AL MCGUIRE
SAM FULLER
BILL STRECKER

BRIAN CROXON
JOHN O'KEEFE

BILL DEMMER
STEVE JENKINS

****
* d i g i t a l *
****

TO: BILL DEMMER
DON MCINNIS
cc: SAM FULLER

DATE: SAT 27 JUN 1981 18:12 EST
FROM: GORDON BELL
DEPT: ENG STAFF
EXT: 223-2236
LOC/MAIL STOP: ML12-1/A51

SUBJECT: NAUTILUS: WHY NOT USE MCA'S AND IMPLEMENT COMET

In lookins at the soal for Nautilus, it would seem like one of the key candidates would be to make a Comet (in terms of microcode) and resister structure usins MCA's and runnnins the machine say 5-6 times faster, sivins us a machine with about the same size and cost as comet, which is what I had been led to believe is the tarset.

It is becomins clear that this is another alternative that we should look at quite seriously. Sam's ECL machine may be fast time to market, but it will be very expensive.

A comet made with mca's should easily be doable in 2 years and would be probably as fast as the ecl 780, but much cheaper.

Given that Bob is tied up for awhile on Venus, what about a really hard look at this?

As an alternative, I have heard that the Japanese have turned around some desisns like this in the 6-9 months timeframe. We might consider havins them build a fast machine usins their sate arrays. We have a quote from TEAC that sets us drives to our electronic specs in 3 months! (and product in 4 months from mmanufacturins)

Call Brian

****
* digital *
****

TO: *GORDON BELL

cc: see "CC" DISTRIBUTION

DATE: MON 18 MAY 1981 8:24 EST
FROM: BRIAN CROXON
DEPT: VAX HRDWR ENG'G
EXT: 247-2416
LOC/MAIL STOP: TW/C04

SUBJECT: ATTACHED MEMO

EMS BC/4.54

------*---*---*---*---*
* d i g i t a l *
------*---*---*---*---*

interoffice memorandum

TO: Gordon Bell ML12-1/A51

DATE: 15 May 1981
FROM: Brian Croxon
DEPT: VAX Hardware Eng.
EXT: 247-2416
LOC/MAIL STOP: TW/C04

cc: Distribution

SUBJ: GESTATION:= 3 years
for NAUTILUS: = Scorpio: = 1 step 1 until complete do.

I will arrange a meeting to start a process that maximizes our learning, vis a vis CAD tools, Japan, Standard, Comet, Venus, Scorpio etc. In reviewing all of the advanced VAX programs it is very apparent that without the collective pooling of CAD developments, we cannot make it in five (5) years let alone get it done in three (3). I would like to push for some very strong committments, in collectively pooling the CAD requirements, and achieving a clear hierarchial approach, highly structured from the top system level, down to the chip/ circuit level.

At the first meeting I would like to develop a small but fully committed team, that quickly develops a clear picture of our needs, resolve what has been done, and what we need to act on immediately too get off the dime.

If I have left anybody out, or am unaware of what may already be going on, please let me know. We don't have $ or time to repeat anything.

cc:

BC/lp
Attachment: Gordon Bell EMS May 14(Nautilus: The Next Step)

Distribution:

Bill Demmer	TW/D19
Bill Strecker	TW/B05
Sam Fuller	ML3-5/H33
Steve Teicher	HL1-1
George Hoff	MR1-2/E47
Roy Rezac	MR1-2/E18

Don McInnis	TW/C04
Steve Jenkins	TW/C04
Bob Stewart	TW/C04
Art Lim	TW/B02
Tom Sherman	TW/A08
Geoff Potter	TW/B02
Ulf Fasergvist	MR1-2/E78
Peter Barck	TW/B02

"CC" DISTRIBUTION:

ART LIM @TWSK	BOB STEWART @TWSK	BILL DEMMER
ULF FAGERQUIST	SAM FULLER	GEORGE HOFF
DON MCINNIS	PETER BARCK @TWSK	GEOFFREY POTTER
ROY REZAC	TOM SHERMAN	STEVE JENKINS @TWSK
BILL STRECKER	STEVE TEICHER	

****
* d i g i t a l *
****

TO: *GORDON BELL

cc: see "CC" DISTRIBUTION

DATE: MON 17 AUG 1981 0:16 EST
FROM: DON MCINNIS
DEPT: DISTRIBUTED MID-SYS
EXT: 247-2118
LOC/MAIL STOP: TW/A08

SUBJECT: RE: COMMENT ON NAUTILUS PLAN

WE HAVE SAID THAT WE WILL EXPLORE ALL AVENUES OF DOING BETTER THAN H1FY86 BUT FOR THE TIME BEING WE ARE NOT READY TO COMMIT TO DOING BETTER THAN THAT . WE HAVE STATED THAT WE WILL HAVE A DESIGN METHODOLOGY DOCUMENT THAT (AMONG OTHER THINGS WILL PUT LIMITS ON GATE ARRAY DENSITIES ; BOARD DENSITIES; AND HAVE DESIGNERS USING HIGHER LEVEL MACROS INSTEAD OF DESIGNING AT THE NAND GATE LEVEL . OUR NUMBER ONE GOAL WITH THE FTA GROUP IS TO BUILD AN AUTOMATIC SYSTEM FOR PRACTICALLY ALL DESIGNS.

STRECKER'S STUDIES HAVE SHOWN THAT VECTORS ARE NOT AN ECONOMICAL WIN EXCEPT FOR THE VERY HIGH END.NAUTILUS IS INTENDED AS A PRICE REPLACEMENT FOR THE 750. SEL OFFERS ARRAY PROCESSOR IN SEVERAL OF THEIR MACHINES. THE FPS PEOPLE SAID THEY HAVE NEVER SEEN THEM AS COMPETITORS AND ARENT CONCERNED ABOUT MINI VENDORS AS ARRAY PROCESSOR COMPETITORS BECAUSE WE WONT BE WILLING TO PUT ENOUGH PRODUCT COST INTO THAT PART OF THE DESIGN.THEY WERE MUCH MORE CONCERNED ABOUT US RELEASING PLAIN VANILLA VAXS AT THE HIGH END AT QUICKER RATE.

CONVERTING NAUTILUS A SIMULAR TECHNOLOGY WILL BE A LOT EASIER THAN WITH CURRENT DESIGNS BECAUSE OF THE DATA BASE MODELS THAT WE WILL HAVE BUILT FOR USING DECSIM AND OUR TIMING VERIFIER.

AFTER SEVERAL SURVEYS , TI WAS THE CLEAR CHOICE.
AMONG THE JAPANESE , ONLY FUJITSU WAS WILLING TO SELL US ANY TECHNOLOGY THAT LOOKED AT ALL INTERESTING FOR NAUTILUS. THEY HAVENT BEEN WILLING TO GIVE US MUCH DATA UNTIL THEY HAVE WORKING PARTS IN HAND. JIM GROCHMAL FROM THE NAUTILUS GROUP WILL BE MAKING THE TRIP TO JAPAN IN SEPTEMBER TO GET MORE INFORMATION ON JAPANESE GATE ARRAYS.
THE FUJISU PART APPEARS THAT IT MIGHT BE CLOSE TO TI IN PERFORMANCE AND SCHEDULE.

JEFF'S SOLE ROLE IS THE CAD PROCESS. I AM ALSO TALKING TO L. ABEL AND B. KUSIK ABOUT A POSSIBLE ROLE IN NAUTILUS CAD.

IT'S NOT TOO EARLY TO WORK WITH MANUFACTURING. I HAD A TWO DAY WOODS MEETING LAST WEEK WITH BURLINGTON TO WORK OUT A JOINT SET OF GOALS AND STRATEGIES. INCLUDED IN THAT WAS A SERIES OF STRATEGIES TO REDUCE TIME TO MARKET.

"CC" DISTRIBUTION:

****
* d i g i t a l *
****

TO: ULF FAGERQUIST

cc: see "CC" DISTRIBUTION

DATE: THU 21 MAY 1981 8:57 EST
FROM: BRIAN CROXON
DEPT: VAX HRDWR ENG'G
EXT: 247-2416
LOC/MAIL STOP: TW/C04

SUBJECT: ATTACHED MEMO

BC/4.59

EMS

------*---*---*---*---*
* d i g i t a l *
------*---*---*---*---*

interoffice memorandum

TO: Ulf Fagerquist

cc: George Hoff
Gordon Bell
Steve Rothman
Bill Demmer
Don McInnis

DATE: 21 May 1981
FROM: Brian Croxon
DEPT: VAX Hardware Eng.
EXT: 247-2416
LOC/MAIL STOP: TW/C04

SUBJ: Reply to Gordon Bell's Memo- "Understanding Comet to help Venus & Nautilus"

We are very willing to start getting together as soon as possible in sharing our experiences relative to Comet. I would also like to include the Nautilus/Scorpio Systems people as they have some critical needs in the CAD space as well. Will you set the time and place?

