Box 33
folder 22

Alpha architecture, 1992-1993

Digital Equipment Corporation records
Engineers' papers: Mike Uhler papers: Alpha and PRISM devel
X2675.2004, *Box 33*

(Message inbox:281)
Replied: Mon, 15 Mar 93 15:16:32 -0500
Replied: human::supnik
Return-Path: ad::rock::human::supnik
Received: by muhthr.cdad.hlo.dec.com (5.57/ULTRIX-fma-041391);
        id AA25854; Mon, 15 Mar 93 15:08:01 -0500
Date: Mon, 15 Mar 93 15:08:01 -0500
From: ad::rock::human::supnik (BXB1-2/C4, 293-5690, SECRETARY: 293-5691 15-Mar-19
To: @dis$.cdad.hlo.dec.com:alpha_arch
Subject: Next Alpha architecture meeting 26-mar-93

The next meeting of the Alpha architecture working team is scheduled for March 26, 1993. I hope that we will have reports on each of the areas to be investigated; those for which progress has not been made should probably be closed. As I understand the situation:

1. Console. Work in progress to significantly reduce implementation size. Still requiring work: layering of PAL, HAL, console, etc; separate or merged SRM vs ARC consoles.

2. Policed function fields. Nothing to report.

3. IEEE handling. Nothing to report.

4. 128b floating point. Two format proposals (Sites, Supnik) made, awaiting written input from EV-6 team to close.

5. Formal bi-endian support. Nothing to report.

6. Partner specific opcode. Nothing to report.

7. ISP extensions for EV-6. Nothing to report.

8. Small system support. Insufficient leverage to pursue; will be dropped.

9. 32b subset. Nothing to report.

10. Speculative execution support. Nothing to report.

11. Byte/word support. Dileep Bhandarkar's I/O reference model team has made substantive progress, and some of their work is ready to review.

I hope that more of the above areas will have progress to report in the next two weeks.

Thanks /Bob Supnik

I attach a copy of the original minutes, for reference.

Alpha Architecture: Review and Next Steps
*** Digital Confidential ***
*** Do not copy or forward ***

Meeting: Jan 22, 1993
Attendees: Dick Sites, Bob Supnik, Rich Grove, Bill Noyce, Dave Laurello, Bob Stewart, Ray Lanza, Gary Lidington, Anton Chernoff
Invited but unable to attend: Brian Allison, Nancy Kronenberg

Review subjects:
- What worked well
- What did not work well
- Areas for further work

* denotes responsible person for action item.

1. The console needs urgent revision. Not only are there two separate consoles (VMS/OSF and NT), but implementation cost is high and impact on systems is prohibitive (VMS/OSF console takes 2MB of system memory, 1MB of flash EEROM).

   Problem: finding a full time owner/driver for a new console.

   Actions: Supnik* to find owner/driver.
   Lidington to review functional requirements, partitioning.
   Lanza to check whether ARC console spec sufficies for OSF booting.
   [Thomas to check same for VMS.]

2. Agreed: Revise SRM so that if future 6-bit opcodes are used for new operates, function fields must be policed in hardware (ie, unused function codes trap). This significantly eases compatibility issues.

   Action: Reith* to draft ECO.

3. IEEE floating point handling is not adequate.

   Agreed: Prohibition on greater hardware support must be lifted.
   Agreed: If IEEE supported, all rounding modes must be in hardware.
   Agreed: Recommendation on default rounding mode in FPCR should be 'unbiased rounding', not 'plus infinity'.
   Needed: Revision of FPCR and exception reporting.

   Action: Chernoff* to lead team, including Noyce, Hobbs, chip implementer (Wolrich or Samudrala) to revise IEEE specification.

4. 128b floating oint data types is needed.

   Agreed: Use HP format (1 bit sign, 15 bit exponent, 113 bit fraction with hidden bit)

   Action: Sites* to draft ECO, with help from FORTRAN team.

5. Bi-endian support should be formalized.

   Agreed: The EV-5 hooks (as proposed by Cray) should become standard.
   To research: Should these be made dynamically accessible to software?

   Actions: Sites* to draft ECO.

6. Policy on partner-specific instructions.

   Agreed: We need a way to allow partners to add purpose-specific

instructions to the architecture. This is difficult given the limited opcode space and the requirements for compatibility among all implementations.

Action: Reith*, Supnik to develop policy.

7. ISP extensions. Many instruction extensions have been proposed. Sites expressed the general view: rather than piecemeal approval, all ideas should be evaluated together, and implemented at some major change point (eg, EV-6).

   Suggestions:
   - integer divide — JUST ADD CODE POINT
   - floating multiply-add
   - integer-floating transfers
   - hardware square root (or inverse) — OR SQRT ASSIST — "USES SAME HARDWARE AS DIVIDE"
   - pipelineable reciprocal or divide step
   - priority encode, population count
   - logical operations in floating point unit — MAY NOT NEED THIS IF INT-FP XFER
   - cache line prefetch, cache line lock
   - better use of degenerate code points (eg, SLL R31, #n, Rc)
   - better constant generation in integer and floating point

   Each suggestion will need detailed justification and proof of payback.

   Action: Uhler*, Clark to lead evaluation as part of EV-6.

8. Better support for small systems. Ideas include: 4KB page support, relaxed policy on virtual synonyms.

   Action: Supnik* to investigate and write ECO if appropriate.

9. 32b subset. HLO is concerned that they can never reach the embedded market with a 64b implementation. Sites believes there is a 32b subset such that binary programs from the 32b version will execute correctly and without change on the 64b machine. Neither VMS nor OSF would support this.

   Dobberpuhl* and Sites to investigate.

10. Speculative execution support. The EV-6 compiler investigation suggests that the architecture needs explicit support of speculative execution.

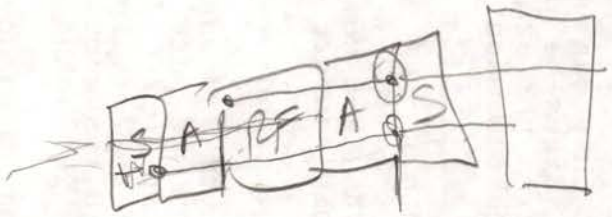
    Action: Lowney*, Grove to investigate and prepare proposal.

11. Byte/word support. The lack of byte and word support creates problems in both I/O support and in applications porting. A short-term proposal (from Mark Woodbury) proposes a compiler/RTL/PAL solution to provide an illusion of byte/word capability for I/O. In the long term, we must decide whether to add byte and word instructions to the architecture for I/O support, memory operations, or both.

    Action: Supnik* to drive evaluation of Woodbury proposal. Requires evaluation by compilers (Glossop, Grove), VMS (Thomas), and OSF/1 (Parenti).

Hardcopies from the meeting will be sent by paper mail.

/Bob



(Message inbox:9)
Return-Path: "HUMAN::SUPNIK"@beantn.pa.dec.com
Message-Id: <9310210100.AA06694@raquet.pa.dec.com>
Date: Wed, 20 Oct 1993 17:58:46 -0700
From: "HUMAN::SUPNIK"@beantn.pa.dec.com (LJO2/D11, 226-2722, Secretary: 226-2723
To: uhler@raquet.pa.dec.com
Subject: LPA-5
X-Vms-To: PADC::DAND,PADC::UHLER,AD::WITEK
X-Vms-Cc: SUPNIK

When I talked to Rich, he said that Alpha's shot at the low power/low cost market couldn't wait until LPA6; LPA5 would have to be a leadership product. I agree. But the disparity between required outcome and available time is pretty large. LPA5's goals would best be served with a new design, yet the time available mandates working from the EV4/LCA core.

Assuming that the design center will be portable devices, 'green' PCs, and very low cost desktop devices, here are some comments on what's been learned, over two years, about performance.

1. Architectural Issues

- The lack of byte and word capability hurts. Instruction text size on Alpha is 22%-23% larger than on Mips (verified on both UNIX and NT). This impacts the Icache, drives up minimum memory requirements, etc.

- A 4KB page is "industry standard" (some OS's, like NetWare, have it essentially hardwired). UNIX and NT both assert that Alpha's 8KB page increases minimum memory requirements. No quantitative proof yet.

[-Floating point performance. For completeness, Alpha's fp performance is hurt by lack of integer/floating transfers and hardware square root. Interestingly, no performance problem has ever been traced to lack of integer divide.]

