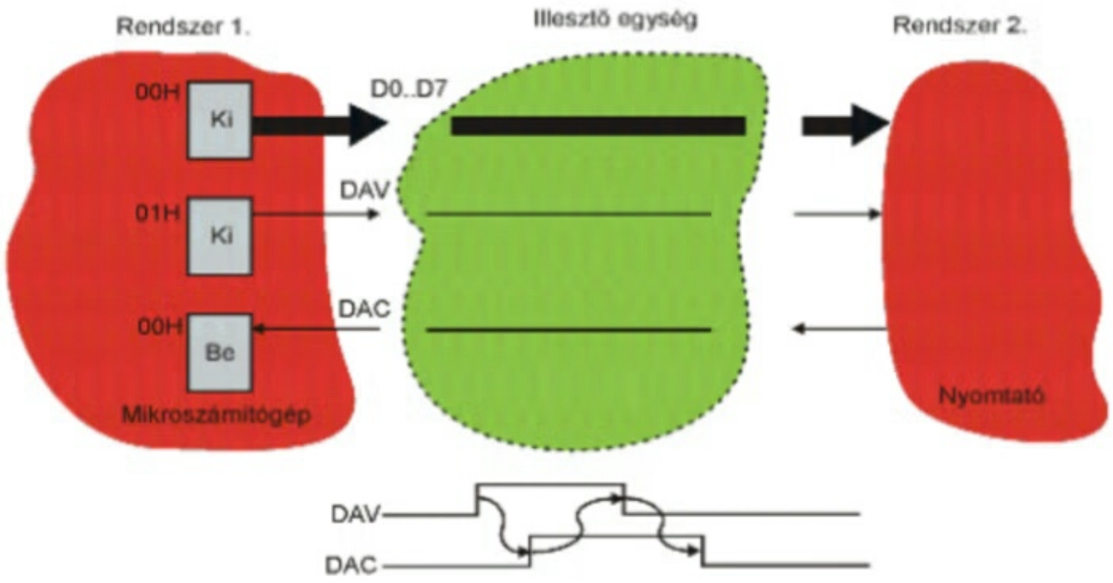
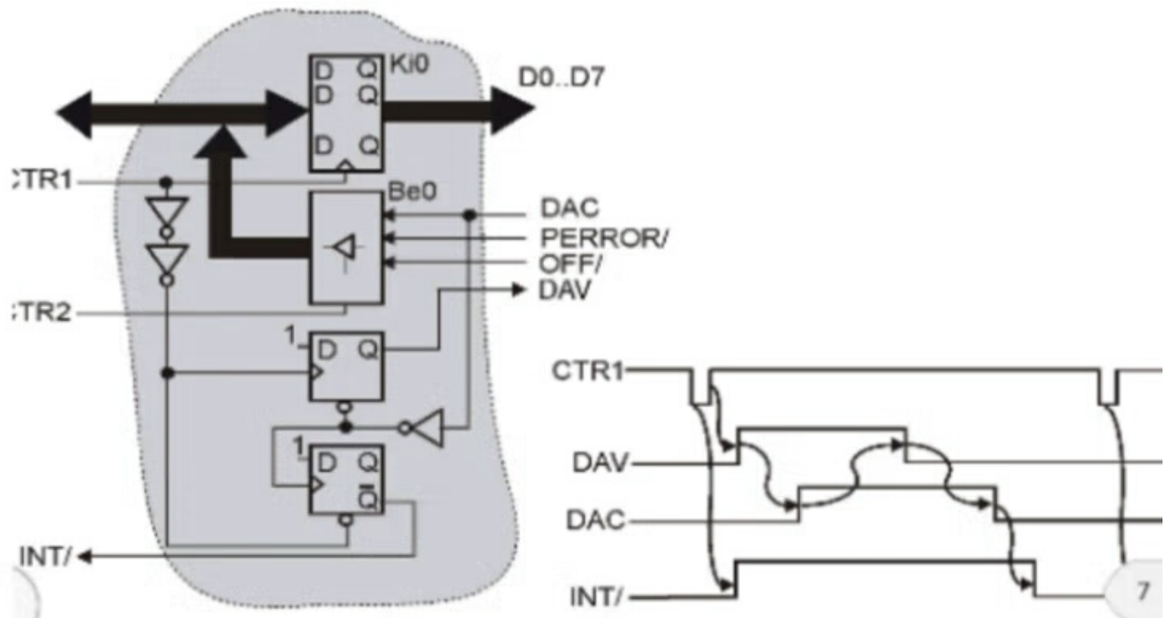
Bevezetés

Nyomtató illesztése IT nélkül és IT-vel. Az adatszeparátor áramkör felépítése és működése.

2. Nyomtató illesztése IT nélkül



3. Nyomtató illesztése IT-vel




```
KARAKTERKI: PUSH AF             ; a nyomtatandó karakter
                                ; mentése a stack-be
VARAKOZAS1: IN A,(00H)          ; DAC beolvasása
            BIT 0,A             ; DAC vizsgálata
            JR NZ,VARAKOZAS1    ; ugrás, ha DAC nem 0
            POP AF              ; a nyomtatandó karakter
                                ; visszahozása
            OUT (00H),A         ; a nyomtatandó karakter
                                ; kivitele
            LD A,1              ; a DAV 1-be állítása
            OUT (01H),A         ; a DAV jel kivitele
VARAKOZAS2: IN A,(00H)          ; a DAC jel beolvasása
            BIT 0,A             ; a DAC jel vizsgálata
            JR Z,VARAKOZAS2     ; várakozás, ha értéke 0
            LD A,0              ; a DAV jel 0-ba állítása
            OUT (01H),A         ; a DAV jel kivitele
            RET                 ; visszatérés
```

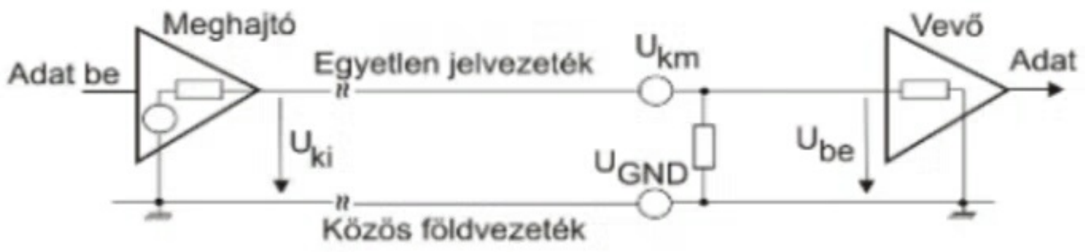
```
SADDR:      DS 2    ; blokk kezdőcím (start address)
NOB:        DS 2    ; átviendő bájtok száma (number of bytes)
PRINTBL:    LD HL,(SADDR)   ; blokk kezdőcím betöltése
            LD DE,(NOB)     ; blokkhossz betöltése
            LD A,D          ; 0 blokkhossz vizsgálata
            OR E            ;
            RET Z           ; visszatérés 0 blokkhossz esetén
WAIT:       IN A,(BE0)      ; a DAC jel beolvasása
            BIT 0,A         ; a DAC jel vizsgálata
            JR NZ,WAIT      ; várakozás, ha a DAC jel nem 0
            LD A,(HL)       ; az aktuális, kiviendő karakter
                            ; betöltése
            OUT (KI0),A     ; az aktuális karakter kivitele
            DEC DE          ; a blokkhossz csökkentése
            LD (NOB),DE     ; a blokkhossz visszaírása a mem-be
            LD A,D          ; 0 blokkhossz vizsgálat
            OR E            ;
            RET Z           ; visszatérés, ha a blokkhossz 0
            INC HL          ; a kezdőcím növelése
            LD (SADDR),HL   ; a kezdőcím visszaírása a memóriába
            EI              ; a megszakítás engedélyezése
            RET             ; visszatérés
```

Mechanikai, logikai és elektromos jellemzők I.
A logikai értékek reprezentálása. Aszimmetrikus, szimmetrikus és áramhurkos rendszerek. Statikus és dinamikus paraméterek, a DC zajtávolság. Kétállapotú, háromállapotú és nyitott kollektoros meghajtók. Az ECL technológia jellemzői. A BTL és GTL technológia jellemzői.

Logikai értékek reprezentálása

A hozzárendelés lehetséges és ténylegesen használt esetei:
- Feszültségtartományok hozzárendelése a logikai értékekhez
  - Aszimmetrikus rendszerek
  - Szimmetrikus, másszóval differenciális rendszerek
- Áramtartományok hozzárendelése a logikai értékekhez, ún. áramhurkos rendszerek

Aszimmetrikus jelátviteli rendszerek

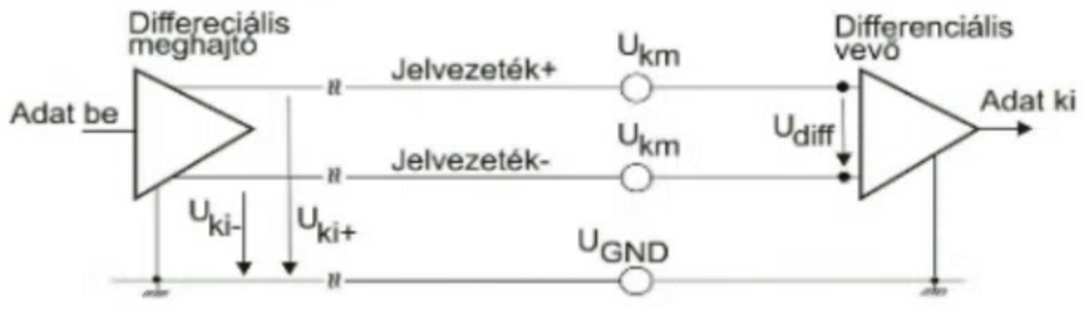


U_{be} eltérése U_{ki}-től
- U_{GND} földvezeték potenciáljának eltolódása
- U_{km} közös módusú zajfeszültség a jelvezetékben

Előnyök
- Egyszerű, kis költségű
- Bitenként csak egy vezeték szükséges
- De zajos környezetben
  - Árnyékolt kábel, és bitenkénti földvezeték szükséges
  - 1m felett sodrott érpár javasolt
  - Ezek azonban megszüntetik a kis költség miatti előnyöket

Hátrányok
- Alacsony zajimmunitás
  - Közös módusú zajok miatt
  - Áthallás nagyobb frekvenciákon
  - Jelentős EMI-t termel
  - Belső zajok a nem lezárt vezetékek miatt

Szimmetrikus jelátviteli rendszerek



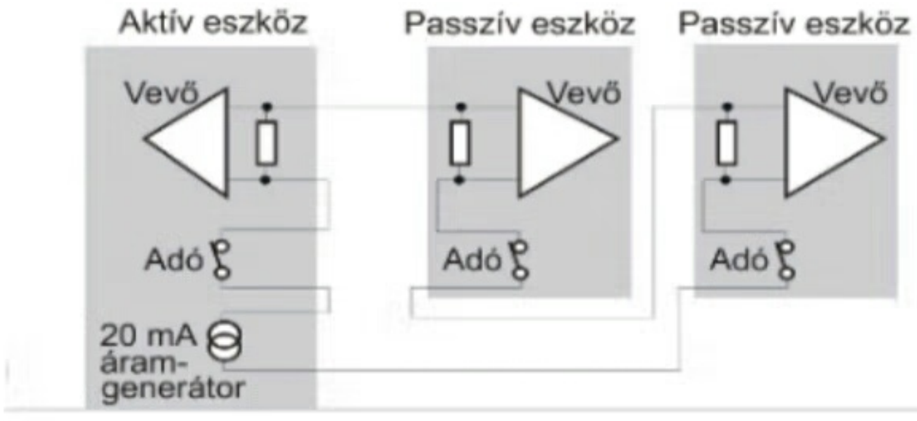
- Két jelvezeték egy bit átviteléhez
  - A logikai értéket a két bemenet közötti feszültség polaritása határozza meg
  - Például logikai 1 értékhez U+ > +2V és U- < +1V kimenetet, logikai 0 értékhez pedig U+ < +1V és U- > +2V kimenetet hozzárendelve, a differenciális vevő bemenetén a differenciá feszültség Udiff >+1V és Udiff <-1V.

Előnyök
- Kevésbé érzékeny a közös módusú zajokra, ezek elvileg kioltjál egymást a vevőnél
- Sodrott érpárral akár 10GHz-ig is használható
- Kevesebb EMI-t termel

Hátrányok
- Magasabb költségek a dupla vezetékszám miatt
- Kezdetben a meghajtók és vevők is drágábbak voltak, ma már nem
- Nagyfrekvencián definit lezárásokat igényel és stabil hullámimpedanciát igényel (ezért sodrott érpár)

Digitális áramhurok

- Áram hiánya, illetve megléte felel meg a két logikai értéknek
- 60-as évek elejétől távnyomtatók (teleprinter)
- Több km-re 1920bps-mal is működhetett
- Nagyfokú zajimmunitás
- RS485/422 kiszorította
- Példa half duplex rendszer (egyidejűleg csak egy adó működhet)



Analóg áramhurok

- Kétvezetékes rendszerben megvalósítja az érzékelő, távadó tápellátását és az adatátvitelt.
- Az érzékelő, távadó úgy szabályozza a hurok áramát, hogy a 4 mA feletti rész a mért mennyiséggel arányos legyen.
- 0 mért érték esetén is a 4 mA-es áram biztosítja számára a tápellátást.

Bipoláris áramhurok

- Ebben a rendszerben a két logikai értékhez ellentétes irányú áramot rendelnek hozzá.
- Lényegében így működik az USB nagy sebességen (480MB/s Vagy a D+ vagy a D- vonalat hajtja meg 17,78mA-es árammal hogy a 45ohmmal a föld felé lezárt vonalai között +400mV vag -400mV feszültség lépjen fel

Meghajtók és vevők paraméterei

– Statikus paraméterek

- C_i, C_o Az áramkör belső kapacitása a bemenetén, illetve a kimenetén.
- I_{CC} Az áramkör V_{CC} bemenetén befolyó áram.
- I_{CEX} Maximális kimeneti szivárgási áram, ha a kimenet magas szintű és a tápfeszültség szintjére van felhúzva.
- $I_{I(hold)}$ Az a bemeneti áram, amely az előző állapotban tart[illegible] áramkört, ha a meghajtó eszköz nagyimpedanciás állapo[illegible] kapcsol át.
- I_{IH} Bemeneti áram magas szinten (a terminálon kifelé folyó áram előjele negatív).
- I_{IL} Bemeneti áram alacsony szinten.
- I_{OFF} Be-kimeneti szivárgási áram kikapcsolt tápfeszültségnél.
- I_{OH} Kimeneti áram magas szinten.
- I_{OL} Kimeneti áram alacsony szinten.

– Statikus paraméterek (folytatás)

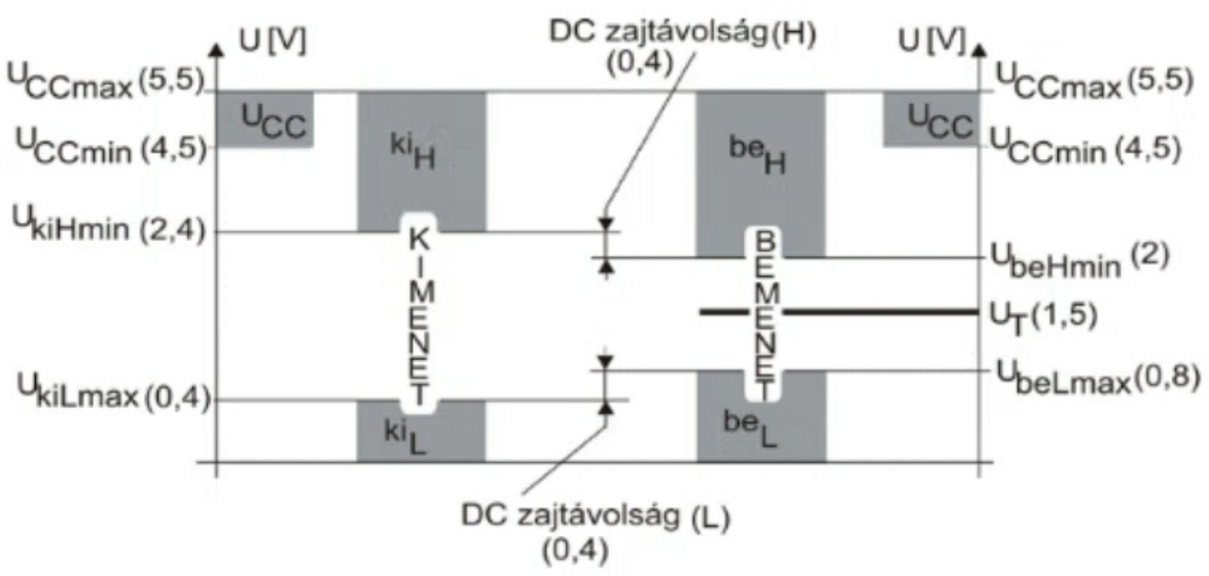
- V_{IH} Magas szintű bemeneti feszültség. Az a minimális feszültsége, amely még a magas szinthez tartozó logikai értéknek megfelelő működést váltja ki.
- V_{IL} Alacsony szintű bemeneti feszültség. Az a maximális feszültség, amely még az alacsony szinthez tartozó logikai értéknek megfelelő működést váltja ki.
- V_{OH} Magas szintű kimeneti feszültség. Az a minimális feszültség, amely még a magas szinthez rendelt logikai értéket reprezentálja.
- V_{OL} Alacsony szintű kimeneti feszültség. Az a maximális feszültség, amely még az alacsony szinthez rendelt logikai értéket reprezentálja.
- V_{T+} Pozitív küszöbszint. Az a bemeneti feszültség, amelyet a negatív küszöbszint felől átlépve az eszköz átkapcsol másik állapotába.
- V_{T-} Negatív küszöbszint. Az a bemeneti feszültség, amelyet a pozitív küszöbszint felől átlépve az eszköz átkapcsol másik állapotába.

– DC zajtávolság (H) magas szinten ($U_{kiHmin} - U_{beHmin}$). Ez a zajtűrésnek is nevezett jellemző a külső zavarjel maximális értékét specifikálja, amely magas szinten még algebrailag hozzáadható a legrosszabb zavarmentes bemeneti jelhez, anélkül hogy a vevő kimenetének helyes logikai értéke megváltozna.

– DC zajtávolság (L) alacsony szinten az ($U_{beLmax} - U_{kiLmax}$). Ez a zajtűrésnek is nevezett jellemző a külső zavarjel maximális értékét specifikálja, amely alacsony szinten még algebrailag hozzáadható a legrosszabb zavarmentes bemeneti jelhez, anélkül hogy a vevő kimenetének helyes logikai értéke megváltozna.

– DC zajtávolság az előbbi két zajtávolság közül a kisebbik.

A paraméterek szemléltetése



– Dinamikus paraméterek

- f_{max} Maximális frekvencia, amellyel a billenőkörök órabemenete még meghajtható a hibamentes működés mellett.
- SR_{LH} Változási meredekség alacsony-magas átmenetkor (lásd az. ábrát).
- SR_{HL} Változási meredekség magas-alacsony átmenetkor (lásd az ábrát).
- t_{dis} Passziválási (disable) idő 3-állapotú és nyitott kollektoros kimenetek esetén, amely eltelik a passziválást kiváltó bemeneti jel fellépte és a kimenet nagyimpedanciás állapotba kerülése között.
- t_{en} Aktiválási (enable) idő 3-állapotú és nyitott kollektoros kimenetek esetén, amely eltelik az aktiválást kiváltó bemeneti jel fellépte és a kimenet nagyimpedanciás állapotból aktív állapotba kerülése között.
- t_f Esési idő, amely alatt a kimenőjel magas szintű definiált pontjáról alacsony szintű definiált pontjára változik (lásd az ábrát).

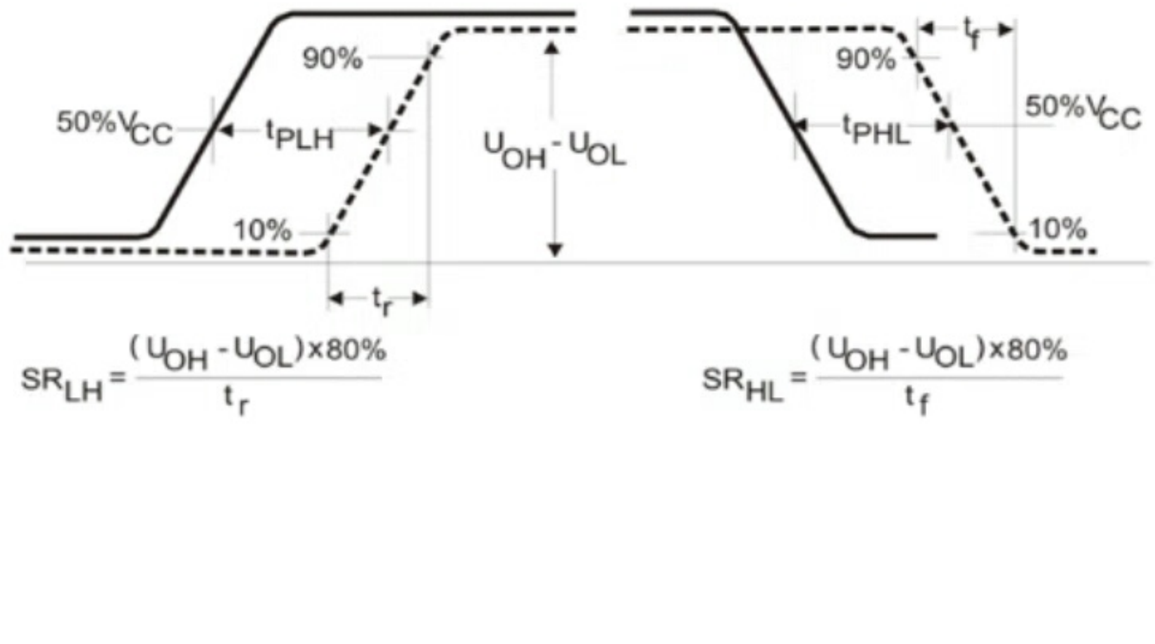
– Dinamikus paraméterek (folytatás)

- t_h Tartási idő, ezalatt a statikus bemenet nem változhat az órajel változását követően.
- t_{PHL} Terjedési idő, a kimenet magas-alacsony átmenetekor. A be-és kimenet specifikált referenciapontjai közötti idő a kimenet definiált magas szintjéről definiált alacsony szintre váltásakor (lásd az ábrát).
- t_{PHZ} Passziválási idő, a kimenet magas-nagyimpedanciás átmenetekor.
- t_{PLH} Terjedési idő, a kimenet alacsony-magas átmenetekor. A be-és kimenet specifikált referenciapontjai közötti idő a kimenet definiált alacsony szintjéről definiált magas szintre váltásakor (lásd az ábrát).

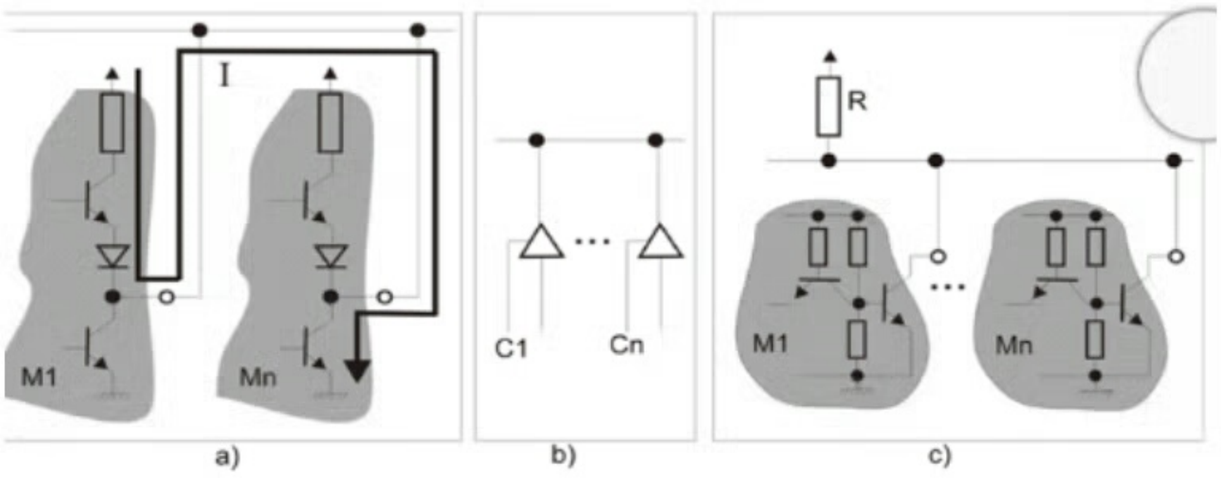
– Dinamikus paraméterek (folytatás)

- t_{PLZ} Passziválási idő, a kimenet alacsony-nagyimpedanciás átmenetekor.
- t_{PZH} Aktiválási idő, a kimenet nagyimpedanciás-magas átmenetekor.
- t_{PZL} Aktiválási idő, a kimenet nagyimpedanciás-alacsony átmenetekor.
- t_r Emelkedési idő, amely alatt a kimenőjel alacsony szintű definiált pontjáról magas szintű definiált pontjára változik (lásd az ábrát).
- t_s Előkészítési idő, amely alatt a statikus bemenet értéke nem változhat az órajel változását megelőzően.

Dinamikus paraméterek értelmezése



Kétállapotú- (a), háromállapotú- (b) és nyitott kollektoros kimenetű (c) meghajtók



A nyitott kollektoros meghajtók felhúzó ellenállásának méretezése

- Ha túl nagy, akkor a maradékáram okozta feszültségesés miatt túl alacsony lesz a magas szint
- Ha túl kicsi, akkor tönkremehet a meghajtó tranzisztor
- A reflexiók miatt a kis érték a kritikus, erre pedig

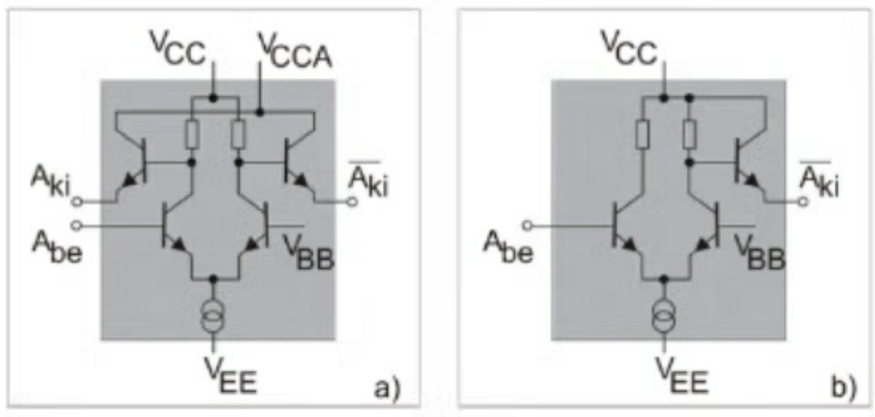
$$(I_{kiL\max} + mI_{beL\min}) > \frac{U_{CC} - U_{kiL\min}}{R_{\min}}$$

$$R_{\min} > \frac{U_{CC} - U_{kiL\min}}{I_{kiL\max} + m \times I_{beL\min}}$$

ECL technológia

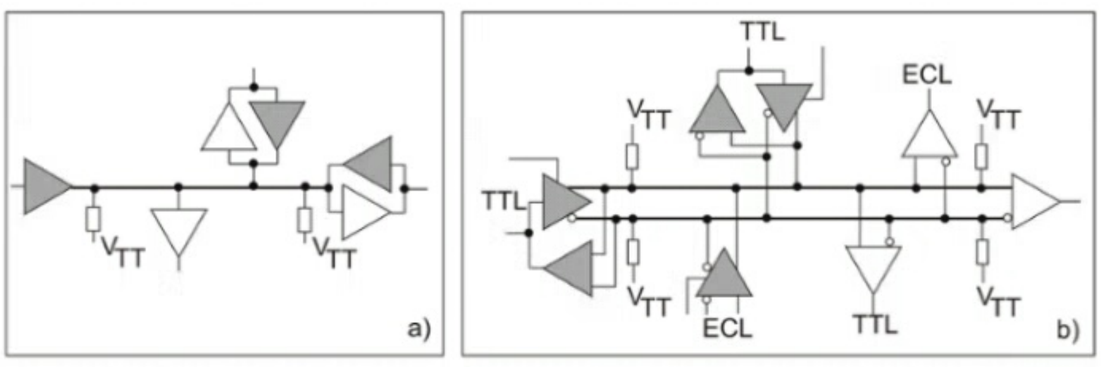
- A sebességet tekintve jelentős előnnyel rendelkezik a többi technológiához képest. Bizonyos ECL elemcsaládok terjedési ideje nem haladja meg a 300ps értéket, s maximális frekvenciájuk 1GHz felett van.
- Sebességük, továbbá nagyfokú zajimmunitásuk és alacsony zajgenerálásuk következtében igen előnyösek a nagyon nagy sebességű rendszerekben.
- További előnyös tulajdonságuk, hogy mind aszimmetrikus mind szimmetrikus meghajtókét/vevőként is használhatók.

Szimmetrikus és aszimmetrikus ECL meghajtó



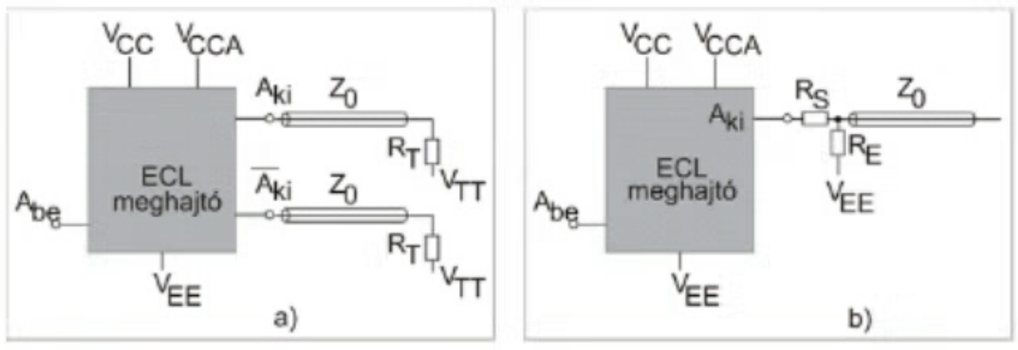
- Differencia erősítőt tartalmaz
- V_{BB} értéke -1,32V-on állandó, s a bemenet névleges L és H szintű feszültsége pedig -1,730V és -0,970V.
- Az eszközön átfolyó áram még átkapcsoláskor is állandó, ezért a táp- és földvezeték zaj minimális.

ECL buszkonfigurációk



- Mindkét esetben az 50 ohm hullámimpedanciájú vonalat mind a két végén 50 ohmos ellenállással le kell zárni U_{TT}=-2V-ra. A meghajtóknak ezért képesek kell lenniük 25 ohm meghajtására.
- A meghajtók bemenete, illetve a vevők kimenete ECL és TTL típusú is lehet, mindkét típusú áramkör hozzáférhető.

ECL lezárás

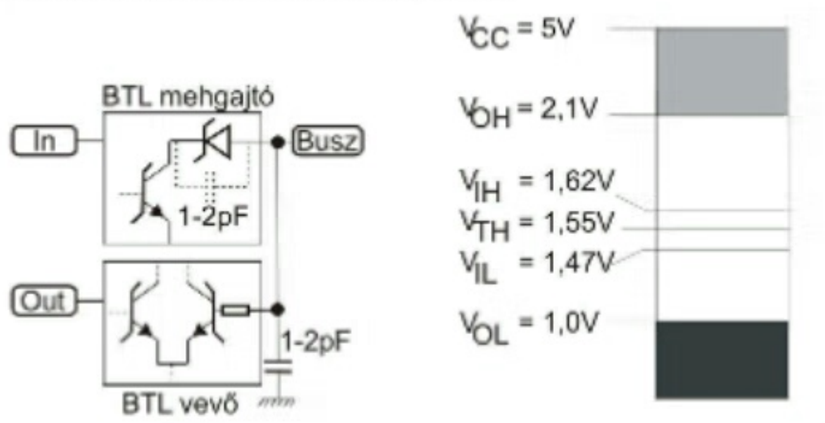


Külső lezárást igényelnek

- (a) Beeső hullámkapcsolású (lezáró ellenállás a hullámimpedanciával azonos)
- (b) Visszavert hullámkapcsolású (a soros ellenállás és az áramkör kimeneti ellenállásának összege egyezik a hullámimpedanciával). Az R_E ellenállás alacsony szintre való átkapcsoláskor a vonal kisütését szolgálja.

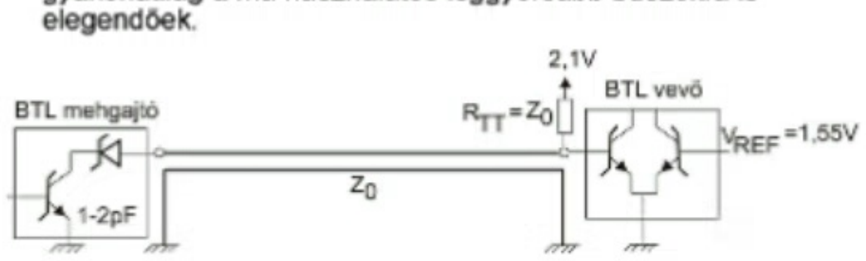
BTL

- A meghajtó kimeneti, ill. a vevő bemeneti kapacitását 1-2pF-re korlátozza
- A kimeneti kis kapacitást a soros Schottky diódával érik el
- 1.1V-os átkapcsolási feszültség, 100mA-es meghajtóképességgel lehetővé teszi a beeső hullámkapcsolást



BTL buszkonfiguráció

- A H szintet a vonal végén lévő, 2,1 V feszültségre felhúzó ellenállás állítja be a meghajtó kikapcsolt állapotában.
- Kétirányú vonalak esetén a vonal minkét végén le kell zárni a felhúzó ellenállással.
- A jelváltás nagysága 1,1V, s mivel a meghajtó 100mA-et tud leadni, még 11 ohmos lezáró ellenállást is meg tud hajtani.
- Mindkét oldali lezárás esetén, azaz ha a buszra nemcsak az egyik végponton, hanem közbülső pontján is csatlakozik meghajtó, akkor a lezáró ellenállás alsó határa 22 ohm. Ezek az értékek gyakorlatilag a ma használatos leggyorsabb buszokra is elegendőek.

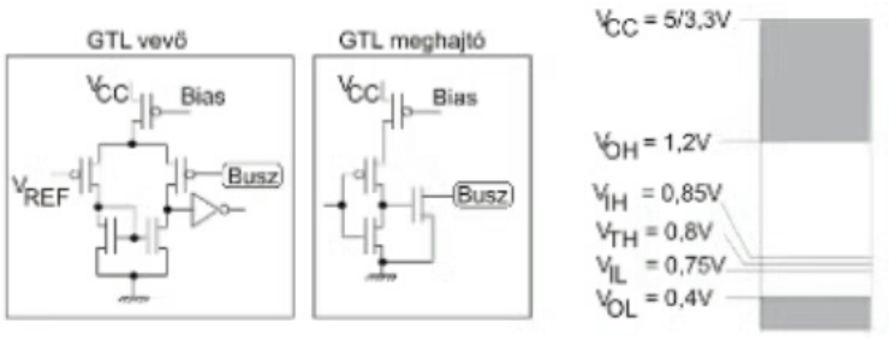


ECL további jellemzői

- TTL technológia → nagy fogyasztás
- BiCMOS technológia: kimenet bipoláris, többi rész CMOS
- Előnyei
  - Telítetlen logika → gyorsabb mint a TTL
  - Aszimmetrikus és szimmetrikus egyaránt megvalósítható
  - Nagy zajimmunitás
  - Alacsony zajgenerálás
  - Akár 25 ohmos hullámimpedanciára is használhatók
- Hátrányok
  - Nagyobb fogyasztás
  - Speciális áramköri elrendezést (layout) és lezárást igényelnek
  - Implementálásuk jóval költségesebb

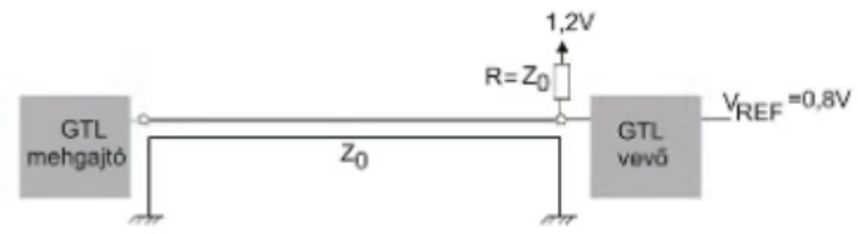
GTL

– Csökkentett kapcsolási feszültségű (kisebb mint 1V), nyitott drain típusú, differenciális bemenetű, CMOS technológiájú elem, melyet a JEDEC is szabványosított.
– A meghajtó és vevő áramköri rajza:



– Az alacsony szint feszültsége 0,4V, s az 1,2V-os magas szint mellett a kapcsolási feszültség 0,8V-re redukálódik.
– A küszöbszint itt is a kapcsolási feszültség közepén van, azaz értéke 0,8V.
– A meghajtó 40mA-es kimeneti áramával 0,8V/0,04A=20 ohmos lezárás hajtható meg. Mindkét végén lezárt busz esetén egy a busz belső pontjára csatlakozó meghajtó ténylegesen 40ohm hullámimpedanciájú, reflexiómentesen lezárt (azaz mindkét végén 40-40 ohmmal lezárt) buszt tud meghajtani.
– A maximális disszipáció 0,8V és 40mA miatt 32mW. Ezért ezek az eszközök már ASIC elemekbe is integrálhatók.

GTL buszkonfiguráció



– A buszkonfigurációban a BTL és GTL között csak a feszültségszintekben van különbség.

Mechanikai, logikai és elektromos jellemzők II.

A távvezeték jellemzése: hullámimpedancia, terjedési sebesség, reflexiós tényezők. A reflexiós folyamatok lejátszódása ideális távvezeték esetén. Be- és kikapcsolási folyamatok szerkesztése a Bergeron módszerrel. Az osztatlan és osztott lezárás. A hullámimpedancia meghatározása méréssel.

A távvezeték jellemzése

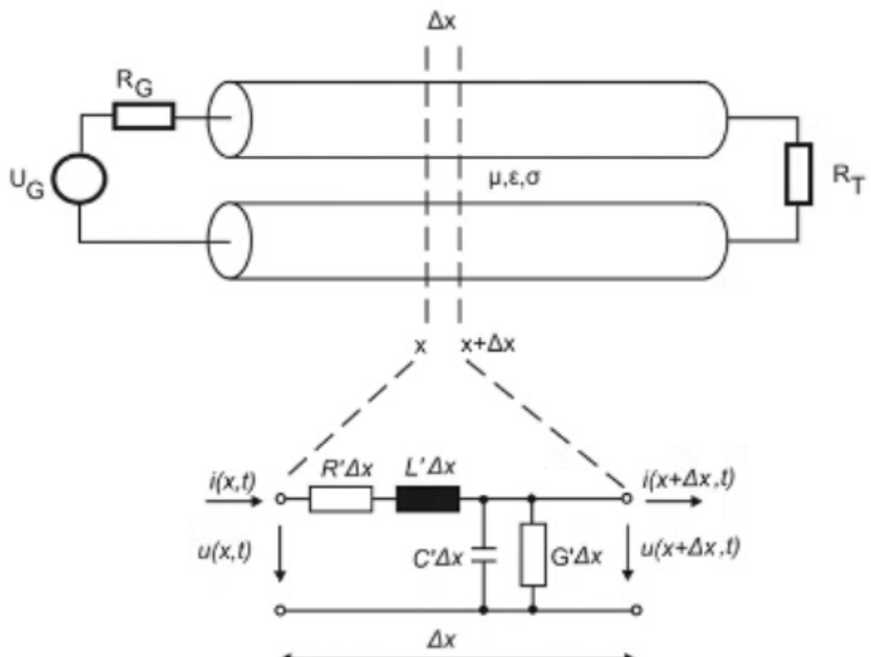
– A hullámimpedancia

$$Z_0 = \sqrt{\frac{j\omega L' + R'}{j\omega C' + G'}} \approx \sqrt{\frac{L'}{C'}} \text{ [ohm]}$$

– A terjedési sebesség

$$v = 1/(\sqrt{L'C'}) \text{ [m/s]}$$

A távvezeték jellemzése (folytatás)



A távvezeték jellemzése (folytatás)

– Reflexiós tényezők: a visszavert hullám amplitúdóját adják meg a beeső hányadában

$$r_G = \frac{R_G - Z_0}{R_G + Z_0} \qquad r_T = \frac{R_T - Z_0}{R_T + Z_0}$$

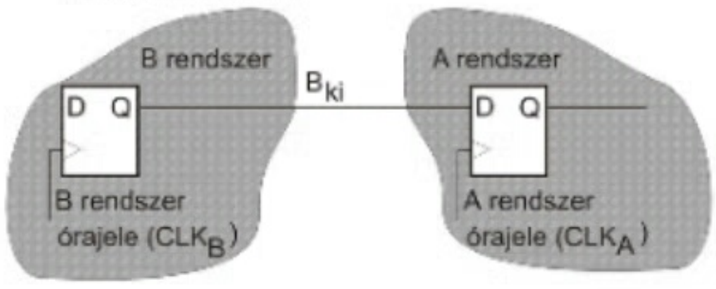
– Értékük -1 és +1 között van
– Negatív esetben ellenkező előjellel verődik vissza a beeső hullám

Mechanikai, logikai és elektromos jellemzők III.

A metastabilitás jelensége. Az illegális állapot felléptének valószínűsége, az MTBF és a t_{meta} kapcsolata. Szinkronizáló áramkörök.

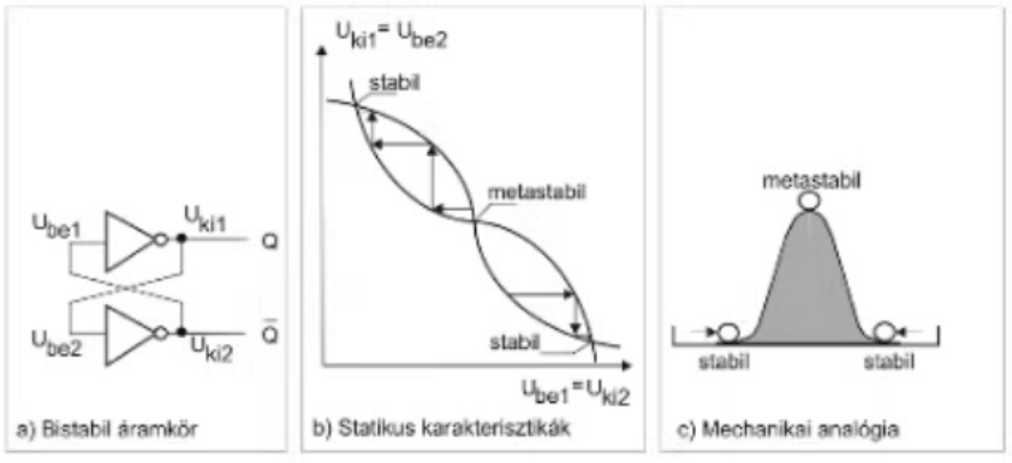
- A helytelen működési feltételeket pedig a tárolóelemek előkészítési idejére (setup time) és tartási idejére (holdup time) előírtak megsértése jelenti.
- Az előkészítési- és tartási időtartományban megváltozó statikus bemeneti jel a tárolóelemet ún. metastabil állapotba vezérelheti, amely se nem a logikai 1, se nem a logikai 0 állapot.

- A tárolóelem meghatározhatatlan ideig ebben az állapotban maradhat, mielőtt áttérne az egyik stabilis állapotába (1 vagy 0).
- Ilyenkor a kimenetére csatlakozó kapuáramkörök egy része logikai 0-nak, más része pedig logikai 1-nek érzékelheti ezt az illegális állapotot, s megint más része tovább terjesztheti ezt az illegális állapotot. Ami komoly nehézségekhez, s hibás működéshez vezethet a digitális rendszerekben.
- Bár egy rendszeren belül a megfelelő tervezéssel elkerülhető a metastatabil állapot, rendszerek összekapcsolásakor (illesztés) elkerülhetetlen.



B_{ki} és CLK_A bármilyen fázishelyzete felléphet, azaz nem kizá a közel egyidejű változa tehát sérül az előkészítési és tartási időre előírt feltétel

A metastabilitás jelensége

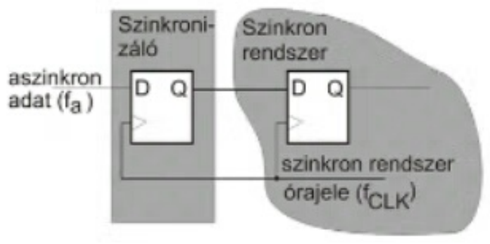


a) Bistabil áramkör. Logikai vizsgálata két stabil állapotot mutat, Q vagy H vagy L szinten lehet

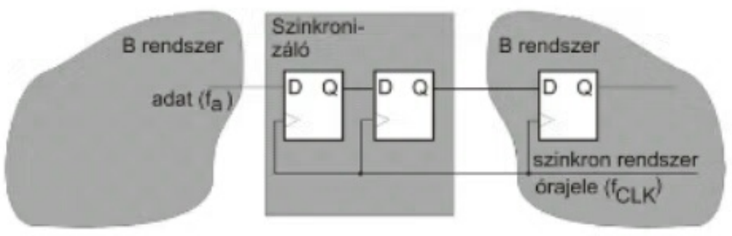
- Három egyensúlyi helyzet, szemben a logikai vizsgálattal kapott kettővel.
- Kettő ezek közül stabil: kimozdítva a rendszert ezekből az állapotaikból, oda visszatérnek
- Egy állapot metastabil, kimozdítva ebből valamelyik stabil állapotába tér át!