BC/lp

"CC" DISTRIBUTION:

*GORDON BELL
DON MCINNIS
BILL DEMMER
STEVE ROTHMAN
GEORGE HOFF

Nautilus

MAY 26 1981

SJ 1-33

------*---*---*---*---*---*
* d i g i t a l *
------*---*---*---*---*---*

INTEROFFICE MEMORANDUM

TO: Gordon Bell

CC: Brian Croxon
Bill Demmer
Don McInnis
Jeff Mitchell
Bob Stewart
Bill Strecker

DATE: 20 May 1981
FROM: Steve Jenkins
DEPT: Adv. VAX Systems Eng.
EXT: 247-2395
LOC/MAILSTOP: TW/C04

SUBJECT: YOUR EMS OF 14 MAY (NAUTILUS: THE NEXT STEP)

I absolutely agree with you about defining a design methodology that is appropriate for the complexity of the program. We did not intend to give you the impression that we didn't recognize the problem or that we were going to design this system in a similar way that we did the 11780 or 11750.

We will be putting together in the next month a Nautilus CAD strategy that will include at least three alternatives: 1) Semi-soft simulator (a hardware accelerator for CAD SIMULATION), 2) DECSIM, and 3) Industry available tools. I hope to review this with you at the end of June.

I have set up a meeting with Dickhut and Supnik in order that we may share our thoughts on techniques for structured designs and system simulations. I already have an ongoing rapport with the VENUS and COMET design teams to learn from their experiences. In addition I will try to schedule meetings with Tom Williams (S1) who I understand will be visiting Sam Fuller soon and with Rezac.

****
* d i g i t a l *
****

TO: AL MCGUIRE @TWSK

cc: see "CC" DISTRIBUTION

DATE: WED 12 AUG 1981 10:33 EDT
FROM: HAP PRINDLE
DEPT: PRODUCT PLANNING
EXT: 231-6239
LOC/MAIL STOP: MR2-3/M84

SUBJECT: NAUTILUS PHASE 0

In preparation for the NAUTILUS review I would like to resubmit some questions which I don't feel have been answered to date.

1. Why do Nautilus at all:
   SCORPIO available sooner
   Multiprocessor primitives available on BI
   Could have price/performance VAX family alternatives from SCORPIO throush MULTISCORPIO to VENUS (1 to 4mips)
   NAUTILUS funds could be applied to interconnects and multiprocessing software
2. We do not feel that the bandwidth of the BI is adequate to compete with P-E and SEL who continues to pick away at our market with their current high bandwidth products. In particular where our customers are beginning to utilize low cost wide band satillite networks(40MBps) we are designing to the bandwidth requirements of rotating storage.

I don't feel your previous responses answered my questions and if LDP has a product that fits in the midrange and is not a new product but a cost reduced VENUS or a transparent SCORPIO multiprocessor similar to the ATLAS implementation at the risht price it will fill our needs to provide flexible network configurations made up of workstations and shared resources to solve the department level requirements.

12-AUG-81 10:39:55 S 31620 MLDP

"CC" DISTRIBUTION:

*GORDON BELL	BERT BRUCE	BILL DEMMER
DAVE DENNISTON	WAYNE FURMAN	JOHN GIUDICE
SUSAN JANCOURTZ	PETER MASUCCI	DON MCINNIS
PETER MOSTECKI	TOM NICHOLS	JOEL SCHWARTZ
MICHAEL SMITH	DICK STRAUSS	FRED WILHELM

Nautilus

****
* d i g i t a l *
****

TO: BRIAN CROXON
DON MCINNIS
cc: see "CC" DISTRIBUTION

DATE: THU 14 MAY 1981 6:51 EST
FROM: GORDON BELL
DEPT: ENG STAFF
EXT: 223-2236
LOC/MAIL STOP: ML12-1/A51

SUBJECT: NAUTILUS: THE NEXT STEP

Let's set about asap to get the maximum learning from Venus, Rezac via the Japanese experience, Stanford S1 et al, and Comet vis a vis the CAD tools. The presentation and approach was now in retrospect clearly out of the dark ages. I believe the VLSI folks under the sponsorship of their managers are spending money in the right areas to get out of the black hole of complexity that the technology is allowing us to enter. No way, do I believe we will be able to complete the Nautlilus design, given the emphasis of the design team, and the status of the design tools. Supnick is probably the best one to start teaching this internally at this time, and perhaps Ron Melanson.

The approach has to be get the tools and discipline NOW, while the technology is being looked at, and then start the design using the tools. This may delay the design start by a year, but I'd expect the output at least two years earlier than with the approach taken.

Therefore, I'd like to see the addition of some VERY strong system programmer, right now who'll be part of the main team to carry out this function, and keep the design top down and structured.

I think we are all tired of getting into these projects and finding that the bookkeeping complexity and details of non-connected parts, etc are killing us. Comet, Venus were both this way, and Fonz ended up this way. NOW is the time to Change! We have to head for the 2 year product cycle, there is nothing stopping this, provided you have the right tools!

Let's have a session after you have the person and when you're ready to set the goals about how the design will be done.

I'd also like to interact while you're grappling with this problem too. I hope we can get together very shortly.

While it's too early to tell the results, Scorpio is being designed in this fashion. There may be the best place to start using Supnik as a guide.

This is not an optional program. No reform; no design. Am tired of seeing designers disappearing into a black hole only to reappear if they do, very tired, old and grey. There aren't that many good ones around to enter the holes.

"CC" DISTRIBUTION:

BILL DEMMER	SAM FULLER	ROY REZAC
BILL STRECKER	STEVE TEICHER	

****
digital
****

TO: BRIAN CROXON
DON MCINNIS
cc: see "CC" DISTRIBUTION

DATE: THU 14 MAY 1981 6:51 EST
FROM: GORDON BELL
DEPT: ENG STAFF
EXT: 223-2236
LOC/MAIL STOP: ML12-1/A51

SUBJECT: NAUTILUS: THE NEXT STEP

Let's set about asap to get the maximum learning from Venus, Rezac via the Japanese experience, Stanford S1 et al, and Comet vis a vis the CAD tools. The presentation and approach was now in retrospect clearly out of the dark ages. I believe the VLSI folks under the sponsorship of their managers are spending money in the right areas to get out of the black hole of complexity that the technology is allowing us to enter. No way, do I believe we will be able to complete the Nautlilus design, given the emphasis of the design team, and the status of the design tools. Supnick is probably the best one to start teaching this internally at this time, and perhaps Ron Melanson.

The approach has to be get the tools and discipline NOW, while the technology is being looked at, and then start the design using the tools. This may delay the design start by a year, but I'd expect the output at least two years earlier than with the approach taken.

Therefore, I'd like to see the addition of some VERY strong system programmer, right now who'll be part of the main team to carry out this function, and keep the design top down and structured.

I think we are all tired of getting into these projects and finding that the bookkeeping complexity and details of non-connected parts, etc are killing us. Comet, Venus were both this way, and Fonz ended up this way. NOW is the time to Change! We have to head for the 2 year product cycle, there is nothing stopping this, provided you have the right tools!

Let's have a session after you have the person and when you're ready to set the goals about how the design will be done.

I'd also like to interact while you're gropping with this problem too. I hope we can get together very shortly.

While it's too early to tell the results, Scorpio is being designed in this fashion. There may be the best place to start using Supnik as a guide.

This is not an optional program. No reform, no design. Am

tired of seeing designers disappearing into a black hole only to reappear if they do, very tired, old and grey. There aren't that many good ones around to enter the holes.

"CC" DISTRIBUTION:

BILL DEMMER	SAM FULLER	ROY REZAC
BILL STRECKER	STEVE TEICHER	

SJ 1-34

digital

INTEROFFICE MEMORANDUM MAY 26 1981

TO: Gordon Bell

DATE: 21 May 1981
FROM: Steve Jenkins
DEPT: Adv. VAX Systems Eng.
EXT: 247-2395
LOC/MAILSTOP: TW/C04

CC: Frank Bomba
Brian Croxon
Bruce Delagi
Bill Demmer
Sam Fuller
Don McInnis
Steve Rothman
Bob Stewart
Bill Strecker
Pat White

SUBJECT: IEEE P896 BACKPLANE BUS

The IEEE P896 BI looks like an extremely attractive bus structure for low end systems. It certainly goes much further than the IEEE 796 (Multibus) did in meeting the requirements of systems in the 80's. In fact, it has reconfirmed my belief that DEC's BI has the appropriate functionality for future systems since the P896 design group is proposing essentially a similar structure.

A comparison of the P896 vs. BI data transfer characteristics follows:

	P896	BI
Data Path	32 bits (multiplexed address/data)	32 bits (multiplexed address/data
Transfer Type	Asynchronous	Syncronous
No. of Signals	66	53
Transfer Size	4, 8 bytes	4, 8, 16 bytes
No. of Drops	32	16
Length	19.6 in. (plus cable at reduced performance)	24 in. plus 3 ft. cable
Data Integrity	ECC, Command/Data confirmation	Parity, Command/ Data confirmation
Maintainability	Jumper Address Selection? No Master Identification. Error logging difficult.	No. Jumpers on Module. Bus Master ID on bus. Error logging easy with synchronous bus.
Performance	1/2 BI?	Up to 13 MB/sec.
Cost	Multiple VLSI chips?	Single VLSI chip.

