TPA XP-1
Párhuzamos számítógép

megvalósíthatósági tanulmány

Írta : Matakovics György
A forgalmi modelleket készítette: Magos László

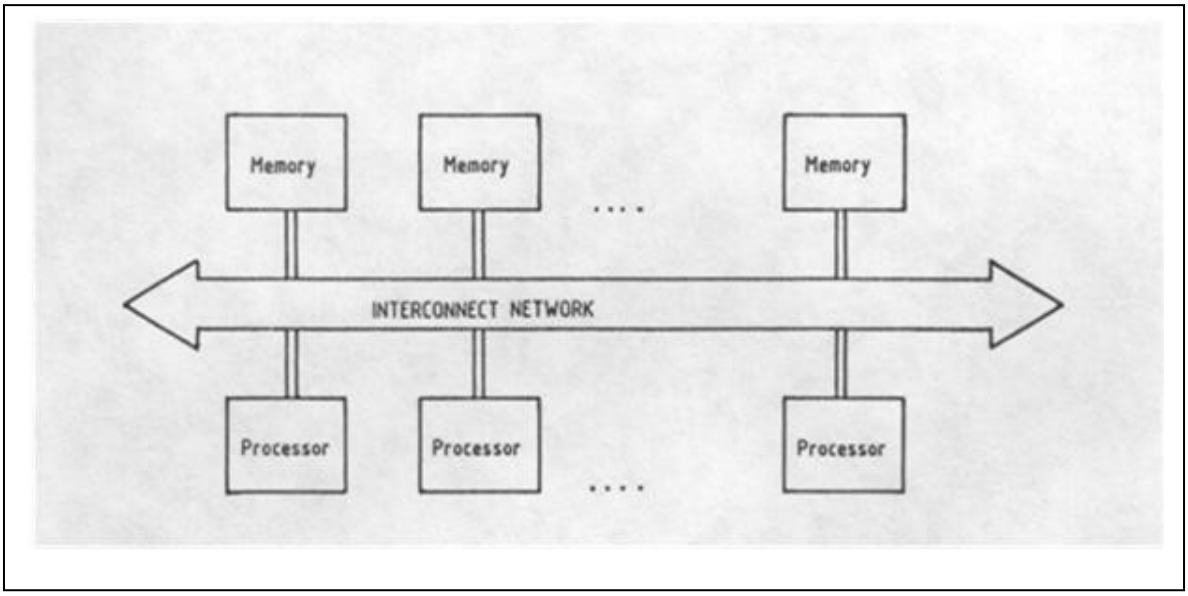
BEVEZETÉS

A számítógépek teljesítménynövelésének hagyományos eszköze az egyedi processzor teljesítményének növelése, melynek az aktuális technológiai lehetőségek szabnak határt. E probléma feloldásának egyik lehetséges módja a feladaton belüli párhuzamosságok feltárása és a feladatok egyidejű végrehajtása több processzoron; illetve multiuser operációs rendszeren belül több felhasználói program egyidejű futtatása. A fentiekkel magyarázható, hogy napjainkban mindinkább teret hódítanak a multiprocesszoros számítógépek. Ezzel a törekvéssel találkozhatunk mind a nagy számítógépgyártó cégeknél (IBM,DEC,Cray,...), mind a kisebb sorozatú .speciális gépek gyártóinál (Elxsi, Thinking Machines, BBN, Ncube, Gould, Convex, FPS, ...).

Az utóbbi években, mióta Japán az ötödik generációs számítógépek kifejlesztésére irányuló programját elindította, a többi számítástechnikai nagyhatalom is hasonló fejlesztésekbe kezdett. Az új generációs számítógépek kikísérletezése közben kialakult verseny volt a fő katalizátora a paralel1 rendszerek fejlesztésének. A párhuzamos architektúrák kialakításának számos előnye van :

bővíthetőség	azonos processzorelemeket alkalmazva a teljesítmény széles skálája átfogható a felhasználói igényeknek megfelelően
megbízhatóság	a rendszer működőképes marad egyes processzorelemek meghibásodása esetén is
ár	multiprocesszoros rendszerek fejlesztési költsége, infrastrukturális beruházási igénye alacsonyabb, mint az egyedi processzor fejlesztési költségei.

A nagy teljesítményű, alacsony árú, 32 bites mikroprocesszorok és a párhuzamos programozási nyelvek (OCCAM, ADA, FORTRAN 8X) megjelenése méginkább segíti a párhuzamos architektúrák kialakítását.
Osztott memóriájú multiprocesszoros rendszerekben az Interconnect Network-ön a processzorok számától és azok teljesítményétől függő forgalom van.



Ha az Interconnect áteresztőképessége kisebb a szükségesnél, akkor a processzoroknál teljesítménycsökkenés következik be.

Mivel a processzor-memória forgalom arányos a processzorok számával és azok teljesítményével, ezért az Interconnect sávszélességét is arányosan növelni kell, hogy elkerüljük a degradációt. Így a nagyszámú és nagyteljesítményű processzort tartalmazó rendszerekben az Interconnect nagyon összetetté és bonyolulttá válik. Emiatt a sávszélesség növelésével párhuzamosan az Interconnect forgalmát csökkenteni kell, amennyire az lehetséges.

A processzorok teljesítményének csökkenését a következő tényezők befolyásolják:

- a több processzor alkalmazása miatt az Interconnect-en kialakuló konfliktushelyzetek

  kommunikációs konfliktus

  az Interconnect Network-ön több processzor akar egyidőben forgalmazni

  memóriakonfliktus

  a memória egyszerre csak egy kérést tud kiszolgálni (sorbanállás)

  válaszidő, melynek leglényegesebb összetevői :

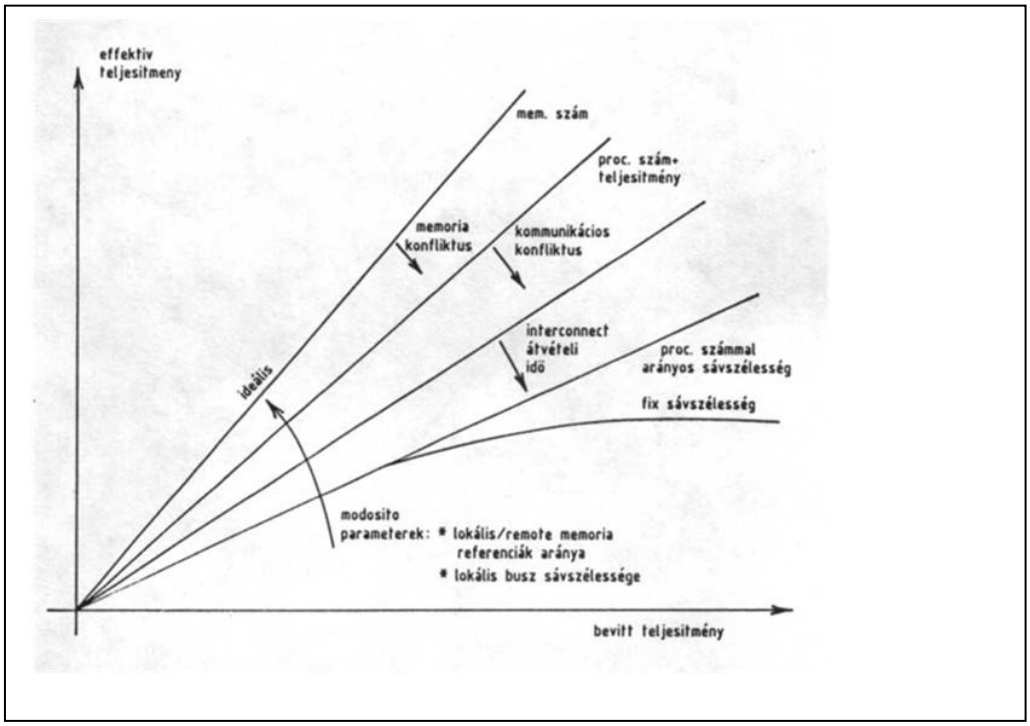
  memória válaszidő (az adat előkeresésének az ideje)

  Interconnect késleltetési idő

  (a válaszidőt - az adat kérésétől az adat megérkezéséig eltelt időt)- a későbbiekben a processzor szempontjából fordulási időnek is fogjuk nevezni.)

2. A MODELLEZÉS LÉNYEGES PARAMÉTEREI

A multiprocesszoros architektúrák vizsgálatához kidolgoztunk egy matematikai/statisztikai modellt, mely alkalmas a Crossbar/Busz/Hierarchikus rendszerek elemzésére. Segítségével egzakt módon tudjuk meghatározni, hogy egy adott processzorelem a környezetétől függően milyen teljesítményűvé degradálódik.



vízszintes tengely: egy egyedi processzorelem névleges teljesítménye [MIPS] ("bevitt" teljesítmény)

függőleges tengely: egy egyedi processzorelem mekkora teljesítményűnek tekinthető a paraméterekkel megadott környezetében [MIPS] ("kivehető " teljesítmény)

A modellhez szükséges leglényegesebb bemeneti paraméterek a következők:

- egyedi processzorteljesítmény [Proc_mips , MIPS]

  leglényegesebb összetevője annak, hogy mekkora forgalmat [Mbyte/sec] generál egy processzorelem az Interconnect-en
- egy utasítás végrehajtásához szükséges byte szám [Proc_bpi]

  hasonlóképpen a forgalomra van hatással
- Crossbar típusúaknál (X) a fordulási idő [Rem_mbps ,Mbyte/sec]
- Busz típusúaknál (B) a busz sávszélessége [Rem_mbps,Mbyte/sec]

Módosító paraméterek:

- lokális/remote memória elérések aránya [Local_rat,%] (Cache/Lokális memória)
- lokális memória elérési sávszélesség [Loc_mbps,Mbyte/sec]

A modellezés részletes leírását az 1.számú melléklet tartalmazza.

INTERCONNECT MEGVALÓSÍTÁSOK

A továbbiakban elemezzünk néhány alapvető Interconnect megvalósítási lehetőséget a konfliktushelyzetek és a fordulási idő szempontjából.

1. Common busz Interconnect
2. Crossbar jellegű Interconnect-ek
3. Multistage Interconnect Network

COMMON BUSZ INTERCONNECT

Állandó sávszélességű Interconnect. Common Access Throughput értéke (az a jellemző, hogy egyidőben hány kérést tud kiszolgálni) 1, (CAT=1) , azaz csak szekvenciális kiszolgálást tesz lehetővé. Viszonylag egyszerű megvalósíthatósága és programozási kényelme miatt a legelterjedtebb megoldás. Mivel már az Interconnect használatánál konfliktushelyzet alakul ki, egy cég sem alkalmazza 32 processzornál többet tartalmazó rendszerben, (még ha a busz sebessége elegendő is lenne 32 processzor adat igényéhez). Sávszélességének növelése komoly technológiai problémákat vet fel. Az Interconnect busz sebességének növelése (ciklusidejének csökkentése a busz hosszának csökkenésével jár (ld. ELXSI 6400 Gigabus 25 ns-os ciklusideje) amely a buszon elhelyezhető processzorok és memóriák számának korlátozásával jár együtt. (ELXSI backplane kb.30 cm.) Ez pedig a memória konfliktusok növekedésével, azaz teljesítménycsökkenéssel párosul.

Buszos rendszerekben mindegyik processzor Cache memóriával csökkenti a buszforgalom/teljesítmény arányt,de még így is ez az érték átlagosan 2-3 Mbyte/Mips.

16 darab 5 Mips (elméleti eredő teljesítmény 80 Mips) teljesítményű processzorhoz legalább 150-200 Mbyte/sec áteresztőképességű Common buszra van szükség, amely legalább 40 ns-os ciklusidőt feltételez. Ilyen sebességű busz megvalósítása jelenlegi technológiai korlátaink miatt nem kivitelezhető.

A Common busz sávszélessége növelhető az arbitráció/command/válasz ciklusok elkülönítésével (split transaction, vagy pended bus),és átlapolásával. Ennek áramköri megvalósítása nagyon fejlett IC gyártási technológia nélkül nem sikerülhet (Ld. buszvezérlő áramkörök pl. VME buszra,VAXBI buszra mintegy 100.000 tranzisztorral).

