**Các phần cần học**

• Beágyazott rendszerek fogalma, alapvető jellemzői

• Beágyazott rendszerek funkcionális, időzítési és megvalósítási kérdései

• Real-time rendszerek fogalma, fajtái és jellemzői

• Hard real-time rendszerek főbb jellemzői, gyakorlati megvalósíthatóság lehetőségei

• Soft real-time rendszerek főbb jellemzői, gyakorlati megvalósíthatóság lehetőségei

• Számítógép acrhitektúrák. Neumann és Harvard féle acrhitektúra felépítése és főbb jellemzői

• 8 bites mikrokontrollerek általános jellemzői, felhasználási területei.

• 32 bites mikrokontrollerek általános jellemzői, felhasználási területei.

• Multiprocesszoros rendszerek fogalma, fajtái és általános jellemzői

• Szorosan csatolt multiprocesszoros rendszerek fajtái és általános jellemzői

• Lazán csatolt multiprocesszoros rendszerek fajtái és általános jellemzői

• Perifériakezelés fajtái, általános jellemzői

• Eszközszintű perifériakezelés

• Logikai perifériakezelés

• Programozható logikai eszközök fajtái, általános jellemzői (ASIC, PLD)

• FPGA felépítése. Logic cell, slice, CLB, IOB, DCM.

• Absztrakciós szintek, tervezési stratégiák.

• Hardver leírónyelvek általános jellemzői, fajtái.

• Számítógép-hálózatok fogalma, általános jellemzői és felépítése

• OSI modell felépítése és jellemzői. Protokoll és az interfész fogalma.

• OSI modell egyes rétegeinek általános jellemzői.

• Ipari hálózatok felépítése és általános jellemzői.

• Gyakrabban használt ipari buszok. RS485, Ipari Ethernet és a CAN.

# Valós idejű rendszerek (RST)

Egy rendszer valós idejűnek tekinthető abban az esetben, ha a külvilág felöl érkező jelzésre meghatározott időn belül reagál.

|  |  |  |
| --- | --- | --- |
| Karakterisztika | Kemény valós idejű | Lágy valós idejű |
| Válasz idő | Egy idő alatt a rendszernek reagálnia kell a környezetből érkező eseményre | Időnként interakciót kell végrehajtania a rendszernek |
| Viselkedés csúcsterhelés esetén | Jósolható módon viselkednek | Lehet nem teljesítik az előír követelményeket ideiglenesen. |
| Ütem vezérlése | Szinkronban kell lennie környezetének állapotával. | befolyásolják környezetüket, ha nem képesek eleget tenni feladatuknak, például megnövelik a feladat végrehajtási idejét. |
| Biztonság |  |  |
| Adatfájlok mérete | Kisci, közepes | Nagy |
| Redundancia | Aktív | Chekpoint-recovery |
| Adat integritás | Short-term, rövidtávú | Long-term, Hosszútávú |
| Hibadetektálás | Autonóm módon történik  Pl: légzsák | Emberi beavatkozás mód  Pl: meghibásodott rendszertől kapott hibaüzenetek |
| Hibás állapot kezelése |  |  |

## Kemény valós idejű

A kemény valós idejű rendszer (hard real-time system, HRT) esetén a rendszernek meghatározott időn belül mindenképpen kell válaszolnia a környezetből érkező jelzésre, mert ha nem, akkor az katasztrófális következményekkel jár.

Pl: felügyeleti biztonságtechnikai rendszer ,repülógép érzékelőjei,

## Lágy valós idejű

A lágy valós idejű rendszer (soft real-time system, SRT) esetén a rendszernek nem kell meghatározott időn belül válaszolnia a környezetből érkező jelzésre, megengedett az idő túllépés is, az eredmény értékes lehet az időkorláton túl is, viszont az idővel degradálódhat is.

Pl: Mérésadatgyűjtő rendszer

## Jellemzői

### Válaszidő (response time)

A válaszidő (response time) az az idő, amely idő alatt a beágyazott rendszernek reagálnia kell a környezetből érkező eseményre, vagy az az idő, amely időnként interakciót kell végrehajtania a rendszernek.

### Viselkedés csúcsterhelés esetén

Csúcsterhelésnek nevezzük (peak-load performance) azt az állapotot, amelyet a rendszer **meghibásodás nélkül elvisel**, illetve ezen maximális terhelés mellett a rendszer még teljesíti a számára előírt követelményeket. A rendszerek különféle módon viselkednek csúcsterhelés esetén.

A kemény valós idejű rendszerek (HRT)esetén jól definiált kell, hogy legyen ez a viselkedés. Tervezéskor biztosítani kell, hogy a beágyazottrendszer minden szituációban az időkorláton belül teljesítse feladatát, hiszen a HRT rendszerek éppen azáltal valósítják meg a velük szemben megfogalmazott elvárásokat, hogy még a ritkán előforduló csúcsterhelések idején is jósolható módon viselkednek.

A lágy valós idejű (SRT)rendszereket átlagos teljesítmény-jellemzőkre tervezzük, a ritkán előforduló csúcsterhelések következményeit -gazdaságossági megfontolásból –elviseljük azt is, hogy a beágyazott rendszer ne teljesítse az előírt követelményeket ideiglenesen.

### Ütem vezérlése (Control of pace)

A beágyazott rendszerek esetében a feladatok végrehajtását különféle módon lehet megoldani. Az ütemezett feladatok azok, amelyek meghatározott időnként végrehajtásra kerülnek, illetve valamilyen külső ütemező esemény hatására hajtódnak végre.

A kemény valós idejű (HRT) rendszernek minden körülmények között szinkronban kell lennie környezetének állapotával. A szinkron állapot fenntartását biztosíthatja belső ütemező, vagy a környezetből érkező szinkron jel.

A lágy valós idejű (SRT) rendszerek befolyásolják környezetüket, ha nem képesek eleget tenni feladatuknak, például megnövelik a végrehajtandó feladat végrehajtási idejét.

### Biztonság

A biztonság kritikusságának mértékétől függően sokféle feladat merülhet fel tervezési időben. Autonóm hibadetektálási mechanizmusokat kell kidolgozni, amelyek valamilyen “talpra állítási” (recovery) akciót indítanak az adott alkalmazás által diktált időviszonyok mellett.

### Adatfájlok mérete

HRT rendszerek kisméretű adatfájlokon dolgoznak, amelyek valós idejű adatbázist alkotnak. Ezek jellemzője az adatintegritás rövid idejűsége, mert az idő múlásával az adatok jelentős része aktualitását veszíti.Az SRT rendszerekben éppen ellenkezőleg a hosszú idejű adatintegritás fontos.

### Redundancia

HRT rendszerek esetén ez a stratégia csak korlátozottan használható mert:

- az időkorlát tartása nehéz, mert a visszagörgetéshez szükséges idő nem, vagy nehezen jósolható,

- a környezetet befolyásoló “utasítás” nem tehető meg nem történtté,

- az ellenőrzési pontnál érvényes adatok az idő múlásával érvényüket veszítik.SRT rendszerekben (pl. tranzakciós rendszerek) hiba esetén a számításokat “visszagörgetik” a legutolsó ellenőrzési ponthoz, amikor még biztosan helyes volt a működés és onnan kezdik a “talpra állítást”.

