ÓBUDAI EGYETEM

SZÁMÍTÓGÉP ARCHITEKTÚRÁK ALAPJAI

17-18. előadás

Durczy Levente

2025/2026/1

Netburst architektúra (Pentium 4 CPU család):

Az 1990-es évek második felében a második generációs szuperskalároknál elérték az architektúra teljesítménybeli határait. Általános célú alkalmazásoknál nem maradt több kimeríthető párhuzamosság utasítás szinten!

Szakkifejezéssel élve: a hatékonyság növelésének extenzív forrásai kimerültek általános célú alkalmazásokban, azonban a piac igényelte a teljesítmény további növekedését (internet robbanásszerű elterjedése)

Hogyan gyorsítsunk?

- Új architektúrák. Például VLIW
- Párhuzamosság egyéb formái. Például szál vagy folyamat szintű párhuzamosság
- Frekvencia erőteljes növelése (Intel)

Az akkori legfejlettebb Intel architektúra, a P6 (Pentium III alapja) nem volt alkalmas 1333 MHz feletti működésre. Fejlesztési cél ⟶ 10 GHz! ⟶ Új architektúra kell: ⟶ Netburst

A frekvencia növelésének a forrásai:

-Gyártási csíkszélesség csökkentése több lépésben (180nm ⟶ 65 nm)
(0,7 –szeresére csökkentik általában egy lépésben! ⟶ Kisebb helyen több tranzisztor (Moore törvény), gyorsabb elektron áramlás ⟶ növelhető a frekvencia!)
-Futószalag fokozatok hosszának csökkentése (egy fokozat hossza: egymás után lévő logikai kapuk száma)
ezt már csak úgy lehet elérni, hogy a meglévő fokozatokat kisebb egységekre bontják
⟶ nő a fokozatok száma! (ez nem cél, ez egy kényszerű következmény)
A maximális frekvenciát a leglassabb futószalag fokozat hossza határozza meg! (a jelnek át kell érnie!)

Több fokozat ⟶ párhuzamosan végrehajtott utasítások száma nő ☺ ⟶ függőségek száma is nő ☹ ⟶ csökkenhet a hatékonyság és egy érték fölött a teljesítmény is!

P4 CPU (2000-2008) jellemzői:

- CISC architektúra (1-17 byte hosszú utasítások)
- Belső RISC mag
- Hosszabb futószalagok (több függőség, de magasabb frekvencia)

Bár hatékonyságban alulmaradt például az AMD-vel szemben, a lényegesen magasabb órajel miatt a teljesítménye elérte, vagy meghaladta konkurenciáét!
Sikerét részben annak köszönheti, hogy jobb védelme volt a túlhevülés ellen, mint a konkurenseknek.
Bevezették a „Thermal monitor”-t:
Órajelmoduláción alapuló megoldás: túlmelegedés esetén eleinte lekapcsolta az órajelet, majd órajel frekvencia csökkentés és magfeszültség csökkentés.

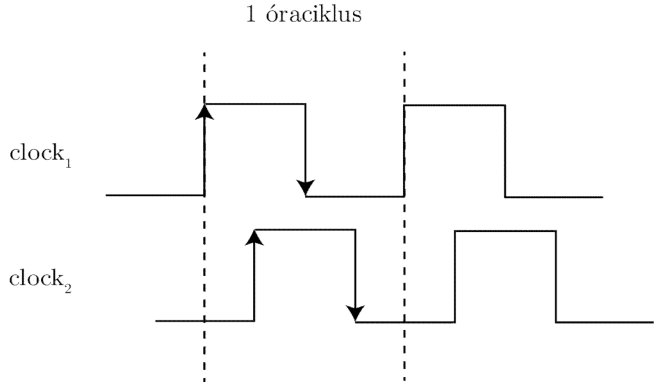
Eredmény: hűtési probléma esetén a CPU lelassul, vagy lefagy, de nem sérül, ellentétben az AMD processzorokkal:

ÓBUDAI EGYETEM

Legfontosabb Újítások:

- Execution Trace Cache:
  Az Execution Trace Cache az L1 utasítás cache-nek felel meg, de CISC helyett már dekódolt RISC utasításokat tartalmaz a végrehajtásuk feltételezett sorrendjében. Ezzel csökkenthető a kiválasztásra fordított idő.
- Hyper futószalag
  Lényege, hogy sok fokozatból áll és kihagyták belőle a dekódoló fokozatot!
  A dekódolás futószalagon kívül történik, azért, hogy az L1 cache-ben (Execution Trace Cache) már dekódolt és átalakított utasítások szerepelhessenek!
  A CISC utasítások dekódolása és átalakítása lassú (változó hossz, CISC-RISC konverzió), ez akadályozza a nagy frekvenciás végrehajtást, ezért került külön a futószalagtól!
  A frekvencia növelés okán a korábbi 10-15 fokozathoz képest 20-31 fokozatot tartalmazott.
  A hosszú futószalagok legnagyobb hátránya az utasítás-végrehajtás fajlagos hatékonyságának csökkenése a rövidebb futószalagokhoz képest (függőségek miatt egy ciklus alatt kevesebb utasítás hajtódik végre), valamint hogy hibás becslés esetén nagyobb a büntetés, több fokozatot kell törölni a futószalagból ➡ fajlagos teljesítmény csökkenés!
- Enhanced Branch Prediction:
  A minél kevesebb hibás becslés kiküszöbölésére bevezettek egy továbbfejlesztett elágazásbecslő logikát. Megnövelt méretű elágazás előrejelző tároló és újabb előrejelző algoritmus bevezetése.
  Ennek hatékonysága 94-97%-os volt. Pentium III-hoz képest körülbelül 33%-kal csökkent a hibás becslések száma! (mit jelent ez?)

- Quad Data Rate Bus:
  Bevezettek egy új belső rendszerbuszt az adatelérés gyorsítására az L1 és L2 gyorsítótárak felé. Mivel a memória nem akkora mértékben gyorsult, mint a processzor, így a külső rendszerbuszt sebességét nem lehetett növelni. A buszfrekvencia négyszeresén továbbítja az adatokat. Ennek eléréséhez kettő órajel generátort alkalmaz 90° fáziseltolással, valamint a felfutó és lefutó élen is történik adattovábbítás. Így órajelciklusonként négy adattovábbítás történik!



- Rapid Execution Engine:
  A Rapid Execution Engine az egyszerű FX műveletek gyors végrehajtására szolgáló végrehajtó egység. Az órajel felfutó és lefutó élére is képes műveletvégzésre, így a végrehajtási idő akár egy fél ciklusra is csökkenhet. A gyorsaság érdekében kevesebb kaput tartalmaz és csak az alapműveletek elvégzésére képes. Ez olyan sikeres hatékonyság növekedést eredményezett, hogy később kettőt is építettek a processzorokba.

- Replay System:

  RISC-szerű utasítások ismételt végrehajtása
  Sok fokozatú futószalag sok függőséggel jár ➡ utasítások várakozása nő ➡ kihasználatlanság nő!
  Kezelés: az ütemező megbecsüli az utasítások végrehajtási idejét, így tudja, hogy az utasítás végrehajtásának kezdetétől mennyi idő múlva fogja igényelni a bemenő operandust. Ezért a függő utasítást már ennyi idővel a szükséges függőség teljesülése előtt kiküldi a végrehajtó egység felé, hogy pont mire felhasználná a bemenő operandust, az már éppen előálljon.
  Ha rossz volt a becslés és hamarabb szükség lenne a forrás operandusra, mint ahogy az előállt, az utasítás egy speciális sorba, a Replay Queue-ba kerül, ahonnan később újra kiküldésre kerül.
  Bár a hatékonyságot mindez csökkenti, elkerülhető vele a futószalag kiürülése, így biztosítja a végrehajtó egységek optimális kihasználását.