The P896 Interrupt/Multiprocessing mechanisms are very similar to the BI except they are implemented using a 2 wire serial line appendage to the bus. Like the BI the P896 has processor to processor directed interrupts, broadcast mechanisms, and capabilities of assigning resources to a processor or set of processors.

Conclusions:

The P896 offers no better performance/functionality than the BI.

The P896 is about 1.5 years behind the BI in it's specification and probably has a slower design gestation period due to the politics of creating an industry standard.

The P896 will probably cost more than the BI. This is hard to access (i.e. I may be wrong) since the cost of multiple higher volume components may be less than a single medium volume component.

The P896 as an industry standard will cause a rich set of highly functional, cost effective devices to be available.

Recommendation:

Based on the fact that the BI is a more suitable structure for the broad range of system requirements (SCORPIO, NAUTILUS, SATURN) in the dimensions of cost, performance, and functionality and the fact that DEC will have an LSI interface to the BI available significantly sooner then P896 LSI interfaces, I recommend that we continue to develop the BI for our next generation of VAX systems. In addition, I feel that an industry standard bus for the backbone of our VAX systems would be a bad business decision in that it opens us up to all sorts of add-on business. Licensing the BI will allow us to control who designs devices to our system in such a closely coupled manner.

Request:

The advanced Scorpio program is critically dependent on the BI (cost & schedule); Nautilus needs the performance & functionality of the BI; Development resources are not available for multiple bus type adaptor projects; significant investment will be made in FY82 for the BI.

I need clear direction from you as to whether or not we should commit the BI to our next generation of VAX systems. If committed, I need your support to keep the necessary forward momentum going over the next two years to achieve my stated goals and deliverables.

Bill Demmer, Steve Jenkins, Dan McNamara, Bob Stewart

Status Report on the P896 Backplane Bus

cc: Sam Fuller, Pat Stratton, Pat White, Shanin, Amrodoon.

Tachi Mackean.

Given the IEEE standardization of the P796 (Intel Multibus) what's the chance of this one passing? F/U 5/22

Andrew A. Allison

Chairman, P896 Subcommittee

IEEE Computer Society Microprocessor Standards Committee

Why not? / Let's use it, NOT BI! Good

This report on the activities of the P896 (Advanced Microcomputer System Backplane Bus) Subcommittee of the Computer Society's Microprocessor Standards Committee describes the origins of the subcommittee and the status of the proposed specification as of its December 11, 1980 meeting. It must be emphasized that, while representing the fruits of many months of effort on the part of a joint US/European working group, the specification is incomplete and subject to significant change. This draft provides a preview of the general form of the functional specification and is intended to elicit technical response and to solicit participation in the work of the subcommittee. It is based on the working document for the P896 workshop held at the National Bureau of Standards facility in Boulder, Colorado on January 19-21, 1981. A summary of the results of that meeting is planned for the next issue of IEEE Micro. ← How'd this go?

A subcommittee on microprocessor standards was set up by the IEEE Computer Society in August, 1977. By the middle of 1978, the committee's efforts toward developing standard specifications for the S-100 (P696) and Multibus* (P796) buses had made clear the need to consider future system bus requirements before the emergence of yet another generation of *de facto* but incompletely specified and incompatible buses.

The working group set up to consider this need concluded that the buses then being specified by the Microprocessor Standards Committee could not be extended to satisfy the requirements anticipated for future microprocessor-based systems. Three major categories of bus—backplane, local network, and residential—were identified. A backplane bus subcommittee was set up in June, 1979, and Project Authorization Request Number P896 was approved by the IEEE Standards Board in September of the same year. EDISG—the European Distributed Intelligence Study Group—set up a subgroup in May 1980 to interact with the IEEE work. EDISG is one of the working groups supported by the Commission of European Communities for promoting standardization in the field of data processing.

P896 is intended to be a manufacturer- and processor-independent bus offering 32-bit multiplexed address and data paths, while also supporting 16- and 8-bit data paths. Fully handshaked bus transfers are expected to be supported, and distributed bus arbitration provides for at least 32 bus masters. Multitask operation is facilitated by the provision of a serial interprocessor link that also incorporates interrupt arbitration. Recommendations will be given for operating system compatibility. IEC standard mechanical specifications are proposed for modules, backplanes, and racks. In recognition of the high overall system cost associated with each signal path, every effort is being made to reduce pin count.

Bus timing and control are specified in such a way as to facilitate increased performance as interface technology permits, with an initial maximum clock rate in excess of 10 MHz. If electrical and mechanical interface constraints are ignored, the upper limit on clock frequency is anticipated to be governed by four end-to-end bus propagation delays plus a limited number of gate delays.

Information on the status of the P896 activity may be obtained from:

Andrew Allison, Chairman
P896 Working Group
27360 Natoma Road
Los Altos Hills, CA 94022

or

Prof J.-D. Nicoud, Vice-Chairman
LAMI-DE-EPFL
Bellerive 16
CH-1007 Lausanne (Switzerland)

*Multibus is a registered trademark of Intel Corporation.

0272-1732/81/0200-0067$00.75

****
digital
****

TO: JOE CARCHIDI
*BILL DEMMER
SAM FULLER
BILL HEFFNER
BILL JOHNSON

DATE: SAT 2 MAY 1981 15:29 EST
FROM: GORDON BELL
DEPT: ENG STAFF
EXT: 223-2236
LOC/MAIL STOP: ML12-1/A51

SUBJECT: LOW COST VAX'S

We are having a hell of a problem in getting the 11's down in cost at the systems level because we are unable to take advantage of the set of component chips available. The ii is an attempt at the hardware level to interface these.

There is a similar problem for software busing. These new peripheral chips are totally different and require new handlers and a way of interfacing to our operating system structures.

We are going to have to bring about some major changes in the way we cope with this hardware from the semi folks.

Currently, we can't use it. If we don't we are going to be uncompetitive.

VAX will have the same problem as Lloyd points out.

I want the BI rethought out. I don't see how we can use it. I would like to use IIndustry Standard chips and provide an Industry Standard Board level bus such as the new, emerging 32 bit version of the Multibus that the IEEE is proposing. No way can we continue to ignore these standards.

ATTACHED: MEMO;62

****
* d i g i t a l *
****

Sent ✓

TO: see "TO" DISTRIBUTION

cc: see "CC" DISTRIBUTION

DATE: FRI 14 AUG 1981
FROM: GORDON BELL
DEPT: ENG STAFF
EXT: 223-2236
LOC/MAIL STOP: ML12-1/A51

SUBJECT: COMMENT ON NAUTILUS PLAN

The 32-bit program office has to do a better job of measuring cost/performance of systems. The overall planning targets and understanding is nil. Price performance is plotted on linear paper, with no actual rates. What are they?

Although price and performance are ok, the gestation time of 4-5 years is ridiculous! (We must have 26%/year improvement when compared with either the 780 or 750.) We aren't going to be able to compete with either Japan, or IBM, nor is it adequate when compared with CDC!

We need vectors added to VAX! Note the preponderance of scientific users. Why can't we add them into the architecture now? When?

Is there anyway to design a machine that can have a planned mid-life kick in terms of better gate arrays? The Japanese use re-implementing machines in different technology to get more out of their engineering investment.

The TI gate array is really unimpressive and looks like a bitch (expensive to design) to use for such a powerful machine. The Japanese may have better parts for higher performance. Shouldn't we look at them for Nautilus?

Overall, the background work is good, especially the cad and design thoughts.

I'm scared generally about planning and project control of what will be a large project (the size in gates of Venus). I don't believe the project is necessarily complex, but is a result of our designers trying to build a minimum cost system using marginal component technology. Why? Would a better way to operate be to state constraints on board and gate array densities, automatic design of gate arrays, and CAD? Can that we have a strong process (CAD, physical implementation including test equipment)? Is this Jeff's sole role? How does he operate with other TW services? It's early to operate with Manufactuirng, but given our poor interface and 1-2 years introduction times, this is a problem too.