Szinkronizáló áramkörök

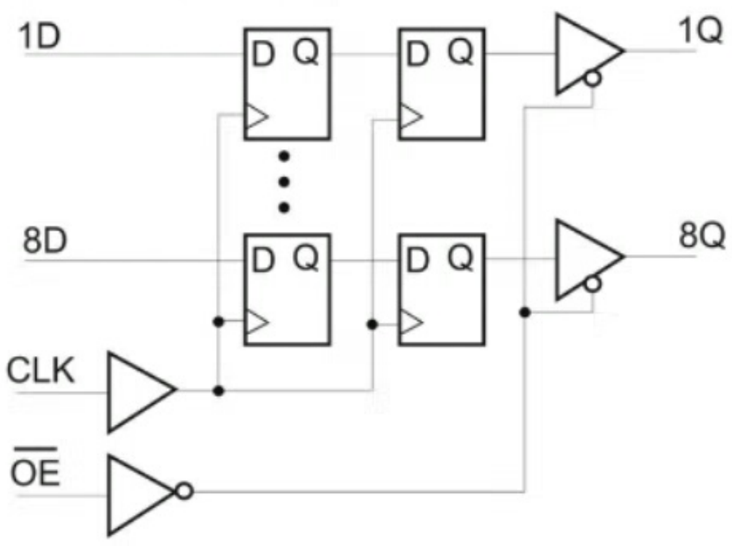
- Egyfokozatú



- Kétfokozatú



- A Texas Instruments szinkronizáló áramköre



Az illegális állapot felléptének valószínűsége

- Mintavételezzünk egy aszinkron jelet egy D tárolóval. Ekkor az illegális állapot felléptének valószínűsége

$$p_{hiba} = \frac{t_s + t_h}{t_{CLK}} = f_{CLK}(t_s + t_h)$$

- ahol t_s, t_h és f_{CLK} rendre az előkészítési és tartási idő, illetve a mintavételi frekvencia.
- Ha az aszinkron jel frekvenciája f_a, akkor a másodpercenként fellépő hibavalószínűség:

$$f_{hiba} = f_{CLK} f_a (t_s + t_h)$$

- Ha $t_s = t_h = 100ps$ és az órajel frekvenciája $f_{CLK} = 500MHz$, akkor

$$p_{hiba} = 0,1$$

- S ha az aszinkron jel frekvenciája 1MHz, akkor

$$f_{hiba} = 100kHz$$

- Ez meglepően nagy érték. Lehet-e egyáltalán működőképes rendszert tervezni?
- Igen. Ha az illegális állapotba be is lép a rendszer, onnan olyan gyorsan ki is kerülhet, ami még nem okoz hibát.
- Tudni kell tehát, mennyi ideig tartózkodik a rendszer az illegális állapotban.

- A gyakorlatban azt adják meg, hogy mennyi az átlagosan eltelt idő két, a t_{meta} időtartamnál hosszabb ideig fennálló metastabil állapot fellépte között. Ha a t_{meta}-nál hosszabb ideig fennálló metastabil állapot hibát okoz a rendszerben, akkor a szóban forgó átlagidő az ún. MTBF (Mean Time Between Failures = hibátlan működés átlagos ideje). Az MTBF a fenti p valószínűség reciproka, tehát

$$MTBF = \frac{k}{t_r} \frac{1}{f_a f_{CLK}} e^{t_{mata}/T} = \frac{1}{C_1 f_a f_{CLK}} e^{C_2 t_{meta}}$$

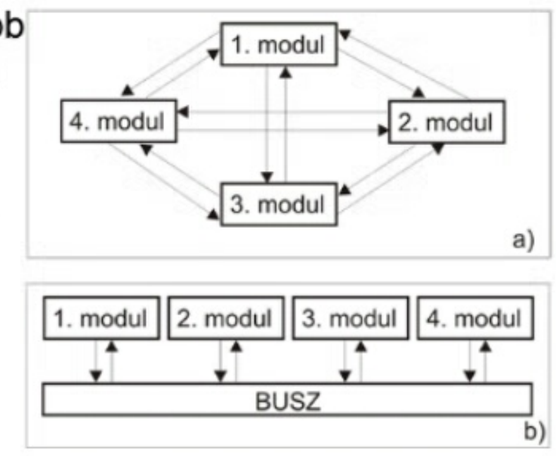
t meta: menyi idő alatt éri el a rendszer az egyik stabilis állapotát

Mechanikai, logikai és elektromos jellemzők IV.

A buszok fogalma és osztályozása szintek, használati jogok és a funkciók szétosztottsága szerint. Statikus és dinamikus arbitráció, allokációs és deallokációs elvek. A dinamikus arbitráció HW megvalósítása: a központi és elosztott, a soros, lekérdezéses és független kérelmeken alapuló megoldás. A logikai és geografikus címzés. Időzítési protokollok: szinkron, aszinkron, szemiszinkron. A szinkron olvasás implementációja és idődiagramja. A szinkron írás és a teljesen reteszelt aszinkron olvasás implementációja. Aszinkron olvasás és írás idődiagram. Jelelcsúszás.

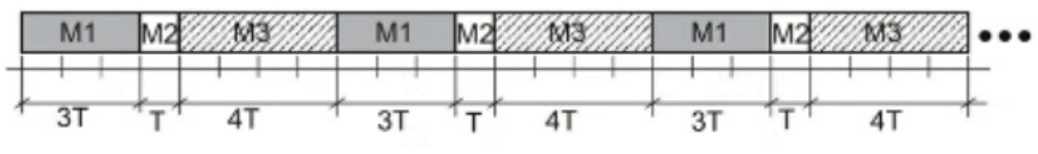
– A busz a vezetékek és csatlakozók egy olyan csoportja, amelynek az a feladata, hogy egy n-bites szó valamennyi bitjét egy adott, a buszra csatlakozó komponensből (adó vagy forrás) egy másik, ugyancsak a buszra csatlakozó komponensbe (vevő vagy nyelő) átvigye.

– Léteznek egyirányú buszok, melyek az adatokat csak az egyik irányban képesek átvinni, szemben a mindkét irányú adatátvitelre alkalmas ún. kétirányú buszokkal.
– A dedikált buszokra az a jellemző, hogy egyetlen adó és egyetlen vevő kapcsolódik rájuk. Ha n komponenst kellene összekötni buszokkal az összes lehetséges módon, akkor egyirányú dedikált buszokat használva n(n-1) buszra lenne szükség (a).
– A közös használatú busz (shared bus) fizikailag kettőnél több komponenst köt össze, jelentősen csökkentve a költségeket.
– De míg a dedikált buszok esetén egyidejűleg n(n-1) kommunikáció folyhat, közös használatú buszokon csak 1.



Buszkérelem és arbitráció

Statikus arbitráció



M1 vezérelheti a buszt a (T8k+0, T8k+1, T8k+2),
M2 vezérelheti a buszt a (T8k+3),
M1 vezérelheti a buszt a (T8k+4 , T8k+5 , T8k+6, T8k+7)

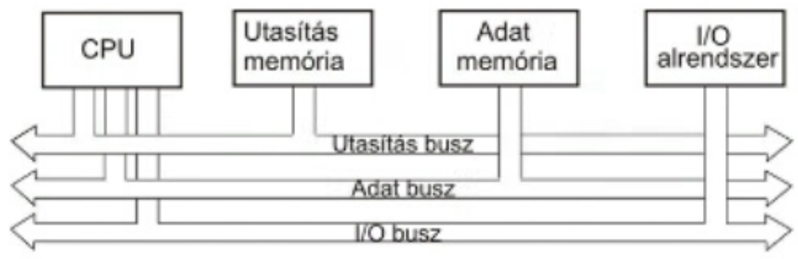
– Az aktuális buszkérelem fellépte és a busz vezérlési jogának odaítélése (busz allokáció) között nincs kapcsolat.
– Előre meghatározott, hogy az egyes potenciális mesterek milyen sorrendben és mennyi időre kapják meg a busz vezérlési jogát.
– Egyszerű, de nem hatékony

Dinamikus arbitráció

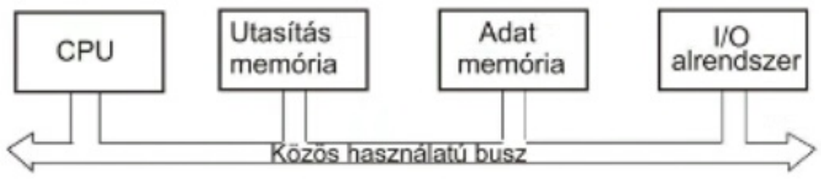
– A buszvezérlési jog megadását (allokáció) és elvételét (deallokáció) dinamikusan, a fellépő igények alapján határozza meg az arbiter.
– Allokációs elvek
  • Prioritásos
  • Igazságos
  • Kombinált
– Deallokációs elvek
  • Elengedés kérelem esetén (Release on Request)
  • Elengedés a tranzakció befejeztével (Release when done)
  • Kényszerített elengedés (Pre-emption)

Buszrendszerek a használati jogok szerint

– Dedikált

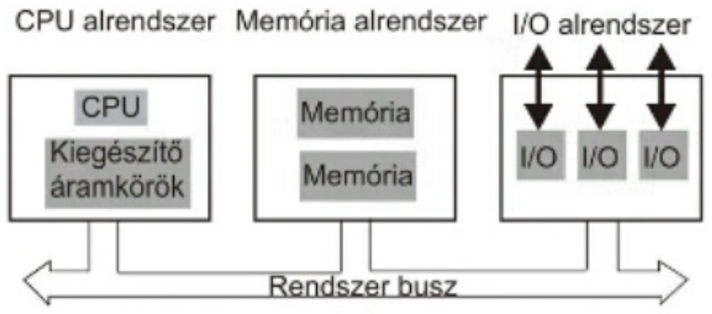


– Közös használatú



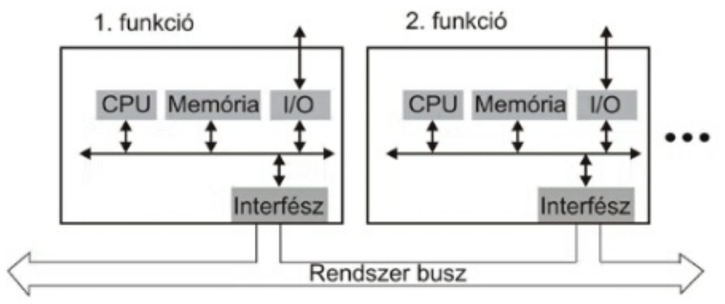
Buszrendszerek a funkciók szétosztottsága szerint

– Erőforrás szerint szétosztott



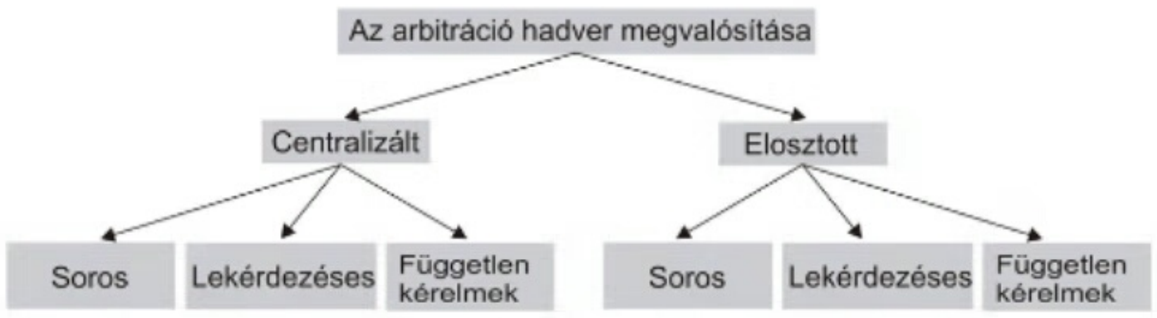
– CPU-memória-I/O busz, amely minden erőforrást összeköt
– Alacsony integráltsági fokon terjedt el
– Egyprocesszoros orientáltság
– Memóriabusz orientáltság
– A CPU és a buszjelek összehangoltak

– Funkciók szerint szétosztott



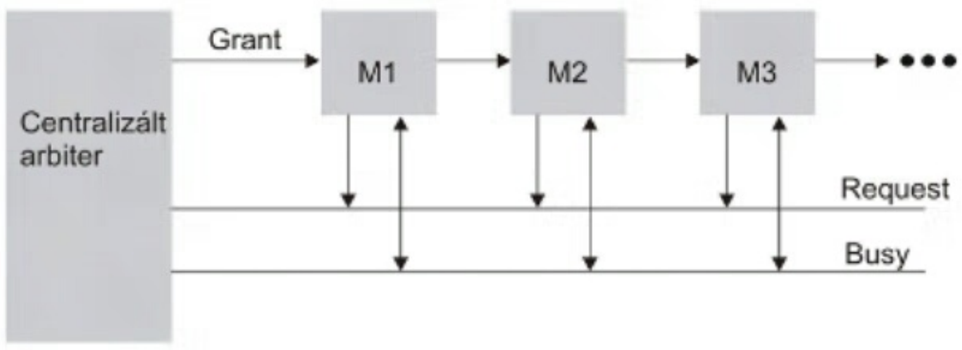
– Az egyes funkciókat egy-egy mikroszámítógép látja el (pl. robot egy-egy szabadsági fokát vezérlő egy-egy mikroszámítógép)
– Többprocesszoros orientáltság
– Üzenetorientált kommunikáció
– CPU és buszjelek függetlensége

Dinamikus arbitráció hardver megvalósításai



- Centralizált: az arbiter funkció egy helyen van megvalósítva
- Elosztott: az arbiter funkció elosztva, a potenciális mesterekben van megvalósítva

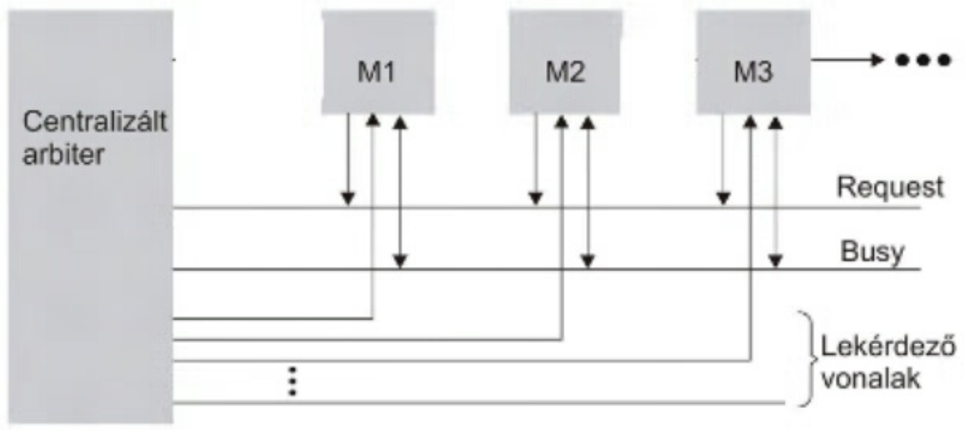
Centralizált, soros (daisy chain) megvalósítás



- **Request (kérelem).** A potenciális mesterek buszhasználati igényüket ezen a közös vonalon jelezhetik (huzalozott VAGY kapcsolat).
- **Busy (foglalt).** A busz foglaltságát jelzi. Mindig az aktuális mesternek kell ezt a vonalat aktív állapotban tartania.
- **Grant (engedély).** Az arbiter ennek a jelnek az aktív állapotával közli, hogy az általa észlelt buszkérelmet elfogadta.

- Ha egy modul foglalja (vezérli) a buszt, akkor a *Busy* jelet aktív állapotban tartja.
- Ha egy modul vezérelni akarja a buszt, akkor a *Request* jelet aktív állapotba vezérli.
- Az arbiter kiad egy aktív *Grant* jelet, ha aktív *Reguest* jelet észlel és a busz szabad, tehát a *Busy* jel nem aktív.
- Egy adott potenciális mester átengedi a bemenetére érkező aktív *Grant* jelet, ha a saját *Request* kérelme nem aktív.
- Egy potenciális mester a következő feltételek együttes teljesülése mellett veheti át a busz vezérlési jogát:
  - saját kérelme aktív,
  - a *Busy* jel nem aktív,
  - észleli a *Grant* jel aktívvá válását.

Centralizált, lekérdezéses (polling) megvalósítás



- Továbbra is egyetlen Request vonal szolgál a kérelem továbbítására és egyetlen Busy vonal a foglaltság jelzésére
- De egyetlen Grant vonal helyett, most annyi lekérdező (polling) vonala van a rendszernek, ahány potenciális mestert tartalmaz

A protokoll

- Az egyes potenciális mesterek a *Request* jel aktív állapotba vezérlésével jelezhetik igényüket.
- Ha a *Request* jel aktív, a *Busy* jel viszont nem, akkor az arbiter egymás után elkezdi aktivizálni a polling vonalakat (egy adott időben csak egyetlen vonalat vezérelve aktív állapotba, s azt egy előre meghatározott ideig aktív állapotban tartva).
- Egy potenciális mester, amelyiknek a saját *Request* jele aktív, érzékelve polling vonalának aktív állapotba váltását, a *Busy* jelet aktív állapotba vezérelheti.
- Az arbiter érzékelve a *Busy* jel aktív állapotba váltását, nem folytatja a pollingot, a lekérdezést.
- Az aktív mester addig használhatja a buszt, amíg a *Busy* jelet aktív állapotban tartja.

SATA

Az ATA interfész jelei, regiszterstruktúra, parancsok. A SATA általános jellemzői és réteges szerkezete. A fizikai réteg: általános jellemzők, jelek és funkciójuk, sávon kívüli jelek, tápellátási üzemmódok, beépített önteszt. Az adatkapcsolati réteg: keretadási szolgáltatások, a 8b/10b kódolási rendszer, a csomagok szerkezete, a primitívek, összekeverés. A transzportréteg funkciói.

Logikai jellemzők

Az interfész jelei (lényegében ISA-16 jelek)

DD(15:0) (Device Data)
- Kétirányú, 8 és 16-bites adat. Regiszter transzfer esetén 8, egyébként 16 bites az adatátvitel.

DIOR- (Device I/O read)
- A host strobe jele az eszköz regisztereinek vagy adatportjának olvasásakor.

DIOW- (Device I/O write)
- A host strobe jele az eszköz regisztereinek és adatportjának írásakor.

CS(1:0) (Chip select)
- A host kimenőjelei a Command Block és a Control Block regisztereinek kiválasztására szolgál (lásd a regiszterstruktúrát).

DA(2:0) (Device address)
- A host kimenőjelei az eszközön regisztereinek és adatportjának megcímzésére.

DASP (Device active, device 1 present)
- Aktív állapotával reset alatt az 1-es eszköz jelezheti jelenlétét ezzel a jellel. Más időben az aktív eszköz állíthatja aktív állapotba.

DMARQ (DMA request)
- Az eszköz DMA kérelmi jele, ha kész az adatátvitelre, akkor aktív állapot vezérli.

DMACK- (DMA acknowledge)
- A host jele a DMARQ jelre, az DMA megindítását jelzi.

INTRQ (Device interrupt)
- Az eszköz megszakításkérelmi jele a host felé.

IORDY (I/O channel ready)
- Passzív (L) állapotára a host az átviteli ciklust meghosszabbítja.

RESET (Hardware reset)
- A host jele az eszközök felé, azok alaphelyzetbe hozására.

CSEL- (Cable select)
- Ha a CSEL engedélyezve van az eszközben, akkor ez a jel állítja be a sorszámát. Ha CSEL 0, akkor az eszköz száma is 0, egyébként 1.

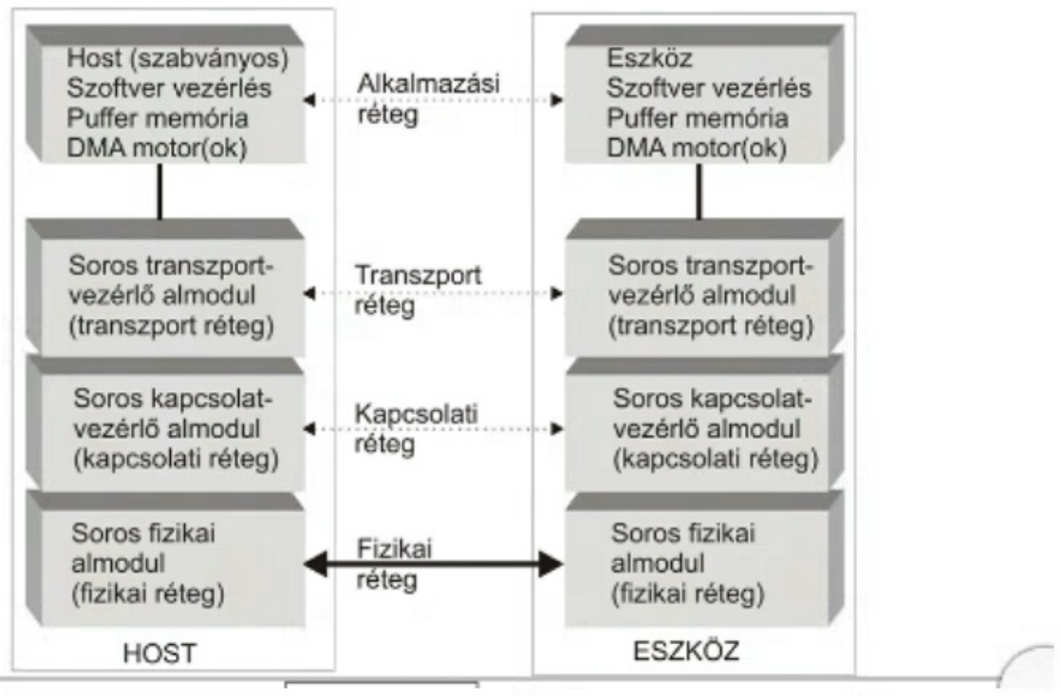
PDIAG-/CBLID- (Passed diagnostics/Cable assembly type identifier
- Az 1-es eszköz ezzel jelzi a 0-s eszköz felé, hogy befejezte a diagnosztizáló tevékenységét. Másrészt a host a tápfeszültség vagy hardver reset után ezt beolvasva detektálhatja, hogy a kábel 80 erű-e.

Parancsok

- Néhány olvasási parancs
  - Read Sector(s) /szektor(ok) olvasása/
  - Read Verify Sector(s) /szektor(ok) olvasása adatátvitel nélkül/
  - Read Sector Buffer /szektor puffer olvasása/
  - Read DMA /Olvasás DMA-val/
    - Multi-word DMA /blokkos DMA/
    - Ultra DMA
  - Read Multiply /mind Read Sector(s), de több szektor képez egyetlen blokkot/
- Néhány írási parancs
  - Write Sector(s) /szektor(ok) írása/
  - Write Verify Sector(s) /mint szektor(ok) írása, de azonnali ellenőrzéssel/
  - Write Sector Buffer /szektor puffer felülírása valamilyen mintával/
  - Write DMA
    - Multi-word DMA /blokkos DMA/
    - Ultra DMA
  - Write Multiply /mint Write Sector(s), de több szektor egyetlen blokként kezelve/

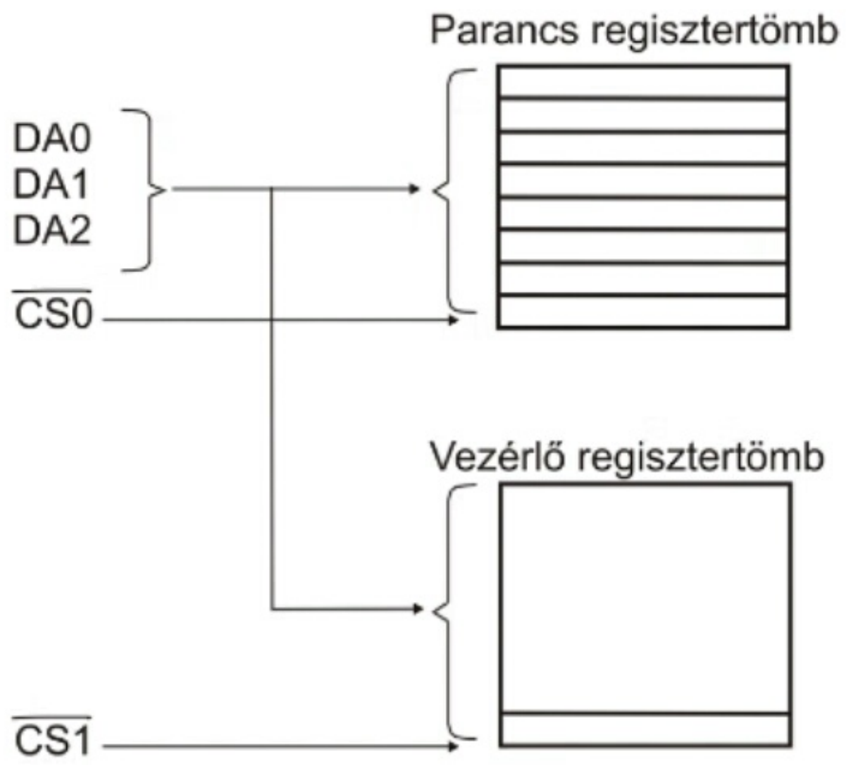
A SATA alrendszerek működése

A SATA réteges szerkezete



Regiszterstruktúra

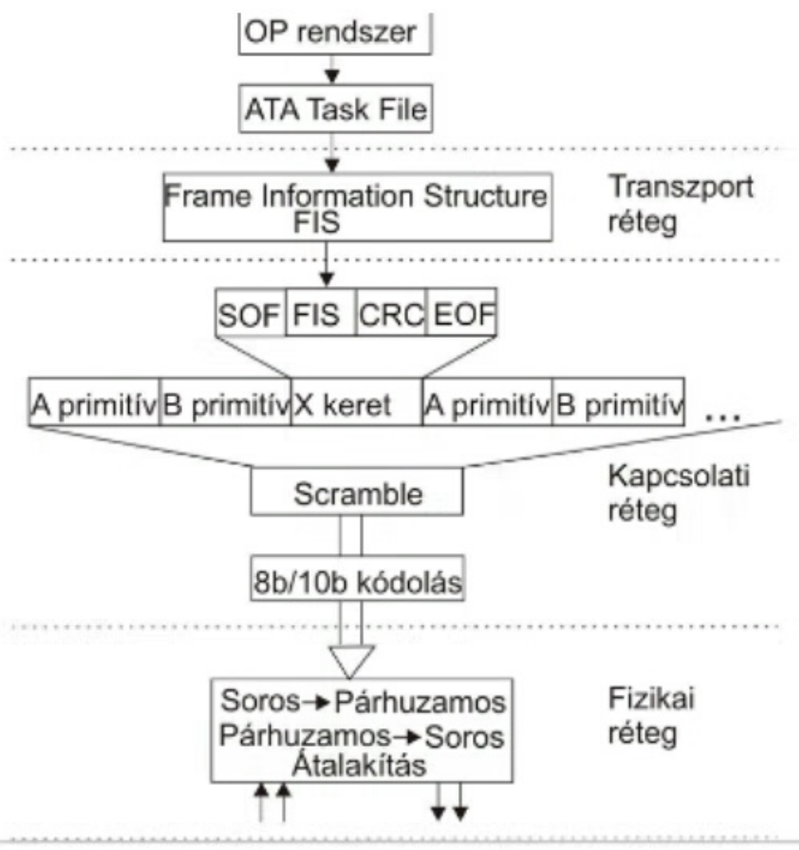
- A host (PC) egy regiszterstrktúrán keresztül kommunikál a meghajtóval
- Az egyes regiszterek kiválasztása a CS(1:0), DA(2:0), DIOR- és DIOW- jelek állapotától függ.
- A CS1 ás CS0 értékeitől függően az egyes regiszterek ún. regiszter blokkokba tartoznak, így meg szokták különböztetni a parancs (command) regisztereket és a vezérlő (control) regisztereket.
- Az IBM PC-ben a fenti jelek „generálása" meghatározott I/O címekre írással és olvasással történik. Az 0-s eszköz parancs regiszterei az 1F0H..1F7H címeken, vezérlő regiszterek pedig a 3F6H címen férhetők hozzá. Az 1-es eszközre ezek a címek 170H..177H és 376H.



SATA általános jellemzői

- A soros ATA egy nagysebességű soros interfész a merevlemezes tárolók illesztésére szolgáló párhuzamos ATA kiváltására, lehetővé téve nagyobb átviteli sebességet.
- Szimmetrikus jelátvitelt alkalmaz, amely a Gigabit technológián és a 8b/10b kódoláson alapszik.
- A soros ATA kifejlesztésekor alapvető szempont volt, hogy a párhuzamos ATA interfészre megírt szoftverek a soros ATA alkalmazásakor változatlanul használhatók legyenek.
- Az eszköz és a soros ATA host adapter együttesen emulálja a párhuzamos ATA eszköz viselkedését a BIOS vagy a szoftver meghajtó felé. A parancs és vezérlő blokk regisztereinek viselkedését, a PIO és a DMA adatátvitelt, a reset-et és a megszakításokat egyaránt emulálja az eszköz és a soros ATI host adapter.

- Az ATA interfész esetén az eszközben lévő parancs- és vezérlő reisztereket, mint árnyékregisztereket leképzi a SATA host adapter (árnyék taszk fájl), s az eszközmeghajtó továbbra is eze regiszterek írásával és olvasásával kommunikál az eszközzel.
- A SATA host adapter ezen árnyék regiszterfájl tartalma alapján állítja össze azokat az csomagokat, melyeket elküld a soros kommunikációs csatornán az eszköz (diszk) felé.



Fizikai réteg

Fontosabb jellemzők

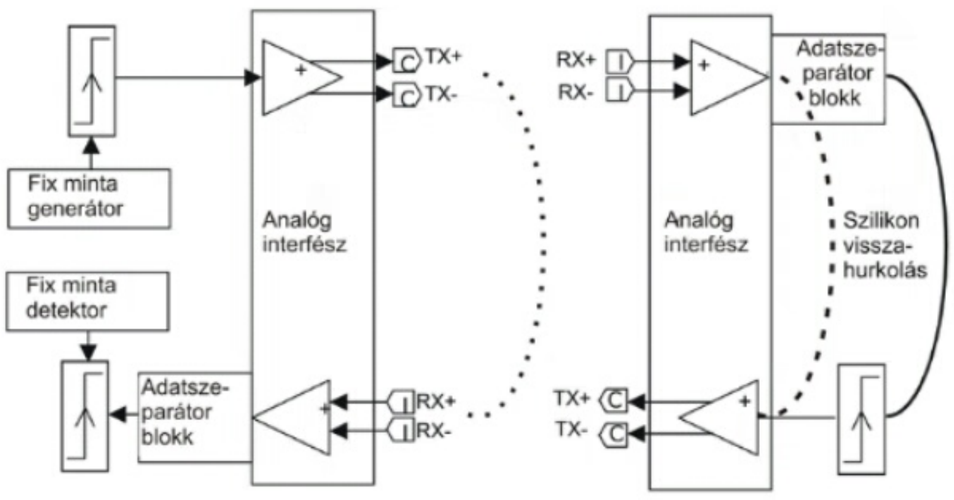
– A párhuzamos ATA esetén az átviteli sebesség növelésének a párhuzamos jelátvitel mellett a TTL jelszint és az aszimmetrikus jelátvitel használat volt a gátja.
– A SATA esetében mindhárom gátat megszüntették, azaz a SATA
  • a párhuzamos jelátvitel helyett **soros jelátvitelt**,
  • a TTL jelszintek helyett **csökkentett jelszinteket**,
  • s az aszimmetrikus jelátvitel helyett **szimmetrikus jelátvitelt** alkalmaz.

Maguk a jellemzők

– 1,5Gb/s, 3Gb/s és 6Gb/s (16Gb/s SATAexpress lásd később) adatátviteli sebesség NRZ kódolással
– Kis jelszintű, differenciális jelátvitel
– A 100 ohmos kábel mindkét végén 100 ohmmal (szimmetrikusan) lezárva
– A link réteg 10, 20, 40, stb. bites párhuzamos kimenetét soros jellé alakítva továbbítja a vevő felé, illetve a vett soros adatokat 10, 20, 40, stb. bites párhuzamos formába alakítva adja át a link rétegnek
– Speciális OOB (Out-of-band = sávon kívüli) jelek vétele és adása
– Tápfeszültség sorrendiség kezelése és sebesség egyeztetés
– Státusjelzés a link réteg felé: eszköz jelen; eszköz hiányzik; eszköz jele, de az egyeztetés sikertelen

Beépített önteszt (BIST=Built In Self Test)

– A beépített önteszt a fizikai réteg elemeinek visszahurkolásos tesztelését teszi lehetővé. Háromféle beépített önteszt áll rendelkezésre:
  • Távoli végen visszahurkolt, újraidőzített (Far-End Retimed) – kötelező
  • Távoli végen visszahurkolt, analóg (Far-End Analog) – opcionális
  • Közeli végen visszahurkolt, analóg (Near-End Analog) – opcionális

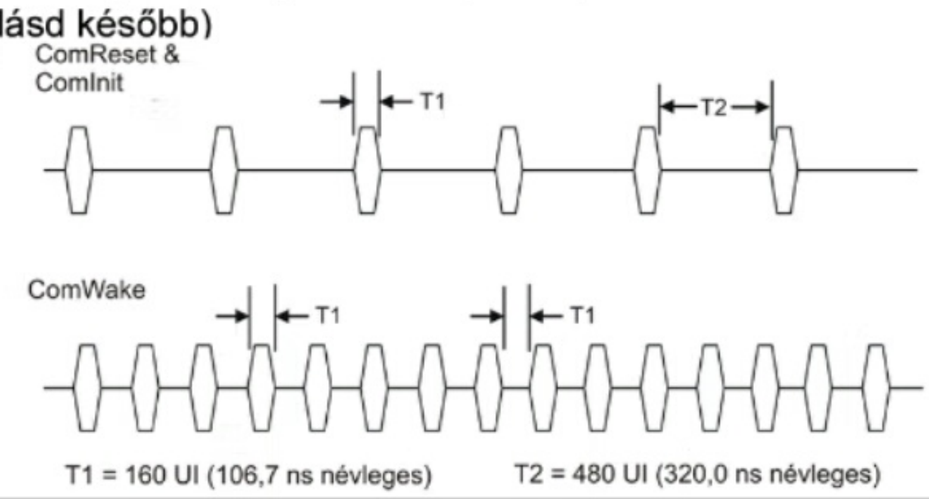


Kódolási rendszerrel szembeni követelmények

– A nagysebességű adatátvitel megköveteli az egyenáramú kiegyenlítettséget
  • A bitfolyamban az egymást követő 1-esek és 0-k száma lehetőleg minél rövidebb bithosszon közel azonos legyen.
  • Ekkor nem tolódik el jelentősen a csatorna egyenfeszültségű szintje, nem töltődnek fel a parazita kapacitások, melyek kisütése a sebesség ellen hat.
– Elegendő állapotváltozást tartalmazzon az órajel kinyerésére.
– Az adat kódszavaktól meg kell tudni különböztetni néhány, vezérlési és státusz célokra használandó kódszót.
– Eleget tesz az 1983-ban publikált 8b/10b kódolási rendszer
  • amely a 8-bites kódszavakat 10 bites kódszavakba transzformálja.
  • Az 1-esek és 0-k számának különbsége legalább 20 bites stringben nem több mint 2, s legfeljebb öt 1-es vagy legfeljebb öt 0-s bit követi közvetlenül egymást.
  • A SATA mellett a PCI Express, IEEE 1394b, SAS, Fibre Channel, SSA, Gigabit Ethernet, InfiniBand, HyperTransport, stb. rendszer is alkalmazza.

Sávon kívüli jelek

– Három sávon kívüli jelet használ a SATA, ezek a COMRESET, COMINIT és a COMWAKE
– A sávon kívüli jeleket 160 bitidőnyi (106,7ns), ALIGN primitíveket tartalmazó lökettel (burst), és a löketek közötti nyugalmi állapottal (nincs differenciális jel, csak közös módú) állítják elő
– ALIGN primitív: speciális duplaszó, szinkronzálási célokat szolgál (lásd később)



Tápellátási üzemmódok

– Aktív üzemmód: A fizikai almodul aktív. Az intefész szinkronizált állapotban van, s képes adatokat adni és venni. Ezt az állapotát a PhyRdy jel jelzi.
– Részleges tápellátási üzemmód: A fizikai almodul csökkentett tápellátást kap. Az interfész külső jelei semleges logikai állapotban (közös módú feszültségen) vannak. Ebből az állapotból a rendszer nem több mint 10µs alatt át tud térni az aktív állapotba.
– Slumber tápellátási üzemmód: Mint a részleges, de az áttérési idő az aktív állapotba akár 10ms is lehet.

Adatkapcsolati réteg

Keretadás szolgáltatások

– Egyezkedés a cél kapcsolati réteggel, s az arbitrációs konfliktusok feloldása
– A transzport rétegtől átvett információ keretbe ágyazása (SOF, CRC, EOF, stb. hozzáadása)
– Adatok átvétele a transzport rétegtől duplaszó (DWORD) formájában
– CRC számítása a transzport réteg adatain
– Keret elküldése és a nyugtázás vétele a cél kapcsolati rétegtől
– A transzport réteg értesítése a hibátlan vagy hibás adásról
– 8b/10b kódolás
– Összekeverés (scrambling) az EMI kibocsátás csökkentése érdekében.

Keretvétel szolgáltatások

– A küldő kapcsolati réteg felé a keret vételére kész állapot nyugtázása
– Adatok átvétele a fizikai rétegtől kódolt karakterek formájában
– 8b/10b dekódolással az adatok duplaszóba transzformálása
– A keret becsomagolásának (SIF, CRC, EOF) eltávolítása
– A CRC számítása a duplaszavakon, s a számított és vett CRC összehasonlítása
– A transzport réteg és a küldő kapcsolati réteg értesítése a hibátlan vagy hibás vételről
– Az összekevert adatok visszatranszformálása.

A kódrendszer jellemzői

– A 8 bites adatoknak 10 bites entitásokat, más néven *szimbólumokat* vagy *karaktereket* feleltet meg.
– A 8 bites bemenet alsó 5 bitjét 6 bitbe, a felső három bitet pedig 4 bitbe transzformálja. Ezek összekapcsolásával jön létre a 10 bites szimbólum.
– Az *adat szimbólumokat* gyakran D.x.y módon jelölik, ahol x a 0..31, y pedig a 0..7 tartományba esik.
– 12 *speciális szimbólum* (vagy *vezérlő karakter*), melyek az *adat szimbólumok* között előfordulhatnak. Ezeket gyakran a keret kezdetének (SOF = Start Of Frame), a keret végének (EOF = End Of Frame), a kapcsolat nyugalmi állapotban, stb. jelzésére használják.
– Különleges szimbólum az ún. *komma szimbólum*, amellyel a 10 bites szimbólumok elhelyezkedését jelzik. Ennek jele K.x.y, s valamennyi D.x.y szimbólumtól különbözik.

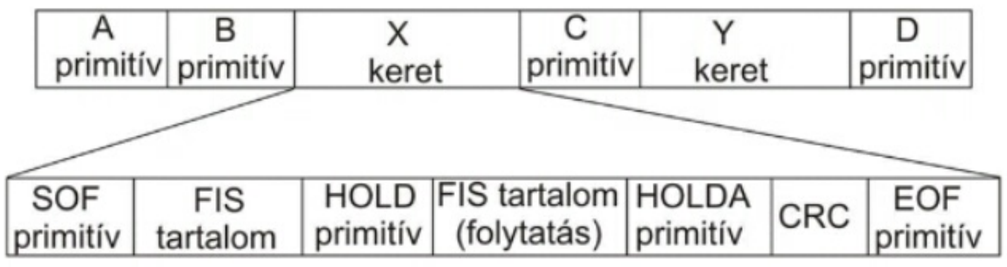
– Mivel a 10 bit esetén a lehetséges kódszavak száma 1024, ezek közül bizonyosakat ki lehet hagyni, hogy egy kódszóban az egymást követő azonos bitek száma ne lépje túl az 5 értéket, továbbá az 1-esek és 0-k számának különbsége ne legyen nagyobb mint kettő. A bemeneti 256 lehetséges kódszó megfeleltetése végrehajtható úgy, hogy
  - Egy részükben a pontosan öt 1-es és öt 0-s bitet tartalmaz
  - Másik részükhöz kétféle kód rendelhető, az egyikben az 1-esek száma hat, a 0-k száma pedig négy, illetve a másikban az 1-esek száma négy, a 0-k száma pedig hat. Például a bemeneti 01101000 (68H) bemeneti kódhoz a 111001 0011 és a 000110 1100 transzformált kódok, a 01110000 (70H) bemeneti kódhoz pedig a 011011 0011 és az 100100 1100 transzformált kódok tartoznak.
– Az 1-esek és 0-k számának közelítő kiegyenlítése ezután már kézenfekvő. Ha például az utoljára elküldött szimbólumban az 1-esek száma hat volt, s ezután a 70H-nak megfelelő transzformált szimbólumot kell elküldeni, akkor a hat és négy 1-est tartalmazó szimbólum közül a négy 1-est tartalmazót kell elküldeni, hogy az 1-esek száma kiegyenlítődjön.

Primitívek

– A soros vonal vezérlésére vagy státuszának jelzésére szolgálnak.
– A primitívek mindig vezérlő karakterrel kezdődnek, az ALIGN primitív a K.28.5 (komma) karakterrel, a többi primitív pedig a K.28.3 karakterrel. A vezérlő karaktert követően három kódolt bájt fejezi be a primitívet.
– ALIGN primitív
  - Ez a primitív (kódolása D27.3-D10.2-D10.2-K28.5) a soros bitfolyamban arra szolgál, hogy a (vevő) fizikai réteg hozzáigazíthassa, szinkronizálhassa működését a vett bitsorozat struktúráihoz. Ezt a primitívet mindig párban küldi el az adó.
  - A kapcsolat létrejötte után a kapcsolati réteg két ALIGN primitívet ad át a fizikai rétegnek, amit 254 nem ALIGN duplaszó követ. Majd ezt a ciklust többször megismételheti.
  - A vevő fizikai réteg általában nem adja tovább ezt a primitívet a kapcsolati réteg felé, ha mégis, akkor az figyelmen kívül hagyja.
– CONT primitív
  - Ez a primitív (kódolása D25.4-D25.4-D10.5-K28.3) lehetővé teszi hosszan ismételendő primitívek kiküszöbölését. Hatására az előtte vett ismételt primitív mindaddig fenntartandó, amíg egy másik primitív vétele be nem következik. A CONT és az ALIGN primitíveken kívül minden más primitív vétele megszünteti a primitívek automatikus ismétlését.
  - Az emittált elektromágneses zavarok csökkentésére az adatok összekeverését is elvégzi a kapcsolati réteg. Ennek hatékonyságát viszont nagymértékben rontanák a hosszan megismételt primitívek. Ezt küszöböli ki a CONT
– SOF és EOF primitívek
  - Egy keret kezdetét és végét jelölik ki.
– HOLD és HOLDA primitívek
  - A HOLD primitívet az átviendő információ közben akkor küldi el az adó, ha nem kész az adásra. Az ellentétes csatornán pedig a vevő küldheti ezt a primitívet, ha nem kész a további információ vételére.
  - A HOLDA primitívet mindaddig küldi a vevő, amíg HOLD primitívet vesz.
  - A vevő HOLD-jára nem jön azonnal az adó HOLDA-ja, ezért a vevőnek már puffere betelése előtt el kel küldemnie a HOLD-t!

Átviteli mód

– Az átviendő információ primitív és keret struktúrákat tartalmaz.
– Egy primitív egyetlen DWORD-ből áll, s különbözik a többi primitívtől, illetve az adatokban előforduló bármilyen mintától. Elsősorban real-time állapotinformáció közlésére, az információátvitel vezérlésére és a host/eszköz közötti kommunikáció koordinálására szolgálnak. A primitívek minden bájtja konstans, s első bájtjuk mindig egy speciális karakter (a K.28.3 vagy a K28.5).
– Egy keret több DWORD-ből áll, mindig egy SOF primitívvel kezdődik, ezt követi az átviendő információ az ún. Keret Információs Struktúra (FIS), amit a CRC és végül az EOF primitív követ. Bizonyos forgalomvezérlő primitívek (HOLD, HOLDA és CONT) elhelyezhetők az SOF és EOF primitívek között a partnerek közötti sebesség illesztésére.