Következmény:

A fajlagos teljesítmény, azaz az IPC (Instructions Per Cycle) csökken. Ez összességében nem jelenti a teljesítmény csökkenését, mivel az órajel magasabb lett. Így ciklusonként kevesebb utasítást hajt végre, de adott időegység alatt több ciklus van.

Esettanulmány: Intel Pentium 4

Kevésbé volt hatékony, mint a korabeli AMD CPU-k, viszont a magasabb órajel miatt nagyobb teljesítményre volt képes.
a P4 frekvenciája 1.5 GHz-ről indult és a Prescott architektúrával (2004) elérték a 3.2 GHz-et.

A P3-hoz képest fejlesztették a multimédiás feldolgozást. 144 új utasítás került be az SSE (FP) és MMX (FX) utasításkészletekbe.

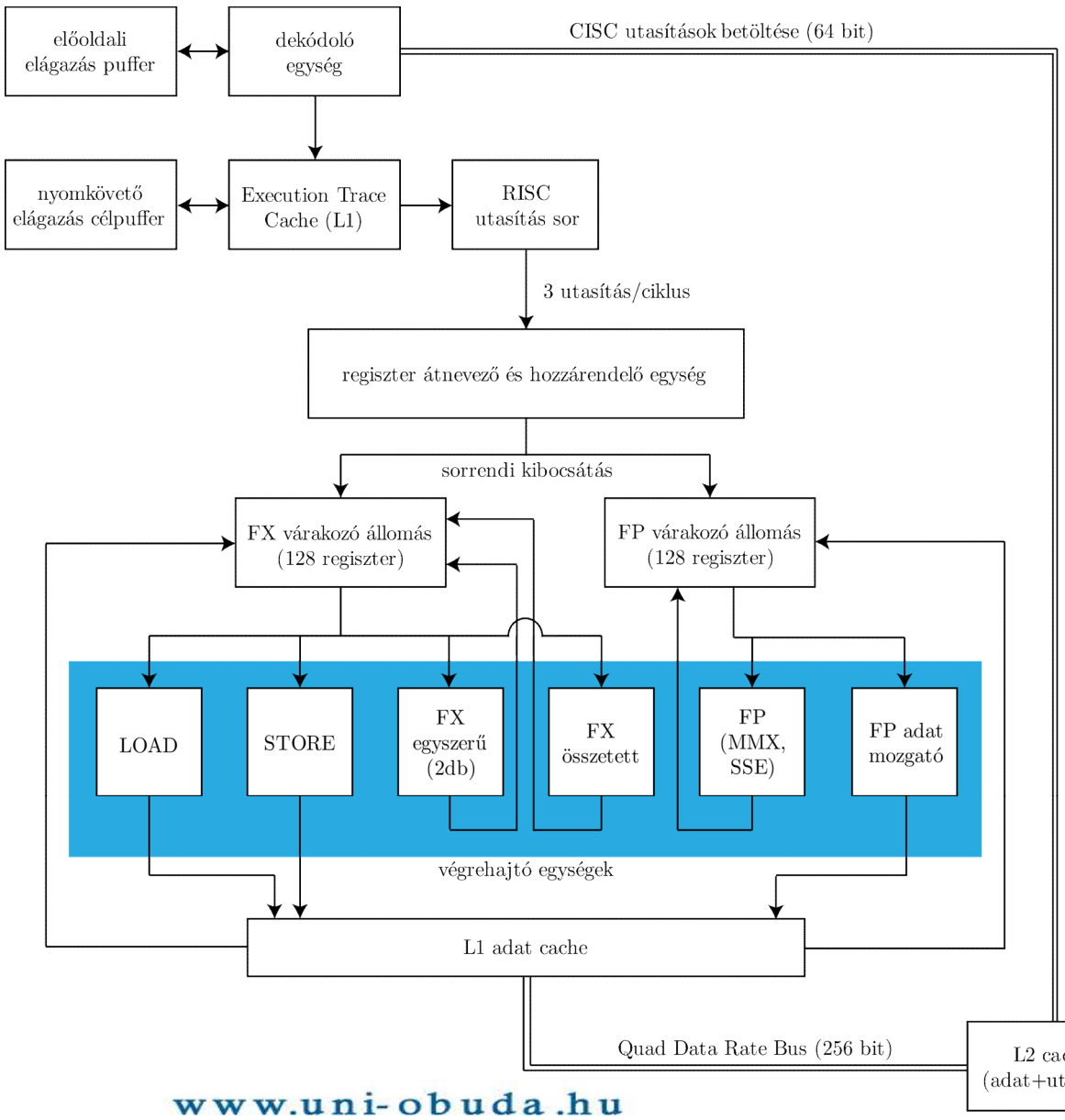
8 db architektúrális regiszter!

A memória alrendszer tartalmazza az egyesített L2 gyorsítótárat és a külső operatív tárnak a memóriabuszon való eléréshez szükséges logikát.

A futószalagon kívüli dekódoló egység a dekódolás során az előoldali elágazás puffert tudja kezelni segítve az elágazások pontosabb becslését. Van egy nyomkövető elágazás célpuffer, ami szintén az elágazások hatékony ütemezését segítette.

L1 és L2 cache között található a Quad Data Rate Bus.
Az L2 cache csatlakozik az operatív tárhoz és innen történik a CISC utasítások betöltése a dekódolóba.

Végrehajtási modell:



Fejlődési korlátok I.:

- Hatékonysági korlát (ILP kimerülés, memória sebesség olló)
  (Növekvő órafrekvenciákon egyre csökkenő teljesítménytöbblet!)
- **Disszipációs korlát**
- Párhuzamos buszok frekvencia korlátja (előző félév)

Elérték a 10 GHz-et?
3-4 GHz-nél új kihívások jelentkeztek.

Ezek egyike a statikus disszipáció, vagyis a szivárgási áramok miatt jelentősen emelkedő hőveszteség!
Szivárgási áram: a tranzisztor kikapcsolt állapotában is folyik minimális áram (a „Gate” nemtökéletes szigetelő!), ez a frekvencia növelésével exponenciálisan nő!
A szivárgási áram fajtái:

- Gate ➡ Drain felé
- Source ➡ Drain felé

3.8-4 GHz-nél „hőkatasztrófa” következett be. Ezen a tranzisztorok csíkszélességének csökkentése is csak egy darabig tudott segíteni.

Disszipáció: dinamikus + statikus

Statikus disszipáció:
A szivárgási áram hőveszteséget (disszipációt) okoz. 3.8-4 GHz környékén bekövetkezett a hőkatasztrófa. Ennek kezelése hagyományos eszközökkel nem megvalósítható, csak vízhűtéssel, ami viszont nem gazdaságos. A statikus disszipáció kiszámítása:

$D_s = V * I_{leak}$ (V a feszültség, I_{leak} pedig a szivárgási áram, ami a frekvencia növekedésével exponenciálisán nő)

Dinamikus disszipáció:
az a disszipáció, amit aktív tranzisztorokon átfolyó áram hőtermelése okoz.