### Adatintegritás

### Vezérlési mód

A beágyazott rendszerek esetében alapvetően kétféle vezérlési módot tudunk értelmezni:

- eseményvezérelt (event triggered) rendszer

Az aszinkron módon érkező megszakítások kiszolgálása, illetve ebből adódóan dinamikus ütemezés szükség.

- idővezérelt (time triggered) rendszer

Minden kommunikáció, illetve feldolgozás központi időzítéshez (órához) szinkronizált módon történik

### Hibadetektálás

A hibadetektálás a valós idejű rendszerek esetében eltérően történhet HRT és SRT rendszerek esetében.

HRT esetében a hibadetektálás autonóm módon történik, ami azt jelenti, hogy külső beavatkozásra nincsen lehetőség. Nézzünk egy egyszerű esetet, például amikor a személygépjármű ütközik egy másik járművel. Ebben az esetben a gépjárművezető személynek nincs esélye, hogy érzékelje az ütközést, illetve bekapcsolja a légzsákot, és ha ezt egy automatika nem tenné meg, akkor az ilyen jellegű balesetek következményei még súlyosabbak lennének.

SRT esetében lehetőség van arra, hogy emberi beavatkozás segítségével történjen meg a hiba detektálása. SRT esetében a hibadetektálásra különféle módszereket lehet alkalmazni:

- bemenetek és kimenetek állapotának visszajelzése (pl. LED-ek segítségével),

- meghibásodott rendszertől kapott hibaüzenetek,

- a berendezés meghibásodáskor vagy felvett állapotának jelzése.Ezen visszajelzések alapján a hibadetektálást végző operátor gyorsabban beazonosíthatja a keletkezett hiba helyét és esetleg az okát is.

### Hibás állapot kezelése

Katasztrófa megakadályozása bénítással (fail-safe) is történhet. Példaképpen a vonatok esetében, ha a biztonsági rendszer meghibásodik, akkor a szabad utat jelző jelzőlámpa rendszerében csak a piros lámpák fognak jelezni.

Másik módszert abban az esetben alkalmazzák, ha a normál működtetés már lehet hatásos, ilyenkor a katasztrófa elhárítás extra eszközökkel (failoperational) történik. Példaképpen, ha a repülőgép robotpilótája már nem képes normál módon működtetni a járművet, akkor a pilótának át kell vennie az irányítást a gép felett, és kézi irányítással lehozni a repülőgépet.

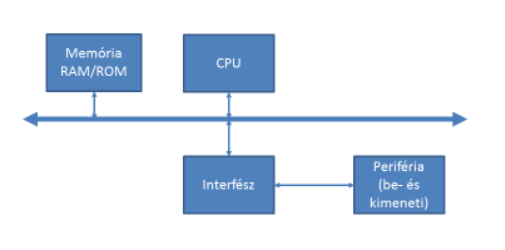
## Kérdések:

* Mutassa be, hogy egy felügyeleti biztonságtechnikai rendszer real-time 6 db jellemzőjét!
  + Kemény valós idejű
* Mutassa be, hogy egy mérésadatgyűjtő rendszer real-time 6 db jellemzőjét!
  + Lágy valós idejű

# Architektúrák

## Neumann architektúra

Kérdés: jellemezze az architektúrát 5 szempont szerint! Milyen architektúra, mi a szerepe az egyes egységeknek, hogyan történik a feladat végrehajtás?



Öt funkcionális egység

1. aritmetikai egység, (ALU)
2. központi vezérlőegység,
3. memóriák,
4. bemeneti egységek
5. kimeneti egységek

Memória:a programot(utásítás) és a adatottárolja

CPU (Central processing unit) központi feldolgozóegység más néven processzor vagy microprocesszor.

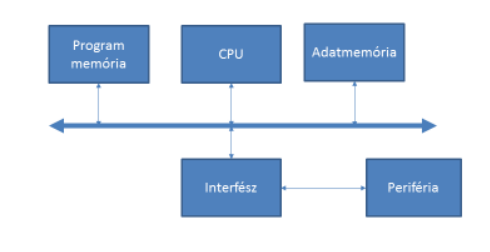
### Alkalmazás

Audio processing

Medical imaging: Xray, MRI,cat scan, fitness tracker, smart watch, digital assistants,

## Havard archiektuúra

Kérdés: Jellemezze az architektúrát 5 szempont szerint! Milyen architektúra, mi a szerepe az egyes egységeknek, hogyan történik a feladat végrehajtás?



Hogyan történik a feladat végrehajtás?

Mi a feladat?

Honnan jön a feldat?

## Mi a különség a neumann és harvard architektura között?

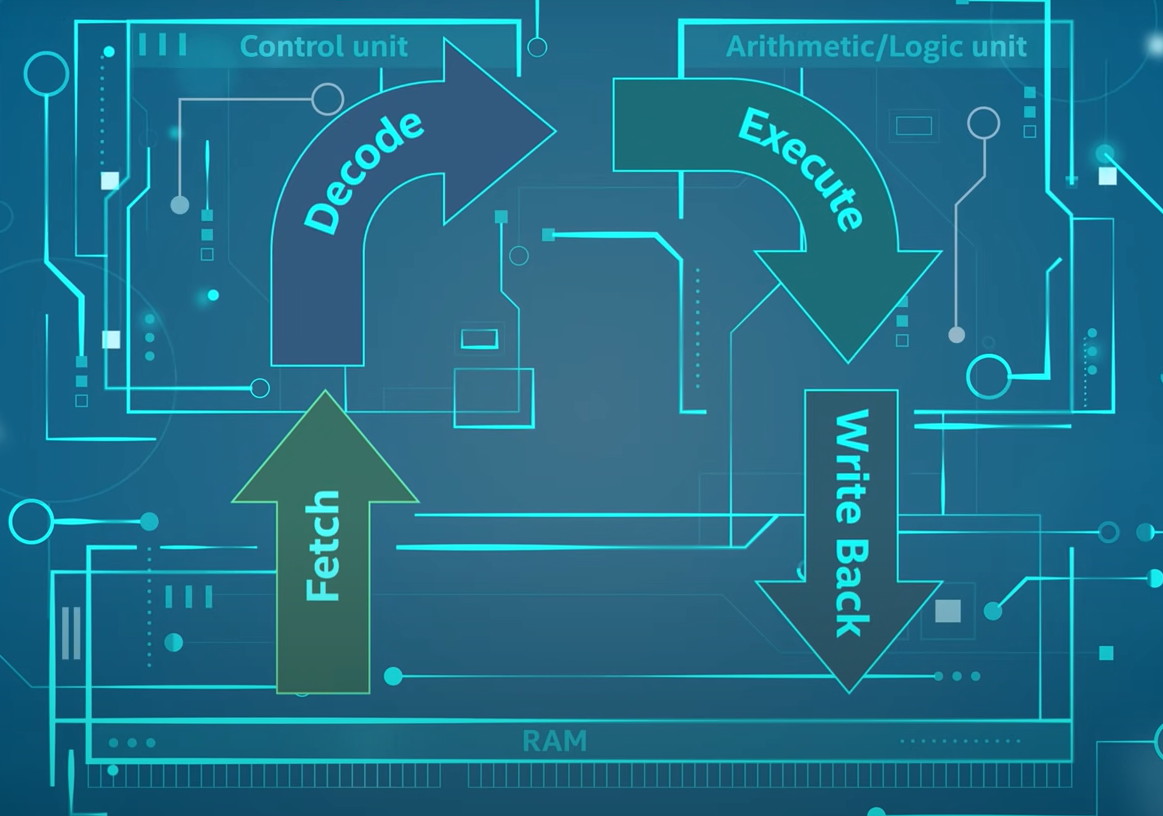
|  |  |  |
| --- | --- | --- |
|  | Neumann | Harvard |
| Memória | Egy közös memóriában tárol | Kettő külön memóriában tárol |
| Hardver igénye | Kevés | Több |
| Helyigény | Ez kevesebb helyet igényel. | Ez több helyet igényel. |
| Sebesség | Általános | Gyorsabb |