2. Chip Implementation Issues

- Long functional unit latencies in EV4/LCA are difficult to cover. The compiler and performance analysis teams highlight:
  > the integer multiplier
  > the shifter (used in byte ops)
  > the address calculation-to-use delay
  > the compare-to-branch delay
  [> the floating point datapath]
  The floating divider, a problem in EV4, is significantly improved in EV45 and LCA45.

- The memory controller (EV4) is single threaded (one memory transaction outstanding). EV4, even after the pass 3 fix, has problems around resumption of operations after fills.

3. System Implementation Issues

- Cache latencies are too long; the block size for the Dcache might be better set at 16B instead of 32B.

- Memory bandwidth is typically inadequate for technical stride-1 vectorizable applications (IBM beats Alpha workstations 2:1 or more).

- Memory latency is invariably too long (500ns+ in Laser)>

4. Suggestions and Conclusions

If the time were available, I would probably suggest something like the Orion design: a simple minded 5 stage integer pipeline with an adequate but not spectacular floating point unit. EV4's internal structure is

build for a Testarosa, when the market wants a Honda Civic.

Since that's not feasible, the question becomes, how to improve EV4/LCA's performance while doing minimal damage to the current investment in circuit design and layout. Some ideas:

- Replace the data memory hierarchy with an on chip writeback cache. This will improve performance, simplify system design, help with the int/float transfer problem, etc. The performance of the R4000 series can help answer whether offchip cache support is still required. (I hope not; two level write back caches are complicated.)

- Optimize the memory controller, particularly with respect to restarting the pipeline after a cache miss.

- Make the Icache set associative. Alpha's performance with shared libraries is erratic because of cache location conflicts.

- Speed up the slow functional units without altering the basic pipeline structure. EV5 has shown that it's possible to construct a one cycle shifter; likewise, faster multiply circuits are available. (Whether they fit in the available space, as the divider did, is unknown. I also don't know if the compare-to-branch latency can be shortened.)

If more radical approaches can be considered:

- Scrap the separate Abox. For integer apps, the extra latency for address calculations to loads wipes out the benefits of double issuing loads and integer ops. (Zarka's data shows that EV4 double issues less than 10% on integer apps.) If VA calculations were done in the main datapath, the integer side of the chip would get simpler and smaller. Perhaps the latency to Dcache can be shortened, which would further improve performance and simplify bypassing.

- A differing floating point unit could be radically smaller and less power hungry: allowing either reduction in die area or increase in the on chip caches. (I don't know whether minor surgery, like halving the size of the multiplier array, would yield enough area to be worthwhile.)

Lastly, if architectural changes can be encompassed:

- Add byte and word capability.

- Support 4KB pages (but aliasing issues?).

/Bob

(Message inbox:10)
Return-Path: sites
Received: by gentilly.pa.dec.com; id AA00634; Thu, 21 Oct 93 10:09:31 -0700
Message-Id: <9310211709.AA00634@gentilly.pa.dec.com>
To: witek, uhler, dobberpuhl, meyer@ad.enet.dec.com, sites
Subject: Summary of potential architecture additions
Date: Thu, 21 Oct 93 10:09:29 -0700
From: sites
X-Mts: smtp

Alpha AXP PHASE II Proposals
rls 19 Oct 93

"Later is dirtier." -- Fred Brooks

Here is my list of things that have been proposed as architecture changes, most targetted for EV-6 time frame. Thoughts/comments? /dick

- Bi-endian LDL/STL xor bit 2; EXT/INS/MSK xor Rb<2:0> add to SRM
- Extended float w/SW emulation; add to SRM (see below)
- Performance counter PALcalls; add to SRM

- Rpcc redefined: top 32 bits os specific [soon]
- Kernel halt redefined: os specific action [soon]
- Allow all of memory as lock range [Jensen], but discourage it
  [VMS interlocked queues time out under heavy I/O load.]

- possible IMB selective [WNT]
- possible 4KB page [WNT]

- LD/ST byte?
- LD/ST word?

- LDO/STO/LDX/STX aligned octawords

- Integer
  Count leading zeros 0..64
  Count trailing zeros 0..64
  Count 1-bits 0..64
  CMPBGT/WGE/WGT byte and word, both GT and GE
  INT->FLT fast move
  FLT->INT fast move
  SL1 shift left 1-bit 0..63 (makes 0000...010...00)
  SLM shift left -1 0..63 (makes 1111...000 masks)
  IDIVL/Q UIDIVL/Q (define overflow, divide by zero;
  zdiv = 1 div for Fortran)

- Cache lines: load all zero? give size of line
- Solid writes: hint to avoid any reads, or to add trailing zero bytes
  instead of read
- Non-cached bulk move mem->mem, mem->I/O, I/O->mem, mem->cpu

- Float
  and/or/xor [don't; in favor of quick int<-->flt]
  recip divide, aka cheap divide
  mul/add and mul/sub
  Give IEEE results w/o TRAPB; not require non-finite traps
  (keep TRAPB for IEEE trap routines)

- Extended float
  format 1/15/112+h little-endian F1/F0 sign in F1 (I lost the coin flip)
  ADDX/SUBX/MULX/DIVX Feven,Feven,Feven
  F30 fetch gets s=0 exp=0, manhi=0 [zero/denormal]
  sw must put 0 in F30 to get true zero operand

```
        F30 store keeps low part only; or unpredict (?)
        MULTTX (X=T*T?) Fany,Fany,Feven [overlapping regs]
        Exception mask 1 reg or 2 reg (?)
        Call Std. (?)
        CPYSEX 15-bit Fany,Fany,Fany 1 reg (?)
        CMPXEQ/LE/LT/UN Feven, Feven,Fany [overlapping regs]
        CVTXT Feven,Fany          Rounds, just like T to S
        CVTTX Fany,Feven          no loss
        CVTXQ Feven,Fany          Rounds, just like T to Q
        CVTQX Fany,Feven          no loss

Summary:
                                                register
                                                pattern
        ADDX    Feven + Feven -> Feven          222     ev6 (PALcode for non-finite)
        SUBX          -                         222     ev6
        MULX          *                         222
        DIVX          /                         222
         [MULTTX Fany * Fany  -> Feven          112 ?]
        CPYSEX Fany    Fany  -> Fany            111     ev6
        CMPXxx Feven : Feven -> Fany            221     ev6
        CVTTX  --      Fany  -> Feven           112     ev6
        CVTQX  --      Fany  -> Feven           112
        CVTXT  --      Feven -> Fany            221 [F30?] ev6
        CVTXQ  --      Feven -> Fany            221 [F30?]
Could just do even registers only; 1-reg source/result in even
13 instructions

        Fbranch zero? No; 2 branches; or/branch; cmpx/branch; ignore denorm
```

The duck meeting

Agenda

- ✓ Byte/Word load/store
- ✓ 4Kpages
- ✓ big endian
- ✓ new int ops
- ✓ new FP ops
- ✓ 128 bit floating point
- ✓ Cache ops
- ✓ Emulation support
- ✓ 32bit subset
- ✓ Random

Byte/Word load/store

NO

- ✓ no
  - by time ev6 and lpa6 are done either worked around or too late.
  - compilers will not support it since emulation too slow
  - PCI getting rid of bytes/words
- ✓ yes
  - needed for I/O
  - smaller I-stream
  - software folks would shutup

4K pages

NO

- ✓ No hard data
- ✓ do only for 32bit va systems (10:10:12) as a mode
- ✓ again not hard to do - will get people to shutup

Full big endian

YES

- ✓ Dstream
  - ldl/stl xor va bit 2
  - ext/ins/msk xor rbx<2:0>

NO

- ✓ Istream - yes

New int ops

DO IFF WE FIND SOME USES

- ✓ FFS, FFC
- ✓ count leading/trailing zeros 0..64
- ✓ pop count

GET PRECISION FP IN SOFTWARE

- ✓ CMPBGT/WGE/WGT — BOOLEANS GT, EQ, NE WORD EXTENSIONS
- ✓ Int <-> Flt fast move — HIGH

NO

- ✓ gen bit mask rc <- (((-1)>>(63-rb))<<ra)
- ✓ shift left 1-bit 0..63 makes (00..010..00)
- ✓ shift left 1-bit 0..63 makes (11..00)

CONSTANT SHIFTS

NEED

- ✓ int DIV l/q signed /unsigned
- ✓ ldo/sto aligned octaword

NO

Floating Point

- ✓ and/or/xor
- ✓ recip, recip sqrt
- ✓ square root
- ✓ mul/add , mul/sub
- ✓ Give IEEE without trap
  - except denorm
  - with denorm