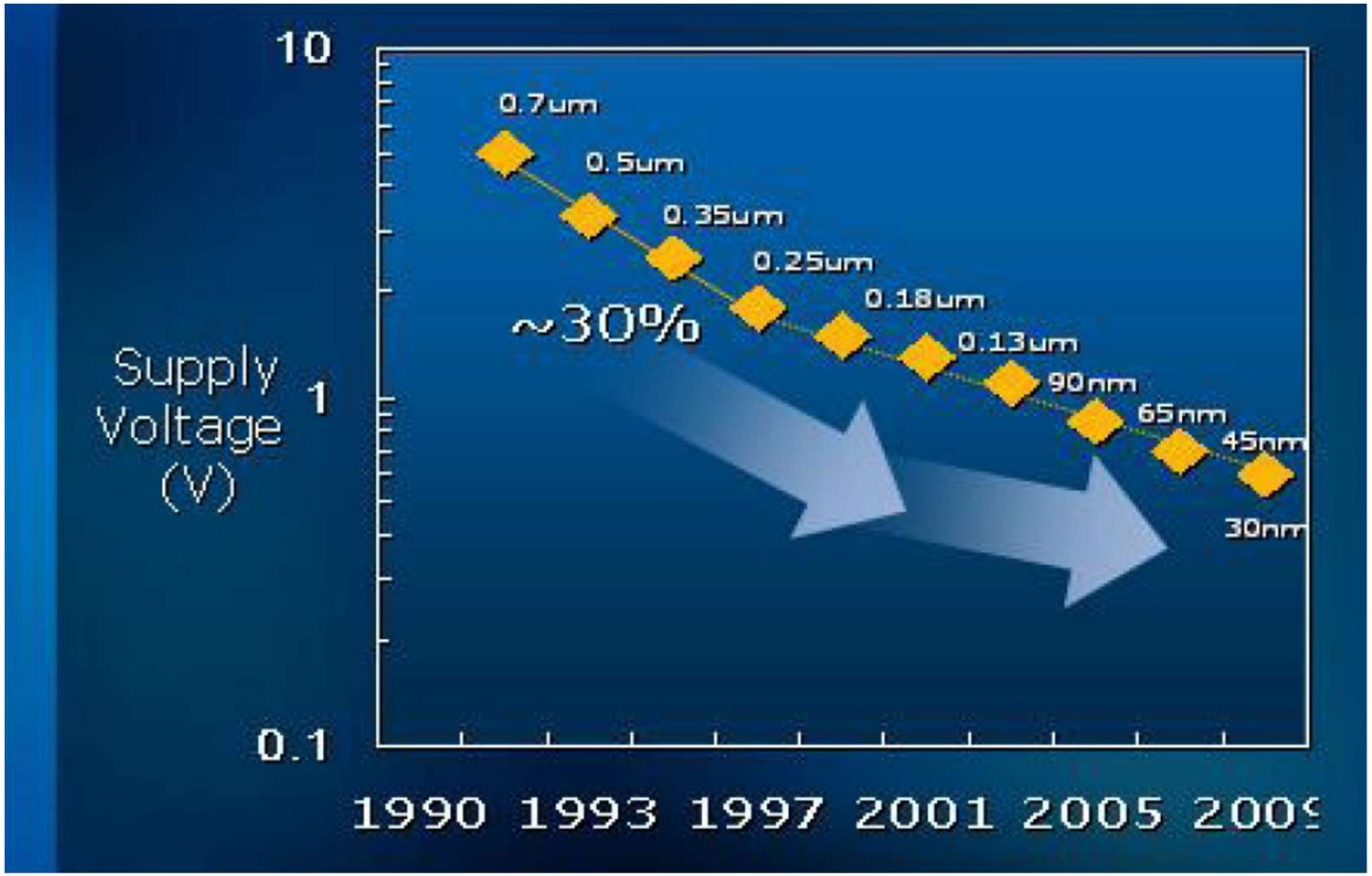
Képlete:

$D_d = A * C * V^2 * f_c$ (f_c növelése esetén lineárisan nő, ha a többi érték változatlan!)

A az aktív kapuk részaránya,
C a kapuk összesített elosztott kapacitása,
V a tápfeszültség,
fc a magfrekvencia.
Látszik, hogy a dinamikus disszipáció négyzetesen függ a feszültségtől, így a frekvencia növelése a feszültség csökkentésével ellensúlyozható (következő ábra)

