Biztonságkritikus rendszerek architektúrája

Rendszertervezés és -integráció előadás

dr. Majzik István

Méréstechnika és Információs Rendszerek Tanszék

Célkitűzés

Hibahatások ellenére a veszély elkerülése

Fail-safe működés

Meghibásodás esetén is biztonságos működés

Fail-stop működés

- A megállás (lekapcsolás) biztonságos állapot
- Detektált hiba esetén le kell állítani a rendszert
- Hibadetektálás a kritikus feladat

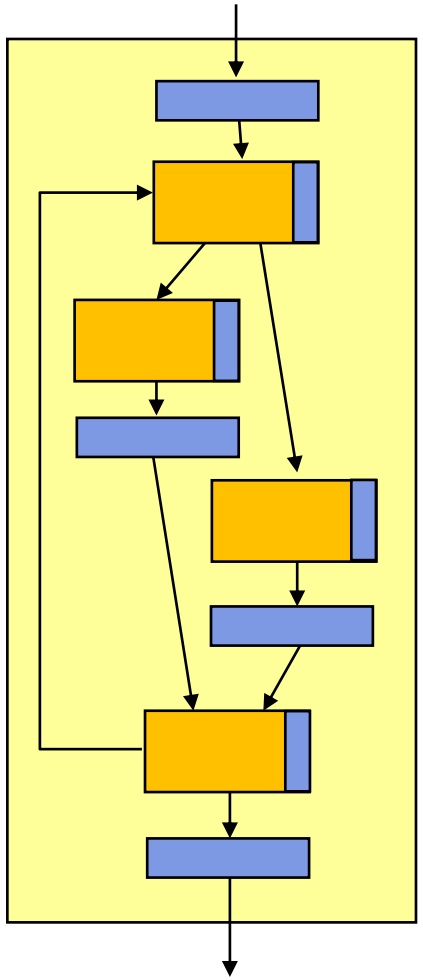
Fail-operational működés

- A megállás (lekapcsolás) nem biztonságos állapot
- Detektált hiba esetén is szükséges szolgáltatás
  - teljes, vagy
  - csökkentett (degradált)
- Hibatűrés szükséges

Jellegzetes megoldások fail-stop működéshez

Egycsatornás feldolgozás önellenőrzéssel

- Egy feldolgozási (végrehajtási) folyamat
- Ütemezett hardver öntesztek
  - Induláskor: Részletes önteszt
  - Futás közben: Lappangó állandósult hibák detektálása
  - Ajánlott többféle módon
- Rendszeres szoftver önellenőrzés
  - Alkalmazásfüggő hibadetektálás
  - Vezérlési folyam, adatfeldolgozás ellenőrzése
- Hátrányok:
  - Hibafedés korlátos: pedig a SIL teljesítése ettől függ!
  - Hibakezelés megvalósítása (pl. lekapcsolás) nem megbízható

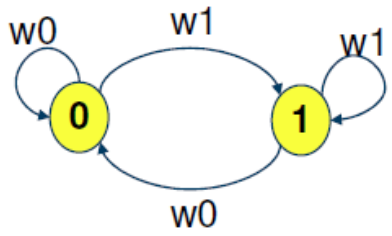


Hibadetektálás megvalósítása

- Alkalmazásfüggő (ad-hoc) technikák
  - Hihetőség vizsgálat (elfogadható tartományok)
  - Időzítések ellenőrzése (túl korán, túl későn …)
  - Visszahelyettesítés (inverz függvénnyel)
  - Struktúra ellenőrzése (pl. kettős láncolással)
- Alkalmazásfüggetlen mechanizmusok (áttekintés)
  - Hardver támogatású ellenőrzések
    - CPU szintű: Illegális utasításkód, user/supervisor mód stb.
    - MMU szintű: Memóriatartományok védelme
  - OS szintű ellenőrzések
    - Rendszerhívások érvénytelen paraméterei
    - Erőforrások hozzáférésének védelme

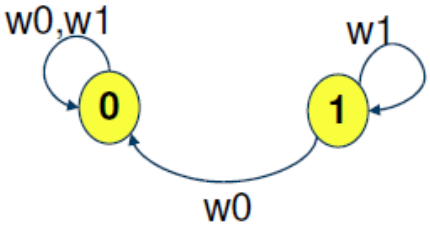
Példa: Memória tesztelése

Hibátlan cella állapotai:

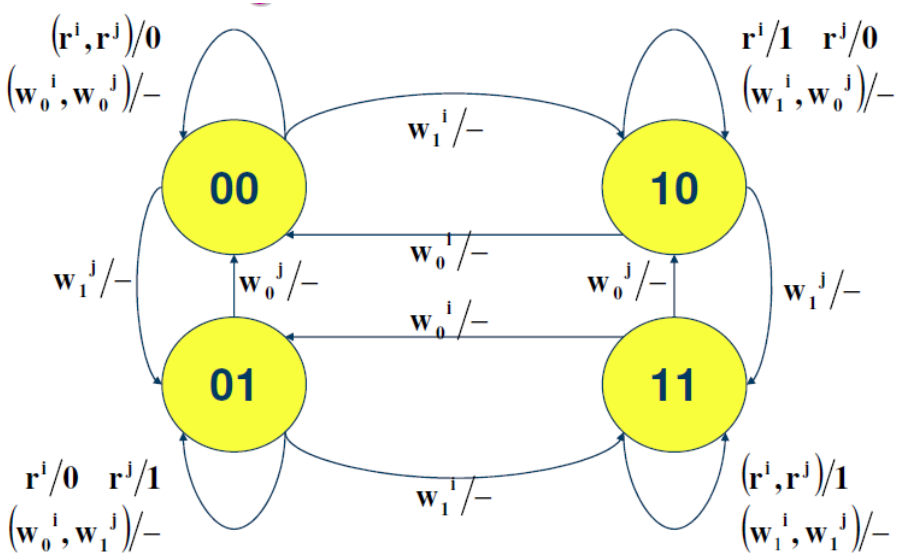


Leragadási hibák esetén:

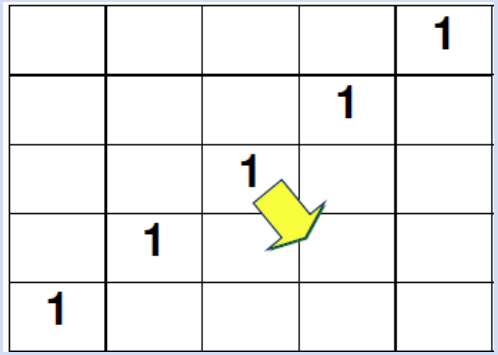
Tranzíciós hiba esetén (w1):



Átmenetek összeragadás tesztjéhez:



Teszt: „Marching” algoritmusok (w/r):



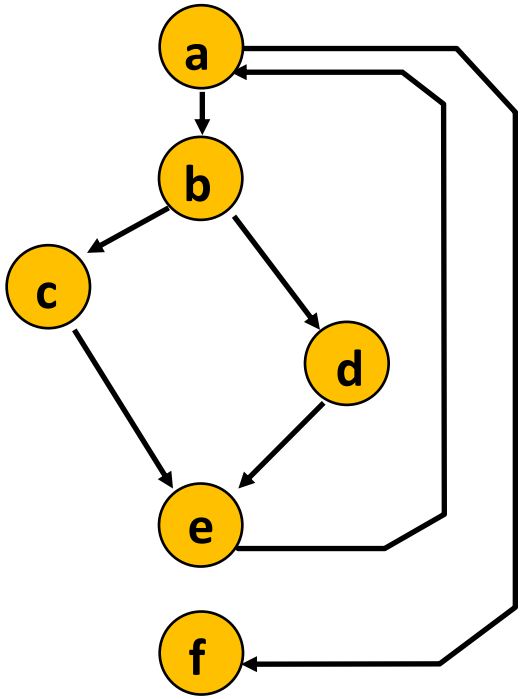
Példa: Szoftver utasítás-szekvencia ellenőrzése

- Végrehajtási útvonalak ellenőrzése (monitor)
  - Referencia: Vezérlési gráf a forráskód alapján

Forráskód:

```
a:  for (i=0; i<MAX; i++) {
b:        if (i==a) {
c:              n=n-i;
          } else {
d:              m=m-i;
          }
e:        printf("%d\n",n);
    }
f:  printf("Ready.")
```

Vezérlési gráf:



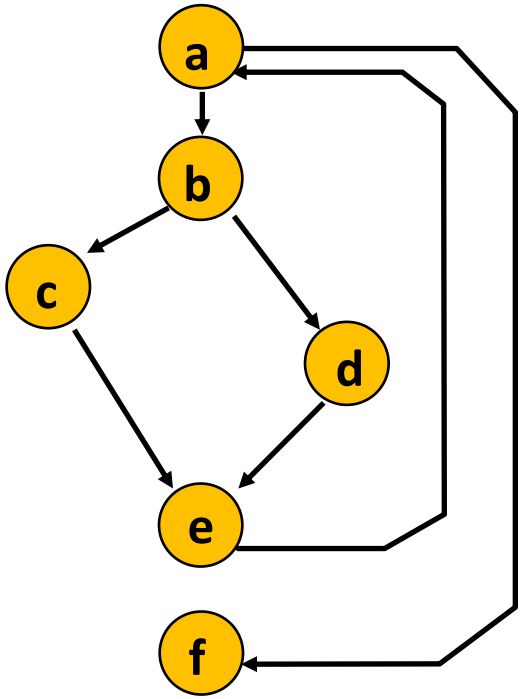
Példa: Szoftver utasítás-szekvencia ellenőrzése

- Végrehajtási útvonalak ellenőrzése (monitor)
  - Referencia: Vezérlési gráf a forráskód alapján
  - Aktuális futás: Jelzőszámok alapján ellenőrizhető

Felműszerezett forráskód:

```
a:  S(a); for (i=0; i<MAX; i++) {
b:      S(b); if (i==a) {
c:          S(c); n=n-i;
        } else {
d:          S(d); m=m-i;
        }
e:      S(e); printf("%d\n",n);
    }
f:  S(f); printf("Ready.")
```

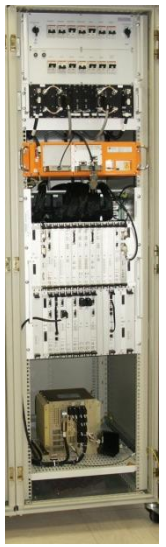
Vezérlési gráf (referencia):



Példa: SAFEDMI mozdonyvezetői kezelőfelület

Mozdony-vezető

DMI



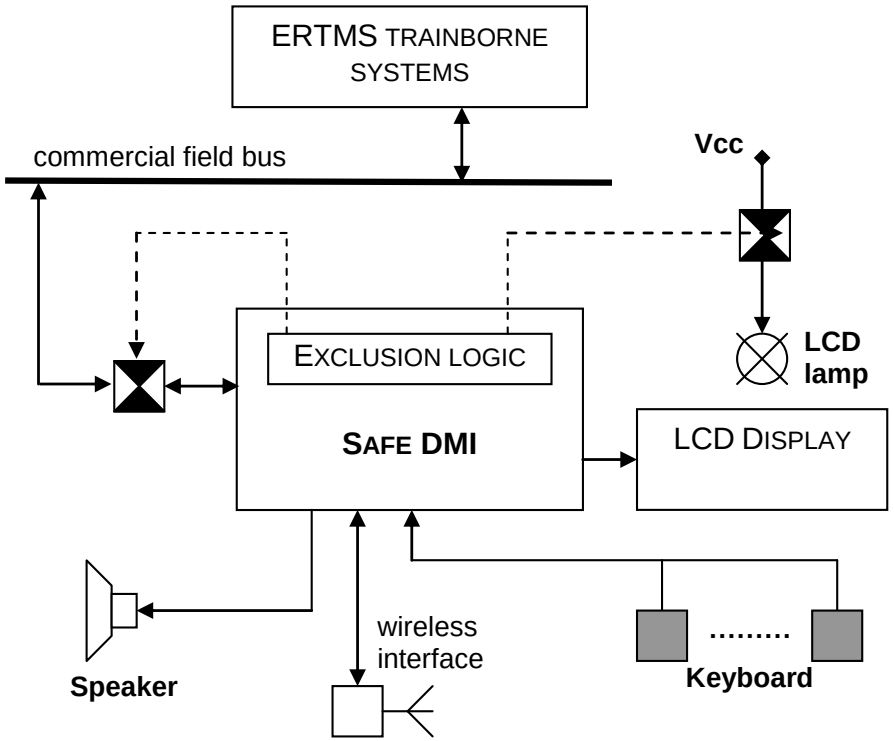
EVC
European Vital Computer

Maintenance Centre

- Safety Integrity Level 2
  - Információ megjelenítés
  - Parancs feldolgozás
  - EVC kommunikáció
- Biztonságos vezeték nélküli kommunikáció
  - Konfiguráció
  - Diagnosztika
  - Szoftver frissítés

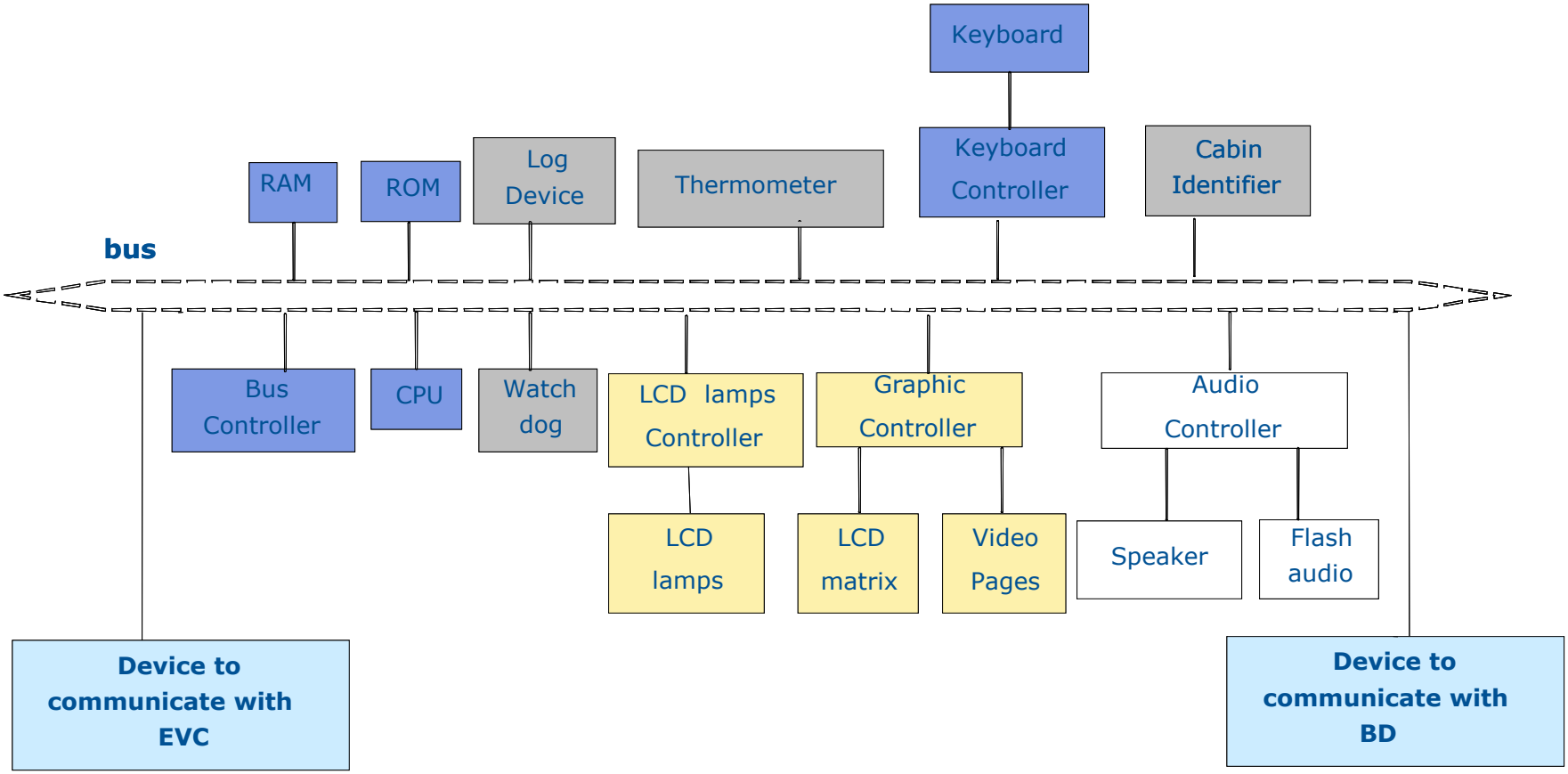
Példa: SAFEDMI hardver architektúra

- Fail-stop működés
- Reaktív fail-safety (hibadetektálás és -kezelés)
- Generikus (COTS) hardver elemek; a hibadetektálás és hibakezelés megvalósítása szoftver alapú megoldásokkal