128 bit Floating point

- ✓ double register load/store
- ✓ hw vs software
- ✓ support ops for sw

Cache ops

- ✓ block zero
- ✓ zero to end of block
- ✓ block invalidate
- ✓ block flush
- ✓ IMB(VA)?
- ✓ Non-cached bulk move

Emulation support

- ✓ CC bits
- ✓ Instruction dispatch
- ✓ short writes to Reg file

32bit subset

- ✓ No too much software work

Random

- ✓ Allow all of memory as lock - but discourage it
- ✓ Performance count PALcalls
- ✓ Rpcc top 32 bits os specific
- ✓ Kern halt redifined to os specific

The duck meeting

Agenda

- ✓Byte/Word load/store
- ✓4Kpages
- ✓big endian
- ✓new int ops
- ✓new FP ops
- ✓128 bit floating point
- ✓Cache ops
- ✓Emulation support
- ✓32bit subset
- ✓Random

Byte/Word load/store

NO

- ✓no
  - by time ev6 and lpa6 are done either worked around or too late.
  - compilers will not support it since emulation too slow
  - PCI getting rid of bytes/words
- ✓yes
  - needed for I/O
  - smaller I-stream
  - software folks would shutup

4K pages

NO

- ✓No hard data
- ✓do only for 32bit va systems (10:10:12) as a mode
- ✓again not hard to do - will get people to shutup

Full big endian

YES

- ✓Dstream
  - ldl/stl xor va bit 2
  - ext/ins/msk xor rbx<2:0>

NO

- ✓Istream - yes

New int ops

DO IFF WE FIND SOME USES

- ✓FFS, FFC
- ✓count leading/trailing zeros 0..64
- ✓pop count

GET PRECISION FP IN SOFTWARE

- ✓CMPBGT/WGE/WGT — BOOLEANS GT, EQ, NE WORD EXTENSIONS
- ✓Int <-> Flt fast move — HIGH

NO

- ✓gen bit mask rc <- (((-1)>>(63-rb))<<ra)
- ✓shift left 1-bit 0..63 makes (00..010..00)
- ✓shift left 1-bit 0..63 makes (11..00)

CONSTANT MULTIPLY

NEED

- ✓int DIV l/q signed /unsigned
- ✓ldo/sto aligned octaword

NO

Floating Point

- ✓ and/or/xor
- ✓ recip, recip sqrt
- ✓ square root, 1/SQRT
- ✓ mul/add, mul/sub
- ✓ Give IEEE without trap
  - except denorm
  - with denorm

128 bit Floating point

- ✓ double register load/store
- ✓ hw vs software
- ✓ support ops for sw

Cache ops

- ✓ block zero
- ✓ zero to end of block
- ✓ block invalidate
- ✓ block flush
- ✓ IMB(VA)?
- ✓ Non-cached bulk move

Emulation support

- ✓ CC bits
- ✓ Instruction dispatch
- ✓ short writes to Reg file

32bit subset

- ✓ No too much software work

Random

- ✓ Allow all of memory as lock - but discourage it
- ✓ Performance count PALcalls
- ✓ Rpcc top 32 bits os specific
- ✓ Kern halt redifined to os specific

The duck meeting

Agenda

- ✓ Byte/Word load/store
- ✓ 4Kpages
- ✓ big endian
- ✓ new int ops
- ✓ new FP ops
- ✓ 128 bit floating point
- ✓ Cache ops
- ✓ Emulation support
- ✓ 32bit subset
- ✓ Random

Byte/Word load/store

NO

- ✓ no
  - by time ev6 and ipc6 are done either worked around or too late.
  - compilers will not support it since emulation too slow
  - PCI getting rid of bytes/words
- ✓ yes
  - needed for I/O
  - smaller I-stream
  - software folks would shutup

4K pages

NO

- ✓ No hard data
- ✓ do only for 32bit va systems (10:10:12) as a mode
- ✓ again not hard to do - will get people to shutup

Full big endian

YES

- ✓ Dstream
  - ldl/stl xor va bit 2
  - ext/ins/msk xor rbx<2:0>

NO

- ✓ Istream - yes

New int ops

DO IFF WE FIND SOME USES

- ✓ FFS, FFC
- ✓ count leading/trailing zeros 0..64
- ✓ pop count

} GET PRECISION FP IN SOFTWARE

- ✓ CMPBGT/WGE/WGT < EQUIVS GT, EQ, NE WORD EXTENSIONS
- ✓ Int <-> Flt fast move – HW it

NO

- ✓ gen bit mask rc <- (((-1)>>(63-rb))<<ra)
- ✓ shift left 1-bit 0..63 makes (00..010..00)
- ✓ shift left 1-bit 0..63 makes (11..00)

} CONSTANT GENERATION

NEED

- ✓ int DIV I/q signed /unsigned
- ✓ ldo/sto aligned octaword

NO

Floating Point

- ✓ and/or/xor
- ✓ recip, recip sqrt
- ✓ square root, 1/SQRT
- ✓ mul/add, mul/sub
- ✓ Give IEEE without trap
  - except denorm
  - with denorm

128 bit Floating point

- ✓ double register load/store
- ✓ hw vs software
- ✓ support ops for sw

Cache ops

- ✓ block zero
- ✓ zero to end of block
- ✓ block invalidate
- ✓ block flush
- ✓ IMB(VA)?
- ✓ Non-cached bulk move

Emulation support

- ✓ CC bits
- ✓ Instruction dispatch
- ✓ short writes to Reg file

32bit subset

- ✓ No too much software work

Random

- ✓ Allow all of memory as lock - but discourage it
- ✓ Performance count PALcalls
- ✓ Rpcc top 32 bits os specific
- ✓ Kern halt redifined to os specific

The duck meeting

Agenda

- ✓ Byte/Word load/store
- ✓ 4Kpages
- ✓ big endian
- ✓ new int ops
- ✓ new FP ops
- ✓ 128 bit floating point
- ✓ Cache ops
- ✓ Emulation support
- ✓ 32bit subset
- ✓ Random

Byte/Word load/store

NO

- ✓ no
  - by time ev6 and lpa6 are done either worked around or too late.
  - compilers will not support it rince emulation too slow
  - PCI getting rid of bytes/words
- ✓ yes
  - needed for I/O
  - smaller I-stream
  - software folks would shutup

4K pages

NO

- ✓ No hard data
- ✓ do only for 32bit va systems (10:10:12) as a mode
- ✓ again not hard to do - will get people to shutup

Full big endian

YES

- ✓ Dstream
  - ldl/stl xor va bit 2
  - ext/ins/msk xor rbx<2:0>

NO

- ✓ Istream - yes

New int ops

DO IFF WE FIND SOME USES

- ✓ FFS, FFC
- ✓ count leading/trailing zeros 0..64
- ✓ pop count

GET PRECISION FP IN SOFTWARE

- ✓ CMPBGT/WGE/WGT — BOOLEANS GT, EQ, NE AND EXTENSIONS
- ✓ Int <-> Flt fast move — HW IT

NO

- ✓ gen bit mask rc <- (((-1)>>(63-rb))<<ra)
- ✓ shift left 1-bit 0..63 makes (00..010..00)
- ✓ shift left 1-bit 0..63 makes (11..00)

CONSTANT GENERATION

NEED

- ✓ int DIV l/q signed /unsigned
- ✓ ldo/sto aligned octaword

Floating Point

- ✓ and/or/xor
- ✓ recip, recip sqrt
- ✓ square root, 1/SQRT
- ✓ mul/add, mul/sub — NO
- ✓ Give IEEE without trap
  - except denorm
  - with denorm

128 bit Floating point

- ✓ double register load/store
- ✓ hw vs software
- ✓ support ops for sw

Cache ops

- ✓ block zero
- ✓ zero to end of block
- ✓ block invalidate
- ✓ block flush
- ✓ IMB(VA)?
- ✓ Non-cached bulk move

Emulation support

- ✓ CC bits
- ✓ Instruction dispatch
- ✓ short writes to Reg file

32bit subset

- ✓ No too much software work

Random

- ✓ Allow all of memory as lock - but discourage it
- ✓ Performance count PALcalls
- ✓ Rpcc top 32 bits os specific
- ✓ Kern halt redifined to os specific

