA ill. AGP bővítőhelyek mellett általában 3-4 PCI slot is található volt.
- A külső környezeti zavarok, zaj elkerülése végett, a PCI elemeket rövid úton kell összekötni, így a PCI jelek egy oldalán vannak kivezetve (PCI Speedway), sok földelést használva.
- Saját POST (Power On Self Test) önellenőrző kóddal is rendelkezik, a hibák felderítése végett, ami a számítógép bekapcsolásakor inicializálódik.

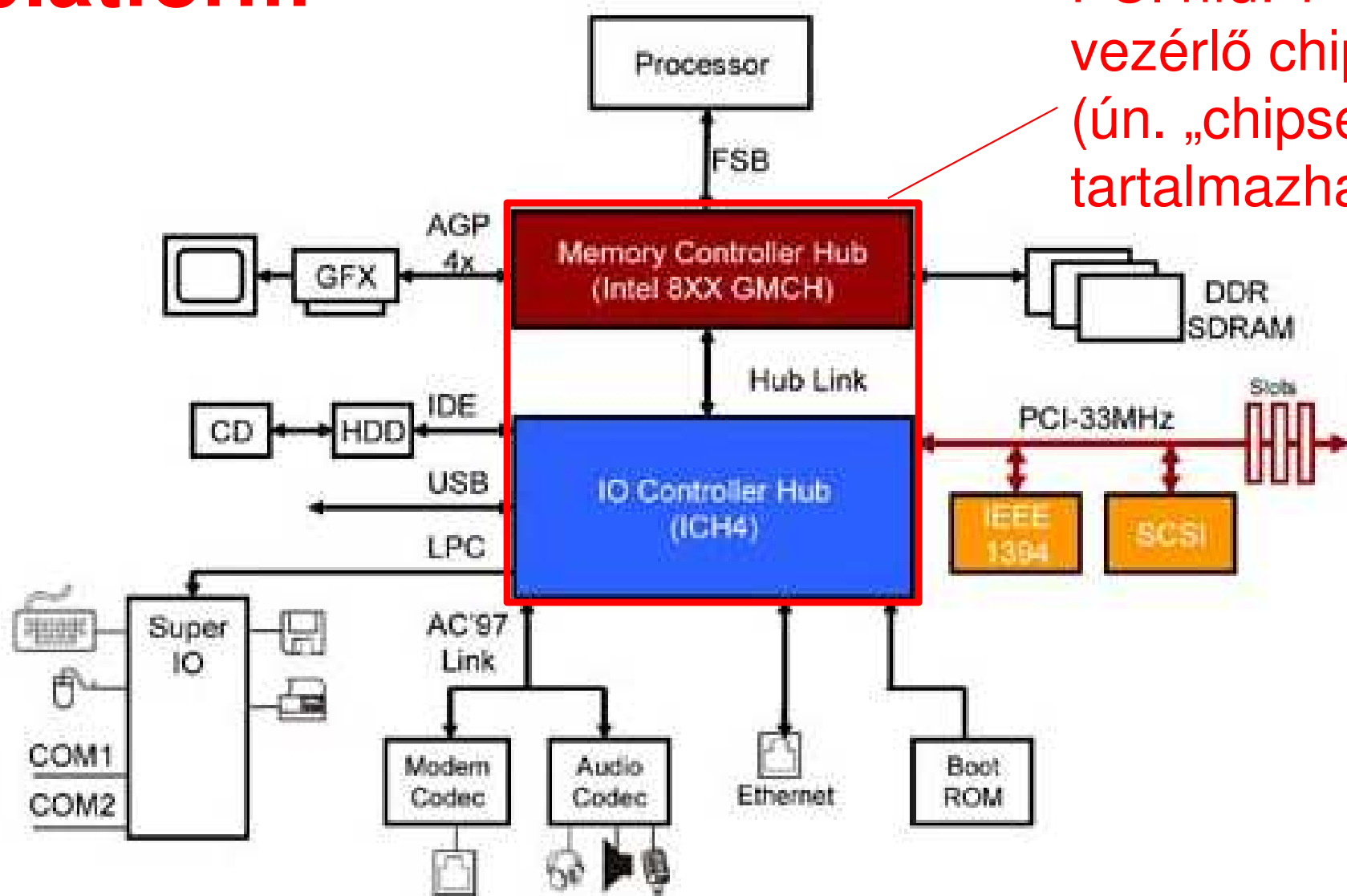
PCI-X



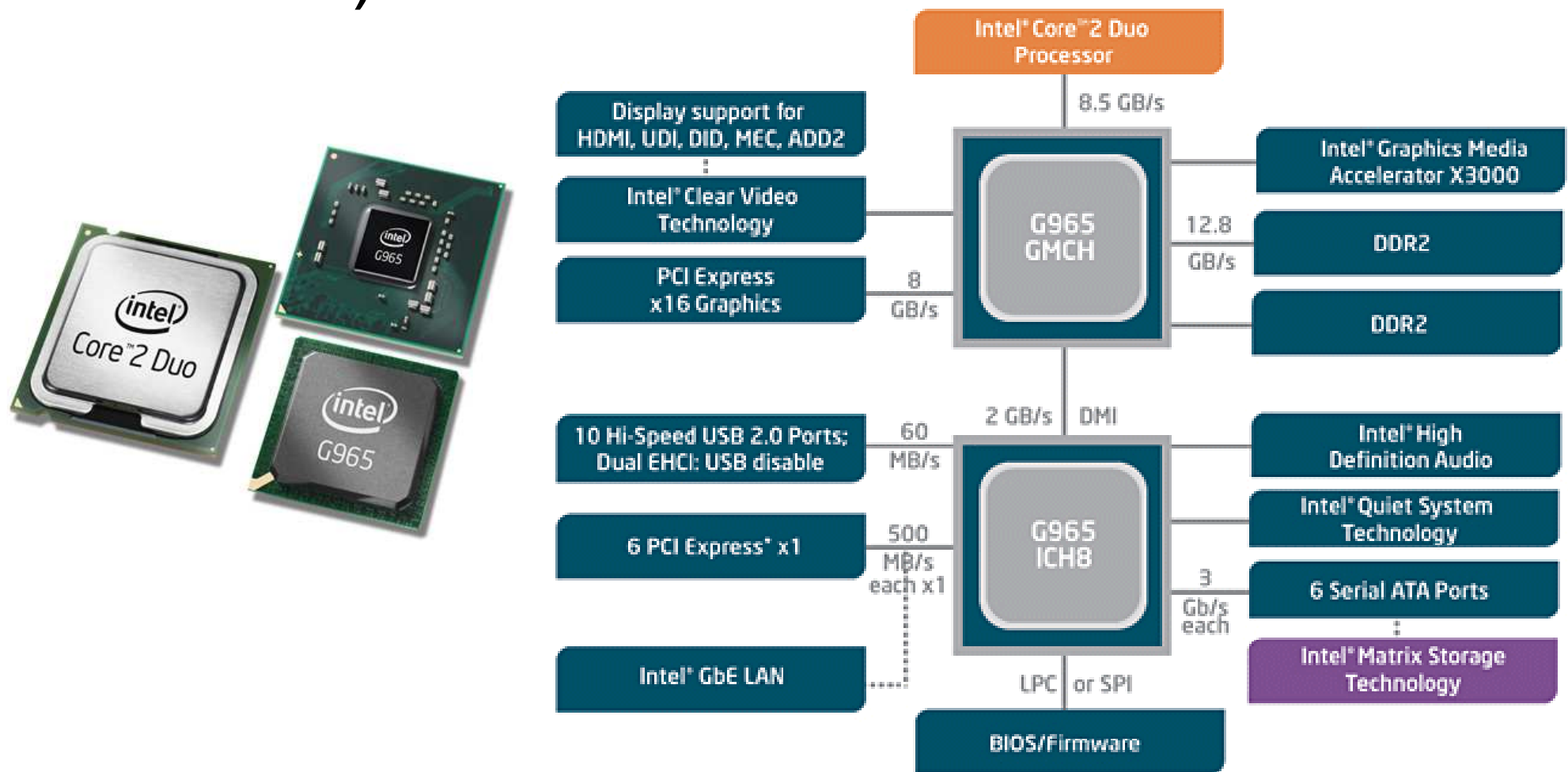
- PCI módosított változata
- **PCI-X** mint **extended**, bővített 64 bites
 - PCI-64bit/66.6MHz max 532 MByte/s
 - PCI-64bit/133.3MHz max 1064 Mbyte/s
 - PCI-X 1.0 & PCI-X 2.0
- Főként szerver alaplapon található:
 - nagyteljesítményű kártyák integrálása

Példa: 33MHz-es PCI busz alapú platform

PCI híd: 1 vagy 2 vezérlő chip-et (ún. „chipset”-et) tartalmazhat.

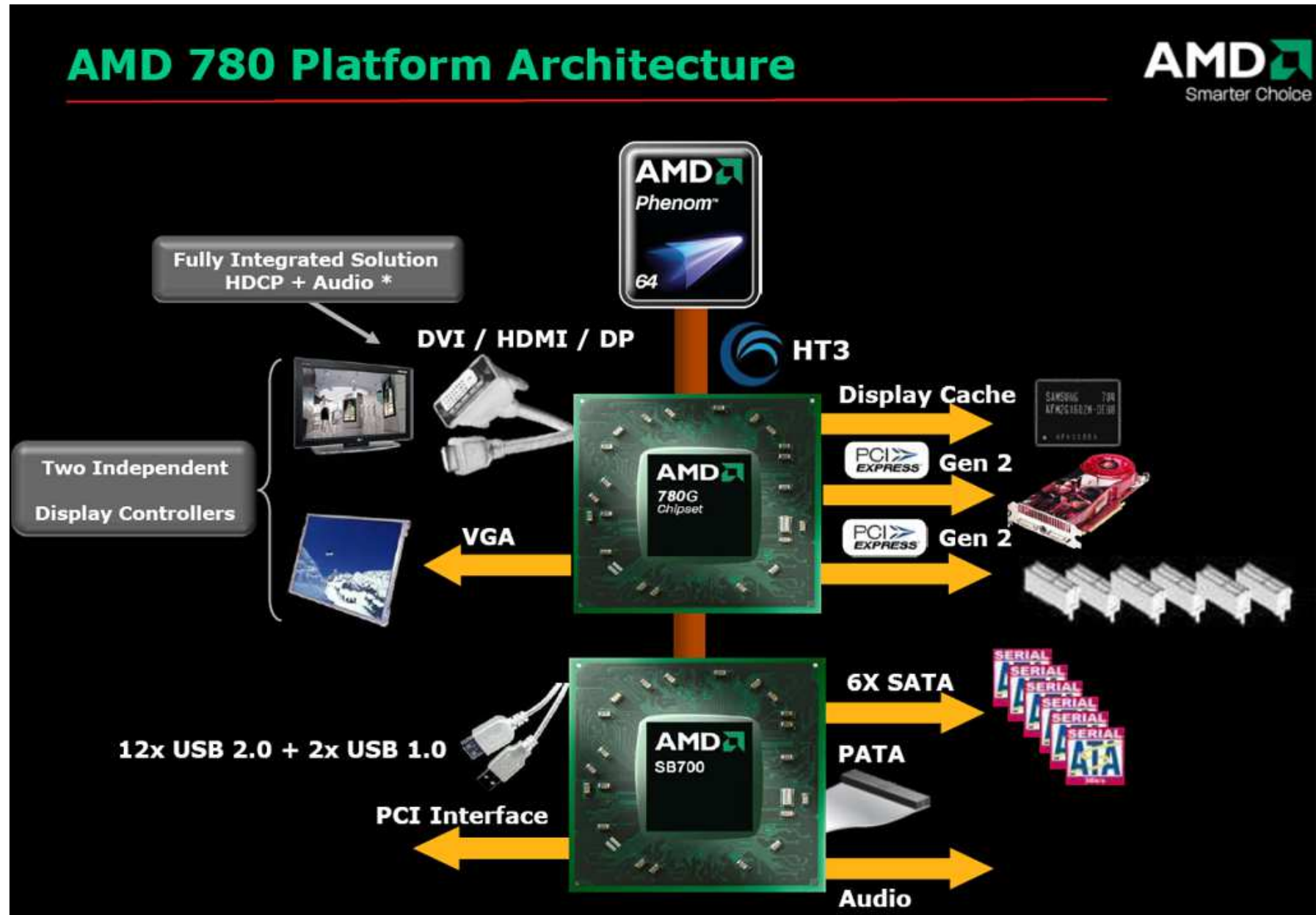


Példa: Intel chipset (Intel ICH8 – déli híd)