"TO" DISTRIBUTION:

AL MCGUIRE DON MCINNIS STEVE JENKINS

"CC" DISTRIBUTION:

BRIAN CROXON
JOHN O'KEEFE

BILL DEMMER
BILL STRECKER

SAM FULLER

750 CPU DEVELOPMENT

MACHINE BLOCK DIAGRAM
- ESTIMATE OF GA & BOARD COUNT

PROBE 1ST WAFERS

1ST ARRAY SLOW

MOST GAs IN LAYOUT

1ST GA SPEC

REV B GA SLOW

BB DEBUG STARTS

LAST PROTO PCB INTO LAYOUT

ALL BB CHIPS AVAIL

WEN GA SPEED PROBLEM

PROTO RUNS UETP AT SPEED

BUILT 28 PROTOS

DMT COMPLETE FVS

DMT 1/3

Comet gate start 6/76

1976	1977				1978				1979				1980			
10,11,12	J,F,M	A,M,J	J,A,S	O,N,D	J,F,M	A,M,J	J,A,S	O,N,D	J,F,M	A,M,J	J,A,S	O,N,D	J,F,M	A,M,J	J,A,S	O,N,D
Q277	Q377	Q477	Q180	Q280	Q380	Q480	Q181	Q281	Q381	Q481	Q180	Q280	Q380	Q480	Q181	Q281

MOVE TO TEWKSBURY

BB WIRELIST TO KANTA

RELEASE ALL GA FOR PROTOS INTO LAYOUT

ALL PG TAPES FOR PROTOS SENT OUT

POWER ON PROTO #1
DISCOVERED LAST SPEED PROBLEM

NEW ETCH FOR SPEED FIXES SIGNED OFF

GA NOT CHARACTERIZED

CAD TOOLS FOR PCBs & GAs BREAKING

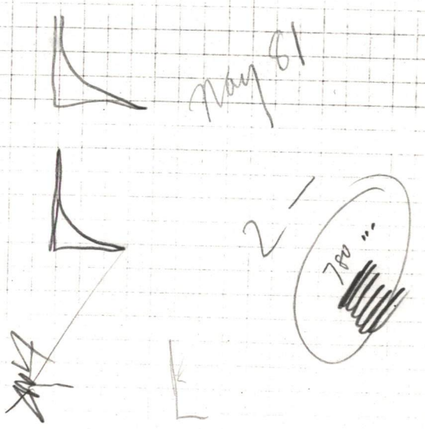
BREAKDOWN IN CAD SYSTEMS AFTER MOVE

TURNAROUND TIME FOR GA & PCB LAYOUTS

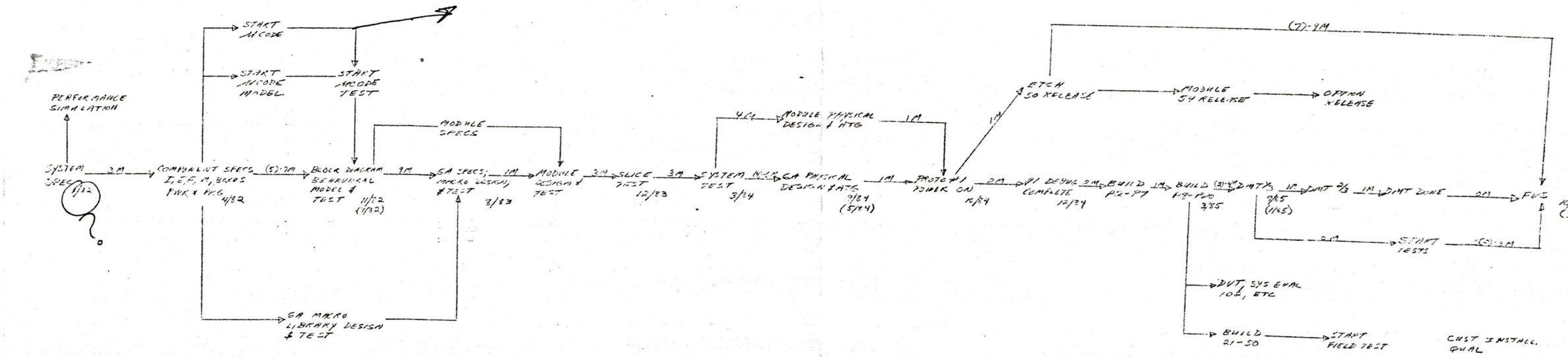
WORCHESTER UNPREDICTABLE

AXE NOT IN PLACE FOR 750 UNTIL LATE 1980

FAA & MBA FUNDING ISSUES = SCHEDULE SLIP



DJM 12/31



NOTES:

① ALL MILESTONES ARE EVENT COMPLETE & DATES ARE MONTH END
② TEST PRIOR TO PHYSICAL DESIGN IMPLIES SIMULATION & TIMING VERIFICATION
③ OVERLAP IN START OF MILESTONES ARE NOT SHOWN
④ SCHEDULE IS DIRECTLY COUPLED TO FUNDING LEVEL

NAUTILUS CRITICAL PATH NETWORK OVERVIEW
2/27/81 DJM

Mauzlus

****
* d i g i t a l *
****

TO: BILL DEMMER

cc: see "CC" DISTRIBUTION

DATE: WED 2 DEC 1981 9:19 PM EST
FROM: GORDON BELL
DEPT: ENG STAFF
EXT: 223-2236
LOC/MAIL STOP: ML12-1/A51

SUBJECT: RE: NAUTILUS ACCELERATION

I love it and will support it.

Would like to review just how you intend to pull it off in that time frame, if you use many custom gate arrays. Since I'm going away for 3 weeks December 20, please make sure we can get together somehow for a few hours and exchange views. (My worries about complexity and gate arrays, and how you are going to get the product out in that time frame.) If necessary, let's do it over dinner.

Great!

"CC" DISTRIBUTION:

BOB STEWART @TWSK
SAM FULLER
STEVE JENKINS @TWSK
BRIAN CROXON
DON MCINNIS
BILL STRECKER
MARY JANE FORBES
JOHN O'KEEFE

****
* d i g i t a l *
****

TO: *GORDON BELL

cc: see "CC" DISTRIBUTION

DATE: WED 2 DEC 1981 12:39 PM EST
FROM: BILL DEMMER
DEPT: 32 BIT SYSTEMS
EXT: 247-2112
LOC/MAIL STOP: TW/D19

SUBJECT: NAUTILUS ACCELERATION

After a review of several alternatives I would conclude the following is our best choice for the Nautilus program:

- Accelerate Current Design to Ship Q2 FY85 (3 years)

  2.5 x 780 Perf @ 750 Cost

- Include MP (or attached processor) capability at

  FCS (to allow Nautilus to serve as Venus backup)

- Requires additional $1.5M devel. expense in FY83

The other major alternative discussed was the use of discrete 100K ECL which would yield a 5 plus times 780 perf at a substantial cost increase over the 780. While this would key off the basic structure of the Nautilus design it would require a redo of the individual functions within the processor, thus the time to market cannot be substantially improved over the above recommendation. I would not want this approach to be taken by the Nautilus group as it would leave a major gap in the Mid-range of the VAX product line.

/vjt
1.14

"CC" DISTRIBUTION:

BOB STEWART @TWSK
BRIAN CROXON
SAM FULLER
DON MCINNIS
JOHN O'KEEFE
STEVE JENKINS @TWSK
BILL STRECKER

Copy to Alan, Jud, Teicher, Rowe, Burdick, Bill Johnson (Hudson), Duane Dickhut, glorioso, Vehbi, Rezac, Sam, Gator Ray

From: HYDRA::MCINNIS 15-DEC-1981 14:50
To: ISHTAR::DEROSA
Subj: DESIGN METHODOLOGY DOC. FOR GORDON

This is great.

· automatic "state of design" via filling the documents.

– Can you go to 2 ga's vs 1 ga if problem
– Name
– auto design state collector?
process
what does the set of documents look like

TITLE: NAUTILUS DESIGN METHODOLOGY DOCUMENT

ABSTRACT: This document describes the design process to be used for the Nautilus product from the initial conception to the start of the construction of the first prototype. Key to this process is the substitution of functional and timing simulations for a breadboard.

STATUS: COMPANY CONFIDENTIAL

AUTHORS: Rob Rusik
Steve Jenkins

DATE:	REVISION:	CHANGE:
Dec. 2, 1981	P0	First draft
Dec. 14, 1981	P0.5	Section 3 Revised

Let me recommend all major products get to this state!

Gordon

1.0 OBJECTIVES

The purpose of this document is to describe "how" the design occurs and the tools available. Subsequent documents will describe the "how" details after design completion.

The design objective simply stated is to create an equivalent cost VAX 11750 replacement product at 2 to 3 times the VAX 11780 performance for customer availability by ~~the end of 1985~~ (June,1985).

A formal design methodology is expected to produce a predictable process and a high quality product. The process described within this document strives to establish a design discipline which is easy to follow, defines important checkpoints, yet does not overburden the designer with excessive rules and procedures. If the proper balance is acheived then an optimized time to market should result with the additional up-front time paying off with reduced number of redesign passes.

2.0 DESIGN PROCESS

The processes described below all have a common theme which permit a concept to be developed until it becomes too complex to maintain and is then partitioned into lower level concepts and so forth. The lowest level concept is one which can be readily implemented by a design engineer. At this point the pieces are integrated to form the whole and, if the proper steps occurred, the end product should be of high quality.

The decomposition of the problem into smaller bounded tasks allows distributed resources to work in parallel on the solution. These distributed resources can either be engineers or computers with the important point being that failure of one or a small number of resources can be compensated for more easily. However, if the integration of the pieces is to be successful, then the boundary conditions of the dispersed tasks must be rigidly defined.

2.1 Logic Design

The initial phase of the logic design for the CPU, FPU, and Memory occurs at the conceptual level without a great deal of attention being paid to the physical partitioning. At this point it is important to acheive an understanding of the functional operation of the system and then subsystems.

Once this has been acheived, then physical partitioning is done and specific design tasks are assigned. Detailed implementation occurs and as it progresses, the descriptions of the physical boundaries become more stable.