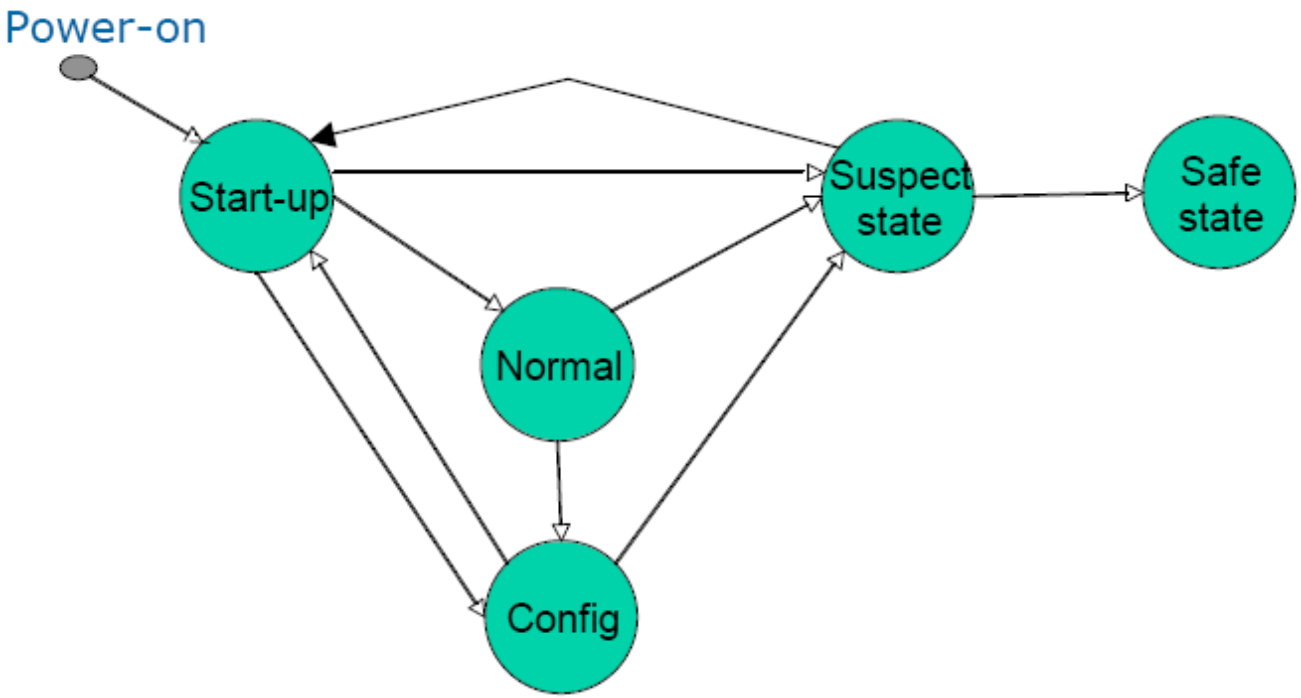
Példa: SAFEDMI hardver architektúra

Komponensek:



Példa: SAFEDMI hibakezelés

- Alap üzemmódok:
  - Startup, Normal, Configuration, Safe módok
- Hibakezelés: Suspect állapot
  - Átmeneti állapot
  - Hibaszámlálás: Tranziens – állandósult hiba megkülönböztetéséhez



Példa: SAFEDMI elindítás utáni öntesztek

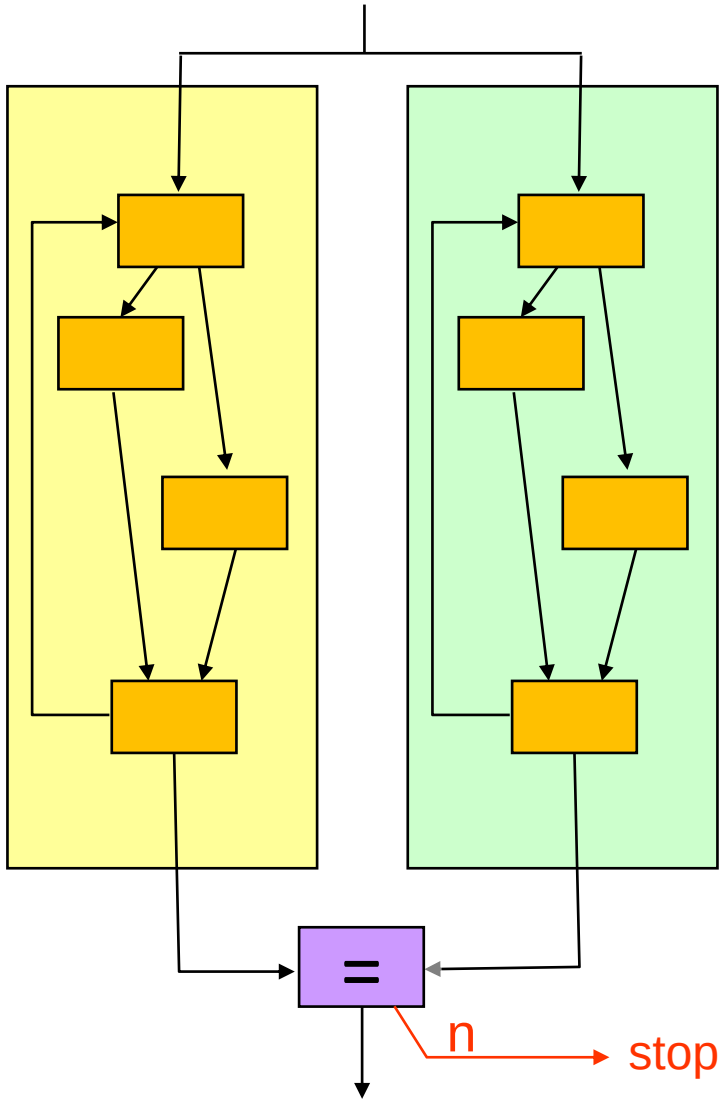
- CPU öntesztelése:
  - Külső watchdog áramkör (periodikus életjelek)
  - Alap funkcionalitás → Komplex funkciók (aritmetikai és logikai egység, utasításdekódolás, regiszter dekódolás, belső buszok megmozgatása)
- Memória tesztelése:
  - „Marching” algoritmusok (leragadás, összeragadás tesztelése)
- Szoftver integritás (EPROM tartalom):
  - Hibadetektáló kódolás
- Perifériák tesztelése:
  - Operátori (mozdonyvezetői) segítséggel (pl. hang érzékelés)

Példa: SAFEDMI működés közbeni ellenőrzések

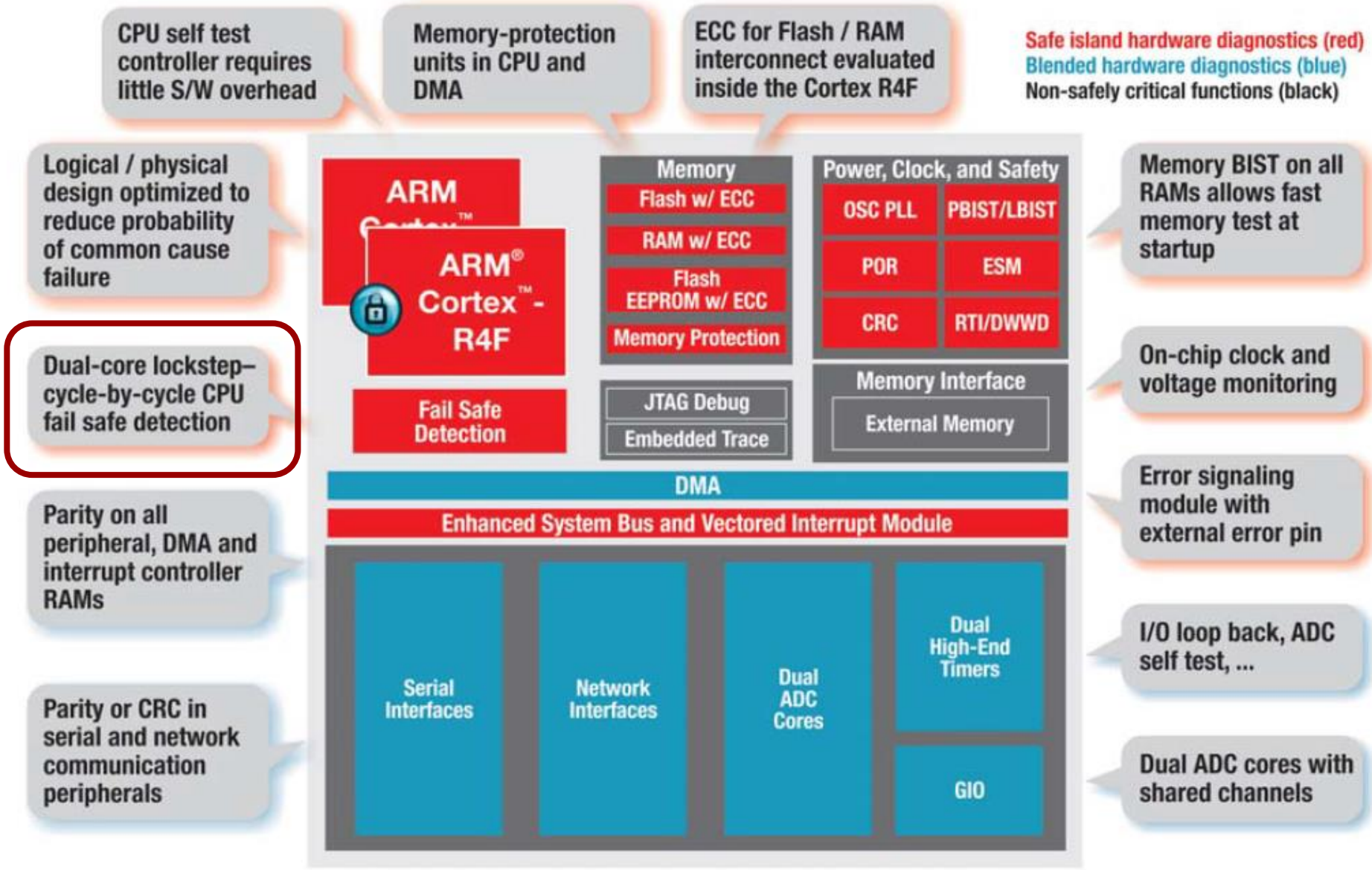
- Hardver perifériák:
  - Ütemezett tesztek: Kisebb erőforrásigényűek (CPU, memória, …)
  - Adat elfogadhatósági tesztek: I/O műveletek esetén
- Kommunikációs és konfigurációs funkciók:
  - Adat elfogadhatósági tesztek (hihetőségi vizsgálatok)
  - Hibadetektáló és hibajavító kódok
    - Adott hibagyakoriság fölött a kapcsolat bontása
- Üzemmód váltások és kezelői bemenetek feldolgozása:
  - Végrehajtási útvonalak (vezérlési gráf) ellenőrzése
  - Időtúllépés (time-out) ellenőrzése
  - Megerősített parancskiadás (jóváhagyás szükséges)
- Grafikus adatmegjelenítés (bitképek összeállítása):
  - Duplikált végrehajtás és összehasonlítás

Két- vagy többcsatornás feldolgozás komparálással

- Két vagy több feldolgozási csatorna
  - Közös bemenet
  - Kimenetek komparálása
  - Eltérés esetén leállás
- Nagy hibafedés
- Komparátor kritikus elem
  - De egyszerű!
- Speciális „komparálás”:
  - Operátori ellenőrzés
- Hátrányok:
  - Közös eredetű hiba?
  - Hosszú lappangási idő

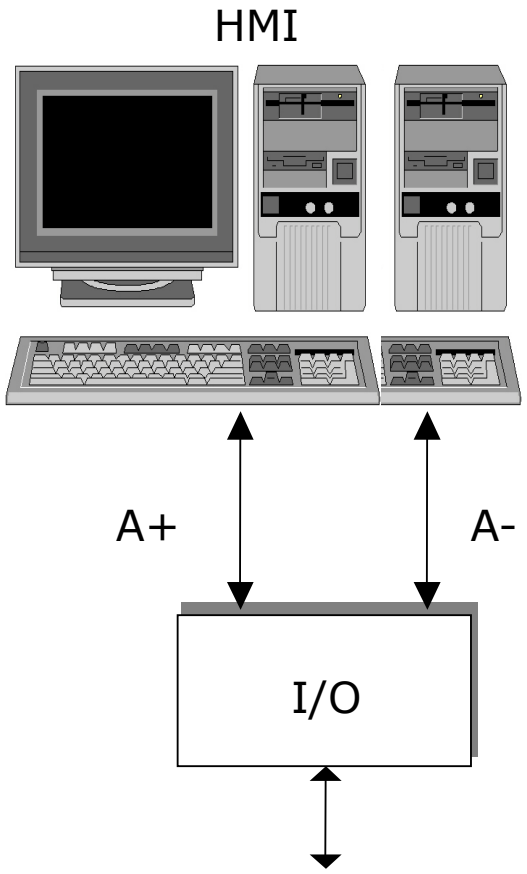


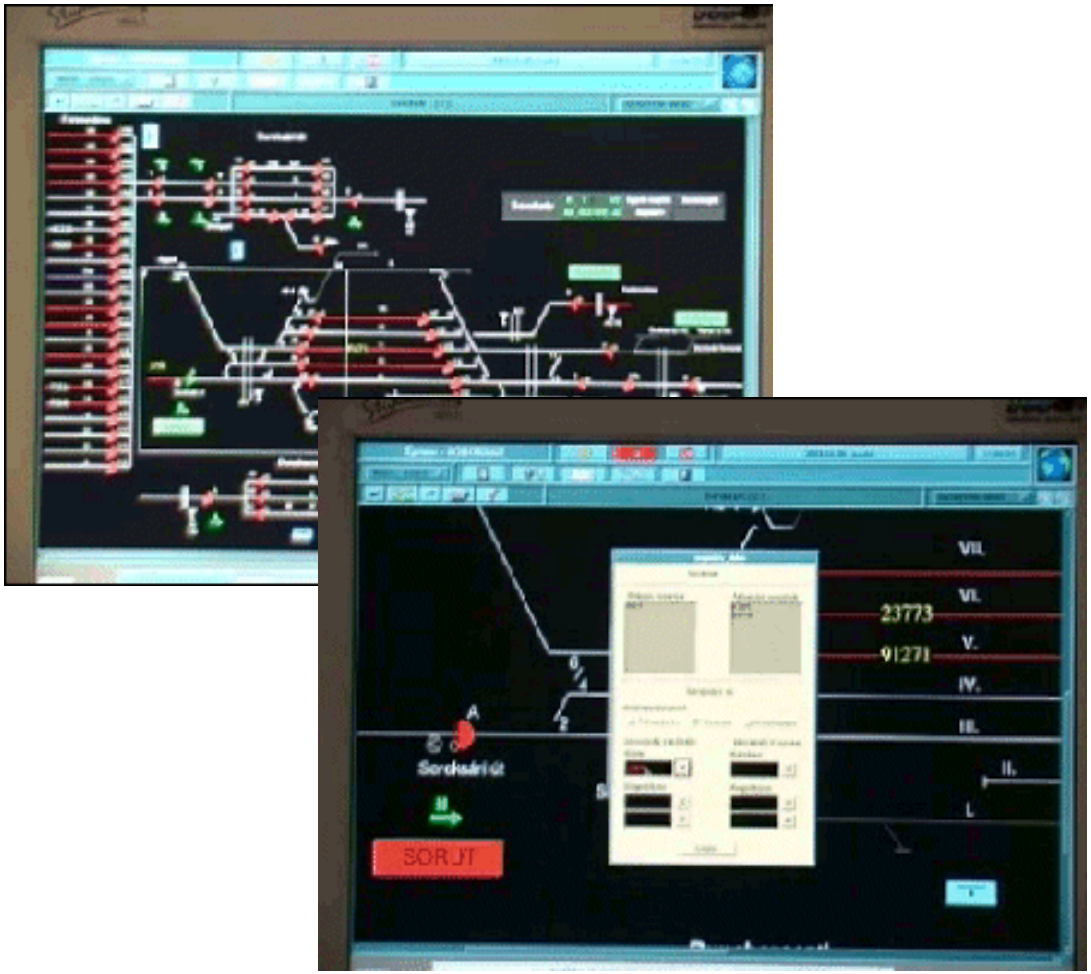
Példa: Safety microcontroller (itt: TI Hercules)