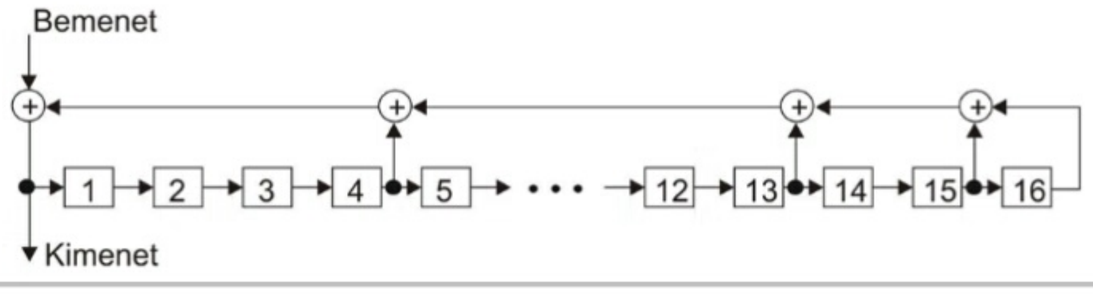
Az átviteli struktúra

Összekeverés (scrambling)

– A keretek tartalmát az SOF és az EOF között a CRC-t beleértve az emittált elektromágneses interferencia csökkentése érdekében speciálisan transzformálni (scrambling = összekeverés) kell. A transzformálás egy lineáris léptető regiszterrel történik, melynek generátor polinomja

$$g(x) = x^{16} + x^{15} + x^{13} + x^4 + 1$$

  s melyet FFFFH értékkel kell inicializálni.
– A transzformálás duplaszavakon történik



Transzport réteg

– Összeállítja az ún. Keret Információs Struktúrát (FIS = File Information Structure) a kapcsolati réteg számára
– Átveszi a kapcsolati rétegtől és szétbontja a FIS-t.
– A FIS összeállítása
  - Összegyűjti a szükséges FIS tartalmat a kért FIS típusnak megfelelően.
  - A FIS tartalmat a helyes sorrendbe rendezi.
  - Értesíti a kapcsolati réteget a szüksége keret elküldésére és átadja a FIS tartalmat a kapcsolati rétegnek.
  - Menedzseli a puffer/FIFO forgalmat, s értesíti a kapcsolati réteget a szükséges forgalomvezérlésről.
  - Átveszi a vétel nyugtázását a kapcsolati rétegtől.
  - Értesíti a kérelmet kezdeményező felsőbb réteget a hibátlan vagy hibás adatátvitelről.
– A FIS szétbontása
  - Meghatározza a FIS típusát.
  - Szétosztja a FIS tartalmát a FIS típusa által meghatározott helyekre.
  - A host transzport rétege esetén a FIS vétele egy, az eszköznek elküldendő FIS összeállítását válthatja ki.
  - Értesíti a felsőbb réteget a hibátlan vagy hibás vételről.

USB

Verziók, fontosabb jellemzők. Busz topológia, funkciók és végpontok, keretek, adatfolyam típusok. Mechanikai jellemzők Elektromos jellemzők: jelszintek, sebességidentifikáció. Kódolás. Logikai jellemzők: tranzakciók, token-, adat- és kézfogásos csomagok, a PID. USB leírók és a konfigurálás. Az USB 3.0 járulékos tulajdonságai és szolgáltatásai.

- A fenti hiányosságokat felismerve hét vállalat (Compaq, Digital Equipment, IBM, Intel, Microsoft és Northern Telecom) összefogott, hogy kidolgozzon egy specifikációt elsősorban a hagyományos interfészek (RS232, nyomtató interfész, PS2 interfész, stb.) egyetlen interfészbe foglalására.
- Ennek az eredménye lett az 1996 január 15-én megjelent

  USB 1.0 változatú

  specifikációja.
- Ez a változat két adatátviteli sebességet specifikált:
  - alacsony sebességű (low speed, 1,5Mb/s)
  - teljes sebességű (full speed, 12Mb/s).
- A két adatátviteli sebességgel az a lehetőséget biztosították, hogy akár a kis költségű perifériák (egér, billentyűzet, stb.) akár a nagyobb sebességű perifériák (nyomtató, szkenner, stb.) ugyanazt a soros buszt használhassák.

- 1998 szeptember 23-án az

  USB 1.1

  specifikációja, amely lényegében a korábbi inkompatibilitások-hoz vezető időzítéseket tisztázta, de lényeges funkcionális továbbfejlesztést nem tartalmazott.
- 2000 április 27-én jelent meg az

  USB 2.0

  specifikációja, amely az eddigi két adatátviteli sebesség mellett egy harmadikat, a

  480Mb/s

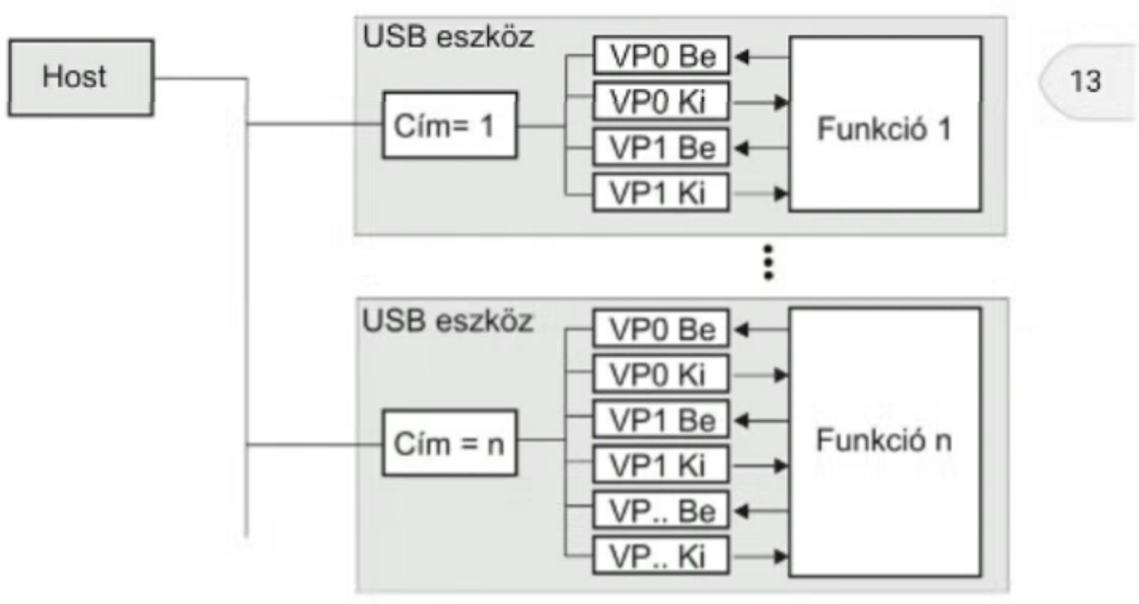
  nagysebességű üzemmódot is specifikálta.
- Ezzel a sebességgel tovább bővült az USB-n keresztül illeszthető eszközosztályok száma. Ezek közé tartoznak az egerek, billentyűzetek, szkennerek, nyomtatók, külső tárolók, hálózati komponensek, digitális kamerák, stb. Több eszköz esetén(digitális kamera, szkenner) az USB lett a szabványos csatlakoztatási mód. Az USB-n keresztül csak az igen nagy adatátviteli sebességet igénylő perifériák nem csatlakoztathatók, mint például diszplék és monitorok, valamint a felső kategóriás digitális videó komponensek.

- 2008.11.17-én jelentették be az

  USB 3.0

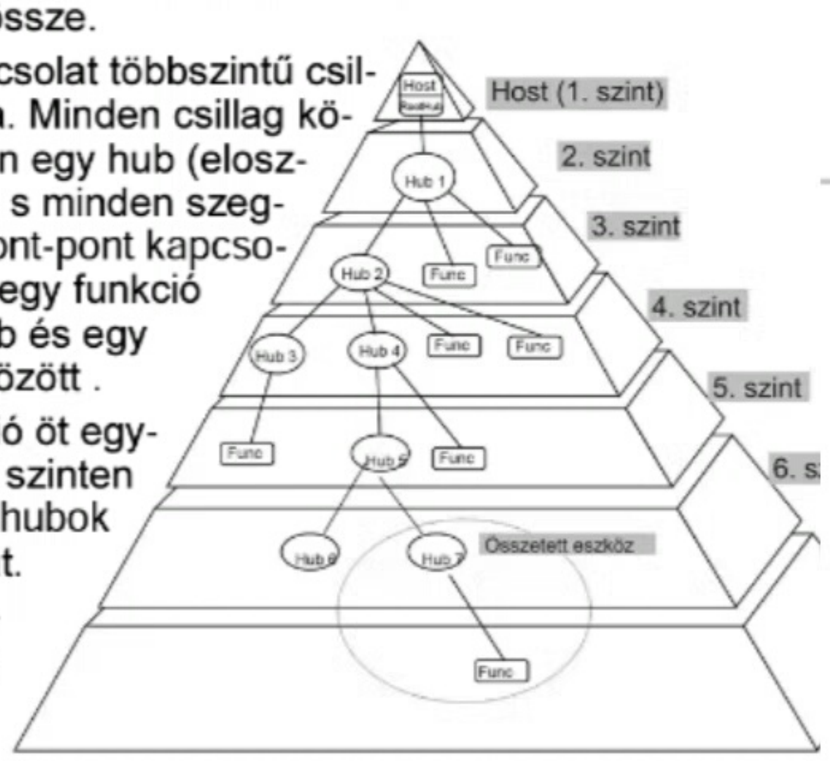
  specifikációt, amely egy negyedik átviteli sebességet (SuperSpeed) vezetett be, melynek értéke 5Gb/s. Ez a sebesség durván 400MB/s sávszélességet biztosít! Előnyei a korábbi változatokkal:
  - Sebesség 5.0Gb/s szemben az USB 2.0 480Mb/s sebességével
  - Gyorsabb eszközidentifikáció
  - Fejlettebb tápellátás menedzsment és hatékonyság
  - Nagyobb tápáramellátási képesség az eszközök felé
  - Optikai és hagyományos USB-vezetékes kapcsolat
- 2013.07.31. Az USB 3.1 specifikáció megjelenése
  - Sebesség 10.0Gb/s
- Részletesen az USB 2.0 specifikáció, USB 3.0 új jellemzői később

Funkciók és végpontok



Busz topológia

- Az USB az USB eszközöket az USB hosttal köti össze.
- A fizikai kapcsolat többszintű csillag topológia. Minden csillag középpontjában egy hub (elosztó) található, s minden szegmens egy pont-pont kapcsolat a hub és egy funkció vagy egy hub és egy másik hub között .
- A specifikáció öt egymást követő szinten engedi meg hubok alkalmazását.
- A max. csatlakoztatható eszközök száma 127.



- Minden USB rendszer csak egy hostot tartalmaz. Az USB host számítógép közötti interfész egységet host vezérlőnek (= Host Controller) nevezik. A host rendszer részét képező gyökér hub egy vagy több csatlakozási ponttal rendelkezik
- Az USB eszközök két csoportban oszthatók:
  - hubok
  - funkciók.
- A hubok járulékos csatlakozási pontokat biztosítanak a rendszerben, a funkciók pedig meghatározott képességet prezentálnak a host felé mint például ISDN kapcsolat, billentyűzet, egér, stb.
- Az USB eszközök szabványos interfészt mutatnak a host felé: az USB protokollt használják, reagálnak a szabványos USB műveletekre (konfigurálás, reset, stb.).

- A funkcióknak, mint USB eszközöknek egyedi címük van, továbbá a funkción belül 16 bemeneti és 16 kimeneti megcímezhető ún. végponttal rendelkezhetnek.
- A végpontok mint adatforrások és adatnyelők tekinthetők (a ki- és bemenet iránya a host felől értendő). Például, ha a szoftver meghajtó egy csomagot küld a végpont 1-nek, akkor az eszköz VP1 Ki pufferébe kerül, ahonnan az eszközt vezérlő processzor kiolvashatja (értesítést is kap a processzor, hogy érvényes adat van a pufferben).
- Ha a funkció válaszolni akar, nem küldhet közvetlenül a buszra adatokat, mivel a buszt a host vezérli. Ezért az adatokat az VP1 Be pufferbe írja, s azok mindaddig ott lesznek, amíg a host egy IN csomagot nem küld, amely felszólítja a végpontot az adatok elküldésére.
- Minden eszköznek támogatnia kell a 0-s végpontot (VP0). Ez a végpont kapja meg a vezérlési és státus kérelmeket. A többi végpont támogatása opcionális.

Keretek

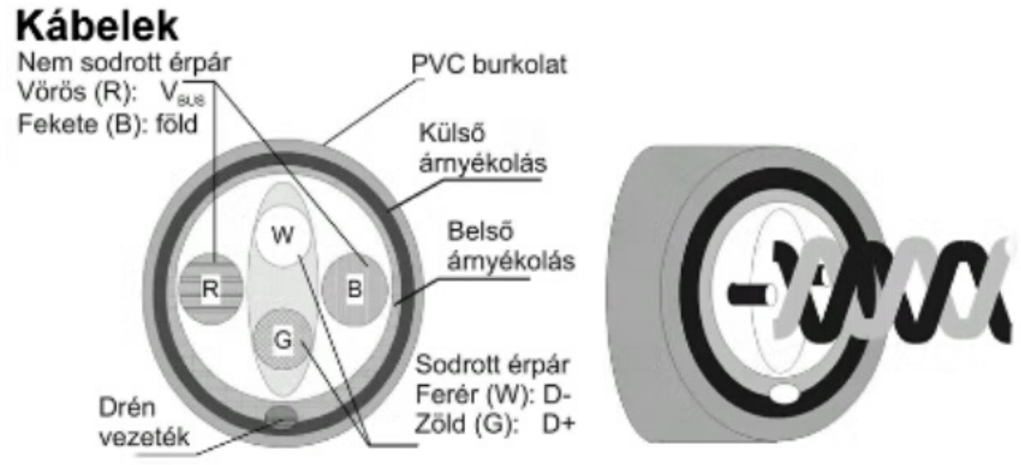
- Az USB-n az adatátvitel keretekbe osztva megy végbe. A gyökér hub minden milliszekundumban elküld egy SOF csomagot (Start-of-Frame = Keret kezdet). Két egymást közvető SOF csomag közötti időtartamot keretnek nevezzük. A nagysebességű hubok ezen kívül 8 mikrokeretet is generálnak egy kereten belül járulékos SOF csomagokkal.
- A keretekben megvalósítandó tranzakciókat adatfolyam típus (lásd később) szerint ütemezi a rendszer. Így a keretben rendelkezésre álló idő 90%-át az azonos idejű adatátvitel és megszakításos átvitel foglalhatja el, a fennmaradó 10% a vezérlési- és a tömegadat átvitelére használható.

Adatfolyam típusok

- Az USB a host és az eszköz között funkcionális adat- és vezérlésátvitelt biztosít un. egyirányú és kétirányú csövekkel.
- Az adatátvitel a host szoftvere és az eszköz egyik végpontja között megy végbe.
- A host szoftverének és az eszköz végpontjának ilyen összerendelését csőnek nevezzük. Egy adott csövön az adatátvitel független a többi csövön végbemenő adatátviteltől.
- Egy USB eszköz több csövet tartalmazhat. Az egyik csövén például olvasási, a másikon pedig írási folyamat mehet végbe.
- A csöveken végbemenő adatátvitel négyféle lehet:
  - Vezérlésátvitel (Control Transfer): Elsősorban az eszköz csatlakoztatásakor a konfigurálás céljaira használatos. Emellett egyéb eszközspecifikus célokat is elláthat, például az eszköz csöveinek vezérlését.
  - Tömegadat-átvitel (Bulk Data Transfer): Nagymennyiségű adat írása vagy olvasása különösebb időkorlát nélkül (például nyomtató vagy szkenner esetén). Az adatátvitel szekvenciális. A megbízható adatátvitelt hardver szinten a hibafelismerés és hiba esetén a korlátozott számú ismétlés teszi lehetővé. A tömegadat-átvitel számára rendelkezésre álló sávszélesség a busz egyéb aktivitásaitól függ.
  - Megszakításos átvitel (Interrupt Data Transfer): Véletlenszerűen fellépő, minimális latenciával átviendő, kismennyiségű adatok esetén használják (például egy pozícionáló eszköz által beviendő koordináta érték).
  - Azonos idejű adatátvitel (Isochronous Data Transfer): Az USB sávszélességének előre egyeztetett részét használja előre egyeztetett latenciával. Az isochronous adat folytonos és valós idejű a keletkezésében, továbbításában és az elfogyasztásában (tipikus példák az audio és a videó adatok). A potenciális tranziens hibák nem kerülnek kijavításra, azaz ebben az átviteli módban nincs csomagismétlés.

- Az USB 2.0 hibamelléklete (errata) egy mini-USB B csatlakozót is specifikált. Erre az igény a mobiltelefonok, kamerák, stb. oldaláról jelentkezett, mivel az eredeti B csatlakozó túlságosan nagy volt, hogy ezekbe az eszközökbe lehetett volna integrálni.
- További igényként jelentkezett, hogy a mobil eszközök korlátozott host funkciót is elláthassanak. Az USB 2.0 On-The-Go kiegészítése ennek lehetőségeit specifikálta, s definiált egy mini-A csatlakozópárt, valamint egy mini-AB csatlakozó aljzatot. Az utóbbiba akár a mini-A, akár a mini-B csatlakozó dugó is bedugható.
- Az On-The-Go specifikációban definiált duális szerepű USB eszköz ugyanis akár korlátozott hostként, akár közönséges perifériaként is funkcionálhat anélkül, hogy a buszról le kellene csatlakoztatni

Kábelek



- A teljes és nagysebességű árnyékolt USB kábelek sodrott érpáron viszik át a differenciális jelet. Kissebességű esetben az USB 1.1 változata szerint nem kötelező az árnyékolás és a sodrott érpár használata. Az USB 2.0 azonban szigorította a kis sebességű kábelek specifikációját. Ez már megköveteli a belső árnyékolást, s bár nem írja elő, de javasolja a külső árnyékolást és a sodrott érpár használatát.

Mechanikai jellemzők

Csatlakozók

- Az USB csatlakozók négy érintkezőt tartalmaznak. Ebből kettő a differenciális jel átvitelére, egy-egy pedig a +5V és a föld átvitelére szolgál.
- A szabvány kétféle csatlakozót specifikál, egy A és egy B típusút.
- Az A típusú mindig a host felé irányul (ún. upsteram csatlakozó), a B típusú meg a hosttól el (ún. downstream csatlakozó).
- A két csatlakozó típus nem cserélhető fel, így például kizárt, hogy egy hub hosttól elmutató kapuját egy másik elmutató kapujával összekapcsoljuk (illegális visszahurkolás).
- A számítógép USB csatlakozói, valamint a huboknak az eszközök felé mutató csatlakozói tehát A típusúak, a B típusú csatlakozók pedig az eszközökön a host felé mutatnak. Gyakran persze az eszközök fix kábellel rendelkeznek, ilyenkor nem látható rajtuk a B típusú csatlakozó.

- Az USB 1.0 specifikálta a kábelszegmensek maximális hosszát. Ez kis sebességű esetben maximum 3m, teljes sebességű esetben pedig 5m lehet.
- Ellentmondásnak tűnhet, hogy kisebb sebességen rövidebb a megengedett hossz. Ennek oka, a gyengébb minőségű kábel. Figyelembe véve, hogy legfeljebb 5 hub kapcsolható sorba, a legtávolabbi eszköz teljes sebességű esetben is legfeljebb 30m-re lehet a hosttól.
- Az USB 1.1 változta már elhagyta a maximális hossz megadását, s ezt indirekten írta elő az időzítési és feszültség követelményekkel. Nevezetesen kis sebességű esetben a kábel hosszát a jelek fel- és lefutási idői, a szegmens kapacitív terhelése és a VBUS, GDN vonalakon fellépő feszültségesés korlátozza.
- Teljes- és nagysebességű esetben a hosszt a megengedett jelcsillapítás, a kábel terjedési ideje és a VBUS, GDN vonalakon fellépő feszültségesés korlátozza. Az eredeti 3m és 5m korlátok azonban továbbra is jó irányadók.

Elektromos jellemzők

Az USB kisjelszintű, differenciális jelátvitelt alkalmaz. Kis- és teljes sebességű esetben differenciális feszültséggel, nagysebességű esetben pedig differenciális árammal specifikálja a két logikai állapotot. Ezen kívül bizonyos jelzésekre ún. single ended zero (SE0, minkét jelvezeték 0 szintre kapcsolása) állapotot is használ.

Adó-vevők és jelszintek

- A kis- teljes és nagysebességű eszközök eltérő sajátossággal rendelkeznek, s ez az USB 1.1, illetve USB 2.0 specifikációnak eleget tevő hubokra is érvényes. Az 1.1 specifikációjú hubokhoz kis- és teljes sebességű eszközök csatlakoztathatók, a 2.0 specifikációjúhoz pedig mindhárom sebességű eszköz.
- A 2.0 specifikációjú eszközök azonban nem kötelesek támogatni a nagysebességű üzemmódot, ha viszont támogatják, akkor a kissebességűt nem támogathatják.

USB 1.1 jelszintek

- A differenciális feszültség az adónál minimum 1,9V, a vevőnél pedig 0,2V. Tehát jelentős zajtartalékkal rendelkezik a rendszer.
- Az aszimmetrikus 0 busz állapot (Single Ended Zero = SE0) a csomag végét (End Of Package = EOP), lekacsolást vagy törlést jelenthet. Az aszimmetrikus 1 állapot az aszimmetrikus 0 állapot inverze, ez azonban tiltott állapot, hibamentes esetben soha nem fordulhat elő.
- A kis- és a teljes sebességű esetben a diffrenciális jelszintek egymásnak inverzei. Ezért bevezették a J és a K állapotokat, hogy a sebességtől függetlenül egységesen tárgyalni lehessen a jelátvitelt.
- A tétlen állapotban egyetlen meghajtó sem aktív. Ilyenkor csak a felhúzó ellenállások határozzák meg a buszvonalak szintjét. Kis sebességnél a D- vonal, teljes sebességnél pedig a D+ vonal a sokkal pozitívabb.
- Ha az eszköz a kis fogyasztású, felfüggesztett állapotban van, akkor a K adatállapot váltja ki a visszakapcsolást az aktív állapotba.

- A csomag kezdet (Start-of-Packet =SOF) állapotot az jelzi, hogy a tétlen állapot K adatállapotba megy át.
- A csomag végét (End-of-Package = EOP) az jelzi, hogy a busz legalább egy bitideig SE0 állapotban kerül, melyet legalább 1 bitideig J adatállapot követ.
- Egy downstream kapu lekapcsolt állapotba kerül, ha kimenetén a SE0 jel több mint 2,5µs ideig fennáll.
- Egy downstream kapu rákapcsolt állapotba kerül, ha a tétlen állapota legalább 2,5µs, de legfeljebb 2,0ms-ig fennáll.
- Ha a SE0 állapot több mint 10ms-ig fennáll, akkor az eszköznek a reset állapotba kell kerülni, de már 2,5µs elteltével is reset állapotba kerülhet. A nagy sebességű üzemmódra is alkalmas teljes sebességű eszközök a sebesség azonosítást a reset állapotban végzik el. A reset állapotot elhagyva az eszköznek a korrekt sebességgel kell működnie a kiindulási 00h címmel.

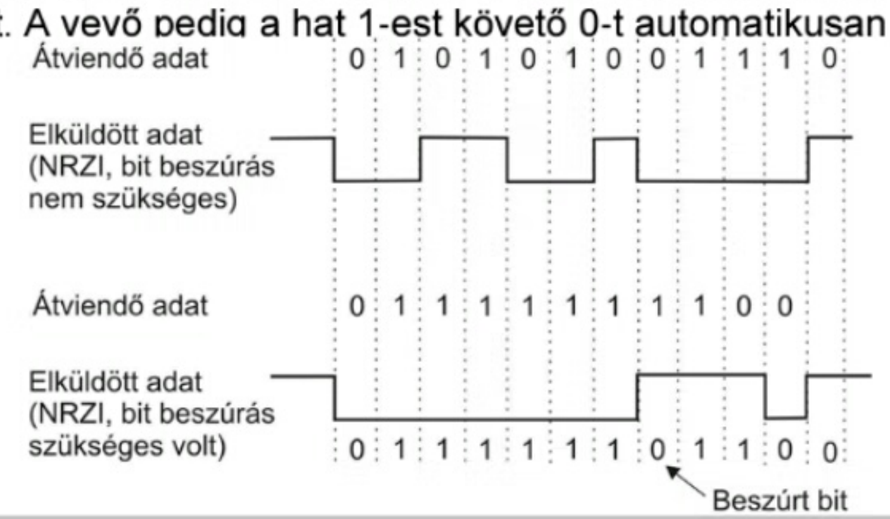
Sebesség identifikáció

- Az USB eszközöknek azonosítani kell magukat a host (hub) felé, hogy milyen kommunikációs sebességgel üzemelnek. Ezt a D+ vagy a D- vonal 3,3V-ra felhúzásával jelzik.
- A felhúzó ellenállások azt is lehetővé teszik, hogy a host (hub) detektálhassa az eszköz jelenlétét vagy hiányát a vonal másik végén. Ha ugyanis nem kapcsolódik eszköz egy downstream kapura, akkor mindkét vezeték közel földpotenciálon van a host (hub) két lehúzó ellenállása miatt.

Nagysebességű eszközök:

Kódolás

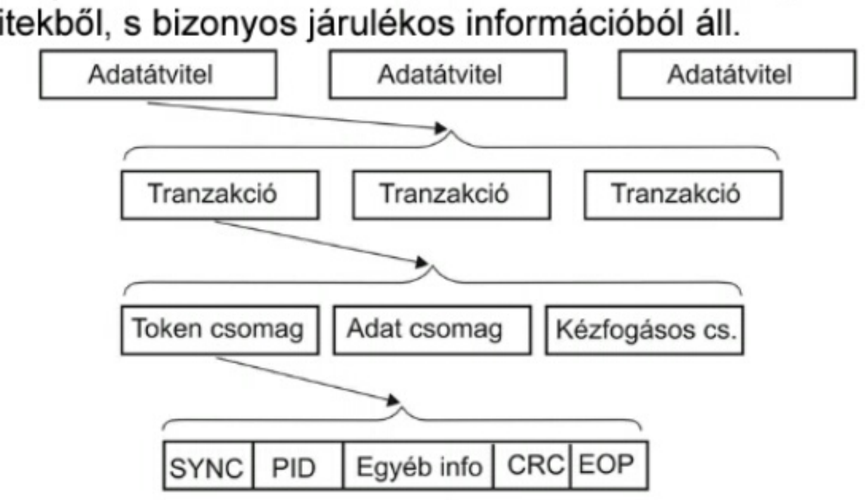
- Az USB rendszer a buszon a jelek kódolására az NRZI módszert használja bitbeszúrással kiegészítve. NRZI esetén „1” írásakor nem változik a vonal állapota, minden „0” írásakor viszont ellentétesre változik. Sok 1-es esetén nem változik a jel, aminek következtében a vevő kieshetne a szinkronizált állapotból. Ezért hat egymást követő 1-es után az adó automatikusan beszúr egy 0-t. A vevő pedig a hat 1-est követő 0-t automatikusan „eldobja”.



Logikai jellemzők

Az adatátvitel elemei az USB buszon

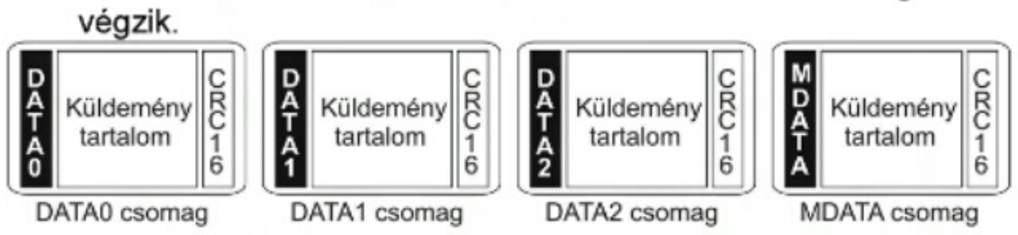
- Bizonyos mennyiségű adat átvitele egy vagy több tranzakcióból, egy tranzakció egy, kettő vagy három csomagból, egy csomag pedig azonosító mezőből (PID = Packet Identifier), PID ellenőrző bitekből, s bizonyos járulékos információból áll.



- A token csomagok a tranzakciót vezetik be és határozzák meg annak típusát. Az adatcsomagokban a küldeménytartalom kerül átvitelre. A kézfogásos adatok az adatok vételét nyugtázzák, míg a speciális csomagok a „sebesség konverzió” valósítják meg. Az egyes csomagokat, a minden csomagban közös szinkronizációs és csomagvég részek elhagyásával a következőkben ismertetjük.

Adatcsomagok

- Az IN, OUT vagy SETUP token csomagokkal kezdeményezett adatátvitelt a DATA0, DATA1, DATA2 és MDATA csomagok végzik.



- A küldeménytartalom 0..1023 bájt lehet, amit a 16 bites CRC követ. A hibafelismerés előmozdítására többféle adatcsomag létezik. Az adó általában a DATA0 és DATA1 csomagokat felváltva küldi. Így, ha a vevő például két ADAT0 csomagot vesz közvetlenül egymás után, akkor ez azt jelenti, hogy elveszett egy DATA1 csomag, ami hibára utal.
- A DATA2 és MDATA csomagokat csak az azonos idejű átvitelé (isochronous) használja, ezekre nem térünk ki.

Kézfogásos csomagok

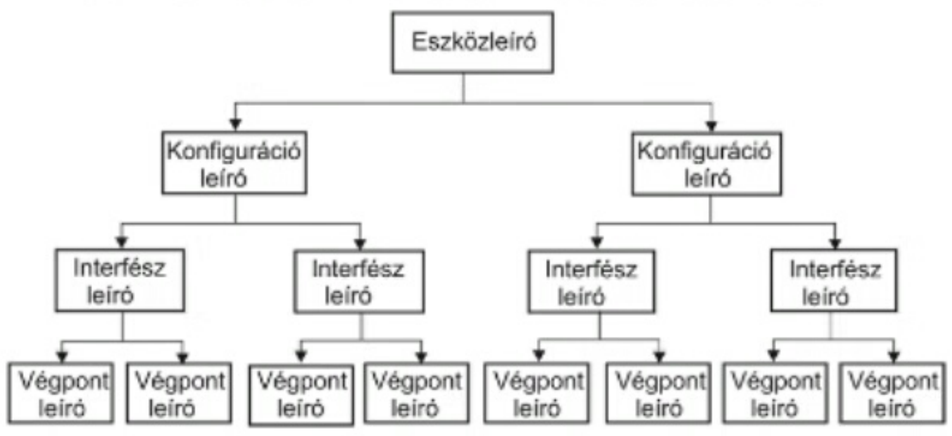
- A kézfogásos csomagokat a vevő küldi el az adó felé.

- Ezekkel jelzi, hogy a vétel milyen státusszal fejeződött be. Fajtái az ACK, NAK, NYET és STALL csomagok. ezek a csomagok csak a PID mezőt tartalmazzák, nincs CRC mezejük. A PID redundanciája teszi lehetővé a hibafelismerést.
- Az egyes csomagok jelentése a következő:
  - Az ACK a token és/vagy adatcsomag helyes vételét nyugtázza.

USB leírók és a konfigurálás

- Egy eszközt csatlakoztatva az USB buszra, a rendszer szoftvernek fel kell ismerni a csatlakoztatott eszköz típusát, végpontjainak jellemzőit, stb., hogy a megfelelő eszközmeghajtót betölthesse, a tranzakciókat megfelelően ütemezhesse, stb.
- Ezt a folyamatot konfigurálásnak nevezzük, az angol nyelvű szakirodalom gyakran az enumeration (felsorolás) kifejezést is használja. Az eszközöknek a konfiguráció során beolvasott jellemzőit a 0-s végponton keresztül hozzáférhető ún. eszközleírók tartalmazzák.

Az eszközleírók hierarchikus rendszere



Eszközleíró

- Egy USB eszköznek csak egyetlen eszközleírója van. Talmazza a szükséges információt a 0-s végpontról (default kommunikációs cső), továbbá olyan általános információt, mint a termék típusa, gyártója, melyik USB változattal kompatíbilis, a 0-s végpont maximális csomagmérete, az egyes jellemzők string leíróinak indexe, stb. Megadja még az eszköz által támogatott konfigurációk számát.

Konfigurálás

- A host szoftver feladata az USB buszra csatlakoztatott összes eszköz detektálása és konfigurálása, melyre gyakran az enumeration (felsorolás) kifejezést használjuk. Mi azonban a továbbiakban ehelyett is a konfigurálást használjuk.
- A konfigurálás a gyökér hubon kezdődik. A szoftver törli (reset), majd engedélyezi az összes kaput. Ezután a hub megállapítja, hogy melyik kapujára milyen sebsségű eszköz van csatlakoztatva. A csatlakoztatott eszközök státuszbitjeit aktív helyzetbe állítja. A szoftver az aktív státuszbiten keresztül felismerve egy eszköz csatlakoztatott állapotát, törli az USB eszközt, egyedi címet rendel hozzá, beolvassa a konfigurációs adatokat, betölti a megfelelő eszközmeghajtókat, s befejezi a konfigurálást. S ezt a gyökér hub minden kapujára elvégzi.
- A különböző operációs rendszerek különböző szoftver komponenseket definiálnak az USB eszközök konfigurálására, s a konfigurálást egyedi szekvenciában hajthatják végre. A következőkben konkrét operációs rendszerre vonatkoztatás nélkül, a konfigurálás elsődleges lépéseinek sorozatát adjuk meg. Ezek a lépések a gyökér hub egy kapujára a következők(a következő műveletek vezérlés tranzakción keresztül valósulnak meg. A szoftver az egyes eszközök 0-s végpontjával kommunikálva olvassa be a leírókat és konfigurálja az eszközt):
  1. A host utasítja a hubot kapuinak feszültség alá helyezésére, ha ez még nem történt meg.
  2. Ha a hub detektálja egy eszköz csatlakoztatását, s annak sebességét, akkor beállítja a csatlakoztatási státusz bitet.
  3. A host lekérdezi a hubot és értesül egy eszköz csatlakoztatásáról.
  4. A host kiad egy törlést a kapun keresztül a csatlakoztatott eszköznek (min. 10ms). A nagysebességű üzemmód azonosítása ezalatt történik (csirip).
  5. A host ismét megvizsgálja a hub státuszát, ha az eredetileg teljes sebességű üzemmódot jelzett. Ekkor eldől, nem-e nagysebességű a csatlakoztatott eszköz.
  6. Az USB eszköz ekkor a kiindulási 0-s című.
  7. A host GetDescriptor kérelmekkel beolvassa a szabványos leírókat. A beolvasott leírókat elemzi (parsing), hogy megállapítsa az eszköz jellemzőit.
  8. A host a kiindulási cím helyett egy egyedi címet rendel hozzá az eszközhöz.
  9. A host ellenőrzi, hogy az eszköz kiszolgálásához szükséges erőforrások rendelkezésre állnak-e.
  10. A host egy konfigurációs értéket küld az eszköznek, ez meghatározza, hogyan melyik konfigurációban használja majd a rendszer. Ekkor az eszköz az adott konfigurációban szereplő leírók tartalma szerint üzemelhet és kész a kliens szoftver kérelmeinek fogadására és a konfigurációban megadott áram fogyasztására.

Az USB 3.0 specifikáció

- Átviteli sebesség
  - SuperSpeed (SS) 5 Gb/s
- Megnövelt sávszélesség
  - Full duplex adatátvitel (adás és vétel külön érpáron)
- Megnövelt meghajtóképesség
  - 900 mA
- Hatékonyabb buszkihasználás
  - NRDY és ERDY csomagok
  - Készenlét aszinkron jelzése (nincs polling)
  - Host kérelem – végpont elfogadás/visszautasítás
  - Elfogadás esetén adatküldés vagy fogadás
  - Visszautasítás esetén (NRDY) ha kész a végpont, ERDY csomaggal jelez
- Adatkódolás
  - SATA-hoz, PCIe-hez hasonló
  - 8b/10b kódolás
  - Összekeverés (scrambling)
- Közvetlen kábelhossz nem definiált, csak az elektromos paraméterek
  - Gyakorlatilag 5m (USB 2.0) helyett csak 3m
- Lefelé kompatibilitás a 2.0 specifikációval
  - 2.0-ás és 3.0-ás A típusú csatlakozók kompatibilisek
  - 3.0-ás B típus a 2.0-ás módosított változata
- Forgó média támogatás
  - Tömegadatátvitel kiegészítve Stream protokollal
- Teljesítmény menedzsment (U0…U3)

PCI
Általános jellemzők. Mechanikai jellemzők. Elektromos jellemzők: CMOS jelleg, visszavert hullámra kapcsolás. Logikai jellemzők: jelek és funkciójuk, buszparancsok, memória-, IO-, és konfigurációs címtartomány, hierarchikus buszrendszer, olvasási- és írási tranzakció idődiagramja, báziscímregiszterek és a báziscímek kiosztásának mechanizmusa.
PCI Express
Rendszerarchitektúra és a kapcsolók felépítése. A fizikai réteg adójának logikai rendszere: blokkvázlat és részfunkciók (pufferelés, karakterbeszúrás, bájtszétszedés, összekeverés, kódolás és párhuzamos-soros átalakítás). Vevőalrendszer funkciók:órajelkinyerés, soros-párhuzamos átalakítás, dekódolás, elasztikus pufferelés. A villamos réteg: karakter-egymásrahatás (alul- és túlhangsúlyozás). Csomag alapú réteges protokoll: tranzakciós csomagok.

A PCI busz

Bevezetés

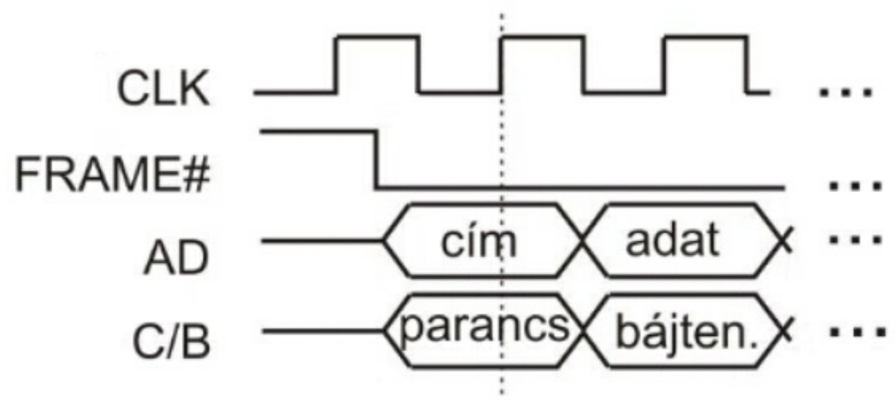
– A PCI busz az egymást követő változatok során sebességében és szolgáltatásaiban bővült, A fontosabb jellemzők és szolgáltatások (ezek egy része csak adott változattól felfelé áll rendelkezésre):
  - Processzor függetlenség
  - 256 funkcionális eszköz kiszolgálása buszonként
  - Hierarchikus rendszerben 256 PCI busz működhet együtt
  - Kis fogyasztás
  - Támogatott blokkátvitel
  - 33MHz vagy 66MHz maximális frekvencia
  - 32 vagy 64 bites adatbusz
  - Blokkátvitelkor az írási és olvasási idő közelít az 1 órajelhez, nem blokkátvitel esetén az írási idő 2, olvasási idő 3 órajel
  - Konkurens buszműködés
  - Buszmester támogatás
  - Rejtett busz arbitráció
  - Tranzakció-integritás ellenőrzés
  - Három címtartomány: memória (4GB, opcionálisan 64GB), I/O (64kB, opcionálisan 4GB), konfigurációs
  - Szoftver átlátszóság
  - Különböző méretű bővítőkártyák

Elektromos jellemzők

– Két fontos tulajdonság
  - A PCI busznak két olyan sajátossága van, amelyek döntően meghatározzák elektromos jellemzőit, s a buszon használandó adó/vevőáramkörök karakterisztikáit. Egyrészt a PCI busz egy CMOS busz, másrészt ún. visszavert hullámra kapcsolt (reflected vawe switching) busz. Mindkét jellemző domináns a fogyasztás csökkentése, s az ún. „zöld” architektúra szempontjából.
  - A fogyasztás csökkenése egyrészt abból adódik, hogy a CMOS-on alapuló buszon az állandósult állapotban (miután a kapcsolási tranziens lecsengett) az áram minimális, a meghajtó áramokat lényegében csak a felhúzó ellenállások „fogyasztják el”.
  - A fogyasztás csökkenése másrészt a visszavert hullámra kapcsolásból, a lezáratlanságból következik.

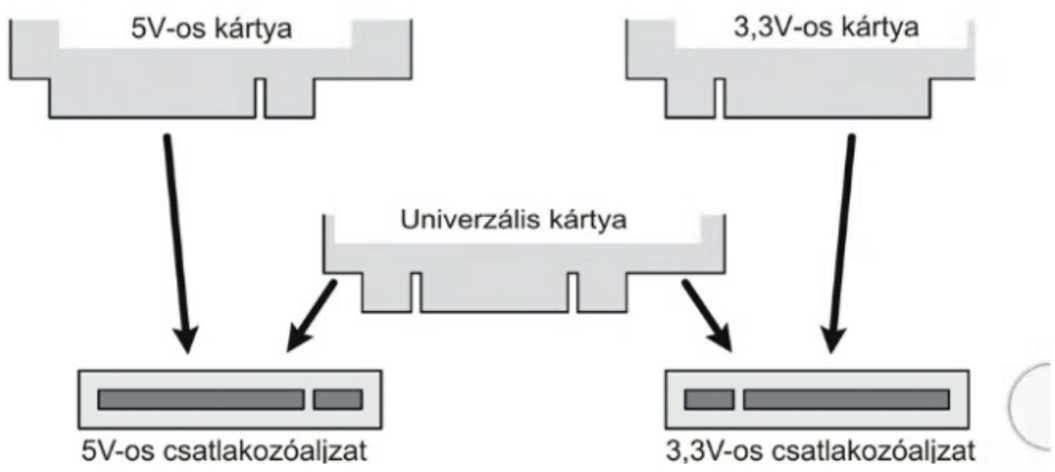
Buszparancsok

– Az initiator a buszparanccsal közli a target eszközzel, hogy milyen típusú tranzakciót hajt végre. A tranzakció első fázisa a címfázis, amelynek kezdetét a FRAME# jel aktív állapotba váltása jelzi. Ekkor a multiplexelt AD vonalak a target eszköz címét, a multiplexelt C/BE vonalak pedig a kódolt parancsot tartalmazzák.



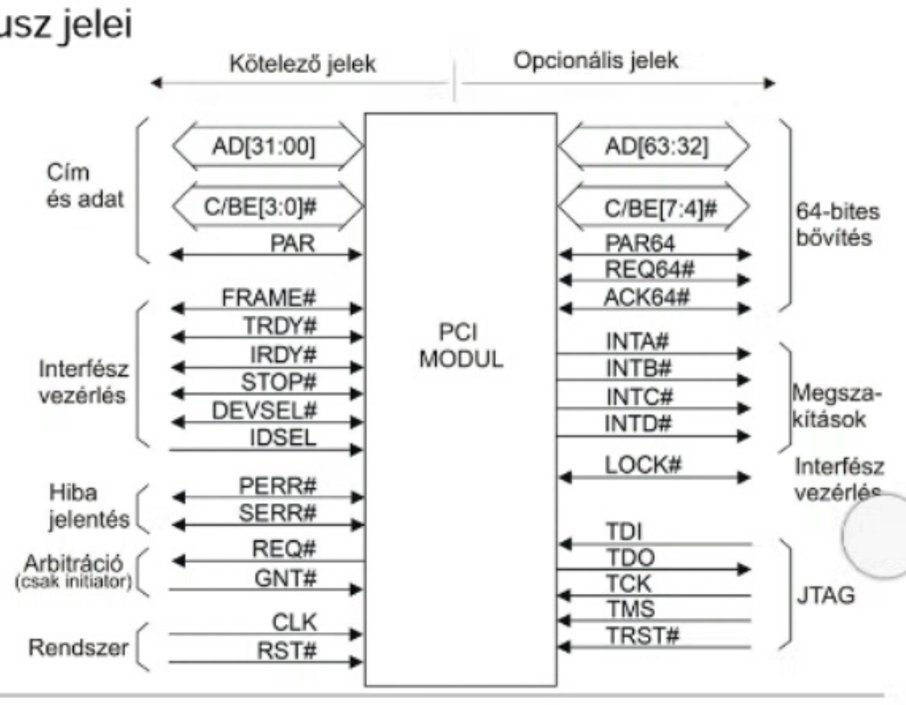
Mechanikai jellemzők

– Szabványos hosszúságú és rövid kártyák
– 5V, 3,3V vagy mindkettővel üzemeltethető kártyák
– 32 bites rendszer 120, a 64 bites rendszer 184 érintkezővel
– A kártyák alaplapra merőlegesek, vagy azzal párhuzamosak lehetnek (kiemelő kártya)



Logikai jellemzők

A PCI busz jelei



Jelölések:

– **in** Szabványos bemeneti jel.
– **out** Totem Pole kimenet, szabványos aktív meghajtó.
– **t/s** Háromállapotú, kétirányú be-kimeneti jel.
– **s/t/s** Fenntartott háromállapotú, aktív nullás jel. Amely eszköz ezt alacsony szinten meghajtja, annak nagyimpedanciás állapotba való vezérlése előtt legalább egy órajelig magas szinten kell tartania. Az új eszköz pedig csak a nagyimpedanciás állapotba vezérlést követően legalább egy órajel elteltével hajthatja csak meg ezt a vonalat. Ennek a vonalnak a felhúzása a központi erőforrás részéről kötelező az inaktív állapot fenntartása céljából.
– **o/d** Nyitott drain típusú jel, amely megengedi a huzalozott VAGY kapcsolat létrehozását. Az inaktív állapotot a központi erőforrásnak felhúzással tartja fenn

IO címtartomány

A 32 címbit mindegyike részt vesz a címzésben, s együttesen egy bájtot címeznek meg. A bájt engedélyezés bitek pedig csak bizonyos megszorításokkal választhatók meg szabadon. Nevezetesen a megcímzett bájt engedélyezése a duplaszavas struktúrában aktív kell legyen (ez 0 értéket jelent), az alatta lévő bájtokat tiltani kell, a felette lévő bájtok engedélyezhetők és tilthatók. Ha minden bájt tiltva van (1111), akkor AD[1::0] bármilyen értéket felvehet.