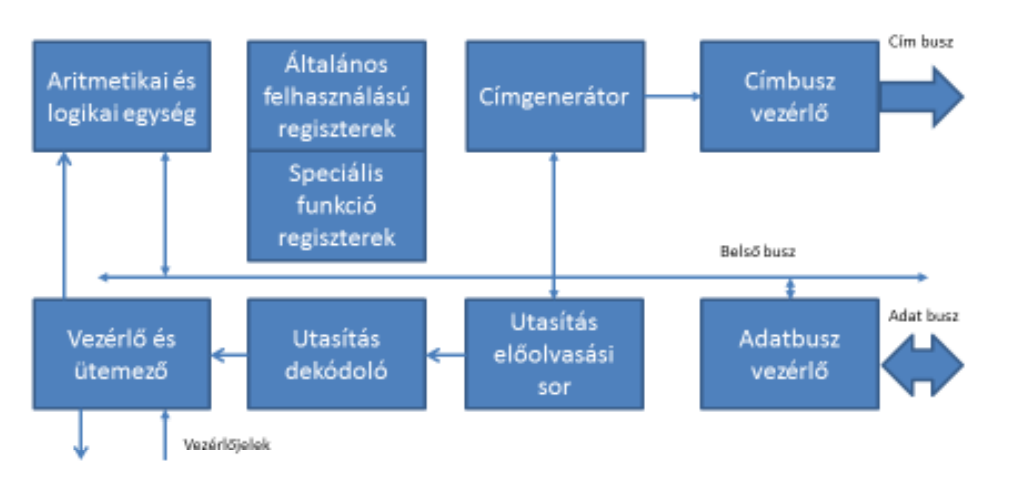
# CPU

CPU feladata: utásítások végrehajtása

4 step instruction cycle:

1. Fetching instruction from memory
2. Decoding instruction
3. Exciting instruction
4. Stroring the result back in memory
5. And return to step 1





Ez milyen achitektura???

Ez neumann achitektura

Hol van a memory??

Nincs itt

Mik a CPU főbb elemei és mi a szerepe az egyes egységének?

* ALU (Aritmetikai és logikai egység): Aritmetkiai és logikai műveletek végrehajtás
  + Aritmetikai: +,-,x,:
  + Logikai: AND, OR, NOT, XOR
* Regiszterek: Általános felhasználású regiszterek és speciális funkciójú regiszterek: Ideilennes adatokat tartalmaz.
* Utasítás dekódoló: Az utasításokat értelmezi.
* Vezérlő- és ütemező egység(CU: control unit): Ez vezérli az ALU-t, meghatároza milyen műveletet kell elvégeznie az ALU

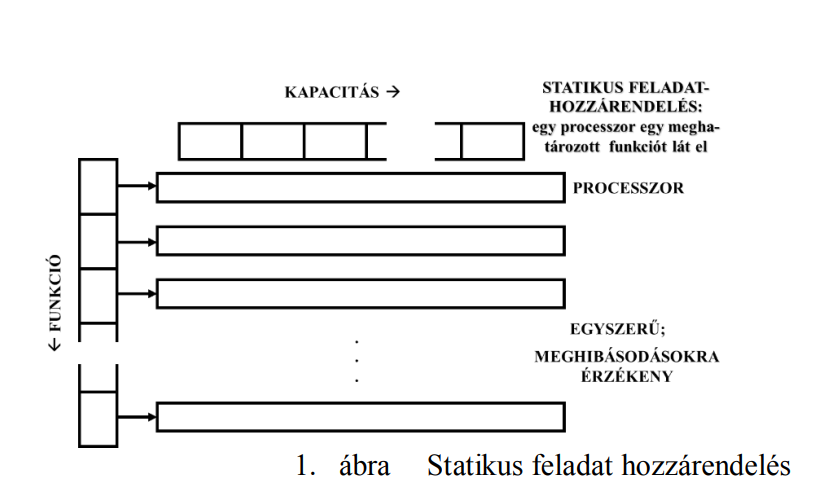
Utasítás ciklus:

1. Utásítás beolvasás
2. Beolvasott utásíás dekódolása
3. Utásítás végrahajtása
4. Eredmény tárolása
5. Következő utasítás címének kiszámítása
6. Vissza 1. lépés

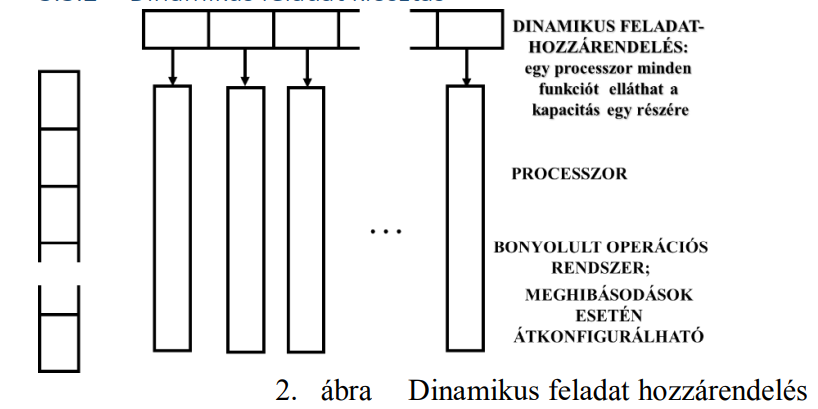
# Mikrokontroleres vagy mikroprocesszoros megvalósítások

## Feladat kiosztás

### Statikus feladat kiosztás



### Dinamikus feladat kiosztás



### Szorosan csatolt multiprocesszoros rendszere

A szorosan csatolt rendszerek esetében a kommunikáció közös erőforráson keresztül valósul meg, egyetlen operációs rendszer van. Bonyolult, de gyors működésű.