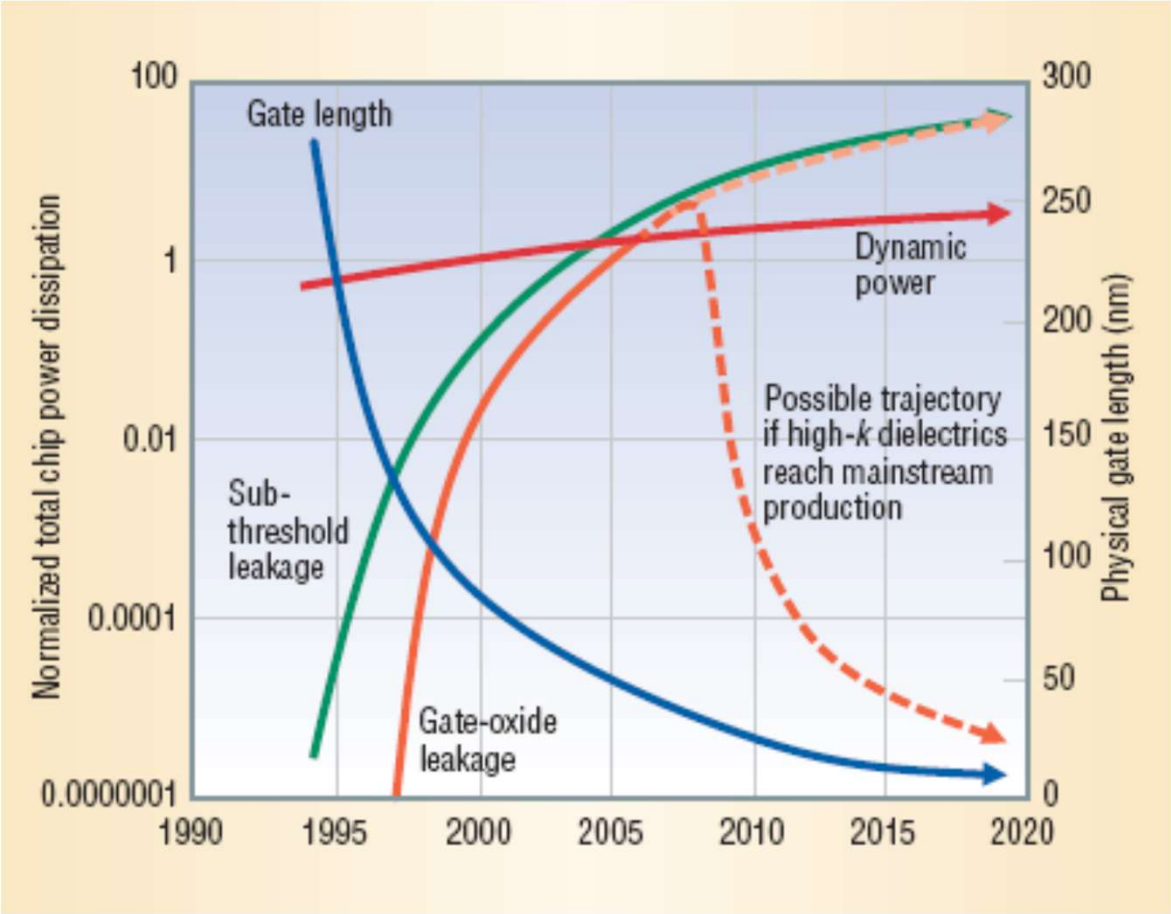
A disszipációs korlát

A tápfeszűltség skálázása a gyártási technológiával



Forrás: Gelsinger P. IDF, June 7 2006

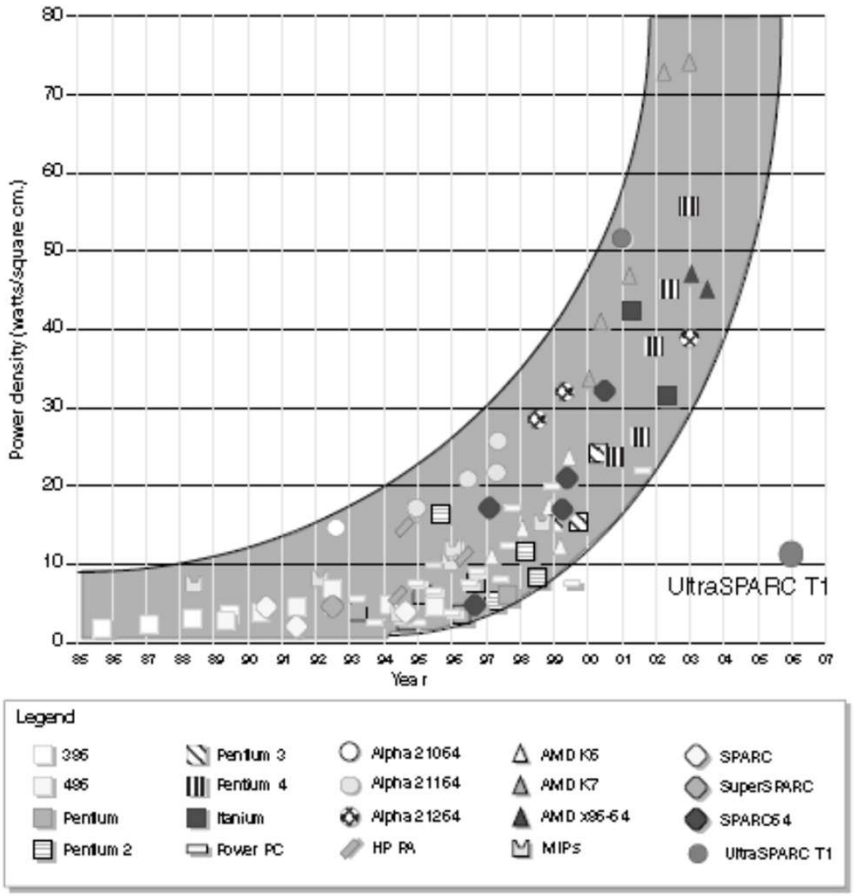
A disszipációs korlát



6.1 ábra: A dinamikus és a statikus disszipáció növekedési trendje

Forrás: N. S. Kim et al., „Leakage Current: Moore's Law Meets Static Power", Computer, Dec. 2003, pp. 68-75.

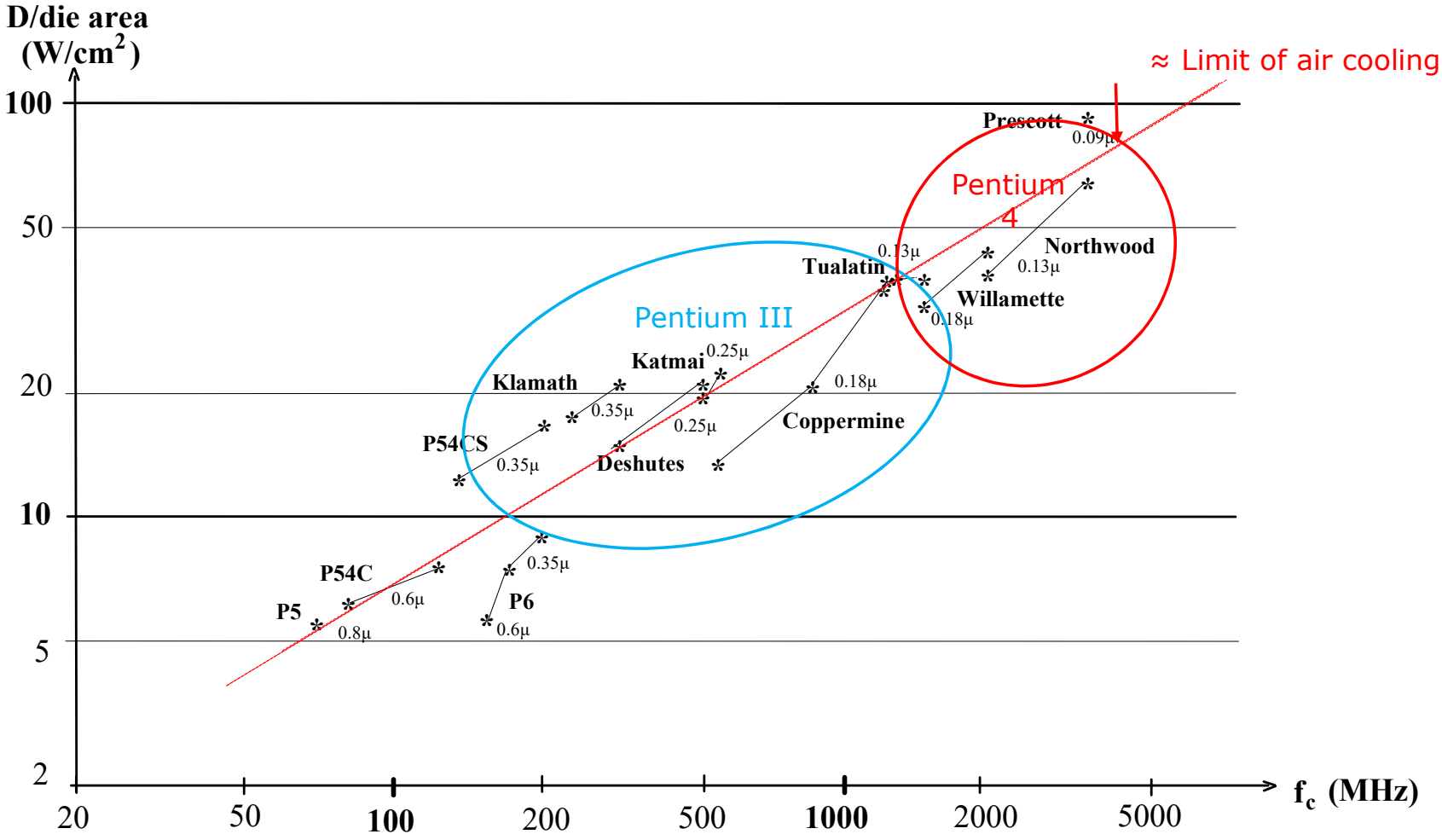
A disszipációs korlát



6.5. ábra: A fajlagos disszipáció értékének növekedése (általában)

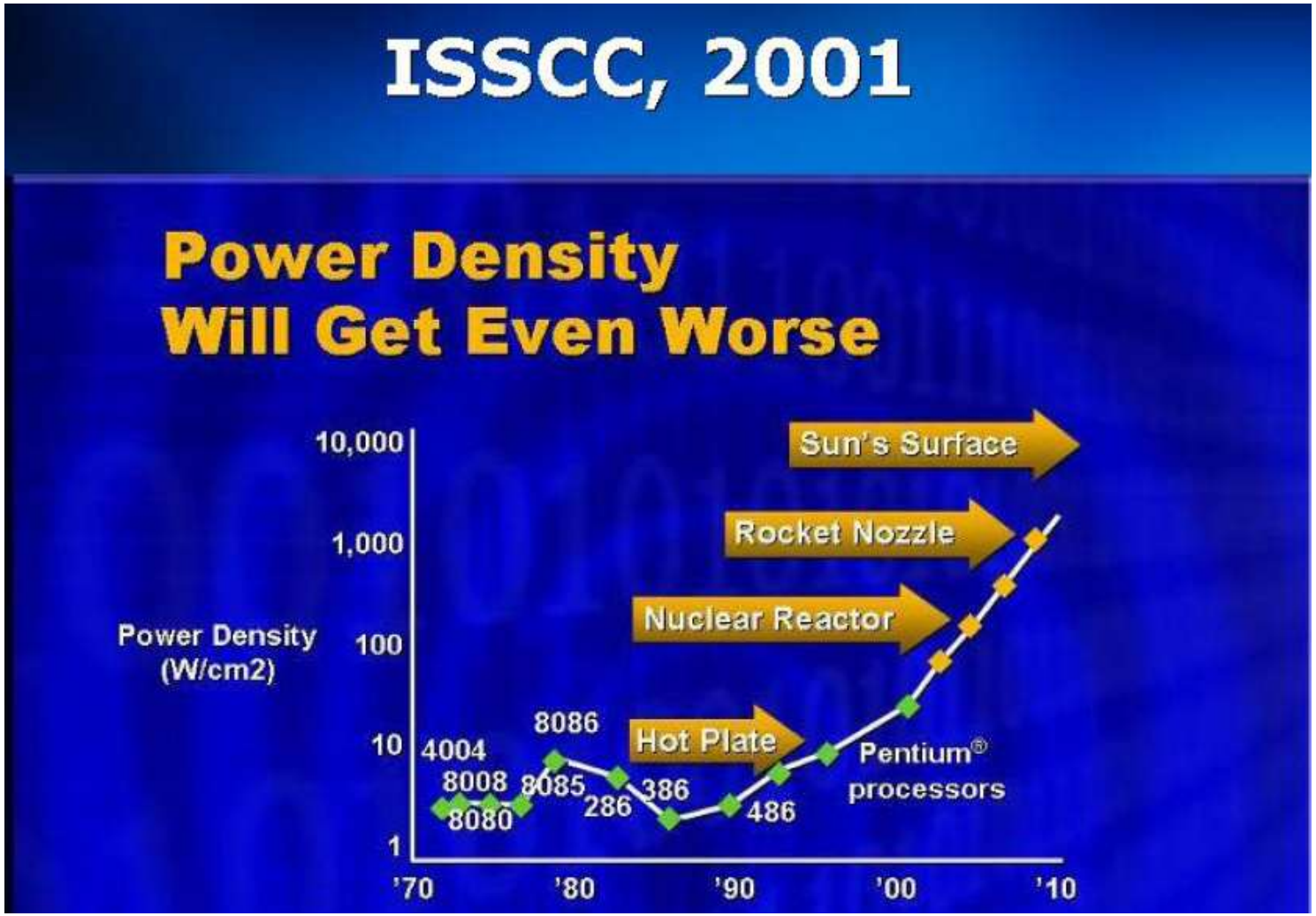
Forrás: R Hetherington, „The UltraSPARC T1 Processor” White Paper, Sun Inc., 2005