3.1.1. NÉHÁNY BUSZOS ARCHITEKTÚRÁJÚ SZÁMÍTÓGÉP FŐBB JELLEMZŐI

CPU típus	PROC. szám	Telj. MIPS	Architektúra Interconnect	Shared memória	Cache	CPU
Multimax	22	16	Bus	Globális	van	egyedi
Balance 8000	12	8.4	Bus	Globális	van	
21000	30	21	Bus	Globális	van	NS32032
Elxsi 6400	12	156	Bus	Globális	van	egyedi
VAX 8840	4	22	Bus	Globális	van	egyedi

3.1.2.. NÉHÁNY BUSZ FŐBB JELLEMZŐI

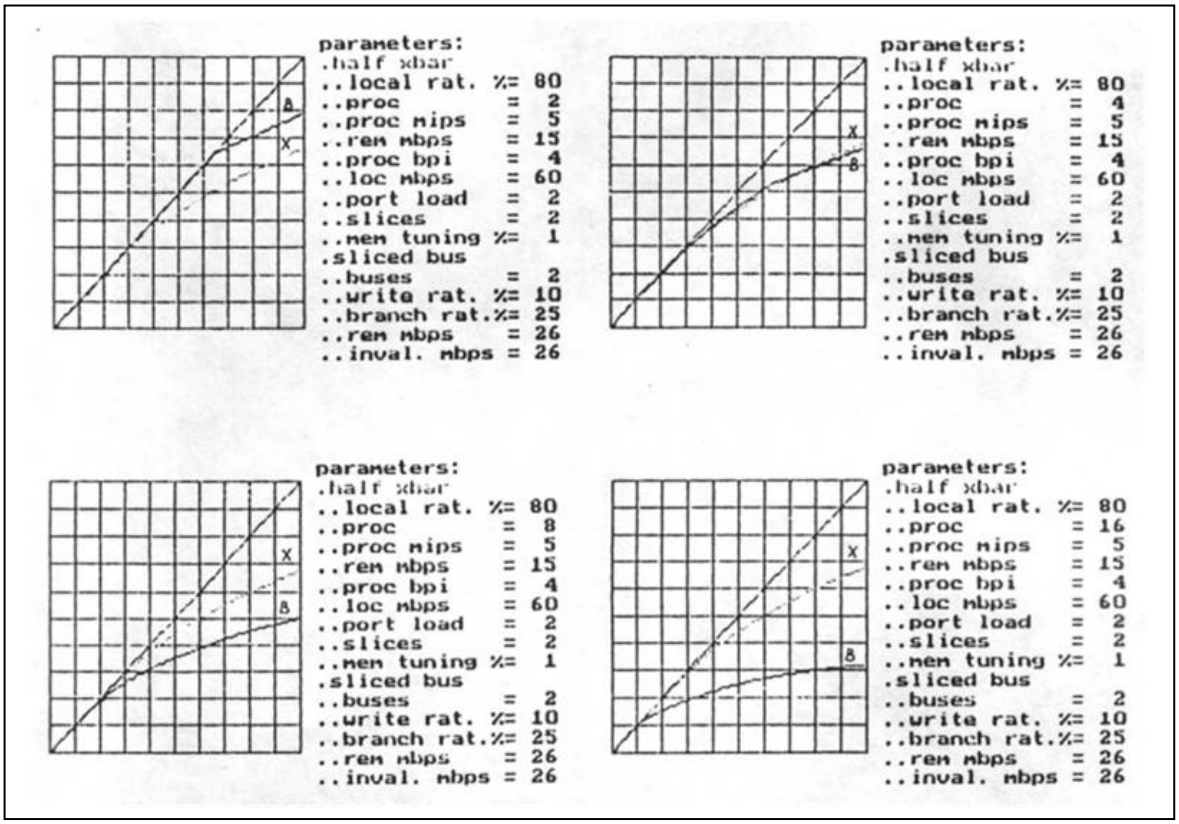
Busz	tipus	ciklusidő	sávszélesség
SBI VAX 11/780	szinkron	200 nsec	13.3 Mbyte/sec
VAXBI VAX 8000	szinkron	200 nsec	13.3 Mbyte/sec
XMI VAX 6000	szinkron	60 nsec	100 Mbyte/sec
GIGABUS ELXSI 6400	szinkron	25 nsec	320 Mbyte/sec
NANOBUS ENCORE MULTIMAX	szinkron	80 nsec	100 Mbyte/sec
VME BUS	aszinkron		40 Mbyte/sec
NAUTILUS VAX 8000	szinkron	45 ns	45 Mbyte/sec

A felsorolt értékek throghput sávszélesség értékek. A fizikailag elérhető effektív sávszélesség érték egy processzor szempontjából kb. 15-20 Mbyte/sec.)

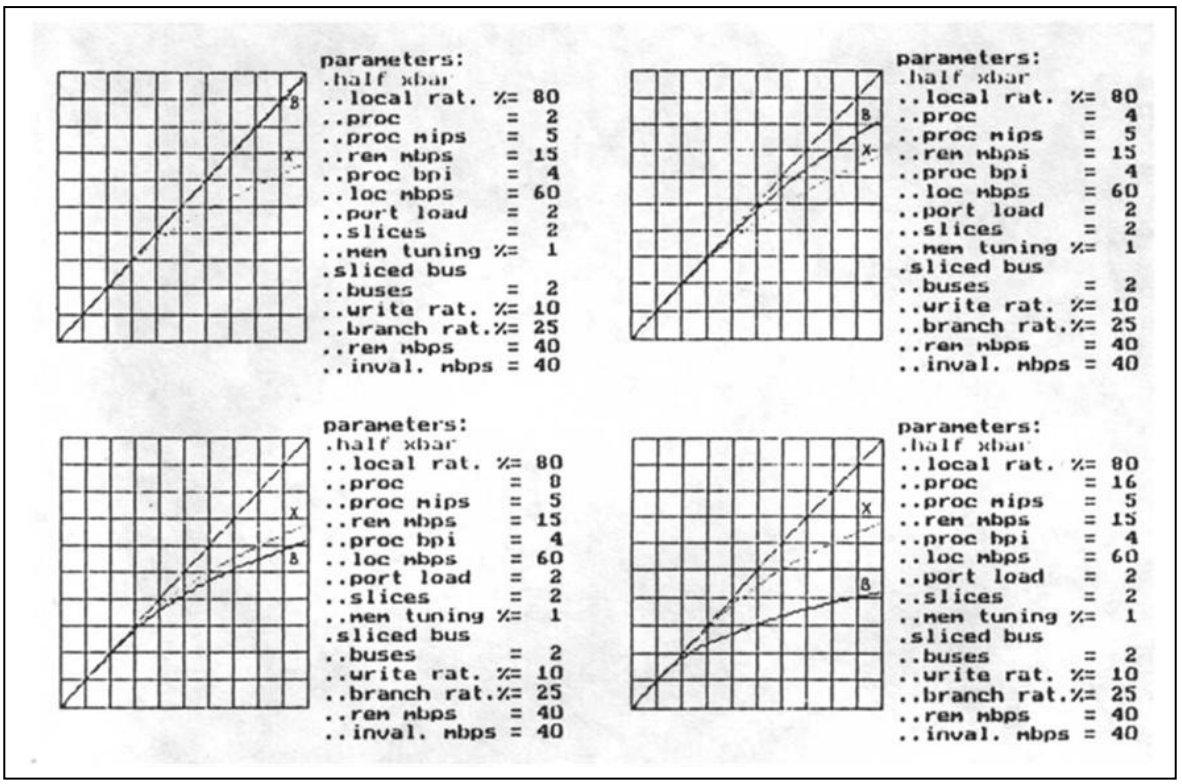
COMMON BUSZOS ARCHITEKTÚRÁK ELEMZÉSE

Modellünk segítségével elemezzük a Common buszra épülő rendszereket. A modellben a processzorok cache memóriával rendelkeznek, a busz hasított ciklusú szinkron busz (B).

Referenciaként az általunk kialakított, pont-pont kapcsolatú multi-port memóriás half-crossbar architektúra teljesítménygörbéje (X) látható. (A 45 fokos egyenes az ideális esetet mutatja.)

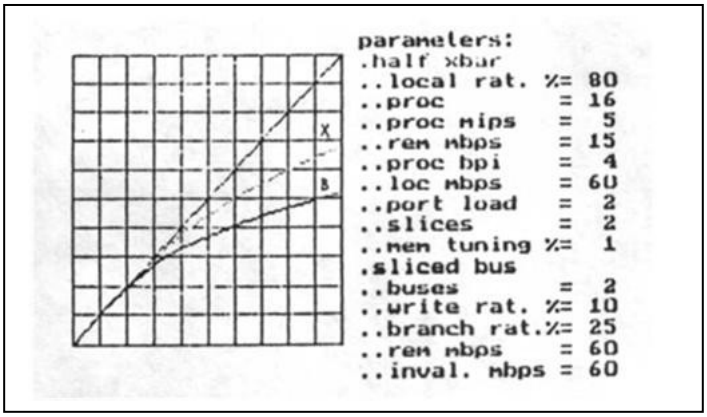


Változtatott paraméter:processzorok száma (proc), a processzorok 5 MIPS névleges teljesítményűek,a busz sávszélessége(rem_mbps 26 MB/sec) A busz alacsony áteresztőképessége legfeljebb 4 processzor alkalmazását teszi lehetővé.(A processzorszám növelésével az egyes processzorelemeken a teljesítményveszteség rohamosan nő (B).



Változtatott paraméter:a processzorok száma. A processzorok 5 MIPS névleges teljesítményűek, a busz sávszélessége (rem_mbps 40 MB/sec).

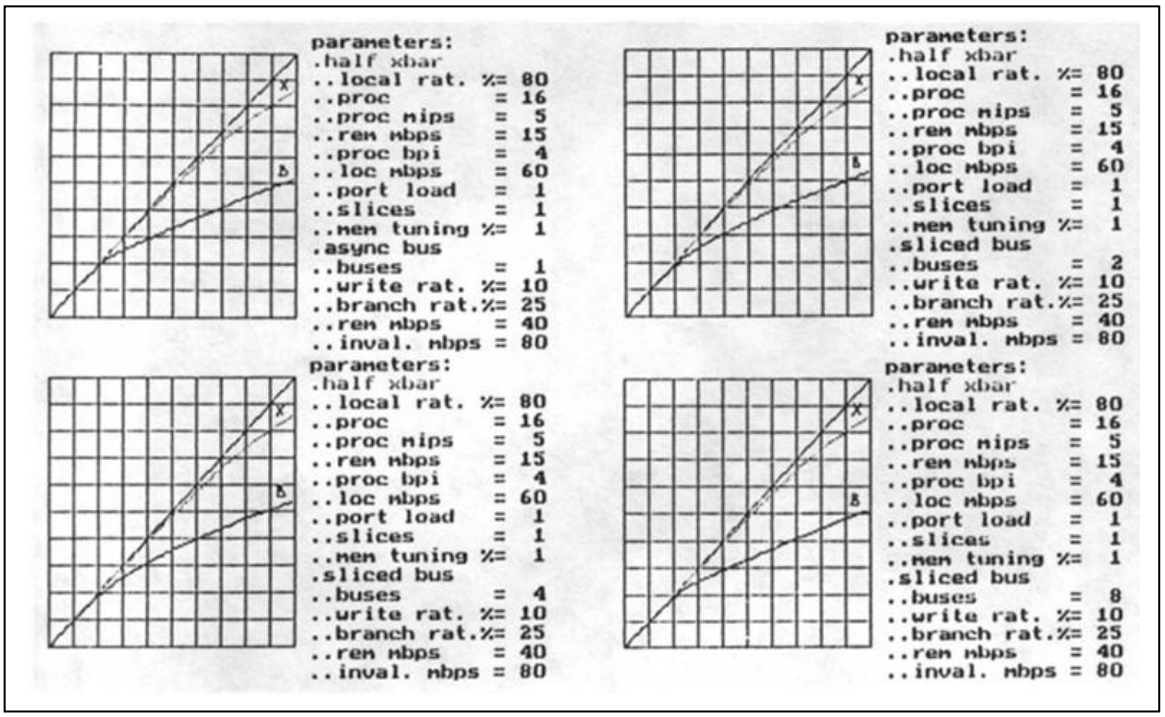
Növelve a busz sávszélességét,megközelítően 8 processzornál még elfogadható a degradáció (B).



16 processzornál még a 60 Mbyte/sec sávszélesség is kevés, pedig ez minimum 60-80 nsec-os ciklusidőt jelent.

HASÍTOTT CIKLUSÚ COMMON BUSZ ELŐNYEI

Érdekességként megmutatjuk, hogy a hasított ciklus meddig jár előnyökkel és azok milyen mértékűek:



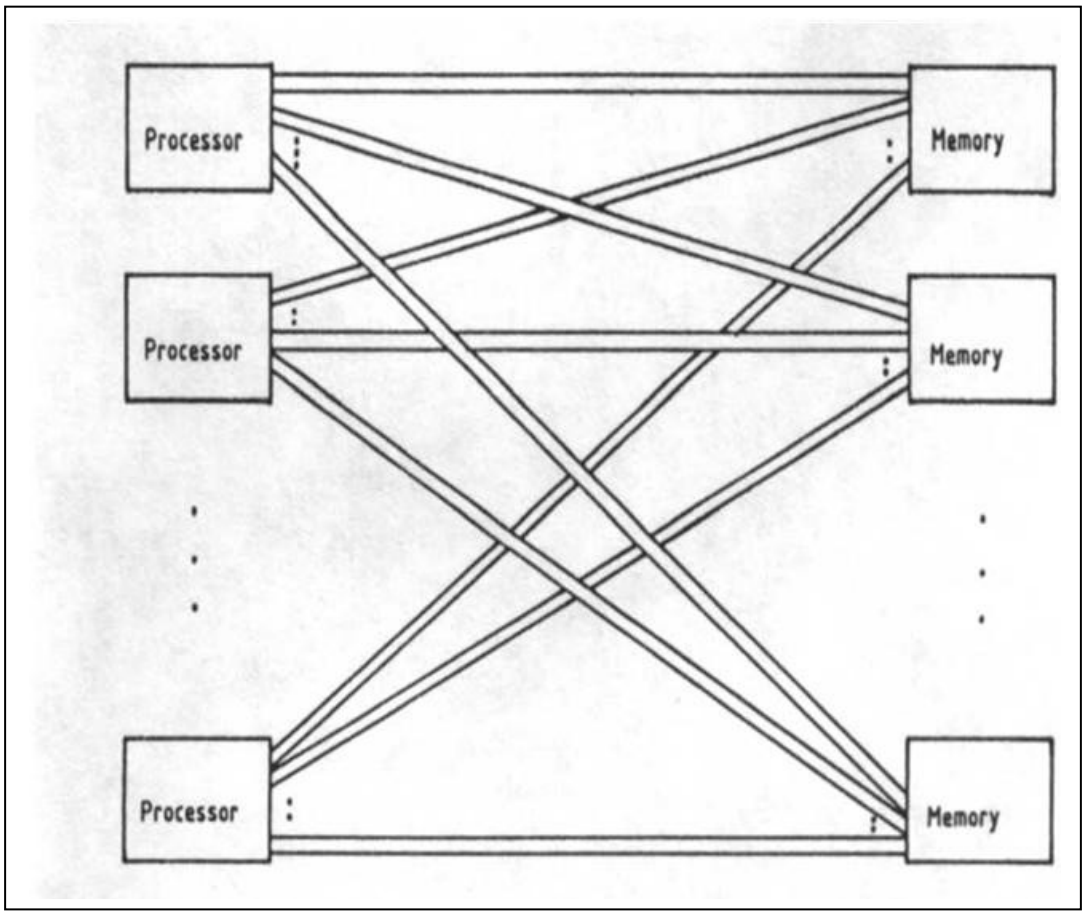
A busz 40 Mbyte/sec sávszélességű. A hasított ciklust a buszok számának növelésével lehet interpretálni.(ld. 1.számú mellékletben részletesen) Pozitív eredményt a két vagy négy ciklusra hasítás ad (buses=2,4). Természetesen ez az 5 MIPS teljesítményű processzorok környezetére igaz. Ugyanis a hasítás csak olyan buszon végezhető el,ahol a memória legalább olyan "lassú", vagy a buszciklusok olyan gyorsak,hogy a memória késleltetési ideje alatt egy másik buszciklus is lejátszódhat. Nyilvánvaló, hogy a busz csak akkor hasítható 4, vagy 8 ciklusra,ha a memória elég lassú.(Hacsak nem 10 nsec-os buszciklusidővel dolgozunk.) Ez pedig az 5 MIPS teljesítményű processzornál már degradációt okoz (utolsó ábra).