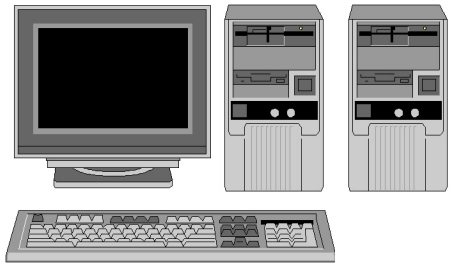
Példa: Egy SCADA rendszer

- Supervisory Control and Data Acquisition

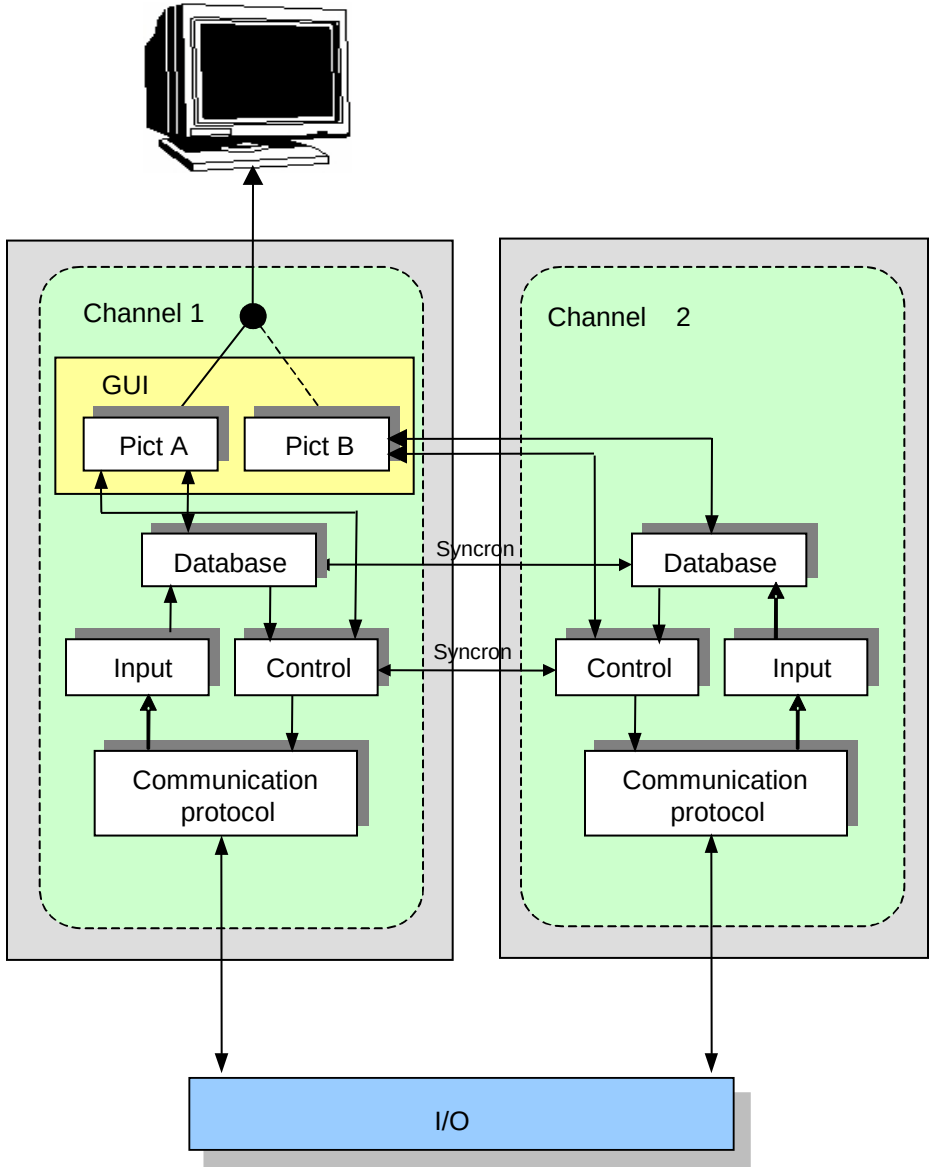




Példa: SCADA szoftver architektúra

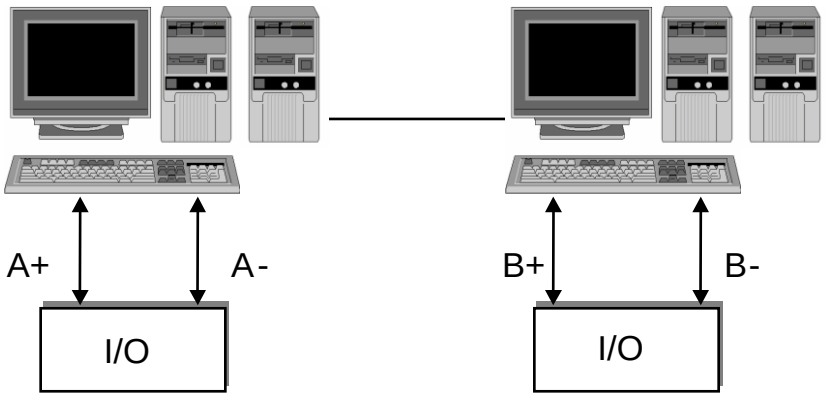


- Két csatorna
- A két csatorna által számolt bitmap váltakozó megjelenítése (operátor az összehasonlító; eltérés esetén villogás)
- Szinkronizáció: Belső hibadetektálás (mielőtt a kimenetre kerülne)



Példa: SCADA telepítési opciók

- Két csatorna ugyanazon a számítógépen: Közös platform hatásainak kezelése
  - Statikusan linkelt szoftver modulok
  - Időben, memóriában és diszken elkülönülő végrehajtás
  - Diverz adattárolás
    - Bináris adatokra (jelek): Inverz adatábrázolás (ponált/negált)
    - Technológiai adatbázis különböző indexelése
- Két csatorna különböző szervereken
  - Szinkronizáció dedikált belső hálózaton
- Rendelkezésre állás növelése (hibatűrés):
  - Újraindulás (tranziens hibára)
  - Kétszer „2-ből 2” séma



Példa: SCADA hibadetektálás összefoglalása

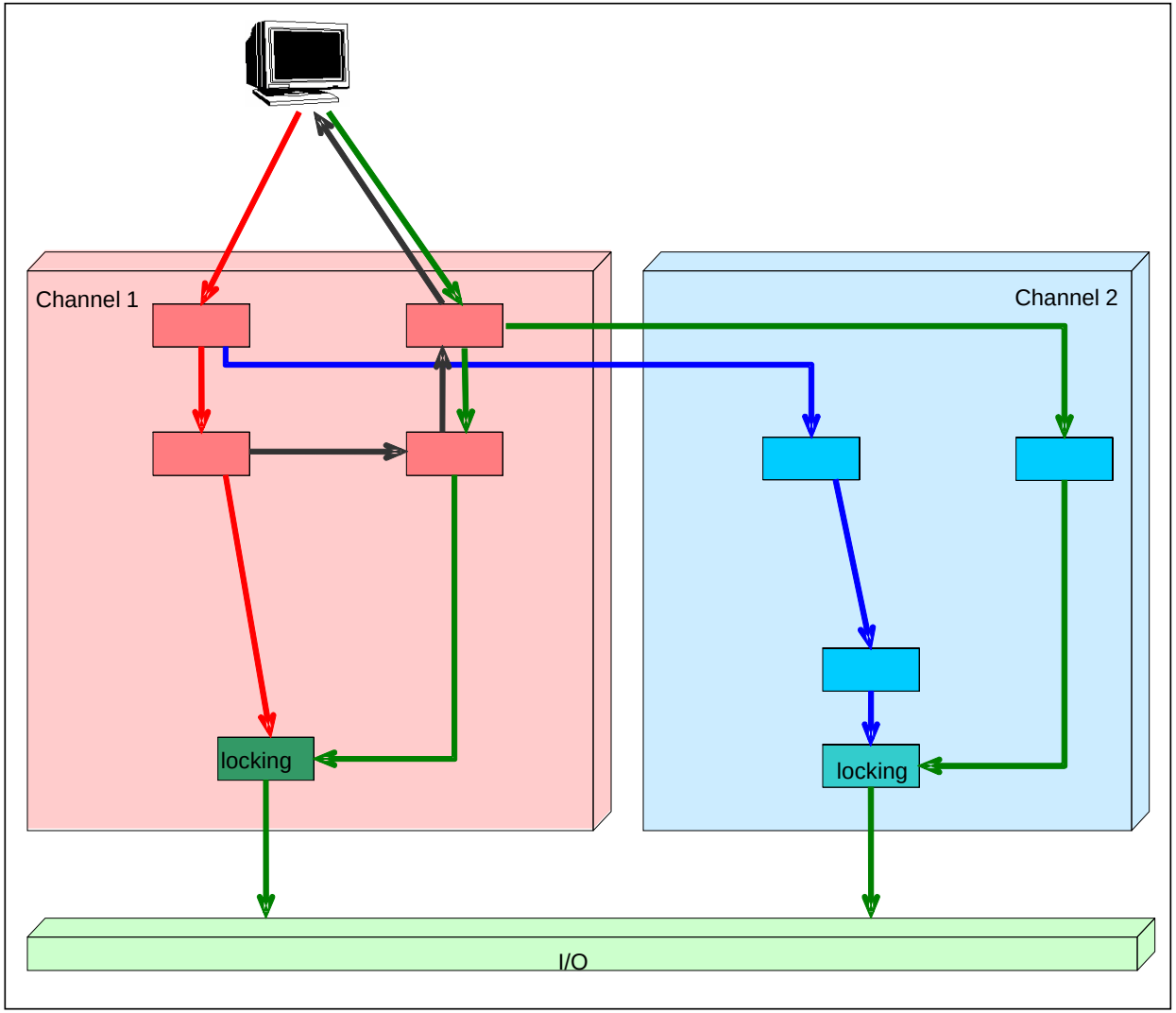
Véletlen hardver hibákra működés közben:

- Csatornák komparálása: Operátori illetve I/O
  - Operátornak: Villogó RGB-BGR szimbólum jelzi a rendszeres képernyőfrissítést és színhelyességet
- Watchdog processz
  - Többi processz futásának ellenőrzése (rendszeres életjelek vizsgálata)
- Az adatbázis tartalmának rendszeres összehasonlítása
  - Lappangó hibák detektálása

Szándékolatlan vezérlésre, közös hibákra:

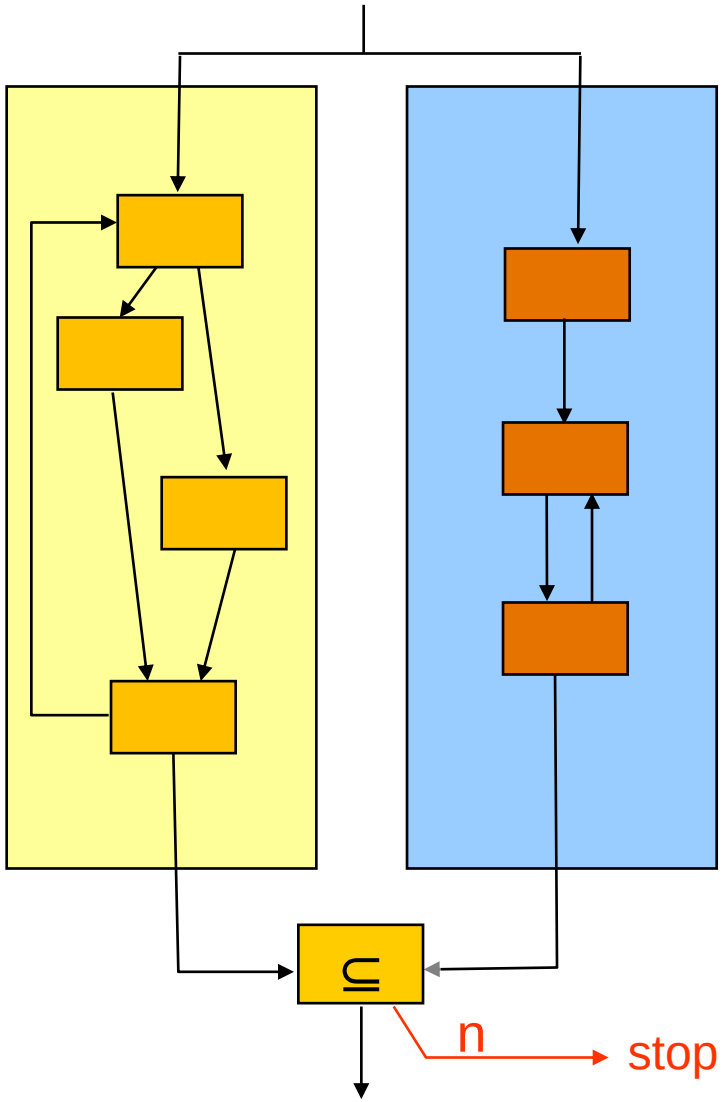
- Megerősített (háromfázisú) parancskiadás:
  - Előkészítés (de a beavatkozás zárolva diverz modulokkal)
  - Visszaolvasás független modulokon keresztül
  - Operátori jóváhagyás (eltérő GUI műveletekkel)