AD[1::0]	Kezdő bájt	Érvényes BE[3::0]#
00	0. bájt	xxx0 vagy 1111
01	1. bájt	xx01 vagy 1111
10	2. bájt	x011 vagy 1111
11	3. bájt	0111 vagy 1111

Memória címtartomány

– Ebben az esetben mindig duplaszót címez a mester. Tehát AD[1::0] nem vesz részt a címdekódolásban, viszont azt határozza meg, hogy milyen címsorrendben történjen az adatátvitel. Nevezetesen:

AD[1::0]	Címsorrend
00	Lineárisan növekvő
01	Fenntartva
10	Cache blokk visszacsatoltan
11	Fenntartva

– Lineárisan növekvő esetben minden adatfázis után 32 bites rendszerben 4-gyel (egy duplaszó címtartománya), 64 bites rendszerben pedig 8-cal (két duplaszó címtartománya) növele a teljes 32 bites cím (ami nyilván nem érinti a címsorrendet meghatározó alsó két címbitet).

– Cache blokk visszacsatoltan esetén az adatátvitel lineárisan a cache blokk végéig, azután a cache blokk elején folytatódik a cache blokk eddig nem írt részén. Ha a teljes cache blokk hozzáférése megtörtént, de a tranzakció még nem fejeződött be, akkor a következő cache blokkban ugyanolyan relatív pozícióban folytatódik, mint amilyen relatív pozícióban a megelőző cache blokkban elkezdődött.

Konfigurációs címtartomány

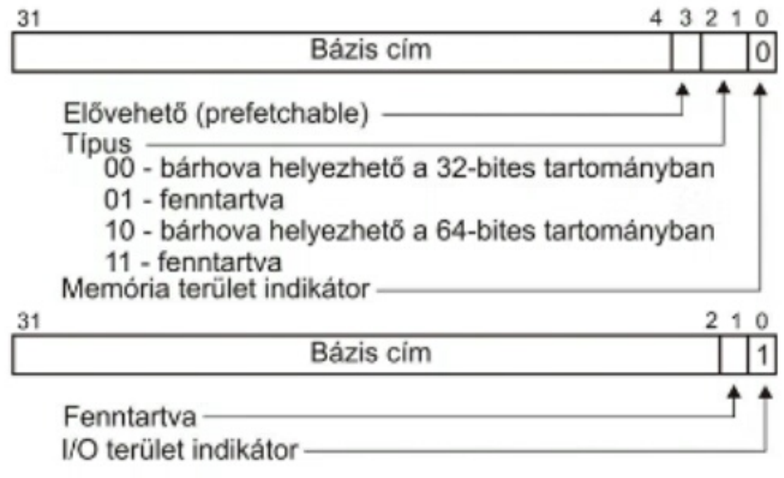
– A konfigurációs címtartomány egyrészt az adott eszközre vonatkozó információkat tartalmazza, másrészt az operációs rendszer bizonyos adatokat ír bele. Célja, hogy a buszra csatlakozó eszközök összehangolt működése biztosítható legyen a működésüket szabályozó paraméterek megfelelő beállításával (például a báziscímek központi kiosztásával az eszköz címtartományok átfedésének kiküszöbölése, a megfelelő meghajtó szoftverek betöltése, stb.).

– Bázis címregiszterek.

– Ezek a regiszterek lehetővé teszik a konfigurációs szoftver számára, hogy meghatározhassa az egyes eszközök által igényelt memória és I/O erőforrásokat.

– Ezt beolvasva a konfigurációs szoftver olyan kezdőcímeket rendel hozzá az egyes eszközökhöz, hogy azok címtartományai ne fedhessék át egymást (amit hagyományosan mikrokapcsolókkal, manuálisan kellett elvégezni), s ezeket a kezdőcímeket beírja az eszközök bázis címregisztereibe.

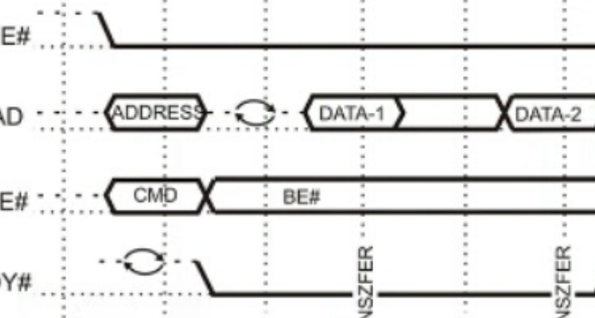
– A 0 típusú fejléc hat bázis címregisztert támogat, így egy eszköznek (pontosabban egy funkciónak) hat különböző címtartománya lehet.



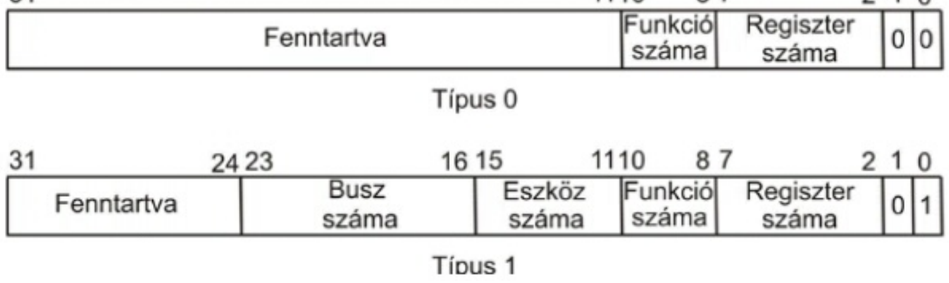
– Memória terület esetén a csak olvasható bit 2,1 azt specifikálja, hogy a 32 vagy a 64 bites tartományba képzendő-e le a cím, s ez közvetve a bázis címregiszter hosszára is utal. Korábbi változatokban a 01 érték az 1MB alatti tartományt specifikálta, a 2.2 változatban már ezt elhagyták. I/O terület esetén csak 32 bites címtartomány van. Az elővehetőség azt jelenti, hogy mellékhatások nélkül olvasható a memória, azaz annak olvasása nem változtatja meg annak tartalmát

Konfigurációs parancsok

– Az egyetlen PCI buszra csatlakoztatható eszközök kis száma kicsi. Ezt a korlátot a PCI rendszer a hierarchikus rendszerben összekapcsolt PCI buszokkal (szegmensekkel) szünteti meg. Az egyes szegmenseket ún. PCI-PCI hidak kapcsolják össze.

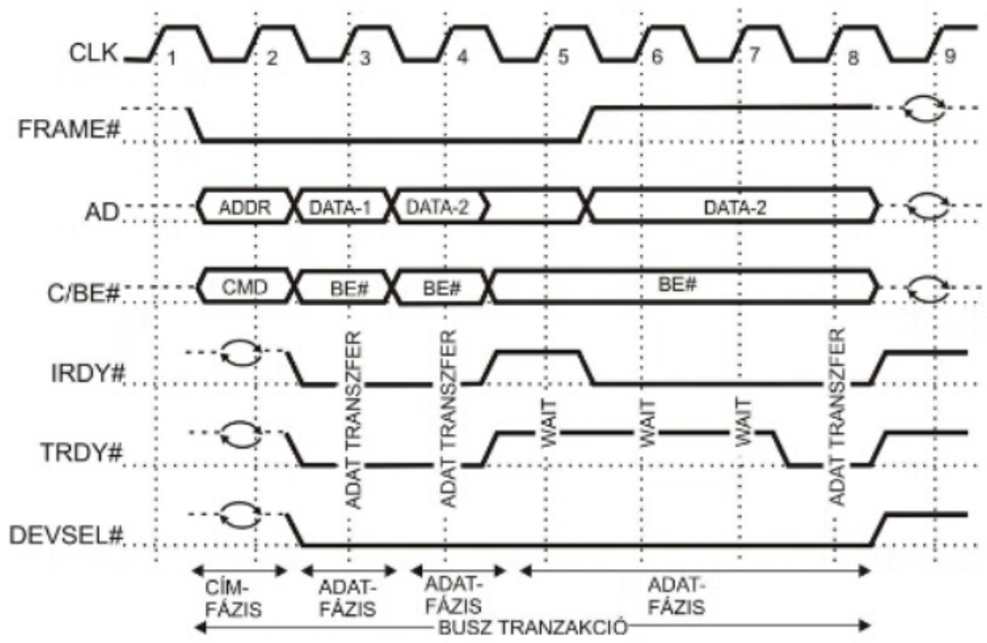


– Ebben a rendszerben egy PCI-PCI hídnak tudnia kell, hogy egy konfigurációs hozzáférést mikor kell továbbítania a híd mögötti eszközök felé. Ezért két konfigurációs tranzakciót (és egyben címtípust) használ a PCI rendszer



Olvasás tranzakció

Írás tarnzakció



– A báziscím kiosztása.

- A báziscím kiosztásához először meg kell határozni a szükséges memória vagy I/O tartomány nagyságát. Erre a következő mechanizmus szolgál. A konfigurációs szoftver csupa 1-est ír a bázis címregiszterbe, az eszköznek a szükséges blokkméretbe eső biteket 0-ba kell állítani, s ezt visszaolvasva a konfigurációs szoftver megkapja a szükséges blokkméretet. A minimális blokkméret 16 bájt (az alsó négy bit láttuk más jelentésű) a maximális 2Gbájt. A blokkméret csak 2 egész hatványai szerint változhat. A könnyebb megértést az alábbi példa segítheti:

Lépés	1 MB-os példa
1. Csupa 1-es beírása	0xFFFFFFFF
2. Visszaolvasás	0xFFF00008
3. Alsó négy csak olvasható bit maszkolása	0xFFF00000
4. 1-es komplemens képzése	0x000FFFFF
5. 1 hozzáadásával megvan a blokkméret.	0x00100000

- Ha a konfigurációs szoftver a fenti mechanizmussal megkapta az összes eszköz szükséges címtartományát, eldöntheti, hogy melyik eszköznek milyen kezdőcímet ad, amit beír az eszköz bázis címregiszterébe (a fenti esetben például 0x03400000). Figyelem: a kezdőcím mindig természetes módon hozzárendelt, azaz a blokkméret egész számú többszöröse.

Rendszerarchitektúra

– A PCI Express egy nagysebességű, pont-pont típusú, skálázható, soros busz. Egy PCI Express kapcsolat (link) két szimplex csatornából áll, egyik az adás, másik a vétel számára. Ezek a csatornák kisfeszültségű, differenciális jelátvitelt használnak két-két vezetékkel. Az átvitt soros információ 8b/10b kódolású. Az órajel az átvitt információból nyerhető ki.

– Szigorúan véve a PCI Express nem egy busz. A busz egy olyan adatút, amelyhez egyidejűleg több eszköz csatlakozhat, s azt megosztva használhatják. A PCI Express egy pon-pont típusú kapcsolat, azaz két eszközt kapcsol össze, s más eszköz nem osztozhat ezen a kapcsolaton. De mivel a PCI és PCI-X tényleges buszok következő generációja, ezért a szakirodalommal összhangban mi is busznak nevezzük.

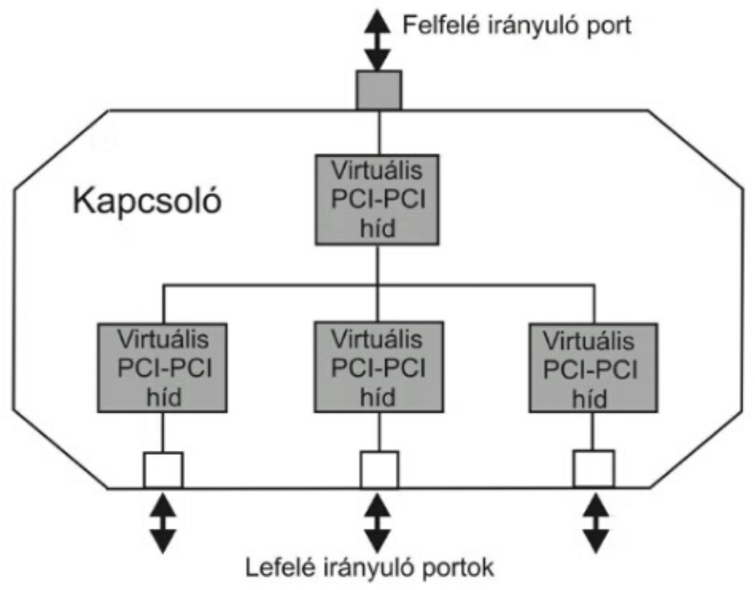
– A PCI Express sávszélessége skálázhatóan növelhető újabb adó-vevő kapcsolat ún. lane (ösvény) hozzáadásával a két eszköz között.

– A specifikáció x1, x4, x8 és x16 lane alkalmazását teszi lehetővé.

– Az alap x1 ösvényt tartalmazó kapcsolat sávszélessége 2,5Gb/s. Mivel a kapcsolat kétirányú, ezért az adatátviteli sebesség 5Gb/s.

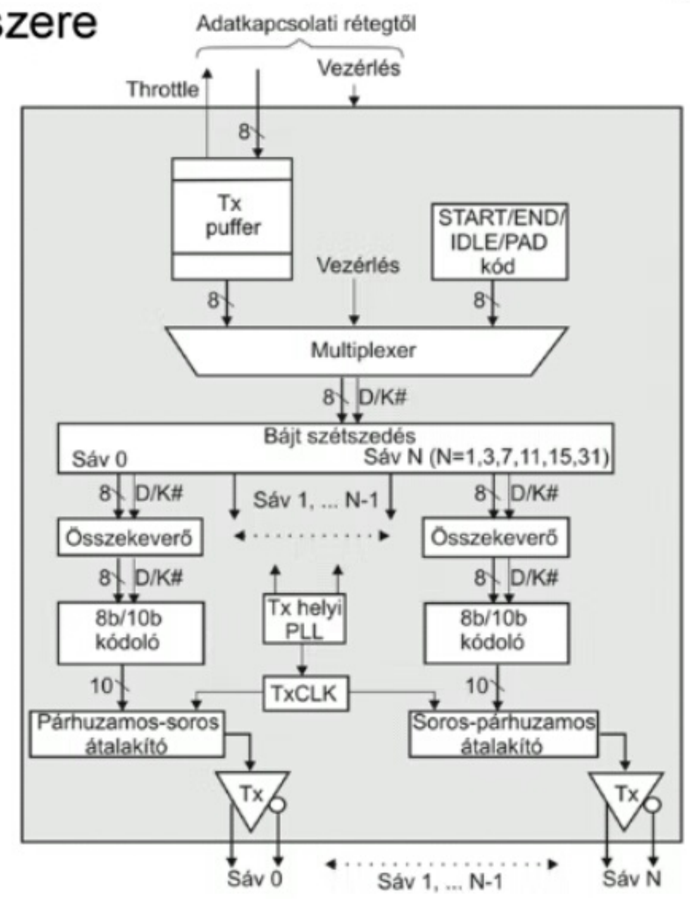
– A 8b/10b kódolás miatt azonban egy bájt átviteléhez 10 bit szükséges, így az időegység alatt ténylegesen átvitt bájtok száma 500MB/s. Az átvitt hasznos sávszélesség ennél is kisebb, mert az adatok becsomagoltan, hibajavító kódolással kerülnek átvitelére, s a csomagok különböző vezérlő karaktereket is tartalmaznak.

A kapcsoló felépítése



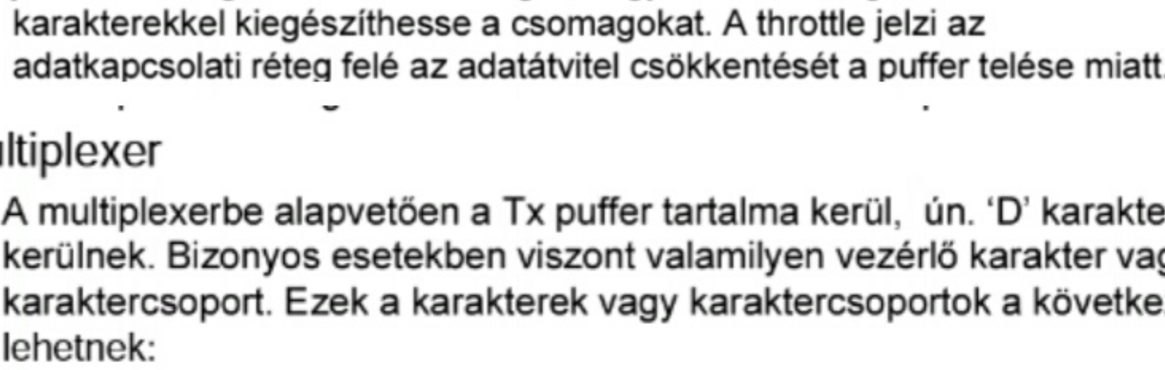
ADG

logikai alrendszere



– Tx puffer
  - A Tx puffer a átveszi a tranzakciós réteg és az adatkapcsolati réteg csomagjait. A csomagokkal együtt az adatkapcsolati réteg a „vezérlő" jellel jelzi a csomagok kezdetét és végét, hogy a fizikai réteg a START és END karakterekkel kiegészíthesse a csomagokat. A throttle jelzi az adatkapcsolati réteg felé az adatátvitel csökkentését a puffer telése miatt.

– Multiplexer
  - A multiplexerbe alapvetően a Tx puffer tartalma kerül, ún. 'D' karakterek kerülnek. Bizonyos esetekben viszont valamilyen vezérlő karakter vagy karaktercsoport. Ezek a karakterek vagy karaktercsoportok a következők lehetnek:
    – START és END kontroll karakterek. Más nevükön 'K' karaktereket a multiplexer minden tranzakciós csomag (TLP) és adatkapcsolati csomag (DLLP) elé, illetve mögé beszúrja. Ezek a „keretező" karakterek megkönnyítik a vevő számára a csomaghatárok felismerését. Kétféle START és kétféle END karaktert használ a rendszer. Ezek a következők STP (tranzakciós csomag kezdete), SDP (adatkapcsolati csomag kezdete), END (hibátlan csomag vége) és EDB (hibás csomag vége).



– Rendezett csoportok (ordered-set). Nx4 karakterből állhatnak, s mindig a comma (COM) vezérlő karakterrel kezdődnek, s típustól függően D vagy K karaktereket tartalmazhatnak. Speciális eseményeket közvetítenek. Ezek: átviteli csatorna tréning, eliminálható rendezett csoport, elektromos alvó állapot kiváltása, teljesítményszint változtatása.

– Logikai nyugalmi állapot. Ha nincs elküldendő csomag (logikai nyugalmi állapot) akkor ún. Idle karaktereket kapuz be a multiplexer. Így a vevő PLL-je fenn tudja tartani a szinkronizálást.

– Bájt szétszedés
  - Ha egy kapu több sávot valósít meg, akkor a bájt szétszedő logika a csomagban lévő bájtokat sorban egymás után az egymást követő sávoknak adja át. Példaként egy négysávos rendszert a következő ábra szemléltet.

Összekeverő

– Az összekeverés célja, hogy kiküszöbölje az ismétlődő mintákat az adatfolyamban. Az ismétlődő minták ugyanis diszkrét frekvenciákon viszonylag nagy energiát képviselnek, s ezek egyrészt jelentős elektromágneses zavarokat (EMI) okozhatnak, másrészt többsávos rendszerekben az egyes sávok közötti áthallás is komoly gondot okozhatna. Az összekeverés eredményeként a 0-k kvázi-véletlen eloszlásúak lesznek az adatfolyamban, diszkrét energiákon nem jelentkezik tetemesebb energia, mintegy „kifehéredik" a zaj.

– Az összekeverés például egy lineáris visszacsatolt léptető regiszterrel előállított bitsorozattal való összekeveréssel valósítható meg, melynek a karakterisztikus polinomja:

– $G(x) = x^{16} + x^5 + x^4 + x^3 + 1$

– A szabvány részletesen specifikálja, hogy mely karaktereket kell és melyeket nem szabad. összekeverni. Az adatkaraktereket kell, a 'K' karaktereket viszont nem szabad. Emellett még egyéb előírások is betartandók ezek részleteire azonban nem térünk ki.

8b/10b kódolás

– Ennek a kódolásnak a mechanizmusát és előnyeit a SATA kapcsán már tárgyaltuk, itt nem foglalkozunk vele.

Párhuzamos-soros átalakító

– Minden sávban a 8b/10b kódolótól érkező 10 bites karaktereket 'abcdeifghj' sorrendben, a legkisebb helyértékű bittől kezdve bitsoros jelekké alakítja. A 8b/10b kódoló 250MHz frekvenciával adja át a karaktereket a párhuzamos-soros átalakítónak, amely ennek tízszeresével, azaz 2,5GHz frekvenciával (TxCLK órajellel) működik.

Órajel kinyerés

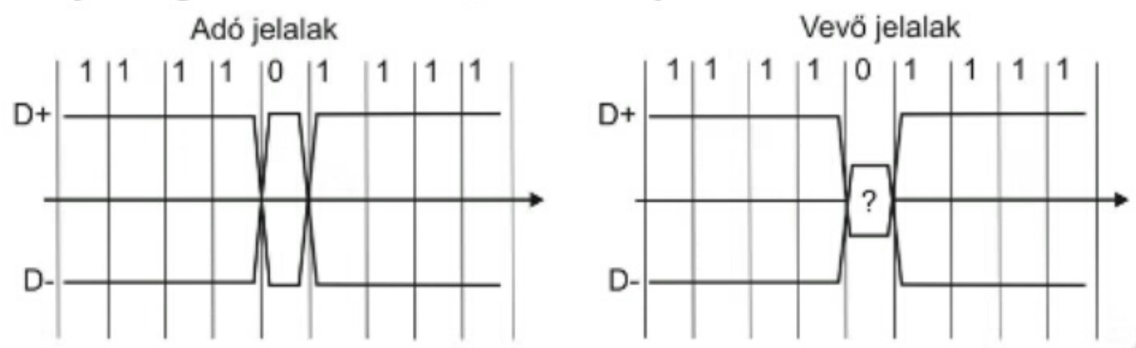
- A vevő bemeneti részében a vett soros jelből az RxCLK kinyerő PLL (PLL = Fáziszárt hurok) állítja elő a vevő órajelét (a 8b/10b kódolás biztosítja, hogy legalább minden 5-ik bit esetén változzon a jel).
- Amikor a vevő órajele szinkronba kerül az adó órajelével, akkor azt mondjuk, hogy megtörtén a bit-zárás. Ez szükséges ahhoz, hogy a bemenet értékei a helyes időpontokban legyenek beléptetve a soros-párhuzamos átalakítóba.
- Az adó a csatorna tréningje során hosszú TS1 és TS2 rendezett csoportokat küld a vevőnek, amely ezt használva éri el a bit-zárást.

Soros-párhuzamos átalakítás

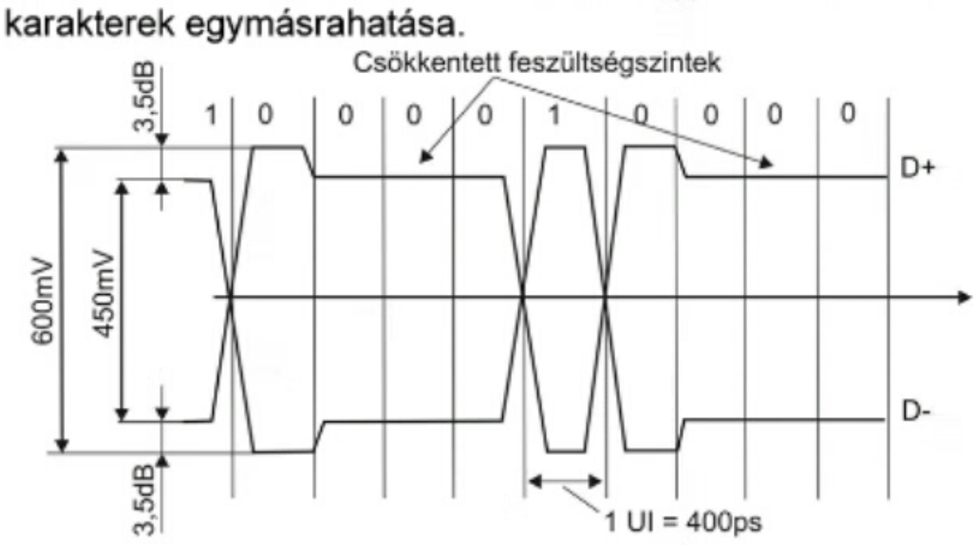
- Bit-zárás esetén helyes bitek lépnek be a soros-párhuzamos átalakítóba, azonban még a karakterhatár felismerésére is szükség van, azaz hogy mikor kell a soros-párhuzamos átalakító (ez egy soros léptető regiszter párhuzamos kimenetekkel) kimenetét átírni az elasztikus pufferbe. A vevőnek tehát fel kell ismernie a 10-bites karakterhatárokat.
- A karakterhatárok felismerését a COM (K28.5) karakterek teszik lehetővé, melyek nem mennek át összekeverésen. Ebben a karakterben két azonos bitet öt ellentétes bit követ (0011111010b vagy 1100000101b). Ha hiba nem lép fel, akkor egyetlen más karakter sem rendelkezik ezzel a tulajdonsággal, ami megkönnyíti detektálását. Ha a COM detektor érzékel egy COM karaktert, akkor a COM karaktert közvető bit biztosan egy 10 bites karakter első bitje. Ekkor inicializálja a soros párhuzamos átalakítót, szakkifejezéssel megtörtént a karakter-zárás.

Karakter egymásrahatás (inter-symbol interferencia)

- Minél nagyobb frekvencián üzemel egy adatátviteli csatorna, annál kisebb az 1 bit átvitelére szolgáló idő. 2,5Gb/s esetén ez a bitidő, az ún. Unit Interval (UI) 400ps. A csatlakozók, a hozzávezetések és a csatorna viszont bizonyos kapacitással rendelkezik, s ezen kapacitás eredőjének a töltését kell rövid idő alatt az ellentétesre változtatni, ha az átviendő jel ellentétesre változik. Ha hosszabban nem változik az átviendő jel, akkor nyilván jobban feltöltődik az eredő kapacitás, s kevésbé lesz sikeres a kisütése a jel megváltozása esetén, ami hibás jel vételéhez vezethet.



- A probléma megoldására az *alulhangsúlyozást* és a *túlhangsúlyozást* használják.
- Alulhangsúlyozás. Ha több azonos bit követi egymást az átvitelben, akkor ez első bit kivételével a többi bitet csökkentett feszültséggel kapcsolja a csatornára az adó. Ezzel csökkenteni lehet az eredő kapacitás feltöltődését, s így csökkenthető a karakterek egymásrahatása.



Elasztikus puffer

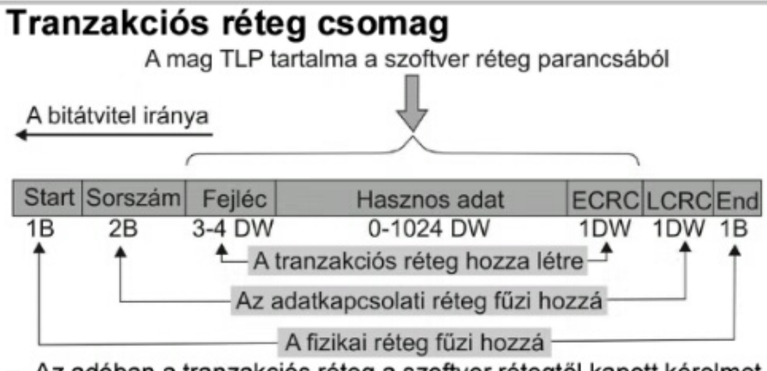
- A vevő logikai egységeinek nagy része helyi órajellel működik. Ennek frekvenciája nyilván nem egyezik meg pontosan az adó órajelének frekvenciájával. A szabvány a frekvencia pontosságára a következőt írja elő: 2,5GHz ± 300ppm. Ez azt jelenti, ha az adó órajele +300ppm, a vevő órajele pedig -300ppm értékkel tér el a névlegestől, akkor minden 1666-ik órajelre egy órajel elcsúszás lép fel.

- A vett karakterek a bemeneti jelsorozatból kinyert órajel ütemének megfelelően íródnak be az elasztikus pufferbe, onnan viszont a helyi órajel ütemében olvasódnak ki, s íródnak be a 8b/10b dekódolóba.
- Az elasztikus puffer az adó felől érkező SKP karakterek törlésével vagy beszúrásával kompenzálja a két órajel frekvenciájának eltérését. A SKP karakterek az ún. eliminálható rendezett csoportok részei, amely csoport egy COM karakterre indul, s azt három SKIP karakter követi. Az adó periodikusan (minden 1180..1538 karakter után) küld ilyen eliminálható rendezett csoportokat a vevő felé. A SKP karakterek hasznos információt nem hordoznak, a felsőbb szinteken a csomagok összeállításánál már figyelmen kívül maradnak.

8b/10b dekódoló

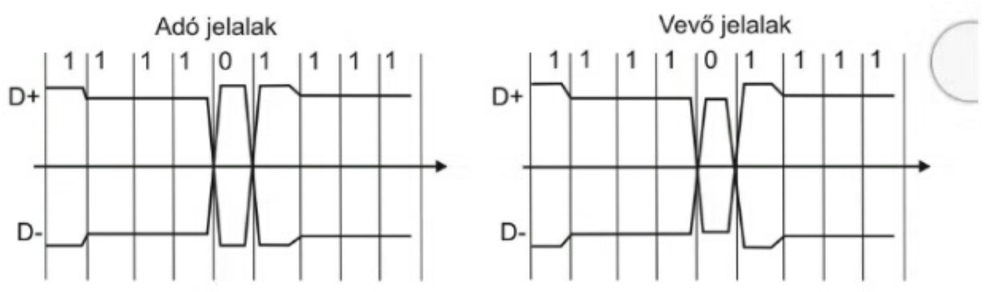
- A 10 bites, 8b/10b kódú karaktereket visszaalakítja 8b kódjukba, azaz a 8b/10b kódoló inverz funkcióját valósítja meg. Ez az egység párhuzamosan hibadetektálást is végez.

Tranzakciós réteg csomag



- Az adóban a tranzakciós réteg a szoftver rétegtől kapott kérelmet egy tranzakciós réteg csomaggá (TLP = Transaction Layer Packet) alakítja át. Ebben a rétegben a TLP egy fejlécből, az azt követő hasznos adatokból és opcionálisan egy ún. End-toEnd CRC (ECRC) hibajavító kódból áll. Ezt a csomagot a lejjebb lévő rétegek járulékos információval egészítik ki. Nevezetesen az adatkapcsolati réteg elöl sorszámot, hátul pedig hibajavító kódot fűz a TLP csomaghoz, melyet végül a fizikai réteg Start és End karakterekkel lát el, keretez be. A fizikai csatornára tehát már a fenti ábrán látható csomag kerül ki (az ábrán a B betű bájtot, DW pedig duplaszót jelöl).
- A vevőben a fordított folyamat megy végbe.
- Megjegyzés: A szoftver réteget az angol nyelvű szakirodalom device core-nak (eszköz magnak) is nevezi.

- Túlhangsúlyozás. Ebben az esetben az azonos bitek közül az elsőt megnövelt feszültséggel kapcsolja a csatornára az adó. Ez segít az ellentétesre feltöltött kapacitás kisütésében.



Számítógép-architektúrák

Az utasításkészlet architektúra, a mikroarchitektúra és a számítógép-architektúra definíciója. A jó architektúra 8 jellemzője és rövid leírásuk. A koncepcionális szakadék. Függvényhívás megvalósítása HLL támogatás nélkül és támogatással. Az ENTER n,m utasítás leírása. A fordítás és interpretálás összehasonlítása. A BASIC interpreter működése. A szoftver kompatibilitás és portabilitás fogalma. Az öt számítógép-generáció legfőbb jellemzője.

Utasításkészlet architektúra (Instruction Set Architecture = ISA)

- Az ISA a számítógép programozásához kapcsolódva tartalmazza az utasításokat, az adattípusokat, regisztereket, címzési módokat, a memória architektúrát, a megszakítások és kivételek, valamint a külső be-kivitel kezelését.
- Időről-időre új utasításokkal bővítik: pl. az IA-32 bővítése MMX, SSE, SSE2, SSE3, SSE4-vel
- HW és SW közötti interfésznek is tekinthető
- Mint specifikáció szolgál a mikroarchitektúra megtervezéséhez
- Elválasztja a statikus feladatokat (fordítási idő) a dinamikus feladatoktól (futási idő)

A jó architektúra jellemzői

- Konzisztencia (következetesség)
- Ortogonalitás
- Megfelelőség (propriety)
- Takarékosság (parsimony)
- Átlátszóság (transzparency)
- Általánosság (generality)
- Teljesség (completness)
- Nyitottság (open endness)

Mikroarchitektúra/Implementáció

- Az ISA-t a szilíciumlapkán megvalósító hardver működésének teljes mértékű leírása
- Tartalmazza a processzor egyes blokkjainak, közöttük a magok, végrehajtó egységek (mint pl. lebegőpontos- és egész típusúak), ugrás prediktorok, csővezetékek, gyorsítótárak, periféria-támogatás, stb. leírását.
- Gyakran, főleg tudományos körökben a mikroarchitektúrát számítógép szervezésnek nevezik

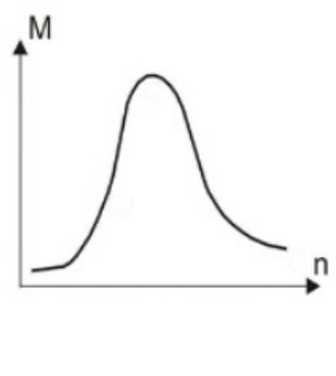
Számítógép architektúra

- Az utasításkészlet-architektúra és a mikroarchitektúra együttesen képezi a számítógép-architektúrát.
- Más definíciók
  - Amhdal (1964), IBM vezető tervezője szerint: a számítógép funkcionális megjelenése a közvetlen (assembly szintű) felhasználó felé. Ma ez lényegében az ISA.
  - Stone (1975): A komponesek halmaza és szervezésük. Ma ez lényegében a mikroarchitektúra vagy szervezés.
  - Németh (1991): a számógép architektúrája annak térbeli, időbeli és funkcionális elrendezése.
    - Modulok $M = \{m_i, i=1,2,\ldots N\}$
    - Csatolás $R = \{r_{ij}, i=1,2,\ldots N; i \neq j\}$
    - Architektúra $A = (M,R)$

Megfelelőség

A specifikált funkciók hozzátartoznak a rendszer alapvető, lényeges követelményeihez.

- Ellentéte a felslegesség (extraneousness), ami nem kell a lényegi cél eléréséhez.
- Pl.: Autó sebességváltó funkcionálisan felesleges, mégis miért van? Oka az M/n jelleggörbe. Automata sebességváltós autókban ki van küszöbölve



Megfelelőség (folytatás)

- MC68000 mikroprocesszor CLR (törlés) utasítása: ez először olvas, csak azután töröl.
  - Így egyszerűbb volt megcsinálni
  - Nemcsak felesleges, de csak írható regisztereknél hibát okozhat. Kijavították.
- 2-es komplemens kódban csak egyetlen NULLA van,1-es komplemens kódban +NULLA és a –NULLA is létezik.
  - A két nulla létezése felesleges összehasonlításokat követel, ami a matematikában nincs.

Konzisztencia (következetesség)

- Egy rész ismeretében következtethetünk a többi rész architekturális jellemzőire
  - Például, ha nem ismerjük, hogy a logaritmust számító utasítást hogyan hajtja végre a számítógép, konzisztens architektúra esetén feltehetjük, hogy a többi aritmetikai utasítás ugyanezt az adatábrázolási formát, kerekítési rendszert, pontosságot, kivételkezelést, jelzőbit-beállító funkciót, stb. használja.
  - Példa inkonziszenciára:
    MC6820 processzor
    ADDQ (add quick) közvetlen operanusa 3 bites
    MOVEQ (move quick) közvetlen operanusa 8 bites

Ortogonalitás

- Geometriában: egy koordináta változtatása nem változtatja meg a többi koordináta értékét
- Általánosan: egy jellemzőosztályon belüli változás nem hat más jellemzőosztály értékeire
- Itt: koncepcionálisan független funkciók specifikálása független legyen egymástól
  - Például MC6820
    MOVE mindkét operandusa megadható effektív címmel
    ADD csak egyik operandusa adható meg effektív címmel.
    A művelet és operandus specifikáció nem ortogonális.

Takarékosság

- A jó architektúra gazdaságos is. A rendszer szempontjából inkorrekt funkciókat nem tartalmaz.
- Például az MC68020 mikroprocesszorban a mikroprogram lehetővé tenné, hogy egy 32 bites lebegőpontos szám 1-es komplemensét is képezzük. De minek?
- A 60-as években hallottam: „Lehetne Svájcban szocializmust csinálni? Igen – de minek. Hiszen ott anélkül is jólét van. Volt a válasz"

Átlátszóság

Átlátszó a rendszer, ha az implementáció vagy realizáció szintjén meghozott döntés nem hat vissza az utasítás architektúra szintjére (Information Hiding).

- Súlyosan sértette az átlátszóságot a RISC processzorok kezdeti csővezeték implementációja. A csővezeték szervezés ugyanis megkövetelte az ún. késleltetett elágazás alkalmazását.
- Példa

```
ADD A,B
JMP   cím,Z
SUB   A,C    // ezt mindenképpen végrehajtotta a proc.
             // ezért a fordítóprogram beszúrt egy NOP-ot
ADD A,B
JMP cím,Z
NOP
SUB A,C      // eddigre már kiértékelődik az elágazás
```

Általánosság

– Egy funkció sokféle használatának lehetőségére utal.
– A tervező egy funkciót ne korlátozzon a saját használati tapasztalatai alapján. Számítania kell a sok kreatív felhasználóra.

Nyitottság

– Egy architektúra nyitott, ha például a jövőbeli fejlesztésekre fenntartott műveleti kód- és címterülettel rendelkezik.
– Más példa: HTML fix elemkészletű, XML bővíthető elemkészletű.

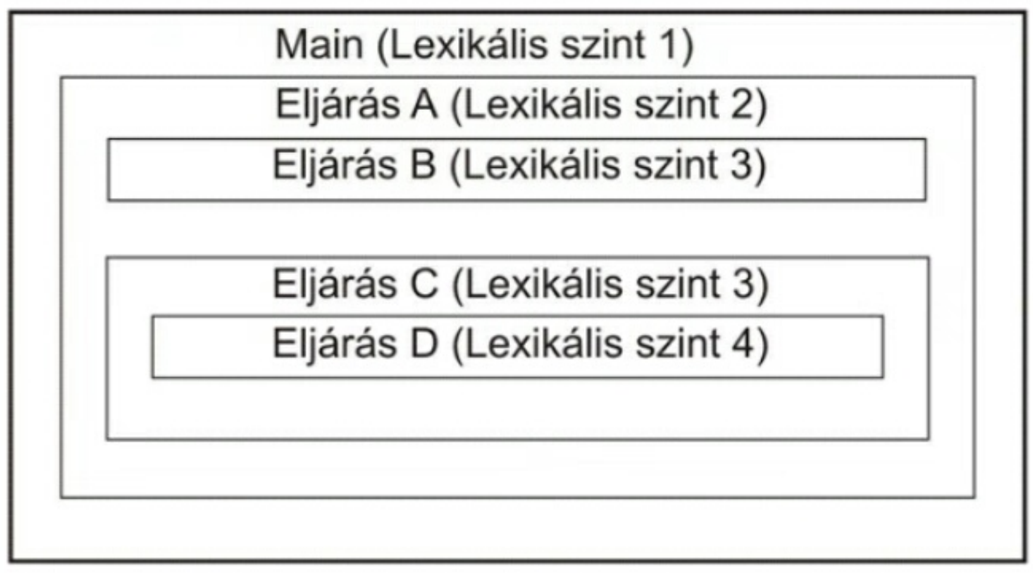
Teljesség

– Ha egy adott funkcióosztály összes funkciója jelen van az architektúrában, akkor az architektúra ezen funkcióosztály szempontjából teljes.
– Ha például az utasításkészlet tartalmaz szorzó utasítást, akkor a teljességhez tartalmaznia kell osztó utasítást is.
– Például az MC6809 volt az első 8 bites mikroprocesszor, amelynek utasításkészlete tartalmazott szorzást. De osztást nem tartalmazott, ezért utasításkészlet architektúrája nem volt teljes.

Az ENTER n,m utasítás

A a blokkstruktúrájú nyelvekben használt hatáskörszabályoknak megfelelő veremkeret-struktúrát hoz létre.

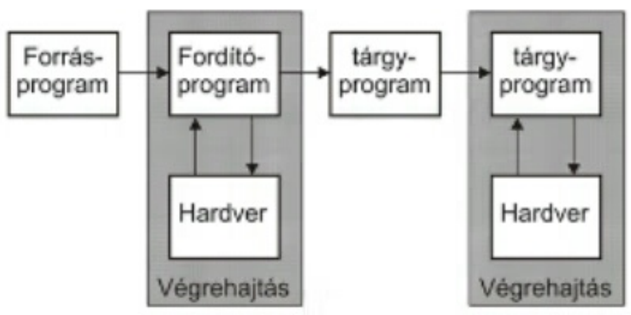
- n a lokális változóknak fenntartandó bájtok száma
- m a lexikális beágyazottság (hány veremkeretmutatót kell átmásolni a megelőző veremkeretekből



Szintek közötti viszony (Fordítás és interpretálás)

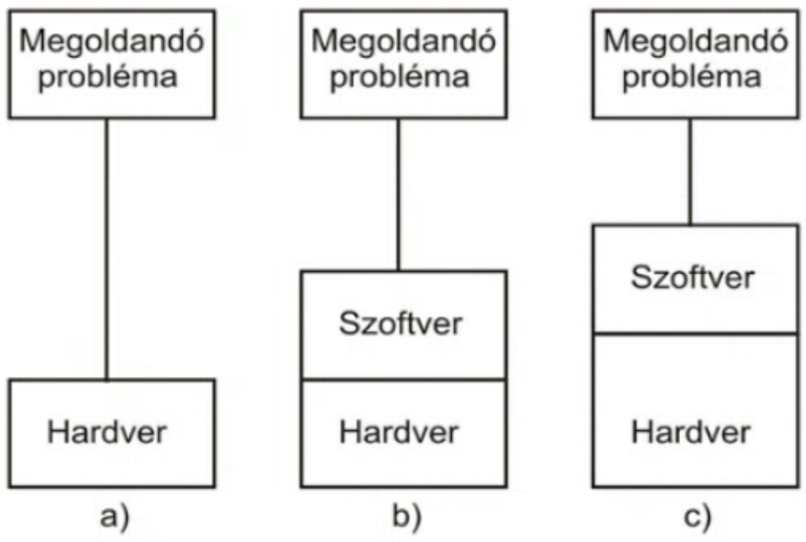
Fordítás

– Egyből tárgykódba vagy közbülső kódba további fordítás céljára
– az n-edik szint minden komponensét az (n-1)-ik szint komponenseinek egy csoportjára képezzük le
– A problémát az n-edik szinten reprezentáljuk, de a végrehajtás a (n-1)-ik vagy az alatti szinten megy végbe.
– Compiler = fordító ??? (összeállít, összeszerkeszt)



A fejlődés lépései

Az emberi gondolkodás absztrakt és nagyon távol áll a számítógép áramkörei által végrehajtott műveletek szintjétől, nagy az ún. koncepcionális szakadék vagy fogalmi távolság.