Integrált grafikus vezérlővel:
Graphics Media Accelerator 3000 (GMA 3000)

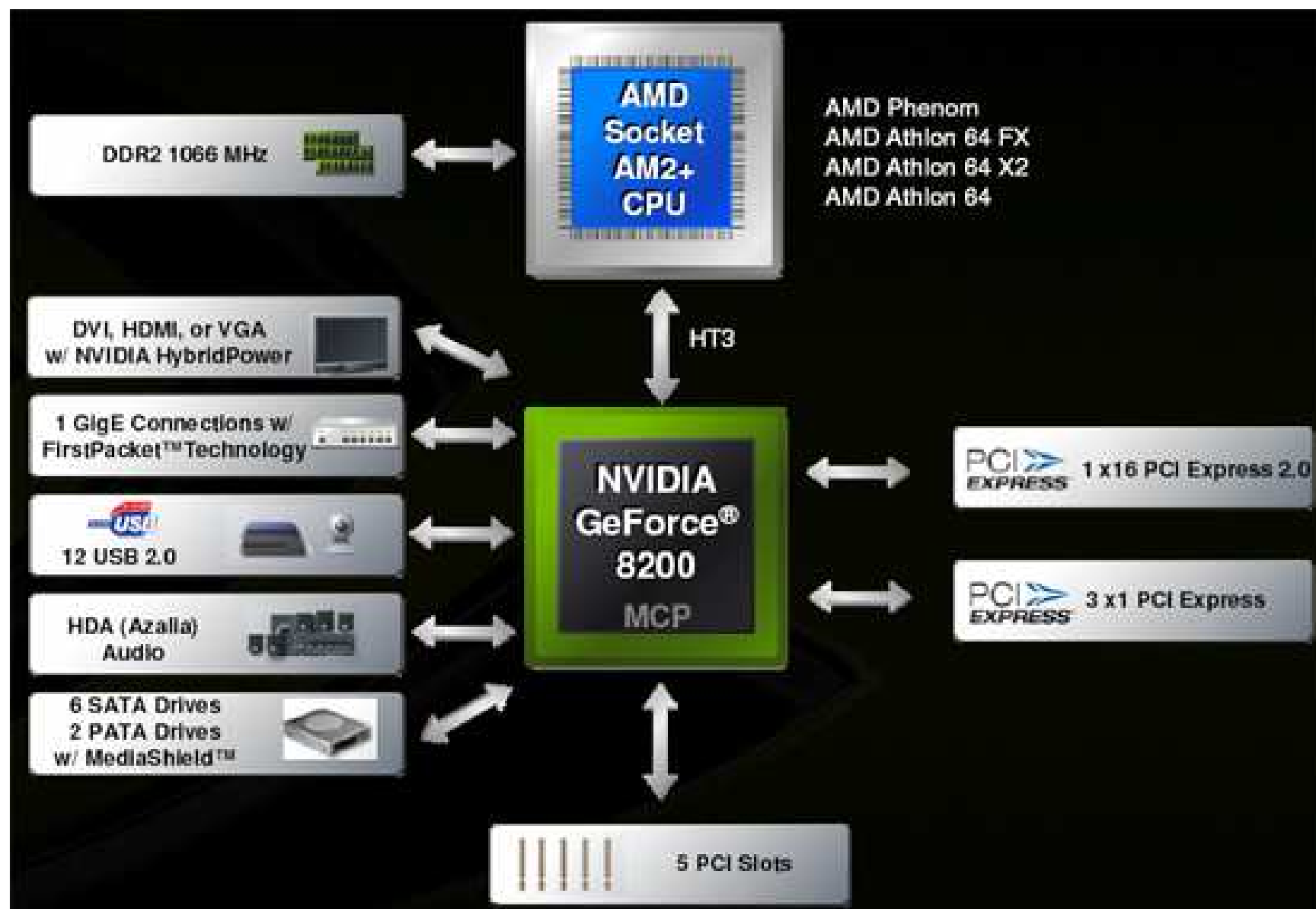
PI: Újabb generáció AMD780G



*2008.
március

PI: NVIDIA GF8200 AMD CPU-khoz

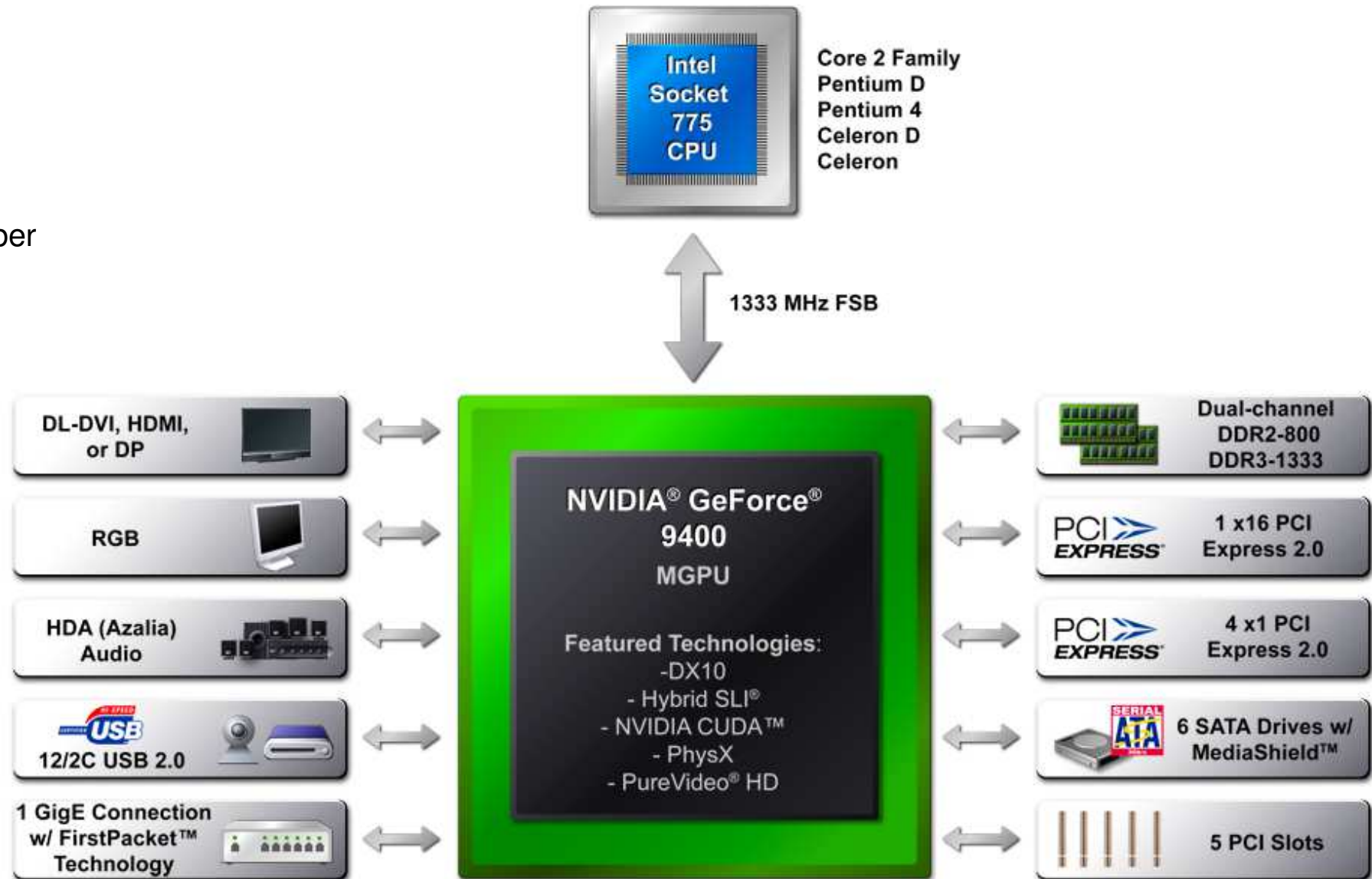
(MCP78 – Egy chipset-en É-D híd együtt!)



*2008.
május

PI: NVIDIA GF9400 MGPU Intel CPU-khoz (MGPU – Egy chipset-en É-D híd együtt!)

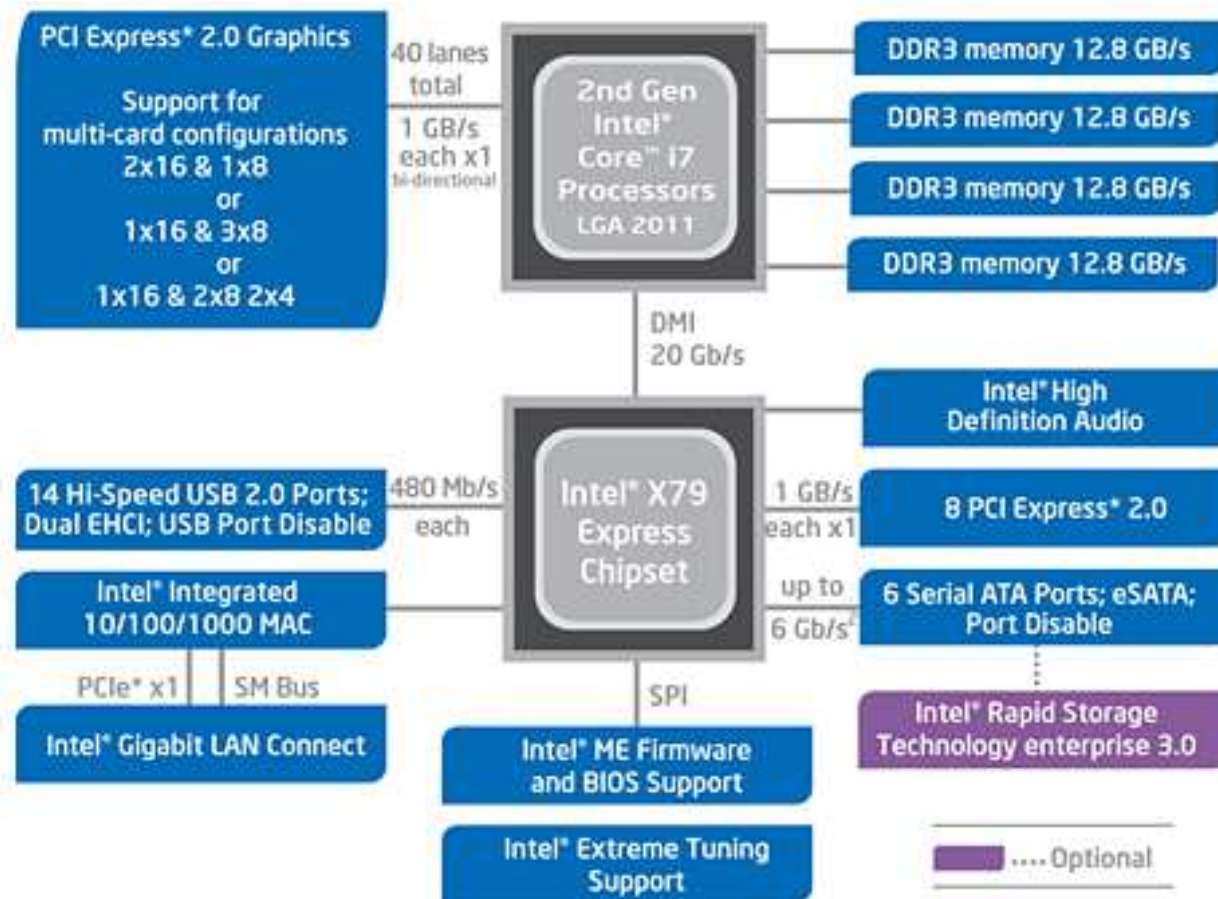
*2008.
november



PI: Intel Sandy Bridge-E (2011)

- Nincs északi híd (PCI-E integráltan ill., a DDR3 memória vezérlése is a CPU feladata), csak déli híd maradt (X79)

