Biztonsági folyamatirányító rendszerek szoftvere

Tartalom

- Szoftverek szerepe a folyamatirányító rendszerekben
- Szoftverek megbízhatósága
- Szoftver életciklus
- Biztonsági szoftverekkel szemben támasztott követelmények
- Szoftverírás
- Tesztelés

Programozott irányítórendszerek

- Célgépek
  - Nincs operációs rendszer
  - Egyszerű szoftver
  - Pl. egy-chipes mikrokontrollerek
- Univerzális alkalmazású rendszerek
  - Moduláris hardver (általában kártya rendszerű)
  - Tagolt szoftverfelépítés

Tagolt szoftverfelépítés

KONFIGURÁCIÓ

Az adott alkalmazási hely konfigurációja; általában adatbázis

ALKALMAZÓI/FELHASZNÁLÓI SZOFTVER

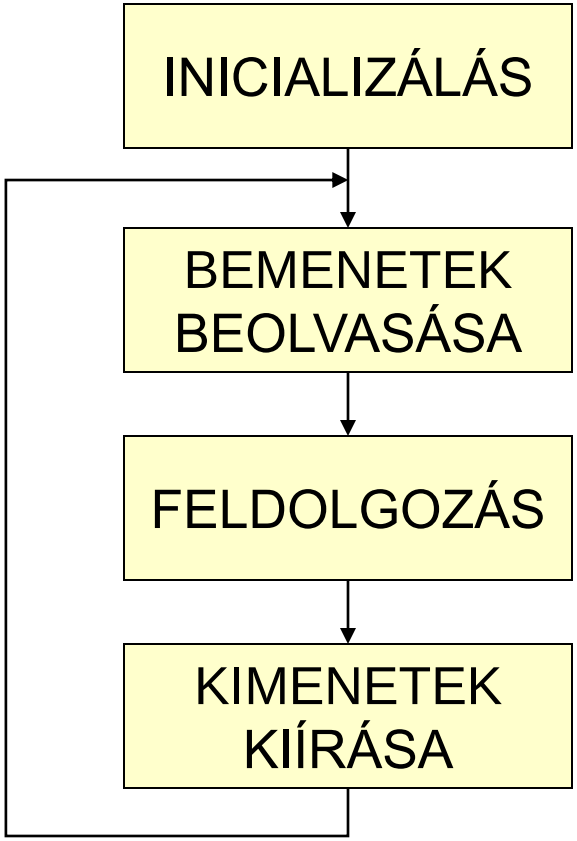
Alkalmazás-specifikus funkciók,
pl. általános jelzőlámpa vezérlés

OPERÁCIÓS RENDSZER

A folyamatirányítással kapcsolatos általános alapfunkciók

Az operációs rendszer feladatai

A folyamatirányító rendszerek szoftvere általában ciklikus működésű:



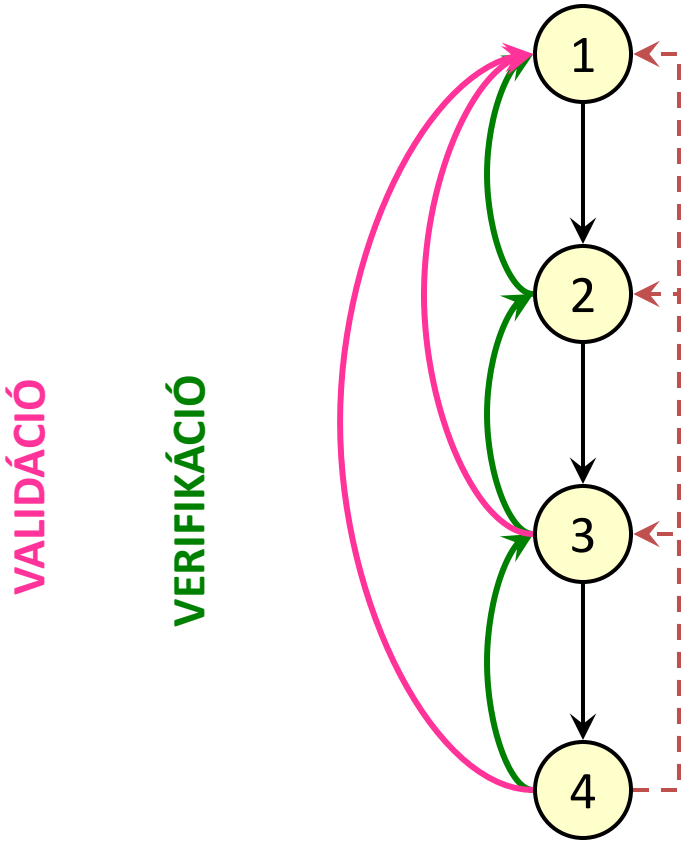
Szoftverek megbízhatósága

- A szoftver akkor működőképes – és így biztonságos –, ha a követelményeket (specifikáció) jól fogalmaztuk meg, és a megvalósítás (implementáció) is helyes.

- A szoftverek megbízhatósága (helyes működésének valószínűsége) **független az időtől** (amennyiben nem változtattuk meg).

- Szoftverek esetében nem beszélhetünk meghibásodásról:
  - A szoftver megbízhatóságát „csak” az eredeti, szisztematikus, specifikációs, tervezési és megvalósítási hibák csökkentik.
  - A szoftvert tároló alkatrész meghibásodása hardver-meghibásodás.
  - DE! Emberi beavatkozással egy jó szoftvert is el lehet rontani, például:
  - **Újra-bekerülési hiba**: átlagosan minden harmadik hibajavítással újabb hibát idézünk elő.

Szoftver-életciklus



SPECIFIKÁCIÓ

A szoftver feladatának meghatározása

SZOFTVER TERVEZÉS

KÓDOLÁS

KARBANTARTÁS

- Hibák javítása
- Módosítások

Követelmények a biztonsági szoftverekkel szemben

Cél: **hibamentesség**.

Szoftver-megbízhatóságot növelő módszerek

- Jól strukturáltság
- Moduláris felépítés
- Áttekinthetőség – szükséges az ellenőrzéshez is
  - Modulonként kevés be/kimenet (lehetőleg 1-1) → könnyű tesztelni
  - Jól definiált interfészek
  - Feltétel nélküli ugrások (GOTO) kerülése
  - Tesztelhetőség kialakítása – „tesztelés-barát tervezés”
- Jól dokumentáltság
  - Funkciók leírása
  - Interfészek leírása
- Nem-biztonsági részek: arra kell ügyelni, hogy a nem-biztonsági rész semmilyen módon ne legyen hatással a biztonsági részekre – **visszahatásmentesség**.

Szoftverírás

- Programozási koncepciók
  - **Defenzív programozás**: számítok arra, hogy a programozás során hibákat fogok elkövetni.
    - Passzív ellenőrzések: pl. ellenőrző összegek, hihetőség-vizsgálat
    - Aktív ellenőrzések: adatáramlástól független ellenőrzés, így tesztelhetők a ritkán aktív lefutási ágak is.
  - CASE: Computer Aided Software Engineering
    - Automatikus programgenerátorok: a generátor program helyességét kell bizonyítani.

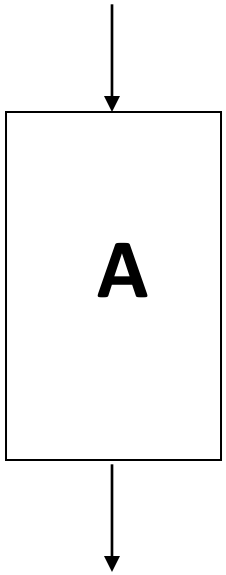
Szoftver tesztelés

- Az eredeti hibák kiküszöbölésének leggyakoribb eljárása a **hibaeltávolítás**. Lépései:
  - tesztelés
  - diagnózis
  - javítás.