HLL támogatás nélkül

A hívás

```
public MOV AX,1
PUSH AX
MOV AX,2
PUSH AX
CALL f
```

A függvény

```
PUSH EBP     ; a hívó keretmutató mentése a verembe
MOV EBP,ESP  ; a hívott veremkeretének beállítása
SUB ESP,2    ; helyfoglalás a 2 bájt (egy szó)
             ; hosszúságú lokális változó számára
   ...       ; a függvény funkciójának végrehajtása
   ...               ;
MOV AX,…     ; visszatérési érték az AX-ben
MOV ESP,EBP  ; a lokális változók helyének
             ; felszabadítása
POP EBP      ; és a hívott program keretmutatójának
             ; visszaállítása
RET 4
```


HLL támogatással

A hívás

```
PUSH 1
PUSH 2
CALL f
```

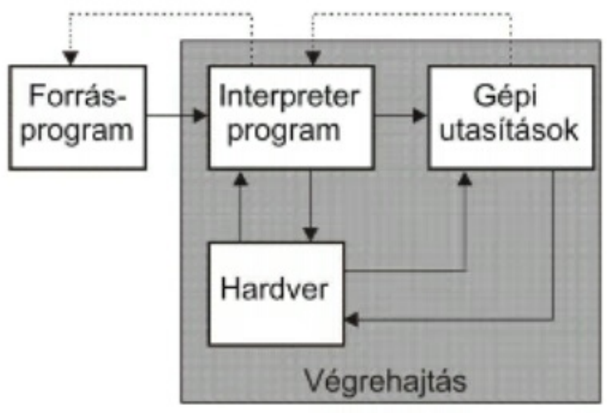
A hívott függvény

```
ENTER 2,0 ; a hívó keretmutatójának mentése a verembe
          ; a hívott keretének beállítása, helyfoglalás
          ; lokális változói számára (2 bájt)
...       ; a függvény funkciójának végrehajtása
...
LEAVE     ; a lokális változók helyének felszabadítása
          ; a hívó keretmutatójának visszaállítása
RET 4     ; visszatérés a hívás utáni helyre, 4 bájt
          ; felszabadítása a veremben
```


Szintek közötti viszony (Fordítás és interpretálás)

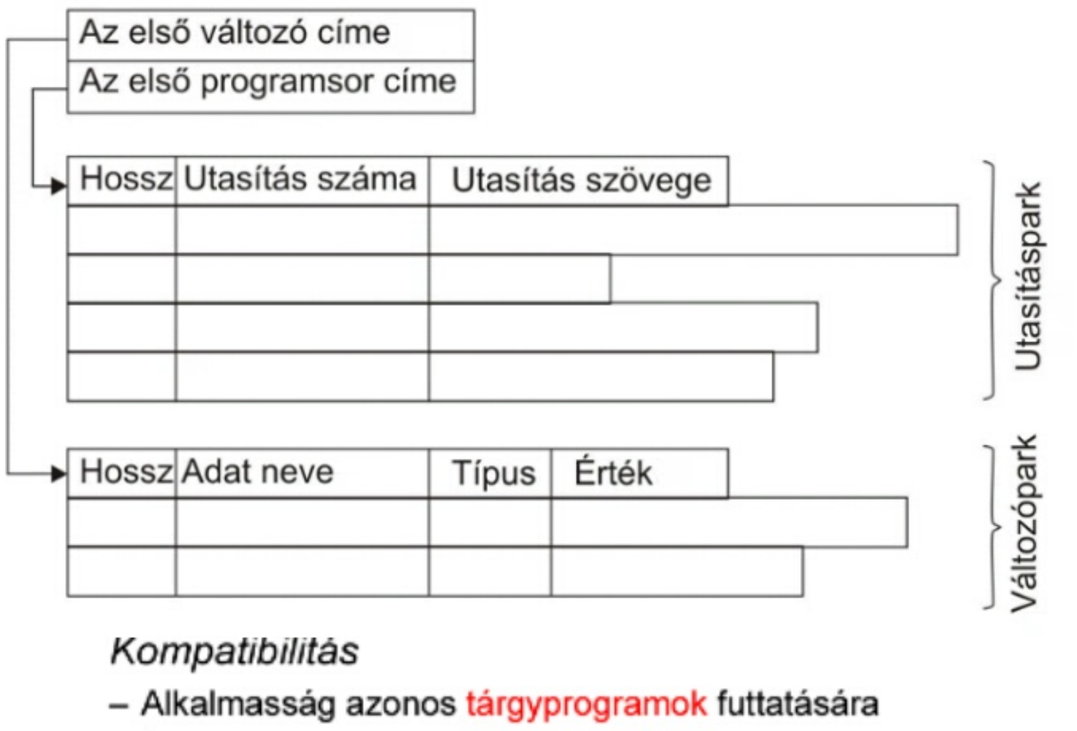
Interpretálás

– Az interpretálás (értelmezés) folyamata egybeesik a végrehajtás folyamatával, s így egyidejűleg jelen vannak az n-ik és az (n-1)-ik szint komponensei.
– Azaz a probléma reprezentálása és futtatása egyaránt az n-ik szinten megy végbe.



Szintek közötti viszony (Fordítás és interpretálás)

Példa: BASIC interpreter



Kompatibilitás

- Alkalmasság azonos tárgyprogramok futtatására
- Általában felfelé kompatibilitás: Pentium 4-en futhat a 486-os tárgyprogram, de fordítva nem (MMX, SSE miatt)

Szoftver portabilitás (hordozhatóság)

Az **A** gépre megírt program a **B** gépen is futtatható

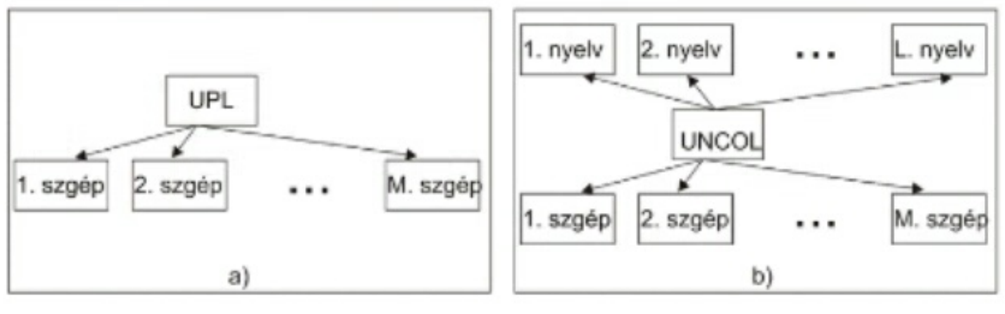
Többféle megoldás:

- Megoldás magasszintű nyelven
- Megoldás a gépi nyelv szintjén

Szoftver portabilitás (hordozhatóság)

Megoldás magasszintű nyelven

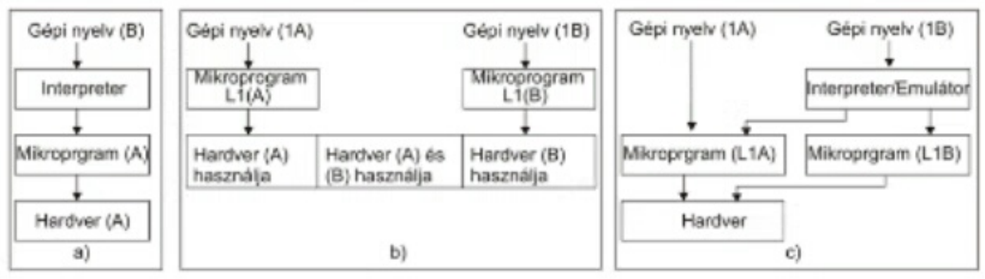
- Brute force módszer: minden géptípusra (M) minden nyelvi fordító (L), M*L fordítóprogram
- Universal Programming Language (UPL)
  M fordítóprogram (JAVA ilyen kezd lenni)
- Universal Computer Oriented Language (UNCOL)
  M+L fordító



Szoftver portabilitás (hordozhatóság)

Megoldás gépi nyelvi szinten

a) interpretálás
b) emulálás
c) interpretálás és emulálás együttesen



Számítógépgenerációk

Első megközelítésben a generációk a megvalósító technológia szerint különböznek: elektroncső – tranzisztor – IC – LSI – ULSI/VHSIC

Generáció	Technológia és architerktúra	Softver és alkalmazások	Reprezentatív rendszer
1. (1945-54)	Elektroncsövek és relés memória. PC, ACC regiszterek és fixpontos aritmetika.	Gépi kód, assembly, programozott I/O, nincs szubrutin és szerkesztés (linking)	ENIAC, Princetone IAS, IBM 701
2. (1955-64)	Tranzisztor és ferritgyürüs memória. Lebegőpontos aritmetika, multiplexelt memória hozzáférés.	HLL, szubrutinkönyvtár, kötegelt feldolgozás (batch processing)	IBM 7090, CDC 1604, Univac LARC

Számítógépgenerációk

Generáció	Technológia és architerktúra	Softver és alkalmazások	Reprezentatív rendszer
3. (1965-74)	SSI és MSI áramkörök, mikroprogramozás, csővezetékes szervezés, gyorsítótár	Multiprogramozás, időosztásos operációs rendszerek, többfelhasználós alkalmazások	IBM 360-370, PDP 8
4. (1975-90)	LSI, VLSI, félvezetős memória, multiprocesszorok, vektor és szuperszámítógép-ek	Multiprocesszoros operációs rendszerek, párhuzamos nyelvek és fordítók	VAX 9000
5. (1991-)	ULSI/VHSIC processzorok, skálázható masszív párhuzamos architektúrák	MPP, AI alkalmazások	TCM, CM-5, Intel Paragon, IBM BlueGene, Roadrunner

Mikroarchitektúrák

Amhdal törvény párhuzamos processzorokra, skalár és szuperskalár csővezetékes processzorokra. Az Earle latch. Az egyszeres pontosságú lebegőpontos formátum, a lebegőpontos szorzóegység blokkvázlata. A 4x4 bites bináris szorzó soros és párhuzamos megvalósítása. A TYP csővezeték és fizikai megvalósítása. Példa RAW, WAR és WAW adatfüggőségre. Vezérlési függőség. RAW kiküszöbölése a TYP csővezetékben: feldolgozás felfüggesztéssel, előrecsatolással. Előrecsatolás megvalósítása az ALU és MEM fázisok kimenetéről. Vezérlési hazárd esetén 4 stall ciklust beiktató megoldás. A szuperskalár csővezeték fajtái és rövid jellemzésük. A TYP1 szuperskalár csővezeték fázisai, az egyes fázisok rövid jellemzése. A célcímszámítás és feltételkiértékelés tipikus esetei. Az elágazási célpuffer (BTB). Az elágazásjövendölés módszerei: fix, szoftver támogatott, relatív cím alapján történő, előtörténeten alapuló. A P6 mikroarchitektúra: a mikroarchitektúra egységei és funkcióik.

Párhuzamos processzorok teljesítőképessége

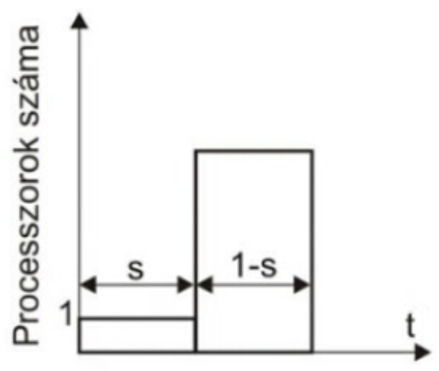
Amhdal törvény

A hatékonyság (H) az N processzor kihasználtságával jellemezhető, azaz azzal az időhányaddal, amelyben mind az N processzor dolgozik.

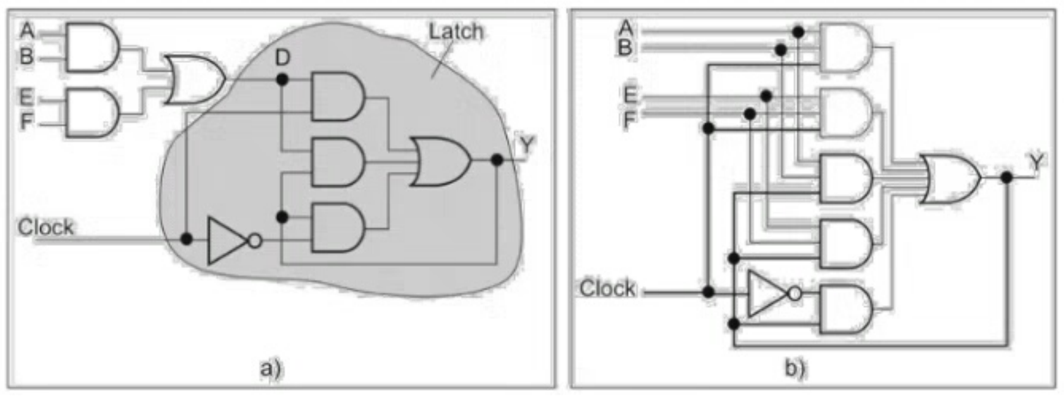
$$H = \frac{s + N \times (1 - s)}{N} = 1 - s \times \left(1 - \frac{1}{N}\right)$$

ahol

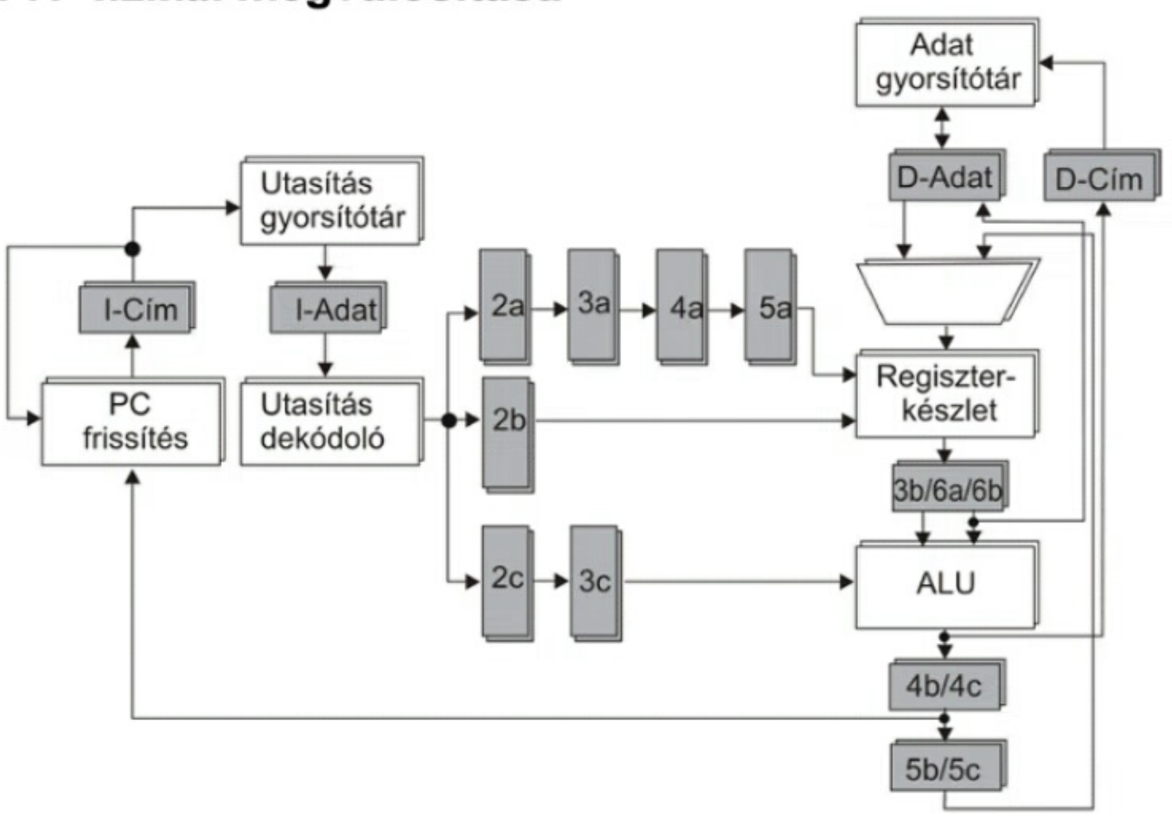
- *H* a hatékonyság,
- *s* a számítási idő azon hányada, amelyben a rendszer skalár számításokat végez
- *N* a processzorok száma (skaláris művelet esetén csak egy processzor dolgozik)
- Ha N nagy, akkor H értéke (1-s)-hez tart.



- Earle latch (J.G.Earle, IBM 360/91, 1965)



TYP fizikai megvalósítása



Az adatfüggőségek szemléltetése

Igazi függőség (RAW)	R3 ← R1 op R2 R5 ← R3 op R4
Anti függőség (WAR)	R3 ← R1 op R2 R1 ← R4 op R5
Kimeneti függőség (WAW)	R3 ← R1 op R2 R5 ← R3 op R4 R3 ← R6 op R7

Szuperskalár csővezeték hatékonysága

– Az Amhdal törvényt használjuk N párhuzamos csővezetékre

– Túl pesszimista felosztás: szigorúan szekvenciális és vektorizálható részek

– A nemvektorizálható részeknél legyen M a kisebb fokú párhuzamosság, ekkor a sebességnövekedés:

$$S = \frac{1}{\frac{(1-p)}{M} + \frac{p}{N}}$$

–Legyen p=0.75 és elérendő S=4

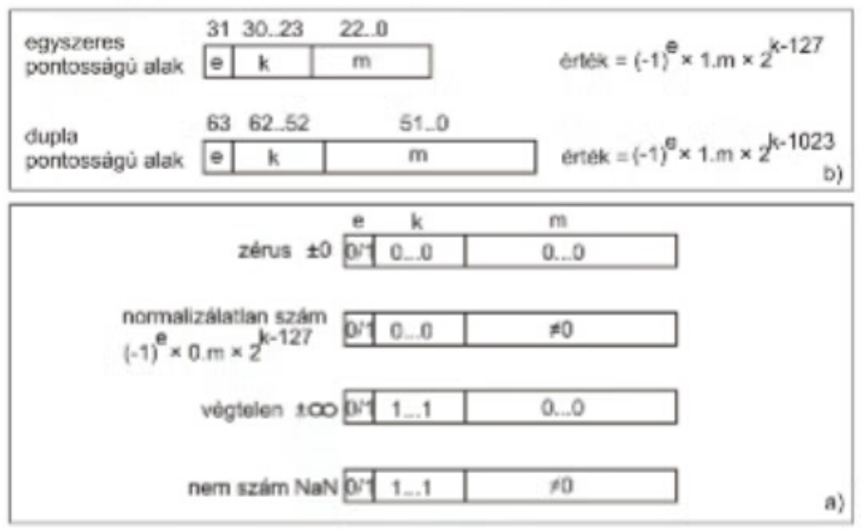
– Ha M=2 akkor N=6 csővezetékkel elérhető

– Ha M=1 akkor csak N=100 csővezetékkel érhető el!!!

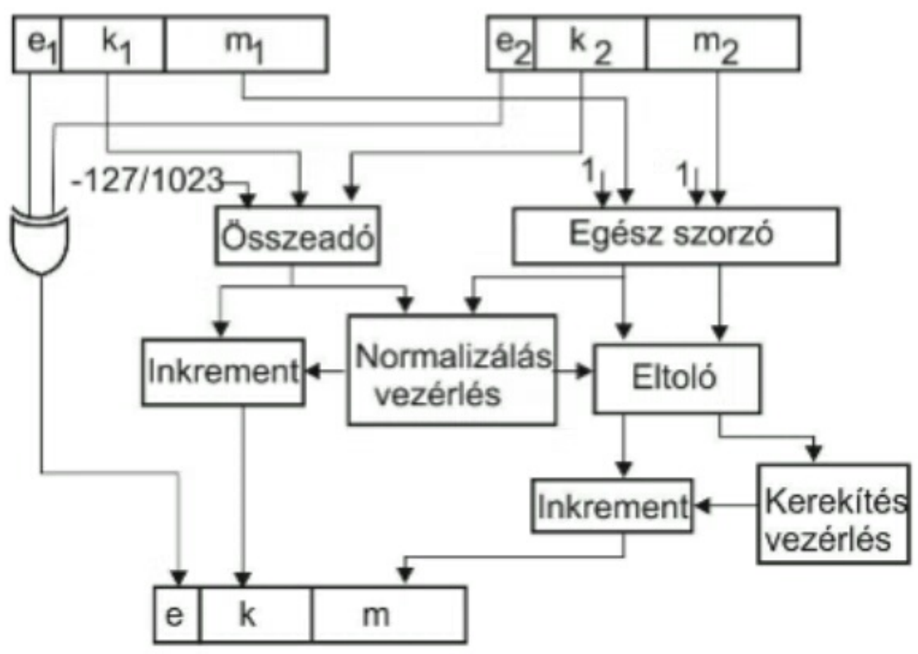
Aritmetikai csővezeték

– A fixpontos ALU csak egy, a lebegőpontos viszont több ciklust igényel

– A lebegőpontos ábrázolás (IEEE 754)



– A lebegőpontos szorzóegység

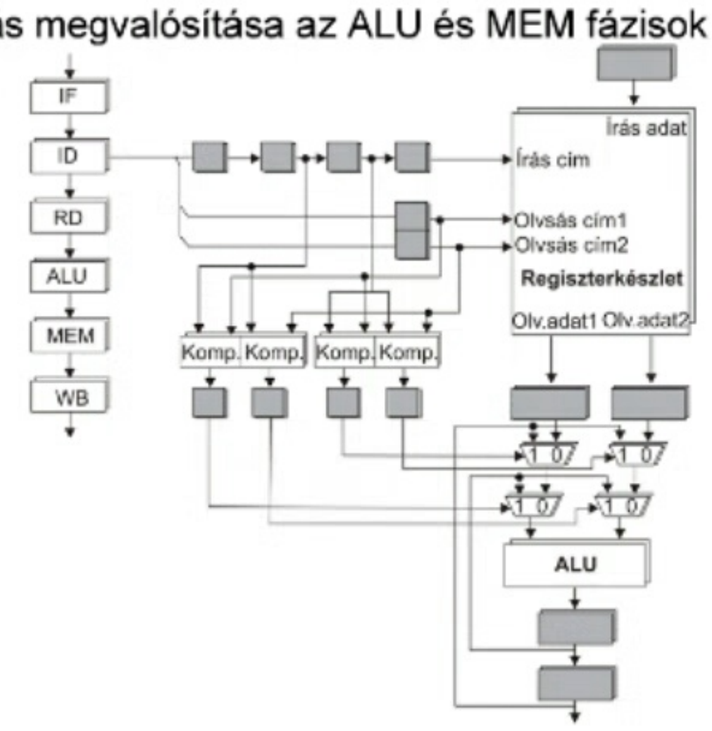


Vezérlési függőség

- Ezt a feltételes ugró utasítások okozhatják.
- Egy ilyen utasítás kimenetele határozza meg, hogy a sorrendben következő vagy egy másik utasítással kell-e folytatni a program végrehajtását.
- Normális esetben az utasítások betöltése a gépi ciklusonként megnövelt utasításszámláló értékének megfelelően történik.
- Ha a feltételes ugrás feltétele nem teljesült, akkor a szekvenciális utasításbetöltés helyes. Ellenkező esetben viszont hibás a következő utasítás(ok) betöltése.
- Az a nehézség, hogy a kétértelműség mindaddig fennáll, amíg a feltétel ki nem értékelődik.

A vezérlési függőség mint egy regiszter RAW adatfüggőség is tekinthető a PC mint regiszter vonatkozásában. A feltételes ugrás írja a PC-t, az utasításlehívás viszont olvassa. A feltételes ugró utasítás az ugráscímmel módosítja a PC-t, ha a feltétel teljesül, egyébként a PC a sorrendben következő utasítás címével frissítődik. A TYP csővezetékben a PC írása az ugráscímmel a MEM fázisban történik, ugyanakkor az PC olvasására már az IF fázisban sor kerül. Tehát RAW hazárd léphet fel a PC vonatkozásában feltételes ugró utasítások esetén.

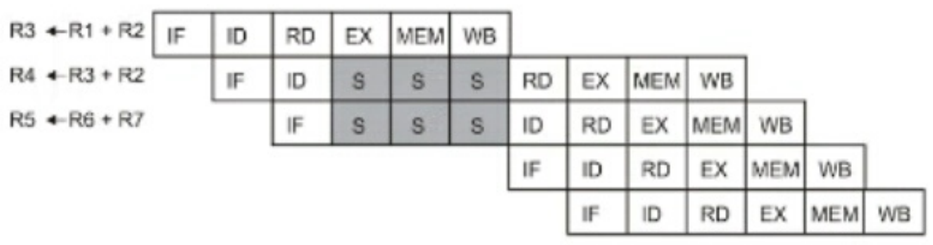
Előrecsatolás megvalósítása az ALU és MEM fázisok kimenetéről



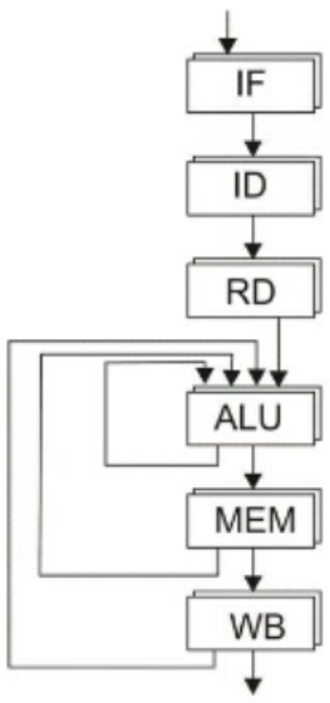
A RAW hazárd kiküszöbölése a TYP-ben

- Feldolgozás felfüggesztésével (rontja a hatékonyságot)
- Előrecsatolással (hatékonyabb)
- Elágazásjövendöléssel (vezérlési hazárd esetén)

Feldolgozás felfüggesztése (stall)

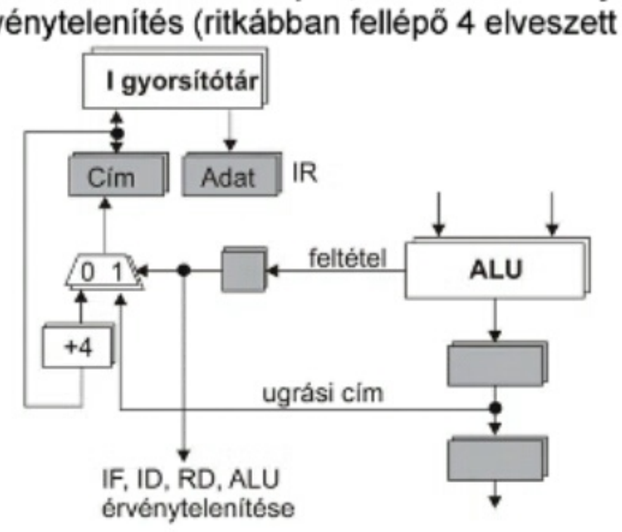


Előrecsatolás (forwarding) elve



Vezérlési hazárd

1. 4 stall ciklus beiktatása míg kiértékelődik a feltétel és/vagy kiszámítódik az ugrási cím
2. Szekvenciális továbblépés, de ha feltétel teljesült, akkor érvénytelenítés (ritkábban fellépő 4 elveszett ciklus)

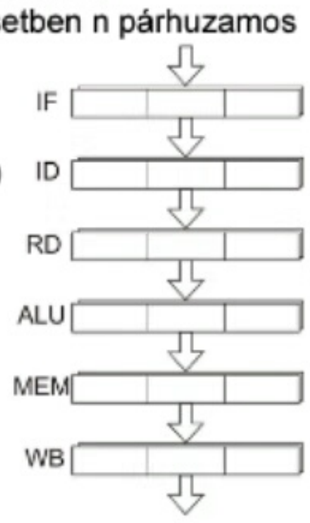


A párhuzamos csővezetékek fajtái

- Egységes szerkezetű csővezeték
- Diverzifikált csővezeték
- Dinamikus csővezeték

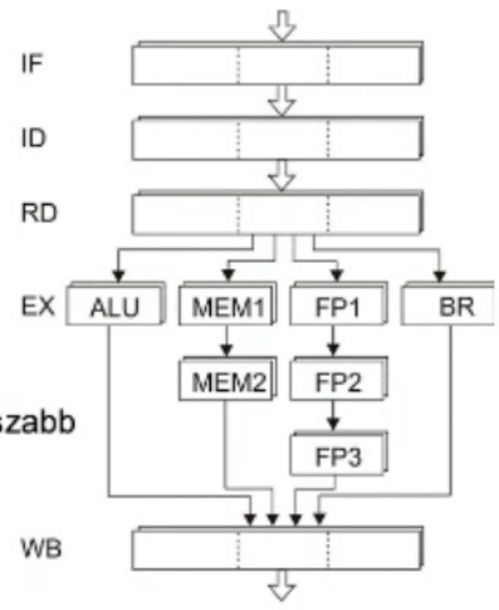
Egységes szerkezetű csővezeték

- Az egyes csővezetékek azonos fázisokat tartalmaznak
- Kétféle párhuzamosság: időbeli és térbeli
- Tetemesebb hardvert igényel mint az időbeli párhuzamosság. Legáltalánosabb esetben n párhuzamos csővezeték esetén
  - Az egységek többszörözése (n)
  - Kommunikáció biztosítása a párhuzamos fázisok között (n^2 kapcsolat)
  - Regiszterblokk, valamint a gyorsítótár író és olvasó kapuinak többszörözése (n)



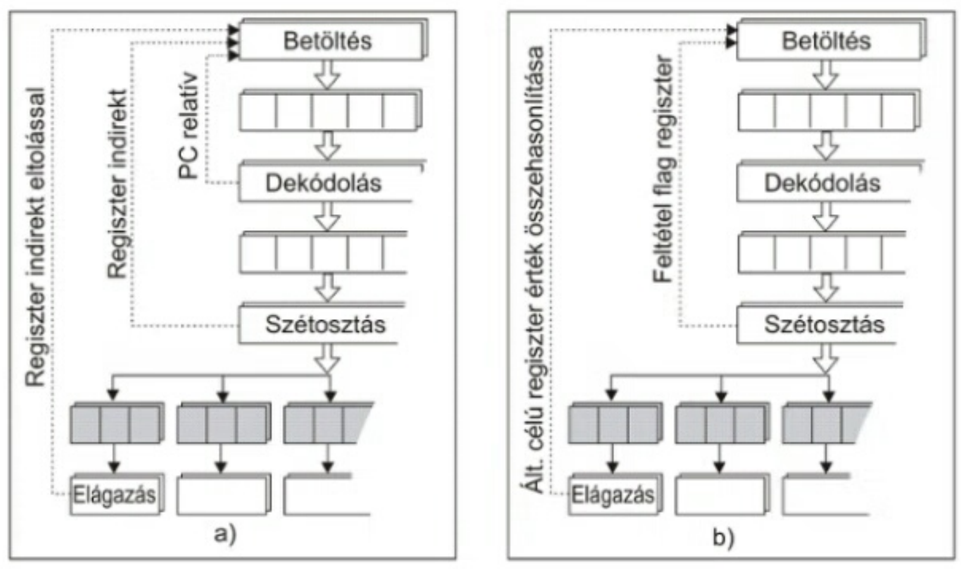
Diverzifikált csővezeték

- A különböző utasítások különböző végrehajtási időket igényelnek
- Nem azonos az egyes csővezetékek fokozatainak száma
- De csak akkor van teljesen kihasználva, ha az utasítások típusa azonos a csővezeték-típusokkal
- Felfüggesztődhet egy csővezeték, ha egy hosszabb csővezeték még nem írt vissza egy regisztert (pontozótábla)



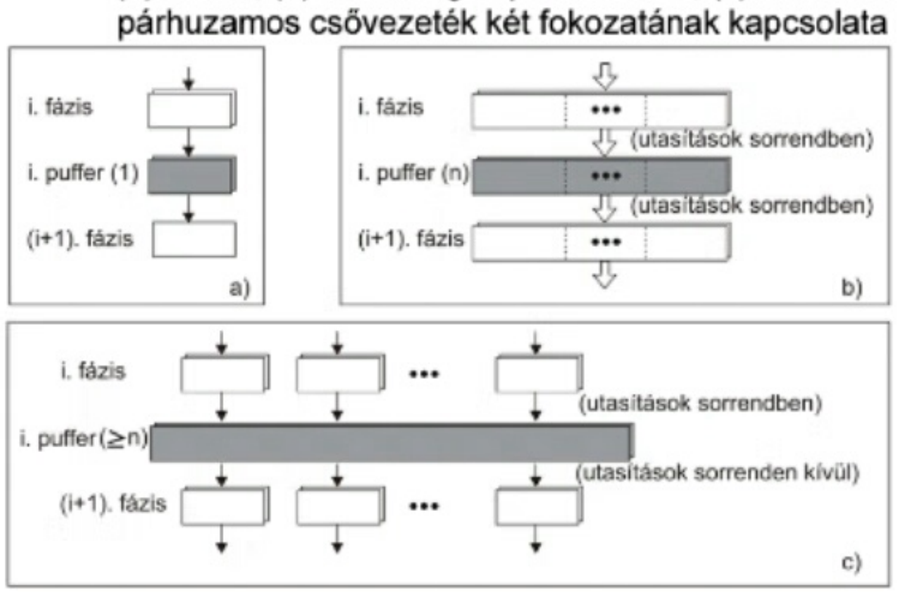
Célcímszámítás és feltétel kiértékelés

- A célcím kiszámítása a címzési módtól függően 1,2 vagy 3 ciklust igényel
- A feltétel kiértékelése az utasítás típusától függően 2 vagy 3 ciklust igényel



Dinamikus csővezeték(sorrenden kívüli végrehajtás)

(a) skalár, (b) közönséges párhuzamos, (c) dinamikus párhuzamos csővezeték két fokozatának kapcsolata



Elágazásjövendölés

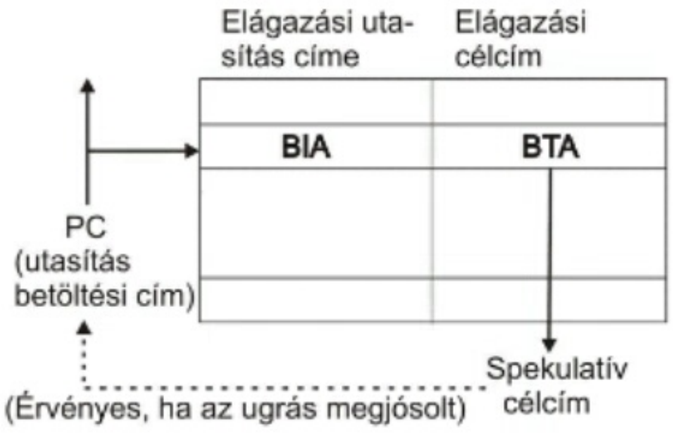
- Különböző spekulatív módszereket használnak: az elágazási utasítás korábbi kimenetei alapján jósolnak
- Jövendölés két komponense: célcím és elágazási feltétel
- A feltétel kiértékelődése után azonban az ellenőrzést el kell végezni, s téves jövendölés esetén a hibás utat eldobni, s a helyesen felépíteni
- Ismertetett módszerek
  - Elágazási célpuffer (Branch Target Buffer = BTB)
  - Elágazásjövendölés az előtörténet alapján
  - Adatpív, kétszintű elégazásjövendölés
  - Felgyógyulás a hibás elágazásjövendölésből

Több módszer az elágazás jövendölésére

- Legegyszerűbb: az ugrás nem teljesül feltételezés
- Szoftver támogatás ISA módosításával
  - Az elágazás kiértékelődése előtt a következő utasítás töldödik be és hajtódik végre
  - Nem túl jó: ciklus végén mindig hibásan jósol
- Szoftver támogatás ISA módosításával
  - Egy extra bitet a fordítóprogram állít be, ez utal arra, hogy valószínű-e a feltétel teljesülése (Motorola M88110)
- IMB RS/6000
  - Az elágazási utasítás és a célutasítás relatív helye alapján jósol
  - Ha az elágazási utasítás címe nagyobb mint a célutasításé, akkor elágazást jósol (valószínűleg ciklus végén van az elágazás)
  - Egyébként a soron következőt javasolja

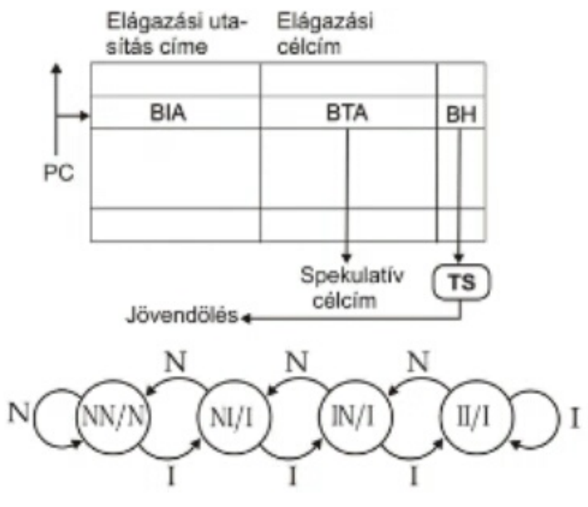
Elágazási célpuffer

- Az elágazási utasítás által legutoljára használt célcímet tartalmazza
- Hozzáférés az IF fázisban a PC tartalmával
- BTA azonnal betölthető a PC-be, ha a jövendölés szerint az elágazás végrehajtandó



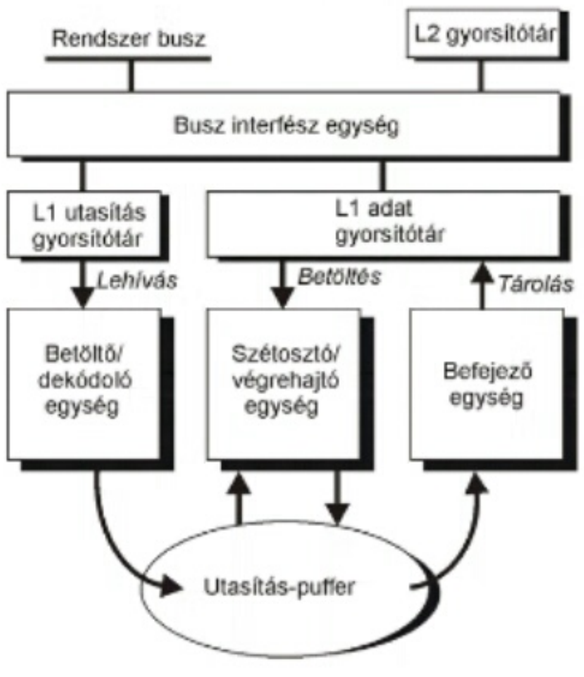
Jövendölés az előtörténet alapján

- A BTA-ban a BIA és BTA mellett egy harmadik bejegyzés is van, az elágazás előtörténet (Branch History = BH)
- Tipikus 2-bites elágazásjövendölés



- Kiindulási állapot NN, s a kimenet az állapothoz van hozzárendelve
- Előnyben részesíti az elágazás végrehajtását, mivel három állapotában kimenete elágazást jövendöl (ciklus esetén jól működik), azaz hosszabb N vagy I sorozatot tételez fel.
- IININ esetén viszont csak 50%, NNININ esetén pdig 0% a helyes jövendölés.
- Pentium "telítődő számlálós megoldás": NN/N, NI/N, IN/I, II/I hozzárendelés
- Milyen állapotgép lenne a legjobb?
- 1992 IBM RS/6000 gépen szimuláció.
  - 2-bites elágazásjövendölések esetén a lehetséges állapotgépek száma 2^{16}
  - A triviálisan érdekteleneket elhagyva marad 5248
  - A legtöbb benchmark program a telítődő számlálót hozta ki (89%-94% helyes elágazásjövendölés)

A P6 mikroarchitektúra egységei

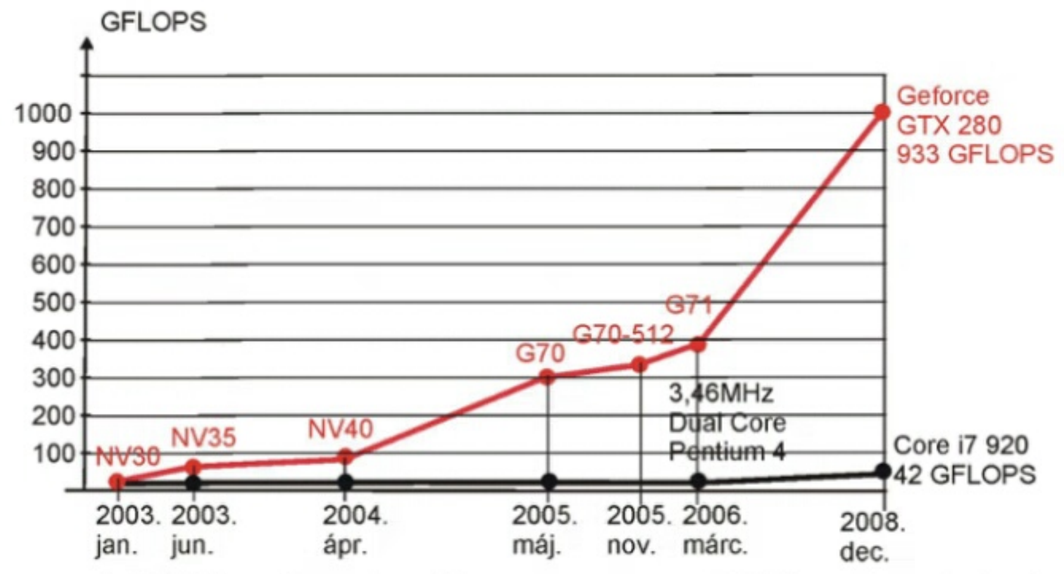


- Betöltő/dekódoló egység
  Ez az egység sorrendben hívja le a program utasításait az utasítás gyorsítótárból, mikroutasításokba dekódolja. Az adatáramlás ebben az egységben az utasításáramlásnak felel meg. Az utasitás elővétel maga spekulatív.
- Szétosztó/végrehajtó egység
  Ez egy sorrendtől eltérő végrehajtású egység, amely fogadja az adatáramlást, a mikroutasítások végrehajtását az adatfüggőség és a rendelkezésre álló erőforrások függvényében ütemezi, s átmenetileg tárolja a spekulatív végrehajtás eredményeit.
- Befejező egység
  Ez az egység az eredeti utasítássorrendben tárolja az átmeneti, spekulatív eredményeket a permanens architektúrális állapotban.
- Busz interfész egység
  Részben sorrendben működő egység. Közvetlenül kommunikál az L2 gyorsítótárral négy konkurens hozzáférést biztosítva. Ez az egység a tranzakciókat is vezérli a memória felé MESI szimatoló protokollal.

Speciális processzorok

Grafikus processzorok: a fejlődés trendje, és a BitBlt művelet. GPU változatok a videó memória szerint. Az egyszerű grafikus csővezeték fokozatai, s az egyes fokozatokban elvégzett feldolgozás. A CPU és a GPU párhuzamos működésének leírása, dedikált feldolgozó egységeket, ill. univerzális feldolgozó egységeket alkalmazó GPU-k összehasonlítása.

A sebesség fejlődése



- A GPU-k sebessége 10-20-szorosa a CPU-k sebességének
  Oka: nagymértékű párhuzamosság (lásd később az architektúránál)

GPU változatok a videó memória szerint

- Dedikált grafikus megoldások
  - A számítógéptől független, saját grafikus memóriával rendelkeznek
  - Ezeknek a megoldásoknak a teljesítménye a legnagyobb
  - Figyelem: a dedikált megoldás nem feltétlenül jelent ki-behelyezhető grafikus kártyát
- Integrált grafikus megoldások
  - A számítógép memóriájának egy részét használják a grafika céljaira
  - Ezeknek a megoldásoknak a teljesítménye elmarad a dedikált változatokétól, de lényegesen olcsóbbak
- Hibrid grafikus megoldások
  - Kisebb saját grafikus memóriával is rendelkeznek, ami mellett a számítógép memóriáját is használják
  - Árban és teljesítményben az előbbi két megoldás között helyezkednek el

Transzformáció

Input: háromszögek helyi koordináta rendszerben

Művelet: A háromszögek globális koordináta rendszerbe transzformálása

Kimenet: Háromszögek folyama közös koordináta rendszerben

A funkcionalitás fejlődése 1: Primitív pixelmanipuláció

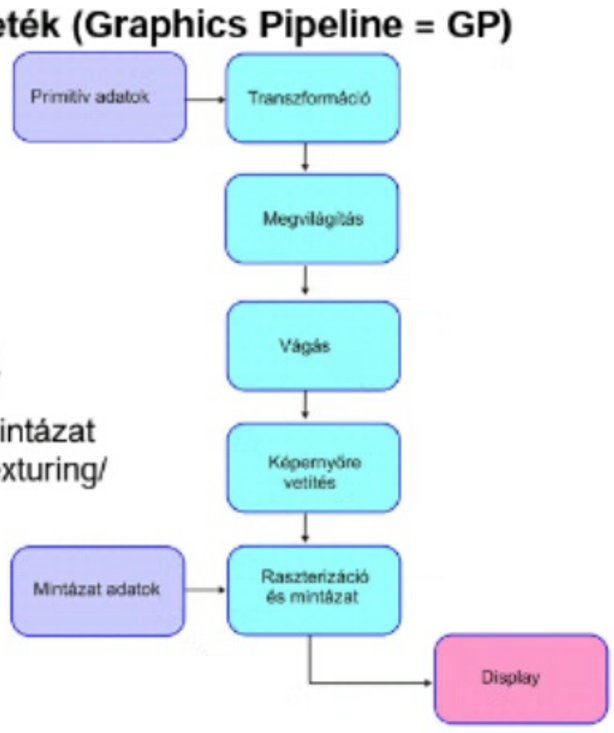
- A primitív pixelmanipulációban az elsőnek megjelent és legalapvetőbb a bit-blokk transzfer (BitBlt = Bit-Block Transfer) volt. Ez egy olyan számítógépes grafikai eljárás, amely több bittérképet egyetlen bittérképbe kombinál.
- Legtöbbször csak két bittérkép vesz részt a műveletben, a forrás- és a cél bittérkép. A forrás és a cél bitjei a megadott raszter művelettel (Raster OPeration = ROP) kombinálva, az eredmény visszaíródik a cél bittérképbe.
- A raszter művelet tipikusan logikai ÉS, VAGY, Kizáró VAGY és a NEM művelet.
- A BitBlt legtipikusabb alkalmazása egy transzparens foltgrafikának egy másik grafikára való helyezése az ún. maszkolt BitBlt, ami egy AND és egy OR rasztermüvelettel valósítható meg.