(Az 1.ábra a VME busznak felel meg. Aszinkron, nem hasítható ciklusú busz.)

3. 2. CROSSBAR JELLEGŰ INTERCONNECTEK

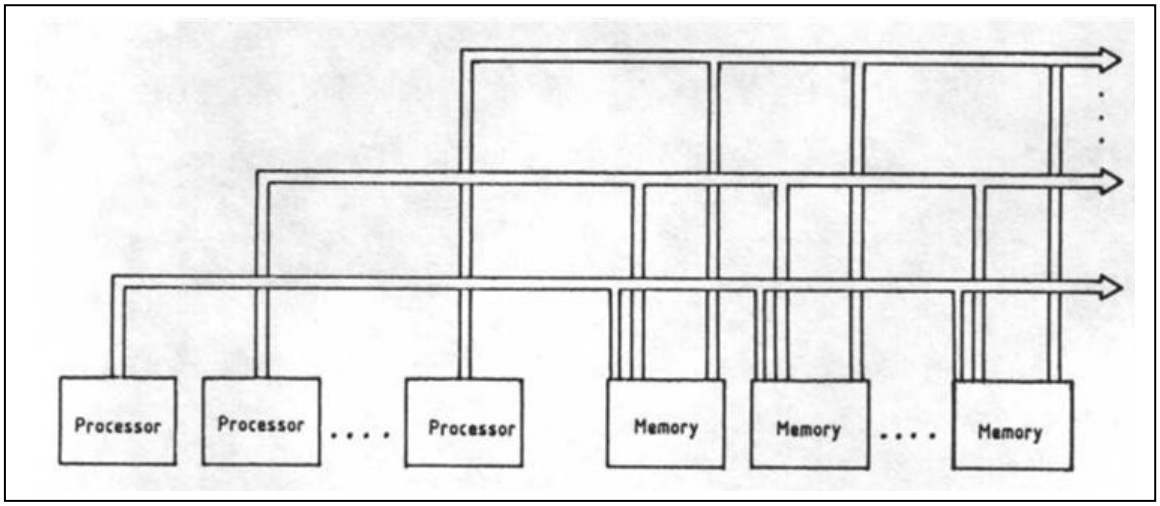
Olyan Interconnect típusok,melyekben nem alakulhat ki kommunikációs konfliktus. Sávszélességük arányos a processzorok számával, párhuzamos Interconnectek. Modulárisak, csak memóriakonfliktus helyzetet tartalmaznak. Megvalósítások:

3.2.1. PONT/PONT KAPCSOLATÚ MULTIPORT MEMÓRIÁS KIALAKÍTÁS



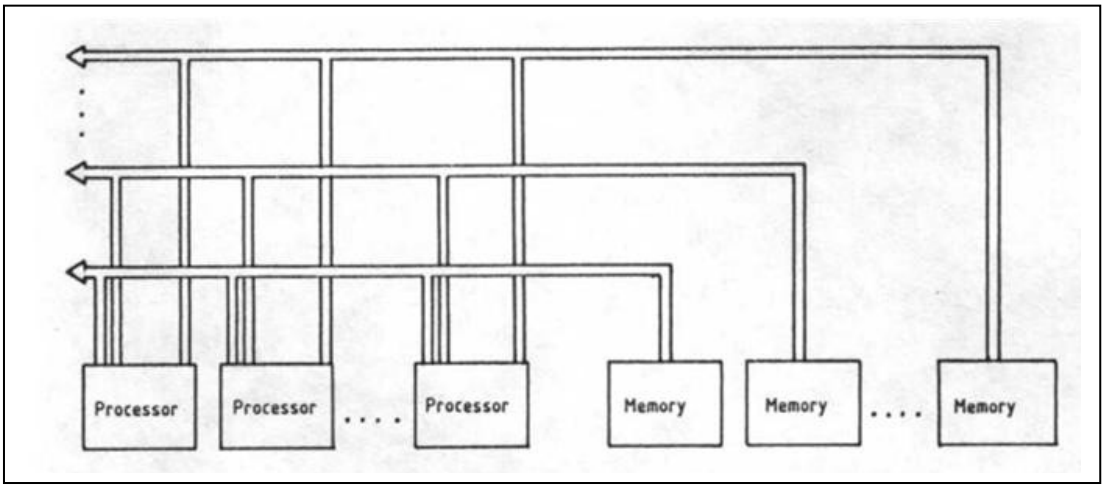
Pont-pont kapcsolattal az Interconnect késleltetési ideje jelentősen csökkenthető. A megoldás sok kábelt,nagyszámú adó/vevő áramkört tartalmaz, mely nem teszi lehetővé, hogy 4-8-nál több processzort tartalmazó rendszerben használjuk. Memóriakonfliktus kialakulhat, melynek teljesítménycsökkentő szerepe a memóriamodulok számának növelésével, (mely tovább bonyolítja a kialakítást), és azok interleave-elésével tompítható. (Ezt az architektúrát pl. a Burroughs cég alkalmazza.)

3.2.2. PONT/PONT KAPCSOLATÚ MULTIPORT MEMÓRIÁS KIALAKÍTÁS EGYSZERŰSÍTETT, "BUSZOSÍTOTT" VÁLTOZATA



"Buszosított" kialakítás. A processzor/memória egyedi sávszélességek emiatt rosszabbak, mint a pont/pont kapcsolattal.

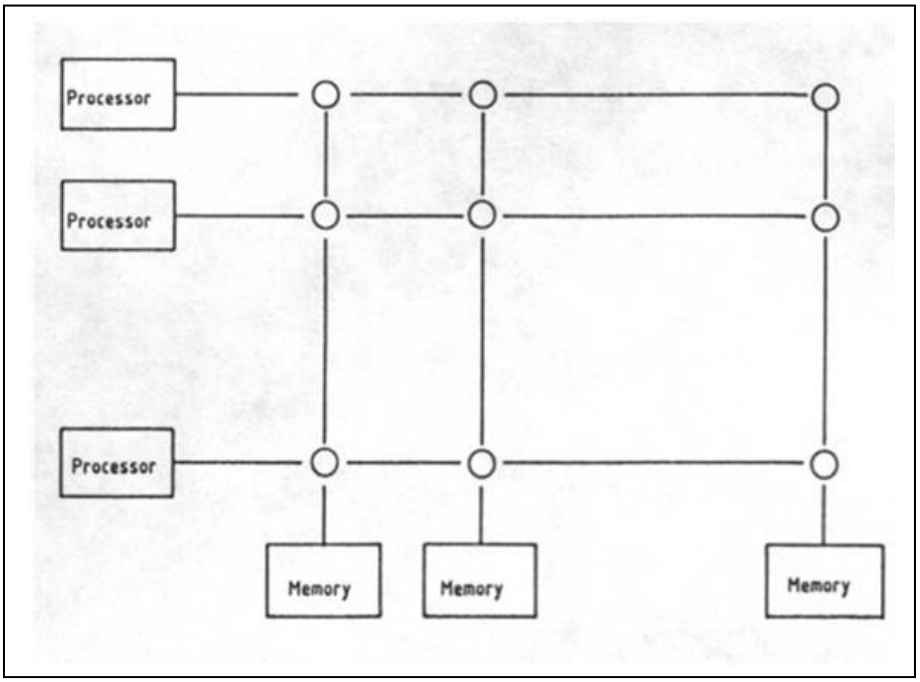
3.2.3. MULTIPLIKÁLT BUSZ INTERCONNECT (a buszok száma,a memóriák száma, és a processzorok száma megegyezik.)



Áramköri megoldása mindkét utóbbi rendszernek szinte azonos. A multiport memória alkalmazása kedvezőbb, mert memória oldalon a kérések eltárolhatók, pipeline technikával a memóriamodulok sávszélessége növelhető. A multiple busz változatnál a busz arbitráció miatti ütközés növeli az átviteli időt.

Mindkét architektúra csak korlátozott számú (legfeljebb 16) processzor esetén alkalmazható a backplane csatlakozópontok magas száma, Interconnect áramköri elemek mennyisége miatt.

3.2.4. KAPCSOLÓÁRAMKÖRÖS CROSSBAR INTERCONNECT

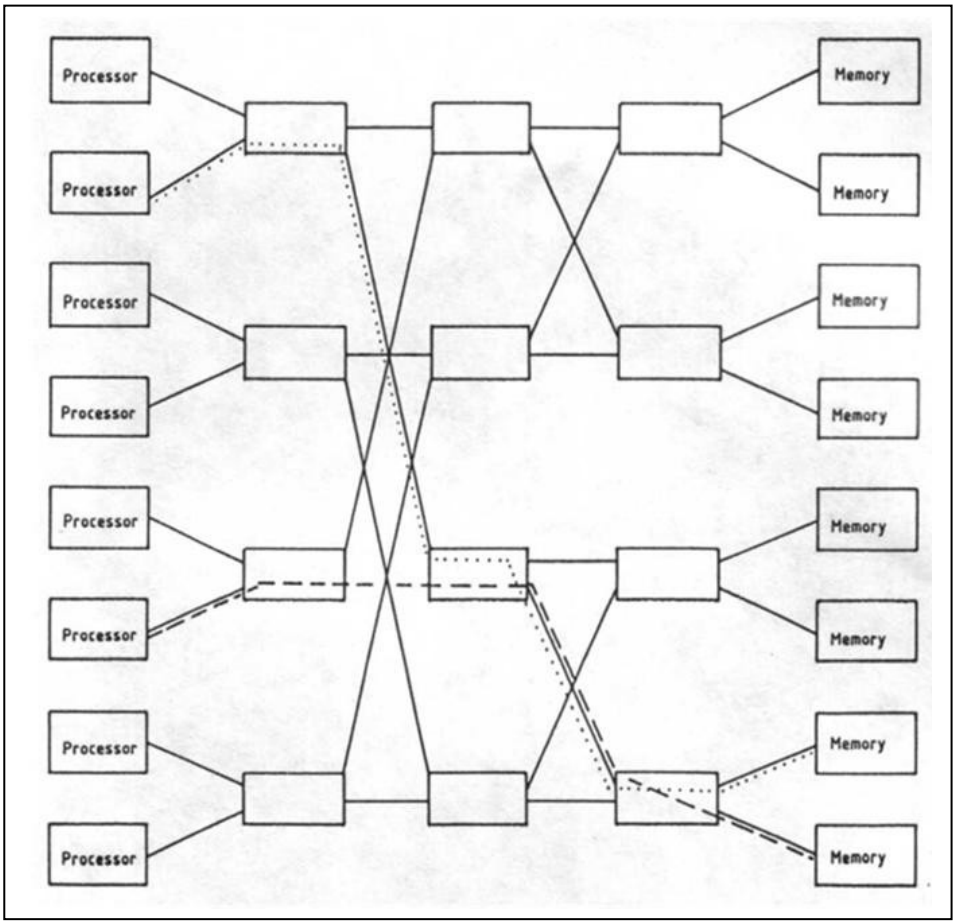


A kapcsolóelemek száma bitenként N processzor és memória esetén N*N.16 processzor esetén 32 bit széles adatátviteli csatorna mellett 8192.Vezérlése általában centralizált, ami sorrendiséget visz a párhuzamos Interconnect-be. Emiatt a vezérlés megtervezése nagyon kritikus.

Megvalósított típusok:

Típus	Processzorok száma	Teljesítmény	Crossbar sávszélesség
C.mmp	16		
Alliant/FX series	8	120 MIPS	376 Mbyte/sec
Convex C200	4		800 Mbyte-sec

3.3. MULTISTAGE INTERCONNECT NETWORK



Indirect binary n-cube vagy Banyan kialakítás (2x2 kapcsolóelemmel)

A kapcsoló-áramkörös Crossbar némileg egyszerűsített, elemszámot csökkentő változata. Sávszélessége arányos a processzorok számával, moduláris. Kommunikációs konfliktus kialakulhat annak eredményeként, hogy a kapcsoló-áramkörös Crossbar megoldáshoz képest a kapcsolóelemek számát csökkentették. Nagy számú processzor esetén az egyetlen lehetséges, fizikailag és technológiailag is megvalósítható megoldás.