Példa: SCADA megerősített parancskiadás

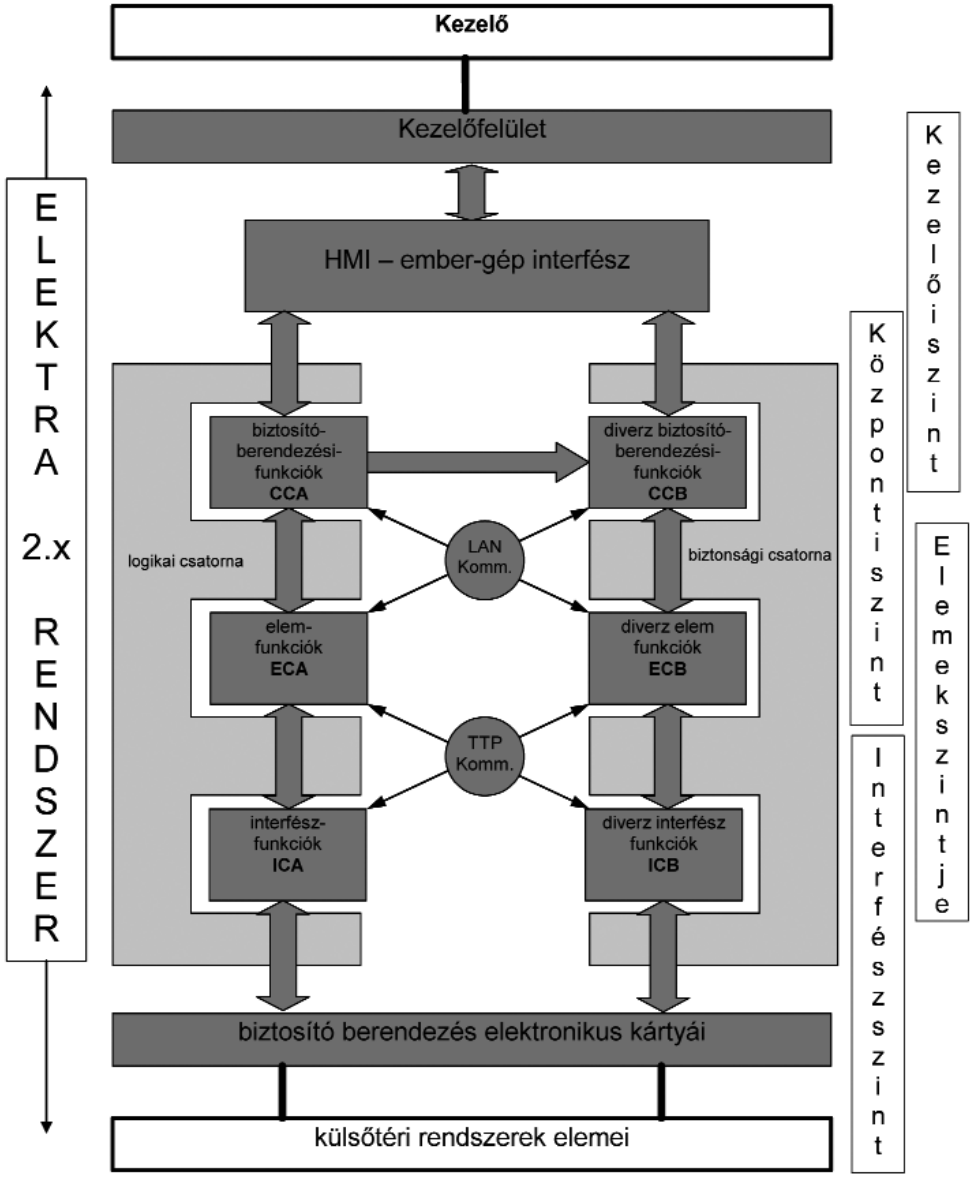


Kétcsatornás feldolgozás független ellenőrzéssel

- Független csatorna
  - „Safety bag”: csak biztonsági ellenőrzés
  - Eltérő tervezésű
  - Megengedhető, biztonságos viselkedés ellenőrzése
- Példa:
  - Elektra biztosítóberendezés
  - Szabályok az elsődleges csatorna működésének ellenőrzésére



Példa: Alcatel (Thales) Elektra



Két csatorna:

- Logikai csatorna: CHILL (CCITT High Level Language) eljárás-orientált programnyelv
- Biztonsági csatorna: PAMELA (Pattern Matching Expert System Language) szabály-orientált programnyelv

Összefoglalás: Architektúrák

Fail-safe működés

Meghibásodás esetén is biztonságos működés

Fail-stop működés

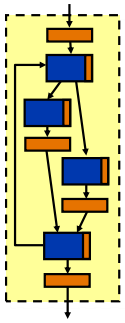
- A megállás (lekapcsolás) biztonságos állapot
- Detektált hiba esetén le kell állítani a rendszert
- Hibadetektálás a kritikus feladat

Fail-operational működés

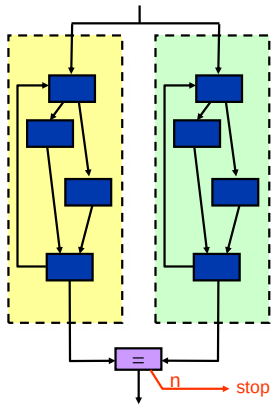
- A megállás (lekapcsolás) nem biztonságos állapot
- Detektált hiba esetén is szükséges szolgáltatás
  - teljes, vagy
  - csökkentett (degradált)
- Hibatűrés szükséges

Összefoglalás: Architektúrák fail-stop működéshez

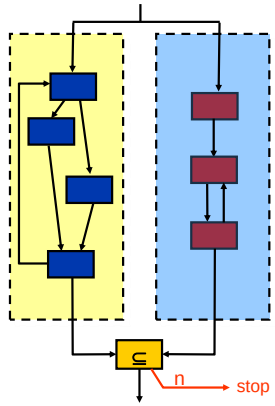
1. Egycsatornás feldolgozás beépített ellenőrzéssel
   - Hardver: Bekapcsoláskor önteszt (POST) és ütemezett öntesztek (BIST)
   - Szoftver: Online (ön)ellenőrzés



2. Kétcsatornás feldolgozás komparálással
   - Replikált feldolgozási csatornák közös bementekkel (probléma: közös hibák a csatornákban)
   - Csatornák kimenetének komparálása



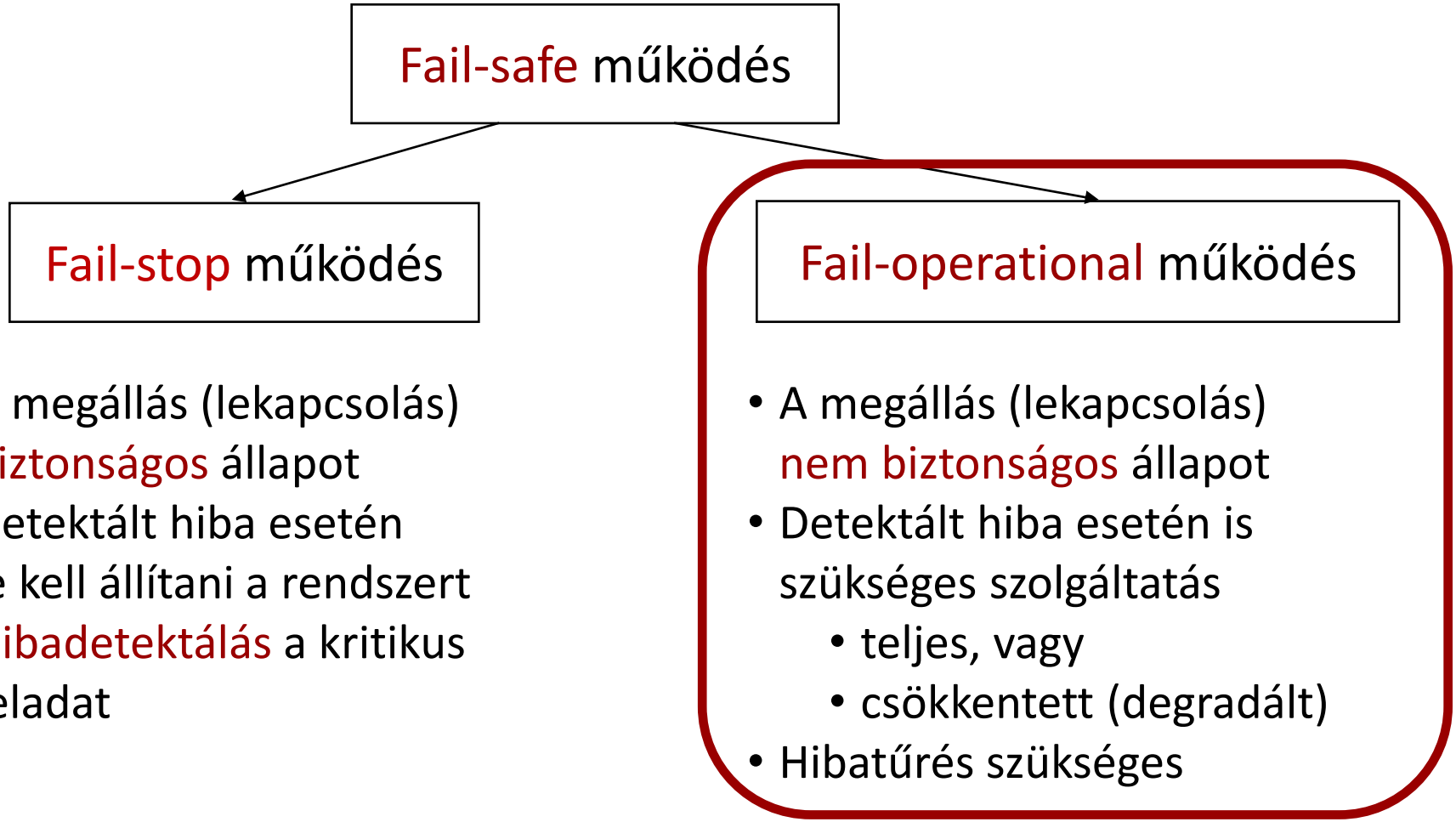
3. Kétcsatornás feldolgozás független ellenőrzéssel
   - Független, diverz ellenőrző csatorna
   - Az elsődleges csatorna kimenetének biztonsági ellenőrzése



Jellegzetes megoldások fail-operational működéshez: Hibatűrés

Célkitűzés

Hibahatások –> veszély elkerülése



- A megállás (lekapcsolás) biztonságos állapot
- Detektált hiba esetén le kell állítani a rendszert
- Hibadetektálás a kritikus feladat

- A megállás (lekapcsolás) nem biztonságos állapot
- Detektált hiba esetén is szükséges szolgáltatás
  - teljes, vagy
  - csökkentett (degradált)
- Hibatűrés szükséges

Hibatűrő rendszerek

- Hibatűrés célja: Szolgáltatást nyújtani hiba esetén is
  - Leállás helyett autonóm hibakezelés működés közben
  - Beavatkozás a hibaok → hibajelenség láncba
- Alapfeltétel: Redundancia (tartalékolás):
  Többlet erőforrások a hibás komponensek kiváltására
  - Hardver
  - Szoftver
  - Információ
  - Idő

  } Együttes megjelenés is
- Redundancia típusai
  - Hideg: Normál esetben passzív; lassú átkapcsolás hiba esetén
  - Langyos: Normál esetben csökkentett terhelés
  - Meleg (forró): Normál esetben aktív; gyors átkapcsolás

A redundancia típusai

Redundancia / tulajdonság	Hideg tartalék (passzív redundancia)	Langyos tartalék (másodlagos funkciók)	Meleg tartalék (aktív redundancia)
Alapelv	Csak hiba esetén aktiválva	Csökkentett terheléssel működik	Ugyanúgy működik, mint az elsődleges
Előnye	Nem hibásodik meg a passzív komponens	Kisebb meghibásodási tényezőjű tartalék	Gyorsan átveheti az elsődleges helyét
Hátránya	Lassan veszi át az elsődleges helyét	Közepes sebességű feladat átvétel	Azonos meghibásodási tényező
Példa	Kikapcsolt tartalék számítógép	Naplózó számítógép belép elsődlegesként	Árnyék számítógép

Példa: Információ redundancia: Hibajavító kódolás

- Hibadetektáló kódolás (EDC): Csak jelezni tudja a hibát
  - Paritásbit: Hamming távolság megnövelése, 1 bithiba detektálható
  - Ellenőrző összeg: Fájlok, hosszabb üzenetek esetén alkalmazható
- Hibajavító kódolás (ECC): Hibahely azonosítás, információ javítás
  - Nagyobb Hamming távolság: Javítható hibák
    - Pl.: (7,4) bites Hamming kód: 1 bithiba javítható, 1 és 2 bithiba detektálható
  - Blokkos információ: Bonyolultabb kódok (pl. Reed-Solomon kódok)
    - Pl.: (255, 223) bájtos RS kód: 16 bájthiba javítható
- Korlátozott hibajavító képesség
  - Információ tárolás: Hosszú idő alatt olyan sok hiba felgyűlhet, hogy a hibajavító kód nem boldogul vele
  - Tárakhoz: Periodikus olvasás és javítva visszaírás (pl. „memory scrubbing”)

4 adatbit, 3 redundáns bit

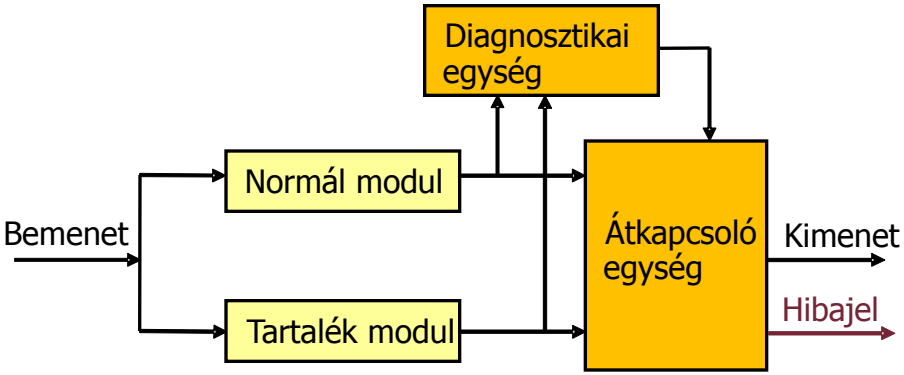
Milyen redundancia használandó?

- Hardver tervezési hibák: (< 1%)
  - Hardver redundancia, eltérő tervezésű
  - Gyakran nem számítanak rá (bevált komponensek)
- Hardver állandósult működési hibák: (~ 20%)
  - Hardver redundancia (pl. tartalék processzor)
- Hardver időleges működési hibák: (~ 70-80%)
  - Idő redundancia (pl. utasítás újravégrehajtás)
  - Információ redundancia (pl. hibajavító kódok)
  - Szoftver redundancia (pl. állapotmentés és helyreállítás)
- Szoftver tervezési hibák: (~ 10%)
  - Szoftver redundancia, eltérő tervezésű variánsok

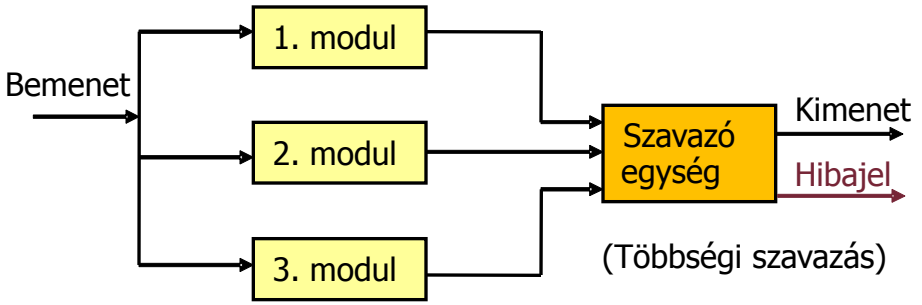
1. Állandósult hardver hibák kezelése

Többszörözés:

- Kettőzés:
  - Komparálással: csak hibadetektálás!
  - Hibatűrés: Diagnosztikai támogatás és átkapcsolás



- TMR: Triple-modular redundancy
  - Hiba maszkolása többségi szavazással
  - Szavazó kritikus elem (de egyszerű)