Grafikus csővezeték (Graphics Pipeline = GP)

- Bemenete egy háromdimenziós "látvány", melyet egymást követő feldolgozási fokozatokban egy kétdimenziós raszteres képpé alakít át.
- A két legleterjedtebb, szabványos grafikus csővezeték modellt az OpenGL és a Direct3D specifikációja tartalmazza.
- A modern GPU-k lehetővé teszik a grafikus csővezeték egyes fokozatainak programozott végrehajtását.
- Az egyes fokozatokat feldolgozó programokat shader-nek (árnyékoló) nevezik, mivel kezdetben alapvetően csak ezt a fázist valósították meg.
- Először röviden ismertetjük a GP egyes fokozatait, majd két tipikus GPU egységet, melyek alkalmasak a GP fokozatainak végrehajtására
  - Az egyes fokozatokhoz dedikált egységeket hozzárendelő típus
  - A különböző fokozatokhoz azonos feldolgozó egységeket használó típus (ez a legmodernebb változat)

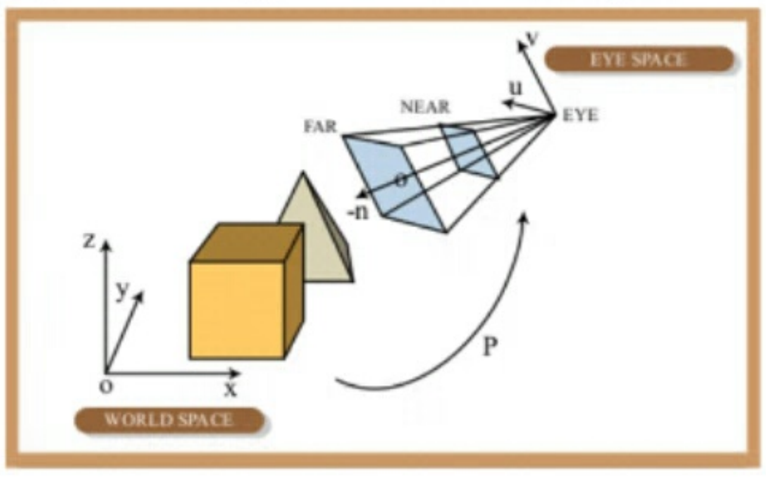
Grafikus csővezeték (Graphics Pipeline = GP)

- Transzformáció
- Megvilágítás /Illumination/
- Vágás /Clipping/
- Képernyőre vetítés /Screen Projection/
- Raszterizáció és mintázat /Rasterization & Texturing/



Transzformáció - folytatás

- Az egyes háromszögcsúcspontok (vertex) transzformálása eltolási és forgatási mátrixokkal
- Ez nagyon sok lebegőpontos mátrixműveletet jelent



Megvilágítás

Input: megvilágítatlan háromszögek közös koordináta rendszerben

Művelet: A háromszögek megvilágítása különböző fényforrásokkal

- Minden fényforrás külön feldolgozva
- A részeredmények összegzés (Phong Lighting Equation)
- PLE ismételt vektorszorzást igényel

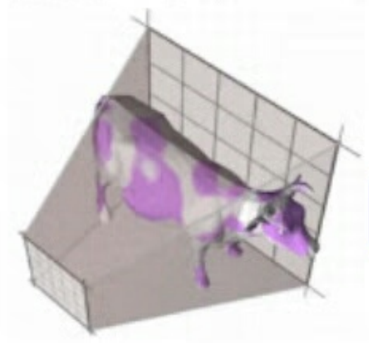
Kimenet: Háromszögek kiszámított színekkel

Vágás

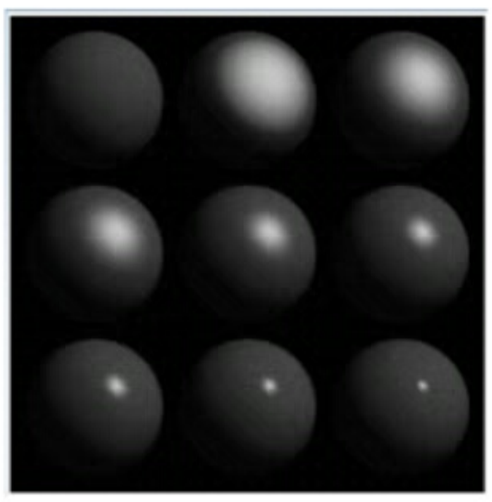
Input: a teljes látvány

Művelet: a néző látóterében nem látható részek eltávolítása

Kimenet: a vágott látvány



Megvilágítás illusztrálása



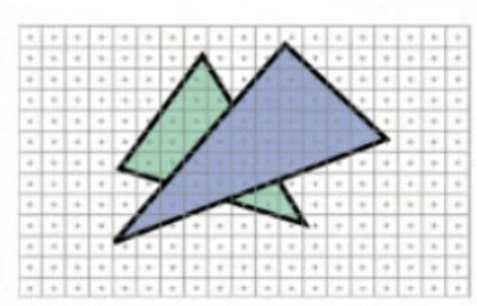
Raszterizáció

Input: háromszögek sorozata

Művelet: a háromszögek pixelekbe konvertálása

- meghatározni: melyik háromszög fed le egy pixelt
- Mintázat (textura) adatok olvasása és alkalmazása
- A generált pixelek attribútumai a csúcspontok attribútumai alapján számítódnak ki perspektivikus interpolációval.

Kimenet: pixelek sorozata

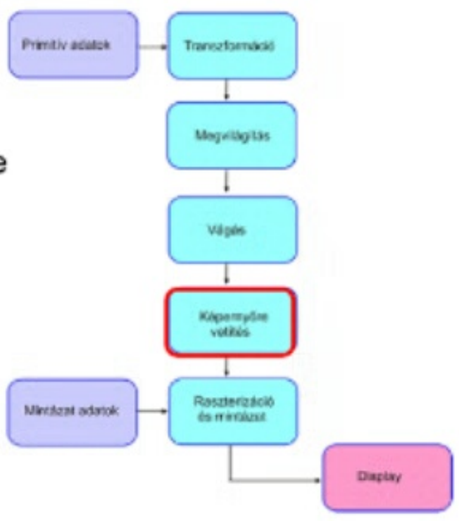


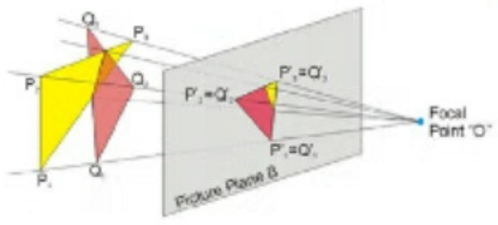
Képernyőre vetítés

Input: 3D-s, színezett háromszögek sorozata

Művelet: a háromszögek leképzése 3D-ből 2D-be

Kimenet: háromszögek sorozata a kimeneti képernyő koordináta rendszerében





Tipikus követelmények a FIR szűrő alapján

- Gyors hozzáférés az adatokhoz és az utasításokhoz
- Szorzás és összeadás (MAC = Multiply-Accumulate) gyors, konkurens végrehajtása (tipikusan egy ciklusban)
- Csúsztatható ablak fenntartása valamilyen értékek (például a bemeneti, mintavételezett és digitalizált jelek) halmazán → Cirkuláris puffer alkalmazása

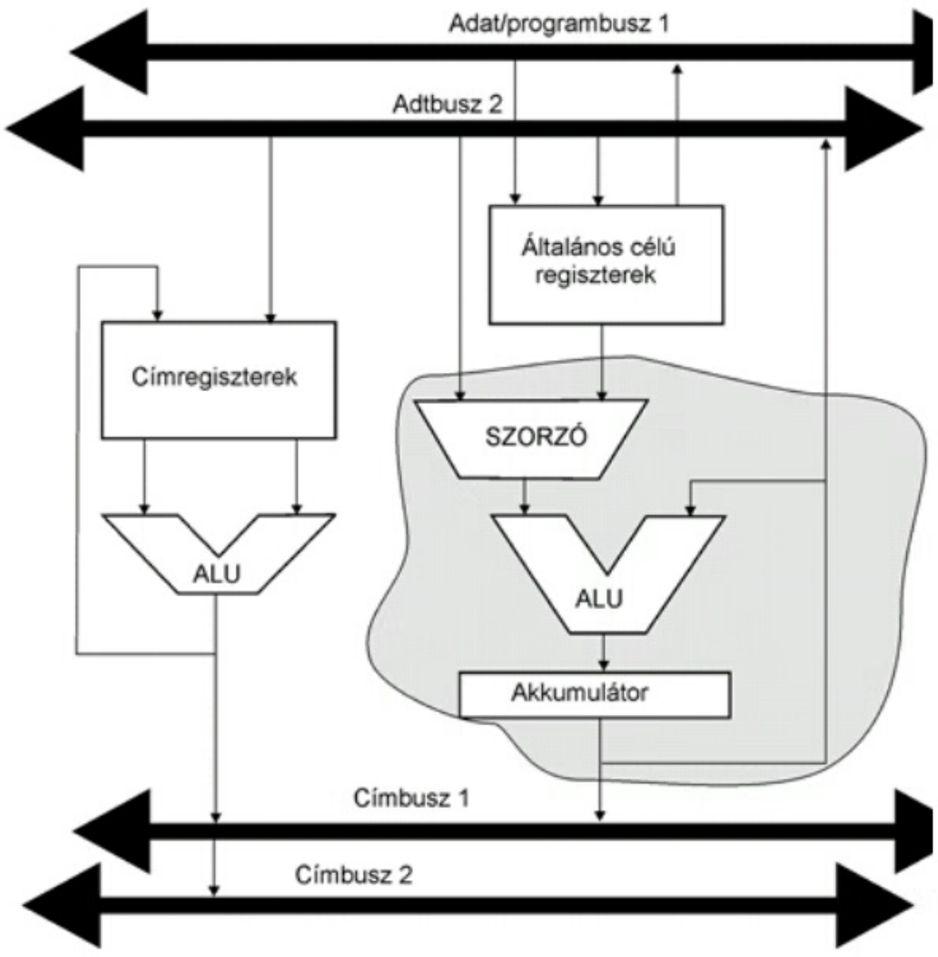
Univerzális shraderek használata

- A megjelenítendő képek típusától nagymértékben függ, hogy tetemesebb vertex vagy tetemesebb pixel feldolgozást kell-e végezni
- Ezért egy adott számú vertex-, ill. pixel feldolgozóval az egyik típus esetén a vertex-, a másik típus estén meg a pixel feldolgozó lesz kihasználatlan
- A megoldás: univerzális feldolgozók használata, melyek akár a vertex, akár a pixel feldolgozásra alkalmasak, s így teljes mértékben kihasználható a feldolgozó kapacitás

Gyors hozzáférés az adatokhoz

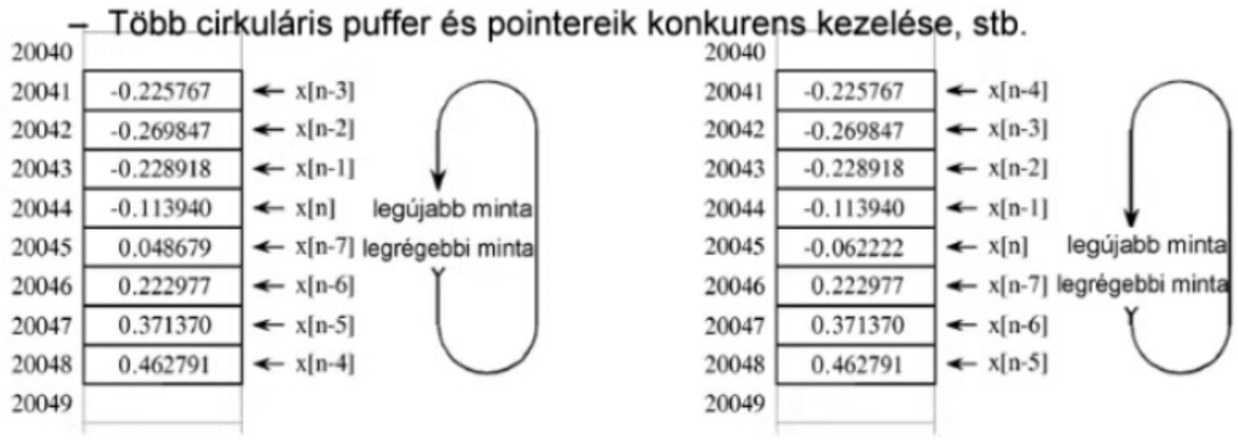
- Von Neumann architektúra nem megfelelő
  - Szűk keresztmetszet az adatok átvitelekor
- Harvard architektúra (duális memória)
  - Jobb
- Szuper Harvard architektúra

Klasszikus DSP archiektúra MAC egységgel



Cirkuláris puffer

- A cirkuláris (moduló címzésű) puffer kezelése négy paraméterrel
  - Puffer kezdete
  - Puffer vége
  - Címzés lépésköz
  - A legújabb minta címe
- Optimális kezelés
  - Cím inkrementálás és olvasás/írás egy ciklusban
  - Több cirkuláris puffer és pointereik konkurens kezelése, stb.



Ciklusszámláló

- Zero-Overhead Looping
- For-next ciklus a ciklusszámláló explicit dekrementálása és tesztelése nélkül
- Az egyszerű ciklusszámláló egyetlen utasítás n-szeri végrehajtását vezérli.
- A fejlett cikluszámláló m utasítás n-szeri vérehajtását vezérli.

Hardver verem

- Gyors megszakítások, regiszter push/pop és call/return utasítások esetén használható.
- Sokkal gyorsabb mint a rendszer memóriában leképzett verem.

Speciális címzési módok

- Moduló címzési mód (cirkuláris puffer megvalósítása)
- Fordított bitsorrendű címzés
  - FFT (Fast Fourier Transform) esetén a feldolgozás bizonyos fázisában a címbiteket fordított sorrendben kell használni

Mi a hálózati processzor?

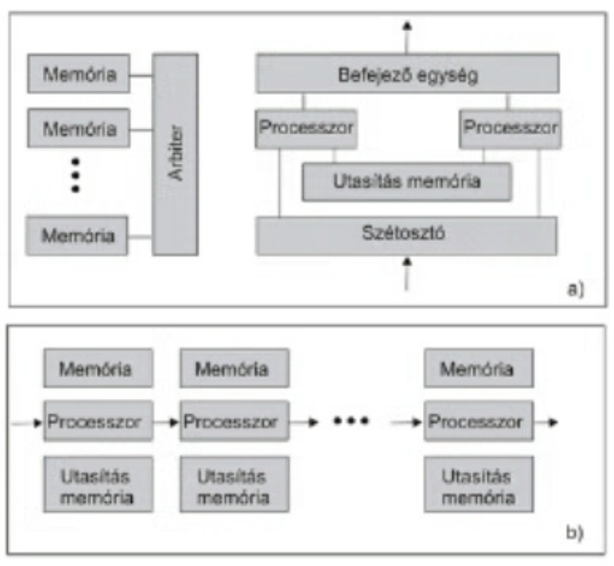
- Egyszerű definíció: a hálózati processzor, a továbbiakban NP (Network Processor) egy olyan programozható eszköz amelyet speciálisan az adatcsomagoknak „vezeték sebesség” melletti feldolgozására terveztek meg.
- Milyen feldolgozást? A legfontosabbak
  - Ellenőrző összeg vizsgálata
  - Mezőkinyerés
  - Mintaillesztés, csomagosztályozás
  - Célhálózat megállapítása
  - Keresés (lookup)
  - Útvonalválasztás
  - Feldarabolás és összerakás
  - Feldolgozás
  - Fejléckezelés
  - Várakozási sor kezelés
  - Ellenőrző összeg generálása
  - Elszámolási információk gyűjtése
  - Statisztikák gyűjtése

NP rendszerarchitektúra

Két alapvető modell:

a) végrehajtás a befejezésig (RTC=run-to-completion) feldolgozási modell

b) csővezetékes feldolgozási modell



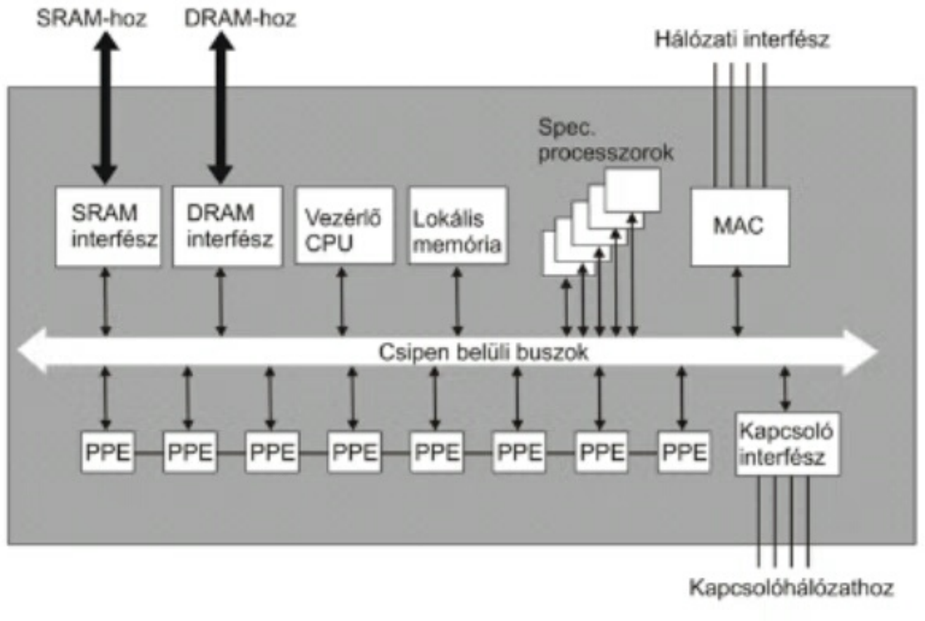
Csővezeték modell

- Az egyes CPU-k más és más feladatosztályok végrehajtására optimalizáltak és optimalizált utasításkészlettel rendelkeznek.
- Az alkalmazói program particionálva van az egyes processzorok között.
- Gyengesége:
  - A feldolgozási tevékenységet egyenletesen kell elosztani az egyes processzorok között.
  - A programot is megfelelően particionálni kell az egyes csővezeték fázisokhoz.
  - Természetesen a program nem mindig particionálható ideálisan, ami kihasználatlan processzor ciklusokhoz vezethet egyes fázisokban.
  - Továbbá járulékos ciklusokra van szükség a csomagok továbbítására az egyes fázisok között.
  - A legnagyobb kihívása a csővezetékes modell programozásának a változtatásokhoz kapcsolódik. Ugyanis egy kis változtatás után a kód particionálását elölről kell kezdeni.

RTC modell

- A programozó egyetlen szálat lát, amely hozzáfér a teljes utasításmemória tartományhoz, valamint az összes közös használatú erőforráshoz, mint például vezérlő memória, táblák, számlálók, stb.
- Szimmetrikus multiprocesszorokon alapszik, ebben az esetben több CPU osztozik egy közös memórián. A CPU-k mint egy feldolgozási erőforráscsoport tagjai szerepelnek, szimultán működnek, s feldolgozást végeznek vagy feldolgozásra várnak.
- Két almodell:
  - Az egyes processzorok az allokált csomagot megszakítás nélkül dolgozzák fel.
  - Egy processzoron több szál fut, egy-egy csomagot egy-egy szál képvisel, s a szálak között még a csomag teljes feldolgozása előtt váltás történhet (pl. várakozás a külső memóriára).

Egy tipikus NP



– Interfész a környezettel
  - A hálózati interfészen keresztül fogadja a feldolgozandó csomagokat, majd azokat a feldolgozás után ugyanezen az interfészen keresztül küldi el. Ez a MAC-ot (Media Access Control), amely a 7-rétegű OSI modellben az adatkapcsolati réteg (2-es réteg) funkcióját látja el. Fejletlenebb hálózati processzorok nem tartalmazzák a MAC-ot, s csak a 3-as rétegtől kezdve felfelé implementálják az egyes rétegek funkcióit.
  - Nagyteljesítményű hálózati eszközökben több hálózati processzort használnak. Ilyen esetben a feldolgozott csomag nem csak a MAC-on keresztül hagyhatja el a processzort, hanem a kapcsolóhálózati interfészen keresztül egy másik hálózati processzorhoz továbbítódhat. A kapcsolóhálózati interfész vagy annak egy része egy számítógépes busz interfész (például PCI) is lehet.
  - Az SRAM interfészen keresztül külső statikus memóriához csatlakozik a processzor, amely az útvonalválasztáshoz szükséges táblázatokat és más, gyors hozzáférést igénylő adatokat tárolhat. A DRAM tipikusan a feldolgozás alatti csomagokat és felügyeleti információt tárol.

– A kulcselemek: a PPE elemek (Packet Processing Engine). A csomagfeldolgozásra specializált utasításkészletű RISC magok, rendszerint külön utasítás és adat gyorsítótárral.
– A PPE-ket használata a már említett modell szerint kétféle lehet. Az egyik megoldásban minden PPE egyforma. Amikor csomag érkezik a hálózati processzorhoz, az az egyik szabad PPE-hez irányítódik. Ha nincs szabad PPE, akkor a csomag a DRAM-ban tárolt várakozási sorba kerül, amíg egy PPE fel nem szabadul. Ha ezt a módszert alkalmazzák, akkor a PPE-k közötti vízszintes kapcsolat nem létezik, mert nem kell kommunikálniuk egymással.
– A másik PPE elrendezés a csővezeték. Ebben az esetben mindegyik PPE egy meghatározott feldolgozási lépést hajt végre, majd az adott csomag pointerét átadja a csővezetékben a rákövetkező PPE-nek. Mindkét elrendezésben a PPE-k teljesen programozhatók.

– Fejlettebb rendszerekben a PPE-k többszálúságot is használnak, s több regiszterkészletük van. Amikor egy PPE megakad, mert például adatot kell olvasnia a DRAM-ból, akkor azonnal át tud váltani egy másik szálra.
– A PPE-k mellett minden hálózati processzor egy vezérlőprocesszort is tartalmaz (általános célú RISC CPU). Ez a csomagkezeléssel szorosan nem összefüggő feladatok elvégzésére szolgál, mint például az útvonalválasztó táblák frissítése.
– Speciális processzorok, gyorsító egységek a mintaillesztés és más kritikus művelet végrehajtására. Ezek valójában kis ASIC elemek, amelyek nagyon jók egyetlen művelet elvégzésére, például az útvonalválasztó táblából egy cím megtalálására.
– A hálózati processzor komponensei a lapkán elhelyezett buszokon keresztül kommunikálnak egymással, melyek sebessége több Gb/s (az ábrán egyetlen buszként reprezentáltuk a több buszt).

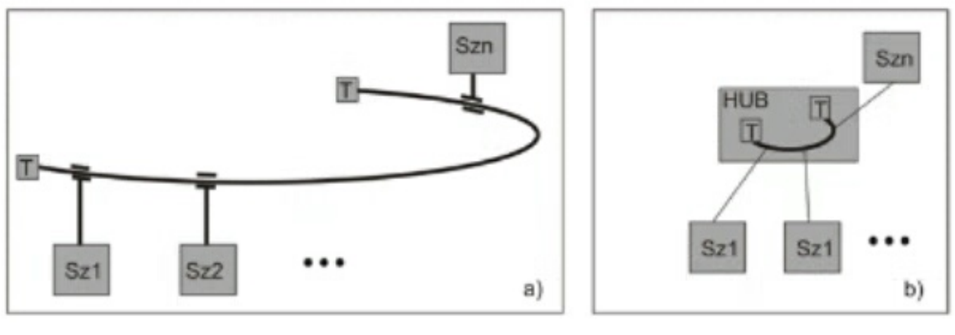
– A későbbiekben az Ethernet sodrott érpárt kezdett használni átviteli közegként, ekkor az eredeti koaxiális gerinchálózat tulajdonképpen egy elosztón, idegen szóval egy hub-on belülre zsugorodott, az ún. összezsugorodott gerinchálózattá (b. ábra).
– A hub kapui pont-pont interfészt valósítanak meg az egyes számítógépek felé.
– Egy adott kapun beérkező csomag valamennyi kapun keresztül az összes számítógép felé továbbítódik, s egy számítógép csak akkor kezdhet adni, ha a hub-ra csatlakozó egyetlen számítógép sem ad.
– A hub-ok használata esetén tehát a működés megegyezik az eredeti megoldás működésével, azaz minden eszköz az ütközési doménben van.
– A jelszinteket azonban regenerálja, ezért nagyobb kiterjedésű hálózatokban nagyon fontos.
– Az OSI modellben a hubok 1-es szintű eszközök.

Switch (kapcsoló)

– Az Ethernet ütközéses domén jellegét a kapcsolók kiküszöbölik.
– Topológiailag hasonlítanak egy hub-okhoz, működésük azonban jelentősen eltér. A bejövő Ethernet csomag cél MAC címe alapján azt csak azon kapuján keresztül továbbítják, amelyre csatlakozó számítógép MAC címe megegyezik a cél MAC címmel.
– Minden Ethernet interfész rendelkezik egy 48 bites MAC (Media Access Controller) címmel, melyet a gyártó rendel hozzá a hálózati interfész kártyához (NIC=Networl Interface Card), s azon egy ROM tárolja.
– A kapcsoló megjegyzi, hogy az egyes kapuira milyen MAC című hosztok csatlakoznak, s a bejövő csomagot csak arra a kapujára továbbítja, amelyre csatlakozó hoszt MAC címe

HUB (elosztó)

– Az Ethernet fizikai rétege (OSI 1-es szint) egyetlen koaxiális kábelből állt, erre csatlakozott fizikailag a hálózat valamennyi eszköze.



– Az Ethernet hálózaton minden hoszt figyelte a kábel forgalmát, mielőtt adni kezdett volna, s csak forgalommentes esetben (megfelelő időt kivárva) kezdhette el az adást. Mivel a kábel csomópontjai közötti jelterjedési idő véges, ezért az előbbi feltétel betartása mellet is több hoszt elkezdhetett adni, ami az üzenetek ütközéséhez vezetett. Ezt az adók észlelve leállították az adásukat, s bizonyos időt kivárva kezdtek újra neki az adásnak.
– Az Ethernet hálózat ún. ütközéses domén, mivel minden eszköznek várakoznia kell a vonal szabad voltára, s az egyes eszközök interferálhatnak egymással.
– Az egyetlen adatátviteli médium (koaxiális kábel) komoly hátránya, hogy bármelyik rácsatlakozó eszköz meghibásodása vagy a kábel egy adott helyen fellépő hibája a teljes hálózat működésképtelenségét okozhatja.

Router (útvonalválasztó)

– Az útvonalválasztók hasonlóan működnek mint a kapcsolók, azaz ezek is irányítják és „szűrik” a hálózati forgalmat. Ezt azonban nem a csomag MAC címe alapján teszik, hanem egy adott protokoll szerint, egy a MAC címnél jobban becsomagolt paraméter alapján.
– Az útvonalválasztók az OSI modell szerint 3-as szintű hálózati elemek. Ezért ezek különböző 1-es szintű hálózatokat kapcsolhatnak össze (Ethernet, Token Ring, SONET sőt még az RS232-t is).
– Az otthoni használatra szánt útvonalválasztók a hálózati címtranszformáció (Network Address Translation = NAT) lehetőségét is biztosítják, így egyetlen ISP címet több hoszt közösen használhat.

Gateway

– Ezek az eszközök az OSI modell felső szintjein működnek, s az információ transzformálását végzik el teljesen eltérő hálózati architektúrák és adatformátumok között.

Többprocesszoros rendszerek
Osztályozás: feladat hozzárendelési mód, processzorok közti kapcsolat, utasítás-és adatáram, memória megosztottság és vezérlési mechanizmus szerint. Példával demonstrálja a vezérléses-, adatáramlásos és igényvezérelt típust. Az ILLIAC -IV blokkvázlata, síkbeli hőmérsékletelosztás számítása tömbprocesszorral. Az omega hálózat és útválasztási stratégiája. Gyorsítótár koherencia: lényege, szoftver megoldások, hardver megoldások (könyvtár alapú és szimatoló).

Osztályozás

Osztályozási szempontok

- Feladat hozzárendelési mód
- Processzorok közti kapcsolat
- Utasítás és adatáram
- Memória megosztottság
- Vezérlési mechanizmus

Feladat hozzárendelési mód

Dinamikus feladat hozzárendelési mód

- Egy processzor a rendszerkapacitás egy részén minden funkciót elláthat, s futási időben dől el, hogy egy processzornak milyen feladatot kell ellátni.
- Előnye: egy processzor meghibásodása csak a rendszer egy részére terjed ki, s a processzorok teljesítőképessége jól kihasználható.
- Hátránya: bonyolult szoftver igényel, s egy processzor teljesítőképessége korlátozza a megvalósítható funkciókat.

Statikus feladat hozzárendelési mód

- Egy-egy processzor meghatározott funkciót lát el, a funkció hozzárendelés rögzített, már a futás idő előtt el van döntve, s az alatt nem változik.
  - Előnye: egyszerűbb szoftvert igényel, s a processzorok felépítése a feladatnak megfelelően optimalizálható.
  - Hátránya: Az egyes processzorok terhelése jelentős mértékben eltérő lehet, s ez a rendszer érzékenyebb a meghibásodásokra.

Processzorok közti kapcsolat

- A processzoroknak kommunikálni kell egymással, hiszen a közös feladat egyes részeit oldják meg. A kommunikáció jellege szorosan- és lazán csatolt lehet.

Szorosan csatolt rendszerek

- A processzorok közötti kapcsolat egy közös erőforráson (rendszerint memórián) keresztül valósul meg. Ezért nevük: osztott memóriás többprocesszoros rendszerek (shared memory multiprocessor system).
  - Fontos jellemzőjük: a processzorok egy közös operációs rendszer alatt működnek.
  - Kis processzorszám mellett igen hatékonyak, de sok processzor esetén gyakorlatilag már nem növekszik eredő teljesítményük a processzorok számát növelve.

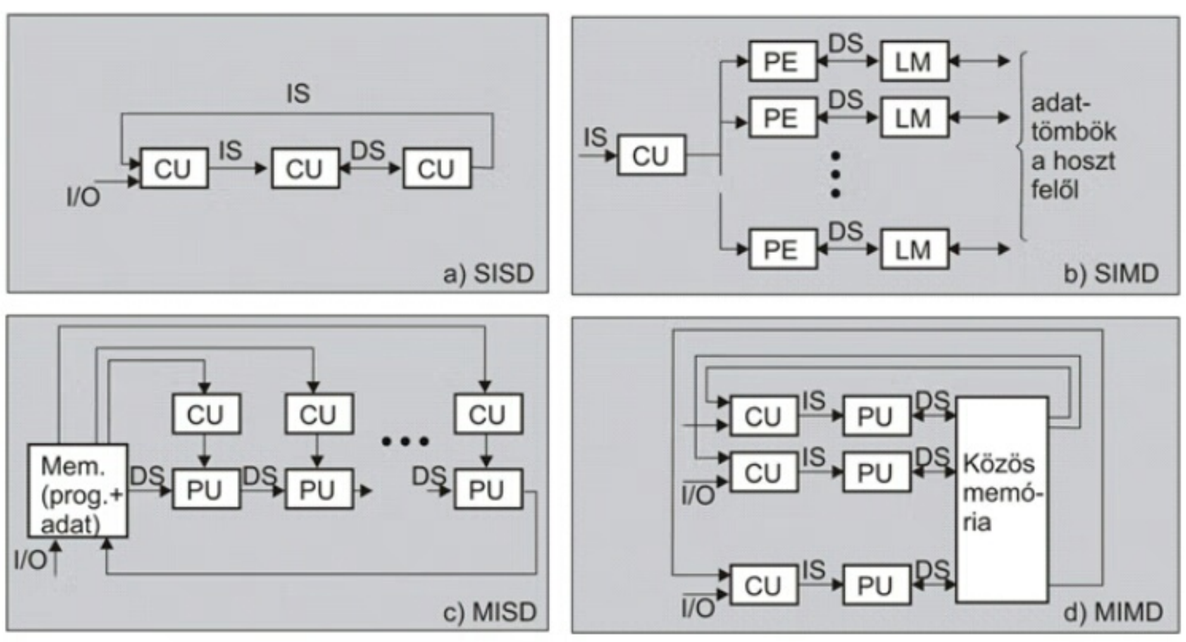
Lazán csatolt rendszerek

- A processzorok közötti kapcsolat üzenet orientált, azaz a processzorok üzenetekkel kommunikálnak egymással (message passing multiprocessor system).
  - Az információcsere kis mennyiségű és lassú az osztott memóriás rendszerekhez képest.
  - Az egyes processzorok különálló operációs rendszerek felügyelete alatt működnek.

Utasítás és adatáram jelleg

Ez a csoportosítás Flynn-től (1972) származik, s a processzorok közti utasítás- és adatáram alapján különbözteti meg a többprocesszoros rendszereket, melyek négy csoportba sorolhatók :

- SISD (Single Instruction Single Data)
- SIMD (Single Instruction Multiply Data)
- MISD (Multiply Instruction Multiply Data)
- MIMD (Multiply Instruction Multiply Data)



CU= Control Unit, PU=Processing Unit, MU=Memory Unit,
IS=Instruction Stream, DS=Data Stream,
PE=Processing Element, LM=Local Memory)

Memória megosztottság

A memória megosztottság szerint két fő csoportra oszthatók. Ezek:

– közös memóriát használó
– privát, elosztott memóriát használó

rendszerek.

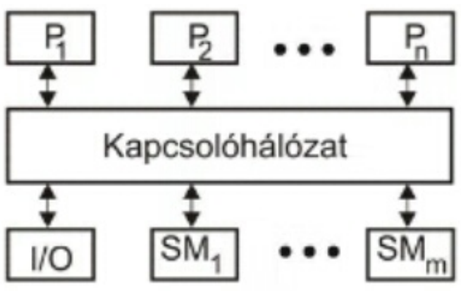
Tulajdonképpen a szorosan és lazán csatolt felosztásnak felelnek meg.

A közös memóriát használó rendszerek további három alcsoportra oszthatók. Ezek:

– UMA (Uniform Memory Access),
– NUMA (Non Uniform Memory Access)
– COMA (Cache Only Memory Architecture)

modellek.

UMA modell



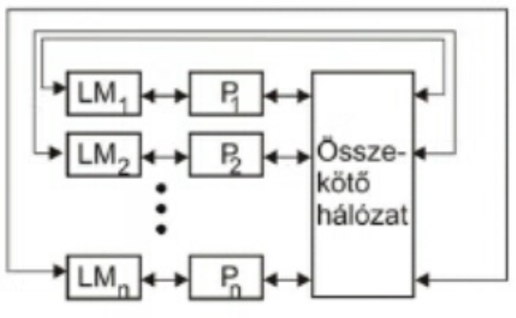
– A memória mindegyik processzor számára közös, s a hozzáférési idő mindegyik számára azonos. Az utóbbi tulajdonságból származik a modell neve. A perifériák közös használatára ugyanez igaz.
– A kapcsolóhálózat busz, crossbar vagy többfokozatú hálózat (multistage network). A párhuzamos folyamatok koordinálása, szinkronizálása és a processzorok közti kommunikáció vezérlése a közös memóriabeli megosztott változókra épül.

– Szimmetrikus UMA rendszer: az összes processzor azonos módon hozzáférhet minden memóriához és perifériához. Ekkor bármelyik processzor alkalmas arra, hogy az executive programokat (OS mag, I/O kiszolgáló rutinok, stb.) futtassa.
– Aszimmetrikus UMA rendszer: a processzoroknak csak egy részhalmaza férhet hozzá mindenhez, ezért csak ezek alkalmasak az executive programok futtatására. Ebben az esetben Mester (MP) és Attached (AP) processzorokat különböztetünk meg. De MP és AP számára a közös memória osztott használatú.

NUMA modell

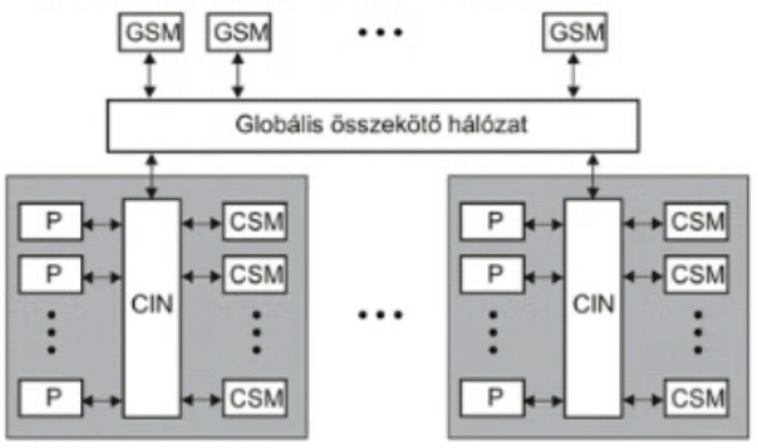
– Ebben a modellben a hozzáférési idő a hozzáférendő szó memóriabeli helyétől függ. Két altípusa van: lokális memóriájú NUMA és a hierarchikus struktúrájú NUMA.

Lokális memóriájú NUMA



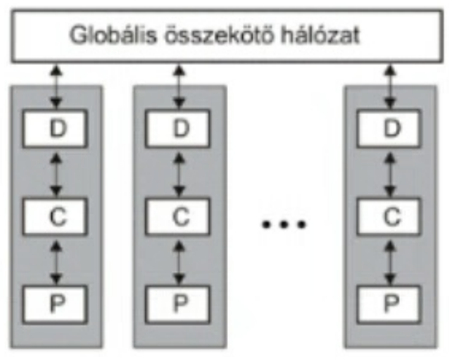
P = Processzor
LM = Lokális memória

Hierarchikus struktúrájú NUMA



GSM = Global Shared Memory (globális osztott memória)
CSM = Cluster Shared Memory (fürt osztott memória)
CIN = Cluster Interconnect Network (fürt összekötő hálózat)
P = Processzor

COMA modell



D = Directory (könyvtár)
C = Cache (gyorsítótár)
P = Processzor

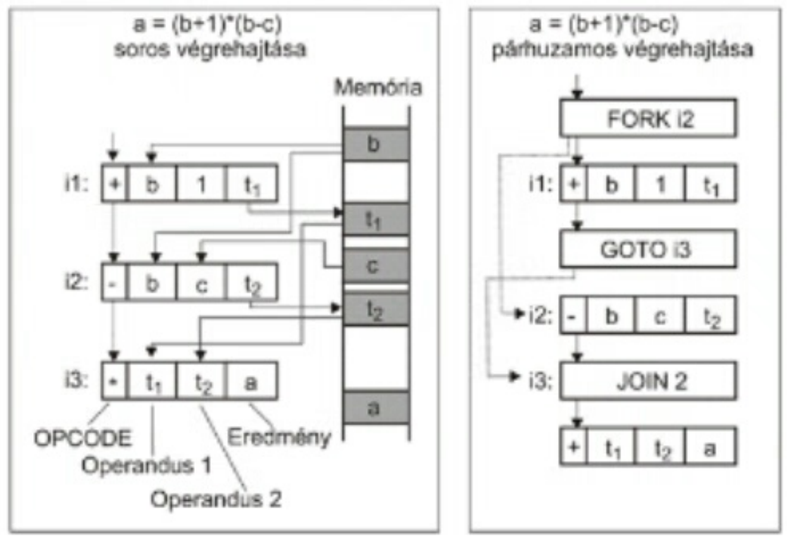
Vezérlési mechanizmus

Négy típusuk van.

Ezek:

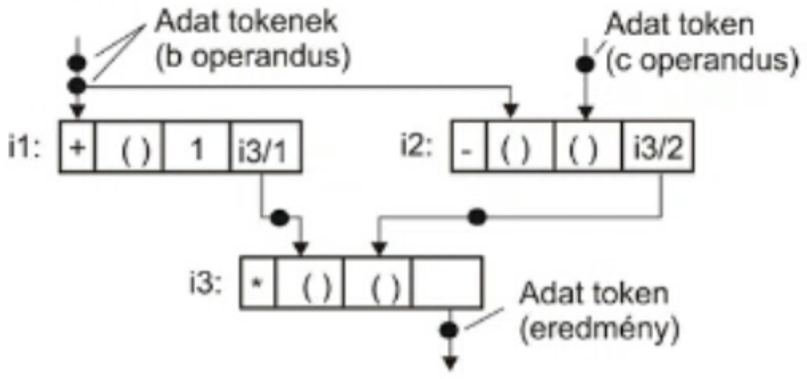
– vezérlésáramlásos (control flow)
– adatáramlásos (data flow)
– igényvezérelt (demand driven)
– mintavezérelt (pattern driven)

Vezérlésáramlásos



– A program explicite előírja az utasítások végrehajtási sorrendjét, a megvalósítás az utasításszámlálón és a közös memóriában tárolt megosztott változókon alapszik.
– A vezérlési áram az utasítások árama a memóriából a CPU-ba, s ez vezérli a végrehajtást.

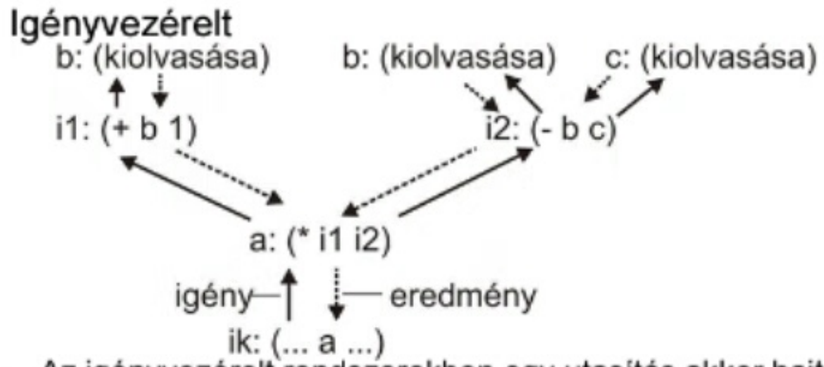
Adatáramlásos



– Ebben a modellben egy utasítás akkor és azonnal végrehajtódik, ha a végrehajtásához szükséges adatok és erőforrások rendelkezésre állnak (eager evaluation= fürge végrehajtás).

– A számítási eredmények (az ún. adat tokenek) közvetlenül az utasítások között kerülnek átadásra. Egy utasítás által generált adatnak több példánya jön létre, s kerül közvetlenül átvitelre az azt igénylő utasításokhoz. Ha egy utasítás felhasznál (elfogyaszt) egy tokent, akkor az többé már nem áll rendelkezésre más utasítások számára
– Az adatáramlásos modell nem használ utasításszámlálót és olyan változókat, melyek a globális memória celláinak felelnének meg. Azonban szüksége van egy olyan speciális mechanizmusra, amely detektálja az adat rendelkezésre állását és párosítja a tokeneket a neki megfelelő utasításokkal.
– Nevét onnan kapta, hogy az adatok áramlása vezérli a program végrehajtását. Bár teljes mértékben az adatáramlásos modellen alapuló kommerciális számítógépek még nincsennek, a sorrenden kívüli végrehajtást alkalmazó szuperskalár processzorok magja, amint már láttuk, adatáramlásos elven működik.

Igényvezérelt



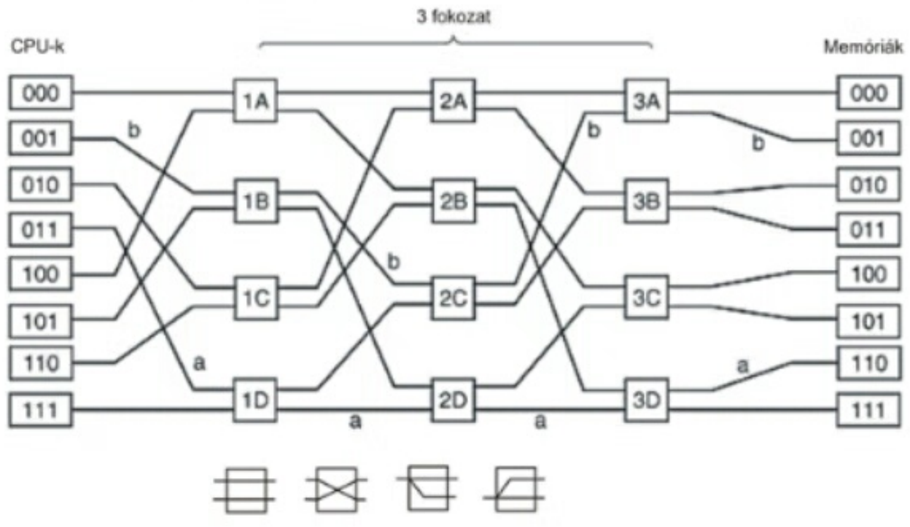
– Az igényvezérelt rendszerekben egy utasítás akkor hajtódik végre, ha eredményére szükség van (lazy evaluation=lusta végrehajtás).
– A modell a fentről-lefelé (tp-down) megközelítést alkalmazza.
– Előző példánkat tekintve először az **a** értékének meghatározást igényli, ez azonban a két zárójeles kifejezés értékét igényli, azok pedig a zárójeles kifejezésekben szereplő műveletek elvégzését. A legbelső szintre eljutva onnan fordított sorrendben visszaterjesztve az egyes műveletek eredményeit, megkapjuk az **a** értékét.