Megvalósított típusok: (MIN = Multistage Interconnect Network)

CPU típus	Proc. szám	Telj. MIPS	Arch. Interc.	Shared Memory	Cache	CPU
IBM RP3 kísérleti típus	512	1000	MIN	Lokális	van	IBM ROMP
CEDAR	64		MIN	Hierarc.	van	
BBN Butterfly	256	128 600	MIN	Lokális	nincs	MC68000 MC68020

4. AZ INTERCONNECT TÍPUSÁNAK KIVÁLASZTÁSA

Az általunk megoldani kívánt feladatok számára legalább 60-80 MIPS teljesítmény szükséges. A népszerű MOTOROLA 68000 család MC 68030 típusú processzorát felhasználva ideális esetben 16 processzorral 80 MIPS csúcsteljesítményt érhetünk el. Ezért a továbbiakban a vizsgálódásainkat korlátozzuk 16 processzorelemre.
Ehhez keresve a leghatékonyabb Interconnectet (megvalósíthatóság, technológiai igények) azt mondhatjuk , hogy a Crossbar jellegű (kommunikációs konfliktus nincs), multiport memóriás, buszosított kialakítás rendelkezik a legjobb paraméterekkel.

Idézzük fel újra, hogy a teljesítménycsökkenésre milyen tényezők vannak hatással:

konfliktushelyzetek

kommunikációs	Crossbar jellegű architektúra esetén nincs ilyen konfliktus
memória	ütközés a memóriahasználaton

átviteli idő (fordulási idő)

5. MEMÓRIASZERVEZÉS

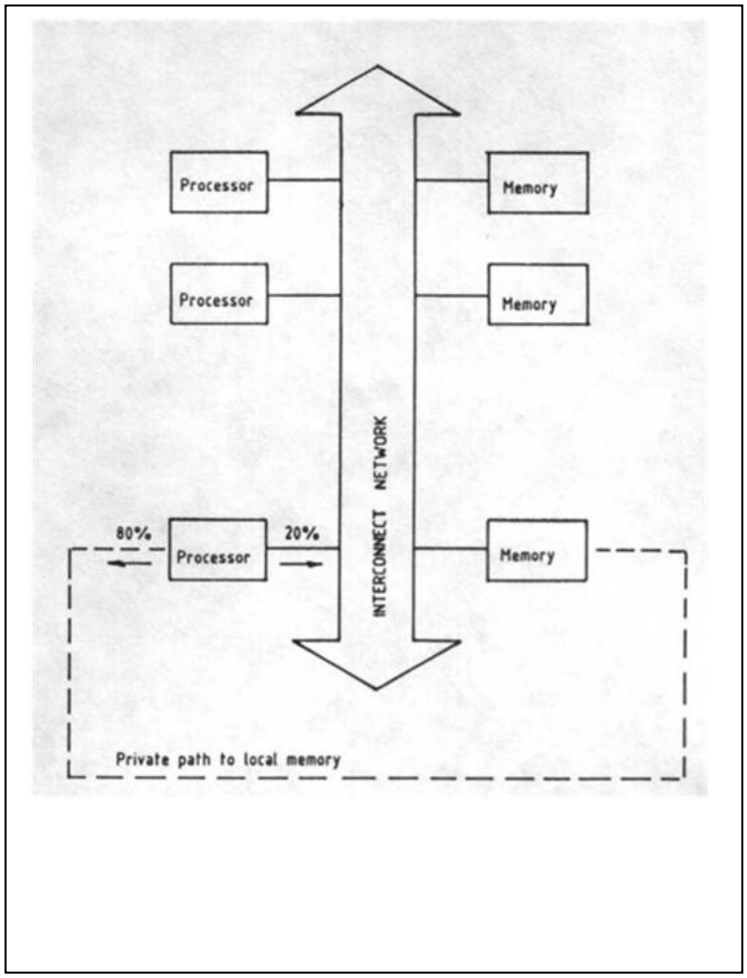
A memóriakonfliktusok száma hatékony memóriaszervezéssel jelentősen csökkenthető, mely lényegesen javítja a rendszer eredő teljesítményét. Multiprocesszoros rendszerekben memóriát két alapvető módon helyezhetünk el:

Globális	kizárólag az Interconnect-hez rendelt memória. Mindegyik processzor számára a CPU/Memória ciklusidő azonos.
Lokális	a teljes memóriát elosztjuk a processzorok között,azaz a processzorokhoz rendeljük. Megkülönböztetünk

privát memória	csak egy CPU éri el
osztott memória	mindegyik CPU számára elérhető memória, de az egyik CPU nem az Inter-connect-en keresztül, hanem egy gyors alternatív úton fér hozzá.

5.1. LOKÁLIS MEMÓRIA KIALAKITÁSA

A lokális/osztott memória alkalmazása a következőt jelenti:



Az Interconnect-en a processzor forgalmának csak egy része jelenik meg. A nagyobbik forgalom egy speciális, külön úton zajlik. (Analógiaként ld. VME BUS/VMSB/VMX)

Két módon készíthetünk lokális, (vagy lokális-szerű) memóriát:

distributed valódi lokális memória,mely a global memória egy része a processzorhoz rendelve

cache memória gyorsítótár,a lokalitást automatikusan biztosítja

5.2. LOKÁLIS MEMÓRIATÍPUSOK ÖSSZEHASONLÍTÁSA

Mindkét lokális memóriatípus a következő előnyökkel jár:

1. Csökken az Interconnect forgalma
   a. A kommunikációs konfliktus csökken, ezért ugyanazon az Interconnect-en több processzor helyezhető el.
   b. A memóriakonfliktus csökken és ez a rendszer hatékonyságát javítja.
2. Az Interconnect átviteli ideje , késleltetése a lokális hozzáféréseknél nem jelentkezik, így az átlagos átviteli idő csökken.

A két megoldás közti különbség:

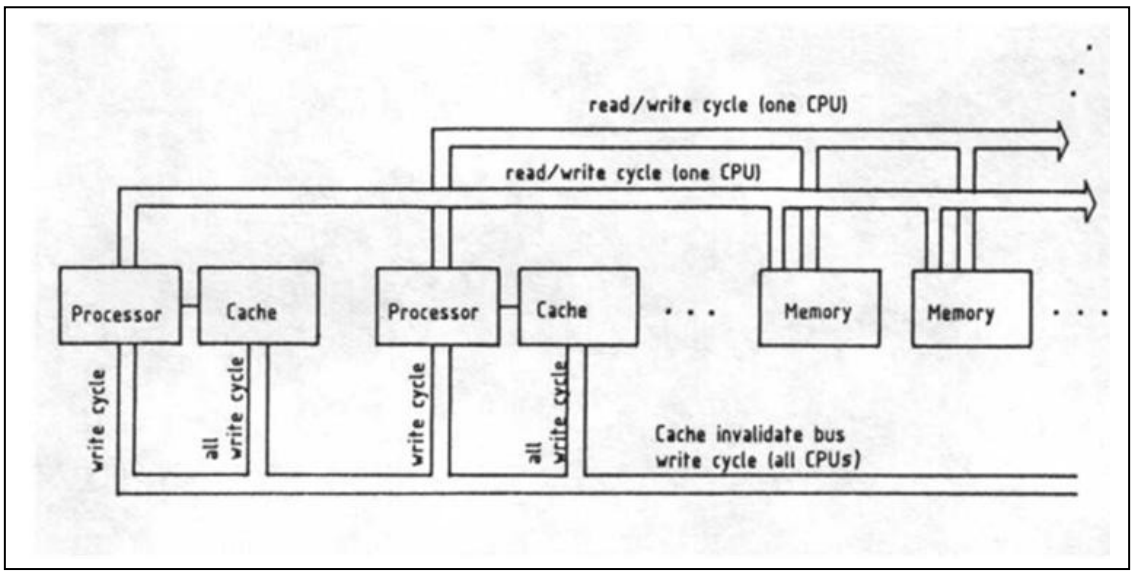
distributed	ebben az esetben a lokalitásról a software-nek kell gondoskodnia
cache	a lokalitás automatikusan teljesül (de végső soron ez is software függő)

Következtetés:

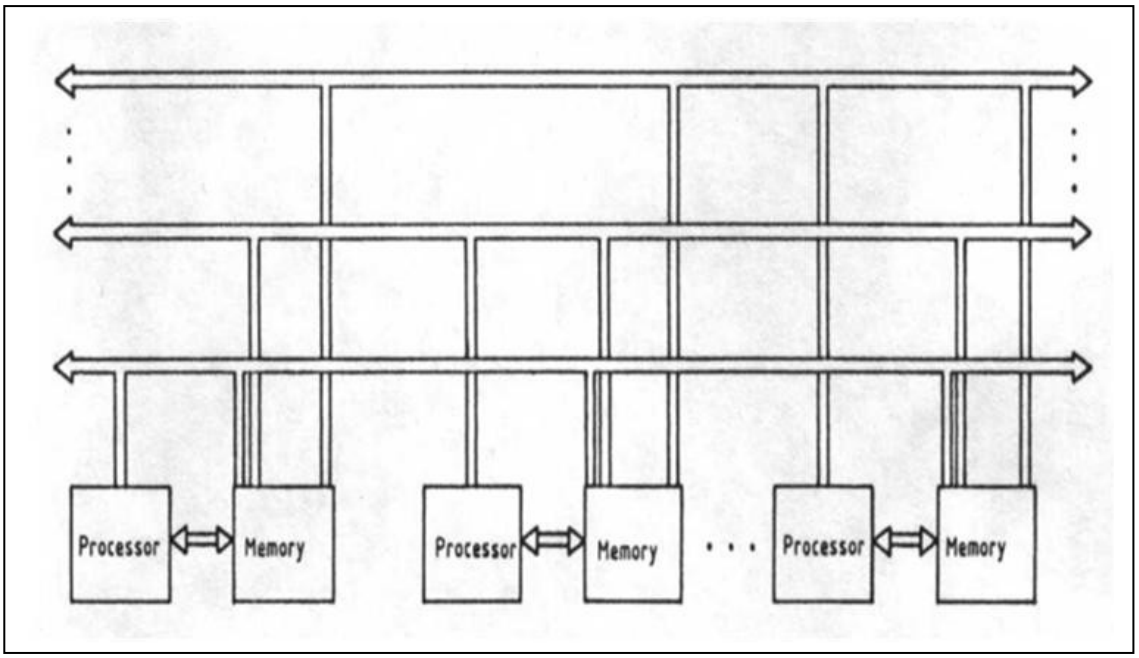
lokális, vagy lokális szerű memória kell, bármilyen Interconnect-nél, mert az javítja az architektúra jellemzőit (Interconnect átviteli idő, memóriakonfliktus)

A következőkben nézzük meg , hogy a konfliktusmentes multiport memóriás architektúra kialakítás mellett a rendszer paraméterei hogyan alakulnak a két memórakialakítás esetén.

5.2.1. CACHE MEMÓRIA



5.2.2. DISTRIBUTED MEMÓRIA



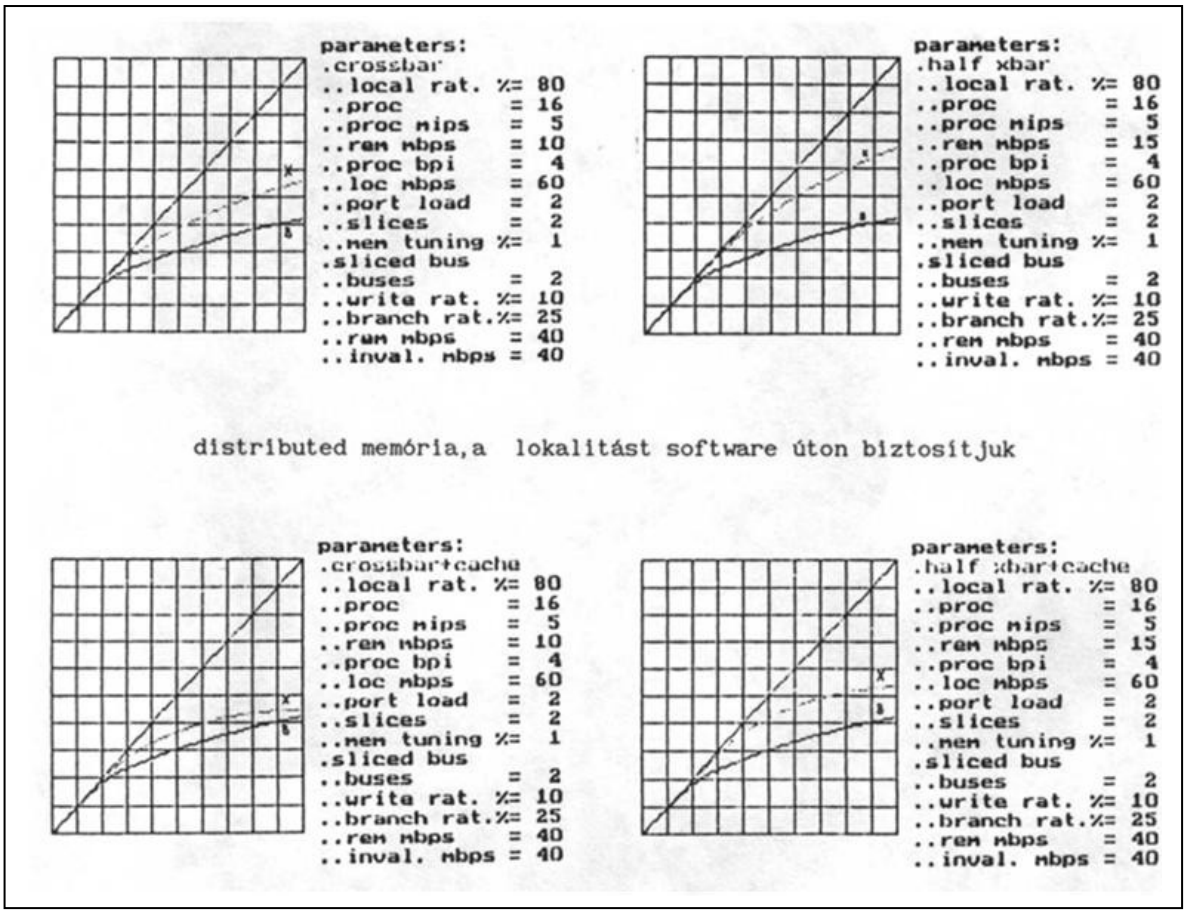
5.2.3. LOKÁLIS MEMÓRIATÍPUSOK ELEMZÉSE

Elemzéseinkben éljünk a következő feltételezéssel:

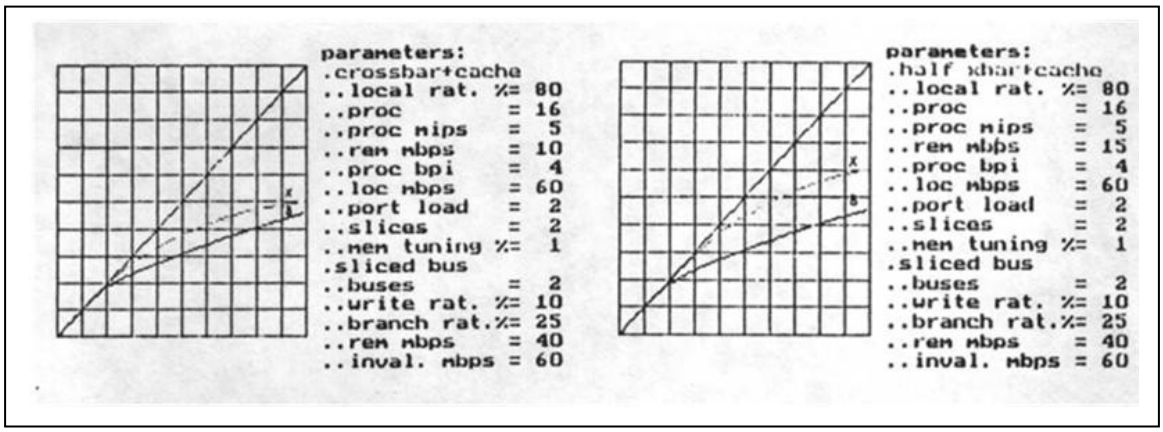
Software úton 80%-20% lokális-remote memória hozzáférési arányt tudunk biztosítani, a Cache memóriánk pedig legalább 90 %-os találati aránnyal működne.

Tekintsünk két Crossbar jellegű multiport memóriás architektúrát. (A Half-Crossbar magyarázatát ld. a 6.2. pont alatt)

CROSSBAR (X) HALF-CROSSBAR (X)



A lokalitást cache memória biztosítja. A cache konzisztenciáról a Cache Invalidate Bus gondoskodik. (inval_mbps 40 Mbyte/sec)



Az egyik szűk keresztmetszetet az invalidációs busz jelenti. Ha ennek sávszélességét (inval mbps) növeljük, akkor az architektúra paraméterei javulnak. (Ezen a gondon segítene a write-back, vagy copy-back cache

stratégia, de egykártyás processzorokat nézve, ezt a változatot el kell vetnünk igen magas áramköri igénye miatt.)
Tehát software úton biztosított lokál itassál hatékonyabb rendszert tudunk kialakítani,melynek okai mégegyszer:
Cache memória esetén az Invalidációs buszon a processzorok teljes (összegzett) írásciklusa megjelenik .Ez egy 80 MIPS csúcsteljesítményű 16 processzoros gép esetén kb. 50 Mbyte-sec forgalmat jelent (write-through cache esetén).A monitorozás a cache konzisztencia érdekében ilyen terhelés mellett
teljesítménycsökkenéshez vezet (Kommunikációs konfliktus az Invalidate busz használatán).
A cache memória nem tudja feloldani az adatstruktúrákon történő ütközésekből származó konfliktust. Ez azt jelenti,hogy a közös adatstruktúrát hiába tartalmazza a cache,mert azt a többi processzor bármikor módosíthatja (és azt invalidációs ciklus követi a cache-ben), és ez újra remote forgalmat eredményez.
Ha a cache stratégiában az invalidálás (melyből származó konfliktus a dual TAG cache-sel kiküszöbölhető) helyett a cache frissítést alkalmazzuk, akkor a cache data memórián alakul ki konfliktushelyzet. Tehát: biztosítsuk software úton a lokalitást distributed memóriás rendszerben.

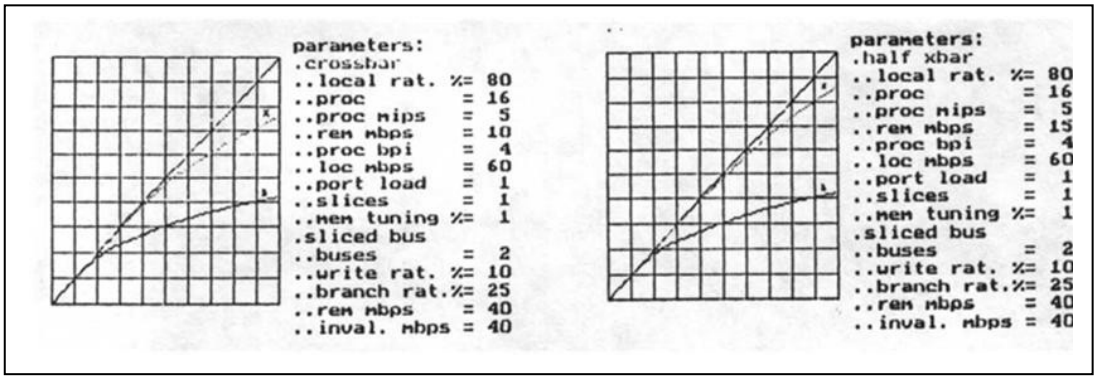
Folytassuk a választott distributed multiport memóriás rendszer elemzését: 16 portos memória , buszosított rendszerben.

6. A DISTRIBUTED MEMÓRIÁS MULTIPORT KIALAKÍTÁS ELEMZÉSE

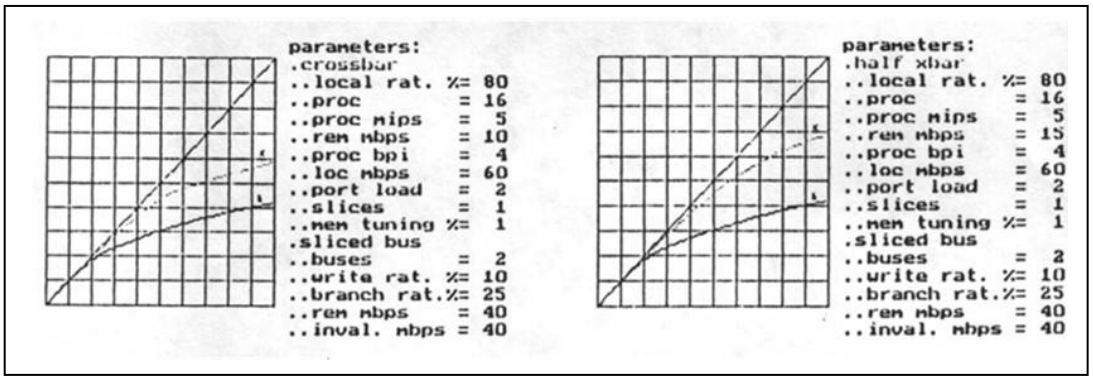
6.1. A MULTIPORT MEMÓRIA PORTSZÁMÁNAK CSÖKKENTÉSE

Próbáljunk egyszerűsíteni a 16 portos memórián úgy, hogy 8 portossá tesszük, és egy porton két processzor osztozik. Ez csökkenti az áramköri igényeket, illetve a tervezést egyszerűsíti.
(Gondoljunk a Common buszos "split transaction",vagy "pended bus" stratégiára: a csatorna a memória elérési ideje alatt kihasználható egy másik processzor számára.)

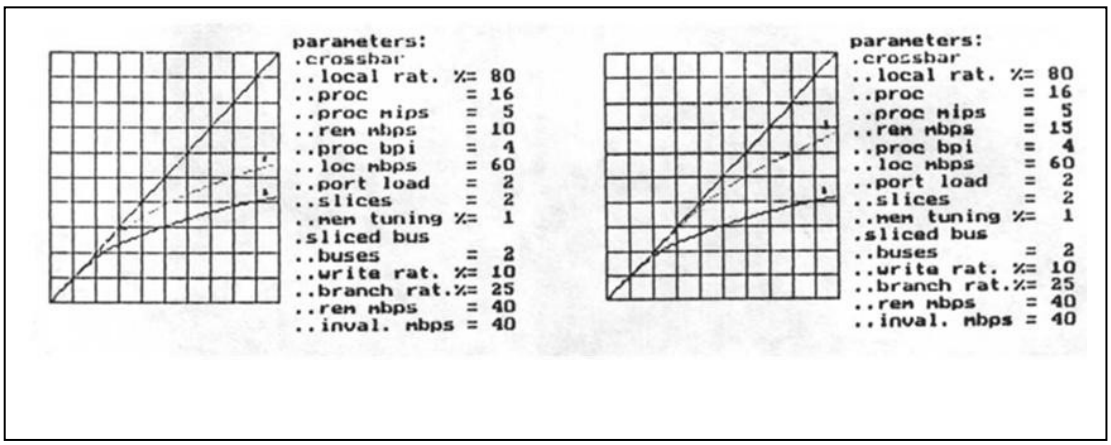
CROSSBAR HALF-CROSSBAR



egy csatornán egy processzor (port load = 1)



egy csatornán két processzor (port load = 2)



egy csatornán két processzor hasított ciklussal (slices = 2)

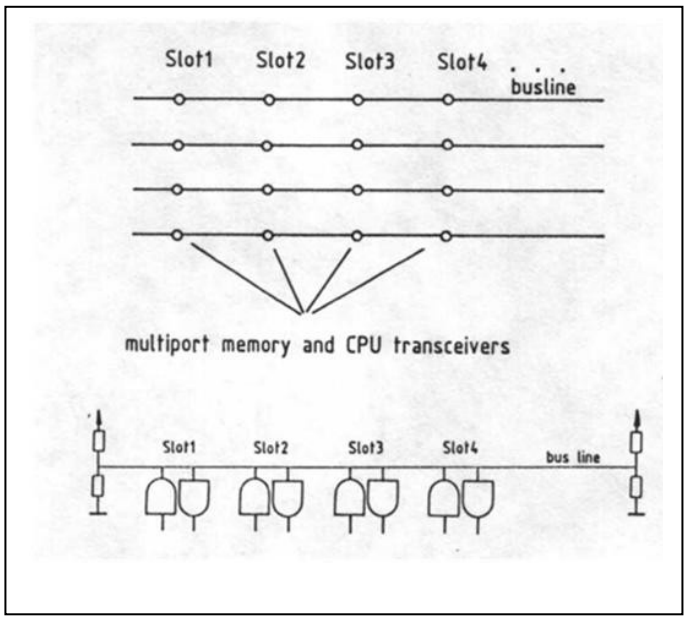
A csatornaszám csökkentése jól láthatóan befolyásolja a rendszer eredő teljesítményét. Az első fázisban , annak ellenére hogy a rendszer degradálódásához vezet,mégis 8 csatornás hálózatot célszerű építenünk, mert áramköri igénye jóval kisebb a 16 csatornásénál, a fele akkora méretű backplane elektromos jellemzői jobban kézben tarthatók. A második fázisban természetesen 16 csatornás backplane-t készítünk.

6.2. BUSZOSÍTÁS, VAGY PONT/PONT KAPCSOLAT

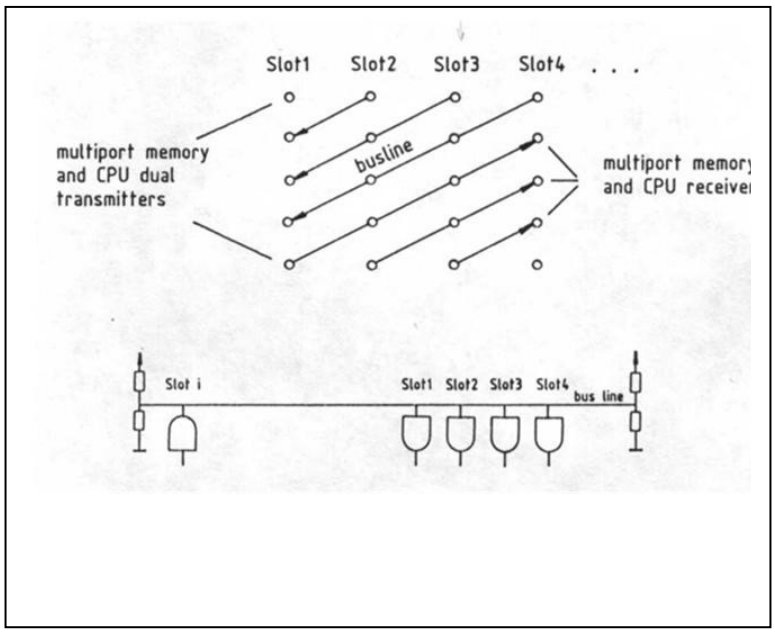
A buszosítás már egy egyszerűsítés eredménye volt, de azzal a következménnyel járt, hogy az egyedi processzor/memória (most már processzor/multiport) sávszélesség csökkent, azaz a csatorna késleltetési ideje megnövekedett.