*2011.
november

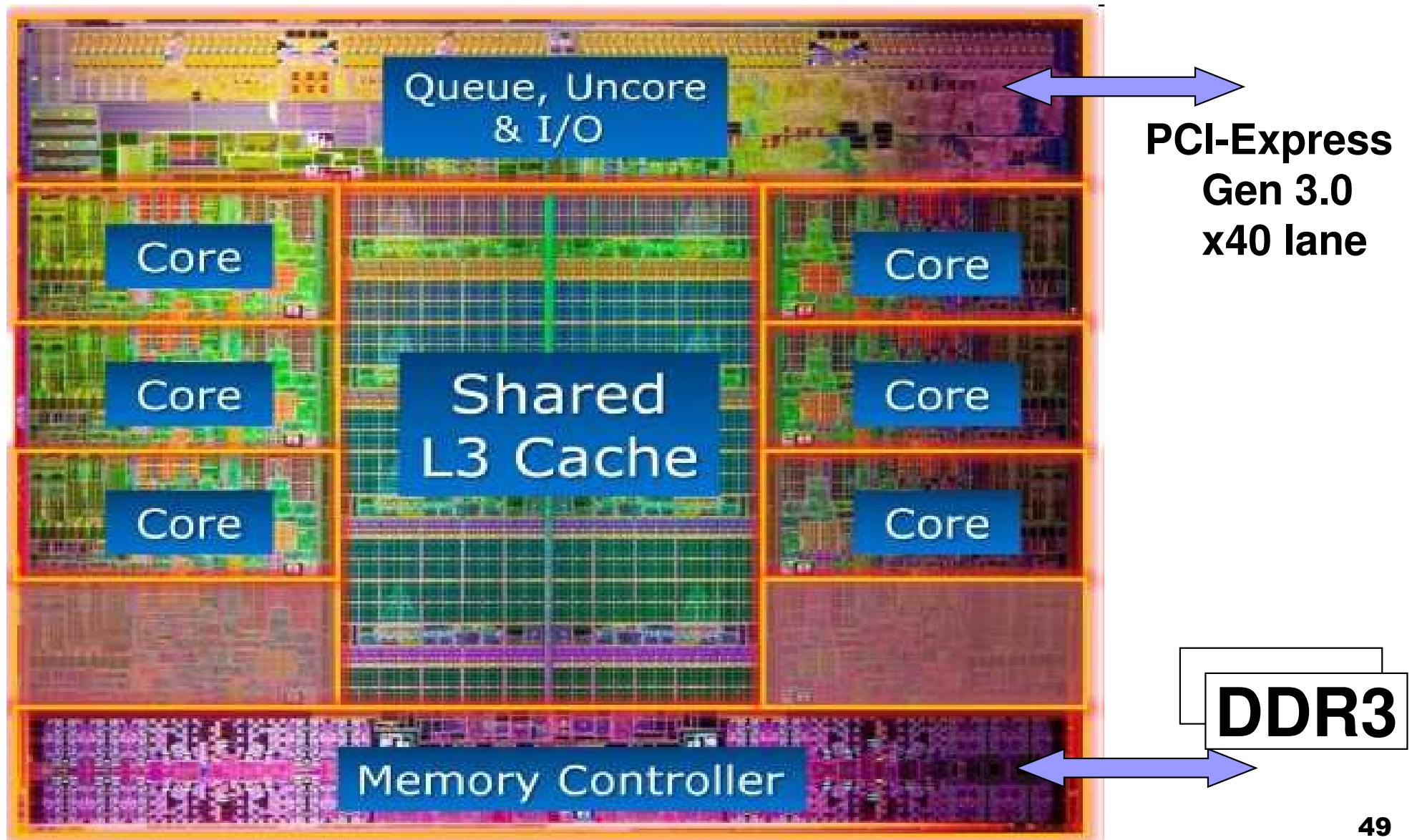


¹Theoretical maximum bandwidth

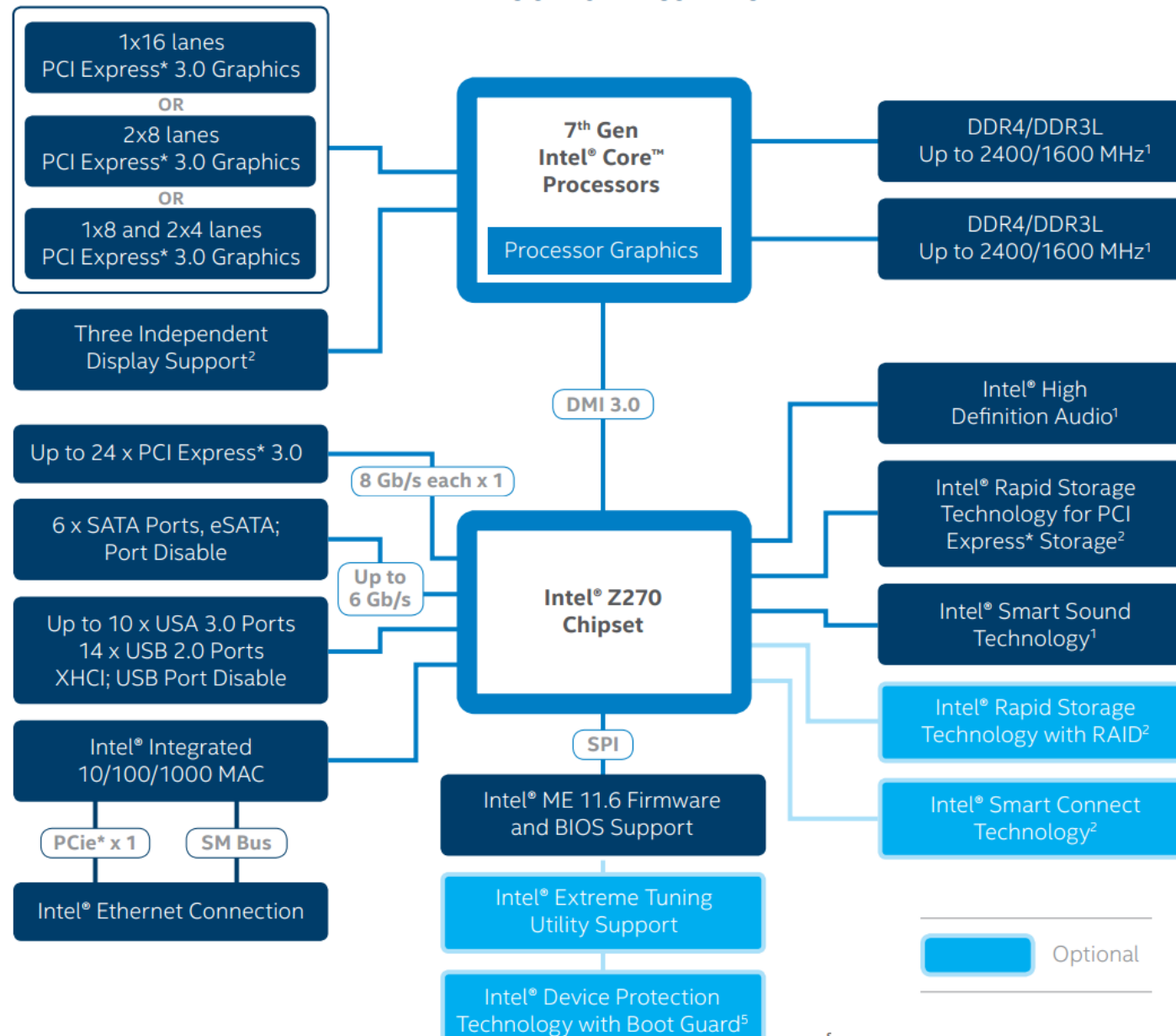
²All SATA ports capable of 3 Gb/s. 2 ports capable of 6 Gb/s.

Intel® X79 Express Chipset Block Diagram

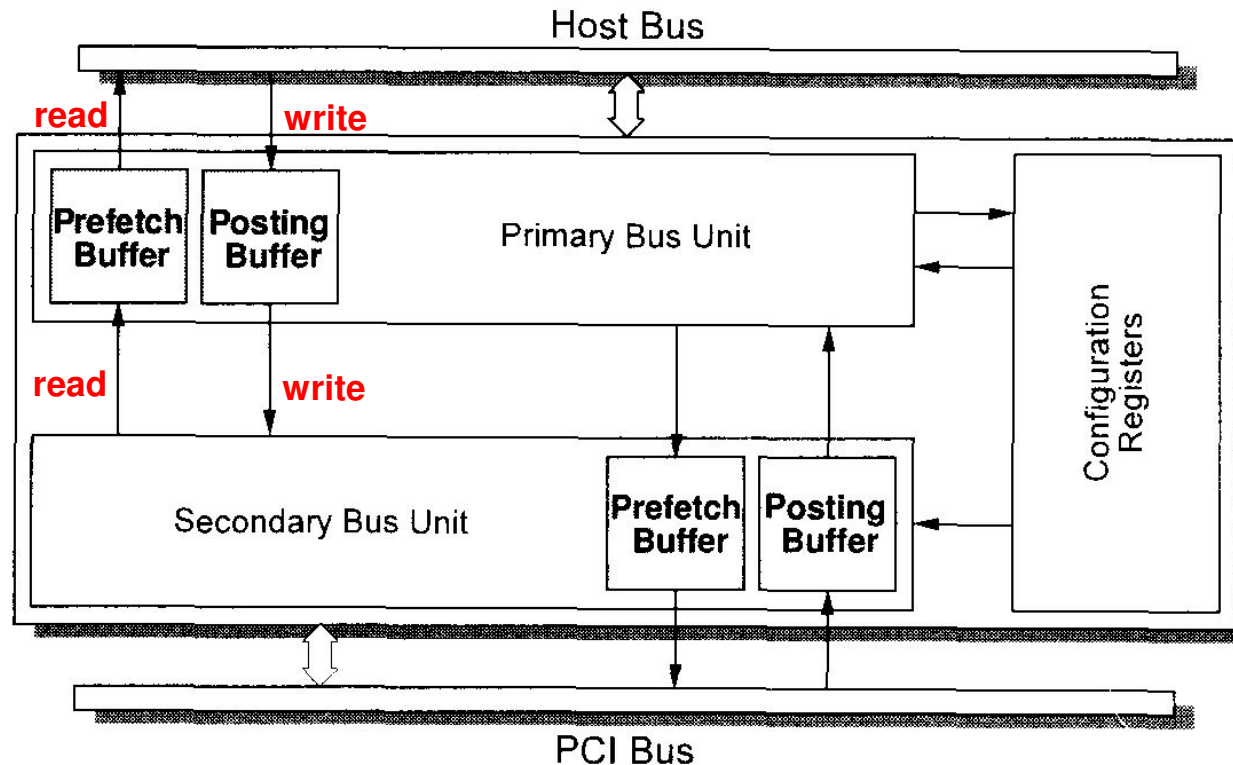
Intel Core i7-3960X – Sandy Bridge-E



INTEL® Z270 CHIPSET BLOCK DIAGRAM



PCI Bridge (CPU ↔ PCI)



- A PCI BRIDGE a PCI buszt köti össze a rendszerbusszal (CPU LOCAL BUS –al és MEMORY BUS –al egyben), tehát a **HOST busszal**. A konfigurációs regiszter a PCI BRIDGE részét képezi.
- A Host busz kapcsolódik az elsődleges busz egységhez, amelyben (*Read*) *Prefetch* és (*Write*) *Posting bufferek* találhatóak. Ez az egység közvetlenül is tud kommunikálni a másodlagos buszegységgel, amely a PCI buszhoz kapcsolódik. De a két egység megcímezhető a konfigurációs regiszteren keresztül is (szoftveres és regiszteres konfiguráció).

PCI busz fontosabb jelei #1

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.