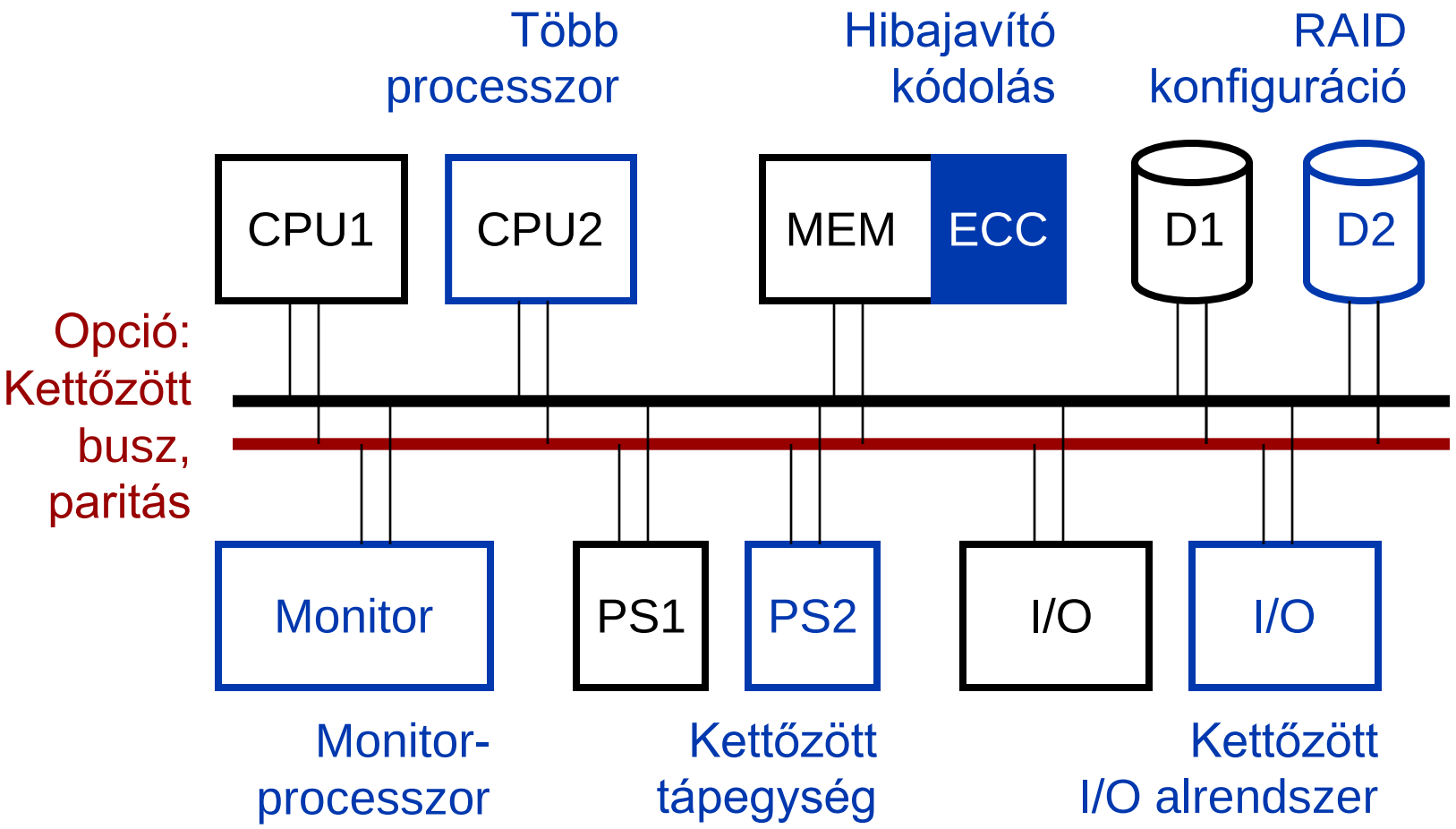


- NMR: N-modular redundancy
  - Hiba maszkolása többségi szavazással
  - Missziós idő túlélése nagyobb esélyű, utána javítás jöhet
  - Repülőgép fedélzeti eszközök: 4MR, 5MR

A többszörözés megvalósítása

- Berendezés/számítógép szint:
  - Szerverek: Nagy rendelkezésre állású szerver fürtök
    - Pl. Linux HA Clustering, Windows Server Failover Clustering
  - Szoftver támogatás: feladatátvétel (failover)
- Kártya szint:
  - Futásidőbeli átkonfigurálás “hot-swap”
    - Pl. compactPCI, HDD, tápegységek
  - Szoftver támogatás: monitorozás, konfigurációkezelés
- Alkatrész szint:
  - Alkatrész szintű többszörözés
    - TMR
    - Önellenőrző áramkörök (kódolt információval dolgoznak)

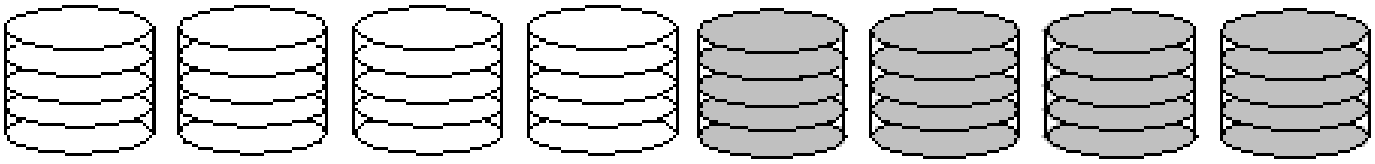
Példa: Hardver redundancia



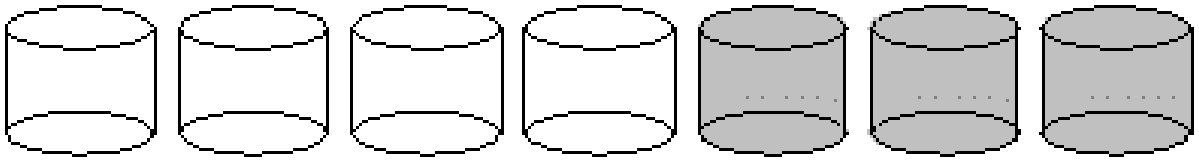
pl. IBM xSeries

Példa: RAID diszk egységek

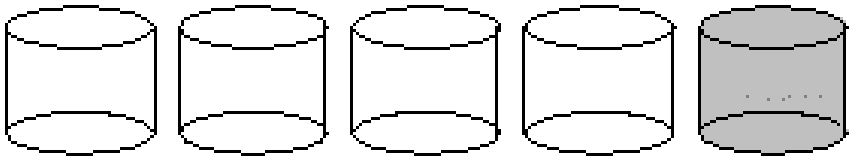
(Redundant Array of Independent Disks)



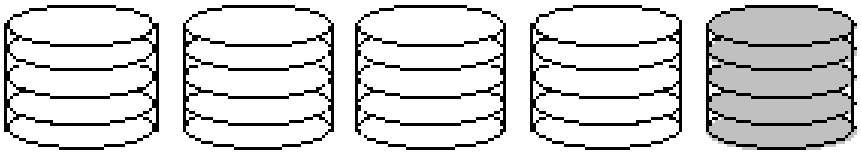
RAID-1: Tükrözés



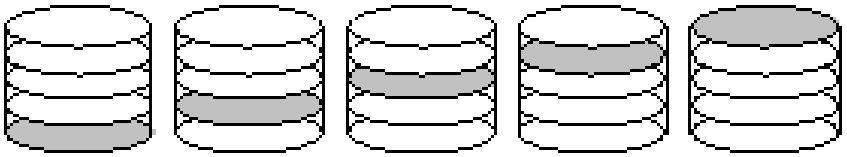
RAID-2: ECC (mint memória)



RAID-3: Bitszintű paritás

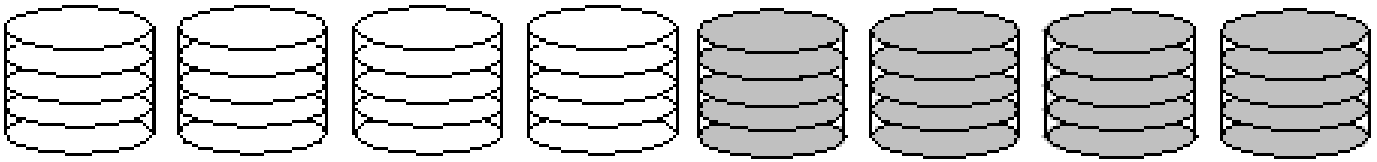


RAID-4: Blokkszintű paritás



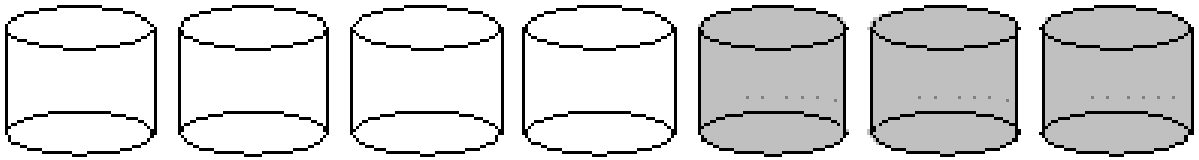
RAID-5: Elosztott blokkszintű paritás

Duplikált diszkek



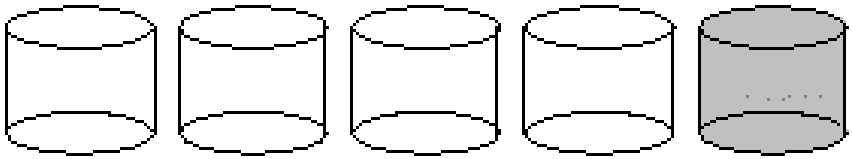
RAID-1: Tükrözés

Hibajavító kódolás



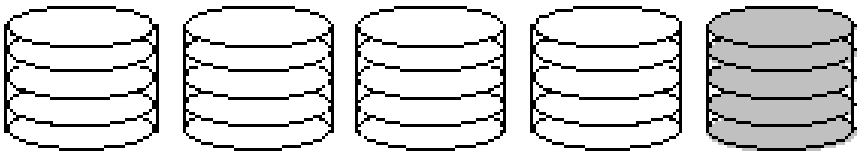
RAID-2: ECC (mint memória)

Azonosítható a hibás diszk: Paritás elég a javításhoz



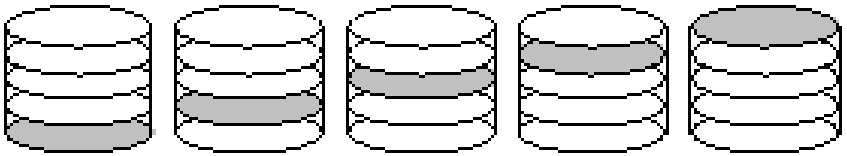
RAID-3: Bitszintű paritás

Konkurens hozzáférés a blokkokhoz



RAID-4: Blokkszintű paritás

Paritás diszk sem szűk keresztmetszet



RAID-5: Elosztott blokkszintű paritás

2. Időleges hardver hibák kezelése

Megközelítés: Szoftver alapú

- Ismételt végrehajtás esetén a hiba nem jelentkezik
- Hibahatások kiküszöbölése a fontos

A hiba kezelhető hibamentes állapot beállításával (és részben ismételt végrehajtással)

Feladatok:

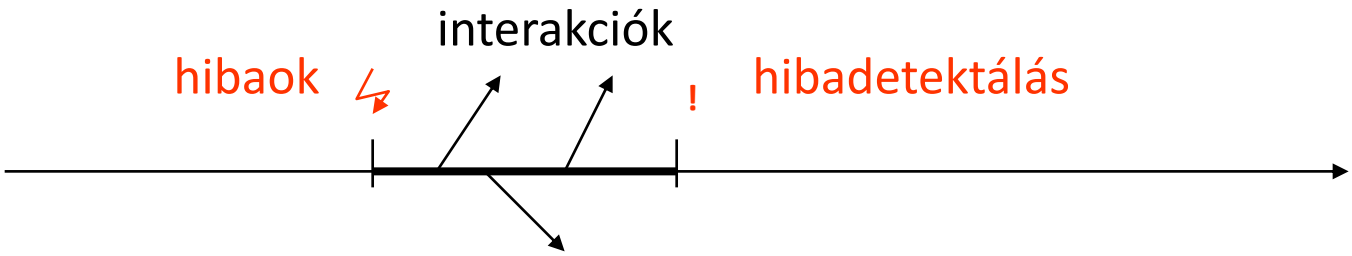
1.) Hibadetektálás
2.) Hibahatás felmérése
3.) Helyreállítás
4.) Meghibásodás (hibaok) kezelése

1. fázis: Hibadetektálás

- Alkalmazásfüggetlen (mechanizmus):
  - Illegális utasítások felismerése
  - Védelmi szintek, jogosultságok ellenőrzése
  - …
- Alkalmazásfüggő (ad-hoc):
  - Időzítés ellenőrzése
  - Visszahelyettesítés (algoritmus)
  - Hihetőség vizsgálat
  - Struktúra ellenőrzés
  - Diagnosztikai ellenőrzés
  - …

2. fázis: Hibahatások felmérése

- Motiváció: Hibadetektálás késleltetési ideje alatt a hiba terjedhet a komponensek között



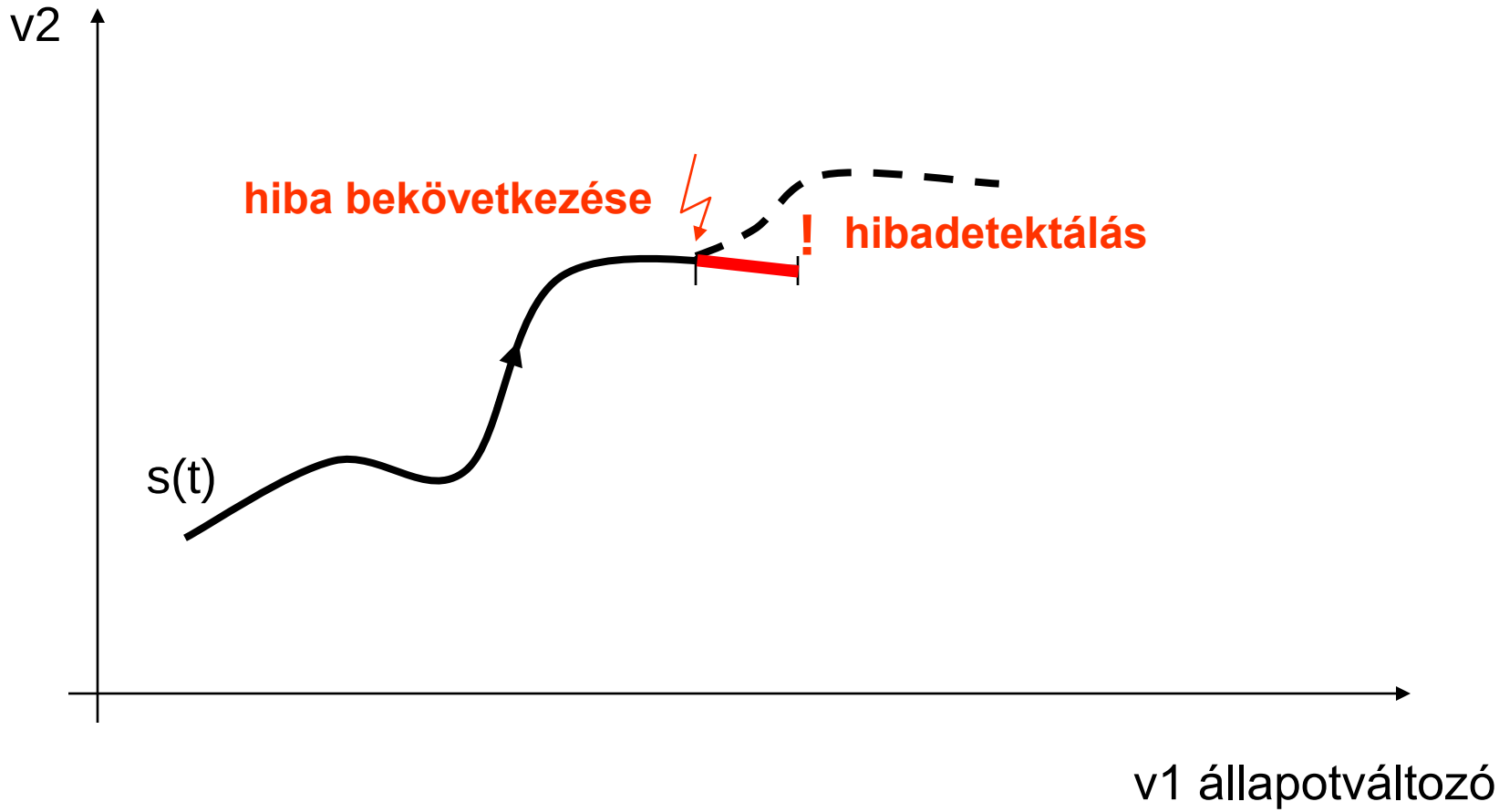
- Hibaterjedés behatárolása: Interakciók ellenőrzése
  - Bemeneti ellenőrzések elvégzése (pl. hihetőség vizsgálat)
  - Kimeneti ellenőrzések elvégzése (legyen „fail-silent”)
- Hiba által érintett komponensek meghatározása:
  - Naplózás: Erőforrás használat, kommunikáció
  - A hibadetektálás késleltetési ideje alatt történt interakciók követése