The duck meeting

Agenda

- ✓Byte/Word load/store
- ✓4Kpages
- ✓big endian
- ✓new int ops
- ✓new FP ops
- ✓128 bit floating point
- ✓Cache ops
- ✓Emulation support
- ✓32bit subset
- ✓Random

Byte/Word load/store

NO

- ✓no
  - by time ev6 and lpa6 are done either worked around or too late.
  - compilers will not support it since emulation too slow
  - PCI getting rid of bytes/words
- ✓yes
  - needed for I/O
  - smaller I-stream
  - software folks would shutup

4K pages

NO

- ✓No hard data
- ✓do only for 32bit va systems (10:10:12) as a mode
- ✓again not hard to do - will get people to shutup

Full big endian

YES

- ✓Dstream
  - ldl/stl xor va bit 2
  - ext/ins/msk xor rbx<2:0>

NO

- ✓Istream - yes

New int ops

DO IFF WE FIND SOME USES

- ✓FFS, FFC
- ✓count leading/trailing zeros 0..64
- ✓pop count

} GET PRECISION FP IN SOFTWARE

- ✓CMPBGT/WGE/WGT < COULD HAVE GT, EQ, NE WORD EXTENSIONS
- ✓Int <-> Flt fast move – HIGH

NO

- ✓gen bit mask rc <- (((-1)>>(63-rb))<<ra)
- ✓shift left 1-bit 0..63 makes (00..010..00)
- ✓shift left 1-bit 0..63 makes (11..00)

} CONSTANT GENERATION

NEED

- ✓int DIV l/q signed /unsigned
- ✓ldo/sto aligned octaword

ADD

Floating Point

- ✓ and/or/xor
- ✓ recip, recip sqrt
- ✓ square root, 1/sqrt
- ✓ mul/add, mul/sub
- ✓ Give IEEE without trap
  - except denorm
  - with denorm

128 bit Floating point

- ✓ double register load/store
- ✓ hw vs software
- ✓ support ops for sw

Cache ops

- ✓ block zero
- ✓ zero to end of block
- ✓ block invalidate
- ✓ block flush
- ✓ IMB(VA)?
- ✓ Non-cached bulk move

Emulation support

- ✓ CC bits
- ✓ Instruction dispatch
- ✓ short writes to Reg file

32bit subset

- ✓ No too much software work

Random

- ✓ Allow all of memory as lock - but discourage it
- ✓ Performance count PALcalls
- ✓ Rpcc top 32 bits os specific
- ✓ Kern halt redifined to os specific

The duck meeting

Agenda

- ✓ Byte/Word load/store
- ✓ 4Kpages
- ✓ big endian
- ✓ new int ops
- ✓ new FP ops
- ✓ 128 bit floating point
- ✓ Cache ops
- ✓ Emulation support
- ✓ 32bit subset
- ✓ Random

Byte/Word load/store

NO

- ✓ no
  - by time ev6 and lpa6 are done either worked around or too late.
  - compilers will not support it since emulation too slow
  - PCI getting rid of bytes/words
- ✓ yes
  - needed for I/O
  - smaller I-stream
  - software folks would shutup

4K pages

NO

- ✓ No hard data
- ✓ do only for 32bit va systems (10:10:12) as a mode
- ✓ again not hard to do - will get people to shutup

Full big endian

YES

- ✓ Dstream
  - ldl/stl xor va bit 2
  - ext/ins/msk xor rbx<2:0>

NO

- ✓ Istream - yes

New int ops

DO IFF WE FIND SOME USES

- ✓ FFS, FFC
- ✓ count leading/trailing zeros 0..64
- ✓ pop count

GET PRECISION FP IN SOFTWARE

- ✓ CMPBGT/WGE/WGT — COULD USE GT, EQ, NE WORD EXTENSIONS
- ✓ Int <-> Flt fast move — HW IT

NO

- ✓ gen bit mask rc <- (((-1)>>(63-rb))<<ra)
- ✓ shift left 1-bit 0..63 makes (00..010..00)
- ✓ shift left 1-bit 0..63 makes (11..00)

CONSTANT GENERATION

NEED

- ✓ int DIV I/q signed /unsigned
- ✓ ldo/sto aligned octaword

ADD

Floating Point

- ✓ and/or/xor
- ✓ recip, recip sqrt – BOTH APPROXIMATE
- ✓ square root, 1/SQRT
- ✓ mul/add, mul/sub – NO
- ✓ Give IEEE without trap
  - except denorm
  - with denorm

} HIGH, BUT NO SUPPORT YET

128 bit Floating point

- ✓ double register load/store
- ✓ hw vs software
- ✓ support ops for sw – STORE ROOT TO STUDY

Cache ops

- ✓ block zero
- ✓ zero to end of block
- ✓ block invalidate
- ✓ block flush
- ✓ IMB(VA)?
- ✓ Non-cached bulk move

Emulation support

- ✓ CC bits
- ✓ Instruction dispatch
- ✓ short writes to Reg file

NO

32bit subset

- ✓ No too much software work

NO

Random

- ✓ Allow all of memory as lock - but discourage it
- ✓ Performance count PALcalls — ITB, DTB, ALIGN
- ✓ Rpcc top 32 bits os specific
- ✓ Kern halt redifined to os specific

Alpha I/O Architecture

System Model & Nomenclature

Saro Saravanan

VMS Engineering

January 14, 1992

DIGITAL INTERNAL USE ONLY

Functional Building Blocks -

Hardware Block:

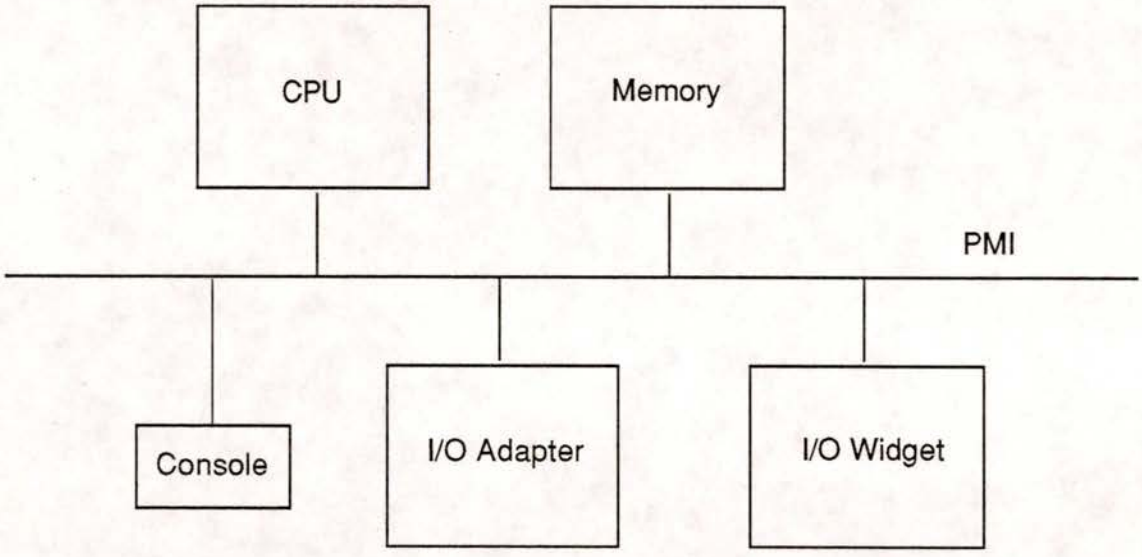
- An assembly of hardware components that implements a specific function. Part of, whole, or multiple HARDWARE MODULEs.
  - A HARDWARE MODULE is a physical building block - board with hardware components.

Interconnect:

- Physical medium carrying address, data, and control signals between hardware blocks
- A hardware block on an interconnect is called a NODE on that interconnect

The Generic Alpha System

Figure 1: Alpha System Model



CPU

- Consists of an Alpha processor, cache memory, and cache management and addressing logic

Memory

- Consists of RAM chips along with interface and control logic

Console

- Consists of a console processor, console storage, and console software. Allows an operator to monitor and control the system.