**Szervezések**

**Crossbar szervezés**: A crossbar olyan kapcsoló, amely több processzorot több adatterülethez köt mátrixos módon

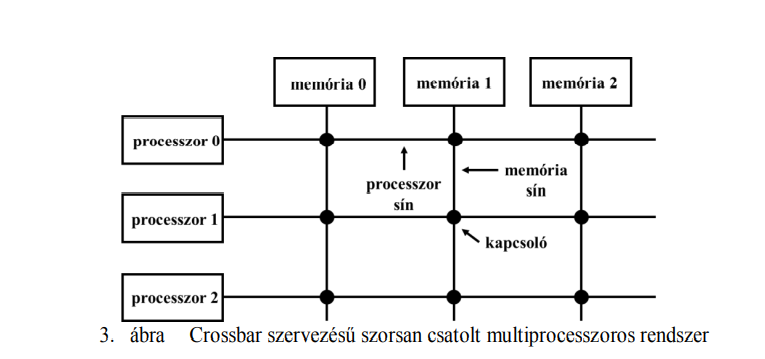


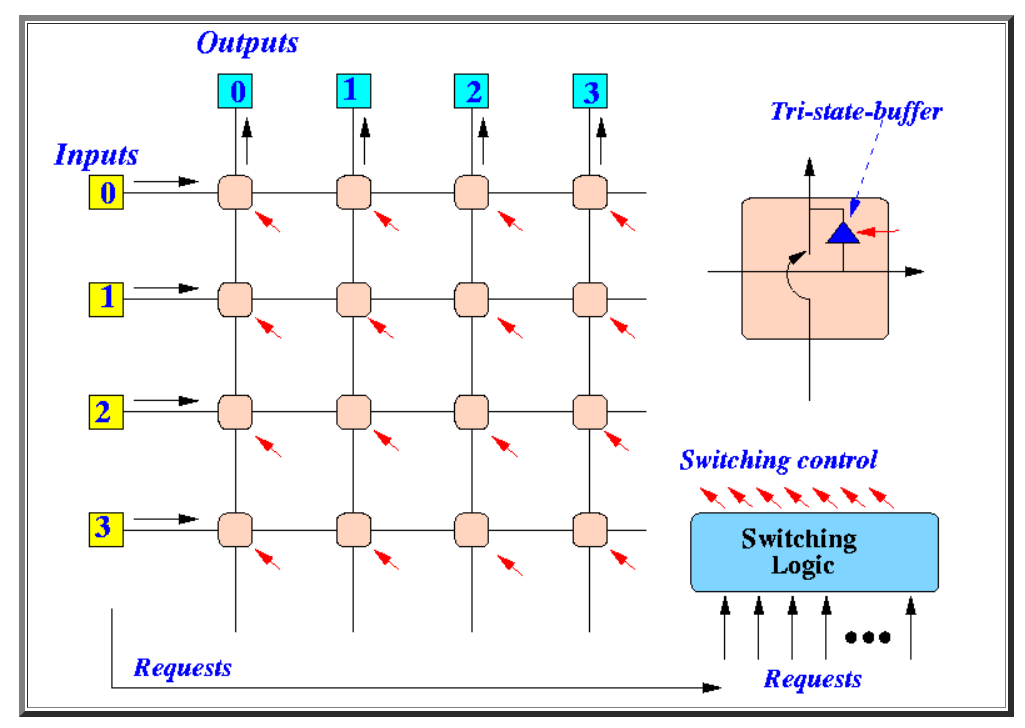
Előnyok:

* Egyszerű felépítés
* Könyen átkonfigurálható

Hátrányok:

* Csak kis rendszerek esetén használható





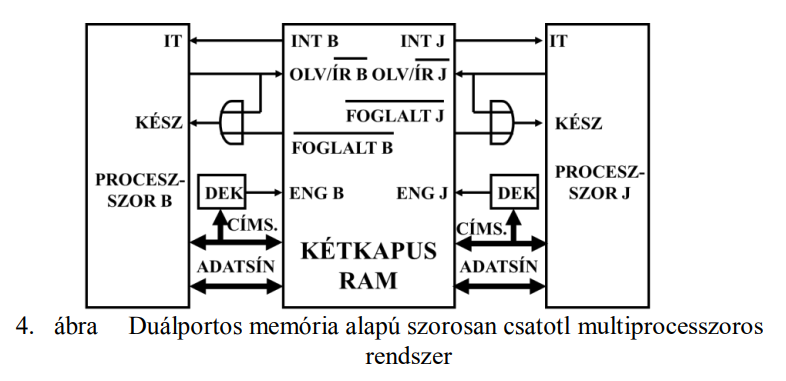
<http://www.mathcs.emory.edu/~cheung/Courses/355/Syllabus/90-parallel/CrossBar.html>

The crossbar switch can switch inputs (yellow color squares) to the outputs (cyan color squares) - imagine each box is a CPU or a memory module.

The setting of the switching fabric is under the control of a **switching logic** circuit

The sources (inputs) first **send request** signals to the **switching logic** circuit.

The switching logic circuit will set the switching fabric to route the requests from the input to their destinations.



Egymástól független két bemeneti/kimeneti oldal

Ha két oldalról azonos címhez kívánunk fordulni, akkor a beépített vezérlő áramkör a „később" jövő felé foglalt jelet ad ki. Ezt csak íráskor kell figyelembe venni, várakozásra késztetve a megfelelő oldali processzort

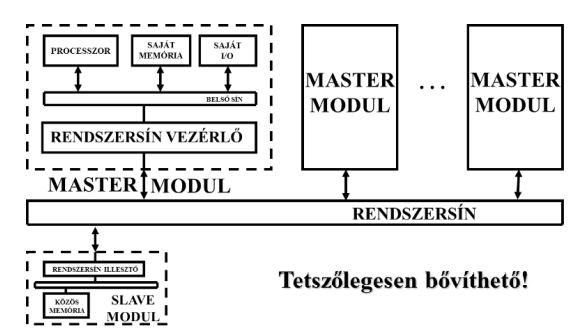
Az utolsó regiszterbe a jobboldali processzor valamit beír. Ezt a veloldali felé egy INT B jelzi. A beloldali processrzo kiolvasva a regisztert törli a jezést (hardver SZEMAFOR)

Előny: egyszerű

Hatrányok:

* Csak kevés (2-4) port valósítható meg a bonyolult kapurendszer miatt->akkor csak kis renszer építhető ki
* Kis mimória kapacitás valüsíható meg a bonyolult kapurrendzer miatt
* Nehéz átkonfigurálni (khó cấu hình lại)

**Renszersínre alapozott szervezés**

****

**Master modul:**

- magához ragadhatja a rendszersín vezérlését.

**Slave modul:**

- nem ragadhatja magához a rendszersín vezérlését;

- közös erőforrás(oka)t tartalmaz.

• A rendszer (elvben) tetszőlegesen bővíthető és átkonfigurálható.

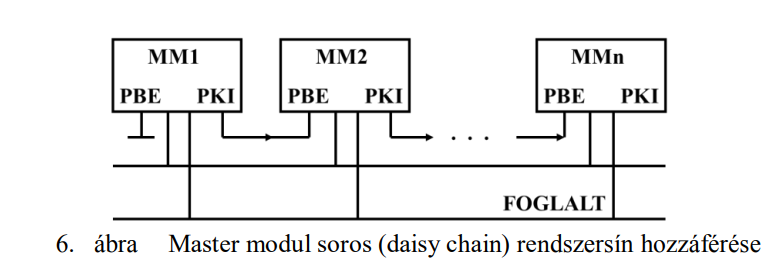
• Az egyes processzorokon futó folyamatok (a slave modulban lévő) közös erőforráson keresztül kívánnak kommunikálni egymással. Fel kell oldani a közös erőforrás használatáért folyó versengést.