- **CLK**: (Clock) PCI busz órajele (0-20-33MHz)
- **AD0-AD31**: (AD: Address/Data) multiplexált cím/adat vezetékek 32 bites üzemmódban (ahol AD00-AD07 byte-címzés esetén az LSB, míg AD24-AD31 byte-címzés esetén az MSB-t jelöli).
- **AD32-AD63**: 64 bites üzemmódban
- **IRDY**: (Initiator Ready) adatok olvasásakor (negált jel)
- **TRDY**: (Target Ready): adat írásakor (negált jel)
- **DEVSEL**: (Device Select) ez egy nyugtajel, a target ezzel nyugtázza, hogy a címet dekódolta
- **IDSEL**: a Chip Selectnek felel meg, adatírás vagy konfiguráció során lehet hozzáférni a chip-hez
- **STOP**: az adatforgalom megszakítását jelzi a target-nek
- **FRAME**: adatátviteli ciklus jelzése (negált jel)

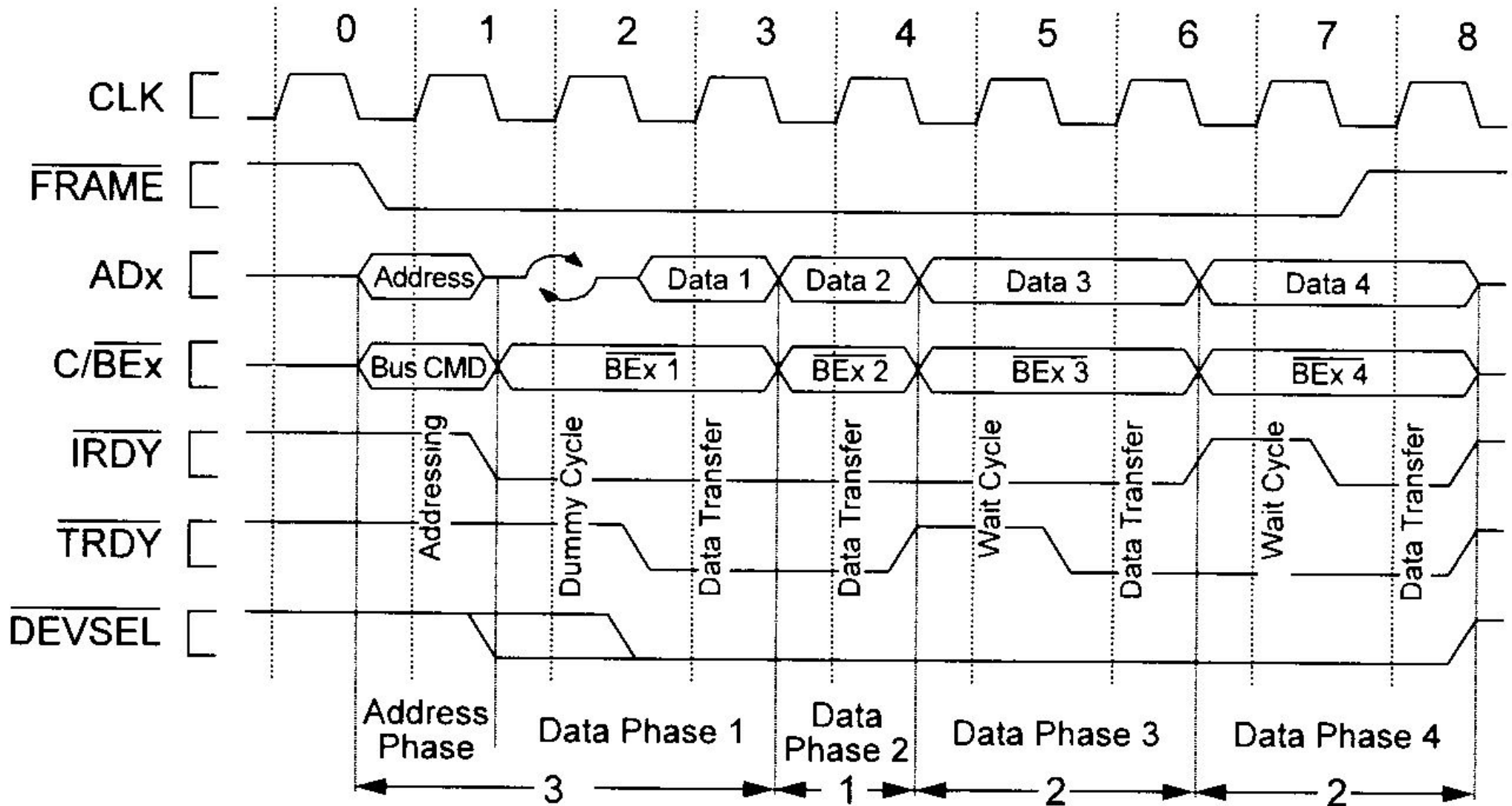
PCI busz fontosabb jelei #2

- **PAR:** (Parity) adatok és címek paritás ellenőrzése
- **C/BE0 – C/BE3:** (Command & Byte Enable) egy-egy jel egy byte-ot foglal magába. Megmutatja, hogy melyik byte tartalmaz érvényes adatot, ill. írunk, vagy olvasunk-e.
- **PERR – SERR:** (Parity error ill System error) hibajelek
- **INTA – INTD:** megszakításjelek (felfutó élre vezéreltek)
- **REQ:** (Request) sínhozzárendelés a kéréshez
- **GNT:** (Grant) sínhozzárendelés engedélyezéshez
- **TCK, TDI, TDO, TRST:** (Test Clock, Test Data in, Test Data Out, Test Reset) PCI sín tesztelésének jelei (JTAG-hez)
- **RST:** (Reset) regiszterek tartalmának törlése, és a PCI jeleinek kiinduló helyzetbe állítása

PCI írási/olvasási tranzakciók

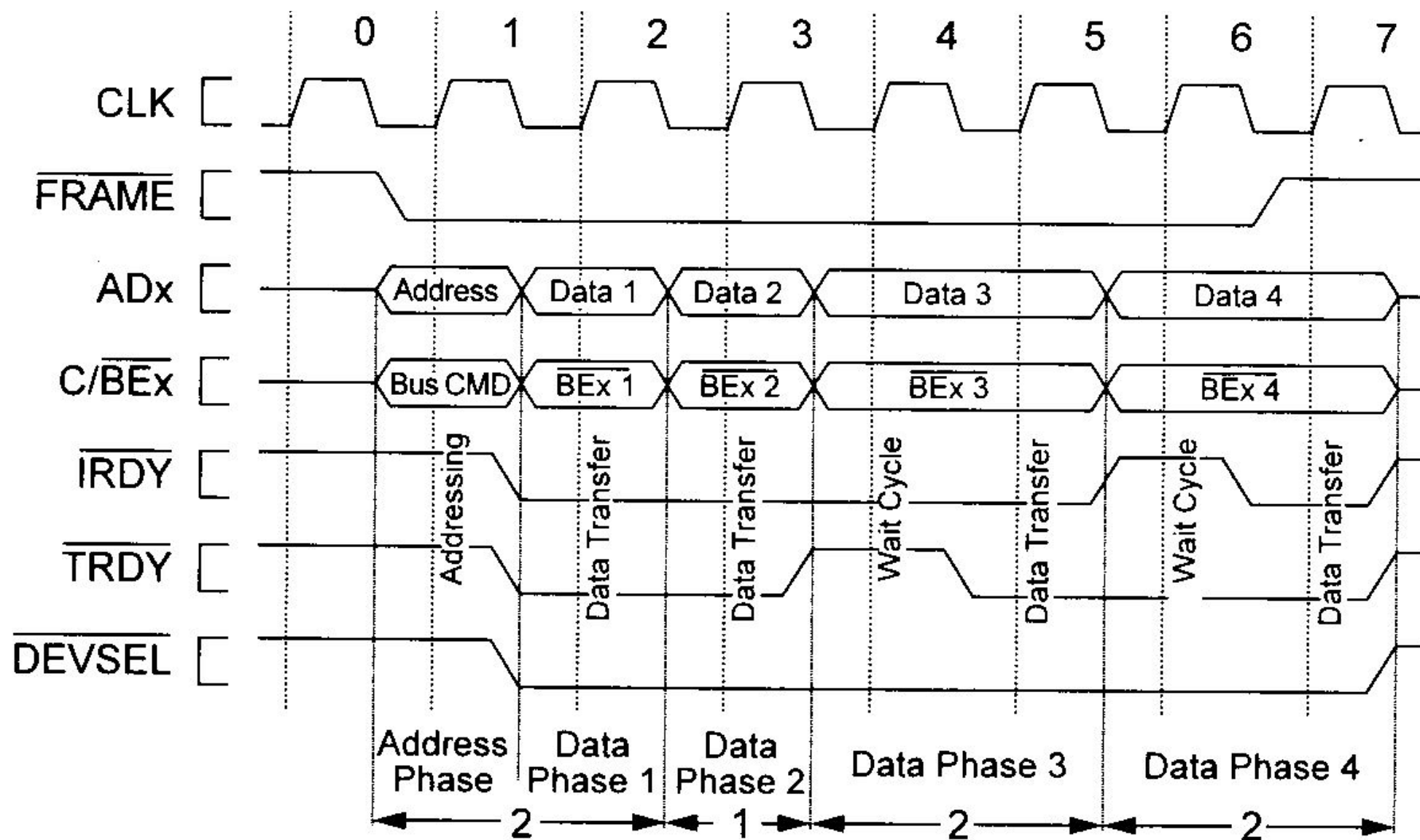
- **Burst = „löketszerűen”** egyszerre több adatot szeretnénk kiolvasni/írni. 1 cím kiadása után tipikusan **4 adat jön (4-es burst)**. A címzési fázis utáni első órajelciklusban az átvitel irányát módosítani kell, a közös multiplexált ADDR/DATA busz miatt. Olvasásnál ezért van szükség egy üres ciklusra (*Dummy*). Például: olvasásnál 3-x-x-x, vagy írásnál 2-x-x-x
- Miután az arbitráció során az egység sikeresen megkapta a vezérlést, a FRAME jellel inicializáljuk az adatátviteli folyamatot. Olvasásnál a címzés egy ciklus idejű, miután egy üres ciklus jön, majd az első adat megérkezéséhez szükséges ciklus (Esetünkben az első fázis 3 ciklus idejű: címzés + üres ciklus(ok)+TRDY adat). Az IRDY jel majd a TRDY jel alacsony jelszintre váltása után kezdődhet meg az adatok továbbítása *löketszerűen*, egymás után x-x-x ciklusonként.
- Olvasásnál: pl. 3-1-2-2= először 3 órajelciklust vár az első adatig, majd utána 1-2-2 ciklusonként jönnek az adatok (közben *wait* ciklusok lehetnek!)
- Írásnál nincs üres (dummy) ciklus, a cím kiadása után egyből mennek az adatok 2-1-2-2 ciklusonként, az **optimum 2-1-1-1)**

PCI olvasási ciklus (3-1-2-2 burst)



* Optimális 3-1-1-1.

PCI írási ciklus (2-1-2-2 burst)



* Optimális 2-1-1-1.



PCI-Express busz

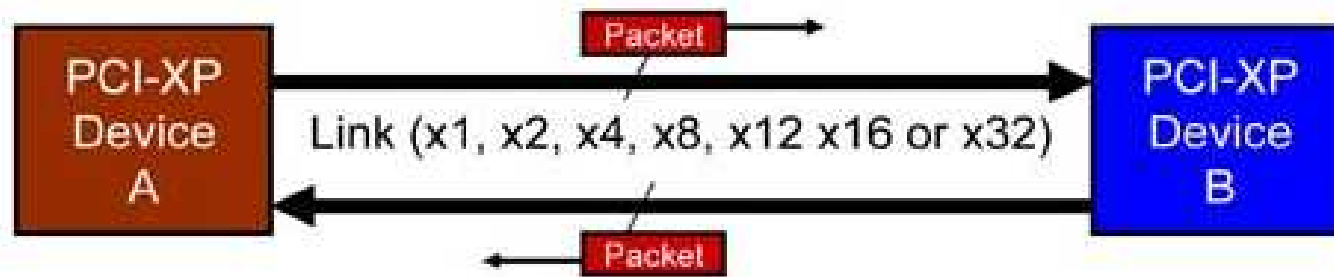
Kapcsolódó segédanyag:

- Addison.Wesley.PCI.Express.System.Architecture.eBook.chm
- http://www.pcisig.com/news_room/faqs/faq_express/

PCI-Express Busz



- Nagyteljesítményű, nagysebességű, P2P: **pont-pont kapcsolati protokoll, szinkron soros** buszrendszer (2001)
- **Duális simplex** (két-egyirányú) kommunikációt biztosít (ma már full-duplex)
 - Link: x1, x2, x4, x8, x12, x16, vagy max x32 lane-ből áll



- Lane (sáv): jelpárok a két irányban
- Hivatalos Jelölése: PCI-E, PCIe ! (≠ PCI-X !)
- Cél: felváltani a PCI, PCI-X és AGP buszokat
- 2003 óta létező szabvány. Jelenleg v5.0.