Mintavezérelt

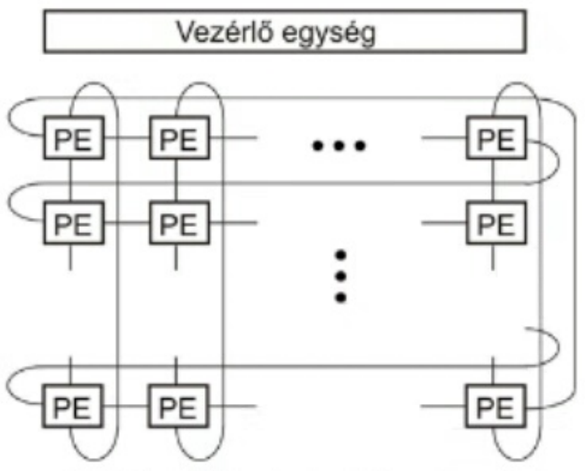
– Az adatokban fellépő minták határozzák meg, hogy milyen programrészt kell végrehajtani.
– Tipikus példa a mintavezérelt rendszerre az XSLT transzformációs nyelv. Ez ún. sablonokat (template) definiál, s a forrásállománynak a definiált sablonokra illeszkedő részein végrehajtja a sablonokban megadott feldolgozást.
– A mintavezérelt rendszert a következő egyszerű példán keresztül szemléltetjük.

Omega hálózat

– $N=2^K$ bemenet esetén K fokozatra, s fokozatonként N/2 kapcsolóegységre van szükség, kapcsolóegységenként négy kapcsolóval.
– Egy adott fokozat kimenetei a következő fokozat bemene-teire tökéletes keveréssel csatlakoznak (két csoportra osztjuk a kimeneteke és felváltva veszünk ki egyet-egyet a két csoportból). Az utolsó fokozatnál nincs keverés.



– Az Omega hálózat blokkoló hálózat, általában nem minden kombinációban kapcsolható össze minden bemenet minden kimenettel.
– Útválasztás: Az egyes kapcsolóegységek a célállomány meglévő címének legmagasabb bitjét leveszik, ha az 0 volt, akkor felül, ellenkező esetben alul hagyja el a kapcsolóegységet a jel. Visszafelé a levett és a kapcsolóegységekben tárolt bitek jelölik ki az irányt. Ha az 0 volt, akkor a felső bemenet felé, ellenkező esetben az alsó bemenet felé kell visszafelé haladni



– Kommunikációs hálózata korlátozott, minden PE csak a négy szomszédjával tudott kommunikálni. Azonban bármelyik PE hat PE-n keresztül bármelyik másik PE-vel kommunikálhat.

Alkalmazási példa

Síkbeli hőmérséklet eloszlás számítása, ha a kerületen adott a kezdeti hőmérséklet

– A hőmérsékleteloszlást a Laplace egyenlet írja le

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=0$$

– Ennek közelítő megoldása

$$\frac{\partial T}{\partial x}\approx\frac{T(x+\Delta x)-T(x)}{\Delta x}$$

$$\frac{\partial^2 T}{\partial x^2}\approx\frac{\left.\frac{\partial T}{\partial x}\right|_{x+\Delta x}-\left.\frac{\partial T}{\partial x}\right|_{x}}{\Delta x}=\frac{\frac{T(x+\Delta x)-T(x)}{\Delta x}-\frac{T(x)-T(x-\Delta x)}{\Delta x}}{\Delta x}=$$

$$\frac{T(x+\Delta x)-2T(x)+T(x-\Delta x)}{\Delta x^2}$$

– A fentieket az y szerinti második parciális deriváltra is elvégezve kapjuk:

$$T(x)\approx\frac{1}{4}\left[T(x+\Delta x,y)+T(x-\Delta x,y)+T(x,y+\Delta y)+T(x,y-\Delta y)\right]$$

$$T=\frac{1}{4}[T_N+T_S+T_E+T_W]$$

– Azaz egy pont hőmérséklete a tőle északra (North), délre (South), keletre (East) és nyugatra (West) fekvő szomszédos pontok hőmérsékleteinek átlaga. Ha a kerület hőmérséklete ismert, akkor a fenti képletet használva, s a belső pontoknak valamilyen kezdeti hőmérsékletértéket adva, sorozatos iterációval egyre pontosabban meghatározható a belső pontok hőmérséklete.
– A fenti példa jól mutatja, hogy az SIMD rendszerek nem általános célúak, hanem sokkal inkább speciális feladatok megoldására előnyösek. Tipikus területek: differenciál-egyenletekkel leírható feladatok, képfeldolgozás, stb.

Gyorsítótár koherencia

Mi a gyorsítótár koherencia?

- A többprocesszoros rendszerekben a processzorok egy vagy kétszintű gyorsítótárat is tartalmaznak.
- Ez előnyös a nagyobb teljesítőképesség elérése céljából, de felvet egy nehézséget az ún. gyorsítótár koherencia vagy konzisztencia problémáját.
- Ennek lényege: Ugyanazon adat több másolata is egyidejűleg létezhet több gyorsítótárban, s ha egy processzor szabadon módosíthatja saját másolatát, akkor a memóriában és a többi gyorsítótárban hibás lesz az adat. Más szóval inkonzisztens lesz az adat a gyorsítótárban és a memóriában.

- A gyorsítótár írása kétféleképpen történhet:
  - Késleltetett írással (Write back): Az írás rendszerint csak a gyorsítótárba történik, s az operatív memória frissítése csak akkor következik be, ha a sort ki kell üríteni.
  - Írásáteresztéssel (Write through): Minden írás a gyorsítótárba és a memóriába is megtörténik, ami biztosítja, hogy az operatív memória mindig konzisztens.
- Késleltetett írású módszer inkonzisztenciát okoz. Ha egy adat két gyorsítótárban is benn van, s az egyik frissítődik, akkor a másik gyorsítótár „nem fogja tudni", hogy a benne lévő adat érvénytelen. De még az írásáteresztéses módszernél is fellép az inkonzisztencia, hacsak a gyorsítótárak nem figyelik a memóriaforgalmat vagy nem kapnak közvetlen értesítést az adat megváltoztatásáról.
- A koherencia szoftver és hardver úton egyaránt fenntartható.

Szoftver megoldások

- Járulékos hardver nélkül a fordítóprogram és az operációs rendszer segítségével kezelik a problémát.
- Attraktív, mivel a potenciális probléma detektálása a futási idő helyett a fordítási időbe, s a komplexitás a hardverből a szoftverbe tevődik át.
- Azonban a szoftver megoldásoknak általában konzervatív döntéseket kell hozniuk, ami a kód kevésbé hatékony kihasználásához vezet.
- A fordítóprogram alapú megoldások analizálják a programot annak meghatározása céljából, hogy mely adatok gyorsítótárba helyezése nem biztonságos, s ezeket az adatokat megjelölik. S az operációs rendszer vagy a hardver megakadályozza a megjelölt adatok gyorsítótárba helyezését.

- A legegyszerűbb megoldás minden megosztott (közös használatú) adat gyorsítótárba helyezésének tiltása. Ez túlságosan konzervatív, mivel a megosztott adatstruktúrát gyakran egy bizonyos időperiódusokban esetleg csak egyetlen processzor használja.
- Hatékonyabb megközelítések analizálják a programot, s meghatározzák a megosztott adatok biztonságos periódusait. A fordítóprogram a generált tárgyprogramba olyan utasításokat helyez el, melyek biztosítják a koherenciát a kritikus időszakokban.

- Tipikusan a központi vezérlő tárolja azt az információt, hogy melyik processzor mely sorok másolatával rendelkezik.
- Ha egy processzor írni akarja a sort, akkor kizárólagosságot kell kérnie rá a központi vezérlőtől.
- Mielőtt a központi vezérlő ezt megadná, értesíti a sor érvénytelenségéről azokat a processzorokat, amelyek gyorsítótárukban tartalmazzák ennek a sornak a másolatát.
- Mikor a központi vezérlő visszakapta a sor érvénytelenítésének nyugtázását ezektől a processzoroktól, akkor megadja a kizárólagosságot az adott sorra az azt kért processzornak.
- Ha egy processzor egy más processzornak kizárólagosan garantált sort akar olvasni, akkor az olvasási hibáról értesíti a vezérlőt.
- Ekkor a vezérlő utasítja a sort kizárólagos joggal használó processzort, hogy írja vissza azt a központi memóriába. Ezután ez a sor olvasásra megosztottan használható.

- Hátránya enne a megoldásnak, hogy központilag szűk keresztmetszetű lehet és jelentős időveszteség lép fel a központi és a lokális vezérlők közötti kommunikáció miatt.
- Nagy rendszerekben, melyek több buszt vagy más komplex kommunikációs kapcsolatot használnak, viszont hatékonyan alkalmazható.

Hardver megoldások

- Futási időben észlelik a potenciális inkonzisztens állapotokat.
- Mivel ezek a problémával csak akkor foglalkoznak, amikor az fellép, a gyorsítótár hatékonysága sokkal nagyobb mint a szoftver megoldásokban.
- Ráadásul ezek a megoldások transzparensek a programozó és a fordítóprogram számára, ezért csökkentik a szoftverfejlesztésre nehezedő nyomást.
- Sokféle hardver megoldás ismert, ezek azonban két fő osztályba sorolhatók.
  - Az egyik osztály a könyvtár alapú (directory) protokollok osztálya,
  - a másik pedig a szimatoló protokollok osztálya.

Könyvtár alapú protokollok

- Ezek a protokollok begyűjtik és tárolják, hogy az egyes gyorsítótár soroknak mely gyorsítótárakban van másolata.
- Tipikusan egy központi vezérlőt (amely a memóriavezérlő része) és a központi memóriában tárolt könyvtárat tartalmaznak.
- A könyvtár globális állapot információt tartalmaz a lokális gyorstótárakról.
- Amikor egy lokális gyorsítótár vezérlő kérelmet küld, a központi vezérlő kiadja a megfelelő parancsokat a GYT és a memória vagy a GYT-ak közötti adatátvitelre, s frissíti az állapot információt. Ezért minden olyan lokális változásról, amely befolyásolhatja egy sor globális állapotát, értesíteni kell a központi vezérlőt.

Szimatoló protokollok

- Ezek a protokollok szétosztják a felelősséget a GYT koherencia fenntartására a egyes processzorok GYT vezérlői között.
- A GYT-nak fel kell ismernie, hogy egy sora megosztott-e. S amikor egy megosztott soron frissítés lép fel, akkor azt a többi gyorsítótárral is közölni kell „broadcast" jelleggel.
- Minden GYT vezérlőnek szimatolnia kell a buszon, hogy a broadcast jelzésről tudomást szerezzen, s arra megfelelően reagáljon.
- A szimatoló protokollok ideálisak a busz alapú többprocesszoros rendszerekben, mivel a közös használatú busz egyszerű eszközt nyújt a broadcastingra és a szimatolásra
- Kétféle megközelítés:
  - írás érvénytelenítés
  - írás frissítés (vagy írás broadcast).

Írás érvénytelenítés

- Több olvasója, de csak egy írója lehet egy sornak.
- Indulásképen a sor megosztott lehet olvasásra.
- Ha egy GYT írni akarja a sort, akkor először kiad egy érvénytelenítést a sorra a többi GYT felé, ezáltal kizárólagossá teszi a sort saját GYT-ában.
- Ezután mindaddig írhatja, amíg egy másik GYT nem kéri ugyanezt a sort.

Írásfrissítés

- Több írója és olvasója is lehet egy sornak.
- Amikor egy processzor frissíteni akar egy sort, a frissítendő szót a többi GYT-nak is elküldi, s az illető szót tartalmazó GTY-ak is frissítik a szót tartalmazó sort.

Egylapkás rendszerek

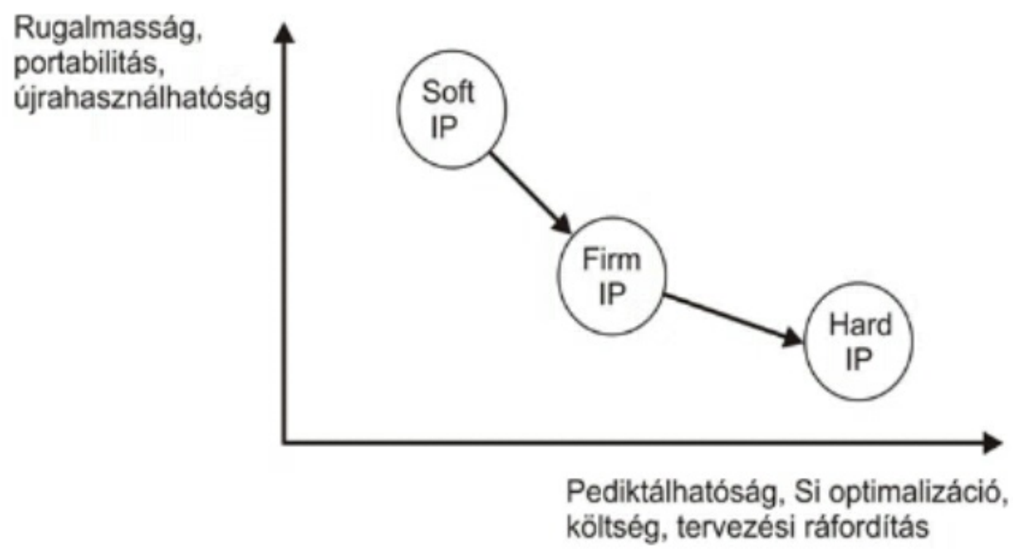
Moore törvény és következménye a digitális tervezésre. Az egylapkás rendszerek definíciója. Újrahasználható IP-k: digitális, szoft, hard, firm, valamint analóg/vegyes jelű IP blokkok. Programozható IP: HW és SW programozhatóság. IP licensz faiták. Kommunikációs infrastruktúrák: busz alapú és socket alapú megoldások. Az OCP socket jellemzői és jelei.

- Egylapkás rendszer: előre megtervezett és verifikált blokkokból építjük fel. Többféle nevük: Intellectual Property (IP) blokkok, IP magok, virtuális komponensek.
- Egyetlen lapkára integrál
  - *processzorokat,*
  - *memória modulokat,*
  - *I/O perifériákat ,*
  - *felhasználás-specifikus funkciókat,*
  - *analóg- , digitális és sokszor RF funkciókat,*
  - *szoftvert,*
  - *újrahasznosítható szellemi tulajdonok (HW+SW).*
- Az FPGA eszközök kapacitásának állandó növekedésével ezek az eszközök is alkalmasak lettek az SoC megvalósítására:
  - *SoPC /System on Programmable Chip/ vagy*
  - *PSoC /Programmable System on Chip/*
- Definíció:

(Csaknem) komplett rendszer integrálása egyetlen csipen

Digitális IP magok

- Az IP magok aktuális formái: soft-, hard- és firm IP bkokkok
- Jellemzőik:



- Drasztikus termelékenység-növekedés
  - Blokkok megtervezése helyett csak integrálásuk a feladat
  - Design For test (DFT) technikák implementálása
  - Rendszer szintű validáció és verifikáció
    - Validáció: annsk ellenőrzése, hogy a rendszer eleget tesz a felhasználó által támasztott igényeknek
    - Verifikáció: annak ellenőrzése, hogy a rendszer eleget tesz-e a specifikációnak

Soft IP magok

- Specifikálásuk RTL vagy magasabb szintű leírással
- Digitális esetben alkalmasabbak, mivel HDL technológiától független és kapuszintig optimalizálható
- Előnyök: rugalmasság, portabilitás, újrahaszálhatóság
- Hátrányok: Időzítések és tápellátási jellemzők nem garantáltak
  - Különböző technológiákban megvalósítva különböző minőség
  - HDL target sokszor titkosított formában adott, nem lehet módosítani
    - Ez megnehezíti az eljárást, de megakadályozza a tervezési hibák bevitelét a blokkba

Újrahsználható IP

- SoC legfontosabb feltétele: újrahasználható IP blokkok rendelkezésre állása → Plug-and-Play integráció (helyezd be és működik)
- IP bokkok: SoC legmagasabb szintű építőelemei
- Könyvtárak: különböző időzítésű, lapkaterületű és teljesítmény-konfigurációs IP blokkok
- Fajtáik:
  - digitális IP blokkok
    - soft IP blokkok
    - hard IP blokkok
    - firm IP blokkok
  - analóg/vegyes jelű blokkok
  - programozható IP blokkok

Hard IP magok

- Fix layouttal rendelkeznek
- Optimalizáltak egy adott alkalmazásra egy specifikus technológiai eljárásban
- Előnyük: megjósolhatók az időzítési, fogyasztási, stb. paramétereik
- Előre kvalifikáltak: létrehozójuk legyártott lapkán tesztelte őket

Digitális IP blokkok

- Napjainkban ezek a legelterjedtebbek
- Tervezési folyamatuk:
  - Specifikálás és dokumentáció
  - Implementálás szabványos módszerekkel
  - Teljes mértékű verifikáció
- Az 1. fázis a megfelelő dokumentáció generálása
- A 2. fázis kódtervezés, szintézis és a tesztelés tervezése
- A 3. fázis több mint 50%-át jelentheti a teljes tervezési folyamatnak. Mivel az IP-t sokszor felhasználhatják a legkülönbözőbb alkalmazásokban, nem tolerálhatók a hibák a tervezésben. A verifikáció célja a kód vonatkozásában a 100%-os, a funkcionalitás vonatkozásában pedig a közel 100%-os hibamentesség elérése.

Firm IP magok

- Parametrizált áramkörin leírással adottak, a tervező céljainak megfelelően optimalizálhatja őket
- A soft IP magokhoz képet paramétereik jobban megjósólhatók
- Kompromisszumot jelentik a soft- és hard IP magok között

Analóg/vegyes jelű blokkok

- A digitális tervezés jól definiált top-down módszerekkel elvégezhető, de az analóg/vegyes jelű (analog/mixed-signal, a továbbiakban AMS) tervezés tradicionálisan ad hoc folyamat.
- Ha analóg és digitális részek egyaránt jelen vannak ugyanazon a hordozón, akkor az analóg rész tervezése jóval időigényesebb lehet, még ha jóval kisebb területet is foglal el a hordozón, mint a digitális rész.
- Ugyanakkor a vizsgálatok azt mutatják, hogy az SoC-k több mint 70 % tartalmaz AMS blokkokat.

Analóg/vegyes jelű blokkok

- A digitális tervezés jól definiált top-down módszerekkel elvégezhető, de az analóg/vegyes jelű (analog/mixed-signal, a továbbiakban AMS) tervezés tradicionálisan ad hoc folyamat.
- Ha analóg és digitális részek egyaránt jelen vannak ugyanazon a hordozón, akkor az analóg rész tervezése jóval időigényesebb lehet, még ha jóval kisebb területet is foglal el a hordozón, mint a digitális rész.
- Ugyanakkor a vizsgálatok azt mutatják, hogy az SoC-k több mint 70 % tartalmaz AMS blokkokat.
- Tipikus AMS komponensek: műveleti erősítők, AD és DA átalakítók, fáziszárt hurkok (PLL), soros-párhuzamos adó-vevők, szűrők, feszültség referenciák, rádiófrekvenciás modulok, feszültségszabályozók, analóg komparátorok, stb. Nagy részük hard IP formában érhető el, s egy konkrét gyártás-technológiához kapcsolódik. Ezért nehéz más alkalmazáshoz vagy gyártástechnológiák esetén ezeket felhasználni.

Programozható IP blokkok

- SoC-n belüli programozhatóság, azaz lapka szintű újrahasználhatóság
- Két formája:
  - hardver programozhatóság programozható logikai magok használatával,
  - szoftver programozhatóság beágyazott processzorok használatával.
- A programozhatóság célja rugalmas infrastruktúra biztosítása.
- A rugalmasság két formáját különböztetjük meg:
  - gyártás előttit (prefabrication) és
  - gyártás utánit (postfabrication).

- Gyártás előtti programozhatóság
  - Strukturált ASIC-on alapszik.
  - A strukturált ASIC alsó layout szintjei rögzítettek, s a lapka már részben legyártott.
  - A programozás a felső réteg és az összekötő réteg módosításával történik a végső gyártás előtt.
  - Mivel csak néhány lépést kell végrehajtani a gyártás befejezéséhez, az átfutási idő jelentősen lerövidül.
  - De a lapka nem módosítható a gyártás befejeztével.

- Gyártás utáni programozhatóság
  - Lehetővé teszi az IP blokk módosítását annak legyártása után.
  - A módosítás egy beágyazott processzoron futó szoftver/firmver átírásával (szoftver konfigurálhatóság) vagy a beágyazott programozható hardver átkonfigurálásával (hardver programozhatóság) történhet.
  - Nagy előny, hogy az IC viselkedése, funkciója az igények megváltozásával módosítható (pl. kommunikációs szabványok vagy felhasználói igények változásával).
  - Nincs szükség új IC kifejlesztésére, ami tipikusan több hónapot és jelentős költségeket igényelne.
  - Viszont hátrányként jelentkezik a nagyobb fogyasztás, kisebb sűrűség és a kisebb sebesség.

- Hardver programozhatóság
  - Egy vagy több programozható logikai mag biztosítja a lapkán.
  - A programozható magok nagyon hasonlóak az FPGA eszközökhöz.
  - A gyártás folyamán integrálódnak be az eszközbe és „elkötelezetlen" kapukat, billenőköröket és programozható kapcsolatokat tartalmaznak.
  - A programozást kiváltó bitek többféleképpen tárolhatók.
    - Statikus RAM elemeket használata, ekkor az SoC többször átprogramozható.
    - Másik megoldás esetén a strukturált ASIC-ban a gyártás utolsó fázisaiban a konfigurálható összekötéseket az igényeknek megfelelően vagy létrehozzák vagy nem.

  - Szoftver programozhatóság
    - A beágyazott szoftver lehetővé teszi, hogy egyetlen SoC alkalmas legyen különböző feladatokra.
    - Minél több funkció megoldható szoftverrel, annál rugalmasabbá tehető az SoC.
    - A komplex vezérlési funkciók implementálására a szoftver az alkalmasabb, az adatfeldolgozó műveletekre inkább a hardver.
    - Ha a gyorsaság alapvető szempont, akkor hardvert kell használnunk.
    - A használt hardver akár programozható logikai mag lehet, ennek sebessége kb. ötszöröse a szoftveres megoldás sebességének. ASIC használata esetén viszont tipikusan 50-szeres sebesség érhető el a szoftver megoldáshoz képest.
    - A szoftver újrahasználatához kódkönyvtárak, adatstruktúrák, operációs rendszer kernelek és real-time operációs rendszerek (pl. RTOS) állnak rendelkezésre.

IP licensz fajták

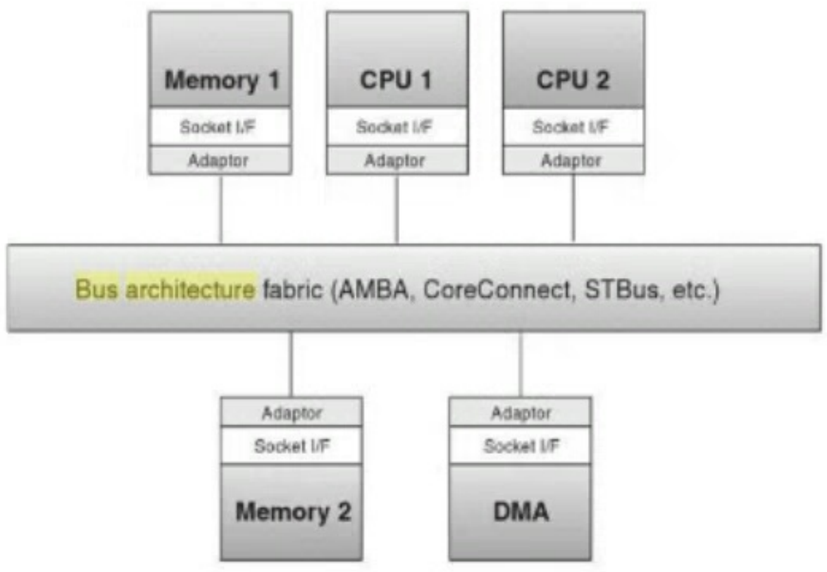
- *Perpetual (határidő nélküli) implementációs licensz.* A licensz vásárlója határidő nélkül tervezheti és gyárthatja a vásárolt mag alapú eszközeit.
- *Term (határidős) licensz.* Korlátozott számú termék tervezése rögzített időn belül (ált. 3 év), de határidő nélküli gyártás.
- *Per use licensz.* Az adott IP-vel egyetlen termék tervezhető rögzített időn belül (ált. 3 év), de a gyártási idő határidő nélküli.
- *University „DesignStart" program.* Bizonyos fizikai és egyes processzor IP-k akadémiai célokra ingyenesen letölthetők.

Szabványos busz alapú kommunikációs architektúrák

- ARM Microcontroller Bus Architecture (AMBA 2.0, AMBA 3.0 – AXI)
- IBM CoreConnect
- STMicroelectronics STBus
- Sonics SMART Interconnect
- Wishbone OpenCores
- Altera Avalon, stb.

- A busz alapú kommunikációs architektúrák részleteivel itt nem foglalkozunk, egyik képviselőjüknek az AXI busznak a részletesebb tárgyalására később térünk ki.

Socket alapú kommunikációs architektúra



- Csak azokat az interfészeket specifikálják, melyekkel a komponesek a busz alapú kommunikációs architektúrához csatlakozhatnak.
- Nem specifikálják a busz architekturális implementációját, csak az interfészt specifikálják.
- A feldolgozási komponenst teljes mértékben szétcsatolják a kommunikációs komponenstől és annak implementációjától.
- A komponensek szabványos interfésszel tervezhetők, anélkül hogy egy adott kommunikációs architektúra implementációhoz illeszkedni kellene. Ez nagymértékben növeli az IP-k újrahaszálhatóságát.
- Így csak egyetlen követelménynek kell eleget tenni: rendelkezésre kell állni a célbuszon a socket adapternek, amelyik a komponens interfészt leképezi a busz protokolljára.

Az OCP socket

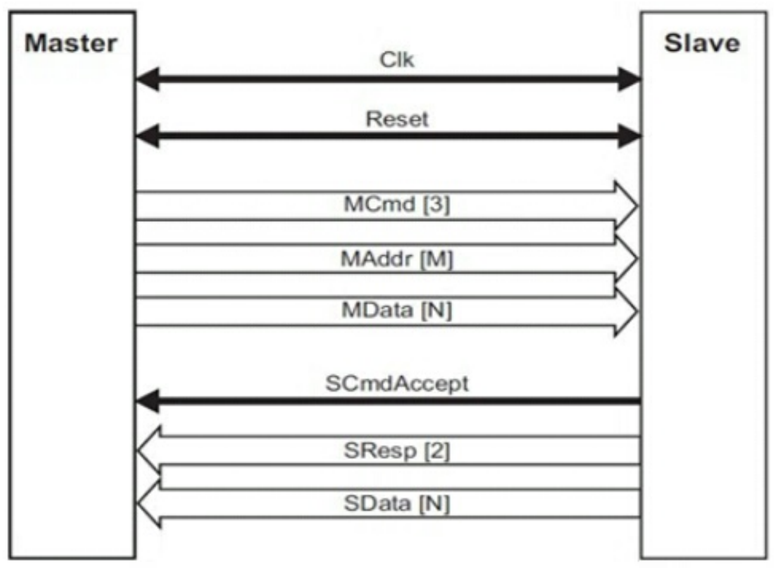
- Az OPC (Open Socket Architecture) a mai nyílt szabványú socketek egyik legjobb példája.
- Nagymértékben parametrizált, konfigurálható és bővíthető, továbbá támogatására hatékony tervezési és verifikációs eszközök állnak rendelkezésre.
- Egy szabványos mag interfész definiálása a 90-es évek elején indult el a VSI Alliance létrhozásával.
- Definiálta a Virtual Component Interface-t (VCI), mint az IP mag interfészt. Bár ez igen értékes lépés volt, erőfeszítései nem eredményeztek egy széles körben elfogadott szabványt.

- Az OCP socket egy komplett specifikáció, amely a mag kommunikációs követelményeknek, a busz struktúráknak és a kapcsolóhálózati technológiáknak széles skáláját képes támogatni.
- Az OCP socket buszfüggetlen és kezeli az SoC három domináns kommunikációs típusát, az adat-, vezérlés- és teszt kommunikációt. Kiterjedt tervezési, validációs és implementációs eszköztámogatása van, továbbá támogatja a SystemC tranzakciós interfészt.
- Az OCP socket a követelményeknek megfelelően konfigurálható (egyszerű, kisteljesítményű mag interfésztől a fejlett processzor és memória számára sokkal összetettebb és nagyobb teljesítményű interfészig).
- Az individuális magtervezők az OCP használatával függetlenül tervezhetik funkcionális IP magjaikat.
- Az OCP protokoll specifikáció szerint az OCP socket mester-szolga elvet használ egyirányú szinkron jelekkel, melyek mintavétele az OCP órajel felfutó élénél történik.

- Az OCP az adatáramlásban az átviteleket explicite fázisokra osztva valósítja meg.
- A request (kérelem) fázisban a mester ellátja a szolgát az átvitel végrehajtásához szükséges információval.
- Egyszerű írás esetén a kérelem fázis az adatátvitelt is elvégzi.
- A response (válasz) fázisban a szolga a transzfer befejezéséről értesíti a mestert.
- Olvasás esetén a válasz fázis tartalmazza az adatátvitelt.
- Ezek a fázisok egymáshoz képest csővezetékszerűen működhetnek, azaz a mester több kérelmem fázist is elindíthat, mielőtt a szolga válaszolna.
- A kézfogásos és folyamvezérlő jelek széles csoportja lehetővé teszi mindkét oldal számára az átvitel ütemezésének és a csővezeték mélységének kezelését.

Az alapvető OCP jelek

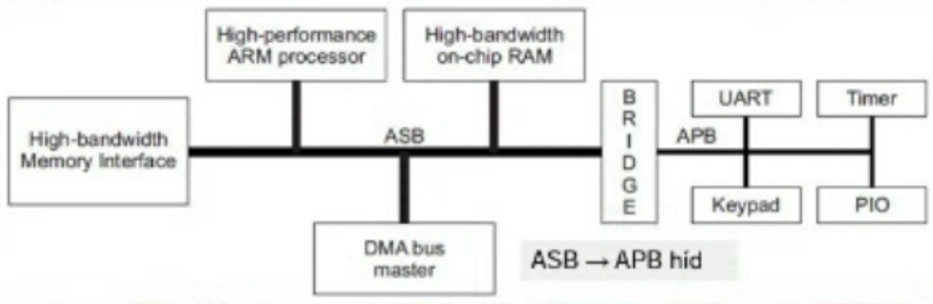
- mester parancs-, cím- és írásadat jelek (MCmd = MasterCommand és MAddr = MasterAddress, MData = Masterdata)
- szolga kézfogásos-, válasz- és olvasási adat jelek (SCmdAccept, SResp és Sdata)
- órajel (Clk) és reset (Reset_N).



AMBA – AXI

AMBA változatok. Az AHB, ASB és APB, s a velük felépített tipikus rendszer. Az AXI általános jellemzése: az öt csatorna, az ID-vel azonosított átvitelek, az outstanding request. Tipikus AXI rendszer. Az AXI protokoll. Fontosabb jelek. Tranzakciók: Valid-Ready kézfogás esetei, egy négy olvasási átvitelt tartalmazó olvasási löket idődiagramja, egy írási idődiagram.

ASB és APB



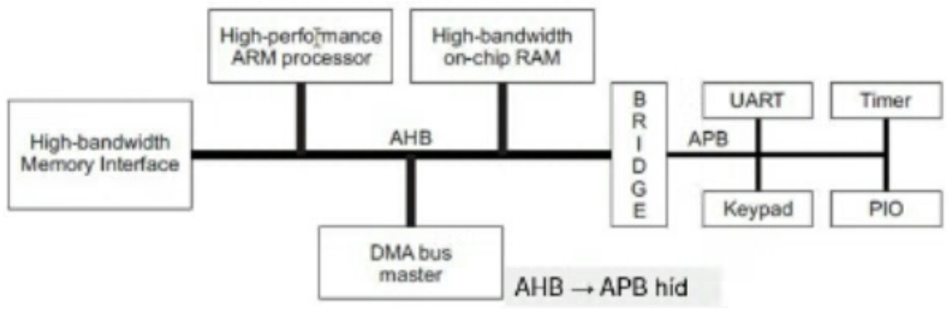
ASB
- Nagy teljesítmény
- Csővezetékes végrehajtás
- Burst (löket) átvitel
- Több buszmester

APB
- Kis fogyasztás
- Egyszerű interfész
- Sok periféria esetén használható

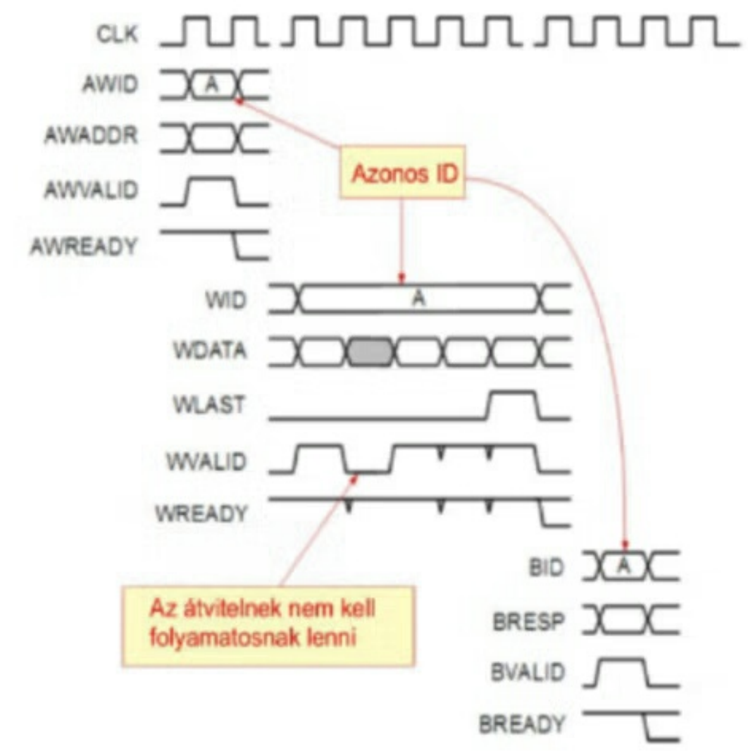
Tranzakció orientált átvitel

- Cím-, adat- és válaszátvitelek
- A tranzakciók ID[3:0] jelekkel azonosítják magukat
- Minden transzfer szintén ID[3:0]-vel azonosítja magát a tranzakcióhoz
- Így sorrenden kívüli befejeződés és interlív lehetséges
- A transzferek a célállomáson sorrendbe rendezhetők az ID alapján
- Lásd a következő ábrát

AHB



AHB
- Nagy teljesítmény
- Csővezetékes végrehajtás
- Burst (löket) átvitel
- Több buszmester
- Osztott tranzakció

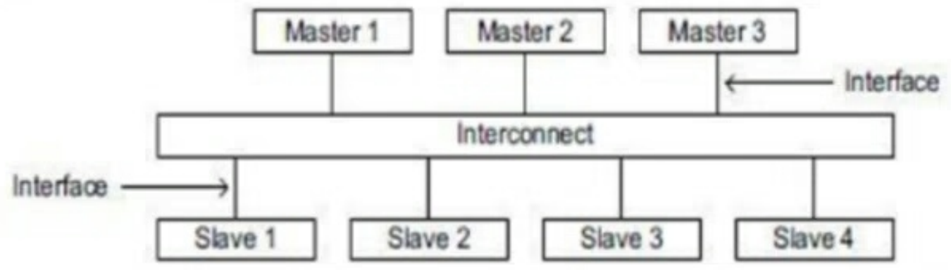


- Az AXI protokoll löket alapú és 5 független csatornát definiál:
  - Read address csatorna és Write address csatorna
    - Címet és vezérlési információt visz át
  - Read data csatorna
    - Adatokat és válaszinformációt visz át
    - Buszszélesség: 8,16, 32, 64, 128, 256, 512, 1024
  - Write data csatorna
    - Buszszélesség: 8,16, 32, 64, 128, 256, 512, 1024
  - Write response channel
    - Írási válaszinfo

~ Write address

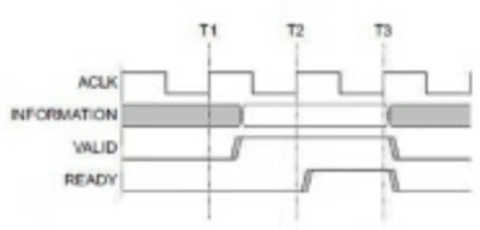
Tipikus rendszer

Részei:
- Mesterek
- Szolgák
- Kapcsolóhálózat (interconnect)

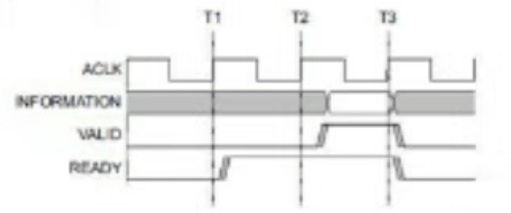


Tranzakciók

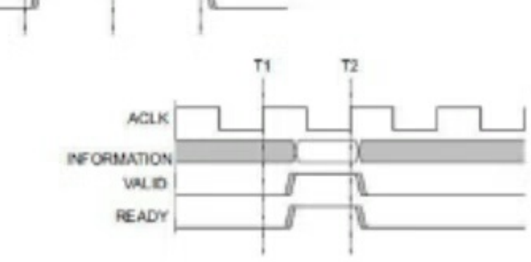
A VALID és READY kézfogásos jelek időzítési változatai



Valid Ready előtt

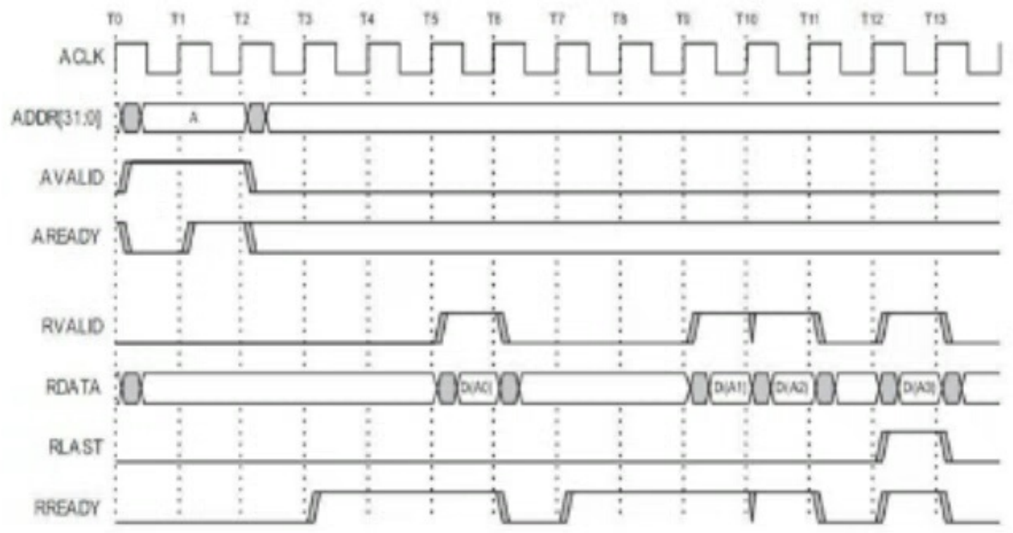


Ready Valid előtt



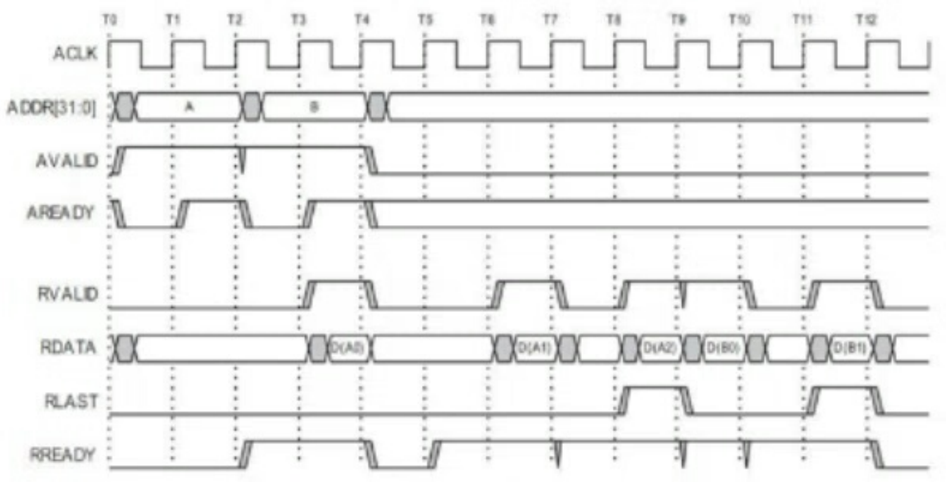
Valid és Ready együtt

Olvasás 1.



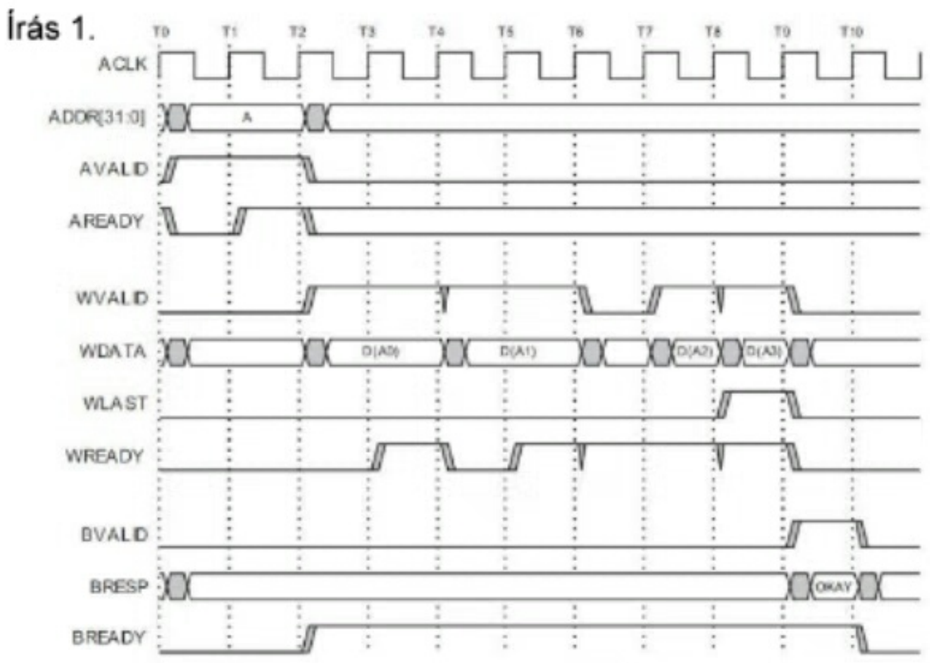
ADDR == AWADDR, A VALID == ARVALID és A READY == ARREADY

Olvasás 2.



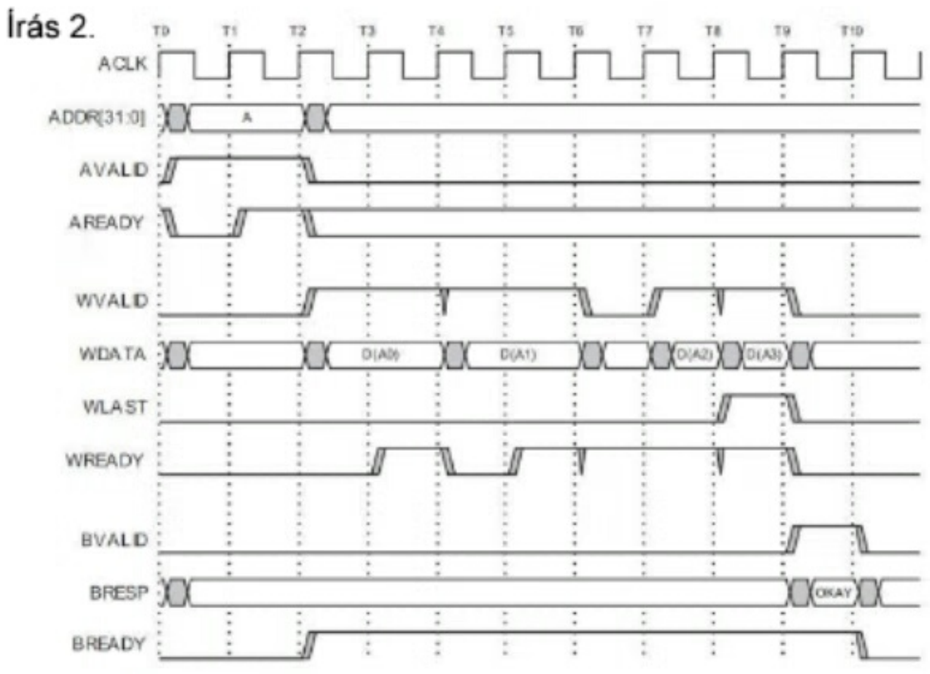
ADDR == AWADDR, A VALID == ARVALID és A READY == ARREADY

Írás 1.