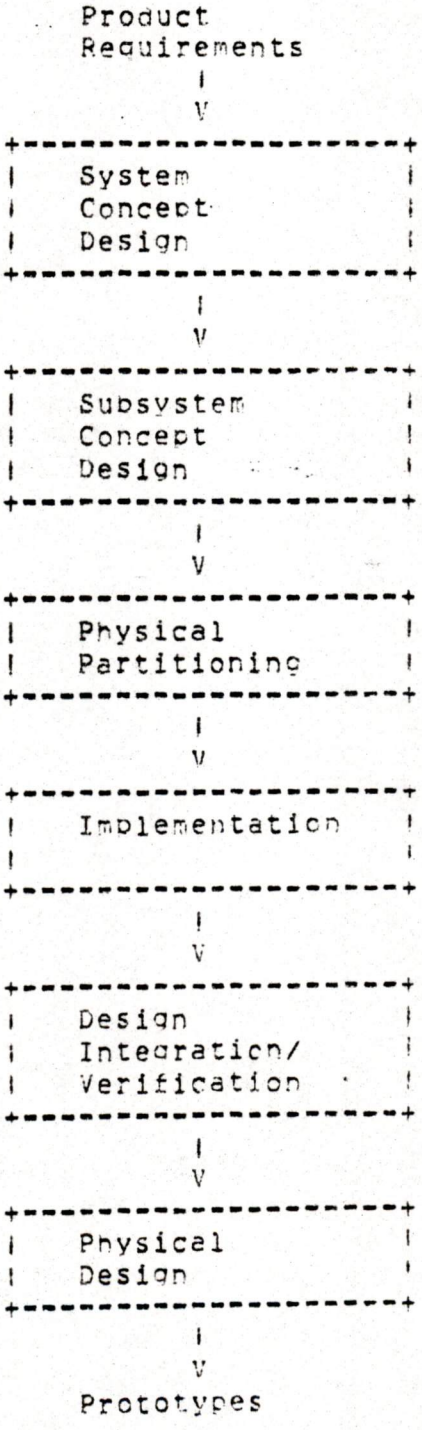
When the design is completed design verification tests are performed on a functional simulator starting at the lowest physical unit (GA) and successively building back up to the complete system. The use of a simulator as a debugging tool will eliminate the need of constructing a breadboard prior to prototypes. The modular approach to integrating the design elements should allow a smooth and therefore speedy debug cycle to occur since most faults are contained to a smaller area and uncovered in a very parallel way.

It is important to note that while working at any of the higher levels of design it still becomes necessary to "peek" at the more detailed levels to be able to make realistic estimates of timing considerations and design alternatives. For instance, the functional description of the execution box may require probe designing of a slice of the data path logic in order to acheive a reasonable feel about cycle time.

In the last stage, physical design occurs such as GA, Module, and Backplane layouts. Design changes which occur as a result of the physical design process are integrated back into the whole design in a controlled manner. An example of this would be logic changes required to compensate for longer

than predicted interconnect paths.

Logic Design Flow



2.1.1 System Concept Design

This portion of the design concerns itself with defining the overall functionality of the CPU, FPU, and Memory; with predicting cost and performance; and with describing the adaptors and attachments which make up the I/O section of the system. During this time the major technologies will be selected and the design theme established.

Accomplishments:

1. System Specification drafted
2. Design requirements for the I/O system sent to the appropriate engineering groups.
3. Boundary conditions described such as RD port, SI, CI, NI,etc.
4. System block diagram and machine flow determined.
5. Performance simulation complete.
6. Product Requirements Not Met justification document drafted.

Tools:

1. Graphics Editor for diagrams.
2. Text Editor for specifications.
3. Performance simulator for first order performance estimate.

Reviews:

1. Internal review of System specification, followed by Phase 1A review after cleanup.
2. Internal review of Product Reqs. Not Met document, followed by review with product lines and external groups, followed by Phase 1A review.

2.1.2 Subsystem Concept Design

The major sections of the system are broken apart and assigned to team leaders for more detailed analysis. Physical partitioning constraints are now becoming clearer. Refinements are made to estimates of product cost and performance. Non-physical boundaries are established such as internal register bit assignments, traps and interrupt vectors, etc.

Accomplishments:

1. Subsystem specifications drafted which include complete functional descriptions such as bits in registers, equations for condition codes, etc.
2. Detailed block diagrams, flows, and timing budgets done.
3. Ucode control word defined.
4. Performance simulations completed.
5. Macro logic design library established.

6. Design rules written.
7. Interconnect catalogue (Net List CREF) established with the major interconnects within and among the subsystem units listed and their characteristics described.

Tools:

1. Graphics editor for diagrams.
2. Text editor for documents.
3. Performance simulator with improved system model.

Reviews:

1. Subsystem specifications reviewed internally, followed by review with external groups after cleanup.

2.1.3 Physical Partitioning

Partitioning of the functional units is formalized and design tasks are assigned to individual implementors within the design teams. Major partitions are Gate arrays, Modules, and Backplanes. The theme hereafter is to design with partitions in mind.

Accomplishments:

1. Short functional descriptions are written for each of the GA's and Modules.
2. Parts count is estimated.
3. Cost analysis is refined.
4. The Interconnect Catalogue is updated with the major interconnects among GA's and Modules listed and their characteristics described.

Tools:

1. Text Editor for documents.
2. Graphics Editor.

Reviews:

1. Internal review of GA and Module descriptions by system architect with applicable design team and other team leaders.

2.1.4 Implementation

Macro design library, Component Catalogue, and Interconnect Catalogue are updated in a controlled manner as the design progresses through to completion.

The design is accomplished using macro gate descriptions for commonly used functions and customized gate configurations when certain optimizations are required. Macro designs will be done by specific designers for use by the CPU designers. The two level

design concept is analagous to programming in high level language with specialized routines (for speed or size) done in machine language.

Timing Verification, Loading Analysis, and Testability tools are used during the implementation phase to assist engineers in assessing the design feasability. All timing values of interconnect delays are estimates based on statistical inference from test GA layout samples. Fully automatic trial layouts will be performed by the design engineers to establish layout feasibility and to more accurately estimate interconnect delays.

All details of the design are completed with more accurate predictions of performance and cost established. Design consistency among the seperate teams is checked via design review forums.

Accomplishments:

1. Design prints are completed with timing and functional descriptions updated.
2. Initial timing verification is completed.
3. Parts lists are created.
4. Failure rate analysis is completed.
5. Power consumption calculations are completed.
6. Testability of GA's and Modules is analysed.
7. Loading analysis is complete.

Tools:

1. Graphics editor for prints.
2. Text editor for documents.
3. Timing Verifier as a design aid.
4. Testability Analyzer
5. Power and Failure rate calculation tool used on parts lists.
6. Loading Analysis tool.
7. Trial Layout and Interconnect Delay Analyzer

Reviews:

1. Internal design reviews/tutorials are held with the design teams.

2.1.5 Design Integration/Verification

The process of building together the system from the individual design pieces begins now. Testing before integration occurs starting with the individual Gate Arrays, progressing to the Modules, then to the Subsystem, and finally to the system level. Release procedures from stage to stage are determined.

Hand generated test cases are developed for each stage which

cannot be stimulated by firmware or udiagnostics. If applicable micro-instructions will be used as the test vehicle at the subsystem level.

A progression of test will be performed at the system level. This will start with udiagnostics to macro level diagnostics, and then to AXE (Architectural Exercisor) tests to demonstrate conformance to the VAX ISP. The amount of test cases run will depend upon the performance of the simulator being used and the computer resources available.

Accomplishments:

1. Increased confidence in the design is established.
2. GA specifications are compiled by combining existing functional descriptions with gate level models and timing verifier output.
3. Module specifications are compiled by combining existing functional descriptions with gate and behavioral models and timing verifier output.
4. Logic and ucode designs are merged.

Tools:

1. Graphics editor for design changes.
2. Functional Simulator working with mixed mode of gates and behavorial models.
3. Timing Verifier for final analysis using estimated interconnect delays.
4. Testability Analyzer if applicable.
5. Power and Failure calculation tool against final parts lists.
6. Text editor for documents.

Reviews:

1. Formal Design Reviews with external groups.
2. Document/file reviews with TI for GA option submission.
3. Document/file reviews with design services for GA option and module submissions.

2.1.6 Physical Design

GA's, Modules and Backplane etchboards are submitted for layout. Parts are kitted for early prototypes. Logic prints and Parts lists are placed under a revision control system.

Actual interconnect timing values are fed back to the design data base and used for final timing verification. Changes are made to correct inconsistencies. Functional simulation is performed on any design unit which changes. Design unit is defined as including all physical design segments within interconnect boundaries which are unaltered as a result of the change.

Accomplishments:

1. Final Timing Verification is completed.
2. Logic prints, Timing diagrams, Parts lists, etc. are updated and formally released.
3. Final hypothesis is made for failure rate.
4. Physical designs are completed.