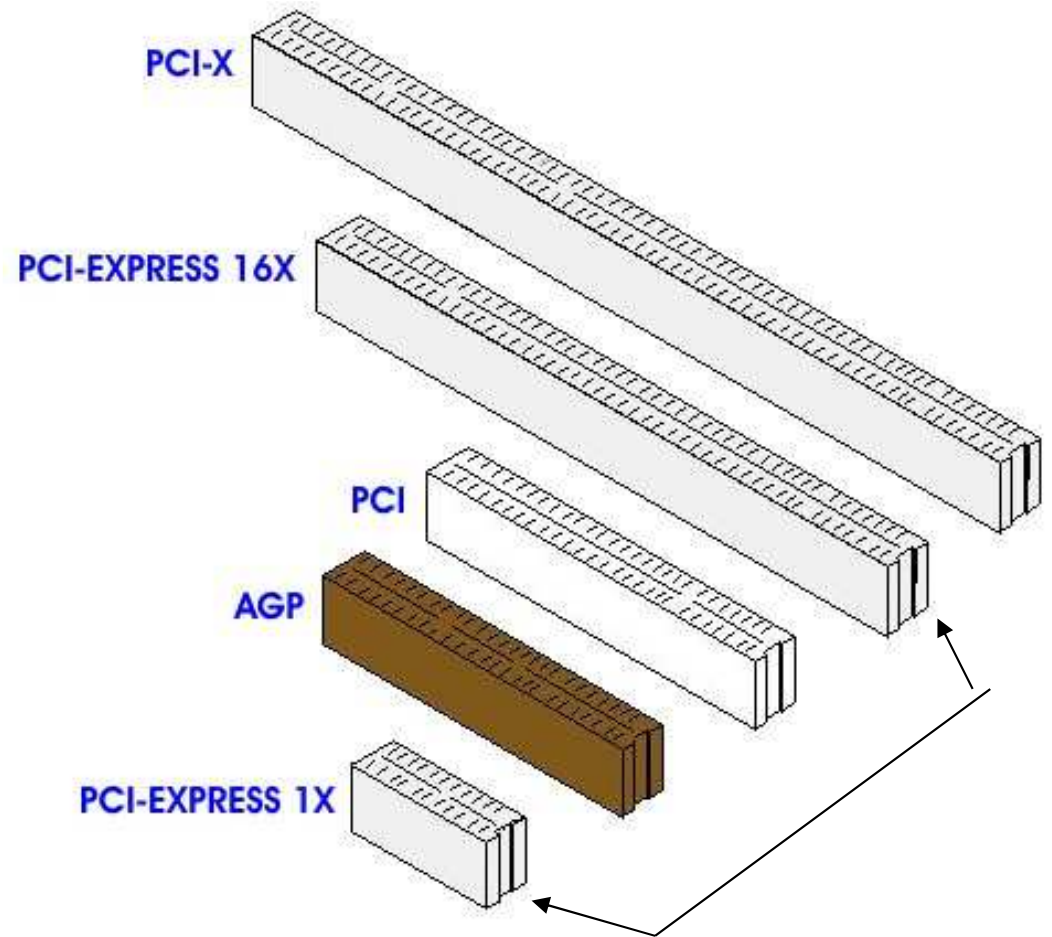
PCI Express tulajdonságai

- A PCI-nál az eszközök osztoztak a sínen, míg a PCI Expressnél egy *switchen/hub-on* keresztül érik el (P2P topológia) a sít (minden eszköz úgy látja, mintha saját külön sínnel rendelkezne).
Szinkron Soros kapcsolat!
- A **switch** gondoskodik a P2P kapcsolatok létrehozásáról és a vezérli a sín adatforgalmát. A switch és az eszközök közötti kapcsolatokat **link**-nek nevezik.
- Egy PCIe link duál szimplex (ma már duplex), azaz az adó és a vevő két egyirányú (kétirányú) csatornán keresztül forgalmaz. Minden link egy vagy több **lane**-ből (sáv) állhat.
- Egy **lane** egy *bájt* egyidejű átvitelét teszi lehetővé (250MB/s), ami a gyakorlatban maximálisan ~2,5 GB/s adatátviteli sebességet jelent.
- A PCIe x1, x2, x4, x8, x12, x16 és x32 lane-ből álló linkek létrehozását támogatja. A switch alkalmazása lehetővé teszi a rendelkezésre álló sávszélesség jobb kihasználását és az adatforgalom fontosság szerinti osztályozását
- Alacsony fogyasztás, illetve az energiatakarékossági funkciók támogatása

ALKALMAZÁS: grafikus kártyák, merevlemezek, SSD-k, WiFi, Ethernet vezérlők, stb.

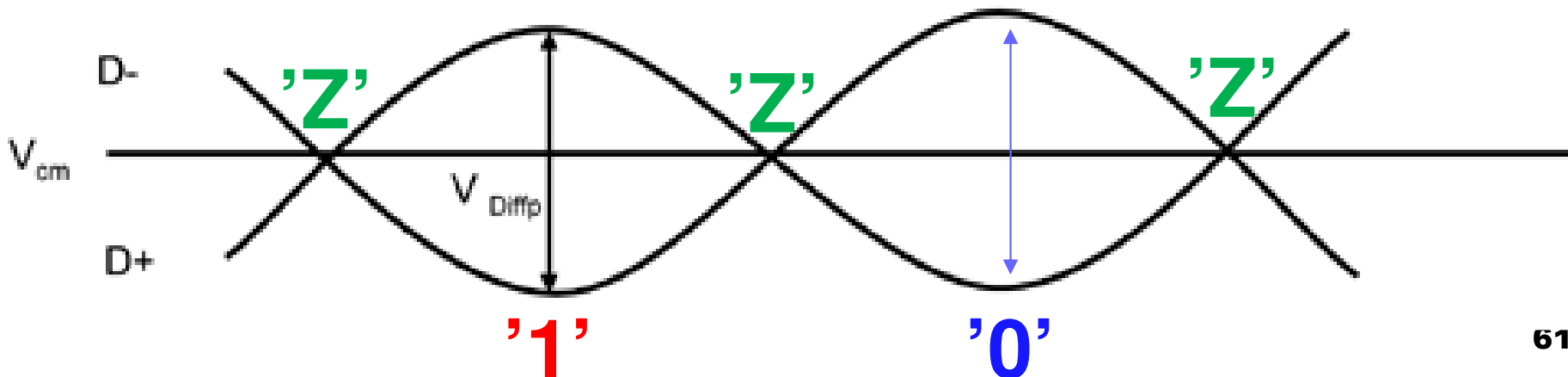
Bővítőhelyek (slotok)

- PCI (32 bit)
- PCI-X (64 bit)
- **PCI-Express**
 - x1
 - x16
 - x32
- AGP (Intel Accelerated Graphic Port)
 - x1-x8