Processor-Memory Interconnect (PMI)

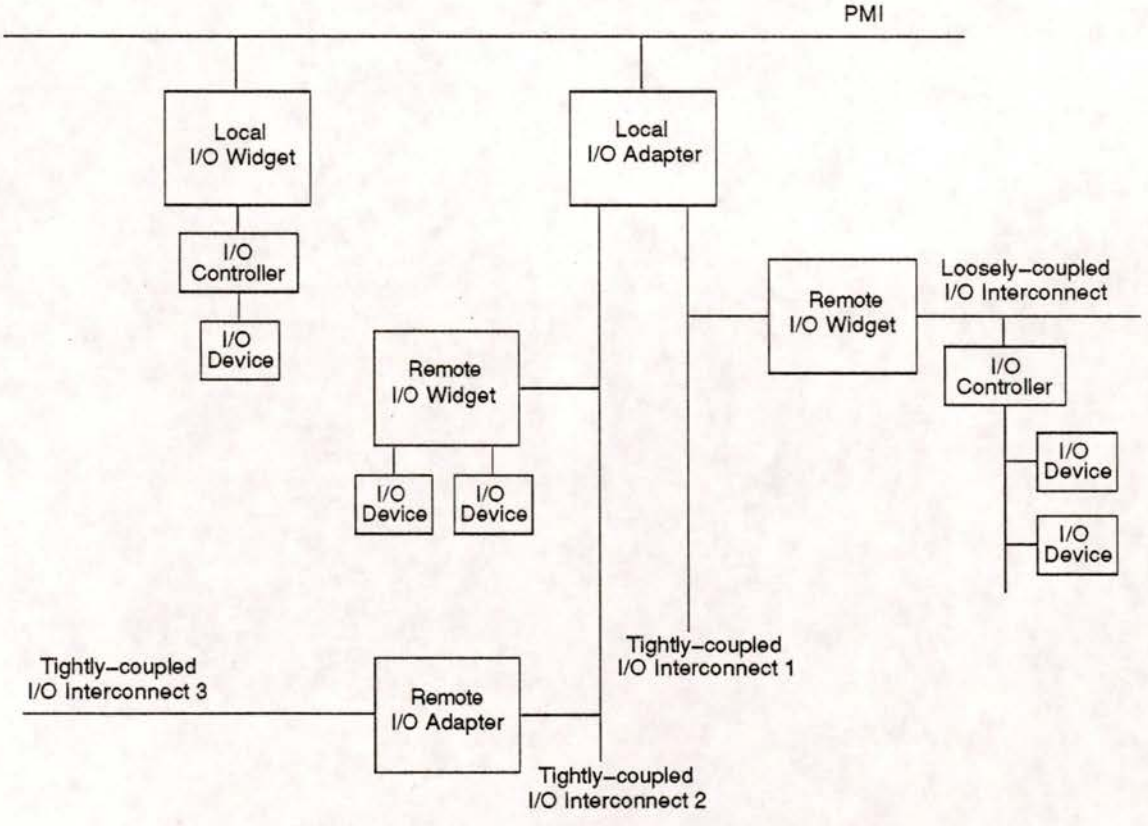
- Interconnect between CPU and memory

I/O Interconnect

- An interconnect whose primary purpose is I/O. Two classes:
  - Tightly-coupled: Address space accessible to the CPU through direct or mailbox access. Examples: XMI, Lbus, TC, FB+, etc.

- Loosely-coupled: Address space inaccessible to the CPU through direct or mailbox access. Examples: SCSI, NI, FDDI, CI, etc.

Figure 2: Alpha I/O Hardware Blocks and Interconnects



I/O Adapter

- Hardware block that connects one or more tightly-coupled interconnects to another tightly-coupled interconnect

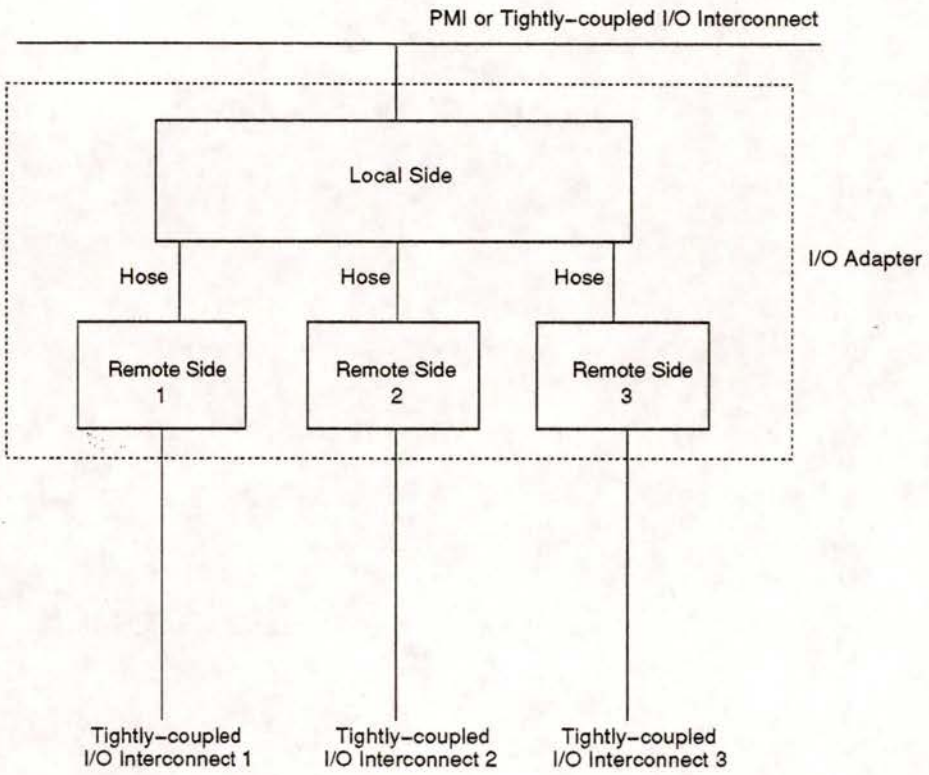
I/O Widget

- Hardware block that connects one or more loosely-coupled interconnects to the PMI or to a tightly-coupled interconnect

Local vs Remote

- Local adapter/widget connects directly to the PMI, whereas remote does not
- Local= closer to PMI, Remote= farther

Figure 3: Components of an I/O Adapter



I/O Device

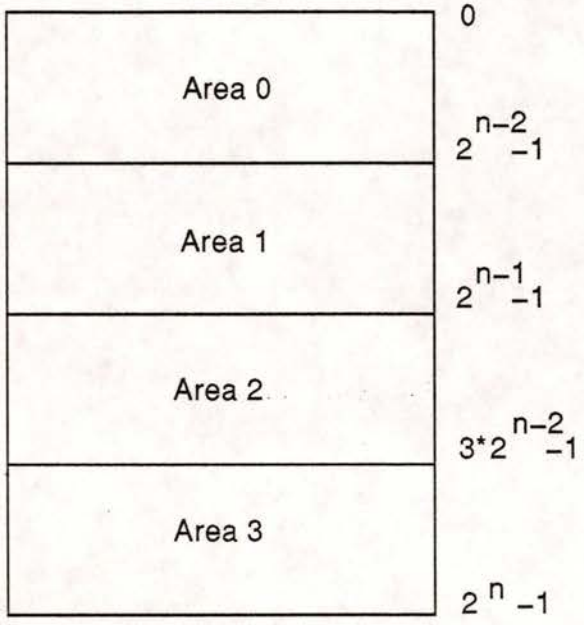
- Target of an I/O operation: can be physical (disk, tape, etc.) or "pseudo"

I/O Controller

- Provides control logic and interface registers to enable software to control one or more devices

Physical Address Space

Figure 4: PA Space



- Collection of all physical locations accessible by a PMI node - extent defined by # of PA bits implemented
- Four areas - each used for either memory references or I/O location references, but not both
- Area 0 used for memory only
- Source of LD and target of ST can be either memory or I/O location

Memory Space

- Part of processor's PA space through which it accesses memory

I/O Space

- Part of processor's PA space through which it accesses HARDWARE INTERFACE REGISTERs (old name: CSRs)
- Local I/O Space = processor's I/O space
- Node Space = Part of the local I/O space through which a node's interface registers are accessed
- Remote I/O space = any tightly-coupled interconnect's address space

Device Driver I/O

General Model

- Driver requests and initiates a device function
- Controller carries out the requested function
- Controller informs the driver of request completion
- Driver determines completion status

Simple Model

- Driver writes one or more interface registers to initiate request
- Controller carries it out
- Controller generates an interrupt to inform the driver of completion
- Driver reads one of more interface registers to determine status

Sophisticated Model

- Driver inits one or more command buffers to request device function
- Driver accesses an interface register to notify the controller of available command buffer(s)
- Controller performs function(s)
- Controller records completion status in one or more response buffers and generates an interrupt to notify driver of one or more request completions
- Driver determines status of completed operations from response buffers