A 8 portos memória és processzorelem backplane-jét alakítsuk ki új módon:

klasszikus busz kialakítás (több adó-áramkör egy vonalon)

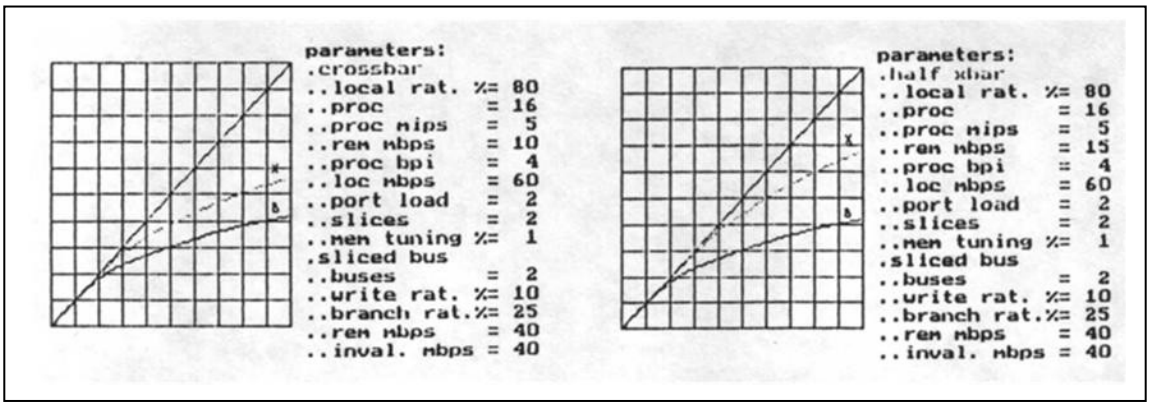


módosított kialakítás (egy adóáramkör egy vonalon)



Az új kialakítással pont/pont kapcsolatot tudunk megvalósítani, mely 30-40 %-kal jobb késleltetési idő elérését teszi lehetővé. A módosított kialakítás a hagyományos buszos elrendezésből úgy származtatható, hogy fizikailag külön írás- és olvasásciklusokat megvalósító egyirányú buszokat alkalmazunk. Forgalmi szempontból az adatátviteli csatornák így nem függetlenek teljesen egymástól, azaz egy processzor adatátviteli ciklusa és a vele egy kártyán elhelyezett memória olvasási ciklusa ütközhet egymással. Ezt a kialakítást nevezzük Half-Crossbar-nak .

CROSSBAR HALF-CROSSBAR



A Half-crossbar esetben 50%-kal nagyobb remote sávszélességgel modelleztünk.(rem mbps = 15 mbyte/sec)

A két egymás ellen ható tényező közül az átviteli idő csökkenése a nagyobb súlyú, azaz ez többet javít a rendszeren, mint amennyit ront a "half duplex " multiport csatorna forgalmi szempontból.

VEKTORPROCESSZOROK

Minden skalár processzorelemet kiegészítünk egy vektorprocesszorral. A csatolt vektorprocesszorokat a host számítógéphez való kapcsolódásuk szerint két csoportra oszthatjuk.

7.1. INTEGRÁLT VEKTORPROCESSZOROK

Az integrált vektorprocesszorok a host számítógép erőforrásait hatékonyan képesek használni. A feladatok megoldása során a host számítógéppel szorosan együtt dolgoznak. A speciális kapcsolódás miatt egy ilyen segédprocesszor csak egy adott típusú számítógéphez kapcsolható.

7.2. PERIFÉRIÁLIS VEKTORPROCESSZOROK

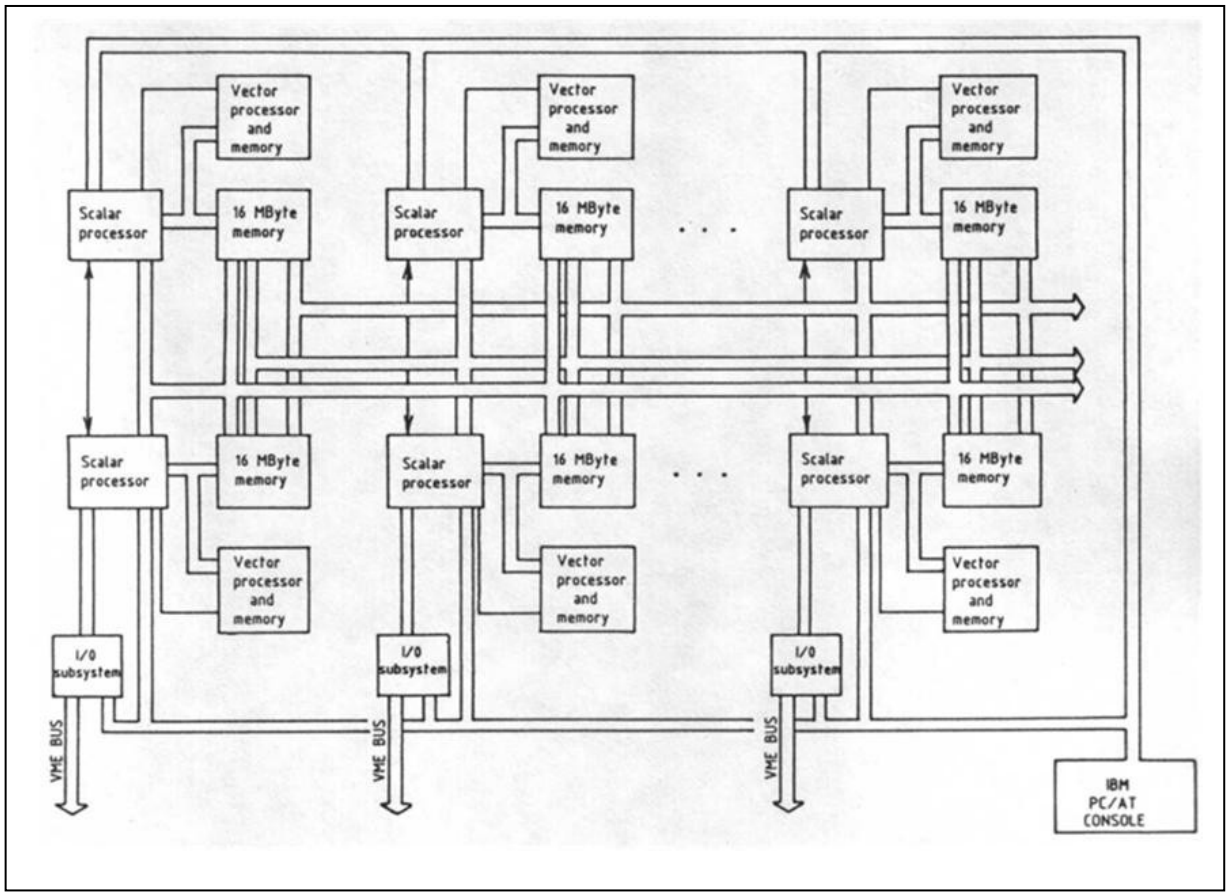
A perifériális vektorprocesszorok lazábban kapcsolódnak a host számítógéphez. Általában nagy méretű adatmemóriájuk van, nagy részük DMA-s perifériaként, kisebb részük közös memóriával kapcsolódik a host számítógéphez. A kapcsolódás helye általában nem speciális, egy-egy vektorprocesszor különböző gyártótól származó számítógép standard interface buszára kapcsolható. A host számítógép perifériaként kezeli a segédprocesszort, feltölti adatokkal, esetleg betölti a programot, elindítja a műveletet, majd ha a segédprocesszor kész, átveszi az eredményt. Ez meglehetősen sok adminisztrációs művelet elvégzését igényli. Ez használatukat korlátozó tényező, ugyanis a perifériás processzor kezeléséhez szükséges rendszerhívások által igénybe vett viszonylag hosszú idő olyan algoritmusokra szűkíti be az alkalmazásuk területét , melyeknél az egy rendszerhívásra eső aritmetikai műveletek száma igen nagy.

7.3. PROGRAMOZÁSUK

A programozó a vektorprocesszor típusától függően kétféleképpen kezelheti azt. Egyes gyártók szubrutincsomagokkal árulják vektorprocesszoraikat. A felhasználó ebben az esetben saját maga döntheti el, hogy mely feladatok kerüljenek a vektorprocesszorhoz. Az átadni kívánt feladatokhoz kiválasztja a megfelelő szubrutinhívást, és beilleszti a programjába. Ennek a megoldásnak két fő hátránya van. Az első, hogy a programozó többnyire nem tudja optimálisan szétosztani a feladatokat, ezáltal a vektorprocesszor nyújtotta lehetőségek részben kihasználatlanok maradnak. A másik, hogy a korábban kifejlesztett, vagy vásárolt programok nem képesek kezelni a vektorprocesszort. Az igazi megoldás a párhuzamosságok automatikus feltárása optimalizáló fordítóprogram segítségével. Ilyen módon a már meglévő programok egyszerű újrafordítással hatékonyan képesek használni a vektorprocesszort.

Tekintettel arra, hogy egy optimalizáló fordító megírása jelentős időt és energiát igényel, a fejlesztés első fázisában egy piacon kapható, standard interface-re kapcsolódó, optimalizáló fordítóval rendelkező perifériás vektorprocesszort alkalmazunk.

8. A VÁLASZTOTT KIALAKÍTÁS BLOKKVÁZLATA

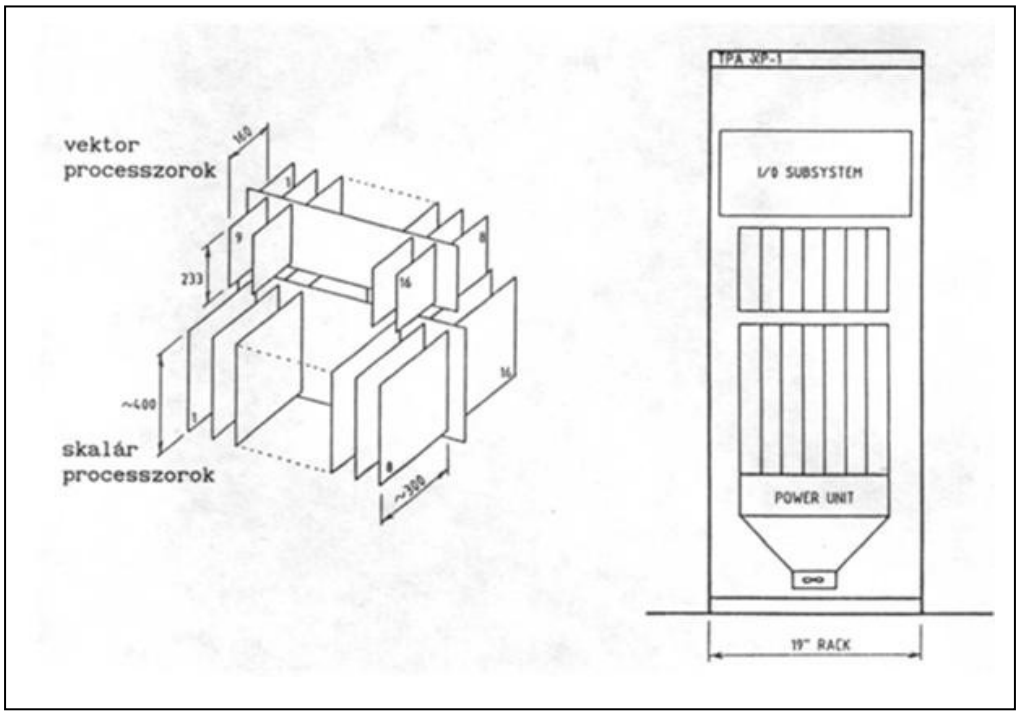


A rendszer 16 skalár és 16 vektorprocesszort tartalmaz. A memória elhelyezése distributed, a memóriaegységek 8 portosak, egy porton két processzor osztozik. Minden skalárprocesszorhoz egy perifériális vektorprocesszor tartozik. A konfiguráció VME buszra illeszkedő perifériavezérlőkkel bővíthető.