3. fázis: Helyreállítás

- Előrelépő helyreállítás:
  - Hibamentes állapot beállítása szelektív korrekcióval
  - A detektált hiba és a hibahatás függvénye
  - Előre figyelembe vett hibák esetén
- Visszalépő helyreállítás:
  - Állapot beállítása korábbi hibamentes állapotra
  - Hibától függetlenül megvalósítható
  - Állapotmentés és visszaállítás minden komponensre
- Kompenzáció:
  - Többlet információ alapján a hibahatás kompenzálható

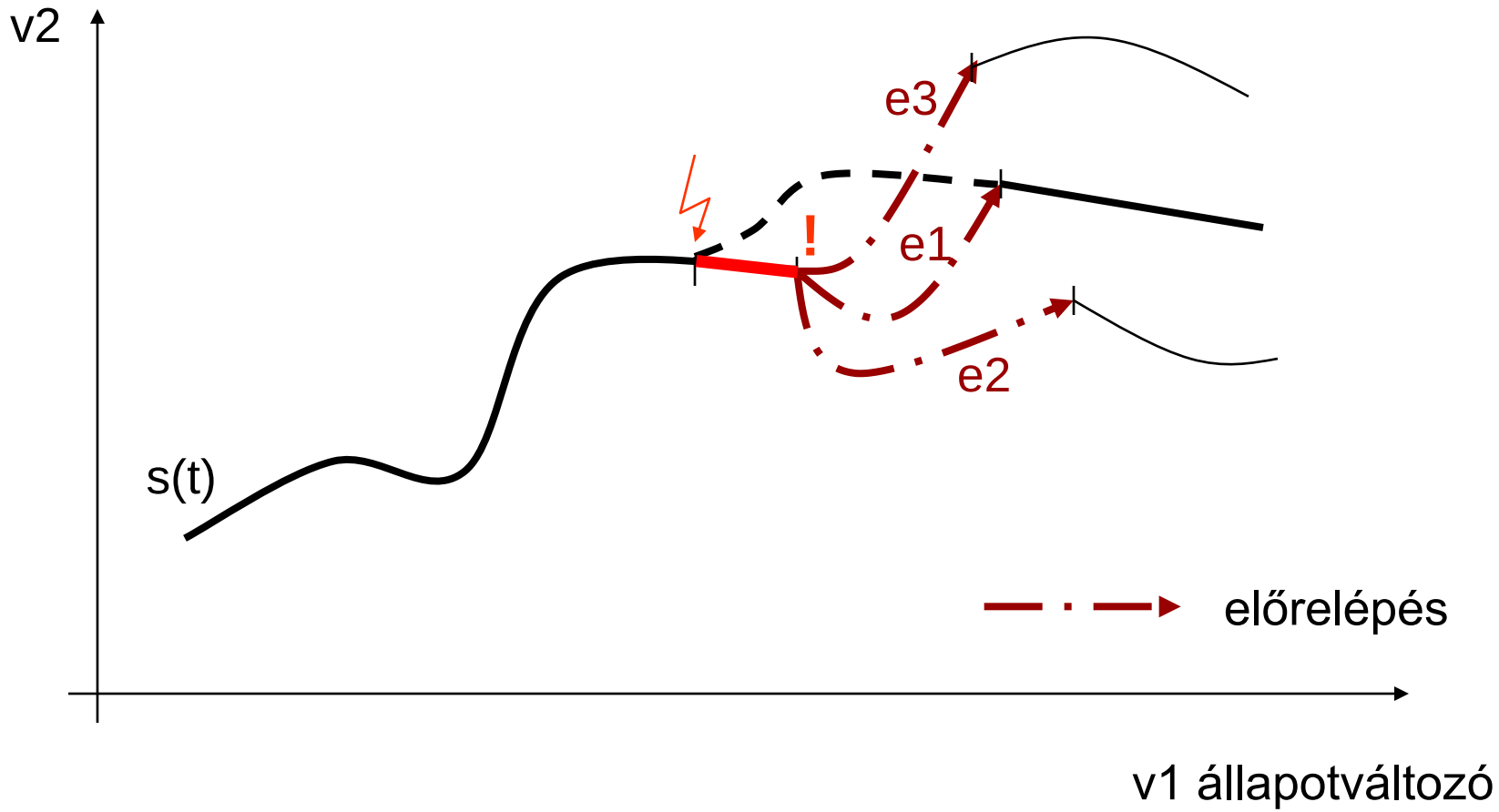
A helyreállítás lehetőségei

- A rendszer állapotterében: Hibadetektálás



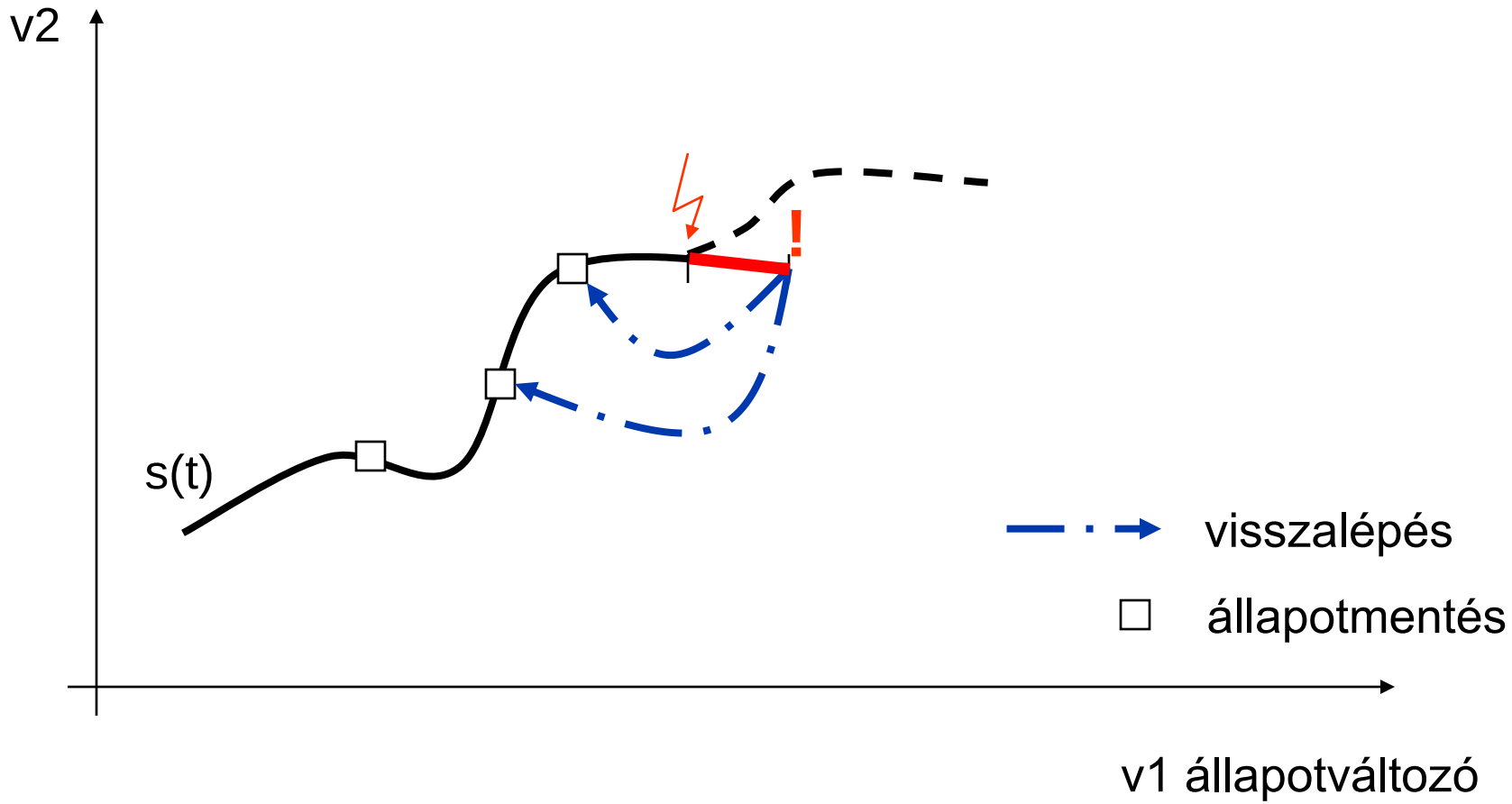
A helyreállítás lehetőségei

- A rendszer állapotterében: Előrelépő helyreállítás



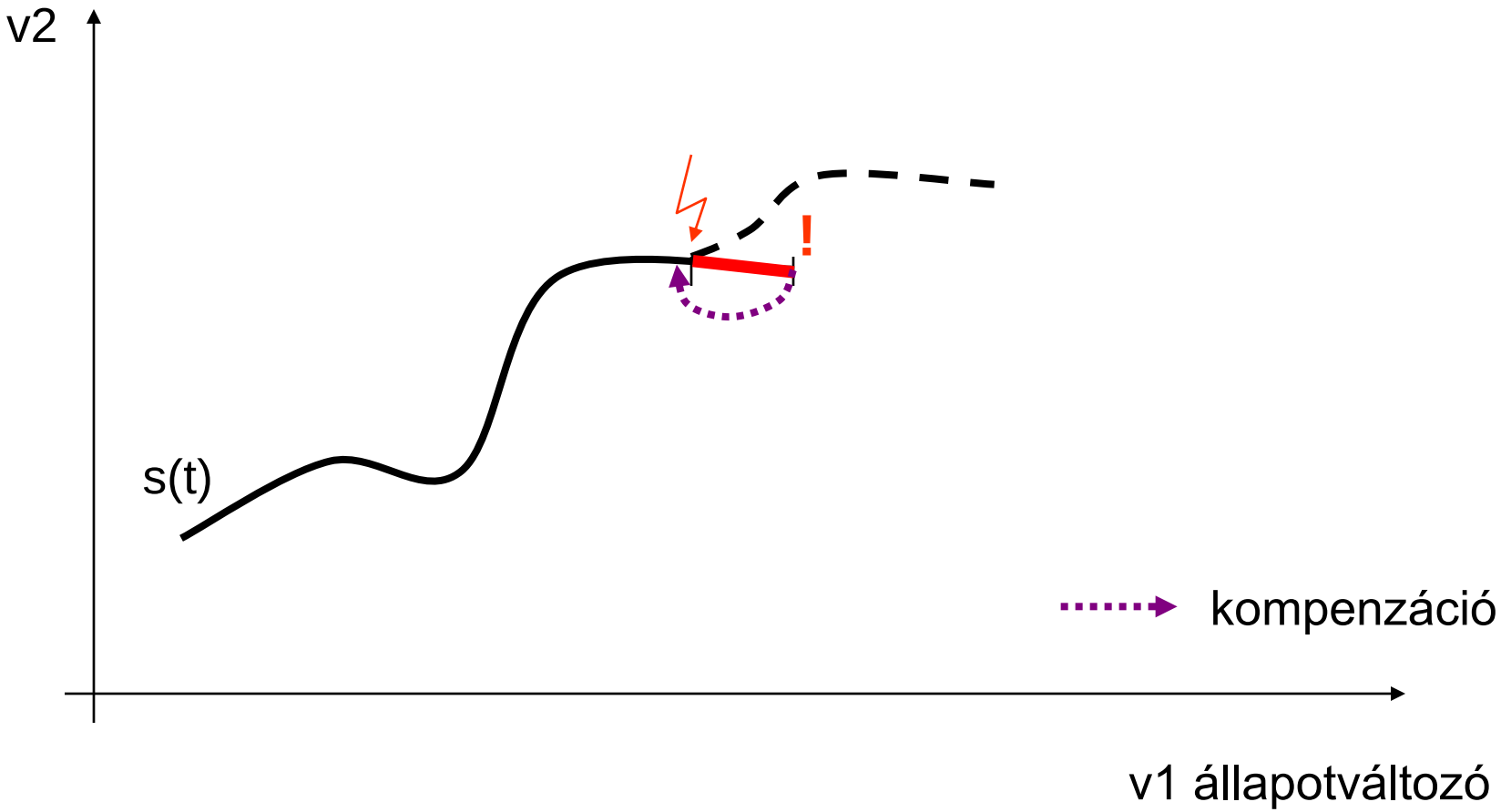
A helyreállítás lehetőségei

- A rendszer állapotterében: Visszalépő helyreállítás



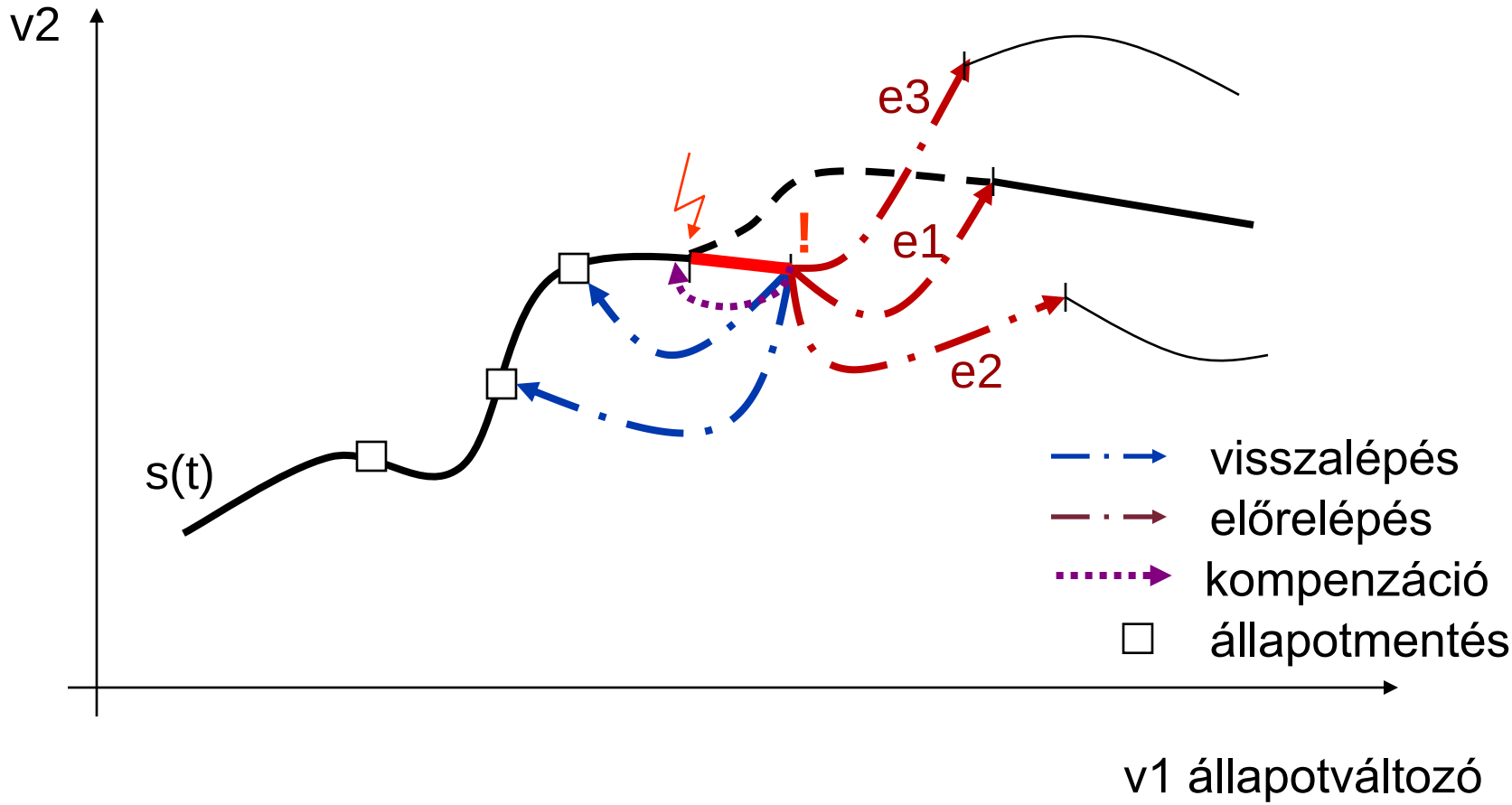
A helyreállítás lehetőségei

- A rendszer állapotterében: Kompenzáció



A helyreállítás lehetőségei

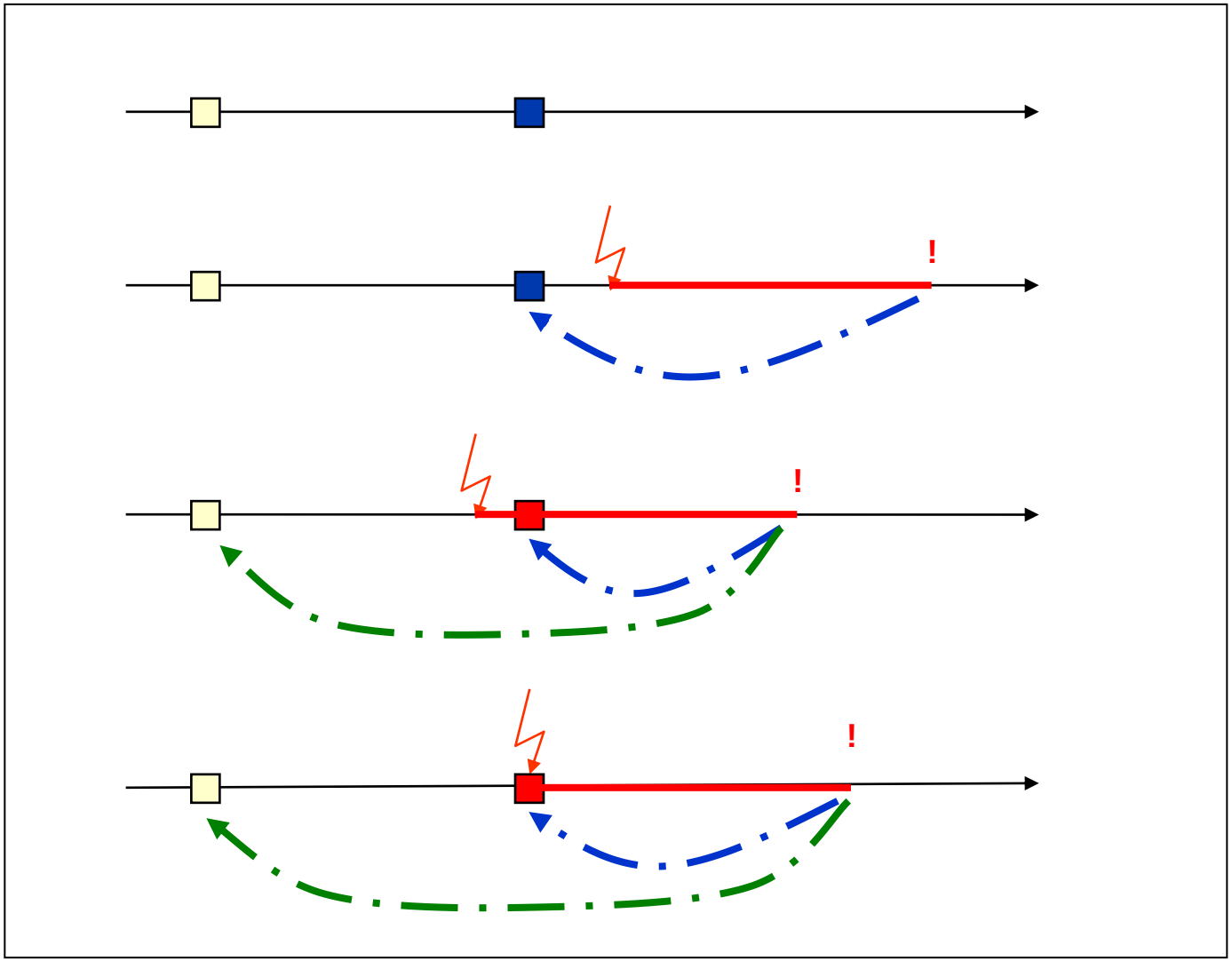
- A rendszer állapotterében: A lehetőségek



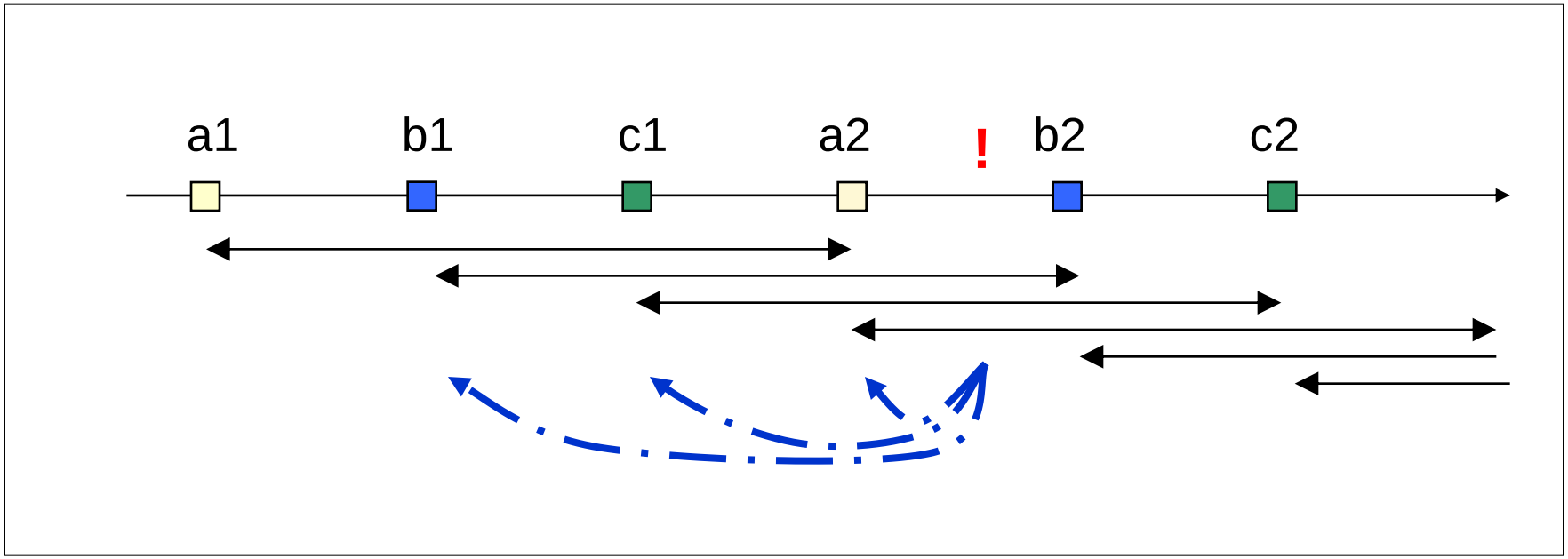
Visszalépő helyreállítás

- Állapotmentés és visszaállítás alapján
  - Checkpoint: állapotmentés (időpontja)
  - Műveletek:
    - Állapotmentés: időközönként, üzenetek után; stabil tárba
    - Visszaállítás: stabil tárból az operatív memóriába
    - Eldobás: adott számú checkpoint megtartása
  - Analógia: „autosave”
- Műveletek visszavonása alapján
  - Hiba: téves (szándékolatlan) művelet
  - Műveletek naplózása szükséges a visszavonáshoz