- **Tesztelés**
  - Statikus: a rendszer működtetése nélküli tesztelés. Ez végrehajtható
    - a rendszeren magán vagy
    - a rendszer alkalmas modelljén végrehajtott vizsgálatokkal.
  - Dinamikus tesztelés: a rendszer működtetése révén
    - megfelelőségi tesztek / pozitív tesztek
    - hibakereső tesztelés
    - funkcionális/black boksz tesztek és strukturális tesztek

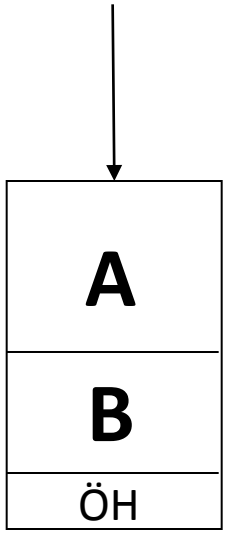
Biztonsági architektúrák

- 1 hardver, 1 szoftver
  - Lehet, h. a szoftver jól van megírva,
  - de a hardver véletlen hibái ellen semmi nem véd.



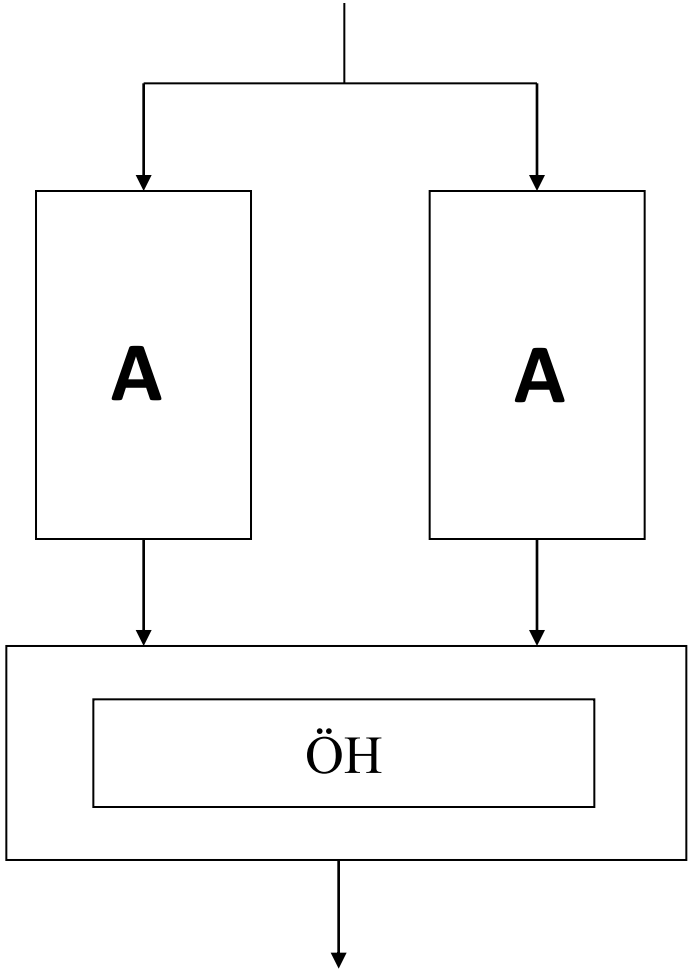
Biztonsági architektúrák

- 1 hardver, 2 szoftver
  - Két különböző (diverz) szoftver fut (A és B) ugyanazon a gépen.
  - Két szoftver futhat párhuzamosan, vagy egymás után.
  - Az összehasonlító felfedi, ha a két szoftver mást mond → felfedhetők a specifikációs és programozási hibák
  - Mivel a két program eltérő, ezért egy HW hiba nem egyformán hat a két szoftverre, így a véletlen HW hibák is felfedhetők
- Pl. Ebilock (svéd) elektronikus bb.