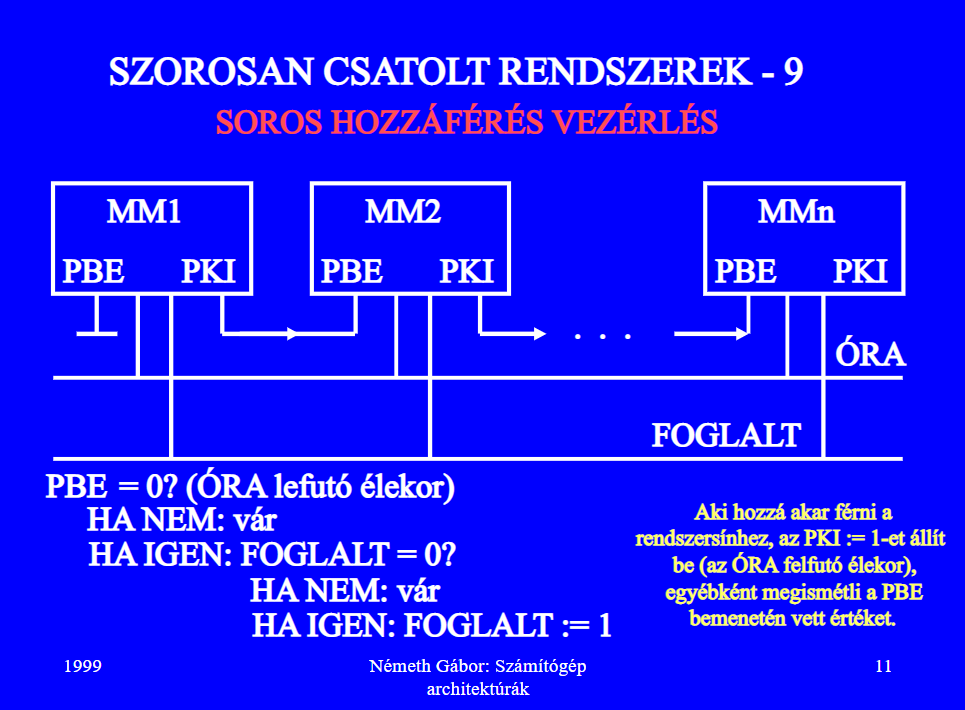
• DE: a folyamatokat futtató master moduloknak előbb hozzá kell férniük a rendszersínhez.

• A master modulok versenyeznek a rendszersínhez való hozzáférés jogáért:

- SOROS (daisy chain), vagy

- PÁRHUZAMOS hozzáférés vezérlés.





ELŐNYE:

nagyon egyszerű.

• HÁTRÁNYAI:

- A terjedési késleltetés miatt csak nagyon kis

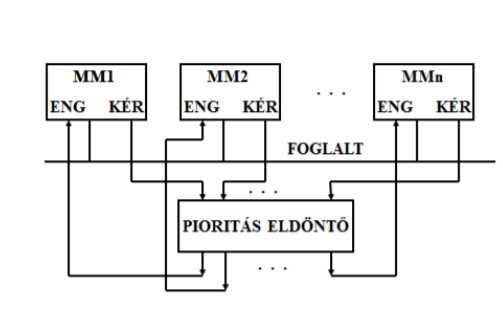
rendszerekben alkalmazható (meg kell várni a legbaloldalibbtól is az igény

végigfutását a láncon).

- Nehéz átkonfigurálni (vezetéket kell elvágni és összekötni), de a

kis rendszerek (nagy megbízhatóság) miatt erre menetközben nincs szükség.

- A rögzített prioritás miatt éhezés léphet fel.



• A prioritás lehet rögzített (a prioritás eldöntő egyszerű kombinációs

áramkör), vagy változó (pl. körbenforgó - round robin - ekkor hosszabb idő

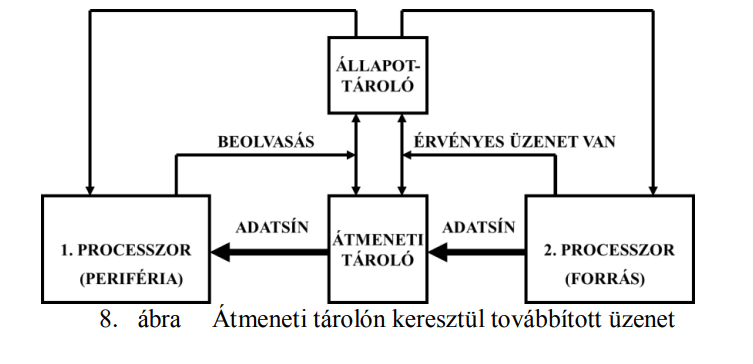
alatt egyenlő esélyt kap minden modul).

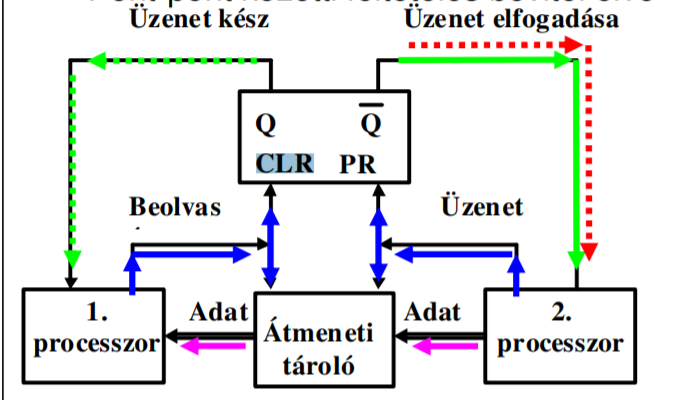
• A prioritás eldöntőt nagyobb rendszereknél elosztott módon valósítjuk meg

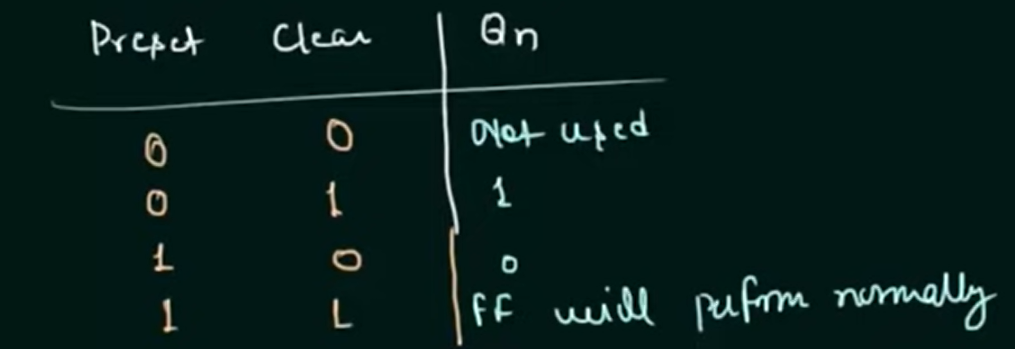
(ellenkező esetben a rendszer kritikus része lenne).