  - Analógia: „többszintű undo”
- Kombinált módszer is lehetséges

Visszalépő helyreállítás forgatókönyvei



Checkpoint tartományok



Optimalizálás szempontjai:

- Korlátos tár az állapotmentéshez (hány állapotmentés lehet)
- Állapotmentés gyakorisága (mennyi számítást vesztünk el)
- Hibadetektálás lappangási ideje (milyen jó a detektálás)

4. fázis: A hibaok kezelése

- Időleges hibák:
  - Előre- vagy visszalépő helyreállítás elég
- Állandósult hibák:

  Helyreállítás nem lesz sikeres (újra hibadetektálás)
  - Hiba lokalizálása
    - Diagnosztikai célú tesztelés
  - Újrakonfigurálás
    - Hibatűrés: Hibás komponens feladatainak teljes átvétele
    - Degradált működés: Egyes funkciók fenntartása
    - „Graceful degradation”: A kritikus funkciók fenntartása
  - Javítás, csere

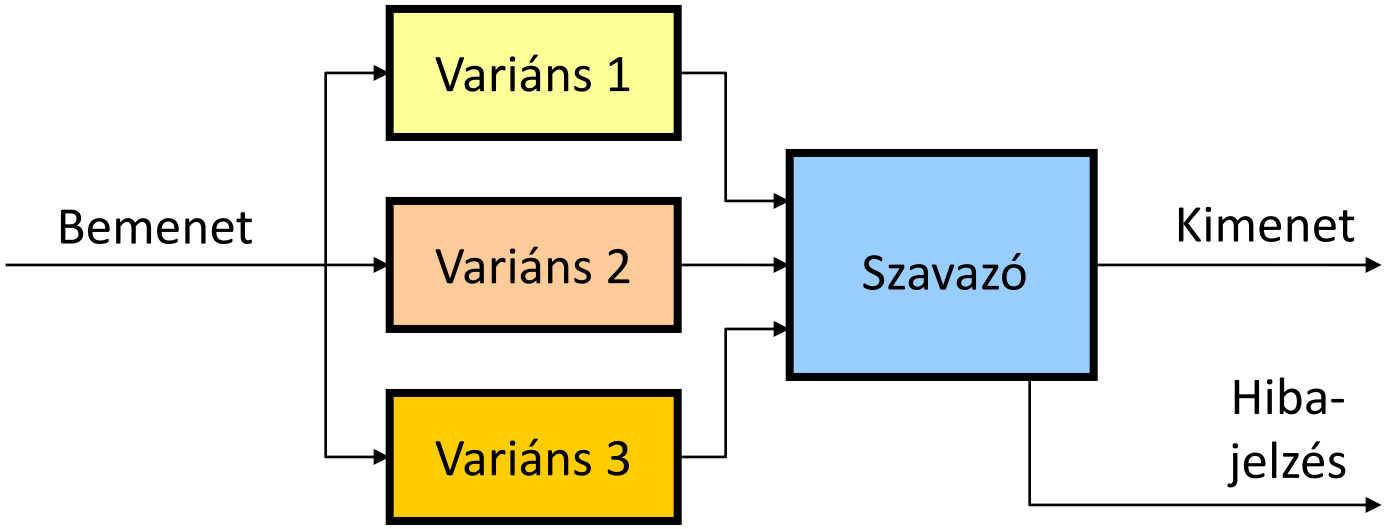
3. Szoftver tervezési hibák kezelése

- Ismételt végrehajtás nem segít (állandósult hiba)
- Redundáns modulok: Eltérő tervezés szükséges!
  Variánsok: azonos specifikáció, de
  - eltérő algoritmus, adatstruktúrák
  - más fejlesztési környezet, programnyelv
  - elszigetelt fejlesztés

  az azonos hibák bekövetkezésének csökkentésére
- Variánsok végrehajtásának technikái:
  - N-verziós programozás
  - Javító blokkok

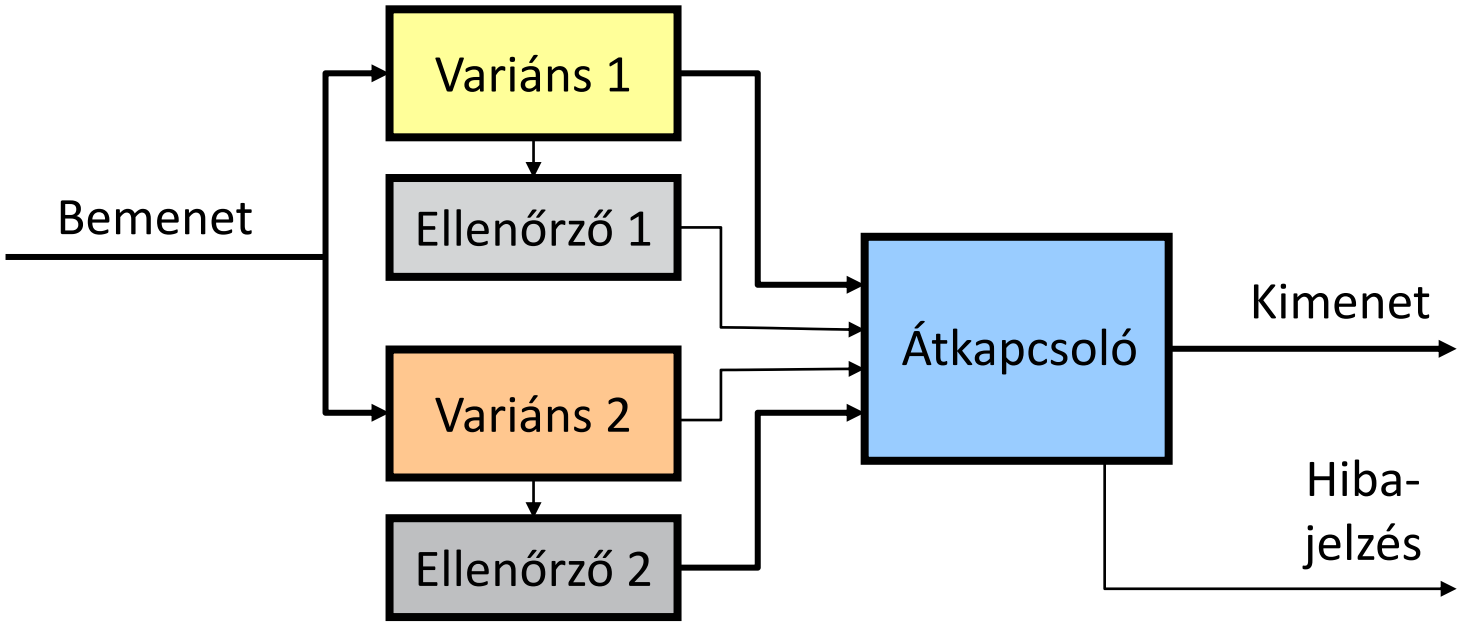
N-verziós programozás

- Aktív statikus redundancia:
  Minden variáns végrehajtása (párhuzamosan)
  - Ugyanazon bemenetek
  - Többségi szavazás
    - Elfogadható értéktartományt kell megadni a kimenetekre
    - Szavazó egyszeres hibapont (SPOF), de egyszerű