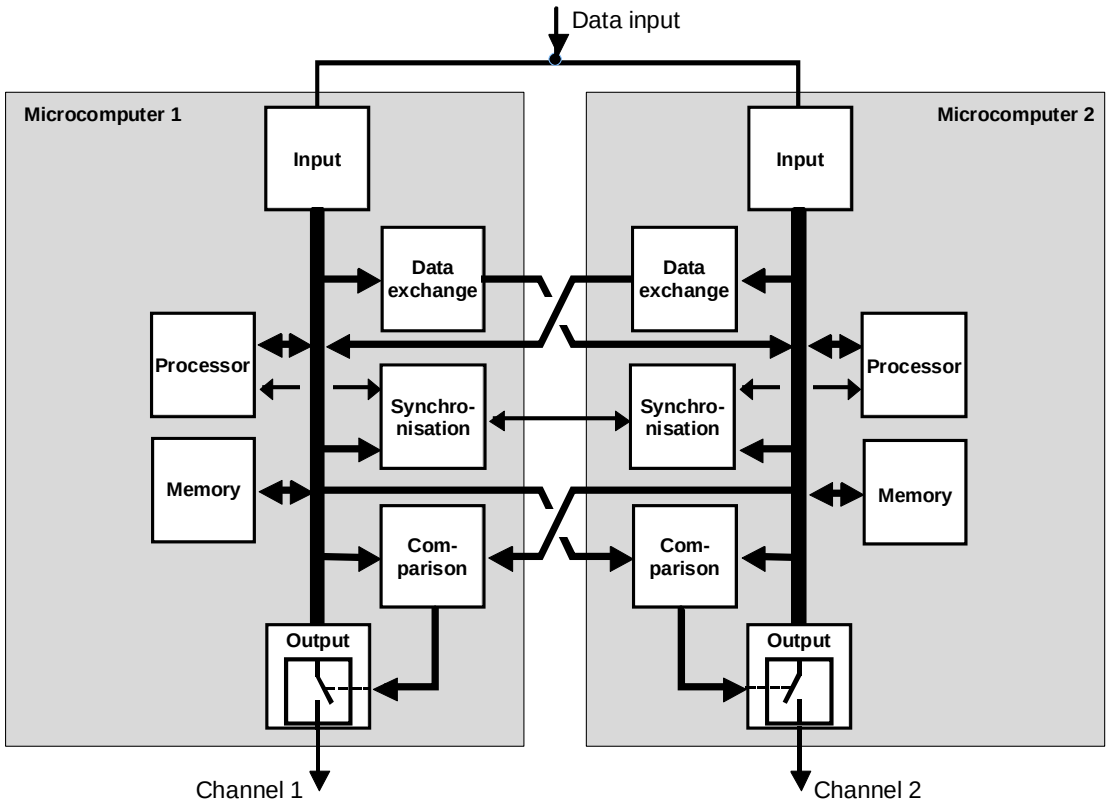


Biztonsági architektúrák

- 2 hardver, 1 szoftver
  - 2-ből 2 rendszer (2v2)
  - Véd a hardver véletlen meghibásodásai ellen
  - A szoftvert „eleve jóra” kell készíteni, mert az architektúra nem véd a specifikációs és programozási hibák ellen.
- Pl. Siemens SIMIS-elv