Interface Register Access

Direct Access

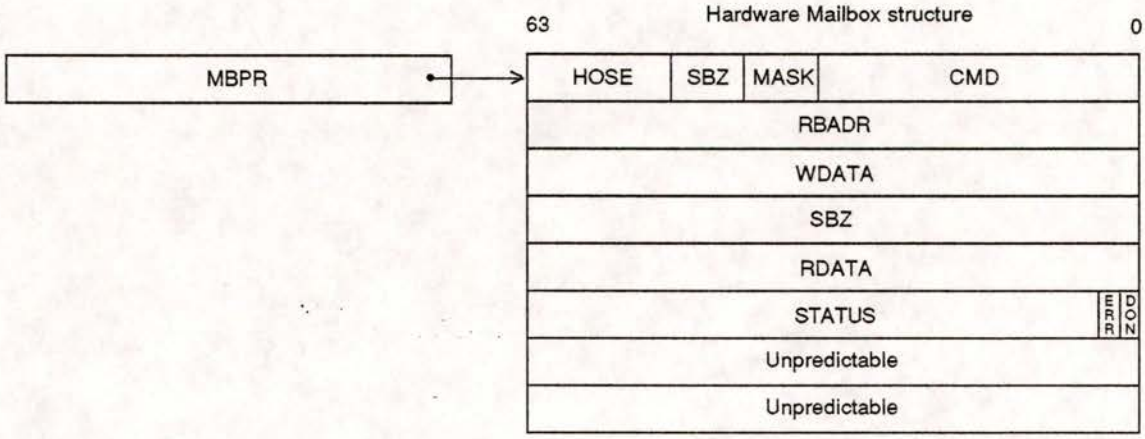
- An interface register with an address in local I/O space is directly accessible with a LD or ST intstruction
- Example: Interface register for any PMI node

Mapped Access

- A remote I/O widget's interface register can be directly accessed if the system implementation maps the addresses into the processor's I/O space.
- Example: Any TC widget's interface registers on Flamingo
- Mapped access is a special form of direct access

Hardware Mailbox Access

Figure 5: The Alpha Hardware Mailbox



- Mechanism for accessing remote interface registers - an alternative for mapped access
- Achieves the same effect as a VAX instruction to read/write a device register, for example:
  - MOVL R3, DEVICE_REGISTER_OFFSET(R4)

Steps taken by driver for hardware mailbox access:

- Allocate hardware mailbox in memory
- Deposit a remote interconnect command in CMD (read/write)
- Write a mask in MASK to indicate size of target location (byte, word, longword, or quadword)
- Identify the target I/O interconnect in HOSE
- Write the tightly-coupled I/O interconnect address of remote interface register in RBADR
- If CMD specifies WRITE, initialize WDATA with data to be written
- Zero all other fields
- Write the hardware mailbox physical address to the MBPR

- Driver polls DON bit
- For a WRITE operation, adapter can set DON immediately after it has read the mailbox (or it can wait)
- For a READ operation, adapter can set DON only after it has written the data to RDATA
- For READ operations only, adapter can set the ERR bit - additional status is contained in the rest of STATUS in that case

Direct Memory Access

DMA from a local I/O widget

- Simple PMI operation

DMA from a remote I/O widget

- Requires assistance of intervening I/O adaper(s)
- Local I/O adapter must map a part of the tightly-coupled I/O interconnect's address space into the processor's memory space
- Need map registers

Tightly- vs Loosely-coupled I/O Interconnects

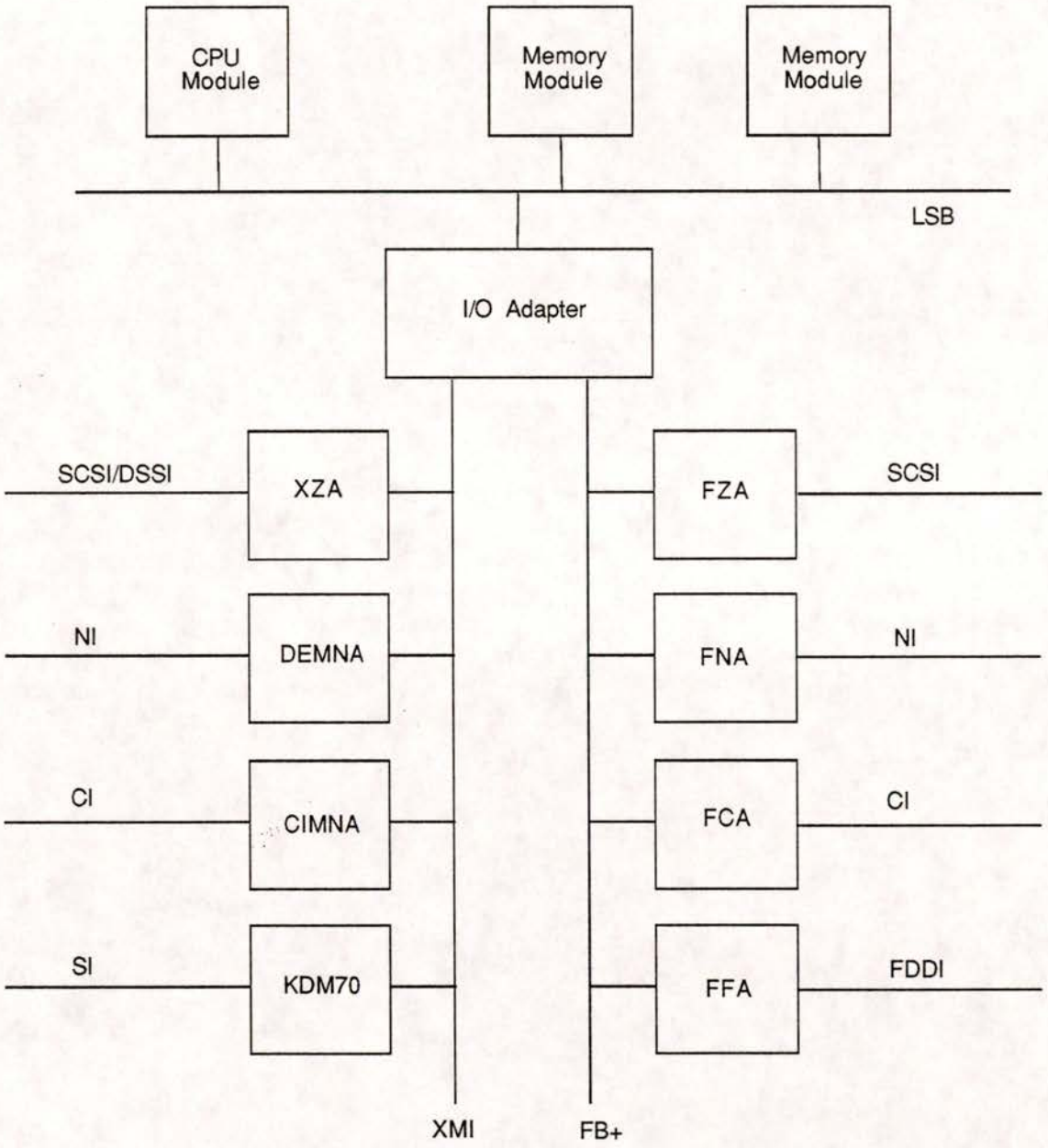
Tightly-coupled:

- CPU can access its address space through direct or mailbox access
- Typically limited in range to the system cabinet