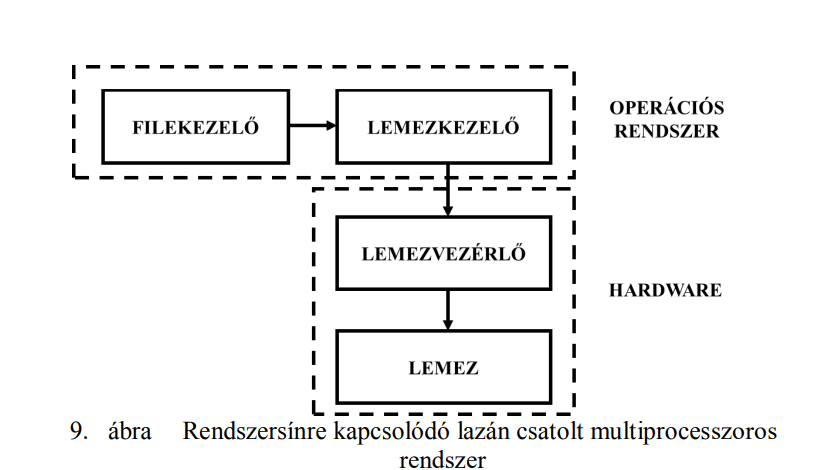
### Lazán csatolt multiprocesszoros rendszerek

Elv: az üzenet átvitelének idejére a forrás processzor a fogadót perifériájának tekinti.









#### Postafiók elv



* Egyszerre csak egyetlen folyamat férhet hozzá a postafiókhoz (kölcsönös kizárás).
* A postafiók használatát (KRITIKUS SZEKCIÓ) megfelelő szemafór(ok) kezelésével biztosítjuk.
* A szemafór vizsgálatának és átállításának oszthatatlan műveletnek kell lennie (egyébként egy másik folyamat is úgy érezhetné, hogy hozzáférhet a postafiókhoz).
* Ezt a P és V primitívekkel oldjuk meg (sok processzor rendelkezik ilyen hardver utasításokkal).

# Perifériakezelési módszerek

• **Eszközszintű kezelés**: a perifériális eszköz fizikai sajátosságainak megfelelő illesztési felületet és utasításkészletet biztosítunk.

* Kis rendszerek, beépített rendszerek esetén előnyös.
* Jól kihasználhatók a processzor és a periféria sajátosságai.

• **Logikai kezelés**: általánosított illesztési felületeket és beviteli/kiviteli eljárásokat alkalmazunk.

## Eszközszintű perifériakezelés

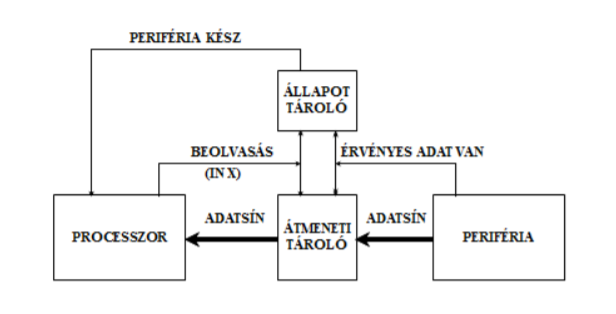
• FELTÉTEL NÉLKÜLI BEVITEL/KIVITEL

A processzor és a periféria nincs szinkronizálva.(A processzornak és a perifériának mindig rendelkezésre kell állnia.)

Egyszerű IN és OUT utasítások.

Pl: kapcsoló leolvassás, LED kigyújtás

• JELZŐBITES (FELTÉTELES) BEVITEL/KIVITEL



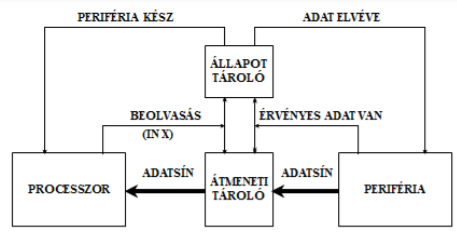
Az együttműködésért kizárólag a processzor felelős.

(A processzornak mindig rendelkezésre kell állnia.)

Példa: Hangbemenet, hálozati kártya, hang kártya

olvassás művelet=> az állapotregiszterrel az adatirány elkerülhető

• SZEMAFOROS (FELTÉTELES) BEVITEL/KIVITEL



## Logikai perifériakezelés

• Sokféle perifériális eszköz:

 általánosított beviteli/kiviteli eljárások és illesztési felület.

• I/O **processzor**ra, ill. **csatorná**ra alapozott kezelés:

 rögzített feladatú autonom modulok, felszabadítják a processzort a periféria részletes kezelése alól;

 I/O processzor: átviteli műveletekre optimalizált + általános adatfeldolgozási képesség.

# Ipari hálozatokban alkalmazott kommunikációs hálózatok

## RS485()

CCITT( International Telegraph and Telephone Consultative Committee)

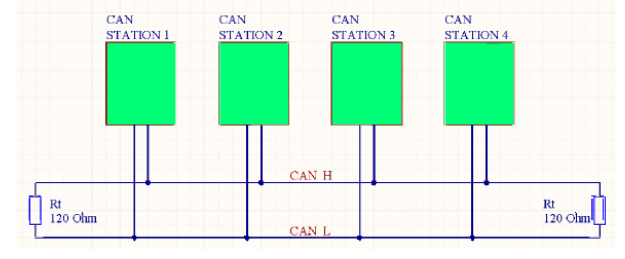
Az RS-485 buszkommunikáció alapja az UART és egy SN75176 vonali meghajtó IC segítségével.