Tools:

1. GA placement and routing tools.
2. Module placement and routing tools.
3. Layout Editor.
4. Design Rules Checker.
5. Interconnect Verifier.
6. Timing Verifier.
7. Interconnect Delay Analyzer.
8. Functional Simulator.
9. Loading Analysis Tool.
10. Graphics Editor.
11. Power and Failure Rate calculation tool.
12. Test Program Generation Tools.
13. Text editor.

Reviews:

1. Internal Readiness review to determine go/no-go for prototype fabrication.

2.2 Firmware Design

The early firmware development will support the ideas being developed for the System specification for machine organization and performance estimates.

During the Definition phase the work is partitioned into manageable tasks. The major partitions are kernal code (i.e.,memory management,traps,interrupts,operand specifier decoding) and instruction specific code. Coding standards and macros are defined to establish consistency in the design.

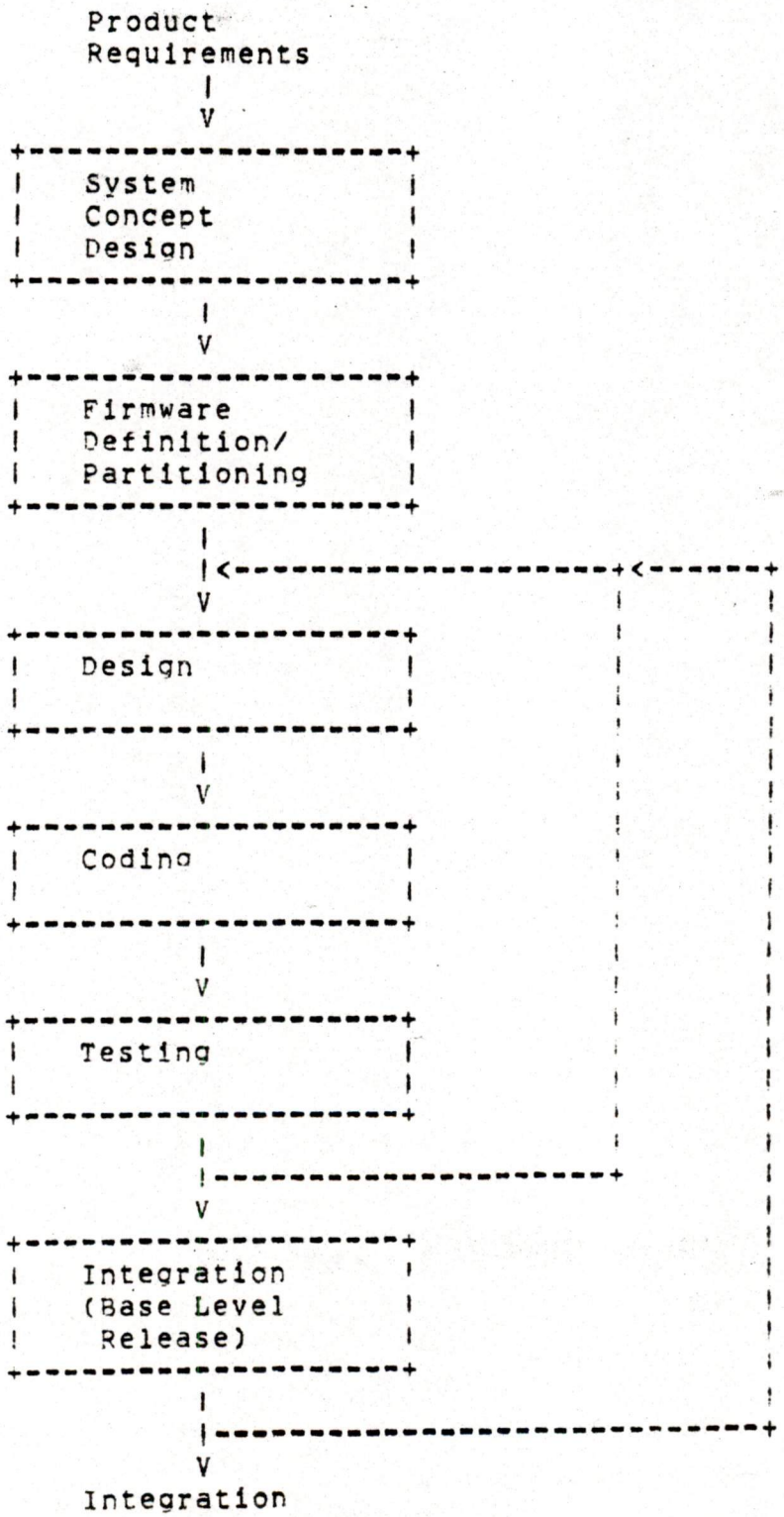
In order to ensure the project as a whole is progressing on schedule or to identify as early as possible any potential problem areas, there will be a scheduled series of internal base level releases. Testing of these base level releases will be performed on a behavioral model of the machine.

Each base level will involve design, coding, and testing cycles by the individual programmers prior to integration into the larger body of code. Code reviews (walkthroughs) will occur to ensure conformance to standards, completion of documentation and testing.

Integration of the code modules take place as the final activity of a base level release. Naming conflicts are identified and resolved and additional testing occurs using some hand generated test cases, but mainly with AXE generated test cases.

At the completion of the last base level release the firmware is ready to be integrated with the detailed hardware design. Success at this point will be determined by our ability to maintain tracking between the evolution of the hardware and firmware via joint reviews and thorough documentation.

Firmware Design Flow



2.2.1 System Concept Design

The firmware effort during this phase will be direct support of the hardware concept development. Hardware/firmware tradeoffs will be understood and performance predictions established via probe coding of instruction routines.

Accomplishments:

1. General firmware flows identified.
2. Initial performance estimates established (i.e.,number of cycles).
3. Input to system specification completed.

Tools:

1. Text Editor for documents.

Reviews:

None.

2.2.2 Firmware Definition/Partitioning

The definition phase is that in which the design is partitioned into a series of manageable tasks with standards and correctness criteria established. Partitioning is at the function and instruction group level and may be refered to as code modules.

Coding standards that embody the concept of "coding for the reader" will be established to ensure that the firmware is maintainable by others.

Performance estimates are refined and an initial code size is postulated. More detail is understood as to instruction processing. A Macro definition file is started with a control process for changes. Initial boundary interfaces for interaction of the code modules is defined.

Accomplishments:

1. Firmware specification drafted which defines coding standards, field definitions, Macro definitions, code module functional descriptions, and testing criteria.
2. Firmware control word defined.
3. Global code flows established.

Tools:

1. Text Editor.
2. Micro-Assembler

Reviews:

1. Internal review of specification by system architect, code design team, and hardware team leaders, followed by review with external groups after cleanup.

2.2.3 Firmware Design

Detailed specifications for code modules are created which define the algorithms down to the micro-operations level, but not the microword level. Interfaces to kernal code are formalized with hardware features assumed to be stable.

Individual programmers wil likely cycle through the design/code/test sequence for subsections of their coding task in a base level release.

Accomplishments:

1. Instruction algorithms and kernal code interfaces known
2. Firmware specification updated.

Tools:

1. Text Editor
2. Micro-Assembler

Reviews:

None.

2.2.4 Coding

Detail implementation of the algorithms to the microword completed in conformance to the specification. Individual instruction performance is finalized. Macro definitions are evolving and test cases are coded.

VMS software interface boundaries are defined such as exception parameters, etc. as a result of kernal code completion.

Accomplishments:

1. Code module design completed.
2. Instruction performance known.
3. Software interface known.

Tools:

1. Text Editor
2. Micro-Assembler
3. Code Management System

Reviews:

None.

2.2.5 Testing

Demonstration that code meets correctness criteria by individual programmers using hand generated test cases and AXE generated instruction tests are performed. Test files for base level testing are created.

Accomplishments:

1. Doneness criteria for code integration is completed.

Tools:

1. Text Editor
2. Micro-Assembler
3. Code Management System
4. Behavioral model simulator with debugging features installed
5. Architectural verification tool

Reviews:

1. Internal design reviews/tutorials are held with the design teams, followed by external design reviews with VAX experts.

2.2.6 Integration (Base Level Release)

Code modules are now integrated into the main code body and placed under revision control. Previously generated test cases and AXE test cases are used to validate the base level release. Additional test methods include: micro PC traces to ensure all microwords are executed and, for a given operation, that all paths have been taken, stress testing (fault testing) throughout the code body, and assertion preprocessing or the ability to make certain assumptions about the state of the environment at a given body of code and to test that indeed the expected state is achieved.

Testing of the base level will continue in parallel with the development of the next base level in order to accumulate as many AXE test cases as possible. It is believed that as many as 100,000 cases are needed to adaquatly verify an entire VAX firmware implementation. This testing will overflow into the hardware testing and be greatly accelerated due to the increased speed over the simulator.

Several base level releases are planned in order to better understand the design progress during the project. The completion of the last base level release will produce tested firmware which will then be integrated with the hardware and system testing occurs using a hardware gate model.

Accomplishments:

1. Increased confidence in the design is established.

Tools:

1. Text Editor
2. Micro-Assembler
3. Code Management System
4. Behavioral model simulator
5. Architectural verification tool

Reviews:

None.