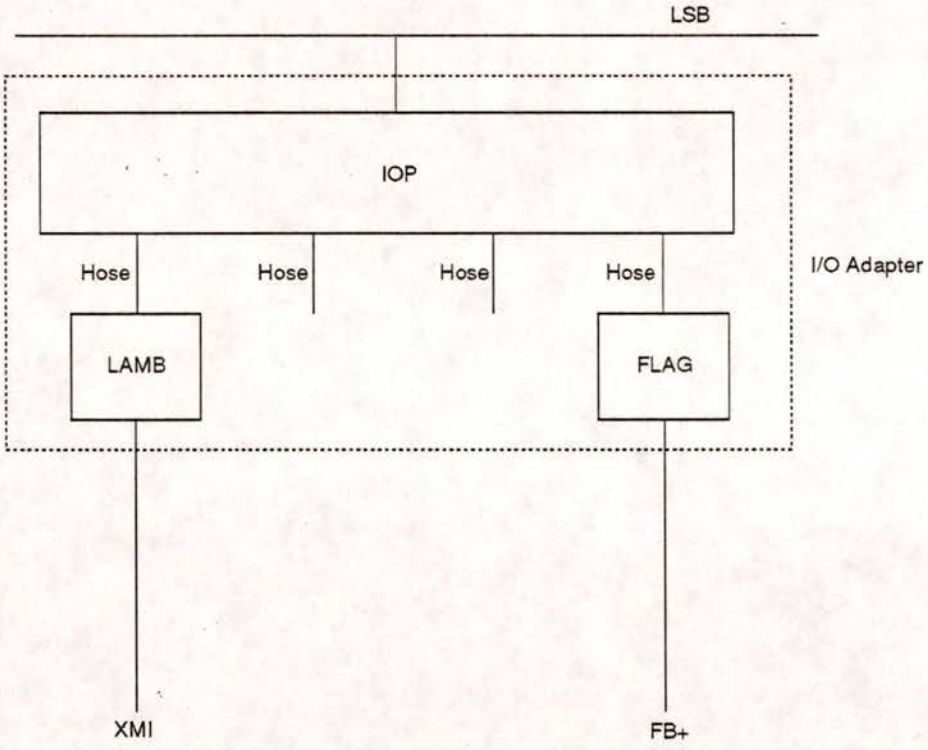
Loosely-coupled:

- CPU cannot access its address space through direct or mailbox access
- Tends to "snake around the room"