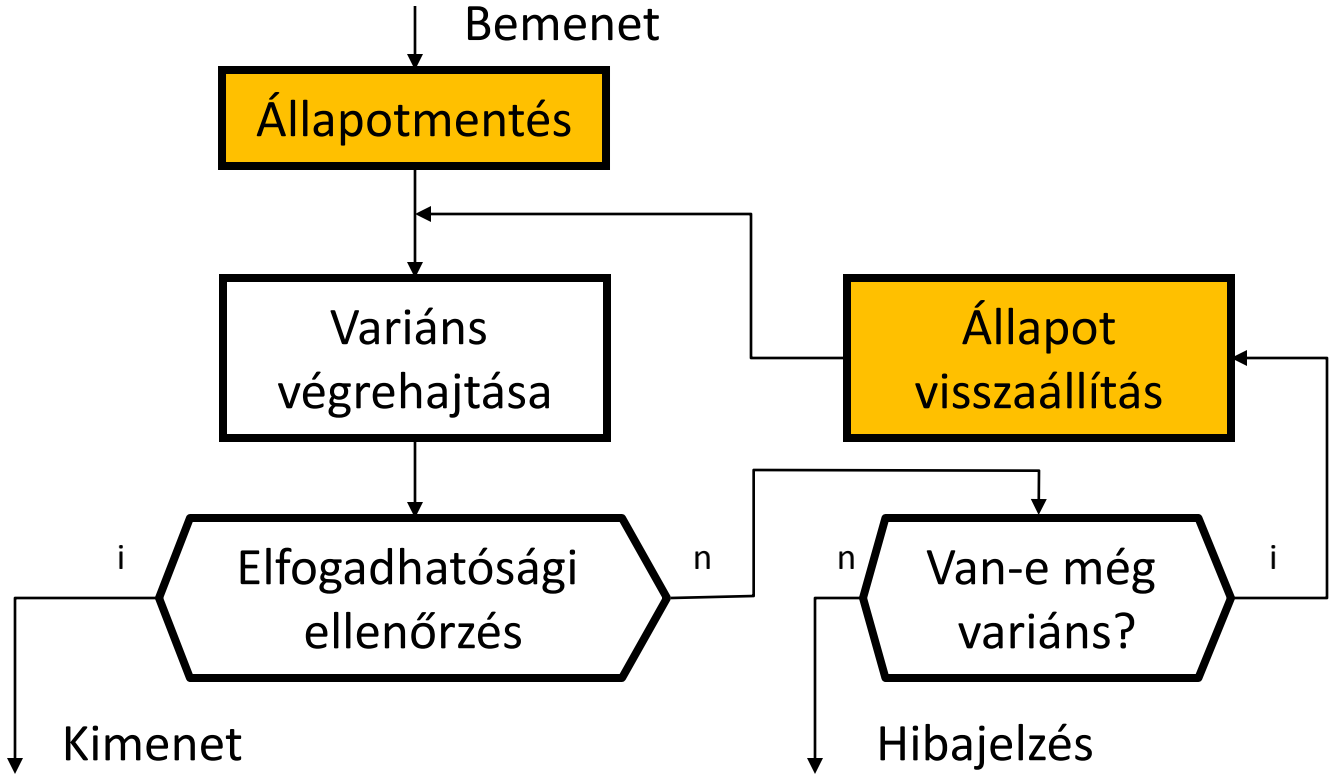
N-önellenőrző programozás

- Aktív dinamikus redundancia:
  - N számú önellenőrző komponens
  - Hibadetektálás esetén átkapcsolás az elsődlegesről az önellenőrző tartalékra



Javító blokkok

- Passzív redundancia: Csak hiba esetén aktiválódik
  - Variánsok kimenetének elfogadhatósági ellenőrzése
  - Hiba esetén tartalék variáns (soros) végrehajtása



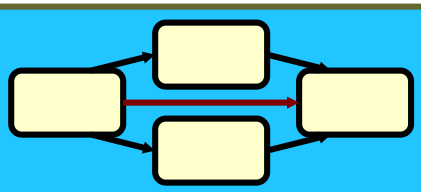
Összehasonlítás

Tulajdonság/típus	**N-verziós prg.**	**Javító blokkok**
Ellenőrzés	Szavazás, relatív	Elfogadhatóság abszolút
Végrehajtás	Párhuzamos	Soros
Időigény	Leglassabb variáns (vagy time-out)	Hibák számától függő
Redundancia aktiválása	Mindig	Csak hiba esetén
Tolerált hibák	[(N-1)/2]	N-1
Hibakezelés	Maszkolás	Helyreállítás

Hibatűrés tesztelése

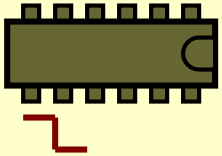
- Meghibásodás (hiba) előidézése: Hibainjektálás
  - Hardver:
    - Hibaok létrehozása:
      busz jelek kényszerítése, tápfeszültség tüske,
      részecske-besugárzás, hőterhelés
    - Hardverfüggő, lassú
  - Szoftver:
    - Hibahatás létrehozása (rendszerállapot megváltoztatása):
      regiszterek, memóriabitek átállítása (pl. Unix ptrace())
    - Rugalmas, gyors
    - Hibaokoknak való megfelelés kérdéses
  - Hibrid
- Hatás monitorozása (működés közben)

A hibainjektálás szintjei

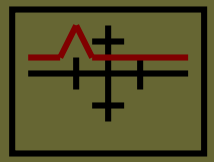


Hibaszimuláció a forráskódban vagy modell szinten a fejlesztőeszközben

Regisztertartalom módosítása, memóriakép felülírása



Síneken haladó vagy áramkörök lábán megjelenő jelek módosítása



Radioaktív sugárzás, ioninjektálás, tápfeszültség zavarása, hőmérséklet

Ár, valósághűség

90

Összefoglalás: Hibatűrés technikái

1. Hardver tervezési hibák
   - Diverz redundáns hardver komponensek
2. Állandósult hardver működési hibák
   - Hardver redundancia: többszörözés
3. Időleges hardver működési hibák
   - Szoftver redundancia
     1. Hibadetektálás
     2. Hibahatás felmérés
     3. Helyreállítás: előrelépő vagy visszalépő
     4. Hibaok kezelése
   - Információ redundancia: Hibajavító kódolás
   - Idő redundancia: Ismétlés
4. Szoftver tervezési hibák
   - Diverz redundáns szoftver komponensek (NVP, RB)

Szabvány szerinti módszerek

- IEC 61508:
  Functional safety in electrical / electronic / programmable electronic safety-related systems
- Szoftver architektúra tervezés

Table A.2 – Software design and development: software architecture design (see 7.4.3)

	Technique/Measure*	Ref	SIL1	SIL2	SIL3	SIL4
1	Fault detection and diagnosis	C.3.1	---	R	HR	HR
2	Error detecting and correcting codes	C.3.2	R	R	R	HR
3a	Failure assertion programming	C.3.3	R	R	R	HR
3b	Safety bag techniques	C.3.4	---	R	R	R
3c	Diverse programming	C.3.5	R	R	R	HR
3d	Recovery block	C.3.6	R	R	R	R
3e	Backward recovery	C.3.7	R	R	R	R
3f	Forward recovery	C.3.8	R	R	R	R
3g	Re-try fault recovery mechanisms	C.3.9	R	R	R	HR
3h	Memorising executed cases	C.3.10	---	R	R	HR
4	Graceful degradation	C.3.11	R	R	HR	HR
5	Artificial intelligence - fault correction	C.3.12	---	NR	NR	NR
6	Dynamic reconfiguration	C.3.13	---	NR	NR	NR
7a	Structured methods including for example, JSD, MASCOT, SADT and Yourdon.	C.2.1	HR	HR	HR	HR
7b	Semi-formal methods	Table B.7	R	R	HR	HR
7c	Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	---	R	R	HR
8	Computer-aided specification tools	B.2.4	R	R	HR	HR

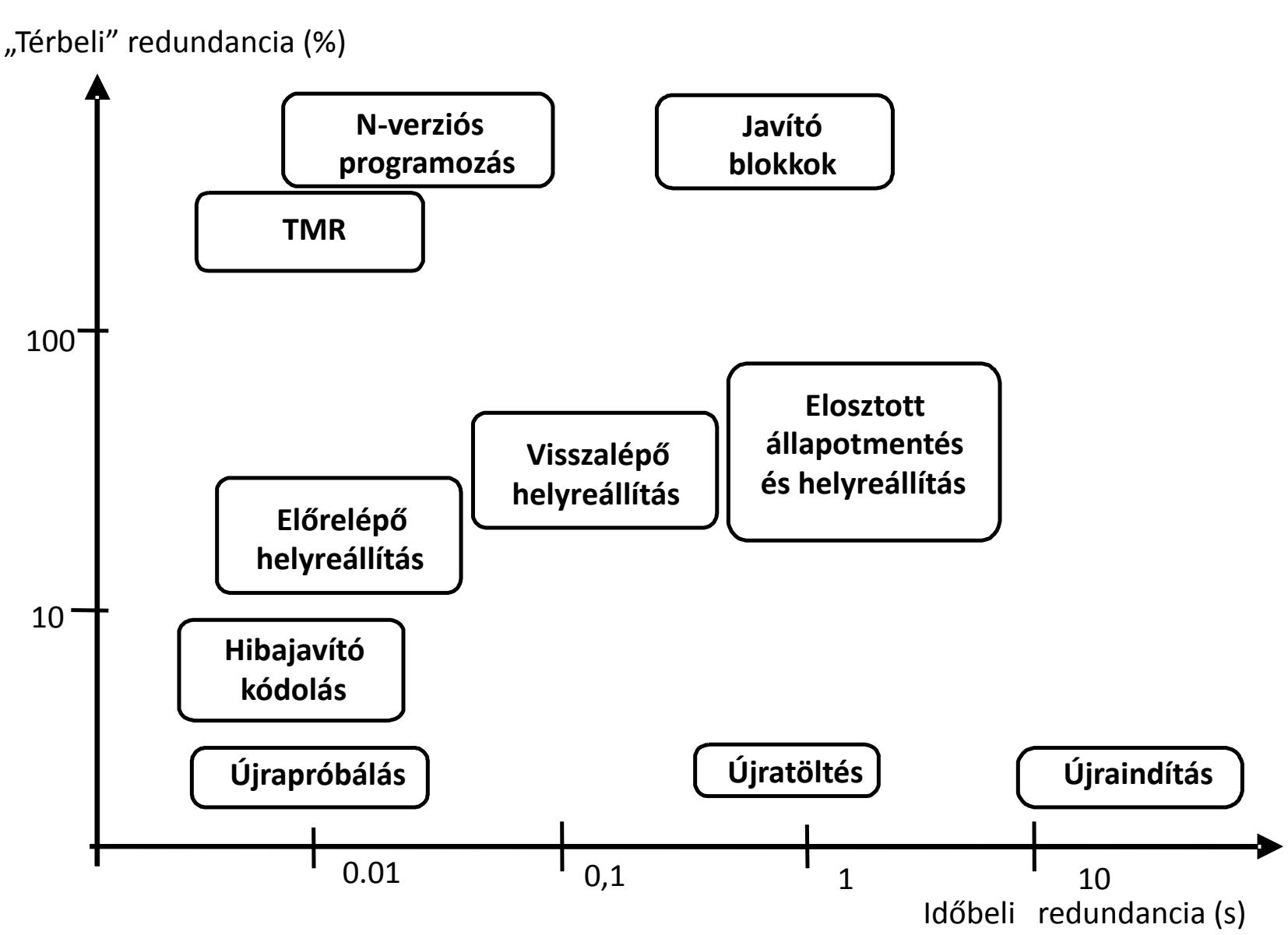
NOTE – The measures in this table concerning fault tolerance (control of failures) should be considered with the requirements for architecture and control of failures for the hardware of the programmable electronics in IEC 61508-2.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

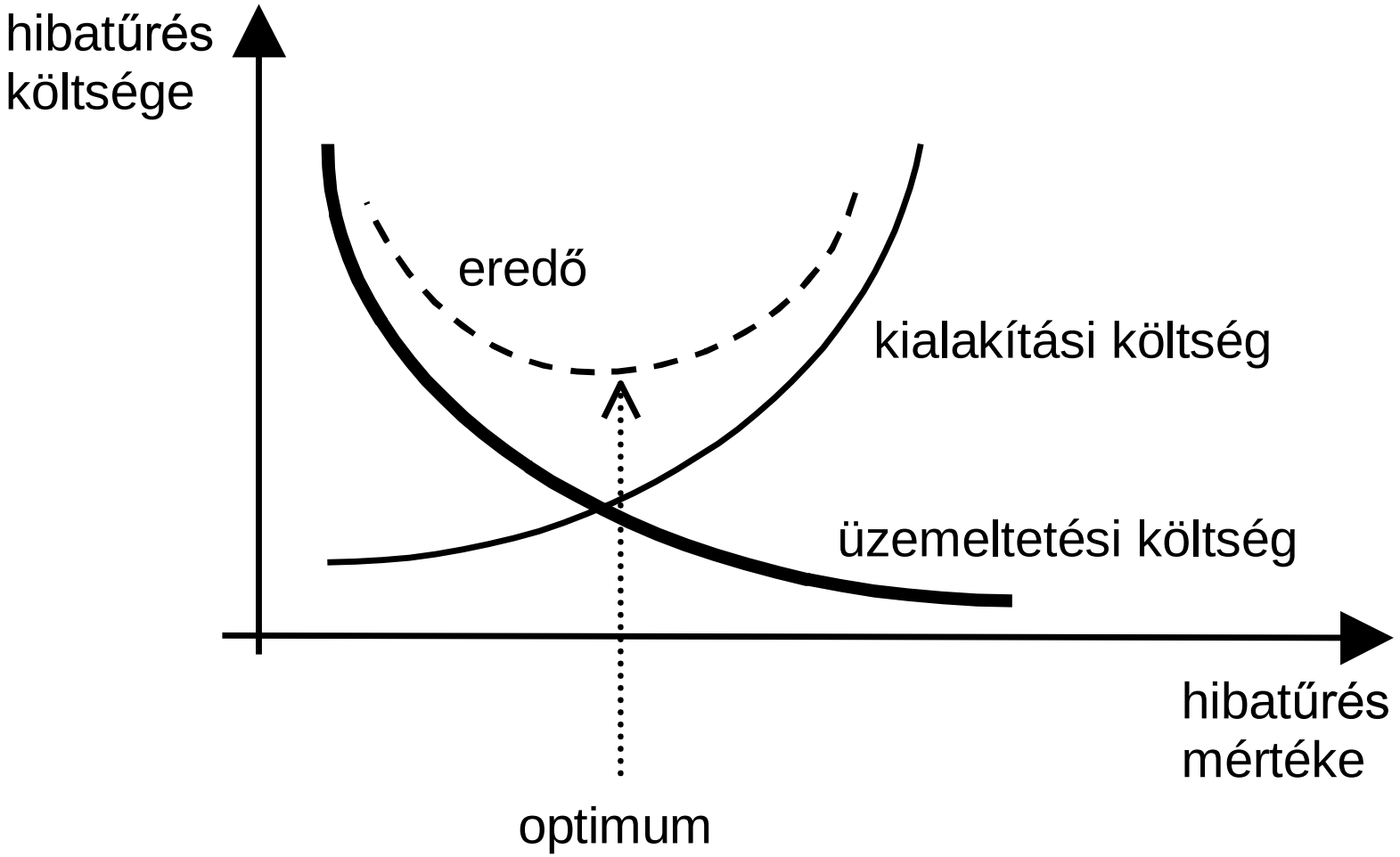
Az időigény összefoglalása

- Tiszta idő redundancia: Újrapróbálás (retry)
  - Alacsony hardver szinten: processzor (mikro)utasítás
  - Magasabb szinten: funkció, taszk ismételt végrehajtás
  - Időleges hibák esetén hatásos
- Idő overhead “velejárója” a többi típusnak
  - Hard RT rendszerek: tervezési szempont, hogy mennyire garantálható a hibakezelés ideje
  - Preferált megoldások:
    - Állandósult hardver hibák: maszkolás, meleg tartalék
    - Időleges hardver hibák: előrelépő helyreállítás
    - Szoftver tervezési hibák: N-verziós programozás

Az időigény áttekintése



Költségoptimalizálás



Összefoglalás: Architektúra tervezés

- Fail-stop megoldások
  - Egycsatornás feldolgozás öntesztte
  - Kétcsatornás feldolgozás komparálással
  - Kétcsatornás feldolgozás független ellenőrzéssel
- Fail-operational (hibatűrő) megoldások
  - Hardver tervezési hibák: Diverz redundáns hardver komponensek
  - Állandósult hardver működési hibák: Többszörözött hardver komponensek
  - Időleges hardver működési hibák:
    - Szoftver redundancia: Hibadetektálás és helyreállítás
    - Információ redundancia: Hibajavító kódolás
    - Idő redundancia: Végrehajtás ismétlése
  - Szoftver tervezési hibák: Diverz redundáns szoftver komponensek (NVP, RB)