2.3 Console Code Design

2.4 Microdiagnostic Design

2.5 Power System Design

2.6 Mechanical Design

3.0 Design System

The CAD system which will support the NAUTILUS design methodology, technology and product development will be known as NEPTUNE. This section describes the NEPTUNE architecture and identifies the tools which will be used as part of this system.

3.1 Goals

Establish a production-worthy INTEGRATED system of CAD tools for the development of gate array-based systems.

Drive this CAD system development by NAUTILUS product development demands.

Preserve the investments in CAD tools, systems and expertise beyond NAUTILUS product development.

Support the introduction of new methodologies; Structured Logic Design, Rigorous Specifications(?), Hierarchical Simulation, Timing Verification and Level Sensitive Scan Design.

Establish a CAD system architecture which will enable us to ennance, add and retire individual tools over time.

Improve the predictability of the physical design process by incrementally establishing feasibility.

Provide as clean and consistant a user interface as possible given the diverse sources of tools.

Be predictable => NO SURPRISES!

The overall goal is to make the designer's job easier and to bring a higher quality design to the marketplace faster.

3.2 NEPTUNE Architecture

The architecture of NEPTUNE can be described in terms of; management of design data, how the user will deal with CAD processes and a taxonomy of the tools which will be used. Obviously, this architecture must be viewed from the perspective of the design methodology described in this document. Although a specific computing environment and set of tools are identified in the following sections, the architecture must enable us to evolve these specifics without major disruption of the users.

3.2.1 Design Data Management

During the development process, and well into production, design data represents the family jewels. NEPTUNE assumes that it will have to accomodate tools which require and produce data in formats which are not under our control. Accepting this requirement, design data management (which might more accurately be called design file management) boils down to:

naming, storing and accessing files of data who's format is tool driven

structuring relationships among files to reflect structure within the design (e.g. logical and/or physical hierarchy)

controlling ownership, access and version history of files

providing global and local library access and maintainance control

Today, hierarchical design is in vogue and the Design Data Manager (DDM) must support it. The hierarchy may reflect logical structure, physical partitioning or a combination. Some parts of the design may have a greater hierarchical depth than others. The current NAUTILUS design hierarchy is still being discussed.

The NEPTUNE DDM (Design Data Manager) will be layered on the VMS file system. It draws heavily from CHAS and we may be able to use major portions of the CHAS code if we can establish a satisfactory process for providing highly responsive support and evolution. CHAS views the hierarchy as a structured set of BLOCKS. All blocks are the same in their potential to have associated sets of files. A BASE DESCRIPTION, common to all blocks in the hierarchy, establishes a template identifying the attributes (types of data files) which can be associated with any block. For each block, an instantiation of the base description is essentially a set of attribute name, value (list) pairs which indicate what data exists for that block and where it can be found. Base descriptions for each block in the hierarchy are accessed via an ISAM (Index Sequential Access Method) file. All of this sounds overly complicated: maybe an example would help. Suppose I wanted to run the Timing Verifier on the ALU of the E Box. A command procedure (discussed in Section 3.2.2) operating on my behalf would retrieve the appropriate net list by locating the base description of the NAUTILUS.E_Box.Data_Path.ALU (my hierarchical path name) in the table (ISAM file) of base descriptions. It would then look for the attribute of CURRENT_NET_LIST and find associated with that attribute the actual name of the file containing the net list. We will return to this example and extend it in the section on command procedures. Specific attributes (data types) are discussed in Section 3.2.4.

This approach makes it easy to establish structure among blocks; the base descriptin can contain attributes of PARENT(S) and CHILDREN. In fact, it would be easy to deal with several structures although I'm not quite sure how one would use such a capability. Adding new attributes (data file types) will be easy and nondisruptive. The user doesn't have to worry about where the actual data is stored, an especially attractive feature in a hetrogeneous network (see Section 3.3). Hacks, like limitations on file name length, will be hidden from the user. And finally, but very important, it is not anticipated that there will be much of a performance cost for all of this good stuff.

3.2.2 Command Procedures

Command procedures can be thought of as (small) programs which combine CAD tools into CAD processes. Their purpose is to simplify the user's interaction by buffering him from detail and change within NEPTUNE. Using the DDM, a command procedure locates the required data files and invokes a series of programs which may include; data extraction/derivation (e.g. create a current net list from the current schematics data file), format converstion (e.g. convert the net list to the 'new improved' format required by the Timing Verifier), actual tools (e.g. run the SCALD Compiler followed by the Timing Verifier) and store the results (e.g. Compiler and Timing Verifier Reports) via the DDM. Command procedures will also control how (and where) programs are run; within the user's current process (i.e. while he waits), as a subprocess (i.e. enabling the user to do something else while waiting) or submit as a batch job. Conditional capabilities within the command procedure enable it to make decisions (e.g. IF the schematics data file is more recent than the net list data file, THEN run the program which will derive a current net list data file).

Command procedures will be implemented by means of the VMS CLI (Command Language Interpreter). Although a standard set of command procedures will be provided, I would expect many of the users to modify, extend and create some of their own. This is just what happens with DCL (Digital Command Language) on VMS.

3.2.3 Tool Taxonomy

	Capture, Edit & Synthesis	Transformations	Simulation & Analysis
Conceptual (Architectural) Design	Text Editor Code Mgt System Graphics Editor		Performance Simulator
Physical Partitioning	Text Editor Graphics Editor Code Mgt System	Net List Extractor Net List Cross Reference	
Logical Implementation	Text Editor Graphics Editor Code Mgt System	Net List Extractor Net List Compiler Net List Cross Reference ROM & PLA Compiler	Timing Verifier Loading, Power & Failure Rate Analyzer Testability Analyzer Trial Layout & Interconnect Delay Analyzer Circuit & Transmission Line Analyzer
Microcode	Text Editor Code Mgt System	Microcode Assembler Behavioral Model Compiler	Behavioral Simulator
Integration & Design Verification	Text Editor Graphics Editor Code Mgt System	Net List Extractor Net List Cross Reference ROM & PLA Compiler	Timing Verifier Failure Rate Analyzer Testability Analyzer Trial Layout & Interconnect

			Delay Analyzer Logic Simulator
Test	Text Editor Auto (stuck-at) Pattern Generation Auto AC Auto DC Code Mgt System	Test Program Generation	Fault Simulator
Gate Array Physical Design	Auto Placement Auto Routing Layout Editor		Interconnect Delay Analyzer Design Rule Checker Interconnect Verifier
PC & Backplane Physical Design	Auto Placement Auto Routing Layout Editor		Design Rule Checker Interconnect Verifier

3.2.4 Data Types

Data Type	Source	Consumer
Text	Text Editor	Text Editor
Performance Simulator Model Source	Text Editor	Performance Simulator Compiler
Behavioral Model Source	Text Editor	Behavioral Model Compiler
Microcode Source	Text Editor	Microcode Assembler
ROM Source	Text Editor	ROM Assembler
PLA Source	Text Editor	PLA Assembler
Schematic (Block Diagram)	Graphics Editor	Net List Extractor Schematic Plot Extractor
Net List	Net List Extractor Hierarchical Net List Compiler	Hierarchical Simulator Timing Verifier Loading, Power & Failure Rate Analyzer Fault Simulator Auto (stuck-at) Pattern Generator Auto Placement Auto Router Interconnect Verifier Trial Layout & Interconnect Delay Analyzer Circuit & Transmission Line Analyzer
Simulator Stimulus & Response	Text Editor Simulator	Simulator Timing Diagram Plot

	Auto (stuck-at) Pattern Generator	Extractor Auto AC Auto DC Test Program Generator
Gate Array Layout	Auto Placement Auto Router Layout Editor	Layout Editor Interconnect Delay Analyzer Design Rule Checker Interconnect Verifier Layout Plot Extractor Tooling Post Processor
PC & Backplane Layout	Auto Placement Auto Router Layout Editor	Layout Editor Design Rule Checker Interconnect Verifier Layout Plot Extractor Tooling Post Processor
Plots	Schematic Plot Extractor Layout Plot Extractor Timing Diagram Plot Extractor	Plot Server

3.3 Computational Environment

We will clearly distinguish between those tools (or maybe command procedures => CAD processes) which are INTERACTIVE and those which are not. In each case, we will match the computing environment to the tool so as to meet the user's expectation in a PREDICTABLE fashion. To accomplish this, we must; accurately characterize the computational requirements of each CAD tool (or process), accurately characterize the demand schedule, provide the necessary computing environment and discipline ourselves relative to how we load the various resources.

I characterize interactive tools as those which are sufficiently responsive that the user's train of thought is not interrupted by computational pauses. Consistancy of response turns out to be almost as important as response time itself. Tools that fall into this catagory are basically editors; text, graphics and layout. For the bulk of interactions, users expect editors to respond instantaneously (i.e. < 0.5 seconds). Users are understandingly more patient for the smaller number of commands which require more significant computational activity (e.g. reading or writing text, schematics or layout files). Still, they tend to become distracted if the pause exceeds ten seconds. This threshold of tollerance obviously varies from user to user.