A disszipációs korlát



6.2. ábra: Intel processzorok fajlagos disszipációja

A disszipáció növekedés problémájának felismerése (ISSCC 2001, Gelsinger P. Intel)



A disszipációs korlát konzekvenciái:

- Magasabb órafrekvencia ⟶ Magasabb disszipáció ⟶ órafrekvenciák stagnálása
- Az órafrekvencia növelésén alapuló fejlesztési irány háttérbe szorulása
- A processzorok tervezésében a disszipáció csökkentő technikák előtérbe kerülése
- A disszipáció csökkentése céljából bevezették a dinamikus feszültség és frekvencia szabályozást (DVFS)

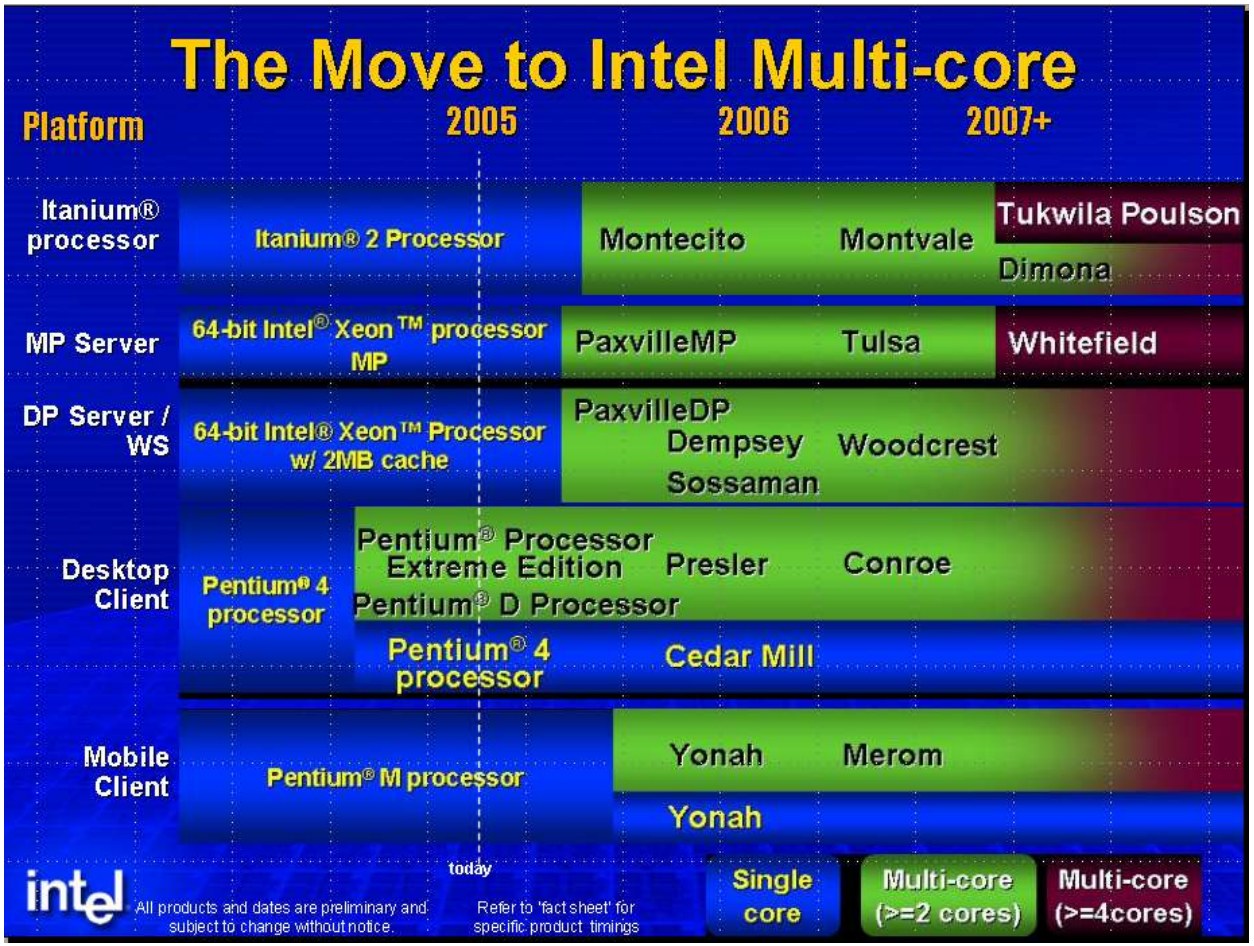
  Működése:

  - meghatározzák a szükséges teljesítményt
  - frekvencia hozzáillesztése a teljesítményhez
  - az adott órafrekvencia fenntartásához szükséges minimális feszültség beállítása

  (később kiegészül az AVS-el (Adaptive Voltage Scaling))

Eredmény:
Magasabb szintű párhuzamosságok, majd többmagos architektúrák megjelenése és elterjedése.

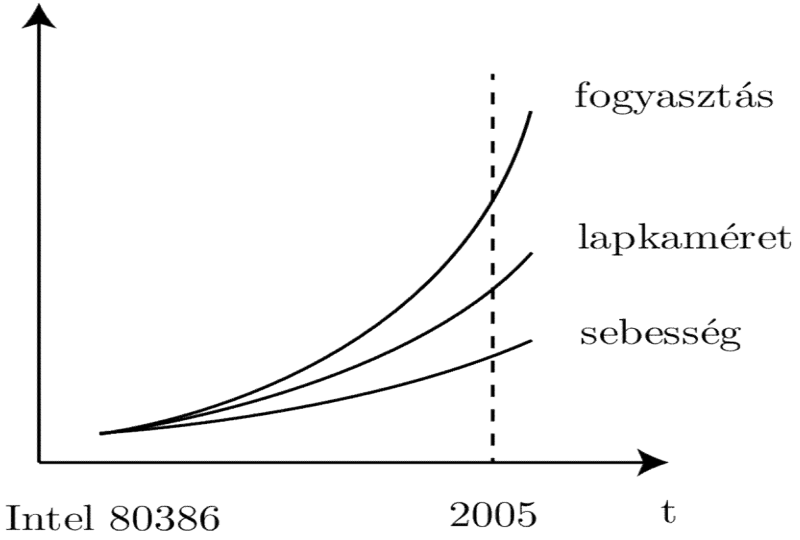
Paradigmaváltás



Többmagos processzorok robbanásszerű elterjedése az Intel processzorok példáján

Szál szinten párhuzamos architektúrák:

Az Intel 80386-os processzora óta bevezetett újítások a fogyasztás és a lapkaméret növekedésével jártak, viszont ehhez képest a teljesítmény csak kisebb mértékben nőtt.
Tehát a teljesítmény növekedése nem állt egyenes arányban a komplexitás növekedésével.
Ez elsősorban az egymagos processzorokra igaz.
Következmény: egyre több tranzisztor kell és nő a fogyasztás.
A teljesítmény további növeléséhez el kellett térni a klasszikus tervezési elvektől, ahol csak utasítás szinten használták ki a párhuzamosságot.
A fejlődés következő lépcsőjét a szál szintű párhuzamosság kihasználása jelentette.