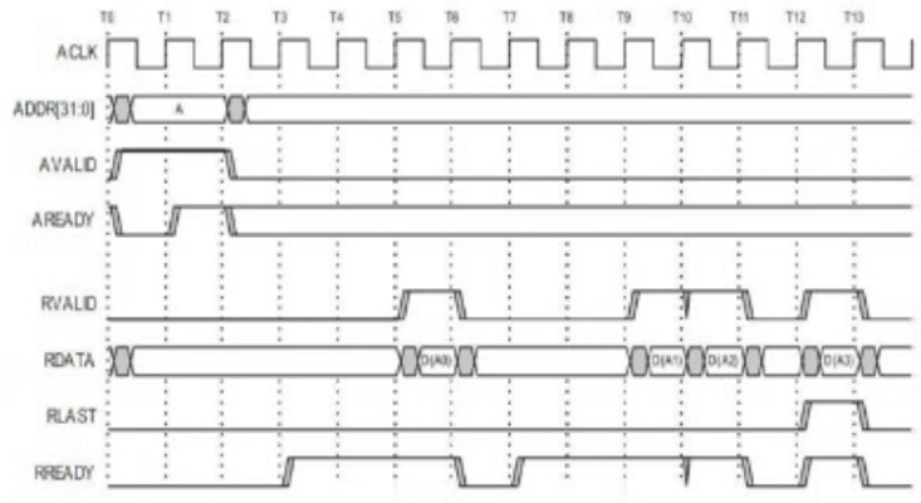
ADDR == AWADDR, A VALID == AWVALID és A READY == AWREADY

Írás 2.



ADDR == AWADDR, A VALID == AWVALID és A READY == AWREADY

Olvasás 3.



ADDR == AWADDR, A VALID == ARVALID és A READY == ARREADY

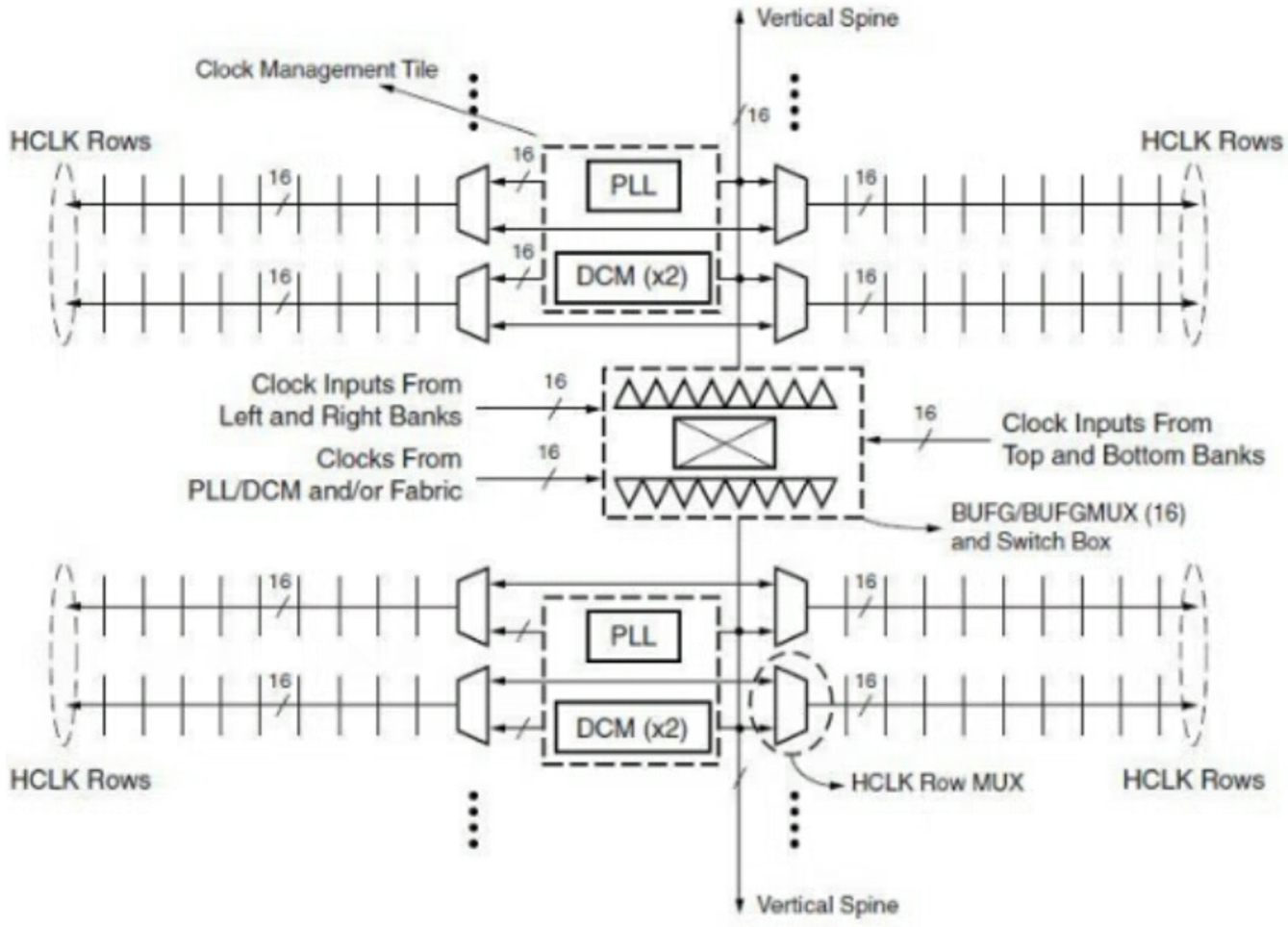
Xilinx FPGA családok
Spartan-6 órajelhálózat és a PLL egység felépítése, a kimenő órajel számítása. A CLB, IO, Block RAM és DSP szelet. A konfiguráció lépései. A PicoBlaze blokkdiagramja. A MikroBlaze fix és opcionális elemei. A MicroBlaze MCS blokkdiagramja.

1.1 Spartan-6

A következő fejezetekben a Spartan-6 FPGA-kban található erőforrásokat mutatom be.

1.1.1 Órajelhálózat felépítése

Az órajelhálózat CMT-kből (Clock Manegement Tile) és az elosztó hálózatból tevődik össze. Minden CMT egy PLL-t és 2 DCM-et (Digital Clock Manager) tartalmaz. A hálózat blokkvázlata a következő ábrán látható:



Fizikailag az FPGA közepén található egy kapcsoló egység, amely szétosztja az FPGA erőforrásai felé különböző órajeleket. Ezt a kapcsoló egységet három órajelforrás hajthatja meg:

- Órajelek a felső vagy az alsó bankokból
- Órajelek a bal vagy a jobb oldali bankokból
- Órajelek a PLL-ből, DCM-ből vagy valamilyen belső logikai jelről

A kapcsoló egység bemeneteire csak azok az I/O-k köthetőek rá, melyeket globális órajelbemeneteknek nevezünk. Egy FPGA-ban összesen 32 ilyen I/O található.

A kapcsoló egységek közvetlenül az órajelrendszer függőleges „gerincét" hajtják meg, amihez vízszintesen csatlakoznak a HCLK sorok (Horizontal Clock), melyek a helyi logikákat látják el órajellel, ahogy az alábbi ábrán is látható.

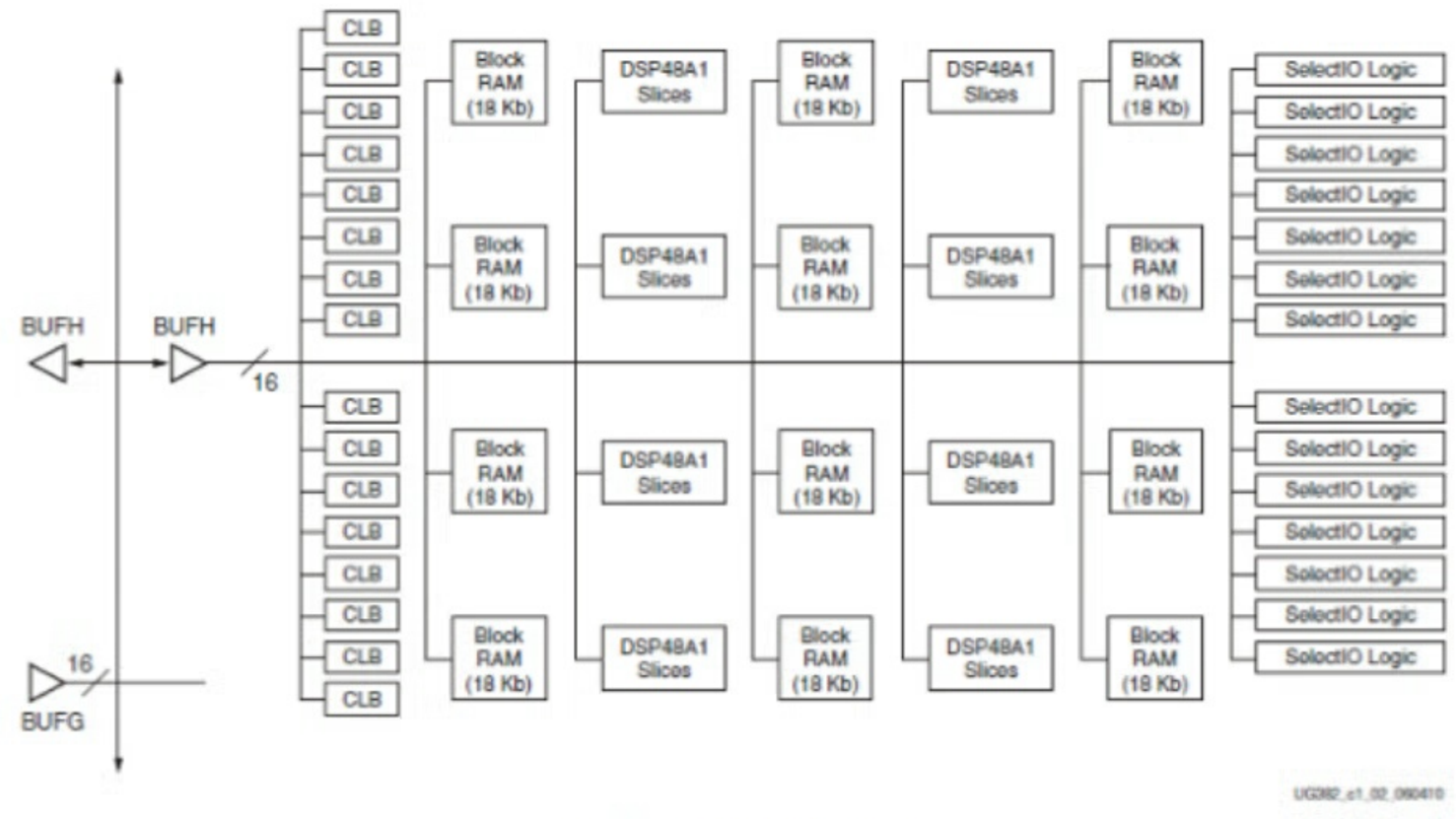


Figure 1-2: **BUFH Routing**

A globális órajelhálózat sajátossága, hogy alacsony késleltetést biztosít az FPGA logikai erőforrásaihoz, ezáltal biztosítva a lehető legmagasabb működési frekvenciát.

1.1.2 PLL

A Spartan-6 család tagjai 2..6 PLL egységgel rendelkeznek. Egy PLL egység blokkvázlata az alábbi ábrán látható.

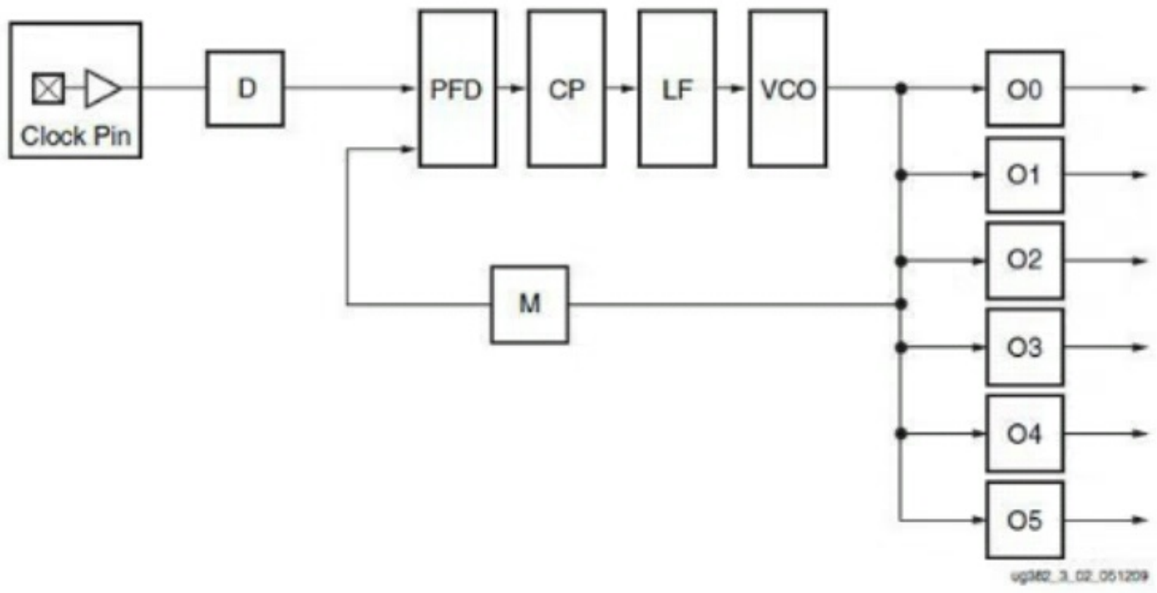


Figure 3-2: **Block Diagram of the Spartan-6 FPGA PLL**

Egy digitális PLL-ről van szó melynek működési elve a következő. A bemenő órajelet egy D programozható előosztó osztja le. A PFD (Phase-Freqency Detector) ennek a leosztott órajelnek a frekvenciáját, ill. fázisát hasonlítja össze a visszacsatolt órajelhez képest, és a különbséggel arányos jelet állít elő a kimenetén, amely a CP-n (Charge Pump) és az LF-en (Loop Filter) keresztül a VCO-t (Voltage Controlled Oscillator) hajtja meg. A VCO a bemeneti feszültséggel arányos frekvenciájú órajelet állít elő, amely az M osztón keresztül vissza van csatolva a PFD bemenetére.

A bemenő és a kimenő órajelek közötti összefüggés az alábbi:

$$f_{OUT} = \frac{f_{IN} \times M}{D \times Ox}$$

A PLL felépítése a mikrovezérlőkben található PLL-ekhez hasonló, azonban egy fontos sajátosság, hogy összesen 6 különböző kimenete van, melyek saját kimeneti osztóval rendelkeznek (O0..O5). Ezáltal lehetőség van különböző frekvenciájú, akár egymással szinkron órajelek előállítására. Ez azért fontos, mert egy időkritikus feladat magas órajelet igényelhet, azonban egy nem időkritikus funkciónak célszerű alacsonyabb órajelet választani a fogyasztás és az FPGA-terv megvalósíthatósága érdekében. Ha a különböző frekvenciájú órajelek egymással szinkronban járnak (az egyik órajel a másiknak az egész számú többszöröse), akkor a különböző órajelről járó logikák egymással könnyen összekapcsolhatóak lesznek.

1.1.3 CLB felépítése

A Spartan-6 alapvető logikai egysége a CLB (Configurable Logic Block), mely egyaránt használható kombinációs és sorrendi hálózatok kialakítására. Egy CLB-ben 2 ún. logikai szelet (slice) található,

---

amelyek egy kapcsolómátrix segítségével vannak kapcsolatban az FPGA többi erőforrásával. A szeletek között nincsen közvetlen kapcsolat.

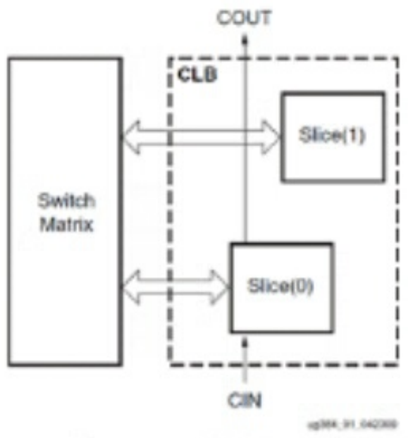


Figure 1: **Arrangement of Slices within the CLB**

A CLB-k kialakításánál a tervezők arra törekedtek, hogy minél többféleképpen lehessen felhasználni ugyanazt a logikai szeletet. Így például kialakíthatunk belőlük egy egyszerű kombinációs vagy sorrendi hálózatot, multiplexert, teljes összeadókat, RAM-ot vagy akár shift regisztert is. Attól függően, hogy egy adott logikai szelet milyen funkciók kialakítását teszi lehetővé három fajtát különböztethetünk meg:

- SLICEX
- SLICEL
- SLICEM

A különböző szeletek funkcionalitását az alábbi táblázat foglalja össze:

Table 1: **Slice Features**

Feature	SLICEX	SLICEL	SLICEM
6-Input LUTs	√	√	√
8 Flip-flops	√	√	√
Wide Multiplexers		√	√
Carry Logic		√	√
Distributed RAM			√
Shift Registers			√

Minden CLB úgy épül fel, hogy van benne egy SLICEX szelet, a másik pedig vagy egy SLICEL, vagy egy SLICEM. Így a szeletek 50%-a SLICEX, míg 25-25%-a SLICEL, ill. SLICEM.

1.1.4 SelectIO felépítése

A Spartan-6 FPGA-nak a környezetével való kapcsolatát az IO-egységei biztosítják. Ezek az egységek szintén széles körűen konfigurálhatóak, a legfontosabb paraméterek a következőek (az alapértelmezett értékek félkövér betűtípussal vannak jelölve):

- Jelszint (pl. **LVCMOS**, LVTTL)
- Slew rate (lassú, **normál**, gyors)

- Meghajtó erőssége (2, 4, 6, 8, **12**, 16 vagy 24 mA)
- Bemeneti lezárás (**nincs**, 25Ω, 50Ω, vagy 75Ω)
- Kimeneti impedancia (**nincs**, 25Ω, 50Ω, vagy 75Ω)

Mindegyik láb viselkedhet bemenetként, kimenetként, ill. be- és kimenetként. Mindegyik I/O támogatja a háromállapotú kimenetet, és a differenciális jelátvitelt. Differenciális jelátvitel esetén két I/O lábat kell párban használni.

A nagyobb rugalmasság érdekében az IO lábakat bankokra osztják, a Spartan-6 FPGA-kban 4, vagy 6 ilyen bank létezik. Ezeket a bankokat különböző feszültségről lehet járatni a kívánt jelszintnek megfelelően. Ezáltal az FPGA illesztését a környezetével egyszerűbben meg tudjuk valósítani.

1.1.5 Block RAM

Egy Block RAM egység 18 kb-es, továbbá használható 2 független 9kb-es blokként is. Ez egy szinkron működésű dual-port RAM, a két port egymástól teljesen függetlenül (órajel, adatszélesség) működik. Az adatokat paritásbitekkel lehet védeni, a paritásbitek száma a választott adatszélességtől függ.

A lehetséges konfigurációkat a következő táblázatok tartalmazzák:

Table 1: **Simple Dual-Port Mode Allowed Combinations for 9 Kb Block RAM**

			Port A								
			No Parity Bits						With Parity Bits		
			8K x 1	4K x 2	2K x 4	1K x 8	512 x 16	256 x 32	1K x 9	512 x 18	256 x 36
Port B	No parity bits	8K x 1	Use true dual-port mode					None Allowed	None Allowed		
		4K x 2									
		2K x 4									
		1K x 8									
		512 x 16									
		256 x 32	None Allowed					Allowed			
	With parity bits	1K x 9	None Allowed						Use true dual-port mode		None Allowed
		512 x 18									
		256 x 36							None Allowed		Allowed

9 Kb Block RAM—True Dual-Port Operation

Table 3: **True Dual-Port Mode Allowed Combinations for 18 Kb Block RAM**

		Port A								
		No Parity Bits						With Parity Bits		
		16K x 1	8K x 2	4K x 4	2K x 8	1K x 16	512 x 32	2K x 9	1K x 18	512 x 36
Port B	16K x 1	All Allowed						None Allowed		
	8K x 2									
	4K x 4									
	2K x 8									
	1K x 16									
	512 x 32									
	2K x 9	None Allowed						All Allowed		
	1K x 18									
	512 x 36									

A táblázatokból az derül ki, hogy a lehetséges adatszélesség paritásbitek nélkül 2 különböző egész számú hatványai lehetnek: 1, 2, 4, 8, 16, vagy 32 bit, paritásbitekkel pedig 9, 18 vagy 36 (minden bájt mellé tartozik egy paritásbit). Azonban a 9 kb-es konfiguráció esetén a 32 / 36 bites adatszélességet csak úgy lehet elérni, hogy az A porton csak írási, a B porton pedig csak olvasási műveletet hajthatunk végre (Simple Dual-Port Mode).

1.1.6 DSP-szelet

Egy DSP-szelet leegyszerűsített blokkdiagramja a következőképpen néz ki:

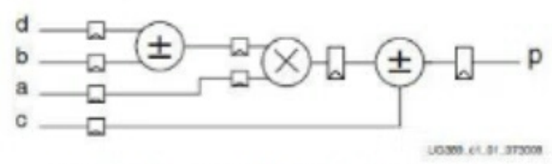


Figure 1-1: **DSP48A1 Slice with Pre-Adder**

A DSP48A1 kettes komplemensben ábrázolt számokkal dolgozik. Támogat egy 18-bites előösszeadót (pre-adder), amelynek a kimenete is 18-bites. Ezt egy 18-bites szorzó egység követi, melynek a kimenete 36-bites, és előjelesen 48 bitre van kiterjesztve. Végül egy újabb összeadó következik, melynek 48 bites bemenetei vannak, és a 48 bites végeredményt szolgáltatja.

Ez a struktúra gyorsabban képes összeadást elvégezni, mintha azt a CLB-ben valósítottuk volna meg. Látható, hogy pl. egy MAC utasítás is egyszerűen végrehajtható, melynek segítségével könnyen meg tudunk valósítani egy FIR-szűrőt (Finite Impulse Response). Innen származik a DSP48A1 elnevezés.

Az egyes műveletvégző egységek között regiszterfokozatok is találhatóak, melyek a csővezetékes működést biztosítják, ha erre szükség van.

1.2 Spartan-6 konfigurálása

A Spartan-6 FPGA-k programozásához konfigurációs adatokat kell az eszközre tölteni. Ennek a mérete a konkrét eszköztől függ, 3 és 33 Mbit között változhat. Mivel az FPGA ezeket az adatokat egy SRAM típusú belső latch-ben tárolja, ezért kikapcsoláskor a konfigurációs adatok elvesznek. Ahhoz, hogy a legközelebbi bekapcsoláskor is fel tudjuk programozni az eszközt, szükség van a konfigurációs adatok külső flashben való tárolására.

A Spartan-6 FPGA-k párhuzamos és soros interfészt is támogatnak a konfigurációs adatok beolvasásához, továbbá egyaránt viselkedhet mesterként és szolgaként is. Ha az FPGA a mester, akkor a konfigurációs órajelet ő hajtja meg, ellenkező esetben a külső eszköz. A soros interfész sebessége növelhető azáltal, hogy nem csak 1, hanem 2 (dual) vagy 4 (quad) adatvonalat használhatunk. A párhuzamos interfész adatszélessége 8 vagy 16 bit lehet. Az interfész típusának megállapítása induláskor hardveres konfiguráció alapján lehetséges. Ezt úgy valósították meg, hogy van két mód bemenete az FPGA-nak melyeket a kialakított interfésznek megfelelően kell elhúzni. A lehetséges konfigurációs interfészek tehát:

- Párhuzamos interfész (BPI) – Mester
- Párhuzamos interfész (BPI) – Szolga
- Soros interfész (SPI) – Mester
- Soros interfész (SPI) – Szolga

A fentieken felül természetesen lehetőség van a konfiguráció beírására a JTAG interfészen keresztül is.

Az alábbiakban a konfiguráció lépéseit vesszük sorba.

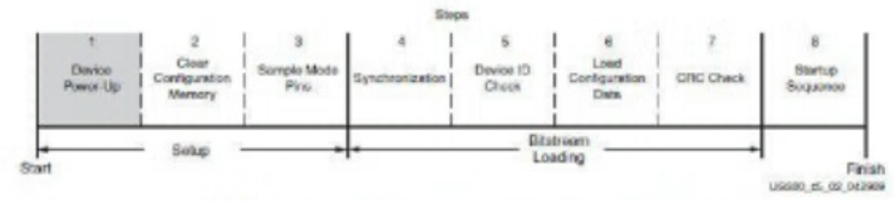


Figure 5-2: **Spartan-6 FPGA Configuration Process**

1. *Bekapcsolás*

   A Spartan-6 FPGA-nak 3 tápfeszültségre van szüksége az induláshoz:

   - V_{CCO2}
   - V_{CCAUX}
   - V_{CCINT}

   A V_{CCINT} az FPGA core tápfeszültségét biztosítja, nagysága 1,2V. A V_{CCAUX} a konfigurációs logikát látja el tápfeszültséggel, míg a V_{CCO2} a 2. banknak a tápfeszültsége. A soros interfészhez elegendő ez a tápfeszültség, mivel a konfigurációs lábak ebben a bankban találhatóak. Párhuzamos interfész esetén más bankok konfigurációs lábaira is szükség van, értelemszerűen ilyenkor ezeknek is biztosítani kell a megfelelő tápfeszültséget. A V_{CCAUX} és a V_{CCINT} feszültségek 2,5V vagy 3,3V lehetnek.

2. *Konfigurációs memória törlése*
3. *Mód bemenetek (M[1:0]) beolvasása*

   A konfigurációs órajel első felfutó élére olvassuk be a mód bemeneteket, melyek kiválasztják az interfészt a konfigurációs adatok beolvasásához.

4. *Szinkronizáció*

   Az FPGA a kiválasztott interfészen keresztül megpróbálja beolvasni a konfigurációt a 0-s címről kezdődően. A beolvasott adatokat a szinkronizációs szó (0xAA995566) megtalálásáig figyelmen kívül hagyjuk.

5. *Eszközazonosító ellenőrzése*

   Ez egy biztonsági lépés annak megakadályozására, hogy egy másik FPGA-ra lefordított tervet töltsünk be. Az FPGA kiolvassa az eszközazonosítót a Flash-ből és összehasonlítja a sajátjával. Csak akkor próbálja meg betölteni a konfigurációt, ha a két érték egyezik.

6. *Konfigurációs adatok betöltése*
7. *CRC-ellenőrzés*

   Ez is egy biztonsági lépés az adatok konzisztenciájának ellenőrzése céljából. Ez a lépés kihagyható, viszont egy hibás konfiguráció hibás működéshez, sőt az FPGA fizikai tönkretételéhez vezethet.

8. *FPGA elindulása*

   Ennél az utolsó lépésnél beállítható, hogy az FPGA milyen lépésekben induljon el. Az alapértelmezett sorrend a következő:

   a. DONE kimenet elengedése
   b. Kimenetek aktiválása
   c. Globális írásengedélyezés: RAM-ok és flip-flopok működésének az engedélyezése
   d. Belső End Of Startup jel magasba vált

   A fentieken kívül lehetőség van beiktatni egy olyan lépést is, ahol megvárjuk, amíg bizonyos DCM egységek, és a felhasznált összes PLL kimeneti órajelei stabilizálódnak.

A konfigurációs idő legnagyobb részét tipikusan a konfigurációs adatok beolvasása viszi el. Ezt az időt a következőképpen tudjuk meghatározni:

$$t_{konf} = \frac{N}{f_{CLK} \times W}$$

ahol N a konfigurációs bitek száma (3..33 Mbit közötti), f_{CLK} a konfigurációs órajel frekvenciája (2..26 MHz), W pedig a konfigurációs interfész bitszélessége. Számoljunk ki egy konkrét esetet az alábbi paraméterekkel:

- XC6SLX25 FPGA -> konfigurációs bitek száma = 6 440 432
- Legnagyobb konfigurációs órajel -> 26 MHz
- Soros interfész (1 bit szélesség)

A fenti képlet alapján a konfigurációs idő ebben az esetben:

$$t_{konf,x1} = 248\ ms$$

Abban az esetben, ha 8-bites párhuzamos interfészt használunk, ez az idő a nyolcadára csökken:

$$t_{konf,x8} = 31\ ms$$

Megállapíthatjuk, hogy soros esetben is elfogadható indulási időt kapunk (vannak olyan mikrokontrollerek, amelyeknek az indulási ideje szintén ez a nagyságrend!), mely azonban drasztikusan lecsökkenthető párhuzamos interfész választásával. Ennek az ára, hogy több lábat kell a konfiguráció céljára felhasználnunk.

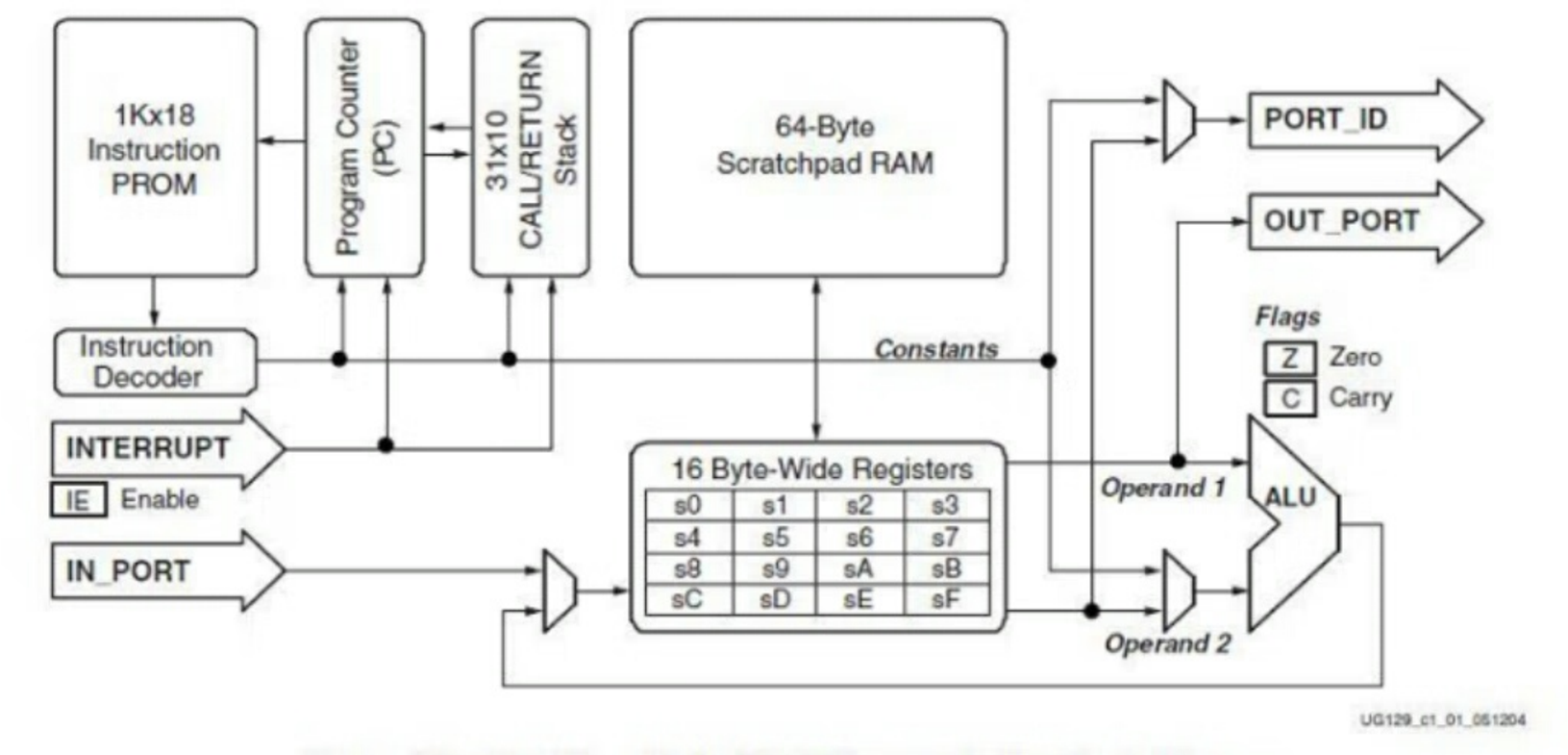


Figure 1-1: **PicoBlaze Embedded Microcontroller Block Diagram**

Az ábrán látható, hogy a Microblaze processzor egységei és interfészei közül sok választható elem van. Természetesen az alapvető működéshez szükséges elemek kötelezően benne vannak, ezek az alábbiak:

- 32 darab 32 bites általános célú regiszter
- Az utasítások beolvasásához szükséges interfész (Instruction-side Bus IF), a hozzá tartozó puffer és az utasításdekódoló egység
- Programszámláló
- Adatok írásához / olvasásához szükséges interfész (Data-side Bus IF)
- ALU
- Speciális célú regiszterek

A választható elemek az alábbiak:

- Cache az utasítás- és / vagy az adatoldalon
- Memory Management Unit (MMU)
- Az ugró utasítások gyorsítására használatos Branch Target Cache
- ALU kiegészítések: barrel shifter, szorzás, osztás, FPU
- Stream interfészek nagy tömegű adatmozgatáshoz

Ezeken kívül még számos dolog beállítható a processzorban, például az utasításpipeline lehet 3- vagy 5-fokozatú.

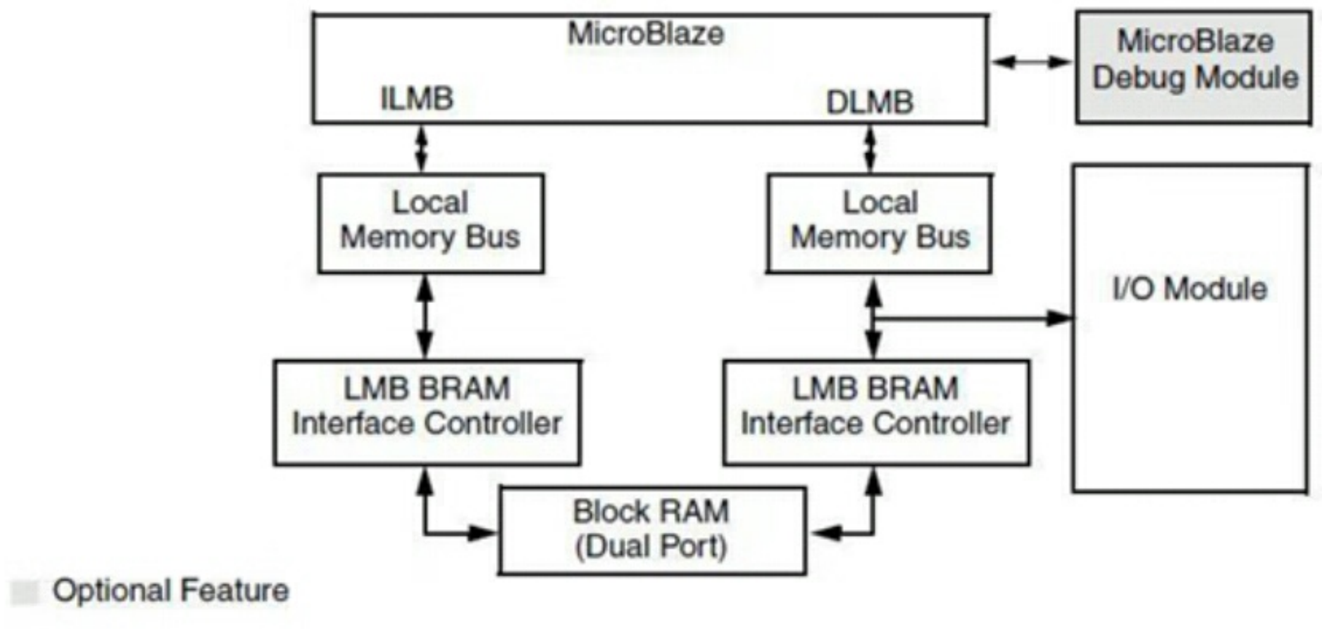


Figure 1: **MicroBlaze Micro Controller System (MCS)**

4.2.1 PS

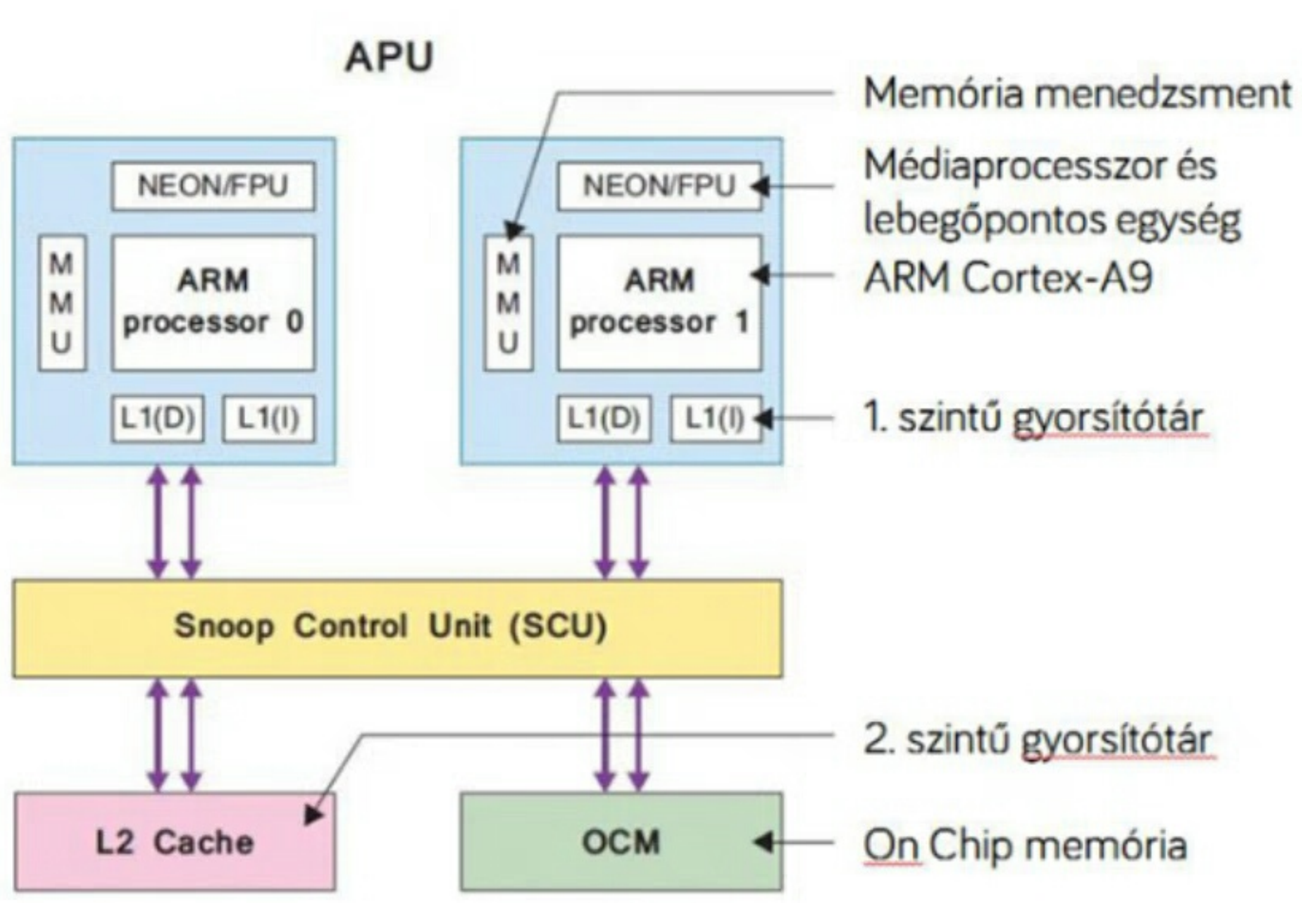
A PS az alábbi 4 fő egységből áll:

- Application processor unit (APU)
- Memória interfészek
- I/O perifériák (IOP)
- Central Interconnect

Vizsgáljuk meg ezeket az egységeket részletesebben.

4.2.1.1 APU

Az APU blokkvázlatát az alábbi ábra szemlélteti.



Az APU egy ARM Cortex-A9 kétmagos processzor 2,5 DMIPS/MHz teljesítménnyel. Mindegyik mag rendelkezik egy dupla pontosságú lebegőpontos valamint egy NEON aritmetikai egységgel. Kétszintű cache gyorsítja az adathozzáférést: egy 1. szintű 32kB-os (4-utas set-asszociatív) és egy 2. szintű 512kB-os (8-utas set-asszociatív). Továbbá rendelkezésre állnak egy dual-port RAM (256kB), egy 8-csatornás DMA, egy CoreSight debugger, egy megszakításvezérlő (GIC), három WDT (egy-egy a processzormagoknak és egy rendszer WDT), és két Triple Timer/Counter (TTC).

4.2.2 Interfészek PS és PL között

A PS és a PL között az alábbi AXI-interfészek állnak a rendelkezésünkre:

- Két 32 bites AXI-Master (kis- és közepes sebességekre)
- Két 32 bites AXI-Slave (kis- és közepes sebességekre)
- Négy 64 / 32 bites nagyteljesítményű AXI-Slave interfész (HP AXI)
- Egy 64 bites AXI-Slave interfész cache-koherens porttal (ACP)

Ezek közül a HP-AXI interfész blokkvázlata az alábbi:

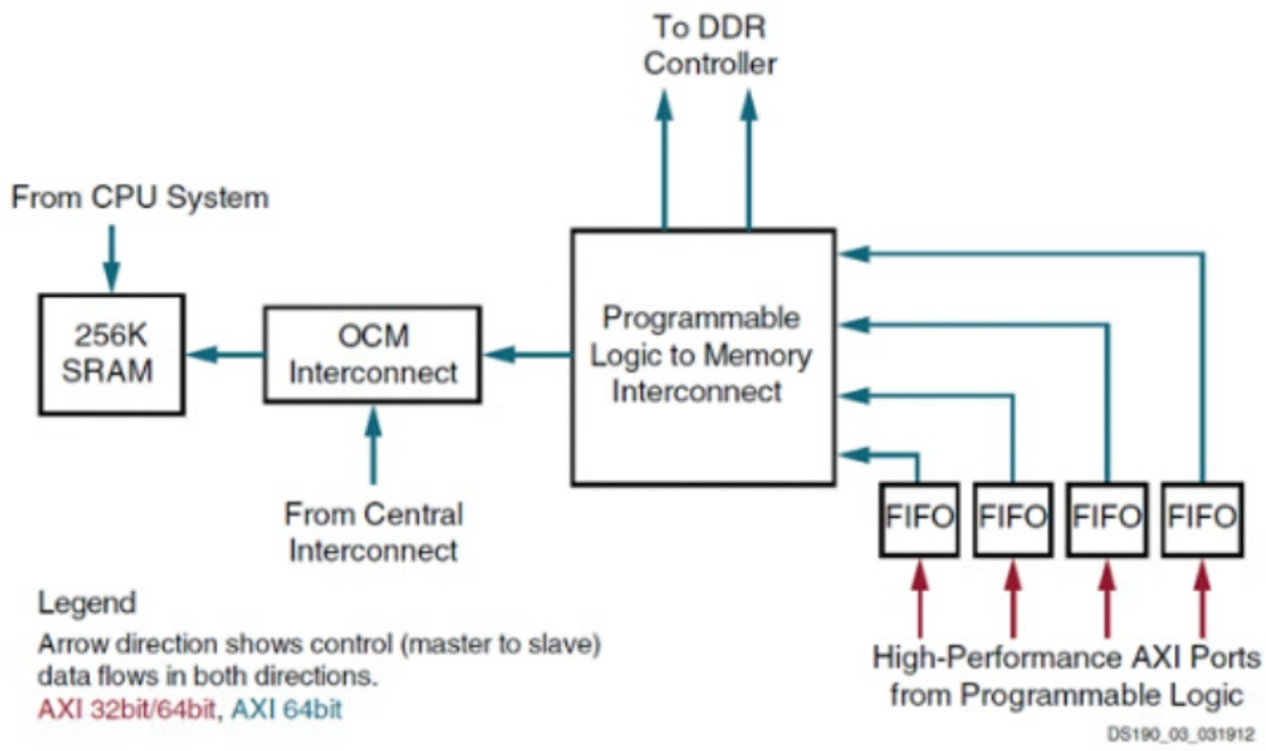


Figure 3: **PL Interface to PS Memory Subsystem**

Amint látható ez az interfész pufferelt adatátvitelt tesz lehetővé a FIFO-k segítségével. Mindegyik interfész eléri a DDR és az OCM memóriát is. Mivel ezeket a memóriákat a PS is látja, ezért az interfészt nagysebességű adatátvitelre használhatjuk a PS és a PL között. A HP-AXI esetében dedikáltan a PL az adatátvitel mestere.

4.2.3 Rendszerindulás

Az SoC indulása az alábbi lépésekből tevődik össze:

1. Boot-ROM elindulása
2. Boot mode bemenetek mintavételezése
3. FSBL (First Stage Bootloader) másolása az OCM-be
4. FSBL elindítása
5. PS és PL konfigurálása

A tápfeszültség megjelenése után a PS a boot ROM-ban tárolt kódot indítja el. Ez a kód beolvassa a boot mód bemeneti lábakat, amely alapján eldönti, hogy milyen típusú külső memóriából kell a FSBL-t beolvasnia. A támogatott memóriák a következőek:

- SPI Flash
- NAND Flash
- NOR Flash
- SD-kártya

Ezután az FSBL kódját bemásolja az OCM-be, majd futattja azt. Az FSBL felelős a PS és a PL megfelelő konfigurálásáért. A PL konfigurálása opcionális, ha a rendszerünknek nincs szüksége rá, akkor ez a lépés kihagyható.