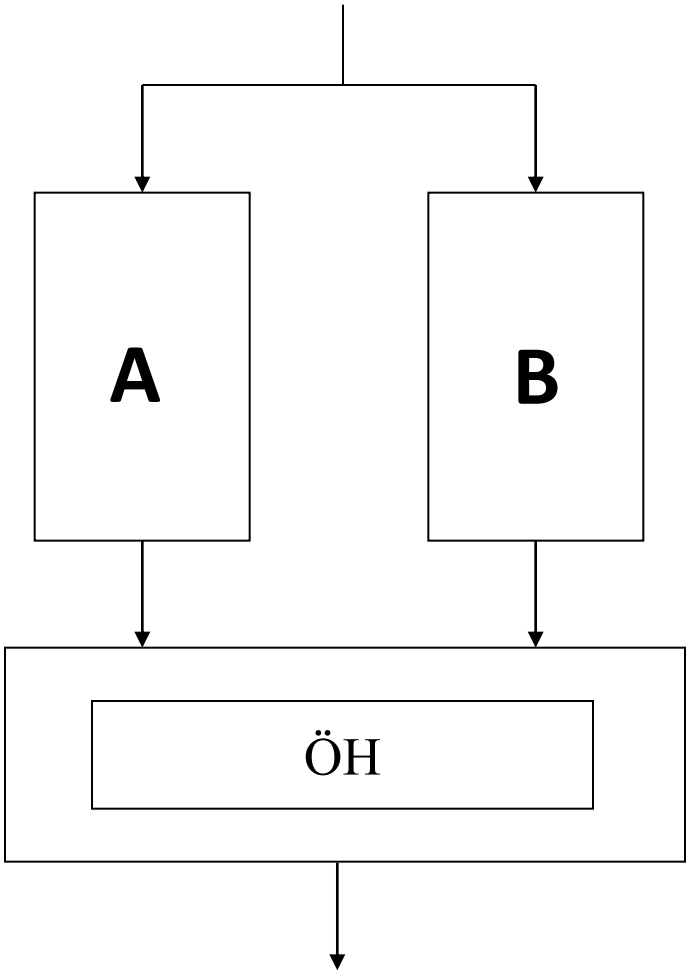
Simis-elv
2v2 konfiguráció



- 2 mikroszámítógép
- óraszinkron
- utasításszinkron
- a mikroszámítógépektől független 2 összehasonlító
- összehasonlítja a kimeneteket és a
- processzor tartalmakat (memóriát)

Biztonsági architektúrák

- 2 hardver, 2 szoftver
  - Az architektúra véd a véletlen hardver hibák ellen és
  - a szoftver hibák ellen.
  - A két csatornában eltérő specifikációval, eltérő programnyelven kifejlesztett programok futnak
- Pl. Thales Elektra