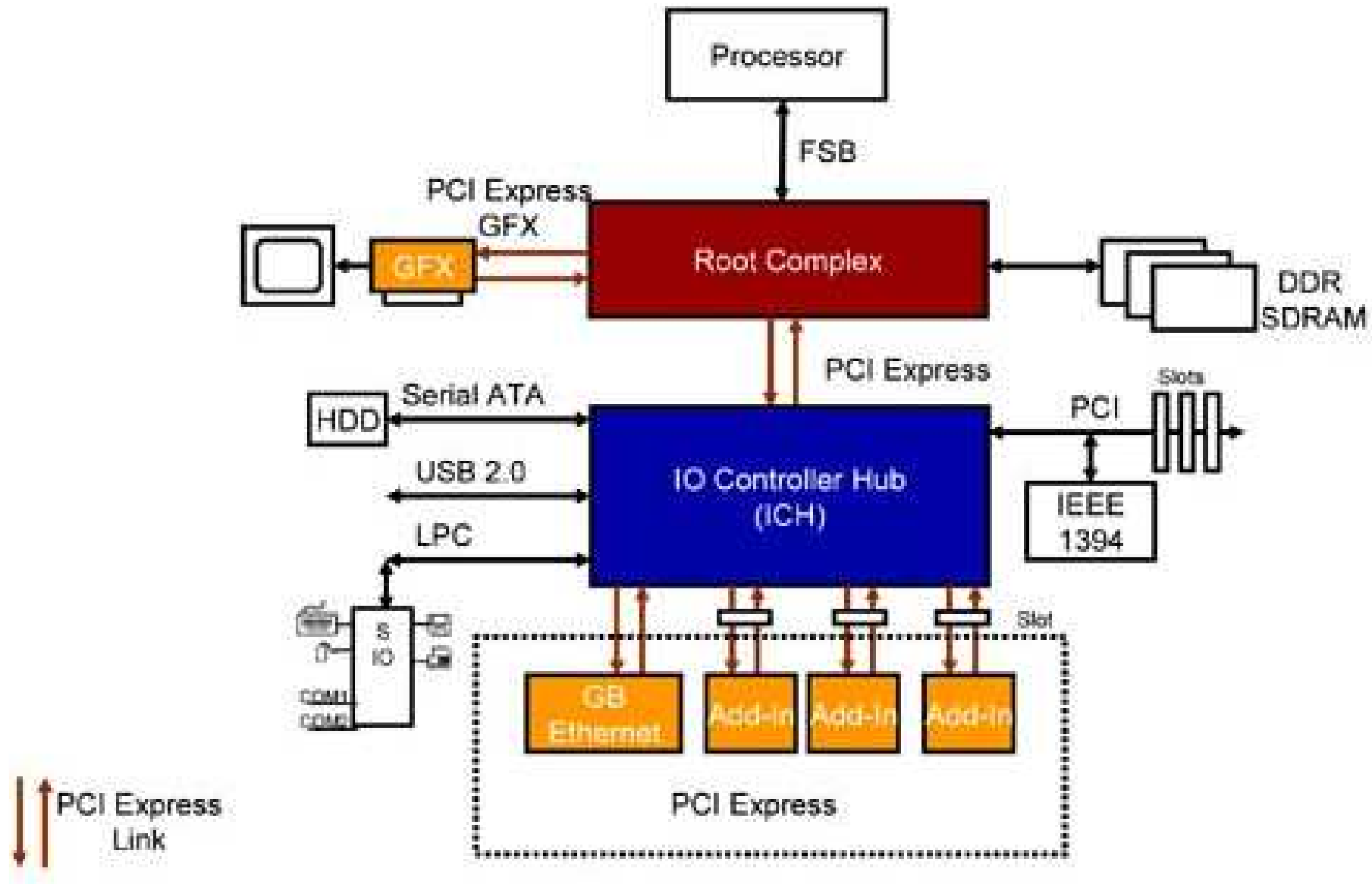


Működése: Differenciális jel

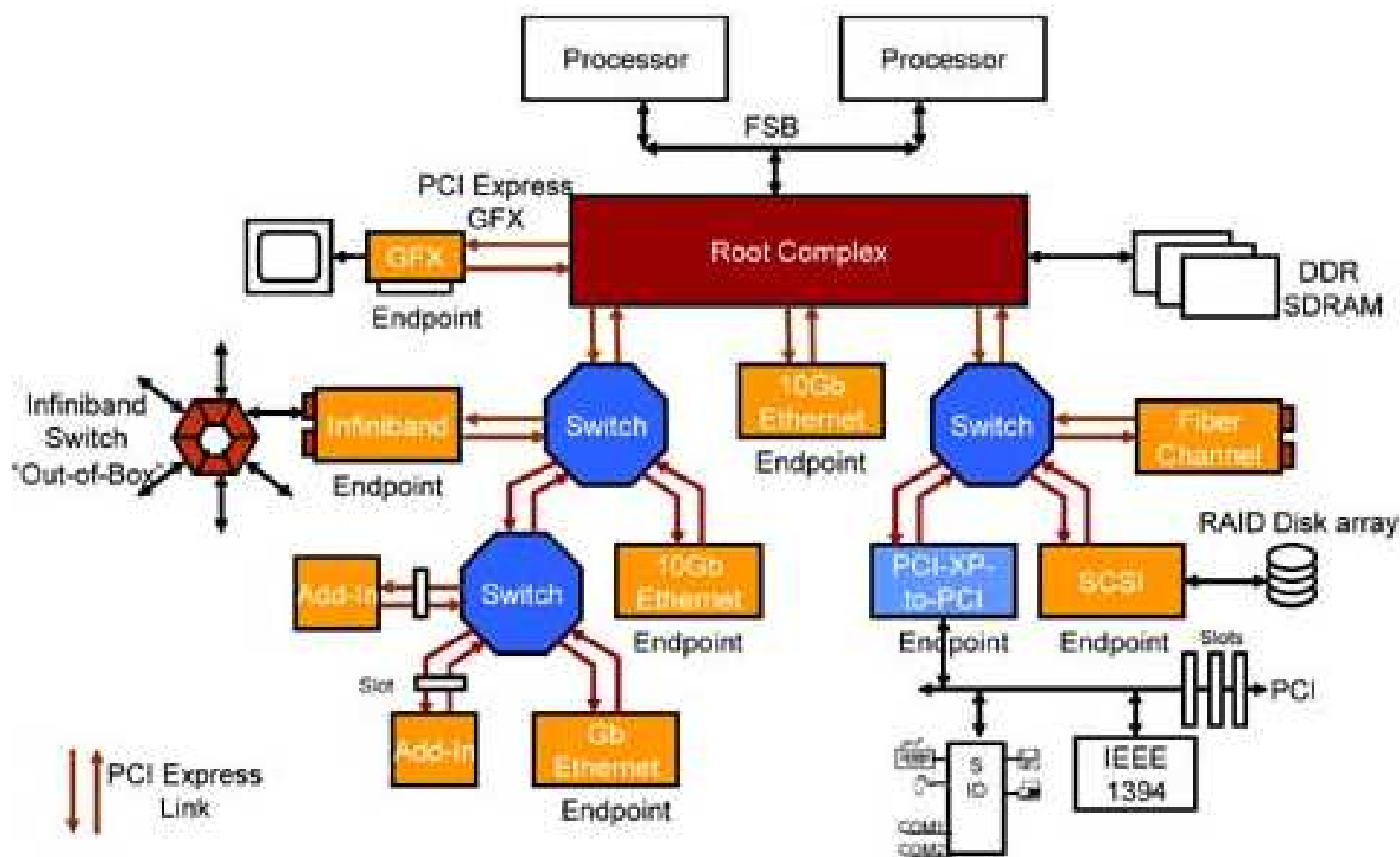
- Differenciális driver-ek és receiver-ek találhatók mindenegyes PCIe portnál.
- Ha *pozitív* feszültség különbség van a D+ és D- terminálisok között, akkor **logikai '1'**-et reprezentál.
- Ha *negatív* feszültség differencia van a D+ és D- között, akkor pedig **logikai '0'**-t mutat.
- Ha *nincs* feszültség különbség a D+ és D- között, az azt jelenti, hogy a driver nagy-impedanciás ún. **tri-state ('Z')** állapotban van, amely az eszköz (tétlen) állapotát, és a *line low-power* állapotát jelzi.
- PCI Express iel elektromos karakterisztikáia (VDS):



Olcsó asztali gépes PCI-Express rendszer



PCI-Express szerver alapú, multi-processoros rendszer esetén



- **Switch:** több-portos eszköz, Link-ek kapcsolhatók hozzá

PCI-Express v1.0 tulajdonságai (2003)

- Csomag (Packet) kapcsolt protokoll
- Sáv szélességek és órajelek:
 - Max 2.5 Gbit/sec/sáv/irány (2.5 GHz)
 - 8b/10b kódolás (10bits/byte)
 - Data rate: 250 MBit/sec/sáv/irány (x1 lane!)
 - Transfer rate: Max. elméleti határ ~**2.5 GT/s** (x16 lane: 16 sávon) // **Gigatransfer** (GT/s):
- **T, Transfer** = adatátviteli műveletek/szavak száma, amely egy adott csatornán másodpercenként bekövetkezhet. Hívják még Transfer rate-nek vagy Sample rate-nek is.
- Pl. Tegyük fel, hogy egy szó 1 byte-os = 8 bites
 - Ekkor 1 GT/s szó = $8 \cdot 10^9$ Byte /s = 7.45 GByte/s ~ 8x8 Gbit/s = 64 Gbit/s (ez már bitrate-t jelent!)

PCI-Express v1. x összesített sávszélesség adatok

- Max 2.5 GBit/s == 2.5 GTransfer/s
- 8b/10b kódolás (osztva 10-el)
- n Lane: „x n” (szorozva)
- szorozva 2-vel (irányonként)

Table 1-3. PCI Express v1.0 Aggregate Throughput for Various Link Widths							
PCI Express Link Width	x1	x2	x4	x8	x12	x16	x32 (nem támog.)
Aggregate Bandwidth (GB/s)	0.5	1	2	4	6	8	16

PCI-Express v2.x tulajdonságai (2007)

- Csomag (Packet) kapcsolt protokoll
- Sáv szélességek (órajelek) duplájára növekedtek:
 - 5 GBit/sec/sáv/irány (5 GHz)
 - Data rate: 500 MB/sec/sáv/irány (x1 lane)
 - **Transfer rate:**
 - ~**5 GT**/s (x1 lane)
 - Max. elméleti határ : ~80 GT (x16 lane)
 - Nagyobb fogyasztású (250-300W) eszközöket is támogat
 - Már külső eszközöket is támogat (10m-es kábel)
 - Input-Output Virtualization (IOV): hatékonyabbá teszi az ugyanazon hardveren futó virtuális gépek működését azáltal, hogy segíti a PCIe -eszközök megosztását - kapcsolati seb. SW-szintű beállítása
 - PCIe v1.1-re visszafelé kompatibilis

PCI-Express v3.x tulajdonságai (2011-től)

- Data rate: 1 GBit/sec/sáv/irány (x1 lane)
- Sáv szélesség ismét meg fog duplázódni: max 8 GT/s (128b/130b kódolás).
 - „PCIe 2.0 delivers 5GTs but employed an 8b/10b encoding scheme which took 20 % overhead on the overall raw bit rate. By modifying the requirement for the 128/130b encoding scheme, PCIe 3.0's effectively delivers double PCIe 2.0 bandwidth: 1.5 % overhead!”
 - 1 Gbit/sec/sáv/irány (x1 lane)
- Továbbfejlesztett jel és adat integritás:
 - receiver-transmitter,
 - PLL (phased-locked-loop)
 - csatorna részek optimalizálás
- PCIe v.2.x-re visszafelé kompatibilis
- Bevezetése: 2011. januárjában jelent meg.

PCI-Express v4.x tulajdonságai (2016)

- Data rate: 2 GB/sec/sáv/irány (x1 lane, max 16 lane)
- Sáv szélesség ismét meg fog duplázódni: max ~16 GT/s (8b/10b kódolás módosításával).
 - Több változtatás
 - Továbbfejlesztett jel és adat integritás:
 - Power optimizations (active / idle states)
 - Max power consumption: 375 W total (1×75 W + 2×150 W)
- PCIe v.3.0-re visszafelé kompatibilis
- Bevezetése: 2017 végtől (többszöri csúszás)
 - BroadCom 200 Gbit Ethernet vezérlőjében először
 - AMD X580 chipset (2019. jan jelent csak meg)

PCI-Express v5.x tulajdonságai (2019)

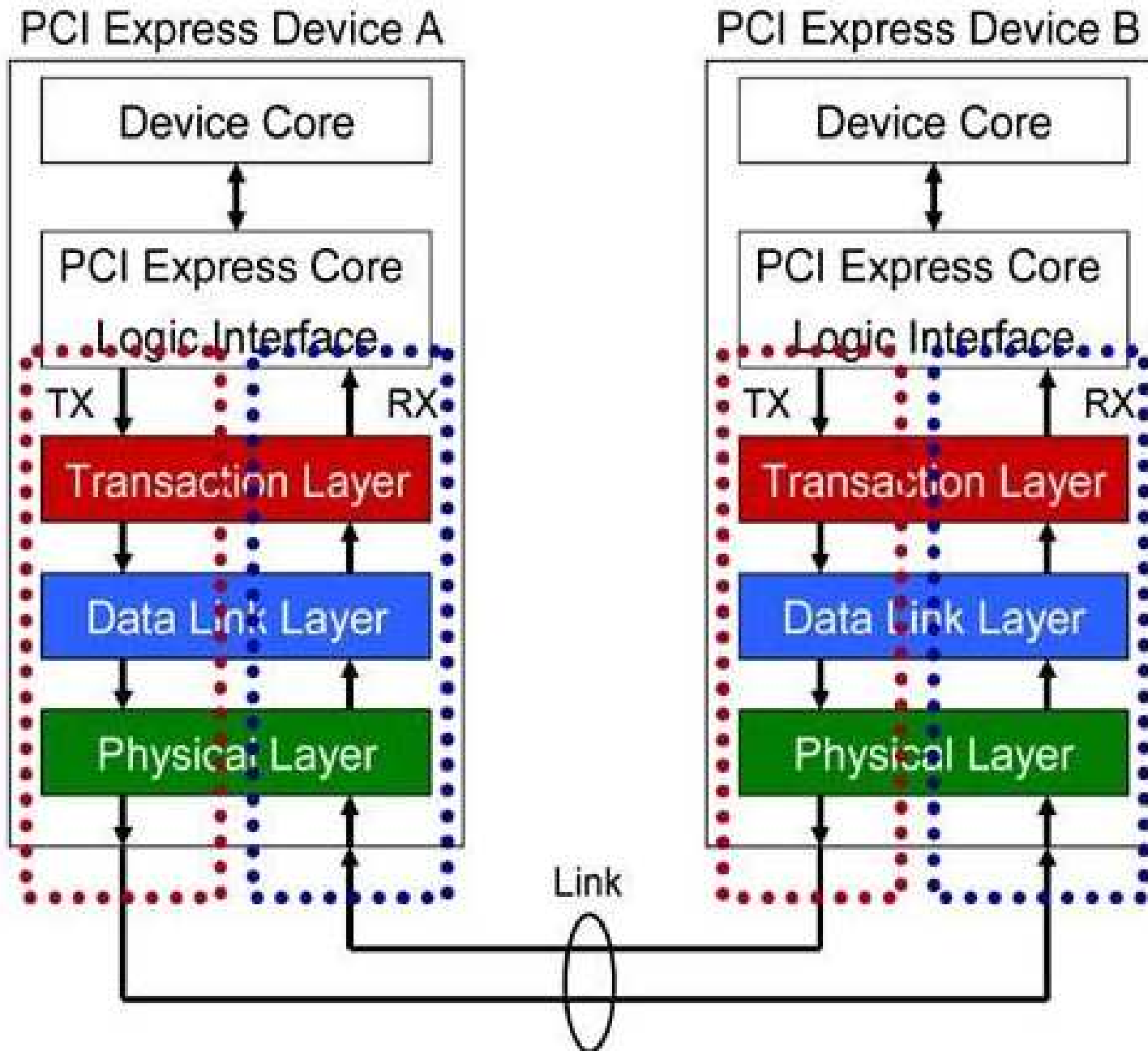
- Data rate: ~4 GB/sec/sáv/irány (x1 lane, max 16 lane)
- Sáv szélesség ismét duplázódik: max ~32 GT/s (8b/10b kódolás módosításával).
 - Továbbfejlesztett jel és adat integritás:
 - Power optimizations (active / idle states)
 - Max power consumption: 375 W total (1×75 W + 2×150 W)
- PCIe v.4.0-re visszafelé kompatibilis
- Tervezett bevezetése: 2019 eleje (csúszás)
 - BroadCom 200 Gbit Ethernet vezérlőjében először
- Jelenleg már a v.6.x szabványon dolgoznak, várható megjelenés 2021. ~64 GT/s IO sáv szélességet garantálnak.