A szál a program legkisebb önállóan végrehajtható része ⟹ párhuzamosan futtatható.
Míg az utasítás szintű párhuzamosság felderítésére önmagában is képes a hardver, a szálak kihasználásához az OS támogatására is szükség van.
Cél az egyetlen fizikai magban a feldolgozás során keletkező üresjáratok kihasználása!

A párhuzamosság formái:
- implicit: a programozó szekvenciális programot ír, a hardver és a szoftver párhuzamosítja a kódot. Ide tartozik minden szintű párhuzamosság.
- explicit: a programozó kifejezetten párhuzamos programot ír

Fontos, hogy egy szálban nincs párhuzamosság, ezért a teljesítmény növeléséhez több szál létrehozása szükséges. Ez egy magasabb szintű párhuzamosság, így összetettebb is.

Szálak származtatása:

- Különböző alkalmazásokból (multiprogramozás).
- Ugyanabból az alkalmazásból:
  - Multitasking
  - Multithreading

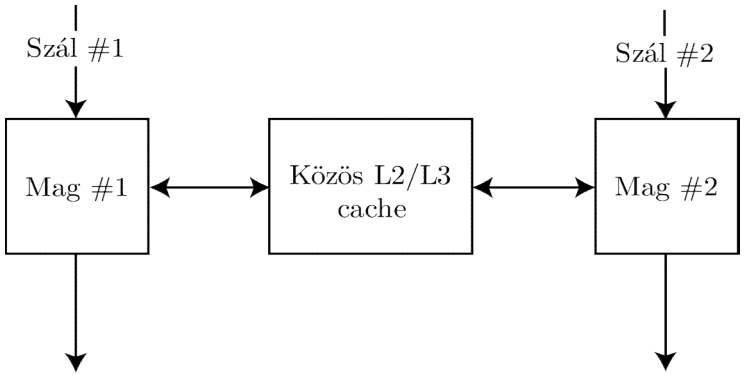
ÓBUDAI EGYETEM

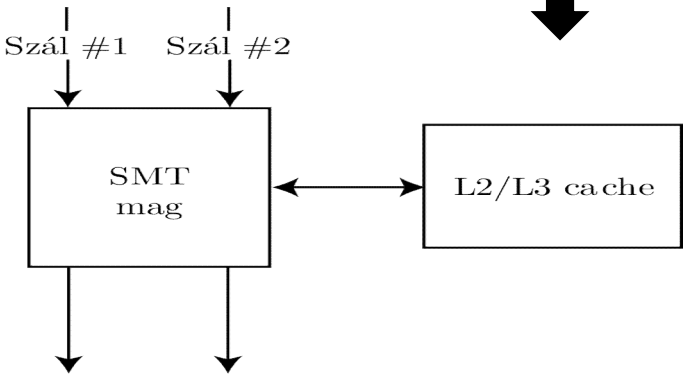
Többszálúság csoportosítása:

- Szoftveres többszálóság: többszálú alkalmazások, vagy OS futtatása egyszálú CPU-n, például időosztásos technikával
- Hardveres többszálúság: többszálú alkalmazás futtatása többszálas CPU-n

Többszálú CPU-k típusai:

- SMP – Symmetric multiprocessing: ugyanarra a lapkára implementálva kettő vagy több magú processzor párhuzamosan futtatja a szálakat
- SMT – Simultaneous multithreading: egy SMT mag képes arra, hogy két szálat párhuzamosan tudjon futtatni. Körülbelül +5% komplexitás növekedéssel akár 30%-os teljesítmény növekedés érhető el segítségével (ügyességen alapuló megoldás)





Több szál futtatása P4 CPU-n (Hyper Threading nélkül):

Ha nincs meg a hardveres támogatás akkor az új szál futtatása előtt az aktív szál kontextusát el kell menteni a memóriába, ezután a másik szál kontextusát be kell tölteni. A kontextus váltás akár 2-3 ezer óraciklus is lehetett! ⟶ LASSÚ!
(a durván szemcsézett szálszintű párhuzamosságnak felel meg)

OS szálkezelése:

A modern operációs rendszerek minden folyamatra külön lap táblát tartanak nyilván, ez az ún. TLB (Translation Lookaside Buffer: lapkezelő segédpuffer). Ezekben található az adott folyamat minden elérhetősége (virtuális címtartomány!), ami a memóriában vagy a háttértáron tárolódik. Tehát a szálkezelés hardveres és szoftveres támogatást is igényel.
Taszk (folyamat) váltáskor a TLB teljes tartalmát érvényteleníteni kell, ezért minden memória-hivatkozás TLB hibával fog járni, amíg a TLB meg nem telik az új taszk gyakran használt bejegyzéseivel.
Ha egy folyamaton belül van párhuzamosság, az gyorsan működik, mert nem kell sem a TLB-t, sem a cache-t üríteni! Csak a regisztereket és a PC-t kell átállítani.