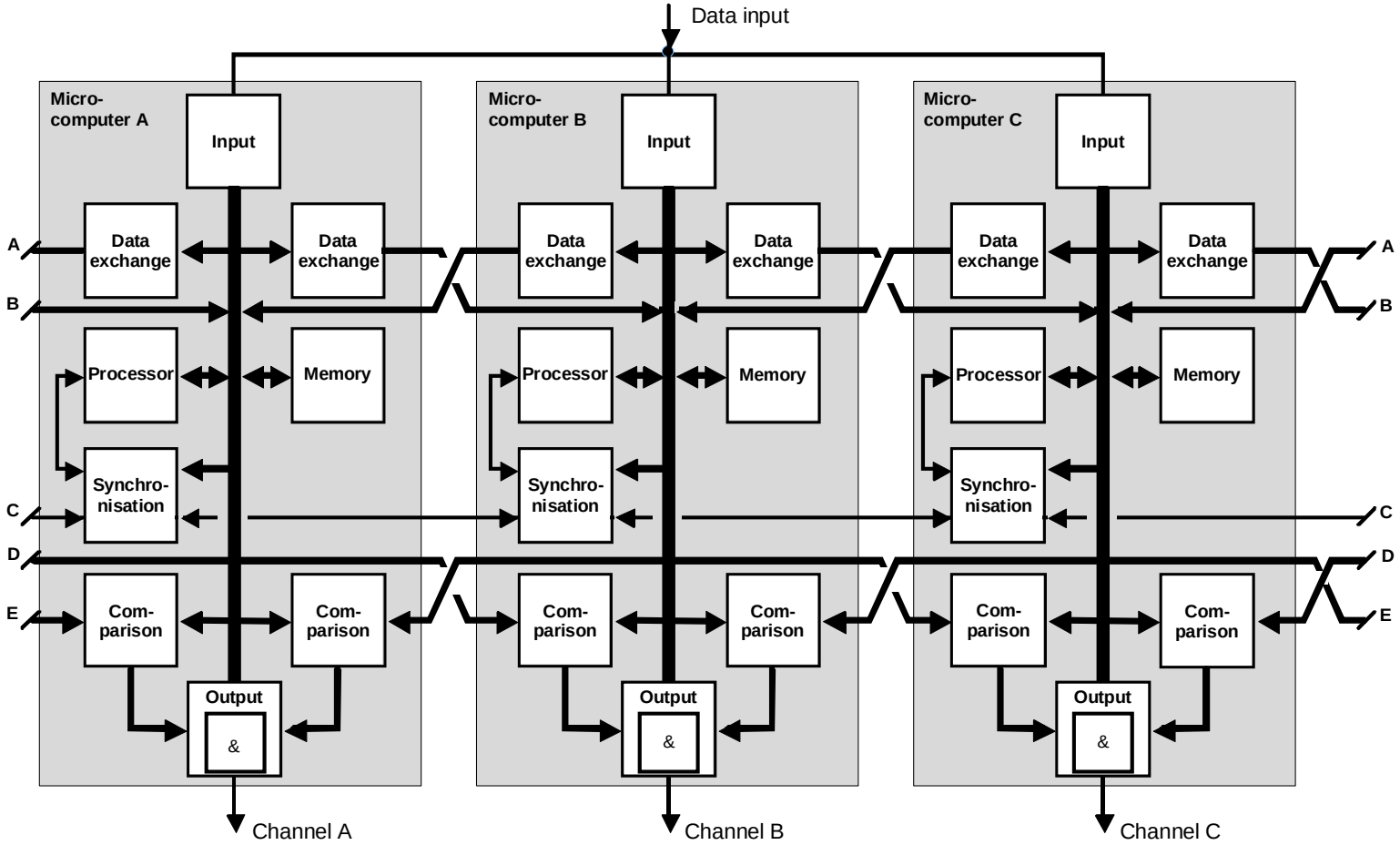


Rendelkezésre állás

- Az eddig bemutatott architektúrák biztonságosak ugyan, de már egy hiba esetén is működésképtelenek.
- Módszerek a rendelkezésre állás növelésére:
  1. Tartalékolás
     - Egycsatornás rendszer: redundancia
     - 2v2 → 2×(2v2) (pl. SIMIS IS: SIMIS PC)
     - 2v2 → 2v3 (pl. SIMIS IS: ECC számítógépek)

2v3

- 3 mikroszámítógép
- 6 független összehasonlító egység
- a 3. csatorna is aktív
- hiba esetén a hibás csatorna/modul leáll
- tovább működik 2v2 rendszerként