9. SPECIFIKÁCIÓ

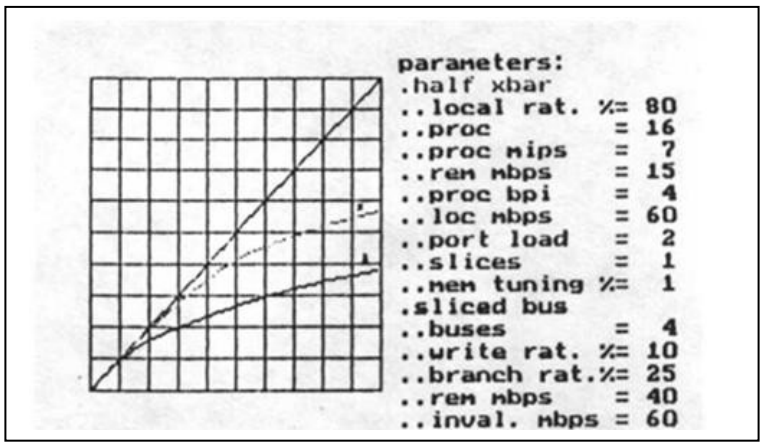
Skalár processzor	Processzorszám	max. 16
	CPU típusa	Motorola MC68030
	CPU ciklusidő:	40ns
	Teljesítmény:	5-80 Mips (csúcs) 5-64 Mips (effektív) 1.5-24 Mflops(csúcs) 1.5-21 Mflops (eff.)
	Hardver FPP:	Weitek WTL3168
	Virtuális memória:	4 Gbyte
	Battery backup:	van
Vektorprocesszor	processzorszám	max. 16
	Teljesítmény	20 Mflops (vektor)
Operatív memória	Ciklusidő	l00 ns (lokális) 350ns (remote)
	Védelem	Paritás bitek
	Bővíthetőség	16 Mbyte-onként
	Max. méret:	256 Mbyte
Input/Output control	Busz típusa:	VME bus
	1/0 csat. száma:	max. 8
	Sávszélesség	40Mbyte/sec/csatorna
Software	Operációs rendszer	UNIX
	Op. rendszer típusa	Multitasking/multiuser (Single copy)

10. MECHANIKAI KONSTRUKCIÓ

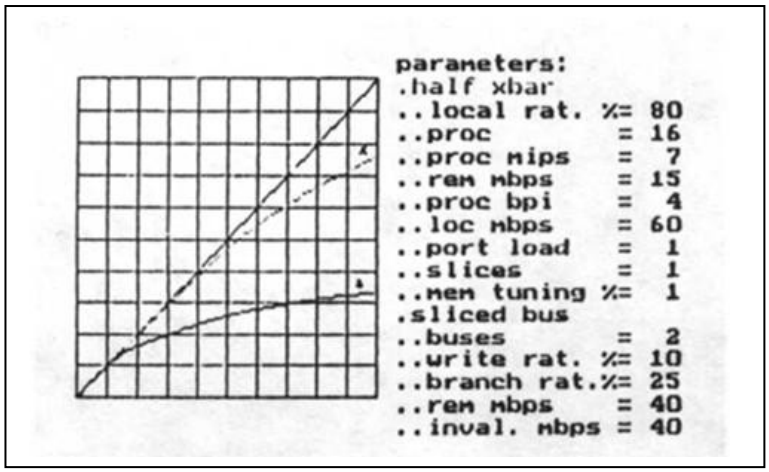


11. A FEJLESZTÉS MÁSODIK FÁZISA,A TOVÁBBLÉPÉS LEHETŐSÉGEI, IRÁNYA

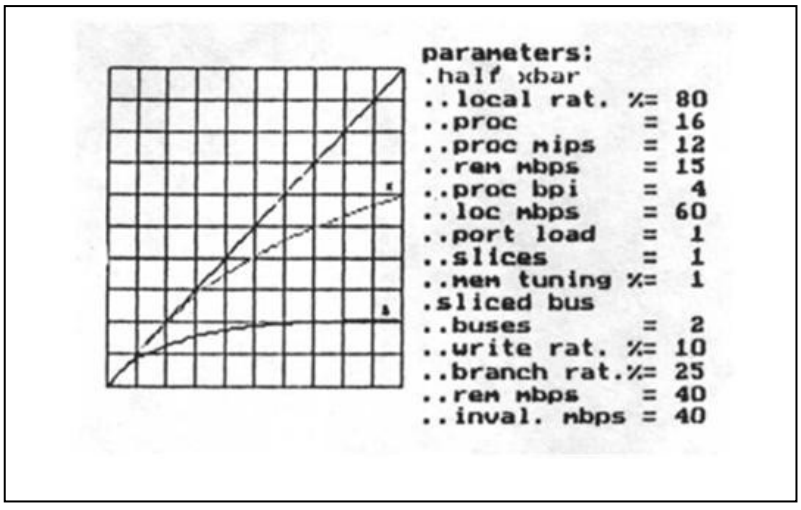
Végezetül feltétlen beszélni kell a továbblépés lehetőségeiről. Egyik lépés, hogy a MOTOROLA 68030 25MHz-es változata helyett a 33 MHz-eset használjuk.



Annak ellenére, hogy az egyedi processzorteljesítmény így 40%-kal nő, a rendszer teljesítménye csak 20 %-kal növekszik, mert szűk keresztmetszetet hagytunk az architektúrában. Át kell térni a 16 csatornás Interconnectre (port load=1). Ez már nem valósítható meg standard áramköri elemekkel, céláramkörök szükségesek.



Ezt a kialakítást már elfogadható degradáció mellett elviseli egy kb. 12 MIPS teljesítményű processzor is (pl. MOTOROLA 68040 25 MHz).



Az első fázisban létrehozott példány tapasztalatai birtokában tűzzük ki célul ennek a Crossbar jellegű architektúrának a továbbfejlesztését.

BUDAPEST 1989.október

1. számú melléklet

Többprocesszoros architektúrák forgalmi elemzése

Tartalomjegyzék

1. A forgalmi modell képességei

1.1. Figyelembe vehető architektúrák

Célunk a multiprocesszoros rendszerek olyan leírása , amelyben figyelembe vehető bizonyos tervezési döntések hatása a rendszer teljesítőképességére. A későbbi pontokban részletezett alapokra építve alakult ki egy program, amelynek kezdeti változatai a rendszer input/output jellemzésére voltak alkalmasak, és a fő paraméter a processzorszám volt. A program egy újabb változata input-input jellegű szűk keresztmetszetet vizsgál, fő paramétere az egyedi processzor teljesítménye.

Az újabb verzió már nem foglalkozik a Crossbar hálózatok egyes változataival, így például a Multistage hálózatokkal, helyette a hierarchikus buszrendszerekhez közeli Crossbar kialakításokat elemzi. A több processzorral terhelt Crossbar csatornán /porton/ kívül alkalmas még egy fizikailag egyszerűsített Crossbar interface, a 6.2. pontban részletezett "módosított buszkialakítás", figyelembevételére is.

Szintén az újabb verziójú program foglalkozik a Multiplikált buszos megoldásoknak valósághűbb leírásával. Multiplikált busz alatt eredetileg a modellezésben egy, a Crossbar-ral rokon megoldást értünk, amely azonban lényegesen kevesebb buszt tartalmaz, mint a Crossbar. A modell alkalmas időosztásos buszoknak több független busszal történő leírására. Ennek a cache-sel ellátott változatát elemeztük, figyelembe véve a memória interleave-elési lehetőséget is.

1.2. Paraméterezési megoldások Alapvető követelmény a modellezés eredményével szemben, hogy alkalmas legyen különböző megoldási változatok közötti összehasonlításra. Ennek következtében kerültek a Crossbar-ra és a Multiplikált buszra jellemző görbék egy koordinátarendszerbe, és ilyen módon elég sok paraméter együttes figyelembevételére nyílik lehetőség. A koordinátarendszer vízszintes tengelyén tüntettük fel egy processzor névleges teljesítményét, a függőleges tengely pedig a valóságos, az architektúrái is környezet hatására lecsökkent teljesítményt méri. Ideális esetben a környezet hatása elhanyagolható, az eredményül kapott mips-degradációs görbe egy 45°-os egyeneshez közelít.

A paramétereket részletesen később ismertetjük, most csak fő csoportjait emeljük ki:

processzorfüggő

buszfüggő

memóriafüggő

feladatfüggő

paraméterek segítik egy-egy megoldási változat körülírását. Külön ki- emelendő, hogy számos olyan paramétert is elfogad a program, ami nem, vagy nehezen megvalósítható adott körülmények között. Jó példa erre a cache invalidációs busz sebessége, amelynek igen nagy értékeinél például egy kettős tag-es cache-hez közeli forgalmi helyzetet modellezünk, habár nem annak pontos szimulációja a cél, hanem a forgalom érzékenységének kimutatása a paraméterre vonatkozóan.

2. A modellezés elvi alapjai

2.1. A publikált throughput modellek

Egyes modellek a lokális memória nélküli crossbar hálózatot írták le egy igen egyszerű összefüggés segítségével. Azt kellett meghatározni, mekkora a valószínűsége (ql) annak, hogy legalább egy processzor foglalkozik egy tetszőlegesen kiválasztott memóriával, amikor adott a processzorok forgalom kibocsátó képessége egy valószínűség-értékkel (qO). Ez az érték úgy adódik,hogy a processzor Mbyte/sec-ben megadott teljesítményét arányítjuk az Interconnect sebességéhez. Az így kapott 1-nél kisebb-egyenlő értéket forgalmazási valószínűségnek tekintjük. Ha az érték 1-nél nagyobb, akkor a processzor-oldalon egy közvetlen mips-degradációs faktor is jelentkezik. Ha a processzorok és a memóriák száma azonosan

m , és független valószínűségi változóknak tekintjük az egyes processzorok forgalmait, a következő összefüggés igaz:

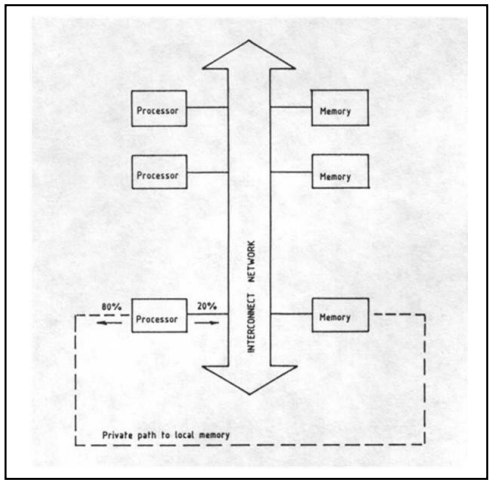
$$ql = 1 - \prod_{i=1}^{m} (1-qi) = 1-(1-qO/m)^m \qquad (1)$$

ahol: q0: az azonosnak tekintett processzorok forgalma

qi: a különbözőeknek vett processzorok forgalma a kiválasztott memóriára vonatkoztatva m: a processzorok és a memóriák száma ql: a memóriaoldali forgalom, vagy throughput

A későbbiekre is vonatkozik, hogy a throughput abszolút értéke ql-nek felel meg, de gyakran érdekes a relatív érték is (ql/qO). Erre nézve viszont általánosabb definíció adható Multiplikált buszos rendszerekben: a kimeneti és a bemeneti várható érték aránya.

Igényesebb elemzések figyelembe veszik a lokális , vagy lokális-szerű memória létezését, és forgalommegosztó hatását is. Egy ilyen elrendezést mutatunk be az 1.ábrán.



1.ábra

Többnyire a processzorteljesítmény mips-ekben van megadva, ennek aránya a lokális, illetve az Interconnect busz sebességéhez tulajdonképpen kétféle processzorforgalmat definiál, és mindkettő szerepel is a throughput modell javított változatában:

(2

$$ql = l-(l-r*qloc)*(l-(l-r)*qrem/(m-D)$$

ahol: qloc: az input forgalom cache felé

qrem: az input forgalom remote irányban

ql: az eredő throughput

r: a lokális forgalom aránya az egészhez

m: a processzorok száma

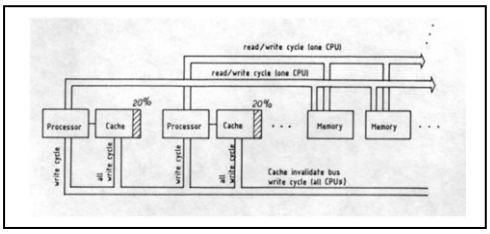
Ebből a képletből két irányba szokásos továbblépni: az egyik a már említett Multiple buszos throughput definíció, a másik a Multistage architektúrák viselkedése.

A Multiple busszal kapcsolatban megemlítjük, hogy a várható érték számítása során alkalmazott Bernoul1i-eloszlást a buszok számának megfelelően csonkítani kell.

A több crossbar kapcsolót tartalmazó Multistage architektúrák vizsgálatánál a rekurzív módszer terjedt el. Ez azt jelenti,hogy minden egyes 'stage'-re kiszámítjuk a throughput-ot az előző kimenet adatát bemenetnek használva.

2.2. Multiplikált buszos modellek kiegészítése

Az ilyen típusú valóságos architektúrák rendelkeznek cache-sel, és ez a fajta lokális memória-megoldás többnyire jó választást jelent ezekben a rendszerekben. Jelentős méretű cache alkalmazása esetén az egyes feladatok "jól belefférnek " a cache-be, ennek ellenére bármilyen nagy méretű cache-t választunk is, a programok lokalitását ez sem tudja 100 % -osan biztosítani, mert a multiprocesszoros környezetben a közös adatstruktúrát bármelyik processzor megváltoztathatja. A model1jeinkben ez adja a 20%-os remote forgalmat.(1.ábra)
A következő ábrán a satírozott cache terület felel meg a remote arányának (közös adatstruktúrák).



2.ábra

Az ábrán látható cache-invalidációs buszon létrejövő forgalmi torlódás becsléséhez a throughput modellt használva az (1) alapképletünk egy újabb változatát kaphatjuk:

$$ql = l-(l-h*qloc)(l-w*qrem)^{m-1} \qquad (3)$$

ahol: qloc: input forgalom cache felé

qrem: input forgalom remote irányban

w: processzorok irási aránya

h: cache találati arány

m: processzorok száma

ql: eredő (invalidációs) throughput

Ez lehetőséget ad, hogy a cache hatását figyelembevegyük. A cache találati arányt nem tekinthetjük 100%-osnak. mert a feladatok hiába férnek bele a cache-be, olvasáskor annál nagyobb valószínűséggel fogunk /a cache invalidate ciklus által/ átírt adatot olvasni, minél nagyobb a remote adatok aránya. Ezért (a remote forgalmon keresztül) a hit-rate csökkenni fog növekvő processzorszám esetén. Megjegyezzük azonban, hogy kisebb hit-rate esetén kisebb az invalidációs konfliktus is. A hit-rate csökkenését a következő becsléssel vehetjük figyelembe:

h = w+(l-w)*(l-w*(l-r))
m-l

(4)

ahol:
r: a lokális forgalom aránya az egészhez

w: a processzorok írási aránya

h: a cache találati arány

m: a processzorok száma

A Crossbar hálózatban is elképzelhető cache alkalmazása, annak invalidációs problémája nagyon hasonló, a Multiplikált busznál alkalmazottéhoz. A fenti képletek kis módosítással ott is érvényesek.