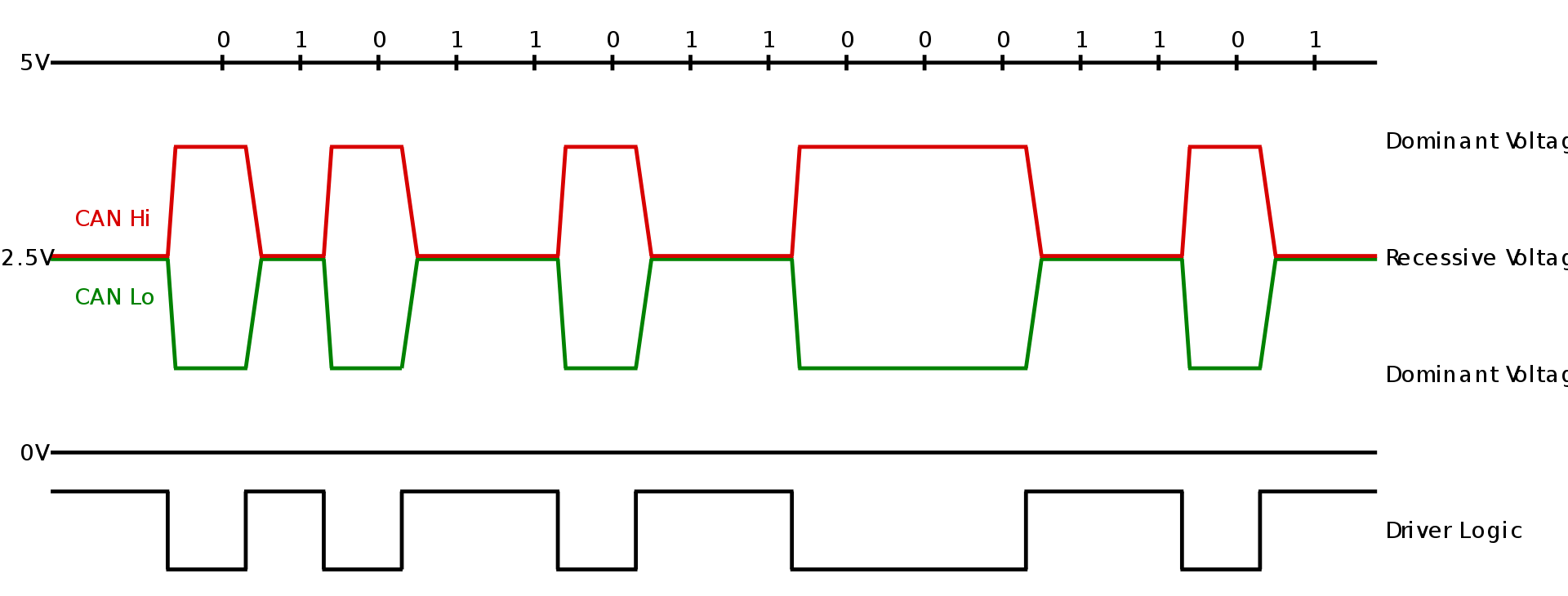
Két irányú half-duplex

Maximális adatátviteli sebessége 10m-ig 35Mbit/sec, 1200m-nél viszont már csak 100kbit/sec sebességet éri el

## CAN

### Fizika réteg

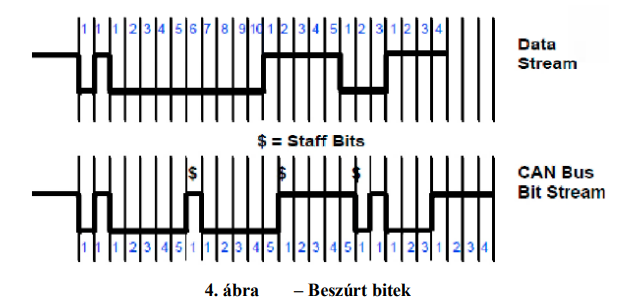




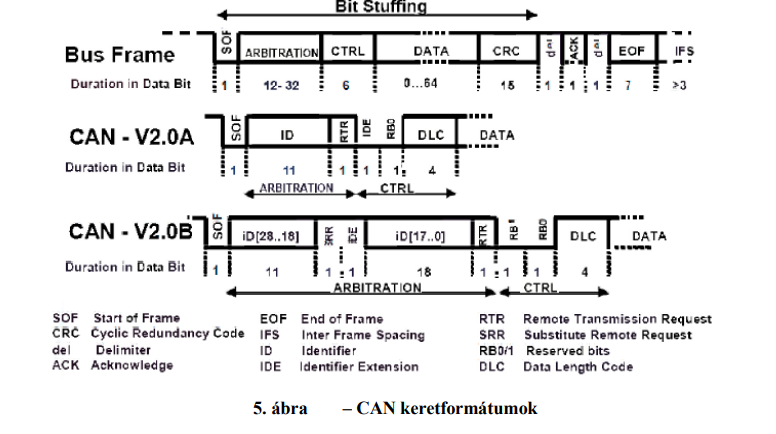
2 vezetéket használ: CAN-H és CAN-L

A szinkronizációnak két fajtája van. Az első esetben az üzenet első bitjére történik a szinkronizálás (SOF). A Start Of Frame (SOF) kezdetekor egy lefutó él keletkezik, amelyre történik meg az első szinkronizáció. Ezt „Hard Syncronization”- nak nevezzük.

A második szinkronizáció előtt meg kell nézni a beszúrt bitet. Abban az esetben, hogy az 5 megegyező bit követi egymást, akkor a CAN szabvány kimondja, hogy egy ellentétes bit értéket be kell szúrni az üzenetbe



### Üzenetek keretformátumai



Látható hogy az atbitrációs részt leszámítva a keretek nagyon hasonlóak. Általánosan kijelenthető egy keretről, hogy a SOF (Start of Frame) bittel kezdődik. Ezt követi az arbitrációs rész, vagyis az üzenet azonosítója, majd egy RTR (Remote Transmission Request) bit következik. Ha ennek a bitnek az értéke „1” akkor adatkérés, ha pedig „0”, akkor adat küldés fog történni. A kiterjesztett keretnél lehetőségünk nyílik beállítani még az IDE (Identifier Extension) bitet, mellyel kiválaszthatjuk, hogy a standard vagy a kiterjesztett formátumú keretet szeretnénk használni.

A következő rész a CTRL field, amely tartalmazza a DLC (Data Length Code) négybites részt, amely meghatározza, hogy az átvitt keret hány bájt adatottartalmazzon. Ennek értéke 0 és 8 között lehet. Ezt a részt követik az adatbájtok,amelyek maximum 64 biten helyezkedhetnek.

Ezután következik egy 15 bites CRC (Cyclic Redundancy Check) mező, amely rész az üzenet hibamentes célba juttatásának ellenőrzéséért felel. Az adó állomás előállít egy 15 bites CRC kódot, amelyet az üzenetbe illeszt, majd ezt a vevő visszakódolja, és ha megegyezik az adatbájtokkal, akkor a második ACK bitet domináns értékkel írja felül, amelyből az adó tudja, hogy az adatátvitel sikeresen megtörtént. Ha egyik vevő sem ezt a bitet domináns 0-ra, akkor ez a bit recesszív értékű marad, amely hatására küldő azt érzékeli, hogy sikertelen volt az átvitel.Az üzenet végét egy 7 bites EOF (End Of Frame) jelzi, melynek minden értéke 1–es.

### CAN üzenetek fajtái

* Data frame (Adat keret): Ez a csomag adatot juttat el az adótól a vevőig.
* Remote frame (Adat kérő keret): A keret lényege, hogy egy eszköz kezdeményezni tudja, hogy egy másik állomás üzenetet küldjön. Fontos, hogy az adatkérő keret azonosítójával azonos azonosítójú választ kell küldenie.
* Error frame (Hibakelező keret): Ezen a kereten keresztül tudnak visszajelezni az állomások, ha hibát észleltek.
* Overload frame (Túlcsordulást jelző keret): Ha egy csomópont már nem tud több üzenetet fogatni, akkor ezzel jelez vissza az adónak, hogy késleltesse a további adat küldését.

### Ipari Ethernet

Lười quá

# FPGA

FPGA felépítése. Logic cell, slice, CLB, IOB, DCM.

• Absztrakciós szintek, tervezési stratégiák.

• Hardver leírónyelvek általános jellemzői, fajtái.