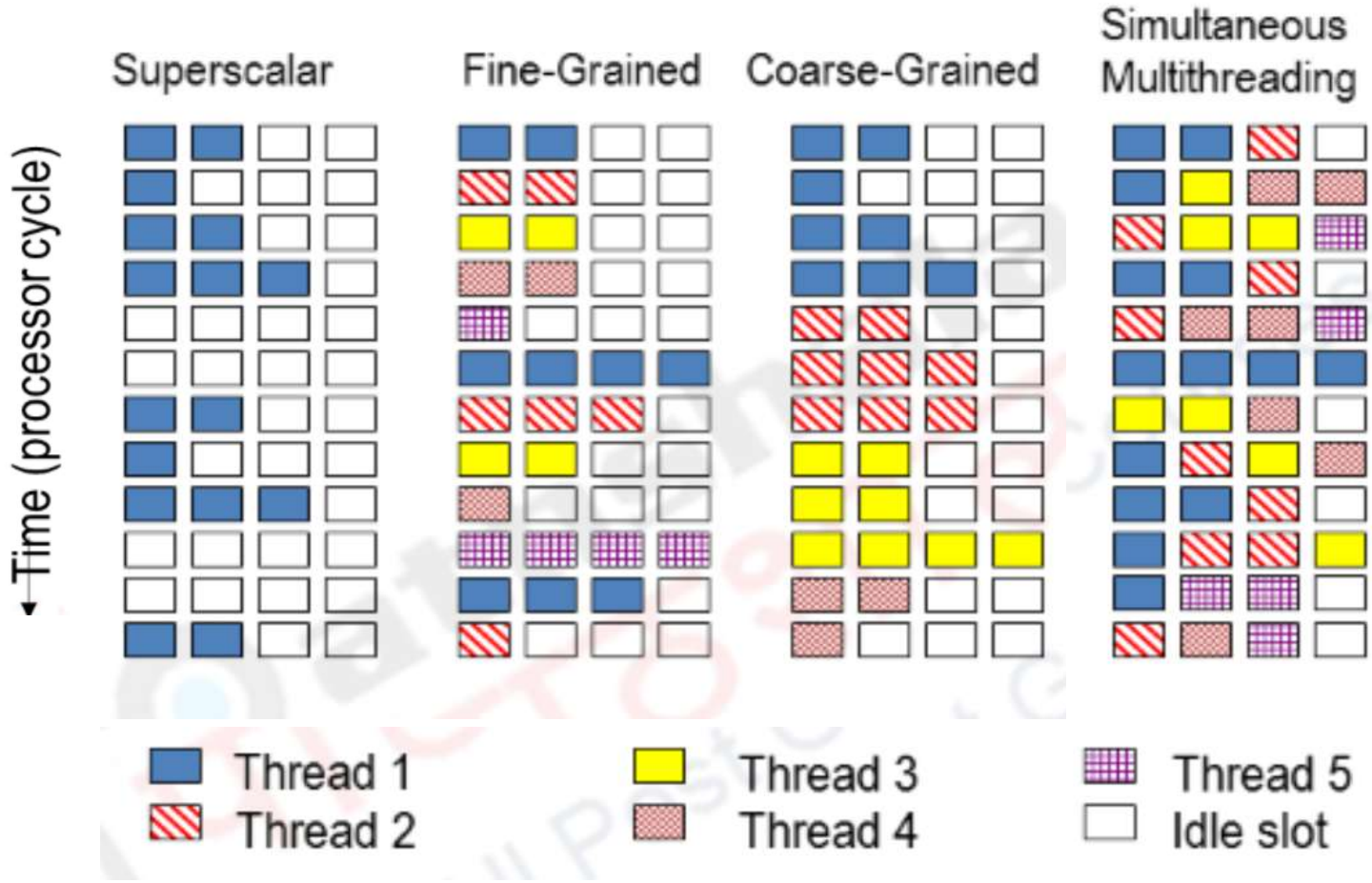
Szál szinten párhuzamos architektúrák osztályozása:

- Finoman szemcsézett
  A CPU kettő vagy több szálat futtat és órajelenként vált a szálak között. A gyakori váltások miatt az ilyen processzorok több regisztert tartalmaznak, így azokban több szál adatai is elférnek (nem kell várni a memóriából való betöltésre).
  Például kétszálú CPU esetén mindkét szál kontextusát tároljuk a regiszterekben és egy kontextus kapcsoló segítségével történik a váltás. Következménye, hogy nincs késleltetés a váltások között. A teljesítmény növekszik, mivel, ha az egyik szál függőségre vár, az nem blokkolja a másik szálat. Cél az üres járatok feltöltése. Akár 20%-os teljesítmény növekedést is eredményezhet.

- Durván szemcsézett / eseményvezérelt (SoEMT – Switch on Event Multithreading)
  Amennyiben az egyik szál futása megakad (például függőség miatt), akkor vált a másik szálra. Probléma, hogy a szál megakadását érzékelni kell, ami általában 1-2 óraciklus időigényű.

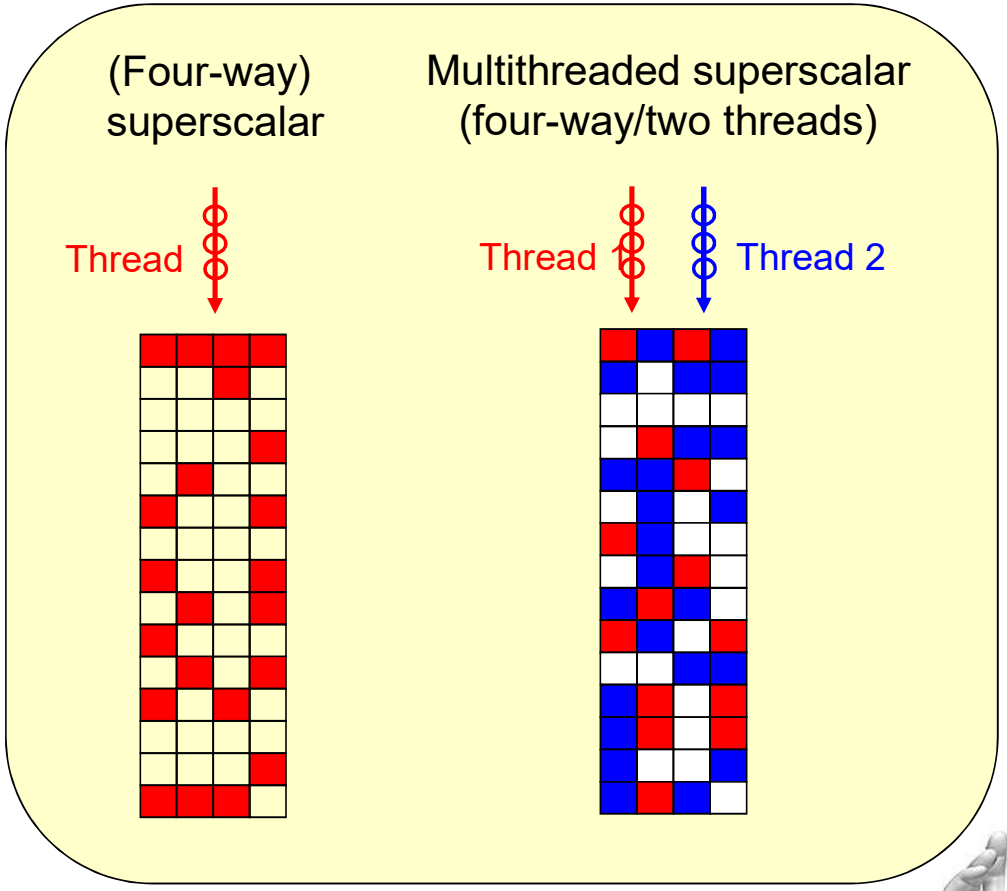
- SMT (Simultaneous multithreading)
  Az elvét 1968-ban írták le, gyakorlati megvalósítása viszont csak a 2000-es években kezdődött el. Ennek egyik ilyen megvalósítása az Intel Hyper Threading. Lényege, hogy az összes szál futtatása párhuzamosan történik. Az SMT-t támogató processzorok elsősorban szuperskalár architektúrák. Mivel sok végrehajtóegység van, ezek időnként kihasználatlanok A függőségek miatt! Ezeket töltjük fel a második szál utasításaival

Példa: különbség az egyszálas és az előző lapon felsorolt többszálas CPU-k működése között:
(mindegyik 4 utasítás széles) jól látszik az üresjárati ciklusok számának csökkenése



A párhuzamosság megvalósításához az alábbi hardveres követelményeknek kell teljesülni:

- Számos erőforrást meg kell többszörözni. Például szálankénti program counter és regiszter tároló készlet. Így a szálak közötti váltás nagyon gyors.
- Az erőforrásokat meg kell tudni osztani a szálak között. Szükség esetén egyesíteni is, ha egy szál megakadása esetén az aktív szál a teljes erőforrást tudja használni.



Intel Hyper Threading:

Az Intel Hyper Threading technológia egy két szálas SMT architektúra.
Először a Northwood alapú Pentium 4-es processzorokba került be.
A két szál utasításait egyszerre hajtja végre sorrenden kívüli kiküldéssel.
Egy mag az OS számára két külön logikai magnak látszódik, ezért úgy ütemezi az utasításokat, mintha két processzoros rendszeren futna.
Tehát egy CPU egyszerre két logikai CPU utasításait hajtja végre.
A processzorban ekkor egyszerre két architektúrális állapot van jelen!
Ehhez plusz tranzisztorok kellenek, de ezek száma az CPU egészéhez mérten elenyészően kevés (~5% méret és fogyasztás növekedés).
Cserébe 0-30% teljesítmény többlet érhető el (alkalmazás függő)!