PCI-Express (v1.0 ... v6.0)

📶 I/O BANDWIDTH DOUBLES
EVERY 3 YEARS

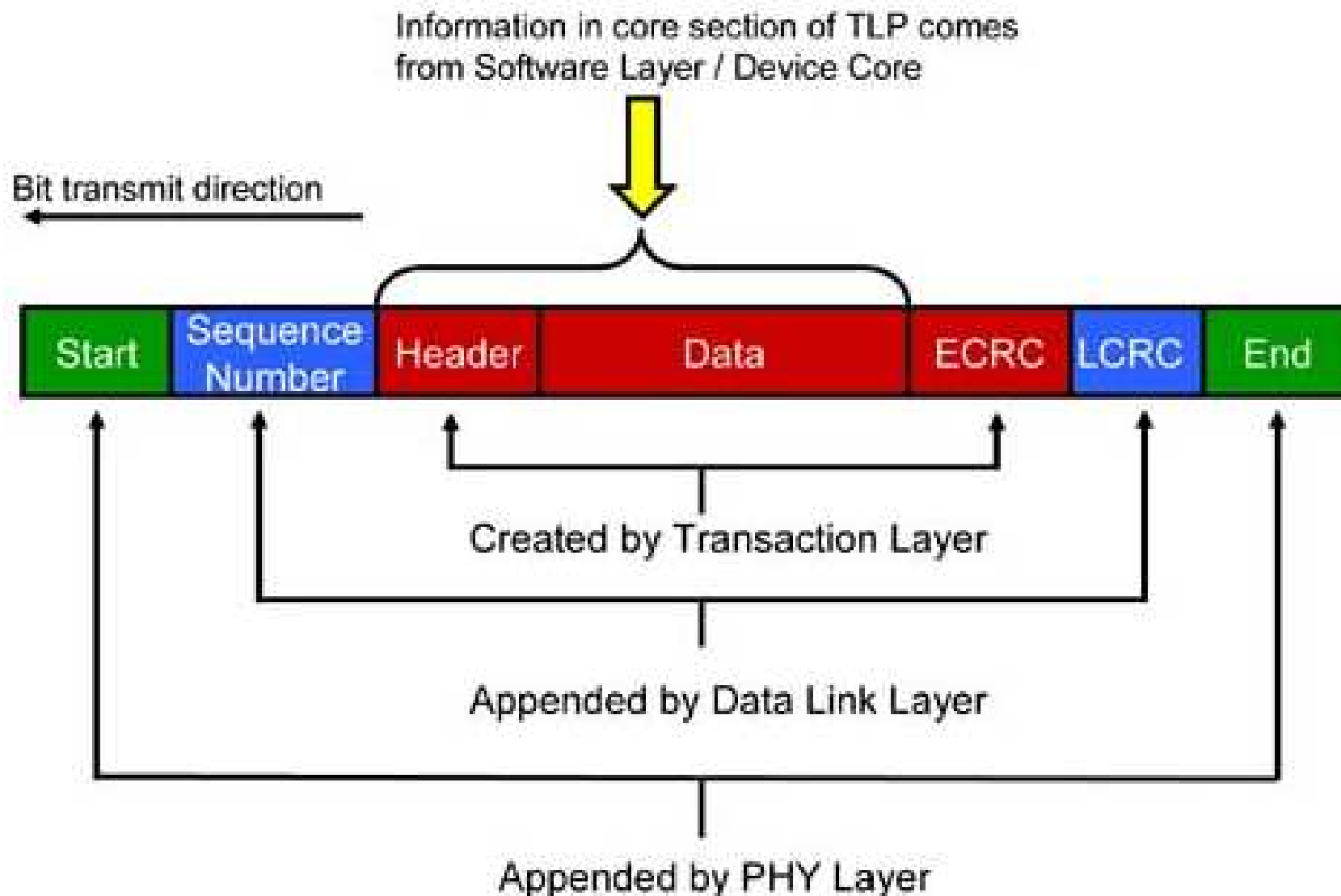


PCI Express rétegszerkezete



- **ISO OSI modell alsó négy! rétegét implementálja**
 - Fizikai,
 - Adatkapcsolati,
 - Hálózati és Szállítási réteg egyben,

Tranzakciós réteg csomagjai (keret)





SCSI busz

Korai párhuzamos SCSI buszrendszer tulajdonságai



- **SCSI= Small Computer System/Standard Interface:** komplex, *intelligens, sínorientált eszköz* (merev-, hájlékonylemez, CD-ROM, szalagos egység, scanner...) interfész. Különféle perifériák illesztésére, a CPU tehermentesítésére fejlesztették ki, az operációs rendszertől független felülettel rendelkeznek. Sokoldalú eszköz, mivel nemcsak PC-s környezetbe, hanem UNIX munkaállomásokba és Apple-Mac gépekbe is integrálható.
- **Tulajdonságai:** A korai párhuzamos SCSI buszon általánosan 8 eszközt definiál (mai soros SCSI-n több x1000 eszköz csatlakoztatható) legnagyobb sebessége 160Mbyte/s, max buszhossz 12m (UW-160) volt. Csatlakoztatható a - típustól függően - több belső és 1 külső (újszerűsége) eszköz is!
- Az eszközöket egyetlen vezérlő a **hostadapter** kezeli, amely a számítógépes rendszer eszközeinek kapcsolatát építi fel a SCSI buszrendszerrel. Nagysebességű, párhuzamos blokkos átvitelt biztosított a CPU és perifériák között. A korai szabványos SCSI csatlónak 50 (v. 68) lába volt, amelyből 9 vezérlő-, 9 adatvezeték.

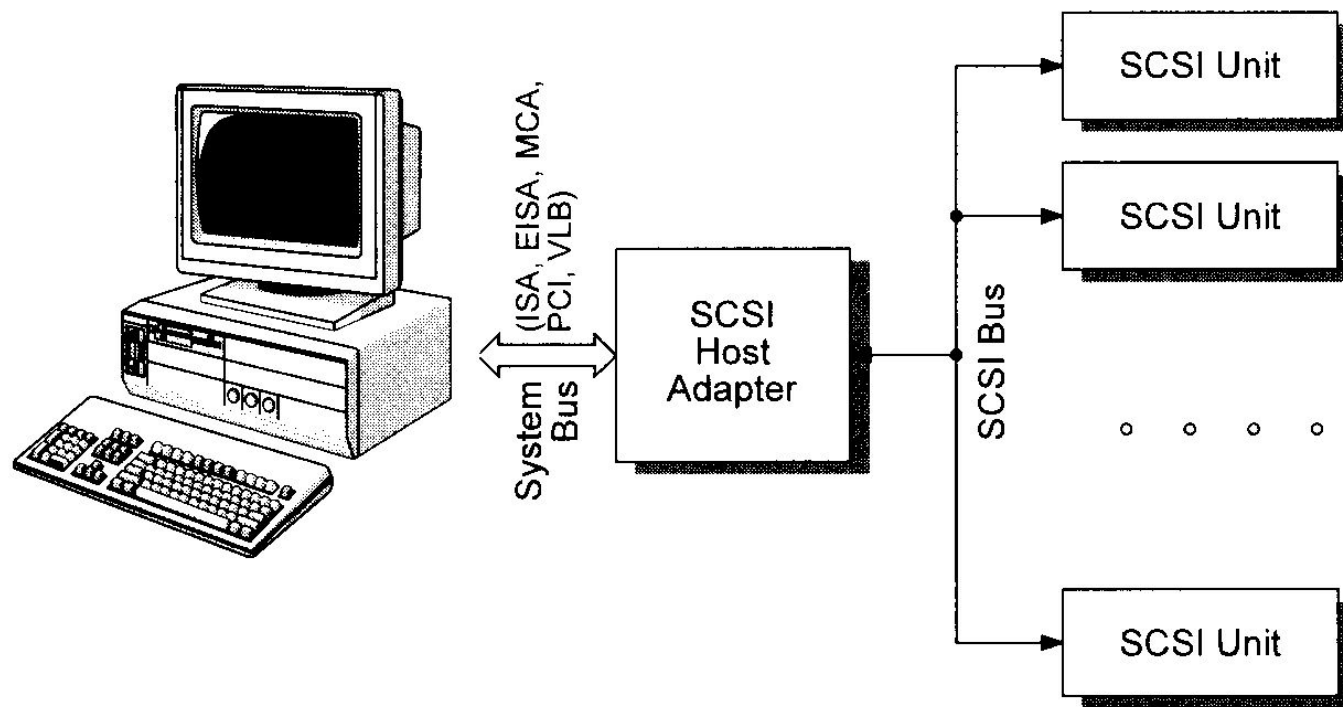
SCSI tulajdonságai (folyt. 1)

- A saját processzorral és memóriával rendelkező intelligens SCSI egységek kezdeményezőként (***initiator***) és fogadóként (***target***) is működhetnek! (lényegében egy I/O processzor eszköz)
- Maximálisan **8 initiator**, és **8 target** működhet egyszerre (de legalább 1 initiator/host-nak kell lennie a rendszerben). A kezdeményező adja ki az utasításokat, a cél pedig feldolgozza, és végrehajtja azokat. Minden egységnek saját különböző címe van (0-7), amelyeket jumperekkel (rövidzár) kell beállítani, hogy az ütközéseket a buszon elkerüljük. A hagyományos SCSI szabványnak megfelelően hostadapter/vezérlő címe mindig 7-es, míg a szalagos egységé pl. a 0-ás, azonosítójuk **SCSI-ID=0** ill. 7.

SCSI tulajdonságai (folyt. 2)

- A SCSI_ID mellett létezik a **LUN** = Logikai Egység Azonosítószám is: azaz minden targethez (perifériához) hozzárendelhető további 8 logikai egység, amiket az SCSI parancsok esetén saját LUN-nal azonosíthatunk.
- Az (korai, párhuzamos) **SCSI-I** busz kommunikáció 8-bites adatbuszon 1-bites paritás ellenőrzés mellett zajlott. Lassú target esetén, érdemesebb volt magasabb prioritású szintet állítani. A busz elején a vezérlő hostadapter van (jumper, kapcsoló, BIOS állítja), a másik végére pedig a lezáró ellenállást (mindig az utolsó eszköznél kellett tenni). Az egységeket egymás után felfűzve, egyforma (50-eres) szalagkábelrel lehetett csatlakoztatni.
- Alapvetően **Aszinkron** / de a mai szabványok a szinkron protokollokat is támogatják!