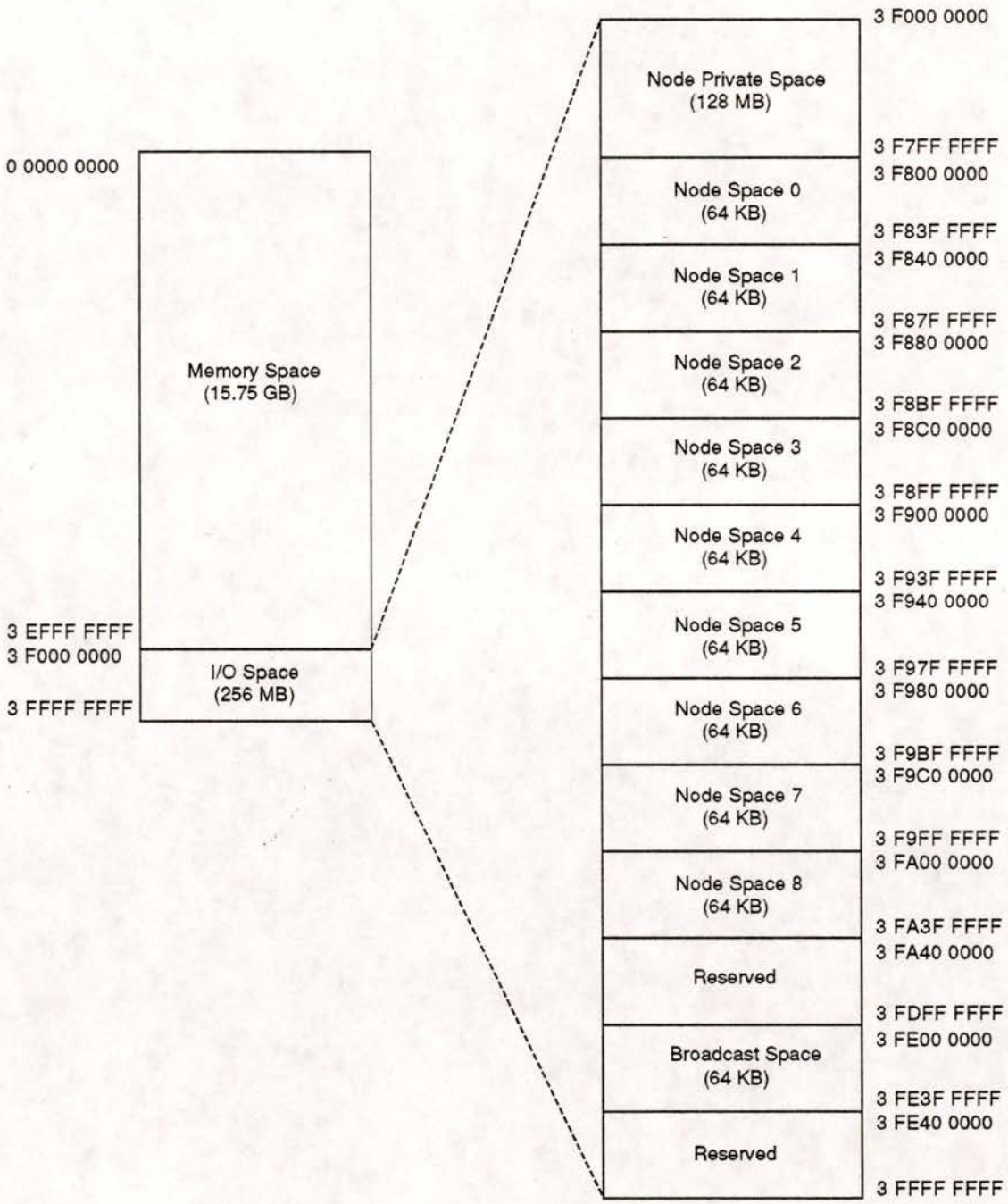
Laser System Configuration



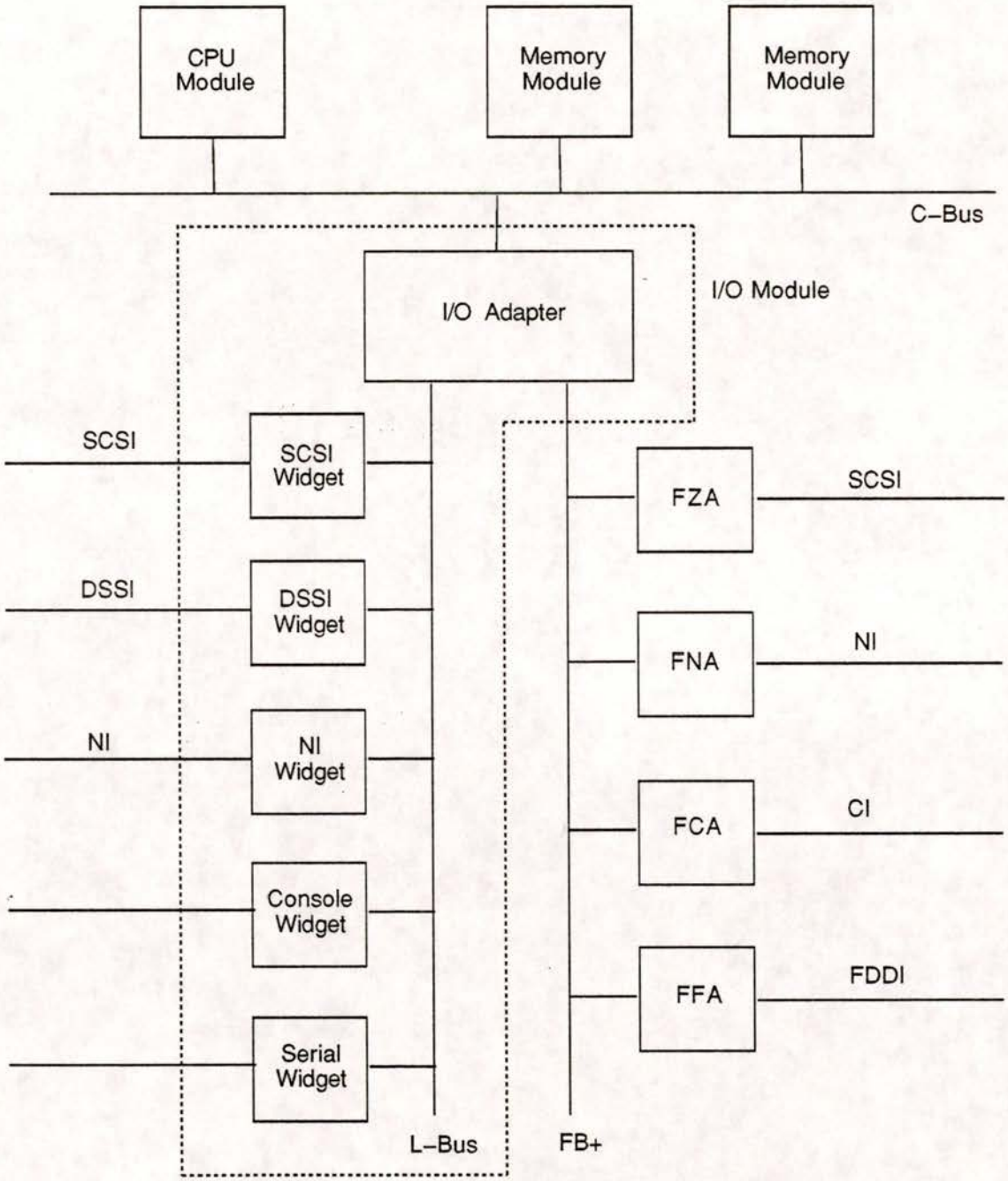
Laser I/O Adapter



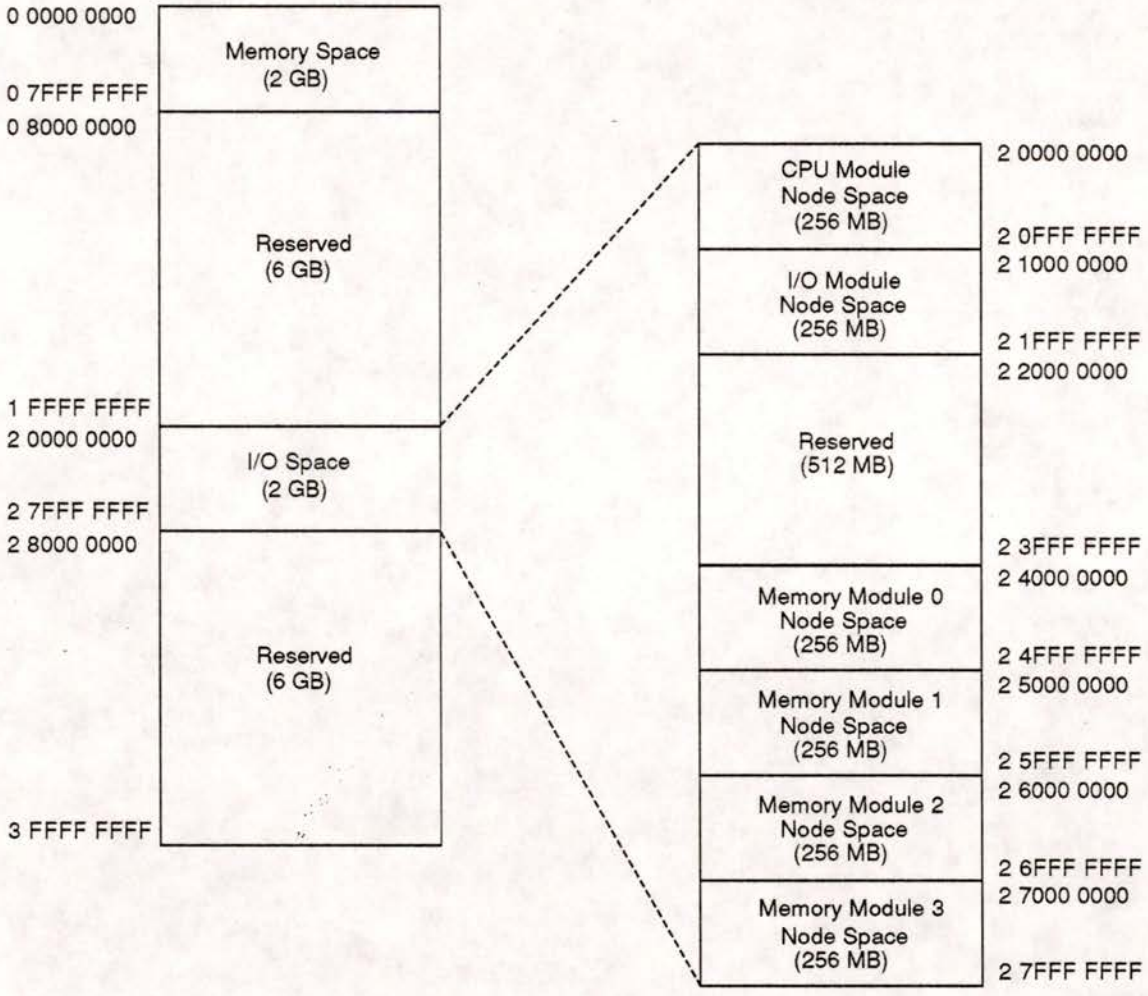
Laser PA Space



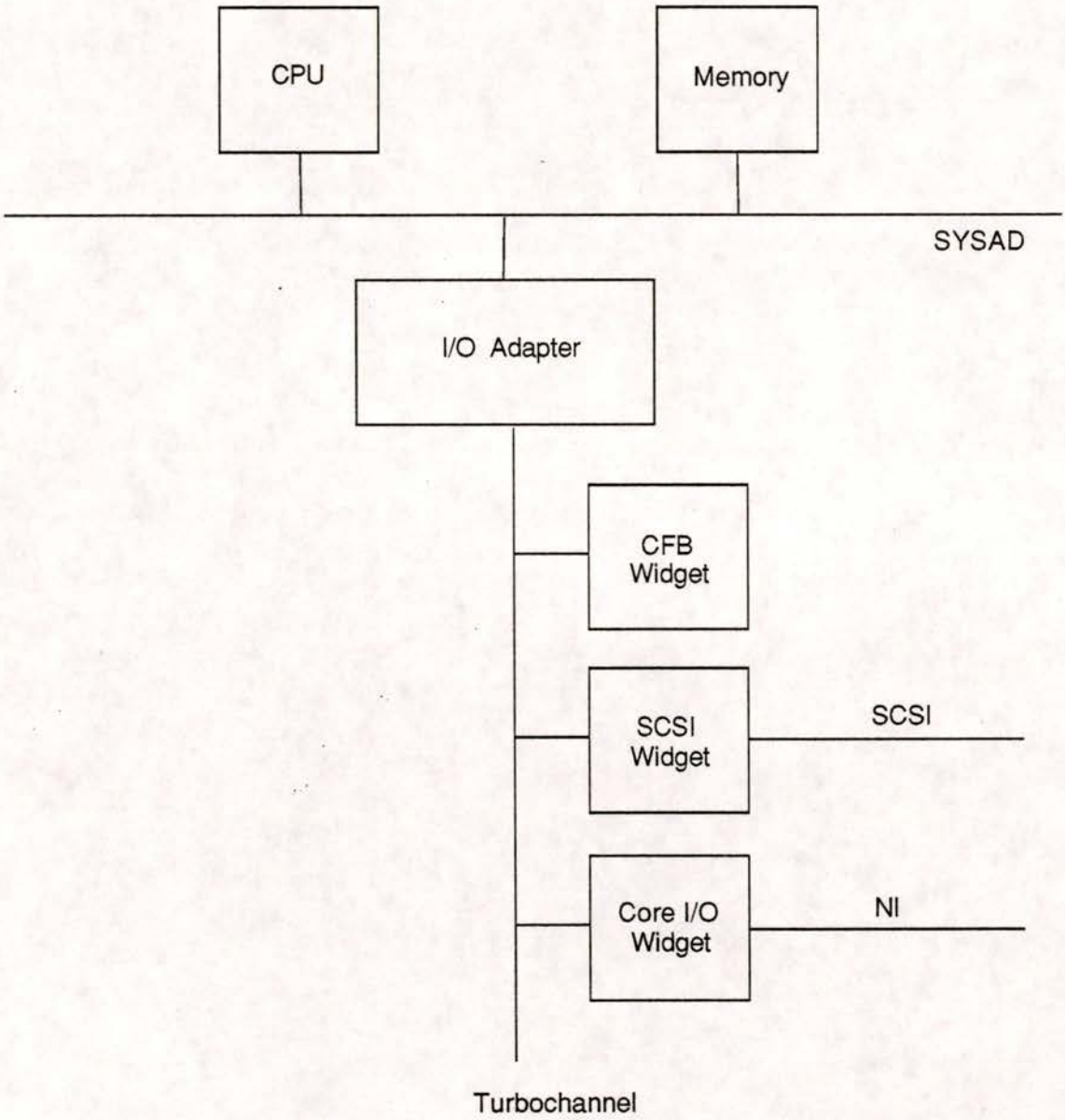
Cobra System Configuration



Cobra PA Space



Flamingo System Configuration



Flamingo PA Space