2.3.
Crossbar interface megvalósítása

Fontos megvalósítási kérdés a kombinált processzor-memóriakártyák Crossbar interface-ének célszerű kialakítása. Lehetőség van bizonyos kommunikációs konfliktust eredményező olyan egyszerűsítésre, amely viszont a backplane fizikai kialakítását egyszerűsíti, és rövidebb fordulási időt eredményez.

A megoldás lényege - forgalmi szempontból- olyan taggal csökkenteni a (2) képletet, amely kifejezi annak a helyzetnek a gyakoriságát, amikor a processzor ír egy remote memóriába, a lokális memóriáját pedig közben kívülről olvassák. (Ez egyfajta félduplex működést eredményez.)

ql = l-(l-r*qloc)*(l-(l-r)*qrem/(m-l))
m-l

-(l-(l-w)*(l-r)*qrem/(m-D)

m-l

*r*qloc*w*(m-l)/m

(5)

ahol:
qloc: az input forgalom cache felé

qrem: az input forgalom remote irányban

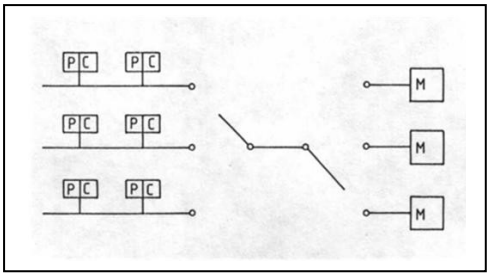
ql: az eredő throughput

r: a lokális forgalom aránya az egészhez w: a processzorok írási aránya

m: a processzorok száma

2.4. Egy throughput modell alkalmazási példa

Multiplikált buszos rendszerek gyakran tartalmaznak interleave-elt memóriákat. Az úgynevezett hasított ciklusú buszok /split transaction/ egy lehetséges felfogásában a processzorokat annyi csoportba osztjuk, ahány időrés van,és így csak az egyes csoportokon belüli processzorok forgalma ütközik intenzíven. Az interleave-elt memóriákon történő ütközés azonban csökkentett intenzitással jelentkezik, hiszen amíg az időréseket uraló processzorok folyamatosan növekvő címeken dolgoznak, addig nincs memória-konfliktus. A következő ábrán ennek a vizsgált rendszernek a vázlatát láthatjuk.



Az ábrán a forgó kapcsoló Jelképezi az időrések és memóriák folyamatosan lépkedő egymáshoz rendelését. A másik kapcsolón a hozzárendelésben jelentkező ugrások hoznak létre kapcsolási helyzetet,amikor egy processzor nem címfolytonos igénnyel jelentkezik, és ezzel sorbanállási szituációba kerül.
A throughputban is megjelenik ennek megfelelően egy 'ugrási arány', amely jelentősen csökkenti az említett sorbanállási helyzet gyakoriságát. Pontosabban:

$$ql = l-mt*(l-r*qloc)*(l-br*(l-r)*qrem/(m-l))^{m-l} \qquad (6)$$

ahol: qloc: az input forgalom a cache felé

qrem: az input forgalom remote irányban

ql: az eredő throughput r: a local/remote

arány

m: a processzorok száma

br: az elágazási utasítások aránya (branch ratio/

mt: 'memory tuning' tényező

Láthatólag feltüntettük a képletben egy más működésű memória tökéletesítési módszer hatását is. Ez olyan eljárások hatását modellezi, amelyek közvetlenül csökkentik a memórián az ütközést, például egy ügyes bufferelési eljárás segítségével.

Természetesen egy konkrét eljárás hatását nehéz leírni egyetlen paraméterértékkel, de mégis lehetőség van eldönteni legalább azt, hogy van-e érdemleges hatása a módosított megoldásnak.

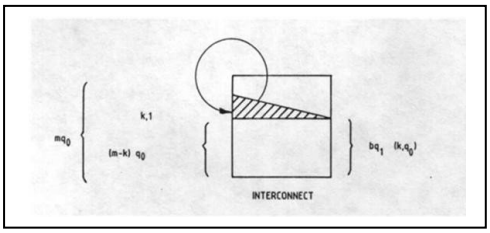
3. Processzorközpontú szemlélet érvényesítése

3.1. A feltételes throughput bevezetése

Egy sokprocesszoros rendszer Jellemzéséhez kevésnek látszik pusztán a throughput megadása. Számos irodalmi hivatkozással élhetnénk a rendszerben jelenlévő üzenetek kiszolgálásig való várakozási idejére vonatkozó becsléseket illetően. Ezek gyakran hivatkoznak a Pollaczek-Hincsin formulára, mint a sorbanállási modellek egyik alapvető összefüggésére. Sajnos, nehéz összevethető eredményeket találni, igaz eredmények főleg csomagkapcsolt rádiórendszerekre kidolgozott modellek adaptációiból adódtak.

Az általunk követett felfogásban a rendszerbeli fordulási időt adottnak tekintjük a busz és a memória együttes késleltetési idejéből adódóan. Növekvő buszforgalom esetén a processzorok számára ez a fordulási idő megnövekszik, ami visszafogja a processzorok forgalmát. Ez az átlagos kényszerű forgalomcsökkenés ('mips-degradáció') már többet mond számunkra a throughputnál, hiszen minket nem igazán a memóriák forgalmi terhelése érdekel, sokkal inkább egy adott architektúrába helyezett processzor kényszerű állási ideje. Ez a rendszer egy input/input jellegű jellemzése, szemben a throughput input-output jellegével.

A következő ábra segít megérteni azt a stacionárius állapotot, amikor a processzorigények egy része 'visszaverődik', ezzel egy kicsit megnövelve a forgalmat, és az így keletkezett throughput csökkenés az új bemeneti igények kiszolgálására éppen elégséges throughputot eredményez.



4.ábra

A visszaverődött forgalom a közvetlen oka a mips degradációnak, amely a 7. képletből számítható. Ezt szinte mindig csak iteratív úton lehet megoldani.

$$m*(1-k/m)*qO = b*ql(k) \qquad (7)$$

ahol: 1-k/m: a mips-degradáció

m: a processzorok száma

k: a blokkolódott proc. száma

qO: az input forgalom

ql(k): a feltételes throughput

b: az interconnect buszainak száma

Láthatóan az eredő throughput függvénye a blokkolódott processzorok számának, ezeket ugyanis egy állandó teljes forgalom jellemzi. Ez meg is jelenik a feltételes throughput képletében.

ql(k) = (k/m)*(1-(1-r)*(1-qrem*(1-r)/(m-1))
m-k

***(1-(1-r)/(m-D)**
k-1

) +

+ (1-k/m)*(1-(1-r*qloc)*(1-qrem*(1-r)/(m-D)
m-k-1
»(1-(1-r)/(m-1))^{k})

ahol:
qloc: az input forgalom cache felé

qrem: az input forgalom remote irányban

ql(k): a feltételes thruput

r: a local/remote arány

m: a processzorok száma

k: a blokkolódott proc. száma

A képletek alkalmazásával kapott eredmények telítéses görbéket adnak növekvő input forgalom mellett, mind Crossbar, mind Multiplikált buszos rendszerekre.

3.2.
Hierarchikus buszok jellemzése

Többszintű buszrendszereknél az alacsonyabb szintű buszon és a magasabb szintű buszon egyaránt történik mips-degradáció, és a kettő nem független egymástól. Több processzor egyszerű felhelyezése egy buszra közvetlenül forgalomnövelő hatású, időosztásos megoldásban pedig a fordulási idő növekedésével jár. Mivel nem lehet interleave-elt memóriát alkalmazni, jelentős memóriakonfliktusra kell számítani.

Hierarchikus busz alatt többnyire olyan Crossbar hálózatot értünk, ahol a Crossbar-interface-re több processzort csatlakoztatunk, / d e a 2.4. pontbeli példa is ilyen természetű ./ Tulajdonképpen throughput szempontból a hierarchikus buszok a Multistage hálózatokkal rokoníthatók, hiszen a throughput Itt is rekurzív módon számítható.

4. A modellező program paraméterei

A program néhány általános jellegű és könnyen értelmezhető paramétert is tartalmaz, amely mind a Crossbar, mind a Multiplikált buszos rendszerben alapvető:

	jelölése:
− processzorok száma	proc
− processzorok mips-értéke	proc mips
− proc. utasításszélessége (byte/inst)	proc bpi
− busz sebességek (Mbyte/sec)	. . . mbps

Ez utóbbi annyi magyarázatot igényel, hogy az mbps érték tartalmazza a teljes fordulási időt, azaz a memória válaszidejét is. A következők állhatnak a ... helyén:

Crossbar esetben:	rem mbps
	loc mbps
multiplikált esetben:	rem mbps
	inv mbps

A vizsgált architektúrák ismeretében nem meglepő a local és a remote busz sebességének megkülönböztetése, azonban az utolsó (inv mbps) kivételt képez, ez ugyanis a cache invalidációkat tartalmazó busz sebessége. Ilyen módon ezt az adatot Crossbar esetben is használjuk, csak ott kisebb a jelentősége.

További processzor-busz viszonyokra jellemző adatok:

	jelölése:
- buszok száma	buses
- crossbar portonkénti processzorok száma	port load
- crossbarport időbeli megosztása	slices

Mielőtt az utolsó paramétercsoportra rátérnénk, amely %-os adatokat tartalmaz, érdemes a bináris választás jellegűekről beszélni. Így választhatjuk ki a Crossbar architektúrán belül a 'half xbar' változatot cache-sel vagy anélkül. A multiplikált buszos megoldásban kiválasztható egy 'async bus' amellyel például egy VME buszt is vizsgálhatunk, vagy egy 'sliced bus' amely hasított ciklusú buszt választ ki/ pended bus , vagy split transaction /.

Lássuk az utolsó paramétercsoportot:

	jelölése:
- local/remote arány	local rat.
- proc. írási arány	write rat.
- proc. ugrási arány	branch rat.
- memória ütközéscsökkentés	mem tuning

A local/remote arány mind a crossbar, mind a multiplikált buszos megoldásban fellép mint a helyi es távoli processzorokhoz tartozó memóriákhoz fordulás aránya.

A proc. írási arány Multiplikált busznál a cache vizsgálatánál, Crossbar esetben a remote cache-nél és a Crossbar interface-nél játszik szerepet.

Multiplikált buszos megoldásnál az interleave-elt memóriákon jelentkező ütközés becsléséhez szükséges paraméter az ugrási arány.

Mindkét rendszernél érdekes az utolsó figyelembe vehető paraméter, amely ütközést csökkentő egyéb memóriakialakítások hatásának becslését teszi lehetővé.

IRODALOMJEGYZÉK

1. Yih-Chyun Jenq: Performance Analysis of a packet switch. . . IEEE,1983.sac.l.No.6.
2. Antoni Zabludowski:Analysis of a multimicroprocessor system with time-shared bus Microprocessing and microprogramming, 1984.No.13.
3. Kang G. Shin.A cost-effective multistage interconnection. .. IEEE,1985.C-34 No.12.
4. Akira Fukuda:Equi1ibrium point analysis of memory interference.. IEEE,1988.C-37 No.5.
5. Chita R. Das: Bandwidth availability of multiple bus multiproc.... IEEE,1985.C-34 No.10.
6. Laxmi N. Bhuyan: Design and performance of generalized interconn... IEEE,1983.C-32 No.12.
7. Chin-Tau A.Lea:The load-sharing Banyán network IEEE,1986.C-35 No.12.
8. Michel Dubois:Throughput analysis of cache-based multiprocessors with multiple buses IEEE,1988.C-37 No.1.
9. M.Ajmone Marsán:Memory interference models... Microprocessing and microprogramming 1981.No.7.
10. Trevor N. Mudge:A semi-Markov model for the performance ... IEEE,1985.C-34 No.10.
11. Clyde P.Kruskal:The performance of multistage interconnection networks for multiprocessors IEEE,1983.C-32 No.12.
12. Kai Hwang: An orthogonal multiprocessor fot parallel scientific computations IEEE,1989.C-38 No.1.
13. Kyungsook Y. Lee:The PM22I interconnection network IEEE,1989.C-38 No. 2.
14. Per Stenström:Reducing contention in shared-memory multiproc. IEEE Computer, 1988
15. Laxmi N.Bhuyan: Performance of multiprocessor interconnection

networks IEEE

Computer, 1989.

16. Philip C.Treleaven: Parallel architecture overview Parallel Computing,1988.No.8.

17. Michel Dubois: Synchronization.coherence, and event ordering in

mult i processors IEEE Computer,

1988.

18. Ulrich Herzog:Performance evaluation principles for vector- and

multiprocessor systems

19. Forró Tibor: Nagy aritmetikai teljesítményű számítástechnikai

eszközök KFKI

tanulmány, 1989.

20. Edward F.Gehringer:A survey of commercial parallel processors