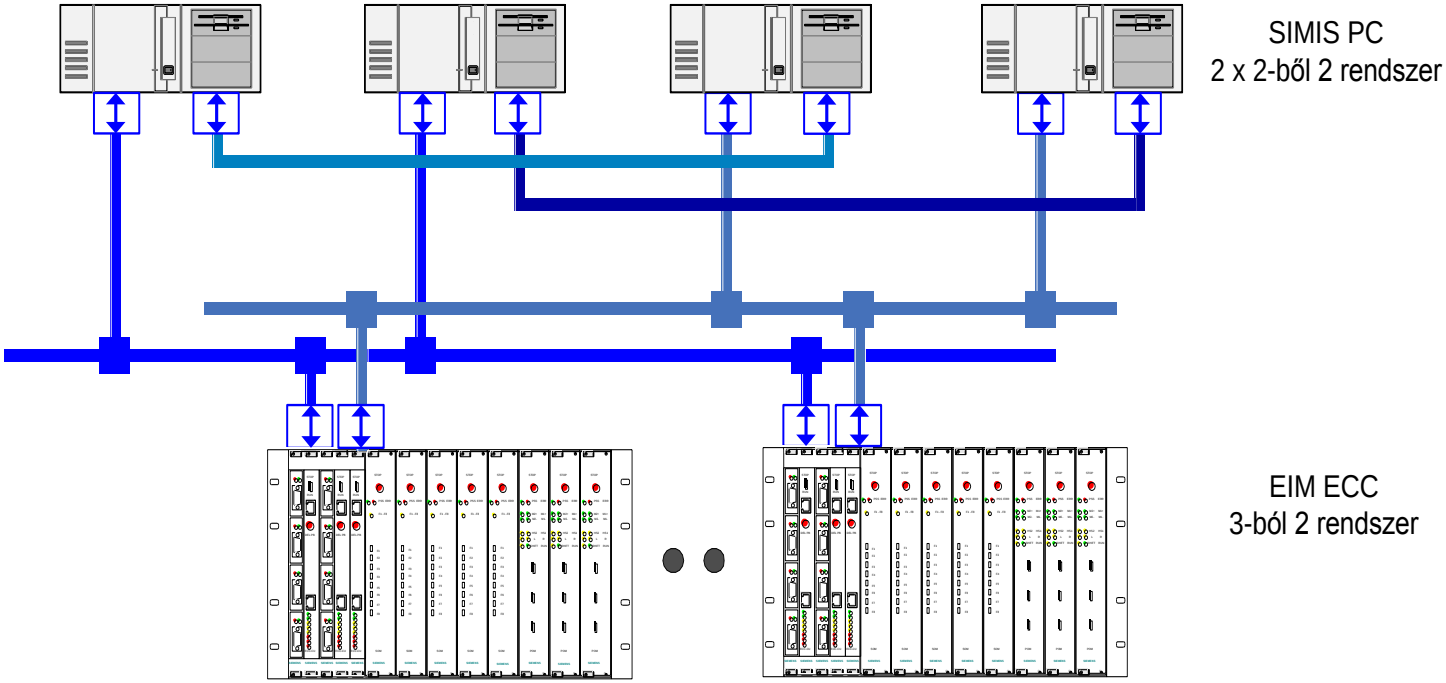


- 3 mikroszámítógép
- 6 független összehasonlító egység
- a 3. csatorna is aktív
- hiba esetén a hibás csatorna/modul leáll
- tovább működik 2v2 rendszerként

SIMIS PC

- 2×(2v2) diverz számítógéprendszer
- Kereskedelmi forgalomban kapható számítógépek és operációs rendszerek
  - AMD, Intel alaplap/processzor
  - Win2000, Linux operációs rendszerek
  - a diverzitás, és az ECC-kben való összehasonlítás miatt lehetséges ezek alkalmazása
- Összehasonlítás az ECC-kben történik (időablakkal)

SIMIS PC



- 2×(2v2) diverz számítógéprendszer
- Kereskedelmi forgalomban kapható számítógépek és operációs rendszerek
  - AMD, Intel alaplap/processzor
  - Win2000, Linux operációs rendszerek
  - a diverzitás, és az ECC-kben való összehasonlítás miatt lehetséges ezek alkalmazása
- Összehasonlítás az ECC-kben történik (időablakkal)

Alcatel – ELEKTRA 1

- Két diverz szoftver csatorna
- A két csatornában lévő gépeknek egyenként van 2-es vagy 3-as redundanciájuk
- A biztosítóberendezés szintekre oszlik:
  - Kezelés/visszajelentés
  - Központi szint
  - Perifériavezérlés
- Az egyes számítógépek és a perifériavezérlőket pont-pont kábelezés köti össze
- Hardverbázis: 486-os processzorok
- Vágányúti logika: táblázatos, vágányutas elvű

ELEKTRA
Rendszerfelépítés