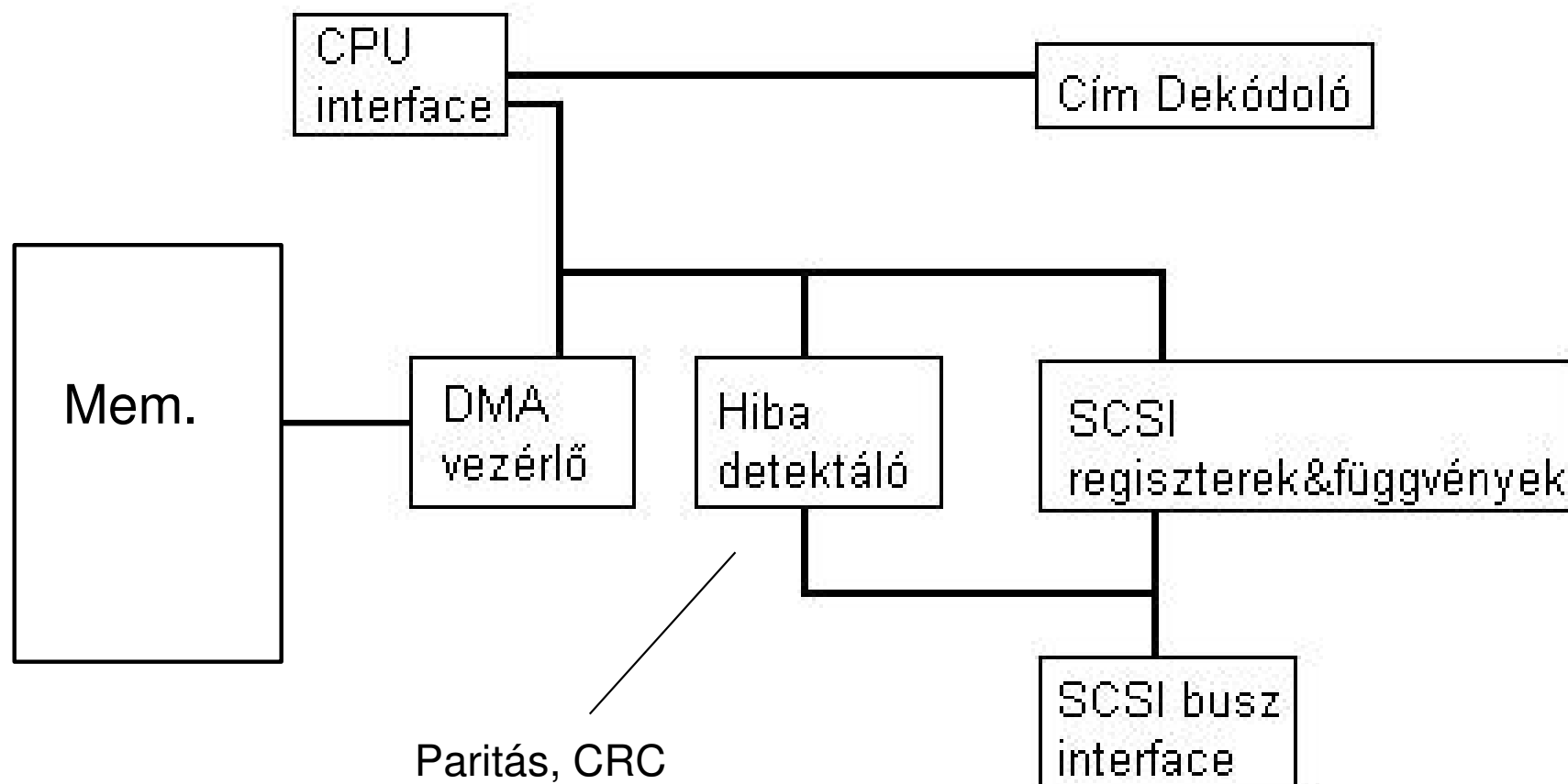
SCSI busz és PC rendszerbusz kapcsolata



- A *SCSI busz* egy host adapteren keresztül kapcsolódik a *rendszer buszhoz*

SCSI blokkdiagram

- SCSI vezérlő általános felépítése



SCSI busz fontosabb vezérlő jelei

- **REQUEST**: handshaking parancskérés, target által kezelt
- **ACKNOWLEDGE**: handshaking üzenet nyugtázása az initiator által
- **BUSY**: buszon a target foglaltságának jelzése (egy eszköz szabad, ha BUSY=0)
- **SELECTION**: initiator kiválasztja a target-et (SELECTION=0, nincs kiválasztva),
 - (SEL=0 esetén a target újra felépíti a kapcsolatot az initiator-ral a busz ideiglenes felszabadítása után)
- **C/D: Control /Data**: a target által kezelt jel, vezérlőadatok, parancsok, állapotinformációk jelzése a buszon
- **I/O : Input/Output**: szintén a target által kezelt vezérlőjel, amely az adatbusz adatforgalmának irányát mutatja
- **MSG= Message**: az üzenetküldés fázisának jelzésére szolgál, a target által kezelt
- **ATTENTION**: vezérlő figyelmezteti a célt
- **RESET**: az összes csatlakoztatott SCSI eszköz „reset-elése”, inicializálása, a sín alaphelyzetbe állítása

SCSI busz fázisok #1

■ Bus-free (szabad?)

- ☐ Nem használja SCSI egység a buszt, és SEL# , BSY# inaktívak

■ Arbitration (döntési mechanizmus)

- ☐ egység aktiválja a **BSY#** és saját SCSI-ID azonosítóját a buszra helyezi
- ☐ Rövid arbitrációs késleltetés után, ha nincs más aktív SCSI-ID magasabb prioritással, akkor az egység fogja vezérelni a buszt és aktiválja **SEL#** jelet

■ Selection (kiválasztás)

- ☐ Initiator kiválasztja a Target-et (bizonyos funkciókat kell végrehajtani)
- ☐ az I/O# jel inaktív
- ☐ Az initiator és target SCSI-ID-jének OR kapcsolatából egy címet állít elő, melyet a buszra helyez
- ☐ target aktiválja BSY# jelet (foglalt lesz a busz)

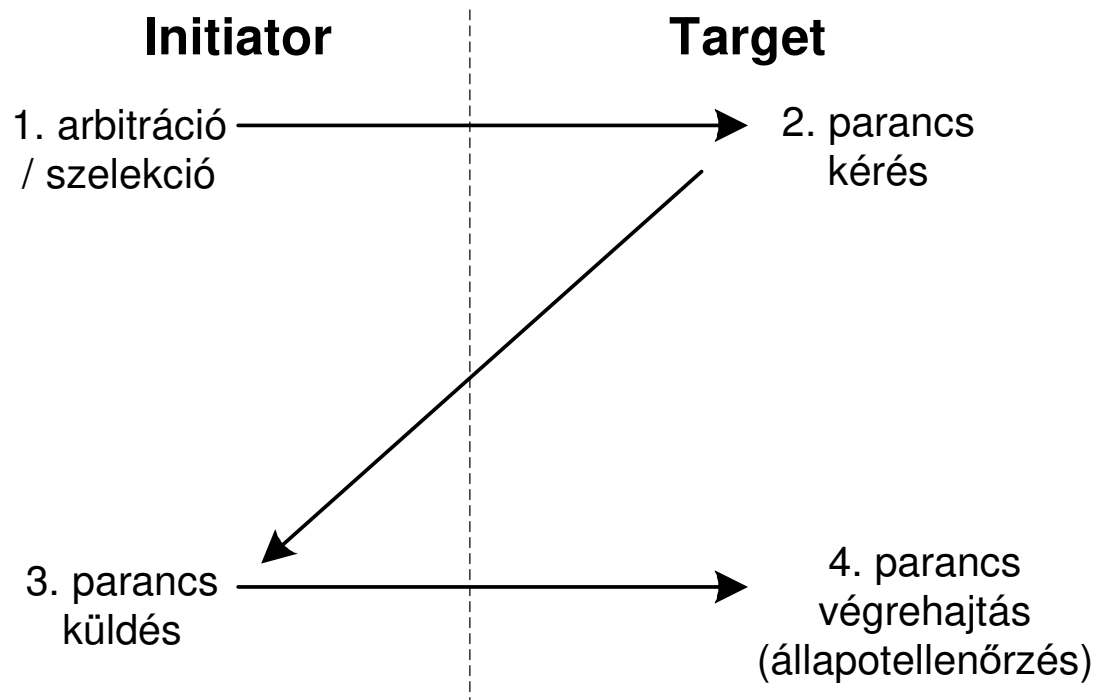
SCSI busz fázisok #2

- Reselection

- Ha szükséges, az initiator újra kapcsolatot létesít a korábbi targettel, miután egy megszakítás történt (~folytatódhat a művelet ott ahol abbamaradt)

- 1. Command: Initiator parancsot küld (Target parancsot igényelhet)
- 2. Data: adatátvitel pl. Init -> Target
- 3. Message: üzenet továbbítás
- 4. Status: Target->Init állapot információ

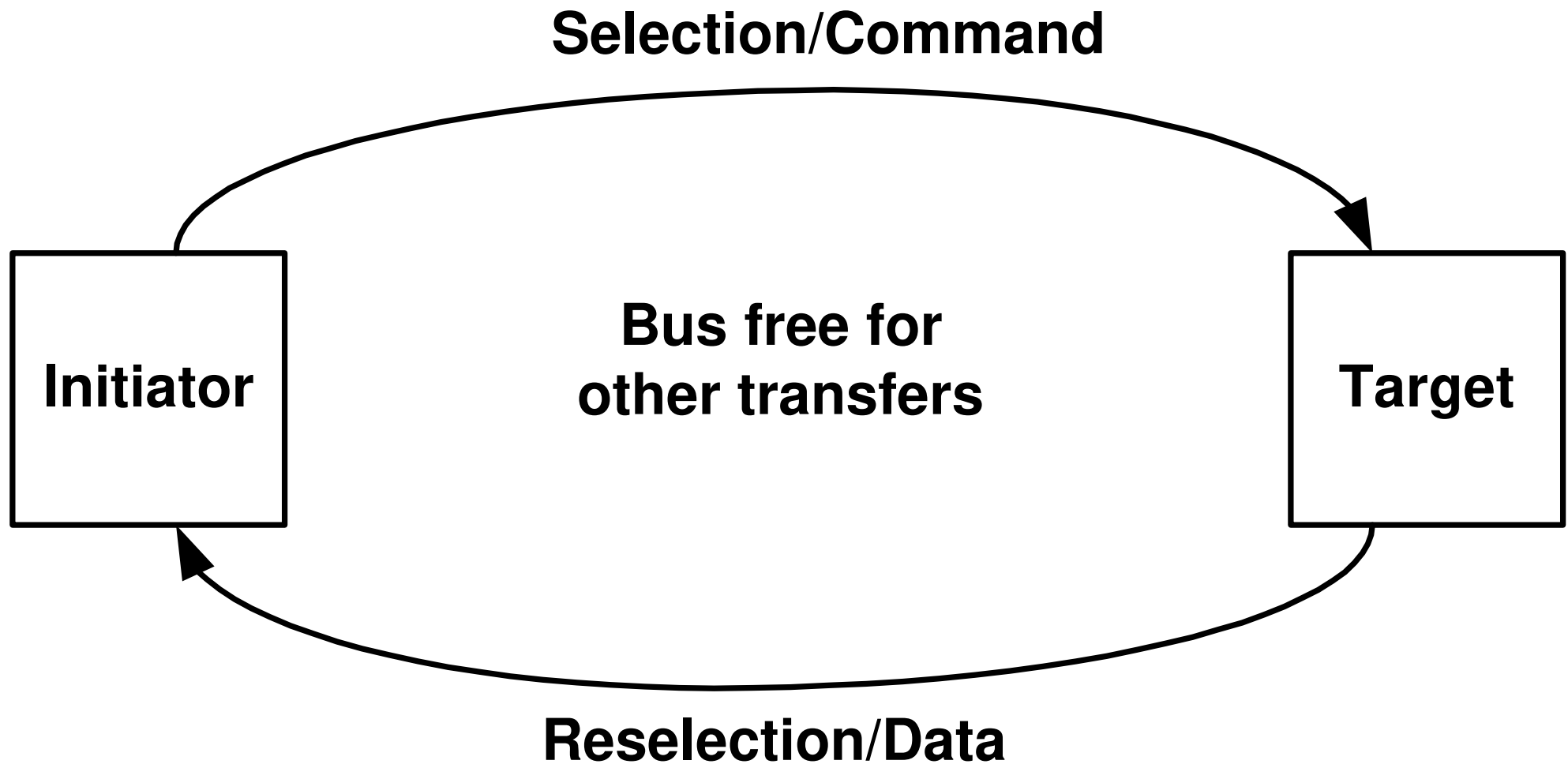
SCSI – kommunikációs lépések



SCSI busz fázisok a következők:

- ☐ - Reselection (ha a target meg lett szakítva, folytathatja a komm.t az initiatorral)
- ☐ - busz szabad (bus free?)
- ☐ - arbitráció / szelekció (ki adjon?)
- ☐ - üzenetküldés (msg)
- ☐ - parancs elmegy (command)
- ☐ - adatkapcsolat (data)
- ☐ - állapotellenőrzés (status)
- ☐ - üzenetzárás, kapcsolatbontás

SCSI bus műveletek (kördiagram)



Fontosabb SCSI szabványok



■ *Párhuzamos:*

- *SCSI I:* 8 bites busz; átviteli seb. *aszinkron* módban 2.5Mbyte/s, szinkron módban 5Mbyte/s; 5Mhz busz-frekvencia; max 7 egység csatlakoztatható, 50 pólusú csatoló
- *SCSI II (Fast SCSI):* első valódi SCSI szabvány, 8 bites busz; 10Mbyte/s; 10 Mhz buszfrequencia; max 7 egység; 50 pólusú
- *Wide-SCSI:* 16 bites szinkron busz; 20Mbyte/s; 10Mhz; 15 egység; 68 pólusú csatlakozó
- *Ultra Wide SCSI:* 16 bites; 40Mbyte/s; 20 Mhz; max 15 egység; 68 pólusú csat.
- *Ultra2 Wide SCSI:* 16 bites; 80Mbyte/s; 40Mhz; max 15 egység; 68/80 pólusú; max kábelhossz. 12m
- *Ultra-3160 SCSI:* 16 bites; 160Mbyte/s; 80Mhz; max 15 egység; 68/80 pólusú; 12m (differenciális jellel működik)
- *Ultra3-320 SCSI:* 16 bites; 320Mbyte/s; max 15 egység; 68/80 pólusú; 12m (differenciális jellel működik)
- *Ultra3-640 SCSI:* 16 bites; 640Mbyte/s, max 15 egység; 68/80 pólusú;
- **Soros (SAS) 1.x-3.x:** 1.5 – 3 – 6... 12 Gbit/s, (Adaptec.com), 6m, 16K eszköz
- **Optikai (Fiber),** max 16 Gbit/s; akár 127 egység, 500m
- **iSCSI:** SCSI over TCP/IP