Üzemmódok:

- ST (single task)
  Egy szál végrehajtása történik amellett, hogy a megosztott erőforrások egyesítésre kerülnek és azok felett az éppen aktív logikai CPU rendelkezik .
  Két állapot lehet, attól függően, hogy melyik logikai CPU aktív: ST0 vagy ST1.
- MT (multi task)
  Több szál végrehajtása történik párhuzamosan.

A CPU először MT módban indul.
Ha az egyik szál megakad, ST módba vált a processzor.
A függőség megszűnésével a CPU visszakerül MT ütemmódba.
Az üzemmódok közötti váltás a „HALT” utasítással segítségével történik, ami megszakítja a CPU futását és energiatakarékos állapotba helyezi.
A „HALT” után attól függően kerül a CPU ST0, vagy ST’ állapotba, hogy melyik logikai CPU-n futott le a HALT.
Ezt csak az OS, vagy más, alacsony szintű alkalmazás adhatja ki!

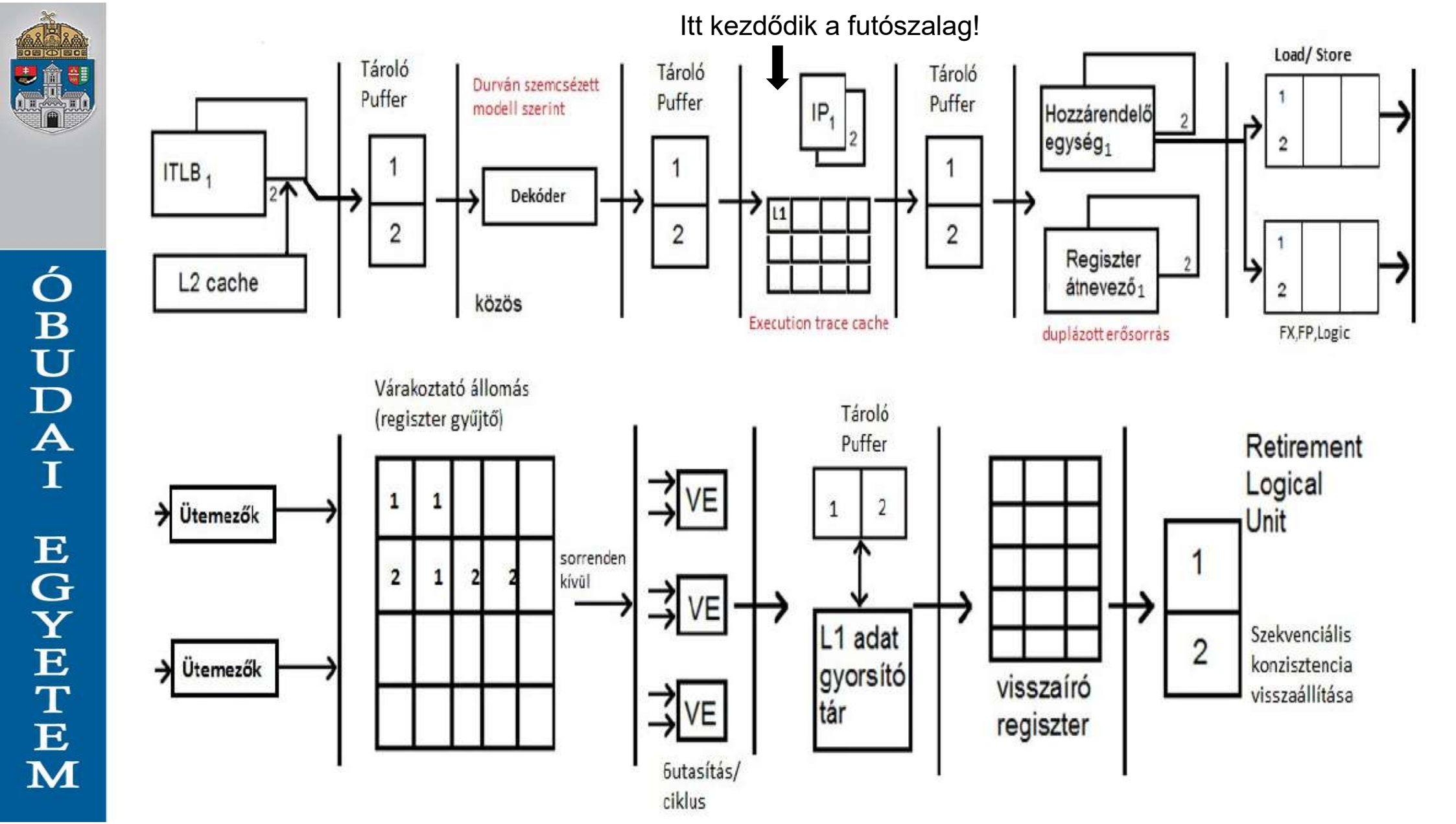
Nem véletlen, hogy Netburst architektúra kapta meg először a HT-t, hiszen a hosszú futószalag miatt sok függőség keletkezett és ez sok várakozást (üresjáratot) eredményezett. Hatékonyság csökkent!

SMT megvalósítási célok:
- kis magméret növekedés
- egyik szál várakozása esetén a másik szál késleltetés nélkül folytathassa a műveleteket (ehhez tárolók kellenek a belső magban)
- Egy szál futtatása esetén a sebesség ugyanolyan gyors legyen, mintha nem lenne megosztva a mag! (egyesíteni kell az erőforrásokat)

Működési vázlat:

Az utasítás címfordítási segédpuffer (ITLB) az L2 cache-ből kapja az utasításokat, majd egy elválasztó pufferen keresztül kerülnek a futószalagon kívüli dekóderbe. A dekóder SoEMT szerint dekódol. Egy tároló pufferen keresztül töltődik fel az Execution Trace Cache (L1) a dekódolt utasításokkal. Újabb elválasztó pufferen keresztül kerülnek az utasítások a regiszterátnevező és hozzárendelő egységbe, mely szétválasztja őket külön pufferekbe. Az ütemező segítségével biztosítja ezek tárolását a várakoztató állomásba, ahova az adat az L1 data cache-ből kerül az utasításhoz. A várakoztató állomásból sorrenden kívüli kiküldés történik a végrehajtó egységek felé. Végrehajtás után az eredmények a visszaíró regiszterbe kerülnek, amiből a RLU biztosítja a szekvenciális konzisztencia visszaállítását.

Minden utasítás az adott Logikai CPU-hoz való tartozást jelző címkével van ellátva, amely végigköveti a RISC utasítást a futószalag fokozatokban.
Ha a megcímzett utasítást az L1 cache-ben nem találja, akkor az ITLB segítségével határozza meg a fizikai címet.
Az Out of Order Execution Unit sebességre optimalizálva hajtja végre az utasításokat (ossza el a végrehajtó egységeket a szálak között). Egy szál esetén az összes erőforrást biztosítja neki.
Az RLU az architektúrális állapotokat figyelve a megfelelő LP-be írja vissza az eredményeket.


Itt kezdődik a futószalag!
ITLB 1
2
L2 cache
Tároló Puffer
1
2
Durván szemcsézett modell szerint
Dekóder
közös
Tároló Puffer
1
2
IP1
2
L1
Execution trace cache
Tároló Puffer
1
2
Hozzárendelő egység1
2
Regiszter átnevező1
2
duplázott erősorrás
Load/ Store
1
2
1
2
FX,FP,Logic
Várakoztató állomás (regiszter gyűjtő)
Ütemezők
Ütemezők
1 1
2 1 2 2
sorrenden kívül
VE
VE
VE
Tároló Puffer
1 2
L1 adat gyorsító tár
visszaíró regiszter
Retirement Logical Unit
1
2
Szekvenciális konzisztencia visszaállítása
Utasítás/ ciklus

ÓBUDAI EGYETEM

Gyakorlati példa:

Intel Nehalem CPU (2008)

Core-i architektúra, részben visszatérés a PIII elvekhez