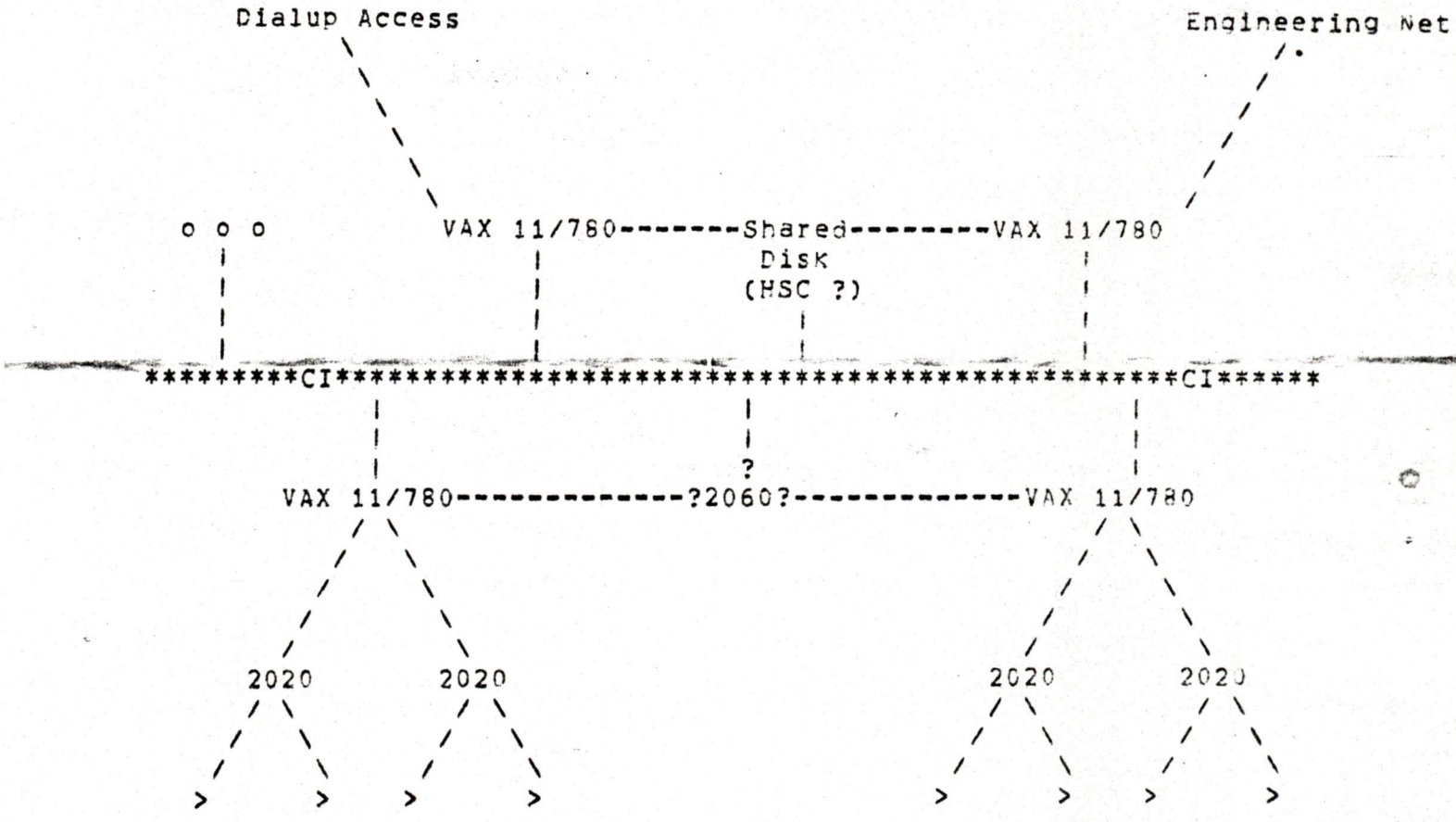
Programs which require minutes, hours, days or more to run will be considered BATCH jobs and the user's expectations set accordingly. Predictibility of turnaround is important for this class of jobs.

Idealy, one would like to have a single (kind of) workstation capable of running any of the interactive tools. Such workstations would be terminal nodes of a network which would provide; batch computes, access to design data, links to other networks and specialized peripherials such as electrostatic plotters. The user would view their workstation as an interactive window to the computational universe. Restricting interactive programs to run on workstations would enable us to match the tool to the (local) computational environment and thus achieve our goal of responsiveness. There are two things which make this ideal difficult to achieve in the near future; the cost of computer equipment and the many uncoordinated sources of tools.

Similarily, one would like to have all tools run on a single machine architecture, VAX/VMS. However, it is likely that we will have a number of programs which, for a number of reasons, cannot be conveniently make to run on VAX/VMS.

All of this brings us to a computational environment based on a VAX/VMS-biased hetrogeneous network. Text editing will be done on multi user nodes (as contrasted to dedicated workstations). Graphics and layout editing will be done on cluster controllers with a fixed, small number (i.e. two) of workstations. A CI will be used as the backbone of this network to facilitate the rapid movement of large amounts of design data.

We still have a lot of experimentation to do before we can decide if it makes sense to put the Graphics Editor Workstations directly in the work space of the designers. There are two issues; do the designers want do on-line design and if they do, does it make sense to put the workstations in their offices or to cluster them in a common area.



> = display & keyboard

A great deal of quantative data must be gathered before we can accurately quantify the actual number and detailed configuration of theses resources.

There are also a number of network issues yet to be understood; where do users receive mail, how can we avoid having to have accounts on each node, etc.

3.4 Tools

Tool	Primary Source	Backup Source
Text Editor	EDT	TECO, EMACS...
Code Mgt System	STEP (CMS)	
Graphics Editor	VALID **	SUDS *, DECDRAW
Performance Simulator	HAPS	
Behavioral Model Compiler Hierarchical Simulator	DECSIM	SAGE2 *
Circuit & Transmission Line Analyser	SPICE & SEI[illegible]	
Net List Extractor	VALID **	SUDS * (2060, not 2020) DECDRAW
Hierarchical Net List Compiler	VALID **	SCALD, DECSIM NETPRO
ROM & PLA Compiler	ROMGEN, PLATO & MICRO2	
Timing Verifier	VALID **	SCALD, ELKIND

Loading, Power & Failure Rate Analyzer	VALID ** <= + ?	SCALD, EPLS + Noelcke
Trial Layout & Interconnect Delay Analyzer	FINCUT <= + ? PINCUT	TI ***
Microcode Assembler	MICRO2	
Microcode Simulator	DECSIM	TUNS
Testability Analyzer		
Auto (stuck-at) Pattern Generation	LASER <= + ?	
Gate Array Auto AC Pattern Generation	AUTO AC <= + ?	
Gate Array Auto DC Pattern Generation	AUTO DC <= + ?	
Fault Simulator	LASER	DECSIM
Gate Array Test Program Generator	TECKIO (*?) <=?	
Gate Array Auto Placement	FINCUT	TI ***

Gate Array Auto Router	CHARIOT	TI ***
Gate Array Layout Editor	VLS	IDEAS *
Gate Array Interconnect Delay Analyzer	CHARIOT <= + ?	TI ***
Gate Array Design Rule Checker	OLIVER <=? (*?)	TI ***
Gate Array Interconnect Verifier	IV <=? (*?)	TI ***
PC & Backplane Auto Placement	PINCUT <=? (*?)	PCLS PLACER (*?) SCICARDS **
PC & Backplane Auto Router	TWIGY (*?)	SCICARDS **
PC & Backplane Layout Editor	VLS	IDEAS * SCICARDS **
PC & Backplane Design Rule Checker	SPACECHECK (*?)	SCICARDS ** <=?
PC & Backplane Interconnect Verifier	CONCHECK (*?)	SCICARDS ** <=?

All programs run on VAX/VMS except:
* Runs on DECsystem-20
** Runs on vendor specific hardware
*** Runs as vendor service

3.5 Support and Evolution

It has been my experience that CAD tools evolve significantly as they are used. We will have to have a viable support and evolution plan for each tool which will guarantee responsive bug fixing and adaptive evolution. At the same time, we must provide production-worthy tools. Tradeoffs in this space are especially difficult. Our history in Digital relative to successfully depending on one another (i.e. on another group) is poor. The prospect of having in-depth expertise in all of the tools listed above is mind boggling. Purchasing tools from vendors without access to source code is scary. We will work these management issues case by case.

4.0 DESIGN RULES

This section would list all the rules that would apply to the design and release of the design:
GA usage rules.
Timing assumptions.
Specifications formats.
Loading.
Physical partitioning.
Design Libraries.
Testing requirements.

5.0 METRICS

This section would list the design process metrics which we believe we can meet:
Design time/ GA.
Simulation Times.
Testing times.
Number of passes.
Etc.

APPENDIX A DESIGN PITFALLS

Warnings of traps to avoid.

APPENDIX B COMPONENT CATALOGUE

List of acceptable components to be used in the Nautilus design. Also included will be the Macro definitions.

APPENDIX C INTERCONNECT CATALOGUE

This will very likely be a Net List CREF which is automatically generated from the schematics.