Három fő részből épülnek fel:

* Általános célú logikai blokkok vagy logikai cellák
* Speciális funkciót(pl. I/O) ellátó dedikált blokkok

A korai FPGA-k esetén még csak az I/O blokk volt az egyetlen speciélis blokk, később megjelentek tobábbi dedikált blokkok, mint pl:

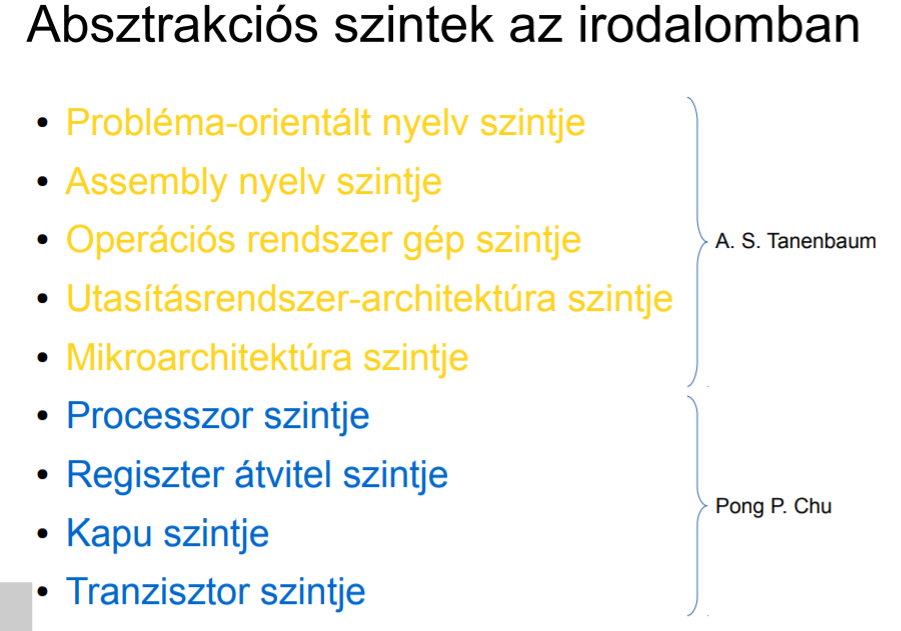
 órajel-menedzsment blokk főként órajel-szintézisre és kondicionálásra,

 memória blokk adatok tárolására,

 hardveres szorzó blokk,

* Belső huzalozás, kapcsolómátrixok

## Absztrakciós szintek:



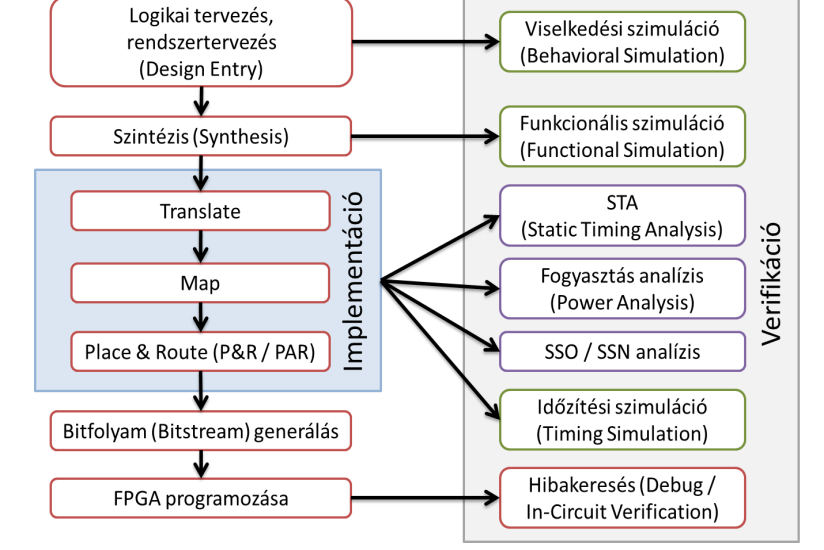
Absztrakciós szintek:

* Rendzser szint
* Algoritmikus szint
* Funkcionális szint (pl. Multiplexer, dekóder, ALU) RTL szint(VHDL, Verilog stb.)
* Logikai szint (Kapuk- Boole algebra)
* Fizikai áromkör szint ( tranziszritor szint)

## Tervezéri stratégiák

Az FPGA alapú rendszertervezés folyamatát a Xilinx Vivado/ISE fejlesztő környezetben történő .

1. Design Entr- Tervezés „belépési” szintje (pl. HDL forrás)
2. Szinmuláció
3. Szintézis és Implementáció
4. Időzítési analízis
5. Bitfolyam létrehozása



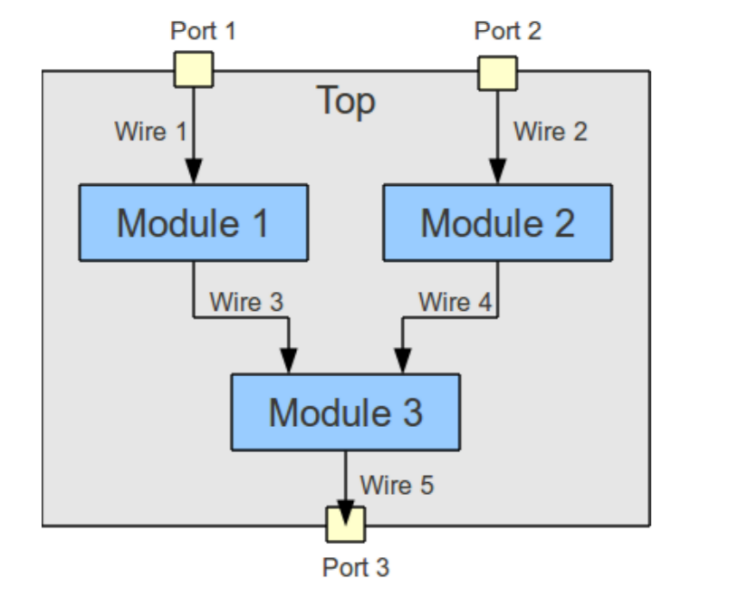
## VHDL

VHDL=VHSIC HDL( Very High Speed Integrated Circuit Hardware Description Language)

VHDL hardver leíró nyelv három alapegysége

* Architektúra: HW egység funkciója és szerkezete
* Interfész(entity): HW egységek közötti kommunikáció(protokoll) jellemzői
* Konfiguráció: architektúrák és interfészek egymáshoz rendelés

Kérdés: Mutassa be egy VHDL kód főbb szintaktikai egységeit (entity, architecture, signal, variable, process) hivatkozva az ábrán látható jelölésekre!



library ieee;

use ieee.std\_logic\_1164.all;

--------------------------------------------------

entity and\_of\_inverter is

port (

port1 : in std\_logic;

port2 : in std\_logic;

port3 : out std\_logic

);

end and\_of\_inverter;

--------------------------------------------------

architecture dataflow of and\_of\_inverter is

signal port1\_inv : std\_logic;

signal port2\_inv : std\_logic;

begin

port1\_inv <= not port1;

port2\_inv <= not port2;

port3 <= port1\_inv and port2\_inv;

end dataflow;